i.MX 7Solo Applications Processor Reference Manual

Document Number: IMX7SRM
Rev. 0.1, 08/2016

Contents

Section number **Title** **Page**

Chapter 1
Introduction

Chapter 2
Memory Map

Chapter 3
Security

Chapter 4
ARM Platform and Debug

Chapter 5
Clocks and Power Management

Chapter 6
SNVS, Reset, Fuse and Boot

Chapter 7
Interrupts and DMA Events

Chapter 8
Chip IO and Pinmux

Chapter 9
External Memory

Chapter 10
Mass Storage

Chapter 11
Connectivity

Chapter 12
Timers

Chapter 13
Multimedia

Chapter 14
ADC and Temperature Monitor

Chapter 15
Low Speed Communication and Interconnects

Chapter 1
Introduction

1.1 Introduction

This chapter introduces the architecture of the i.MX 7Solo Multimedia Applications Processor. The i.MX 7Solo processor represents NXP Semiconductor's latest achievement in integrated multimedia applications processors that are part of a growing family of multimedia-focused products offering high-performance processing optimized for lowest power consumption.

1.2 Target Applications

The i.MX 7Solo is defined as the next-generation Applications Processor for the power sensitive use cases. It has a specific feature set for many mobile battery powered applications. i.MX 7Solo provides all the interfaces necessary for connecting peripherals such as WLAN, BlueTooth and etc.

i.MX 7Solo is a follow-on to the i.MX 6SoloLite, with advanced performance and higher power efficiency.

1.3 Acronyms and Abbreviations

The table below contains acronyms and abbreviations used in this document.

Acronyms and Abbreviated Terms

Term	Meaning
ADC	Analog-to-Digital Converter
AHB	Advanced High-performance Bus

Table continues on the next page...

Term	Meaning
AIPS	ARM IP Bus
ALU	Arithmetic Logic Unit
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
ASRC	Asynchronous Sample Rate Converter
AXI	Advanced eXtensible Interface
BIST	Built-In Self Test
CA7	ARM Cortex A7
CAN	Controller Area Network
CCM	Clock Controller Module
CPU	Central Processing Unit
CSI	CMOS Sensor Interface
CSU	Central Security Unit
CTI	Cross Trigger Interface
DAP	Debug Access Port
DDR	Double data rate
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory
ECC	Error correcting codes
ECSPI	Enhanced Configurable SPI
EIM	External Interface Module
ENET	Ethernet
EPIT	Enhanced Periodic Interrupt Timer
EPROM	Erasable Programmable Read-Only Memory
ETM	Embedded Trace Macrocell
FIFO	First-In-First-Out
GIC	General Interrupt Controller
GPC	General Power Controller
GPIO	General-Purpose I/O
GPR	General-Purpose Register
GPS	Global Positioning System
GPT	General-Purpose Timer
GPU	Graphics Processing Unit
HAB	High-Assurance Boot
I2C or I^2C	Inter-Integrated Circuit
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IOMUX	Input-Output Multiplexer
IP	Intellectual Property
IrDA	Infrared Data Association

Table continues on the next page...

Term	Meaning
JTAG	Joint Test Action Group (a serial bus protocol usually used for test purposes)
LCD	Liquid Crystal Display
LCDIF	Liquid Crystal Display Interface
LDO	Low-Dropout
LIFO	Last-In-First-Out
LRU	Least-Recently Used
LSB	Least-Significant Byte
LUT	Look-Up Table
LVDS	Low Voltage Differential Signaling
MAC	Medium Access Control
MMC	Multimedia Card
MMDC	Multi Mode DDR Controller
MSB	Most-Significant Byte
MT/s	Mega Transfers per second
OCRAM	On-Chip Random-Access Memory
OCOTP	On-Chip One-Time Programmable Controller
PCI	Peripheral Component Interconnect
PCIe	PCI enhanced
PCMCIA	Personal Computer Memory Card International Association
PGC	Power Gating Controller
PIC	Programmable Interrupt Controller
PMU	Power Management Unit
POR	Power-On Reset
PSRAM	Pseudo-Static Random Access Memory
PWM	Pulse Width Modulation
PXP	Pixel Pipeline
QoS	Quality of Service
R2D	Radians to Degrees
RISC	Reduced Instruction Set Computing
ROM	Read-Only Memory
ROMCP	ROM Controller with Patch
RTOS	Real-Time Operating System
Rx	Receive
SAI	Synchronous Audio Interface
SCU	Snoop Control Unit
SD	Secure Digital
SDIO	Secure Digital Input/Output
SDLC	Synchronous Data Link Control
SDMA	Smart DMA
SIM	Subscriber Identification Module
SoC	System-on-Chip

Table continues on the next page...

Term	Meaning
SPBA	Shared Peripheral Bus Arbiter
SPDIF	Sony Phillips Digital Interface
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
SRC	System Reset Controller
TFT	Thin-Film Transistor
TPIU	Trace Port Interface
Tx	Transmit
TZASC	TrustZone Address Space Controller
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
USDHC	Ultra Secured Digital Host Controller
WDOG	Watchdog
WLAN	Wireless Local Area Network
WXGA	Wide Extended Graphics Array

1.4 Features

1.4.1 ARM Cortex-A7 Platform

The i.MX 7Solo Applications Processor is based on ARM Cortex A7 MPCore™ Platform, which has the following features:

- ARM Cortex A7 MPCore™ Processor, including
  - 32KB L1 Instruction Cache
  - 32KB L1 Data Cache
  - Media Processing Engine(MPE) with NEON technology supporting the Advanced Single Instruction Multiple Data version 2 (SIMDv2) architecture
- Support of ARMv7A architecture including:
  - Security extensions for enhanced security
  - Virtualization extensions
  - Large Physical Address (LPA) extension (supported within Cortex-A7 MPCore platform)
- 512KB unified I/D L2 cache
- Snoop control unit(SCU)

- Generic timer
- Integrated Global Interrupt Controller(GIC) configured to support 128 shared peripheral interrupts
- In-order pipeline with direct and indirect branch prediction
- Harvard *Level 1* (L1) memory system with a *Memory Management Unit* (MMU)
- *Level2* (L2) memory system
- APB debug interface that supports integer processor clock ratios for 1:4 only
- Debug trace support through an *Embedded Trace Macrocell* (ETM) interface
- Floating Point Unit (FPU) with support of the VFPv4-D32 architecture

NOTE

This is a superset of VFPv4-D16

1.4.2 Cortex-M4 Core Platform

Cortex-M4 Core Platform include the following:

- 16 KB L1 Instruction Cache
- 16 KB L1 Data Cache
- 64KB TCM
- Integrated Nested Vectored Interrupt Controller (NVIC)
- FPU
- CoreMPU (Memory Protection Unit)

1.4.3 System Bus and Interconnect

System bus and interconnect include the following:

- Network interconnect(NIC-301) AXI arbiter
- Quality of service controller(QoSC) to configure priorities and limits of AXI transcations
- Performance monitor(PERFMON) to monitor AXI bus activity
- Debug monitor(DBGMON) to record AXI transactoins preceding a system reset

1.4.4 Clocking and Resets

Clocking and resets include:

- Clock control module(CCM) provides centralized clock generation and control
  - Simplified clock tree structure
  - Unified clock programming model for each clock root
  - Multicore awareness for resource domains
- System reset controller(SRC) provides reset generation and distribution

1.4.5 Interrupts and DMA

Interrupts and DMA include :

- 128 shared peripheral interrupts routed to Cortex-A7 Global Interrupt Controller(GIC) and Cortex-M4 nested vector interrupt controller(NVIC) for flexible interrupt handling
- Smart direct memory access(SDMA) controller supporting 48 DMA requests

1.4.6 On-Chip Memory

The on-chip memory system consists of the following levels:

- Level 1 Cache - 32KB Instruction, 32KB Data cache
- Level 2 Cache - Unified instruction and data, 512KB
- Boot ROM - including HAB, 96 KB
- Internal multimedia / shared, fast access RAM (OCRAM, 128KB)
- Buffer/general-purpose RAM (OCRAM__GP, 128KB)

1.4.7 External Memory Interface

The external memory interfaces supported on this chip include:

- DRAM
  - 16/32-bit DDR3 up to 533MHz clock (1066MT/s)
  - 16/32-bit DDR3L up to 533MHz clock (1066MT/s)
  - 32-bit LP-DDR2 up to 533MHz clock (1066MT/s)
  - 32-bit LP-DDR3 up to 533MHz clock (1066MT/s)
- 16-bit NOR FLASH
- 16-bit PSRAM/Cellular RAM

- 8-bit NAND-Flash, including support for Raw MLC/SLC devices, BCH ECC up to 62-bit, and ONFi3.2 compliance (clock rates up to 100 MHz and data rates upto 200 MB/sec)
- Quad SPI flash with support for parallel read mode of two identical flash devices
- SD/MMC Ports:
  - SDXC 8-bit, 832 Mbps
  - eMMC 5.0 (DDR) and eSD 3.0

1.4.8 Timers

The timers on this chip include :

- One local generic timer integrated into each Cortex-A7 CPU
- Global system counter with timer bus interface to Cortex-A7 MPCore generic timers
- One local system timer (SysTick) integrated into the Cortex-M4 CPU
- Four general purpose timer (GPT) modules
- Four watchdog timer (WDOG) modules
- Two flex Timer (FTM) modules supporting PWM signal generation and input and output capture

1.4.9 Image/Graphic Accelerators

The i.MX 7Solo makes use of dedicated HW image processing, in order to meet the targeted multimedia performance. The use of HW accelerators enables high performance at low power dissipation and lowers the CPU utilization, allowing it to be used for other tasks.

The i.MX 7Solo incorporate the following graphical hardware accelerators:

- ePXP – Enhanced PiXel Processing engine to off loading key pixel processing operations required to support LCD
  - Multiple input/output format support, including YUV / RGB / Gray Scale
  - Support both RGB/YUV scaling
  - Support overlay with Alpha blending
  - RGB656/RGB444 to RGBW4444 conversion with LUT
  - Color space conversion (CSC) , secondary color space conversion (CSC2), and rotation

1.4.10 Display and Camera Interfaces

i.MX 7Solo has parallel LCD interface to support eReaders with LCD panel.

- LCDIF supporting one parallel 24-bit LCD display with resolution up to 1920x1080 at 60Hz
- MIPI DSI host controller and D-PHY:
  - Supports 2 data lanes and 1 clock lane
  - Maximum bit rate of 1.5 Gbps
- CMOS sensor interface (CSI) supporting up to one parallel 24-bit camera interface
- MIPI CSI-2 controller and D-PHY:
  - Supports 2 data lanes and 1 clock lane
  - Maximum bit rate of 1.5 Gbps

1.4.11 Audio

Audio include the following:

- Three synchronous audio interface (SAI) modules supporting I2S, AC97, TDM, and codec/DSP interfaces
- Medium Quality Sound (MQS) for low-cost stereo audio output

1.4.12 General Connectivity Interfaces

The i.MX 7Solo contains a rich set of general connectivity interfaces, including:

- Three Ultra Secure Digital Host Controller (uSDHC) interfaces:
  - SD/SDIO 3.01 compliance with 208 MHZ SDR signaling to support up to 104 MB/sec
  - Support for SDXC (extended capacity)
- One USB 2.0 OTG controllers with integrated PHY interface
- One USB 2.0 host controller with HSIC interface
- One Gigabit Ethernet controllers with support for Ethernet AVB and IEEE1588
- Four eCSPI (Enhanced CSPI), up to 52 Mbps each
- Two Flexible Controller Area Network (FlexCAN) modules supporting the CAN 2.0B protocol specification
- Two Smartcard Interface Modules (SIMv2) supporting PCIv4 compliance
- Seven universal asynchronous receiver/transmitter (UARTs) modules
  - Providing RS232 interface

    - Supporting 9-bit RS485 multidrop mode
    - One of the five supports 8-wire (uart1) while others four supports 4-wire. (Due to SoC IOMUX limitation, since all UART IP are identical.)
- Four I2C modules
- Four SPI modules
- Four PWM modules
- Keypad module (KPP)

1.4.13 Security

Security functions are enabled and accelerated by the following hardware:

- ARM TrustZone including the TZ architecture (separation of interrupts, memory mapping, etc.) :
    - ARM Cortex A7 MPCore TrustZone support
    - TrustZone-aware Global Interrupt Controller(GIC)
    - TrustZone Address Space Controller (TZASC)
    - TrustZone watchdog timer
- SJC – System JTAG Controller, protecting JTAG from debug port attacks by regulating or blocking the access to the system debug features
- Cryptographic Acceleration and Assurance Module (CAAM)
    - PKHA block to support Public Key Cryptography with RSA 4096 and Elliptic Curve (ECC) algorithms
    - Real-time integrity checker (RTIC)
    - DPA(Differential Power Analysis) protection
    - True random number generation (RNG)
    - Manufacturing protection support
- Secure Non-Volatile Storage(SNVS), including
    - Secure Real Time Clock (RTC)
    - 10 External Tamper pins that can be configured to support 5 active meshes
- Cental Security Unit(CSU)
- On-chip RAM (OCRAM) secure region protection using OCRAM controller
- Security Sensor detection of physical attacks using temperature/voltage/frequency detection
- RNGB-True Random Number Generation
- HAB-High Assurance Boot(Secure Boot)

1.4.14 Multicore Support

Multicore support contains :

- Resource domain controller (RDC) to support isolation and safe sharing of system resources
- Messaging unit(MU)
- Hardware Semaphore(SEMA42)
- Shared bus topology
- ARMv7 exclusive access support for external memory interfaces that support a global monitor

1.4.15 Analog

The i.MX 7Solo Applications Processor has two 12-bit general purpose ADC modules

1.4.16 GPIO and Pin Multiplexing

- Seven general-purpose input/output (GPIO) modules with interrupt capability
- Input/output multiplexing controller (IOMUXC) to provide centralized pad control.

1.4.17 Power Management

The i.MX 7Solo power management unit consists of :

- Temperature sensor with programmable trip points
- General Power Controller (GPC) to provide coordination of system power states
- Flexible power domain partitioning with internal power switches to support efficient power management

1.4.18 System Debug

The i.MX 7Solo processor system debug features are :

- ARM CoreSight debug and trace architecture

- Trace Port Interface Unit (TPIU) to support off-chip real-time trace
- Embedded Trace FIFO (ETF) with 4 KB internal storage to provide trace buffering
- Unified trace capability for Cortex-A7 and Cortex-M4 CPUs
- Cross Triggering Interface (CTI)
- Support for 5-pin (JTAG) and 2-pin (cJTAG, SWD) debug interfaces

NOTE

GPR1 bits 28-29 must be set in order to enable debug mode for on-chip modules.

1.5 Architectural Overview

This section contains the i.MX 7Solo architectural details.

1.5.1 Simplified Block Diagram

The high-level block diagram is shown in the figure below. This diagram provides a view of the major sub-systems and logical connectivity.

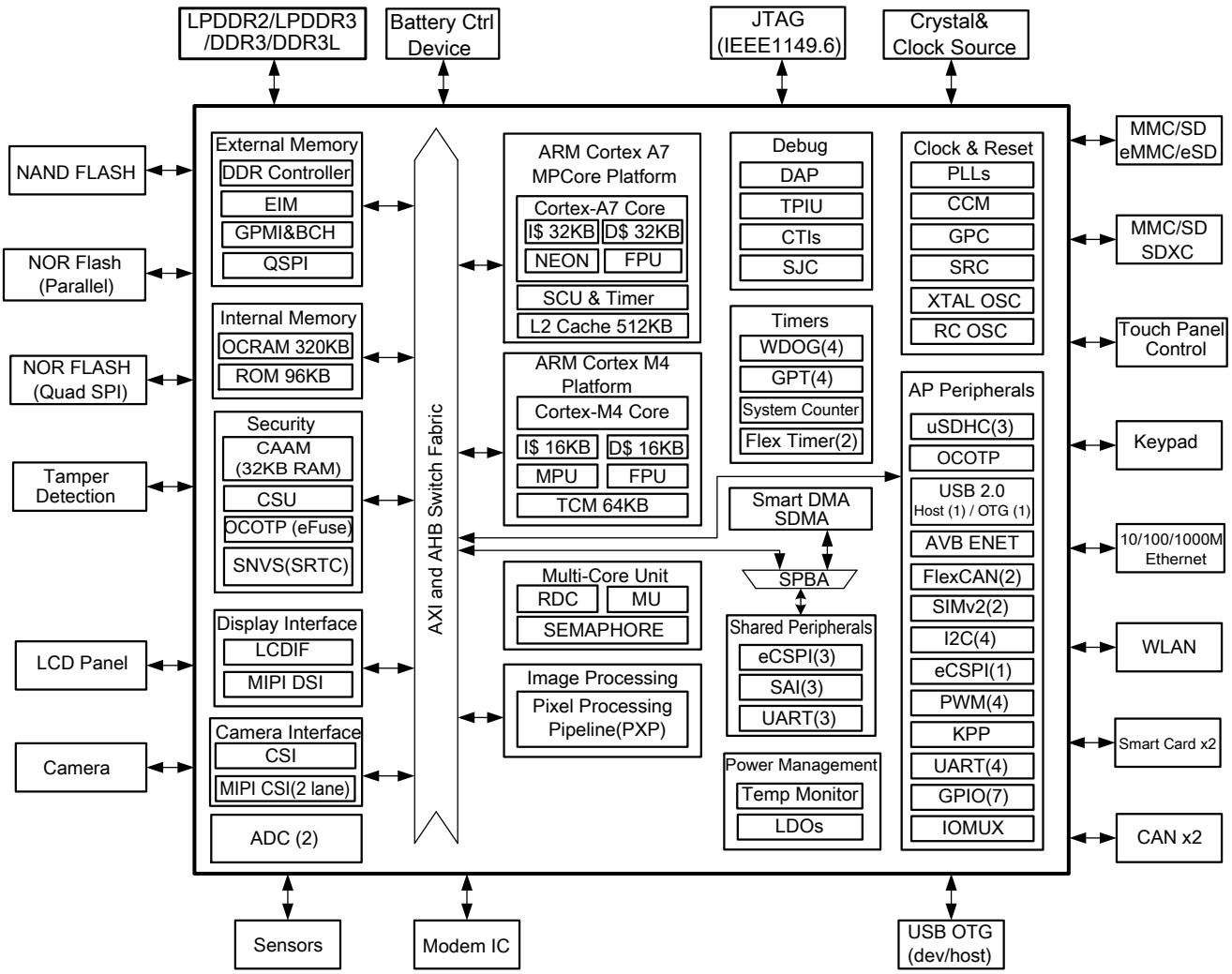


Figure 1-1. High-Level Block Diagram

1.6 Primary Boot Options

The i.MX 7Solol supports boot from following devices:

- NOR Flash
- NAND flash (including SLC and MLC)
- SDIO / MMC / SDXC
- eSD 3.0/eMMC 5.0 (fast boot)
- SPI (serial flash)
- USB (recovery mode) and plug detect (USB host stack not required in ROM, can be stored in an external boot device, e.g. serial flash)
- USB Solid-State Drive (via plug-in mode)

- Ethernet (via plug-in mode)
- QSPI

The Cortex-A7 is the primary boot on the i.MX7Solo. The Cortex-M4 core can be enabled during boot as a secondary core to handle timing-critical tasks, but it cannot be used as the boot core.

1.7 Operation Temperature

The i.MX 7Solo supports both consumer temperature range and extended consumer temperature range. The operation condition for these 2 different parts will be:

- The junction temperature will be 0 to 95 °C for consumer part
- The junction temperature will be -20 to 105°C for extended consumer part

NOTE

Junction temperature (Tj) is the average temperature of the whole chip. Although the junction temperature may be within the specification, certain regions of the chip develop hot spots that can be greater than Tj. These hot spots must stay below 105 C to ensure the chip is reliable and operates within cell library limits.

1.8 Package

There will be 2 packages supported for i.MX 7Solo:

- 12mm x 12mm MAPBGA, 0.4mm pitch
- 19mm x 19mm MAPBGA, 0.75mm pitch

1.9 Endianness Support

i.MX 7Solo supports Little Endian mode only.

Chapter 2
Memory Map

2.1 Memory

This chapter introduces the memory architecture of the chip. The system memory high-level partition is defined below:

2.1.1 Memory system overview

2.1.1.1 On-chip L1, L2 caches, TCM

Cortex-A7 MPcore Platform

- Level 1 Cache (2x per Cortex-A7 Core)
  - Instruction (32 KB)
  - Data (32 KB)
- Level 2 Cache, shared by the two Cortex-A7 cores:
  - Unified instruction and data (512 KB)

Cortex M4 Platform

- Cache
  - Instruction (32 KB)
  - Data (32 KB)
- Tightly-Coupled-Memory
  - TCML on Code Bus (32KB)
  - TCMH on System Bus (32KB)

2.1.1.2 On-chip memories

2.1.1.3 External L3 memories

The chip supports external memories, via the following memory interfaces / controllers:

- DDR Controller
  - x32/x16 LPDDR2-1066
  - x32/x16 LPDDR3-1066
  - x32/x16 DDR3-1066
  - x32/x16 DDR3L-1066
- NOR Flash
  - Dual-Channel Quad-SPI (Q-SPI) interface. (include support for in-place execution)
  - 16-bit/8-bit Parallel NOR Flash interface. (include support for in-place execution)
- Expansion SD/MMC (eSD/eMMC) and SDXC ports
  - Conforms to the SD Host Controller Standard Specification version 3.0;
  - Compatible with the MMC System Specification version 4.5;
  - Card bus clock frequency up to 208 MHz;
  - Up to 3 controllers supported;
- RawNAND Flash
  - 8-bit NAND FLASH, up to 4 devices supported by 4 chip-selects and 1 ganged ready/busy;
  - ONFI 2.x complaint, synchronous clock rate of up to 100 MHz with data rate of up to 200 MB/s;
  - Support ONFI NAND for Micron and Hynix and Toggle NAND for Toshiba and Samsung.
  - BCH62 for ECC, up to 200MB/s

2.1.2 Cortex-A7 Memory Map

Start Address	End Address	Region	Size	Description
8000_0000	FFFF_FFFF	DDR Address	2048MB	DDR Main memory
7000_0000	7FFF_FFFF	FLASH	256MB	Reserved
6000_0000	6FFF_FFFF		256MB	QSPI1
4000_0000	5FFF_FFFF	Reserved	512MB	Reserved
3800_0000	3FFF_FFFF	Reserved	128MB	Reserved
3600_0000	37FF_FFFF	QSPI RX Buffers	32MB	Reserved

Table continues on the next page...

Start Address	End Address	Region	Size	Description
3400_0000	35FF_FFFF		32MB	QSPI1 RX Buffer
3390_0000	33FF_FFFF	Reserved	7MB	Reserved
3382_0000	338F_FFFF	Reserved	896KB	Reserved
3380_0000	3381_FFFF	Reserved	128KB	Reserved
3310_0000	337F_FFFF	MMAP Peripherals	7MB	Reserved
3301_0000	330F_FFFF		960KB	Reserved
3300_8000	3300_FFFF		32KB	Reserved
3300_0000	3300_7FFF		32KB	APBH DMA
3280_0000	32FF_FFFF	Reserved	8MB	Rerserved
3270_0000	327F_FFFF	GPV_7	1MB	Reserved
3260_0000	326F_FFFF	GPV_6	1MB	"m4" configuration port
3250_0000	325F_FFFF	GPV_5	1MB	"display" configuration port
3240_0000	324F_FFFF	GPV_4	1MB	"enet" configuration port
3230_0000	323F_FFFF	GPV_3	1MB	"per_m" configuration port
3220_0000	322F_FFFF	GPV_2	1MB	"per_s" configuration port
3210_0000	321F_FFFF	GPV_1	1MB	"wakeup" configuration port
3200_0000	320F_FFFF	GPV_0	1MB	"main" configuration port
3140_0000	31FF_FFFF	Reserved	12MB	Reserved
3110_0000	313F_FFFF	ARM Peripherals	3MB	Reserved
3101_0000	310F_FFFF		960KB	Reserved
3100_8000	3100_FFFF		32KB	Reserved
3100_0000	3100_7FFF		32KB	GIC
30C0_0000	30FF_FFFF	Periph (AIPS)	4MB	Reserved
3080_0000	30BF_FFFF		4MB	AIPS-3, See IP listing on separate map.
3040_0000	307F_FFFF		4MB	AIPS-2, See IP listing on separate map.
3000_0000	303F_FFFF		4MB	AIPS-1, See IP listing on separate map.
2800_0000	2FFF_FFFF	EIM	128MB	EIM - CS0 (NOR/SRAM)
2000_0000	27FF_FFFF	Reserved	128MB	Reserved
1000_0000	1FFF_FFFF	Reserved	256MB	Reserved
0800_0000	0FFF_FFFF	Reserved	128MB	Reserved
0400_0000	07FF_FFFF	Reserved	64MB	Reserved
0100_0000	03FF_FFFF	Reserved	48MB	Reserved
00C0_0000	00FF_FFFF	Reserved	4MB	Reserved
00B0_0000	00BF_FFFF	Reserved	1MB	Reserved
00A0_0000	00AF_FFFF	OCRAM	1MB	Reserved
0094_8000	009F_FFFF		736KB	Reserved
0094_0000	0094_7FFF		32KB	OCRAM_PXP
0092_0000	0093_FFFF		128KB	OCRAM_GP
0090_0000	0091_FFFF		128KB	OCRAM 128KB
0081_0000	008F_FFFF	Reserved	960KB	Reserved
0080_8000	0080_FFFF	TCM	32KB	Reserved

Table continues on the next page...

Start Address	End Address	Region	Size	Description
0080_0000	0080_7FFF		32KB	TCMU
007F_8000	007F_FFFF		32KB	TCML
007F_0000	007F_7FFF		32KB	Reserved
0070_0000	007E_FFFF	Reserved	960KB	Reserved
0060_0000	006F_FFFF	Reserved	1MB	Reserved
0050_0000	005F_FFFF	Reserved	1MB	Reserved
0040_0000	004F_FFFF	Reserved	1MB	Reserved
0020_0000	003F_FFFF	Reserved	2MB	Reserved
0019_0000	001F_FFFF	Reserved	448KB	Reserved
0018_8000	0018_FFFF	OCRAM_S	32KB	Reserved
0018_0000	0018_7FFF		32KB	OCRAM_S
0011_0000	0017_FFFF	Reserved	448KB	Reserved
0010_8000	0010_FFFF	CAAM	32KB	Reserved
0010_0000	0010_7FFF		32KB	CAAM (32K secure RAM)
0002_0000	000F_FFFF	Reserved	896KB	Reserved
0001_8000	0001_FFFF	Boot ROM	32KB	Reserved
0001_7000	0001_7FFF		4KB	Boot ROM - Protected 4KB area
0000_0000	0001_6FFF		92KB	Boot ROM (ROMCP)

2.1.3 Cortex-M4 Memory Map

Start Address	End Address	Region	Size	Description
E010_0000	FFFF_FFFF	Reserved	511MB	Reserved
E000_0000	E00F_FFFF	CM4 PPB	1MB	CM4 PPB
8000_0000	DFFF_FFFF	DDR Address	1536MB	DDRC
7000_0000	7FFF_FFFF	FLASH	256MB	Reserved
6000_0000	6FFF_FFFF		256MB	QSPI1
4000_0000	5FFF_FFFF	Reserved	512MB	Reserved
3800_0000	3FFF_FFFF	Reserved	128MB	Reserved
3600_0000	37FF_FFFF	QSPI RX Buffers	32MB	Reserved
3400_0000	35FF_FFFF		32MB	QSPI1 RX Buffer
3390_0000	33FF_FFFF	Reserved	7MB	Reserved
3382_0000	338F_FFFF	Reserved	896KB	Reserved
3380_0000	3381_FFFF	Reserved	128KB	Reserved
3310_0000	337F_FFFF	MMAP Peripherals	7MB	Reserved
3301_0000	330F_FFFF		960KB	Reserved
3300_8000	3300_FFFF		32KB	Reserved

Table continues on the next page...

Start Address	End Address	Region	Size	Description
3300_0000	3300_7FFF	APBH DMA	32KB	APBH DMA
3280_0000	32FF_FFFF	Reserved	8MB	Reserved
3270_0000	327F_FFFF	Reserved	1MB	Reserved
3260_0000	326F_FFFF	GPV_6	1MB	"m4" configuration port
3250_0000	325F_FFFF	GPV_5	1MB	"display" configuration port
3240_0000	324F_FFFF	GPV_4	1MB	"enet" configuration port
3230_0000	323F_FFFF	GPV_3	1MB	"per_m" configuration port
3220_0000	322F_FFFF	GPV_2	1MB	"per_s" configuration port
3210_0000	321F_FFFF	GPV_1	1MB	"wakeup" configuration port
3200_0000	320F_FFFF	GPV_0	1MB	"main" configuration port
3100_0000	31FF_FFFF	Reserved	16MB	Reserved
30C0_0000	30FF_FFFF	Periph (AIPS)	4MB	Reserved
3080_0000	30BF_FFFF		4MB	AIPS-3, See IP listing on separate map.
3040_0000	307F_FFFF		4MB	AIPS-2, See IP listing on separate map.
3000_0000	303F_FFFF		4MB	AIPS-1, See IP listing on separate map.
2800_0000	2FFF_FFFF	EIM	128MB	EIM - CS0 (NOR/SRAM)
2400_0000	27FF_FFFF	Reserved	64MB	Reserved
2200_0000	23FF_FFFF	Reserved	32MB	Reserved
2100_0000	21FF_FFFF	Reserved	16MB	Reserved
2040_0000	20FF_FFFF	Reserved	12MB	Reserved
2030_0000	203F_FFFF	CM4 ALIAS SYSTEM	1MB	Reserved
2024_8000	202F_FFFF		736KB	Reserved
2024_0000	2024_7FFF		32KB	OCRAM_PXP
2022_0000	2023_FFFF		128KB	OCRAM_GP
2020_0000	2021_FFFF		128KB	OCRAM_128KB
2019_0000	201F_FFFF		448KB	Reserved
2018_8000	2018_FFFF		32KB	Reserved
2018_0000	2018_7FFF		32KB	OCRAM_S
2011_0000	2017_FFFF		448KB	Reserved
2010_8000	2010_FFFF		32KB	Reserved
2010_0000	2010_7FFF		32KB	CAAM (32K secure RAM)
2004_0000	200F_FFFF	Reserved	768KB	Reserved
2003_8000	2003_FFFF	Reserved	32KB	Reserved
2002_0000	2003_7FFF	Boot ROM (all 96KB)	96KB	Boot ROM (ROMCP)
2001_0000	2001_FFFF	Reserved	64KB	Reserved
2000_8000	2000_FFFF	TCM	32KB	Reserved
2000_0000	2000_7FFF		32KB	TCMU
1FFF_8000	1FFF_FFFF		32KB	TCML
1FFF_0000	1FFF_7FFF		32KB	Reserved
1000_0000	1FFE_FFFF	CM4 ALIAS CODE	256MB-64 KB	DDR Code alias

Table continues on the next page...

Start Address	End Address	Region	Size	Description
0C00_0000	0FFF_FFFF		64MB	Reserved
0800_0000	0BFF_FFFF		64MB	QSPI1
0400_0000	07FF_FFFF		64MB	EIM - CS0 (NOR/SRAM)
0100_0000	03FF_FFFF	Reserved	48MB	Reserved
00C0_0000	00FF_FFFF	Reserved	4MB	Reserved
00B0_0000	00BF_FFFF	Reserved	1MB	Reserved
00A0_0000	00AF_FFFF	OCRAM	1MB	Reserved
0094_8000	009F_FFFF		736KB	Reserved
0094_0000	0094_7FFF		32KB	OCRAM_PXP
0092_0000	0093_FFFF		128KB	OCRAM_GP
0090_0000	0091_FFFF		128KB	OCRAM_128KB
0081_0000	008F_FFFF	Reserved	960KB	Reserved
0080_8000	0080_FFFF	Reserved	32KB	Reserved
0080_0000	0080_7FFF	ROM (Low 64KB)	32KB	Boot ROM (ROMCP)
007F_8000	007F_FFFF		32KB	Boot ROM (ROMCP)
007F_0000	007F_7FFF	Reserved	32KB	Reserved
0070_0000	007E_FFFF	Reserved	960KB	Reserved
0060_0000	006F_FFFF	Reserved	1MB	Reserved
0050_0000	005F_FFFF	Reserved	1MB	Reserved
0040_0000	004F_FFFF	Reserved	1MB	Reserved
0020_0000	003F_FFFF	Reserved	2MB	Reserved
0019_0000	001F_FFFF	Reserved	448KB	Reserved
0018_8000	0018_FFFF	OCRAM_S	32KB	Reserved
0018_0000	0018_7FFF		32KB	OCRAM_S
0011_0000	0017_FFFF	Reserved	448KB	Reserved
0010_8000	0010_FFFF	CAAM	32KB	Reserved
0010_0000	0010_7FFF		32KB	CAAM (32K secure RAM)
0002_0000	000F_FFFF	Reserved	896KB	Reserved
0001_8000	0001_FFFF	Reserved	32KB	Reserved
0001_7000	0001_7FFF	ROM (High 64KB)	4KB	Boot ROM - Protected 4KB area
0000_8000	0001_6FFF	ROM (High 64KB)	60KB	Boot ROM (ROMCP)
0000_0000	0000_7FFF	CM4 ALIAS CODE	32KB	OCRAM_S ALIAS

2.1.4 DMA memory map

The Smart DMA memory map is defined in the following table.

Table 2-1. SDMA Peripheral Memory Map

Address	Size	Peripheral
0xE000	4 KB	Reserved for SDMA internal registers
0xD000	4 KB	Reserved
0xC000	4 KB	SAI3
0xB000	4 KB	SAI2
0xA000	4 KB	SAI1
0x9000	4 KB	UART2
0x8000	4 KB	UART3
0x7000	4 KB	Reserved for SDMA internal registers
0x6000	4 KB	UART1
0x5000	4 KB	Reserved
0x4000	4 KB	eCSPI3
0x3000	4 KB	eCSPI2
0x2000	4 KB	eCSPI1
0x1000	4 KB	Reserved
0x0000	4 KB	Reserved for SDMA internal memory

2.1.5 AIPS Memory Map

Table 2-2. AIPS-1 memory map

Start Address	End Address	Region	NIC Port	Size
303F_0000	303F_FFFF	AIPS-1 (s_b_0)	Reserved	64KB
303E_0000	303E_FFFF		CSU	64KB
303D_0000	303D_FFFF		RDC	64KB
303C_0000	303C_FFFF		SEMAPHORE2	64KB
303B_0000	303B_FFFF		SEMAPHORE1	64KB
303A_0000	303A_FFFF		GPC	64KB
3039_0000	3039_FFFF		SRC	64KB
3038_0000	3038_FFFF		CCM	64KB
3037_0000	3037_FFFF		SNVS_HP	64KB
3036_0000	3036_FFFF		Analog	64KB
3035_0000	3035_FFFF		OCOTP_CTRL	64KB
3034_0000	3034_FFFF		IOMUXC_GPR	64KB
3033_0000	3033_FFFF		IOMUXC	64KB

Table continues on the next page...

Table 2-2. AIPS-1 memory map (continued)

Start Address	End Address	Region	NIC Port	Size
3032_0000	3032_FFFF		KPP	64KB
3031_0000	3031_FFFF		ROMCP	64KB
3030_0000	3030_FFFF		GPT4	64KB
302F_0000	302F_FFFF		GPT3	64KB
302E_0000	302E_FFFF		GPT2	64KB
302D_0000	302D_FFFF		GPT1	64KB
302C_0000	302C_FFFF		IOMUXC_LPSR	64KB
302B_0000	302B_FFFF		WDOG4	64KB
302A_0000	302A_FFFF		WDOG3	64KB
3029_0000	3029_FFFF		WDOG2	64KB
3028_0000	3028_FFFF		WDOG1	64KB
3027_0000	3027_FFFF		IOMUXC_LPSR_GPR	64KB
3026_0000	3026_FFFF		GPIO7	64KB
3025_0000	3025_FFFF		GPIO6	64KB
3024_0000	3024_FFFF		GPIO5	64KB
3023_0000	3023_FFFF		GPIO4	64KB
3022_0000	3022_FFFF		GPIO3	64KB
3021_0000	3021_FFFF		GPIO2	64KB
3020_0000	3020_FFFF		GPIO1	64KB
301F_0000	301F_FFFF		AIPS-1 Configuration	64KB
3010_0000	301E_FFFF	AIPS-1 Glob. Module Enable	Reserved	960KB
3000_0000	300F_FFFF		DAP Memory Map	1024KB

Table 2-3. AIPS-2 memory map

Start Address	End Address	Region	Allocation	Size
307F_0000	307F_FFFF	AIPS-2 (s_b_1)	Reserved	64KB
307E_0000	307E_FFFF		AXI_DEBUG_MON	64KB
307D_0000	307D_FFFF		PERFMON2	64KB
307C_0000	307C_FFFF		PERFMON1	64KB
307B_0000	307B_FFFF		Reserved	64KB
307A_0000	307A_FFFF		DDRC	64KB
3079_0000	3079_FFFF		DDR_PHY	64KB
3078_0000	3078_FFFF		TZASC	64KB
3077_0000	3077_FFFF		Reserved	64KB
3076_0000	3076_FFFF		MIPI DSI	64KB
3075_0000	3075_FFFF		MIPI CSI	64KB
3074_0000	3074_FFFF		Reserved	64KB
3073_0000	3073_FFFF		LCDIF	64KB

Table continues on the next page...

Table 2-3. AIPS-2 memory map (continued)

Start Address	End Address	Region	Allocation	Size
3072_0000	3072_FFFF		Reserved	64KB
3071_0000	3071_FFFF		CSI	64KB
3070_0000	3070_FFFF		PXP	64KB
306F_0000	306F_FFFF		-	64KB
306E_0000	306E_FFFF		Reserved	64KB
306D_0000	306D_FFFF		-	64KB
306C_0000	306C_FFFF		System Counter_CTRL	64KB
306B_0000	306B_FFFF		System Counter_CMP	64KB
306A_0000	306A_FFFF		System Counter_RD	64KB
3069_0000	3069_FFFF		PWM4	64KB
3068_0000	3068_FFFF		PWM3	64KB
3067_0000	3067_FFFF		PWM2	64KB
3066_0000	3066_FFFF		PWM1	64KB
3065_0000	3065_FFFF		FlexTimer2	64KB
3064_0000	3064_FFFF		FlexTimer1	64KB
3063_0000	3063_FFFF		eCSPI4	64KB
3062_0000	3062_FFFF		ADC2_WRAPPER	64KB
3061_0000	3061_FFFF		ADC1_WRAPPER	64KB
3060_0000	3060_FFFF		Reserved	64KB
305F_0000	305F_FFFF		AIPS-2 configuration	64KB
3054_0000	305E_FFFF	AIPS-2 Glob. Module Enable	Reserved	704KB
3050_0000	3053_FFFF		Reserved	256KB
3044_0000	304F_FFFF	AIPS-2 Glob. Module Enable	Reserved	768KB
3040_0000	3043_FFFF		Reserved	256KB

Table 2-4. AIPS-3 memory map

Start Address	End Address	Region	NIC Port	Size
30BF_0000	30BF_FFFF	AIPS-3 (s_b_2)	-	64KB
30BE_0000	30BE_FFFF		ENET1	64KB
30BD_0000	30BD_FFFF		SDMA	64KB
30BC_0000	30BC_FFFF		EIM	64KB
30BB_0000	30BB_FFFF		QSPI	64KB
30BA_0000	30BA_FFFF		SIM2	64KB
30B9_0000	30B9_FFFF		SIM1	64KB
30B8_0000	30B8_FFFF		Reseved	64KB
30B7_0000	30B7_FFFF		Reseved	64KB
30B6_0000	30B6_FFFF		uSDHC3	64KB
30B5_0000	30B5_FFFF		uSDHC2	64KB

Table continues on the next page...

Table 2-4. AIPS-3 memory map (continued)

Start Address	End Address	Region	NIC Port	Size
30B4_0000	30B4_FFFF		uSDHC1	64KB
30B3_0000	30B3_FFFF		USB3 (HOST)	64KB
30B2_0000	30B2_FFFF		-	64KB
30B1_0000	30B1_FFFF		USB1 (OTG1)	64KB
30B0_0000	30B0_FFFF		Reserved	64KB
30AF_0000	30AF_FFFF		Reserved	64KB
30AE_0000	30AE_FFFF		Reserved	64KB
30AD_0000	30AD_FFFF		USB PL301	64KB
30AC_0000	30AC_FFFF		SEMAPHORE-HS	64KB
30AB_0000	30AB_FFFF		MU-B	64KB
30AA_0000	30AA_FFFF		MU-A	64KB
30A9_0000	30A9_FFFF		UART7	64KB
30A8_0000	30A8_FFFF		UART6	64KB
30A7_0000	30A7_FFFF		UART5	64KB
30A6_0000	30A6_FFFF		UART4	64KB
30A5_0000	30A5_FFFF		I2C4	64KB
30A4_0000	30A4_FFFF		I2C3	64KB
30A3_0000	30A3_FFFF		I2C2	64KB
30A2_0000	30A2_FFFF		I2C1	64KB
30A1_0000	30A1_FFFF		FlexCAN2	64KB
30A0_0000	30A0_FFFF		FlexCAN1	64KB
309F_0000	309F_FFFF		AIPS-3 Configuration	64KB
3094_0000	309E_FFFF	AIPS-3 Glob. Module Enable	Reserved	704KB
3090_0000	3093_FFFF		CAAM	256KB
308F_0000	308F_FFFF	AIPS-3 (s_b_2, via SPBA) Glob. Module Enable	SPBA	64KB
308E_0000	308E_FFFF		Reserved for SDMA internal registers	64KB
308D_0000	308D_FFFF		Reserved	64KB
308C_0000	308C_FFFF		SAI3	64KB
308B_0000	308B_FFFF		SAI2	64KB
308A_0000	308A_FFFF		SAI1	64KB
3089_0000	3089_FFFF		Reserved for SDMA internal registers	64KB
3088_0000	3088_FFFF		UART3	64KB
3087_0000	3087_FFFF		UART2	64KB
3086_0000	3086_FFFF		UART1	64KB
3085_0000	3085_FFFF		Reserved	64KB
3084_0000	3084_FFFF		eCSPI3	64KB
3083_0000	3083_FFFF		eCSPI2	64KB
3082_0000	3082_FFFF		eCSPI1	64KB

Table continues on the next page...

Table 2-4. AIPS-3 memory map (continued)

Start Address	End Address	Region	NIC Port	Size
3081_0000	3081_FFFF		Reserved	64KB
3080_0000	3080_FFFF		Reserved for SDMA internal memory	64KB

2.1.6 DAP Memory Map

Table 2-5. DAP Memory Map Table

Start Address	End Address	Region	NIC Port	Size	Allocation
300C_0000	300F_FFFF	ARM Cortex A7 MPCore Platform - DAP	AIPS-1, Global Map (s_b_1)	256KB	Reserved
3008_B000	300B_FFFF			212KB	Reserved
3008_A000	3008_AFFF			4KB	Reserved
3008_9000	3008_9FFF			4KB	CoreSight Hugo CTI1
3008_8000	3008_8FFF			4KB	CoreSight Hugo CTI0
3008_7000	3008_7FFF			4KB	TPIU
3008_6000	3008_6FFF			4KB	TMC_ETR
3008_5000	3008_5FFF			4KB	CoreSight Hugo ATB_REPLICATOR
3008_4000	3008_4FFF			4KB	TMC_ETB
3008_3000	3008_3FFF			4KB	ATB_FUNNEL
3008_2000	3008_2FFF			4KB	TSGEN_READ
3008_1000	3008_1FFF			4KB	TSGEN_CTRL
3008_0000	3008_0FFF			4KB	CoreSight Hugo ROM Table
3007_F000	3007_FFFF			4KB	Reserved
3007_E000	3007_EFFF			4KB	Reserved
3007_D000	3007_DFFF			4KB	CPU1 ETM
3007_C000	3007_CFFF			4KB	CPU0 ETM
3007_B000	3007_BFFF			4KB	Reserved
3007_A000	3007_AFFF			4KB	Reserved
3007_9000	3007_9FFF			4KB	CPU1 CTI
3007_8000	3007_8FFF			4KB	CPU0 CTI
3007_7000	3007_7FFF			4KB	Reserved
3007_6000	3007_6FFF			4KB	Reserved
3007_5000	3007_5FFF			4KB	Reserved
3007_4000	3007_4FFF			4KB	Reserved
3007_3000	3007_3FFF			4KB	CPU1 PMU
3007_2000	3007_2FFF			4KB	CPU1 Debug
3007_1000	3007_1FFF			4KB	CPU0 PMU

Table continues on the next page...

Table 2-5. DAP Memory Map Table (continued)

Start Address	End Address	Region	NIC Port	Size	Allocation
3007_0000	3007_0FFF			4KB	CPU0 Debug
3006_1000	3006_FFFF			60KB	Reserved
3006_0000	3006_0FFF			4KB	CA7 ROM Table
3004_2000	3005_FFFF			120KB	Reserved
3004_1000	3004_1FFF			4KB	CA7_ATB_FUNNEL
3004_0000	3004_0FFF			4KB	CA7_DEBUG ROM Table
3000_1000	3003_FFFF			252KB	Reserved
3000_0000	3000_0FFF			4KB	DAP ROM Table

Chapter 3
Security

3.1 System Security

3.1.1 Overview

Security is a common requirement for platforms built using the i.MX 7D, although the specific needs vary greatly depending on the platform and market. The type and cost of assets to be protected on a portable consumer device are very different from those to be protected on automotive or industrial platforms, and the same applies to the kind of attacks and level of resources threatening those assets. The platform designer must select an appropriate set of counter measures to meet the relevant platform security needs.

For the platform designer to meet the requirements for each market, the i.MX 7D incorporates a range of security features which can be used individually or in concert to underpin the platform security architecture. Most of the i.MX 7D security features provide protection against particular kinds of attack and can be configured at various levels according to the required degree of protection. These features are designed to work together and can be integrated with appropriate software to create defensive layers. In addition to protection features, the i.MX 7D includes a general purpose accelerator to enhance the performance of selected industry standard cryptographic algorithms.

The following is an introduction to the i.MX 7D security components.

- Cryptographic module with AES 128/256 symmetric algorithms, elliptic curve and RSA public key algorithms, random number generator entropy source and NIST-validated DRBG, cryptographic key protection, run-time integrity checking.
- High Assurance Boot (HAB) feature in the System Boot up to RSA-4096 signature verification
- Secure Non Volatile Storage (SNVS)
- TrustZone (TZ) Architecture in the ARM Cortex A7 Platform, TrustZone aware Interrupt Controller (GIC) and TrustZone Watchdog Timer (WDOG-2)

- TrustZone Address Space Controller (TZC-380) - providing security address region control functions on DDR memory space.
- On-chip RAM (OCRAM) with TrustZone protection using OCRAM controller.
- 32 Kbyte of on-chip Secure RAM
- On chip OTP (OCOTP) with on-chip electrical fuses
- Central Security Unit (CSU)
- Resource Domain Controller (RDC)
- Secure JTAG Controller (SJC)
- Locked mode in the Smart Direct Memory Access (SDMA) controller

Detailed descriptions of the components are provided in the *Multimedia Applications Processor Security Reference Manual*.

3.1.2 Central Security Unit (CSU)

3.1.2.1 CSU Overview

The CSU manages the system security policy for peripheral access on the SoC. The CSU allows trusted code to set individual security access privilges on each of the peripherals, using one of eight security access privilege levels. Also, according to programmed policy, the CSU may assign bus master security privileges during bus transactions.

3.1.2.2 CSU Features

The Central Security Unit (CSU) sets access control policies between bus masters and bus slaves, allowing peripherals to be separated into distinct security domains. This protects against unauthorized access to data e.g. when software programs a DMA bus master to access addresses that the software itself is prohibited from accessing directly. Configuring DMA bus master privileges in the CSU consistent with software privileges defends against such attempted accesses.

CSU has the following security related features:

- Peripheral access policy - Appropriate bus master privileges and identity are required to access each peripheral.
- Masters privilege policy - CSU overrides bus master privilege signals, i.e. user/supervisor secure/non-secure, according to access control policy.

3.1.2.3 CSU Functional Description

The CSU enables secure software to set bus privilege security policy within the platform.

Security policies may be set, and optionally locked in the CSU registers. These privilege values may originate in the command sequence file (CSF) which is processed by the High Assurance Boot (HAB) itself or by an HAB authenticated image which executes after the initial boot ROM phase.

3.1.2.3.1 CSU Peripheral Access Policy

According to its programmed policy, the CSU determines the bus master privileges and the masters that are allowed to access each of the slave peripherals.

There are four security modes of operation (i.e. bus privileges) in the system distinguished by security (TrustZone/non-TrustZone) and privilege (Supervisor/User) setting of the module. Below is the list of these security modes from the highest security level to the lowest:

- TrustZone (Secure) Privilege (Supervisor) Mode - Highest Security Level
- TrustZone (Secure) non-Privilege (User) Mode - Medium Security Level
- non-TrustZone (Regular) Privilege (Supervisor) Mode - Medium Security Level
- non-TrustZone (Regular) non-Privilege (User) Mode - Lowest Security Level

This functionality is implemented as follows:

The Configure Slave Level (CSL) Register value for a specified peripheral resource defines the output signal -- csu_sec_level for that peripheral. The value of this signal determines by what master privileges a peripheral is accessible. The relationship between the value of the csu_sec_level signal and security operation mode is shown in the table below. The CSL registers reside in the CSU module. Details, describing CSL register fields and how they are programmed to control access privileges for specific peripherals, can be found in the Security Reference Manual (see System Security Overview section).

Table 3-1. Permission Access Table

CSU_SEC_LEVEL[2:0]	Non-Secure User Mode	Non-Secure Spvr Mode	Secure (TZ) User Mode	Secure (TZ) Spvr Mode	CSL register value
(0) 000	RD+WR	RD+WR	RD+WR	RD+WR	8'b1111_1111
(1) 001	None	RD+WR	RD+WR	RD+WR	8'b1011_1011
(2) 010	RD	RD	RD+WR	RD+WR	8'b0011_1111
(3) 011	None	RD	RD+WR	RD+WR	8'b0011_1011
(4) 100	None	None	RD+WR	RD+WR	8'b0011_0011
(5) 101	None	None	None	RD+WR	8'b0010_0010
(6) 110	None	None	RD	RD	8'b0000_0011
(7) 111	None	None	None	None	Any other value

3.1.3 Cryptographic Acceleration and Assurance Module (CAAM)

3.1.3.1 CAAM Overview

CAAM is a cryptographic acceleration device that accelerates symmetric block and stream cipher algorithms, hashing algorithms, and random number generation. It has an integral DMA engine that allows CAAM to fetch its command programs, read input data and write the resulting output.

CAAM works with the SNVS to provide platform assurance features, including support for High Assurance Boot, detection of and response to potential tamper events, and short-term and long-term protection of secret data such as public-private keypairs, Digital Rights Management keys and proprietary software.

CAAM provides the following features:

- Cryptographic acceleration of hashing, encryption and decryption
- Random Number Generation using a true random number generator and a NIST-compliant pseudo random number generator
- Secure Memory for access-controlled protection of critical data, and automatic zeroization in the case of tampering
- Offloading of cryptographic functions via programmable command descriptor language
- Independent queues for job-specific commands and results
- Register Bus Interface used for configuration, control and status
- DMA, including scatter/gather support for data
- Short-term encryption/decryption of cryptographic keys
- Long-term protection of secret data via device-specific encryption keys

3.1.4 Secure Non-Volatile Storage (SNVS)

3.1.4.1 SNVS Overview

SNVS is a hardware device that includes a security state machine and security violation detection circuits that, together with High Assurance Boot software, determine whether the chip is currently in a secure state.

When the security state machine indicates a secure state, the SNVS allows use of special cryptographic keys to decrypt long-term secrets such as public/private keypairs, Digital Rights Management keys and proprietary software. When the SNVS detects a potential security violation, such as a tamper alert, the SNVS sends an interrupt to alert the Operating System of the event. The SNVS also includes a general purpose real-time counter.

The SNVS includes the following features:

- Security State Machine driven by High Assurance Boot software and security violation detection circuits
- Master Key Control that protects the integrity and secrecy of the Master Key (OTPMK) stored in fuses
- Security violation indicators that detect JTAG events, power glitches, Master Key ECC check failure, and HAB/software-reported and hardware-reported security violations
- 256-bit Zeroizable Master Key that can be automatically erased in the event of a security breach
- Secure Realtime Counter that can continue running by coincell power domain when main chip power is off
- Non-volatile Monotonic Counter used to protect against “roll-back” attacks
- Non-volatile General Purpose Register can be used to store a 32-bit value, persistant across power cycles
- Non-Secure Real Time Counter with programmable alarm and periodic interrupt to wakeup system

3.1.4.2 Tamper Detection

Tamper Detection is a special mechanism provided through a chip pin to signal when the device encounters unauthorized opening or tampering.

When not in use, the Tamper Detection signal is pulled-down internally. In case of use, it should be connected to a Tamper Detection contact in a target system (Normally closed, pulled-up to the VDD_SNVS_IN).

An always-ON power supply (coincell battery) should be present in the system. If the tamper detection feature is enabled by software then opening of the tamper contact:

- Activate security related hardware (e.g. automatic and immediate erasure of the Zeroizable Master Key and deny access and erase secure memory contents)

3.1.5 High-Assurance Boot (HAB)

The HAB, which is the high-assurance boot feature in the system boot ROM, detects and prevents the execution of unauthorized software (malware) during the boot sequence.

When the unauthorized software is permitted to gain control of the boot sequence, it can be used for a variety of goals, such as exposing stored secrets; circumventing access controls to sensitive data, services, or networks, or for repurposing the platform. The unauthorized software can enter the platform during upgrades or reprovisioning, or when booting from the USB connections or removable devices.

The HAB protects against unauthorized software by:

- Using digital signatures to recognize the authentic software. This enables you to boot the device to a known initial state and run the software signed by the device manufacturer.

3.1.6 RDC Overview

The Resource Domain Controller (RDC) provides robust support for the isolation of destination memory mapped locations such as peripherals and memory to a single core, a bus master, or set of cores and bus masters.

Many of today's processors have multiple cores for increased performance and flexibility. In some cases, the cores serve different functions (e.g. user level applications versus real time machine control) and in such cases the software for each core may be developed by different providers.

Without careful collaboration between the two operating systems inadvertent malfunction or degradation in performance may result. Similarly, malware present on one core should not be able to affect the operating conditions of the other domain. The RDC provides isolation to allow for more robust and secure operation on the chip. Details can be found in the RDC chapter.

3.1.7 System JTAG Controller (SJC)

The JTAG port provides debug access to hardware blocks, including the ARM processor and the system bus. This enables program control and manipulation as well as visibility to the chip peripherals and memory.

The JTAG port must be accessible during initial platform development, manufacturing tests, and general troubleshooting. Given its capabilities, JTAG manipulation is a known attack vector for accessing sensitive data and gaining control over software execution. The System JTAG Controller (SJC) protects against the whole range of attacks based on unauthorized JTAG manipulation. It also provides a JTAG port that conforms to the IEEE 1149.1 and IEEE 1149.6 (AC) standards for BSR (boundary-scan) testing.

The SJC provides these security levels:

- The JTAG Disabled-JTAG use is permanently blocked.
- The No-Debug-All security sensitive JTAG features are permanently blocked.
- The Secure JTAG-JTAG use is restricted (as in the No-Debug level) unless a secret-key challenge/response protocol is successfully executed.
- The JTAG Enabled-JTAG use is unrestricted.

The security levels are selected via the e-fuse configuration.

3.2 Resource Domain Controller (RDC)

3.2.1 Overview

The Resource Domain Controller (RDC) provides robust support for the isolation of destination memory mapped locations such as peripherals and memory to a single core, a bus master, or set of cores and bus masters.

Many of today's processors have multiple cores for increased performance and flexibility. In some cases, the cores serve different functions (e.g. user level applications versus real time machine control) and in such cases the software for each core may be developed by different providers.

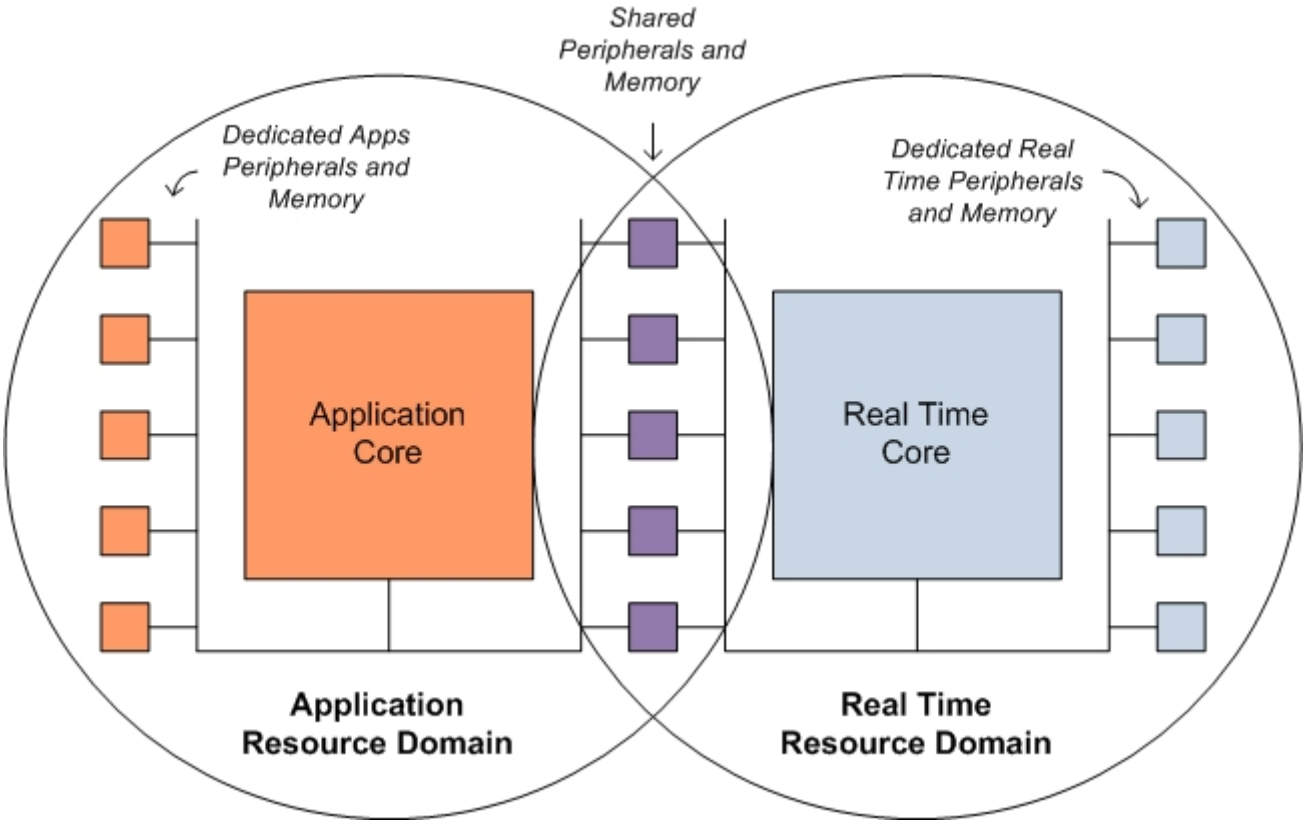


Figure 3-1. Dedicated and Shared Peripherals

For efficiency reasons the code on the cores may share chip resources such as peripherals and memory. The sharing of chip resources between the somewhat independent processing domains allows for the opportunity of data collisions where information stored in peripherals or memory by a process on one core is overwritten by software running on another core. Without careful collaboration between the two operating systems inadvertent malfunction or degradation in performance may result.

The RDC provides a mechanism to allow boot time configuration code to establish resource domains by assigning cores, bus masters, peripherals and memory regions to domain identifiers. Once configured, bus transactions are monitored to restrict accesses initiated by cores and bus masters to their respective peripherals and memory.

For shared peripherals, the RDC provides a semaphore-based locking mechanism to provide for temporary exclusivity while the domain software uses the peripheral. Once the software of one domain has finished the task and finished with the peripheral then it may release the semaphore making the peripheral available to the other domain.

3.2.1.1 Features

Resource domain subsystem has the following features:

- Assignment of cores, bus masters, peripherals, and memory regions to a resource domain
- Fixed memory resolution of 128 Bytes for small address spaces and 4 KB for large address spaces
- Four resource domain identifiers
- Memory read/write access controls for each resource domain and region
- Optional semaphore-based, hardware-enforced exclusive access of shared peripherals to a resource domain
- Prioritized access permissions for overlapping memory regions
- Automatic restoration of resource domain access permissions to memory regions in the power-down domain

3.2.2 Functional Description

The RDC is the central location for creation of isolated resource domains and for the enablement of semaphore-based access also known as “safe sharing”. Configuration software assigns one of four resource domain identifiers to each core and bus master, and allocates each memory region and peripheral to one or more resource domains.

Memory Read or Write access privileges for each resource domain are declared for each memory region. In addition, the software configuration determines which shared peripherals (those peripherals allocated to more than one domain) require safe sharing by setting the semaphore-required configuration for each peripheral.

The RDC configuration information is sent to the fabric ports, memories gaskets, semaphore controller, and peripherals to control access based on domain assignments. The fabric uses the domain identifier associated with each port to include this information along with the bus transaction. When the slave gasket encounters a bus transaction it makes a comparison of the transaction domain ID to the RDC-provided list of allowed domains. If the transaction domain ID is on the list then access may be permitted.

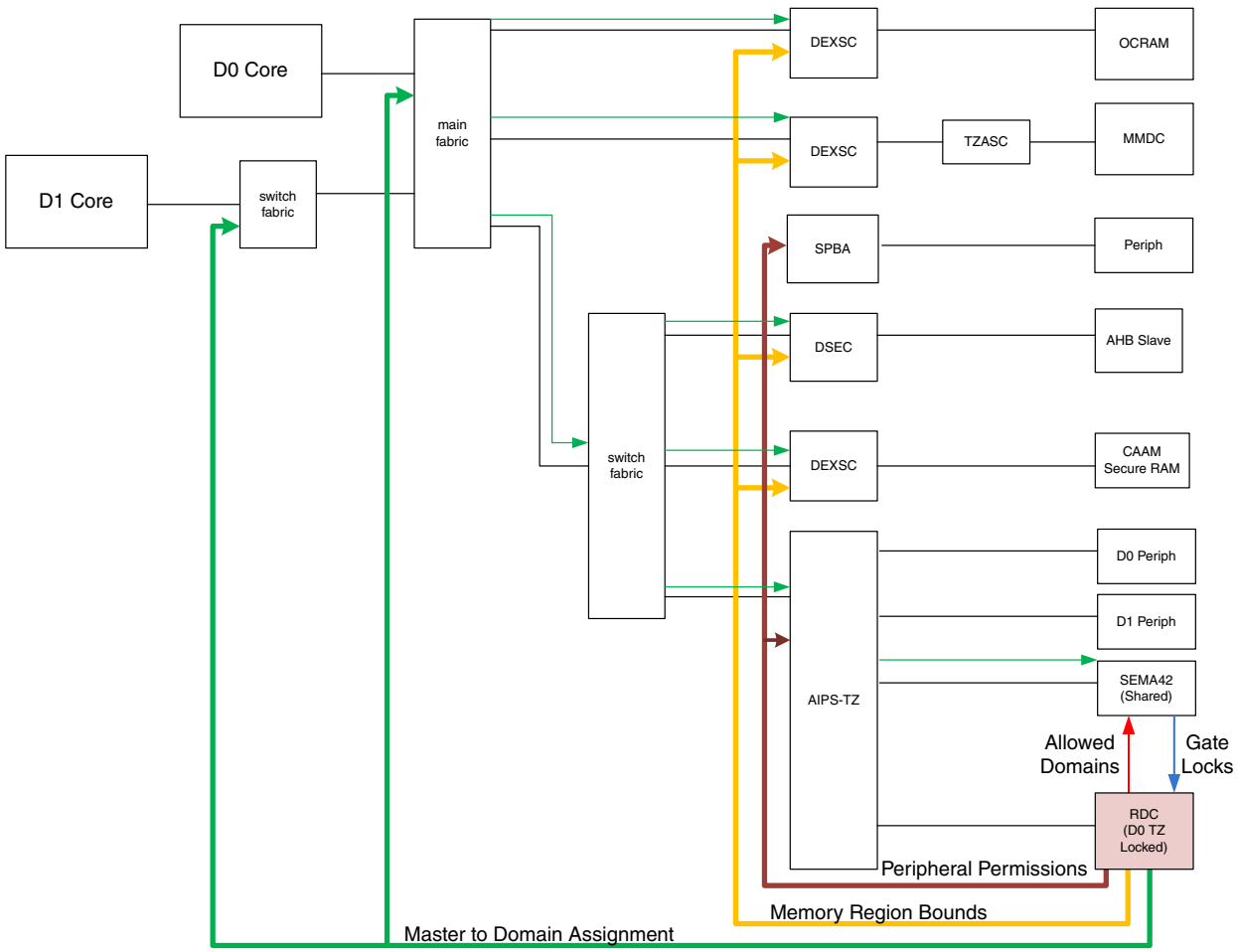


Figure 3-2. Example RDC Connections

For shared peripherals, RDC permits more than one domain access to a single peripheral. RDC also provides three ways to control synchronized use of shared peripherals. These methods include hardware-enforced synchronization, software-based semaphores, or no synchronization. The latter may be suitable for well-tuned multi-core operating systems that handle synchronization in the core platform, for instance.

For hardware-enforced synchronization, also known as "safe sharing", ownership of the peripheral must be claimed in the semaphore controller before access is allowed to the shared peripheral. The "semaphore required" bit (SREQ) is set in the PDAP register corresponding to the shared peripheral which causes the RDC to require that a semaphore is obtained by a domain before access by that domain to the shared peripheral is allowed. During the time that the domain has the semaphore in possession its bus masters have exclusive access to the peripheral.

When the semaphore is released then no domain masters have access until the semaphore is obtained again. When the SREQ is set, RDC does not allow masters to obtain semaphores of peripherals to which it is not allocated; the master must have designated access in the D-registers of the corresponding PDAP register (e.g. D3R bit set for Domain 3 access of the shared peripheral). There is a one-to-one mapping between the semaphore controller gate and the resource domain controller peripheral. The mapping of PDAP registers and peripherals can be found in the Peripheral Map section of the RDC chapter.

3.2.2.1 Domain ID

The RDC provides for an isolation of domain resources by use of a identifier called the Domain ID (DID). A core and its resources including memory, bus masters, and peripherals are all associated with a single DID. When software or a DMA attempts to access a peripheral or memory, the corresponding bus transaction includes the DID along with the other bus control information such as Read, Write, and privilege mode.

3.2.2.2 Resource Assignment

The RDC allows assignment of peripherals and memories to one or more domains while each bus master or core is placed in one of four domains. The masters are assigned a domain in the MDA register. A peripheral is given R/W access permissions to each domain in the PDAP register. Memory regions are bound by address space in start and end registers, the MRSA and MREA. Each memory region is assigned one or more allowed domains and R/W permissions in the MRC control register. Memory regions must be enabled before the permissions are active. Otherwise the permissions are not restricted.

The RDC itself should be isolated to ensure that only a trustworthy resource manager can configure the RDC registers. This process may either be present initially, during secure boot, or during the runtime in the secure world, for example. If the operating system does not support a runtime trusted execution then during the secure boot process the RDC configuration can be locked to prevent further modification after the operating systems are running.

NOTE

The CCM supports multicore awareness based on resource domain assignments programmed into the RDC. Refer to the CCM chapter regarding the relationship between core resource domains and their respective CCM resources. Failing to follow the proper sequence when updating the resource domain

assignments of the core can result in clocks being inadvertently gated.

3.2.2.3 Safe Sharing

For shared peripherals, the RDC can be configured to require a domain to obtain a semaphore lock before access to the peripheral is allowed. This feature helps prevent collisions from processes on separate cores that may want to use the same peripheral at the same time. The RDC sends a list of eligible domains to the semaphore module for each gate/peripheral. The eligible domains are those that are set in the peripheral domain access permissions (PDAP) registers. There is a one-to-one correspondence between semaphore gates and peripherals so each gate in the semaphore block represents a peripheral. The RDC receives semaphore locks from the hardware semaphore module (SEMA42). A semaphore lock is acquired when a core or bus master from a given domain requests a lock for a particular gate. The semaphore module compares the request's domain ID against the list of eligible domain IDs. If the domain ID is on the list and the lock is available then the lock is set and a signal is sent back to the RDC module indicating a lock has been acquired for a particular gate and to which domain ID the lock belongs. The RDC then restricts access to the corresponding peripheral to only transactions originating from the domain that has the lock. Another domain, though on the shared list to access the peripheral, must then wait until the lock is released before acquiring the lock and gaining access to the peripheral. To enable this feature of hardware enforcement for the semaphore locks, the SREQ bit is set in the RDC resource register.

If the SREQ is set, then when a process determines it needs a shared peripheral, it must first lock the resource in the semaphore module. Once the resource is locked, the semaphore module sends a signal to the RDC indicating the domain has access to the resource. The RDC will then set the access permissions to allow that domain access to the peripheral.

For a domain to acquire a lock on a peripheral, the domain must have been assigned to the peripheral in the RDC Peripheral Domain Access Permissions register (PDAP). The semaphore module only allows safe-sharing locks for those domains that are assigned to the peripheral. The semaphore module does not consider the access type (Read or Write) when allowing domains to acquire locks.

The SEMA42 module implements hardware-enforced semaphores as an IPS-mapped slave peripheral device. The feature set includes:

- Module definition supporting 64 hardware-enforced gates in a multi-processor configuration, where up to 15 processors can be supported; cp*X* is meant to represent core processor *X*
  - Gates appear as an *n*-entry byte-size array with read and write accesses (*n* = 16, 32, 64).
    - Processors lock gates by writing "Master_index" to the appropriate gate and must read back the gate value to verify the lock operation was successful. The Master_index value for the processors can be found in Table 14-7. Also note that after locking, the gate register contains the master_id value of the locking processor (in bits [3:0]), and also the value of the locking domain (in bits [5:4]).
    - Once locked, the gate is unlocked by a write of zeroes from the locking processor.
    - The number of implemented gates is specified by a hardware configuration define.
  - Each hardware gate appears as a 16-state, 4-bit state machine.
    - 16-state implementation

      if gate = 0x0, then state = unlocked
      if gate = 0x1, then state = locked by processor (master_index) 0
      if gate = 0x2, then state = locked by processor (master_index) 1
      …
      if gate = 0xF, then state = locked by processor (master_index) 14
    - Uses the logical bus master number (master_index) as a reference attribute plus the specified data patterns to validate all write operations.
    - Once locked, the gate can (and must) be unlocked by a write of zeroes from the locking processor.
  - Secure reset mechanisms are supported to clear the contents of individual gates, as well as a clear_all capability.
- Memory-mapped IPS slave peripheral platform module
  - Interface to the IPS bus for programming-model accesses

3.2.2.4 Resource Domain Control and Security Considerations

Conceptually, the RDC configuration is independent of the processor privilege mode and security domain. It is intended to allow for isolation between core processing environments to prevent collisions and increase reliability. Access between resource domains is mutually exclusive and each domain should be in control of its own privilege modes and access rights.

However, it is important to realize multi-core processors may have a multiple resource domains but only one overarching security domain. Chip security controls reside in one resource domain. In this configuration, a domain can affect at least one level of access privileges in the other domain. This may be acceptable but clarity and care is needed to ensure expected functionality.

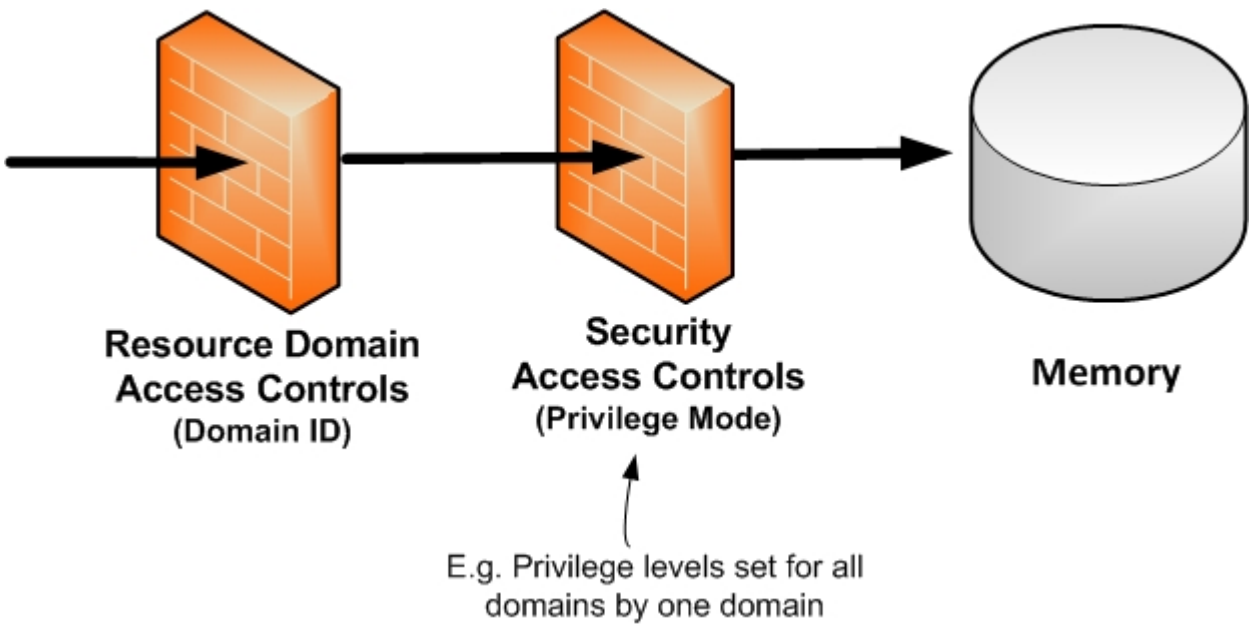


Figure 3-3. Access Control to Memory

Therefore, access to the security controls should be restricted to the most trustworthy operating mode of the core and privilege levels should be coordinated to ensure that shared peripherals and memory regions are accessible by both cores. For instance, if a memory region is designated for secure accesses then all domain masters that share that region must have secure privileges.

3.2.3 Modes of Operation

The RDC provides access controls to the resource domain subsystem. When the device is in a low power mode then some memory regions in the subsystem may be powered off. RDC responds to the impacted memory regions by automatically reconfiguring the memory regions once power returns and blocking access to those memory regions until the reconfiguration process is complete.

3.2.3.1 Low Power Modes

The RDC loads configuration information for memory regions (MRSA, MRSE, MRC) into access control mechanisms (gaskets) at the memory interface. The location of this configuration information may reside inside power domains that lose power during sleep modes for energy savings. To restore configuration information upon return from sleep mode, the RDC receives a global power control signal indicating power is restored. The RDC then automatically reconfigures the memory regions with the configuration information.

During reconfiguration, access is blocked to the previously powered down memories. When the RDC completes reconfiguration it issues an interrupt and allows access to the memory regions. Only the powered down regions are blocked during the reconfiguration. Memory regions in the "always-on" power domain (still powered during sleep mode) remain available according to the programmed access rights. If no memory regions were enabled then the powered down regions are available immediately when power is restored.

The figure below shows the Global Power Control signal which RDC uses to invalidate the configuration upon deassertion and to restore the configuration when re-asserted. The configuration is valid and bus transactions allowed once the memory regions have been restored.

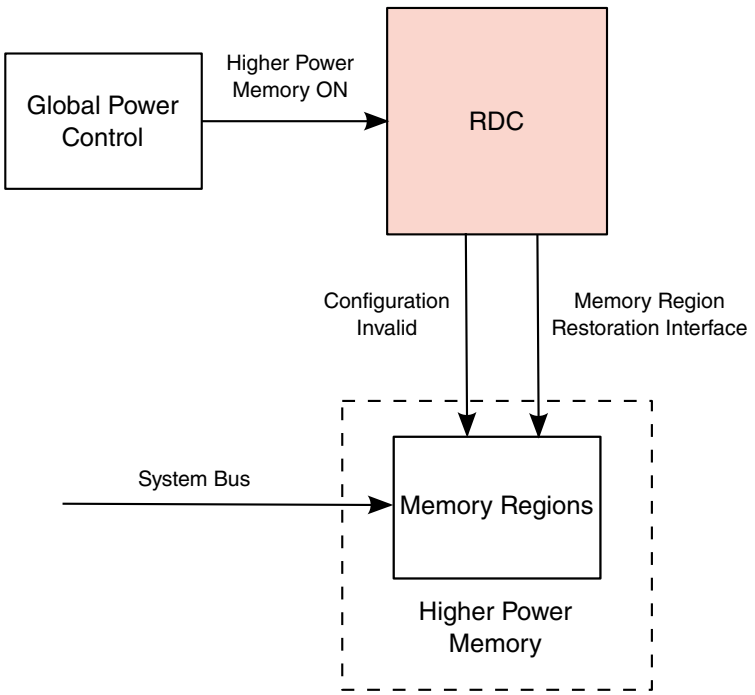


Figure 3-4. Memory Restoration Signaling

3.2.4 Programming Interface

This section provides product specific details describing the mapping of resources - peripherals, bus masters, and memory regions - to corresponding resource domain controls RDC registers.

The RDC and RDC_SEMA42 register maps are combined in this chapter. The base address for the one RDC map and two SEMA42 maps are each separated by 4KB. While there are two SEMA42 submodules and therefore two sets of SEMA42 registers, this chapter describes one. Please refer to the peripheral memory map for the base addresses of the RDC and SEMA42 modules.

3.2.4.1 Master Assignment Registers

Table 3-2. Master Assignment Mapping

Master	RDC MDA register
A7 Core 0	RDC_MDA0
A7 Core 1	RDC_MDA0

Table continues on the next page...

Table 3-2. Master Assignment Mapping (continued)

Master	RDC MDA register
M4 Core	RDC_MDA1
-	RDC_MDA2
CSI	RDC_MDA3
-	RDC_MDA4
LCDIF	RDC_MDA5
Display Port	RDC_MDA6
PXP	RDC_MDA7
Coresight	RDC_MDA8
DAP	RDC_MDA9
CAAM	RDC_MDA10
SDMA (peripheral DMA port)	RDC_MDA11
SDMA (burst DMA port)	RDC_MDA12
APBHDMA	RDC_MDA13
RAWNAND	RDC_MDA14
uSDHC1	RDC_MDA15
uSDHC2	RDC_MDA16
uSDHC3	RDC_MDA17
NC	RDC_MDA18
USB	RDC_MDA19
NC	RDC_MDA20
Test Port	RDC_MDA21
ENET1 TX	RDC_MDA22
ENET1 RX	RDC_MDA23
-	RDC_MDA24
-	RDC_MDA25
SDMA port	RDC_MDA26

3.2.4.2 Peripheral Mapping

Each peripheral has a corresponding resource domain assignment register in the RDC and semaphore lock register in the RDC_SEMA42 module. The following table shows allocation of the RDC PDAP and RDC_SEMA4 GATE registers for peripheral resource domain assignment.

NOTE

Access control of the RDC registers can be programmed using the respective PDAP register. The default setting of the PDAP register for the RDC allows access from all domains. Use

caution when restricting access of the RDC registers to avoid conditions where access to the RDC registers is needed but no master is assigned to a domain with access rights to the RDC.

Table 3-3. RDC Peripheral Mapping

Peripheral	RDC PDAP register	RDC_SEMA42 block/gate register
GPIO1	RDC_PDAP0	SEMA42 B1/G0
GPIO2	RDC_PDAP1	SEMA42 B1/G1
GPIO3	RDC_PDAP2	SEMA42 B1/G2
GPIO4	RDC_PDAP3	SEMA42 B1/G3
GPIO5	RDC_PDAP4	SEMA42 B1/G4
GPIO6	RDC_PDAP5	SEMA42 B1/G5
GPIO7	RDC_PDAP6	SEMA42 B1/G6
IOMUXC_LPSR_GPR	RDC_PDAP7	SEMA42 B1/G7
WDOG1	RDC_PDAP8	SEMA42 B1/G8
WDOG2	RDC_PDAP9	SEMA42 B1/G9
WDOG3	RDC_PDAP10	SEMA42 B1/G10
WDOG4	RDC_PDAP11	SEMA42 B1/G11
IOMUXC_LPSR	RDC_PDAP12	SEMA42 B1/G12
GPT1	RDC_PDAP13	SEMA42 B1/G13
GPT2	RDC_PDAP14	SEMA42 B1/G14
GPT3	RDC_PDAP15	SEMA42 B1/G15
GPT4	RDC_PDAP16	SEMA42 B1/G16
ROMCP	RDC_PDAP17	SEMA42 B1/G17
KPP	RDC_PDAP18	SEMA42 B1/G18
IOMUXC	RDC_PDAP19	SEMA42 B1/G19
IOMUXC_GPR	RDC_PDAP20	SEMA42 B1/G20
OCOTP_CTRL	RDC_PDAP21	SEMA42 B1/G21
ANATOP_DIG	RDC_PDAP22	SEMA42 B1/G22
SNVS HP	RDC_PDAP23	SEMA42 B1/G23
CCM	RDC_PDAP24	SEMA42 B1/G24
SRC	RDC_PDAP25	SEMA42 B1/G25
GPC	RDC_PDAP26	SEMA42 B1/G26
SEMAPHORE1	RDC_PDAP27	SEMA42 B1/G27
SEMAPHORE2	RDC_PDAP28	SEMA42 B1/G28
RDC	RDC_PDAP29	SEMA42 B1/G29
CSU	RDC_PDAP30	SEMA42 B1/G30
Reserved	RDC_PDAP31	SEMA42 B1/G31
RESERVED	RDC_PDAP32	SEMA42 B1/G32
ADC1 WRAPPER	RDC_PDAP33	SEMA42 B1/G33
ADC2 WRAPPER	RDC_PDAP34	SEMA42 B1/G34
eCSPI4	RDC_PDAP35	SEMA42 B1/G35

Table continues on the next page...

Table 3-3. RDC Peripheral Mapping (continued)

Peripheral	RDC PDAP register	RDC_SEMA42 block/gate register
Flex Timer 1	RDC_PDAP36	SEMA42 B1/G36
Flex Timer 2	RDC_PDAP37	SEMA42 B1/G37
PWM1	RDC_PDAP38	SEMA42 B1/G38
PWM2	RDC_PDAP39	SEMA42 B1/G39
PWM3	RDC_PDAP40	SEMA42 B1/G40
PWM4	RDC_PDAP41	SEMA42 B1/G41
System Counter Read	RDC_PDAP42	SEMA42 B1/G42
System Counter Compare	RDC_PDAP43	SEMA42 B1/G43
System Counter Control	RDC_PDAP44	SEMA42 B1/G44
Reserved	RDC_PDAP45	SEMA42 B1/G45
Reserved	RDC_PDAP46	SEMA42 B1/G46
Reserved	RDC_PDAP47	SEMA42 B1/G47
PXP	RDC_PDAP48	SEMA42 B1/G48
CSI	RDC_PDAP49	SEMA42 B1/G49
Reserved	RDC_PDAP50	SEMA42 B1/G50
LCDIF	RDC_PDAP51	SEMA42 B1/G51
Reserved	RDC_PDAP52	SEMA42 B1/G52
MIPI CSI	RDC_PDAP53	SEMA42 B1/G53
MIPI DSI	RDC_PDAP54	SEMA42 B1/G54
Reserved	RDC_PDAP55	SEMA42 B1/G55
TZASC	RDC_PDAP56	SEMA42 B1/G56
DDR PHY	RDC_PDAP57	SEMA42 B1/G57
DDRC	RDC_PDAP58	SEMA42 B1/G58
Reserved	RDC_PDAP59	SEMA42 B1/G59
PERFMON1	RDC_PDAP60	SEMA42 B1/G60
PERFMON2	RDC_PDAP61	SEMA42 B1/G61
AXI DEBUG MON	RDC_PDAP62	SEMA42 B1/G62
QoSC	RDC_PDAP63	SEMA42 B1/G63
FlexCAN1	RDC_PDAP64	SEMA42 B2/G0
FlexCAN2	RDC_PDAP65	SEMA42 B2/G1
I2C1	RDC_PDAP66	SEMA42 B2/G2
I2C2	RDC_PDAP67	SEMA42 B2/G3
I2C3	RDC_PDAP68	SEMA42 B2/G4
I2C4	RDC_PDAP69	SEMA42 B2/G5
UART4	RDC_PDAP70	SEMA42 B2/G6
UART5	RDC_PDAP71	SEMA42 B2/G7
UART6	RDC_PDAP72	SEMA42 B2/G8
UART7	RDC_PDAP73	SEMA42 B2/G9
MU - A	RDC_PDAP74	SEMA42 B2/G10

Table continues on the next page...

Table 3-3. RDC Peripheral Mapping (continued)

Peripheral	RDC PDAP register	RDC_SEMA42 block/gate register
MU - B	RDC_PDAP75	SEMA42 B2/G11
SEMAPHORE - HS	RDC_PDAP76	SEMA42 B2/G12
USB PL301	RDC_PDAP77	SEMA42 B2/G13
Reserved	RDC_PDAP78	SEMA42 B2/G14
Reserved	RDC_PDAP79	SEMA42 B2/G15
Reserved	RDC_PDAP80	SEMA42 B2/G16
USB1 (OTG1)	RDC_PDAP81	SEMA42 B2/G17
Reserved	RDC_PDAP82	SEMA42 B2/G18
USB3 (HOST)	RDC_PDAP83	SEMA42 B2/G19
uSDHC1	RDC_PDAP84	SEMA42 B2/G20
uSDHC2	RDC_PDAP85	SEMA42 B2/G21
uSDHC3	RDC_PDAP86	SEMA42 B2/G22
Reserved	RDC_PDAP87	SEMA42 B2/G23
Reserved	RDC_PDAP88	SEMA42 B2/G24
SIM1	RDC_PDAP89	SEMA42 B2/G25
SIM2	RDC_PDAP90	SEMA42 B2/G26
QSPI	RDC_PDAP91	SEMA42 B2/G27
WEIM	RDC_PDAP92	SEMA42 B2/G28
SDMA	RDC_PDAP93	SEMA42 B2/G29
ENET1	RDC_PDAP94	SEMA42 B2/G30
Reserved	RDC_PDAP95	SEMA42 B2/G31
Reserved for SDMA	RDC_PDAP96	SEMA42 B2/G32
Reserved	RDC_PDAP97	SEMA42 B2/G33
eCSPI1	RDC_PDAP98	SEMA42 B2/G34
eCSPI2	RDC_PDAP99	SEMA42 B2/G35
eCSPI3	RDC_PDAP100	SEMA42 B2/G36
Reserved	RDC_PDAP101	SEMA42 B2/G37
UART1	RDC_PDAP102	SEMA42 B2/G38
UART2	RDC_PDAP105	SEMA42 B2/G39
UART3	RDC_PDAP104	SEMA42 B2/G40
Reserved for SDMA	RDC_PDAP103	SEMA42 B2/G41
SAI1	RDC_PDAP106	SEMA42 B2/G42
SAI2	RDC_PDAP107	SEMA42 B2/G43
SAI3	RDC_PDAP108	SEMA42 B2/G44
Reserved	RDC_PDAP109	SEMA42 B2/G45
Reserved for SDMA	RDC_PDAP110	SEMA42 B2/G46
SPBA	RDC_PDAP111	SEMA42 B2/G47
ULT1 / DAP	RDC_PDAP112	SEMA42 B2/G48
Reserved	RDC_PDAP113	SEMA42 B2/G49

Table continues on the next page...

Table 3-3. RDC Peripheral Mapping (continued)

Peripheral	RDC PDAP register	RDC_SEMA42 block/gate register
Reserved	RDC_PDAP114	SEMA42 B2/G50
reserved	RDC_PDAP115	SEMA42 B2/G51
CAAM	RDC_PDAP116	SEMA42 B2/G52
Reserved	RDC_PDAP117	SEMA42 B2/G53

3.2.4.3 Memory Region Map

The number of memories with domain isolation support varies per device. The number of memory regions for a particular memory and the size of those regions varies per memory gasket. Each region of memory has a set of registers to define the boundaries of the region based on start and end addresses, a control register to set the domain access permissions and enable the region, and a status register to determin if access was denied to a region.

For this device, refer to the table below to determine the memories with domain support, the number of regions for each memory, the region resolution, the identifying numbers for the sets of memory region registers, and the addresses of the RDC registers to access the sets of Memory Region registers.

Table 3-4. Memory Region Mapping

Memory/Port	Number of Regions	Region Resolution	Memory Region Register Set Number (e.g. MRSA, MREA, MRC, MRVS)	Register Address Range
MMDC	8	4 KB	0-7	0x800-0x87C
QSPI	8	4 KB	8-15	0x880-0x8FC
WEIM	8	4 KB	16-23	0x900-0x97C
PCIe	8	4 KB	24-31	0x980-0x9FC
OCRAM	5	128 B	32-36	0xA00-0xA4C
OCRAM_S	5	128 B	37-41	0xA50-0xA9C
OCRAM_GP	5	128 B	42-46	0xAA0-0xAEC
OCRAM_PXP	5	128 B	47-51	0xAF0-0xB3C

3.2.5 RDC Memory Map/Register Definition

RDC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303D_0000	Version Information (RDC_VIR)	32	R	0376_E204h	3.2.5.1/224
303D_0024	Status (RDC_STAT)	32	R/W	0000_0100h	3.2.5.2/225
303D_0028	Interrupt and Control (RDC_INTCTRL)	32	R/W	0000_0000h	3.2.5.3/226
303D_002C	Interrupt Status (RDC_INTSTAT)	32	R/W	See section	3.2.5.4/226
303D_0200	Master Domain Assignment (RDC_MDA0)	32	R/W	0000_0000h	3.2.5.5/227
303D_0204	Master Domain Assignment (RDC_MDA1)	32	R/W	0000_0000h	3.2.5.5/227
303D_0208	Master Domain Assignment (RDC_MDA2)	32	R/W	0000_0000h	3.2.5.5/227
303D_020C	Master Domain Assignment (RDC_MDA3)	32	R/W	0000_0000h	3.2.5.5/227
303D_0210	Master Domain Assignment (RDC_MDA4)	32	R/W	0000_0000h	3.2.5.5/227
303D_0214	Master Domain Assignment (RDC_MDA5)	32	R/W	0000_0000h	3.2.5.5/227
303D_0218	Master Domain Assignment (RDC_MDA6)	32	R/W	0000_0000h	3.2.5.5/227
303D_021C	Master Domain Assignment (RDC_MDA7)	32	R/W	0000_0000h	3.2.5.5/227
303D_0220	Master Domain Assignment (RDC_MDA8)	32	R/W	0000_0000h	3.2.5.5/227
303D_0224	Master Domain Assignment (RDC_MDA9)	32	R/W	0000_0000h	3.2.5.5/227
303D_0228	Master Domain Assignment (RDC_MDA10)	32	R/W	0000_0000h	3.2.5.5/227
303D_022C	Master Domain Assignment (RDC_MDA11)	32	R/W	0000_0000h	3.2.5.5/227
303D_0230	Master Domain Assignment (RDC_MDA12)	32	R/W	0000_0000h	3.2.5.5/227
303D_0234	Master Domain Assignment (RDC_MDA13)	32	R/W	0000_0000h	3.2.5.5/227
303D_0238	Master Domain Assignment (RDC_MDA14)	32	R/W	0000_0000h	3.2.5.5/227
303D_023C	Master Domain Assignment (RDC_MDA15)	32	R/W	0000_0000h	3.2.5.5/227
303D_0240	Master Domain Assignment (RDC_MDA16)	32	R/W	0000_0000h	3.2.5.5/227
303D_0244	Master Domain Assignment (RDC_MDA17)	32	R/W	0000_0000h	3.2.5.5/227
303D_0248	Master Domain Assignment (RDC_MDA18)	32	R/W	0000_0000h	3.2.5.5/227
303D_024C	Master Domain Assignment (RDC_MDA19)	32	R/W	0000_0000h	3.2.5.5/227
303D_0250	Master Domain Assignment (RDC_MDA20)	32	R/W	0000_0000h	3.2.5.5/227
303D_0254	Master Domain Assignment (RDC_MDA21)	32	R/W	0000_0000h	3.2.5.5/227
303D_0258	Master Domain Assignment (RDC_MDA22)	32	R/W	0000_0000h	3.2.5.5/227
303D_025C	Master Domain Assignment (RDC_MDA23)	32	R/W	0000_0000h	3.2.5.5/227
303D_0260	Master Domain Assignment (RDC_MDA24)	32	R/W	0000_0000h	3.2.5.5/227
303D_0264	Master Domain Assignment (RDC_MDA25)	32	R/W	0000_0000h	3.2.5.5/227
303D_0268	Master Domain Assignment (RDC_MDA26)	32	R/W	0000_0000h	3.2.5.5/227
303D_0400	Peripheral Domain Access Permissions (RDC_PDAP0)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0404	Peripheral Domain Access Permissions (RDC_PDAP1)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0408	Peripheral Domain Access Permissions (RDC_PDAP2)	32	R/W	0000_00FFh	3.2.5.6/228
303D_040C	Peripheral Domain Access Permissions (RDC_PDAP3)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0410	Peripheral Domain Access Permissions (RDC_PDAP4)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0414	Peripheral Domain Access Permissions (RDC_PDAP5)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0418	Peripheral Domain Access Permissions (RDC_PDAP6)	32	R/W	0000_00FFh	3.2.5.6/228

Table continues on the next page...

RDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303D_041C	Peripheral Domain Access Permissions (RDC_PDAP7)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0420	Peripheral Domain Access Permissions (RDC_PDAP8)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0424	Peripheral Domain Access Permissions (RDC_PDAP9)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0428	Peripheral Domain Access Permissions (RDC_PDAP10)	32	R/W	0000_00FFh	3.2.5.6/228
303D_042C	Peripheral Domain Access Permissions (RDC_PDAP11)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0430	Peripheral Domain Access Permissions (RDC_PDAP12)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0434	Peripheral Domain Access Permissions (RDC_PDAP13)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0438	Peripheral Domain Access Permissions (RDC_PDAP14)	32	R/W	0000_00FFh	3.2.5.6/228
303D_043C	Peripheral Domain Access Permissions (RDC_PDAP15)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0440	Peripheral Domain Access Permissions (RDC_PDAP16)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0444	Peripheral Domain Access Permissions (RDC_PDAP17)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0448	Peripheral Domain Access Permissions (RDC_PDAP18)	32	R/W	0000_00FFh	3.2.5.6/228
303D_044C	Peripheral Domain Access Permissions (RDC_PDAP19)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0450	Peripheral Domain Access Permissions (RDC_PDAP20)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0454	Peripheral Domain Access Permissions (RDC_PDAP21)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0458	Peripheral Domain Access Permissions (RDC_PDAP22)	32	R/W	0000_00FFh	3.2.5.6/228
303D_045C	Peripheral Domain Access Permissions (RDC_PDAP23)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0460	Peripheral Domain Access Permissions (RDC_PDAP24)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0464	Peripheral Domain Access Permissions (RDC_PDAP25)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0468	Peripheral Domain Access Permissions (RDC_PDAP26)	32	R/W	0000_00FFh	3.2.5.6/228
303D_046C	Peripheral Domain Access Permissions (RDC_PDAP27)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0470	Peripheral Domain Access Permissions (RDC_PDAP28)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0474	Peripheral Domain Access Permissions (RDC_PDAP29)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0478	Peripheral Domain Access Permissions (RDC_PDAP30)	32	R/W	0000_00FFh	3.2.5.6/228
303D_047C	Peripheral Domain Access Permissions (RDC_PDAP31)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0480	Peripheral Domain Access Permissions (RDC_PDAP32)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0484	Peripheral Domain Access Permissions (RDC_PDAP33)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0488	Peripheral Domain Access Permissions (RDC_PDAP34)	32	R/W	0000_00FFh	3.2.5.6/228
303D_048C	Peripheral Domain Access Permissions (RDC_PDAP35)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0490	Peripheral Domain Access Permissions (RDC_PDAP36)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0494	Peripheral Domain Access Permissions (RDC_PDAP37)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0498	Peripheral Domain Access Permissions (RDC_PDAP38)	32	R/W	0000_00FFh	3.2.5.6/228
303D_049C	Peripheral Domain Access Permissions (RDC_PDAP39)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04A0	Peripheral Domain Access Permissions (RDC_PDAP40)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04A4	Peripheral Domain Access Permissions (RDC_PDAP41)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04A8	Peripheral Domain Access Permissions (RDC_PDAP42)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04AC	Peripheral Domain Access Permissions (RDC_PDAP43)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04B0	Peripheral Domain Access Permissions (RDC_PDAP44)	32	R/W	0000_00FFh	3.2.5.6/228

Table continues on the next page...

RDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303D_04B4	Peripheral Domain Access Permissions (RDC_PDAP45)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04B8	Peripheral Domain Access Permissions (RDC_PDAP46)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04BC	Peripheral Domain Access Permissions (RDC_PDAP47)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04C0	Peripheral Domain Access Permissions (RDC_PDAP48)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04C4	Peripheral Domain Access Permissions (RDC_PDAP49)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04C8	Peripheral Domain Access Permissions (RDC_PDAP50)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04CC	Peripheral Domain Access Permissions (RDC_PDAP51)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04D0	Peripheral Domain Access Permissions (RDC_PDAP52)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04D4	Peripheral Domain Access Permissions (RDC_PDAP53)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04D8	Peripheral Domain Access Permissions (RDC_PDAP54)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04DC	Peripheral Domain Access Permissions (RDC_PDAP55)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04E0	Peripheral Domain Access Permissions (RDC_PDAP56)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04E4	Peripheral Domain Access Permissions (RDC_PDAP57)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04E8	Peripheral Domain Access Permissions (RDC_PDAP58)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04EC	Peripheral Domain Access Permissions (RDC_PDAP59)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04F0	Peripheral Domain Access Permissions (RDC_PDAP60)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04F4	Peripheral Domain Access Permissions (RDC_PDAP61)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04F8	Peripheral Domain Access Permissions (RDC_PDAP62)	32	R/W	0000_00FFh	3.2.5.6/228
303D_04FC	Peripheral Domain Access Permissions (RDC_PDAP63)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0500	Peripheral Domain Access Permissions (RDC_PDAP64)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0504	Peripheral Domain Access Permissions (RDC_PDAP65)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0508	Peripheral Domain Access Permissions (RDC_PDAP66)	32	R/W	0000_00FFh	3.2.5.6/228
303D_050C	Peripheral Domain Access Permissions (RDC_PDAP67)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0510	Peripheral Domain Access Permissions (RDC_PDAP68)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0514	Peripheral Domain Access Permissions (RDC_PDAP69)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0518	Peripheral Domain Access Permissions (RDC_PDAP70)	32	R/W	0000_00FFh	3.2.5.6/228
303D_051C	Peripheral Domain Access Permissions (RDC_PDAP71)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0520	Peripheral Domain Access Permissions (RDC_PDAP72)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0524	Peripheral Domain Access Permissions (RDC_PDAP73)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0528	Peripheral Domain Access Permissions (RDC_PDAP74)	32	R/W	0000_00FFh	3.2.5.6/228
303D_052C	Peripheral Domain Access Permissions (RDC_PDAP75)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0530	Peripheral Domain Access Permissions (RDC_PDAP76)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0534	Peripheral Domain Access Permissions (RDC_PDAP77)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0538	Peripheral Domain Access Permissions (RDC_PDAP78)	32	R/W	0000_00FFh	3.2.5.6/228
303D_053C	Peripheral Domain Access Permissions (RDC_PDAP79)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0540	Peripheral Domain Access Permissions (RDC_PDAP80)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0544	Peripheral Domain Access Permissions (RDC_PDAP81)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0548	Peripheral Domain Access Permissions (RDC_PDAP82)	32	R/W	0000_00FFh	3.2.5.6/228

Table continues on the next page...

RDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303D_054C	Peripheral Domain Access Permissions (RDC_PDAP83)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0550	Peripheral Domain Access Permissions (RDC_PDAP84)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0554	Peripheral Domain Access Permissions (RDC_PDAP85)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0558	Peripheral Domain Access Permissions (RDC_PDAP86)	32	R/W	0000_00FFh	3.2.5.6/228
303D_055C	Peripheral Domain Access Permissions (RDC_PDAP87)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0560	Peripheral Domain Access Permissions (RDC_PDAP88)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0564	Peripheral Domain Access Permissions (RDC_PDAP89)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0568	Peripheral Domain Access Permissions (RDC_PDAP90)	32	R/W	0000_00FFh	3.2.5.6/228
303D_056C	Peripheral Domain Access Permissions (RDC_PDAP91)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0570	Peripheral Domain Access Permissions (RDC_PDAP92)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0574	Peripheral Domain Access Permissions (RDC_PDAP93)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0578	Peripheral Domain Access Permissions (RDC_PDAP94)	32	R/W	0000_00FFh	3.2.5.6/228
303D_057C	Peripheral Domain Access Permissions (RDC_PDAP95)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0580	Peripheral Domain Access Permissions (RDC_PDAP96)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0584	Peripheral Domain Access Permissions (RDC_PDAP97)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0588	Peripheral Domain Access Permissions (RDC_PDAP98)	32	R/W	0000_00FFh	3.2.5.6/228
303D_058C	Peripheral Domain Access Permissions (RDC_PDAP99)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0590	Peripheral Domain Access Permissions (RDC_PDAP100)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0594	Peripheral Domain Access Permissions (RDC_PDAP101)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0598	Peripheral Domain Access Permissions (RDC_PDAP102)	32	R/W	0000_00FFh	3.2.5.6/228
303D_059C	Peripheral Domain Access Permissions (RDC_PDAP103)	32	R/W	0000_00FFh	3.2.5.6/228
303D_05A0	Peripheral Domain Access Permissions (RDC_PDAP104)	32	R/W	0000_00FFh	3.2.5.6/228
303D_05A4	Peripheral Domain Access Permissions (RDC_PDAP105)	32	R/W	0000_00FFh	3.2.5.6/228
303D_05A8	Peripheral Domain Access Permissions (RDC_PDAP106)	32	R/W	0000_00FFh	3.2.5.6/228
303D_05AC	Peripheral Domain Access Permissions (RDC_PDAP107)	32	R/W	0000_00FFh	3.2.5.6/228
303D_05B0	Peripheral Domain Access Permissions (RDC_PDAP108)	32	R/W	0000_00FFh	3.2.5.6/228
303D_05B4	Peripheral Domain Access Permissions (RDC_PDAP109)	32	R/W	0000_00FFh	3.2.5.6/228
303D_05B8	Peripheral Domain Access Permissions (RDC_PDAP110)	32	R/W	0000_00FFh	3.2.5.6/228
303D_05BC	Peripheral Domain Access Permissions (RDC_PDAP111)	32	R/W	0000_00FFh	3.2.5.6/228
303D_05C0	Peripheral Domain Access Permissions (RDC_PDAP112)	32	R/W	0000_00FFh	3.2.5.6/228
303D_05C4	Peripheral Domain Access Permissions (RDC_PDAP113)	32	R/W	0000_00FFh	3.2.5.6/228
303D_05C8	Peripheral Domain Access Permissions (RDC_PDAP114)	32	R/W	0000_00FFh	3.2.5.6/228
303D_05CC	Peripheral Domain Access Permissions (RDC_PDAP115)	32	R/W	0000_00FFh	3.2.5.6/228
303D_05D0	Peripheral Domain Access Permissions (RDC_PDAP116)	32	R/W	0000_00FFh	3.2.5.6/228
303D_05D4	Peripheral Domain Access Permissions (RDC_PDAP117)	32	R/W	0000_00FFh	3.2.5.6/228
303D_0800	Memory Region Start Address (RDC_MRSA0)	32	R/W	Undefined	3.2.5.7/229
303D_0804	Memory Region End Address (RDC_MREA0)	32	R/W	Undefined	3.2.5.8/230
303D_0808	Memory Region Control (RDC_MRC0)	32	R/W	0000_00FFh	3.2.5.9/230

Table continues on the next page...

RDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303D_080C	Memory Region Violation Status (RDC_MRVS0)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0810	Memory Region Start Address (RDC_MRSA1)	32	R/W	Undefined	3.2.5.7/229
303D_0814	Memory Region End Address (RDC_MREA1)	32	R/W	Undefined	3.2.5.8/230
303D_0818	Memory Region Control (RDC_MRC1)	32	R/W	0000_00FFh	3.2.5.9/230
303D_081C	Memory Region Violation Status (RDC_MRVS1)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0820	Memory Region Start Address (RDC_MRSA2)	32	R/W	Undefined	3.2.5.7/229
303D_0824	Memory Region End Address (RDC_MREA2)	32	R/W	Undefined	3.2.5.8/230
303D_0828	Memory Region Control (RDC_MRC2)	32	R/W	0000_00FFh	3.2.5.9/230
303D_082C	Memory Region Violation Status (RDC_MRVS2)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0830	Memory Region Start Address (RDC_MRSA3)	32	R/W	Undefined	3.2.5.7/229
303D_0834	Memory Region End Address (RDC_MREA3)	32	R/W	Undefined	3.2.5.8/230
303D_0838	Memory Region Control (RDC_MRC3)	32	R/W	0000_00FFh	3.2.5.9/230
303D_083C	Memory Region Violation Status (RDC_MRVS3)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0840	Memory Region Start Address (RDC_MRSA4)	32	R/W	Undefined	3.2.5.7/229
303D_0844	Memory Region End Address (RDC_MREA4)	32	R/W	Undefined	3.2.5.8/230
303D_0848	Memory Region Control (RDC_MRC4)	32	R/W	0000_00FFh	3.2.5.9/230
303D_084C	Memory Region Violation Status (RDC_MRVS4)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0850	Memory Region Start Address (RDC_MRSA5)	32	R/W	Undefined	3.2.5.7/229
303D_0854	Memory Region End Address (RDC_MREA5)	32	R/W	Undefined	3.2.5.8/230
303D_0858	Memory Region Control (RDC_MRC5)	32	R/W	0000_00FFh	3.2.5.9/230
303D_085C	Memory Region Violation Status (RDC_MRVS5)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0860	Memory Region Start Address (RDC_MRSA6)	32	R/W	Undefined	3.2.5.7/229
303D_0864	Memory Region End Address (RDC_MREA6)	32	R/W	Undefined	3.2.5.8/230
303D_0868	Memory Region Control (RDC_MRC6)	32	R/W	0000_00FFh	3.2.5.9/230
303D_086C	Memory Region Violation Status (RDC_MRVS6)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0870	Memory Region Start Address (RDC_MRSA7)	32	R/W	Undefined	3.2.5.7/229
303D_0874	Memory Region End Address (RDC_MREA7)	32	R/W	Undefined	3.2.5.8/230
303D_0878	Memory Region Control (RDC_MRC7)	32	R/W	0000_00FFh	3.2.5.9/230
303D_087C	Memory Region Violation Status (RDC_MRVS7)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0880	Memory Region Start Address (RDC_MRSA8)	32	R/W	Undefined	3.2.5.7/229
303D_0884	Memory Region End Address (RDC_MREA8)	32	R/W	Undefined	3.2.5.8/230
303D_0888	Memory Region Control (RDC_MRC8)	32	R/W	0000_00FFh	3.2.5.9/230

Table continues on the next page...

RDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303D_088C	Memory Region Violation Status (RDC_MRVS8)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0890	Memory Region Start Address (RDC_MRSA9)	32	R/W	Undefined	3.2.5.7/229
303D_0894	Memory Region End Address (RDC_MREA9)	32	R/W	Undefined	3.2.5.8/230
303D_0898	Memory Region Control (RDC_MRC9)	32	R/W	0000_00FFh	3.2.5.9/230
303D_089C	Memory Region Violation Status (RDC_MRVS9)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_08A0	Memory Region Start Address (RDC_MRSA10)	32	R/W	Undefined	3.2.5.7/229
303D_08A4	Memory Region End Address (RDC_MREA10)	32	R/W	Undefined	3.2.5.8/230
303D_08A8	Memory Region Control (RDC_MRC10)	32	R/W	0000_00FFh	3.2.5.9/230
303D_08AC	Memory Region Violation Status (RDC_MRVS10)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_08B0	Memory Region Start Address (RDC_MRSA11)	32	R/W	Undefined	3.2.5.7/229
303D_08B4	Memory Region End Address (RDC_MREA11)	32	R/W	Undefined	3.2.5.8/230
303D_08B8	Memory Region Control (RDC_MRC11)	32	R/W	0000_00FFh	3.2.5.9/230
303D_08BC	Memory Region Violation Status (RDC_MRVS11)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_08C0	Memory Region Start Address (RDC_MRSA12)	32	R/W	Undefined	3.2.5.7/229
303D_08C4	Memory Region End Address (RDC_MREA12)	32	R/W	Undefined	3.2.5.8/230
303D_08C8	Memory Region Control (RDC_MRC12)	32	R/W	0000_00FFh	3.2.5.9/230
303D_08CC	Memory Region Violation Status (RDC_MRVS12)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_08D0	Memory Region Start Address (RDC_MRSA13)	32	R/W	Undefined	3.2.5.7/229
303D_08D4	Memory Region End Address (RDC_MREA13)	32	R/W	Undefined	3.2.5.8/230
303D_08D8	Memory Region Control (RDC_MRC13)	32	R/W	0000_00FFh	3.2.5.9/230
303D_08DC	Memory Region Violation Status (RDC_MRVS13)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_08E0	Memory Region Start Address (RDC_MRSA14)	32	R/W	Undefined	3.2.5.7/229
303D_08E4	Memory Region End Address (RDC_MREA14)	32	R/W	Undefined	3.2.5.8/230
303D_08E8	Memory Region Control (RDC_MRC14)	32	R/W	0000_00FFh	3.2.5.9/230
303D_08EC	Memory Region Violation Status (RDC_MRVS14)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_08F0	Memory Region Start Address (RDC_MRSA15)	32	R/W	Undefined	3.2.5.7/229
303D_08F4	Memory Region End Address (RDC_MREA15)	32	R/W	Undefined	3.2.5.8/230
303D_08F8	Memory Region Control (RDC_MRC15)	32	R/W	0000_00FFh	3.2.5.9/230
303D_08FC	Memory Region Violation Status (RDC_MRVS15)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0900	Memory Region Start Address (RDC_MRSA16)	32	R/W	Undefined	3.2.5.7/229
303D_0904	Memory Region End Address (RDC_MREA16)	32	R/W	Undefined	3.2.5.8/230
303D_0908	Memory Region Control (RDC_MRC16)	32	R/W	0000_00FFh	3.2.5.9/230

Table continues on the next page...

RDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303D_090C	Memory Region Violation Status (RDC_MRVS16)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0910	Memory Region Start Address (RDC_MRSA17)	32	R/W	Undefined	3.2.5.7/229
303D_0914	Memory Region End Address (RDC_MREA17)	32	R/W	Undefined	3.2.5.8/230
303D_0918	Memory Region Control (RDC_MRC17)	32	R/W	0000_00FFh	3.2.5.9/230
303D_091C	Memory Region Violation Status (RDC_MRVS17)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0920	Memory Region Start Address (RDC_MRSA18)	32	R/W	Undefined	3.2.5.7/229
303D_0924	Memory Region End Address (RDC_MREA18)	32	R/W	Undefined	3.2.5.8/230
303D_0928	Memory Region Control (RDC_MRC18)	32	R/W	0000_00FFh	3.2.5.9/230
303D_092C	Memory Region Violation Status (RDC_MRVS18)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0930	Memory Region Start Address (RDC_MRSA19)	32	R/W	Undefined	3.2.5.7/229
303D_0934	Memory Region End Address (RDC_MREA19)	32	R/W	Undefined	3.2.5.8/230
303D_0938	Memory Region Control (RDC_MRC19)	32	R/W	0000_00FFh	3.2.5.9/230
303D_093C	Memory Region Violation Status (RDC_MRVS19)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0940	Memory Region Start Address (RDC_MRSA20)	32	R/W	Undefined	3.2.5.7/229
303D_0944	Memory Region End Address (RDC_MREA20)	32	R/W	Undefined	3.2.5.8/230
303D_0948	Memory Region Control (RDC_MRC20)	32	R/W	0000_00FFh	3.2.5.9/230
303D_094C	Memory Region Violation Status (RDC_MRVS20)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0950	Memory Region Start Address (RDC_MRSA21)	32	R/W	Undefined	3.2.5.7/229
303D_0954	Memory Region End Address (RDC_MREA21)	32	R/W	Undefined	3.2.5.8/230
303D_0958	Memory Region Control (RDC_MRC21)	32	R/W	0000_00FFh	3.2.5.9/230
303D_095C	Memory Region Violation Status (RDC_MRVS21)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0960	Memory Region Start Address (RDC_MRSA22)	32	R/W	Undefined	3.2.5.7/229
303D_0964	Memory Region End Address (RDC_MREA22)	32	R/W	Undefined	3.2.5.8/230
303D_0968	Memory Region Control (RDC_MRC22)	32	R/W	0000_00FFh	3.2.5.9/230
303D_096C	Memory Region Violation Status (RDC_MRVS22)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0970	Memory Region Start Address (RDC_MRSA23)	32	R/W	Undefined	3.2.5.7/229
303D_0974	Memory Region End Address (RDC_MREA23)	32	R/W	Undefined	3.2.5.8/230
303D_0978	Memory Region Control (RDC_MRC23)	32	R/W	0000_00FFh	3.2.5.9/230
303D_097C	Memory Region Violation Status (RDC_MRVS23)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0980	Memory Region Start Address (RDC_MRSA24)	32	R/W	Undefined	3.2.5.7/229
303D_0984	Memory Region End Address (RDC_MREA24)	32	R/W	Undefined	3.2.5.8/230
303D_0988	Memory Region Control (RDC_MRC24)	32	R/W	0000_00FFh	3.2.5.9/230

Table continues on the next page...

RDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303D_098C	Memory Region Violation Status (RDC_MRVS24)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0990	Memory Region Start Address (RDC_MRSA25)	32	R/W	Undefined	3.2.5.7/229
303D_0994	Memory Region End Address (RDC_MREA25)	32	R/W	Undefined	3.2.5.8/230
303D_0998	Memory Region Control (RDC_MRC25)	32	R/W	0000_00FFh	3.2.5.9/230
303D_099C	Memory Region Violation Status (RDC_MRVS25)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_09A0	Memory Region Start Address (RDC_MRSA26)	32	R/W	Undefined	3.2.5.7/229
303D_09A4	Memory Region End Address (RDC_MREA26)	32	R/W	Undefined	3.2.5.8/230
303D_09A8	Memory Region Control (RDC_MRC26)	32	R/W	0000_00FFh	3.2.5.9/230
303D_09AC	Memory Region Violation Status (RDC_MRVS26)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_09B0	Memory Region Start Address (RDC_MRSA27)	32	R/W	Undefined	3.2.5.7/229
303D_09B4	Memory Region End Address (RDC_MREA27)	32	R/W	Undefined	3.2.5.8/230
303D_09B8	Memory Region Control (RDC_MRC27)	32	R/W	0000_00FFh	3.2.5.9/230
303D_09BC	Memory Region Violation Status (RDC_MRVS27)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_09C0	Memory Region Start Address (RDC_MRSA28)	32	R/W	Undefined	3.2.5.7/229
303D_09C4	Memory Region End Address (RDC_MREA28)	32	R/W	Undefined	3.2.5.8/230
303D_09C8	Memory Region Control (RDC_MRC28)	32	R/W	0000_00FFh	3.2.5.9/230
303D_09CC	Memory Region Violation Status (RDC_MRVS28)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_09D0	Memory Region Start Address (RDC_MRSA29)	32	R/W	Undefined	3.2.5.7/229
303D_09D4	Memory Region End Address (RDC_MREA29)	32	R/W	Undefined	3.2.5.8/230
303D_09D8	Memory Region Control (RDC_MRC29)	32	R/W	0000_00FFh	3.2.5.9/230
303D_09DC	Memory Region Violation Status (RDC_MRVS29)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_09E0	Memory Region Start Address (RDC_MRSA30)	32	R/W	Undefined	3.2.5.7/229
303D_09E4	Memory Region End Address (RDC_MREA30)	32	R/W	Undefined	3.2.5.8/230
303D_09E8	Memory Region Control (RDC_MRC30)	32	R/W	0000_00FFh	3.2.5.9/230
303D_09EC	Memory Region Violation Status (RDC_MRVS30)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_09F0	Memory Region Start Address (RDC_MRSA31)	32	R/W	Undefined	3.2.5.7/229
303D_09F4	Memory Region End Address (RDC_MREA31)	32	R/W	Undefined	3.2.5.8/230
303D_09F8	Memory Region Control (RDC_MRC31)	32	R/W	0000_00FFh	3.2.5.9/230
303D_09FC	Memory Region Violation Status (RDC_MRVS31)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0A00	Memory Region Start Address (RDC_MRSA32)	32	R/W	Undefined	3.2.5.7/229
303D_0A04	Memory Region End Address (RDC_MREA32)	32	R/W	Undefined	3.2.5.8/230
303D_0A08	Memory Region Control (RDC_MRC32)	32	R/W	0000_00FFh	3.2.5.9/230

Table continues on the next page...

RDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303D_0A0C	Memory Region Violation Status (RDC_MRVS32)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0A10	Memory Region Start Address (RDC_MRSA33)	32	R/W	Undefined	3.2.5.7/229
303D_0A14	Memory Region End Address (RDC_MREA33)	32	R/W	Undefined	3.2.5.8/230
303D_0A18	Memory Region Control (RDC_MRC33)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0A1C	Memory Region Violation Status (RDC_MRVS33)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0A20	Memory Region Start Address (RDC_MRSA34)	32	R/W	Undefined	3.2.5.7/229
303D_0A24	Memory Region End Address (RDC_MREA34)	32	R/W	Undefined	3.2.5.8/230
303D_0A28	Memory Region Control (RDC_MRC34)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0A2C	Memory Region Violation Status (RDC_MRVS34)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0A30	Memory Region Start Address (RDC_MRSA35)	32	R/W	Undefined	3.2.5.7/229
303D_0A34	Memory Region End Address (RDC_MREA35)	32	R/W	Undefined	3.2.5.8/230
303D_0A38	Memory Region Control (RDC_MRC35)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0A3C	Memory Region Violation Status (RDC_MRVS35)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0A40	Memory Region Start Address (RDC_MRSA36)	32	R/W	Undefined	3.2.5.7/229
303D_0A44	Memory Region End Address (RDC_MREA36)	32	R/W	Undefined	3.2.5.8/230
303D_0A48	Memory Region Control (RDC_MRC36)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0A4C	Memory Region Violation Status (RDC_MRVS36)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0A50	Memory Region Start Address (RDC_MRSA37)	32	R/W	Undefined	3.2.5.7/229
303D_0A54	Memory Region End Address (RDC_MREA37)	32	R/W	Undefined	3.2.5.8/230
303D_0A58	Memory Region Control (RDC_MRC37)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0A5C	Memory Region Violation Status (RDC_MRVS37)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0A60	Memory Region Start Address (RDC_MRSA38)	32	R/W	Undefined	3.2.5.7/229
303D_0A64	Memory Region End Address (RDC_MREA38)	32	R/W	Undefined	3.2.5.8/230
303D_0A68	Memory Region Control (RDC_MRC38)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0A6C	Memory Region Violation Status (RDC_MRVS38)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0A70	Memory Region Start Address (RDC_MRSA39)	32	R/W	Undefined	3.2.5.7/229
303D_0A74	Memory Region End Address (RDC_MREA39)	32	R/W	Undefined	3.2.5.8/230
303D_0A78	Memory Region Control (RDC_MRC39)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0A7C	Memory Region Violation Status (RDC_MRVS39)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0A80	Memory Region Start Address (RDC_MRSA40)	32	R/W	Undefined	3.2.5.7/229
303D_0A84	Memory Region End Address (RDC_MREA40)	32	R/W	Undefined	3.2.5.8/230
303D_0A88	Memory Region Control (RDC_MRC40)	32	R/W	0000_00FFh	3.2.5.9/230

Table continues on the next page...

RDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303D_0A8C	Memory Region Violation Status (RDC_MRVS40)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0A90	Memory Region Start Address (RDC_MRSA41)	32	R/W	Undefined	3.2.5.7/229
303D_0A94	Memory Region End Address (RDC_MREA41)	32	R/W	Undefined	3.2.5.8/230
303D_0A98	Memory Region Control (RDC_MRC41)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0A9C	Memory Region Violation Status (RDC_MRVS41)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0AA0	Memory Region Start Address (RDC_MRSA42)	32	R/W	Undefined	3.2.5.7/229
303D_0AA4	Memory Region End Address (RDC_MREA42)	32	R/W	Undefined	3.2.5.8/230
303D_0AA8	Memory Region Control (RDC_MRC42)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0AAC	Memory Region Violation Status (RDC_MRVS42)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0AB0	Memory Region Start Address (RDC_MRSA43)	32	R/W	Undefined	3.2.5.7/229
303D_0AB4	Memory Region End Address (RDC_MREA43)	32	R/W	Undefined	3.2.5.8/230
303D_0AB8	Memory Region Control (RDC_MRC43)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0ABC	Memory Region Violation Status (RDC_MRVS43)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0AC0	Memory Region Start Address (RDC_MRSA44)	32	R/W	Undefined	3.2.5.7/229
303D_0AC4	Memory Region End Address (RDC_MREA44)	32	R/W	Undefined	3.2.5.8/230
303D_0AC8	Memory Region Control (RDC_MRC44)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0ACC	Memory Region Violation Status (RDC_MRVS44)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0AD0	Memory Region Start Address (RDC_MRSA45)	32	R/W	Undefined	3.2.5.7/229
303D_0AD4	Memory Region End Address (RDC_MREA45)	32	R/W	Undefined	3.2.5.8/230
303D_0AD8	Memory Region Control (RDC_MRC45)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0ADC	Memory Region Violation Status (RDC_MRVS45)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0AE0	Memory Region Start Address (RDC_MRSA46)	32	R/W	Undefined	3.2.5.7/229
303D_0AE4	Memory Region End Address (RDC_MREA46)	32	R/W	Undefined	3.2.5.8/230
303D_0AE8	Memory Region Control (RDC_MRC46)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0AEC	Memory Region Violation Status (RDC_MRVS46)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0AF0	Memory Region Start Address (RDC_MRSA47)	32	R/W	Undefined	3.2.5.7/229
303D_0AF4	Memory Region End Address (RDC_MREA47)	32	R/W	Undefined	3.2.5.8/230
303D_0AF8	Memory Region Control (RDC_MRC47)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0AFC	Memory Region Violation Status (RDC_MRVS47)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0B00	Memory Region Start Address (RDC_MRSA48)	32	R/W	Undefined	3.2.5.7/229
303D_0B04	Memory Region End Address (RDC_MREA48)	32	R/W	Undefined	3.2.5.8/230
303D_0B08	Memory Region Control (RDC_MRC48)	32	R/W	0000_00FFh	3.2.5.9/230

Table continues on the next page...

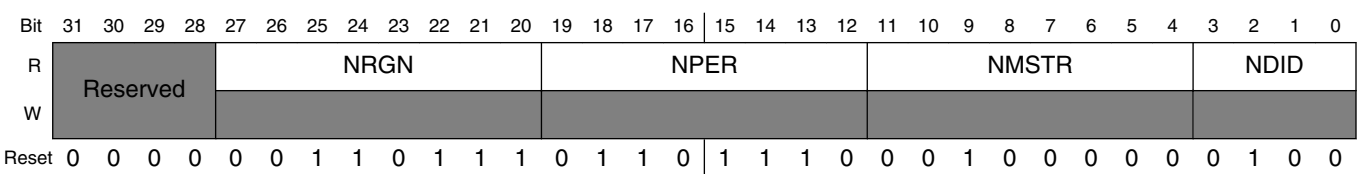
RDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303D_0B0C	Memory Region Violation Status (RDC_MRVS48)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0B10	Memory Region Start Address (RDC_MRSA49)	32	R/W	Undefined	3.2.5.7/229
303D_0B14	Memory Region End Address (RDC_MREA49)	32	R/W	Undefined	3.2.5.8/230
303D_0B18	Memory Region Control (RDC_MRC49)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0B1C	Memory Region Violation Status (RDC_MRVS49)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0B20	Memory Region Start Address (RDC_MRSA50)	32	R/W	Undefined	3.2.5.7/229
303D_0B24	Memory Region End Address (RDC_MREA50)	32	R/W	Undefined	3.2.5.8/230
303D_0B28	Memory Region Control (RDC_MRC50)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0B2C	Memory Region Violation Status (RDC_MRVS50)	32	R/W	0000_0000h	3.2.5.10/ 232
303D_0B30	Memory Region Start Address (RDC_MRSA51)	32	R/W	Undefined	3.2.5.7/229
303D_0B34	Memory Region End Address (RDC_MREA51)	32	R/W	Undefined	3.2.5.8/230
303D_0B38	Memory Region Control (RDC_MRC51)	32	R/W	0000_00FFh	3.2.5.9/230
303D_0B3C	Memory Region Violation Status (RDC_MRVS51)	32	R/W	0000_0000h	3.2.5.10/ 232

3.2.5.1 Version Information (RDC_VIR)

The VIR provides version information including the number of domains, number of master slots, number of peripheral slots, and number of memory regions.

Address: 303D_0000h base + 0h offset = 303D_0000h



RDC_VIR field descriptions

Field	Description
31–28 Reserved	This field is reserved.
27–20 NRGN	Number of Memory Regions Indicates the number of memory regions in this instance of the RDC.
19–12 NPER	Number of Peripherals Indicates the number of peripherals that can be isolated or safe-shared

Table continues on the next page...

RDC_VIR field descriptions (continued)

Field	Description
11–4 NMSTR	Number of Masters Indicates the number of masters supported by this instance of RDC.
NDID	Number of Domains Indicates the number of domain ids supported by this instance of the RDC. Add one to the register value to get the actual number of domains.

3.2.5.2 Status (RDC_STAT)

Address: 303D_0000h base + 24h offset = 303D_0024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

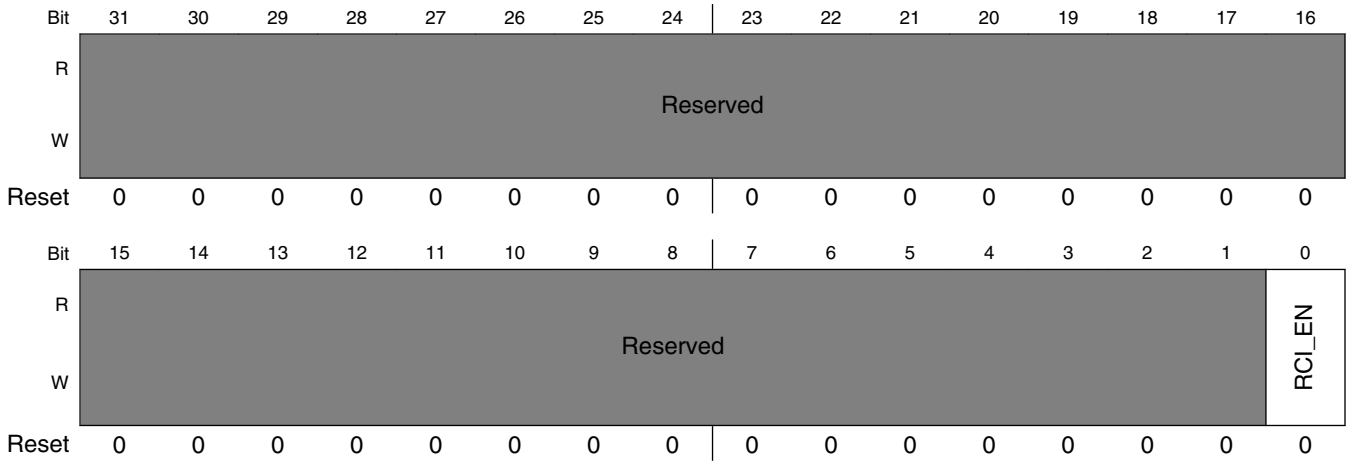
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved							PDS	Reserved				DID			
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

RDC_STAT field descriptions

Field	Description
31–9 Reserved	This field is reserved.
8 PDS	Power Domain Status Indicates if the "Power Down" memory regions are powered and available. Power Down memory regions are only those memory regions susceptible to power outage for power savings are unavailable if this is zero. "Always-On" memory regions remain available. Always On memory regions are those regions that are not powered down unless the entire SoC is powered down. This signal remains low until all access controls have been restored to the domain. 0 Power Down Domain is OFF 1 Power Down Domain is ON
7–4 Reserved	This field is reserved.
DID	Domain ID The Domain ID of the core or bus master that is reading this. The value is different for requests from different domains.

3.2.5.3 Interrupt and Control (RDC_INTCTRL)

Address: 303D_0000h base + 28h offset = 303D_0028h



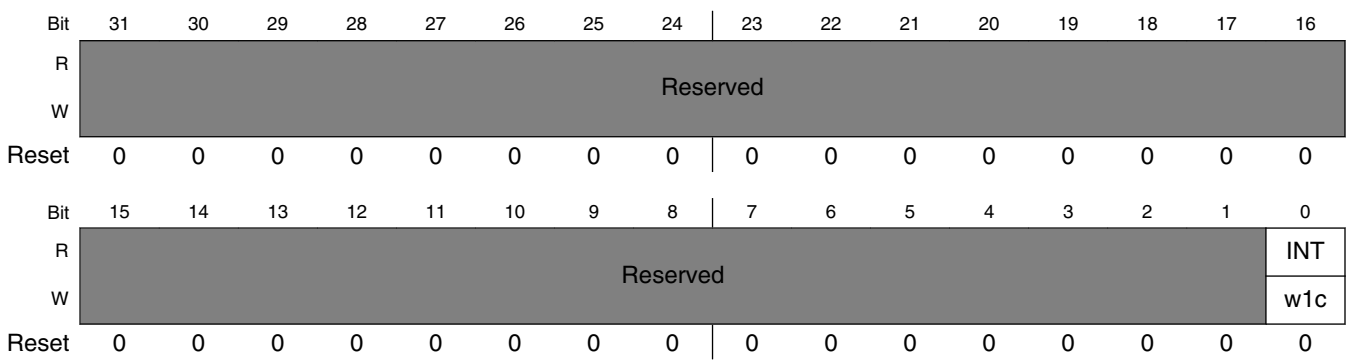
RDC_INTCTRL field descriptions

Field	Description
31–1 Reserved	This field is reserved.
0 RCI_EN	Restoration Complete Interrupt Interrupt generated when the RDC has completed restoring state to a recently re-powered memory regions. 0 Interrupt Disabled 1 Interrupt Enabled

3.2.5.4 Interrupt Status (RDC_INTSTAT)

Indication of Interrupt Pending for State Restoration

Address: 303D_0000h base + 2Ch offset = 303D_002Ch



RDC_INTSTAT field descriptions

Field	Description
31–1 Reserved	This field is reserved.
0 INT	Interrupt Status Indicates state of interrupt signal for state restoration. This is that status of the interrupt enabled in RDC_INTCTRL. Write one to interrupt status to clear it. 0 No Interrupt Pending 1 Interrupt Pending

3.2.5.5 Master Domain Assignment (RDC_MDA*n*)

Address: 303D_0000h base + 200h offset + (4d × i), where i=0d to 26d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LCK	Reserved														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DID	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RDC_MDA*n* field descriptions

Field	Description
31 LCK	0 Not Locked 1 Locked
30–2 Reserved	This field is reserved.
DID	Domain ID Indicates the domain to which the Master is assigned 00 Master assigned to Processing Domain 0 01 Master assigned to Processing Domain 1 10 Master assigned to Processing Domain 2 11 Master assigned to Processing Domain 3

3.2.5.6 Peripheral Domain Access Permissions (RDC_PDAP*n*)

Address: 303D_0000h base + 400h offset + (4d × i), where i=0d to 117d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LCK	SREQ	Reserved													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								D3R	D3W	D2R	D2W	D1R	D1W	D0R	D0W
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

RDC_PDAP*n* field descriptions

Field	Description
31 LCK	Peripheral Permissions Lock When set prevents further modification of the Peripheral Domain Access Permissions (sticky bit until reset) 0 Not Locked 1 Locked
30 SREQ	Semaphore Required When set the hardware semaphore state enforces the semaphore lock. If a domain has access permissions and a semaphore has locked a shared peripheral then only the domain holding the semaphore signal can access this peripheral. 0 Semaphores have no effect 1 Semaphores are enforced
29–8 Reserved	This field is reserved.
7 D3R	Domain 3 Read Access 0 No Read Access 1 Read Access Allowed
6 D3W	Domain 3 Write Access 0 No Write Access 1 Write Access Allowed
5 D2R	Domain 2 Read Access 0 No Read Access 1 Read Access Allowed
4 D2W	Domain 2 Write Access 0 No Write Access 1 Write Access Allowed
3 D1R	Domain 1 Read Access

Table continues on the next page...

RDC_PDAP*n* field descriptions (continued)

Field	Description
	0 No Read Access 1 Read Access Allowed
2 D1W	Domain 1 Write Access 0 No Write Access 1 Write Access Allowed
1 D0R	Domain 0 Read Access 0 No Read Access 1 Read Access Allowed
0 D0W	Domain 0 Write Access 0 No Write Access 1 Write Access Allowed

3.2.5.7 Memory Region Start Address (RDC_MRSA*n*)

Address: 303D_0000h base + 800h offset + (16d × i), where i=0d to 51d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SADR																Reserved															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

RDC_MRSA*n* field descriptions

Field	Description
31–7 SADR	Start address for memory region Lower bound (inclusive) modulo the defined granularity byte size of a region. The region size (granularity) is defined for each Memory/Port in the Memory Region Map section. Region boundaries are aligned to the minimum possible region size for the Memory/Port.
Reserved	This field is reserved.

3.2.5.8 Memory Region End Address (RDC_MREA*n*)

Address: 303D_0000h base + 804h offset + (16d × i), where i=0d to 51d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EADR																Reserved															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

RDC_MREA*n* field descriptions

Field	Description
31–7 EADR	Upper bound for memory region Upper bound (exclusive) modulo the defined granularity byte size of a region. The region size (granularity) is defined for each Memory/Port in the Memory Region Map section. Region boundaries are aligned to the minimum possible region size for the Memory/Port.
Reserved	This field is reserved.

3.2.5.9 Memory Region Control (RDC_MRC*n*)

Address: 303D_0000h base + 808h offset + (16d × i), where i=0d to 51d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LCK	ENA	Reserved													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								D3R	D3W	D2R	D2W	D1R	D1W	D0R	D0W
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

RDC_MRC*n* field descriptions

Field	Description
31 LCK	Region Lock Locks all region fields from further modification except ENA, which can be set but not reset after LCK is set. LCK is a sticky bit. 0 No Lock. All fields in this register may be modified. 1 Locked. No fields in this register may be modified except ENA, which may be set but not cleared.
30 ENA	Region Enable Activates the memory region. If the region is not activated then the permissions and address boundaries have not affect and the region will be fully accessible.

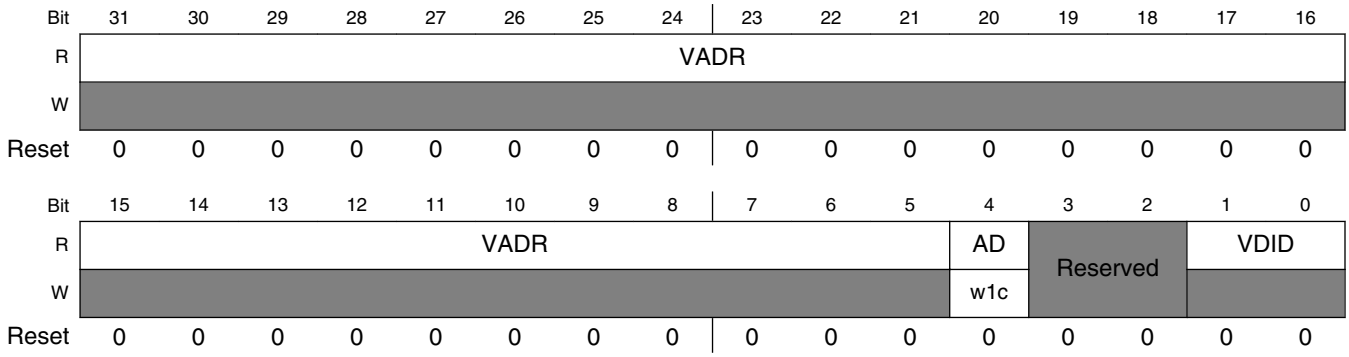
Table continues on the next page...

RDC_MRC*n* field descriptions (continued)

Field	Description
	0 Memory region is not defined or restricted. 1 Memory boundaries, domain permissions and controls are in effect.
29–8 Reserved	This field is reserved.
7 D3R	Domain 3 Read Access to Region 0 Processing Domain 3 does not have Read access to the memory region 1 Processing Domain 3 has Read access to the memory region
6 D3W	Domain 3 Write Access to Region 0 Processing Domain 3 does not have Write access to the memory region 1 Processing Domain 3 has Read access to the memory region
5 D2R	Domain 2 Read Access to Region 0 Processing Domain 2 does not have Read access to the memory region 1 Processing Domain 2 has Read access to the memory region
4 D2W	Domain 2 Write Access to Region 0 Processing Domain 2 does not have Write access to the memory region 1 Processing Domain 2 has Write access to the memory region
3 D1R	Domain 1 Read Access to Region 0 Processing Domain 1 does not have Read access to the memory region 1 Processing Domain 1 has Read access to the memory region
2 D1W	Domain 1 Write Access to Region 0 Processing Domain 1 does not have Write access to the memory region 1 Processing Domain 1 has Write access to the memory region
1 D0R	Domain 0 Read Access to Region 0 Processing Domain 0 does not have Read access to the memory region 1 Processing Domain 0 has Read access to the memory region
0 D0W	Domain 0 Write Access to Region 0 Processing Domain 0 does not have Write access to the memory region 1 Processing Domain 0 has Write access to the memory region

3.2.5.10 Memory Region Violation Status (RDC_MRVS*n*)

Address: 303D_0000h base + 80Ch offset + (16d × i), where i=0d to 51d



RDC_MRVS*n* field descriptions

Field	Description
31–5 VADR	Violating Address The address of the denied access. The first access violation is captured. Subsequent violations are ignored until the status register is cleared. Contents are cleared upon reading the register. Clearing of contents occurs only when the status is read by the memory region's associated domain ID (s).
4 AD	Access Denied Access to a memory region denied. This bit is cleared when this bit is written by one of the allowed domains.
3–2 Reserved	This field is reserved.
VDID	Violating Domain ID The domain ID of the denied access. The first access violation is captured. Subsequent violations are ignored until the status register is cleared. Contents are cleared upon reading the register. 00 Processing Domain 0 01 Processing Domain 1 10 Processing Domain 2 11 Processing Domain 3

3.2.6 RDC SEMA42 Memory Map/Register Definition

Only Supervisor Mode accesses are allowed on these registers. User accesses generate an error termination.

RDC_SEMAPHORE memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303B_0000	Gate Register (RDC_SEMAPHORE1_GATE0)	8	R/W	00h	3.2.6.1/236
303B_0001	Gate Register (RDC_SEMAPHORE1_GATE1)	8	R/W	00h	3.2.6.1/236
303B_0002	Gate Register (RDC_SEMAPHORE1_GATE2)	8	R/W	00h	3.2.6.1/236
303B_0003	Gate Register (RDC_SEMAPHORE1_GATE3)	8	R/W	00h	3.2.6.1/236
303B_0004	Gate Register (RDC_SEMAPHORE1_GATE4)	8	R/W	00h	3.2.6.1/236
303B_0005	Gate Register (RDC_SEMAPHORE1_GATE5)	8	R/W	00h	3.2.6.1/236
303B_0006	Gate Register (RDC_SEMAPHORE1_GATE6)	8	R/W	00h	3.2.6.1/236
303B_0007	Gate Register (RDC_SEMAPHORE1_GATE7)	8	R/W	00h	3.2.6.1/236
303B_0008	Gate Register (RDC_SEMAPHORE1_GATE8)	8	R/W	00h	3.2.6.1/236
303B_0009	Gate Register (RDC_SEMAPHORE1_GATE9)	8	R/W	00h	3.2.6.1/236
303B_000A	Gate Register (RDC_SEMAPHORE1_GATE10)	8	R/W	00h	3.2.6.1/236
303B_000B	Gate Register (RDC_SEMAPHORE1_GATE11)	8	R/W	00h	3.2.6.1/236
303B_000C	Gate Register (RDC_SEMAPHORE1_GATE12)	8	R/W	00h	3.2.6.1/236
303B_000D	Gate Register (RDC_SEMAPHORE1_GATE13)	8	R/W	00h	3.2.6.1/236
303B_000E	Gate Register (RDC_SEMAPHORE1_GATE14)	8	R/W	00h	3.2.6.1/236
303B_000F	Gate Register (RDC_SEMAPHORE1_GATE15)	8	R/W	00h	3.2.6.1/236
303B_0010	Gate Register (RDC_SEMAPHORE1_GATE16)	8	R/W	00h	3.2.6.1/236
303B_0011	Gate Register (RDC_SEMAPHORE1_GATE17)	8	R/W	00h	3.2.6.1/236
303B_0012	Gate Register (RDC_SEMAPHORE1_GATE18)	8	R/W	00h	3.2.6.1/236
303B_0013	Gate Register (RDC_SEMAPHORE1_GATE19)	8	R/W	00h	3.2.6.1/236
303B_0014	Gate Register (RDC_SEMAPHORE1_GATE20)	8	R/W	00h	3.2.6.1/236
303B_0015	Gate Register (RDC_SEMAPHORE1_GATE21)	8	R/W	00h	3.2.6.1/236
303B_0016	Gate Register (RDC_SEMAPHORE1_GATE22)	8	R/W	00h	3.2.6.1/236
303B_0017	Gate Register (RDC_SEMAPHORE1_GATE23)	8	R/W	00h	3.2.6.1/236
303B_0018	Gate Register (RDC_SEMAPHORE1_GATE24)	8	R/W	00h	3.2.6.1/236
303B_0019	Gate Register (RDC_SEMAPHORE1_GATE25)	8	R/W	00h	3.2.6.1/236
303B_001A	Gate Register (RDC_SEMAPHORE1_GATE26)	8	R/W	00h	3.2.6.1/236
303B_001B	Gate Register (RDC_SEMAPHORE1_GATE27)	8	R/W	00h	3.2.6.1/236
303B_001C	Gate Register (RDC_SEMAPHORE1_GATE28)	8	R/W	00h	3.2.6.1/236
303B_001D	Gate Register (RDC_SEMAPHORE1_GATE29)	8	R/W	00h	3.2.6.1/236
303B_001E	Gate Register (RDC_SEMAPHORE1_GATE30)	8	R/W	00h	3.2.6.1/236
303B_001F	Gate Register (RDC_SEMAPHORE1_GATE31)	8	R/W	00h	3.2.6.1/236
303B_0020	Gate Register (RDC_SEMAPHORE1_GATE32)	8	R/W	00h	3.2.6.1/236
303B_0021	Gate Register (RDC_SEMAPHORE1_GATE33)	8	R/W	00h	3.2.6.1/236
303B_0022	Gate Register (RDC_SEMAPHORE1_GATE34)	8	R/W	00h	3.2.6.1/236
303B_0023	Gate Register (RDC_SEMAPHORE1_GATE35)	8	R/W	00h	3.2.6.1/236
303B_0024	Gate Register (RDC_SEMAPHORE1_GATE36)	8	R/W	00h	3.2.6.1/236
303B_0025	Gate Register (RDC_SEMAPHORE1_GATE37)	8	R/W	00h	3.2.6.1/236

Table continues on the next page...

RDC_SEMAPHORE memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303B_0026	Gate Register (RDC_SEMAPHORE1_GATE38)	8	R/W	00h	3.2.6.1/236
303B_0027	Gate Register (RDC_SEMAPHORE1_GATE39)	8	R/W	00h	3.2.6.1/236
303B_0028	Gate Register (RDC_SEMAPHORE1_GATE40)	8	R/W	00h	3.2.6.1/236
303B_0029	Gate Register (RDC_SEMAPHORE1_GATE41)	8	R/W	00h	3.2.6.1/236
303B_002A	Gate Register (RDC_SEMAPHORE1_GATE42)	8	R/W	00h	3.2.6.1/236
303B_002B	Gate Register (RDC_SEMAPHORE1_GATE43)	8	R/W	00h	3.2.6.1/236
303B_002C	Gate Register (RDC_SEMAPHORE1_GATE44)	8	R/W	00h	3.2.6.1/236
303B_002D	Gate Register (RDC_SEMAPHORE1_GATE45)	8	R/W	00h	3.2.6.1/236
303B_002E	Gate Register (RDC_SEMAPHORE1_GATE46)	8	R/W	00h	3.2.6.1/236
303B_002F	Gate Register (RDC_SEMAPHORE1_GATE47)	8	R/W	00h	3.2.6.1/236
303B_0030	Gate Register (RDC_SEMAPHORE1_GATE48)	8	R/W	00h	3.2.6.1/236
303B_0031	Gate Register (RDC_SEMAPHORE1_GATE49)	8	R/W	00h	3.2.6.1/236
303B_0032	Gate Register (RDC_SEMAPHORE1_GATE50)	8	R/W	00h	3.2.6.1/236
303B_0033	Gate Register (RDC_SEMAPHORE1_GATE51)	8	R/W	00h	3.2.6.1/236
303B_0034	Gate Register (RDC_SEMAPHORE1_GATE52)	8	R/W	00h	3.2.6.1/236
303B_0035	Gate Register (RDC_SEMAPHORE1_GATE53)	8	R/W	00h	3.2.6.1/236
303B_0036	Gate Register (RDC_SEMAPHORE1_GATE54)	8	R/W	00h	3.2.6.1/236
303B_0037	Gate Register (RDC_SEMAPHORE1_GATE55)	8	R/W	00h	3.2.6.1/236
303B_0038	Gate Register (RDC_SEMAPHORE1_GATE56)	8	R/W	00h	3.2.6.1/236
303B_0039	Gate Register (RDC_SEMAPHORE1_GATE57)	8	R/W	00h	3.2.6.1/236
303B_003A	Gate Register (RDC_SEMAPHORE1_GATE58)	8	R/W	00h	3.2.6.1/236
303B_003B	Gate Register (RDC_SEMAPHORE1_GATE59)	8	R/W	00h	3.2.6.1/236
303B_003C	Gate Register (RDC_SEMAPHORE1_GATE60)	8	R/W	00h	3.2.6.1/236
303B_003D	Gate Register (RDC_SEMAPHORE1_GATE61)	8	R/W	00h	3.2.6.1/236
303B_003E	Gate Register (RDC_SEMAPHORE1_GATE62)	8	R/W	00h	3.2.6.1/236
303B_003F	Gate Register (RDC_SEMAPHORE1_GATE63)	8	R/W	00h	3.2.6.1/236
303B_0040	Reset Gate Write (RDC_SEMAPHORE1_RSTGT_W)	16	R/W	0000h	3.2.6.2/237
303B_0040	Reset Gate Read (RDC_SEMAPHORE1_RSTGT_R)	16	R/W	0000h	3.2.6.3/239
303C_0000	Gate Register (RDC_SEMAPHORE2_GATE0)	8	R/W	00h	3.2.6.1/236
303C_0001	Gate Register (RDC_SEMAPHORE2_GATE1)	8	R/W	00h	3.2.6.1/236
303C_0002	Gate Register (RDC_SEMAPHORE2_GATE2)	8	R/W	00h	3.2.6.1/236
303C_0003	Gate Register (RDC_SEMAPHORE2_GATE3)	8	R/W	00h	3.2.6.1/236
303C_0004	Gate Register (RDC_SEMAPHORE2_GATE4)	8	R/W	00h	3.2.6.1/236
303C_0005	Gate Register (RDC_SEMAPHORE2_GATE5)	8	R/W	00h	3.2.6.1/236
303C_0006	Gate Register (RDC_SEMAPHORE2_GATE6)	8	R/W	00h	3.2.6.1/236
303C_0007	Gate Register (RDC_SEMAPHORE2_GATE7)	8	R/W	00h	3.2.6.1/236
303C_0008	Gate Register (RDC_SEMAPHORE2_GATE8)	8	R/W	00h	3.2.6.1/236
303C_0009	Gate Register (RDC_SEMAPHORE2_GATE9)	8	R/W	00h	3.2.6.1/236

Table continues on the next page...

RDC_SEMAPHORE memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303C_000A	Gate Register (RDC_SEMAPHORE2_GATE10)	8	R/W	00h	3.2.6.1/236
303C_000B	Gate Register (RDC_SEMAPHORE2_GATE11)	8	R/W	00h	3.2.6.1/236
303C_000C	Gate Register (RDC_SEMAPHORE2_GATE12)	8	R/W	00h	3.2.6.1/236
303C_000D	Gate Register (RDC_SEMAPHORE2_GATE13)	8	R/W	00h	3.2.6.1/236
303C_000E	Gate Register (RDC_SEMAPHORE2_GATE14)	8	R/W	00h	3.2.6.1/236
303C_000F	Gate Register (RDC_SEMAPHORE2_GATE15)	8	R/W	00h	3.2.6.1/236
303C_0010	Gate Register (RDC_SEMAPHORE2_GATE16)	8	R/W	00h	3.2.6.1/236
303C_0011	Gate Register (RDC_SEMAPHORE2_GATE17)	8	R/W	00h	3.2.6.1/236
303C_0012	Gate Register (RDC_SEMAPHORE2_GATE18)	8	R/W	00h	3.2.6.1/236
303C_0013	Gate Register (RDC_SEMAPHORE2_GATE19)	8	R/W	00h	3.2.6.1/236
303C_0014	Gate Register (RDC_SEMAPHORE2_GATE20)	8	R/W	00h	3.2.6.1/236
303C_0015	Gate Register (RDC_SEMAPHORE2_GATE21)	8	R/W	00h	3.2.6.1/236
303C_0016	Gate Register (RDC_SEMAPHORE2_GATE22)	8	R/W	00h	3.2.6.1/236
303C_0017	Gate Register (RDC_SEMAPHORE2_GATE23)	8	R/W	00h	3.2.6.1/236
303C_0018	Gate Register (RDC_SEMAPHORE2_GATE24)	8	R/W	00h	3.2.6.1/236
303C_0019	Gate Register (RDC_SEMAPHORE2_GATE25)	8	R/W	00h	3.2.6.1/236
303C_001A	Gate Register (RDC_SEMAPHORE2_GATE26)	8	R/W	00h	3.2.6.1/236
303C_001B	Gate Register (RDC_SEMAPHORE2_GATE27)	8	R/W	00h	3.2.6.1/236
303C_001C	Gate Register (RDC_SEMAPHORE2_GATE28)	8	R/W	00h	3.2.6.1/236
303C_001D	Gate Register (RDC_SEMAPHORE2_GATE29)	8	R/W	00h	3.2.6.1/236
303C_001E	Gate Register (RDC_SEMAPHORE2_GATE30)	8	R/W	00h	3.2.6.1/236
303C_001F	Gate Register (RDC_SEMAPHORE2_GATE31)	8	R/W	00h	3.2.6.1/236
303C_0020	Gate Register (RDC_SEMAPHORE2_GATE32)	8	R/W	00h	3.2.6.1/236
303C_0021	Gate Register (RDC_SEMAPHORE2_GATE33)	8	R/W	00h	3.2.6.1/236
303C_0022	Gate Register (RDC_SEMAPHORE2_GATE34)	8	R/W	00h	3.2.6.1/236
303C_0023	Gate Register (RDC_SEMAPHORE2_GATE35)	8	R/W	00h	3.2.6.1/236
303C_0024	Gate Register (RDC_SEMAPHORE2_GATE36)	8	R/W	00h	3.2.6.1/236
303C_0025	Gate Register (RDC_SEMAPHORE2_GATE37)	8	R/W	00h	3.2.6.1/236
303C_0026	Gate Register (RDC_SEMAPHORE2_GATE38)	8	R/W	00h	3.2.6.1/236
303C_0027	Gate Register (RDC_SEMAPHORE2_GATE39)	8	R/W	00h	3.2.6.1/236
303C_0028	Gate Register (RDC_SEMAPHORE2_GATE40)	8	R/W	00h	3.2.6.1/236
303C_0029	Gate Register (RDC_SEMAPHORE2_GATE41)	8	R/W	00h	3.2.6.1/236
303C_002A	Gate Register (RDC_SEMAPHORE2_GATE42)	8	R/W	00h	3.2.6.1/236
303C_002B	Gate Register (RDC_SEMAPHORE2_GATE43)	8	R/W	00h	3.2.6.1/236
303C_002C	Gate Register (RDC_SEMAPHORE2_GATE44)	8	R/W	00h	3.2.6.1/236
303C_002D	Gate Register (RDC_SEMAPHORE2_GATE45)	8	R/W	00h	3.2.6.1/236
303C_002E	Gate Register (RDC_SEMAPHORE2_GATE46)	8	R/W	00h	3.2.6.1/236
303C_002F	Gate Register (RDC_SEMAPHORE2_GATE47)	8	R/W	00h	3.2.6.1/236

Table continues on the next page...

RDC_SEMAPHORE memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303C_0030	Gate Register (RDC_SEMAPHORE2_GATE48)	8	R/W	00h	3.2.6.1/236
303C_0031	Gate Register (RDC_SEMAPHORE2_GATE49)	8	R/W	00h	3.2.6.1/236
303C_0032	Gate Register (RDC_SEMAPHORE2_GATE50)	8	R/W	00h	3.2.6.1/236
303C_0033	Gate Register (RDC_SEMAPHORE2_GATE51)	8	R/W	00h	3.2.6.1/236
303C_0034	Gate Register (RDC_SEMAPHORE2_GATE52)	8	R/W	00h	3.2.6.1/236
303C_0035	Gate Register (RDC_SEMAPHORE2_GATE53)	8	R/W	00h	3.2.6.1/236
303C_0036	Gate Register (RDC_SEMAPHORE2_GATE54)	8	R/W	00h	3.2.6.1/236
303C_0037	Gate Register (RDC_SEMAPHORE2_GATE55)	8	R/W	00h	3.2.6.1/236
303C_0038	Gate Register (RDC_SEMAPHORE2_GATE56)	8	R/W	00h	3.2.6.1/236
303C_0039	Gate Register (RDC_SEMAPHORE2_GATE57)	8	R/W	00h	3.2.6.1/236
303C_003A	Gate Register (RDC_SEMAPHORE2_GATE58)	8	R/W	00h	3.2.6.1/236
303C_003B	Gate Register (RDC_SEMAPHORE2_GATE59)	8	R/W	00h	3.2.6.1/236
303C_003C	Gate Register (RDC_SEMAPHORE2_GATE60)	8	R/W	00h	3.2.6.1/236
303C_003D	Gate Register (RDC_SEMAPHORE2_GATE61)	8	R/W	00h	3.2.6.1/236
303C_003E	Gate Register (RDC_SEMAPHORE2_GATE62)	8	R/W	00h	3.2.6.1/236
303C_003F	Gate Register (RDC_SEMAPHORE2_GATE63)	8	R/W	00h	3.2.6.1/236
303C_0040	Reset Gate Write (RDC_SEMAPHORE2_RSTGT_W)	16	R/W	0000h	3.2.6.2/237
303C_0040	Reset Gate Read (RDC_SEMAPHORE2_RSTGT_R)	16	R/W	0000h	3.2.6.3/239

3.2.6.1 Gate Register (RDC_SEMAPHORE*x*_GATE*n*)

Each semaphore gate is implemented in a 4-bit finite state machine, right-justified in a byte data structure. The hardware uses the logical bus master number (master_index) in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. Attempted writes with a data value that is neither the unlock value nor the appropriate lock value (master_index + 1) are simply treated as "no operation" and do not affect any gate state. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes. Processor dex values can be found in Table 4-29.

Address: Base address + 0h offset + (1d × i), where i=0d to 63d

Bit	7	6	5	4	3	2	1	0
Read	0		LDOM		GTFSM			
Write								
Reset	0	0	0	0	0	0	0	0

RDC_SEMAPHORE*x*_GATE*n* field descriptions

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 LDOM	Read-only bits. They indicate which domain had currently locked the gate. 00 The gate is locked by domain 0. (True if bits [3:0] do not equal 0000.) 01 The gate has been locked by domain 1. 10 The gate has been locked by domain 2. 11 The gate has been locked by domain 3.
GTFSM	Gate Finite State Machine. The state of the gate reflects the last processor that locked it, which can be useful during system debug. The hardware gate is maintained in a 16-state implementation, defined as: 0000 The gate is unlocked (free). 0001 The gate has been locked by processor with master_index = 0. 0010 The gate has been locked by processor with master_index = 1. 0011 The gate has been locked by processor with master_index = 2. 0100 The gate has been locked by processor with master_index = 3. 0101 The gate has been locked by processor with master_index = 4. 0110 The gate has been locked by processor with master_index = 5. 0111 The gate has been locked by processor with master_index = 6. 1000 The gate has been locked by processor with master_index = 7. 1001 The gate has been locked by processor with master_index = 8. 1010 The gate has been locked by processor with master_index = 9. 1011 The gate has been locked by processor with master_index = 10. 1100 The gate has been locked by processor with master_index = 11. 1101 The gate has been locked by processor with master_index = 12. 1110 The gate has been locked by processor with master_index = 13. 1111 The gate has been locked by processor with master_index = 14.

3.2.6.2 Reset Gate Write (RDC_SEMAPHORE*x*_RSTGT_W)

Although the intent of the hardware gate implementation specifies a protocol where the locking processor must unlock the gate, it is recognized that system operation may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset.

To support this special gate reset requirement, the RDC Semaphores module implements a "secure" reset mechanism that allows a hardware gate (or all the gates) to be initialized by following a specific dual-write access pattern. Using a technique similar to that required for the servicing of a software watchdog timer, the secure gate reset requires two consecutive writes with predefined data patterns from the same processor to force the clearing of the specified gate(s). The required access pattern is:

1. A processor performs a 16-bit write to the RDC_SEMA42RSTGT memory location. The least significant byte (RDC_SEMA42RSTGT[RSTGDP]) must be 0xE2; the most significant byte is a "don't_care" for this reference.
2. The same processor then performs a second 16-bit write to the RDC_SEMA42RSTGT location. For this write, the lower byte (RDC_SEMA42RSTGT[RSTGDP]) is the logical complement of the first data pattern (0x1D) and the upper byte (RDC_SEMA42RSTGT[RSTGTN]) specifies the gate(s) to be reset. This gate field can specify a single gate be cleared, or else that all gates are to be cleared. If the same processor writes incorrect data on the second access or another processor performs the second write access, the special gate reset sequence is aborted and no error signal will be asserted.
3. Reads of the RDC_SEMA42RSTGT location return information on the 2-bit state machine (RDC_SEMA42RSTGT[RSTGSM]) that implements this function, the bus master performing the reset (RDC_SEMA42RSTGT[RSTGMS]), and the gate number(s) last cleared (RDC_SEMA42RSTGT[RSTGTN]). Reads of the RDC_SEMA42RSTGT register do not affect the secure reset finite state machine in any manner.

Address: Base address + 40h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RSTGTN								0							
Write									RSTGDP							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RDC_SEMAPHOREx_RSTGT_W field descriptions

Field	Description
15–8 RSTGTN	Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write. If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates.
RSTGDP	Reset Gate Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the gate reset mechanism. For the first write, RSTGDP = 0xE2 while the second write requires RSTGDP = 0x1D.

3.2.6.3 Reset Gate Read (RDC_SEMAPHOREx_RSTGT_R)

Although the intent of the hardware gate implementation specifies a protocol where the locking processor must unlock the gate, it is recognized that system operation may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset.

To support this special gate reset requirement, the RDC Semaphores module implements a "secure" reset mechanism that allows a hardware gate (or all the gates) to be initialized by following a specific dual-write access pattern. Using a technique similar to that required for the servicing of a software watchdog timer, the secure gate reset requires two consecutive writes with predefined data patterns from the same processor to force the clearing of the specified gate(s). The required access pattern is:

1. A processor performs a 16-bit write to the RDC_SEMA42RSTGT memory location. The least significant byte (RDC_SEMA42RSTGT[RSTGDP]) must be 0xE2; the most significant byte is a "don't_care" for this reference.
2. The same processor then performs a second 16-bit write to the RDC_SEMA42RSTGT location. For this write, the lower byte (RDC_SEMA42RSTGT[RSTGDP]) is the logical complement of the first data pattern (0x1D) and the upper byte (RDC_SEMA42RSTGT[RSTGTN]) specifies the gate(s) to be reset. This gate field can specify a single gate be cleared, or else that all gates are to be cleared. If the same processor writes incorrect data on the second access or another processor performs the second write access, the special gate reset sequence is aborted and no error signal will be asserted.
3. Reads of the RDC_SEMA42RSTGT location return information on the 2-bit state machine (RDC_SEMA42RSTGT[RSTGSM]) that implements this function, the bus master performing the reset (RDC_SEMA42RSTGT[RSTGMS]), and the gate number(s) last cleared (RDC_SEMA42RSTGT[RSTGTN]). Reads of the RDC_SEMA42RSTGT register do not affect the secure reset finite state machine in any manner.

Address: Base address + 40h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RSTGTN								0		RSTGSM		RSTGMS			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RDC_SEMAPHOREx_RSTGT_R field descriptions

Field	Description
15–8 RSTGTN	Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write.

Table continues on the next page...

RDC_SEMAPHOREx_RSTGT_R field descriptions (continued)

Field	Description
	If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates.
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 RSTGSM	Reset Gate Finite State Machine. Reads of the RDC_SEMA42RSTGT register return the encoded state machine value. Note the RSTGSM = 10 state is valid for only a single machine cycle, so it is impossible for a read to return this value. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as: 00 Idle, waiting for the first data pattern write. 01 Waiting for the second data pattern write. 10 The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. The "01" state persists for only one clock cycle. Software will never be able to observe this state. 11 This state encoding is never used and therefore reserved.
RSTGMS	Reset Gate Bus Master. This 4-bit read-only field records the logical number of the bus master performing the gate reset function. The reset function requires that the two consecutive writes to this register must be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs. The association between system bus master port numbers, the associated bus master device, and the logical processor number is SoC-specific. Consult the device reference manual for this information.

Chapter 4
ARM Platform and Debug

4.1 ARM Cortex A7 Platform (CA7)

4.1.1 Overview

The Cortex-A7 MPCore platform is a high-performance, low-power processor compliant with the ARMv7-A architecture. The Cortex-A7 MPCore platform has dual symmetric ARM®Cortex®-A7 cores with a L1 cache subsystem, integrated L2 cache subsystem, and integrated Generic Interrupt Controller (GIC).

Each Cortex-A7 MPCore includes the following:

- 32 KB L1 Instruction Cache
- 32 KB L1 Data Cache
- Media Processing Engine (MPE) with NEON technology supporting the Advanced Single Instruction Multiple Data version 2 (SIMDv2) architecture
- Floating Point Unit (FPU) with support of the VFPv4-D32 architecture

NOTE

This is a superset of VFPv4-D16

- Support of ARMv7A architecture including:
  - Security extensions for enhanced security
  - Virtualization extensions
  - Large Physical Address (LPA) extension (LPA only supported within Cortex-A7 MPCore platform)

The Cortex-A7 MPCore platform includes the following:

- Integrated Global Interrupt Controller (GIC) configured to support 128 shared peripheral interrupts
- Generic timer
- Snoop control unit (SCU)
- 512 KB unified L2 cache

- Interconnect using a single 128-bit wide bus AMBA AXI bus
- ARMv7.1 ARM debug architecture that complies with the Coresight debug/trace architecture

4.1.2 External Signals

The following table describes the external signals of ARM:

Table 4-1. ARM External Signals

Signal	Description	Pad	Mode	Direction
ARM_EVENTI	Input event signal	LCD_RESET	ALT2	I
ARM_EVENTO	Output event signal	LCD_DATA18	ALT2	O
ARM_TRACE00	Trace signal	LCD_DATA00	ALT2	O
ARM_TRACE01	Trace signal	LCD_DATA01	ALT2	O
ARM_TRACE02	Trace signal	LCD_DATA02	ALT2	O
ARM_TRACE03	Trace signal	LCD_DATA03	ALT2	O
ARM_TRACE04	Trace signal	LCD_DATA04	ALT2	O
ARM_TRACE05	Trace signal	LCD_DATA05	ALT2	O
ARM_TRACE06	Trace signal	LCD_DATA06	ALT2	O
ARM_TRACE07	Trace signal	LCD_DATA07	ALT2	O
ARM_TRACE08	Trace signal	LCD_DATA08	ALT2	O
ARM_TRACE09	Trace signal	LCD_DATA09	ALT2	O
ARM_TRACE10	Trace signal	LCD_DATA10	ALT2	O
ARM_TRACE11	Trace signal	LCD_DATA11	ALT2	O
ARM_TRACE12	Trace signal	LCD_DATA12	ALT2	O
ARM_TRACE13	Trace signal	LCD_DATA13	ALT2	O
ARM_TRACE14	Trace signal	LCD_DATA14	ALT2	O
ARM_TRACE15	Trace signal	LCD_DATA15	ALT2	O
ARM_TRACE_CLK	Clock signal	LCD_DATA16	ALT2	O
ARM_TRACE_CTL	Control signal	LCD_DATA17	ALT2	O

4.1.3 Clocks

This section will discuss the Cortex-A7 clocks. For the specification of the maximum Cortex-A7 core frequency, please see the product datasheet. The following table describes the clock sources for ARM.

Bus Clocks: The AXI master port of the Cortex-A7 MPCore platform is designed to run at half the speed (1:2 ratio) of the Cortex-A7 core.

Debug Clocks: The APB debug interface is designed to run at quarter the speed (1:4 ratio) of the Cortex-A7 core.

4.1.4 Platform Configuration

The revision and configuration of components the Cortex-A7 MPCore platform are detailed below.

Table 4-2. Component Revision

Component	Revision
Cortex-A7 MPCore	MP020-MS-28610-r0p5-00rel0
ETM-A7	TM956-MS-28610-r0p0-00rel0

Table 4-3. Cortex-A7 MPCore Global Configuration

Option	Selected Value	Comments
Instruction cache size	32 KB	L1 instruction cache size per core
Data cache size	32 KB	L1 data cache size per core
L2 cache controller	Present	Integrated L2 cache controller
L2 data RAM cycle latency	3 cycles	
Shared Peripheral Interrupts	128	128 shared peripheral interrupts
Number of processors	2	Dual Cortex-A7 MPCore
Integrated GIC	True	GIC included

Table 4-4. Cortex-A7 Core-Level Configuration

Option	Selected Value	Comments
NEON and/or FPU	FPU and NEON	FPU and NEON are both included in each processor

4.1.5 Low-Power and Performance

This section will discuss the low-power and performance features of the Cortex-A7 Core Platform.

The Cortex-A7 MPCore includes the following low-power features:

- Power-efficient processing provided by Cortex-A7 CPU
- 28nm LP process technology
- Flexible power domain partitioning with separate domains for the following blocks:
  - CPU0 and respective L1 caches
  - CPU1 and respective L1 caches
  - SCU and L2 cache controller
  - L2 cache memory
  - Debug including ETM
- Power-efficient timer events using combination of local generic timers and the global system counter

4.2 ARM Cortex M4 Platform (CM4)

4.2.1 Overview

This block details the ARM Cortex-M4 core. The Cortex-M4 implements the ARMv7-ME instruction set architecture (ISA). It provides compatibility with Cortex-M3 and adds significant new capabilities with DSP and SIMD extensions. The basic multiply-accumulate instructions support operations up to 32 x 32 + 64. Cortex-M4 also includes a single-precision floating-point unit (FPU), which includes an extension register file of thirty-two 32-bit floating-point data registers. Cortex-M4 complex includes the FPU and two 32-bit system bus interfaces. The Cortex-M4 implementation includes two tightly-coupled local memories and two cache memories connected to these bus interfaces although the device implementation connects to the 64-bit system bus interconnect and supports a 32-byte cache line size.

- L1 2-way set-associative 16 KB Instruction/Data cache with 32B line size length

The ARM Cortex-M4 core provides additional general processing capability to the SoC with lower power and fast interrupt response time. The typical use cases for the Cortex-M4 include automotive applications and systems interfaced with CAN that must be able to meet the CAN wakeup requirement, which would otherwise be difficult for the Cortex-A7 to do while running a complex OS.

4.2.1.1 Cortex-M4 Block Diagram

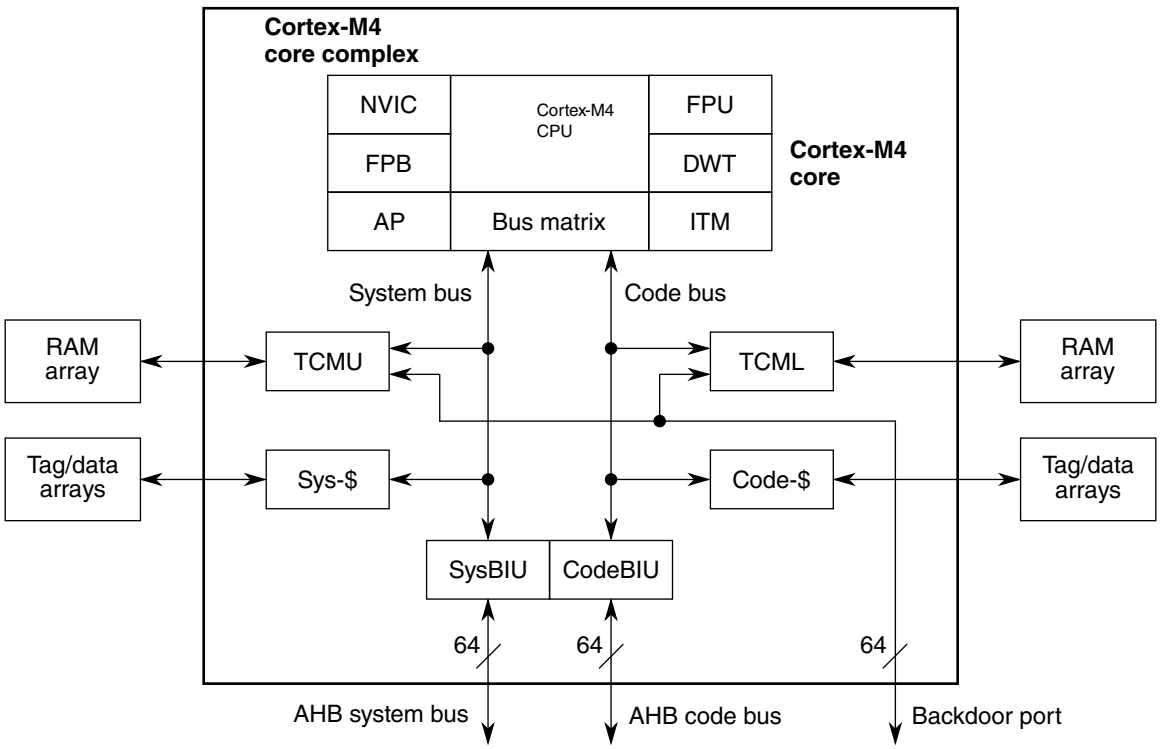


Figure 4-1. Cortex-M4 Block Diagram

Cortex-M4 core features a single issue, three stage pipeline microarchitecture. A high-level spatial pipeline block diagram of the CPU is shown below. The stages of the pipeline include:

- Fe - Instruction fetch stage where data is returned from instruction memory
- De - Instruction decode stage, generation of Load/Store Unit (LSU) address using forwarded register ports and immediate offset of LR register branch forwarding
- Ex - Instruction execute stage, single pipeline with multi-cycle stalls, LSU address/data pipelining to AHB interface, multiply/divide and ALU with branch result

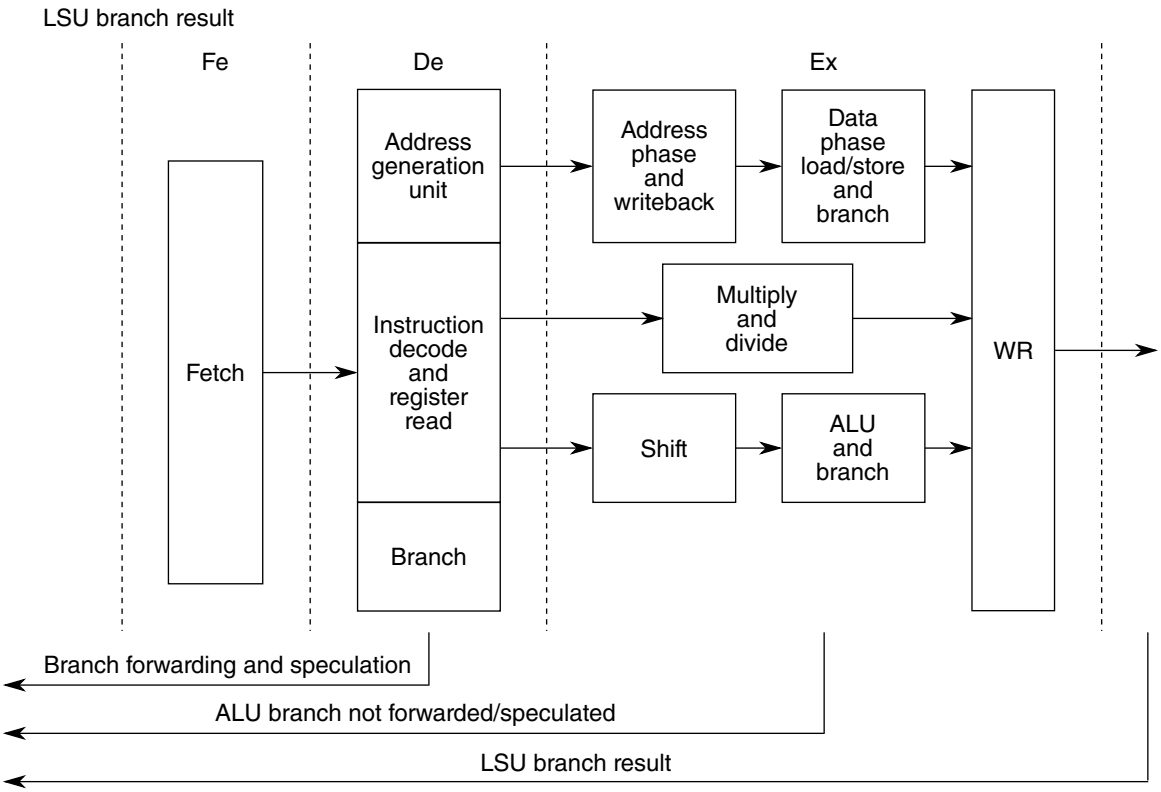


Figure 4-2. Cortex-M4 Pipeline Block Diagram

4.2.2 Cortex-M4 Platform Features

4.2.2.1 Core Module Features

The following are the Cortex-M4 core module features:

- Cortex M4 Core
  - 3-stage pipelined processor supporting ARM v7-M ISA with DSP extensions
  - Single precision FPU supporting the variant of the ARMv7-M Floating-Point Extension (FPv4-SP)
  - MPU - Memory Protection Unit supporting up to 8 regions
  - Modified Harvard connections to AHB_LMEM and the crossbar switch
  - NVIC - Nested Vectored Interrupt Controller with 128 interrupt connections and 16 levels
- AHB LMEM
  - TCMC - Tightly Coupled Memory Controller with support for 64 Kbyte RAM
  - Two 16 KB of combined data/instructions caches to minimize the performance impact of memory access latencies. One for the system bus and one for I/D bus.

4.2.2.2 Network Interconnect Features

The CoreLink Network Interconnect (PL301) is a 2nd generation highly configurable IP component. The PL301 in the core platform has 2 master and 1 slave bus connections.

- Network Interconnect (PL301)
  - 2 on-platform masters, 64-bit AHB interfaces
  - 1 off-platform slave, 64-bit AXI interface

4.2.3 Cortex-M4 Instruction Fetches on the System Bus

The Cortex-M4 processors implement multiple 32-bit bus interfaces that support a Harvard memory architecture. Specifically, the cores provide a modified Harvard connection with 2-cycle pipelined AMBA-AHB code and system buses. The modified Harvard memory architecture results since the bus interfaces are activated by address range and include both instruction fetches and operand data references on a given bus port. A traditional Harvard architecture separates instruction fetches and operand data references onto specific bus ports regardless of access address.

The code bus is typically used for instruction fetching and data accesses of PC-relative data, while the system bus is typically used for operand data references to the on- and off-chip memories and peripheral accesses. This bus structure fully supports concurrent instruction fetch and data accesses, but the Cortex-M4 implementations can generate both types of references on each bus. Additionally, there is a separate 32-bit Private Peripheral Bus (PPB) connection to several important modules (for example, the Nested Vectored Interrupt Controller) accessible to only the core. By placing the various code and data sections in the appropriate locations within the memory map, overall system performance can be maximized.

To provide a “clean timing interface” on the core's system bus, instruction and vector fetch requests to this bus are registered. This increases fetch time by an additional cycle of latency because instructions fetched from the system bus take a minimum of two cycles. This also means that back-to-back instruction fetches from the system bus are not possible.

Instruction fetch requests to the code bus are not registered. It is recommended that performance critical code be located such that it fetches from the ICode bus interface as defined by addresses < 0x2000_0000 (the system bus interface includes the addresses ≥ 0x2000_0000 and < 0xE000_0000 and the Private Peripheral Bus is used for addresses ≥ 0xE000_0000).

NOTE

In the device, the memory map includes aliased address spaces that are mapped into the ICode region for code sections that reside in the system address space. As a simple example, the DDR address space is located in the system region of the memory map, but a subset of this space is aliased so that it appears in the ICode region that instructions mapped into the DDR space can be executed as maximum performance.

4.2.4 Major Platform Bus Interfaces

- The platform supports the AMBA AHB and AXI bus protocol.
  - HBSTRB is the only v6 extension that is supported
  - HRESP[1] (SPLIT/RETRY) is not supported.
  - HTRANS[1:0] = 2b01 (BUSY) is not supported
  - HBURST - The platform and its memory controllers support the full range of AHB burst sizes
  - M0, M1 are 64-bit AHB-lite master bus interfaces, S1 is 64-bit AXI slave bus interface
  - Support AXI standard exclusive access on system bus address (above 0x2000_0000). In MX7ULT1, only a small region of MMDC support it.

4.2.5 Cortex-M4 Boot Requirements

- Cortex-A7 always boots as the primary core.
- Cortex-M4 does not have a boot ROM and at POR is not provided a clock
- Cortex-A7 user code is responsible for the following:
  - Loading and authenticating Cortex-M4 firmware by HAB API or with Cortex-A7 firmware together as a unified image by boot ROM.
  - Launching the Cortex-M4 by enabling its clock and clearing its reset bit in SRC (See System Reset Controller Chapter for more details).

4.2.6 Clocks and Resets

The platform inputs several reset signals. The following table describes the use of each reset.

Table 4-5. Platform Reset Descriptions

Reset Name	Description
ipg_core_async_reset_b	asynchronous reset for CORTEX-M4
ipg_hard_async_reset_b	asynchronous system reset for platform modules
ipg_hard_async_po_reset_b	asynchronous power-on reset used in CM4 Core
debug_reset_b	debug reset

The platform inputs several clocks. The following table describes each clock input:

Table 4-6. Platform Clock Descriptions

Clock Input Name	Description
dap_clk	DAP Bus Clock
tcmc_hclk	TCMC Clock
cm4_cti_clk	CTI Clock
cm4_hclk	Gated CPU Clock. Platform output "cm4_gate_hclk" can be used as the enable signal.
cm4_fclk	Free-running CPU Clock
ipg_clk_nic	Gated PL301 Clock

The platform clocks, cm4_fclk, cm4_hclk, tcmc_hclk and ipg_clk_nic must be equal in frequency and phase.

4.2.7 Debug Configuration

The following table presents a brief description of each one of the debug components.

Table 4-7. Debug Components Description

Module	Description
AHB-AP	AHB Master Interface from JTAG to debug module and SOC system memory maps
ROM Table	Identifies which debug IP is available.
Core Debug	Singlestep, Register Access, Run, Core Status
CoreSight ATB Funnel	The Funnel combines multiple trace streams onto a single ATB bus.

Table continues on the next page...

Table 4-7. Debug Components Description (continued)

Module	Description
CoreSight ATB Upsizer	The ATB upsizer converts the ATB bus from 8-bit to 32-bit or from 32-bit to 64-bit.
CoreSight ATB Async Bridge	The ATB asynchronous bridge is the interface to different trace clock domain.
CoreSight CTI	Cross trigger interface to M4
ETM (Embedded Trace Macrocell)	ETMv3.5 Architecture
ITM	S/W Instrumentation Messaging + Simple Data Trace Messaging + Watchpoint Messaging
DWT (Data and Address Watchpoints)	4 data and address watchpoints
FPB (Flash Patch and Breakpoints)	The FPB implements hardware breakpoints and patches code and data from code space to system space. The FPB unit contains two literal comparators for matching against literal loads from Code space, and remapping to a corresponding area in System space. The FPB also contains six instruction comparators for matching against instruction fetches from Code space, and remapping to a corresponding area in System space. Alternatively, the six instruction comparators can individually configure the comparators to return a Breakpoint Instruction (BKPT) to the processor core on a match, so providing hardware breakpoint capability.
MCM (Miscellaneous Control Module)	The MCM provides miscellaneous control functions.

4.2.8 Platform JTAG Requirements

The Miscellaneous Debug Module (MDM) contains the DAP (Debug Access Port) status, control and ID registers as well as the DAP mux. Access to these DAP registers is through the SWJ-DP. The read-only DAP status register is located at DAPADDR[31:0] = 32'h0100_0000, while the DAP control register is located at DAPADDR=32'h0100_0004. The IDR is located at DAPADDR = 32'h0100_00FC.

The platform's 32-bit *dap_status[31:1]* input vector is registered in the MDM. Bit 0 is reserved for platform itself.

The *mdm_ap_control[31:0]* platform output vector controls SoC-defined functions. The *mdm_ap_control[0]* is reserved for platform itself.

The IDR is a read-only register with a value of 32'h001C_0000.

4.2.9 Local Memory Controller (LMEM)

The Local Memory Controller provides the ARM®Cortex-M4™ processor with tightly-coupled processor-local memories and bus paths to all slave memory spaces.

4.2.9.1 LMEM Block Diagram

The Cortex-M4 processor has a modified 32-bit Harvard bus architecture. Using a 32-bit address space, low-order addresses (0x0000_0000 through 0x1FFF_FFFF) use the Processor Code (PC) bus, and high-order addresses (0x2000_0000 through 0xFFFF_FFFF) use the Processor System (PS) bus. As the bus names imply, normal operation has code accesses on the PC bus and data accesses on the PS bus.

This device has been augmented with tightly-coupled memories for the PC and PS buses. The memories include RAMs and caches. These local memories provide zero wait state access to RAM and cacheable address spaces.

The local memory controller includes four memory controllers and their attached memories:

- SRAM lower (SRAM_L) controller via the PC bus
- SRAM upper (SRAM_U) controller via the PS bus
- Cache memory controller via the PC bus
- Cache memory controller via the PS bus

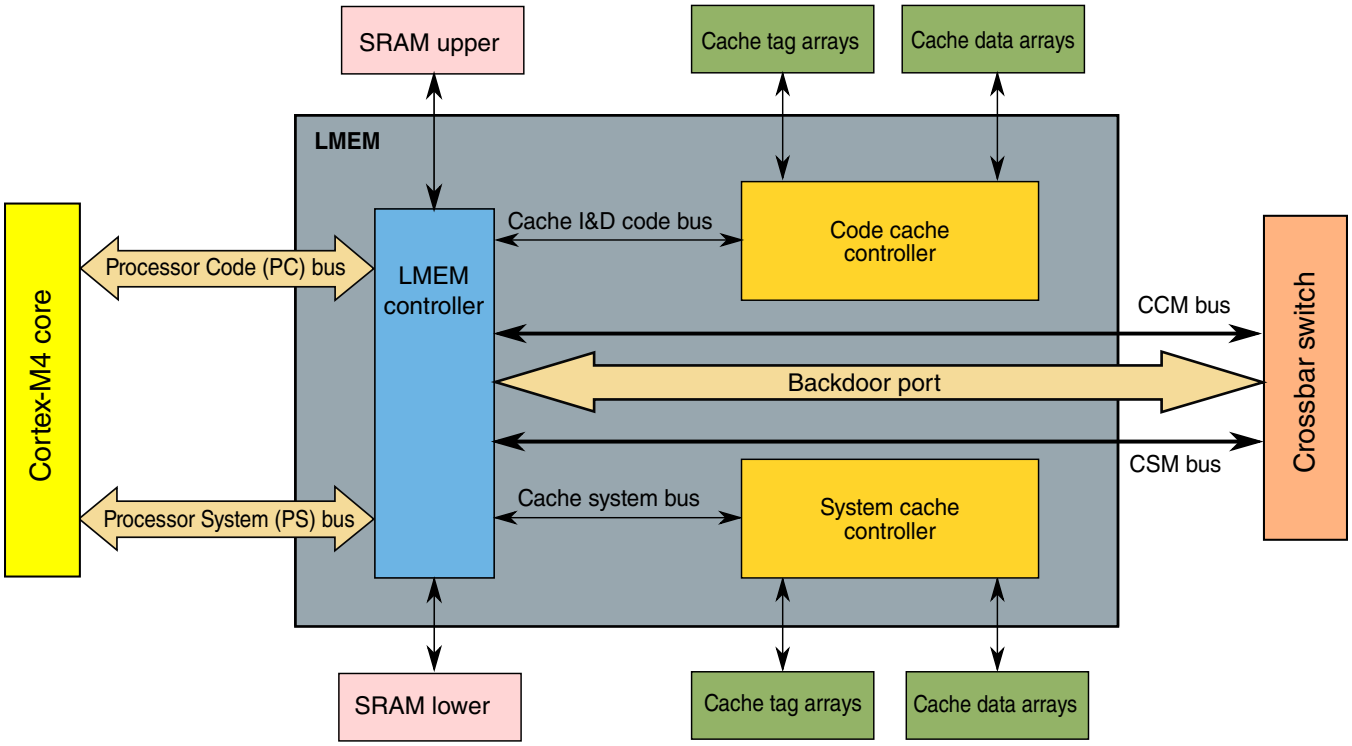


Figure 4-3. Local memory controller block diagram

NOTE

The SRAM and cache controllers reside within the LMEM, but the single-port synchronous RAM arrays used by these controllers are external.

The LMEM contains address decode logic for the PC and PS buses. This logic routes the core's accesses to the various system resources. The address spaces are device-specific and are specified in the device's Chip Configuration chapter.

4.2.9.2 Cache features

A cache is a block of high-speed memory locations containing address information (commonly known as a tag) and the associated data. The purpose is to decrease the average time of a memory access. Caches operate on two principles of locality:

- Spatial locality — An access to one location is likely to be followed by accesses from adjacent locations (for example, sequential instruction execution or usage of a data structure).
- Temporal locality — An access to an area of memory is likely to be repeated within a short time period (for example, execution of a code loop).

To minimize the quantity of control information stored, the spatial locality property is used to group several locations together under the same tag. This logical block is commonly known as a cache line.

When data is loaded into a cache, access times for subsequent loads and stores are reduced, resulting in overall performance benefits. An access to information already in a cache is known as a cache hit, and other accesses are called cache misses.

Normally, caches are self-managing, with the updates occurring automatically. Whenever the processor wants to access a cacheable location, the cache is checked. If the access is a cache hit, the access occurs immediately. Otherwise, a location is allocated and the cache line is loaded from memory. Different cache topologies and access policies are possible. However, they must comply with the memory coherency model of the underlying architecture.

Caches introduce a number of potential problems, mainly because of:

- memory accesses occurring at times other than when the programmer would normally expect them,
- the existence of multiple physical locations where a data item can be held.

The local memory controller supports three modes of operation:

1. Write-through — access to address spaces with this cache mode are cacheable.
    - A read miss on the input bus causes a line read on the output bus of a 32-byte-aligned memory address containing the desired address. This miss data is loaded into the cache and is marked as valid and not modified.
    - A write-through read hit to a valid cache location returns data from the cache with no output bus access.
    - A write-through write miss bypasses the cache and writes to the output bus (no allocate on write miss policy for write-through mode spaces).
    - A write-through write hit updates the cache hit data and writes to the output bus.
2. Write-back — access to address spaces with this cache mode are cacheable.
    - A write-back read miss on the input bus will cause a line read on the output bus of a 32-byte-aligned memory address containing the desired address. This miss data is loaded into the cache and marked as valid and not modified.
    - A write-back read hit to a valid cache location will return data from the cache with no output bus access.
    - A write-back write miss will do a "read-to-write" (allocate on write miss policy for write-back mode spaces). A line read on the output bus of a 16 byte aligned memory address containing the desired write address is performed. This miss data is loaded into the cache and marked as valid and modified; and the write data will then update the appropriate cache data locations.

3. Non-cacheable — access to address spaces with this cache mode are not cacheable. These accesses bypass the cache and access the output bus.

4.2.9.3 LMEM Function

The LMEM receives the following requests:

- Core master bus requests on the Processor Code (PC) bus,
- Core master bus requests on the Processor Space (PS) bus, and
- SRAM controller requests from all other bus masters on the backdoor port.

The LMEM address decode logic routes these accesses and also provides any crossbar switch slave target logic. Finally, the Local Memory controller provides the needed MPU connections for checking all SRAM controller and cacheable accesses.

The programming model for the Code and System Caches is accessed via the core's Private Peripheral Bus (PPB).

4.2.9.3.1 Processor Code accesses

Processor Code accesses are routed to the SRAM_L if they are mapped to that space. All other PC accesses are routed to the Code Cache Memory Controller. This controller then processes the cacheable accesses as needed, while bypassing the non-cacheable, cache write-through, cache miss, and cache maintenance accesses to the CCM bus and the crossbar switch using the Master0 port.

4.2.9.3.2 Processor Space accesses

Processor Space accesses are routed to the SRAM_U if they are mapped to that space. All other PS accesses are routed to the PS Cache Memory Controller. This controller then processes the cacheable accesses as needed, while bypassing the non-cacheable, cache write-through, cache miss, and cache maintenance accesses to the CCM bus and the crossbar switch using the Master1 port.

4.2.9.3.3 Backdoor port accesses

All LMEM backdoor port accesses are for the SRAM controller. These accesses go to the SRAM_L or the SRAM_U depending on their specific address.

4.2.9.3.4 SRAM Function

4.2.9.3.4.1 SRAM Configuration

The figure below shows how the SRAM controller is configured.

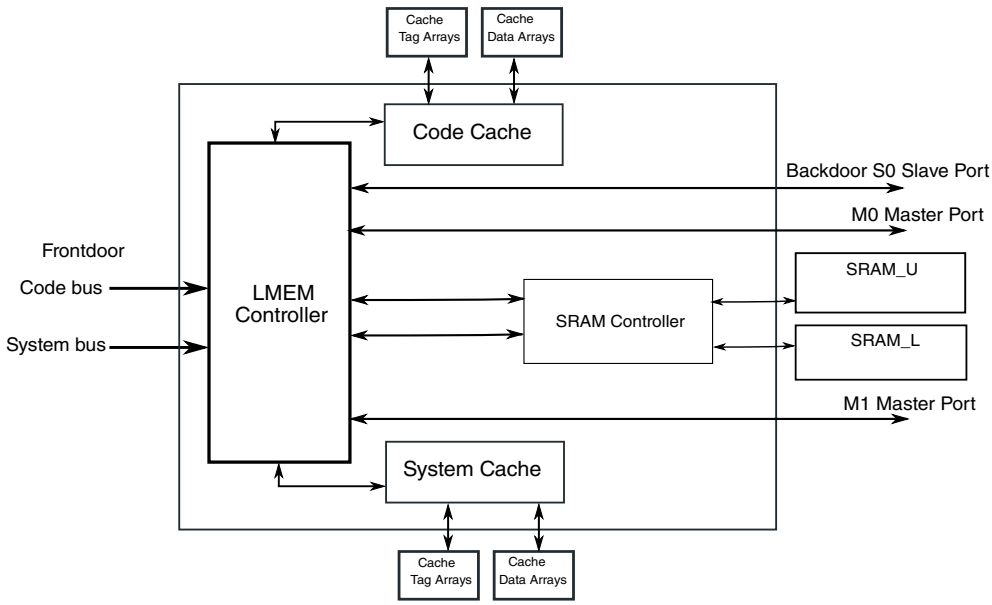


Figure 4-4. SRAM Configuration

4.2.9.3.4.2 SRAM Arrays

The on-chip SRAM is split into two logical arrays, SRAM_L and SRAM_U.

From equal-sized memories, valid address ranges for SRAM_L and SRAM_U are then defined as:

- SRAM_L = 0x1FFF_8000 - (0x1FFF_8000 + SRAM_size/2)
- SRAM_U = 0x2000_0000 - (0x2000_0000 + SRAM_size/2)

4.2.9.3.4.3 SRAM accesses

The SRAM is split into two logical arrays that are 64-bits wide:

- SRAM_L — Accessible by the code bus of the Cortex-M4 core and by the backdoor port.
- SRAM_U — Accessible by the system bus of the Cortex-M4 core and by the backdoor port.

The backdoor port makes the SRAM accessible to the non-core bus masters (such as DMA).

The figure below illustrates the SRAM accesses within the device.

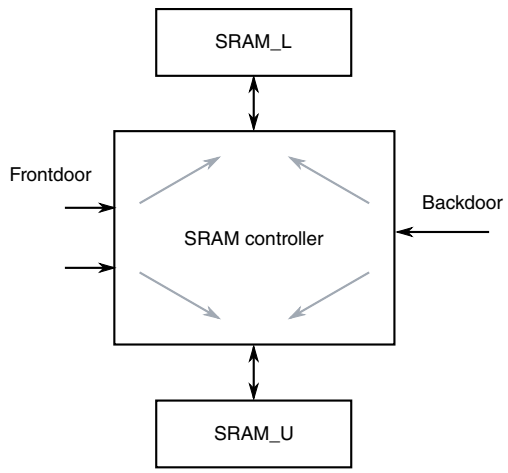


Figure 4-5. SRAM access diagram

The following simultaneous accesses can be made to different logical halves of the SRAM:

- Core code and core system
- Core code and non-core master
- Core system and non-core master

NOTE

Two non-core masters cannot access SRAM simultaneously. The required arbitration and serialization is provided by the crossbar switch. The SRAM_{L,U} arbitration is controlled by the SRAM controller based on the configuration bits in the MCM module.

4.2.9.3.5 Cache Function

The caches on this device are structured as follows. Both caches have a 2-way set-associative cache structure with a total size of 32 KBytes. The caches have a 32-bit address, 64-bit data paths and a 32-byte line size. The cache tags and data storage use single-port, synchronous RAMs.

For these 16-KByte caches, each cache TAG function uses two 256 x 22-bit RAM arrays and the cache DATA function uses two 1024 x 32-bit RAM arrays. The cache TAG entries store 20 bits of upper address as well as a modified and valid bit per cache line. The cache DATA entries store eight byes of code or data.

All normal cache accesses use physical addresses. This leads to the following cache address use:

CACHE - 16 KByte size = (256 sets) x (32-byte lines) x (2-way set associative)

TAG:

- Only address[31:29] and [21:13] are implemented. All other bits are tied to zero. Only the memories below are supported:
  - address[31] - DDR space, first 2M (0x8000_0000 - 0x801F_FFFF)
  - address[30:29] - QSPI channel A, first 2M (0x6000_0000 - 0x601F_FFFF)
  - address[30:29]+[21] - QSPI channel A, second 2M (0x6020_0000 - 603F_FFFF)
  - address[29]+[21] - OCRAM (0x2020_0000 - 0x203F_FFFF)

NOTE

To use cache, user needs to configure MPU to set those memories as cacheable and all the other memories set as non-cacheable.

DATA

- address[31:13] not used
- address[12:5] used to select one of 256 sets
- address[4:2] used to select one of eight 32-bit words within a set
- address[1:0] used to select the byte within the 32-bit word

4.2.9.3.6 Cache Control

The Code and System Caches are disabled at reset. Cache tag and data arrays are not cleared at reset. Therefore, to enable the caches, cache commands must be done to clear and initialize the required tag array bits and to configure and enable the caches.

4.2.9.3.6.1 Cache set commands

The cache set commands may operate on:

- all of way 0,
- all of way 1, or
- all of both ways (complete cache).

Cache set commands are initiated using the upper bits in the CCR register. Cache set commands perform their operation on the cache independent of the cache enable bit, CCR[ENCACHE].

A cache set command is initiated by setting the CCR[GO] bit. This bit also acts as a busy bit for set commands. It stays set while the command is active and is cleared by the hardware when the set command completes.

Supported cache set commands are given in the table below. Set commands work as follows:

- Invalidate – Unconditionally clear valid and modify bits of a cache entry.
- Push – Push a cache entry if it is valid and modified, then clear the modify bit. If entry not valid or not modified, leave as is.
- Clear – Push a cache entry if it is valid and modified, then clear the valid and modify bits. If entry not valid or not modified, clear the valid bit.

Table 4-8. Cache Set Commands

CCR[27:24]				Command
PUSHW1	INVW1	PUSHW0	INVW0	
0	0	0	0	NOP
0	0	0	1	Invalidate all way 0
0	0	1	0	Push all way 0
0	0	1	1	Clear all way 0
0	1	0	0	Invalidate all way 1
0	1	0	1	Invalidate all way 1; invalidate all way 0 (invalidate cache)
0	1	1	0	Invalidate all way 1; push all way 0
0	1	1	1	Invalidate all way 1; clear all way 0
1	0	0	0	Push all way 1
1	0	0	1	Push all way 1; invalidate all way 0
1	0	1	0	Push all way 1; push all way 0 (push cache)
1	0	1	1	Push all way 1; clear all way 0
1	1	0	0	Clear all way 1
1	1	0	1	Clear all way 1; invalidate all way 0
1	1	1	0	Clear all way 1; push all way 0
1	1	1	1	Clear all way 1; clear all way 0 (clear cache)

After a reset, complete an invalidate cache command before using the cache. It is possible to combine the cache invalidate command with the cache enable. That is, setting CCR to 0x8500_0003 will invalidate the cache and enable the cache and write buffer.

4.2.9.3.6.2 Cache line commands

Cache line commands operate on a single line in the cache at a time. Cache line commands can be performed using a physical or cache address.

- A cache address consists of a set address and a way select. The line command acts on the specified cache line.
- Cache line commands with physical addresses first search both ways of the cache set specified by bits [11:4] of the physical address. If they hit, the commands perform their action on the hit way.

Cache line commands are specified using the upper bits in the CLCR register. Cache line commands perform their operation on the cache independent of the cache enable bit (CCR[ENCACHE]). Using a cache address, the command can be completely specified using the CLCR register. Using a physical address, the command must also use the CSAR register to specify the physical address.

A line cache command is initiated by setting the line command go bit (CLCR[LGO] or CSAR[LGO]). This bit also acts a a busy bit for line commands. It stays set while the command is active and is cleared by the hardware when the command completes.

The CLCR[27:24] bits select the line command as follows:

Table 4-9. Cache Line Commands

CLCR[27:24]			Command
LACC	LADSEL	LCMD	
0	0	00	Search by cache address and way
0	0	01	Invalidate by cache address and way
0	0	10	Push by cache address and way
0	0	11	Clear by cache address and way
0	1	00	Search by physical address
0	1	01	Invalidate by physical address
0	1	10	Push by physical address
0	1	11	Clear by physical address
1	0	00	Write by cache address and way
1	0	01	Reserved, NOP
1	0	10	Reserved, NOP
1	0	11	Reserved, NOP
1	1	*xx*	Reserved, NOP

4.2.9.3.6.2.1 Executing a series of line commands using cache addresses

A series of line commands with incremental cache addresses can be performed by just writing to the CLCR.

- Place the command in CLCR[27:24],
- Set the way (CLCR[WSEL]) and tag/data (CLCR[TDSEL]) controls as needed,

- Place the cache address in CLCR[CACHEADDR], and
- Set the line command go bit (CLCR[LGO]).

When one line command completes, initiate the next command by following these steps:

- Increment the cache address (at bit 2 to step through data or at bit 4 to step through lines), and
- Set the line command go bit (CLCR[LGO]).

4.2.9.3.6.2.2 Executing a series of line commands using physical addresses

Perform a series of line commands with incremental physical addresses using the following steps:

- Write to the CLCR.
  - Place the command in CLCR[27:24]
  - Set the tag/data (CLCR[TDSEL]) control
- Place the physical address in CSAR[PHYADDR] and set the line command go bit (CSAR[LGO]).

When one line command completes, initiate the next command by following these steps:

- Increment the physical address (at bit 2 to step through data or at bit 4 to step through lines), and
- Set the line command go bit (CSAR[LGO]).

The line command go bit is shared between the CLCR and CSAR registers, so that the above steps can be completed in a single write to the CSAR register.

4.2.9.3.6.2.3 Line command results

At completion of a line command, the CLCR register contains information on the initial state of the line targeted by the command. For line commands with cache addresses, this information is read before the line command action is performed from the targeted cache line. For line commands with physical addresses, this information is read on a hit before the line command action is performed from the hit cache line or has initial valid bit cleared if the command misses. In general, if the valid indicator (CLCR[LCIVB] is cleared, the targeted line was invalid at the start of the line command and no line operation was performed.

Table 4-10. Line command results

CLCR[22:20]			For cache address commands	For physical address commands
LCWAY	LCIMB	LCIVB		
0	0	0	Way 0 line was invalid	No hit
0	0	1	Way 0 valid, not modified	Way 0 valid, not modified
0	1	0	Way 0 line was invalid	No hit
0	1	1	Way 0 valid and modified	Way 0 valid and modified
1	0	0	Way 1 line was invalid	No hit
1	0	1	Way 1 valid, not modified	Way 1 valid, not modified
1	1	0	Way 1 line was invalid	No hit
1	1	1	Way 1 valid and modified	Way 1 valid and modified

At completion of a line command other than a write, the CCVR (Cache R/W Value Register) contains information on the initial state of the line tag or data targeted by the command. For line commands, CLCR[TDSEL] selects between tag and data. If the line command used a physical address and missed, the data is don't care. For write commands, the CCVR holds the write data.

4.2.10 Miscellaneous Control Module (MCM)

The Miscellaneous Control Module (MCM) provides a myriad of miscellaneous control functions.

4.2.10.1 MCM features

The MCM includes the following features:

- Program-visible information on the platform configuration and revision

4.2.10.2 MCM Interrupts

The MCM generates the following interrupt requests:

- Normal interrupt

4.2.10.2.1 Normal interrupt

The MCM's normal interrupt is generated if any of the following is true:

- ISCR[ETBI] is set, when
  - The ETB counter is enabled, ETBCC[CNTEN] = 1
  - The ETB count expires
  - The response to counter expiration is a normal interrupt, ETBCC[RSPT] = 01

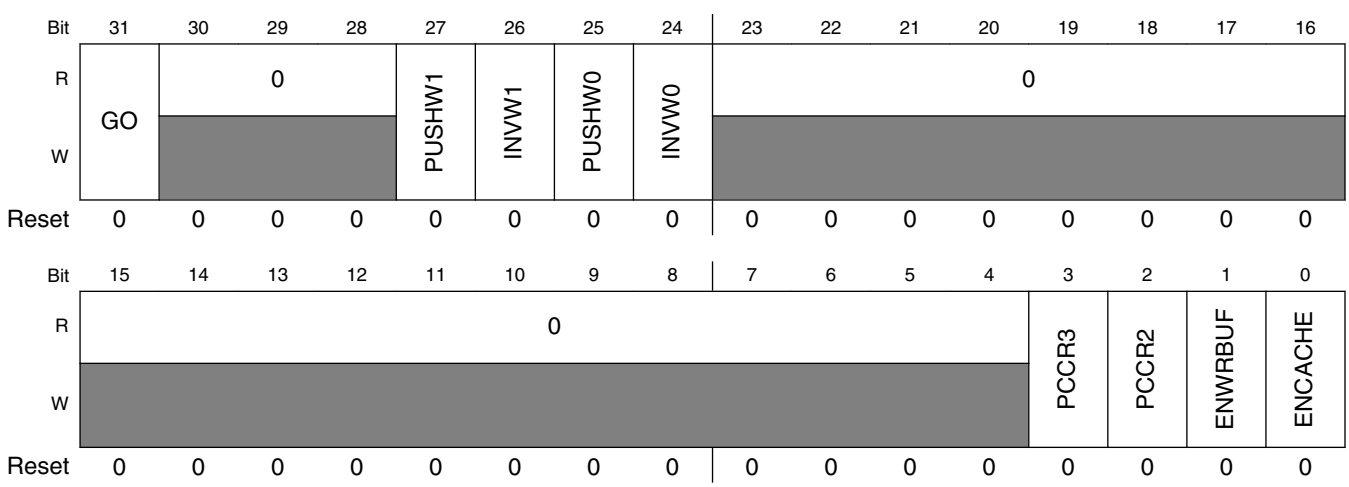
4.2.11 LMEM Memory Map/Register Definition

LMEM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E008_2000	Cache control register (LMEM_PCCCR)	32	R/W	0000_0000h	4.2.11.1/ 262
E008_2004	Cache line control register (LMEM_PCCLCR)	32	R/W	0000_0000h	4.2.11.2/ 264
E008_2008	Cache search address register (LMEM_PCCSAR)	32	R/W	0000_0000h	4.2.11.3/ 266
E008_200C	Cache read/write value register (LMEM_PCCCVR)	32	R/W	0000_0000h	4.2.11.4/ 267
E008_2800	Cache control register (LMEM_PSCCR)	32	R/W	0000_0000h	4.2.11.5/ 268
E008_2804	Cache line control register (LMEM_PSCLCR)	32	R/W	0000_0000h	4.2.11.6/ 269
E008_2808	Cache search address register (LMEM_PSCSAR)	32	R/W	0000_0000h	4.2.11.7/ 272
E008_280C	Cache read/write value register (LMEM_PSCCVR)	32	R/W	0000_0000h	4.2.11.8/ 273

4.2.11.1 Cache control register (LMEM_PCCCR)

Address: E008_2000h base + 0h offset = E008_2000h



LMEM_PCCCR field descriptions

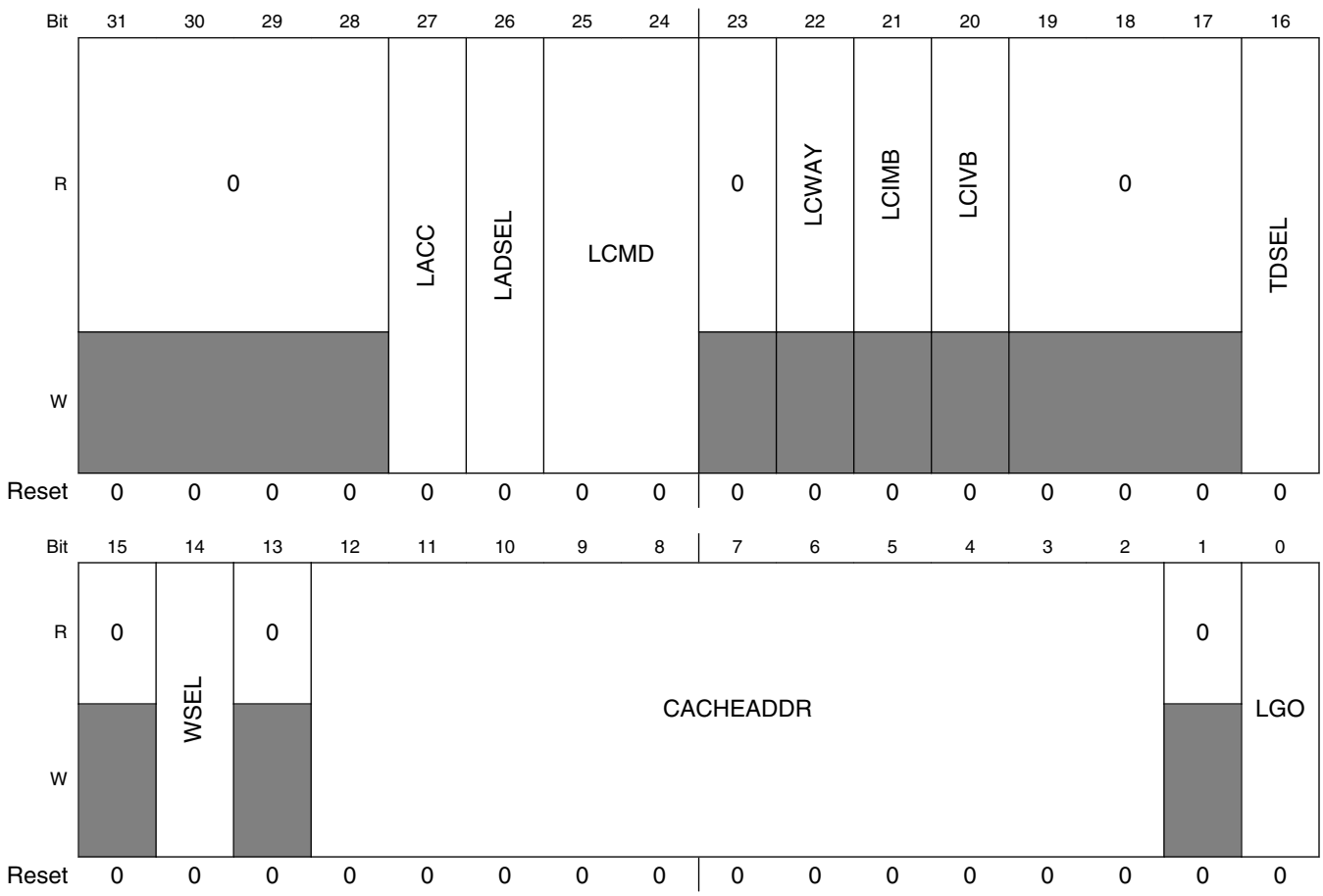
Field	Description
31 GO	Initiate Cache Command Setting this bit initiates the cache command indicated by bits 27-24. Reading this bit indicates if a command is active **NOTE:** This bit stays set until the command completes. Writing zero has no effect. 0 Write: no effect. Read: no cache command active. 1 Write: initiate command indicated by bits 27-24. Read: cache command active.
30–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 PUSHW1	Push Way 1 0 No operation 1 When setting the GO bit, push all modified lines in way 1
26 INVW1	Invalidate Way 1 **NOTE:** If the PUSHW1 and INVW1 bits are set, then after setting the GO bit, push all modified lines in way 1 and invalidate all lines in way 1 (clear way 1). 0 No operation 1 When setting the GO bit, invalidate all lines in way 1
25 PUSHW0	Push Way 0 0 No operation 1 When setting the GO bit, push all modified lines in way 0
24 INVW0	Invalidate Way 0 **NOTE:** If the PUSHW0 and INVW0 bits are set, then after setting the GO bit, push all modified lines in way 0 and invalidate all lines in way 0 (clear way 0). 0 No operation 1 When setting the GO bit, invalidate all lines in way 0.
23–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 PCCR3	Forces no allocation on cache misses (must also have ACCR2 asserted)
2 PCCR2	Forces all cacheable spaces to write through
1 ENWRBUF	Enable Write Buffer 0 Write buffer disabled 1 Write buffer enabled
0 ENCACHE	Cache enable 0 Cache disabled 1 Cache enabled

4.2.11.2 Cache line control register (LMEM_PCCLCR)

This register defines specific line-sized cache operations to be performed using a specific cache line address or a physical address.

If a physical address is specified, both ways of the cache are searched, and the command is only performed on the way which hits.

Address: E008_2000h base + 4h offset = E008_2004h



LMEM_PCCLCR field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LACC	Line access type 0 Read 1 Write
26 LADSEL	Line Address Select When using the cache address, the way must also be specified in CLCR[WSEL].

Table continues on the next page...

LMEM_PCCLCR field descriptions (continued)

Field	Description
	When using the physical address, both ways are searched and the command is performed only if a hit. 0 Cache address 1 Physical address
25–24 LCMD	Line Command 00 Search and read or write 01 Invalidate 10 Push 11 Clear
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 LCWAY	Line Command Way Indicates the way used by the line command.
21 LCIMB	Line Command Initial Modified Bit If command used cache address and way, then this bit shows the initial state of the modified bit If command used physical address and a hit, then this bit shows the initial state of the modified bit. If a miss, this bit reads zero.
20 LCIVB	Line Command Initial Valid Bit If command used cache address and way, then this bit shows the initial state of the valid bit If command used physical address and a hit, then this bit shows the initial state of the valid bit. If a miss, this bit reads zero.
19–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 TDSEL	Tag/Data Select Selects tag or data for search and read or write commands. 0 Data 1 Tag
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 WSEL	Way select Selects the way for line commands. 0 Way 0 1 Way 1
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–2 CACHEADDR	Cache address CLCR[11:4] bits are used to access the tag arrays CLCR[11:2] bits are used to access the data arrays
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LMEM_PCCLCR field descriptions (continued)

Field	Description
0 LGO	Initiate Cache Line Command Setting this bit initiates the cache line command indicated by bits 27-24. Reading this bit indicates if a line command is active **NOTE:** This bit stays set until the command completes. Writing zero has no effect. **NOTE:** This bit is shared with CSAR[LGO] 0 Write: no effect. Read: no line command active. 1 Write: initiate line command indicated by bits 27-24. Read: line command active.

4.2.11.3 Cache search address register (LMEM_PCCSAR)

The CSAR register is used to define the explicit cache address or the physical address for line-sized commands specified in the CLCR[LADSEL] bit.

Address: E008_2000h base + 8h offset = E008_2008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PHYADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PHYADDR														0	LGO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LMEM_PCCSAR field descriptions

Field	Description
31–2 PHYADDR	Physical Address PHYADDR represents bits [31:2] of the system address. CSAR[31:12] bits are used for tag compare CSAR[11:4] bits are used to access the tag arrays CSAR[11:2] bits are used to access the data arrays
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 LGO	Initiate Cache Line Command Setting this bit initiates the cache line command indicated by bits 27-24. Reading this bit indicates if a line command is active **NOTE:** This bit stays set until the command completes. Writing zero has no effect. **NOTE:** This bit is shared with CLCR[LGO]

Table continues on the next page...

LMEM_PCCSAR field descriptions (continued)

Field	Description
	0 Write: no effect. Read: no line command active. 1 Write: initiate line command indicated by bits CLCR[27:24]. Read: line command active.

4.2.11.4 Cache read/write value register (LMEM_PCCCVR)

The CCVR register is used to source write data or return read data for the commands specified in the CLCR register.

Address: E008_2000h base + Ch offset = E008_200Ch

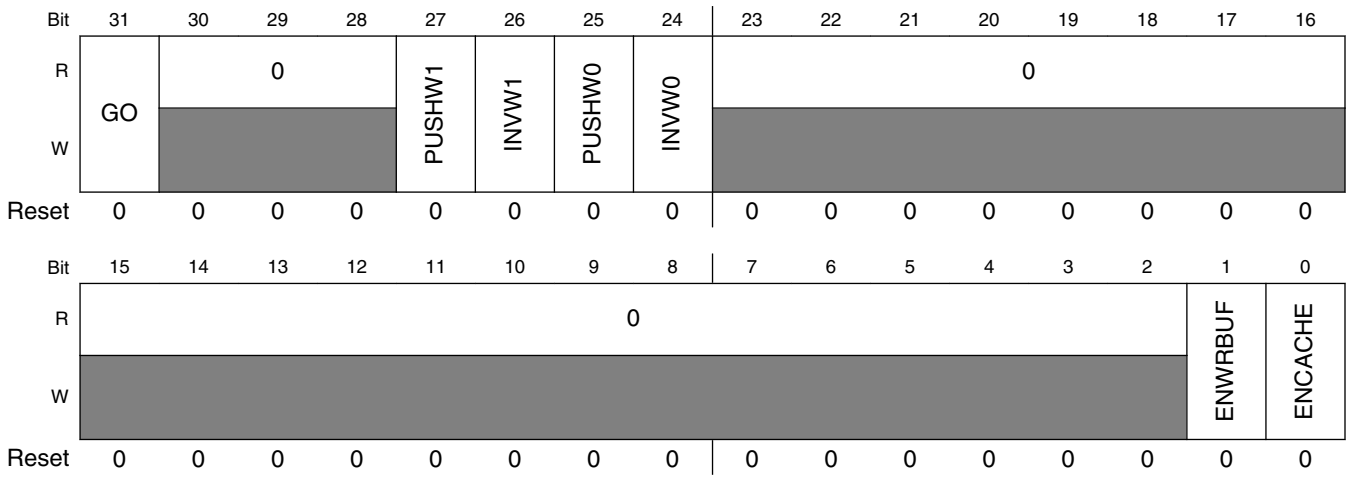
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LMEM_PCCCVR field descriptions

Field	Description
DATA	Cache read/write Data For tag search, read or write: • CCVR[31:12] bits are used for tag array R/W value • CCVR[11:4] bits are used for tag set address on reads; unused on writes • CCVR[3:2] bits are reserved For data search, read or write: • CCVR[31:0] bits are used for data array R/W value

4.2.11.5 Cache control register (LMEM_PSCCR)

Address: E008_2000h base + 800h offset = E008_2800h



LMEM_PSCCR field descriptions

Field	Description
31 GO	Initiate Cache Command Setting this bit initiates the cache command indicated by bits 27-24. Reading this bit indicates if a command is active **NOTE:** This bit stays set until the command completes. Writing zero has no effect. 0 Write: no effect. Read: no cache command active. 1 Write: initiate command indicated by bits 27-24. Read: cache command active.
30–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 PUSHW1	Push Way 1 0 No operation 1 When setting the GO bit, push all modified lines in way 1
26 INVW1	Invalidate Way 1 **NOTE:** If the PUSHW1 and INVW1 bits are set, then after setting the GO bit, push all modified lines in way 1 and invalidate all lines in way 1 (clear way 1). 0 No operation 1 When setting the GO bit, invalidate all lines in way 1
25 PUSHW0	Push Way 0 0 No operation 1 When setting the GO bit, push all modified lines in way 0
24 INVW0	Invalidate Way 0

Table continues on the next page...

LMEM_PSCCR field descriptions (continued)

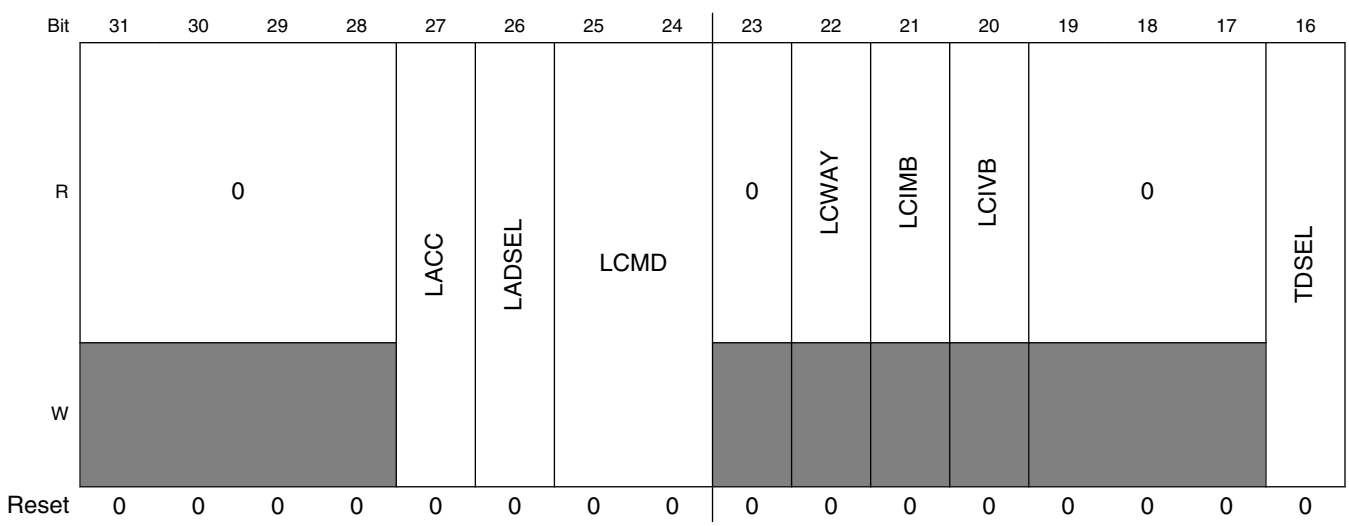
Field	Description
	NOTE: If the PUSHW0 and INVW0 bits are set, then after setting the GO bit, push all modified lines in way 0 and invalidate all lines in way 0 (clear way 0). 0 No operation 1 When setting the GO bit, invalidate all lines in way 0.
23–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 ENWRBUF	Enable Write Buffer 0 Write buffer disabled 1 Write buffer enabled
0 ENCACHE	Cache enable 0 Cache disabled 1 Cache enabled

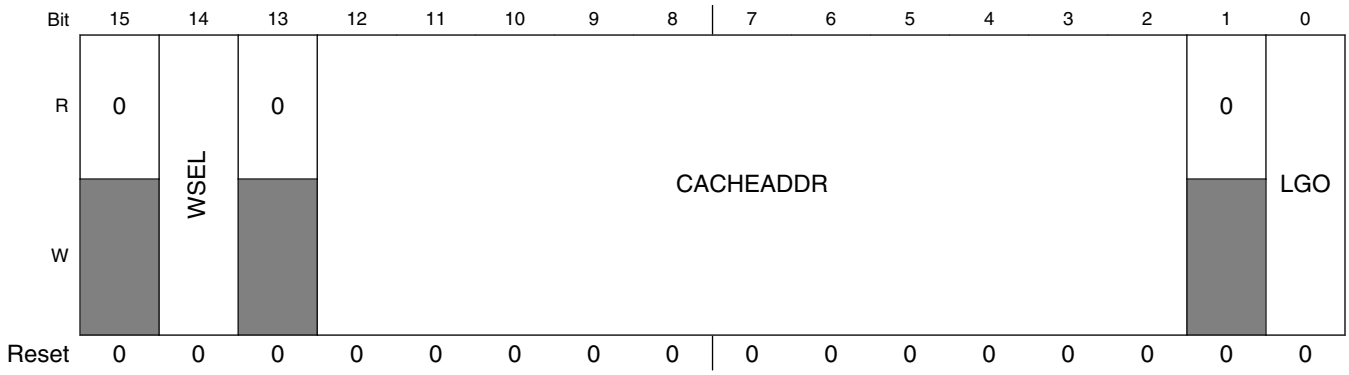
4.2.11.6 Cache line control register (LMEM_PSCLCR)

This register defines specific line-sized cache operations to be performed using a specific cache line address or a physical address.

If a physical address is specified, both ways of the cache are searched, and the command is only performed on the way which hits.

Address: E008_2000h base + 804h offset = E008_2804h





LMEM_PSCLCR field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LACC	Line access type 0 Read 1 Write
26 LADSEL	Line Address Select When using the cache address, the way must also be specified in CLCR[WSEL]. When using the physical address, both ways are searched and the command is performed only if a hit. 0 Cache address 1 Physical address
25–24 LCMD	Line Command 00 Search and read or write 01 Invalidate 10 Push 11 Clear
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 LCWAY	Line Command Way Indicates the way used by the line command.
21 LCIMB	Line Command Initial Modified Bit If command used cache address and way, then this bit shows the initial state of the modified bit If command used physical address and a hit, then this bit shows the initial state of the modified bit. If a miss, this bit reads zero.
20 LCIVB	Line Command Initial Valid Bit If command used cache address and way, then this bit shows the initial state of the valid bit If command used physical address and a hit, then this bit shows the initial state of the valid bit. If a miss, this bit reads zero.
19–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LMEM_PSCLCR field descriptions (continued)

Field	Description
16 TDSEL	Tag/Data Select Selects tag or data for search and read or write commands. 0 Data 1 Tag
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 WSEL	Way select Selects the way for line commands. 0 Way 0 1 Way 1
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–2 CACHEADDR	Cache address CLCR[11:4] bits are used to access the tag arrays CLCR[11:2] bits are used to access the data arrays
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 LGO	Initiate Cache Line Command Setting this bit initiates the cache line command indicated by bits 27-24. Reading this bit indicates if a line command is active **NOTE:** This bit stays set until the command completes. Writing zero has no effect. **NOTE:** This bit is shared with CSAR[LGO] 0 Write: no effect. Read: no line command active. 1 Write: initiate line command indicated by bits 27-24. Read: line command active.

4.2.11.7 Cache search address register (LMEM_PSCSAR)

The CSAR register is used to define the explicit cache address or the physical address for line-sized commands specified in the CLCR[LADSEL] bit.

Address: E008_2000h base + 808h offset = E008_2808h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PHYADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PHYADDR														0	LGO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

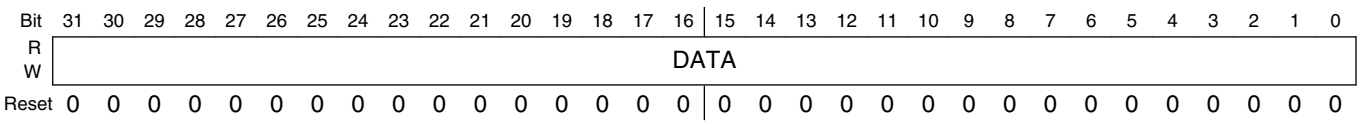
LMEM_PSCSAR field descriptions

Field	Description
31–2 PHYADDR	Physical Address PHYADDR represents bits [31:2] of the system address. CSAR[31:12] bits are used for tag compare CSAR[11:4] bits are used to access the tag arrays CSAR[11:2] bits are used to access the data arrays
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 LGO	Initiate Cache Line Command Setting this bit initiates the cache line command indicated by bits 27-24. Reading this bit indicates if a line command is active **NOTE:** This bit stays set until the command completes. Writing zero has no effect. **NOTE:** This bit is shared with CLCR[LGO] 0 Write: no effect. Read: no line command active. 1 Write: initiate line command indicated by bits CLCR[27:24]. Read: line command active.

4.2.11.8 Cache read/write value register (LMEM_PSCCVR)

The CCVR register is used to source write data or return read data for the commands specified in the CLCR register.

Address: E008_2000h base + 80Ch offset = E008_280Ch



LMEM_PSCCVR field descriptions

Field	Description
DATA	Cache read/write Data For tag search, read or write: • CCVR[31:12] bits are used for tag array R/W value • CCVR[11:4] bits are used for tag set address on reads; unused on writes • CCVR[3:2] bits are reserved For data search, read or write: • CCVR[31:0] bits are used for data array R/W value

4.2.12 MCM Memory Map/Register Definition

MCM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E008_0008	Crossbar Switch (AXBS) Slave Configuration (MCM_PLASC)	16	R	0002h	4.2.12.1/ 274
E008_000A	Crossbar Switch (AXBS) Master Configuration (MCM_PLAMC)	16	R	0003h	4.2.12.2/ 274
E008_000C	Crossbar Switch (AXBS) Control Register (MCM_PLACR)	32	R/W	0000_0000h	4.2.12.3/ 275
E008_0020	Fault address register (MCM_FADR)	32	R	Undefined	4.2.12.4/ 275
E008_0024	Fault attributes register (MCM_FATR)	32	R	Undefined	4.2.12.5/ 276
E008_0028	Fault data register (MCM_FDR)	32	R	Undefined	4.2.12.6/ 278

4.2.12.1 Crossbar Switch (AXBS) Slave Configuration (MCM_PLASC)

PLASC is a 16-bit read-only register identifying the presence/absence of bus slave connections to the device's crossbar switch.

Address: E008_0000h base + 8h offset = E008_0008h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								ASC							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

MCM_PLASC field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ASC	Each bit in the ASC field indicates whether there is a corresponding connection to the crossbar switch's slave input port. 0 A bus slave connection to AXBS input port *n* is absent 1 A bus slave connection to AXBS input port *n* is present

4.2.12.2 Crossbar Switch (AXBS) Master Configuration (MCM_PLAMC)

PLAMC is a 16-bit read-only register identifying the presence/absence of bus master connections to the device's crossbar switch.

Address: E008_0000h base + Ah offset = E008_000Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								AMC							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

MCM_PLAMC field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

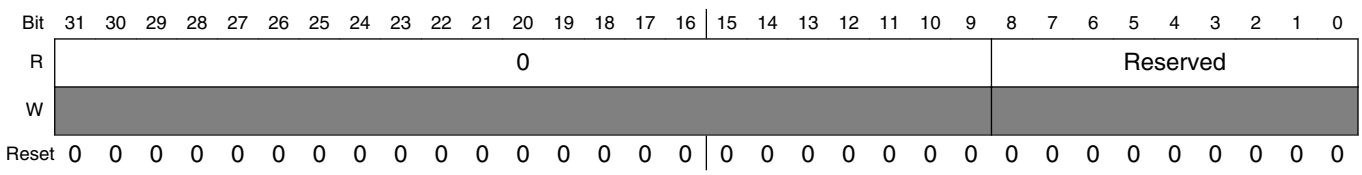
MCM_PLAMC field descriptions (continued)

Field	Description
AMC	Each bit in the AMC field indicates whether there is a corresponding connection to the AXBS master input port. 0 A bus master connection to AXBS input port *n* is absent 1 A bus master connection to AXBS input port *n* is present

4.2.12.3 Crossbar Switch (AXBS) Control Register (MCM_PLACR)

The PLACR register selects the arbitration policy for the crossbar masters.

Address: E008_0000h base + Ch offset = E008_000Ch



MCM_PLACR field descriptions

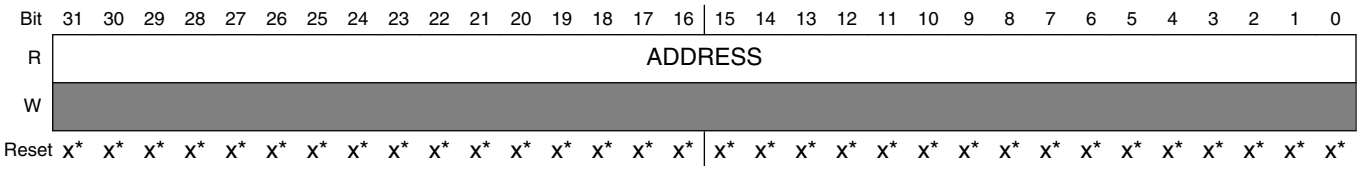
Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
Reserved	This field is reserved.

4.2.12.4 Fault address register (MCM_FADR)

When a properly-enabled cache write buffer error interrupt event is detected, the faulting address is captured in the MCM_FADR register. The MCM logic supports capturing a single cache write buffer bus error event; if a subsequent error is detected before the captured error information has been read from the corresponding registers and the MCM_ISCR[CWBER] indicator cleared, the MCM_FATR[BEOVR] flag is set. However, no additional information is captured.

The bits in this register are set by hardware and signaled by the assertion of MCM_ISCR[CWBER]. Attempted writes to this location are terminated with an error.

Address: E008_0000h base + 20h offset = E008_0020h



* Notes:
- x = Undefined at reset.

MCM_FADR field descriptions

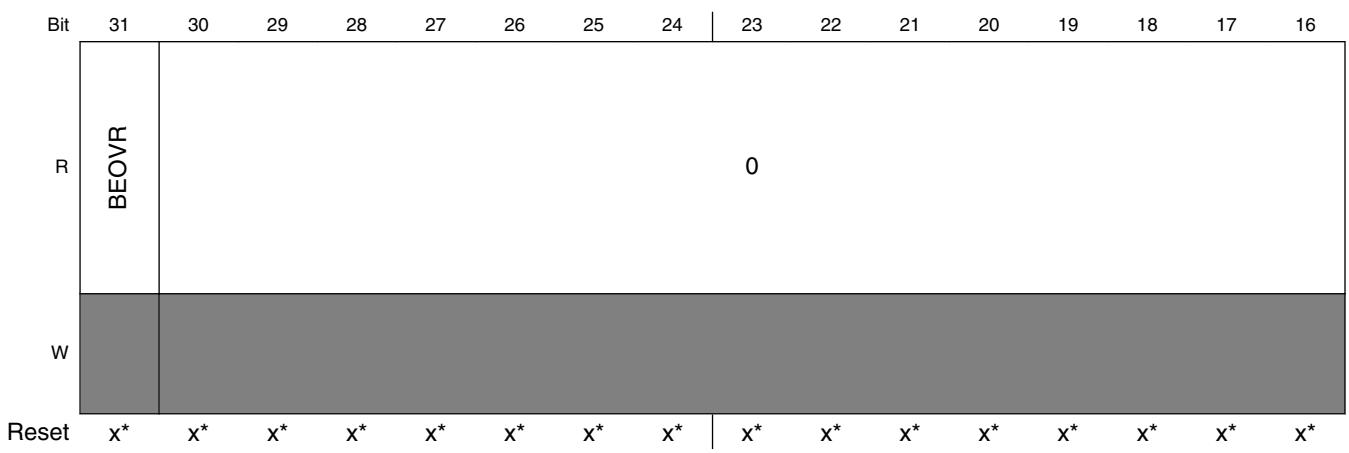
Field	Description
ADDRESS	Fault address

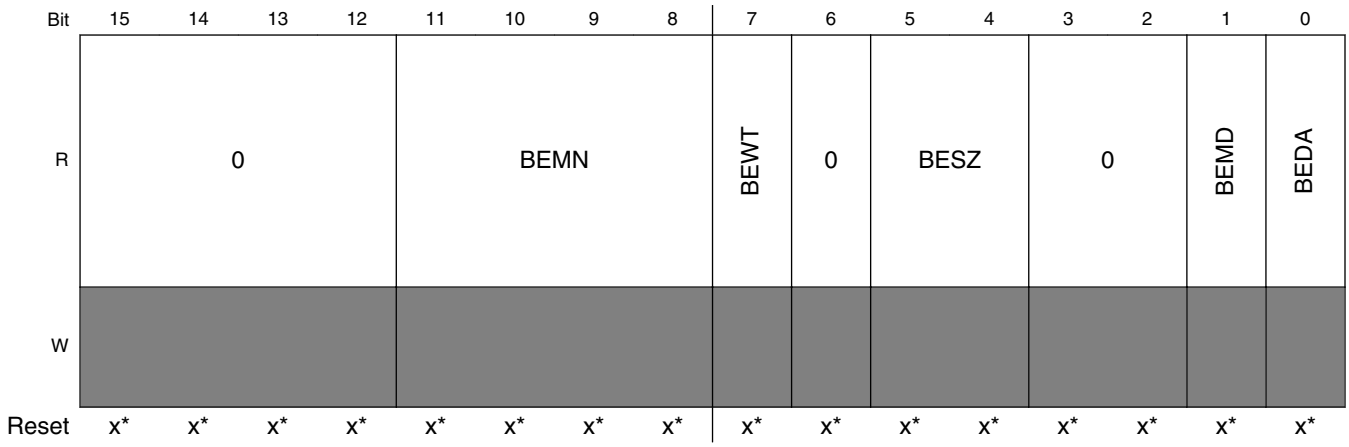
4.2.12.5 Fault attributes register (MCM_FATR)

When a properly-enabled cache write buffer error interrupt event is detected, the faulting attributes are captured in the MCM_FATR register.

The bits in this register are set by hardware and signaled by the assertion of MCM_ISCR[CWBER]. Attempted writes to this location are terminated with an error.

Address: E008_0000h base + 24h offset = E008_0024h





* Notes:
- x = Undefined at reset.

MCM_FATR field descriptions

Field	Description
31 BEOVR	Bus error overrun Indicates if anothercache write buffer bus error is detected before system software has retrieved all the error information from the original event, this overrun flag is set. The window of time is defined from the detection of the original cache write buffer error termination until the MCM_ISCR[CWBER] is written with a 1 to clear it and rearm the capture logic. This bit is set by the hardware and cleared whenever software writes a 1 to the CWBER bit. 0 No bus error overrun 1 Bus error overrun occurred. The FADR and FDR registers and the other FATR bits are not updated to reflect this new bus error.
30–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 BEMN	Bus error master number Crossbar switch bus master number of the captured cache write buffer bus error. For this device, this value is always 0x1.
7 BEWT	Bus error write Indicates the type of system bus access when the error was detected. Since this logic is monitoring data transfers from the cache write buffer, this bit is always a logical one, signaling a write operation. 0 Read access 1 Write access
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 BESZ	Bus error size Indicates the size of the cache write buffer access when the error was detected. 00 8-bit access 01 16-bit access

Table continues on the next page...

MCM_FATR field descriptions (continued)

Field	Description
	10 32-bit access 11 Reserved
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 BEMD	Bus error privilege level Indicates the privilege level of the cache write buffer access when the error was detected. 0 User mode 1 Supervisor/privileged mode
0 BEDA	Bus error access type Indicates the type of cache write buffer access when the error was detected. This attribute is always a logical one signaling a data reference. 0 Instruction 1 Data

4.2.12.6 Fault data register (MCM_FDR)

When a properly-enabled cache write buffer error interrupt event is detected, the faulting data is captured in the MCM_FDR register.

The bits in this register are set by hardware and signaled by the assertion of MCM_ISCR[CWBER]. For byte and halfword writes, only the accessed byte lanes contain valid data; the contents of the other bytes are undefined. Attempted writes to this location are terminated with an error.

Address: E008_0000h base + 28h offset = E008_0028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

MCM_FDR field descriptions

Field	Description
DATA	Fault data

4.3 Messaging Unit (MU)

4.3.1 Overview

The Messaging Unit module enables two processors within the SoC to communicate and coordinate by passing messages (e.g. data, status and control) through the MU interface. The MU also provides the ability for one processor to signal the other processor using interrupts.

Because the MU manages the messaging between processors, the MU uses different clocks (from each side of the different peripheral buses). Therefore, the MU must synchronize the accesses from one side to the other. The MU accomplishes synchronization using two sets of matching registers (Processor A-facing, Processor B-facing).

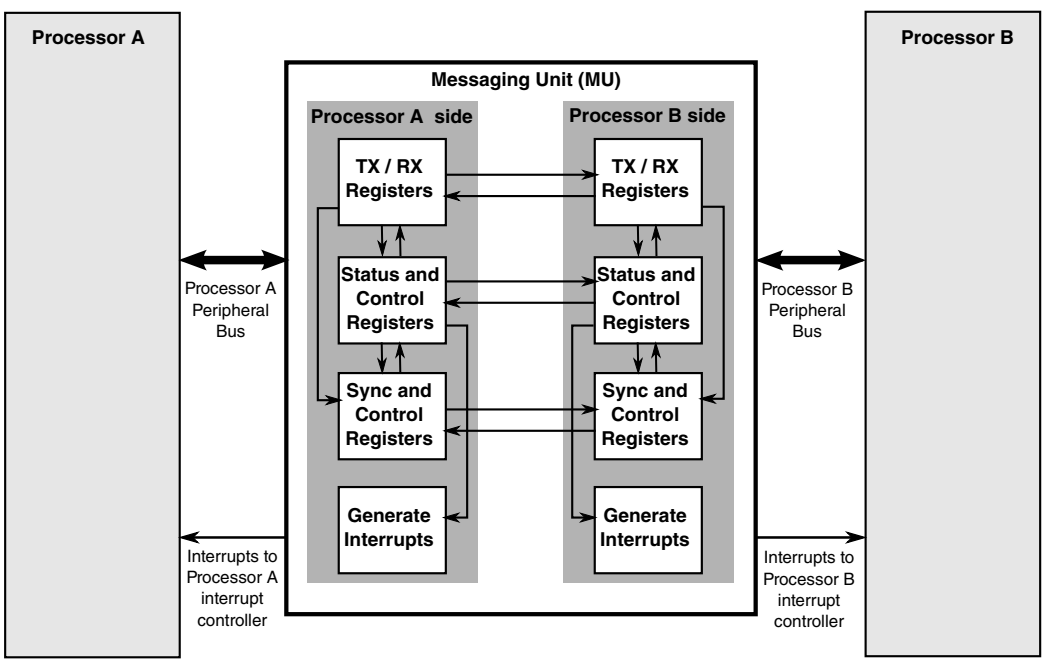


Figure 4-6. MU Block Diagram

4.3.1.1 Features

The MU includes the following features:

- Messaging control by interrupts or by polling

- The Processor B can take the Processor A out of low-power modes by asserting one of the above twelve interrupts to the Processor A and vice versa
- Symmetrical processor interfaces with each side supporting the following:
  - Four general-purpose interrupt requests reflected to the other side
  - Three general-purpose flags reflected to the other side
  - Four receive registers with maskable interrupt
  - Four transmit registers with maskable interrupt

4.3.1.2 Modes of Operation

The MU supports the modes described in the indicated sections:

- Operating Modes
- Low Power Modes

4.3.2 External Signals

There are no Messaging Unit signals directly available at the chip boundary of the SoC.

4.3.3 Functional Description

Table 4-11. Major Features of the MU

Major Feature	Description
Interprocessor Interrupts	• The MU has 12 interrupt sources on each side (Processor A-side, Processor B-side) that are used for signaling the other processor. The interrupts can be used for notification of RX/TX events and general-purpose signaling between the processors.
MU Reset	• The Processor A can issue a reset to the entire MU, using a control bit (MUR) in the Processor A Control Register (ACR). • The MUR bit is a self-clearing bit.

Processor B Boot Configuration	• The Boot Source for Processor B can be configured with the BBOOT bits in the ACR register.
	• Boot Source Options are:
	  • DMEM Base Address
	  • IMEM Base Address
	  • Address 0x00
	• The value at reset is loaded from Flash IFR

Processor B Reset Hold	• Processor B can be held in reset following any reset event. This is done by setting the BRSTH bit in the ACR register. • Processor B will be released from reset when this bit is cleared. • The value at reset is loaded from Flash IFR.
Processor A/B Clock Enable	• The Processor A/B platform clock can be enabled to continue running when Processor A/B enters Stop Mode, until Processor B/A also enters Stop Mode. This allows Processor B/A to continue accessing peripherals on Processor A/B's AIPS bus even when it has entered a Stop Mode.

Table continues on the next page...

Table 4-11. Major Features of the MU (continued)

Major Feature	Description
Status and Control Communications between Cores	• The MU provides a way for the two cores to communicate using the status and control registers present on both the Processor B and Processor A sides of the MU. • The status register of one MU side reflects the status of the other MU side. • The control register is used for control operations, such as enabling an interrupt and sending an interrupt to the other processor.
Synchronized Message Transfers between Cores	• The transfer of data messages between cores uses transmit empty and receive full flags provided on both sides of the MU. • The update of these transmit and receive flags is accomplished using a synchronization mechanism. There is inherent latency between updating the flag on one side and reflecting its status on other side. For more about latency, see Event Update Timing
Accessing Shared Memory Directly and Avoiding Collisions	• For sending data or messages from one MU-side to the other MU-side, the MU provides 4 transmit registers and 4 receive registers on each side of the MU. • The Processor A or Processor B can access shared memory resources of the SoC directly. However, to avoid simultaneous access to shared memory by both cores, the MU provides a method (to prevent simultaneous access) using interrupts and transmit-receive registers for both processors.
Support for Different Clocks in the Two Cores	• The heart of the MU module is the event control mechanism, which synchronizes the access of one MU-side to the other MU-side, because these two MU-sides can operate using different clocks. • Formulated event update latency.
Memory-Mapped Registers	• The MU is connected as a peripheral under the Peripheral bus on both sides—on the Processor A-side, the Processor A Peripheral Bus, and on the Processor B-side, the Processor B Peripheral Bus.

4.3.3.1 Processor A Side Memory-Mapping

The messaging, control, and status registers of the Processor A-side for the MU are mapped to the Processor A memory as a regular peripheral. The Peripheral bus data bus is 32 bits wide inside the MU module.

4.3.3.2 Processor B Side Memory-Mapping

The messaging, control, and status registers of the Processor B-side for the MU are mapped to the Processor B memory as a regular peripheral. The Peripheral bus data bus is 32 bits wide inside the MU module.

4.3.3.3 MU Messaging

The MU provides 32-bit status and control registers to the Processor B and Processor A sides for control operations (such as interrupts and reset), and for status checking of the other MU-side.

For messaging, the MU has four, 32-bit write-only transmit registers and four, 32-bit read-only receive registers on the Processor B and Processor A-sides. These registers are used for sending messages to each other. These messages can be also be controlled using the 3 general purpose flags provided in the control and status registers of either MU-side.

4.3.3.3.1 Programmer Model

The messaging logic is used in conjunction with external memory. You have various messaging methods, which you can use to implement a messaging protocol. Some of these messages could mean “I have just written a message of N words, starting at offset X in the memory,” or “I have just finished reading the previous data block that was sent.” Having the messaging logic independent from the memory array does not restrict you to a predefined hardware protocol. On the other hand, the software needed to manage the messaging is short and straightforward.

Most of the messaging mechanisms are symmetric; they are duplicated and are available on both the Processor B-side and the Processor A-side. The messaging mechanisms are:

- Four, 32-bit write-only transmit registers, which are each reflected in four, read-only receive registers in the other processor’s side. You can use these registers to transfer 32-bit word messages or frame information of messages written to the shared memory (number of words, initial address, and message type code).
- A write to a transmit register on the transmitter side clears a “transmitter empty” bit in the Status Register on the transmitter side, and sets a “receiver full” bit in the Status Register on the receiver side. The setting of the bit at the receiver side can optionally trigger an interrupt at the receiver side (maskable receive interrupt).
- A read of one of the receive registers at the receiver side clears the “receiver full” bit in the Status Register at the receiver side, and sets the “transmitter empty” bit in the Status Register on the transmitter side. The setting of the “transmitter empty” bit can optionally trigger an interrupt at the transmitter side (maskable transmit interrupt).
- Four general purpose flags are reflected in the Status Register on the receiver side
- A read/write access to any reserved location and a write to a read-only register on the Processor A-side of the MU will generate a module transfer error acknowledge to the Processor A.
- A read/write access to any reserved location and write to a read-only register on the Processor B-side of the MU will generate a module transfer error acknowledge to the Processor B.

4.3.3.3.2 Messaging Examples

The following are messaging examples:

- **Passing short messages:** Transmit register(s) can be used to pass short messages from one to four words in length. For example, when a four-word message is desired, only one of the registers needs to have its corresponding interrupt enable bit set at the receiver side; the message's first three words are written to the registers whose interrupt is masked, and the fourth word is written to the other register (which triggers an interrupt at the receiver side).
- **Passing frame information:** Transmit registers can be used to pass frame information for long messages written to the shared system (SDRAM and SyncFLASH). Such frame information normally includes a start address, number of words, and perhaps a message type code.
- **Passing event notices and requests:** Events and requests that do not include data words can be signaled from the Processor B to the Processor A using the general interrupts, such as acknowledging that a long message was read from the shared system memory.
- **Passing fixed length data:** Formatted data with a fixed length can be written in predetermined locations in the shared memory. A processor can use a general interrupt (Processor A or Processor B) to signal the other processor that the data is ready.
- **Passing announcements:** The three flags can be used by a processor to announce its current program state or other billboard messages to the other processor.

Figure 4-7 shows the MU registers schematic.

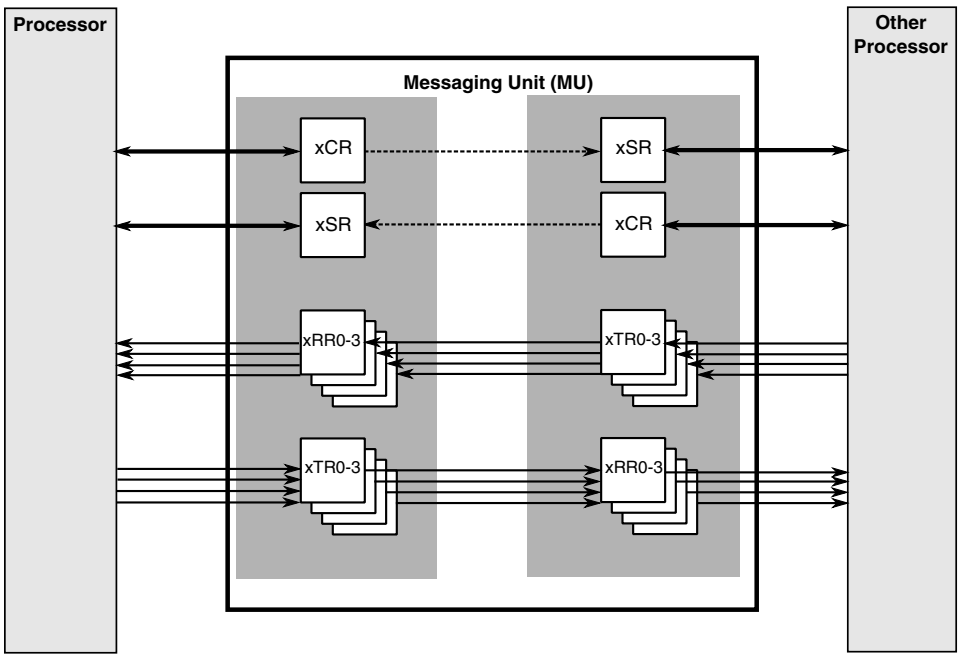


Figure 4-7. MU Registers

4.3.3.4 Operating Modes

This section describes all functional operation modes of the module.

4.3.3.5 Low Power Modes

This section describes the low power operating modes of the MU module.

4.3.3.5.1 Low Power Clocks and Synchronization

The Processor B and the Processor A clocks operate at different frequencies and from different sources. The MU design does not assume any frequency relationship between the Processor A and the Processor B clocks. Be aware, however, that the frequency relationship affects the MU's throughput performance.

- The data buffers and control logic of each MU-side operate with its corresponding clock.

4.3.3.5.2 Processor Low Power Modes

The Processors have four power modes:

- Run
- WAIT

- STOP
- DSM

The Processor can be awakened from a low-power mode by any enabled Processor side MU interrupt, as reflected in the xSR “status” register (RF0–3, TE0–3, GIP0–3 bits are set) and enabled in the xCR control register. Using these bits, the Processor can actively control when to wake the other Processor.

While the Processor is in STOP mode (such that the xSR register bits cannot be updated with events), special logic drives the enabled Processor interrupts directly from the other Processor-side (instead of from the xSR register).

While the Processor is in STOP mode, the asynchronous Processor interrupt will be asserted to wake the Processor:

- If any transmit data register of the other Processor-side is full, because of a write to it (transmit data register); that is, its “empty” bit in the xSR register is cleared while its corresponding receive interrupt is enabled on the Processor-side.
- If any receive data register of the other Processor-side is empty, because of a read on the other Processor -side; that is, its “full” bit in the xSR register is cleared while its corresponding transmit interrupt is enabled on the Processor-side.
- If any general purpose interrupt is set in the xCR register while the corresponding interrupt is enabled on the Processor-side.
- If the other Processor issues a non-maskable interrupt to the Processor.

The logic enables the other Processor to operate independently while the Processor is in any power mode (including STOP). However, the Processor power mode change protocol should be handled with care regarding:

- The interrupts that are enabled on the Processor-side
- The events that could be triggered by the other Processor-side
- The compatibility with the other Processor protocol of entering STOP mode

If the Processor is in STOP mode and an event on the other Processor is triggered, the EP bit (in the xSR register) will remain high until the Processor wakes up.

Before entering STOP mode, the Processor programmer should verify that the EP bit (in the xSR register) is cleared. This check is needed to ensure that all pending updates from the Processor, including the power mode change when STOP or WAIT is executed, will be updated in the xSR register.

- If the other Processor is in STOP mode or DSM mode, the EP bit (in the xSR register) may be stuck high; in this case, the Processor need not check the EP bit before entering STOP mode.

4.3.3.6 Event Update Timing

Each processor's MU messaging side (Processor B or Processor A) has a hardware mechanism to send "event update requests" to the other processor's side. An "event" is considered when any information change should be reflected at the Status Register of the receiving processor. The event update latency is the delay between the event being ready at one processor and the resulting update at the Status Register of the other processor.

- The minimum event latency is "1 clock of the sending side" + "2 1/2 clocks of the receiving side". The minimum case is if there is no event pending when the new event occurs.
- The maximum event latency is "6 clocks of the sending side" + "6 1/2 clocks of the receiving side." The maximum case is if the event occurred just after a previous event was sent to the other side. The event update latency will vary between the above-mentioned minimum and maximum latencies, depending on the time at which the subsequent event is triggered.

4.3.3.7 Interrupts

The MU controls the Processor B interrupt requests to the Processor A, and the Processor A interrupt requests to the Processor B. This section describes all the interrupts that the module generates.

4.3.3.7.1 Interrupts to the Processors

There are 12 interrupt sources from the MU to the Processors:

- Four receive interrupts (asserted when the Processors receive full bits are set and enabled in the xCR register) for each of the receive registers
- Four transmit interrupts (asserted when the Processor transmit empty bits are set and enabled in the xCR register) for each of the transmit registers
- Four general purpose interrupts (asserted when the GIP bits are set and enabled in the xCR register)

All the interrupts are maskable in the Processor Control Register (xCR). The MU does not assume any internal priority of these interrupts. Multiple interrupts (for example, Receive 0 and Receive 1 interrupts or any of the transmit and general purpose interrupts) can be asserted at one time. The priority of these interrupts should be resolved by the interrupt controller at the chip level.

The General Purpose Interrupt Pending bits (GIP0, GIP1, GIP2, and GIP3) should be cleared by the software (as part of the interrupt service routine) to de-assert the request to the interrupt controller.

4.3.3.7.2 General Purpose Interrupt Clearing Sequence

When a Processor writes to the general interrupt bit (GIR), the write event is synchronized to the other Processor clock to set the general interrupt request pending bit (GIP). When the GIP bit is set, and if the general purpose interrupt is enabled on the transmitting Processor side (GIE bit is set), then the receiving Processor general purpose interrupt is issued to the transmitting Processor. The transmitting Processor clears this interrupt by writing a “1” on the GIP bit. The interrupt is de-asserted as soon as the GIP bit is written. The write event of the GIP bit is synchronized to the other Processor clock. The synchronized signal clears the GIR bit. The software should not write the GIR bit again until the GIR bit is cleared.

4.3.3.8 Interrupt Messaging Protocols

4.3.3.8.1 Messaging Protocols using Interrupts

The example below describes a four-word messaging sequence sent by the Processor to the other Processor.

In this example, the first, second, and third receive interrupts are disabled, and the fourth receive interrupt is enabled. We write registers sequentially for n = 0, 1, 2, 3. For n = 0, 1, 2, the interrupts are disabled, therefore no interrupt will go to the other core (although interrupt conditions occur). For n = 3, the interrupt is enabled, and the last Receive Interrupt request is generated.

1. Write Sequence
   - The Processor writes the message information sequentially to its Transmit Registers 0, 1, 2.
   - When the write to the Transmit Register 3 occurs, the RF3 bit of the xSR is set after synchronization, and it immediately trigger the Receive 3 interrupt to the other Processor.
2. Read Sequence
   - The other Processor receives the Receive 3 interrupt and starts reading the message transferred from the receive registers.
   - After Receive Register 3 is read, the interrupt bit is cleared.

Figure 4-8 shows the programmer’s model of a messaging protocol using transmit and receive registers. Use Table 4-12 and Figure 4-8 to understand the generalized protocol sequence.

Table 4-12. Interrupt Messaging Protocol (Generalized)

Sequence	Action	Description
1	Processor A Data write	A data write to the ATRn register by Processor A is immediately reflected in the Processor B BRRn register.
2	Clear Tx Empty bit and Set Rx Full bit	The data write to the ATRn register • Clears the transmitter empty bit (TEn) in the Processor A Transmit Status Register • Sets the receiver full bit (RFn) in the Processor B Receive Status Register
3	Generate Receive Interrupt request	The setting of the receiver full bit (RFn) in the Receive Status Register generates a Receive Interrupt request to Processor B.
4	Processor B Data read	After receiving the Receive Interrupt request, Processor B performs a data read of the BRRn register.
5	Clear Rx Full bit and Set Tx Empty bit	Reading the data out of the BRRn register • Clears the receiver full bit (RFn) in the Processor B Receive Status Register • Sets the transmitter empty bit (TEn) in the Processor A Transmit Status Register
6	Generate Transmit Interrupt request	The setting of the transmitter empty bit (TEn) in the Transmit Status Register generates a Transmit Interrupt request to Processor A.

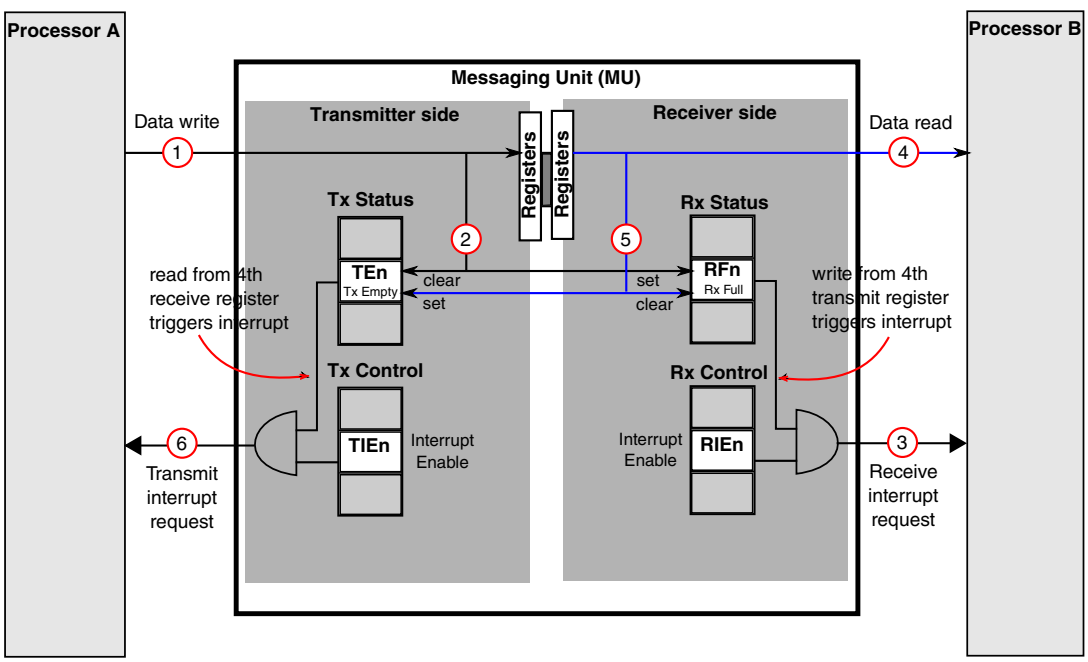


Figure 4-8. Messaging Model Using Transmit and Receive Registers

NOTE

The Transmit registers can be used to pass frame information on long messages written to the shared memory. Such frame

information would typically include an initial address, number of words, and perhaps a message type code.

The messaging hardware can be used by software to implement messaging protocols for a wide array of message types. Full support is given for both interrupt and polling management schemes.

4.3.3.8.2 Messaging Protocols using Event Interrupts

Events and requests that do not include data words can be signaled from the Processor B to the Processor A using the two general interrupts.

Formatted data with a fixed length can be written in predetermined locations in the shared memory. A processor can use a general purpose interrupt to signal the other processor that the data is ready.

The three flags can be used by a processor to announce to the other processor the program state it is currently in, or to announce similar messages.

Table 4-13 and Figure 4-9 describe the event sequence when the Processor triggers an interrupt.

Table 4-13. Interrupt Messaging Protocol (Generalized)

Sequence	Action	Description
1	Processor A sets General Interrupt request bit	Processor A sets its associated General Interrupt request bit (GIRn = 1) in the control register (ACR).
2	General Interrupt Request Pending status bit is set	The General Interrupt Request Pending status bit (GIPn) in the status register (BSR) is set to "1"
3	General Interrupt request to Processor B is generated	Setting the GIPn bit generates the General Interrupt request to Processor B (Interrupt Request Enable bit, GIEn, must be set for Processor B)
4	Processor B reads status register	The Processor B reads the GIPn bit in the BSR register.
5	Processor B services the interrupt	-
6	Processor B sets GIPn bit to clear interrupt	The Processor B writes "1" to the corresponding GIPn bit to clear the interrupt
7	GIRn bit is cleared	Setting the GIPn bit to "1" clears the General Interrupt request bit (GIRn) in the Processor A control register (ACR).

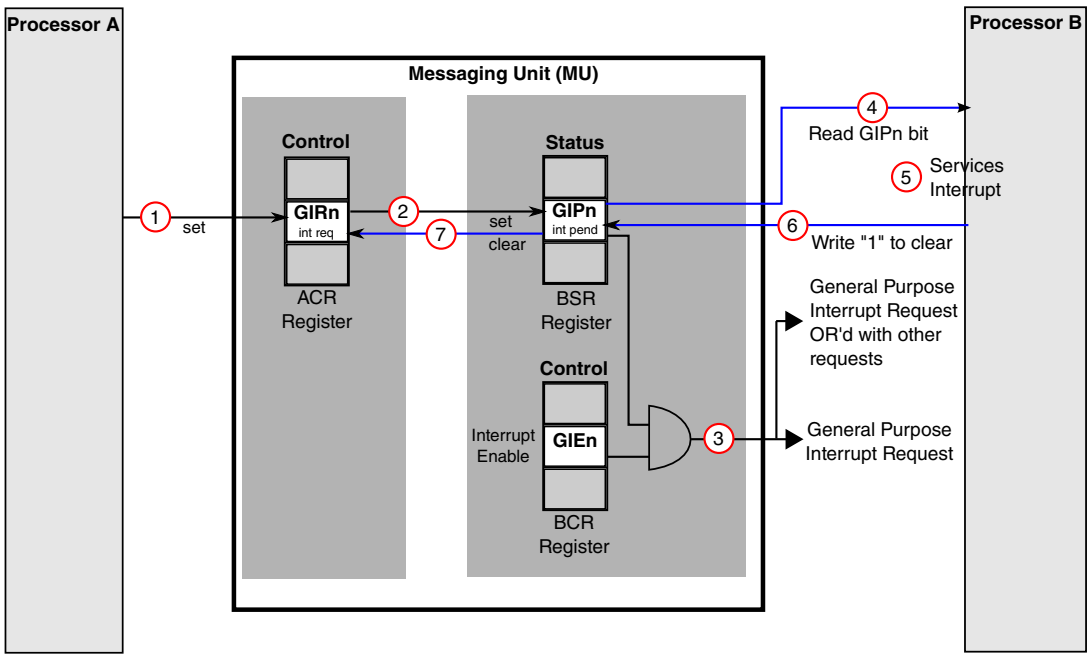


Figure 4-9. Messaging Model Using a General Purpose Interrupt

4.3.3.9 Exclusive Access to Shared Memory

You can use the MU to signal one processor about its current access to the shared memory, so that the data is not overwritten by the other processor during the exclusive memory access period.

The following tables describe the signaling protocol that the Processor A uses to inform the Processor B about its current access (write) to the shared memory, assuming that the set of bits and registers (GIR0 bit, BRR0 register, BTR0 register, GIR0 bit, ARR0 register, ATR0 register) are reserved to support exclusive access to the shared memory protocol.

Table 4-14. How the Processor A Performs an Exclusive Access to Shared Memory

Sequence	Action	Description
1	Processor A sends GIRn request to Processor B using Processor A control register	When the Processor A wants to perform an exclusive access to the shared memory, the Processor A sends an GIR0 request to the Processor B.
2	Processor A sends an exclusive-access request using a transmit data register (ATRn)	The Processor A will send an exclusive-access request (command, location, and length of target access) to Processor B using a selected transmit data register (ATR0).
3	Processor A waits for a dedicated interrupt from Processor B	The Processor A waits for a dedicated interrupt (as an acknowledgement) triggered by the Processor B before proceeding.

Table continues on the next page...

Table 4-14. How the Processor A Performs an Exclusive Access to Shared Memory (continued)

Sequence	Action	Description
4	Processor A accesses shared memory	After receiving a dedicated interrupt from the Processor B, Processor A proceeds.

Table 4-15. How the Processor B Scans for Transaction Information

Sequence	Action	Description
1	Processor B receives an interrupt from a receive data register (BRRn)	-
2	Processor B reads the receive data register (BRRn)	-
3	Processor B scans the receive data register contents	For transaction information (whether Processor A has requested an exclusive-access)

Table 4-16. How the Processor B Accepts Exclusive Access by Processor A

Sequence	Action	Description
1	Processor B triggers a dedicated interrupt	Processor B acknowledges the Processor A request by triggering a dedicated interrupt (ack) to the Processor A.
2	Processor B sends a code message to Processor A	Along with the acknowledge interrupt, the Processor B sends a code message to the Processor A through the selected transmit register (BTRn). The message informs the Processor A that it can exclusively access the shared memory.

Table 4-17. How the Processor B Rejects Exclusive Access by Processor A

Sequence	Action	Description
1	Processor B ignores Processor A request for exclusive access	If the Processor B does not want to give go-ahead permission to the Processor A, Processor B ignores the exclusive access request.

4.3.3.10 Packet Data Transfers

The following example describes the packet transfer sequence between the Processor B and Processor A subsystems:

Table 4-18. Packet Data Transfer Sequence

Action	Sequence	Description
Processor B requests DMA	1	The Processor B sends a DMA request to initiate the packet data transfer

Table continues on the next page...

Table 4-18. Packet Data Transfer Sequence (continued)

Action	Sequence	Description
DMA data transfer	2	DMA acknowledges.
	3	DMA starts transferring data from the specified Processor B location to the specified shared memory
	4	DMA interrupts the Processor B to signal that the packet transfer has finished.
Processor B informs Processor A that data is in shared memory	5	Using an MU Processor B-side transmit register, the Processor B sends a packet information message to the Processor A to inform the Processor A of the arrival of new packet data that is stored in shared memory . The message contains the command, location, and length of packet data information.
Processor A receives interrupt	6	The Processor A receives an interrupt (assuming its corresponding Processor A MU-side receive interrupt is enabled), and the pending processing task becomes active and processes packet data from memory.
Processor A reads data, writes data	7	The Processor A reads or processes packet data from shared memory.
	8	The Processor A writes the result from packet processing to a separate buffer.
Processor A informs Processor B that transfer is finished	9	After the processing of the packet data finishes, the Processor A informs the Processor B (using the MU Processor A-side transmit register, ATRn).
Processor A sends interrupt to Processor B (request for more data)	10	The Processor B receives the next interrupt from the Processor A, in which the Processor A requests more packet data.

4.3.3.11 MU Resets

The MU has two sources of reset, and each reset has a different function from the MU or system perspective.

- One asynchronous system that is connected to both sides of the MU interface.
- One programmable hardware reset (MUR bit) in the ACR register (on the Processor A-side).

Table 4-19. MU programmable resets

Reset	Description
Processor A MU reset	• Processor A MU Reset bit (MUR) of the ACR register • The MUR reset affects the messaging section on both the Processor A and the Processor B sides. The MUR reset causes all control and status registers to return to their default values and all internal states to be cleared. • It is up to the Processor A software to decide whether to use the MUR reset or not. • The instruction immediately following assertion of the MUR bit should not write to MU registers. Such a write may be overwritten by the reset sequence and the register will remain with the reset value. You should wait at least one instruction (after assertion of the MUR bit) before attempting a write to MU registers.

After issuing MUR bit reset events, the Processor A programmer can verify that the reset sequence on the Processor B-side has ended, by checking the RS bit in the ASR register.

NOTE

MUR bit assertion is a delicate operation because it affects the other side's registers asynchronously. MUR bit assertion may cause unpredictable behavior if, for example, the Processor B is concurrently testing an MU register bit (TE bit in Processor B SR register). Before asserting the MUR bit, you should verify that the Processor B is not presently engaged in an MU signalling activity.

4.3.4 Software Restrictions

This section describes certain software restrictions when accessing the MU.

4.3.4.1 General Restrictions

This section lists the restrictions that apply to both the sides (Processor A, Processor B) of the MU.

4.3.4.1.1 Write-After-Write to a Transmit Register

A write to a transmit register signals the receiver side that data is ready for retrieval.

- Writing to the transmit register again without verifying that the data was retrieved is prohibited, because the transmitter side has no way of knowing the exact time that the receiver will attempt to retrieve the data.
- Before attempting to write the transmit register again, the transmitter side should wait for a "Transmitter Empty" interrupt, or should poll the "Transmitter Empty" bit in the Status Register.
- Failure to follow this restriction may result in the wrong data being read on the receiver side of the MU.

4.3.4.1.2 Read-After-Read from a Receive Register

A read of a receive register signals the transmitter side that data can be written to that register. In the same way, the receiver processor should not read a receive register before receiving a "Receiver Full" interrupt or polling the "Receiver Full" bit in the Status Register.

- Reading the receive register again without verifying that the data was written is prohibited, because the receiver side has no way of knowing the exact time that the transmitter will attempt to write the data.
- Before attempting to read the receive register again, the receiver side should wait for a “Receiver Full” interrupt, or should poll the “Receiver Full” bit in the Status Register.
- Failure to follow this restriction may result in the wrong data being written on the transmitter side of the MU.

4.3.4.2 Processor Restrictions

This section lists the restrictions that apply each side of the processor in the MU.

4.3.4.2.1 Before Entering Low Power Mode

Before entering Low Power mode, the Processor should verify that the Processor Event Pending (EP) bit in the Status Register is cleared.

- If the Event Pending bit (EP) is still set to “1”, then the Processor should wait and poll the EP bit until it is cleared, before executing the LPM instruction.
- Note that if the other Processor is in Low Power mode (programmed for clock gating in CCM), the EP bit may be stuck high. In this case, the other Processor clock must be turned ON to get the EP bit cleared before the Processor can enter Low Power mode.
- To discover which power mode the other Processor is in, the Processor can check the PM bits in the xSR register.

4.3.4.2.2 Before Setting a General Interrupt Request Bit (GIR0–3)

Before setting a General Interrupt Request bit (GIR0–3), you must verify that the GIRn bit is cleared, which means that a general interrupt is not pending. Generally, setting the GIRn bit while the bit is set to “1” will be ignored, but in some cases it may issue a second interrupt. This restriction is meant to prevent this indeterministic behavior.

4.3.4.2.3 Reset Bit Restrictions

The reset bit (MUR, HR) restrictions are:

- Before asserting the MUR bit in the ACR register, verify that the Processor B-side is not engaged in some MU activity.
- Do not write to an MU register in the instruction immediately after the assertion of the MUR bit in the ACR register, because the written data can be overridden by the reset value.

4.3.5 MU Processor A-side Memory Map/Register Definition

This section contains the detailed register descriptions for the Processor A-side MU registers.

MU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30AA_0000	Processor A Transmit Register 0 (MU_ATR0)	32	R/W	0000_0000h	4.3.5.1/295
30AA_0004	Processor A Transmit Register 1 (MU_ATR1)	32	R/W	0000_0000h	4.3.5.2/296
30AA_0008	Processor A Transmit Register 2 (MU_ATR2)	32	R/W	0000_0000h	4.3.5.3/297
30AA_000C	Processor A Transmit Register 3 (MU_ATR3)	32	R/W	0000_0000h	4.3.5.4/297
30AA_0010	Processor A Receive Register 0 (MU_ARR0)	32	R	0000_0000h	4.3.5.5/298
30AA_0014	Processor A Receive Register 1 (MU_ARR1)	32	R	0000_0000h	4.3.5.6/299
30AA_0018	Processor A Receive Register 2 (MU_ARR2)	32	R	0000_0000h	4.3.5.7/299
30AA_001C	Processor A Receive Register 3 (MU_ARR3)	32	R	0000_0000h	4.3.5.8/300
30AA_0020	Processor A Status Register (MU_ASR)	32	R/W	00F0_0080h	4.3.5.9/301
30AA_0024	Processor A Control Register (MU_ACR)	32	R/W	0000_0000h	4.3.5.10/ 304

4.3.5.1 Processor A Transmit Register 0 (MU_ ATR0)

Use Processor A Transmit Register 0 (ATR0, 32-bit, write-only) to transmit a message or data to the Processor B.

- You can only write to the ATR0 register when the TE0 bit in ASR register is set to “1”.
- Reading the ATR0 register returns all zeros.

Address: 30AA_0000h base + 0h offset = 30AA_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ATR0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MU_ATR0 field descriptions

Field	Description
ATR0	Processor A Transmit Register 0. (Write-only)

MU_ATR0 field descriptions (continued)

Field	Description
	• Data written to the ATR0 register is reflected on the Processor B-side in the Processor B Receive Register 0 (BRR0). The ATR0 and BRR0 registers are not double-buffered—a write to the ATR0 register overrides the data readable at the BRR0 register. • A write to the transmit register clears a “transmitter empty” bit (TE0) in the Processor A Status Register (ASR) on the transmitter side, and sets a “receiver full” bit (RF0) in the Processor B Status Register (BSR) on the receiver side (optionally triggering an interrupt 0 on the Processor B-side). • Any write to the ATR0 register will update all status information.

4.3.5.2 Processor A Transmit Register 1 (MU_ATR1)

Use Processor A Transmit Register 1 (ATR1, 32-bit, write-only) to transmit a message or data to the Processor B.

- You can only write to the ATR1 register when the TE1 bit in ASR register is set to “1”.
- Reading the ATR1 register returns all zeros.

Address: 30AA_0000h base + 4h offset = 30AA_0004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ATR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MU_ATR1 field descriptions

Field	Description
ATR1	Processor A Transmit Register 1. (Write-only) • Data written to the ATR1 register is reflected on the Processor B-side in the Processor B Receive Register 1 (BRR1). The ATR1 and BRR1 registers are not double-buffered—a write to the ATR1 register overrides the data readable at the BRR1 register. • A write to the transmit register clears a “transmitter empty” bit (TE1) in the Processor A Status Register (ASR) on the transmitter side, and sets a “receiver full” bit (RF1) in the Processor B Status Register (BSR) on the receiver side (optionally triggering an interrupt 1 on the Processor B-side). • Any write to the ATR1 register will update all status information.

4.3.5.3 Processor A Transmit Register 2 (MU_ATR2)

Use Processor A Transmit Register 2 (ATR2, 32-bit, write-only) to transmit a message or data to the Processor B.

- You can only write to the ATR2 register when the TE2 bit in ASR register is set to "1".
- Reading the ATR2 register returns all zeros.

Address: 30AA_0000h base + 8h offset = 30AA_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ATR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MU_ATR2 field descriptions

Field	Description
ATR2	Processor A Transmit Register 2. (Write-only) • Data written to the ATR2 register is reflected on the Processor B-side in the Processor B Receive Register 2 (BRR2). The ATR2 and BRR2 registers are not double-buffered—a write to the ATR2 register overrides the data readable at the BRR2 register. • A write to the transmit register clears a "transmitter empty" bit (TE2) in the Processor A Status Register (ASR) on the transmitter side, and sets a "receiver full" bit (RF2) in the Processor B Status Register (BSR) on the receiver side (optionally triggering an interrupt 2 on the Processor B-side). • Any write to the ATR2 register will update all status information.

4.3.5.4 Processor A Transmit Register 3 (MU_ATR3)

Use Processor A Transmit Register 3 (ATR3, 32-bit, write-only) to transmit a message or data to the Processor B.

- You can only write to the ATR3 register when the TE3 bit in ASR register is set to "1".
- Reading the ATR3 register returns all zeros.

Address: 30AA_0000h base + Ch offset = 30AA_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ATR3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MU_ATR3 field descriptions

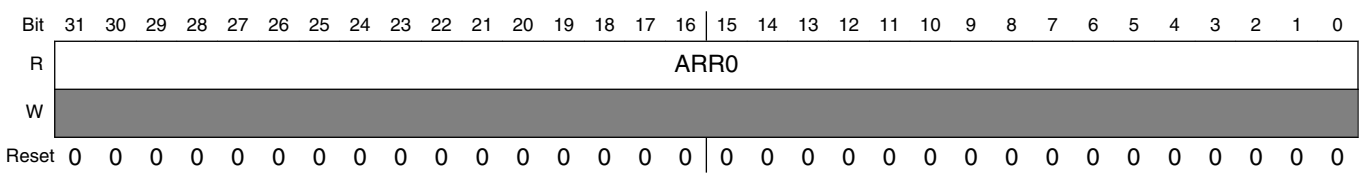
Field	Description
ATR3	Processor A Transmit Register 3. (Write-only) • Data written to the ATR3 register is reflected on the Processor B-side in the Processor B Receive Register 3 (BRR3). The ATR3 and BRR3 registers are not double-buffered—a write to the ATR3 register overrides the data readable at the BRR3 register. • A write to the transmit register clears a “transmitter empty” bit (TE3) in the Processor A Status Register (ASR) on the transmitter side, and sets a “receiver full” bit (RF3) in the Processor B Status Register (BSR) on the receiver side (optionally triggering an interrupt 3 on the Processor B-side). • Any write to the ATR3 register will update all status information.

4.3.5.5 Processor A Receive Register 0 (MU_ARR0)

Use Processor A Receive Register 0 (ARR0, 32-bit, read-only) to receive a message or data from the Processor B.

- Data written to the BTR0 register is immediately reflected in the ARR0 register.
- You can only read the ARR0 register when the RF0 bit in the ASR register is set to “1”.
- Writing to the ARR0 register generates an error response to the Processor A.

Address: 30AA_0000h base + 10h offset = 30AA_0010h



MU_ARR0 field descriptions

Field	Description
ARR0	Processor A Receive Register 0. (Read-only) • Reflects the data written to Processor B Transmit Register 0 (BTR0). • Reading the ARR0 register clears the “receiver full” bit (RF0) in the Processor A Status Register (ASR) on the receiver side, and sets the “transmitter empty” bit (TE0) in the Processor B Status Register on the transmitter side (optionally triggering a transmit interrupt 0 on the Processor B-side). • Any read of the ARR0 register will update all status information.

4.3.5.6 Processor A Receive Register 1 (MU_ARR1)

Use Processor A Receive Register 1 (ARR1, 32-bit, read-only) to receive a message or data from the Processor B.

- Data written to the BTR1 register is immediately reflected in the ARR1 register.
- You can only read the ARR1 register when the RF1 bit in the ASR register is set to "1".
- Writing to the ARR1 register generates an error response to the Processor A.

Address: 30AA_0000h base + 14h offset = 30AA_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ARR1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MU_ARR1 field descriptions

Field	Description
ARR1	Processor A Receive Register 1. (Read-only) • Reflects the data written to Processor B Transmit Register 1 (BTR1). • Reading the ARR1 register clears the "receiver full" bit (RF1) in the Processor A Status Register (ASR) on the receiver side, and sets the "transmitter empty" bit (TE1) in the Processor B Status Register on the transmitter side (optionally triggering a transmit interrupt 1 on the Processor B-side). • Any read of the ARR1 register will update all status information.

4.3.5.7 Processor A Receive Register 2 (MU_ARR2)

Use Processor A Receive Register 2 (ARR2, 32-bit, read-only) to receive a message or data from the Processor B.

- Data written to the BTR2 register is immediately reflected in the ARR2 register.
- You can only read the ARR2 register when the RF2 bit in the ASR register is set to "1".
- Writing to the ARR2 register generates an error response to the Processor A.

Address: 30AA_0000h base + 18h offset = 30AA_0018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ARR2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MU_ARR2 field descriptions

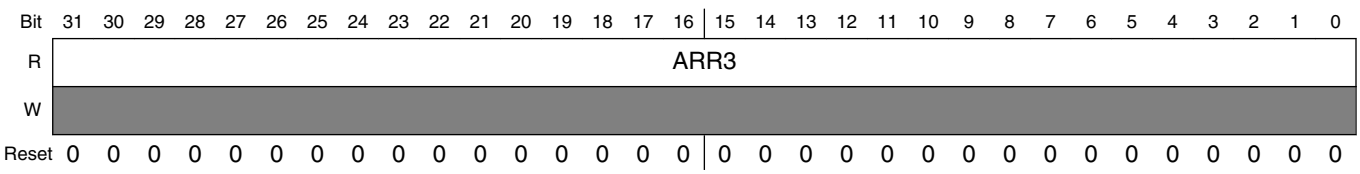
Field	Description
ARR2	Processor A Receive Register 2. (Read-only) • Reflects the data written to Processor B Transmit Register 1 (BTR2). • Reading the ARR2 register clears the “receiver full” bit (RF2) in the Processor A Status Register (ASR) on the receiver side, and sets the “transmitter empty” bit (TE2) in the Processor B Status Register on the transmitter side (optionally triggering a transmit interrupt 2 on the Processor B-side). • Any read of the ARR2 register will update all status information.

4.3.5.8 Processor A Receive Register 3 (MU_ARR3)

Use Processor A Receive Register 3 (ARR3, 32-bit, read-only) to receive a message or data from the Processor B.

- Data written to the BTR3 register is immediately reflected in the ARR3 register.
- You can only read the ARR3 register when the RF3 bit in the ASR register is set to “1”.
- Writing to the ARR3 register generates an error response to the Processor A.

Address: 30AA_0000h base + 1Ch offset = 30AA_001Ch



MU_ARR3 field descriptions

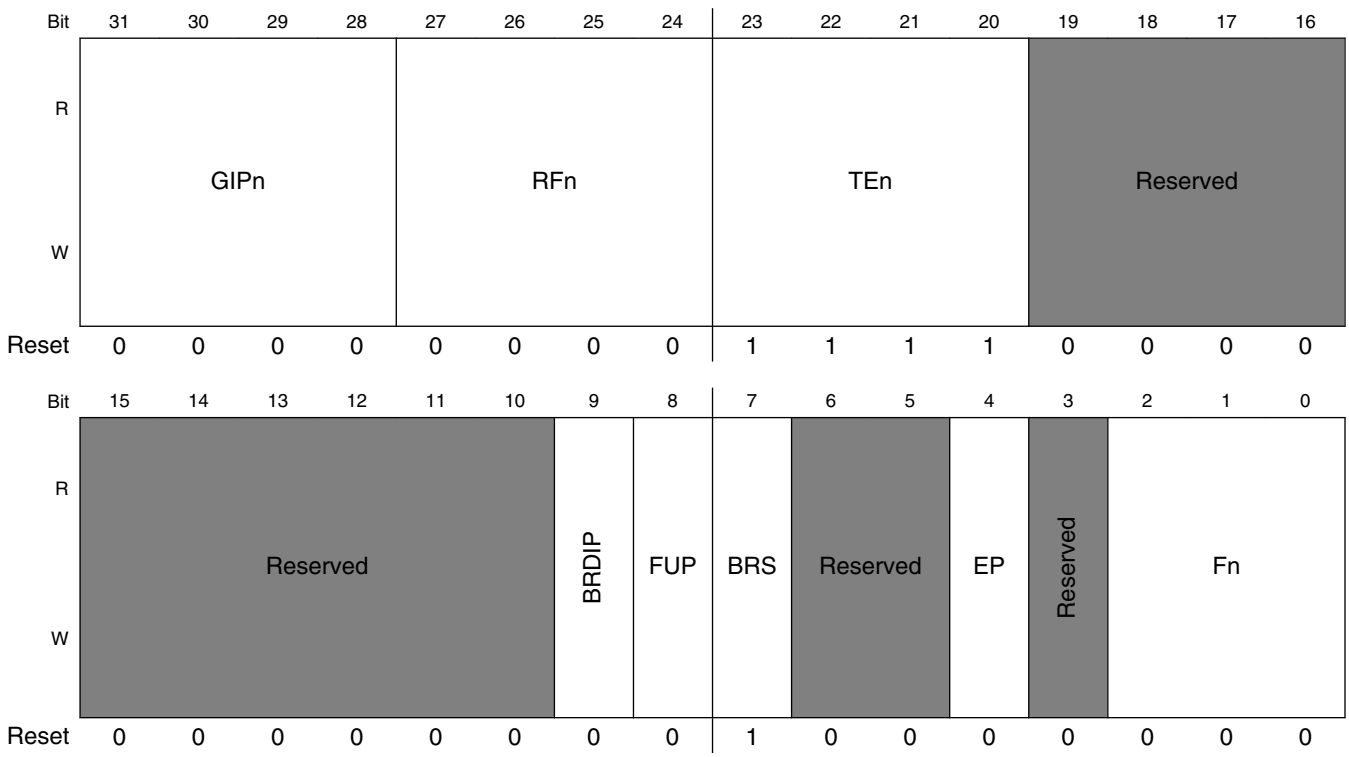
Field	Description
ARR3	Processor A Receive Register 3. (Read-only) • Reflects the data written to Processor B Transmit Register 3 (BTR3). • Reading the ARR3 register clears the “receiver full” bit (RF3) in the Processor A Status Register (ASR) on the receiver side, and sets the “transmitter empty” bit (TE3) in the Processor B Status Register on the transmitter side (optionally triggering a transmit interrupt 3 on the Processor B-side). • Any read of the ARR3 register will update all status information.

4.3.5.9 Processor A Status Register (MU_ASR)

Use the Processor A Status Register (ASR, 32-bit, read-write) to show interrupt status from the Processor B, general purpose flags, and to set dual function control-status bits.

- Some dual-purpose bits are set by the MU logic, and cleared by the Processor A-side programmer
- Other dual-purpose bits are set by the Processor A-side programmer, and cleared by the MU logic.

Address: 30AA_0000h base + 20h offset = 30AA_0020h



MU_ASR field descriptions

Field	Description
31–28 GIPn	For n = {0, 1, 2, 3} Processor A General Interrupt Request n Pending. (Read-Write) • GIPn bit signals the Processor A that the GIRn bit in the BCR register on the Processor B-side was set from “0” to “1”. If the GIEn bit in the ACR register is set to “1”, a General Interrupt n request is issued. • The GIPn bit is cleared by writing it back as “1”. Writing “0”, or writing “1” when the GIPn bit is cleared is ignored. Use this feature in the interrupt routine, where the GIPn bit is cleared in order to de-assert the interrupt request source at the interrupt controller. The proper bit clearing sequence is: clear an Processor A register, set the desired bit in it (Processor A register), and write it to the ASR register, thus clearing the GIPn bit. • GIPn bit is cleared when the MU is reset.

Table continues on the next page...

MU_ASR field descriptions (continued)

Field	Description
	0 Processor A general purpose interrupt n is not pending. (default) 1 Processor A general purpose interrupt n is pending.
27–24 RFn	For n = {0, 1, 2, 3} Processor A Receive Register n Full. (Read-only) • The RFn bit is set to “1” when the BTRn register is written on the Processor B-side. • After the RFn bit is set to “1”, the RFn bit signals the Processor A-side that new data is ready to be read by the Processor A in the ARRn register, and a Receive n interrupt is issued on the Processor A-side (if the RIEn bit in the ACR register has been set to “1”). • RFn bit is cleared when the ARRn register is read, and when the MU is reset. 0 ARRn register is not full (default). 1 ARRn register has received data from BTRn register and is ready to be read by the Processor A.
23–20 TEn	For n = {0, 1, 2, 3} Processor A Transmit Register n Empty. (Read-only) • The TEn bit is set to “1” after the BRRn register is read on the Processor B-side. • After the TEn bit is set to “1”, the TEn bit signals the Processor A-side that the ATRn register is ready to be written on the Processor A-side, and a Transmit n interrupt is issued on the Processor A-side (if the TEn bit in the ACR register is set to “1”). • TEn bit is cleared after the ATRn register is written on the Processor A-side. • TEn bit is set to “1” when the MU is reset. 0 ATRn register is not empty. 1 ATRn register is empty (default).
19–10 -	This field is reserved. Reserved.
9 BRDIP	Processor B Reset De-asserted Interrupt Pending. (Read-Write) • BRDIP bit signals the Processor A-side that the Processor B-side has come out of reset. • BRDIP bit is set to “1” after the MU Processor B-side comes out of reset, after synchronization. The interrupt generated by a Processor B-side reset de-assertion is ORed with the Processor A general purpose interrupt 3. The Processor A general purpose interrupt 3 is issued when the Processor B-side comes out of reset (if the interrupt is enabled). • To clear the BRDIP bit, write “1”, which also clears general purpose interrupt 3. • When Processor A-side of MU comes out of reset BRDIP bit has value “0”(default).Then Processor A sees the status of Processor B-side and if Processor B-side has come out of reset then BRDIP bit goes high.This takes 5-6 clock cycles.And if you read BRDIP bit now you will see it as high although its reset value was "0". 0 The Processor A general purpose interrupt 3, because of a Processor B-side reset de-assertion, is cleared (default). 1 The Processor B-side is out of reset.
8 FUP	Processor A Flags Update Pending. (Read-only) • FUP bit is set to “1” when the Processor A-side sends a Flags Update request to the Processor B-side. • A Flags Update request is generated when the ABF[2:0] bits of the ACR register change. No flag update changes are allowed while the FUP bit is set to “1”. Any write to the ABF[2:0] bits, while the FUP bit is set to “1”, will not generate a Flags Update event, and the ABF[2:0] bits will stay unchanged. • FUP bit is cleared when this Flags Update request is internally acknowledged (that the flag is updated) from the MU Processor B-side, and during MU reset.

Table continues on the next page...

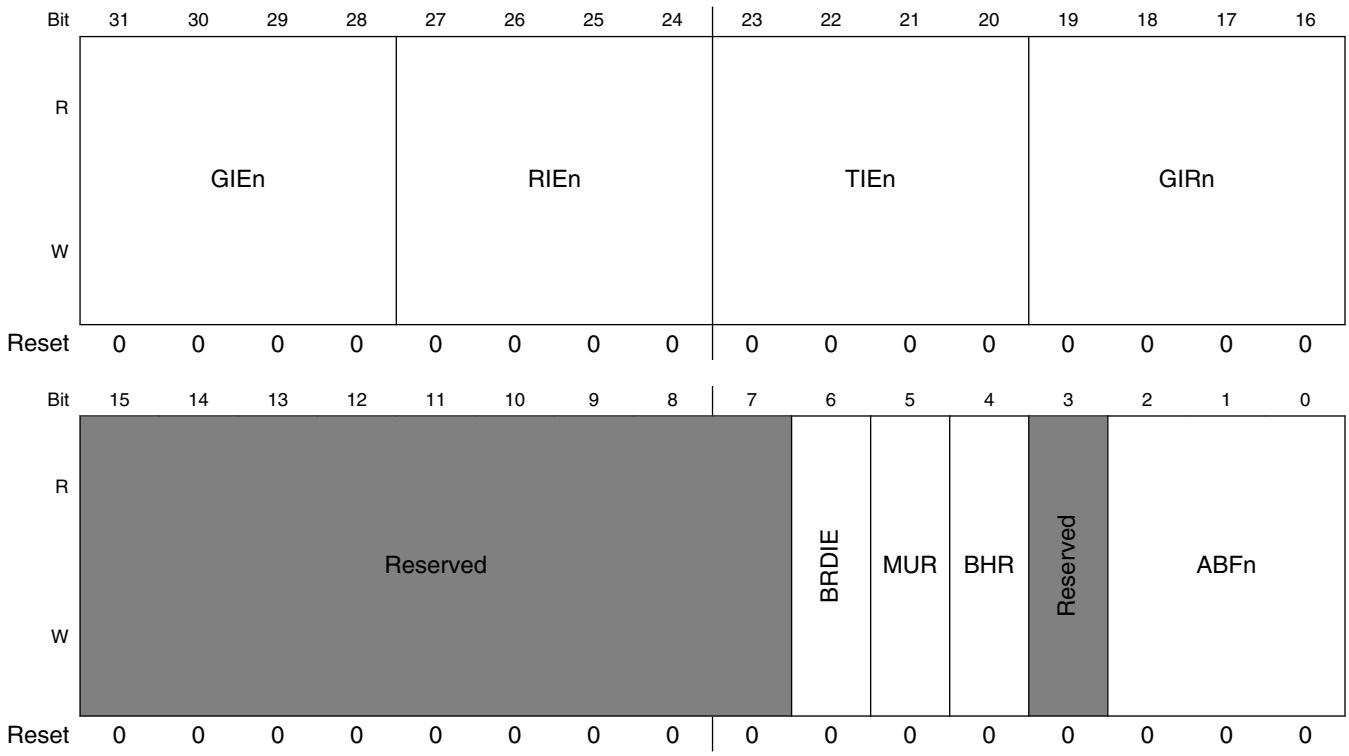
MU_ASR field descriptions (continued)

Field	Description
	0 No flags updated, initiated by the Processor A, in progress (default) 1 Processor A initiated flags update, processing
7 BRS	Processor B-side Reset State. (Read-only) • BRS bit indicates if the Processor B-side of the MU is in a reset state or not. • If the BRS bit is set to “1”, then the Processor B-side of the MU is still in the reset state. • If the BRS bit is cleared, then the Processor B-side of the MU are out of reset. • The BRS bit is set to “1” during: a Processor B system reset, or an MU reset (caused by setting the MUR bit at the ACR register). • The BRS bit is cleared when the reset sequence on the Processor B-side of the MU ends. After issuing any of the reset events mentioned previously, you should verify that the BRS bit is cleared before starting any accesses. • When Processor A side of MU comes out of reset BRS bit has value “1”(default).Then Processor A sees the status of Processor B-side and if Processor B-side has come out of reset then BRS bit goes low.This takes 5-6 clock cycles.And if you read BRS bit now you will see it as low although its reset value was "1" . 0 The Processor B-side of the MU is not in reset. 1 The Processor B-side of the MU is in reset.
6–5 -	This field is reserved. Reserved
4 EP	Processor A-Side Event Pending. (Read-only) • EP bit is set to “1” when the Processor A-side mechanism sends an event update request to the Processor B-side. • EP bit is cleared when the event update acknowledge is received. An “event” is any hardware message that is reflected in the BSR register on the Processor B-side (for example, “transmit register 0 written”). During normal operations, you do not have to deal with the state of the EP bit because the event update mechanism works automatically. • To ensure events have been posted to Processor B before entering STOP mode, you should verify that the EP bit is cleared. If EP bit is set to “1”, you should wait and continue to poll it (EP bit) before entering STOP mode. • Reading the ASR register (to check the EP bit) should be the last access to the MU that should be performed before entering STOPor WAIT modes; otherwise, the EP bit may be set by subsequent additional actions. • The EP bit is cleared when the MU resets. 0 The Processor A-side event is not pending (default). 1 The Processor A-side event is pending.
3 -	This field is reserved. Reserved.
Fn	For n = {0, 1, 2} Processor A-Side Flag n. (Read-only) • Fn bit is the Processor A-side flag that reflects the values written to the BAFn bit in the Processor B control register. • Every time that the BAFn bit is written, the BAFn bit write event updates the Fn bit after the event update latency, which is measured in terms of the number of clocks of the Processor B and the Processor A. 0 BAFn bit in BCR register is written 0 (default). 1 BAFn bit in BCR register is written 1.

4.3.5.10 Processor A Control Register (MU_ACR)

Use the Processor A Control Register (ACR, 32-bit, read-write) to enable the MU interrupts on the Processor A-side, and trigger events and interrupts on the Processor B-side (general purpose interrupt, flag update).

Address: 30AA_0000h base + 24h offset = 30AA_0024h



MU_ACR field descriptions

Field	Description
31–28 GIEn	For n = {0, 1, 2, 3} Processor A General Purpose Interrupt Enable n. (Read-Write) • GIEn bit enables Processor A General Interrupt n. • If GIEn bit is set to “1” (enabled), then a General Interrupt n request is issued when the GIPn bit in the ASR register is set to “1”. • If GIEn is cleared (disabled), then the value of the GIPn bit is ignored and no General Interrupt n request will be issued. • GIEn bit is cleared when the MU resets. 0 Disables Processor A General Interrupt n. (default) 1 Enables Processor A General Interrupt n.
27–24 RIEn	For n = {0, 1, 2, 3} Processor A Receive Interrupt Enable n. (Read-Write) • RIEn bit enables Processor A Receive Interrupt n.

Table continues on the next page...

MU_ACR field descriptions (continued)

Field	Description
	• If RIEn bit is set to “1” (enabled), then an Processor A Receive Interrupt n request is issued when the RFn bit in the ASR register is set to “1”. • If RIEn bit is cleared (disabled), then the value of the RFn bit is ignored and no Processor A Receive Interrupt n request will be issued. • RIEn bit is cleared when the MU resets. 0 Disables Processor A Receive Interrupt n. (default) 1 Enables Processor A Receive Interrupt n.
23–20 TIEn	For n = {0, 1, 2, 3} Processor A Transmit Interrupt Enable n. (Read-Write) • TIEn bit enables Processor A Transmit Interrupt n. • If TIEn bit is set to “1” (enabled), then an Processor A Transmit Interrupt n request is issued when the TEn bit in the ASR register is set to “1”. • If TIEn bit is cleared (disabled), then the value of the TEn bit is ignored and no Processor A Transmit Interrupt n request will be issued. • TIEn bit is cleared when the MU resets. 0 Disables Processor A Transmit Interrupt n. (default) 1 Enables Processor A Transmit Interrupt n.
19–16 GIRn	For n = {0, 1, 2, 3} Processor A General Purpose Interrupt Request n. (Read-Write) • Writing “1” to the GIRn bit sets the GIPn bit in the BSR register on the Processor B-side. If the GIEn bit in the BCR register is set to “1” on the Processor B-side, a General Purpose Interrupt n request is triggered. • The GIRn bit is cleared if the GIPn bit (in the BSR register on the Processor B-side) is cleared by writing it (GIPn bit) as “1”, thereby signalling the Processor A that the interrupt was accepted (cleared by the software). The GIPn bit cannot be written as “0” on the Processor A-side. • To ensure proper operations, you must verify that the GIRn bit is cleared (meaning that there is no pending interrupt) before setting it (GIRn bit). • GIRn bit is cleared when the MU resets. 0 Processor A General Interrupt n is not requested to the Processor B (default). 1 Processor A General Interrupt n is requested to the Processor B.
15–7 -	This field is reserved. Reserved.
6 BRDIE	Processor B Reset De-assertion Interrupt Enable. (Read-Write) • BRDIE bit enables Processor A General Interrupt 3. • If BRDIE bit is set to “1”, then General Interrupt 3 request is issued to the Processor A when the BRDIP bit in the ASR register is set to “1”. • If BRDIE is cleared, then the value of the BRDIP bit is ignored and no General Interrupt 3 request will be issued. • The BRDIE bit is cleared when the MU resets. 0 Disables the Processor A General Purpose Interrupt 3 request due to the Processor B reset de-assertion to the Processor A. Processor B reset deassertion causes Processor B and MU-Processor B side to come out of reset thus setting BRDIP bit to “1”. 1 Enables Processor A General Purpose Interrupt 3 request due to the Processor B reset de-assertion to the Processor A.
5 MUR	Processor A MU Reset.

Table continues on the next page...

MU_ACR field descriptions (continued)

Field	Description
	• Setting MUR bit to “1” resets both the Processor B and the Processor A sides of the MU module, forcing all control and status registers to return to their default values (except the BHR bit in the ACR register and BHRM bit in BCR register), and all internal states to be cleared. • Before setting the MUR bit to “1”, it is advisable to interrupt the Processor B, because setting the MUR bit may affect the ongoing Processor B program. • After setting the MUR bit, you should monitor the value of the BRS bit in the ASR register to know when the reset sequence on the Processor B-side has ended. • MUR bit can only be written as “1”. • MUR bit is always read as “0”. • MUR bit is cleared during the MU reset sequence. 0 N/A. Self clearing bit (default). 1 Asserts the Processor A MU reset.
4 BHR	Processor B Hardware Reset. (Read-Write) • BHR bit asserts and de-asserts the hardware reset of the Processor B. • Set BHR bit to “1” to start a hardware reset of the Processor B. • Clear the BHR bit to de-assert the Processor B hardware reset input. • Assert the BHR bit for a minimum of 3 clock cycles of network clock (sampling clock in SRC) clock. The BRS bit in MU_ASR register (b[7]) indicates the state of the Processor B. As soon as the Processor B goes into Reset (BRS bit is set to “1”), the BHR bit can be de-asserted. • Strobe-setting the BHR bit will not cause an internal MU reset but will be routed outside MU to Processor B domain reset logic • After clearing the BHR bit, monitor the value of the BRS bit at the ASR to know when the Processor B reset sequence has ended. • The BHR reset issued by the Processor A to the Processor B is maskable by the Processor B (according to the settings of the BHRM bit in the BCR register). If the BHRM bit (in the BCR register) is set to “1”, then the BHR reset is masked; if the BHRM bit (in the BCR register) is cleared (default), then the BHR reset is enabled. • The BHR bit does not return to the reset value during the software (MUR) reset. 0 De-assert Hardware reset to the Processor B. (default) 1 Assert Hardware reset to the Processor B.
3 -	This field is reserved. Reserved
ABFn	For n = {0, 1, 2} Processor A to Processor B Flag n. (Read-Write) • ABFn bit is a read-write flag that is reflected in Fn bit in the BSR register on the Processor B-side. • ABFn bit is cleared when the MU resets. 0 N/A. Self clearing bit (default). 1 Asserts the Processor A MU reset.

4.3.6 MU Processor B-side Memory Map/Register Definition

This section contains the detailed register descriptions for the Processor B-side MU registers.

MU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30AB_0000	Processor B Transmit Register 0 (MU_BTR0)	32	R/W	0000_0000h	4.3.6.1/307
30AB_0004	Processor B Transmit Register 1 (MU_BTR1)	32	R/W	0000_0000h	4.3.6.2/308
30AB_0008	Processor B Transmit Register 2 (MU_BTR2)	32	R/W	0000_0000h	4.3.6.3/308
30AB_000C	Processor B Transmit Register 3 (MU_BTR3)	32	R/W	0000_0000h	4.3.6.4/309
30AB_0010	Processor B Receive Register 0 (MU_BRR0)	32	R	0000_0000h	4.3.6.5/310
30AB_0014	Processor B Receive Register 1 (MU_BRR1)	32	R	0000_0000h	4.3.6.6/310
30AB_0018	Processor B Receive Register 2 (MU_BRR2)	32	R	0000_0000h	4.3.6.7/311
30AB_001C	Processor B Receive Register 3 (MU_BRR3)	32	R	0000_0000h	4.3.6.8/312
30AB_0020	Processor B Status Register (MU_BSR)	32	R/W	00F0_0080h	4.3.6.9/313
30AB_0024	Processor B Control Register (MU_BCR)	32	R/W	0000_0000h	4.3.6.10/ 316

4.3.6.1 Processor B Transmit Register 0 (MU_BTR0)

Use Processor B Transmit Register 0 (BTR0, 32-bit, write-only) to transmit a message or data to the Processor A.

- You can only write to the BTR0 register when the TE0 bit in BSR register is set to “1”.
- Reading the BTR0 register returns all zeros.

Address: 30AB_0000h base + 0h offset = 30AB_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BTR0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MU_BTR0 field descriptions

Field	Description
BTR0	Processor B Transmit Register 0. (Write-only) • Data written to the BTR0 register is reflected on the Processor A-side in the Processor A Receive Register 0 (ARR0). The BTR0 and ARR0 registers are not double-buffered—a write to the BTR0 register overrides the data readable at the ARR0 register. • A write to the transmit register clears a “transmitter empty” bit (TE0) in the Processor B Status Register (BSR) on the transmitter side, and sets a “receiver full” bit (RF0) in the Processor A Status Register (ASR) on the receiver side (optionally triggering an interrupt 0 on the Processor A-side). • Any write to the BTR0 register will update all status information.

4.3.6.2 Processor B Transmit Register 1 (MU_BTR1)

Use Processor B Transmit Register 1 (BTR1, 32-bit, write-only) to transmit a message or data to the Processor A.

- You can only write to the BTR1 register when the TE1 bit in BSR register is set to "1".
- Reading the BTR1 register returns all zeros.

Address: 30AB_0000h base + 4h offset = 30AB_0004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BTR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MU_BTR1 field descriptions

Field	Description
BTR1	Processor B Transmit Register 1. (Write-only) • Data written to the BTR1 register is reflected on the Processor A-side in the Processor A Receive Register 1 (ARR1). The BTR1 and ARR1 registers are not double-buffered—a write to the BTR1 register overrides the data readable at the ARR1 register. • A write to the transmit register clears a "transmitter empty" bit (TE1) in the Processor B Status Register (BSR) on the transmitter side, and sets a "receiver full" bit (RF1) in the Processor A Status Register (ASR) on the receiver side (optionally triggering an interrupt 1 on the Processor A-side). • Any write to the BTR1 register will update all status information.

4.3.6.3 Processor B Transmit Register 2 (MU_BTR2)

Use Processor B Transmit Register 2 (BTR2, 32-bit, write-only) to transmit a message or data to the Processor A.

- You can only write to the BTR2 register when the TE2 bit in BSR register is set to "1".
- Reading the BTR2 register returns all zeros.

Address: 30AB_0000h base + 8h offset = 30AB_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BTR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MU_BTR2 field descriptions

Field	Description
BTR2	Processor B Transmit Register 2. (Write-only) • Data written to the BTR2 register is reflected on the Processor A-side in the Processor A Receive Register 2 (ARR2). The BTR2 and ARR2 registers are not double-buffered—a write to the BTR2 register overrides the data readable at the ARR2 register. • A write to the transmit register clears a “transmitter empty” bit (TE2) in the Processor B Status Register (BSR) on the transmitter side, and sets a “receiver full” bit (RF2) in the Processor A Status Register (ASR) on the receiver side (optionally triggering an interrupt 2 on the Processor A-side). • Any write to the BTR2 register will update all status information.

4.3.6.4 Processor B Transmit Register 3 (MU_BTR3)

Use Processor B Transmit Register 3 (BTR3, 32-bit, write-only) to transmit a message or data to the Processor A.

- You can only write to the BTR3 register when the TE3 bit in BSR register is set to “1”.
- Reading the BTR3 register returns all zeros.

Address: 30AB_0000h base + Ch offset = 30AB_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BTR3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MU_BTR3 field descriptions

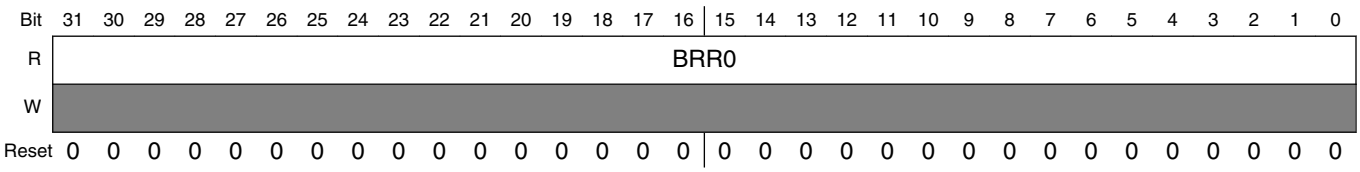
Field	Description
BTR3	Processor B Transmit Register 3. (Write-only) • Data written to the BTR3 register is reflected on the Processor A-side in the Processor A Receive Register 3 (ARR3). The BTR3 and ARR3 registers are not double-buffered—a write to the BTR3 register overrides the data readable at the ARR3 register. • A write to the transmit register clears a “transmitter empty” bit (TE3) in the Processor B Status Register (BSR) on the transmitter side, and sets a “receiver full” bit (RF3) in the Processor A Status Register (ASR) on the receiver side (optionally triggering an interrupt 3 on the Processor A-side). • Any write to the BTR3 register will update all status information.

4.3.6.5 Processor B Receive Register 0 (MU_BRR0)

Use Processor B Receive Register 0 (BRR0, 32-bit, read-only) to receive a message or data from the Processor A.

- Data written to the ATR0 register is immediately reflected in the BRR0 register.
- You can only read the BRR0 register when the RF0 bit in the BSR register is set to "1".
- Writing to the BRR0 register generates an error response to the Processor B.

Address: 30AB_0000h base + 10h offset = 30AB_0010h



MU_BRR0 field descriptions

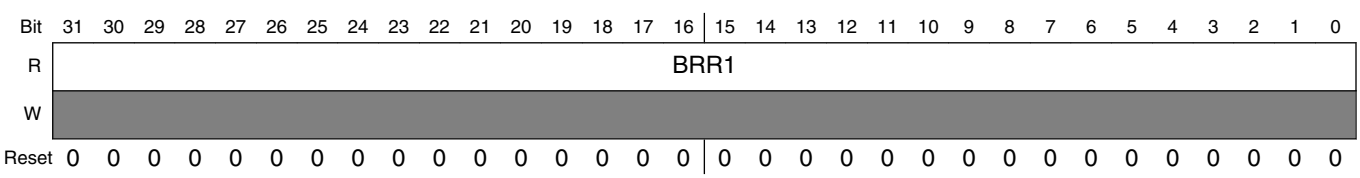
Field	Description
BRR0	Processor B Receive Register 0. (Read-only) • Reflects the data written to Processor A Transmit Register 0 (ATR0). • Reading the BRR0 register clears the "receiver full" bit (RF0) in the Processor B Status Register (BSR) on the receiver side, and sets the "transmitter empty" bit (TE0) in the Processor A Status Register on the transmitter side (optionally triggering a transmit interrupt 0 on the Processor A-side). • Any read of the BRR0 register will update all status information.

4.3.6.6 Processor B Receive Register 1 (MU_BRR1)

Use Processor B Receive Register 1 (BRR1, 32-bit, read-only) to receive a message or data from the Processor A.

- Data written to the ATR1 register is immediately reflected in the BRR1 register.
- You can only read the BRR1 register when the RF1 bit in the BSR register is set to "1".
- Writing to the BRR1 register generates an error response to the Processor B.

Address: 30AB_0000h base + 14h offset = 30AB_0014h



MU_BRR1 field descriptions

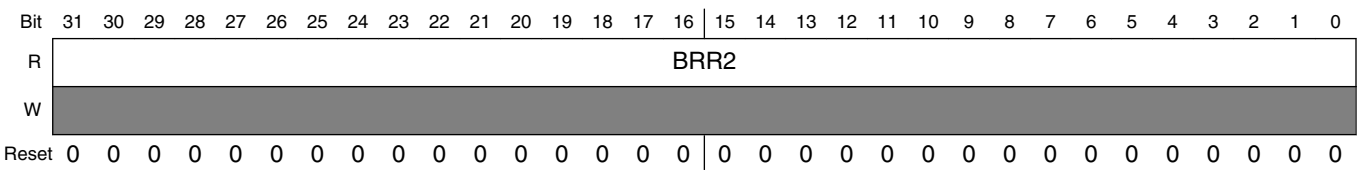
Field	Description
BRR1	Processor B Receive Register 1. (Read-only) • Reflects the data written to Processor A Transmit Register 1 (ATR1). • Reading the BRR1 register clears the “receiver full” bit (RF1) in the Processor B Status Register (BSR) on the receiver side, and sets the “transmitter empty” bit (TE1) in the Processor A Status Register on the transmitter side (optionally triggering a transmit interrupt 1 on the Processor A-side). • Any read of the BRR1 register will update all status information.

4.3.6.7 Processor B Receive Register 2 (MU_BRR2)

Use Processor B Receive Register 2 (BRR2, 32-bit, read-only) to receive a message or data from the Processor A.

- Data written to the ATR2 register is immediately reflected in the BRR2 register.
- You can only read the BRR2 register when the RF2 bit in the BSR register is set to “1”.
- Writing to the BRR2 register generates an error response to the Processor B.

Address: 30AB_0000h base + 18h offset = 30AB_0018h



MU_BRR2 field descriptions

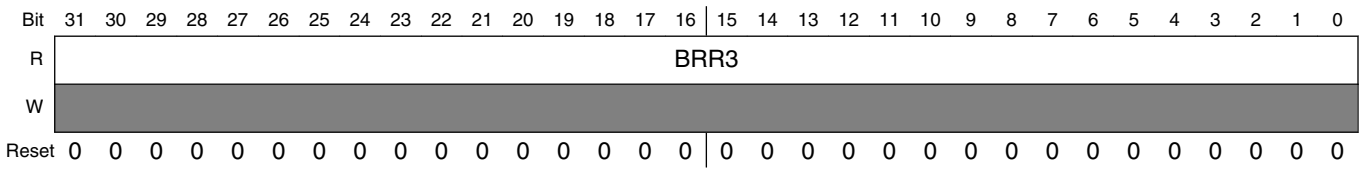
Field	Description
BRR2	Processor B Receive Register 2. (Read-only) • Reflects the data written to Processor A Transmit Register 1 (ATR2). • Reading the BRR2 register clears the “receiver full” bit (RF2) in the Processor B Status Register (BSR) on the receiver side, and sets the “transmitter empty” bit (TE2) in the Processor A Status Register on the transmitter side (optionally triggering a transmit interrupt 2 on the Processor A-side). • Any read of the BRR2 register will update all status information.

4.3.6.8 Processor B Receive Register 3 (MU_BRR3)

Use Processor B Receive Register 3 (BRR3, 32-bit, read-only) to receive a message or data from the Processor A.

- Data written to the ATR3 register is immediately reflected in the BRR3 register.
- You can only read the BRR3 register when the RF3 bit in the BSR register is set to "1".
- Writing to the BRR3 register generates an error response to the Processor B.

Address: 30AB_0000h base + 1Ch offset = 30AB_001Ch



MU_BRR3 field descriptions

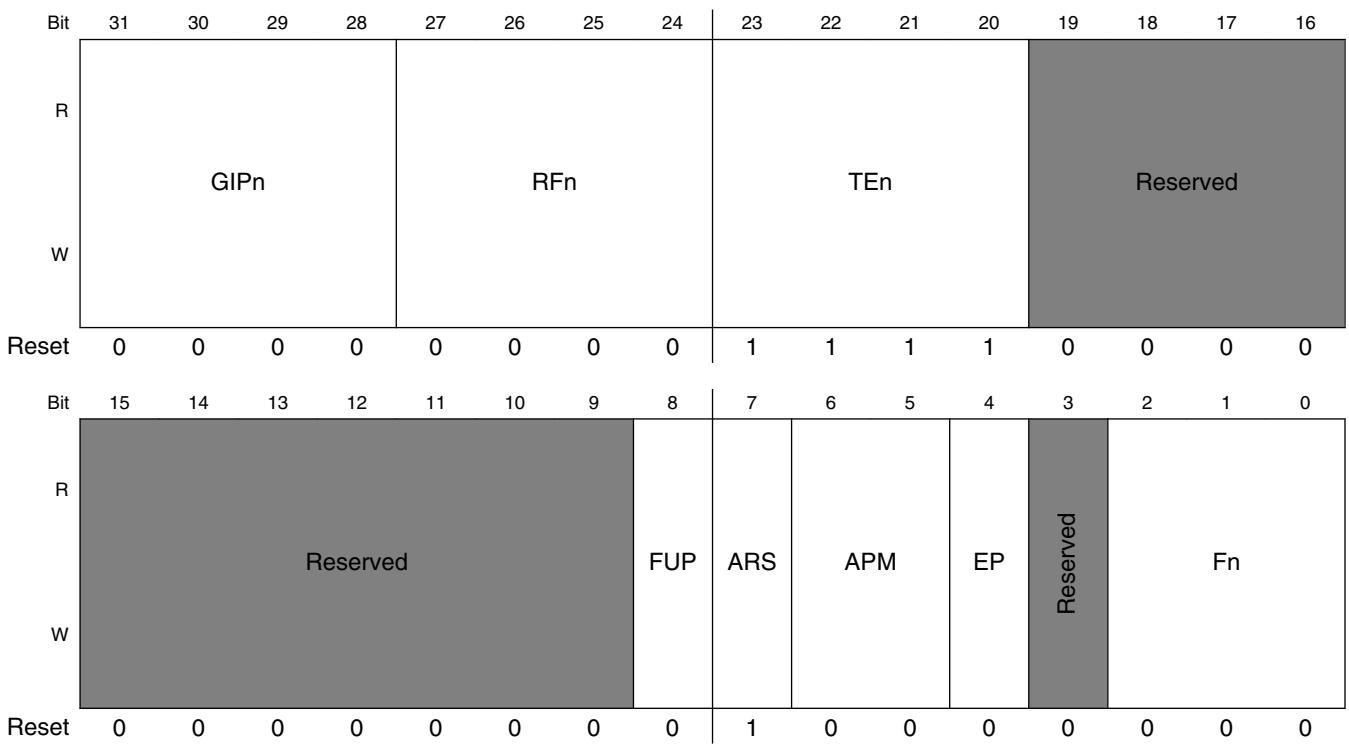
Field	Description
BRR3	Processor B Receive Register 3. (Read-only) • Reflects the data written to Processor A Transmit Register 3 (ATR3). • Reading the BRR3 register clears the "receiver full" bit (RF3) in the Processor B Status Register (BSR) on the receiver side, and sets the "transmitter empty" bit (TE3) in the Processor A Status Register on the transmitter side (optionally triggering a transmit interrupt 3 on the Processor A-side). • Any read of the BRR3 register will update all status information.

4.3.6.9 Processor B Status Register (MU_BSR)

Use the Processor B Status Register (BSR, 32-bit, read-write) to show interrupt status from the Processor B, general purpose flags, the Processor A power mode, and to set dual function control-status bits.

- Dual-purpose bits are set by the Processor B-side programmer, and cleared by the MU logic.

Address: 30AB_0000h base + 20h offset = 30AB_0020h



MU_BSR field descriptions

Field	Description
31–28 GIPn	For n = {0, 1, 2, 3} Processor B General Interrupt Request n Pending. (Read-Write) • GIPn bit signals the Processor B that the GIRn bit in the ACR register on the Processor A-side was set from “0” to “1”. If the GIEn bit in the BCR register is set to “1”, a General Interrupt n request is issued. • The GIPn bit is cleared by writing it back as “1”. Writing “0”, or writing “1” when the GIPn bit is cleared is ignored. Use this feature in the interrupt routine, where the GIPn bit is cleared in order to de-assert the interrupt request source at the interrupt controller. • GIPn bit is cleared when the MU is reset. 0 Processor B general purpose interrupt n is not pending. (default) 1 Processor B general purpose interrupt n is pending.

Table continues on the next page...

MU_BSR field descriptions (continued)

Field	Description
27–24 RFn	For n = {0, 1, 2, 3} Processor B Receive Register n Full. (Read-only) • RFn bit signals to the Processor B-side that new data was written by the Processor A to the ATRn register, and is ready to be read by the Processor B in the BRRn register. • The RFn bit is set to “1” when the ATRn register is written on the Processor A-side. • After the RFn bit is set to “1”, the RFn bit signals the Processor B-side that new data is ready to be read by the Processor B in the BRRn register, and a Receive n interrupt is issued on the Processor A-side (if the RIEn bit in the BCR register has been set to “1”). • RFn bit is cleared when the BRRn register is read, and when the MU is reset. 0 BRRn register is not full (default). 1 BRRn register has received data from ATRn register and is ready to be read by the Processor B.
23–20 TEn	For n = {0, 1, 2, 3} Processor B Transmit Register n Empty. (Read-only) • When TEn = “1”, it signals to the Processor B-side that the BTRn register is ready to be written on the Processor B-side. • The TEn bit is set to “1” after the ARRn register is read on the Processor A-side. • Setting TEn bit will issue a transmit n interrupt on the Processor B-side (if the TIEn bit in the BCR register is set to “1”. • TEn bit is cleared after the BTRn register is written on the Processor B-side. • TEn bit is set to “1” when the MU is reset. 0 BTRn register is not empty. 1 BTRn register is empty (default).
19–9 Reserved	This field is reserved.
8 FUP	Processor B Flags Update Pending. (Read-only) • FUP bit is set to “1” when the Processor B-side sends a Flags Update request to the Processor A-side. • A Flags Update request is generated when the BAF[2:0] bits of the BCR register change. No flag update changes are allowed while the FUP bit is set to “1”. Any write to the BAF[2:0] bits, while the FUP bit is set to “1”, will not generate a Flags Update event, and the BAF[2:0] bits will stay unchanged. • FUP bit is cleared when this Flags Update request is internally acknowledged (that the flag is updated) from the MU Processor A-side, and during MU reset. 0 No flags updated, initiated by the Processor B, in progress (default) 1 Processor B initiated flags update, processing
7 ARS	Processor A Reset State. (Read-only) • ARS bit indicates if the Processor A-side of the MU is in a reset state or not. • If the ARS bit is set to “1”, then the Processor A-side of the MU is still in the reset state. • If the ARS bit is cleared, then both the Processor A and the Processor A-side of the MU are out of reset. • The ARS bit is set to “1” during: a Processor A system reset, or an MU reset (caused by setting the MUR bit at the BCR register).

Table continues on the next page...

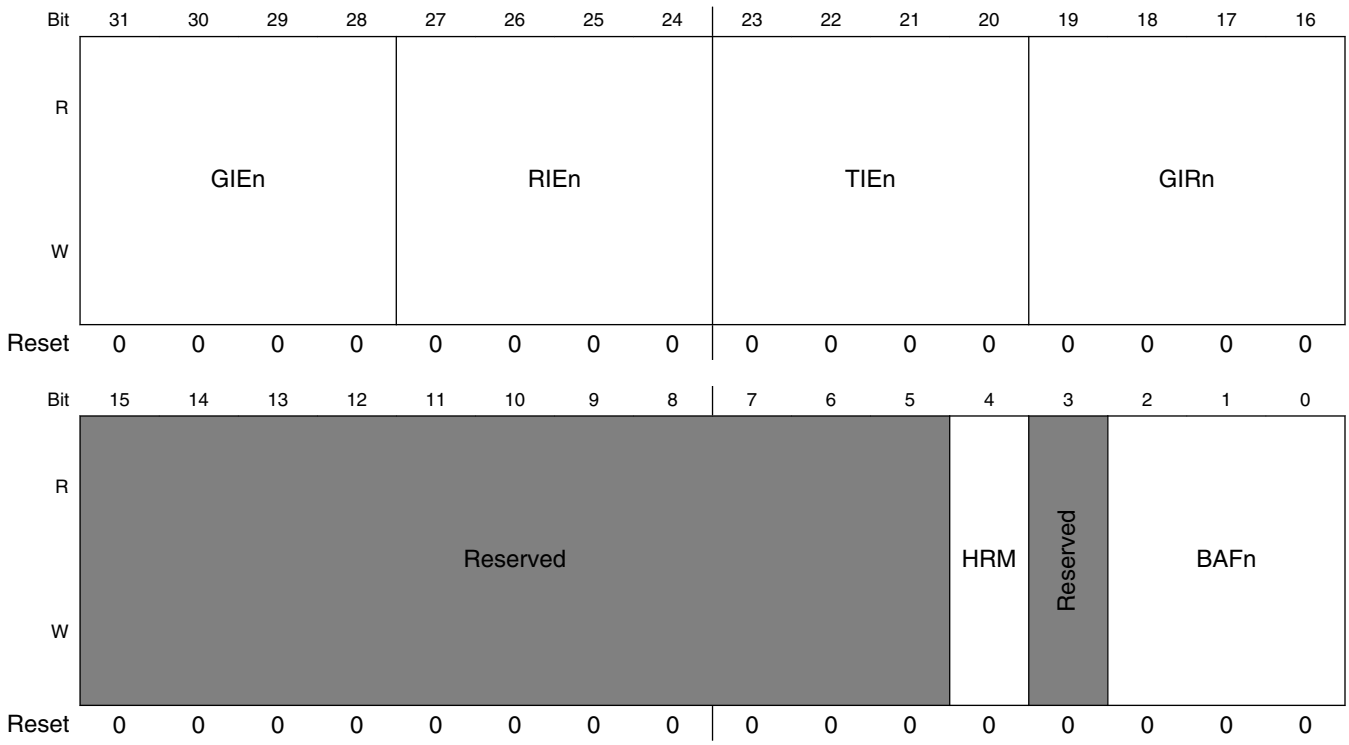
MU_BSR field descriptions (continued)

Field	Description
	• The ARS bit is cleared when the reset sequence on the Processor A-side of the MU ends. After issuing any of the three reset events mentioned previously, you should verify that the ARS bit is cleared before starting any accesses. • When Processor B side of MU comes out of reset ARS bit has value “1”(default).Then Processor B sees the status of Processor A side and if Processor A has come out of reset then ARS bit goes low.This takes 5-6 clock cycles.And if you read ARS bit now you will see it as low although its reset value was "1" . 0 The Processor A or the Processor A-side of the MU is not in reset. 1 The Processor A or the Processor A-side of the MU is in reset.
6–5 APM	Processor A Power Mode. (Read-only) • APM[1:0] bits indicate the Processor A power mode. 00 The System is in Run Mode. 01 The System is in WAIT Mode. 10 Reserved. 11 The System is in STOP Mode.
4 EP	Processor B-Side Event Pending. (Read-only) • EP bit is set to “1” when the Processor B-side mechanism sends an event update request to the Processor A-side. • EP bit is cleared when the event update acknowledge is received. An “event” is any hardware message that is reflected in the ASR register on the Processor A-side (for example, “transmit register 0 written”). During normal operations, you do not have to deal with the state of the EP bit because the event update mechanism works automatically. • To ensure events have been posted to Processor A before entering STOP mode, you should verify that the EP bit is cleared. If EP bit is set to “1”, you should wait and continue to poll it (EP bit) before entering STOP mode. • Reading the BSR register (to check the EP bit) should be the last access to the MU that should be performed before entering STOP mode; otherwise, the EP bit may be set by subsequent additional actions. • Due to Processor B pipeline effects, three NOP operations (or their timing equivalent) should be given after an instruction that sets an event before the EP bit can reflect this event. • The EP bit is cleared when the MU resets. 0 The Processor B-side event is not pending (default). 1 The Processor B-side event is pending.
3 Reserved	This field is reserved.
Fn	For n = {0, 1, 2} Processor B-Side Flag n. (Read-only) • Fn bit is the Processor B-side flag that reflects the values written to the ABFn bit in the Processor A control register. • Every time that the ABFn bit is written, the ABFn bit write event updates the Fn bit after the event update latency, which is measured in terms of the number of clocks of the Processor A and the Processor B. 0 ABFn bit in ACR register is written 0 (default). 1 ABFn bit in ACR register is written 1.

4.3.6.10 Processor B Control Register (MU_BCR)

Use the Processor B Control Register (BCR, 32-bit, read-write) to enable the MU interrupts on the Processor B-side, and trigger events and interrupts on the Processor A-side (wake from STOP, hardware reset, flag update).

Address: 30AB_0000h base + 24h offset = 30AB_0024h



MU_BCR field descriptions

Field	Description
31–28 GIEn	For n = {0, 1, 2, 3} Processor B General Purpose Interrupt Enable n. (Read-Write) • GIEn bit enables Processor B General Interrupt n. • If GIEn bit is set to “1” (enabled), then a General Interrupt n request is issued when the GIPn bit in the BSR register is set to “1”. • If GIEn is cleared (disabled), then the value of the GIPn bit is ignored and no General Interrupt n request will be issued. • GIEn bit is cleared when the MU resets. 0 Disables Processor B General Interrupt n. (default) 1 Enables Processor B General Interrupt n.
27–24 RIEn	For n = {0, 1, 2, 3} Processor B Receive Interrupt Enable n. (Read-Write) • RIEn bit enables Processor B Receive Interrupt n.

Table continues on the next page...

MU_BCR field descriptions (continued)

Field	Description
	• If RIEn bit is set to “1” (enabled), then an Processor B Receive Interrupt n request is issued when the RFn bit in the BSR register is set to “1”. • If RIEn bit is cleared (disabled), then the value of the RFn bit is ignored and no Processor B Receive Interrupt n request will be issued. • RIEn bit is cleared when the MU resets. 0 Disables Processor B Receive Interrupt n. (default) 1 Enables Processor B Receive Interrupt n.
23–20 TIEn	For n = {0, 1, 2, 3} Processor B Transmit Interrupt Enable n. (Read-Write) • TIEn bit enables Processor B Transmit Interrupt n. • If TIEn bit is set to “1” (enabled), then an Processor B Transmit Interrupt n request is issued when the TEn bit in the BSR register is set to “1”. • If TIEn bit is cleared (disabled), then the value of the TEn bit is ignored and no Processor B Transmit Interrupt n request will be issued. • TIEn bit is cleared when the MU resets. 0 Disables Processor B Transmit Interrupt n. (default) 1 Enables Processor B Transmit Interrupt n.
19–16 GIRn	For n = {0, 1, 2, 3} Processor B General Purpose Interrupt Request n. (Read-Write) • Writing “1” to the GIRn bit sets the GIPn bit in the ASR register on the Processor A-side. If the GIEn bit in the ACR register is set to “1” on the Processor A-side, a General Purpose Interrupt n request is triggered. • The GIRn bit is cleared if the GIPn bit (in the ASR register on the Processor A-side) is cleared by writing it (GIPn bit) as “1”, thereby signalling the Processor B that the interrupt was accepted (cleared by the software). The GIPn bit cannot be written as “0” on the Processor B-side. • To ensure proper operations, you must verify that the GIRn bit is cleared (meaning that there is no pending interrupt) before setting it (GIRn bit). • GIRn bit is cleared when the MU resets. 0 Processor B General Interrupt n is not requested to the Processor A (default). 1 Processor B General Interrupt n is requested to the Processor A.
15–5 Reserved	This field is reserved.
4 HRM	Processor B Hardware Reset Mask. (Read-Write) • The Processor A can give a hardware reset to the Processor B by setting the BHR bit in the ACR Register to “1”. • When the HRM bit is set to “1” by the Processor B, the BHR reset issued by the Processor A is masked (disabled by the Processor B). • When the HRM bit is cleared, the BHR reset issued by the Processor A to the Processor B is not masked (enabled by the Processor B). 0 BHR bit in ACR is not masked, enables the hardware reset to the Processor B (default after hardware reset). 1 BHR bit in ACR is masked, disables the hardware reset request to the Processor B.
3 Reserved	This field is reserved.
BAFn	For n = {0, 1, 2} Processor B to Processor A Flag n. (Read-Write)

Table continues on the next page...

MU_BCR field descriptions (continued)

Field	Description
	• BAFn bit is a read-write flag that is reflected in Fn bit in the ASR register on the Processor A-side. • BAFn bit is cleared when the MU resets. 0 Clears the Fn bit in the ASR register. 1 Sets the Fn bit in the ASR register.

4.4 Semaphore (SEMA4)

4.4.1 Overview

The IPS_Semaphores module provides a platform IPS slave device which implements 16 hardware-enforced gates with the following features:

- Module definition supports 16 hardware-enforced gates in a dual-processor configuration, where cp0 is core processor 0 and cp1 is core processor 1
  - Hardware gates appear as a 16-entry byte-size array with read and write accesses
    - Processors lock gates by writing "processor_number+1" to the appropriate gate and must read back the gate value to verify the lock operation was successful
    - Once locked, the gate is unlocked by a write of zeroes from the locking processor
  - Optional interrupt notification after a failed lock write provides a mechanism to indicate when the gate is unlocked
  - Secure reset mechanisms are supported to clear the contents of individual semaphore gates or notification logic, as well as a clear_all capability
  - Programming model allocates memory space to support up to 8 processors and up to 64 gates

A simplified block diagram of the Semaphores module is shown in Figure 4-10. In the diagram, the register blocks named gate0, gate1,…, gate 15 include the finite state machines implementing the semaphore gates plus the interrupt notification logic.

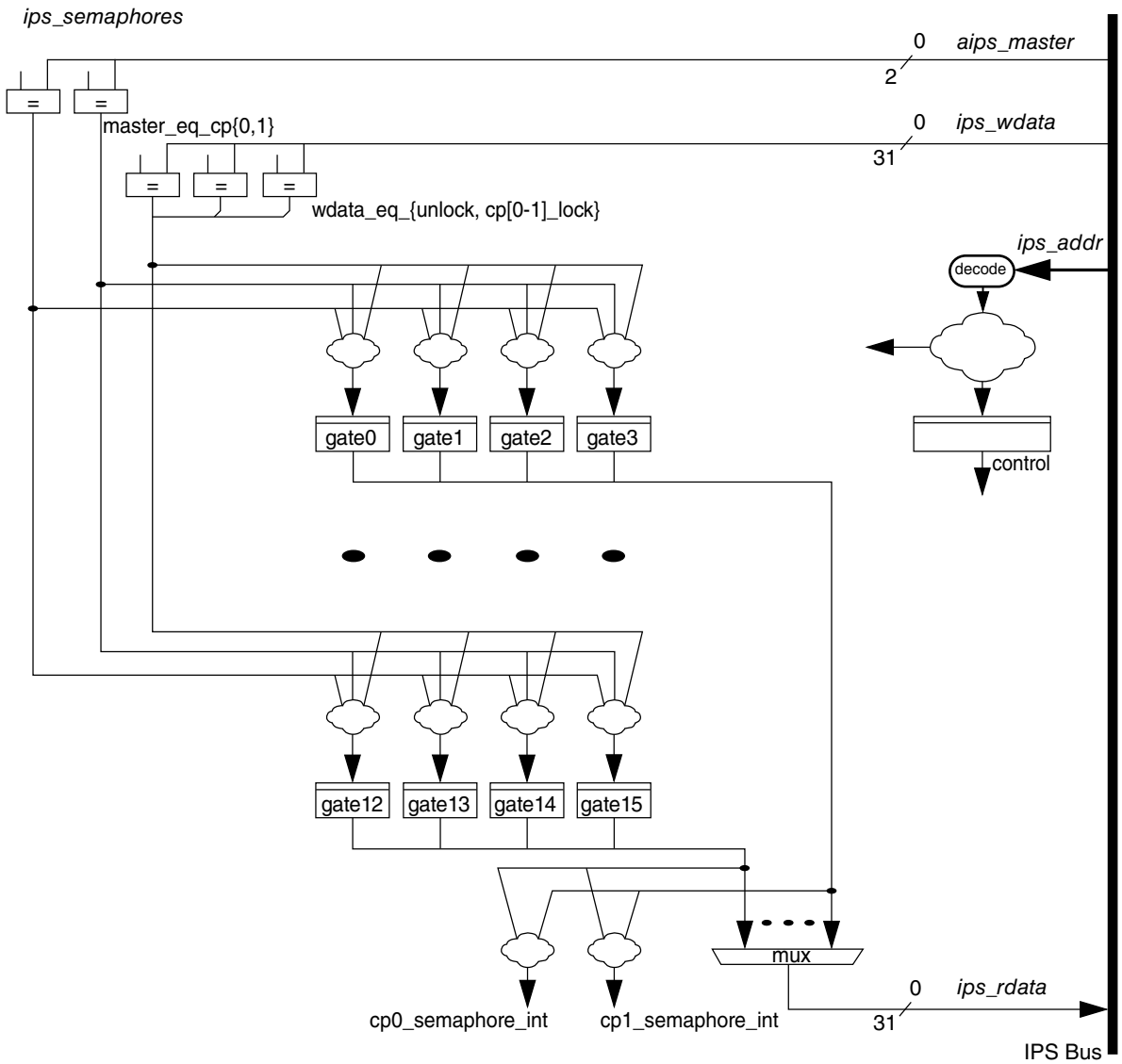


Figure 4-10. IPS_Semaphores Block Diagram

4.4.1.1 Features

The Semaphores module implements hardware-enforced semaphores as an IPS-mapped slave peripheral device. The feature set includes:

- Support for 16 hardware-enforced gates in a dual-processor configuration
  - Each hardware gate appears as a 3-state, 2-bit state machine, with all 16 gates mapped as a byte-size array
    - 3-state implementation

      if gate = 0b00, then state = unlocked
      if gate = 0b01, then state = locked by processor 0

if gate = 0b10, then state = locked by processor 1

- Uses the bus master number as a reference attribute plus the specified data patterns to validate all write operations
- Once locked, the gate can (and must) be unlocked by a write of zeroes from the locking processor

- Optional interrupt notification after a failed lock write provides a mechanism to indicate when the gate is unlocked
- Secure reset mechanisms are supported to clear the contents of individual gates or notification logic, as well as a clear_all capability

- Memory-mapped IPS slave peripheral platform module
  - Interface to the IPS bus for programming-model accesses
  - Two outputs (one per processor) for interrupt notification of failed lock writes

4.4.1.2 Modes of Operation

The Semaphores module does not support any special modes of operation. As a slave peripheral memory-mapped device located on the platform's IPS slave bus, it responds based strictly on the memory addresses of the connected bus. The IPS bus is used to access the Semaphores ' programming model.

4.4.2 External Signal Description

The Semaphores module does not include any external interfaces.

4.4.3 Functional Description

In this section, the functional operation of the Semaphores module, specifically the state machines of the SEMA4_GATEn and SEMA4_CPnNTF registers are detailed.

4.4.3.1 SEMA4_GATEn Operation

Recall each of the SEMA4_GATEn registers implements a 2-bit, 3-state machine. The state transitions for each gate are shown in the following figure.

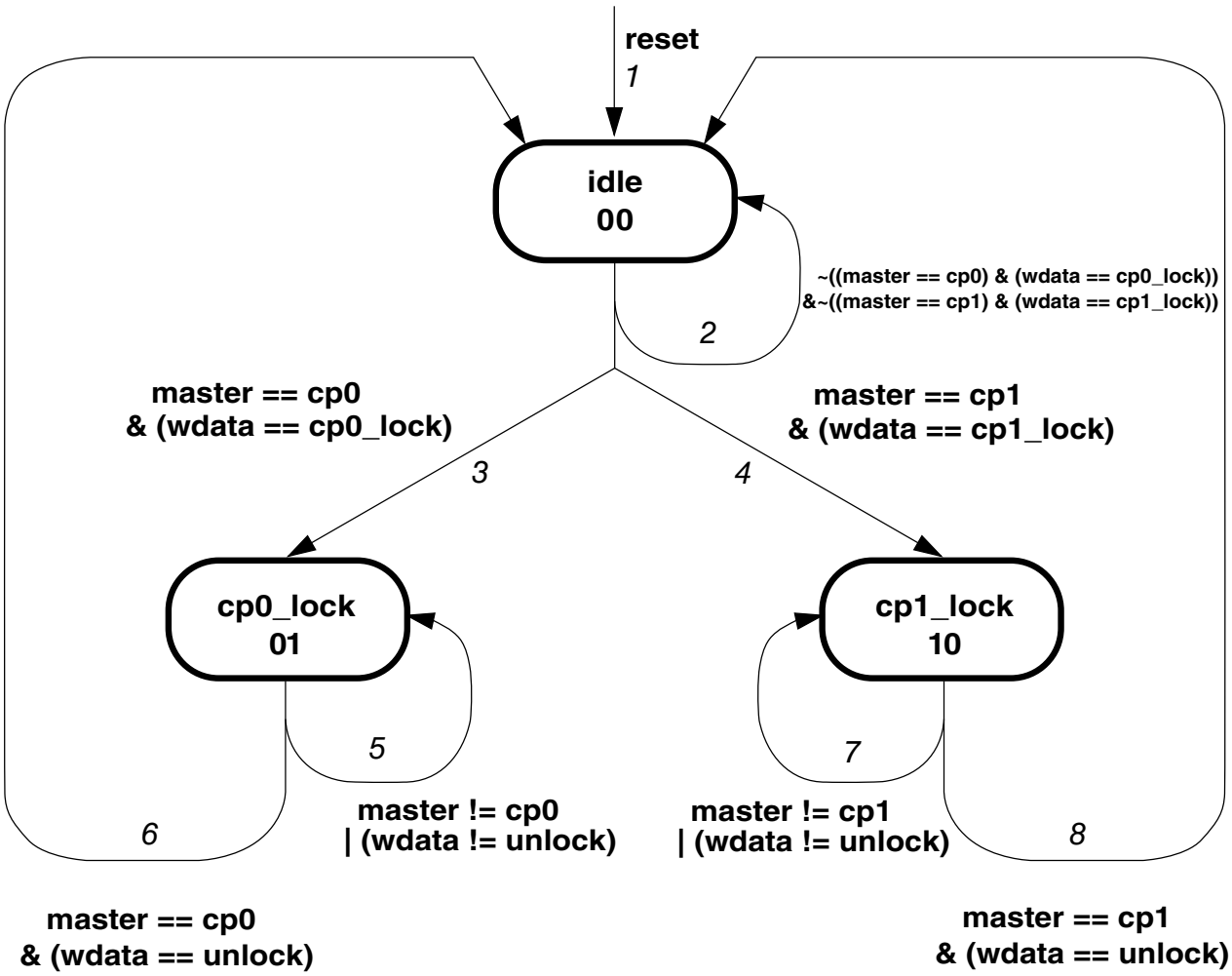


Figure 4-11. SEMA4_GATEn State Machine

The bus master number is used to identify core processor 0 (cp0) or core processor 1 (cp1). The Standard (or Reduced) Product Platform passes the AHB bus master number (`hmaster[2:0]`) through the AIPS (or AIPS-Lite) controller and drives an `aips_master[2:0]` output to the Semaphores module as an IPS sideband signal.

The state transitions for SEMA4_GATEn are defined in the following table.

Table 4-20. SEMA4_GATEn State Transitions

Current State	Next State	Transition	Description
–	idle	1	Any reset, whether a system reset or an individual gate reset, unconditionally forces the gate into the idle state.
idle	idle	2	Unless a write of the appropriate lock value from the corresponding processor occurs, the gate remains in the idle state.
idle	cp0_lock	3	When a write of the "cp0_lock" data value is initiated by processor 0, the gate transitions into the cp0_lock state.

Table continues on the next page...

Table 4-20. SEMA4_GATEn State Transitions (continued)

Current State	Next State	Transition	Description
idle	cp1_lock	4	When a write of the "cp1_lock" value is initiated by processor 1, the gate transitions into the cp1_lock state.
cp0_lock	cp0_lock	5	Once in this state, the gate remains here if any attempted write is not from cp0 with the unlock data value.
cp0_lock	idle	6	The gate returns to the idle (unlocked) state once a write from cp0 with the unlock data value occurs.
cp1_lock	cp1_lock	7	Once in this state, the gate remains here if any attempted write is not from cp1 with the unlock data value.
cp1_lock	idle	8	The gate returns to the idle (unlocked) state once a write from cp1 with the unlock data value occurs.

4.4.3.2 SEMA4_CPnNTF Operation

The failed lock write notification interrupt request is implemented in a 3-bit, 5-state machine which records failed lock attempts and transitions based on gate locking and unlocking. Two specific states are encoded and program-visible as SEMA4_CP0NTF[GNn] and SEMA4_CP1NTF[GNn]. See the following figure.

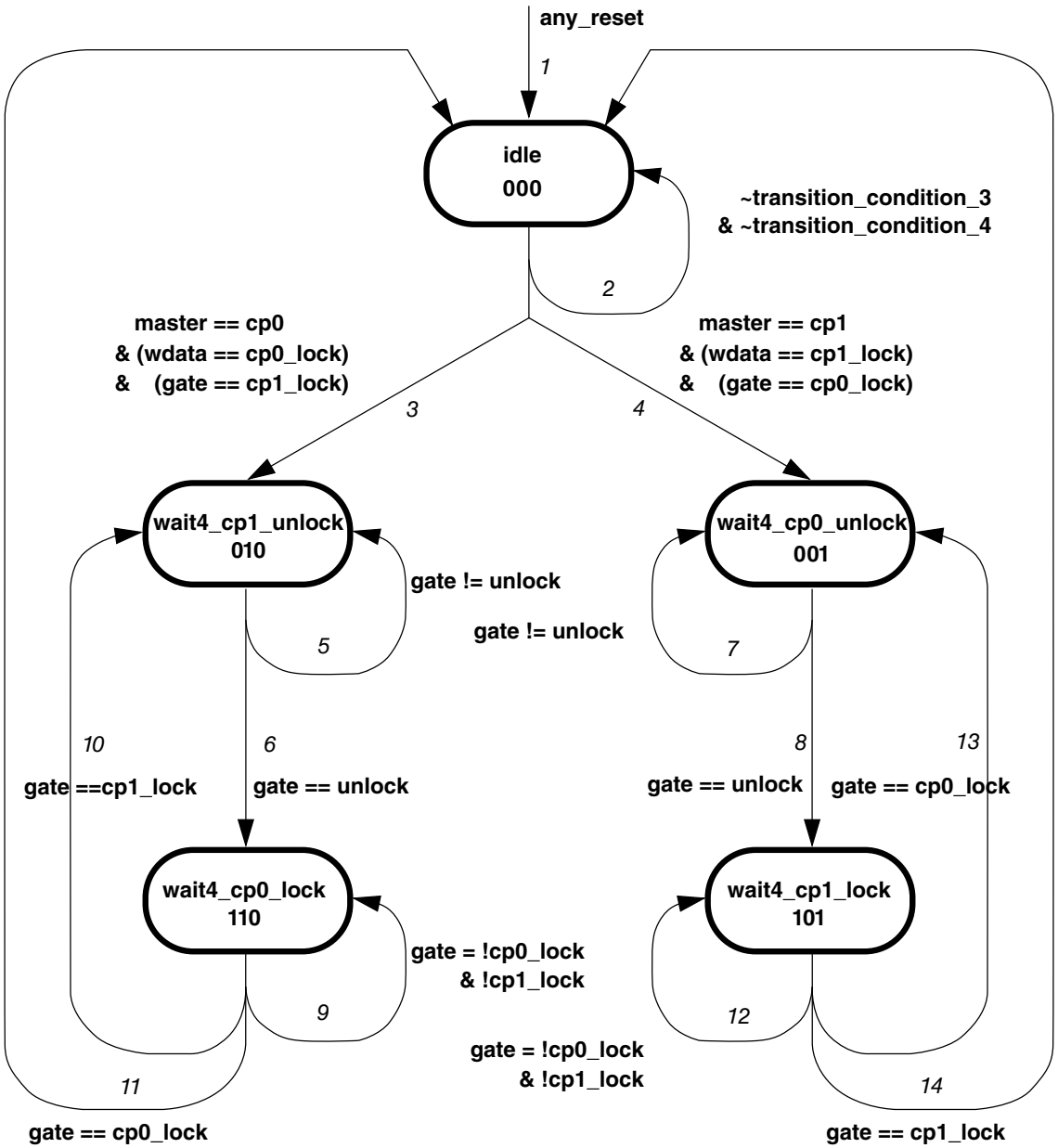


Figure 4-12. IRQ Notification State Machine

The state transitions of the IRQ notification function are defined in the following. Specific states of this machine are program-visible as the SEMA4_CPnNTF registers. In particular, two states are program-visible:

```
if state = wait4_cp0_lock (0b110) // generate cp0_semaphore_int if properly enabled
                then SEMA4_CP0NTF[GNn] = 1; else SEMA4_CP0NTF[GNn] = 0
if state = wait4_cp1_lock (0b101) // generate cp1_semaphore_int if properly enabled
                then SEMA4_CP1NTF[GNn] = 1; else SEMA4_CP1NTF[GNn] = 0
```

Table 4-21. IRQ Notification State Transitions

Current State	Next State	Transition	Description
–	idle	1	Any reset, including a system reset or an individual notification or secure gate reset, unconditionally forces the machine into the idle state.
idle	idle	2	Unless a write of the appropriate lock value from the corresponding processor to an already-locked gate occurs, the machine remains in the idle state.
idle	wait4_cp1_unlock	3	When a write of the "cp0_lock" data value is initiated by processor 0 but the gate is already locked by cp1, the machine transitions into this state, where it waits for cp1 to unlock the gate.
idle	wait4_cp0_unlock	4	When a write of the "cp1_lock" data value is initiated by processor 1but the gate is already locked by cp0, the machine transitions into this state, where it waits for cp0 to unlock the gate
wait4_cp1_unlock	wait4_cp1_unlock	5	Once in this state, the machine remains here until the gate is unlocked.
wait4_cp1_unlock	wait4_cp0_lock	6	From this state, the machine transitions into the next state, waiting for cp0 to lock the gate, once it has been unlocked.
wait4_cp0_unlock	wait4_cp0_unlock	7	Once in this state, the machine remains here until the gate is unlocked.
wait4_cp0_unlock	wait4_cp1_lock	8	From this state, the machine transitions into the next state, waiting for cp1 to lock the gate, once it has been unlocked.
wait4_cp0_lock	wait4_cp0_lock	9	In this state, the machine generates the notification interrupt (if properly-enabled) and remains here until the gate is locked by processor 0 or the gate is again locked by processor 1.
wait4_cp0_lock	wait4_cp1_unlock	10	In this state, the machine generates the notification interrupt (if properly-enabled) and transitions if the gate is again locked by processor 1. With this transition, the notification interrupt request is negated.
wait4_cp0_lock	idle	11	In this state, the machine generates the notification interrupt (if properly-enabled) and transitions if the gate is finally locked by processor 0. With this transition, the notification interrupt request is negated.
wait4_cp1_lock	wait4_cp1_lock	12	In this state, the machine generates the notification interrupt (if properly-enabled) and remains here until the gate is locked by processor 1or the gate is again locked by processor 0.
wait4_cp1_lock	wait4_cp0_unlock	13	In this state, the machine generates the notification interrupt (if properly-enabled) and transitions if the gate is again locked by processor 0. With this transition, the notification interrupt request is negated.
wait4_cp1_lock	idle	14	In this state, the machine generates the notification interrupt (if properly-enabled) and transitions if the gate is finally locked by processor 1. With this transition, the notification interrupt request is negated.

The Semaphores module generates two interrupt request output signals, one per processor, combining the SEMA4_CPnINE and SEMA4_CPnNTF registers, where the boolean equations are:

```
cp0_semaphore_int
    =    sema4_cp0ine[ine0]    &    sema4_cp0ntf[gn0]
    |    sema4_cp0ine[ine1]    &    sema4_cp0ntf[gn1]
    |    sema4_cp0ine[ine2]    &    sema4_cp0ntf[gn2]
    ...
    |    sema4_cp0ine[ine15]    &    sema4_cp0ntf[gn15]
cp1_semaphore_int
    =    sema4_cp1ine[ine0]    &    sema4_cp1ntf[gn0]
    |    sema4_cp1ine[ine1]    &    sema4_cp1ntf[gn1]
    |    sema4_cp1ine[ine2]    &    sema4_cp1ntf[gn2]
    ...
    |    sema4_cp1ine[ine15]    &    sema4_cp1ntf[gn15]
```

4.4.4 Initialization Information

The reset state of the IPS_Semaphores module allows it to begin operation without the need for any further initialization. All the internal state machines are cleared by any reset event, allowing the module to immediately begin operation.

4.4.5 Application Information

In an operational multi-core system, most interactions involving the Semaphores module involves reads and writes to the SEMA4_GATEn registers for implementation of the hardware-enforced software gate functions. Typical code segments for gate functions perform the following operations:

- To lock (close) a gate
    - The processor performs a byte write of "logical_processor_number + 1" to gate[i]
    - The processor reads back gate[i] and checks for a value of "logical_processor_number + 1"

If the compare indicates the expected value, then the gate is locked; proceed with the protected code segment. If the compare does not indicate the expected value, the lock operation failed; repeat the process beginning with byte write to gate[i] in spin-wait loop, or proceed with another execution path and wait for failed lock interrupt notification.

A simple C-language example of a `gateLock` function is shown in the following figure. This function follows the Hennessy/Patterson example described in Multi-Core Programming 101: Software Gates.

```
#define UNLOCK     0
#define CP0_LOCK   1
#define CP2_LOCK   2

void gateLock (n)
int   n;                                          /* gate number to lock */
{
   int i;
   int current_value;
   int locked_value;

   i = processor_number();  /* obtain logical CPU number */

   if (i == 0)
      locked_value = CP0_LOCK;
   else
      locked_value = CP1_LOCK;

   /* read the current value of the gate and wait until the state == UNLOCK */
   do {
        current_value = gate[n];
   } while (current_value != UNLOCK);

   /* the current value of the gate == UNLOCK. attempt to lock the gate for this
        processor. spin-wait in this loop until gate ownership is obtained */
   do {
      gate[n] = locked_value;  /* write gate with processor_number + 1 */
      current_value = gate[n]; /* read gate to verify ownership was obtained */
   } while (current_value != locked_value);
```

Figure 4-13. Sample gateLock Function

- To unlock (open) a gate
    - After completing the protected code segment, the locking processor performs a byte write of zeroes to gate[i], opening (unlocking) the gate

A few comments on the logical CPU number are appropriate. In this example, a reference to `processor_number()` is used to retrieve this hardware configuration value. Typically, the logical processor numbers are defined by a hardwired input vector to the individual cores. The exact method for accessing the logical processor number varies by architecture. For PowerPC cores, there is a processor ID register (PIR) which is SPR 286 and contains this value. A single instruction can be used to move the contents of the PIR into a general-purpose register: mfspr rx,286 where rx is the destination GPRn. Other architectures may support a specific instruction to move the contents of the logical processor number into a general-purpose register, e.g., rdcpn rx for a "read CPU number" instruction.

If the optional failed lock IRQ notification mechanisms are used, then accesses to the related registers (SEMA4_CPnINE, SEMA4_ CPnNTF) are required. Note that there is no required negation of the failed lock write notification interrupt as the request is automatically negated by the Semaphores module once the gate has been successfully locked by the "failing" processor.

Finally, in the event a system state requires a software-controlled reset of a gate or IRQ notification register(s), accesses to the secure reset control registers (SEMA4_RSTGT, SEMA4_RSTNTF) are required. For these situations, it is recommended that the appropriate IRQ notification enable(s) (SEMA4_CPnINE) bits be disabled *before* initiating the secure reset 2-write sequence to avoid any race conditions involving spurious notification interrupt requests.

4.4.6 Memory map and register definition

The Semaphores module provides an IPS programming model mapped to an SPP-standard on-platform 16 KB space. The description here specifies a dual-core configuration with 16 semaphore gates. All the register names are prefixed with "Sema4" as an abbreviation for the full module name.

The programming model is referenced using 8-, 16- and 32-bit accesses. Reads can use any reference size, while writes are generally restricted to the size of the register. Exceptions to the write size restrictions are detailed in the individual register descriptions. Attempted references using inappropriate access sizes, to undefined (reserved) addresses, or with a non-supported access type (for example, a write to a read-only register) generate an IPS error termination.

Finally, the programming model allocates space for a definition with up to 64 gates and up to 8 processor cores, even though this definition is considerably larger than any currently-planned module implementations. The number of gates and supported processor cores are independent; there is no relationship between these two system variables.

The 16 KB Semaphores programming model map is shown in the following table.

SEMA4 memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30AC_0000	Semaphores Gate 0 Register (SEMA4_Gate00)	8	R/W	00h	4.4.6.1/328
30AC_0001	Semaphores Gate 1 Register (SEMA4_Gate01)	8	R/W	00h	4.4.6.2/329
30AC_0002	Semaphores Gate 2 Register (SEMA4_Gate02)	8	R/W	00h	4.4.6.3/330
30AC_0003	Semaphores Gate 3 Register (SEMA4_Gate03)	8	R/W	00h	4.4.6.4/331
30AC_0004	Semaphores Gate 4 Register (SEMA4_Gate04)	8	R/W	00h	4.4.6.5/332
30AC_0005	Semaphores Gate 5 Register (SEMA4_Gate05)	8	R/W	00h	4.4.6.6/333
30AC_0006	Semaphores Gate 6 Register (SEMA4_Gate06)	8	R/W	00h	4.4.6.7/334
30AC_0007	Semaphores Gate 7 Register (SEMA4_Gate07)	8	R/W	00h	4.4.6.8/335

Table continues on the next page...

SEMA4 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30AC_0008	Semaphores Gate 8 Register (SEMA4_Gate08)	8	R/W	00h	4.4.6.9/336
30AC_0009	Semaphores Gate 9 Register (SEMA4_Gate09)	8	R/W	00h	4.4.6.10/ 337
30AC_000A	Semaphores Gate 10 Register (SEMA4_Gate10)	8	R/W	00h	4.4.6.11/ 338
30AC_000B	Semaphores Gate 11 Register (SEMA4_Gate11)	8	R/W	00h	4.4.6.12/ 339
30AC_000C	Semaphores Gate 12 Register (SEMA4_Gate12)	8	R/W	00h	4.4.6.13/ 340
30AC_000D	Semaphores Gate 13 Register (SEMA4_Gate13)	8	R/W	00h	4.4.6.14/ 341
30AC_000E	Semaphores Gate 14 Register (SEMA4_Gate14)	8	R/W	00h	4.4.6.15/ 342
30AC_000F	Semaphores Gate 15 Register (SEMA4_Gate15)	8	R/W	00h	4.4.6.16/ 343
30AC_0040	Semaphores Processor n IRQ Notification Enable (SEMA4_CP0INE)	16	R/W	0000h	4.4.6.17/ 344
30AC_0048	Semaphores Processor n IRQ Notification Enable (SEMA4_CP1INE)	16	R/W	0000h	4.4.6.17/ 344
30AC_0080	Semaphores Processor n IRQ Notification (SEMA4_CP0NTF)	16	R	0000h	4.4.6.18/ 346
30AC_0088	Semaphores Processor n IRQ Notification (SEMA4_CP1NTF)	16	R	0000h	4.4.6.18/ 346
30AC_0100	Semaphores (Secure) Reset Gate n (SEMA4_RSTGT)	16	R/W	0000h	4.4.6.19/ 348
30AC_0104	Semaphores (Secure) Reset IRQ Notification (SEMA4_RSTNTF)	16	R/W	0000h	4.4.6.20/ 349

4.4.6.1 Semaphores Gate 0 Register (SEMA4_Gate00)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate

register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + 0h offset = 30AC_0000h

Bit	0	1	2	3	4	5	6	7
Read	0						GTFSM	
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate00 field descriptions

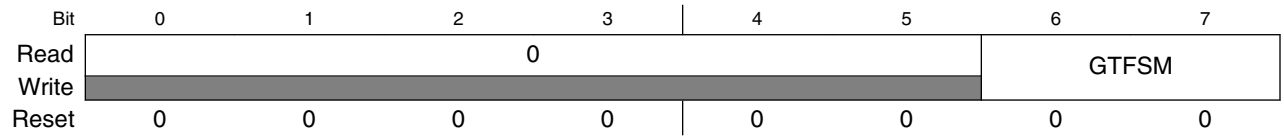
Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.2 Semaphores Gate 1 Register (SEMA4_Gate01)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + 1h offset = 30AC_0001h



SEMA4_Gate01 field descriptions

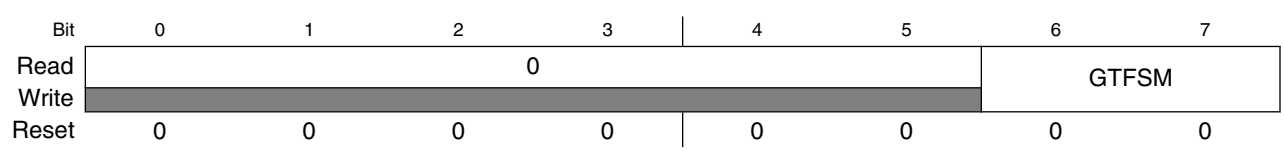
Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.3 Semaphores Gate 2 Register (SEMA4_Gate02)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + 2h offset = 30AC_0002h



SEMA4_Gate02 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.4 Semaphores Gate 3 Register (SEMA4_Gate03)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + 3h offset = 30AC_0003h

Bit	0	1	2	3	4	5	6	7
Read	0						GTFSM	
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate03 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SEMA4_Gate03 field descriptions (continued)

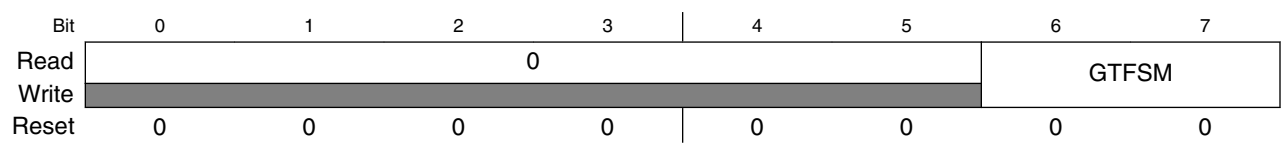
Field	Description
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.5 Semaphores Gate 4 Register (SEMA4_Gate04)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + 4h offset = 30AC_0004h



SEMA4_Gate04 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation .

Table continues on the next page...

SEMA4_Gate04 field descriptions (continued)

Field	Description
	NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.6 Semaphores Gate 5 Register (SEMA4_Gate05)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + 5h offset = 30AC_0005h

Bit	0	1	2	3	4	5	6	7
Read	0						GTFSM	
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate05 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free).

Table continues on the next page...

SEMA4_Gate05 field descriptions (continued)

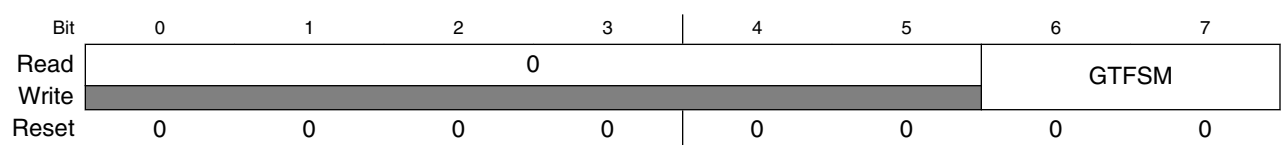
Field	Description
	01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.7 Semaphores Gate 6 Register (SEMA4_Gate06)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + 6h offset = 30AC_0006h



SEMA4_Gate06 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.8 Semaphores Gate 7 Register (SEMA4_Gate07)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + 7h offset = 30AC_0007h

Bit	0	1	2	3	4	5	6	7
Read	0						GTFSM	
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate07 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.9 Semaphores Gate 8 Register (SEMA4_Gate08)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + 8h offset = 30AC_0008h

Bit	0	1	2	3	4	5	6	7
Read	0						GTFSM	
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate08 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.10 Semaphores Gate 9 Register (SEMA4_Gate09)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + 9h offset = 30AC_0009h

Bit	0	1	2	3	4	5	6	7
Read	0						GTFSM	
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate09 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.11 Semaphores Gate 10 Register (SEMA4_Gate10)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + Ah offset = 30AC_000Ah

Bit	0	1	2	3	4	5	6	7
Read	0						GTFSM	
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate10 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.12 Semaphores Gate 11 Register (SEMA4_Gate11)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + Bh offset = 30AC_000Bh

Bit	0	1	2	3	4	5	6	7
Read	0						GTFSM	
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate11 field descriptions

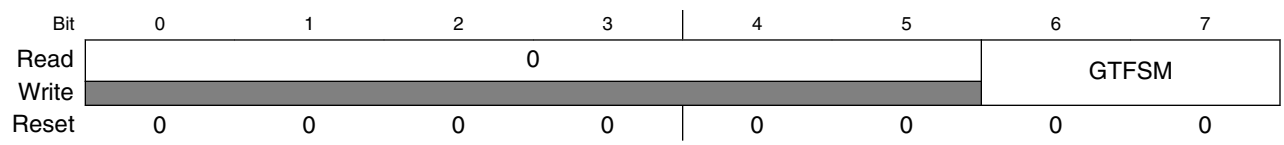
Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.13 Semaphores Gate 12 Register (SEMA4_Gate12)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + Ch offset = 30AC_000Ch



SEMA4_Gate12 field descriptions

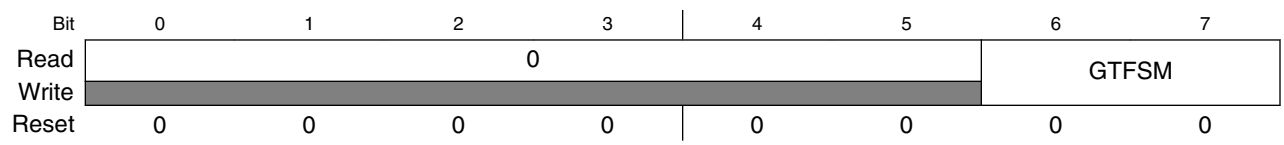
Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.14 Semaphores Gate 13 Register (SEMA4_Gate13)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + Dh offset = 30AC_000Dh



SEMA4_Gate13 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.15 Semaphores Gate 14 Register (SEMA4_Gate14)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + Eh offset = 30AC_000Eh

Bit	0	1	2	3	4	5	6	7
Read	0						GTFSM	
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate14 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.16 Semaphores Gate 15 Register (SEMA4_Gate15)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 30AC_0000h base + Fh offset = 30AC_000Fh

Bit	0	1	2	3	4	5	6	7
Read	0						GTFSM	
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate15 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . **NOTE:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

4.4.6.17 Semaphores Processor n IRQ Notification Enable (SEMA4_CP*n*INE)

The application of a hardware semaphore module provides an opportunity for implementation of helpful system-level features. An example is an optional mechanism to generate a processor interrupt after a failed lock attempt. Recall traditional software gate functions execute a spin-wait loop in an effort to obtain and lock the referenced gate. With this module, the processor that fails in the lock attempt could continue with other tasks and allow a properly-enabled notification interrupt to return its execution to the original lock function.

The optional notification interrupt function consists of two registers for each processor: an interrupt notification enable register (SEMA4_CPnINE) and the interrupt request register (SEMA4_CPnNTF). To support implementations with more than 16 gates, these registers can be referenced with aligned 16- or 32-bit accesses. For the SEMA4_CPnINE registers, unimplemented bits read as zeroes, and writes are ignored.

Address: 30AC_0000h base + 40h offset + (8d × i), where i=0d to 1d

Bit	15	14	13	12	11	10	9	8
Read Write	INE8	INE9	INE10	INE11	INE12	INE13	INE14	INE15
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	INE0	INE1	INE2	INE3	INE4	INE5	INE6	INE7
Reset	0	0	0	0	0	0	0	0

SEMA4_CP*n*INE field descriptions

Field	Description
15 INE8	Interrupt Request Notification Enable 8. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 8. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
14 INE9	Interrupt Request Notification Enable 9. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 9. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
13 INE10	Interrupt Request Notification Enable 10. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 10. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
12 INE11	Interrupt Request Notification Enable 11. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 11.

Table continues on the next page...

SEMA4_CP*n*INE field descriptions (continued)

Field	Description
	0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
11 INE12	Interrupt Request Notification Enable 12. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 12. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
10 INE13	Interrupt Request Notification Enable 13. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 13. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
9 INE14	Interrupt Request Notification Enable 14. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 14. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
8 INE15	Interrupt Request Notification Enable 15. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 15. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
7 INE0	Interrupt Request Notification Enable 0. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 0. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
6 INE1	Interrupt Request Notification Enable 1. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 1. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
5 INE2	Interrupt Request Notification Enable 2. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 2. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
4 INE3	Interrupt Request Notification Enable 3. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 3. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
3 INE4	Interrupt Request Notification Enable 4. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 4. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
2 INE5	Interrupt Request Notification Enable 5. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 5.

Table continues on the next page...

SEMA4_CP*n*INE field descriptions (continued)

Field	Description
	0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
1 INE6	Interrupt Request Notification Enable 6. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 6. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
0 INE7	Interrupt Request Notification Enable 7. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 7. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.

4.4.6.18 Semaphores Processor n IRQ Notification (SEMA4_CP*n*NTF)

The Semaphores module optionally allows the processor that fails in the lock attempt to continue with other tasks and allow a properly-enabled notification interrupt to return its execution to the original lock function rather than simply execute in a spin-wait loop.

The optional notification interrupt mechanism consists of two registers for each processor: an interrupt notification enable register (SEMA4_CPnINE) and the read-only notification interrupt request register (SEMA4_CPnNTF). To support implementations with more than 16 gates, these registers can be referenced with aligned 16- or 32-bit accesses. For the SEMA4_CPnNTF registers, unimplemented bits read as zeroes.

The notification interrupt is generated via a unique finite state machine, one per hardware gate. This machine operates in the following manner:

1. When an attempted lock fails, the FSM enters a first state where it waits until the gate is unlocked.
2. Once unlocked, the FSM enters a second state where it generates an interrupt request to the “failed lock” processor.
3. When the “failed lock” processor succeeds in locking the gate, the IRQ is automatically negated and the FSM returns to the idle state. However, if the other processor again locks the gate, the FSM returns to the first state, negates the interrupt request, and then waits for the gate to be unlocked (again).

The notification interrupt request is implemented in a 3-bit, 5-state machine, where tw specific states are encoded and program-visible as SEMA4_CP0NTF[GNn] and SEMA4_CP1NTF[GNn].

Address: 30AC_0000h base + 80h offset + (8d × i), where i=0d to 1d

Bit	15	14	13	12	11	10	9	8
Read Write	GN8	GN9	GN10	GN11	GN12	GN13	GN14	GN15
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	GN0	GN1	GN2	GN3	GN4	GN5	GN6	GN7
Reset	0	0	0	0	0	0	0	0

SEMA4_CP*n*NTF field descriptions

Field	Description
15 GN8	Gate 8 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 8. For more details, see SEMA4_CPnNTF Operation .
14 GN9	Gate 9 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 9. For more details, see SEMA4_CPnNTF Operation .
13 GN10	Gate 10 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 10. For more details, see SEMA4_CPnNTF Operation .
12 GN11	Gate 11 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 11. For more details, see SEMA4_CPnNTF Operation .
11 GN12	Gate 12 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 12. For more details, see SEMA4_CPnNTF Operation .
10 GN13	Gate 13 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 13. For more details, see SEMA4_CPnNTF Operation .
9 GN14	Gate 14 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 14. For more details, see SEMA4_CPnNTF Operation .
8 GN15	Gate 15 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 15. For more details, see SEMA4_CPnNTF Operation .
7 GN0	Gate 0 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 0. For more details, see SEMA4_CPnNTF Operation .
6 GN1	Gate 1 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 1. For more details, see SEMA4_CPnNTF Operation .
5 GN2	Gate 2 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 2. For more details, see SEMA4_CPnNTF Operation .
4 GN3	Gate 3 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 3. For more details, see SEMA4_CPnNTF Operation .
3 GN4	Gate 4 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 4. For more details, see SEMA4_CPnNTF Operation .
2 GN5	Gate 5 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 5. For more details, see SEMA4_CPnNTF Operation .
1 GN6	Gate 6 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 6. For more details, see SEMA4_CPnNTF Operation .
0 GN7	Gate 7 Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate 7. For more details, see SEMA4_CPnNTF Operation .

4.4.6.19 Semaphores (Secure) Reset Gate n (SEMA4_RSTGT)

Although the intent of the hardware gate implementation specifies a protocol where the locking processor must unlock the gate, it is recognized that system operation may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset.

To support this special gate reset requirement, the Semaphores module implements a "secure" reset mechanism which allows a hardware gate (or all the gates) to be initialized by following a specific dual-write access pattern. Using a technique similar to that required for the servicing of a software watchdog timer, the secure gate reset requires two consecutive writes with predefined data patterns from the same processor to force the clearing of the specified gate(s). The required access pattern is:

1. A processor performs a 16-bit write to the SEMA4_RSTGT memory location. The most significant byte (SEMA4_RSTGT[RSTGDP]) must be 0xe2; the least significant byte is a "don't_care" for this reference.
2. The same processor then performs a second 16-bit write to the SEMA4_RSTGT location. For this write, the upper byte (SEMA4_RSTGT[RSTGDP]) is the logical complement of the first data pattern (0x1d) and the lower byte (SEMA4_RSTGT[RSTGTN]) specifies the gate(s) to be reset. This gate field can specify a single gate be cleared, or that all gates are cleared.
3. Reads of the SEMA4_RSTGT location return information on the 2-bit state machine (SEMA4_RSTGT[RSTGSM]) which implements this function, the bus master performing the reset (SEMA4_RSTGT[RSTGMS]) and the gate number(s) last cleared (SEMA4_RSTGT[RSTGTN]). Reads of the SEMA4_RSTGT register do not affect the secure reset finite state machine in any manner.

Address: 30AC_0000h base + 100h offset = 30AC_0100h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	RSTGTN								RSTGSM_RSTGMS_RSTGDP							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SEMA4_RSTGT field descriptions

Field	Description
15–8 RSTGTN	Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write. If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates. The corresponding secure IRQ notification state machine(s) are also reset.
RSTGSM_ RSTGMS_ RSTGDP	**NOTE:** This field contains subfields that vary depending on whether it is being read or written. Sub-fields indicated as having read access are valid only for read operations. Sub-fields indicated as having

Table continues on the next page...

SEMA4_RSTGT field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>write access are valid only for write operations. Bit numbering in the descriptions begins with the most significant bit numbered 0. See the following table for details.
<table>
<tr><th>Access</th><th>Sub-Field</th><th>Description</th></tr>
<tr><td rowspan="4">Read-Only</td><td>7-6
Reserved</td><td>Reserved. Always reads 0.</td></tr>
<tr><td>5-4
RSTGSM</td><td>Reset Gate Finite State Machine. Reads of the SEMA4_RSTGT register return the encoded state machine value. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as:
00 Idle, waiting for the first data pattern write.
01 Waiting for the second data pattern write.
10 The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. Note that the RSTGSM = 0b10 state is valid for only a single machine cycle, so it is impossible for a read to return this value
11 This state encoding is never used and therefore reserved.</td></tr>
<tr><td>3
Reserved</td><td>Reserved. Always reads 0.</td></tr>
<tr><td>2-0
RSTGMS</td><td>Reset Gate Bus Master. This 3-bit read-only field records the logical number of the bus master performing the gate reset function. The reset function requires that the two consecutive writes to this register be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs. The association between system bus master port numbers, the associated bus master device and the logical processor number is SoC-specific. See the chip configuration chapter for this information.</td></tr>
<tr><td>Write-Only</td><td>7-0
RSTGDP</td><td>Reset Gate Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the gate reset mechanism. For the first write, RSTGDP = 0xe2 while the second write requires RSTGDP = 0x1d.</td></tr>
</table></td></tr>
</table>

4.4.6.20 Semaphores (Secure) Reset IRQ Notification (SEMA4_RSTNTF)

As with the case of the secure reset function and the hardware gates, it is recognized that system operation may require a reset function to re-initialize the state of the IRQ notification logic without requiring a system-level reset.

To support this special notification reset requirement, the Semaphores module implements a "secure" reset mechanism which allows an IRQ notification (or all the notifications) to be initialized by following a specific dual-write access pattern. When successful, the specified IRQ notification state machine(s) are reset. Using a technique

similar to that required for the servicing of a software watchdog timer, the secure reset mechanism requires two consecutive writes with predefined data patterns from the same processor to force the clearing of the IRQ notification(s). The required access pattern is:

1. A processor performs a 16-bit write to the SEMA4_RSTNTF memory location. The most significant byte (SEMA4_RSTNTF[RSTNDP]) must be 0x47; the least significant byte is a "don't_care" for this reference.
2. The same processor then performs a second 16-bit write to the SEMA4_RSTNTF location. For this write, the upper byte (SEMA4_RSTNTF[RSTNDP]) is the logical complement of the first data pattern (0xb8) and the lower byte (SEMA4_RSTNTF[RSTNTN]) specifies the notification(s) to be reset. This field can specify a single notification be cleared, or that all notifications are cleared.
3. Reads of the SEMA4_RSTNTF location return information on the 2-bit state machine (SEMA4_RSTNTF[RSTNSM]) which implements this function, the bus master performing the reset (SEMA4_RSTNTF[RSTNMS]) and the notification number(s) last cleared (SEMA4_RSTNTF[RSTNTN]). Reads of the SEMA4_RSTNTF register do not affect the secure reset finite state machine in any manner.

Address: 30AC_0000h base + 104h offset = 30AC_0104h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	RSTNTN								RSTNSM_RSTNMS_RSTNDP							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SEMA4_RSTNTF field descriptions

Field	Description
15–8 RSTNTN	Reset Notification Number. This 8-bit field specifies the specific IRQ notification state machine to be reset. This field is updated by the second write. If RSTNTN < 64, then reset the single IRQ notification machine defined by RSTNTN, else reset all the notifications.
RSTNSM_ RSTNMS_ RSTNDP	**NOTE:** This field contains subfields that vary depending on whether it is being read or written. Sub-fields indicated as having read access are valid only for read operations. Sub-fields indicated as having write access are valid only for write operations. Bit numbering in the descriptions begins with the most significant bit numbered 0. See the following table for details.

Access	Sub-Field	Description
Read-Only	7-6 Reserved	Reserved. Always reads 0.
	5-4 RSTNSM	Reset Notification Finite State Machine. Reads of the SEMA4_RSTNTF register return the encoded state machine value. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as: 00 Idle, waiting for the first data pattern write. 01 Waiting for the second data pattern write.

Table continues on the next page...

SEMA4_RSTNTF field descriptions (continued)

Field	Description		
	Access	**Sub-Field**	**Description**
			10 The 2-write sequence has completed. Generate the specified notification reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. Note the RSTNSM = 10 state is valid for only a single machine cycle, so it is impossible for a read to return this value. 11 This state encoding is never used and therefore reserved..
		3 Reserved	Reserved. Always reads 0.
		2-0 RSTNMS	Reset Notification Bus Master. This 3-bit read-only field records the logical number of the bus master performing the notification reset function. The reset function requires that the two consecutive writes to this register be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs. The association between system bus master port numbers, the associated bus master device and the logical processor number is SoC-specific. See the chip configuration chapter for this information.
	Write-Only	7-0 RSTNDP	Reset Notification Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the notification reset mechanism. For the first write, RSTNDP = 0x47 while the second write requires RSTNDP = 0xb8.

4.5 On-Chip RAM Memory Controller (OCRAM)

4.5.1 Overview

There is 1 OCRAM controller implemented in i.MX 6UltraLite. One controller is for the 128KB on-chip RAM.

Various options are provided for adding a pipeline or wait-states in a read/write access, in order to ensure flexible timing control at both high and low frequencies.

The internal block diagram is shown in the figure below.

Figure 4-14. On-chip RAM Block Diagram

4.5.2 Basic Functions

4.5.2.1 Read/Write Arbitration

The detailed rules used in arbitration are as follows:

- If there is no granted read or write in the last cycle, and there is only a read request or a write request, the request will be granted.
- If there is no granted read or write in the last cycle, and there are both read or write requests coming in at the same time, the read request will be granted first.
- If a granted read/write transaction has just finished, the write/read request will have the higher priority in the next cycle.
- If the first read/write access request in a transaction is granted, all the data transfer in this burst will be finished before the next arbitration begins, that is, the round-robin arbitration mechanism is based on AXI transaction, not data access.

4.5.3 Advanced Features

This section describes some advanced features designed to avoid timing issues when the on-chip RAM is working at high frequency.

4.5.3.1 Read Data Wait State

When the wait state is enabled, it will take 2 cycles for each read access (each beat of a read burst).

This can avoid the potential timing problem caused by the longer memory access time at higher frequency.

When this feature is disabled, it only takes 1 clock cycle to finish a read transaction. That is, read data is available in the next cycle of read request becomes valid on the bus.

4.5.3.2 Read Address Pipeline

When this feature is enabled, the read address from the AXI master is delayed 1 cycle before it can be accepted by the on-chip RAM.

This can avoid setup time issues for the read access on the memory cell at high frequency. Enabling this feature can cost, at most, 1 more clock cycle for each AXI read transaction, that is, at most 1 more clock cycle for each read burst with multiple beats of data.

When this feature is disabled, the read address from the AXI master can be accepted by the on-chip RAM without delay, and data can become ready for master at next clock cycle (if no other access and no read data wait).

4.5.3.3 Write Data Pipeline

When this feature is enabled, the write data from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM.

This can avoid setup time issue for the write access on the memory cell at high frequency. Enabling this feature would cost at most 1 more clock cycle for each AXI write transaction, that is, at most 1 more clock cycle for each write burst with multiple beats of data.

When this feature is disabled, the write data from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write address is also ready at this cycle).

4.5.3.4 Write Address Pipeline

When this feature is enabled, the write address from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM.

This can avoid setup time issue for the write access on the memory cell at high frequency. Enabling this feature would take at most 1 more clock cycle for each AXI write transaction, that is, at most 1 more clock cycle for each write burst with multiple beats of data.

When this feature is disabled, the write address from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write data is also ready at this cycle).

4.5.4 Programmable Registers

There are no programmable registers in this block;

4.6 Network Interconnect Bus System (NIC-301)

4.6.1 Overview

This section provides an overview of the NIC-301 (Network Inter-Connect) AXI arbiter IP.

The NIC-301 (by ARM Ltd.) is a configurable AXI arbiter between several masters and slaves. The NIC-301 IP is designed so that many configuration options are selected at the hardware design stage, determined by SoC characteristics and needs, while several other configuration options are software-controlled.

This chapter covers in brief the NIC-301 functionality, while providing configuration details on the NIC-301 instances used in the chip. For complete details on the NIC-301 design, see the ARM specification, *AMBA® Network Interconnect (NIC-301) Technical Reference Manual, version r2p3*.

NOTE

The NIC-301 default settings are configured by Freescale's board support package (BSP), and in most cases should not be modified by the customer. The default settings have gone through exhaustive testing during the validation of the part, and have proven to work well for the part's intended target applications. Changes to the default settings may result in a degradation in system performance.

4.6.1.1 Block diagram

The NIC-301 AXI arbiter (or "CoreLink Network Interconnect") by ARM, provides configurable AXI-based interconnect logic, for connecting a number of masters (initiators) to several slaves (targets), via a configurable bus switches and bridging components.

The bus system is composed of

Each instance can include one or more bus switches, with additional logic.

This chapter provides details of the various instances and the selected configuration parameters.

The top level diagram of the bus system is shown in the following figure.

4.6.1.2 NIC-301 Main Features

Key features of the NIC-301 module include the following:

- Address space memory mapping, including 'remap' functions.
- Programmer's view, for software-configured parameters, via "GPV" ports.
- Support for cross-clock domain synchronization.

4.6.1.3 Modes and Operations

The NIC-301 supports a normal functional mode only.

4.6.2 External Signals

There are no external I/O interfaces for NIC-301.

4.6.3 Memory Map and Register Definition

This section includes the block memory map and detailed register descriptions.

Access to NIC-301 registers is provided through the global programmer's view (GPV) ports. Each GPV port provides access to the configuration registers of certain IP . The GPV base addresses are listed in the table below:

4.6.3.1 Memory Map

The NIC-301 memory map, is dependent on the selected configuration option at time of creation.

A "template" map is provided in Table 4-22 below.

4.6.3.2 Configuration programmers model

The GPV's contain configuration registers, partitioned into a number of individual 4KB blocks.

The general structure of the registers is provided by the following tables:

- Address map of the programmers model, Table 4-22
- AMIB Registers, Table 4-23
- ASIB Registers, Table 4-24

Table 4-22. Address map of the programmers model

Address Offset from Base Address	Registers	Notes
0x000F_F000	Internal interface p registers	Maximum p = 61
	...	Note [1]
0x000C_4000	Internal interface 2 registers	
0x000C_3000	Internal interface 1 registers	
0x000C_2000	Internal interface 0 registers	
0x000C_1000	Slave interface m registers	Maximum m = 127
	...	Note [2]
0x0004_4000	Slave interface 2 registers	
0x0004_3000	Slave interface 1 registers	
0x0004_2000	Slave interface 0 registers	
0x0004_1000	Master interface n registers	Maximum n = 63
	...	Note [3]
0x0000_4000	Master interface 2 registers	
0x0000_3000	Master interface 1 registers	
0x0000_2000	Master interface 0 registers	
0x0000_1000	ID registers	
0x0000_0000	Address control registers	Configurable base address [4]

1. Index refers to BI registers index
2. Index refer to ASIB registers index
3. Index refer to AMIB registers index
4. Reserved for internal use

4.6.3.2.1 Address control and ID registers

Registers at offsets 0x0–0xFFC are reserved for internal use.

4.6.3.2.2 AMBA master interface block (AMIB) configuration registers

The table below lists only the registers that affect the user. All other addresses are treated as "reserved". Non-implemented or reserved addresses inside NIC domains are read as ZEROS and writes operations are ignored.

Table 4-23. AMIB Registers

Offset	Register	Access	Width	Reset Value
0x024	fn_mod2	RW	1	0

Table continues on the next page...

Table 4-23. AMIB Registers (continued)

Offset	Register	Access	Width	Reset Value
	Bypass merge. This register is only present if upsizing or downsizing. See upsizing/downsizing data width functions in AMBA Network Interconnect TRM.			
0x040	wr_tidemark	RW	4	Note [1]

1. Reset value varies, default value chosen at RTL creation time is designed to suit normal operation.

4.6.3.2.3 ASIB (AMBA slave interface block) configuration registers

The table below lists only the registers that affect the user. All other addresses are treated as "reserved". Non-implemented or reserved addresses inside NIC domains are read as ZEROS and writes operations are ignored.

Table 4-24. ASIB Registers

Offset	Register	Access	Width	Reset Value
0x040	wr_tidemark Valid only for AXI slaves with WFIFO >=4.	RW	1	Note [1]
0x100	read_qos	RW	4	Note [2]
0x104	write_qos	RW	4	Note [2]

1. Reset value varies, default value chosen at RTL creation time is designed to suit typical operation cases.
2. QoS default is set at RTL creation time, and is listed in specific NIC-301 configuration tables in NIC-specific parameters, as parameters "QoS qv_value" in ASIB / AMIB parameter tables.

4.6.3.3 Register Descriptions

The NIC-301 registers are dependent upon the selected configuration, the type of ports, hardware-selected features and whether they have a GPV view. The addressing is associated to a specific port, by looking at the port's index number, under "apb_slave" column.

The memory map template is provided in the Configuration programmers model above.

4.7 AHB to IP Bridge (AIPSTZ)

4.7.1 Overview

This section provides an overview of the AHB to IP Bridge (AIPSTZ). This particular peripheral is designed as the bridge between AHB bus and peripherals with the lower bandwidth IP Slave (IPS) buses.

4.7.1.1 Features

The following list summarizes the key features of the bridge:

- The bridge supports the IPS slave bus signals.
- The bridge supports 8-, 16-, and 32-bit IPS peripherals. (Accesses larger than the size of a peripheral are not supported, except to 32-bit memory.)
- The bridge supports a pair of IPS accesses for 64-bit and certain misaligned AHB transfers to 32-bit memory in 64-bit platforms.
- The bridge directly supports up to 32 64-Kbyte external IPS peripherals, and 2 global external IPS peripheral spaces. The bridge occupies 1 MBytes of total address space.
- The bridge provides configurable per-block and per-master access protections. Access permissions are based on bus master (e.g. DMA or core) privilege levels and resource domain. More details on the protection features and configuration can be found in the Security Reference Manual
- Peripheral read transactions require a minimum of 2 hclk clocks, and unbuffered write transactions require a minimum of 3 hclk clocks.
- The bridge uses one single asynchronous reset and one global clock.

4.7.2 Clocks

The following table describes the clock sources for AIPSTZ. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 4-25. AIPSTZ Clocks

Clock name	Clock Root	Description
hclk	ahb_clk_root	Module clock

4.7.3 Functional Description

The AIPS bridge serves as a protocol translator between the AHB system bus and the IP bus.

Support is provided for generating a pair of 32-bit IP bus accesses when targeted by a 64-bit system bus access, or a misaligned access which crosses a 32-bit boundary. No other bus-sizing access support is provided.

The AHB to IP bridge is the interface between the AHB and on-chip IPS peripherals, which are sub-blocks containing readable/writable control and status registers.

The AHB master reads and writes these registers through the AIPSTZ. The bridge generates block enables, the block address, transfer attributes, byte enables and write data as inputs to the IPS peripherals. The bridge captures read data from the IPS interface and drives it on the AHB.

Each bridge that connects to the IPS (or peripherals) are referred as AIPS. The chip has three separate AIPS modules, and peripherals are grouped and assigned under each AIPS block. The list of peripherals are indicated as n-1, n-2, and n-3 for AIPS-1, AIPS-2, and AIPS-3 respectively.

AIPS occupies a 1-Mbyte portion of the address space. The register maps of the IPS peripherals are located on 64-Kbyte boundaries. Each IPS peripheral is allocated one 64-Kbyte block of the memory map, and is activated by one of the block enables from the bridge. Up to thirty-two 64-Kbyte external IPS peripherals may be implemented, occupying contiguous blocks of 64-Kbytes. Two global external IPS block enables are available for the remaining address space to allow for customization and expansion of addressed peripheral devices. In addition, a single "non-global" block enable is also asserted whenever any of the thirty-two non-global block enables is asserted.

The bridge is responsible for indicating to IPS peripherals if an access is in supervisor or user mode. It may block user mode accesses to certain IPS peripherals or it may allow the individual IPS peripherals to determine if user mode accesses are allowed. In addition, peripherals may be designated as write-protected.

The bridge supports the notion of "trusted" masters for security purposes. Masters may be individually designated as trusted for reads, trusted for writes, or trusted for both reads and writes, as well as being forced to look as though all accesses from a master are in user-mode privilege level. Refer to AIPSTZ Memory Map/Register Definition for more information.

The AIPSTZ prevents access to a peripheral if the transaction originated from a source from a resource domain that has been explicitly omitted. Resource domains are assigned in the RDC submodule. Please refer to the RDC chapter for programming details.

All peripheral devices are expected to only require aligned accesses equal to or smaller in size than the peripheral size. An exception to this rule is supported for 32-bit peripherals to allow memory to be placed on the IPS.

4.7.4 Access Protections

The AIPSTZ bridge provides programmable access protections for both masters and peripherals. It allows the privilege level of a master to be overridden, forcing it to user-mode privilege, and allows masters to be designated as trusted or untrusted.

Peripherals may require supervisor privilege level for access, may restrict access to a trusted master only, and may be write-protected. IP bus peripherals are subject to access control policies set in both CSU registers and AIPSTZ registers. An access is blocked if it is denied by either policy.

Masters and peripherals are assigned to one or more resource domains in the RDC submodule (see the RDC chapter for details). Depending on RDC programming, masters transactions through the AIPSTZ may or may not be allowed access to peripherals in different resource domains.

4.7.5 Access Support

Aligned 64-bit accesses, aligned and misaligned word and half word accesses, as well as byte accesses are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the IPS.

Peripheral registers must not be misaligned, although no explicit checking is performed by the AIPS bridge. The bridge will perform two IPS transfers for 64-bit accesses, word accesses with byte offsets of 1, 2, or 3, and for half word accesses with a byte offset of 3. All other accesses will be performed with a single IPS transfer.

Only aligned half word and byte accesses are supported for 16-bit peripherals. All other accesses types are unsupported, and results of such accesses are undefined. They are not terminated with an error response.

Only byte accesses are supported for 8-bit peripherals. All other accesses types are unsupported, and results of such accesses are undefined. They are not terminated with an error response.

4.7.6 Initialization Information

The AIPS bridge should be programmed before use.

The following registers should be initialized: The Master Privilege Registers (AIPSTZ_MPRs), the Peripheral Access Control registers (AIPSTZ_PACRs), and the Off-platform Peripheral Access Control registers (AIPSTZ_OPACRs) described in AIPSTZ Memory Map/Register Definition.

4.7.6.1 Security Block

The AIPSTZ contains a security block that is connected to each off-platform peripheral. This block filters accesses based on write/read, non-secure, and supervisor signals.

Each peripheral can be individually configured to allow or deny each of the following transactions as described in the table below:

Table 4-26. Peripheral Access Configuration options

Config Bit	Write	Non-Secure	Supervisor	Meaning
0	0	0	0	Secure User Read
1	0	0	1	Secure Supervisor Read
2	0	1	0	Non-Secure User Read
3	0	1	1	Non-Secure Supervisor Read
4	1	0	0	Secure User Write
5	1	0	1	Secure Supervisor Write
6	1	1	0	Non-Secure User Write
7	1	1	1	Non-Secure Supervisor Write

Each peripheral has a security configuration (sec_config_X) input for determining whether to allow or deny a given access type. These are 8-bit vectors, with each bit corresponding to one of the transactions above as listed in the Config Bit column of Table 4-26. If the bit is asserted (1'b1), the transaction is allowed. If the bit is negated (1'b0), the transaction is not allowed.

For example, if peripheral 0 is configured as follows:

sec_config_0 [7:0] = 8'b0011_0011

This peripheral can only be accessed by secure transactions. Bits 0, 1, 4, and 5 are asserted and these bits refer to the four types of secure transactions. If an insecure transaction is attempted to this peripheral, it will result in an error.

Eight bits per peripheral across an entire system can result in a large number of configuration bits that must be assigned and controlled, most likely in a series of registers in another block. To reduce the number of register bits required predefined sets of security profiles can be defined and encapsulated in an external security translation block. The table below describes one set of security profiles that has been proposed for use with the AIPSTZ.

Table 4-27. Security Levels

CSU_SEC_LEVEL	Non-Secure User	Non-Secure Supervisor	Secure User	Secure Supervisor
0	RD+WR	RD+WR	RD+WR	RD+WR
1	NOT ALLOWED	RD+WR	RD+WR	RD+WR
2	Read Only	Read Only	RD+WR	RD+WR
3	NOT ALLOWED	Read Only	RD+WR	RD+WR
4	NOT ALLOWED	NOT ALLOWED	RD+WR	RD+WR
5	NOT ALLOWED	NOT ALLOWED	NOT ALLOWED	RD+WR
6	NOT ALLOWED	NOT ALLOWED	Read Only	Read Only
7	NOT ALLOWED	NOT ALLOWED	NOT ALLOWED	NOT ALLOWED

Information regarding CSU is provided in the Security Reference Manual. Contact your Freescale representative for information about obtaining this document.

A 3-bit input, 8-bit output translation block can be used such that only three register bits are required to set the security profile and the translation block will drive the correct 8-bit configuration vector. Each peripheral connected to the AIPSTZ would require this translation block. The top level AIPSTZ has this three bit input line `csu_sec_level[2:0]' corresponding to each peripheral X.

4.7.7 AIPSTZ Memory Map/Register Definition

The memory map for the AIPS SW-visible registers is shown in the table below.

The MPROT and OPACR fields are 4 bits in width. Some bits may be reserved depending on device.

AIPSTZ memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Master Priviledge Registers (AIPSTZ_MPR)	32	R/W	7700_0000h	4.7.7.1/363

Table continues on the next page...

AIPSTZ memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
40	Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR)	32	R/W	4444_4444h	4.7.7.2/365
44	Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR1)	32	R/W	4444_4444h	4.7.7.3/369
48	Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR2)	32	R/W	4444_4444h	4.7.7.4/372
4C	Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR3)	32	R/W	4444_4444h	4.7.7.5/375
50	Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR4)	32	R/W	4444_4444h	4.7.7.6/378

4.7.7.1 Master Priviledge Registers (AIPSTZ_MPR)

Each AIPSTZ_MPR specifies 16 4-bit fields defining the access privilege level associated with a bus master in the platform, as well as specifying whether write accesses from this master are bufferable shown in Table 4-28

The registers provide one field per bus master, where field 15 corresponds to master 15, field 14 to master 14,... field 0 to master 0 (typically the processor core). The master index allocation is shown in Table 4-29.

Table 4-28. MPROT Field

Bit	Field	Description
3	MBW	**Master Buffer Writes** - This bit determines whether the AIPSTZ is enabled to buffer writes from this master.
2	MTR	**Master Trusted for Reads** - This bit determines whether the master is trusted for read accesses.
1	MTW	**Master Trusted for Writes** - This bit determines whether the master is trusted for write accesses.
0	MPL	**Master Privilege Level** - This bit determines how the privilege level of the master is determined.

NOTE

The reset value is set to 0000_0000_7700_0000, which makes master 0 and master 1 (ARM CORE) the trusted masters. Trusted software can change the settings after reset.

Table 4-29. Master Index Allocation

Master Index	Master Name	Comments
Master 0	All masters excluding ARM core	Share the same number allocation.
Master 1	ARM A7 CORE	
Master 3	SDMA	

Address: 0h base + 0h offset = 0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MPROT0				MPROT1				MPROT2				MPROT3				Reserved				MPROT5				Reserved							
Reset	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AIPSTZ_MPR field descriptions

Field	Description
31–28 MPROT0	Master 0 Priviledge, Buffer, Read, Write Control xxx0 **MPL** — Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. xxx1 **MPL** — Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access. xx0x **MTW** — This master is not trusted for write accesses. xx1x **MTW** — This master is trusted for write accesses. x0xx **MTR** — This master is not trusted for read accesses. x1xx **MTR** — This master is trusted for read accesses. 0xxx **MBW** — Write accesses from this master are not bufferable 1xxx **MBW** — Write accesses from this master are allowed to be buffered
27–24 MPROT1	Master 1 Priviledge, Buffer, Read, Write Control xxx0 **MPL** — Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. xxx1 **MPL** — Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access. xx0x **MTW** — This master is not trusted for write accesses. xx1x **MTW** — This master is trusted for write accesses. x0xx **MTR** — This master is not trusted for read accesses. x1xx **MTR** — This master is trusted for read accesses. 0xxx **MBW** — Write accesses from this master are not bufferable 1xxx **MBW** — Write accesses from this master are allowed to be buffered
23–20 MPROT2	Master 2 Priviledge, Buffer, Read, Write Control xxx0 **MPL** — Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. xxx1 **MPL** — Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access. xx0x **MTW** — This master is not trusted for write accesses. xx1x **MTW** — This master is trusted for write accesses. x0xx **MTR** — This master is not trusted for read accesses. x1xx **MTR** — This master is trusted for read accesses.

Table continues on the next page...

AIPSTZ_MPR field descriptions (continued)

Field	Description
	0xxx **MBW** — Write accesses from this master are not bufferable 1xxx **MBW** — Write accesses from this master are allowed to be buffered
19–16 MPROT3	Master 3 Priviledge, Buffer, Read, Write Control. xxx0 **MPL** — Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. xxx1 **MPL** — Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access. xx0x **MTW** — This master is not trusted for write accesses. xx1x **MTW** — This master is trusted for write accesses. x0xx **MTR** — This master is not trusted for read accesses. x1xx **MTR** — This master is trusted for read accesses. 0xxx **MBW** — Write accesses from this master are not bufferable 1xxx **MBW** — Write accesses from this master are allowed to be buffered
15–12 -	This field is reserved. Reserved
11–8 MPROT5	Master 5 Priviledge, Buffer, Read, Write Control. xxx0 **MPL** — Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. xxx1 **MPL** — Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access. xx0x **MTW** — This master is not trusted for write accesses. xx1x **MTW** — This master is trusted for write accesses. x0xx **MTR** — This master is not trusted for read accesses. x1xx **MTR** — This master is trusted for read accesses. 0xxx **MBW** — Write accesses from this master are not bufferable 1xxx **MBW** — Write accesses from this master are allowed to be buffered
-	This field is reserved. Reserved

4.7.7.2 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 4-30

Table 4-30. OPAC Field

Bit	Field	Description
3	BW	**Buffer Writes** - This bit determines whether write accesses to this peripheral are allowed to be buffered.[1]

Table continues on the next page...

Table 4-30. OPAC Field (continued)

Bit	Field	Description
2	SP	**Supervisor Protect** - This bit determines whether the peripheral requires supervisor privilege level for access.
1	WP	**Write Protect** - This bit determines whether the peripheral allows write accesses.
0	TP	**Trusted Protect** - This bit determines whether the peripheral allows accesses from an untrusted master.

1. Buffered writes are not available for AIPSTZ. This bit should be set to '0'.

Address: 0h base + 40h offset = 40h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAC0				OPAC1				OPAC2				OPAC3				OPAC4				OPAC5				OPAC6				OPAC7			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPSTZ_OPACR field descriptions

Field	Description
31–28 OPAC0	Off-platform Peripheral Access Control 0 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
27–24 OPAC1	Off-platform Peripheral Access Control 1 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

Table continues on the next page...

AIPSTZ_OPACR field descriptions (continued)

Field	Description
23–20 OPAC2	Off-platform Peripheral Access Control 2 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
19–16 OPAC3	Off-platform Peripheral Access Control 3 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
15–12 OPAC4	Off-platform Peripheral Access Control 4 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
11–8 OPAC5	Off-platform Peripheral Access Control 5

Table continues on the next page...

AIPSTZ_OPACR field descriptions (continued)

Field	Description
	xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
7–4 OPAC6	Off-platform Peripheral Access Control 6 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
OPAC7	Off-platform Peripheral Access Control 7 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

4.7.7.3 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR1)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 4-30

Address: 0h base + 44h offset = 44h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAC8				OPAC9				OPAC10				OPAC11				OPAC12				OPAC13				OPAC14				OPAC15			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPSTZ_OPACR1 field descriptions

Field	Description
31–28 OPAC8	Off-platform Peripheral Access Control 8 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
27–24 OPAC9	Off-platform Peripheral Access Control 9 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

Table continues on the next page...

AIPSTZ_OPACR1 field descriptions (continued)

Field	Description
23–20 OPAC10	Off-platform Peripheral Access Control 10 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
19–16 OPAC11	Off-platform Peripheral Access Control 11 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
15–12 OPAC12	Off-platform Peripheral Access Control 12 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
11–8 OPAC13	Off-platform Peripheral Access Control 13

Table continues on the next page...

AIPSTZ_OPACR1 field descriptions (continued)

Field	Description
	xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
7–4 OPAC14	Off-platform Peripheral Access Control 14 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
OPAC15	Off-platform Peripheral Access Control 15 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

4.7.7.4 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR2)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 4-30

Address: 0h base + 48h offset = 48h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAC16				OPAC17				OPAC18				OPAC19				OPAC20				OPAC21				OPAC22				OPAC23			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPSTZ_OPACR2 field descriptions

Field	Description
31–28 OPAC16	Off-platform Peripheral Access Control 16 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
27–24 OPAC17	Off-platform Peripheral Access Control 17 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

Table continues on the next page...

AIPSTZ_OPACR2 field descriptions (continued)

Field	Description
23–20 OPAC18	Off-platform Peripheral Access Control 18 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
19–16 OPAC19	Off-platform Peripheral Access Control 19 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
15–12 OPAC20	Off-platform Peripheral Access Control 20 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
11–8 OPAC21	Off-platform Peripheral Access Control 21

Table continues on the next page...

AIPSTZ_OPACR2 field descriptions (continued)

Field	Description
	xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
7–4 OPAC22	Off-platform Peripheral Access Control 22 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
OPAC23	Off-platform Peripheral Access Control 23 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

4.7.7.5 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR3)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 4-30

Address: 0h base + 4Ch offset = 4Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAC24				OPAC25				OPAC26				OPAC27				OPAC28				OPAC29				OPAC30				OPAC31			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPSTZ_OPACR3 field descriptions

Field	Description
31–28 OPAC24	Off-platform Peripheral Access Control 24 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
27–24 OPAC25	Off-platform Peripheral Access Control 25 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

Table continues on the next page...

AIPSTZ_OPACR3 field descriptions (continued)

Field	Description
23–20 OPAC26	Off-platform Peripheral Access Control 26 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
19–16 OPAC27	Off-platform Peripheral Access Control 27 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
15–12 OPAC28	Off-platform Peripheral Access Control 28 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
11–8 OPAC29	Off-platform Peripheral Access Control 29

Table continues on the next page...

AIPSTZ_OPACR3 field descriptions (continued)

Field	Description
	xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
7–4 OPAC30	Off-platform Peripheral Access Control 30 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
OPAC31	Off-platform Peripheral Access Control 31 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

4.7.7.6 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACR4)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 4-30

Address: 0h base + 50h offset = 50h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAC32				OPAC33				Reserved																							
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPSTZ_OPACR4 field descriptions

Field	Description
31–28 OPAC32	Off-platform Peripheral Access Control 32 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
27–24 OPAC33	Off-platform Peripheral Access Control 33 xxx0 **TP** — Accesses from an untrusted master are allowed. xxx1 **TP** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP** — This peripheral allows write accesses. xx1x **WP** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

Table continues on the next page...

AIPSTZ_OPACR4 field descriptions (continued)

Field	Description
-	This field is reserved. Reserved

4.8 Shared Peripheral Bus Arbiter (SPBA)

4.8.1 Overview

The Shared Peripheral Bus Arbiter (SPBA) is a three-to-one IP Bus interface arbiter. Three masters arbitrate for shared peripheral access through the SPBA.

The SPBA has three primary functions:

- The IP Bus Line switches a master to one peripheral
- The Masters arbiter arbitrates between the three masters to solve concurrent access or restricted access to peripherals
- The Control Registers and Ownership Control includes a set of registers which are reachable through software and permit the access scheme to be defined for each peripheral (Resource Ownership and Access Control). It generates signals for the external steering logic of interrupts and DMA signals.

The figure below shows the SPBA block diagram

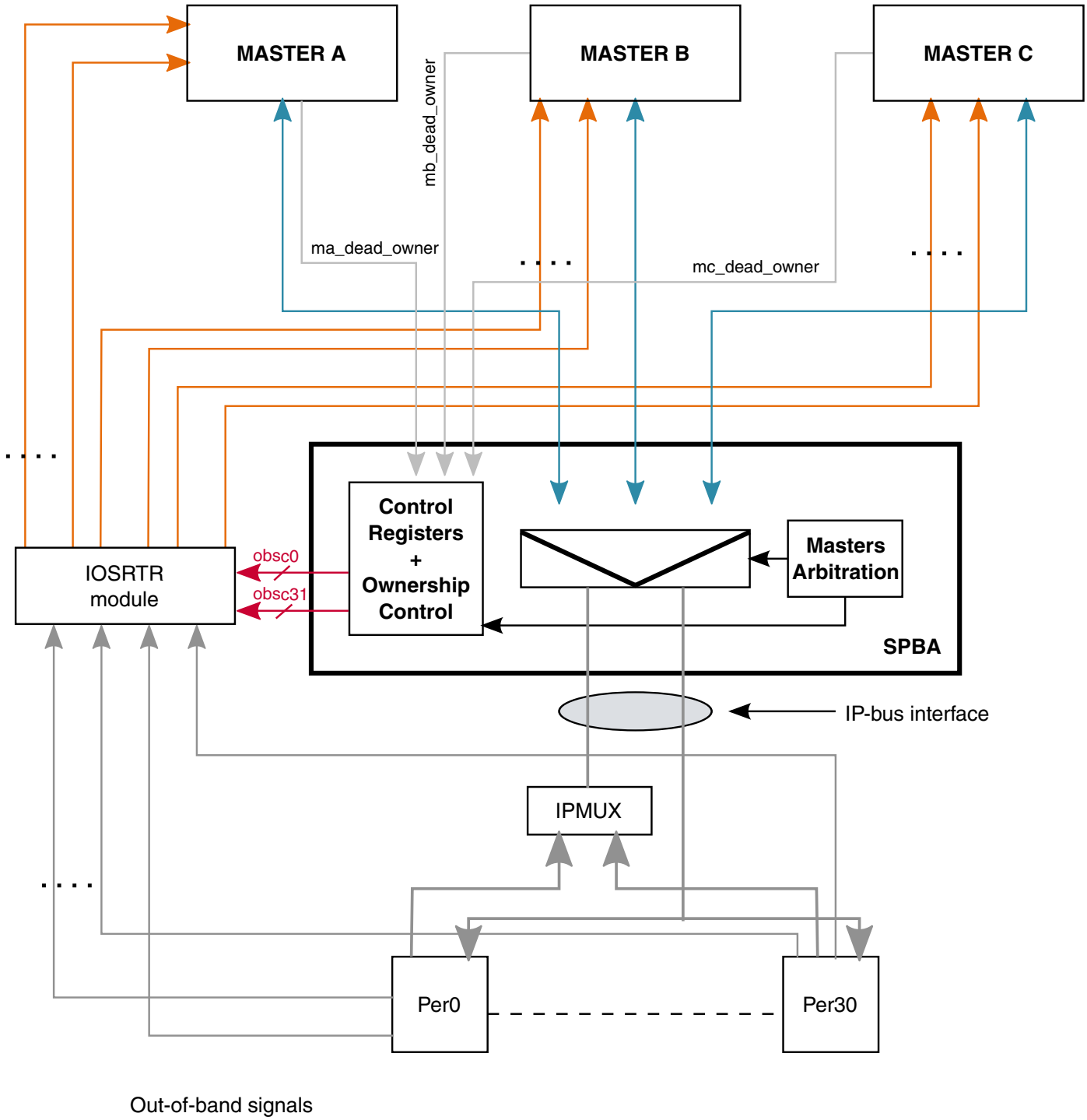


Figure 4-15. SPBA Block Diagram

4.8.1.1 Features

The SPBA includes the following features:

- Three IP Bus masters arbitration: Master A, B and C
- Support for DMA masters
- 32-bit data
- Supports up to 31 shared peripherals, each consuming 64 kilobytes of address space
- SPBA can be considered the 32nd peripheral, used for resource ownership and access control of the 31 peripherals
- Provides 31 sets of out of band steering control (OBSC) signals to the off-block steering logic
- Operating frequency up to 67 MHz
- Clocks: ipg_clk, ipg_clk_s

4.8.1.2 Modes of operation

SPBA behavior is transparent when accessing a peripheral, though it has these distinct modes of operation.

Reset/Abort

The SPBA has a hardware reset which initializes all registers, arbitration and peripherals rights registers (PRRs).

An abort signal input is provided allowing each master to abort its current access and release ownership (in case of master reset sequence).

Functional

Once a master request is granted, its IP Bus signals are steered to the requested peripheral.

Standby

No clock needed. The SPBA needs clocks only during access to the PRRs, arbitration, and abort phases. It generates two clock enable signals indicating when the clocks must be provided.

Configuration

During this phase, a master accesses the SPBA PRRs. The SPBA memory-mapped registers are seen as a shared peripheral.

4.8.2 Clocks

The table found here describes the clock sources for SPBA.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 4-31. SPBA Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

4.8.3 Functional description

4.8.3.1 Masters arbitration

The arbitration mechanism determines which port will control the master port, based on a simple round-robin arbitration scheme.

There are several use cases to consider.

- Only one master request per access. The master is switched to the shared peripheral bus, without arbitration. Figure 4-16 shows the MB request on the global module enable signal, served without wait state.
- If two masters simultaneously access SPBA, the last granted master is held off using the <master>_ips_xfr_wait output signal (default value is high). When the master is granted sips_xfr_wait, shared IP Bus peripheral is connected to <master>_ips_xfr_wait outputs.
- If three masters simultaneously access SPBA, then the last two granted masters are held off using <master>_ips_xfr_wait. Figure 4-17 shows a case in which the last two accesses granted are MA and MB. The requests are used even if they are in the same cycle.
- If after reset, at the first multiple access, no master has been granted, the priority is static: Master A (MA), Master B (MB) and last Master C (MC) port.
- No master request. No master switch to shared peripherals.

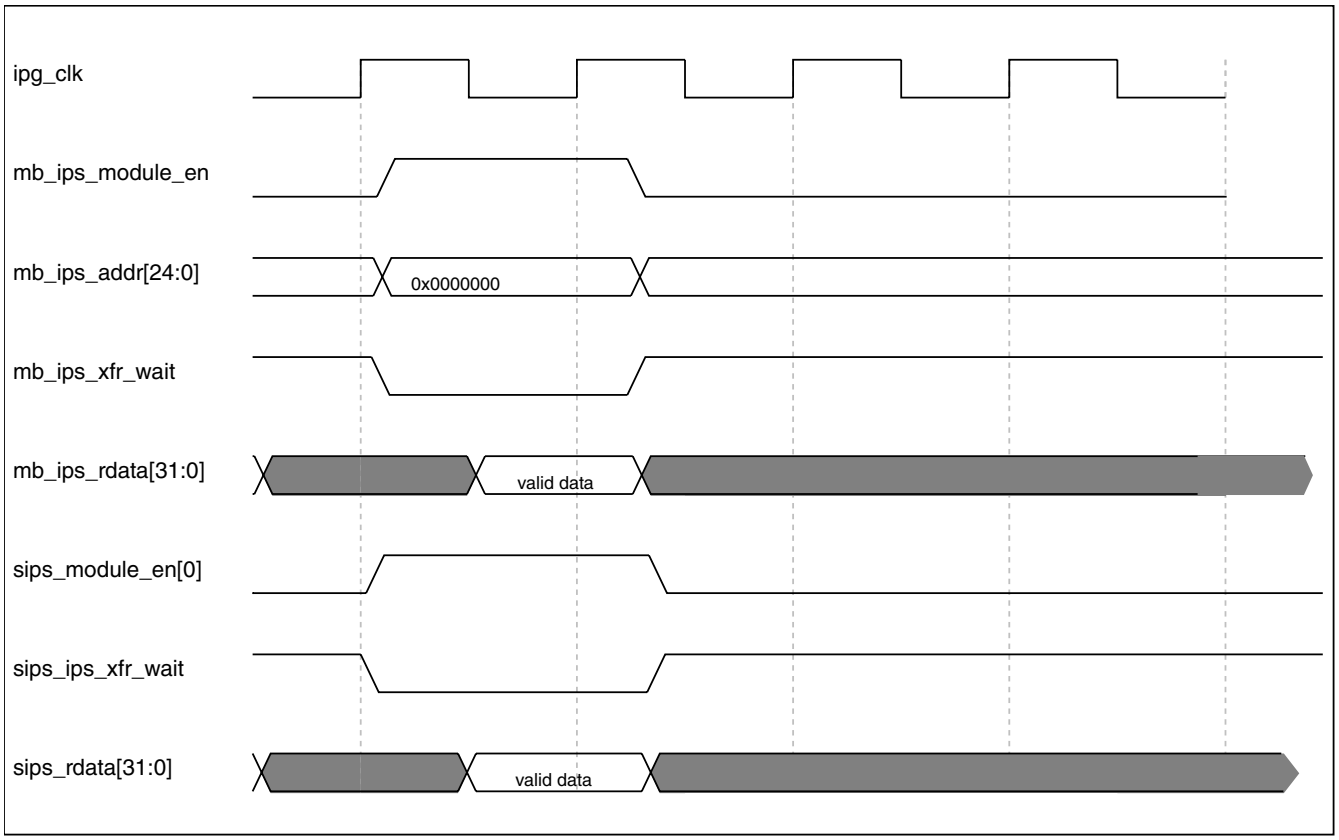


Figure 4-16. Example of one master request, no SPBA arbitration

The following figure assumes MA and MB have been the last two masters granted in the previous transfers (MA then MB).

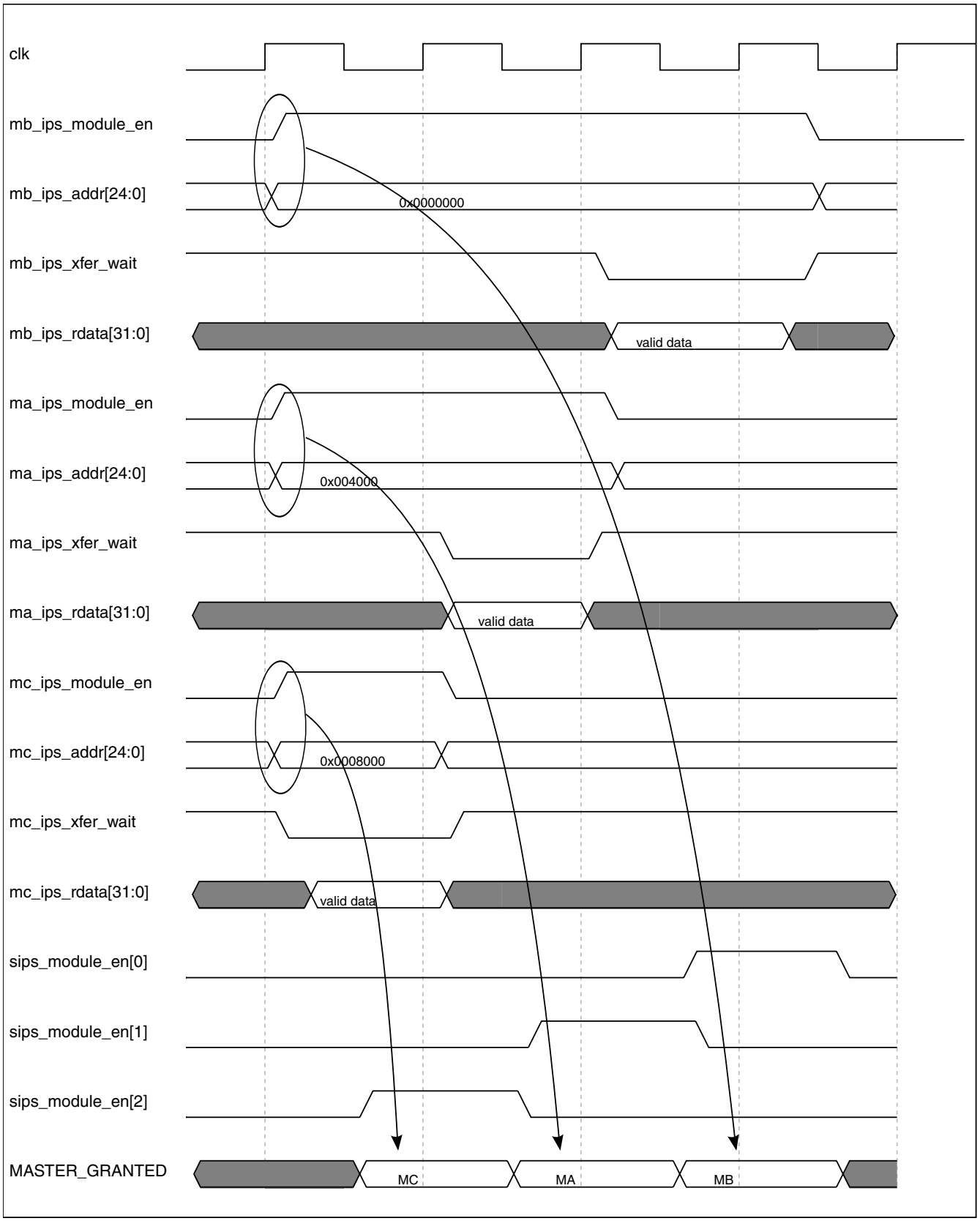


Figure 4-17. Example of three master requests: Masters already granted are "waited";

4.8.4 Resource ownership control

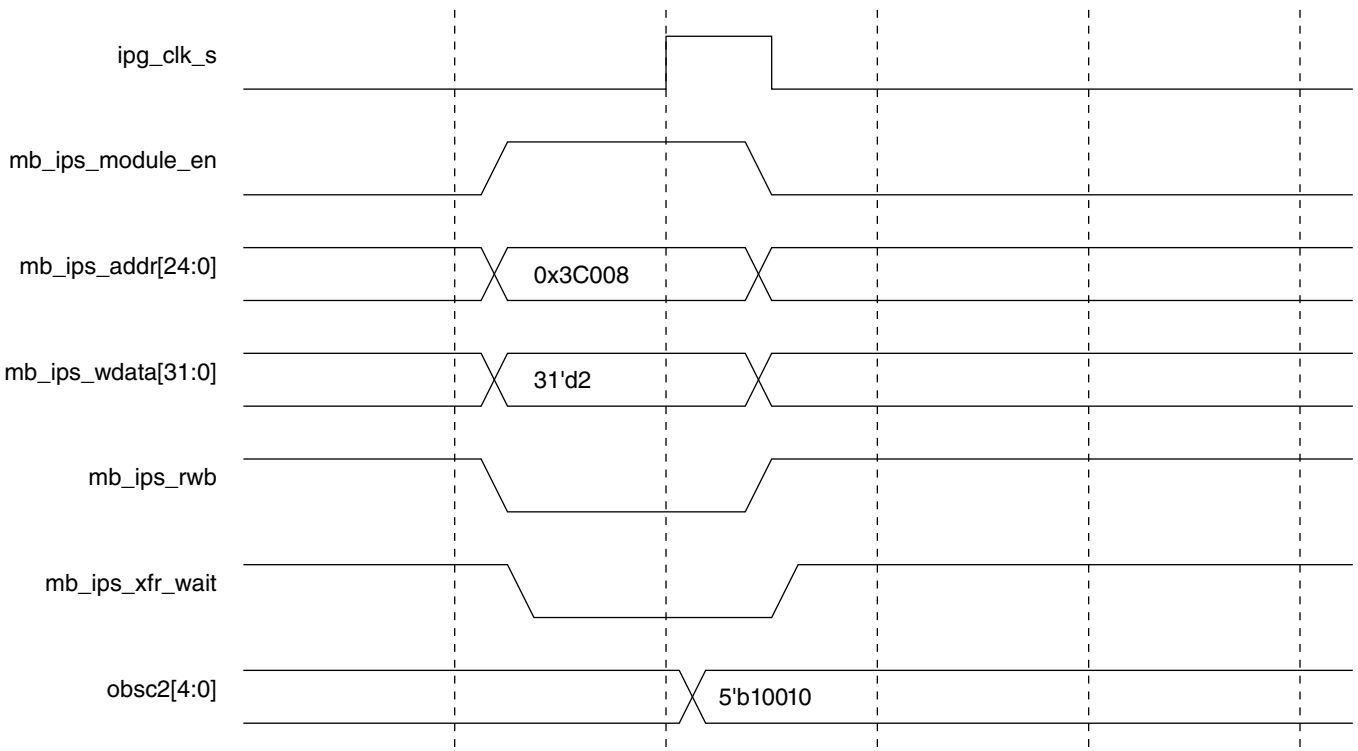
The resource ownership control regulates access to the shared peripherals and determines the steering of out-of-band signals.

4.8.4.1 Access control

4.8.4.1.1 Peripheral access

The peripheral access (resource access) of the requesting master is given by the corresponding RAR bit of the Peripheral Right Register. It determines if the master has access privilege to the resource.

Any attempt at access made by a requesting master whose access privilege bit is not set (in the PRR) is terminated with a bus error (**<master>_ips_xfr_err** is asserted by SPBA logic). The master that owns the resource can lock the peripheral for itself and/or grant other masters access to the peripheral by setting the appropriate bit(s) in the RAR field.



Master B is taking ownership of peripheral 2 by writing 3'b010 in the SPBA peripheral 2 right register (rarfield)
This ownership can be checked on obsc2 output as roi2[1:0] = 2'b10 and rar2[2:0] = 3'b010
(obsc[4:0] = {roi2[1], roi2[0], rar2[2], rar2[1], rar2[0]})

Figure 4-18. Example of one master B gaining ownership of peripheral 2

4.8.4.1.2 Peripheral Right Register access

The ROI bits of the Peripheral Right Register (PRR) determine which master is allowed to make write access to PRR. The identification of the requesting master is compared to the ROI bits of the PRR to determine if the master has ownership of the corresponding register.

Any attempted write access to a PRR already owned by another master will be ignored.

4.8.4.2 Owner election

When the peripheral is not owned by any master (ROI="00", after coming out of reset for instance), the first master to perform successfully a write to the RAR bits of the PRR is granted ownership of the peripheral and its associated PRR.

After writing to the PRR (RAR bit(s)), the master must read it back to make sure that it was granted ownership. If the RMO field is 2'b11, then the ownership claim is successful. If RMO is 2'b10, another master claimed ownership before this master was able to complete its write. This resolves the case in which two or more masters attempt to write the PRR at the same time; only the first master will be granted ownership. However all masters must read the PRR to determine if this case occurred, and if so, whether they were the first master which was granted ownership.

NOTE

A master that has been granted ownership of the PRR does not automatically have the right access to the peripheral; it must still set its own RAR bits in the PRR to access the peripheral.

4.8.4.3 Ending ownership

Ownership may be voluntarily ended by the owning master, or automatically upon assertion of a master-specific dead_owner signal.

The former is appropriate for software-controlled yielding of ownership. The latter is appropriate for automatic yielding of ownership when the owner has gone into reset.

When a master is reset, it clears the ROI bits of the PRRs owned by the corresponding master. When the owner is dead (in reset), all peripherals previously owned by that master must be changed to the un-owned state.

NOTE

It is the programmer's responsibility to make sure the peripherals are placed in an appropriate state before ending ownership.

4.8.4.3.1 Software Controlled Ownership Ending

The ROI bits will be automatically cleared when the master that owns the PRR access right clears (write) the RAR bits (Table 2).

It will then end the ownership of the PRR.

4.8.4.4 The Un-owned State

During the time when the peripheral is un-owned (i.e the ROI field contains all 0's), all masters have full access to it (RAR bits can then be modified by a master if ROI[1:0] = 2'b0).

In such cases it is necessary for software to ensure any necessary coherency in the resource, there is no hardware protection.

4.8.5 SPBA Memory Map/Register Definition

The SPBA control registers (Peripheral Right Registers) are mapped as a virtual shared peripheral.

SPBA can support up to 31 shared peripherals. Each of them has its own Peripheral Right Register (PRR) accessible within the SPBA memory-mapped registers, and consists of the Requesting Master Owner, the Resource Owner ID and the Resource Access Right fields.

SPBA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
308F_0000	Peripheral Rights Register (SPBA_PRR0)	32	R/W	0000_0007h	4.8.5.1/390
308F_0004	Peripheral Rights Register (SPBA_PRR1)	32	R/W	0000_0007h	4.8.5.1/390
308F_0008	Peripheral Rights Register (SPBA_PRR2)	32	R/W	0000_0007h	4.8.5.1/390
308F_000C	Peripheral Rights Register (SPBA_PRR3)	32	R/W	0000_0007h	4.8.5.1/390
308F_0010	Peripheral Rights Register (SPBA_PRR4)	32	R/W	0000_0007h	4.8.5.1/390
308F_0014	Peripheral Rights Register (SPBA_PRR5)	32	R/W	0000_0007h	4.8.5.1/390
308F_0018	Peripheral Rights Register (SPBA_PRR6)	32	R/W	0000_0007h	4.8.5.1/390
308F_001C	Peripheral Rights Register (SPBA_PRR7)	32	R/W	0000_0007h	4.8.5.1/390
308F_0020	Peripheral Rights Register (SPBA_PRR8)	32	R/W	0000_0007h	4.8.5.1/390
308F_0024	Peripheral Rights Register (SPBA_PRR9)	32	R/W	0000_0007h	4.8.5.1/390
308F_0028	Peripheral Rights Register (SPBA_PRR10)	32	R/W	0000_0007h	4.8.5.1/390
308F_002C	Peripheral Rights Register (SPBA_PRR11)	32	R/W	0000_0007h	4.8.5.1/390
308F_0030	Peripheral Rights Register (SPBA_PRR12)	32	R/W	0000_0007h	4.8.5.1/390
308F_0034	Peripheral Rights Register (SPBA_PRR13)	32	R/W	0000_0007h	4.8.5.1/390
308F_0038	Peripheral Rights Register (SPBA_PRR14)	32	R/W	0000_0007h	4.8.5.1/390
308F_003C	Peripheral Rights Register (SPBA_PRR15)	32	R/W	0000_0007h	4.8.5.1/390

Table continues on the next page...

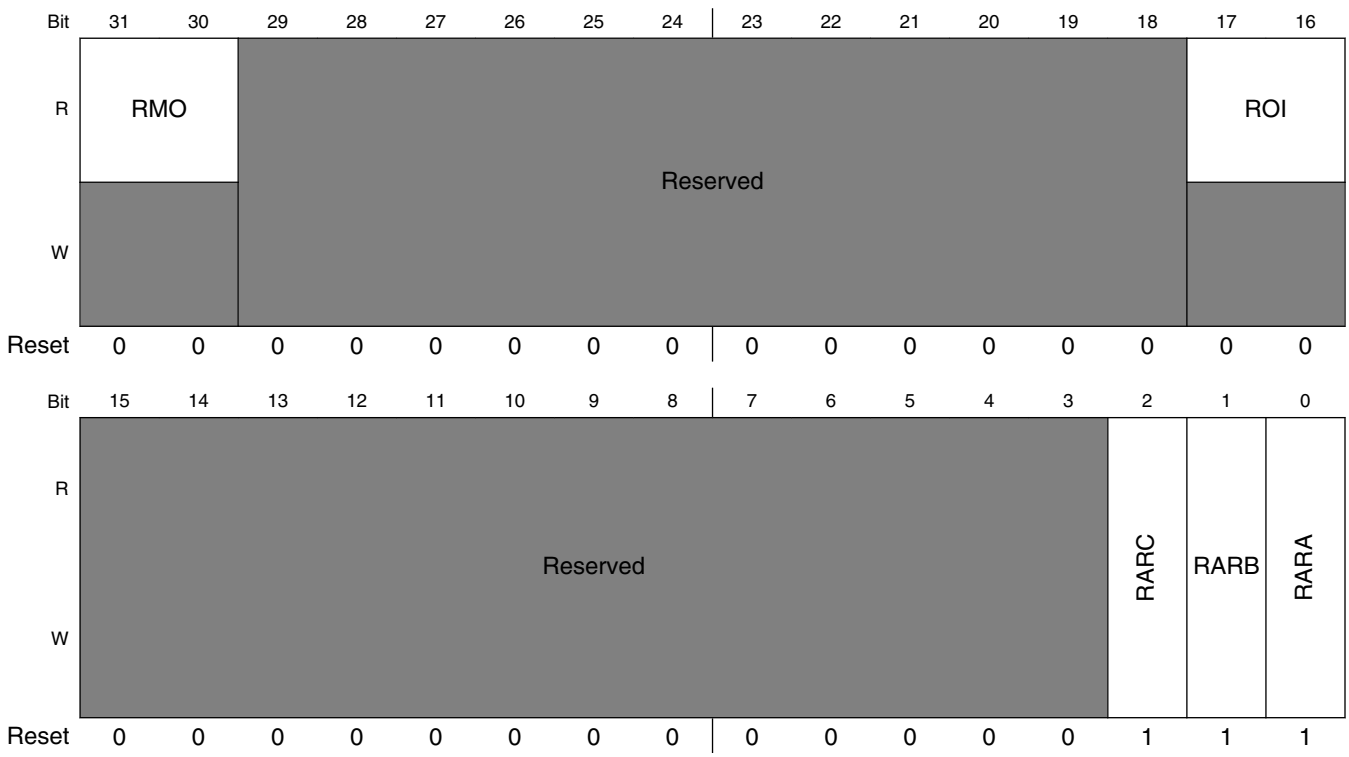
SPBA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
308F_0040	Peripheral Rights Register (SPBA_PRR16)	32	R/W	0000_0007h	4.8.5.1/390
308F_0044	Peripheral Rights Register (SPBA_PRR17)	32	R/W	0000_0007h	4.8.5.1/390
308F_0048	Peripheral Rights Register (SPBA_PRR18)	32	R/W	0000_0007h	4.8.5.1/390
308F_004C	Peripheral Rights Register (SPBA_PRR19)	32	R/W	0000_0007h	4.8.5.1/390
308F_0050	Peripheral Rights Register (SPBA_PRR20)	32	R/W	0000_0007h	4.8.5.1/390
308F_0054	Peripheral Rights Register (SPBA_PRR21)	32	R/W	0000_0007h	4.8.5.1/390
308F_0058	Peripheral Rights Register (SPBA_PRR22)	32	R/W	0000_0007h	4.8.5.1/390
308F_005C	Peripheral Rights Register (SPBA_PRR23)	32	R/W	0000_0007h	4.8.5.1/390
308F_0060	Peripheral Rights Register (SPBA_PRR24)	32	R/W	0000_0007h	4.8.5.1/390
308F_0064	Peripheral Rights Register (SPBA_PRR25)	32	R/W	0000_0007h	4.8.5.1/390
308F_0068	Peripheral Rights Register (SPBA_PRR26)	32	R/W	0000_0007h	4.8.5.1/390
308F_006C	Peripheral Rights Register (SPBA_PRR27)	32	R/W	0000_0007h	4.8.5.1/390
308F_0070	Peripheral Rights Register (SPBA_PRR28)	32	R/W	0000_0007h	4.8.5.1/390
308F_0074	Peripheral Rights Register (SPBA_PRR29)	32	R/W	0000_0007h	4.8.5.1/390
308F_0078	Peripheral Rights Register (SPBA_PRR30)	32	R/W	0000_0007h	4.8.5.1/390
308F_007C	Peripheral Rights Register (SPBA_PRR31)	32	R/W	0000_0007h	4.8.5.1/390

4.8.5.1 Peripheral Rights Register (SPBA_PRR*n*)

This register controls master ownership and access for a peripheral.

Address: 308F_0000h base + 0h offset + (4d × i), where i=0d to 31d



SPBA_PRR*n* field descriptions

Field	Description
31–30 RMO	Requesting Master Owner. This 2-bit register field indicates if the corresponding resource is owned by the requesting master or not. This register is reset to 2'b0 if ROI = 2'b0. 00 **UNOWNED** — The resource is unowned. 01 Reserved. 10 **ANOTHER_MASTER** — The resource is owned by another master. 11 **REQUESTING_MASTER** — The resource is owned by the requesting master.
29–18 -	This field is reserved. Reserved
17–16 ROI	Resource Owner ID. This field indicates which master (one at a time) can access to the PRR for rights modification. This is a read-only register. After reset, ROI bits are cleared ("00" -> un-owned resource).

Table continues on the next page...

SPBA_PRR*n* field descriptions (continued)

Field	Description
	A master performing a write access to the an un-owned PRR will get its ID automatically written into ROI, while modifying RARx bits. It can then read back the RMO, RAR, ROI bits to make sure RMO returns the right value, ROI bits contain its ID and RARx bits are correctly asserted. Then no other master (whom ID is different from the one stored in ROI) will be able to modify RAR fields. Owner master of a peripheral can assert its dead_owner signal, or write 1'b0 in the RARx to release the ownership (ROI[1:0] reset to 2'b0). 00 **UNOWNED** — Unowned resource. 01 **MASTER_A** — The resource is owned by master A port. 10 **MASTER_B** — The resource is owned by master B port. 11 **MASTER_C** — The resource is owned by master C port.
15–3 -	This field is reserved. Reserved
2 RARC	Resource Access Right. Control and Status bit for master C. This field indicates whether master C can access the peripheral. From 0 up to 3 masters can have permission to access a resource (all the master can be granted on a peripheral, but only one access at a time will be granted by SPBA). 0 **PROHIBITED** — Access to peripheral is not allowed. 1 **ALLOWED** — Access to peripheral is granted.
1 RARB	Resource Access Right. Control and Status bit for master B. This field indicates whether master B can access the peripheral. From 0 up to 3 masters can have permission to access a resource (all the master can be granted on a peripheral, but only one access at a time will be granted by SPBA). 0 **PROHIBITED** — Access to peripheral is not allowed. 1 **ALLOWED** — Access to peripheral is granted.
0 RARA	Resource Access Right. Control and Status bit for master A. This field indicates whether master A can access the peripheral. From 0 up to 3 masters can have permission to access a resource (all the master can be granted on a peripheral, but only one access at a time will be granted by SPBA). 0 **PROHIBITED** — Access to peripheral is not allowed. 1 **ALLOWED** — Access to peripheral is granted.

4.9 ROM Controller with Patch (ROMCP)

4.9.1 Overview

The Read Only Memory Controller with ROM Patch (ROMC) acts as an interface between the ARM advanced high-performance bus (AHB - Lite) and the Read Only Memory. The ROMC consists of a ROM Controller and a ROM Patch. The ROM Patch

is used to either patch code routines or fix data tables in the ROM area. There is an IP Bus interface to access the ROM Patch Registers . The figure below depicts the main functional sub-blocks of the ROMC.

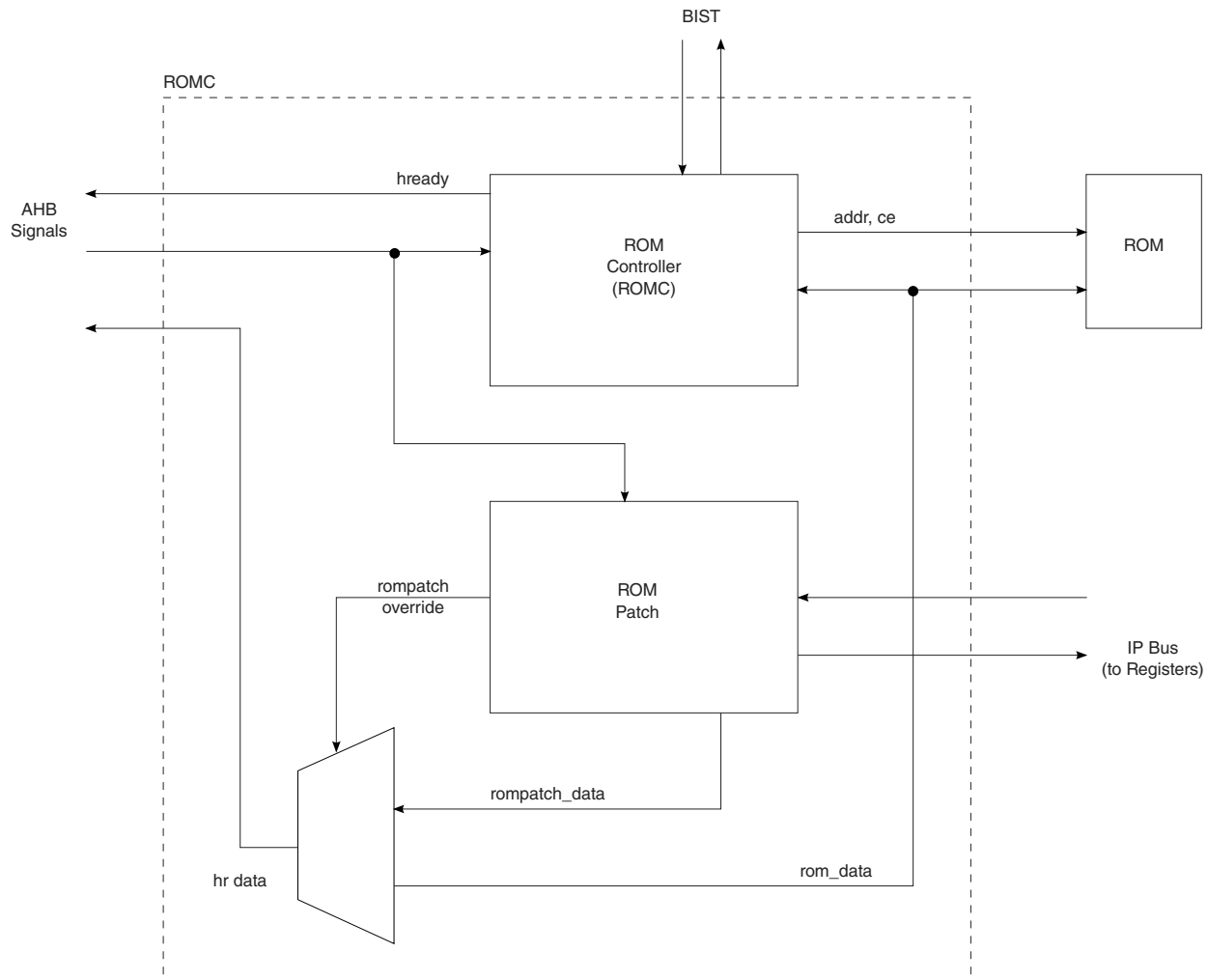


Figure 4-19. ROMC Block Diagram

4.9.1.1 Features

- Supports ROM size ranges from 16 Kbyte up to 4 Mbyte with increments of 1 Kbyte
- Supports opcode patching for a maximum of 16 different addresses in 4 Mbytes of ROM space
- Supports one-word data fixes for a max of 8 memory locations in 4 Mbytes of ROM space
- Supports patching of the Reset Vector (at 0x0000_0000) to allow external booting

4.9.1.2 Modes of Operation

There are two modes of operation: normal mode and BIST mode. In normal mode, the ROMC ensures correct reads from the ROM, assuming the memory complies with the characteristics and requirements for which the ROMC was designed.

4.9.1.2.1 Low Power Mode

There are two clock enables that are used to switch off parts of the ROMC logic when inactive. The first clock enable is used to disable the ROM Controller when the master connected to the AHB interface is not initiating a read to the ROM. The second clock enable is used to disable the registers used to program the ROM patch feature when the registers are not being accessed.

4.9.2 Clocks

The table found here describes the clock sources for ROMCP.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 4-32. ROMCP Clocks

Clock name	Clock Root	Description
hclk	ahb_clk_root	System / bus clock
hclk_reg	ipg_clk_root	System access clock
96krom_CLK	ahb_clk_root	ROM clock

4.9.3 Memory Map

4.9.3.1 ROM Memory Map in detail

The ROMC supports ROM sizes with a range of 16 Kbyte to 4 Mbyte with an increment of 1 Kbyte. The 16 Kbyte lower limit was chosen because the minimum size of security code on an ARM platform is approximately 16 Kbyte of code, which is only accessible in supervisor mode. Note that it is the MMU that controls whether any region of memory is secure.

The exception vectors must be secured as well, and must be put in the same area as the security code. Since they must reside at address 0x0000_0000, the entire 16 Kbyte of ROM which can only be accessible in supervisor mode is located at the very beginning of the platform memory map.

If the user chooses not to use the security code, a memory size smaller than 16 Kbyte can be connected to the platform (minimum of 1 Kbyte). The MMU can be programmed to allow any kind of access into this memory. However, if the ROM size is less than 16 Kbyte, memory aliasing will occur for all invalid addresses greater than the memory size but within the 16 Kbyte of space.

For ROM sizes bigger than 16 Kbyte, the rest of its physical size resides at the address starting at 0x0040_4000 (4 M+16 Kbyte) going up to [0x0040_4000 + (mem. size - 16Kbyte)]. .

4.9.4 Functional Description

This section is divided up into the ROM Controller Functional Description and the ROMC functional description.

4.9.4.1 ROM Controller (ROMC) Functional Description

4.9.4.1.1 Functionality overview

The ROMC serves two main functions. First, as an interface between the AHB-Lite bus on an ARM platform and the ROM. Second, it drives and receives several signals for the BIST engine. In normal mode of operation, the ROMC monitors the AHB-Lite for memory access requests and performs the memory operation to the ROM.

The ROMC includes the option to wait state all accesses from either the ARM or non-ARM masters to ROM in the event that timing requirements will not allow single HCLK clock cycle reads. If a wait state is required, the static inputs rom_wait_arm or rom_wait_alt_mstr can be set to 1 and accesses will take two HCLK clock cycles. If wait states are not required, rom_wait_arm or rom_wait_alt_mstr can be set to 0 and accesses will take one HCLK clock cycle to complete.

4.9.4.2 ROMC Functional Description

4.9.4.2.1 ROMC Disabling

All the bits in the ROMC_ROMPATCHENL register are cleared on Reset, disabling all the address comparators. Once the comparators have been enabled, the ROMC functions of data fixing and opcode patching can be quickly disabled by setting the DIS bit in the ROMC_ROMPATCHCNTL register. This bit is used to enable secure operations in which patching functions need to be disabled. This bit is cleared on Reset.

4.9.4.2.2 ROMC Event Priority

The ROMC has a total of 16 address comparators. The first 8 (0 through 7) comparators can be programmed for the data fixing function (through the 8 data fix enable bits in the ROMC_ROMPATCHCNTL register) while the rest are for opcode patching by default. This allows for potential multiple matching events involving both data fixing and opcode patch types. In these cases the ROMC assigns the highest priority to a data fixing event.

For example, if the ROMC is set up to data fix a certain address with comparator 4 and also opcode patch the same address with comparator 7, it will let comparator 4 have higher priority in indicating a match, and data from ROMC_ROMPATCHD4 will be put on the rompatch_romc_hrdata bus as the override value.

If multiple address matches of the same type level occur concurrently, then the ROMC will choose the source number based on the one with the highest source number. For example, the ROMC is setup to data fix the same location with address comparators 4 and 7, then address comparator 7 will have higher priority in indicating a match, and the value from ROMC_ROMPATCHD7 will be put on the rompatch_romc_hrdata bus as the orverride value. The same priority applies for an opcode patch event, except the override data is in the form of an SWI instruction with the comment field set to the source number with the highest priority.

4.9.4.2.3 Data Fixing

The data fixing feature allows ROM data to be updated by direct replacement when it is being read. This data usually originates from data tables, but can include ARM instructions. To enable data fixing on a certain address, this address value is written in to one of the first eight (0 through 7) of ROMC_ROMPATCHAxx registers and the same numbered bit set in the ROMC_ROMPATCHENL and ROMC_ROMPATCHCNTL registers. The data to be used for replacement is placed in the corresponding ROMC_ROMPATCHDxx.

The ROMC looks for a read access to ROM (either code fetch or data load) by snooping the AHB interface for read transactions. The address is compared with the values stored in the ROMC_ROMPATCHAxx[22:2] registers. If a match occurs from one of the

comparators, the ROMC places the value in the corresponding ROMC_ROMPATCHDxx register on the read data bus by overriding the read data coming from the actual ROM (see the mux in Figure 4-19). The value on the read data bus is maintained until hready is asserted to terminate the access. In data fixing, the entire word is replaced so if a byte or half-word access occurs on a "data fix" location, the entire data word is replaced. The word being replaced is word aligned. (The two LSBs of the matching ROMC_ROMPATCHAxx are ignored in the data fix operation.)

4.9.4.2.4 Opcode Patching

The opcode patch feature provides the ARM core a mechanism to fetch updated versions of code routines that were originally programmed in ROM. This patching mechanism makes use of the SWI (software interrupt instruction) and a table of function pointers residing in writable memory. The opcode being patched is replaced with a SWI instruction by the ROMC. Subsequent processing of the SWI reads from a function pointers table to obtain the address of the replacement code. Execution resumes with this code patch.

To enable opcode patching of a certain address, this address value is written into one of the ROMPATCHAxx registers and the corresponding bit set in the ROMPATCHENL to enable the associated comparator. The register's LSB (ROMC_ROMPATCHxx[0]) should be set if THUMB mode patching is in effect for this address. The ROMC identifies a ROM read access by snooping the AHB interface. The address is compared with the values stored in the ROMC_ROMPATCHAxx[22:2] registers. If a match occurs from one of the comparators, the ROMC generates the opcode of a software interrupt (SWI) instruction with the comment field containing the number of the matching address comparator. This opcode and comment is placed on the read data bus until hready is asserted by the ROM controller to terminate the read access.

The type of SWI generated, (that is, either ARM or THUMB), is determined by the LSB of the ROMC_ROMPATCHAxx register associated with the opcode patch. This bit is cleared for ARM mode (32 bits). The ROMC generates a 32-bit SWI (opcode field is 0xEF, occupying bits [31:24] of the word), with the least significant 5 bits of the 24-bit comment field (bits [23:0]) containing the number of the matching address comparator. The rest of the comment field is filled with zeros. This means that the ROMC will use 16 of the 16777216 possible software interrupts. The ROMC overrides the read data from the ROM.

If the LSB of the matching ROMC_ROMPATCHAxx register is set, the opcode patch is in THUMB mode (16 bits or half word). The ROMC generates a 16-bit SWI instruction (opcode field is 0xDF, occupying bits [15:8] of the half word) with the least significant 5 bits of the 8-bit comment field containing with the source number of the address comparator. The rest of the comments field is filled with zeros. This means that the

ROMC will use 16 of the 256 possible software interrupts. The ROMC puts this 16 bit SWI instruction value on the proper half of the rompatch_romc_hrdata bus. The other half is zeroed out. Which half of the bus contains the SWI opcode and comment depends on the mode (Big Endian or Little Endian) and the bit 1 of the matching ROMC_ROMPATCHAxx register. In Little Endian mode, the lower half is bits {15:0] and the upper half is bits {31:16]. The order is reversed in Big Endian mode.

In Little Endian mode (bigend signal negated), if bit 1 of the matching ROMC_ROMPATCHAxx is cleared (lower half word selected) then the SWI instruction is put on the lower 16 bits of the read data bus and the upper 16 bits are zeroed out. Only the lower 16 bits of the read data bus is overwritten by the ROMC data. If ROMC_ROMPATCHAxx[1] is set (upper half word selected), the SWI instruction is put on the upper 16 bits of the read data bus and the lower 16 bits are zeroed out. Only the upper 16 bits of the read data bus is overwritten.

In Big Endian mode (bigend asserted), if bit 1 of the matching ROMC_ROMPATCHAxx is cleared (lower half word selected) then the SWI instruction is put on the upper 16 bits of the read data bus while the lower 16 bits are zeroed out. Only the upper 16 bits of the read data bus is overwritten. If ROMC_ROMPATCHAxx[1] is set (upper word selected), the SWI instruction is put on the lower 16 bits and the upper 16 bits are zeroed out. Only the lower 16 bits of the read data bus is overwritten.

The eventual execution of the SWI causes the ARM to save the CPSR in SPSR_SVC, the address of the next instruction after the SWI in R14_SVC, enter Supervisor mode, and fetch the SWI vector at 0x8, which then takes it to a handler for further processing as described in the next section.

4.9.4.2.4.1 Typical Software Response to Opcode Patch

When the SWI handler executes it needs to determine whether the SWI was generated by the ROMC. This is done by loading the SWI instruction and extracting its comment field. The state of the ARM core (ARM or THUMB) when the SWI was executed dictates whether to load the instruction word (ARM) or half word (THUMB). This state information can be determined by testing the T bit (bit 5) of the SPSR. If it's set, the execution was in THUMB mode.

By convention, if the comment field of the SWI is greater than 16, the software interrupt was initiated by software (i.e. an operating system call), and a branch is taken to the appropriate handler routine for further processing. If the comment field is less than 16, the SWI was generated by the ROMC performing a code patch operation. In this case, the software then reads from a table of function pointers, using the value in the SWI

comment field as the index into the table. The value that is read is the address of the code patch. This value is loaded into the PC to begin the execution of the code patch. The following code segment illustrates a typical handling of the SWI.

```
stmfd        sp!, {r0-r1,lr}       @ push register onto SWI stack
mrs         r0, spsr        @ get saved status register
tst          r0, #0x20     @ check if call was in THUMB mode
ldrneh      r0, [lr,#-2]       @ yes: load opcode half-word and
bicne       r0, r0, #0xff00   @ yes: extract THUMB comment
ldreq       r0, [lr,#-4]       @ no: load opcode word and
biceq       r0, r0, #0xff000000 @ no: extract ARM comment
                    @ now r0 has comment field
cmp        r0, #16      @ compare to 16 (maximum for ROMC)
ldrlt         lr, =rompatch_tbl_ptr @ < 16: get top of current ROMC
                    @ table; global variable which is
                    @ changeable per context
ldrlt         r1, [lr, r0, lsl #2]   @ < 16: read function pointer from
                    @ table assumed an array of pointers
                    @ patch functions
strlt          r1, [sp, #8]     @ < 16: store function pointer onto
                    @ stack in position of link register
ldmltfd       sp!, {r0-r1,pc}^      @ < 16: "fake" return from SWI, will
                    @ vector core to appropriate patch
                    @ function and set core back to previous
                    @ mode of operating
ldr          r1, =swi_hdler   @ >= 16: pointer to standard SWI
                    @ handler
mov         lr, pc         @ >= 16: set link register
bx          r1           @ >= 16: jump to standard SWI
                    @ handler
ldmfd        sp!, {r0-r1,pc}^      @ >= 16: pop registers from stack
```

4.9.4.2.5 External Boot Feature

Following a Reset event, the ARM issues an instruction fetch of the Reset Vector from address 0x0. This instruction, normally residing in ROM is usually a branch to a Reset handler or boot code which also normally resides in ROM. The ROMC external boot feature allows the bypassing of this code, using a different boot code residing perhaps in external memory.

This feature uses the data fix mechanism and works as follows: if the boot_int signal is negated when a Reset event occurred, the ROMC will perform a data fix of the Reset Vector at 0x0 with the following instruction (opcode 0xE59FF00C):

```
ldr          pc,[pc,#12]      @ read 0x0000_0014 for reset_vector
```

The value of PC when this instruction is executed is 8 so that a PC relative offset of 12 makes the source address 20 or 0x14. When this instruction executes, the ARM core reads from address 0x0000_0014, triggering a ROMC data fix operation which places the value taken from the external boot address on the read data bus, with the two LSBs zeroed out. This value is returned to the ARM to be placed in the PC causing code fetch and execution to start from that address.

4.9.4.2.6 Alternate Masters and ROMC

The ROMC sits on the AHB bus of the internal ROM (ROMC). This means that the ROMC can modify values on the read data bus going to the master. Therefore, any master which reads an opcode patched or data patched location will read patched data.

4.9.5 ROMCP Memory Map/Register Definition

All registers are accessible through an IP Bus and can only be accessed in privileged mode. These registers can only be written with 32-bits stores and are clocked by hclk_reg.

The ROMC register placement was originated from the AWPT design used in the ARM7 platform of Neptune

ROMC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3031_00D4	ROMC Data Registers (ROMC_ROMPATCH0D)	32	R/W	0000_0000h	4.9.5.1/400
3031_00D8	ROMC Data Registers (ROMC_ROMPATCH1D)	32	R/W	0000_0000h	4.9.5.1/400
3031_00DC	ROMC Data Registers (ROMC_ROMPATCH2D)	32	R/W	0000_0000h	4.9.5.1/400
3031_00E0	ROMC Data Registers (ROMC_ROMPATCH3D)	32	R/W	0000_0000h	4.9.5.1/400
3031_00E4	ROMC Data Registers (ROMC_ROMPATCH4D)	32	R/W	0000_0000h	4.9.5.1/400
3031_00E8	ROMC Data Registers (ROMC_ROMPATCH5D)	32	R/W	0000_0000h	4.9.5.1/400
3031_00EC	ROMC Data Registers (ROMC_ROMPATCH6D)	32	R/W	0000_0000h	4.9.5.1/400
3031_00F0	ROMC Data Registers (ROMC_ROMPATCH7D)	32	R/W	0000_0000h	4.9.5.1/400
3031_00F4	ROMC Control Register (ROMC_ROMPATCHCNTL)	32	R/W	0840_0000h	4.9.5.2/401
3031_00F8	ROMC Enable Register High (ROMC_ROMPATCHENH)	32	R	0000_0000h	4.9.5.3/402
3031_00FC	ROMC Enable Register Low (ROMC_ROMPATCHENL)	32	R/W	0000_0000h	4.9.5.4/402
3031_0100	ROMC Address Registers (ROMC_ROMPATCH0A)	32	R/W	0000_0000h	4.9.5.5/403
3031_0104	ROMC Address Registers (ROMC_ROMPATCH1A)	32	R/W	0000_0000h	4.9.5.5/403
3031_0108	ROMC Address Registers (ROMC_ROMPATCH2A)	32	R/W	0000_0000h	4.9.5.5/403
3031_010C	ROMC Address Registers (ROMC_ROMPATCH3A)	32	R/W	0000_0000h	4.9.5.5/403
3031_0110	ROMC Address Registers (ROMC_ROMPATCH4A)	32	R/W	0000_0000h	4.9.5.5/403
3031_0114	ROMC Address Registers (ROMC_ROMPATCH5A)	32	R/W	0000_0000h	4.9.5.5/403
3031_0118	ROMC Address Registers (ROMC_ROMPATCH6A)	32	R/W	0000_0000h	4.9.5.5/403
3031_011C	ROMC Address Registers (ROMC_ROMPATCH7A)	32	R/W	0000_0000h	4.9.5.5/403
3031_0120	ROMC Address Registers (ROMC_ROMPATCH8A)	32	R/W	0000_0000h	4.9.5.5/403
3031_0124	ROMC Address Registers (ROMC_ROMPATCH9A)	32	R/W	0000_0000h	4.9.5.5/403

Table continues on the next page...

ROMC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3031_0128	ROMC Address Registers (ROMC_ROMPATCH10A)	32	R/W	0000_0000h	4.9.5.5/403
3031_012C	ROMC Address Registers (ROMC_ROMPATCH11A)	32	R/W	0000_0000h	4.9.5.5/403
3031_0130	ROMC Address Registers (ROMC_ROMPATCH12A)	32	R/W	0000_0000h	4.9.5.5/403
3031_0134	ROMC Address Registers (ROMC_ROMPATCH13A)	32	R/W	0000_0000h	4.9.5.5/403
3031_0138	ROMC Address Registers (ROMC_ROMPATCH14A)	32	R/W	0000_0000h	4.9.5.5/403
3031_013C	ROMC Address Registers (ROMC_ROMPATCH15A)	32	R/W	0000_0000h	4.9.5.5/403
3031_0208	ROMC Status Register (ROMC_ROMPATCHSR)	32	w1c	0000_0000h	4.9.5.6/404

4.9.5.1 ROMC Data Registers (ROMC_ROMPATCH*n*D)

The ROMC data registers (ROMC_ROMPATCH7D through ROMC_ROMPATCH0D) store the data to use for the 8 1-word data fix events. Each register is associated with an address comparator (7 through 0). When a data fixing event occurs, the value in the data register corresponding to the comparator that has the address match is put on the romc_hrdata[31:0] bus until romc_hready is asserted by the ROM controller to terminate the access. A MUX external to the ROMC will select this data over that of romc_hrdata[31:0] in returning read data to the ARM core. The selection is done with the control bus rompatch_romc_hrdata_ovr[1:0] with both bits asserted by the ROMC.

If more than one address comparators match, the highest-numbered one takes precedence, and the value in corresponding data register is used for the patching event.

Address: 3031_0000h base + D4h offset + (4d × i), where i=0d to 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATAX																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ROMC_ROMPATCH*n*D field descriptions

Field	Description
DATAX	Data Fix Registers - Stores the data used for 1-word data fix operations. The values stored within these registers do not affect the writes to the memory system. They are selected over the read data from ROM when a data fix event occurs. If any part of the 1-word data fix is read, then the entire word is replaced. Therefore, a byte or half-word read will cause the ROMC to replace the entire word. The word is word address aligned.

4.9.5.2 ROMC Control Register (ROMC_ROMPATCHCNTL)

The ROMC control register (ROMC_ROMPATCHCNTL) contains the block disable bit and the data fix enable bits. The block disable bit provides a means to disable the ROMC data fix and opcode patching functions, even when the address comparators are enabled. The External Boot feature is not affected by this bit. The eight data fix enable bits (0 through 7), when set, assign the associated address comparators to data fix operations

NOTE

Bits 27 and 22 always read as 1s.

Address: 3031_0000h base + F4h offset = 3031_00F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		DIS	Reserved												
Reset	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								DATAFIX							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ROMC_ROMPATCHCNTL field descriptions

Field	Description
31–30 -	This field is reserved. Reserved
29 DIS	ROMC Disable -- This bit, when set, disables all ROMC operations. This bit is used to enable secure operations. 0 Does not affect any ROMC functions (default) 1 Disable all ROMC functions: data fixing, and opcode patching
28–8 -	This field is reserved. Reserved
DATAFIX	**Data Fix Enable - Controls the use of the first 8 address comparators for 1-word data fix or for code patch routine.** 0 Address comparator triggers a opcode patch 1 Address comparator triggers a data fix

4.9.5.3 ROMC Enable Register High (ROMC_ROMPATCHENH)

The ROMC enable register high (ROMC_ROMPATCHENH) and ROMC enable register low (ROMC_ROMPATCHENL) control whether or not the associated address comparator can trigger a opcode patch or data fix event. This implementation of the ROMC only has 16 comparators, therefore ROMC_ROMPATCHENH and the upper half of ROMC_OMPATCHENL are read-only. ROMC_ROMPATCHENL[15:0] are associated with comparators 15 through 0. ROMC_ROMPATCHENLH[31:0] would have been associated with comparators 63 through 32.

Address: 3031_0000h base + F8h offset = 3031_00F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ROMC_ROMPATCHENH field descriptions

Field	Description
-	This field is reserved. Reserved

4.9.5.4 ROMC Enable Register Low (ROMC_ROMPATCHENL)

The ROMC enable register high (ROMC_ROMPATCHENH) and ROMC enable register low (ROMC_ROMPATCHENL) control whether or not the associated address comparator can trigger a opcode patch or data fix event. This implementation of the ROMC only has 16 comparators, therefore ROMC_ROMPATCHENH and the upper half of ROMC_ROMPATCHENL are read-only. ROMC_ROMPATCHENL[15:0] are associated with comparators 15 through 0. ROMC_ROMPATCHENLH[31:0] would have been associated with comparators 63 through 32.

Address: 3031_0000h base + FCh offset = 3031_00FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																ENABLE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

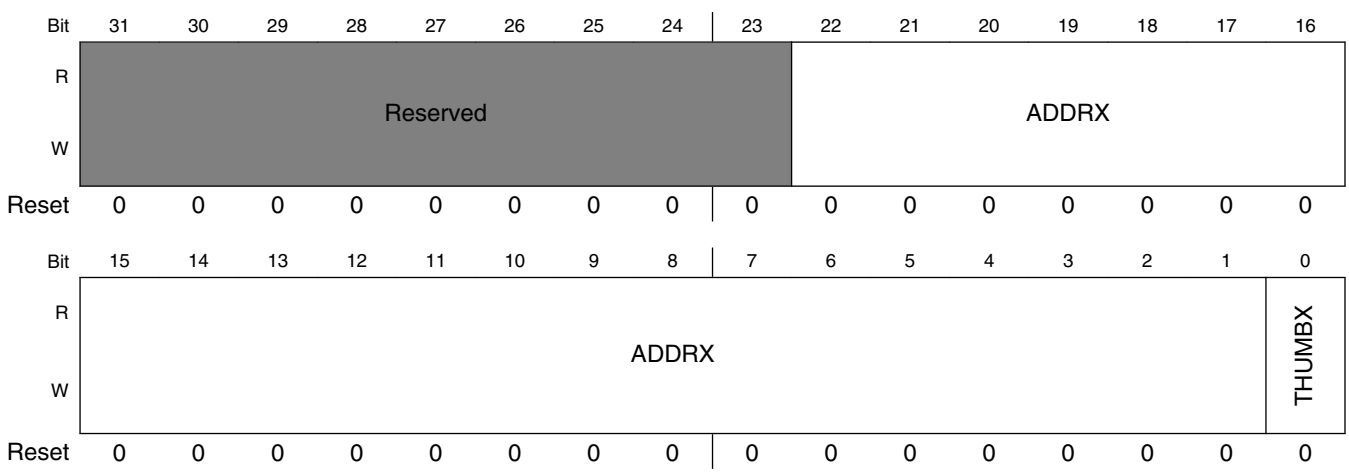
ROMC_ROMPATCHENL field descriptions

Field	Description
31–16 Reserved	This field is reserved.
ENABLE	**Enable** Address **Comparator -** This bit enables the corresponding address comparator to trigger an event. 0 Address comparator disabled 1 Address comparator enabled, ROMC will trigger a opcode patch or data fix event upon matching of the associated address

4.9.5.5 ROMC Address Registers (ROMC_ROMPATCH*n*A)

The ROMC address registers (ROMC_ROMPATCHA0 through ROMC_ROMPATCHA15) store the memory addresses where opcode patching begins and data fixing occurs. The address registers ROMC_ROMPATCHA0 through ROMC_ROMPATCHA15 are each 21 bits wide and dedicated to one 4 Mbyte memory space. Bits 21 through 2 are address bits, to be compared with romc_haddr[21:2] for a match; bit 1 is also an address bit used for half word selection. Bit 0 is the mode bit (set to 1 for THUMB mode). 1-word data fixing can only be used on the first 8 of the address comparators. ROMC_ROMPATCHA0 through ROMC_ROMPATCHA15 are associated each with address comparators 0 through 15.

Address: 3031_0000h base + 100h offset + (4d × i), where i=0d to 15d



ROMC_ROMPATCH*n*A field descriptions

Field	Description
31–23 -	This field is reserved. Reserved

Table continues on the next page...

ROMC_ROMPATCH*n*A field descriptions (continued)

Field	Description
22–1 ADDRX	Address Comparator Registers - Indicates the memory address to be watched. All 16 registers can be used for code patch address comparison. Only the first 8 registers can be used for a 1-word data fix address comparison. Bit 1 is ignored if data fix. Only used in code patch
0 THUMBX	THUMB Comparator Select - Indicates that this address will trigger a THUMB opcode patch or an ARM opcode patch. If this watchpoint is selected to be a data fix, then this bit is ignored as all data fixes are 1-word data fixes. 0 ARM patch 1 THUMB patch (ignore if data fix)

4.9.5.6 ROMC Status Register (ROMC_ROMPATCHSR)

The ROMC status register (ROMC_ROMPATCHSR) indicates the current state of the ROMC and the source number of the most recent address comparator event.

Address: 3031_0000h base + 208h offset = 3031_0208h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved														SW	Reserved
W															w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved										SOURCE					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ROMC_ROMPATCHSR field descriptions

Field	Description
31–18 -	This field is reserved. Reserved
17 SW	ROMC AHB Multiple Address Comparator matches Indicator - Indicates that multiple address comparator matches occurred. Writing a 1 to this bit will clear this it. 0 no event or comparator collisions 1 a collision has occurred
16–6 -	This field is reserved. Reserved
SOURCE	ROMC Source Number - Binary encoding of the number of the address comparator which has an address match in the most recent patch event on ROMC AHB. If multiple matches occurred, the highest priority source number is used.

Table continues on the next page...

ROMC_ROMPATCHSR field descriptions (continued)

Field	Description
	0 Address Comparator 0 matched 1 Address Comparator 1 matched 15 Address Comparator 15 matched

4.10 TrustZone Address Space Controller (TZASC)

4.10.1 Overview

The TrustZone Address Space Controller (TZASC) protects security-sensitive SW and data in a trusted execution environment against potentially compromised SW running on the platform.

The TZASC block diagram is shown in figure below.

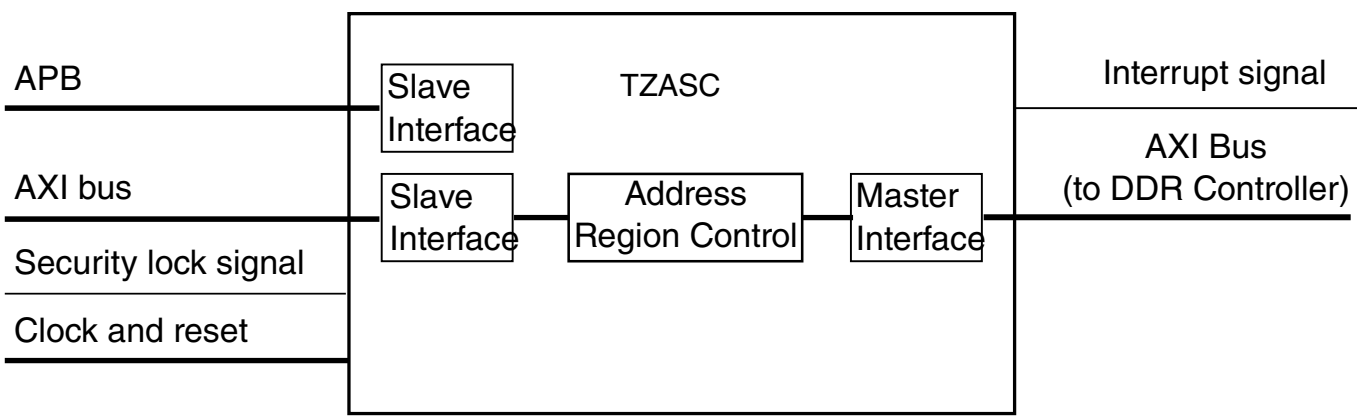


Figure 4-20. TZASC Block Diagram

The TZASC is an IP by ARM ("CoreLink™ TrustZone Address Space Controller TZC-380"), designed to provide configurable protection over program (SW) memory space.

The main features of TZASC are:

- Supports 16 independent address regions
- Access controls are independently programmable for each address region
- Sensitive registers may be locked
- Host interrupt may be programmed to signal attempted access control violations
- AXI master/slave interfaces for transactions
- APB slave interface for configuration and status reporting

4.10.2 Clocks

The table found here describes the clock sources for TZASC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 4-33. TZASC Clocks

Clock name	Clock Root	Description
aclk	mmdc_axi_clk_root	Module clock

4.10.3 Address Mapping in various memory mapping modes

The address configured to the TZASC controller(s) must match the "local addresses" as being passed on to the DDR controller(s).

Memory "aliasing" implications on TZASC settings - in systems which does not utilize the maximal supported DDR space the controller is designed for, the whole DDR memory map becomes "aliased" (replicated) by the size of the physical memory used. In such cases, the TZASC must be configured to protect all aliased regions as well (i.e. effectively reducing the number of available TZASC regions, since all aliased regions must be handled, for each "real" space needing protection).

For complete details on TZASC functionality and the programming model, see the ARM document, “CoreLink™ TrustZone Address Space Controller TZC-380 Technical Reference Manual, (Rev r0p1 or newer)”, available at **http://infocenter.arm.com**.

4.11 System Debug

4.11.1 Debug

4.11.1.1 Debug Architecture

The chapter describes the debug architecture of the chip.

4.11.1.2 Debug System Features

The chip debug will be based on ARM's CoreSight "HUGO" platform, with support for Cortex-A7 and Cortex-M4 core from ARM.

- Support 5-pins (JTAG), 2-pins (cJTAG, ARM SWD) interface
- Support both non-intrusive and halt-mode trace / debug options
- MDM-AP registers for debugger to control mutli-core halt / resume cores
- Trace Memory Controller (TMC) is used to enable capturing trace
  - 4 KB in SOC trace block
  - ETR (4 G memory range, 64-bit wide at 266 MHz) is used to allow routing trace data to system memory
- Support ARM real time trace interface: TPIU (16-bit x 133 MHz)
- Support cross trigger between CA7 and CM4
- Four JTAG security levels, via SJC security functions together with e-Fuse (challenge response, field return, intrusive detection)

4.11.1.3 System level debug architecture

The i.MX7D debug architecture diagram is following:

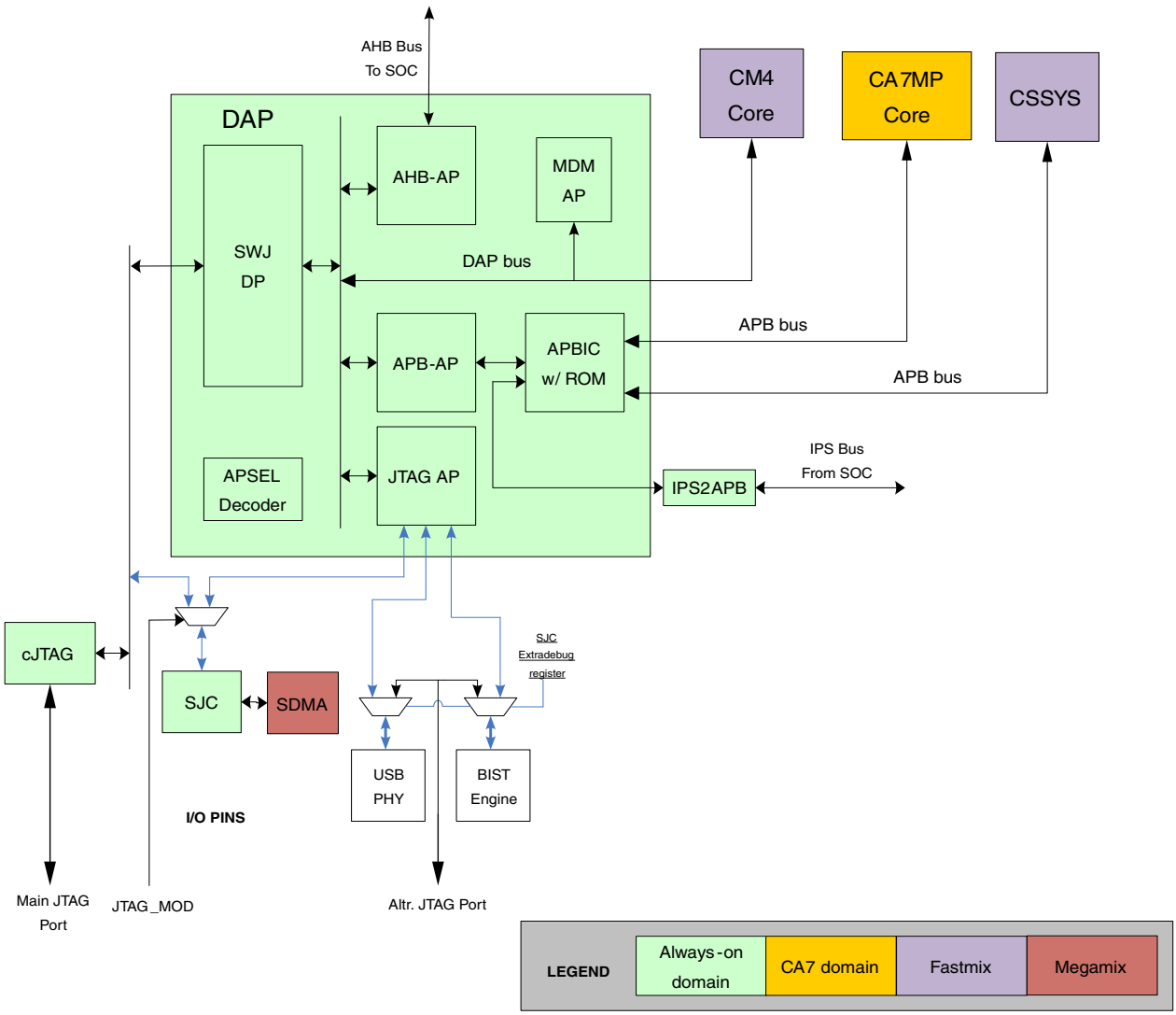


Figure 4-21. Debug Architecture Diagram

- Two JTAG modes are supported. Select via the JTAG_MOD pin.
  - Debug mode: JTAG_MOD = 0, DAP is the only TAP controller in the daisy chain. SJC and SDMA will be attached to JTAG-AP of DAP.
  - Test mode: JTAG_MOD = 1, SJC is the only TAP controller in the daisy chain. 1149.1-compliant, and support 1149.6 AC coupled test.

    More detail JTAG topology, see JTAG topology
- DAP is in always-on domain, to make sure debugger never lose connection with the chip.
- CA7MPCore has one power domain, CM4 and other trace module in SOC fastmix power domain.

4.11.1.4 JTAG topology

The following diagram shows the connectivity between TAP controllers—cJTAG, DAP, SJC, SDMA.

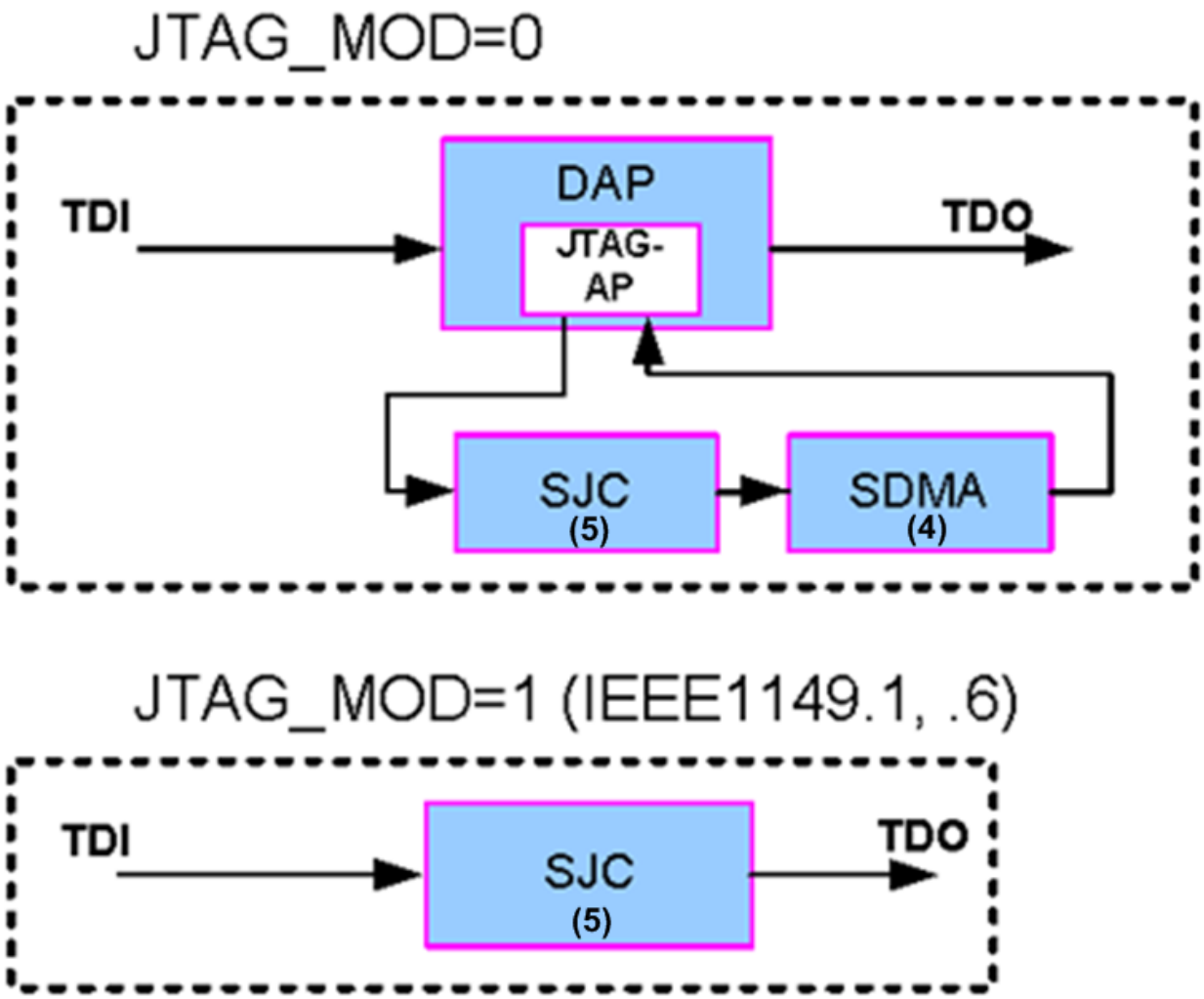


Figure 4-22. JTAG Chain in JTAG_MOD = 0 / 1

- JTAG_MOD = 0, Debug mode: DAP is the only TAP controller in the daisy chain. SJC and SDMA will be attached to JTAG-AP of DAP.
- JTAG_MOD = 1, Test mode: SJC is the only TAP controller in the daisy chain. 1149.1-compliant, and support 1149.6 AC coupled test.
- This structure is IEEE 1149.1 compliant and can be automatically detected by debugger, such as ARM RVI.

4.11.1.4.1 Debug Access Port (DAP) TAP

DAP is a standard ARM component. The DAP comprise a number of components. These components are used to access the DAP from an external debugger and Access Ports to access on-chip debug system resources.

The connectivity of the components in DAP is shown in Figure Figure 4-21.

- AHB-AP, allow the debugger to access to all memory and registers in the system
- APB-AP, allow the debugger to access the Coresight components in CA7 and HUGO
- JTAG-AP, allow the debugger to access SJC, do challenge-response authentication
- Export DAP bus to CM4 to host AHB-AP access port
- MDM-AP, used to host system level JTAG status and control registers.

Table 4-34. DAP Instruction Registers

Code	DAP IR
4'b0000	FSL IDCODE (Freescale IDCODE, consistent with SJC IDCODE) Note that TARGETID input for SW-DP should also be consistent with SJC IDCODE.
4'b1000	ABORT
4'b1010	DPACC
4'b1011	APACC
4'b1110	IDCODE (ARM IDCODE, TDO 32-bit output is 0x5BA00477)
4'b1111	BYPASS
Others	Reserved

4.11.1.5 Debug status and control registers

The Miscellaneous Debug Module (MDM-AP) is very useful in multi-processor run-control.

The debugger can write to MDM-AP control register only if the DBGEN is set. In case the debugger makes an access while DBGEN = 0, the access will be simply ignored and no error will be reported.

Table 4-35 shows the MDM-AP register overview. Besides the registers in the table, there is an IDR at the end of the 4K address range, described in Table 4-36. Table 4-36 is the description and integration guide for MDM-AP registers.

Table 4-35. MDM-AP Registers

	Status (dap_status)	Control (dap_ctrl)	Core halt request (dap_ctrl_halt_req)	Core restart request (dap_ctrl_restart_req)	Core halt ack (dap_status_halt_ack)
0	—	—	CM4 EDBGRQ	CM4 DBGRESTART	CM4 HALTED
1	—	—	—	—	—
2	—	—	CA7 EDBGRQ[0]	CA7 DBGRESTART[0]	CA7 DBGACK[0]
3	—	—	CA7 EDBGRQ[1]	CA7 DBGRESTART[1]	CA7 DBGACK[0]
4	—	—			
5	—	—			
6	—	—	—	—	—
7	—	—	—	—	—
8	—	—	—	—	—
9	—	—	—	—	—
10	—	—	—	—	—
11	—	—	—	—	—
12	—	—	—	—	—
13	—	—	—	—	—
14	—	—	—	—	—
15	—	—	—	—	—
16	—	—	—	—	—
17	—	—	—	—	—
18	—	—	—	—	—
19	—	—	—	—	—
20	—	—	—	—	—
21	—	—	—	—	—
22	CTI Trigger Out	—	—	—	—
23	—	—	—	—	—
24	—	CTI Trigger Input	—	—	—
25	—	CTI Trigger Out ACK	—	—	—
26	—	—	—	—	—
27	—	—	—	—	—
28	—	—	—	—	—
29	—	—	—	—	—
30	M4 DBGRESTARTED	—	—	—	—
31	—	—	—	—	—

Table 4-36. MDM-AP Register Description and Connectivity

Bit	Field	Connect to	Description
Status (dap_status)			
22	CTI trigger out	HUGO.event_dap_trigout	CTI interface trigger out
30	M4 DBGRESTARTED	CM4 integration	Indicate CM4 leaves debug mode
Control (dap_ctrl)			
24	CTI Trigger Input	HUGO.event_dap_trigin	Connects to CTI trigger input
25	CTI Trigger Out ACK	HUGO.eventack_dap_trigout	Connects to CTI trigger ACK input
Core halt request (dap_ctrl_halt_req)			
0	CM4 EDBGRQ	CM4 integration	The external debug request debug event which causing CM4 enter Debug state.
2	CA7 EDBGRQ[0]	CA7 integration	The external debug request debug event which causing CA7 CPU#0 enter Debug state.
3	CA7 EDBGRQ[1]	CA7 integration	The external debug request debug event which causing CA7 CPU#1 enter Debug state.
Core restart request (dap_ctrl_restart_req)			
0	CM4 DBGRESTART	CM4	CM4 debug restart input to CM4 core which causing CM4 leaving debug state.
2	CA7 DBGRESTART[0]	CA7 integration	CA7 debug restart input to CM4 core which causing CA7 CPU#0 leaving debug state.
3	CA7 DBGRESTART[1]	CA7 integration	CA7 debug restart input to CM4 core which causing CA7 CPU#1 leaving debug state.
Core halt ack (dap_status_halt_ack)			
0	CM4 HALTED	CM4 integration	Indicate CM4 has entered debug halted mode
2	CA7 DBGACK[0]	CA7 integration	Indicate CA7 CPU#0 has entered debug halted mode
3	CA7 DBGACK[1]	CA7 integration	Indicate CA7 CPU#1 has entered debug halted mode
Identification Register			
0	Type	N/A	4’h0 (MDM-AP)
1			
2			
3			
4	Varient		4’h3 (MX7)
5			
6			
7			
8	Reserved		Read as 8’h00
9			
10			
11			
12			
13			
14			

Table continues on the next page...

Table 4-36. MDM-AP Register Description and Connectivity (continued)

Bit	Field	Connect to	Description
15			
16	Class		1'b0: Indicates not a memory access
17	JEP 106		11'b00000001110 (Freescale)
18	Identity + Continuation		
19			
20			
21			
22			
23			
24			
25			
26			
27			
28	Revision		4'h0
29			
30			
31			

4.12 System JTAG Controller (SJC)

4.12.1 Overview

The System JTAG Controller (SJC) provides debug and test control with the maximum security.

The test access port (TAP) is designed to support features compatible with the IEEE Standard 1149.1 v2001 (JTAG).

The figure below shows an overview of the JTAG architecture.

Figure 4-23. System JTAG Controller (SJC) Block Diagram

4.12.1.1 Features

The System JTAG Controller (SJC) provides the following capabilities:

- JTAG IEEE1149.1 mandatory instructions, see EXTEST Instruction, SAMPLE/ PRELOAD Instruction , and BYPASS Instruction .
- JTAG IEEE1149.1 optional instructions, see ID_CODE Instruction (IDCODE) , and HIGHZ Instruction.
- JTAG IEEE P1149.1 (standard JTAG) interface to off-chip test and development equipment including an SJC-only mode for true IEEE 1149.1 compliance, used primarily for board-level implementation of boundary scan.
- IEEE P1149.6 (JTAG) mandatory instructions, see EXTEST_PULSE instruction and EXTEST_TRAIN instruction. These two instructions enable edge-detecting behavior on the signal path containing AC pins.
- Debug-related control and status, such as putting selected cores into reset and/or debug mode and the ability to monitor individual core status signals via JTAG.
- Provides means for accessing each OnCE/ICE TAP controller independently to control a target system (see Modes of Operation).
- ExtraDebug logic (see ENABLE_ExtraDebug Instruction).
- The maximum clock speed of the SJC is one-eight of the lowest frequency of the accessed OnCE/ICE. For example in normal operation (no core in low-power mode), this frequency is one-eight of the SDMA frequency if this core is present in the TDI-TDO chain (serially connected with other cores or standalone). The user must also consider the 25 MHz frequency limitation on the CE bus.
- Core compliant modes to support standalone core debuggers (see Modes of Operation).
- Multi-cores daisy chained mode (default one) to support multi-core debuggers (see Modes of Operation).

Detailed information about the SJC is provided in the Security Reference Manual. Contact your Freescale representative for information about obtaining this document.

4.12.1.2 Modes of Operation

The SJC modes are controlled through both the TAP select register (SJC_TSR) and the MOD input port.

The MOD port (typically connected to pad of the same name) selects between two possible topologies of TAP connections, as seen at SoC level:

- Negating it (this should be the default state) selects all the TAPs (SJC, SDMA, DAP and ARM/ETM) to be connected in the TDI-TDO chain, which is referred to as "daisy chain" mode, throughout this chapter.
- Asserting it only selects the SJC TAP to be connected in the TDI-TDO chain.

IEEE1149.1 standard features are enabled by configuring the SJC input pin: MOD. Refer to the following table for MOD settings details:

Table 4-37. SJC Modes

MOD	Name	Description
0	Daisy chain ALL	For common SW debug (High speed and production)
1	SJC only	IEEE 1149.1 JTAG compliant mode

The following figure shows the SJC mode selection flow. The numbers shown in parenthesis below each block name indicates the TAP's IR length.

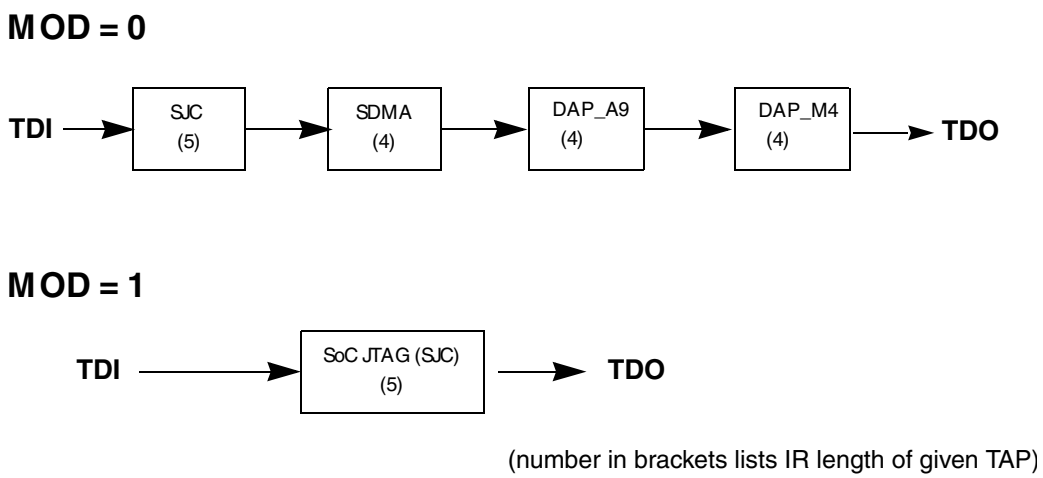


Figure 4-24. SJC Mode Selection Using MOD Pin Sampling

The Connect SDMA bit inside TAP select register controls the SDMA TAP bypass.

- When negated (should be the default state), the SDMA TAP is bypassed with a single D-FF (Flip-flop) during Shift-Dr path
- When asserted SDMA TAP is connected inside the chain
- When taking the SDMA into bypass or out of bypass (by writing to tapsel reg), additional cycle with TMS '0' should be given

The TAP selection block (TSB) provides a simple method of integrating various pieces of IP that have embedded TAPs.

- Provides a way to connect up multiple TAPs within a single SoC
- Identify the SJC TAP as the master TAP which controls the boundary chain (for IEEE 1149.1 standard compliance)
- Follow the state of SJC TAP, and when the Test-Logic-Reset (TLR) state is reached, reset all TAPs

The figure below shows the TAP Selection Block and SOC TAP Chain Scheme.

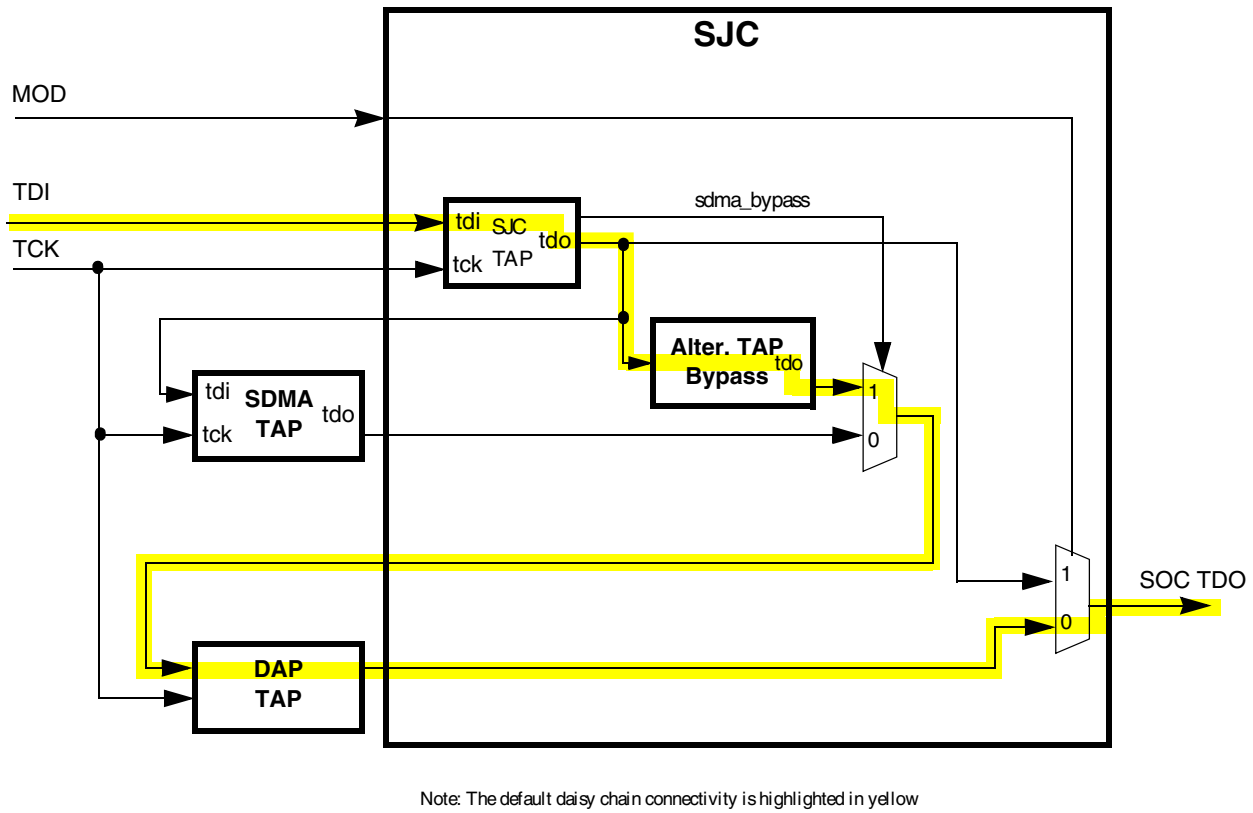


Figure 4-25. TAP Selection Block and SoC TAP Chain Scheme

NOTE

It is the responsibility of the user to ensure that in any configuration of the TAP controllers chosen, all of the TAPs in the chain comply with the demands of TCK clock frequency as well as the required ratio between TCK clock frequency and that of the core's to which the TAP refers.

4.12.2 External Signals

The table found here describes the external signals of SJC.

4.12.2.1 External Signal Overview

The SJC provides test and debug control with a minimum number of contacts.

The figure below shows SJC connections to external contacts and other chip blocks.

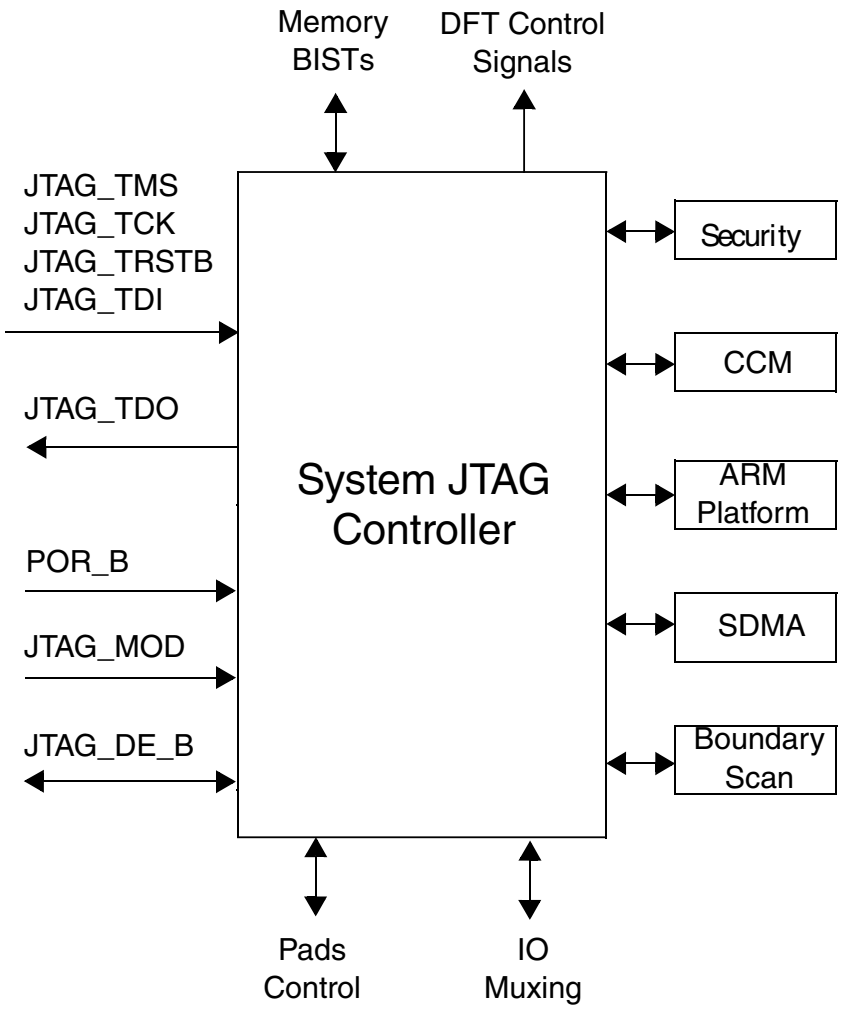


Figure 4-26. SJC Connections

4.12.2.2 TAP Controller

The TAP controller is responsible for interpreting the sequence of logical values on the TMS signal. It is a synchronous state machine that controls the operation of the JTAG logic. The value shown adjacent to each arc represents the value of the TMS signal sampled on the rising edge of TCK signal. For a description of the TAP controller states, refer to the appropriate IEEE 1149.1 document.

The state machine is shown in the following figure.

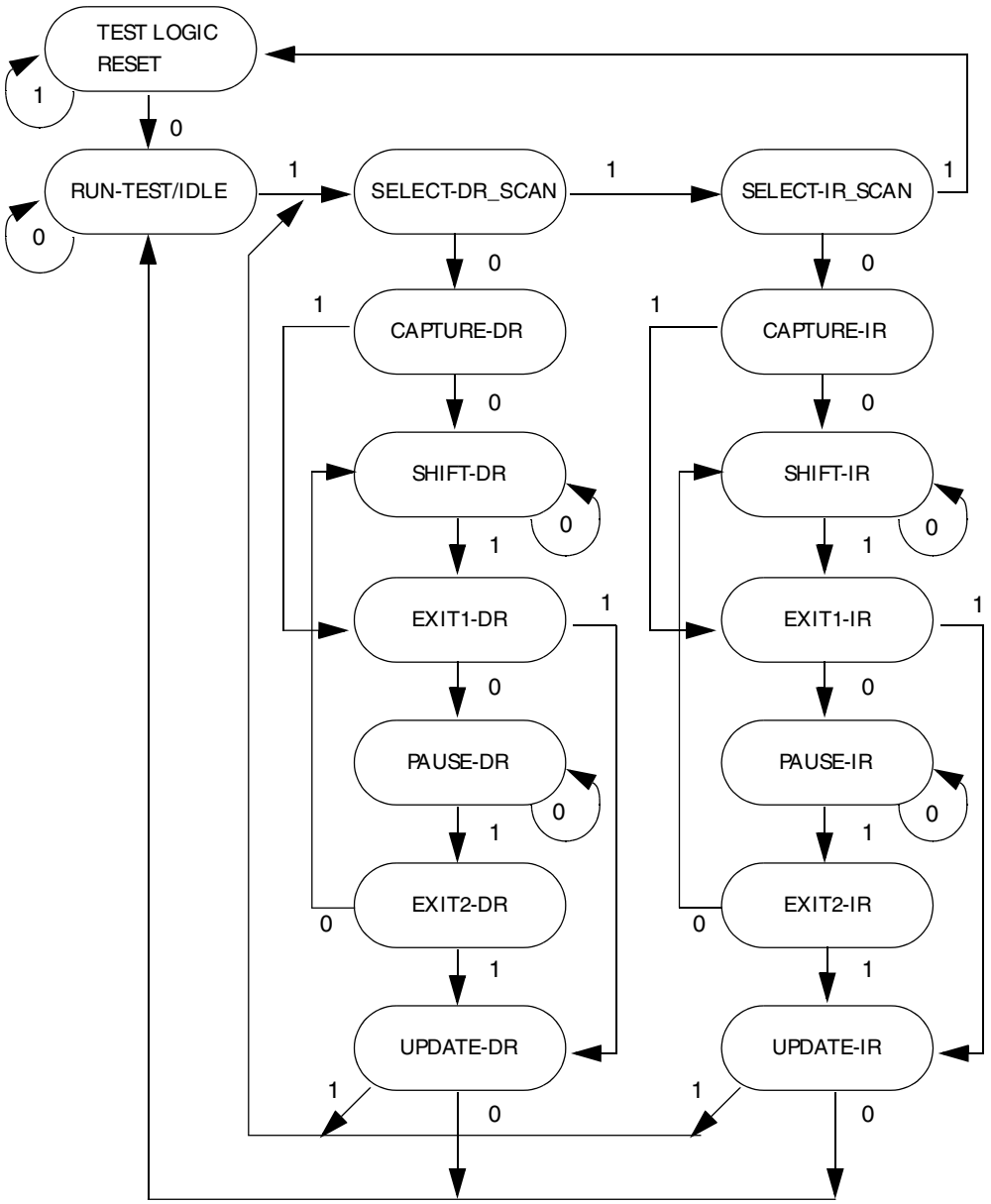


Figure 4-27. TAP Controller State Machine

The change of the JTAG state machine occurs on the rising edge of TCK. TMS and TDI change on the falling edge of TCK. TDO also changes on the falling edge of TCK following entry into the Shift_DR or Shift_IR states (TDO_EN is the enable of the tristate buffer driving the TDO output).

The figure below shows the timings of the SJC signals.

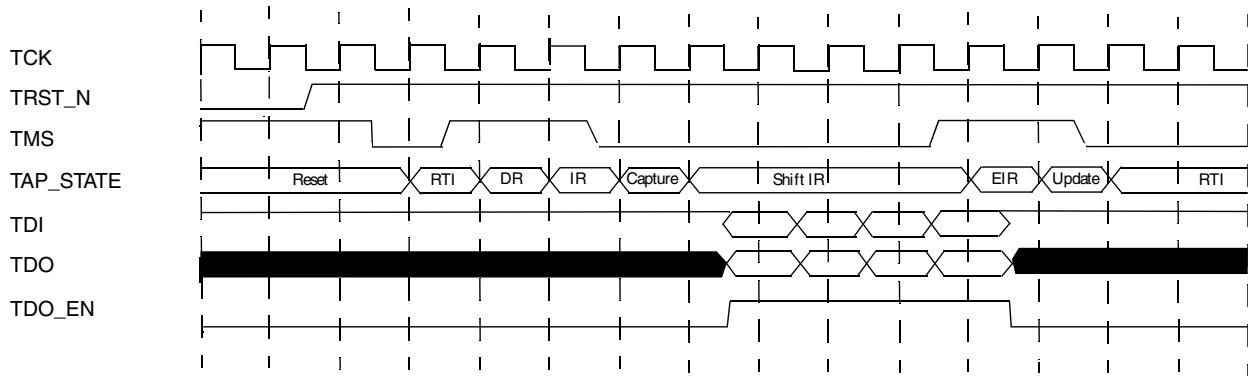


Figure 4-28. SJC Signals Timing Diagram

4.12.2.3 Accessing ExtraDebug Registers

Accessed through the Select-DR-Scan path, the ExtraDebug shift register consists of 39 bits (maximum) comprising a 32-bit data field (max length, see extradebug register description), a 5 bit address field and read/write bit.

The write actually takes place when the JTAG TAP controller enters the Update-DR state. On a read, the data field is ignored (the user should shift only 5 times to enter Read=1 and the address), the read takes place on the next path through DR at the Capture-DR state, the data is shifted-out during the Shift-DR state.

On the second path for a read access, simultaneous write access is not supported: command converter software shifts in zeros so the TAP decodes a write to the CSR (read-only register) which does not have any effect on the circuit.

The number of shift depends on the width of the accessed register as explained in the following diagrams.

First a write access (one path through Select-DR-Scan):

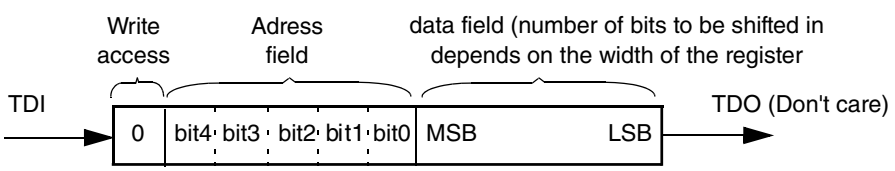


Figure 4-29. TDI/TDO on write access

Then a read access (requires two paths through Jtag DR Scan path):

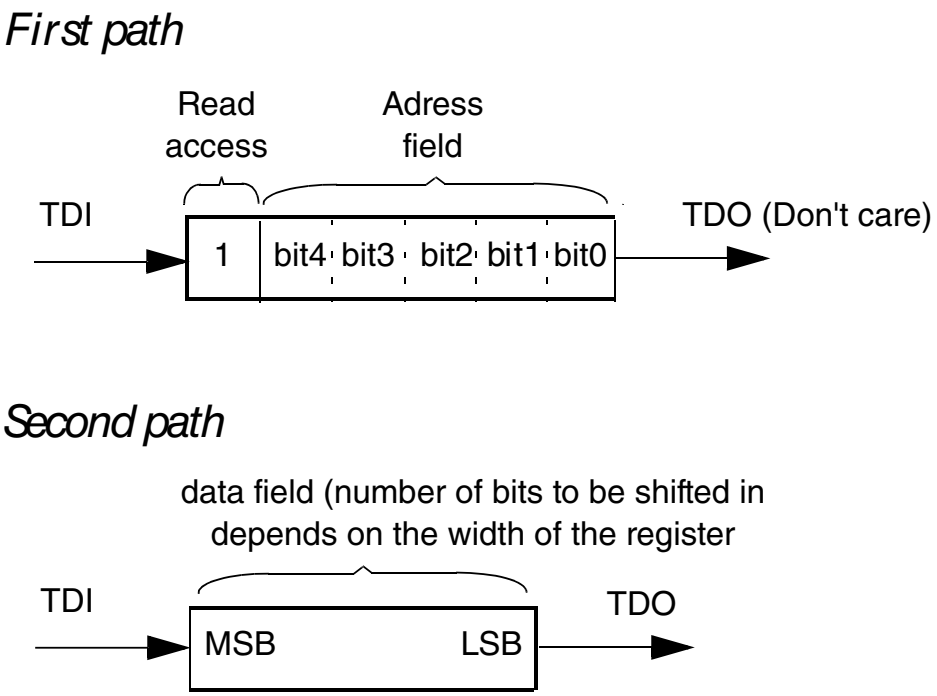


Figure 4-30. TDI/TDO on Read Access

For example, write value 0b1010_1100 to Debug Control Register (address = 0b00110).

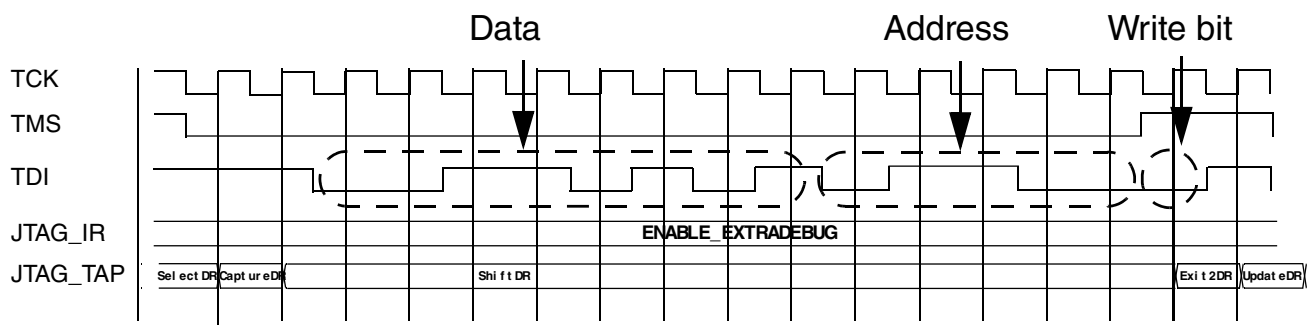


Figure 4-31. Example: Write Access to DCR

The SJC registers have different levels of security (Refer to JTAG Security Modes):

- Secured- accessible only in mode 2 (supposed correct response entered), mode 3 and mode 4.
- Unsecured- accessible in all modes

The level of security of each register is indicated in its name or description, in "Programmable Registers" section.

A single DE_B pin is dedicated for debug request input/output in bidirectional open drain functionality (including an internal pull-up device).

Bits 6:5 in DCR register serve as mask bits, controlling the propagation of external debug request to each recipients (ARM Platform, SDMA).

The bits 1:0 define the propagation enable of IR debug request to recipient cores.

Figure 4-32. DE_B Pin Select Logic

For security reasons, bits for output and input propagation control are at their negated values after reset. A user cannot put the cores in debug mode through DE_B without any Jtag access.

The configuration after reset prevents propagation of debug requests / acknowledges to or from the cores.

4.12.3 TAP Selection Block (TSB)

As described in Modes of Operation, the SJC can access cores in different modes selected through a TSB.

4.12.3.1 Select Mode Using Software

Conceptually, the SJC_TSR is a data register which is accessed through Access TSR IR instruction of SJC TAP.

The following figure shows the process of using reserved IR to access the SJC_TSR.

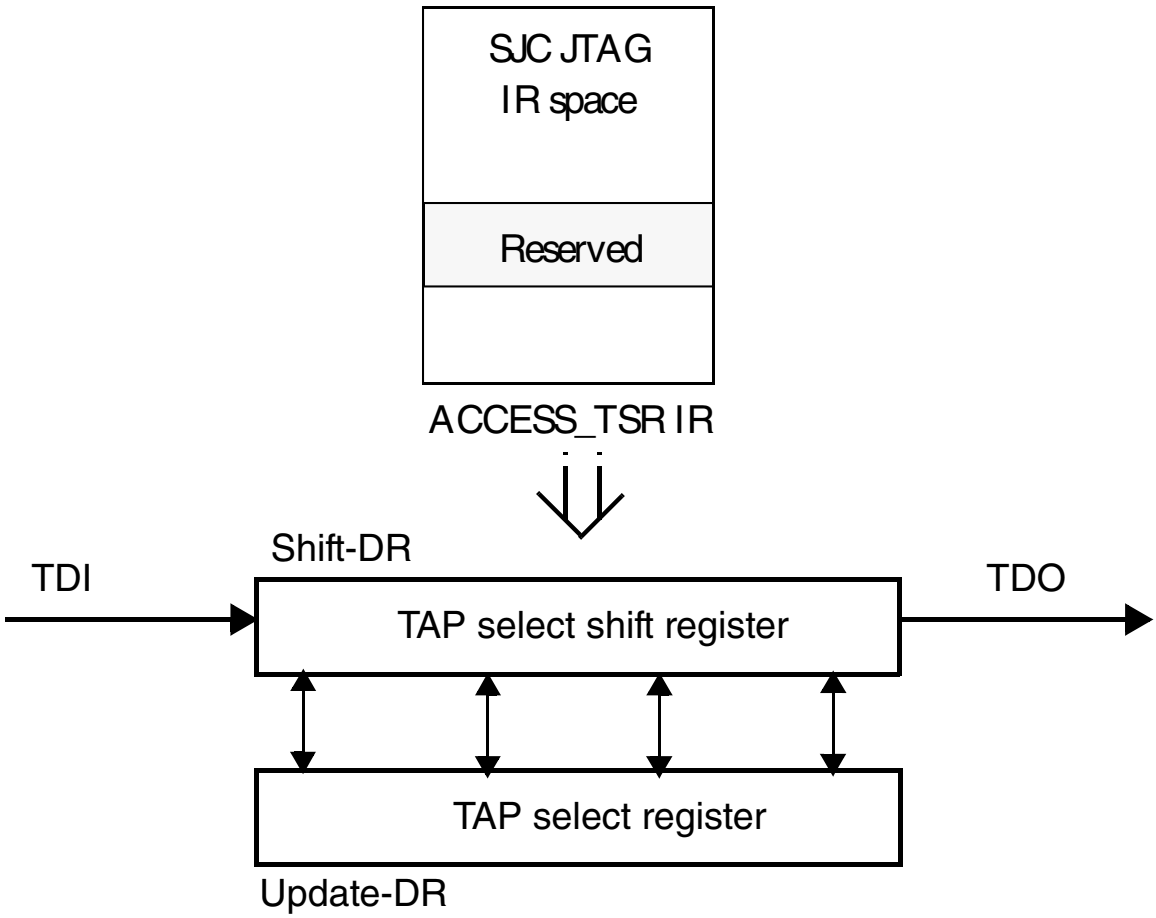


Figure 4-33. Using Reserved IR to Access the TAP Select Register (SJC_TSR)

The SJC_TSR can only be changed during the update-DR state of the TSB JTAG state machine. This is necessary to prevent a TAP that is being selected from losing synchronization with the TSB state machine when the TSB state machine returns to run-test-idle. Therefore, an associated shift register for the SJC_TSR is loaded into the SJC_TSR during the update-DR state (see the figure above). The shift register must also capture the state of the SJC_TSR when in the Capture-DR state for visibility of the contents of the SJC_TSR. See TAP Select Instruction , for more information.

4.12.4 Boundary Scan Register (BSR)

The Boundary Scan Register (BSR) in the JTAG implementation contains bits for all device signal and clock pins and associated control signals.

All SoC bidirectional pins have a single register bit in the boundary scan register for pin data, and are controlled by an associated control bit in the boundary scan register.

4.12.5 SoC JTAG Instruction Register (SJIR)

The SoC JTAG Instruction register can be found here.

The SoC JTAG Instruction register is 5 bits wide.

Table 4-38. SoC JTAG Instruction Register (SJIR)

Code					SJC IR
B4	B3	B2	B1	B0	
0	0	0	0	0	IDCODE
0	0	0	0	1	SAMPLE/PRELOAD
0	0	0	1	0	EXTEST
0	0	0	1	1	HI-Z
0	0	1	0	0	ENABLE_ExtraDebug
0	0	1	0	1	ENTER_DEBUG (secured)
0	0	1	1	0	Reserved
0	0	1	1	1	TAP select
0	1	0	0	0	EXTEST_PULSE
0	1	0	0	1	EXTEST_TRAIN
0	1	0	1	0	Reserved
0	1	0	1	1	Reserved
0	1	1	0	0	Security Output challenge
0	1	1	0	1	Security Enter response
-					Reserved
1	1	1	1	1	BYPASS

The instruction register is reset to 0b00000 in the test-logic-reset controller state which is equivalent to the IDCODE instruction.

During the capture-IR controller state, the parallel inputs to the instruction register are loaded with the code 01 in the least significant bits as required by the standard; the most significant bits are loaded with the values 00, leading to a capture value of 0b00001.

4.12.5.1 ID_CODE Instruction (IDCODE)

Selects the ID register, and the system logic controls the I/O pins. This instruction is provided as a public instruction to allow the manufacturer, part number and version of a component to be determined through the TAP.

The table below shows the ID register configuration.

Table 4-39. ID Configuration Register (IDCODE)

IDCODE				ID Configuration Register												
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Version Information[3:0]				Part Number (Bits 27-16)											
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x
Note:																
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Part Number (Bits 15-12)				Manufacturer Identity											1
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	x	x	x	x	0	0	0	0	0	0	0	1	1	1	0	1
Note:																

Table 4-40. ID Configuration Register Description (IDCODE)

Field	Description
31-28 Version Information	IC/SoC Version information number. Initial value: '0000' This number is subject to changes, for new IC/SoC (System On A Chip) revision releases.
27-12 Part Number	Customer Part Number The 16-bit Part Number value is unique for every Freescale's SoC / IC. See System Debug chapter for exact register value for a specific SoC.
11-1 Manufacturer Identity	Manufacturer Identity Freescale's Manufacturer Identity code. Bits [11:1] - 00000001110
0	Tied to logic 1.

One application of the ID register is to distinguish the manufacturer(s) of components on a board when multiple sourcing is used. As more components emerge which conform to the IEEE 1149.1 standard, it is desirable to allow for a system diagnostic controller unit to blindly interrogate a board design to determine the type of each component in each location. This information is also available for factory process monitoring and for failure mode analysis of assembled boards.

Once the IDCODE instruction is decoded, it selects the ID register which is a 32 Bit data register. Because the bypass register loads a logic 0 at the start of a scan cycle, whereas the ID register loads a logic 1 into its least significant bit, examination of the first bit of data shifted out of a component during a test data scan sequence immediate following exit from Test-Logic-Reset controller state shows whether such a register is included in

the design. When the IDCODE instruction is selected, the operation of the test logic has no effect on the operation of the on-chip system logic as required by the IEEE 1149.1 standard.

4.12.5.2 SAMPLE/PRELOAD Instruction

Selects the boundary scan register and the system logic controls the I/O pins.

The SAMPLE/PRELOAD instruction provides two separate functions:

- First, it provides a means to obtain a snapshot of system data and control signals. The snapshot occurs on the rising edge of TCK in the capture-DR controller state. The data can be observed by shifting it transparently through the boundary scan register.
- The second function of SAMPLE/PRELOAD is to initialize the boundary scan register output cells prior to selection of EXTEST. This initialization ensures that known data appears on the outputs when entering the EXTEST instruction.

NOTE

Because there is no internal synchronization between the JTAG clock (TCK) and the system clock (CLK), the user must provide some form of external synchronization to achieve meaningful results.

For more details on the function and use of SAMPLE/PRELOAD, refer to the appropriate IEEE 1149.1 document.

4.12.5.3 EXTEST Instruction

Selects the boundary scan register, and the 1149.1 test logic has control of the I/O pins.

By using the TAP controller, the register is capable of:

- Scanning user-defined values into the output buffers,
- Capturing values presented to input pins
- Controlling the direction of bidirectional pins,
- Controlling the output drive of tri-statable output pins.

For more details on the function and use of EXTEST, refer to the appropriate IEEE 1149.1 document.

The EXTEST instruction also asserts internal reset for the cores (through CCM, refer to Figure 4-36) to force a predictable internal state while performing external boundary scan operations.

4.12.5.4 HIGHZ Instruction

All output drivers, including the two-state drivers, are turned off (that is, high impedance). The instruction selects the bypass register.

In this mode, all internal pullup resistors on all the pins (except for the TMS, TDI, TCK, TRSTB pins) are disabled. This disabling functionality is not built into SJC, but should be implemented by some logic in the SOC/IO Pads.

For more details on the function and use of HIGHZ, refer to the IEEE 1149.1 document.

The HIGHZ instruction also asserts internal reset for the cores (through CCM, refer to Figure 4-36) to force a predictable internal state while performing external boundary scan operations.

4.12.5.5 BYPASS Instruction

Selects the single Bit bypass register and the system logic controls the I/O pins.

This creates a shift-register path from TDI to the bypass register and, finally, to TDO, circumventing the boundary scan register. This instruction is used to enhance test efficiency when a component other than the SoC Core based device becomes the device under test.

When the bypass register is selected by the current instruction, the shift-register stage is set to a logic zero on the rising edge of TCK in the capture-DR controller state. Therefore, the first bit to be shifted out after selecting the bypass register is always a logic zero.

For more details on the function and use of BYPASS, refer to the appropriate IEEE 1149.1 document.

4.12.5.6 ENABLE_ExtraDebug Instruction

The TDI and TDO pins are connected directly to the ExtraDebug registers, the SJC TAP controller remaining connected to TDI and TMS.

The ExtraDebug shift register consists of 38 bits (maximum) comprising a 32-bits data field (maximum length, see Accessing ExtraDebug Registers,), a 5 bits address field and read/$\overline{\text{write}}$ bit. On a register read, the data field does not need to be filled in. The particular ExtraDebug register connected between TDI and TDO at a given time is

selected by the ExtraDebug controller depending on the ExtraDebug Address being currently decoded. All communication with the ExtraDebug controller is done through the Select-DR-Scan path of the JTAG TAP Controller.

4.12.5.7 ENTER_DEBUG instruction

The ENTER_DEBUG instruction is used to generate a debug request event to SDMA and the ARM Core Platform simultaneously (practically, inherited minimal skew is expected, due to difference in event signal propagation in the different modules).

The TDI and TDO are connected to the Instruction Register (IR). After the acknowledgment of the Debug Mode is received (can be checked by reading the Core Status Register part of the ExtraDebug logic), the user can perform system debug functions on the cores.

NOTE

The ENTER_DEBUG event issue to the cores, can be masked, by bits in DCR register.

NOTE

It is user's responsibility to shift-in another IR value (like IDCODE) before trying to bring the cores out of debug mode, as the debug request signals to the cores remains asserted as long as ENTER_DEBUG IR is in place.

NOTE

The user need to check that cores are in debug mode (watching debug acknowledge signal) before leaving ENTER_DEBUG instruction, otherwise debug request might not take affect.

4.12.5.8 TAP Select Instruction

By means of TAP select instruction a user can access TAP select register and by controlling its only bit SDMA Bypass, control whether SDMA TAP is bypassed or not.

Table 4-41. TAP Select Register (TSR)

	TAP Select Register
	BIT 0
	Connect SDMA
TYPE	rw
RESET	0
Note:	

Table 4-42. TAP Select Register Description

Field	Description
0 SDMA Bypass	Connect SDMA Control whether SDMA TAP is bypassed or not: • 0 - SDMA TAP is bypassed by the alternate TAP inside SJC (emulating 4-bit IR and 1-bit bypass path). • 1 - SDMA TAP is connected to the TDI-TDO chain. **NOTE:** Additional cycle with TMS '0' should be inserted, after writing to this register, to allow the SDMA tap be sync before SDMA get into / out of bypass.

4.12.5.9 EXTEST_PULSE instruction

The EXTEST_PULSE instruction implements new test behaviors for AC pins and simultaneously behaves identically to IEEE Std 1149.1 EXTEST for DC pins.

4.12.5.10 EXTEST_TRAIN instruction

The EXTEST_TRAIN instruction implements new test behaviors for AC pins and simultaneously behaves identically to IEEE Std 1149.1 EXTEST for DC pins.

4.12.6 Security

JTAG manipulation is one of the known hackers' ways of executing unauthorized program code, getting control over the OS and run code in privileged modes.

The SJC provides a debug access to several H/W blocks including the ARM processor and the system bus. This allows for program control and manipulation as well as visibility into system peripherals and memory. The ETM and NEXUS interfaces allow bus transactions to be traced. Together these tools provide the hacker all the access needed to completely comprise the system. Means must be provided to block any malicious JTAG access.

The SJC provides a way of regulating the JTAG access.

The following are the different JTAG security modes:

- **Mode #1: No Debug**-Maximum Security. All security sensitive JTAG features are permanently blocked.

- **Mode #2: Secure JTAG**-High security. JTAG use is regulated by secret key based authentication mechanism.
- **Mode #3: JTAG Enabled**-Low security. JTAG always enabled.

The JTAG security modes are configured using eFUSEs which can burned after packaging by applying electrical signals. The fuse burning is irreversible process, once a fuse is burned (e-fuse or laser fuse) it is impossible to change the fuse back to the un-burned state.

4.12.6.1 JTAG Security Modes

JTAG can be in one of JTAG security modes which is selected by setting the SJC eFUSE configuration. The physical location of the fuses is not in the SJC.

4.12.6.1.1 Mode 1: No Debug - Maximum Security

No Debug JTAG security mode provides the highest security level.

In this mode, all JTAG features are disabled except for:

- ScanBoundary Scan
- MBIST, all modes except for debug modes which enable controlled memory contents output
- PLL BIST
- BIST monitor mode, allowing routing to external pins BIST pass/fail/invoke information
- PLL bypass- Bypass ARM or/and USB PLL.
- Visibility of the following status bits: power mode - normal, standby, stop, shutdown, and so on

These features do not reduce the security level of the product, and they allows to perform important tests and board connectivity checks.

4.12.6.1.2 Mode 2: Secure JTAG - High Security

The Secure JTAG mode limits the JTAG access by using challenge/response based authentication mechanism. Any access to JTAG port is being checked. Only authorized debug devices (that is, devices having the right response) can access the JTAG, unauthorized JTAG access attempts are denied.

The intent of this mode is to allow return field testing. When a secured JTAG device is being returned for debugging, this mode allows authorized re-activation of the JTAG.

4.12.6.1.2.1 Challenge/Response Mechanism in System JTAG Mode

When SJC is in Sysytem JTAG mode the authentication process is as follows:

1. Shift Output Challenge instruction to IR.
2. Passing through Capture-DR state of the SJC and by performing Shift-DR operations Challenge code can be accessed from TDO.
3. Shift Enter Response instruction to IR. By performing Shift-DR, operations enter Response code value through TDI. As Update-DR state is entered, Response code is compared with the correct one.

In Fixed challenge-response pair mode, each part has its individual challenge - response pair which is determined at manufacturing time, and does not change later on. The SJC compares the user's response to the expected response.

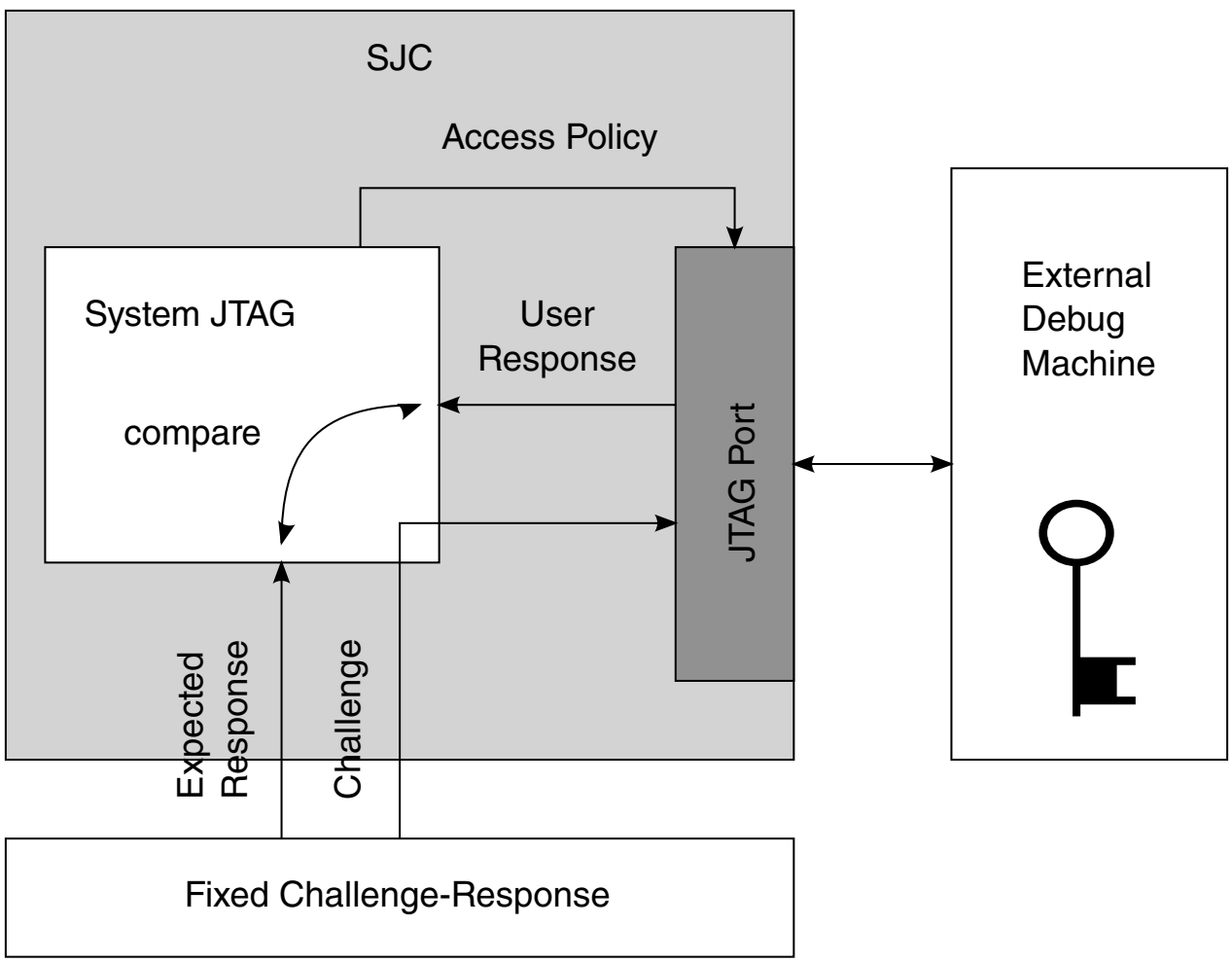


Figure 4-34. Mode #2 - Secure JTAG with Fixed Challenge-response Pair

4.12.6.1.3 Mode 3: JTAG Enabled - Low Security

In the JTAG Enabled JTAG security mode, all JTAG features are enabled.

4.12.6.2 Software Enabled JTAG

To increase the flexibility of the SJC, an option to enable the JTAG via software is added and is available only in Secure JTAG mode. By writing '1' to HAB_JDE (HAB JTAG DEBUG ENABLE) bit in the e-fuse controller module, the JTAG is opened, regardless of its security mode. It is the responsibility of software to assert or negate this bit.

Additionally, a corresponding lock bit is available (in the e-fuse control module) to ensure that only trusted software is able to set the JDE bit. When the LOCK bit is set, no future change of JDE is possible, until the next POR (power-on-reset) cycle.

The platform initialization software should set the LOCK bit for JDE bit before transferring control to the application code.

The S/W JTAG enable allows JTAG enabling without activating the challenge-Response mechanism (which requires JTAG access tool enhancement or special H/W). The JTAG S/W enable does not allow debug in case of boot or memory fault as it requires reset before entering debug.

This feature can be permanently blocked by burning the dedicated e-fuse.

NOTE

The S/W enabled JTAG feature reduces the overall security level of the system as it relies on S/W protections. If this feature is not required, it is strongly recommended to burn the JTAG_HEO e-fuse which disables this feature.

4.12.6.3 Kill Trace

The kill trace signal disables any output of the ETM block. The ETM can be accessed either via JTAG port and/or by direct software code. Blocking the JTAG port also yields assertion of the kill trace signal. This resulted in blocking of trace port. The intention of this action is to block any attempt to break into the system via software manipulation of the debug modules. The kill trace, when active, prevents trace output even in case where it can be activated via chip pin.

The kill trace feature needs to be activated by burning a dedicated e-fuse. If the fuse is left intact, kill trace is never activated as seen in Figure 4-35.

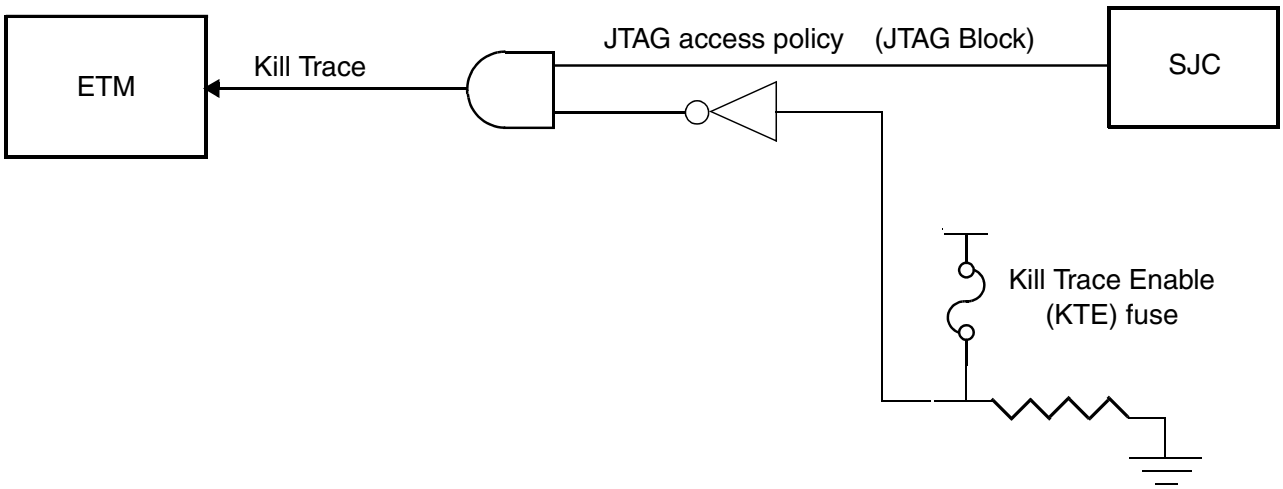


Figure 4-35. Kill Trace eFUSE

The kill trace is asserted when "kill trace enable" fuse is burned and "ipt_secur_block" signal in SJC is asserted, which happens when at least one of the following is true:

- Mode #2 (Secure JTAG) and no code has been entered
- Mode #2 (Secure JTAG) with burned Bypass and Re-enable fuses
- Mode #2 (Secure JTAG) with incorrect response entered
- Mode #1 (No debug)
- TRST_B signal is active
- POR has not ever been asserted

4.12.6.4 SJC Disable Fuse

In addition to the different JTAG security modes that are implemented internally in the System JTAG Controller (SJC), there is an option to disable the SJC functionality by eFUSE configuration. This creates additional JTAG mode that is, JTAG Disabled with highest level of JTAG protection. In this mode all JTAG features are disabled.

Specifically, the following debug features are disabled in addition to the features that were already disabled in No Debug JTAG mode:

- Memory BIST
- Boundary scan register (SJC_BSR)
- Non-Secure JTAG control registers (PLL configuration, Deterministic Reset, PLL bypass)
- Non-Secure JTAG status registers (Core status)
- Chip Identification Code (IDCODE)

4.12.7 Functional Description

This section provides a complete functional description of the block.

4.12.7.1 Static Core Debug

The SJC JTAG TAP controller is fully compatible with the IEEE 1149.1a-2001 Standard Test Access Port and Boundary Scan Architecture specifications.

The ARM platform has an integrated JTAG interface and a TAP controller to manage its own ICE. Also it can access an embedded trace ETM interface, see ARM core and ETM Technical reference guide for more information.

The SDMA has a TAP controller to manage its own OnCE, see SDMA OnCE specifications for more details.

The OnCE and ICE provide a mean of interacting with the cores and their peripherals non-intrusively so that a user may examine registers, memories to facilitate hardware and software development. Refer to TAP Selection Block (TSB), for more information.

4.12.7.2 Reset Mechanism

The following figure shows the SJC reset logic

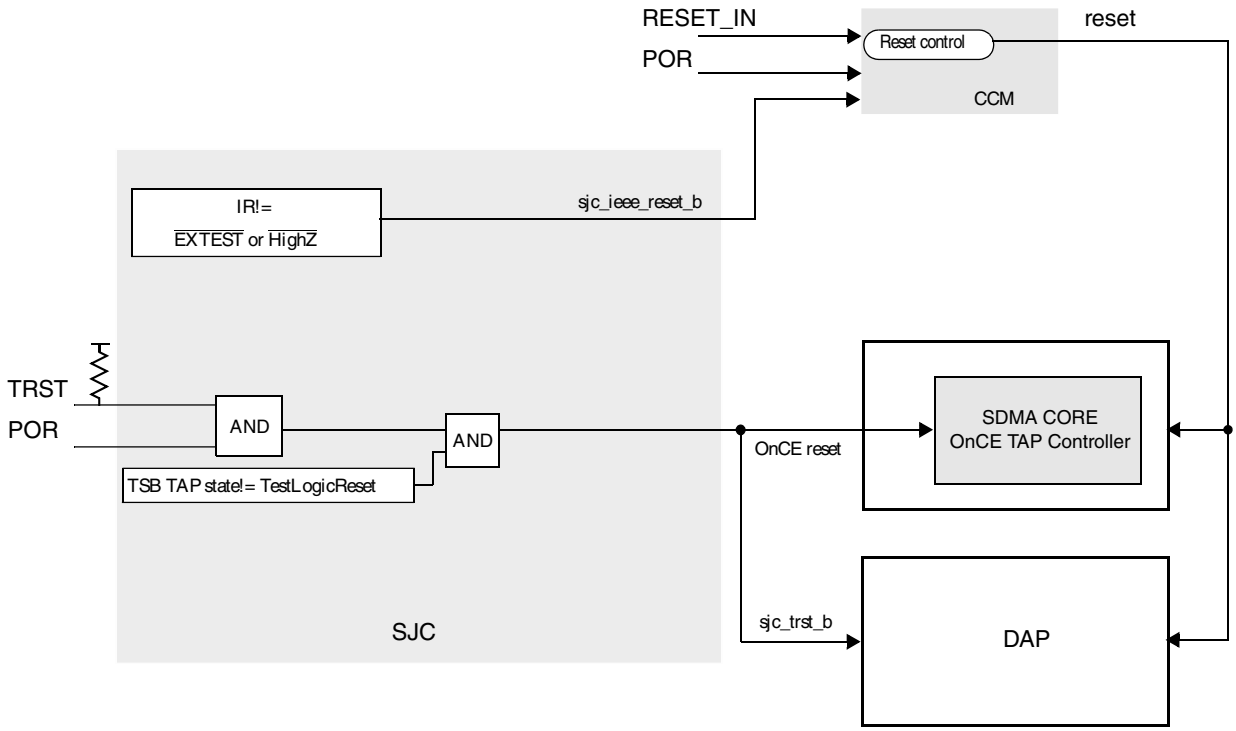


Figure 4-36. SJC Reset Logic

NOTE

- Asserting TRSTB in any scan mode resets the TCR loosing the testmode configuration and selects default TAP.
- SJC generates an IEEE reset signal to the CCM when in one of the IEEE modes HIGHZ or EXTEST. This signal generates a system reset to the cores until exit from one of these modes.
- The TSB generates Once/ICE reset (either TRSTB if implemented or other) when its TAP state reaches Test-Logic-Reset (meaning that TAP accessed is also reaching Test-Logic-Reset).

4.12.8 Initialization/Application Information

The control afforded by the output enable signals using the boundary scan register and the EXTEST instruction requires a compatible circuit-board test environment to avoid device-destructive configurations. The user must avoid situations in which the SJC output drivers are enabled into actively driven networks.

There are two constraints related to the JTAG interface:

- Ensure that the JTAG test logic is kept transparent to the system logic by forcing TAP into the Test-Logic-Reset controller state. During power-up, SJC's internal TRSTB is asserted as IC's POR_B is asserted which forces the TAP controller into this state. After that, if TMS either remains unconnected or is connected to VCC, then the TAP controller cannot leave the Test-Logic-Reset state, regardless of the state of TCK.
- DE_B is an IO pin with pullup and care must be taken of the direction when driving this signal.

4.12.9 SJC Memory Map/Register Definition

In addition to the standard accessible JTAG registers (per IEEE1149.1 standard) listed in SoC JTAG Instruction Register (SJIR) , the chip contains the following registers accessed using the ExtraDebug mechanism, controlled via "ENABLE_ExtraDebug" IR instruction.

NOTE

SJC registers are only accessible by JTAG interface. They are not memory mapped to processor address space, so the absolute addresses provided by default in the SJC memory map are not valid.

This section assumes the JTAG controller is accessed in standalone mode or daisy chained (defined by TAP Selection Block) using the appropriate TSB configuration.

See "System Debug" chapter for more details about the general purpose register descriptions that are unique to this chip.

SJC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	General Purpose Unsecured Status Register 1 (SJC_GPUSR1)	32	R	0000_0000h	4.12.9.1/ 437
1	General Purpose Unsecured Status Register 2 (SJC_GPUSR2)	32	R	0000_0000h	4.12.9.2/ 439
2	General Purpose Unsecured Status Register 3 (SJC_GPUSR3)	32	R	0000_0000h	4.12.9.3/ 439
3	General Purpose Secured Status Register (SJC_GPSSR)	32	R	0000_0000h	4.12.9.4/ 440
4	Debug Control Register (SJC_DCR)	32	R/W	0000_0000h	4.12.9.5/ 441

Table continues on the next page...

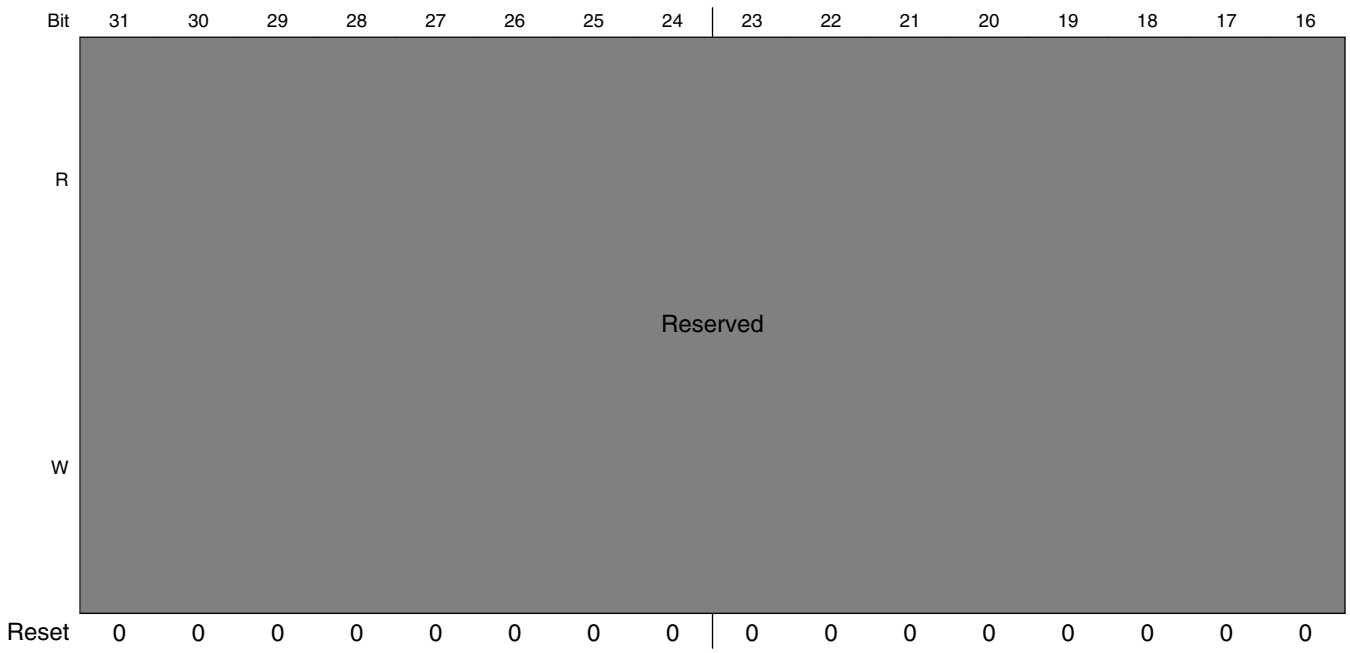
SJC memory map (continued)

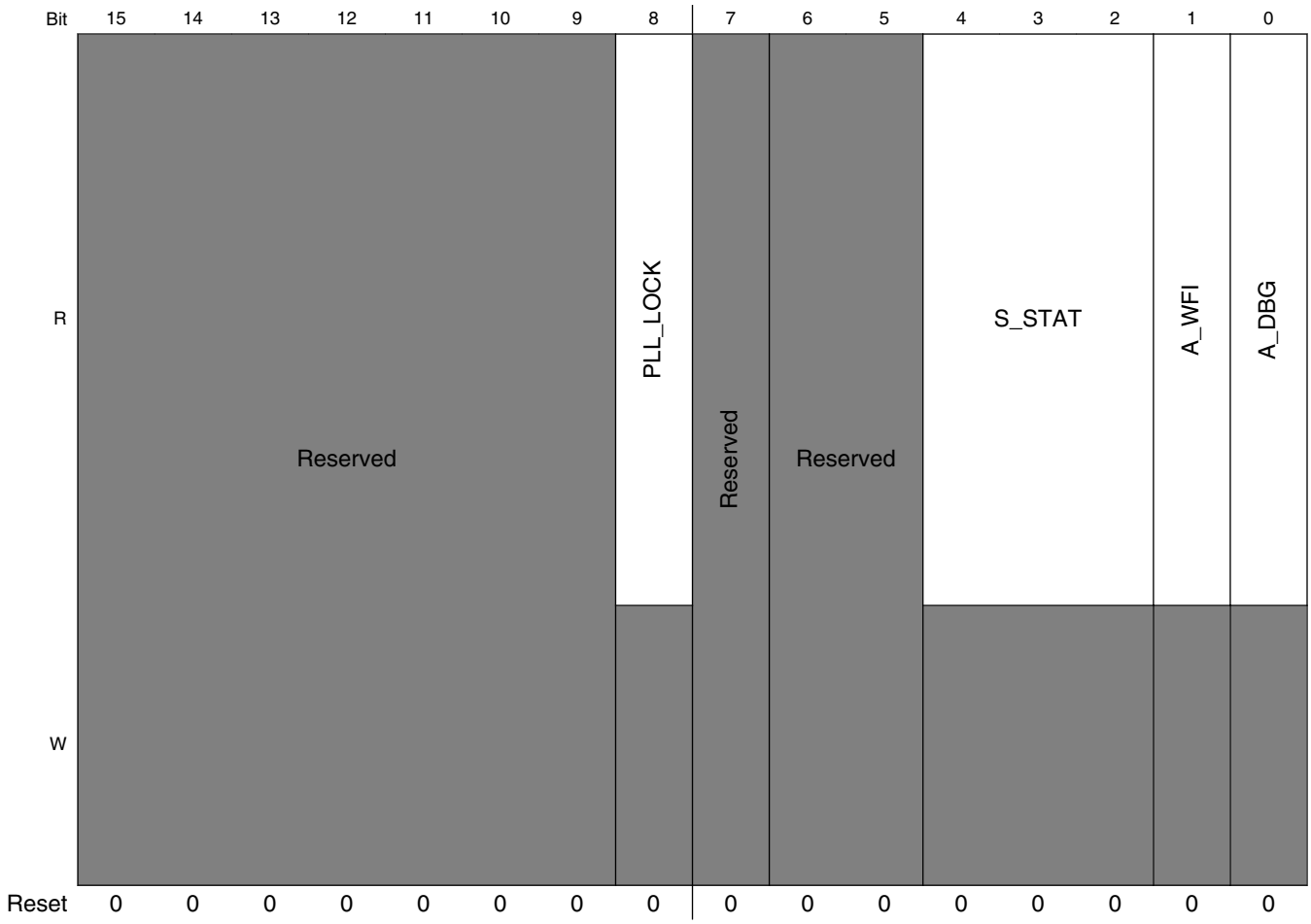
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5	Security Status Register (SJC_SSR)	32	R	See section	4.12.9.6/ 443
7	General Purpose Clocks Control Register (SJC_GPCCR)	32	R/W	0000_0000h	4.12.9.7/ 446

4.12.9.1 General Purpose Unsecured Status Register 1 (SJC_GPUSR1)

The General Purpose Unsecured Status Register 1 is a read only register used to check the status of the different Cores and of the PLL. The rest of its bits are for general purpose use.

Address: 0h base + 0h offset = 0h



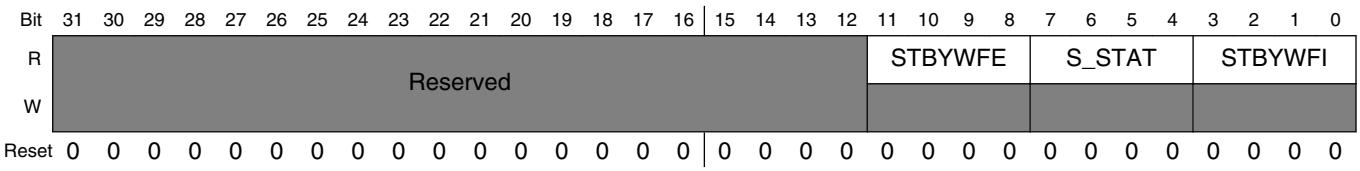


SJC_GPUSR1 field descriptions

Field	Description
31–9 -	This field is reserved. Reserved.
8 PLL_LOCK	PLL_LOCK A Combined PLL-Lock flag indicator, for all the PLL's.
7 -	This field is reserved. Reserved
6–5 -	This field is reserved. Reserved.
4–2 S_STAT	3 LSBits of SDMA core statusH.
1 A_WFI	ARM core wait-for interrupt bit Bit 1 is the ARM core standbywfi (stand by wait-for interrupt). When this bit is HIGH, ARM core is in wait for interrupt mode.
0 A_DBG	ARM core debug status bit Bit 0 is the ARM core DBGACK (debug acknowledge) DBGACK can be overwritten in the ARM core DCR to force a particular DBGACK value. Consequently interpretation of the DBGACK value is highly dependent on the debug sequence. When this bit is HIGH, ARM core is in debug.

4.12.9.2 General Purpose Unsecured Status Register 2 (SJC_GPUSR2)

Address: 0h base + 1h offset = 1h

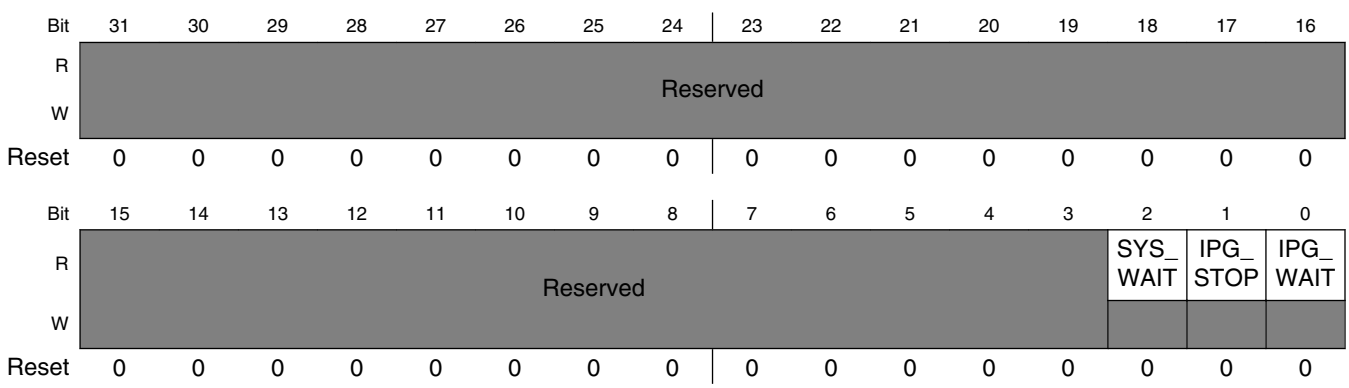


SJC_GPUSR2 field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11–8 STBYWFE	STBYWFE[3:0] Reflecting the "Standby Wait For Event" signals of all cores.
7–4 S_STAT	S_STAT[3:0] SDMA debug status bits: debug_core_state[3:0]
STBYWFI	STBYWFI[3:0] These bits provide status of "Standby Wait-For-Interrupt" state of all ARM cores.

4.12.9.3 General Purpose Unsecured Status Register 3 (SJC_GPUSR3)

Address: 0h base + 2h offset = 2h



SJC_GPUSR3 field descriptions

Field	Description
31–3 -	This field is reserved. Reserved

Table continues on the next page...

SJC_GPUSR3 field descriptions (continued)

Field	Description
2 SYS_WAIT	System In wait Indication on System in wait mode (from CCM).
1 IPG_STOP	IPG_STOP CCM's "ipg_stop" signal indication
0 IPG_WAIT	IPG_WAIT CCM's "ipg_wait" signal indication

4.12.9.4 General Purpose Secured Status Register (SJC_GPSSR)

The General Purpose Secured Status Register is a read-only register used to check the status of the different critical information in the SoC. This register cannot be accessed in secure modes.

Address: 0h base + 3h offset = 3h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPSSR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

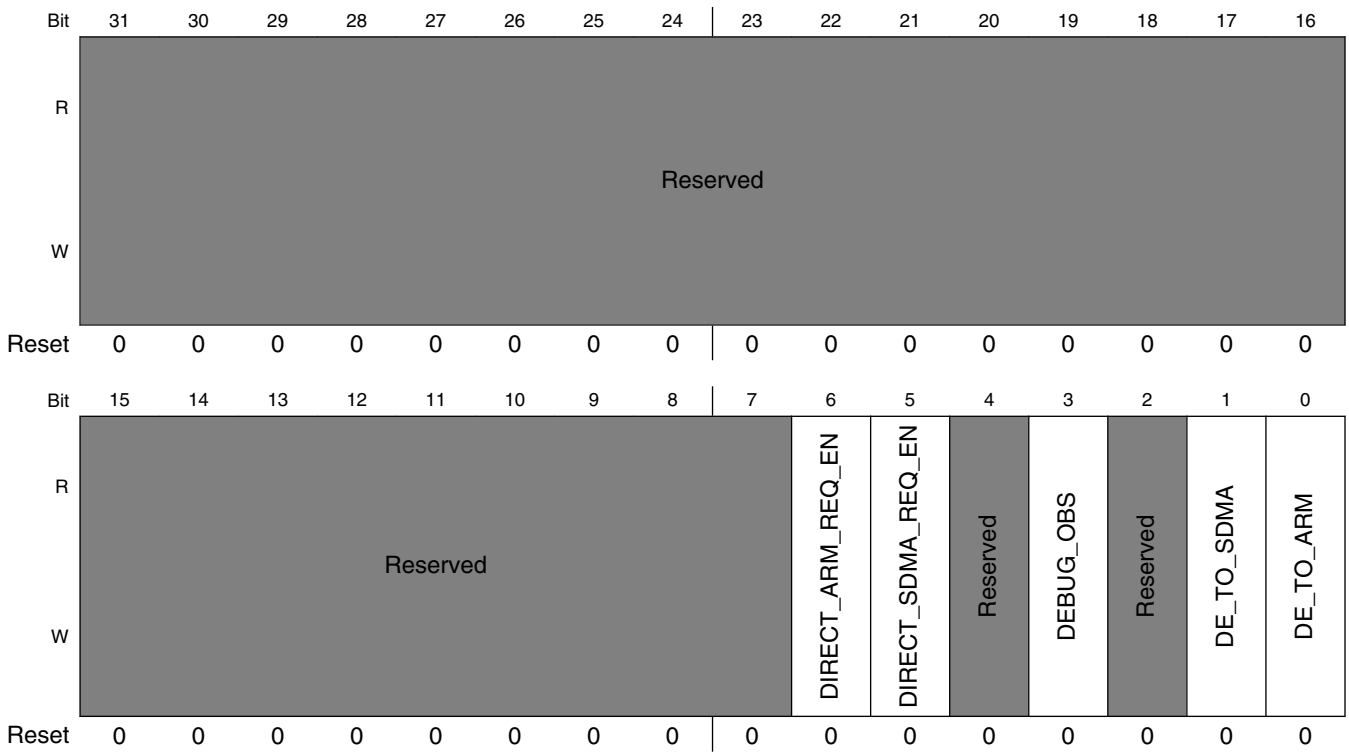
SJC_GPSSR field descriptions

Field	Description
GPSSR	General Purpose Secured Status Register Register is used for testing and debug.

4.12.9.5 Debug Control Register (SJC_DCR)

This register is used to control propagation of debug request from DE_B pad to the cores and debug signals from internal logic to the DE_B pad.

Address: 0h base + 4h offset = 4h



SJC_DCR field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6 DIRECT_ARM_REQ_EN	Pass Debug Enable event from DE_B pin to ARM platform debug request signal(s). This bit controls the propagation of debug request DE_B to the Arm platform. 0 Disable propagation of system debug to (DE_B pin) to Arm platform. 1 Enable propagation of system debug to (DE_B pin) to Arm platform.
5 DIRECT_SDMA_REQ_EN	Debug enable of the sdma debug request This bit controls the propagation of debug request DE_B to the sdma. 0 Disable propagation of system debug to (DE_B pin) to sdma. 1 Enable propagation of system debug to (DE_B pin) to sdma.
4 -	This field is reserved. Reserved

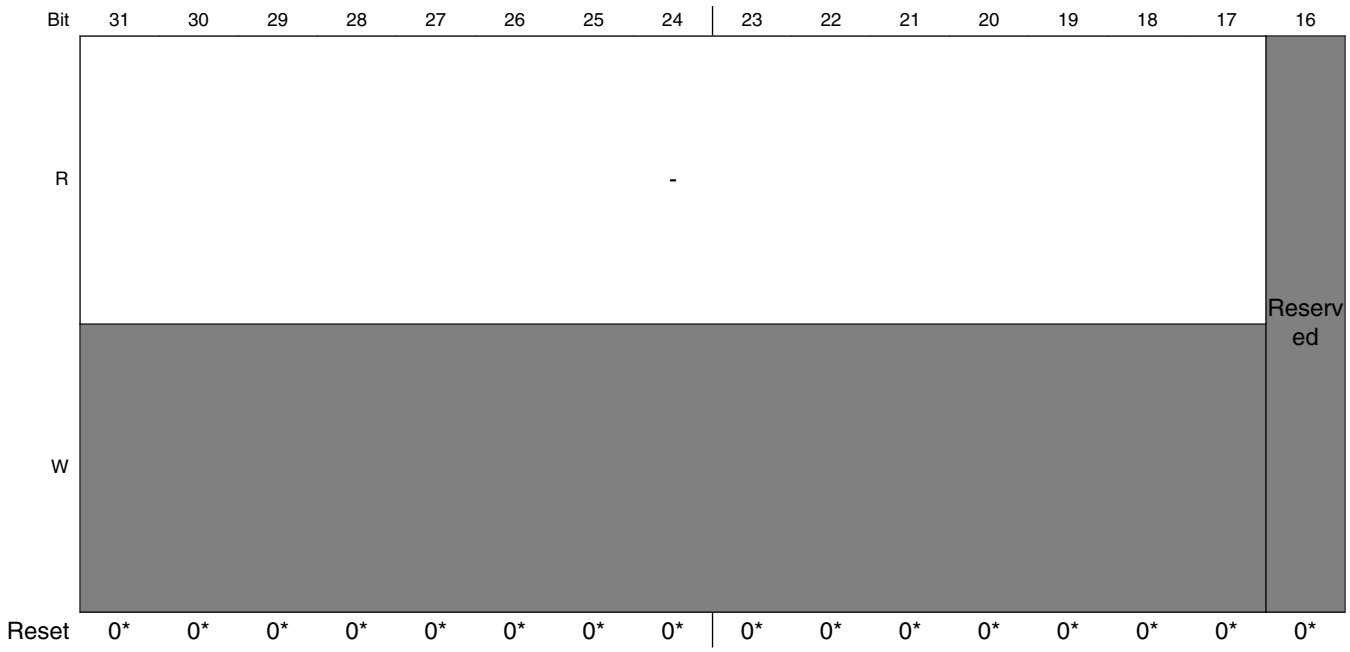
Table continues on the next page...

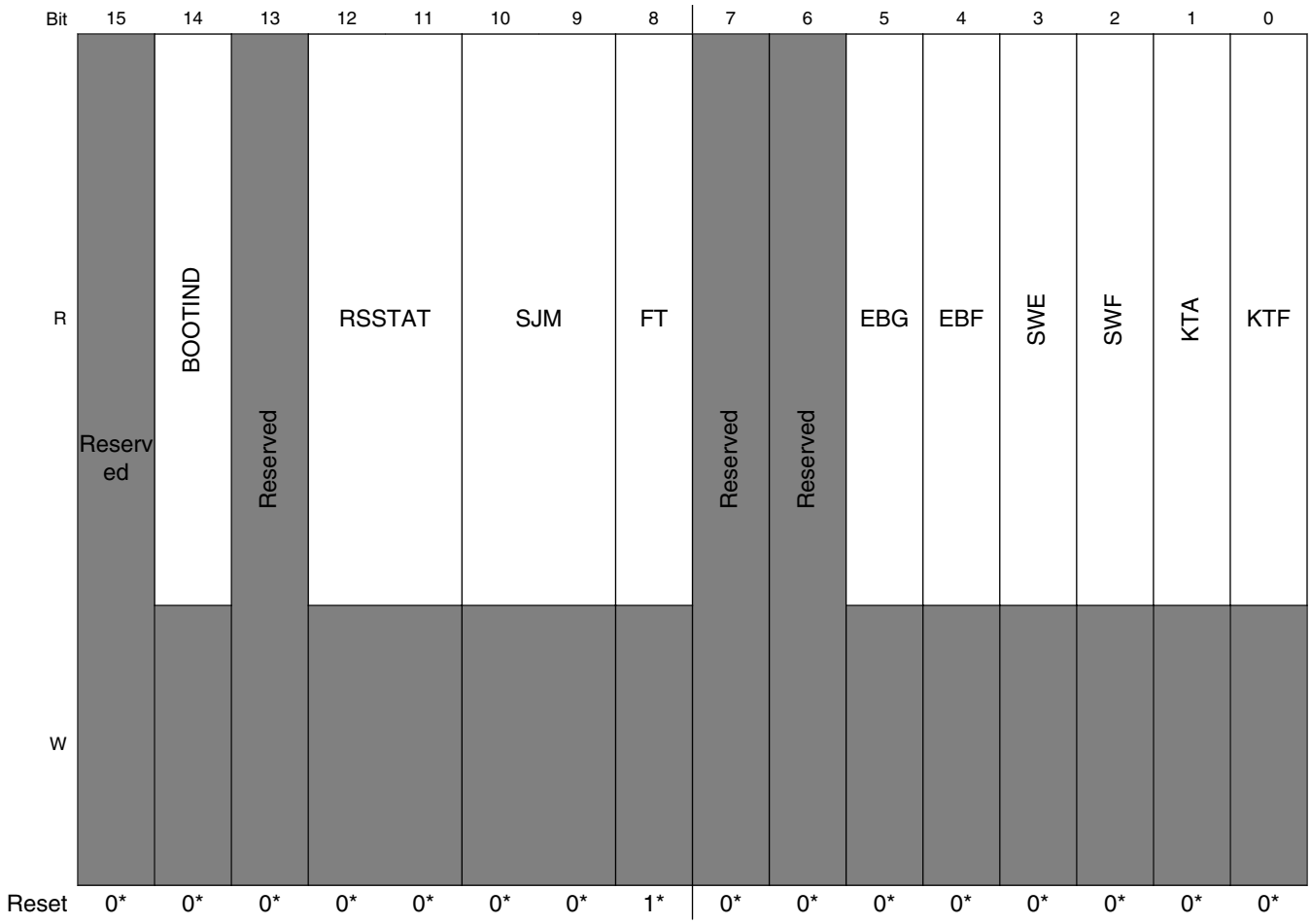
SJC_DCR field descriptions (continued)

Field	Description
3 DEBUG_OBS	Debug observability This bit controls the propagation of the "system debug" input to SJC (driven by the ECT logic), to the DE_B pad. (This logic can be used to pass debug acknowledge event from ECT out to the PAD, for example). The SJC's "system_debug" input is tied to logic HIGH value, therefore, set of "debug_obs" bit, will result in unconditional assertion of DE_B pad. 0 Disable propagation of system debug to DE_B pin 1 Unconditional assertion of pad DE_B
2 -	This field is reserved. Reserved
1 DE_TO_SDMA	SDMA debug request input propagation This bit controls the propagation of debug request to SDMA, when the JTAG state machine is put in "ENTER_DEBUG" IR instruction.. 0 Disable propagation of debug request to SDMA 1 Enable propagation of debug request to SDMA
0 DE_TO_ARM	ARM platform debug request input propagation This bit controls the propagation of debug request to ARM platform ("dbgreq"), when the JTAG state machine is put in "ENTER_DEBUG" IR instruction. 0 Disable propagation of debug request to ARM platform 1 Enable propagation of debug request to ARM platform

4.12.9.6 Security Status Register (SJC_SSR)

Address: 0h base + 5h offset = 5h





* Notes:
- The SJM reset value, reflects the JTAG security state, as defined by status of JTAG_SMODE[1:0] fuses. See the SJM bitfield description for details on valid values.

SJC_SSR field descriptions

Field	Description
31–17 -	Reserved.
16–15 -	This field is reserved. Reserved
14 BOOTIND	Boot Indication Inverted Internal Boot indication, i.e inverse of SRC: "src_int_boot" signal
13 -	This field is reserved. Reserved
12–11 RSSTAT	Response status Response status bits 00 Response wasn't entered 01 Response was entered but not verified

Table continues on the next page...

SJC_SSR field descriptions (continued)

Field	Description
	10 Response was entered and is incorrect 11 Response is correct
10–9 SJM	SJC Secure mode Secure JTAG mode, as set by external fuses. 00 No debug (#1) 01 Secure JTAG (#2) 10 Reserved 11 JTAG enabled (#3)
8 FT	Fuse type Fuse type bit - e-fuse or laser fuse 0 E-fuse technology 1 Laser fuse technology
7 -	This field is reserved. Reserved
6 -	This field is reserved. Reserved
5 EBG	External boot granted External boot enabled, requested and granted 1 granted 0 not granted
4 EBF	External Boot fuse Status of the external boot disable fuse 0 (intact) - external boot is allowed 1 (burned) - external boot is disabled
3 SWE	SW enable SW JTAG enable status 1 enabled 0 disabled
2 SWF	Software JTAG enable fuse Status of the no SW disable JTAG fuse 0 (intact) - SW enable possible 1 (intact) - no SW enable possible
1 KTA	Kill Trace is active 1 active 0 not active
0 KTF	Kill Trace Enable fuse value 0 (intact) - kill trace is never active 1 (burned) - kill trace functionality enabled

4.12.9.7 General Purpose Clocks Control Register (SJC_GPCCR)

This register is used to configure clock related modes in SOC, see System Configuration chapter for more information. Those bits are directly connected to JTAG outputs. Bit 0 of GPCCR controls SDMA clocks invocation. When out of reset, the SDMA is in sleep mode with no SDMA clock running. Unlike events, debug requests does not wake SDMA if it is in sleep mode. The debug request is recognized by the SDMA only when it exits sleep mode upon reception of an event. To be able to enter debug mode even if no event is triggered, the SDMA clock on bit needs to be set prior to sending the debug request (clear at reset).

Address: 0h base + 7h offset = 7h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-														ACLKOFFDIS	SCLKR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SJC_GPCCR field descriptions

Field	Description
31–2 -	Reserved
1 ACLKOFFDIS	Disable/prevent ARM platform clock/power shutdown
0 SCLKR	SDMA Clock ON Register - This bit forces the clock on of the SDMA

Chapter 5
Clocks and Power Management

5.1 Clock, Reset, and Power Management

5.1.1 Introduction

The chip targets many applications where low power consumption, long battery life, always-on and instant-on capabilities, and no need for active cooling are paramount. For this reason, the chip design constantly focuses on reducing current consumption as much as possible, while simultaneously enabling the maximum level of peak performance and a balanced level of sustained performance for target applications. To achieve this, the chip architecture uses a wide range of power-management techniques and their combinations for maximum system design flexibility.

This chapter describes the Clock and Power Management architecture of the chip. It contains information about:

- Structural components of the power and clock management systems
- Clock generation and distribution system
- Power generation and distribution system
- Power mode definition
- Power and clock and management techniques supported.

NOTE

All the numerical values are typical or examples, for accurate values one should use the datasheet.

5.1.2 Components of Clock and Power Management

The centralized clock generation, power generation and distribution are implemented by the following blocks:

- Crystal OSC (XTALOSC): The integrated crystal oscillators generate the 24MHz high speed clock and 32KHz low speed clock, these two clocks are used as the clock source for whole system. See Crystal Oscillator (XTALOSC) for details of the XTALOSC block.
- PLLs and PFDs: These modules generates the clocks with various frequencies required by different functional blocks. See the PLL and PFD section in Clock Controller Module (CCM)) for information on the PLL and PFD architecture, functional description and programming model.
- CCM (Clock Control Module): The CCM module provides control for clock generation, division, distribution, synchronization, and coarse-level gating. See Clock Controller Module (CCM)) for information on the CCM architecture, functional description and programming model.
- LPCG (Low Power Clock Gating): This module distributes the clocks to all blocks in the SoC and handles block level software-controllable and automated clock gating. See Clock Controller Module (CCM) for information on the LPCG architecture and functional description.
- GPC (General Power Controller): This module controls the power state of the SoC. It handles the power gating under low power modes and also manages the power up / power down sequences. See General Power Controller (GPC) chapter for information on the GPC architecture, functional description and programming model.
- SRC (System Reset Controller): This module generates the reset signals to all the modules in the SoC. It will assert the reset signals at appropriate time such as power up. See System Reset Controller (SRC) chapter for information on the GPC architecture, functional description and programming model.
- PMU (Integrated Power Management Unit): This module generates the internal power supplies distribute them to the entire SoC. See Power Management Unit (PMU) chapter for information on the PMU architecture, functional description and programming model.

Together, the modules listed above provide enhanced power-management features with the centralized control for the clock, reset, and power-management signals on the SoC.

5.1.3 Clock Generation

The clock system is designed to provide a scalable, powerful but easy-to-use clock solution.

5.1.3.1 Overview

The overall clock system for the chip is shown in the diagram below.

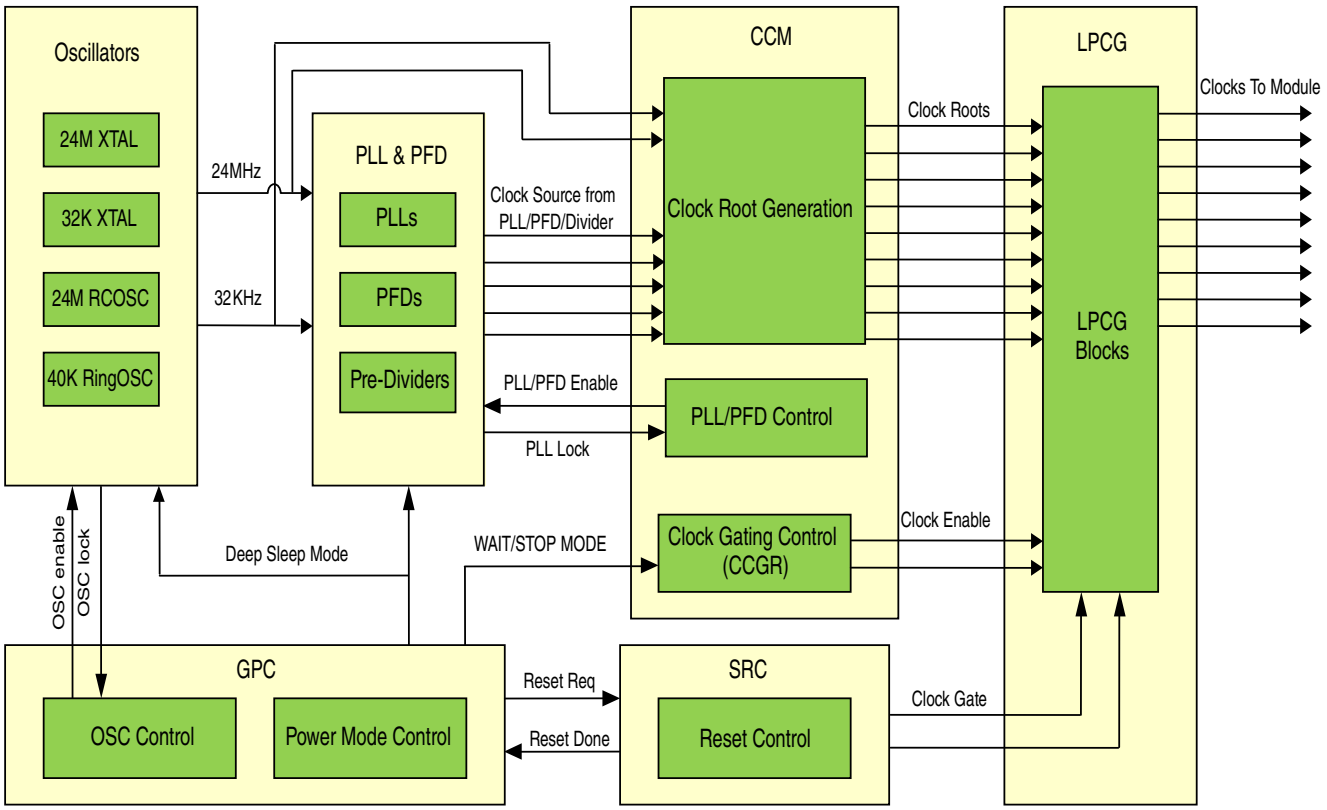


Figure 5-1. Clock System

As shown in the diagram, the OSCs, PLLs, PFDs and Pre-Dividers generate the clock sources with fixed or variable frequency. Then the clock root generation logic inside CCM will generate various clock roots required for core, bus and peripheral blocks. These clock roots will be delivered to each module through LPCG, which contains the clock gating logic for each clock. The control signal for CCGRs in CCM will be the source for clock enable signals. Since some of the clocks need to be gated off during the reset, the SRC module will also send clock gate signals to the LPCG.

In low power mode, GPC will drive the low power mode state signal to CCM, then CCM will enable clock gating based on the configuration. At the same time, CCM may also deassert enable signals for PLLs and PFDs, so these modules can be power off during the low power mode. When the chip enters stop mode, GPC may deassert the OSC enable signal to ANATOP to shut off the OSC, it may also assert the stop mode signal to ANATOP to put the whole ANATOP into low power mode.

The detailed function description for each block will be described in following sections:

5.1.3.2 Oscillator

The chip has 4 integrated oscillators, two for high speed 24MHz clock and two for low speed 32KHz clock.

- 24MHz Crystal Oscillator (XTAL): High frequency oscillator (typical frequency is 24 MHz) used to generate the main system clock. It supplies the PLLs and other peripherals. The 24MHz XTAL clock input can be connected to either an external oscillator or a crystal using the internal oscillator amplifier.
- 24MHz RC Oscillator (RCOSC): Low power RC oscillator used to generate 24MHz clock with less accuracy. It is intended to be used to replace the 24MHz XTAL in low power mode.
- 32KHz Crystal Oscillator (RTC_XTAL): Low frequency real time clock oscillator (typical frequency is 32.768KHz) used for low-frequency functions. It supplies the clock for wake-up circuit, power-down real time clock operation, and slow system and watchdog counters. The 32KHz RTC_XTAL clock input can be connected to either an external oscillator or a crystal using the internal oscillator amplifier.
- 40KHz Ring Oscillator (RingOSC): Internal ring oscillator (typical frequency is 40KHz) used to generate low speed clock when 32KHz RTC_XTAL is not available. It has less accuracy than the 32KHz RTC_XTAL but it starts up much faster.

Either the 24MHz XTAL or the 24MHz RCOSC can be used as the source of the 24MHz clock to CCM and then get delivered to the rest of the SoC. The 24MHz XTAL is selected by default on power up. The switching between 24MHz XTAL and 24MHz RCOSC is fully controlled by SW.

Either the 32KHz XTAL or the 40KHz RingOSC can be used as the source of the 32KHz clock to CCM and then get delivered to the rest of the SoC. Due to the 32KHz XTAL start up time, 40KHz RingOSC is selected as the default clock source during SoC power up. Once the 32KHz XTAL starts up, the clock source will be switched to it automatically and the 40KHz RingOSC will be shut off. If the 32KHz XTAL stops in some case, the 40KHz RingOSC will be enabled and selected as the source. The enable/disable of the 40KHz RingOSC and the clock source switching is done automatically in HW and there is no programmable register for SW to select the clock source. There is only one read-only register to show the current clock source.

When the SoC enters Deep Sleep Mode, the GPC will de-assert the osc_enable signal to shut off the 24MHz XTAL for power saving. When exiting from this mode, GPC will wait for the osc_lock signal to make sure the 24MHz clock source is ready.

See Crystal Oscillator (XTALOSC) chapter for details of the functional descriptions and programming model on these blocks.

5.1.3.3 Clock I/O Port

There are two dedicated IO ports on the chip used for clock generation:

- CLK1P/N: LVDS low jitter differential I/O ports for input or output clocks.
- CLK2: single-ended IO port for input or output clocks.

Both CLK1P/N and CLK2 can take input clocks from outside of the SoC and provide them to the PLLs or to the other modules, or they can take the outputs of the PLLs and provide them outside of the SoC as a functional or reference clock.

5.1.3.4 PLL and PFD

Six PLLs are used in the chip to generate the high speed clocks required by the modules in SoC. Among all the PLLs, the SYS_PLL is equipped with eight Phase Fractional Dividers (PFDs) in order to generate additional frequencies.

- ARM_PLL, used to generate the clock for ARM Cortex-A7 platform. It is a programmable integer frequency multiplier capable of output frequency of up to 1.2GHz.
- SYS_PLL with 8 PFDs, used to generate the clock sources for the whole system. By default, this PLL is a clock source for internal system buses, internal processing logic, NAND/NOR interface modules, etc. SYS_PLL generally runs at a fixed multiplier of 20, producing 480MHz output frequency with 24MHz reference from XTAL. Besides the main output, this PLL drives eight PFDs (PFD0...PFD7). The PFD0, PFD1 and PFD2 will be fixed at 392MHz, 332MHz and 270MHz. With the Pre-Dividers on these PLL/PFD, it can provide clocks with frequency at 480MHz, 392MHz, 332MHz, 270MHz, 196MHz, 166MHz and 135MHz. These clock sources will be used for AXI/AHB bus in the chip, and also be used for those IP blocks that does not require an accurate frequency. The PFD3-PFD7 will be used by some IP blocks that need a programmable clock.
- ENET_PLL, this PLL will have its VCO frequency fixed at 1GHz, the Pre-Dividers on this PLL will divide this clock down to 500MHz, 250MHz, 125MHz, 100MHz and etc. These clocks will be used for the IP blocks that requires a fixed and accurate clock, such as Ethernet, Timer, etc.
- AUDIO_PLL, used to generate reference clock for audio interface. This is a fractional multiplier PLL used for generating a low jitter and high precision audio clock with standardized audio frequencies. The PLLs oscillator frequency range is from 650MHz to 1300MHz, and the frequency resolution is better than 1Hz. This clock is mainly used as a clock for serial audio interfaces and as a reference clock for external audio codecs. It is equipped with a divider on its output and can generate divided by 1, 2, 4, 8 or 16 from the PLL VCO frequency.

- VIDEO_PLL, used to generate reference clock for display interface. This is a fractional multiplier PLL used for generating a low jitter and high precision video clock with standardized video frequencies. The PLLs oscillator frequency range is from 650MHz to 1300MHz, and the frequency resolution is better than 1Hz. This clock is mainly used as a clock for display and video interfaces. It is equipped with dividers on its output and can generate clock divided by 1, 2, 4, 8 or 16 from the PLL VCO frequency.
- DRAM_PLL, used to generate reference clock for DRAM PHY and controller.

Among all the PLLs, the AUDIO_PLL, VIDEO_PLL and DRAM_PLL has the capability to spread spectrum the generated signal. These do not require exact/constant frequency and can be changed as a part of dynamic frequency scaling procedure and/or can be spread-spectrum modulated.

Each PFD works independently by interpolating the VCO of the PLL to which it is connected. It effectively takes the PLL VCO frequency and produces 18/N x Fvco at its output where N ranges from 12 to 35. PFD is a completely digital design with no analog components or feedback loops. The frequency switch time is much faster than a PLL because keeping the base PLL locked and changing the integer N only changes the logical combination of the interpolated outputs of the VCO. Note that the PFD not only enables faster frequency changes than a PLL, but also allows the configuration to be safely changed "on-the-fly" without going through the output clock disabling/enabling process.

Reference input clock for any of the PLLs could be selected individually by the BYPASS_CLK_SRC feed of the PLL control register. Each PLL can use one of the following clock source as its reference clock:

- 24MHz clock from XTAL.
- External reference clock from CLK1P/N.
- External reference clock from CLK2.

A simple diagram for the PLLs, PFDs and Pre-Dividers are shown in the diagram below:

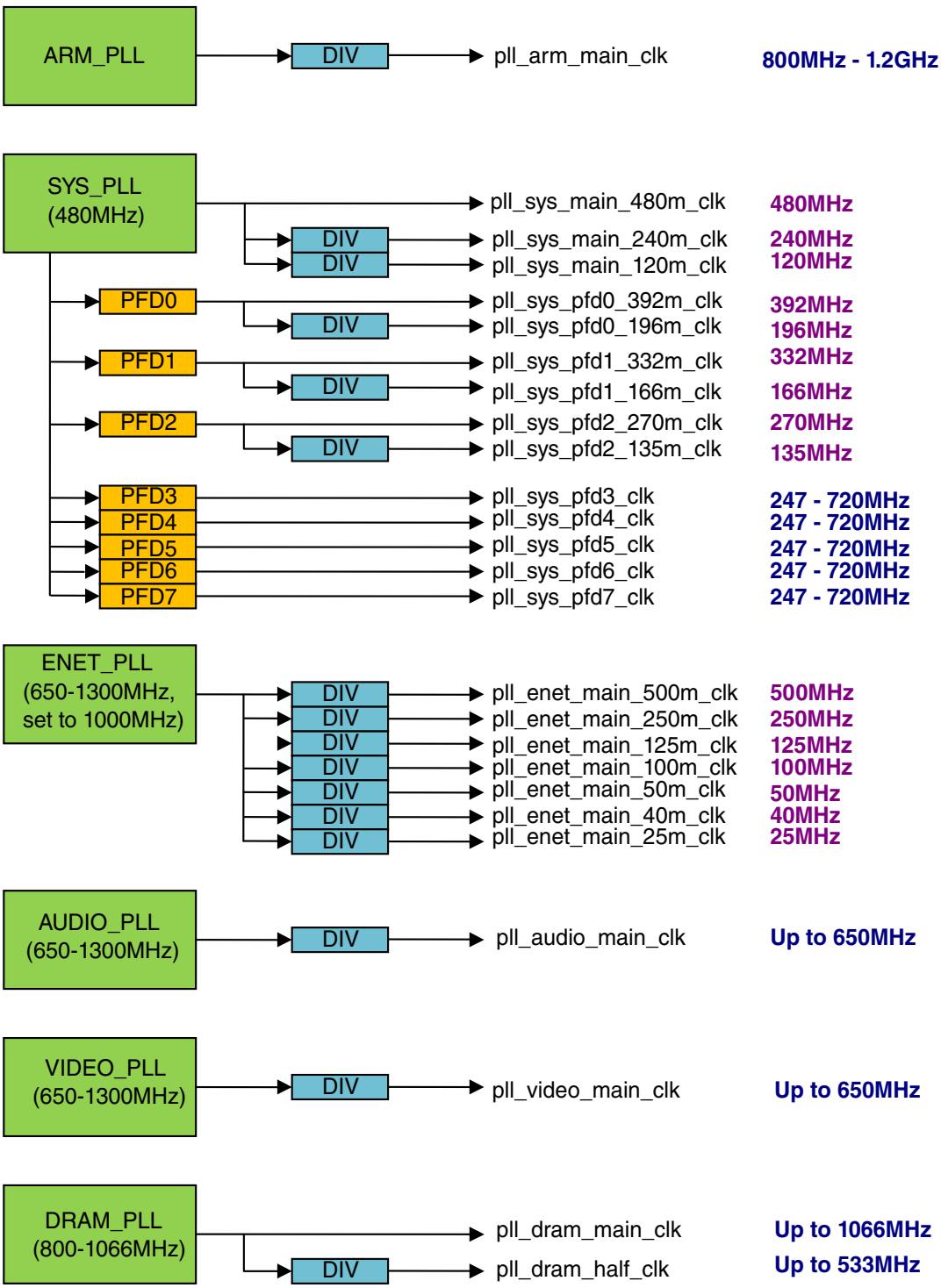


Figure 5-2. PLL and PFD

PLLs configuration and control functions are accessible via individual PLL and PFDs and global configuration and status registers.

Each of the PLLs could be individually configured to "Bypass", "Output Disabled" and "Power Down" modes.

- When configured in "Bypass" PLL pass directly its input reference clocks to the PLL output. Bypassing the PLL is done by setting the BYPASS bit in the control register. For the PLL equipped with PFDs the input reference clock is also bypassed to all PFDs outputs.
- When configured in "Output Disabled" mode, the PLLs output is completely gated and there is neither bypass clock nor PLL generated clock do propagated to PLL output. Main PLL output and PFD outputs have individual "Output Enable" control bits.
- When configured in "Power Down mode" most of the PLL circuitry is switched off. Neither main PLL output nor PFD outputs are available in this mode.

Individual PLL status is reflected in "PLL Lock" bits of the PLL control registers. PLL enable logic which monitors the register value change is implemented to gate off the PLL outputs during the "lock in" period. Outputs are generated to be sent out by monitoring the individual PLL lock flags and filtering out any random initial edges. For PFDs, if it is enabled, its output will be also gated during the “lock in” period.

In addition to the control register for each PLL/PFD, CCM will also have another group of registers dedicated for PLL and PFD enable/disable control in low power mode. Each PLL/PFD will have 2-bit control register as follows:

- 2b’00: the PLL/PFD is always disabled.
- 2’b01: the PLL/PFD will be disabled in STOP/WAIT mode.
- 2’b10: the PLL/PFD will be disabled in STOP mode.
- 2’b11: the PLL/PFD is always enabled.

Every PLL and PFD will have one hardware enable signal from CCM, and also provide one lock flag back to CCM. During the low power mode, GPC will send the WAIT/ STOP mode signal to CCM. CCM can deassert the enable signal to PLL/PFD to disable them in low power mode. When exiting from low power mode, CCM needs to assert the enable signals and wait for the lock flag to make sure the clock is stable, and then provide a lock flag back to GPC as an acknowledge.

CCM Memory Map/Register Definition and CCM Analog Memory Map/Register Definition contains detailed descriptions of the memory mapped registers and control functions of the clock generation sub-module.

5.1.3.5 Clock Root Generation

The clock root generation in CCM will be implemented based on clock slices. Each clock root will have a dedicated clock slice, it takes the clock source from Oscillators, PLL/PFD or Pre-Dividers, and generate the clock root with required frequency.

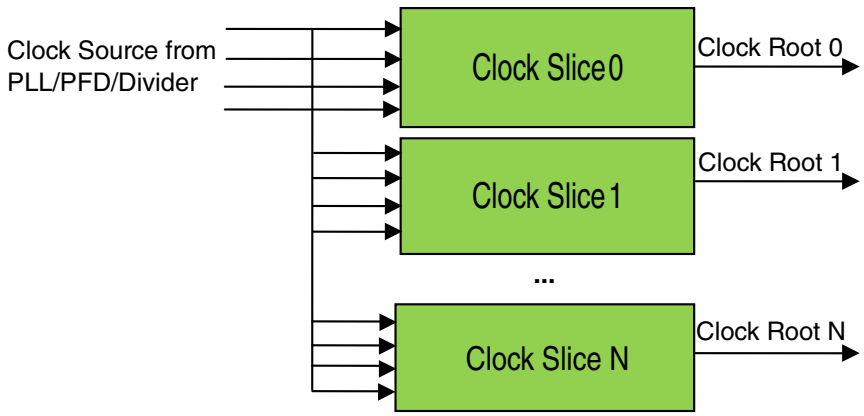


Figure 5-3. Clock Root Generation from Clock Slice

The full clock root definition, which includes the clock slice type, clock source, and default setting, please refer to CCM chapter.

Clock slice is the basic element of the root clock generation. It has unified clock input MUX, Pre-Divider, Post-Divider and glitch-free MUX to generate clock root from clock sources. There are 3 types of clock slices used in the CCM:

- Core clock slice, used for ARM CPU cores.
- Bus clock slice, used for AXI/AHB bus fabric and bus interface for IP modules.
- Peripheral clock slice, used for peripheral IP modules.

The 3 types of clock slice will use a unified programming model to make the SW programming easy-to-use. The control for the MUX, CG and dividers are provided with two 32-bit registers. For details on the programming model of the clock slices, please refer to CCM Memory Map/Register Definition.

5.1.3.5.1 Bus Clock Slice

From function point of view, the bus clock slice is a super-set for the core clock slice and the peripheral clock slice. The bus clock supports 8 functional clock inputs, which are fed into two source select MUX, named SEL_A and SEL_B. The selected clock source will be divided down by an integer pre-divider. To avoid glitches to the pre-dividers when switching clock source on the MUX, there is a clock gating cell used to shut off the clock. The output from the two pre-dividers is selected using a glitch-less MUX. In functional mode, the output of the glitch free MUX will be divided down by the post-divider and used as clock root.

The upper and lower part of the MUX, CG and pre-divider are identical. The duplicated source select MUX and pre-divider are mainly designed to support seamless clock source switching, in order to avoid stop clock. For example, if current clock source is clk0 on SEL_A, and SW want to change clock source from clk0 to clk1, it can configure SEL_B to use clk1 and use the glitch free MUX to switch it from SEL_A to SEL_B.

For bus clock, the clock frequency change needs to happen when the clock is still being used. The bus clock slice provided a mechanism to do handshake with IP modules through hardware signals. When the glitch free MUX or the post-divider is changed, the handshake will start and the change will only be held until an acknowledge is provided. The handshake function can be enabled/disable by register setting.

The AUTO-SLOW block on the final output path is used to automatically divide clock down for power saving based on the idle flag. For a bus block, the idle flag can be asserted when all the active transactions have finished, and be deasserted when there is a new transaction request.

There is also a bypass MUX before the post-divider to allow a bypass clock to be used instead of the 8 functional clock input. This is used during ATE test to take external clock input.

The key function of bus clock slice is shown in the diagram below:

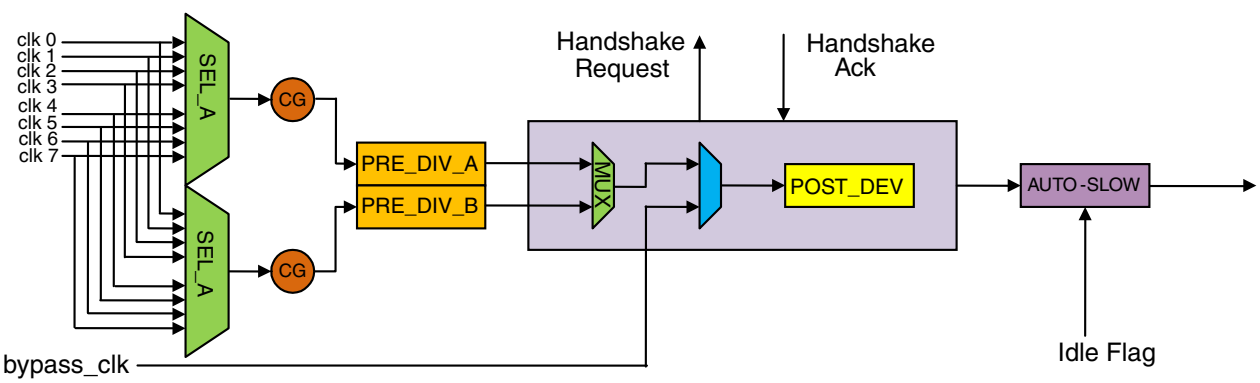


Figure 5-4. Bus Clock Slice

5.1.3.5.2 Core Clock Slice

The core clock slice is a sub-set of the bus clock slice. Since the core clock runs up to 1GHz, the timing of core clock slice needs to be closed at higher frequency.

The key function of core clock slice is shown in the diagram below.

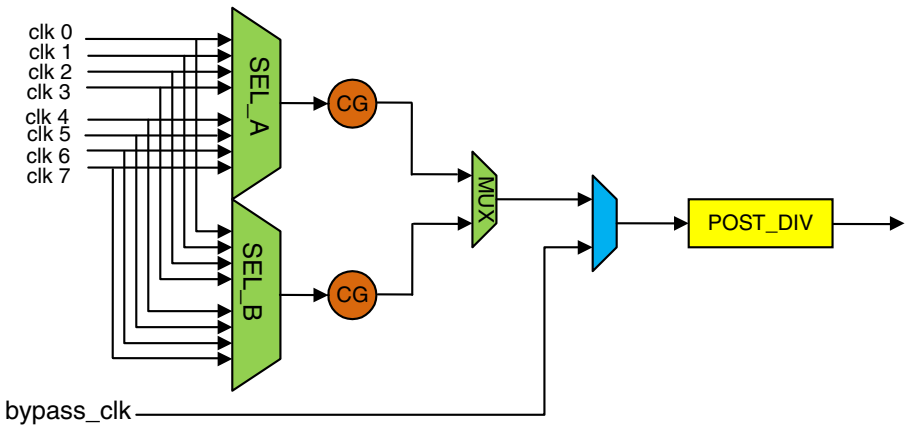


Figure 5-5. Core Clock Slice

5.1.3.5.3 Peripheral Clock Slice

For peripheral clocks, the clock frequency won't be changed when the clock is being used, so there is only one clock source select MUX, one pre-divider and one post divider. Most of the clock slices in CCM are peripheral clock slice.

The key function of peripheral clock slice is shown in the diagram below.

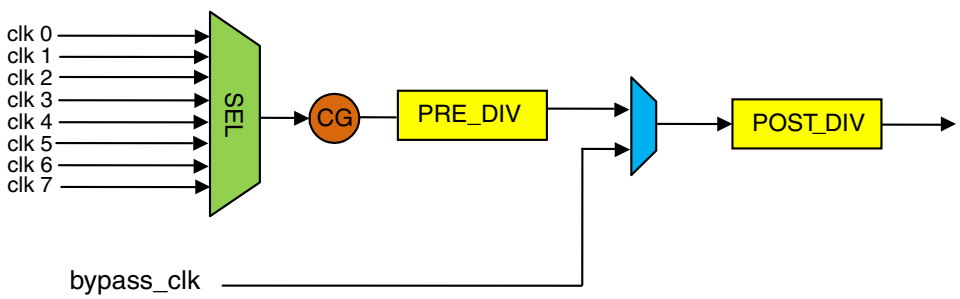


Figure 5-6. Peripheral Clock Slice

5.1.3.6 Low Power Clock Gating (LPCG)

The clock roots generated inside CCM are distributed to the entire SoC through LPCG. The LPCG block receives the root clocks from CCM and splits them to clock branches for each functional block. The clock branches are individually gated clocks. Each LPCG only has one clock root, while each clock root might drive multiple LPCGs.

The clock enables for the LPCG can come from following sources:

- Clock enable signal from CCM - This signal is generated depending on the power mode the system is in. For each power mode, it is defined in the software using the configuration of the CCGR bits in CCM.
- Clock enable signal from the block - This signal is generated by the block based on its internal logic. Not every enable signal from the block is used. Each clock enable signal from the block can be overridden based on the programmable bit in CCM.
- Clock enable signal from the reset controller (SRC) - This signal will enable the clock during the reset procedure.

For the clock enable signal from CCM, the 2-bit CG control register field for each clock is defined as:

- 2b'00: the clock is always gated off.
- 2'b01: the clock will be gated off in STOP/WAIT mode.
- 2'b10: the clock will be gated off in STOP mode.
- 2'b11: the clock is always on.

By proper setting of the CG control registers, the clocks to each functional block could be shut off automatically when chip enters low power mode.

5.1.3.7 DRAM Clock Generation

The DRAM PHY and DRAM controller in the chip requires following clock input:

- PHY_CLK, up to 533MHz, its frequency is the operating frequency of the DRAM interface.
- CONTROLLER_CLK, synchronous to PHY_CLK, always at ½ frequency of PHY_CLK.
- PHY_MCLK, 400MHz to 533MHz, asynchronous to PHY_CLK, but has to be at least 400MHz, no matter what the DRAM frequency is.

To meet all these requirement, following clock structure is used:

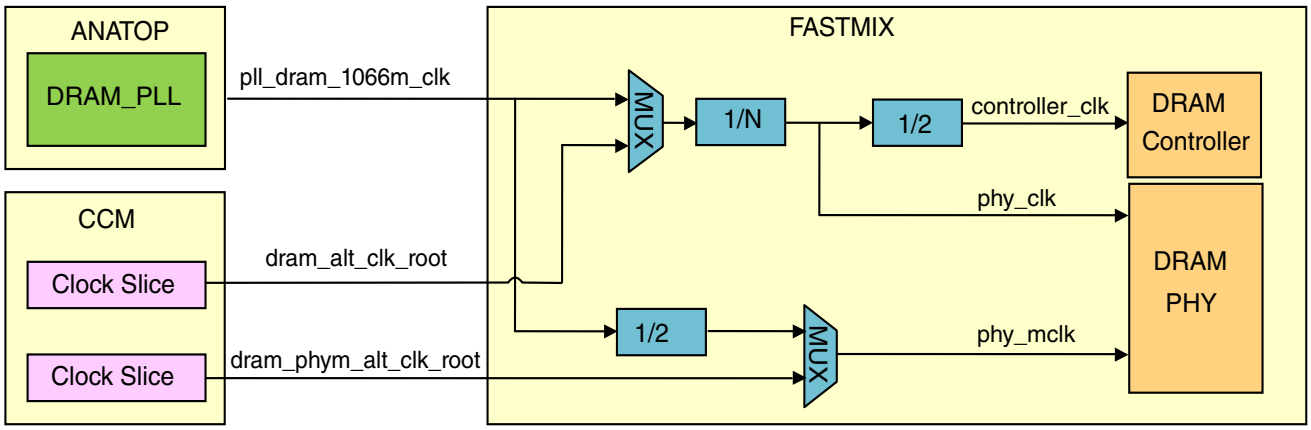


Figure 5-7. DRAM Clock Structure

The dedicate DRAM_PLL is used to generate 2x clock at 1066MHz, and use a divider to divide it by 2 to get 533MHz clock with good duty cycle. This 533MHz clock will be used as the PHY_MCLK. Meanwhile, the 1066MHz clock will also be divided by 2 with the 1/N divider to get 533MHz as the PHY_CLK, and then it will be divided by 2 to get 266MHz clock for the DRAM controller.

For DRAM low speed operations, there is a 1/N divider used to divide the PHY clock down to lower frequency such as 266MHz or 133MHz. The 1/N divider is a 3-bit divider so N can be 2 to 8. The PHY_MCLK will still use the original 533MHz since it has to be 400MHz or higher.

The alternative solution to support DRAM low speed operation is to use the clock root generated from CCM. The dram_phym_alt_clk_root from CCM will be used as PHY_MCLK, and it will be running at 400MHz or above. The dram_alt_clk_root from CCM will be divided by N as the PHY_CLK and also get divided by 2 to be the clock from controller. The advantage of doing this to allow the DRAM _PLL to be shut off to save more power.

5.1.4 Power Management

5.1.4.1 Overview

A high level block diagram of the power management system in the SoC environment is shown in the figure below.

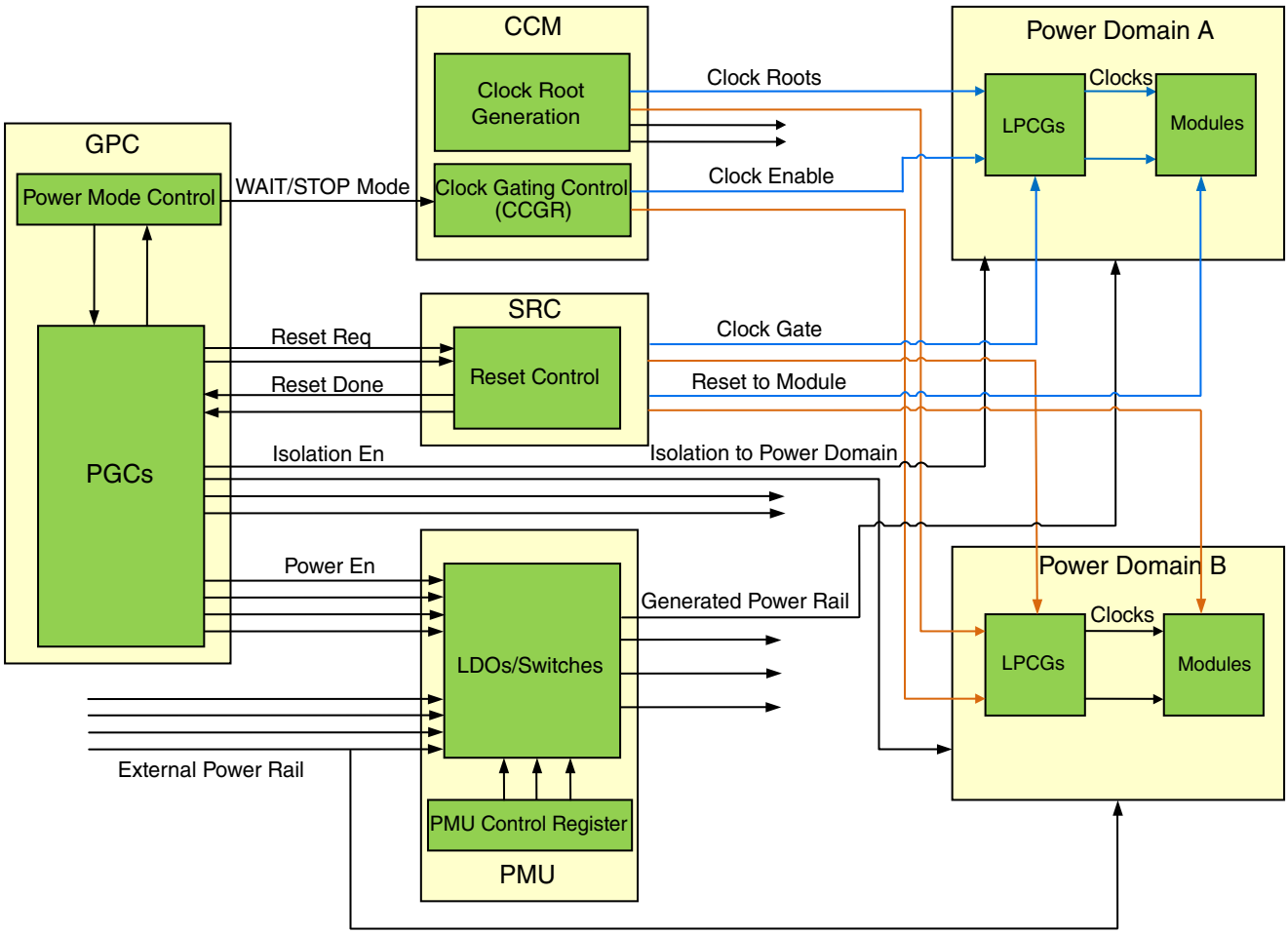


Figure 5-8. Power System

The power generation and management system consists of the GPC, PMU, CCM and SRC.

- The PMU generates the power rails with LDOs or power switches from external power input.
- The GPC is the central controller for all the power system. The key roles of the GPC are chip power mode control and power domains management.
- The CCM generates the clock for modules in each power domain, and also enable/disable the clocks based on the power mode from GPC.
- The SRC generates the reset signal for each power domain during the power up of a domain.

5.1.4.2 PMU

The integrated PMU is designed to simplify the external power interface. It help to reduce the number of external power supplies required, and also manages the power up/ power down sequence for integrated PHY and IO pads.

5.1.4.2.1 PMU Components

The PMU has the following components integrated for power management:

- LDOs
- Power Switches
- PVCC Generator

There are 6 integrated LDO on the chip:

- LDO_1P0A, 1.0V LDOs for analog modules, including XTAL and PLL.
- LDO_1P1D, 1.0V LDO for MIPI PHY.
- LDO_1P2, 1.2V LDO for USB HSIC PHY.
- LDO_USB1_1P0, 1.0V LDO for USB PHY1.
- LDO_SVNS_1P8, 1.8V LDO from coin cell to generate 1.8V power for SNVS and 32K RTC.

There are 8 integrated power switches on the chip:

- SW_CPU0, Power switch for ARM Cortex-A7 CPU0.
- SW_CPU1, Power switch for ARM Cortex-A7 CPU1.
- SW_SCU, Power switch for ARM Cortex-A7 Platform (SCU).
- SW_L2, Power switch for ARM Cortex-A7 L2 Cache memory.
- SW_SOC_PD, Power switch for the power down domain in VDD_SOC.
- SW_FUSE, Power switch for the eFuse programming power supply.
- SW_HSIC_1P0, Power switch for the USB HSIC PHY 1.0V power supply.
- SW_PHY_1P8, Power switch for MIPI PHY 1.8V power supply.

The GPIO pads require an 1.8V PVCC supply for its pre-driver circuit. In order to simplify the power up/power down sequence, each GPIO bank has a dedicated bias generator circuit to generate PVCC from NVCC. There are a total of 13 PVCC Generators on the chip.

NOTE

Due to the pad limitation on the package, the PVCC power generated from different GPIO bank are shorted together into 5 groups.

- NVCC_GPIO1/2 banks share on PVCC supply, named as PVCC_GPIO_1P8_CAP.
- NVCC_ENET1 bank takes on dedicated PVCC supply, named as PVCC_ENET_CAP.
- NVCC_SAI, NVCC_SD1/2/3 banks share one PVCC supply, named as PVCC_SAI_SD_CAP.
- NVCC_I2C, NVCC_SPI, NVCC_UART banks share one PVCC supply, named as PVCC_I2C_SPI_UART_CAP.
- NVCC_LCD banks share one PVCC supply, named as PVCC_EPDC_LCD_CAP.

5.1.4.2.2 Power Distribution

The block diagram below shows the power distribution tree of the chip, including all the LDOs, power switches and PVCC generators.

The power rails on the left side are external power rails supplied to the SoC, while the power rails on the right side are generated by the integrated PMU.

Some of the external power rails are combined together to reduce the total number of external power supplies. For more details on the rules and recommendation on power rails, please refer to “Manage Power Rails and Power Domains”.

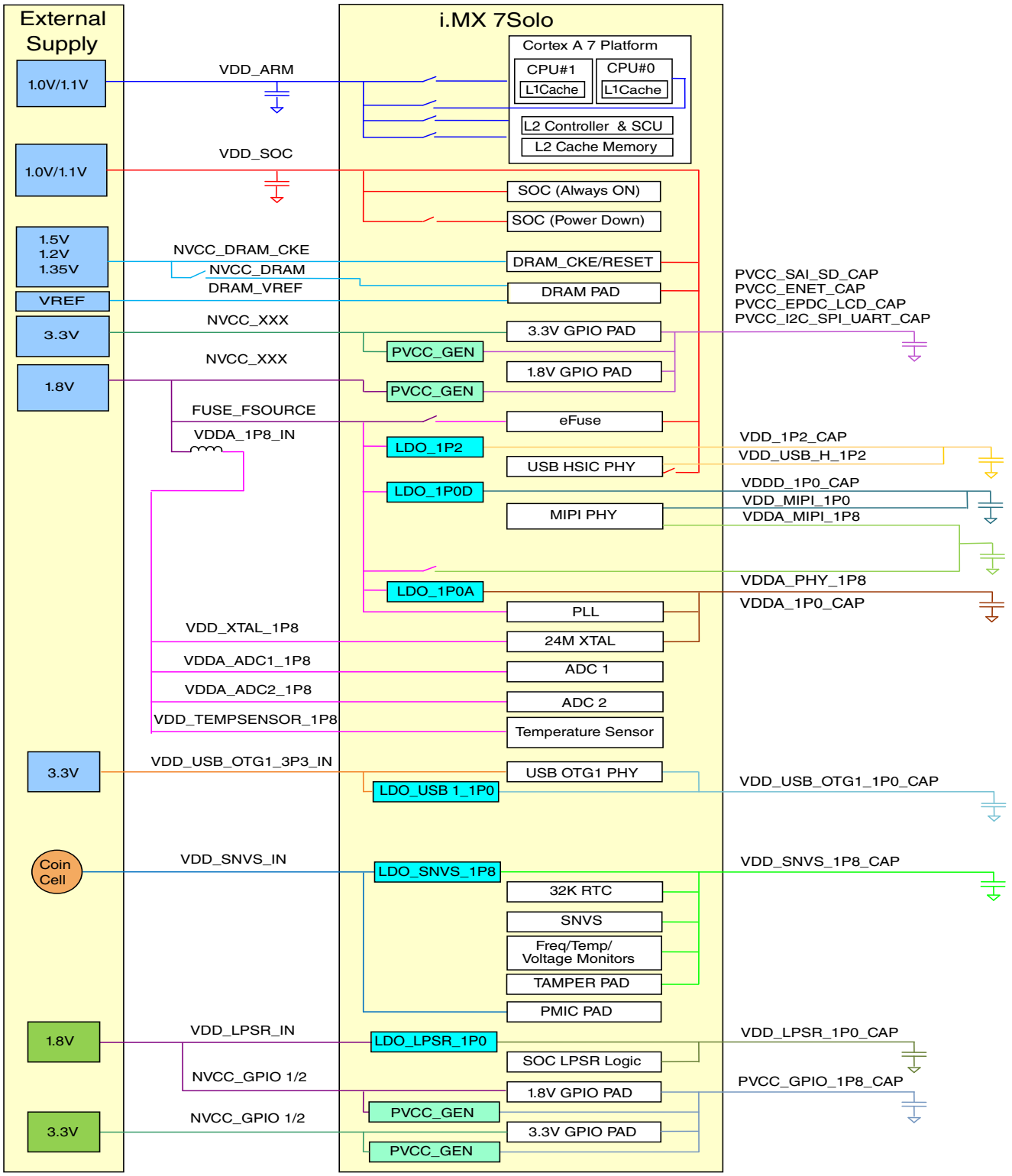


Figure 5-9. Power Diagram

5.1.4.2.3 External Power Supplies

The table below summarizes all the external power rails supplied to the SoC.

Table 5-1. External Power Supply

Power rail	Voltage(V)	Description
VDD_ARM	1.0/1.1	Power supply for ARM A7 platform.
VDD_SOC	1.0/1.1	Power supply for all SOC digital logic except ARM A7
VDDA_1P8	1.8	Power supply for analog modules, also used as the input power source for internal analog LDOs
VDDA_XTAL_1P8	1.8	Power supply for the analog portion of the 24MHz XTAL.
VDDA_ADC1_1P8	1.8	Power supply for the analog portion of the ADC1.
VDDA_ADC2_1P8	1.8	Power supply for the analog portion of the ADC2.
VDDA_TEMPSENSOR_1P8	1.8	Power supply for the analog portion of the Temperature Sensor.
FUSE_FSOURCE	1.8	Power supply for the programming power of FUSE box.
VDD_SNVS_IN	3.0	Power supply for RTC and tamper detection logic, usually supplied by coin cell
VDD_LPSR_IN	1.8	Power supply for low power state retention domain
NVCC_GPIO1	1.8/3.3	Power supply for GPIO bank1 in state retention domain
NVCC_GPIO2	1.8/3.3	Power supply for GPIO bank2 in state retention domain
NVCC_DRAM	1.2/1.35/1.5	DRAM IO power supply 1.2V for LPDDR2/LPDDR3, 1.35V for DDR3L, 1.5V for DDR3
NVCC_DRAM_CKE	1.2/1.35/1.5	DRAM IO power supply for CKE and RESET pad
DRAM_VREF	0.6/0.675/0.75	DRAM IO reference voltage
NVCC_ENET1 NVCC_I2C NVCC_SPI NVCC_UART NVCC_SAI NVCC_LCD NVCC_SD1/2/3	1.8/2.5/3.3	Power supply for GPIO banks outside of state retention domain (Referred as NVCC_XXX in this document)
VDD_USB_OTG1_3P3_IN	3.3	Power supply for USB OTG1 PHY

5.1.4.2.4 Generated Power Supplies

The table below summarizes the power rails generated by integrated PMU

Table 5-2. Generated Power Supply

Power rail	Vtyp	Source	Def	Description
VDDD_1P0_CAP	1.0	LDO_1P0D	OFF	
VDDA_1P0_CAP	1.0	LDO_1P0A	ON	
VDD_1P2_CAP	1.2	LDO_1P2	OFF	
VDD_USB_OTG1_1P0_CAP	1.0	LDO_USB1_1P0	ON	On when VDD_USB_OTG1_3P3_IN is active.
VDD_SNVS_1P8_CAP	1.8	LDO_SNVS_1P8	ON	On when VDD_SNVS_IN active.
VDD_LPSR_1P0_CAP	1.0	LDO_LPSR_1P0	ON	On when VDD_LPSR_IN active.
VDDA_1P8_PHY	1.8	SW_PHY_1P8	OFF	
VDD_ARM_CPU0	1.0	SW_CPU0	ON	Not visible on pads
VDD_ARM_CPU1	1.0	SW_CPU1	ON	Not visible on pads
VDD_ARM_SCU	1.0	SW_SCU	ON	Not visible on pads
VDD_ARM_L2	1.0	SW_L2	ON	Not visible on pads
VDD_SOC_PD	1.0	SW_SOC_PD	ON	Not visible on pads
VDD_USB_H_1P0	1.0	SW_HSIC_1P0	OFF	Not visible on pads
VDD_FUSE	1.8	SW_FUSE	OFF	Not visible on pads
PVCC_GPIO_1P8_CAP	1.8	PVCC_GEN_GPIO1 PVCC_GEN_GPIO2	ON	On when any of the NVCC_GPIO1/2 is active.
PVCC_ENET_CAP	1.8	PVCC_GEN_ENET	ON	On when NVCC_ENET is active.
PVCC_EPDC_LCD_CAP	1.8	PVCC_GEN_LCD	ON	On when NVCC_LCD is active.
PVCC_SAI_SD_CAP	1.8	PVCC_GEN_SAI PVCC_GEN_SD1 PVCC_GEN_SD2 PVCC_GEN_SD3	ON	On when any of the NVCC_SAI or NVCC_SD1/2/3 is active.
PVCC_I2C_SPI_UART_CAP	1.8	PVCC_GEN_I2C PVCC_GEN_SPI PVCC_GEN_UART	ON	On when any of the NVCC_I2C, NVCC_SPI, or NVCC_UART is active.

5.1.4.3 Power Domains

A power domain is a group of blocks or sub-blocks fed by the same power source. The whole chip is built up with a number of power domains, each power domain has one or multiple functional blocks. Each domain has its own power supply, either from external power directly or generated by integrated PMU.

According to the functional blocks inside them, the power domains on the chip can be divided into 3 groups: digital domain, analog domain and IO domain. The domains are named with the same name as its power supply.

The table below summarizes all the power domains on the chip.

Domain	Type	Blocks Inside the Domain
VDD_ARM_CPU0	Digital	ARM Cortex-A7 Core0
VDD_ARM_CPU1	Digital	ARM Cortex-A7 Core1
VDD_ARM_SCU	Digital	ARM Cortex-A7 SCU & L2 Controller
VDD_ARM_L2	Digital	ARM Cortex-A7 L2 memory
VDD_SOC	Digital	See the diagram below
VDD_SOC_PD	Digital	See the diagram below
VDD_LPSR_1P0_CAP	Digital	GPIO-1 controller
VDD_SNVS_IN	IO	PMIC_ON_REQ, PMIC_STBY_REQ, ONOFF pad
VDD_SNVS_1P8_CAP	Analog	SNVS_LP, tamper detection, 32K RTC
VDDA_1P8_IN	Analog	XTAL, ADC1/2, Temperature Sensor, PLL, bandbap
VDD_1P0A_CAP	Digital	XTAL, ADC1/2, Temperature Sensor, PLL
VDD_FUSE	Analog	FUSE
VDD_MIPI_1P0	Digital	MIPI PHY
VDDA_MIPI_1P8	Analog	MIPI PHY
USB_OTG1_3P 3	Analog	USB OTG1 PHY
USB_OTG1_1P0	Digital	USB OTG1 PHY
VDD_USB_H_1P2	Analog	USB HSIC PHY
VDD_USB_H_1P0	Digital	USB HSIC PHY
NVCC_DRAM	IO	DRAM IO except DRAM_CKE and DRAM_RESET
NVCC_DRAM_CKE	IO	DRAM_RESET and DRAM CKE
NVCC_GPIO1	IO	GPIO in NVCC_GPIO1 bank
NVCC_GPIO2	IO	GPIO in NVCC_GPIO2 bank
PVCC_GPIO_1P8_CAP	IO	GPIO in NVCC_GPIO1 and NVCC_GPIO2 bank
NVCC_ENET1	IO	GPIO in NVCC_ENET1 bank
NVCC_I2C	IO	GPIO in NVCC_I2C bank
NVCC_SPI	IO	GPIO in NVCC_SPI bank
NVCC_UART	IO	GPIO in NVCC_UART bank
NVCC_SAI	IO	GPIO in NVCC_SAI bank
NVCC_LCD	IO	GPIO in NVCC_LCD bank
NVCC_SD1	IO	GPIO in NVCC_SD1 bank
NVCC_SD2	IO	GPIO in NVCC_SD2 bank
NVCC_SD3	IO	GPIO in NVCC_SD3 bank

Table continues on the next page...

Domain	Type	Blocks Inside the Domain
PVCC_EPDC_LCD_CAP	IO	GPIO in NVCC_LCD bank
PVCC_ENET_CAP	IO	GPIO in NVCC_ENET1 bank
PVCC_SAI_SD_CAP	IO	GPIO in NVCC_SAI, NVCC_SD1, NVCC_SD2 and NVCC_SD3 bank
PVCC_I2C_SPI_UART_CAP	IO	GPIO in NVCC_I2C, NVCC_SPI and NVCC_UART bank

The diagram below shows the functional block in the main power domains.

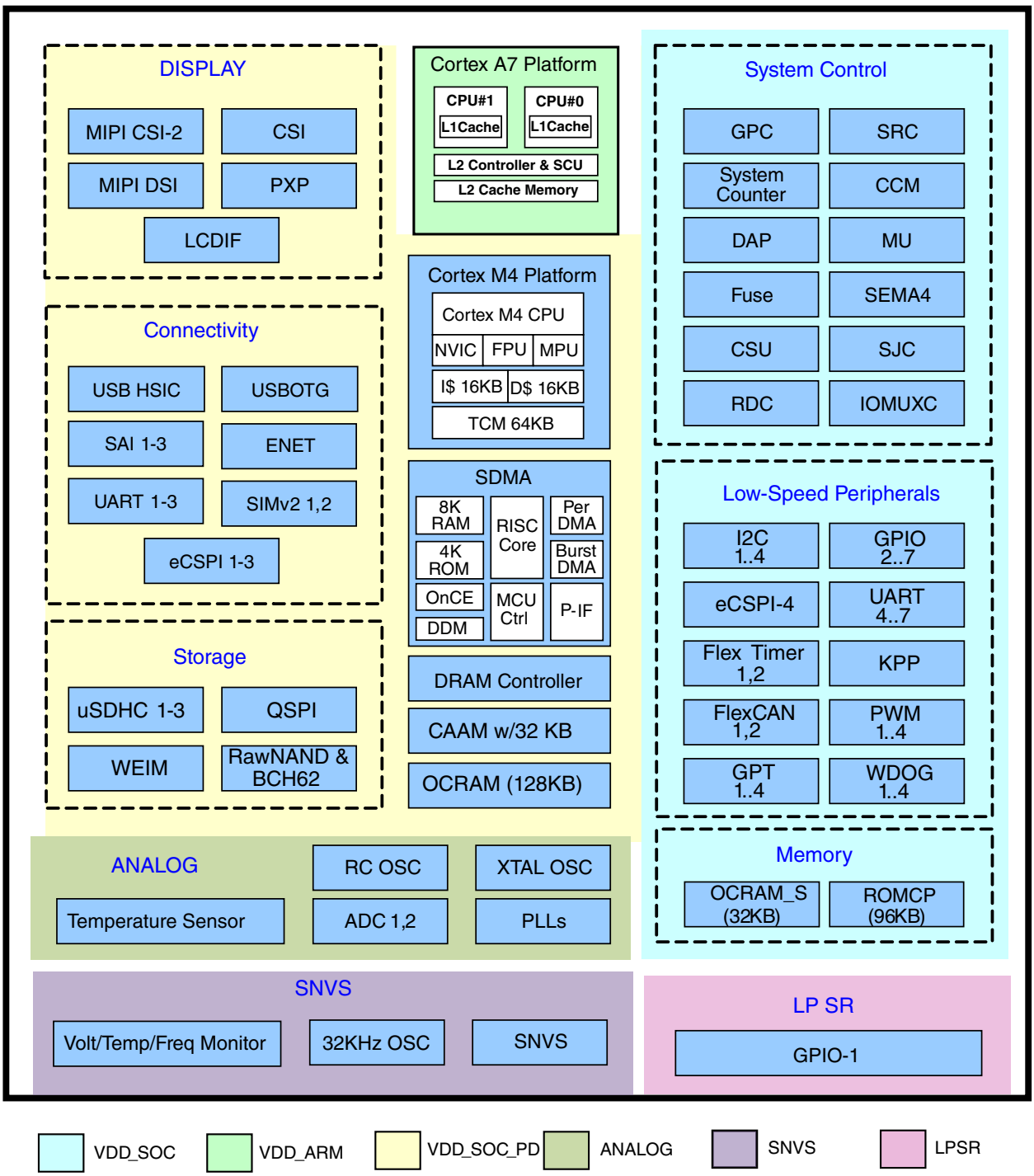


Figure 5-10. Power Domains

NOTE

The integrated PHYs are not shown in the diagram. The USB, MIPI mentioned in the diagram is only the digital controller of these modules.

5.1.4.4 Power Modes

The chip has following power modes:

- OFF Mode: all power rails are off.
- SNVS Mode: only RTC and tamper detection logic is active.
- LPSR Mode: an extension of SNVS mode, with 16 GPIOs in low power state retention mode.
- RUN Mode: all external power rails are on, CPU is active and running, other internal module can be on/off based on application.
- Low Power Mode: most external power rails are still on, CPU is in WFI state or power gated, most of the internal modules are clock gated or power gated. Typically there are 3 low power modes used, System IDLE, Low Power IDLE and SUSPEND.

The valid power mode transition is shown in the diagram below:

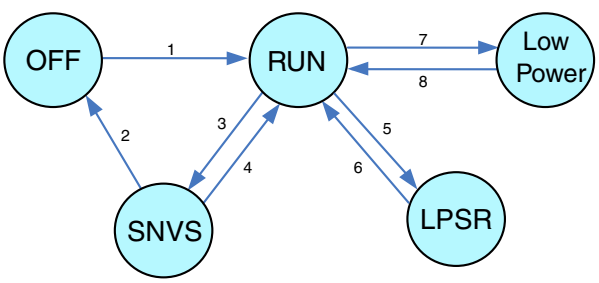


Figure 5-11. Power Modes

The power mode transition condition is defined in the table below:

Table 5-3. Power Mode Transition

Transition	From	To	Condition
1	OFF	RUN	VDD_SVNS_IN supply present.
2	SNVS	OFF	VDD_SNVS_IN supply removal.
3	RUN	SNVS	ONOFF long press, or SW.
4	SNVS	RUN	ONOFF press, or RTC, or tamper event.
5	RUN	LPSR	SW.
6	LPSR	RUN	ONOFF press, or RTC, or tamper event, or GPIO event.
7	RUN	Low Power	SW (CPU execute WFI)
8	Low Power	RUN	RTC, tamper event, IRQ.

The table below summarizes the external power supply state in all the power modes:

Table 5-4. Power Mode

Power rail	OFF	SVNS	LPSR	RUN	Lower Power
VDD_ARM	OFF	OFF	OFF	ON	ON/OFF
VDD_SOC	OFF	OFF	OFF	ON	ON
VDDA_1P8_IN	OFF	OFF	OFF	ON	ON
VDD_SNVS_IN	OFF	ON	ON	ON	ON
VDD_LPSR_IN	OFF	OFF	ON	ON	ON
NVCC_GPIO1/2	OFF	OFF	ON	ON	ON
NVCC_DRAM	OFF	OFF	OFF	ON	ON
NVCC_DRAM_CKE	OFF	OFF/ON	OFF/ON	ON	ON
NVCC_XXX	OFF	OFF	OFF	ON/OFF	ON/OFF
VDD_USB_OTG1_3P3_IN	OFF / ON	OFF / ON	OFF / ON	ON / OFF	ON / OFF

The NVCC_DRAM_CKE can be still ON during SNVS/LPSR mode to keep the CKE/RESET pad in correct state to hold DRAM device in self-refresh mode.

The NVCC_XXX can be off in RUN mode / Low Power mode if all the pads in that IO bank is not used in the application, the NVCC_XXX supply could be tied to GND.

The VDD_USB_OTG1_3P3_IN is fully asynchronous to other power rail, so it can be either ON/OFF in any of the power modes.

5.1.4.4.1 OFF Mode

In OFF mode, all the power rails are shut off.

5.1.4.4.2 SNVS Mode

SNVS mode is also called RTC mode, where only the power for the SNVS domain remain on. In this mode, only the RTC and tamper detection logic is still active.

The external DRAM device can keep in self-refresh when the chip stays in SNVS mode with NVCC_DRAM_CKE still powered. During the state transition between SNVS mode to/from ON mode, the DRAM_CKE pad and DRAM_RESET pad has to always stay in correct state to keep DRAM in self-refresh mode. No glitch / floating is allowed.

5.1.4.4.3 LPSR Mode

LPSR is considered as an extension of the SNVS mode. All the features supported in SNVS mode is also supported in LPSR mode, including the capability of keeping DRAM device in self-refresh.

In LPSR mode, three additional power rails will remain on: the VDD_LPSR_IN, NVCC_GPIO1 and NVCC_GPIO2. These two power rails are used to supply the logic and IO pads in the LPSR domain. The purpose of this mode is to retain the state of 16 GPIO pads, so the other components in the whole system will have their control signal in correct state.

Among all the 16 GPIO pads, the NVCC_GPIO1 supply the power for 8 GPIO pads, and the NVCC_GPIO2 supply the power for the other 8 GPIO pads. This allows the SoC to have some of its GPIO working at 1.8V while others working at 3.3V in the LPSR mode.

When LPSR mode is not needed for the application, the VDD_LPSR can be connected to VDDA_1P8 and NVCC_GPIO1/2 can be connected to the same power supply as NVCC_XXX for other GPIO banks.

In LPSR mode, the supported wakeup source are RTC alarm, ONOFF event, security/ tamper and also the 16 GPIO pads.

5.1.4.4.4 RUN Mode

In RUN mode, CPU is active and running, and the analog / digital peripheral modules inside the processor will be enabled. In this mode, the VDD_SOC, VDD_ARM, VDDA_1P8, VDD_LPSR_IN, VDD_SNVS_IN, NVCC_DRAM, NVCC_DRAM_CKE and NVCC_GPIO1 has to be ON. The other power rails can be ON/OFF based on applications.

Some of the blocks such as USB OTG, USB HSIC, MIPI can be shut off in the RUN mode if they are not active.

In this mode, the PMIC should allow SoC to change voltage of power rails through I2C/SPI interface. Typically, when CPU is doing DVFS, it will switch the VDD_ARM voltage between 1.1V and 1.0V when CPU is frequency is switching between 1GHz and 800MHz(or below).

5.1.4.4.5 Low Power Mode

When the CPU is not running, the processor can enter low power mode. The chip supports a very flexible set of power mode configurations in low power mode.

Typically there are 3 low power modes used, System IDLE, Low Power IDLE and SUSPEND:

- System IDLE - This mode is defined as a mode which CPU can automatically enter when there is no thread running. All the peripheral can still keep working and the CPU will have its state retained so the interrupt response can be very short. The cores shall be able to individually enter the WAIT state.
- Low Power IDLE - This mode is defined for the case when the system wants to have lower power but still needs to keep some of the peripheral alive. Most of the peripherals and analog module and PHYs are shut off. The interrupt response in this mode is expected to be longer than the System IDLE, but its power would be much lower.
- Suspend - This mode is defined as the most power saving mode where all the clocks are off and all the unused analog/PHY and peripherals are off. The external DRAM will stay in self-refresh mode. The exit time from this mode will be much longer.

In System IDLE and Low Power IDLE mode, the voltage on external power supplies remain the same as RUN mode, so the external PMIC is not aware of the state of the processor. If there is any low power setting need to be applied to PMIC, it will be done through I2C/SPI interface before the processor enters low power mode.

When the processor enters SUSPEND mode, it will assert the PMIC_STBY_REQ signal to PMIC. When this signal is asserted, the processor allows the PMIC to shut off VDD_ARM externally. However, in some application scenario, SW want to keep the data in L2 Cache to avoid performance impact on cache miss. In this case, the VDD_ARM cannot be shut off. To support both scenarios, the PMIC should have an option to shut off or keep VDD_ARM when it receives the PMIC_STBY_REQ. This should be configured through I2C/SPI interface before the processor enters SUSPEND mode.

Except the VDD_ARM, the other power rails have to keep active in SUSPEND mode. Since the current on each power rail is greatly reduced in this mode, PMIC can enter its own low power mode to get extra power saving. For example, the PMIC can change the DCDC rails to PFM mode to reduce the power consumption.

The System IDLE, Low Power IDLE and SUSPEND mode are defined as in the table below.

Table 5-5. Low Power Mode Definition

	System IDLE	Low Power IDLE	SUSPEND
LPM Mode	WAIT	WAIT	STOP
ARM A7 CPU0	Clock Gated	OFF	OFF
ARM A7 CPU1	OFF	OFF	OFF

Table continues on the next page...

Table 5-5. Low Power Mode Definition (continued)

	System IDLE	Low Power IDLE	SUSPEND
ARM A7 SCU	ON	OFF	OFF
L2 Cache	ON	ON	OFF
VDD_ARM	ON	ON	OFF[1]
SOC	ON	ON	ON
SOC_PD	ON	ON/OFF	ON/OFF
M4 Core (inside SOC_PD)	Clock Gated	ON/OFF	ON/OFF
PLL	On as needed	OFF	OFF
24MHz XTAL	ON	OFF	OFF
24MHz RCOSC	OFF	ON	OFF
DRAM	Self-Refresh[2]	Self-Refresh	Self-Refresh
DRAM IO Low Power	No	Yes	Yes
NVCC_DRAM	ON	ON	ON
NVCC_DRAM_CKE	ON	ON	ON
LDO_1P0D	On as needed	OFF	OFF
LDO_1P0A	ON	In weak mode	OFF
LDO_1P2	On as needed	OFF	OFF
FUSE	OFF	OFF	OFF
LDO_SNVS	ON	ON	ON
Bandgap	ON	OFF	OFF
DRAM clock	24MHz	OFF	OFF
Main Fabric clock	24MHz	3MHz	OFF
AHB clock	24MHz	3MHz	OFF
IPG clock	12MHz	1.5MHz	OFF
Module clocks	On as needed	On as needed	OFF
MIPI	On as needed	OFF	OFF
USB HSIC	On as needed	OFF or SUSPEND	OFF or SUSPEND
USB OTG1	On as needed	OFF or SUSPEND	OFF or SUSPEND
GPIO wakeup	Yes	Yes	Yes
RTC Wakeup	Yes	Yes	Yes
USB Remote Wakeup	Yes	Yes	Yes [3]
Other wakeup source	Yes	Yes[4]	No

1. Turn off by PMIC when PMIC_STBY_REQ asserted.
2. Automatic enter self-refresh when there is no DRAM access.
3. Need LDO_1P0A and LDO_1P2 when wakeup from USB HSIC.
4. When a module is inside VDD_SOC_PD domain, the VDD_SOC_PD power needs to be on to support wakeup from it.

5.1.4.4.6 Low Power Target

The power consumption for the low power modes are available in the datasheet.

5.1.4.4.7 Exit Time

The exit time for each low power modes are defined as follows. The exit time includes the time needed by SoC power/clock sequence and also the time needed for ARM core state restore, but it does not include the time for any additional OS-level SW operation.

Table 5-6. Low Power Mode Exit Time Target

	System IDLE	Low Power IDLE	SUSPEND
Exit with RCOSC	N/A[1]	500us	N/A[2]
Exit with XTAL[3]	100us	2.5ms	5ms

1. The XTAL is ON in System IDLE mode.
2. Compared with the total exit time from SUSPEND mode, the saving by using RCOSC is not significant. And the flow of enter/exit from STOP mode with RCOSC is very complicated. So exit with RCOSC is not recommended for SUSPEND mode.
3. Assume 1.5ms XTAL lock time.

5.1.4.5 Managing Power Rails and Power Domains

5.1.4.5.1 General Rules

Below are some general rules / recommendations on the power supplies.

The following power rails need to be shorted together:

- VDDA_1P8, VDDA_XTAL_1P8, VDDA_ADC1_1P8, VDDA_ADC2_1P8, VDDA_TEMPSENSEOR_1P8 and FUSE_FSOURCE.

The following power rails can also be shorted together to reduce the external power rails in some applications:

- NVCC_XXX and VDDA_1P8, when the GPIO group is using 1.8V IO power, power filter is needed for the VDDA_1P8 to make sure the noise on IO pads does not go into analog modules.
- VDD_LPSR_IN and NVCC_GPIO1/2, when the GPIO bank1/2 in LPSR domain use 1.8V IO power.
- VDDA_1P8 and VDD_LPSR_IN, when GPIO state retention function is not needed in the application.
- NVCC_GPIO1/2, when these two GPIO banks runs at same IO voltage.
- NVCC_DRAM and NVCC_DRAM_CKE, when DRAM state retention in LPSR/ SNVS mode is not needed in application.

For the power supplies which has a *_CAP pad on the chip, external capacitor(s) is required in one of the follow conditions: (The recommended size/type of the capacitors could be found in the datasheet.)

- This power supply is ON by default.
- This power supply is OFF by default, but will be turned on in the application.

5.1.4.5.2 ARM and SOC Power

ARM and SOC power directly supplied externally through power pads. The ARM supply is separated with SOC because ARM requires 1.1V to run at 1GHz.

The recommended power supply scheme for ARM and SoC is:

- For applications which are very sensitive to power consumption, the VDD_SOC should be always supplied with 1.0V and VDD_ARM should be supplied with 1.1V when it runs at 1GHz and 1.0V when it runs at 800MHz and below.
- For any application that not sensitive to power consumption, these two rails can be connected together and supplied with 1.1V. In this case, ARM CPU can run up to 1GHz.
- For any application that only runs ARM core 800MHz or below, these two power rails can be connected together and supplied with 1.0V.

Both ARM and SOC has integrated power switches to shut off internal modules for power saving in low power modes.

5.1.4.5.3 MIPI PHY

The recommended power supply for MIPIPHY are shown in the table below.

Table 5-7. MIPI PHY Power Supply

Module	Vtyp(V)	Recommended Supply	PAD Name
MIPI PHY	1.0	VDDD_1P0_CAP	VDD_MIPI_ 1P0
MIPI PHY	1.8	VDDA_1P8_PHY	VDDA_MIPI_1P8

If MIPI is not used in the application, VDD_MIPI_1P0 and VDDA_MIPI_1P8 should be tied to ground.

5.1.4.5.4 USB HSIC PHY Power Supply

The recommended power supply for integrated USB HSIC PHYs are shown in the table below.

Table 5-8. Integrated USB HSIC PHY Power Supply

Module	Vtyp(V)	Recommended Supply	PAD name
USB HSIC PHY	1.2	VDD_1P2_CAP	VDD_USB_H_1P2
USB HSIC PHY	1.0	VDD_HSIC_1P0	N/A

The VDD_HSIC_1P0 is supplied internally inside the chip from VDD_SOC through the SW_HSIC_1P0 power switch.

If USB HSIC is not used in the application, the VDD_USB_H_1P2 should be tie to ground and the SW_HSIC_1P0 power switch should always keep off.

5.1.4.5.5 USB OTG PHY Power

The recommended power supply for USB OTG PHY are shown in the table below.

Table 5-9. USB OTG PHY Power Supply

Module	Vtyp(V)	Recommended Supply	PAD name
USB OTG1 PHY	1.0	VDD_USB_OTG1_1P0_CAP	VDD_USB_OTG1_1P0_CAP
USB OTG1 PHY	3.3	VDD_USB_OTG1_3P3_IN	VDD_USB_OTG1_3P3_IN

If USB OTG1 is not used in the application, both the VDD_OTG1_1P0_CAP and VDD_OTG1_3P3_IN should be tie to ground.

5.1.4.5.6 GPIO Power

For the NVCC supplies with same voltage level, they can share the same external power supply rail. However, this implies these NVCC supplies are always ON/OFF at the same time.

All the PVCC supplies are 1.8V and it is possible to short them together on the board. One of the reasons to keep PVCC supplies into several groups is to totally shut off both the PVCC and NVCC supplies to the IO bank for power saving.

For example, when PVCC_ENET_CAP is separated with other PVCC pads, it will be off when NVCC_ENET is off. So the PVCC/NVCC supply for the ENET IO bank is both off. When PVCC_ENET_CAP is shorted to PVCC_EPDC_LCD_CAP, if NVCC_ENET is off and NVCC_LCD is on, the PVCC_ENET_CAP will still be on. There will be leakage current on the PVCC_ENET_CAP for ENET IO bank.

The recommended NVCC/PVCC supply scheme is:

- If LPSR mode is used in the application, the NVCC_GPIO1/2 should be separated with other NVCC supplies, the PVCC_GPIO_1P8_CAP should be separated with other PVCC supplies.
- For applications which is very sensitive to power consumption, if one NVCC supply will be shut off while the other NVCC supplies might be on, keep this NVCC supply separate from other NVCC supplies.
- For applications which is very sensitive to power consumption, if one PVCC supply will be shut off while the other PVCC supplies might be on, keep this PVCC supply separate from other PVCC supplies.

There is one restriction on NVCC_XXX. When more than one NVCC_XXX are supplied by 1.8V, the voltage on these rails has to be the same. It is recommended to use same 1.8V source for these rails.

5.1.4.5.7 DRAM IO Power

In LPSR mode or SNVS mode, the processor support DRAM data retention mode to keep external DRAM device in self-refresh mode. In this mode, NVCC_DRAM is OFF, NVCC_DRAM_CKE is ON and the power to the DRAM chip is ON. Since PMIC usually only has one power rail for DRAM power, an PFET power switch is required to shut off the power from PMIC to NVCC_DRAM. The switch control signal is provided by the processor on its TAMPER9 pad.

The power connection between PMIC, processor and DRAM device is show in the diagram below:

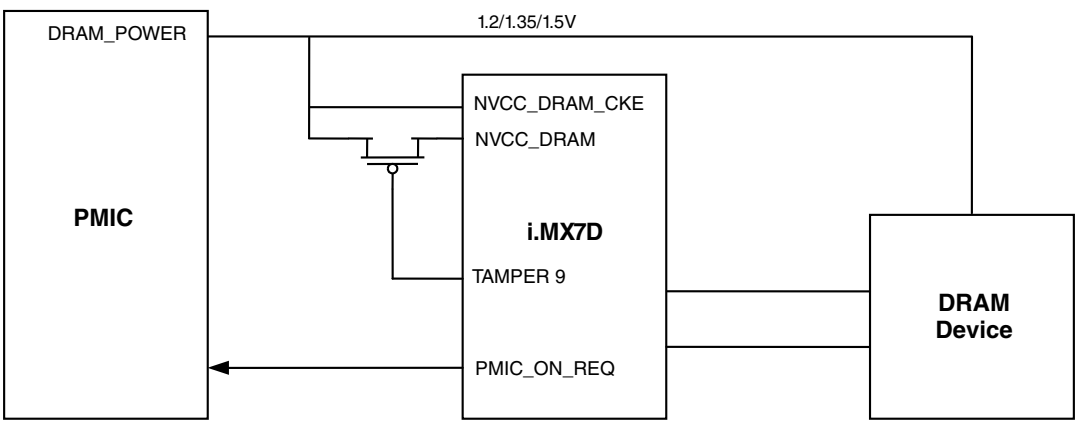


Figure 5-12. DRAM IO Power Connection

To support DRAM data retention, PMIC need to leave DRAM power rail ON in SNVS/ LPSR mode. It is recommended for the PMIC to have configurable register through I2C/SPI interface, to control the ON/OFF state of DRAM power rail when PMIC_ON_REQ signal is de-asserted.

When DRAM data retention is not needed in the application, both NVCC_DRAM_CKE and power to the DRAM chip can be off in LPSR/SNVS mode. In this scenario, the power switch is not required.

5.1.4.5.8 Power Up and Power Down Sequence

For the external power supplies, please refer to the datasheet for details on power sequence requirement. With the integrated PMU, the power up and power down sequence in the chip is simplified to allow customer to use discrete DC-DC/LDO as external power supply. It only have very limited power up/down sequence requirement on some of the rails.

For generated power supplies, the current power architecture guarantees the power up / power down sequence of the integrated PHY/IO when the recommended supply is used. If customer use external PMIC/LDO to supply these modules, the external power supply has to follow the power up / power down sequence required by the PHY/IO.

5.2 Clock Control Module (CCM)

5.2.1 Overview

Clock Control Module (CCM) manages the on-chip module clocks. CCM receives clocks from PLLs and oscillators and creates clocks for on-chip peripherals through a set of multiplexers, dividers and gates. When entering or exiting a low power mode, CCM automatically turns on and off PLLs and peripheral clocks.

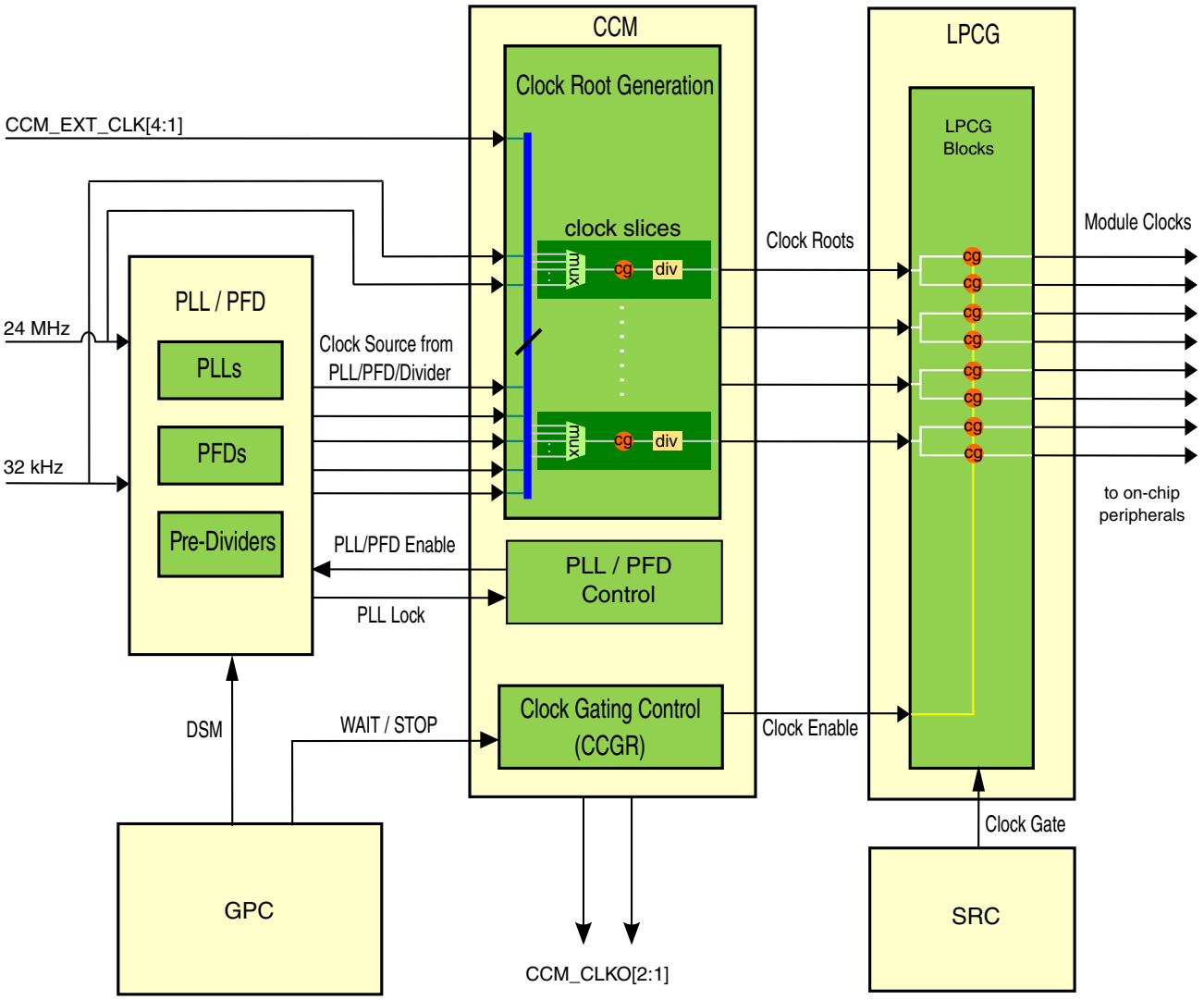


Figure 5-13. CCM Block Diagram

5.2.2 External Signals

The following table describes the external signals of CCM:

Table 5-10. CCM External Signals

Signal	Description	Pad	Mode	Direction
CCM_CLK1_N	Differential Clock 1 Input	CCM_CLK1_N	No muxing	I
CCM_CLK1_P	Differential Clock 1 Output	CCM_CLK1_P	No muxing	O
CCM_CLK2	Clock 2 Output	CCM_CLK2	No muxing	O
CCM_PMIC_STBY_REQ	Goes to PMIC_STBY_REQ pin, which notifies external power management IC to move from	CCM_PMIC_STBY_REQ	No muxing	O

Table continues on the next page...

Table 5-10. CCM External Signals (continued)

Signal	Description	Pad	Mode	Direction
	functional voltage to standby voltage.			
CCM_CLKO1	Clock 1 output for off-chip devices	GPIO1_IO02	ALT5	O
		SD1_CD_B	ALT6	
CCM_CLKO2	Clock 2 output for off-chip devices	GPIO1_IO03	ALT5	O
		SD1_WP	ALT6	
CCM_ENET1_REF_CLK	ENET Reference Clock 1	ENET1_TX_CLK	ALT1	O
		GPIO1_IO02	ALT2	
		GPIO1_IO12	ALT2	
		I2C1_SDA	ALT4	
CCM_ENET2_REF_CLK	ENET Reference Clock 2	EPDC1_BDR0	ALT3	O
		GPIO1_IO03	ALT2	
		GPIO1_IO13	ALT2	
		I2C2_SCL	ALT4	
CCM_ENET3_REF_CLK	ENET Reference Clock 3	GPIO1_IO01	ALT2	O
		GPIO1_IO09	ALT2	
		I2C2_SDA	ALT4	
CCM_EXT_CLK1	External Clock 1	ENET1_TX_CLK	ALT6	I
		GPIO1_IO12	ALT5	
		SD1_DATA0	ALT6	
CCM_EXT_CLK2	External Clock 2	ENET1_RX_CLK	ALT6	I
		GPIO1_IO13	ALT5	
		SD1_DATA1	ALT6	
CCM_EXT_CLK3	External Clock 3	ENET1_CRS	ALT6	I
		GPIO1_IO14	ALT5	
		SD1_DATA2	ALT6	
CCM_EXT_CLK4	External Clock 4	ENET1_COL	ALT6	I
		GPIO1_IO15	ALT5	
		SD1_DATA3	ALT6	
CCM_PMIC_READY	Signal coming from PMIC to indicate that the voltage started to change as result of change in CCM_PMIC_STBY_REQ	GPIO1_IO09	ALT5	I
		GPIO1_IO13	ALT4	
		SAI1_MCLK	ALT3	
		UART1_RXD	ALT2	

5.2.3 Clock Root Selects

The table below details the clock root slices and clock source inputs.

NOTE

The value of all clock root slice registers are zero after POR with the exception of DRAM clock. Please see System Boot for ROM reset values and default frequency settings for the clock root slices.

Table 5-11. Clock Root Table

Offset	Clock Root	Max Frequency (MHz)	Source Select (CCM_TARGET_ROOT[MUX])
0x8000	ARM_A7_CLK_ROOT	1000	000 - OSC_24M 001 - ARM_PLL 010 - ENET_PLL_DIV2 011 - DDR_PLL 100 - SYS_PLL 101 - SYS_PLL_PFD0 110 - AUDIO_PLL 111 - USB_PLL
0x8080	ARM_M4_CLK_ROOT	270	000 - OSC_24M 001 - SYS_PLL_DIV2 010 - ENET_PLL_DIV4 011 - SYS_PLL_PFD2 100 - DDR_PLL_DIV2 101 - AUDIO_PLL 110 - VIDEO_PLL 111 - USB_PLL
0x8800	MAIN_AXI_CLK_ROOT	333	000 - OSC_24M 001 - SYS_PLL_PFD1 010 - DDR_PLL_DIV2 011 - ENET_PLL_DIV4 100 - SYS_PLL_PFD5 101 - AUDIO_PLL 110 - VIDEO_PLL 111 - SYS_PLL_PFD7
0x8880	DISP_AXI_CLK_ROOT	333	000 - OSC_24M 001 - SYS_PLL_PFD1 010 - DDR_PLL_DIV2 011 - ENET_PLL_DIV4 100 - SYS_PLL_PFD6 101 - SYS_PLL_PFD7 110 - AUDIO_PLL

Table continues on the next page...

Table 5-11. Clock Root Table (continued)

Offset	Clock Root	Max Frequency (MHz)	Source Select (CCM_TARGET_ROOT[MUX])
			111 - VIDEO_PLL
0x8900	ENET_AXI_CLK_ROOT	270	000 - OSC_24M 001 - SYS_PLL_PFD2 010 - DDR_PLL_DIV2 011 - ENET_PLL_DIV4 100 - SYS_PLL_DIV2 101 - AUDIO_PLL 110 - VIDEO_PLL 111 - SYS_PLL_PFD4
0x8980	NAND_USDHC_BUS_CLK_ROOT	270	000 - OSC_24M 001 - AHB Clock (before AHB PODF) 010 - DDR_PLL_DIV2 011 - SYS_PLL_DIV2 100 - SYS_PLL_PFD2_DIV2 101 - SYS_PLL_PFD6 110 - ENET_PLL_DIV4 111 - AUDIO_PLL
0x9000	AHB_CLK_ROOT	135	000 - OSC_24M 001 - SYS_PLL_PFD2 010 - DDR_PLL_DIV2 011 - SYS_PLL_PFD0 100 - ENET_PLL_DIV8 101 - USB_PLL 110 - AUDIO_PLL 111 - VIDEO_PLL
0x9080	IPG_CLK_ROOT	135	000 - AHB_CLK_ROOT
0x9800	DRAM_PHYM_CLK_ROOT	533	000 - DDR_PLL 001 - DRAM_PHYM_ALT_CLK_ROOT 010 to 111 - Reserved
0x9880	DRAM_CLK_ROOT	533	000 - DDR_PLL 001 - DRAM_ALT_CLK_ROOT 010 to 111 - Reserved
0xA000	DRAM_PHYM_ALT_CLK_ROOT	533	000 - OSC_24M 001 - DDR_PLL_DIV2 010 - SYS_PLL 011 - ENET_PLL_DIV2 100 - USB_PLL

Table continues on the next page...

Table 5-11. Clock Root Table (continued)

Offset	Clock Root	Max Frequency (MHz)	Source Select (CCM_TARGET_ROOT[MUX])
			101 - SYS_PLL_PFD7 110 - AUDIO_PLL 111 - VIDEO_PLL
0xA080	DRAM_ALT_CLK_ROOT	533	000 - OSC_24M 001 - DDR_PLL_DIV2 010 - SYS_PLL 011 - ENET_PLL_DIV4 100 - USB_PLL 101 - SYS_PLL_PFD0 110 - AUDIO_PLL 111 - SYS_PLL_PFD2
0xA100	USB_HSIC_CLK_ROOT	480	000 - OSC_24M 001 - SYS_PLL 010 - USB_PLL 011 - SYS_PLL_PFD3 100 - SYS_PLL_PFD4 101 - SYS_PLL_PFD5 110 - SYS_PLL_PFD6 111 - SYS_PLL_PFD7
0xA300	LCDIF_PIXEL_CLK_ROOT	150	000 - OSC_24M 001 - SYS_PLL_PFD5 010 - DDR_PLL_DIV2 011 - EXT_CLK3 100 - SYS_PLL_PFD4 101 - SYS_PLL_PFD2 110 - VIDEO_PLL 111 - USB_PLL
0xA380	MIPI_DSI_CLK_ROOT	333	000 - OSC_24M 001 - SYS_PLL_PFD5 010 - SYS_PLL_PFD3 011 - SYS_PLL 100 - SYS_PLL_PFD0_DIV2 101 - DDR_PLL_DIV2 110 - VIDEO_PLL 111 - AUDIO_PLL
0xA400	MIPI_CSI_CLK_ROOT	333	000 - OSC_24M 001 - SYS_PLL_PFD4

Table continues on the next page...

Table 5-11. Clock Root Table (continued)

Offset	Clock Root	Max Frequency (MHz)	Source Select (CCM_TARGET_ROOT[MUX])
			010 - SYS_PLL_PFD3 011 - SYS_PLL 100 - SYS_PLL_PFD0_DIV2 101 - DDR_PLL_DIV2 110 - VIDEO_PLL 111 - AUDIO_PLL
0xA480	MIPI_DPHY_REF_CLK_ROOT	200	000 - OSC_24M 001 - SYS_PLL_DIV4 010 - DDR_PLL_DIV2 011 - SYS_PLL_PFD5 100 - REF_1M 101 - EXT_CLK2 110 - VIDEO_PLL 111 - EXT_CLK3
0xA500	SAI1_CLK_ROOT	67.5	000 - OSC_24M 001 - SYS_PLL_PFD2_DIV2 010 - AUDIO_PLL 011 - DDR_PLL_DIV2 100 - VIDEO_PLL 101 - SYS_PLL_PFD4 110 - ENET_PLL_DIV8 111 - EXT_CLK2
0xA580	SAI2_CLK_ROOT	67.5	000 - OSC_24M 001 - SYS_PLL_PFD2_DIV2 010 - AUDIO_PLL 011 - DDR_PLL_DIV2 100 - VIDEO_PLL 101 - SYS_PLL_PFD4 110 - ENET_PLL_DIV8 111 - EXT_CLK2
0xA600	SAI3_CLK_ROOT	67.5	000 - OSC_24M 001 - SYS_PLL_PFD2_DIV2 010 - AUDIO_PLL 011 - DDR_PLL_DIV2 100 - VIDEO_PLL 101 - SYS_PLL_PFD4 110 - ENET_PLL_DIV8

Table continues on the next page...

Table 5-11. Clock Root Table (continued)

Offset	Clock Root	Max Frequency (MHz)	Source Select (CCM_TARGET_ROOT[MUX])
			111 - EXT_CLK3
0xA700	ENET1_REF_CLK_ROOT	125	000 - OSC_24M 001 - ENET_PLL_DIV8 010 - ENET_PLL_DIV20 011 - ENET_PLL_DIV40 100 - SYS_PLL_DIV4 101 - AUDIO_PLL 110 - VIDEO_PLL 111 - EXT_CLK4
0xA780	ENET1_TIME_CLK_ROOT	125	000 - OSC_24M 001 - ENET_PLL_DIV10 010 - AUDIO_PLL 011 - EXT_CLK1 100 - EXT_CLK2 101 - EXT_CLK3 110 - EXT_CLK4 111 - VIDEO_PLL
0xA900	ENET_PHY_REF_CLK_ROOT	125	000 - OSC_24M 001 - ENET_PLL_DIV40 010 - ENET_PLL_DIV20 011 - ENET_PLL_DIV8 100 - DDR_PLL_DIV2 101 - AUDIO_PLL 110 - VIDEO_PLL 111 - SYS_PLL_PFD3
0xA980	EIM_CLK_ROOT	135	000 - OSC_24M 001 - SYS_PLL_PFD2_DIV2 010 - SYS_PLL_DIV4 011 - DDR_PLL_DIV2 100 - SYS_PLL_PFD2 101 - SYS_PLL_PFD3 110 - ENET_PLL_DIV8 111 - USB_PLL
0xAA00	NAND_CLK_ROOT	533	000 - OSC_24M 001 - SYS_PLL 010 - DDR_PLL_DIV2 011 - SYS_PLL_PFD0

Table continues on the next page...

Table 5-11. Clock Root Table (continued)

Offset	Clock Root	Max Frequency (MHz)	Source Select (CCM_TARGET_ROOT[MUX])
			100 - SYS_PLL_PFD3 101 - ENET_PLL_DIV2 110 - ENET_PLL_DIV4 111 - VIDEO_PLL
0xAA80	QSPI_CLK_ROOT	400	000 - OSC_24M 001 - SYS_PLL_PFD4 010 - DDR_PLL_DIV2 011 - ENET_PLL_DIV2 100 - SYS_PLL_PFD3 101 - SYS_PLL_PFD2 110 - SYS_PLL_PFD6 111 - SYS_PLL_PFD7
0xAB00	USDHC1_CLK_ROOT	200	000 - OSC_24M 001 - SYS_PLL_PFD0 010 - DDR_PLL_DIV2 011 - ENET_PLL_DIV2 100 - SYS_PLL_PFD4 101 - SYS_PLL_PFD2 110 - SYS_PLL_PFD6 111 - SYS_PLL_PFD7
0xAB80	USDHC2_CLK_ROOT	200	000 - OSC_24M 001 - SYS_PLL_PFD0 010 - DDR_PLL_DIV2 011 - ENET_PLL_DIV2 100 - SYS_PLL_PFD4 101 - SYS_PLL_PFD2 110 - SYS_PLL_PFD6 111 - SYS_PLL_PFD7
0xAC00	USDHC3_CLK_ROOT	200	000 - OSC_24M 001 - SYS_PLL_PFD0 010 - DDR_PLL_DIV2 011 - ENET_PLL_DIV2 100 - SYS_PLL_PFD4 101 - SYS_PLL_PFD2 110 - SYS_PLL_PFD6 111 - SYS_PLL_PFD7
0xAC80	CAN1_CLK_ROOT	60	000 - OSC_24M

Table continues on the next page...

Table 5-11. Clock Root Table (continued)

Offset	Clock Root	Max Frequency (MHz)	Source Select (CCM_TARGET_ROOT[MUX])
			001 - SYS_PLL_DIV4 010 - DDR_PLL_DIV2 011 - SYS_PLL 100 - ENET_PLL_DIV25 101 - USB_PLL 110 - EXT_CLK1 111 - EXT_CLK4
0xAD00	CAN2_CLK_ROOT	60	000 - OSC_24M 001 - SYS_PLL_DIV4 010 - DDR_PLL_DIV2 011 - SYS_PLL 100 - ENET_PLL_DIV25 101 - USB_PLL 110 - EXT_CLK1 111 - EXT_CLK3
0xAD80	I2C1_CLK_ROOT	67.5	000 - OSC_24M 001 - SYS_PLL_DIV4 010 - ENET_PLL_DIV20 011 - DDR_PLL_DIV2 100 - AUDIO_PLL 101 - VIDEO_PLL 110 - USB_PLL 111 - SYS_PLL_PFD2_DIV2
0xAE00	I2C2_CLK_ROOT	67.5	000 - OSC_24M 001 - SYS_PLL_DIV4 010 - ENET_PLL_DIV20 011 - DDR_PLL_DIV2 100 - AUDIO_PLL 101 - VIDEO_PLL 110 - USB_PLL 111 - SYS_PLL_PFD2_DIV2
0xAE80	I2C3_CLK_ROOT	67.5	000 - OSC_24M 001 - SYS_PLL_DIV4 010 - ENET_PLL_DIV20 011 - DDR_PLL_DIV2 100 - AUDIO_PLL 101 - VIDEO_PLL

Table continues on the next page...

Table 5-11. Clock Root Table (continued)

Offset	Clock Root	Max Frequency (MHz)	Source Select (CCM_TARGET_ROOT[MUX])
			110 - USB_PLL 111 - SYS_PLL_PFD2_DIV2
0xAF00	I2C4_CLK_ROOT	67.5	000 - OSC_24M 001 - SYS_PLL_DIV4 010 - ENET_PLL_DIV20 011 - DDR_PLL_DIV2 100 - AUDIO_PLL 101 - VIDEO_PLL 110 - USB_PLL 111 - SYS_PLL_PFD2_DIV2
0xAF80	UART1_CLK_ROOT	80	000 - OSC_24M 001 - SYS_PLL_DIV2 010 - ENET_PLL_DIV25 011 - ENET_PLL_DIV10 100 - SYS_PLL 101 - EXT_CLK2 110 - EXT_CLK4 111 - USB_PLL
0xB000	UART2_CLK_ROOT	80	000 - OSC_24M 001 - SYS_PLL_DIV2 010 - ENET_PLL_DIV25 011 - ENET_PLL_DIV10 100 - SYS_PLL 101 - EXT_CLK2 110 - EXT_CLK3 111 - USB_PLL
0xB080	UART3_CLK_ROOT	80	000 - OSC_24M 001 - SYS_PLL_DIV2 010 - ENET_PLL_DIV25 011 - ENET_PLL_DIV10 100 - SYS_PLL 101 - EXT_CLK2 110 - EXT_CLK4 111 - USB_PLL
0xB100	UART4_CLK_ROOT	80	000 - OSC_24M 001 - SYS_PLL_DIV2 010 - ENET_PLL_DIV25

Table continues on the next page...

Table 5-11. Clock Root Table (continued)

Offset	Clock Root	Max Frequency (MHz)	Source Select (CCM_TARGET_ROOT[MUX])
			011 - ENET_PLL_DIV10 100 - SYS_PLL 101 - EXT_CLK2 110 - EXT_CLK3 111 - USB_PLL
0xB180	UART5_CLK_ROOT	80	000 - OSC_24M 001 - SYS_PLL_DIV2 010 - ENET_PLL_DIV25 011 - ENET_PLL_DIV10 100 - SYS_PLL 101 - EXT_CLK2 110 - EXT_CLK4 111 - USB_PLL
0xB200	UART6_CLK_ROOT	80	000 - OSC_24M 001 - SYS_PLL_DIV2 010 - ENET_PLL_DIV25 011 - ENET_PLL_DIV10 100 - SYS_PLL 101 - EXT_CLK2 110 - EXT_CLK3 111 - USB_PLL
0xB280	UART7_CLK_ROOT	80	000 - OSC_24M 001 - SYS_PLL_DIV2 010 - ENET_PLL_DIV25 011 - ENET_PLL_DIV10 100 - SYS_PLL 101 - EXT_CLK2 110 - EXT_CLK4 111 - USB_PLL
0xB300	ECSPI1_CLK_ROOT	80	000 - OSC_24M 001 - SYS_PLL_DIV2 010 - ENET_PLL_DIV25 011 - SYS_PLL_DIV4 100 - SYS_PLL 101 - SYS_PLL_PFD4 110 - ENET_PLL_DIV4 111 - USB_PLL

Table continues on the next page...

Table 5-11. Clock Root Table (continued)

Offset	Clock Root	Max Frequency (MHz)	Source Select (CCM_TARGET_ROOT[MUX])
0xB380	ECSPI2_CLK_ROOT	80	000 - OSC_24M 001 - SYS_PLL_DIV2 010 - ENET_PLL_DIV25 011 - SYS_PLL_DIV4 100 - SYS_PLL 101 - SYS_PLL_PFD4 110 - ENET_PLL_DIV4 111 - USB_PLL
0xB400	ECSPI3_CLK_ROOT	80	000 - OSC_24M 001 - SYS_PLL_DIV2 010 - ENET_PLL_DIV25 011 - SYS_PLL_DIV4 100 - SYS_PLL 101 - SYS_PLL_PFD4 110 - ENET_PLL_DIV4 111 - USB_PLL
0xB480	ECSPI4_CLK_ROOT	80	000 - OSC_24M 001 - SYS_PLL_DIV2 010 - ENET_PLL_DIV25 011 - SYS_PLL_DIV4 100 - SYS_PLL 101 - SYS_PLL_PFD4 110 - ENET_PLL_DIV4 111 - USB_PLL
0xB500	PWM1_CLK_ROOT	67.5	000 - OSC_24M 001 - ENET_PLL_DIV10 010 - SYS_PLL_DIV4 011 - ENET_PLL_DIV25 100 - AUDIO_PLL 101 - EXT_CLK1 110 - REF_1M 111 - VIDEO_PLL
0xB580	PWM2_CLK_ROOT	67.5	000 - OSC_24M 001 - ENET_PLL_DIV10 010 - SYS_PLL_DIV4 011 - ENET_PLL_DIV25

Table continues on the next page...

Table 5-11. Clock Root Table (continued)

Offset	Clock Root	Max Frequency (MHz)	Source Select (CCM_TARGET_ROOT[MUX])
			100 - AUDIO_PLL 101 - EXT_CLK1 110 - REF_1M 111 - VIDEO_PLL
0xB600	PWM3_CLK_ROOT	67.5	000 - OSC_24M 001 - ENET_PLL_DIV10 010 - SYS_PLL_DIV4 011 - ENET_PLL_DIV25 100 - AUDIO_PLL 101 - EXT_CLK2 110 - REF_1M 111 - VIDEO_PLL
0xB680	PWM4_CLK_ROOT	67.5	000 - OSC_24M 001 - ENET_PLL_DIV10 010 - SYS_PLL_DIV4 011 - ENET_PLL_DIV25 100 - AUDIO_PLL 101 - EXT_CLK2 110 - REF_1M 111 - VIDEO_PLL
0xB700	FLEXTIMER1_CLK_ROOT	67.5	000 - OSC_24M 001 - ENET_PLL_DIV10 010 - SYS_PLL_DIV4 011 - ENET_PLL_DIV25 100 - AUDIO_PLL 101 - EXT_CLK3 110 - REF_1M 111 - VIDEO_PLL
0xB780	FLEXTIMER2_CLK_ROOT	67.5	000 - OSC_24M 001 - ENET_PLL_DIV10 010 - SYS_PLL_DIV4 011 - ENET_PLL_DIV25 100 - AUDIO_PLL 101 - EXT_CLK3 110 - REF_1M 111 - VIDEO_PLL
0xB800	SIM1_CLK_ROOT	67.5	000 - OSC_24M

Table continues on the next page...

Table 5-11. Clock Root Table (continued)

Offset	Clock Root	Max Frequency (MHz)	Source Select (CCM_TARGET_ROOT[MUX])
			001 - SYS_PLL_PFD2_DIV2 010 - SYS_PLL_DIV4 011 - DDR_PLL_DIV2 100 - USB_PLL 101 - AUDIO_PLL 110 - ENET_PLL_DIV8 111 - SYS_PLL_PFD7
0xB880	SIM2_CLK_ROOT	67.5	000 - OSC_24M 001 - SYS_PLL_PFD2_DIV2 010 - SYS_PLL_DIV4 011 - DDR_PLL_DIV2 100 - USB_PLL 101 - VIDEO_PLL 110 - ENET_PLL_DIV8 111 - SYS_PLL_PFD7
0xB900	GPT1_CLK_ROOT	100	000 - OSC_24M 001 - ENET_PLL_DIV10 010 - SYS_PLL_PFD0 011 - ENET_PLL_DIV25 100 - VIDEO_PLL 101 - REF_1M 110 - AUDIO_PLL 111 - EXT_CLK1
0xB980	GPT2_CLK_ROOT	100	000 - OSC_24M 001 - ENET_PLL_DIV10 010 - SYS_PLL_PFD0 011 - ENET_PLL_DIV25 100 - VIDEO_PLL 101 - REF_1M 110 - AUDIO_PLL 111 - EXT_CLK2
0xBA00	GPT3_CLK_ROOT	100	000 - OSC_24M 001 - ENET_PLL_DIV10 010 - SYS_PLL_PFD0 011 - ENET_PLL_DIV25 100 - VIDEO_PLL 101 - REF_1M

Table continues on the next page...

Table 5-11. Clock Root Table (continued)

Offset	Clock Root	Max Frequency (MHz)	Source Select (CCM_TARGET_ROOT[MUX])
			110 - AUDIO_PLL 111 - EXT_CLK3
0xBA80	GPT4_CLK_ROOT	100	000 - OSC_24M 001 - ENET_PLL_DIV10 010 - SYS_PLL_PFD0 011 - ENET_PLL_DIV25 100 - VIDEO_PLL 101 - REF_1M 110 - AUDIO_PLL 111 - EXT_CLK4
0xBB00	TRACE_CLK_ROOT	135	000 - OSC_24M 001 - SYS_PLL_PFD2_DIV2 010 - SYS_PLL_DIV4 011 - DDR_PLL_DIV2 100 - ENET_PLL_DIV8 101 - USB_PLL 110 - EXT_CLK2 111 - EXT_CLK3
0xBB80	WDOG_CLK_ROOT	67.5	000 - OSC_24M 001 - SYS_PLL_PFD2_DIV2 010 - SYS_PLL_DIV4 011 - DDR_PLL_DIV2 100 - ENET_PLL_DIV8 101 - USB_PLL 110 - REF_1M 111 - SYS_PLL_PFD1_DIV2
0xBC00	CSI_MCLK_CLK_ROOT	135	000 - OSC_24M 001 - SYS_PLL_PFD2_DIV2 010 - SYS_PLL_DIV4 011 - DDR_PLL_DIV2 100 - ENET_PLL_DIV8 101 - AUDIO_PLL 110 - VIDEO_PLL 111 - USB_PLL
0xBC80	AUDIO_MCLK_CLK_ROOT	135	000 - OSC_24M 001 - SYS_PLL_PFD2_DIV2 010 - SYS_PLL_DIV4

Table continues on the next page...

Table 5-11. Clock Root Table (continued)

Offset	Clock Root	Max Frequency (MHz)	Source Select (CCM_TARGET_ROOT[MUX])
			011 - DDR_PLL_DIV2 100 - ENET_PLL_DIV8 101 - AUDIO_PLL 110 - VIDEO_PLL 111 - USB_PLL
0xBD80	CCM_CLKO1	270	000 - OSC_24M 001 - SYS_PLL 010 - SYS_PLL_DIV2 011 - SYS_PLL_PFD0_DIV2 100 - SYS_PLL_PFD3 101 - ENET_PLL_DIV2 110 - DDR_PLL_DIV2 111 - REF_1M
0xBE00	CCM_CLKO2	270	000 - OSC_24M 001 - SYS_PLL_DIV2 010 - SYS_PLL_PFD0 011 - SYS_PLL_PFD1_DIV2 100 - SYS_PLL_PFD4 101 - AUDIO_PLL 110 - VIDEO_PLL 111 - OSC_32K

5.2.4 Clock Tree

The figure below illustrates the clock sources from the PLLs.

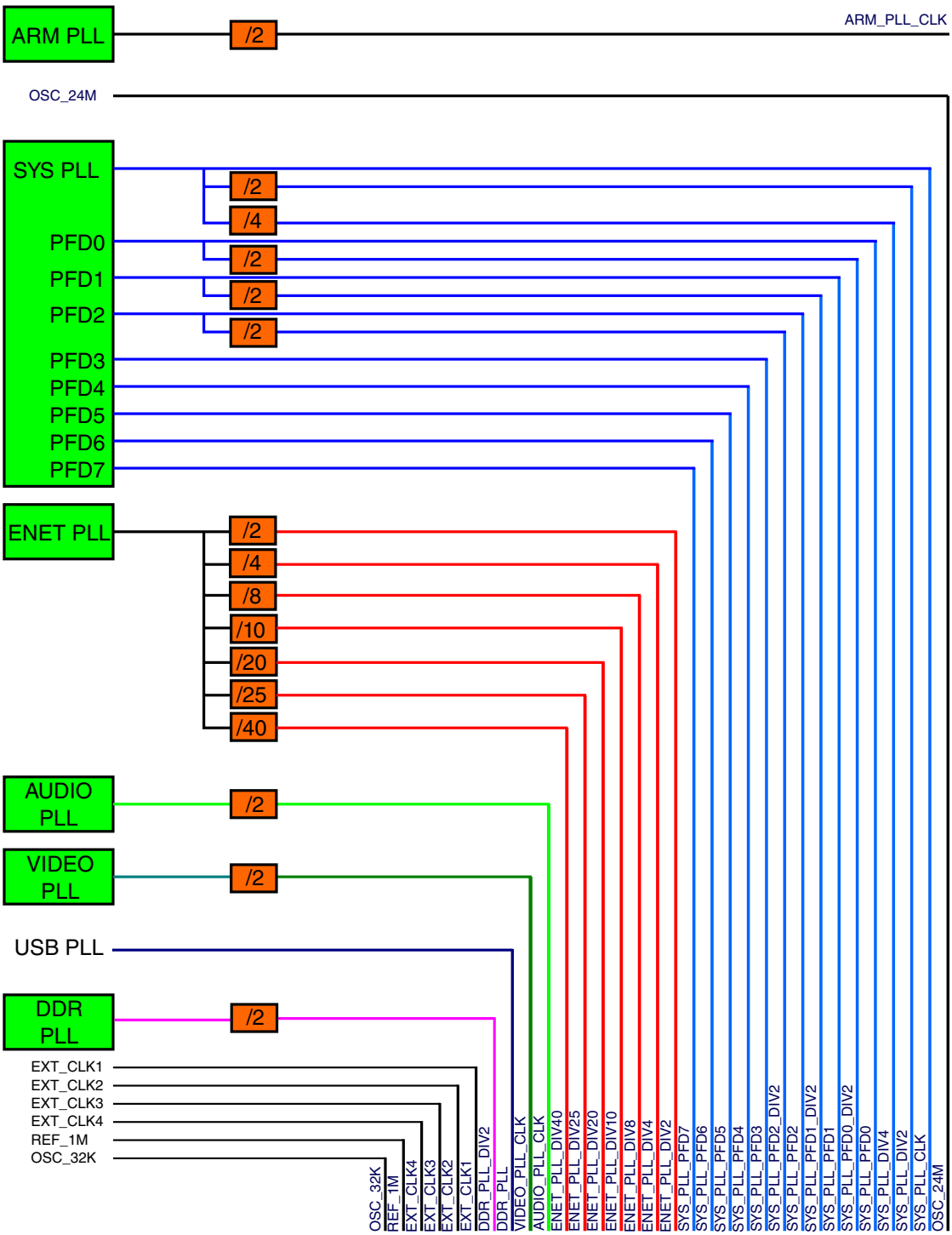
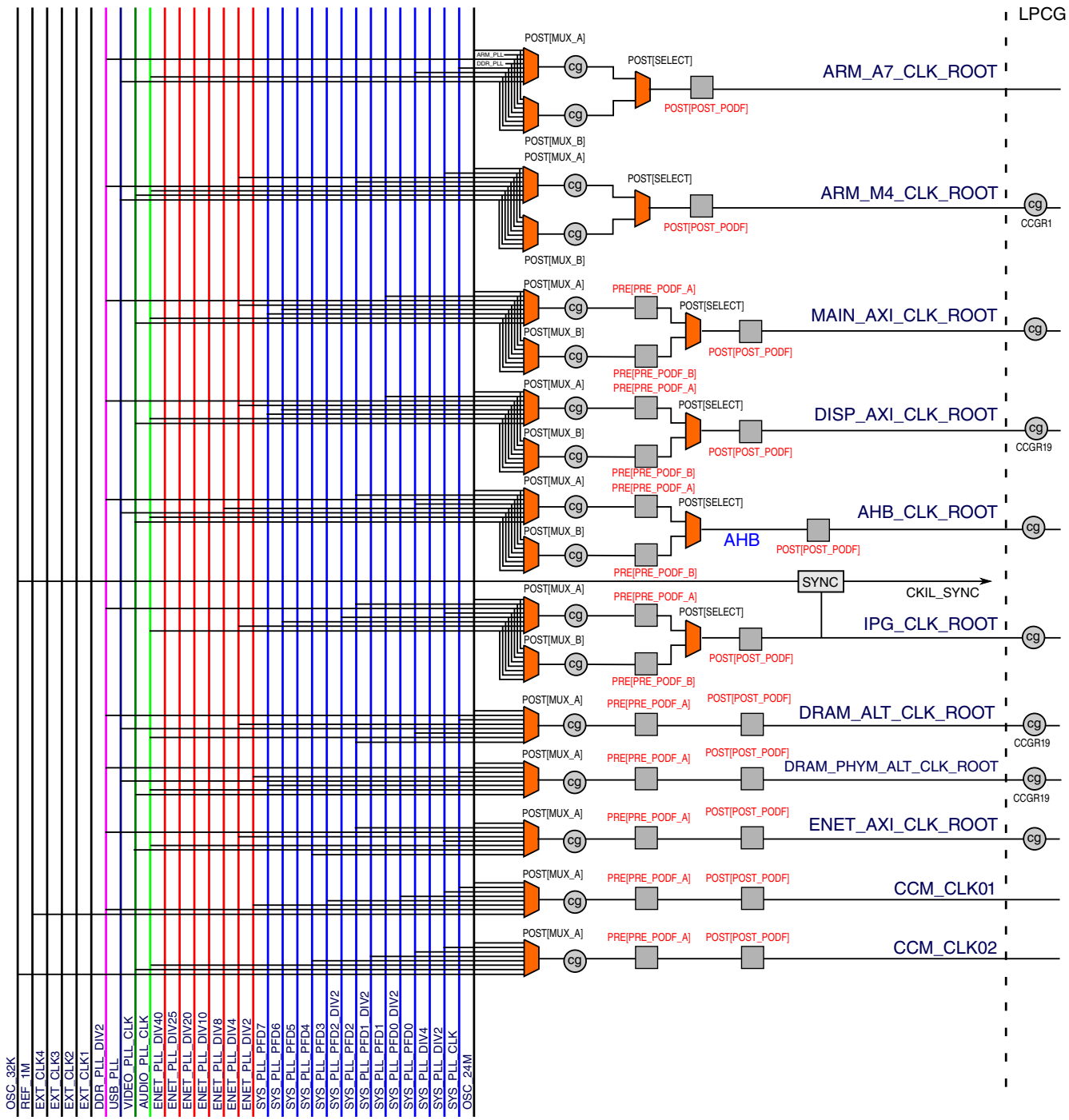
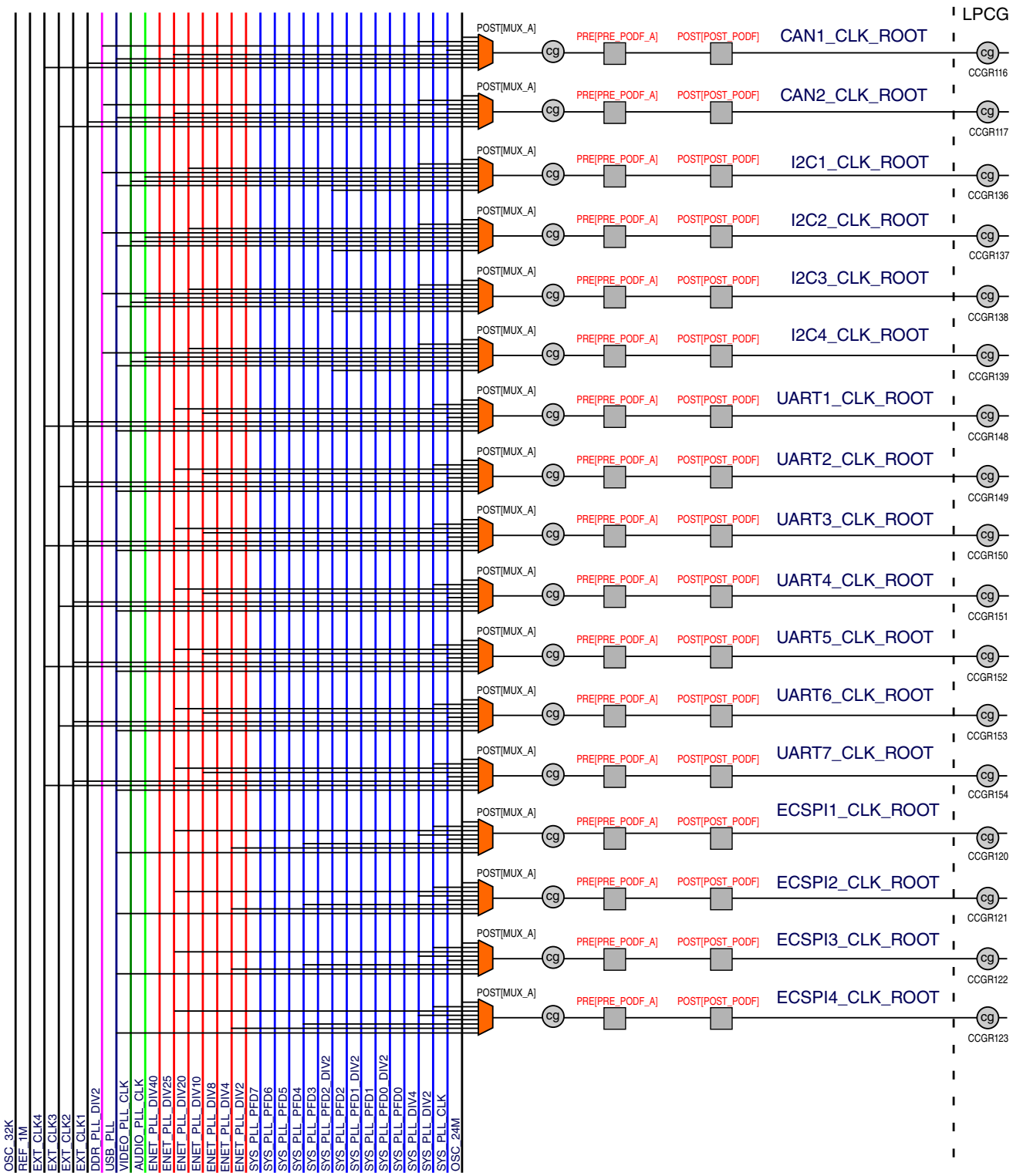
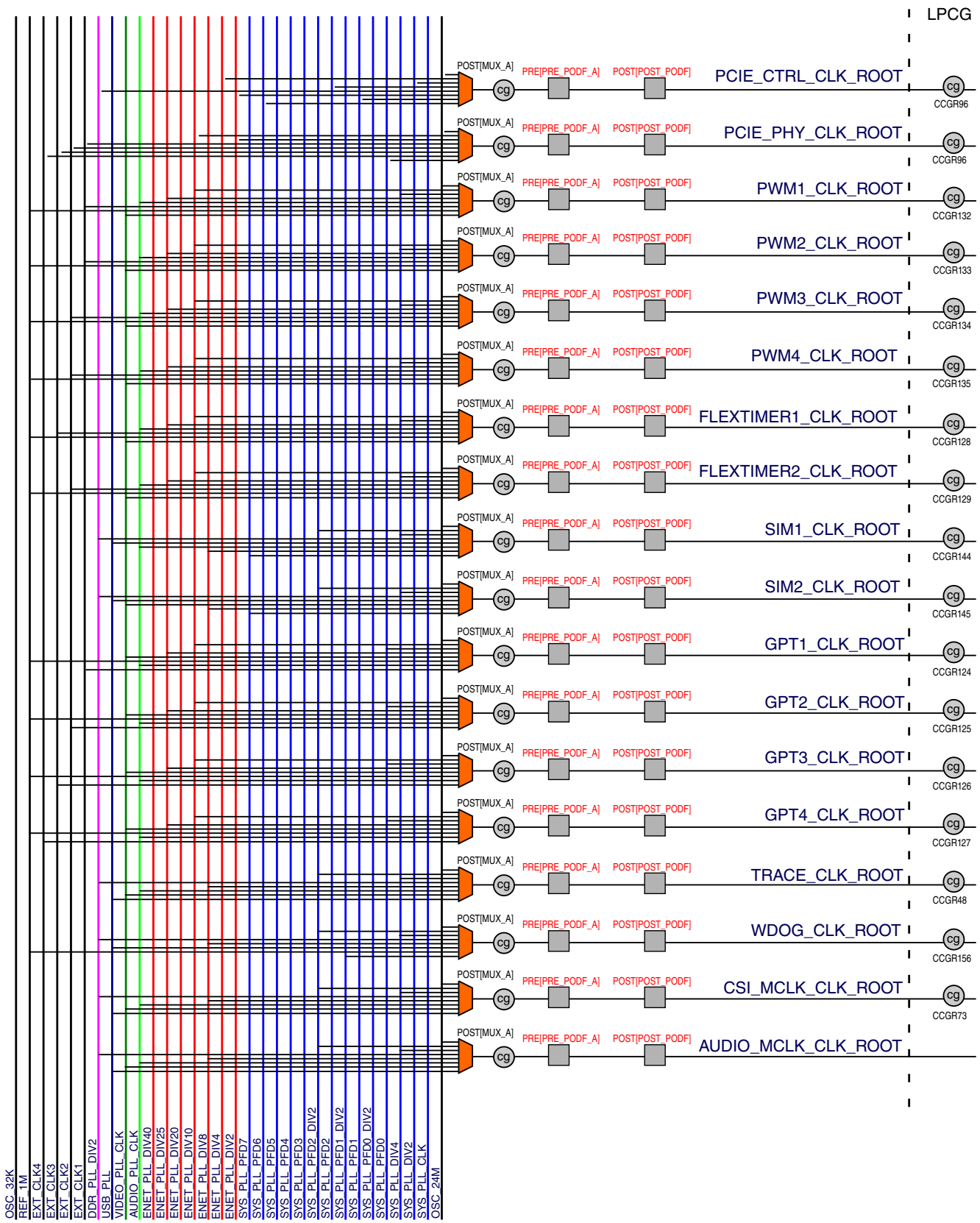


Figure 5-14. CCM input clock sources

The figure below illustrates the clock slices of CCM and clock root generation.


LPCG
ARM_A7_CLK_ROOT
ARM_M4_CLK_ROOT
MAIN_AXI_CLK_ROOT
DISP_AXI_CLK_ROOT
AHB_CLK_ROOT
IPG_CLK_ROOT
DRAM_ALT_CLK_ROOT
DRAM_PHYM_ALT_CLK_ROOT
ENET_AXI_CLK_ROOT
CCM_CLK01
CCM_CLK02
CKIL_SYNC
SYNC
AHB
CCGR1
CCGR19
POST[MUX_A]
POST[MUX_B]
POST[SELECT]
POST[POST_PODF]
PRE[PRE_PODF_A]
PRE[PRE_PODF_B]
ARM_PLL
DDR_PLL
OSC_32K
REF_1M
EXT_CLK4
EXT_CLK3
EXT_CLK2
EXT_CLK1
DDR_PLL_DIV2
USB_PLL
VIDEO_PLL_CLK
AUDIO_PLL_CLK
ENET_PLL_DIV40
ENET_PLL_DIV25
ENET_PLL_DIV20
ENET_PLL_DIV10
ENET_PLL_DIV8
ENET_PLL_DIV4
ENET_PLL_DIV2
SYS_PLL_PFD7
SYS_PLL_PFD6
SYS_PLL_PFD5
SYS_PLL_PFD4
SYS_PLL_PFD3
SYS_PLL_PFD2_DIV2
SYS_PLL_PFD2
SYS_PLL_PFD1_DIV2
SYS_PLL_PFD1
SYS_PLL_PFD0_DIV2
SYS_PLL_PFD0
SYS_PLL_DIV4
SYS_PLL_DIV2
SYS_PLL_CLK
OSC_24M


LPCG
POST[MUX_A]
PRE[PRE_PODF_A]
POST[POST_PODF]
CAN1_CLK_ROOT
CCGR116
CAN2_CLK_ROOT
CCGR117
I2C1_CLK_ROOT
CCGR136
I2C2_CLK_ROOT
CCGR137
I2C3_CLK_ROOT
CCGR138
I2C4_CLK_ROOT
CCGR139
UART1_CLK_ROOT
CCGR148
UART2_CLK_ROOT
CCGR149
UART3_CLK_ROOT
CCGR150
UART4_CLK_ROOT
CCGR151
UART5_CLK_ROOT
CCGR152
UART6_CLK_ROOT
CCGR153
UART7_CLK_ROOT
CCGR154
ECSPI1_CLK_ROOT
CCGR120
ECSPI2_CLK_ROOT
CCGR121
ECSPI3_CLK_ROOT
CCGR122
ECSPI4_CLK_ROOT
CCGR123
OSC_32K
REF_1M
EXT_CLK4
EXT_CLK3
EXT_CLK2
EXT_CLK1
DDR_PLL_DIV2
USB_PLL
VIDEO_PLL_CLK
AUDIO_PLL_CLK
ENET_PLL_DIV40
ENET_PLL_DIV25
ENET_PLL_DIV20
ENET_PLL_DIV10
ENET_PLL_DIV8
ENET_PLL_DIV4
ENET_PLL_DIV2
SYS_PLL_PFD7
SYS_PLL_PFD6
SYS_PLL_PFD5
SYS_PLL_PFD4
SYS_PLL_PFD3
SYS_PLL_PFD2_DIV2
SYS_PLL_PFD2
SYS_PLL_PFD1_DIV2
SYS_PLL_PFD1
SYS_PLL_PFD0_DIV2
SYS_PLL_PFD0
SYS_PLL_DIV4
SYS_PLL_DIV2
SYS_PLL_CLK
OSC_24M


LPCG
POST[MUX_A]
PRE[PRE_PODF_A]
POST[POST_PODF]
cg
PCIE_CTRL_CLK_ROOT
CCGR96
PCIE_PHY_CLK_ROOT
CCGR96
PWM1_CLK_ROOT
CCGR132
PWM2_CLK_ROOT
CCGR133
PWM3_CLK_ROOT
CCGR134
PWM4_CLK_ROOT
CCGR135
FLEXTIMER1_CLK_ROOT
CCGR128
FLEXTIMER2_CLK_ROOT
CCGR129
SIM1_CLK_ROOT
CCGR144
SIM2_CLK_ROOT
CCGR145
GPT1_CLK_ROOT
CCGR124
GPT2_CLK_ROOT
CCGR125
GPT3_CLK_ROOT
CCGR126
GPT4_CLK_ROOT
CCGR127
TRACE_CLK_ROOT
CCGR48
WDOG_CLK_ROOT
CCGR156
CSI_MCLK_CLK_ROOT
CCGR73
AUDIO_MCLK_CLK_ROOT
OSC_32K
REF_1M
EXT_CLK4
EXT_CLK3
EXT_CLK2
EXT_CLK1
DDR_PLL_DIV2
USB_PLL
VIDEO_PLL_CLK
AUDIO_PLL_CLK
ENET_PLL_DIV40
ENET_PLL_DIV25
ENET_PLL_DIV20
ENET_PLL_DIV10
ENET_PLL_DIV8
ENET_PLL_DIV4
ENET_PLL_DIV2
SYS_PLL_PFD7
SYS_PLL_PFD6
SYS_PLL_PFD5
SYS_PLL_PFD4
SYS_PLL_PFD3
SYS_PLL_PFD2_DIV2
SYS_PLL_PFD2
SYS_PLL_PFD1_DIV2
SYS_PLL_PFD1
SYS_PLL_PFD0_DIV2
SYS_PLL_PFD0
SYS_PLL_DIV4
SYS_PLL_DIV2
SYS_PLL_CLK
OSC_24M

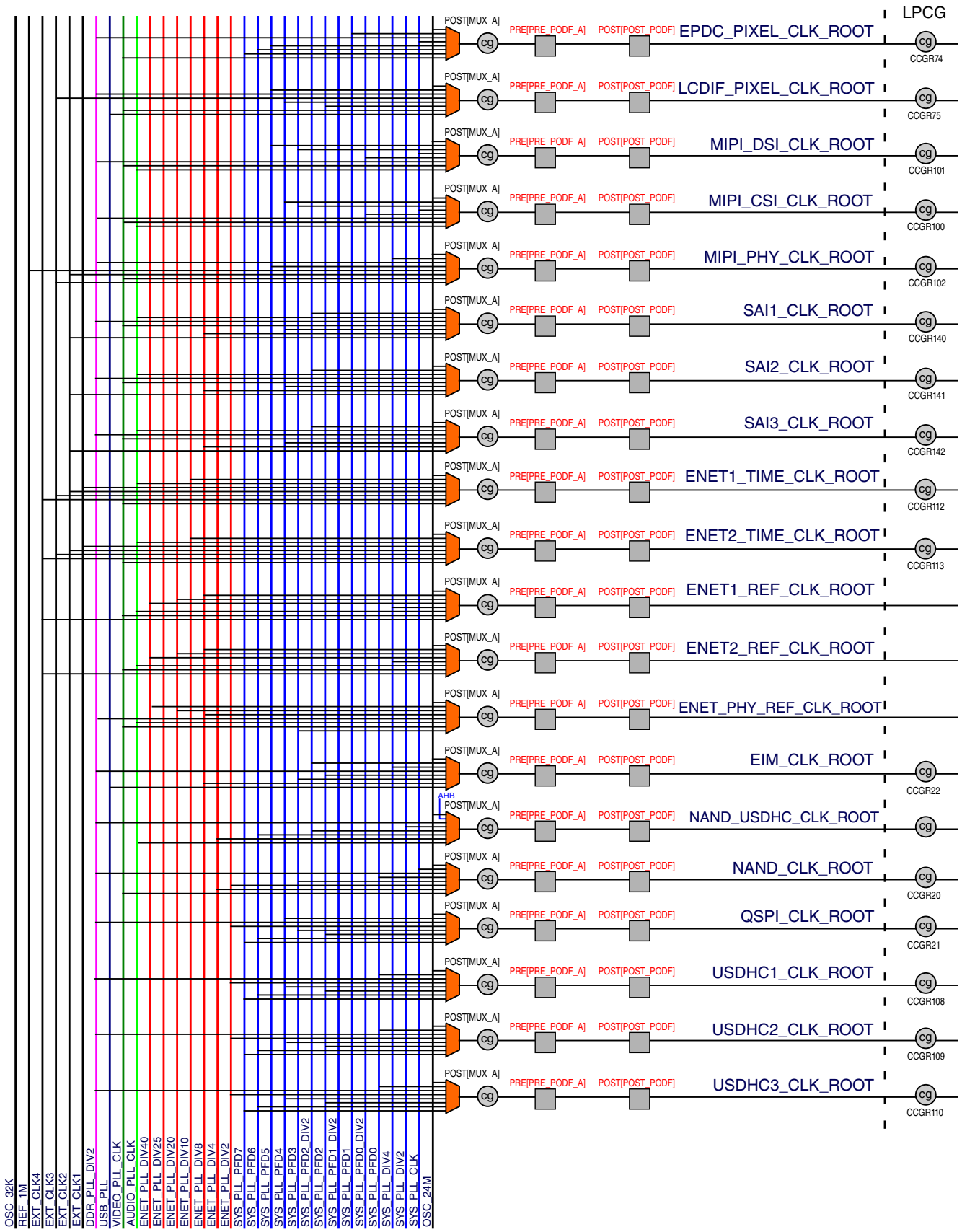


Figure 5-15. CCM Clock Tree Root Slices

5.2.5 System Clocks

The table below shows the CCM output module clock connectivity and gating.

Table 5-12. System Clocks and Gating

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
ADC	adc1.ipg_slave_clk	XTALOSC_OUT (OSC_24M)	clk_enable_adc (CCGR32)
	adc1.slave_clk	XTALOSC_OUT (OSC_24M)	clk_enable_adc (CCGR32)
	adc2.ipg_slave_clk	XTALOSC_OUT (OSC_24M)	clk_enable_adc (CCGR32)
	adc2.slave_clk	XTALOSC_OUT (OSC_24M)	clk_enable_adc (CCGR32)
	adc1.ipg_master_clk	IPG_CLK_ROOT	clk_enable_adc (CCGR32)
	adc1.master_clk	IPG_CLK_ROOT	clk_enable_adc (CCGR32)
	adc2.ipg_master_clk	IPG_CLK_ROOT	clk_enable_adc (CCGR32)
	adc2.master_clk	IPG_CLK_ROOT	clk_enable_adc (CCGR32)
	adc.ipg_clk_24mhz	XTALOSC_OUT (OSC_24M)	clk_enable_adc (CCGR32)
	adc.ipg_clk_24mhz_s	XTALOSC_OUT (OSC_24M)	clk_enable_adc (CCGR32)
AIPSTZ	aips_tz1.hclk	AHB_CLK_ROOT	clk_enable_ipmux1 (CCGR10)
	ipmux1.master_clk	AHB_CLK_ROOT	clk_enable_ipmux1 (CCGR10)
	ipmux1.slave_clk	IPG_CLK_ROOT	clk_enable_ipmux1 (CCGR10)
	aips_tz2.hclk	AHB_CLK_ROOT	clk_enable_ipmux2 (CCGR11)
	ipmux2.master_clk	AHB_CLK_ROOT	clk_enable_ipmux2 (CCGR11)
	ipmux2.slave_clk	IPG_CLK_ROOT	clk_enable_ipmux2 (CCGR11)
	aips_tz3.hclk	AHB_CLK_ROOT	clk_enable_ipmux3 (CCGR12)
	ipmux3.master_clk	AHB_CLK_ROOT	clk_enable_ipmux3 (CCGR12)
	ipmux3.slave_clk	IPG_CLK_ROOT	clk_enable_ipmux3 (CCGR12)
APBHDMA	apbhdma.hclk	NAND_USDHC_BUS_CLK_ROOT	clk_enable_rawnand (CCGR20)
	apbhdma_sec.mst_hclk	NAND_USDHC_BUS_CLK_ROOT	clk_enable_rawnand (CCGR20)
	apbhdma.u_gpmi_bch_input_gpmi_io_clk	NAND_CLK_ROOT	clk_enable_rawnand (CCGR20)
	apbhdma.u_bch_input_apb_clk	NAND_USDHC_BUS_CLK_ROOT	clk_enable_rawnand (CCGR20)
	apbhdma.u_gpmi_bch_input_bch_clk	NAND_USDHC_BUS_CLK_ROOT	clk_enable_rawnand (CCGR20)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
	apbhdma.u_gpmi_input_apb_clk	NAND_USDHC_BUS_CLK_ROOT	clk_enable_rawnand (CCGR20)
CAAM	caam_secure_mem.clk	AHB_CLK_ROOT	clk_enable_caam (CCGR36)
	caam_wrapper.aclk	AHB_CLK_ROOT	clk_enable_caam (CCGR36)
	caam_wrapper.ipg_clk	IPG_CLK_ROOT	clk_enable_caam (CCGR36)
	caam_wrapper.ipg_clk_s	IPG_CLK_ROOT	clk_enable_caam (CCGR36)
	caam_wrapper_exsc.aclk_exsc	AHB_CLK_ROOT	clk_enable_caam (CCGR36)
CM4	m4_sec.mst_hclk	ARM_M4_CLK_ROOT	clk_enable_m4 (CCGR1)
	m4_mem.tcmc_hclk	ARM_M4_CLK_ROOT	clk_enable_m4 (CCGR1)
	m4.dap_clk	AHB_CLK_ROOT	clk_enable_m4 (CCGR1)
	m4.cm4_cti_clk	ARM_M4_CLK_ROOT	clk_enable_m4 (CCGR1)
	m4.cm4_fclk	ARM_M4_CLK_ROOT	clk_enable_m4 (CCGR1)
	m4.cm4_hclk	ARM_M4_CLK_ROOT	clk_enable_m4 (CCGR1)
	m4.ipg_clk_nic	ARM_M4_CLK_ROOT	clk_enable_m4 (CCGR1)
	m4.tcmc_hclk	ARM_M4_CLK_ROOT	clk_enable_m4 (CCGR1)
CSI	csi.csi_hclk	MAIN_AXI_CLK_ROOT	clk_enable_csi (CCGR73)
	csi.ipg_clk	MAIN_AXI_CLK_ROOT	clk_enable_csi (CCGR73)
	csi.ipg_clk_s	MAIN_AXI_CLK_ROOT	clk_enable_csi (CCGR73)
	csi.ipg_clk_s_raw	MAIN_AXI_CLK_ROOT	clk_enable_csi (CCGR73)
	csi_mem.rxfifo_clk	MAIN_AXI_CLK_ROOT	clk_enable_csi (CCGR73)
	csi.ipg_master_clk	IPG_CLK_ROOT	clk_enable_csi (CCGR73)
	csi.master_clk	IPG_CLK_ROOT	clk_enable_csi (CCGR73)
	csi.ipg_slave_clk	MAIN_AXI_CLK_ROOT	clk_enable_csi (CCGR73)
	csi.slave_clk	MAIN_AXI_CLK_ROOT	clk_enable_csi (CCGR73)
	csi.mainclk	MAIN_AXI_CLK_ROOT	clk_enable_csi (CCGR73)
CSU	csu.ipg_clk_s	IPG_CLK_ROOT	clk_enable_qos_csu (CCGR45)
DAP	dap.ipg_clk_s	AHB_CLK_ROOT	clk_enable_debug (CCGR47)
	dap.dapclk_2_2	AHB_CLK_ROOT	clk_enable_debug (CCGR47)
	dap.ipg_master_clk	IPG_CLK_ROOT	clk_enable_debug (CCGR47)
	dap.master_clk	IPG_CLK_ROOT	clk_enable_debug (CCGR47)
DBGMON	dbgmon.ipg_clk_s	IPG_CLK_ROOT	clk_enable_dbgmon (CCGR46)
	dbgmon.apb_axi_ACLK	IPG_CLK_ROOT	clk_enable_dbgmon (CCGR46)
	dbgmon_lpregs.lpregs_clk	IPG_CLK_ROOT	clk_enable_dbgmon (CCGR46)
DDRC	iomux_ddr.ipt_clk_io	IPG_CLK_ROOT	clk_enable_dram (CCGR19)
	ddr.aclk_0	DRAM_CLK_ROOT	clk_enable_dram (CCGR19)
	ddr.core_ddrc_core_clk	DRAM_CLK_ROOT	clk_enable_dram (CCGR19)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
	ddr_phy.clk2x	DRAM_CLK_ROOT	clk_enable_dram (CCGR19)
	ddr_phy.clkm	DRAM_PHYM_CLK_ROOT	clk_enable_dram (CCGR19)
	ddr.ipg_clk_s	IPG_CLK_ROOT	clk_enable_dram (CCGR19)
	tzasc1.ipg_clk_s	DRAM_CLK_ROOT	clk_enable_dram (CCGR19)
	ccm_dram_clk_gen.ccm_dram_alt_clk	DRAM_ALT_CLK_ROOT	clk_enable_dram (CCGR19)
	ddrc_mrr.ipg_clk	IPG_CLK_ROOT	clk_enable_dram (CCGR19)
	mmdc_exsc.aclk_exsc	DISP_AXI_CLK_ROOT	clk_enable_dram (CCGR19)
	mmdc_exsc.ipg_clk	IPG_CLK_ROOT	clk_enable_dram (CCGR19)
	ddr.pclk	IPG_CLK_ROOT	clk_enable_dram (CCGR19)
DDR PHY	ddr_phy.PCLK	IPG_CLK_ROOT	clk_enable_dram (CCGR19)
	ddr_phy.ipg_clk_s	IPG_CLK_ROOT	clk_enable_dram (CCGR19)
ECSPI	ecspi1.ipg_clk_per	ECSPI1_CLK_ROOT	clk_enable_ecspi1 (CCGR120)
	ecspi1.ipg_clk	IPG_CLK_ROOT	clk_enable_ecspi1 (CCGR120)
	ecspi1.ipg_clk_s	IPG_CLK_ROOT	clk_enable_ecspi1 (CCGR120)
	ecspi2.ipg_clk_per	ECSPI2_CLK_ROOT	clk_enable_ecspi2 (CCGR121)
	ecspi2.ipg_clk	IPG_CLK_ROOT	clk_enable_ecspi2 (CCGR121)
	ecspi2.ipg_clk_s	IPG_CLK_ROOT	clk_enable_ecspi2 (CCGR121)
	ecspi3.ipg_clk_per	ECSPI3_CLK_ROOT	clk_enable_ecspi3 (CCGR122)
	ecspi3.ipg_clk	IPG_CLK_ROOT	clk_enable_ecspi3 (CCGR122)
	ecspi3.ipg_clk_s	IPG_CLK_ROOT	clk_enable_ecspi3 (CCGR122)
	ecspi4.ipg_clk_per	ECSPI4_CLK_ROOT	clk_enable_ecspi4 (CCGR123)
	ecspi4.ipg_clk	IPG_CLK_ROOT	clk_enable_ecspi4 (CCGR123)
	ecspi4.ipg_clk_s	IPG_CLK_ROOT	clk_enable_ecspi4 (CCGR123)
EIM	eim_exsc.aclk_exsc	EIM_CLK_ROOT	clk_enable_eim (CCGR22)
	eim_exsc.ipg_clk	IPG_CLK_ROOT	clk_enable_eim (CCGR22)
	sim_s.eimclk	EIM_CLK_ROOT	clk_enable_eim (CCGR22)
	eim.aclk	EIM_CLK_ROOT	clk_enable_eim (CCGR22)
	eim.aclk_slow	EIM_CLK_ROOT	clk_enable_eim (CCGR22)
	eim.ipg_clk_s	IPG_CLK_ROOT	clk_enable_eim (CCGR22)
ENET	enet1.ipg_clk	ENET_AXI_CLK_ROOT	clk_enable_enet1 (CCGR112)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
	enet1.ipg_clk_mac0	ENET_AXI_CLK_ROOT	clk_enable_enet1 (CCGR112)
	enet1.ipg_clk_mac0_s	ENET_AXI_CLK_ROOT	clk_enable_enet1 (CCGR112)
	enet1.ipg_clk_s	ENET_AXI_CLK_ROOT	clk_enable_enet1 (CCGR112)
	enet1.ipg_clk_time	ENET1_TIME_CLK_ROOT	clk_enable_enet1 (CCGR112)
	enet1_mem.mac0_rxmem_clk	ENET_AXI_CLK_ROOT	clk_enable_enet1 (CCGR112)
	enet1_mem.mac0_txmem_clk	ENET_AXI_CLK_ROOT	clk_enable_enet1 (CCGR112)
	enet1.ipg_slave_clk	ENET_AXI_CLK_ROOT	clk_enable_enet1 (CCGR112)
	enet1.slave_clk	ENET_AXI_CLK_ROOT	clk_enable_enet1 (CCGR112)
	enet1.ipg_master_clk	IPG_CLK_ROOT	clk_enable_enet1 (CCGR112)
	enet1.master_clk	IPG_CLK_ROOT	clk_enable_enet1 (CCGR112)
	enet2.ipg_clk	ENET_AXI_CLK_ROOT	clk_enable_enet2 (CCGR113)
	enet2.ipg_clk_mac0	ENET_AXI_CLK_ROOT	clk_enable_enet2 (CCGR113)
	enet2.ipg_clk_mac0_s	ENET_AXI_CLK_ROOT	clk_enable_enet2 (CCGR113)
	enet2.ipg_clk_s	ENET_AXI_CLK_ROOT	clk_enable_enet2 (CCGR113)
	enet2.ipg_clk_time	ENET2_TIME_CLK_ROOT	clk_enable_enet2 (CCGR113)
	enet2_mem.mac0_rxmem_clk	ENET_AXI_CLK_ROOT	clk_enable_enet2 (CCGR113)
	enet2_mem.mac0_txmem_clk	ENET_AXI_CLK_ROOT	clk_enable_enet2 (CCGR113)
	enet2.ipg_slave_clk	ENET_AXI_CLK_ROOT	clk_enable_enet2 (CCGR113)
	enet2.slave_clk	ENET_AXI_CLK_ROOT	clk_enable_enet2 (CCGR113)
	enet2.ipg_master_clk	IPG_CLK_ROOT	clk_enable_enet2 (CCGR113)
	enet2.master_clk	IPG_CLK_ROOT	clk_enable_enet2 (CCGR113)
EPDC	epdc.ipg_clk_s	MAIN_AXI_CLK_ROOT	clk_enable_epdc (CCGR74)
	ocram_epdc_exsc.ipg_clk	IPG_CLK_ROOT	clk_enable_epdc (CCGR74)
	ocram_epdc_exsc.aclk_exsc	MAIN_AXI_CLK_ROOT	clk_enable_epdc (CCGR74)
	epdc.pixclk	EPDC_PIXEL_CLK_ROOT	clk_enable_epdc (CCGR74)
	epdc.aclk	MAIN_AXI_CLK_ROOT	clk_enable_epdc (CCGR74)
	epdc_mem.ocram_clk	MAIN_AXI_CLK_ROOT	clk_enable_epdc (CCGR74)
	epdc.ipg_master_clk	IPG_CLK_ROOT	clk_enable_epdc (CCGR74)
	epdc.master_clk	IPG_CLK_ROOT	clk_enable_epdc (CCGR74)
	epdc.ipg_slave_clk	MAIN_AXI_CLK_ROOT	clk_enable_epdc (CCGR74)
	epdc.slave_clk	MAIN_AXI_CLK_ROOT	clk_enable_epdc (CCGR74)
	ocram_ctrl_epdc.clk	MAIN_AXI_CLK_ROOT	clk_enable_epdc (CCGR74)
FLEXCAN	can1.ipg_clk_pe	CAN1_CLK_ROOT	clk_enable_can1 (CCGR116)
	can1.ipg_clk_pe_nogate	CAN1_CLK_ROOT	clk_enable_can1 (CCGR116)
	can1.ipg_clk	IPG_CLK_ROOT	clk_enable_can1 (CCGR116)
	can1.ipg_clk_chi	IPG_CLK_ROOT	clk_enable_can1 (CCGR116)
	can1.ipg_clk_s	IPG_CLK_ROOT	clk_enable_can1 (CCGR116)
	can1_mem.ram_CLK	IPG_CLK_ROOT	clk_enable_can1 (CCGR116)
	can2.ipg_clk_pe	CAN2_CLK_ROOT	clk_enable_can2 (CCGR117)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
	can2.ipg_clk_pe_nogate	CAN2_CLK_ROOT	clk_enable_can2 (CCGR117)
	can2.ipg_clk	IPG_CLK_ROOT	clk_enable_can2 (CCGR117)
	can2.ipg_clk_chi	IPG_CLK_ROOT	clk_enable_can2 (CCGR117)
	can2.ipg_clk_s	IPG_CLK_ROOT	clk_enable_can2 (CCGR117)
	can2_mem.ram_CLK	IPG_CLK_ROOT	clk_enable_can2 (CCGR117)
FLEXTIMER	ftm1.ipg_clk	FLEXTIMER1_CLK_ROOT	clk_enable_ftm1 (CCGR128)
	ftm1.ipg_clk_s	FLEXTIMER1_CLK_ROOT	clk_enable_ftm1 (CCGR128)
	ftm1.ipg_slave_clk	FLEXTIMER1_CLK_ROOT	clk_enable_ftm1 (CCGR128)
	ftm1.slave_clk	FLEXTIMER1_CLK_ROOT	clk_enable_ftm1 (CCGR128)
	ftm1.ipg_master_clk	IPG_CLK_ROOT	clk_enable_ftm1 (CCGR128)
	ftm1.master_clk	IPG_CLK_ROOT	clk_enable_ftm1 (CCGR128)
	ftm2.ipg_clk	FLEXTIMER2_CLK_ROOT	clk_enable_ftm2 (CCGR129)
	ftm2.ipg_clk_s	FLEXTIMER2_CLK_ROOT	clk_enable_ftm2 (CCGR129)
	ftm2.ipg_slave_clk	FLEXTIMER2_CLK_ROOT	clk_enable_ftm2 (CCGR129)
	ftm2.slave_clk	FLEXTIMER2_CLK_ROOT	clk_enable_ftm2 (CCGR129)
	ftm2.ipg_master_clk	IPG_CLK_ROOT	clk_enable_ftm2 (CCGR129)
	ftm2.master_clk	IPG_CLK_ROOT	clk_enable_ftm2 (CCGR129)
GPIO	gpio1.ipg_clk_s	IPG_CLK_ROOT	clk_enable_gpio1 (CCGR160)
	gpio2.ipg_clk_s	IPG_CLK_ROOT	clk_enable_gpio2 (CCGR161)
	gpio3.ipg_clk_s	IPG_CLK_ROOT	clk_enable_gpio3 (CCGR162)
	gpio4.ipg_clk_s	IPG_CLK_ROOT	clk_enable_gpio4 (CCGR163)
	gpio5.ipg_clk_s	IPG_CLK_ROOT	clk_enable_gpio5 (CCGR164)
	gpio6.ipg_clk_s	IPG_CLK_ROOT	clk_enable_gpio6 (CCGR165)
	gpio7.ipg_clk_s	IPG_CLK_ROOT	clk_enable_gpio7 (CCGR166)
GPT	gpt1.ipg_clk_24m	XTALOSC_OUT (OSC_24M)	clk_enable_gpt1 (CCGR124)
	gpt1.ipg_clk	GPT1_CLK_ROOT	clk_enable_gpt1 (CCGR124)
	gpt1.ipg_clk_highfreq	GPT1_CLK_ROOT	clk_enable_gpt1 (CCGR124)
	gpt1.ipg_clk_s	GPT1_CLK_ROOT	clk_enable_gpt1 (CCGR124)
	gpt1.ipg_slave_clk	GPT1_CLK_ROOT	clk_enable_gpt1 (CCGR124)
	gpt1.slave_clk	GPT1_CLK_ROOT	clk_enable_gpt1 (CCGR124)
	gpt1.ipg_master_clk	IPG_CLK_ROOT	clk_enable_gpt1 (CCGR124)
	gpt1.master_clk	IPG_CLK_ROOT	clk_enable_gpt1 (CCGR124)
	gpt2.ipg_clk_24m	XTALOSC_OUT (OSC_24M)	clk_enable_gpt2 (CCGR125)
	gpt2.ipg_clk	GPT2_CLK_ROOT	clk_enable_gpt2 (CCGR125)
	gpt2.ipg_clk_highfreq	GPT2_CLK_ROOT	clk_enable_gpt2 (CCGR125)
	gpt2.ipg_clk_s	GPT2_CLK_ROOT	clk_enable_gpt2 (CCGR125)
	gpt2.ipg_slave_clk	GPT2_CLK_ROOT	clk_enable_gpt2 (CCGR125)
	gpt2.slave_clk	GPT2_CLK_ROOT	clk_enable_gpt2 (CCGR125)
	gpt2.ipg_master_clk	IPG_CLK_ROOT	clk_enable_gpt2 (CCGR125)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
	gpt2.master_clk	IPG_CLK_ROOT	clk_enable_gpt2 (CCGR125)
	gpt3.ipg_clk_24m	XTALOSC_OUT (OSC_24M)	clk_enable_gpt3 (CCGR126)
	gpt3.ipg_clk	GPT3_CLK_ROOT	clk_enable_gpt3 (CCGR126)
	gpt3.ipg_clk_highfreq	GPT3_CLK_ROOT	clk_enable_gpt3 (CCGR126)
	gpt3.ipg_clk_s	GPT3_CLK_ROOT	clk_enable_gpt3 (CCGR126)
	gpt3.ipg_slave_clk	GPT3_CLK_ROOT	clk_enable_gpt3 (CCGR126)
	gpt3.slave_clk	GPT3_CLK_ROOT	clk_enable_gpt3 (CCGR126)
	gpt3.ipg_master_clk	IPG_CLK_ROOT	clk_enable_gpt3 (CCGR126)
	gpt3.master_clk	IPG_CLK_ROOT	clk_enable_gpt3 (CCGR126)
	gpt4.ipg_clk_24m	XTALOSC_OUT (OSC_24M)	clk_enable_gpt4 (CCGR127)
	gpt4.ipg_clk	GPT4_CLK_ROOT	clk_enable_gpt4 (CCGR127)
	gpt4.ipg_clk_highfreq	GPT4_CLK_ROOT	clk_enable_gpt4 (CCGR127)
	gpt4.ipg_clk_s	GPT4_CLK_ROOT	clk_enable_gpt4 (CCGR127)
	gpt4.ipg_slave_clk	GPT4_CLK_ROOT	clk_enable_gpt4 (CCGR127)
	gpt4.slave_clk	GPT4_CLK_ROOT	clk_enable_gpt4 (CCGR127)
	gpt4.ipg_master_clk	IPG_CLK_ROOT	clk_enable_gpt4 (CCGR127)
	gpt4.master_clk	IPG_CLK_ROOT	clk_enable_gpt4 (CCGR127)
I2C	i2c1.ipg_clk_patref	I2C1_CLK_ROOT	clk_enable_i2c1 (CCGR136)
	i2c1.ipg_clk_s	I2C1_CLK_ROOT	clk_enable_i2c1 (CCGR136)
	i2c1.ipg_slave_clk	I2C1_CLK_ROOT	clk_enable_i2c1 (CCGR136)
	i2c1.slave_clk	I2C1_CLK_ROOT	clk_enable_i2c1 (CCGR136)
	i2c1.ipg_master_clk	IPG_CLK_ROOT	clk_enable_i2c1 (CCGR136)
	i2c1.master_clk	IPG_CLK_ROOT	clk_enable_i2c1 (CCGR136)
	i2c2.ipg_clk_patref	I2C2_CLK_ROOT	clk_enable_i2c2 (CCGR137)
	i2c2.ipg_clk_s	I2C2_CLK_ROOT	clk_enable_i2c2 (CCGR137)
	i2c2.ipg_slave_clk	I2C2_CLK_ROOT	clk_enable_i2c2 (CCGR137)
	i2c2.slave_clk	I2C2_CLK_ROOT	clk_enable_i2c2 (CCGR137)
	i2c2.ipg_master_clk	IPG_CLK_ROOT	clk_enable_i2c2 (CCGR137)
	i2c2.master_clk	IPG_CLK_ROOT	clk_enable_i2c2 (CCGR137)
	i2c3.ipg_clk_patref	I2C3_CLK_ROOT	clk_enable_i2c3 (CCGR138)
	i2c3.ipg_clk_s	I2C3_CLK_ROOT	clk_enable_i2c3 (CCGR138)
	i2c3.ipg_slave_clk	I2C3_CLK_ROOT	clk_enable_i2c3 (CCGR138)
	i2c3.slave_clk	I2C3_CLK_ROOT	clk_enable_i2c3 (CCGR138)
	i2c3.ipg_master_clk	IPG_CLK_ROOT	clk_enable_i2c3 (CCGR138)
	i2c3.master_clk	IPG_CLK_ROOT	clk_enable_i2c3 (CCGR138)
	i2c4.ipg_clk_patref	I2C4_CLK_ROOT	clk_enable_i2c4 (CCGR139)
	i2c4.ipg_clk_s	I2C4_CLK_ROOT	clk_enable_i2c4 (CCGR139)
	i2c4.ipg_slave_clk	I2C4_CLK_ROOT	clk_enable_i2c4 (CCGR139)
	i2c4.slave_clk	I2C4_CLK_ROOT	clk_enable_i2c4 (CCGR139)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
	i2c4.ipg_master_clk	IPG_CLK_ROOT	clk_enable_i2c4 (CCGR139)
	i2c4.master_clk	IPG_CLK_ROOT	clk_enable_i2c4 (CCGR139)
IOMUXC	iomuxc.ipg_clk_s	IPG_CLK_ROOT	clk_enable_iomux (CCGR168)
	iomuxc_gpr.ipg_clk_s	IPG_CLK_ROOT	clk_enable_iomux (CCGR168)
	iomux.ipt_clk_io	LCDIF_PIXEL_CLK_ROOT	clk_enable_iomux (CCGR168)
	iomuxc_lpsr.ipg_clk_s	IPG_CLK_ROOT	clk_enable_iomux_lpsr (CCGR169)
	iomuxc_lpsr_gpr.ipg_clk_s	IPG_CLK_ROOT	clk_enable_iomux_lpsr (CCGR169)
	iomux_lpsr.ipt_clk_io	LCDIF_PIXEL_CLK_ROOT	clk_enable_iomux_lpsr (CCGR169)
KPP	kpp.ipg_clk_s	IPG_CLK_ROOT	clk_enable_kpp (CCGR120)
LCDIF	lcdif.ipg_clk_s	MAIN_AXI_CLK_ROOT	clk_enable_lcdif (CCGR75)
	lcdif.ipg_master_clk	IPG_CLK_ROOT	clk_enable_lcdif (CCGR75)
	lcdif.master_clk	IPG_CLK_ROOT	clk_enable_lcdif (CCGR75)
	lcdif.ipg_slave_clk	MAIN_AXI_CLK_ROOT	clk_enable_lcdif (CCGR75)
	lcdif.slave_clk	MAIN_AXI_CLK_ROOT	clk_enable_lcdif (CCGR75)
	lcdif.pix_clk	LCDIF_PIXEL_CLK_ROOT	clk_enable_lcdif (CCGR75)
	lcdif.apb_clk	MAIN_AXI_CLK_ROOT	clk_enable_lcdif (CCGR75)
MIPI CSI	mipi_csi.ipg_clk_s	IPG_CLK_ROOT	clk_enable_mipi_csi (CCGR100)
	mipi_csi.I_PCLK	IPG_CLK_ROOT	clk_enable_mipi_csi (CCGR100)
	mipi_csi.I_WRAP_CLK	MIPI_CSI_WARP_CLK_ROOT	clk_enable_mipi_csi (CCGR100)
MIPI DSI	mipi_dsi.ipg_clk_s	IPG_CLK_ROOT	clk_enable_mipi_dsi (CCGR101)
	mipi_dsi.i_pclk	IPG_CLK_ROOT	clk_enable_mipi_dsi (CCGR101)
	mipi_dsi.i_EXTSERCLK	MIPI_DSI_EXTSER_CLK_ROOT	clk_enable_mipi_dsi (CCGR101)
	mipi_mem.O_IMG_MEM_RCLK0	CSI_MCLK_CLK_ROOT	clk_enable_mipi_phy (CCGR102)
MIPI PHY	mipi_phy.M_XI	MIPI_DPHY_REF_CLK_ROOT	clk_enable_mipi_phy (CCGR102)
MU	mu.ipg_clk_dsp	IPG_CLK_ROOT	clk_enable_mu (CCGR39)
	mu.ipg_clk_mcu	IPG_CLK_ROOT	clk_enable_mu (CCGR39)
	mu.ipg_clk_s_dsp	IPG_CLK_ROOT	clk_enable_mu (CCGR39)
	mu.ipg_clk_s_mcu	IPG_CLK_ROOT	clk_enable_mu (CCGR39)
OCOTP	ocotp_ctrl_wrapper.ipg_clk	IPG_CLK_ROOT	clk_enable_ocotp (CCGR35)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
	ocotp_ctrl_wrapper.ipg_clk_s	IPG_CLK_ROOT	clk_enable_ocotp (CCGR35)
OCRAM	ocram_exsc.ipg_clk	IPG_CLK_ROOT	clk_enable_ocram (CCGR17)
	ocram_exsc.aclk_exsc	MAIN_AXI_CLK_ROOT	clk_enable_ocram (CCGR17)
	ocram_ctrl.clk	MAIN_AXI_CLK_ROOT	clk_enable_ocram (CCGR17)
	ocram_mem.clk	MAIN_AXI_CLK_ROOT	clk_enable_ocram (CCGR17)
	ocram_s_exsc.aclk_exsc	AHB_CLK_ROOT	clk_enable_ocram_s (CCGR18)
	ocram_s_exsc.ipg_clk	IPG_CLK_ROOT	clk_enable_ocram_s (CCGR18)
	ocram_ctrl_s.clk	AHB_CLK_ROOT	clk_enable_ocram_s (CCGR18)
	ocram_s_mem.clk	AHB_CLK_ROOT	clk_enable_ocram_s (CCGR18)
PCIE	pcie_exsc.ipg_clk	IPG_CLK_ROOT	clk_enable_pcie (CCGR96)
	pcie_exsc.aclk_exsc	MAIN_AXI_CLK_ROOT	clk_enable_pcie (CCGR96)
	pcie_ctrl.mstr_aclk	MAIN_AXI_CLK_ROOT	clk_enable_pcie (CCGR96)
	pcie_ctrl.slv_aclk	MAIN_AXI_CLK_ROOT	clk_enable_pcie (CCGR96)
	pcie_clk_rst.auxclk	IPG_CLK_ROOT	clk_enable_pcie (CCGR96)
	pcie_clk_rst.dbi_axi_clk	MAIN_AXI_CLK_ROOT	clk_enable_pcie (CCGR96)
	pcie_clk_rst.mstr_axi_clk	MAIN_AXI_CLK_ROOT	clk_enable_pcie (CCGR96)
	pcie_clk_rst.slv_axi_clk	MAIN_AXI_CLK_ROOT	clk_enable_pcie (CCGR96)
	pcie_ram.mstr_axi_clk	MAIN_AXI_CLK_ROOT	clk_enable_pcie (CCGR96)
	pcie_ram.slv_axi_clk	MAIN_AXI_CLK_ROOT	clk_enable_pcie (CCGR96)
	pcie_ctrl.pipe_clk	pcie_phy.PC_CLK	clk_enable_pcie (CCGR96)
PCIE PHY	pcie_phy.ipg_clk_s	IPG_CLK_ROOT	clk_enable_pcie (CCGR96)
	pcie_phy.PCLK	IPG_CLK_ROOT	clk_enable_pcie (CCGR96)
	pcie_phy.REFCLK_EXT	PCIE_PHY_CLK_ROOT	clk_enable_pcie (CCGR96)
PERFMON	perfmon1.ipg_clk_s	IPG_CLK_ROOT	clk_enable_perfmon1 (CCGR68)
	perfmon1.axi0_ACLK	DRAM_CLK_ROOT	clk_enable_perfmon1 (CCGR68)
	perfmon1.apb_clk	IPG_CLK_ROOT	clk_enable_perfmon1 (CCGR68)
	perfmon2.ipg_clk_s	IPG_CLK_ROOT	clk_enable_perfmon2 (CCGR69)
	perfmon2.axi0_ACLK	DRAM_CLK_ROOT	clk_enable_perfmon2 (CCGR69)
	perfmon2.apb_clk	IPG_CLK_ROOT	clk_enable_perfmon2 (CCGR69)
PWM	pwm1.ipg_master_clk	IPG_CLK_ROOT	clk_enable_pwm1 (CCGR132)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
	pwm1.master_clk	IPG_CLK_ROOT	clk_enable_pwm1 (CCGR132)
	pwm1.ipg_slave_clk	PWM1_CLK_ROOT	clk_enable_pwm1 (CCGR132)
	pwm1.slave_clk	PWM1_CLK_ROOT	clk_enable_pwm1 (CCGR132)
	pwm1.ipg_clk	PWM1_CLK_ROOT	clk_enable_pwm1 (CCGR132)
	pwm1.ipg_clk_highfreq	PWM1_CLK_ROOT	clk_enable_pwm1 (CCGR132)
	pwm1.ipg_clk_s	PWM1_CLK_ROOT	clk_enable_pwm1 (CCGR132)
	pwm2.ipg_master_clk	IPG_CLK_ROOT	clk_enable_pwm2 (CCGR133)
	pwm2.master_clk	IPG_CLK_ROOT	clk_enable_pwm2 (CCGR133)
	pwm2.ipg_slave_clk	PWM2_CLK_ROOT	clk_enable_pwm2 (CCGR133)
	pwm2.slave_clk	PWM2_CLK_ROOT	clk_enable_pwm2 (CCGR133)
	pwm2.ipg_clk	PWM2_CLK_ROOT	clk_enable_pwm2 (CCGR133)
	pwm2.ipg_clk_highfreq	PWM2_CLK_ROOT	clk_enable_pwm2 (CCGR133)
	pwm2.ipg_clk_s	PWM2_CLK_ROOT	clk_enable_pwm2 (CCGR133)
	pwm3.ipg_master_clk	IPG_CLK_ROOT	clk_enable_pwm3 (CCGR134)
	pwm3.master_clk	IPG_CLK_ROOT	clk_enable_pwm3 (CCGR134)
	pwm3.ipg_slave_clk	PWM3_CLK_ROOT	clk_enable_pwm3 (CCGR134)
	pwm3.slave_clk	PWM3_CLK_ROOT	clk_enable_pwm3 (CCGR134)
	pwm3.ipg_clk	PWM3_CLK_ROOT	clk_enable_pwm3 (CCGR134)
	pwm3.ipg_clk_highfreq	PWM3_CLK_ROOT	clk_enable_pwm3 (CCGR134)
	pwm3.ipg_clk_s	PWM3_CLK_ROOT	clk_enable_pwm3 (CCGR134)
	pwm4.ipg_master_clk	IPG_CLK_ROOT	clk_enable_pwm4 (CCGR135)
	pwm4.master_clk	IPG_CLK_ROOT	clk_enable_pwm4 (CCGR135)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
	pwm4.ipg_slave_clk	PWM4_CLK_ROOT	clk_enable_pwm4 (CCGR135)
	pwm4.slave_clk	PWM4_CLK_ROOT	clk_enable_pwm4 (CCGR135)
	pwm4.ipg_clk	PWM4_CLK_ROOT	clk_enable_pwm4 (CCGR135)
	pwm4.ipg_clk_highfreq	PWM4_CLK_ROOT	clk_enable_pwm4 (CCGR135)
	pwm4.ipg_clk_s	PWM4_CLK_ROOT	clk_enable_pwm4 (CCGR135)
PXP	pxp.ipg_clk_s	MAIN_AXI_CLK_ROOT	clk_enable_pxp (CCGR76)
	ocram_pxp_exsc.ipg_clk	IPG_CLK_ROOT	clk_enable_pxp (CCGR76)
	ocram_pxp_exsc.aclk_exsc	MAIN_AXI_CLK_ROOT	clk_enable_pxp (CCGR76)
	pxp.ipg_master_clk	IPG_CLK_ROOT	clk_enable_pxp (CCGR76)
	pxp.master_clk	IPG_CLK_ROOT	clk_enable_pxp (CCGR76)
	pxp.ipg_slave_clk	MAIN_AXI_CLK_ROOT	clk_enable_pxp (CCGR76)
	pxp.slave_clk	MAIN_AXI_CLK_ROOT	clk_enable_pxp (CCGR76)
	ocram_ctrl_pxp.clk	MAIN_AXI_CLK_ROOT	clk_enable_pxp (CCGR76)
	ocram_pxp.clk	MAIN_AXI_CLK_ROOT	clk_enable_pxp (CCGR76)
	pxp.clk	MAIN_AXI_CLK_ROOT	clk_enable_pxp (CCGR76)
QOS	qos.ipg_clk_s	IPG_CLK_ROOT	clk_enable_qos (CCGR42)
	qos.apb_clk	IPG_CLK_ROOT	clk_enable_qos (CCGR42)
	qos_dispmix.apb_clk	IPG_CLK_ROOT	clk_enable_qos_dispmix (CCGR43)
	qos_megamix.apb_clk	IPG_CLK_ROOT	clk_enable_qos_megamix (CCGR44)
QuadSPI	qspi_sec.mst_hclk	AHB_CLK_ROOT	clk_enable_qspi (CCGR21)
	qspi_sec.ipg_clk	IPG_CLK_ROOT	clk_enable_qspi (CCGR21)
	qspi_sec.ipg_clk_s	IPG_CLK_ROOT	clk_enable_qspi (CCGR21)
	qspi.ahb_clk	AHB_CLK_ROOT	clk_enable_qspi (CCGR21)
	qspi.ipg_clk	IPG_CLK_ROOT	clk_enable_qspi (CCGR21)
	qspi.ipg_clk_s	IPG_CLK_ROOT	clk_enable_qspi (CCGR21)
	qspi.ipg_clk_4xsfif	QSPI_CLK_ROOT	clk_enable_qspi (CCGR21)
RDC	rdc.ipg_clk	IPG_CLK_ROOT	clk_enable_rdc (CCGR38)
	rdc.ipg_clk_s	IPG_CLK_ROOT	clk_enable_rdc (CCGR38)
ROMCP	romcp_sec.mst_hclk	AHB_CLK_ROOT	clk_enable_rom (CCGR16)
	romcp.hclk	AHB_CLK_ROOT	clk_enable_rom (CCGR16)
	romcp.hclk_reg	IPG_CLK_ROOT	clk_enable_rom (CCGR16)
	rom_96k.rom_CLK	AHB_CLK_ROOT	clk_enable_rom (CCGR16)
SAI	sai1.ipg_clk	IPG_CLK_ROOT	clk_enable_sai1 (CCGR140)
	sai1.ipg_clk_s	IPG_CLK_ROOT	clk_enable_sai1 (CCGR140)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
	sai1.ipg_clk_sai_mclk1	SAI1_CLK_ROOT	clk_enable_sai1 (CCGR140)
	sai1.ipt_clk_sai_bclk	SAI1_CLK_ROOT	clk_enable_sai1 (CCGR140)
	sai1.ipt_clk_sai_bclk_b	SAI1_CLK_ROOT	clk_enable_sai1 (CCGR140)
	sai1.ipt_clk_sai_mclk	SAI1_CLK_ROOT	clk_enable_sai1 (CCGR140)
	sai2.ipg_clk	IPG_CLK_ROOT	clk_enable_sai2 (CCGR141)
	sai2.ipg_clk_s	IPG_CLK_ROOT	clk_enable_sai2 (CCGR141)
	sai2.ipg_clk_sai_mclk1	SAI2_CLK_ROOT	clk_enable_sai2 (CCGR141)
	sai2.ipt_clk_sai_bclk	SAI2_CLK_ROOT	clk_enable_sai2 (CCGR141)
	sai2.ipt_clk_sai_bclk_b	SAI2_CLK_ROOT	clk_enable_sai2 (CCGR141)
	sai2.ipt_clk_sai_mclk	SAI2_CLK_ROOT	clk_enable_sai2 (CCGR141)
	sai3.ipg_clk	IPG_CLK_ROOT	clk_enable_sai3 (CCGR142)
	sai3.ipg_clk_s	IPG_CLK_ROOT	clk_enable_sai3 (CCGR142)
	sai3.ipg_clk_sai_mclk1	SAI3_CLK_ROOT	clk_enable_sai3 (CCGR142)
	sai3.ipt_clk_sai_bclk	SAI3_CLK_ROOT	clk_enable_sai3 (CCGR142)
	sai3.ipt_clk_sai_bclk_b	SAI3_CLK_ROOT	clk_enable_sai3 (CCGR142)
	sai3.ipt_clk_sai_mclk	SAI3_CLK_ROOT	clk_enable_sai3 (CCGR142)
SCTR	sctr.ipg_master_clk	IPG_CLK_ROOT	clk_enable_sctr (CCGR34)
	sctr.master_clk	IPG_CLK_ROOT	clk_enable_sctr (CCGR34)
	sctr.scan_clk	XTALOSC_OUT (OSC_24M)	clk_enable_sctr (CCGR34)
	sctr.sys_ctr_base_clk	XTALOSC_OUT (OSC_24M)	clk_enable_sctr (CCGR34)
	sctr.ipg_clk	IPG_CLK_ROOT	clk_enable_sctr (CCGR34)
	sctr.ipg_clk_s	IPG_CLK_ROOT	clk_enable_sctr (CCGR34)
	sctr_gray2bin.clk	ca7_clkin	clk_enable_sctr (CCGR34)
SDMA	sdma.CLK	AHB_CLK_ROOT	clk_enable_sdma (CCGR72)
	ipmux_sdma.master_clk	IPG_CLK_ROOT	clk_enable_sdma (CCGR72)
	ipmux_sdma.slave_clk	IPG_CLK_ROOT	clk_enable_sdma (CCGR72)
	sdma_events_sync.clk	AHB_CLK_ROOT	clk_enable_sdma (CCGR72)
	sdma.sdma_ap_ahb_clk	AHB_CLK_ROOT	clk_enable_sdma (CCGR72)
	sdma.ips_hostctrl_clk	IPG_CLK_ROOT	clk_enable_sdma (CCGR72)
	sdma.sdma_core_clk	IPG_CLK_ROOT	clk_enable_sdma (CCGR72)
SEC	sec_wrapper.clk	IPG_CLK_ROOT	clk_enable_sec_debug (CCGR49)
SEMA42	sema1.clk	IPG_CLK_ROOT	clk_enable_sema1 (CCGR64)
	sema2.clk	IPG_CLK_ROOT	clk_enable_sema2 (CCGR65)
SIM	sim_display.mainclk	MAIN_AXI_CLK_ROOT	clk_enable_sim_display (CCGR5)
	sim_display.mainclk_r	MAIN_AXI_CLK_ROOT	clk_enable_sim_display (CCGR5)
	sim_enet.mainclk	ENET_AXI_CLK_ROOT	clk_enable_sim_enet (CCGR6)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
	sim_enet.mainclk_r	ENET_AXI_CLK_ROOT	clk_enable_sim_enet (CCGR6)
	sim_main.enetclk	ENET_AXI_CLK_ROOT	clk_enable_sim_enet (CCGR6)
	sim_s.gpv4clk	ENET_AXI_CLK_ROOT	clk_enable_sim_enet (CCGR6)
	sim_m.mainclk	AHB_CLK_ROOT	clk_enable_sim_m (CCGR7)
	sim_m.mainclk_r	AHB_CLK_ROOT	clk_enable_sim_m (CCGR7)
	sim_main.per_mclk	AHB_CLK_ROOT	clk_enable_sim_m (CCGR7)
	sim_m.usdhcclk	NAND_USDHC_BUS_CLK_ROOT	clk_enable_sim_m (CCGR7)
	sim_m.usdhcclk_r	NAND_USDHC_BUS_CLK_ROOT	clk_enable_sim_m (CCGR7)
	sim_main.usdhcclk	NAND_USDHC_BUS_CLK_ROOT	clk_enable_sim_m (CCGR7)
	sim_main.mainclk	MAIN_AXI_CLK_ROOT	clk_enable_sim_main (CCGR4)
	sim_main.mainclk_r	MAIN_AXI_CLK_ROOT	clk_enable_sim_main (CCGR4)
	sim_main.per_sclk	AHB_CLK_ROOT	clk_enable_sim_s (CCGR8)
	sim_s.mainclk	AHB_CLK_ROOT	clk_enable_sim_s (CCGR8)
	sim_s.mainclk_r	AHB_CLK_ROOT	clk_enable_sim_s (CCGR8)
	sim_main.wakeupclk	AHB_CLK_ROOT	clk_enable_sim_wakeup (CCGR9)
	sim_wakeup.mainclk	AHB_CLK_ROOT	clk_enable_sim_wakeup (CCGR9)
	sim_wakeup.mainclk_r	AHB_CLK_ROOT	clk_enable_sim_wakeup (CCGR9)
	sim1.ipg_clk	IPG_CLK_ROOT	clk_enable_sim1 (CCGR144)
	sim1.ipg_clk_s	IPG_CLK_ROOT	clk_enable_sim1 (CCGR144)
	sim1.ipg_perclk	SIM1_CLK_ROOT	clk_enable_sim1 (CCGR144)
	sim2.ipg_clk	IPG_CLK_ROOT	clk_enable_sim2 (CCGR145)
	sim2.ipg_clk_s	IPG_CLK_ROOT	clk_enable_sim2 (CCGR145)
	sim2.ipg_perclk	SIM2_CLK_ROOT	clk_enable_sim2 (CCGR145)
	sim_m.apbhdmaclk	NAND_USDHC_BUS_CLK_ROOT	clk_enable_rawnand (CCGR20)
	sim_m.bchclk	NAND_USDHC_BUS_CLK_ROOT	clk_enable_rawnand (CCGR20)
	sim_s.apbhdmaclk	NAND_USDHC_BUS_CLK_ROOT	clk_enable_rawnand (CCGR20)
	sim_display.cm4clk	ARM_M4_CLK_ROOT	clk_enable_m4 (CCGR1)
	sim_main.cm4clk	ARM_M4_CLK_ROOT	clk_enable_m4 (CCGR1)
	sim_main.dramclk	DRAM_CLK_ROOT	clk_enable_dram (CCGR19)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
SNVS	anatop.snvs_ipg_clk	IPG_CLK_ROOT	clk_enable_snvs (CCGR37)
	anatop.snvs_ipg_clk_s	IPG_CLK_ROOT	clk_enable_snvs (CCGR37)
	snvs_hp_wrapper.ipg_clk	IPG_CLK_ROOT	clk_enable_snvs (CCGR37)
	snvs_hp_wrapper.ipg_clk_s	IPG_CLK_ROOT	clk_enable_snvs (CCGR37)
SPBA	spba.ipg_clk	IPG_CLK_ROOT	clk_enable_ipmux3 (CCGR12)
	spba.ipg_clk_s	IPG_CLK_ROOT	clk_enable_ipmux3 (CCGR12)
TRACE	coresight_mem.cs_etf_clk	MAIN_AXI_CLK_ROOT	clk_enable_trace (CCGR48)
	hugo.DBGCLK	MAIN_AXI_CLK_ROOT	clk_enable_trace (CCGR48)
	hugo.traceclkin	TRACE_CLK_ROOT	clk_enable_trace (CCGR48)
TZASC	tzasc1.ipg_slave_clk	DRAM_CLK_ROOT	clk_enable_dram (CCGR19)
	tzasc1.slave_clk	DRAM_CLK_ROOT	clk_enable_dram (CCGR19)
	tzasc1.ipg_master_clk	IPG_CLK_ROOT	clk_enable_dram (CCGR19)
	tzasc1.master_clk	IPG_CLK_ROOT	clk_enable_dram (CCGR19)
	tzasc1.aclk	DRAM_CLK_ROOT	clk_enable_dram (CCGR19)
UART	uart1.ipg_clk	IPG_CLK_ROOT	clk_enable_uart1 (CCGR148)
	uart1.ipg_clk_s	IPG_CLK_ROOT	clk_enable_uart1 (CCGR148)
	uart1.ipg_perclk	UART1_CLK_ROOT	clk_enable_uart1 (CCGR148)
	uart2.ipg_clk	IPG_CLK_ROOT	clk_enable_uart2 (CCGR149)
	uart2.ipg_clk_s	IPG_CLK_ROOT	clk_enable_uart2 (CCGR149)
	uart2.ipg_perclk	UART2_CLK_ROOT	clk_enable_uart2 (CCGR149)
	uart3.ipg_clk	IPG_CLK_ROOT	clk_enable_uart3 (CCGR150)
	uart3.ipg_clk_s	IPG_CLK_ROOT	clk_enable_uart3 (CCGR150)
	uart3.ipg_clk_s	IPG_CLK_ROOT	clk_enable_uart3 (CCGR150)
	uart3.ipg_perclk	UART3_CLK_ROOT	clk_enable_uart3 (CCGR150)
	uart4.ipg_clk	IPG_CLK_ROOT	clk_enable_uart4 (CCGR151)
	uart4.ipg_clk_s	IPG_CLK_ROOT	clk_enable_uart4 (CCGR151)
	uart4.ipg_perclk	UART4_CLK_ROOT	clk_enable_uart4 (CCGR151)
	uart5.ipg_clk	IPG_CLK_ROOT	clk_enable_uart5 (CCGR152)
	uart5.ipg_clk_s	IPG_CLK_ROOT	clk_enable_uart5 (CCGR152)
	uart5.ipg_perclk	UART5_CLK_ROOT	clk_enable_uart5 (CCGR152)
	uart6.ipg_clk	IPG_CLK_ROOT	clk_enable_uart6 (CCGR153)
	uart6.ipg_clk_s	IPG_CLK_ROOT	clk_enable_uart6 (CCGR153)
	uart6.ipg_perclk	UART6_CLK_ROOT	clk_enable_uart6 (CCGR153)
	uart7.ipg_clk	IPG_CLK_ROOT	clk_enable_uart7 (CCGR154)
	uart7.ipg_clk_s	IPG_CLK_ROOT	clk_enable_uart7 (CCGR154)
	uart7.ipg_perclk	UART7_CLK_ROOT	clk_enable_uart7 (CCGR154)
USB	hs.clk	IPG_CLK_ROOT	clk_enable_hs (CCGR40)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
	usb.ipg_ahb_clk	AHB_CLK_ROOT	clk_enable_usb_ctrl (CCGR104)
	usb.ipg_clk_s	IPG_CLK_ROOT	clk_enable_usb_ctrl (CCGR104)
	usb.ipg_clk_s_pl301	IPG_CLK_ROOT	clk_enable_usb_ctrl (CCGR104)
	usb.test_clk_60m	IPG_CLK_ROOT	clk_enable_usb_ctrl (CCGR104)
	hsic_phy.PLL480MCLK	USB_HSIC_CLK_ROOT	clk_enable_usb_hsic (CCGR105)
USB PHY	otg1_phy.CLKCORE	XTALOSC_OUT (OSC_24M)	clk_enable_usb_phy1 (CCGR106)
	otg2_phy.CLKCORE	XTALOSC_OUT (OSC_24M)	clk_enable_usb_phy2 (CCGR107)
USDHC	usdhc1.CLK	NAND_USDHC_BUS_CLK_ROOT	clk_enable_usdhc1 (CCGR108)
	usdhc1.ipg_clk	IPG_CLK_ROOT	clk_enable_usdhc1 (CCGR108)
	usdhc1.ipg_clk_s	IPG_CLK_ROOT	clk_enable_usdhc1 (CCGR108)
	usdhc1.hclk	NAND_USDHC_BUS_CLK_ROOT	clk_enable_usdhc1 (CCGR108)
	usdhc1.ipg_clk_perclk	USDHC1_CLK_ROOT	clk_enable_usdhc1 (CCGR108)
	usdhc2.CLK	NAND_USDHC_BUS_CLK_ROOT	clk_enable_usdhc2 (CCGR109)
	usdhc2.ipg_clk	IPG_CLK_ROOT	clk_enable_usdhc2 (CCGR109)
	usdhc2.ipg_clk_s	IPG_CLK_ROOT	clk_enable_usdhc2 (CCGR109)
	usdhc2.hclk	NAND_USDHC_BUS_CLK_ROOT	clk_enable_usdhc2 (CCGR109)
	usdhc2.ipg_clk_perclk	USDHC2_CLK_ROOT	clk_enable_usdhc2 (CCGR109)
	usdhc3.CLK	NAND_USDHC_BUS_CLK_ROOT	clk_enable_usdhc3 (CCGR110)
	usdhc3.ipg_clk	IPG_CLK_ROOT	clk_enable_usdhc3 (CCGR110)
	usdhc3.ipg_clk_s	IPG_CLK_ROOT	clk_enable_usdhc3 (CCGR110)
	usdhc3.hclk	NAND_USDHC_BUS_CLK_ROOT	clk_enable_usdhc3 (CCGR110)
	usdhc3.ipg_clk_perclk	USDHC3_CLK_ROOT	clk_enable_usdhc3 (CCGR110)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
WDOG	wdog1.ipg_master_clk	IPG_CLK_ROOT	clk_enable_wdog1 (CCGR156)
	wdog1.master_clk	IPG_CLK_ROOT	clk_enable_wdog1 (CCGR156)
	wdog1.ipg_slave_clk	WDOG_CLK_ROOT	clk_enable_wdog1 (CCGR156)
	wdog1.slave_clk	WDOG_CLK_ROOT	clk_enable_wdog1 (CCGR156)
	wdog1.ipg_clk	WDOG_CLK_ROOT	clk_enable_wdog1 (CCGR156)
	wdog1.ipg_clk_s	WDOG_CLK_ROOT	clk_enable_wdog1 (CCGR156)
	wdog2.ipg_master_clk	IPG_CLK_ROOT	clk_enable_wdog2 (CCGR157)
	wdog2.master_clk	IPG_CLK_ROOT	clk_enable_wdog2 (CCGR157)
	wdog2.ipg_slave_clk	WDOG_CLK_ROOT	clk_enable_wdog2 (CCGR157)
	wdog2.slave_clk	WDOG_CLK_ROOT	clk_enable_wdog2 (CCGR157)
	wdog2.ipg_clk	WDOG_CLK_ROOT	clk_enable_wdog2 (CCGR157)
	wdog2.ipg_clk_s	WDOG_CLK_ROOT	clk_enable_wdog2 (CCGR157)
	wdog3.ipg_master_clk	IPG_CLK_ROOT	clk_enable_wdog3 (CCGR158)
	wdog3.master_clk	IPG_CLK_ROOT	clk_enable_wdog3 (CCGR158)
	wdog3.ipg_slave_clk	WDOG_CLK_ROOT	clk_enable_wdog3 (CCGR158)
	wdog3.slave_clk	WDOG_CLK_ROOT	clk_enable_wdog3 (CCGR158)
	wdog3.ipg_clk	WDOG_CLK_ROOT	clk_enable_wdog3 (CCGR158)
	wdog3.ipg_clk_s	WDOG_CLK_ROOT	clk_enable_wdog3 (CCGR158)
	wdog4.ipg_master_clk	IPG_CLK_ROOT	clk_enable_wdog4 (CCGR159)
	wdog4.master_clk	IPG_CLK_ROOT	clk_enable_wdog4 (CCGR159)
	wdog4.ipg_slave_clk	WDOG_CLK_ROOT	clk_enable_wdog4 (CCGR159)
	wdog4.slave_clk	WDOG_CLK_ROOT	clk_enable_wdog4 (CCGR159)

Table continues on the next page...

Table 5-12. System Clocks and Gating (continued)

Module	Module Clock (instance.clock)	Clock Root	Module Clock Gating Enable (CCGR)
	wdog4.ipg_clk	WDOG_CLK_ROOT	clk_enable_wdog4 (CCGR159)
	wdog4.ipg_clk_s	WDOG_CLK_ROOT	clk_enable_wdog4 (CCGR159)

5.2.6 Functional Description

The following sections describe the functional details of CCM.

5.2.6.1 Input Clocks

The table below describes the input clock sources that supply the muxes to the clock slices in the clock root generator. Please see PLL Interface for PLL programming information.

Control Register	Input Clock	Offset	Frequency (MHz)	Description
CCM_PLL_CTRL0	CKIL_SYNC	0x800	0.032	CKIL synchronizer
CCM_PLL_CTRL1	ARM_PLL	0x810	800	ARM PLL
CCM_PLL_CTRL2	ARM_PLL_CLK	0x820	800	ARM PLL main output clock
CCM_PLL_CTRL3	DDR_PLL	0x830	1066	DDR PLL
CCM_PLL_CTRL4	DDR_PLL_DIV2	0x840	533	DDR PLL divided 2 clock output
CCM_PLL_CTRL5	DDR_PLL_CLK	0x850	1066	DDR PLL main output clock
CCM_PLL_CTRL6	SYS_PLL	0x860	480	System PLL
CCM_PLL_CTRL7	SYS_PLL_CLK	0x870	480	System PLL main output clock
CCM_PLL_CTRL8	SYS_PLL_DIV2	0x880	240	System PLL divided 2 clock output
CCM_PLL_CTRL9	SYS_PLL_DiV4	0x890	120	System PLL divided 4 clock output
CCM_PLL_CTRL10	SYS_PLL_PFD0	0x8A0	392	System PLL PFD0 clock
CCM_PLL_CTRL11	SYS_PLL_PFD0_DIV2	0x8B0	196	System PLL PFD0 divided 2 clock
CCM_PLL_CTRL12	SYS_PLL_PFD1	0x8C0	332	System PLL PFD1 clock
CCM_PLL_CTRL13	SYS_PLL_PFD1_DIV2	0x8D0	166	System PLL PFD1 divided 2 clock
CCM_PLL_CTRL14	SYS_PLL_PFD2	0x8E0	270	System PLL PFD2 clock
CCM_PLL_CTRL15	SYS_PLL_PFD2_DIV2	0x8F0	135	System PLL PFD2 divided 2 clock
CCM_PLL_CTRL16	SYS_PLL_PFD3	0x900	-	System PLL PFD3 clock
CCM_PLL_CTRL17	SYS_PLL_PFD4	0x910	-	System PLL PFD4 clock
CCM_PLL_CTRL18	SYS_PLL_PFD5	0x920	-	System PLL PFD5 clock
CCM_PLL_CTRL19	SYS_PLL_PFD6	0x930	-	System PLL PFD6 clock

Table continues on the next page...

Control Register	Input Clock	Offset	Frequency (MHz)	Description
CCM_PLL_CTRL20	SYS_PLL_PFD7	0x940	-	System PLL PFD7 clock
CCM_PLL_CTRL21	ENET_PLL	0x950	1000	ENET PLL
CCM_PLL_CTRL22	ENET_PLL_DIV2	0x960	500	Ethernet PLL divided 2 clock
CCM_PLL_CTRL23	ENET_PLL_DIV4	0x970	250	Ethernet PLL divided 4 clock
CCM_PLL_CTRL24	ENET_PLL_DIV8	0x980	125	Ethernet PLL divided 8 clock
CCM_PLL_CTRL25	ENET_PLL_DIV10	0x990	100	Ethernet PLL divided 10 clock
CCM_PLL_CTRL26	ENET_PLL_DIV20	0x9A0	50	Ethernet PLL divided 20 clock
CCM_PLL_CTRL27	ENET_PLL_DIV25	0x9B0	40	Ethernet PLL divided 25 clock
CCM_PLL_CTRL28	ENET_PLL_DIV40	0x9C0	25	Ethernet PLL divided 40 clock
CCM_PLL_CTRL29	AUDIO_PLL	0x9D0	66	Audio PLL
CCM_PLL_CTRL30	AUDIO_PLL_CLK	0x9E0	66	Audio PLL clock output
CCM_PLL_CTRL31	VIDEO_PLL	0x9F0	66	Video PLL
CCM_PLL_CTRL32	VIDEO_PLL_CLK	0xA00	66	Video PLL clock output
USB OTG2 PHY	USB_PLL		480	USB PHY clock output
XTALOSC	REF_1M		1	1M reference clock sourced from XTALOSC (sync with RCOSC 24M). REF_1M is also tied to 32K XTALOSC source.
external source	OSC_24M		24	Clock output of 24MHz crystal oscillator
external source	OSC_32K		0.032	32K oscillator clock output
external source	EXT_CLK1		-	Clock input from external IO
external source	EXT_CLK2		-	Clock input from external IO
external source	EXT_CLK3		-	Clock input from external IO
external source	EXT_CLK4		-	Clock input from external IO

NOTE

CKIL_SYNC needs to be configured the same as the PLL for ipg_clk

5.2.6.2 CKIL Synchronizer

Some on-chip peripherals need synchronization between their 32K clock inputs and ipg_clk inputs. When ipg_clk on peripheral is to be stopped, the synchronizer must be bypassed to avoid CKIL_SYNC stop. In bypass, the CKIL_SYNC comes directly from oscillator inside anatop.

CCM provide a synchronized version of 32K clock(CKIL_SYNC), which is generated by system IPG_CLK_ROOT. CKIL_SYNC is synchronized to IPG_CLK_ROOT when system is in functional mode. When the system entering low power mode that shutdown IPG_CLK_ROOT CKIL synchronizer need to be bypassed, and raw CKIL is supplied to the system.

Bypass signal control of 32K clock synchronizer comes from source control channel 0. This channel need to be configured exactly the same as the channel that control PLL output that used to generate IPG_CLK_ROOT. Typically, the PLL output is pll_sys_pfd2_135m_clk on channel 15.

Peripherals that use ip_sync may not run their ipg_clk on IPG_CLK_ROOT. So they cannot use synchronized 32K clock provided by CCM. Instead, they synchronized 32K clock use its ipg_clk. Bypass signal on these peripherals on 32K clock synchronizer comes from LPCG control of their ipg_clk. So in this case, to avoid problem on 32K input, LPCG must shutdown together with PLL clock output, which used for the ipg_clk generation.

5.2.6.3 Clock Components

The details of the CCM clock components are detailed below.

5.2.6.3.1 Clock Divider

Synchronized clock dividers can be used to make integer division on source clock frequency. The divider guarantees a clean clock signal on its output during the change of the divide factor. Dividers are performing 1/(N+1) divide. A glitch can cause a sync-divider to go into an unrecoverable state, therefore a clean clock signal must be provided to a sync-divider.

When updating the divider value, a read_en signal should be deasserted to stop the fetch divider value from the related control registers. To change a divider factor value, the bus may enter a transition state and latch into the divide logic. After the divide value is changed, read_en needs to be reasserted. The divider will begin to update its internal divide logic. After it finishes updating, the divider will send a sw_ack signal to inform new value is in effect.

5.2.6.3.2 Safe Multiplexer

Safe multiplexers can guarantee a clean clock signal when switching between 2 clock sources. Both clock inputs must be active when switching. Glitches in the selected source may be transferred to the output clock while the de-selected source is blocked. Glitches that occur during switching may cause an unpredictable state, and will recover in 2~3 cycles.

Safe multiplexers first shutdown the current active clock, then switch. After receiving an acknowledge that the current clock stopped. The multiplexer turns on the new selected clock source. As the acknowledge of turning on the new clock source is received, the source switch finishes.

5.2.6.3.3 Clock Gate

A clock gate cell is used to gate clocks. When gated, the clock will stop at 0. A clock gate cell can accept glitches on its clock inputs. If the gate cell is off, it will block or absorb glitches. If the gate cell is on, it may pass glitches to its output. A gate cell needs 2~3 clock cycles to change states. The state during this period is either on or off and will recover in 2~3 cycles, but cannot be predicted which state it is in during this period.

5.2.6.3.4 8 to 1 Multiplexer

The 8-to-1 multiplexer is a combinational multiplexer that can switch anytime. The multiplexer output does not guarantee a clean clock signal. During switching, the output must be gated or all 8 inputs will require the same value.

5.2.6.4 Clock Slices

There are several types of clock generation slices in CCM. The slices are categorized as Core, Bus, Peripheral (IP), and DRAM.

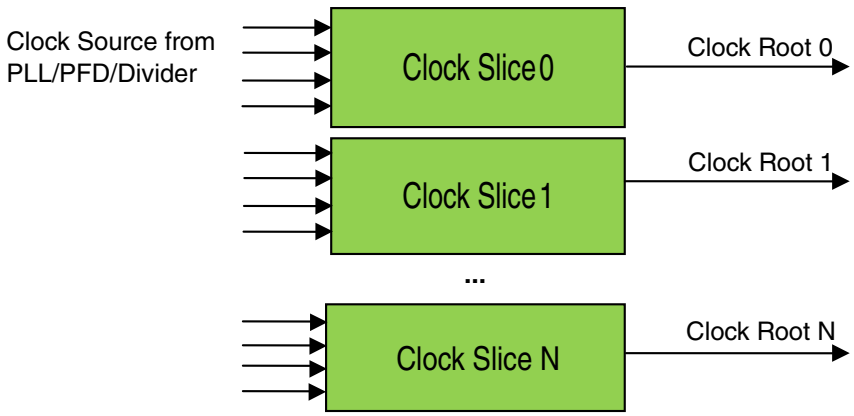


Figure 5-16. Clock Slices

5.2.6.4.1 Core clock slice

Core clock slices are designed for high speed, non-stop clock generation, typically for ARM core. Core clock slice is comprised of a post divider and safe multiplexer. Each has a clock gate and a clock multiplexer inside. To run at high frequency, post divider is 3 bits, which is half the bit width of other slices. The two 8-to-1 multiplexers are not glitch-less, so switching them should only be done when their output is not routed to a clock output.

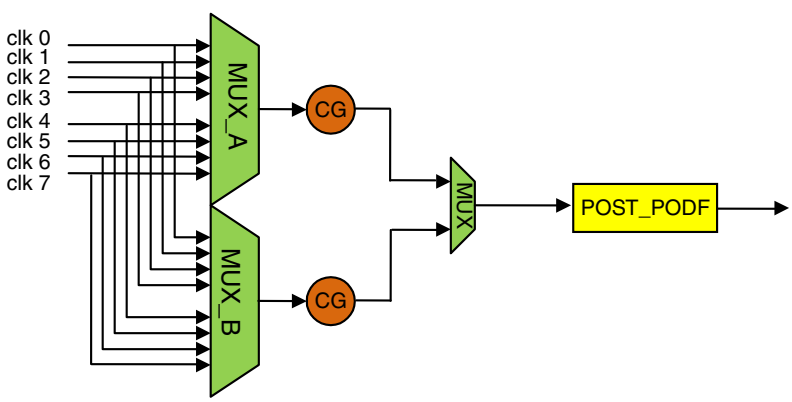


Figure 5-17. Core clock slice

5.2.6.4.2 Bus clock slice

Bus clock slices are comprised of post divider and a safe multiplexer. Each has a pre-divider, clock gate and a clock multiplexer inside. The pre-divider is three bits and provides a maximum division factor of 8. The post-divider is six bits and can divide

down the clock input to 1MHz on clock output. The two 8-to-1 multiplexers are not glitch-less, so switching should only be done when their output is not routed to clock output.

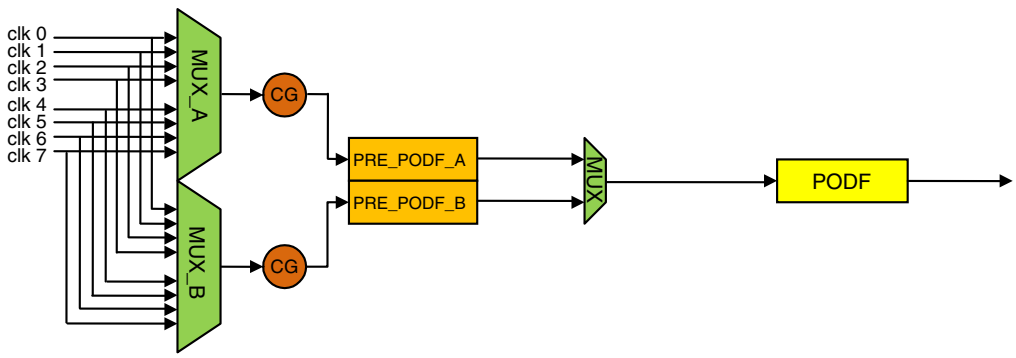


Figure 5-18. Bus clock slice

5.2.6.4.3 Peripheral clock slice

IP peripheral clock slices are comprised of a post-divider, pre-divider, clock gate and a clock multiplexer. The pre-divider is three bit and can divide down by a factor of 8. The post-divider is six bit and can divide down the input clock to 1MHz on clock output. The 8-to-1 multiplexer is not glitch-less, so switching should only be done when their output is not routed to clock output.

IP clock slices must be stopped to change the clock source.

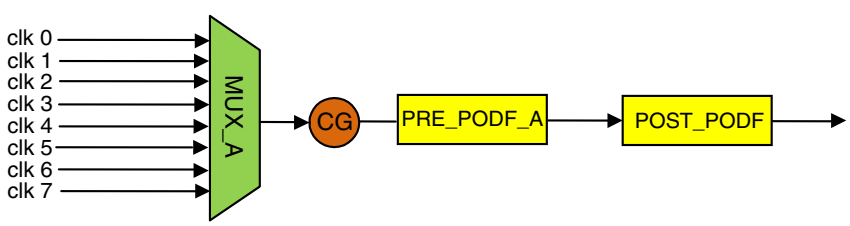


Figure 5-19. Peripheral clock slice

5.2.6.4.4 DRAM clock slice

DRAM clock slice is comprised of a safe multiplexer and a post-divider. The post-divider is a three bit width synchronized divider. This slice generates clock 2x for the DDR controller. Dram_alt_clk is a CCM generated clock while PLL_DRAM comes directly from the PLL, which may ensure a better duty-cycle on clock output.

DRAM PHYM clock slice is comprised of a safe multiplexer and a fixed divider with a divide factor of two. The PHYM slice generates the phy_clkm for DDR PHY. Dram_phym_alt_clk is a CCM generated clock while PLL_DRAM comes directly from the PLL, which may ensure a better duty-cycle on clock output.

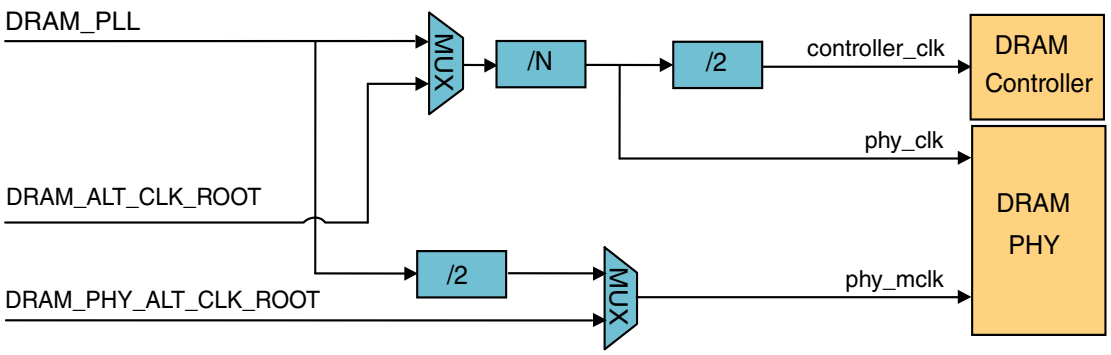


Figure 5-20. DRAM clock slice

5.2.6.5 Clock gate control

CCM can perform automatic clock shutdown of on-chip peripherals according to system low power mode. Each logic domain can respectively declare its dependency level on each clock. If a clock is detected not dependant on any domain, it will be shutdown to save power.

Clock generation inside CCM creates clock root for on-chip peripherals. Before clock root goes into peripherals via low power clock gating cells. By controlling these LPCGs, CCM can manage on-chip peripheral clocks.

Clock gate controls are active clock gating, which means it needs the clock root to be active while low power clock gating. Clock generation module does only multiplexing, gating and dividing on clock sources. So, clock root from generation module will stop when the corresponding clock source stops.

It is important that software maintains the clock source driving the LPCG stay active while the LPCG is turned on or off.

5.2.6.6 Clock source control

CCM can perform automatic PLL shutdown of ANATOP according to system low power mode. Each logic domain can respectively declare its dependency level on each clock. If a PLL is detected not depended by any domain, it will be shutdown to save power.

PLL in ANATOP can be set be controlled by CCM via setting some CCM override bits. Also there is control on 32K IPG_CLK_ROOT synchronized version inside clock source control module.

For PLLs, we have channel on every PLL, every PFD and every divider. PLL is the source of PFDs and dividers. For any clock, its source must be left on when it is kept on. Behavior is undefined if this rule is violated.

For a shutdown clock source, if it is declared depended by writing clock source control registers, the controlling logic will turn on the source immediately, while the setting goes into a shadow register. After clock source get ready, the setting will be accepted by source control logic, and copied from shadow register to setting register.

5.2.6.7 Access control

CCM can implement its own access control based on domain, to provide more precise control on shared resource. Access controls are implemented on clock root generation, clock gate control, and clock source control. Access control logic will never impact on read access, but will block unauthentic write access.

Access controls on clock root generation are independent between every clock root. A sticky authentic fail flag will be set if authentic check failed when a domain is trying to write to some register. There are a whitelist and a semaphore inside access control logic. For each clock root, access control logics are default disabled, after power on reset. Software can enable access control anytime after that. But once access logic is enabled, it cannot be disabled until next power on reset.

Table 5-13. Whitelist

Enabled	Write access will be authenticated before being performed.
Disabled	every access on protected item will be performed.

NOTE

Only domains that are on the whitelist can perform write access to this clock root when access control is enabled.

Table 5-14. Semaphore

Enabled	A domain must obtain the semaphore's ownership before its write access can be authenticated. Only a domain on the whitelist can obtain the ownership and the ownership will last until the domain explicitly releases the ownership. Semaphore obtain will fail if it is already fetched by some other domain.
Disabled	Authentic check will check only on whitelist.

NOTE

Semaphore is intended to help software keep clock root from unexpectedly changing or keep operation.

Access control of clock gate and clock source control is performed in a simple operation. Every domain can only write on the bits for it's own setting. Any write to irrespective domain will be ignored.

5.2.6.8 System level considerations

Clock shutdown strategy

Any a clock shutdown should be first LPCG, then PLL. If LPCG is configured not to shutdown, clock root for then LPCG should not stop either. Violation of this rule will lead system into an unpredictable state.

Core clock root frequency

If core clock is set lower that one third of IPG clock, SRC needs to generate a longer reset signal to match requirement from ARM core. This typically happened when run ARM core at some divided XTAL 24M while IPG clock is supplied by PLL.

DRAM clock

DRAM PHYM clock needs a clock faster that 400MHz.

USB OTG CLOCK

USB clock may not be a reliable clock source in some applications. This clock may stop when USB cable is disconnected.

5.2.7 Programming Guide

5.2.7.1 Set, Clear, and Toggle register features

Every register of CCM have set, clear, toggle feature. Set register locate at address of register base address + 0x04, clear register locate at register base address + 0x08, and toggle located at base address + 0x0c. Read from all 4 registers to get the current register value. Write to base register will set register to write value. Write 1 to set register bits will set them to 1, while write 0 have no effect. Write 1 to clear register bits will clear them to 0, while write 0 have no effect. Write 1 to set register bits will set them to invert value, while write 0 have no effect.

5.2.7.2 PLL Interface

CCM can control PLLs inside CCM_ANALOG when entering or leaving low-power mode. Software must set the PLL override inside CCM_ANALOG before entering low-power mode after power-on reset (POR).

There are four levels of low-power modes in a logic domain:

- Not needed
- Needed in RUN
- Needed in RUN and WAIT
- Needed in RUN, WAIT, and STOP

CCM only takes action while domain status are switching between STOP (DEEP SLEEP mode is considered the same as STOP). There are 4 domains that can be assigned. Any CPU platform can be assigned to any domain by RDC. If a domain is empty, the domain is considered as STOP.

Each domain can declare its dependency to CCM. The use of any clock, without declaring it in its own domain, is not permitted. A domain declares its dependency on a clock by writing the dependency level. Settings against behavior in low-power mode are as follows:

Table 5-15. Domain Dependency

Domain Level	Run	Wait	Stop / Deep sleep
0			
1	Required		
2	Required	Required	
3	Required	Required	Required

Table 5-16. CCGR Program Interface

CCGR		Domain3				Domain2				Domain1				Domain0			
	31-16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Level1	Level0			Level1	Level0			Level1	Level0			Level1	Level0

Each domain can change only bits assigned to control access. Any irrelevant write to it will be ignored. For example, Domain 0 can only write to bits [1:0]. Any bits written to, other than bits [1:0], will be ignored. Other domains can read all of the other domain settings. The default value for domain 0 is 2, and will enter STOP mode after shutting down. When the default value of the other domain setting is 0, it will not be required.

When setting clock source, the settings will not take effect immediately. The setting will enter the shadow register first. If a PLL shutdown or new setting enters the shadow register to declare dependency on the PLL, the PLL will turn on immediately. When the PLL is ready, the setting in shadow register will be updated to the new setting. During this period, the pending bit will be set and cleared. Then CCM will send the PLL control signal as a shadow register and inform GPC the PLL status according to the setting register. In other cases, the setting will be updated from the shadow register immediately. Clock sources have dependency on each other.

NOTE

Do not shutdown the parent clock when the required child clock is active. Attempting to do so will lead to unpredicable and unrecoverable behavior. It is recommended to shutdown the parent clock and child clock together.

5.2.7.3 CCGR Interface

Before a clock root goes to on–chip peripherals, the clock root is distributed through low power clock gates (LPCG). These LPCG are implemented to automatically perform clock shutdown when a domain enters and leaves a low-power state.

There are four levels of low-power modes in a logic domain:

- Not needed
- Needed in RUN
- Needed in RUN and WAIT
- Needed in RUN, WAIT, and STOP

CCM only takes action while domain status are switching between STOP (DEEP SLEEP mode is considered the same as STOP). There are 4 domains that can be assigned. Any CPU platform can be assigned to any domain by RDC. If a domain is empty, the domain is considered as STOP.

Each domain can declare its dependency to CCM. The use of any clock, without declaring it in its own domain, is not permitted. A domain declares its dependency on a clock by writing the dependency level. Settings against behavior in low-power mode are as follows:

Table 5-17. Domain Dependency

Domain Level	Run	Wait	Stop / Deep sleep
0			
1	Required		
2	Required	Required	

Table continues on the next page...

Table 5-17. Domain Dependency (continued)

3	Required	Required	Required

Table 5-18. CCGR Program Interface

CCGR		Domain3				Domain2				Domain1				Domain0			
	31-16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Level1	Level0			Level1	Level0			Level1	Level0			Level1	Level0

Each domain can change only bits assigned to control access. Any irrelevant write to it will be ignored. For example, Domain 0 can only write to bits [1:0]. Any bits written to, other than bits [1:0], will be ignored. Other domains can read all of the other domain settings. The default value for domain 0 is 2, and will enter STOP mode after shutting down. When the default value of the other domain setting is 0, it will not be required.

The table below lists the CCM Clock Gating Register (CCGR) and associated offset for each LPCG enable.

NOTE

Not all CCGRs are mapped.

Table 5-19. CCGR Mapping Table

Gating Register	LPCG Enable	Offset
CCM_CCGR4	sim_main	0x4040
CCM_CCGR5	sim_display	0x4050
CCM_CCGR6	sim_enet	0x4060
CCM_CCGR7	sim_m	0x4070
CCM_CCGR8	sim_s	0x4080
CCM_CCGR9	sim_wakeup	0x4090
CCM_CCGR10	ipmux1	0x40A0
CCM_CCGR11	ipmux2	0x40B0
CCM_CCGR12	ipmux3	0x40C0
CCM_CCGR16	rom	0x4100
CCM_CCGR17	ocram	0x4110
CCM_CCGR18	ocram_s	0x4120
CCM_CCGR19	dram	0x4130
CCM_CCGR20	rawnand	0x4140
CCM_CCGR21	qspi	0x4150
CCM_CCGR22	weim	0x4160
CCM_CCGR32	adc	0x4200

Table continues on the next page...

Table 5-19. CCGR Mapping Table (continued)

CCM_CCGR33	anatop	0x4210
CCM_CCGR34	sctr	0x4220
CCM_CCGR35	ocotp	0x4230
CCM_CCGR36	caam	0x4240
CCM_CCGR37	snvs	0x4250
CCM_CCGR38	rdc	0x4260
CCM_CCGR39	mu	0x4270
CCM_CCGR40	hs	0x4280
CCM_CCGR41	dvfs	0x4290
CCM_CCGR42	qos	0x42A0
CCM_CCGR43	qos_dispmix	0x42B0
CCM_CCGR44	qos_megamix	0x42C0
CCM_CCGR45	csu	0x42D0
CCM_CCGR46	dbgmon	0x42E0
CCM_CCGR47	debug	0x42F0
CCM_CCGR48	trace	0x4300
CCM_CCGR49	sec_debug	0x4310
CCM_CCGR64	sema1	0x4400
CCM_CCGR65	sema2	0x4410
CCM_CCGR68	perfmon1	0x4440
CCM_CCGR69	perfmon2	0x4450
CCM_CCGR72	sdma	0x4480
CCM_CCGR73	csi	0x4490
CCM_CCGR75	lcdif	0x44B0
CCM_CCGR76	pxp	0x44C0
CCM_CCGR100	mipi_csi	0x4640
CCM_CCGR101	mipi_dsi	0x4650
CCM_CCGR102	mipi_mem_phy	0x4660
CCM_CCGR104	usb_ctrl	0x4680
CCM_CCGR105	usb_hsic	0x4690
CCM_CCGR106	usb_phy1	0x46A0
CCM_CCGR108	usdhc1	0x46C0
CCM_CCGR109	usdhc2	0x46D0
CCM_CCGR110	usdhc3	0x46E0
CCM_CCGR112	enet1	0x4700
CCM_CCGR116	can1	0x4740
CCM_CCGR117	can2	0x4750
CCM_CCGR120	ecspi1	0x4780
CCM_CCGR121	ecspi2	0x4790
CCM_CCGR122	ecspi3	0x47A0

Table continues on the next page...

Table 5-19. CCGR Mapping Table (continued)

CCM_CCGR123	ecspi4	0x47B0
CCM_CCGR124	gpt1	0x47C0
CCM_CCGR125	gpt2	0x47D0
CCM_CCGR126	gpt3	0x47E0
CCM_CCGR127	gpt4	0x47F0
CCM_CCGR128	ftm1	0x4800
CCM_CCGR129	ftm2	0x4810
CCM_CCGR132	pwm1	0x4840
CCM_CCGR133	pwm2	0x4850
CCM_CCGR134	pwm3	0x4860
CCM_CCGR135	pwm4	0x4870
CCM_CCGR136	i2c1	0x4880
CCM_CCGR137	i2c2	0x4890
CCM_CCGR138	i2c3	0x48A0
CCM_CCGR139	i2c4	0x48B0
CCM_CCGR140	sai1	0x48C0
CCM_CCGR141	sai2	0x48D0
CCM_CCGR142	sai3	0x48E0
CCM_CCGR144	sim1	0x4900
CCM_CCGR145	sim2	0x4910
CCM_CCGR148	uart1	0x4940
CCM_CCGR149	uart2	0x4950
CCM_CCGR150	uart3	0x4960
CCM_CCGR151	uart4	0x4970
CCM_CCGR152	uart5	0x4980
CCM_CCGR153	uart6	0x4990
CCM_CCGR154	uart7	0x49A0
CCM_CCGR156	wdog1	0x49C0
CCM_CCGR157	wdog2	0x49D0
CCM_CCGR158	wdog3	0x49E0
CCM_CCGR159	wdog4	0x49F0
CCM_CCGR160	gpio1	0x4A00
CCM_CCGR161	gpio2	0x4A10
CCM_CCGR162	gpio3	0x4A20
CCM_CCGR163	gpio4	0x4A30
CCM_CCGR164	gpio5	0x4A40
CCM_CCGR165	gpio6	0x4A50
CCM_CCGR166	gpio7	0x4A60
CCM_CCGR168	iomux	0x4A80
CCM_CCGR169	iomux_lpsr	0x4A90

Table continues on the next page...

Table 5-19. CCGR Mapping Table (continued)

CCM_CCGR170	kpp	0x4AA0

5.2.7.4 Target Interface

Target interface is optimized to simplify software operation. Using this interface, all clock roots are in the same program model with the same register bit field mapping. Software need not care any detail inside clock slice, even clock slice types. Software just writes to the register the settings it desired, and internal hardware logic will generate a safe sequence to achieve that setting.

Target interface needs software provide whether this clock need active, clock source number to be selected, pre divider value, and post divide value. If a clock slice does not support some setting, the setting is simply ignored, and will not do any harm to supported fields.

Freq = (clock source freq)/(pre_div+1)/(post_div+1)

Internal logic sequence of target interface guarantees a clean clock on output without frequency overshoot. But needs software to guarantee at least current selecting and target clock source is active.

Sequence does not variant against current state and target setting. It always take as worst case on every filed. It first open all clocks, then apply slow down change on divider values, the switch to new selected clock source, then change on speed up dividers, finally shutdown it if new setting request that.

Do during change setting via target interface, clock output are always active. For intermediate frequency, it always chooses the lower one to avoid frequency overshoot for ip clock root. For bus, core, and ahb slice, there is an safe multiplexer and two clock arms, the pre divider on current setting will not be changed. If user keeps frequency in safe multiplexer close, the output will be smooth when switching.

The write operation on a target interface will never return until output clock is running at desired setting. Software does not need polling or read any register for clock stable.

STEP	STATE	OPERATION
0	SMART_IDLE	Idle state, no write operation is pending
1	SMART_WAIT_READY	State get this state when receive write access, wait for every field ready
2	SMART_APPLY_GATE1	open all branched, all gates inside clock slices
3	SMART_WAIT_GATE1	Wait for gate applied

Table continues on the next page...

4	SMART_APPLY_PODF1	apply post divider and post divider for if new value generate slower clock
5	SMART_WAIT_PODF1	Wait for divider accept new value
6	SMART_APPLY_GATE2	shutdown spare branch if exist, else shutdown working branch
7	SMART_WAIT_GATE2	Wait for shutdown operation complete
8	SMART_APPLY_MUX	Change multiplexer to new source on spare branch if exist, else switch working one
9	SMART_APPLY_GATE3	open gates on all branches
10	SMART_WAIT_GATE3	Wait for gates opened
11	SMART_APPLY_SWITCH	Switch safe multiplexer if there is one
12	SMART_WAIT_SWITCH	Wait for safe multiplexer switch
13	SMART_APPLY_PODF2	apply post divider and post divider for if new value generate faster clock
14	SMART_WAIT_PODF2	Wait for divider accept new value
15	SMART_APPLY_GATE4	apply clock gate setting, shutdown spare one if there is
16	SMART_WAIT_GATE4	Wait for gate applied
17	SMART_APPLY_AUTO	apply auto and auto divider
18	SMART_DONE	Finish, wait for bus operation complete

5.2.7.5 Normal Interface

Normal interface provide more controllable thing that target interface. And also provide protections against dangerous operation.

Normal interface provides safe sequences to handle each clock component, divider, gate, multiplexer. But it is software that needs to care the order and relationship between updating components.

Writing to this interface will complete immediately, and internal logic will continue try to apply written values to clock components. A busy flag will be assert during applying.

Field access rule:

1. Only one field can be modified a time
2. No field can be modified when any field pending
3. Not violate change condition in following table

FIELD	CHANGE CONDITION	FINISH CONDITION
Auto		immediate
Auto-divider		immediate
Bypass	Gatea active and gateb active	Bypass switch complete

Table continues on the next page...

Post-divider		New divider value applied
Gate B	Bypass disable	New gate value active
Pre-divider B	Bypass disable and gateb not gated	New divider value applied
MUX B	Bypass disable and gateb gating	immediate
Gate A	Bypass	New gate value active
Pre-divider A	Bypass and gatea not gated	New divider value applied
MUX A	Bypass and gatea gating	immediate

- Error will be reported if access rules violated.
- Unsafe or ambiguous access will be ignored.

If a write access as blocked by normal interface, the write operation will be ignored. And a sticky bit “violate” will be set. The bit will last until software clears it explicitly. The violate bit is 4 bits inside CCM, each for a logic domain. Each domain can read and clear the bit for itself, the bits for other domain is neither visible nor clearable.

5.2.8 CCM Memory Map/Register Definition

The Memory Map below represents the full array for CCM.

NOTE

Not all mapped Clock Slices and CCGRs are tied to functional components.

Please see the following for the functional mapping tables and information:

- CCM_PLL_CTRL - Input Clocks
- CCM_TARGET_ROOT - Clock Root Selects
- CCM_CCGR - CCGR Interface

CCM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_0000	General Purpose Register (CCM_GPR0)	32	R/W	0000_0000h	5.2.8.1/ 682
3038_0004	General Purpose Register (CCM_GPR0_SET)	32	R/W	0000_0000h	5.2.8.1/ 682
3038_0008	General Purpose Register (CCM_GPR0_CLR)	32	R/W	0000_0000h	5.2.8.1/ 682
3038_000C	General Purpose Register (CCM_GPR0_TOG)	32	R/W	0000_0000h	5.2.8.1/ 682
3038_0800	CCM PLL Control Register (CCM_PLL_CTRL0)	32	R/W	0000_0002h	5.2.8.2/ 683

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_0804	CCM PLL Control Register (CCM_PLL_CTRL0_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0808	CCM PLL Control Register (CCM_PLL_CTRL0_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_080C	CCM PLL Control Register (CCM_PLL_CTRL0_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0810	CCM PLL Control Register (CCM_PLL_CTRL1)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0814	CCM PLL Control Register (CCM_PLL_CTRL1_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0818	CCM PLL Control Register (CCM_PLL_CTRL1_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_081C	CCM PLL Control Register (CCM_PLL_CTRL1_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0820	CCM PLL Control Register (CCM_PLL_CTRL2)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0824	CCM PLL Control Register (CCM_PLL_CTRL2_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0828	CCM PLL Control Register (CCM_PLL_CTRL2_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_082C	CCM PLL Control Register (CCM_PLL_CTRL2_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0830	CCM PLL Control Register (CCM_PLL_CTRL3)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0834	CCM PLL Control Register (CCM_PLL_CTRL3_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0838	CCM PLL Control Register (CCM_PLL_CTRL3_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_083C	CCM PLL Control Register (CCM_PLL_CTRL3_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0840	CCM PLL Control Register (CCM_PLL_CTRL4)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0844	CCM PLL Control Register (CCM_PLL_CTRL4_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0848	CCM PLL Control Register (CCM_PLL_CTRL4_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_084C	CCM PLL Control Register (CCM_PLL_CTRL4_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0850	CCM PLL Control Register (CCM_PLL_CTRL5)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0854	CCM PLL Control Register (CCM_PLL_CTRL5_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0858	CCM PLL Control Register (CCM_PLL_CTRL5_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_085C	CCM PLL Control Register (CCM_PLL_CTRL5_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0860	CCM PLL Control Register (CCM_PLL_CTRL6)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0864	CCM PLL Control Register (CCM_PLL_CTRL6_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0868	CCM PLL Control Register (CCM_PLL_CTRL6_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_086C	CCM PLL Control Register (CCM_PLL_CTRL6_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0870	CCM PLL Control Register (CCM_PLL_CTRL7)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0874	CCM PLL Control Register (CCM_PLL_CTRL7_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0878	CCM PLL Control Register (CCM_PLL_CTRL7_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_087C	CCM PLL Control Register (CCM_PLL_CTRL7_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0880	CCM PLL Control Register (CCM_PLL_CTRL8)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0884	CCM PLL Control Register (CCM_PLL_CTRL8_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0888	CCM PLL Control Register (CCM_PLL_CTRL8_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_088C	CCM PLL Control Register (CCM_PLL_CTRL8_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0890	CCM PLL Control Register (CCM_PLL_CTRL9)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0894	CCM PLL Control Register (CCM_PLL_CTRL9_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0898	CCM PLL Control Register (CCM_PLL_CTRL9_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_089C	CCM PLL Control Register (CCM_PLL_CTRL9_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_08A0	CCM PLL Control Register (CCM_PLL_CTRL10)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_08A4	CCM PLL Control Register (CCM_PLL_CTRL10_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_08A8	CCM PLL Control Register (CCM_PLL_CTRL10_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_08AC	CCM PLL Control Register (CCM_PLL_CTRL10_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_08B0	CCM PLL Control Register (CCM_PLL_CTRL11)	32	R/W	0000_0002h	5.2.8.2/ 683

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_08B4	CCM PLL Control Register (CCM_PLL_CTRL11_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_08B8	CCM PLL Control Register (CCM_PLL_CTRL11_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_08BC	CCM PLL Control Register (CCM_PLL_CTRL11_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_08C0	CCM PLL Control Register (CCM_PLL_CTRL12)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_08C4	CCM PLL Control Register (CCM_PLL_CTRL12_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_08C8	CCM PLL Control Register (CCM_PLL_CTRL12_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_08CC	CCM PLL Control Register (CCM_PLL_CTRL12_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_08D0	CCM PLL Control Register (CCM_PLL_CTRL13)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_08D4	CCM PLL Control Register (CCM_PLL_CTRL13_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_08D8	CCM PLL Control Register (CCM_PLL_CTRL13_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_08DC	CCM PLL Control Register (CCM_PLL_CTRL13_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_08E0	CCM PLL Control Register (CCM_PLL_CTRL14)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_08E4	CCM PLL Control Register (CCM_PLL_CTRL14_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_08E8	CCM PLL Control Register (CCM_PLL_CTRL14_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_08EC	CCM PLL Control Register (CCM_PLL_CTRL14_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_08F0	CCM PLL Control Register (CCM_PLL_CTRL15)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_08F4	CCM PLL Control Register (CCM_PLL_CTRL15_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_08F8	CCM PLL Control Register (CCM_PLL_CTRL15_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_08FC	CCM PLL Control Register (CCM_PLL_CTRL15_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0900	CCM PLL Control Register (CCM_PLL_CTRL16)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0904	CCM PLL Control Register (CCM_PLL_CTRL16_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0908	CCM PLL Control Register (CCM_PLL_CTRL16_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_090C	CCM PLL Control Register (CCM_PLL_CTRL16_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0910	CCM PLL Control Register (CCM_PLL_CTRL17)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0914	CCM PLL Control Register (CCM_PLL_CTRL17_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0918	CCM PLL Control Register (CCM_PLL_CTRL17_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_091C	CCM PLL Control Register (CCM_PLL_CTRL17_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0920	CCM PLL Control Register (CCM_PLL_CTRL18)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0924	CCM PLL Control Register (CCM_PLL_CTRL18_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0928	CCM PLL Control Register (CCM_PLL_CTRL18_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_092C	CCM PLL Control Register (CCM_PLL_CTRL18_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0930	CCM PLL Control Register (CCM_PLL_CTRL19)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0934	CCM PLL Control Register (CCM_PLL_CTRL19_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0938	CCM PLL Control Register (CCM_PLL_CTRL19_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_093C	CCM PLL Control Register (CCM_PLL_CTRL19_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0940	CCM PLL Control Register (CCM_PLL_CTRL20)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0944	CCM PLL Control Register (CCM_PLL_CTRL20_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0948	CCM PLL Control Register (CCM_PLL_CTRL20_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_094C	CCM PLL Control Register (CCM_PLL_CTRL20_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0950	CCM PLL Control Register (CCM_PLL_CTRL21)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0954	CCM PLL Control Register (CCM_PLL_CTRL21_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0958	CCM PLL Control Register (CCM_PLL_CTRL21_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_095C	CCM PLL Control Register (CCM_PLL_CTRL21_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0960	CCM PLL Control Register (CCM_PLL_CTRL22)	32	R/W	0000_0002h	5.2.8.2/ 683

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_0964	CCM PLL Control Register (CCM_PLL_CTRL22_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0968	CCM PLL Control Register (CCM_PLL_CTRL22_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_096C	CCM PLL Control Register (CCM_PLL_CTRL22_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0970	CCM PLL Control Register (CCM_PLL_CTRL23)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0974	CCM PLL Control Register (CCM_PLL_CTRL23_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0978	CCM PLL Control Register (CCM_PLL_CTRL23_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_097C	CCM PLL Control Register (CCM_PLL_CTRL23_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0980	CCM PLL Control Register (CCM_PLL_CTRL24)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0984	CCM PLL Control Register (CCM_PLL_CTRL24_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0988	CCM PLL Control Register (CCM_PLL_CTRL24_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_098C	CCM PLL Control Register (CCM_PLL_CTRL24_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0990	CCM PLL Control Register (CCM_PLL_CTRL25)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0994	CCM PLL Control Register (CCM_PLL_CTRL25_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0998	CCM PLL Control Register (CCM_PLL_CTRL25_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_099C	CCM PLL Control Register (CCM_PLL_CTRL25_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_09A0	CCM PLL Control Register (CCM_PLL_CTRL26)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_09A4	CCM PLL Control Register (CCM_PLL_CTRL26_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_09A8	CCM PLL Control Register (CCM_PLL_CTRL26_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_09AC	CCM PLL Control Register (CCM_PLL_CTRL26_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_09B0	CCM PLL Control Register (CCM_PLL_CTRL27)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_09B4	CCM PLL Control Register (CCM_PLL_CTRL27_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_09B8	CCM PLL Control Register (CCM_PLL_CTRL27_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_09BC	CCM PLL Control Register (CCM_PLL_CTRL27_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_09C0	CCM PLL Control Register (CCM_PLL_CTRL28)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_09C4	CCM PLL Control Register (CCM_PLL_CTRL28_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_09C8	CCM PLL Control Register (CCM_PLL_CTRL28_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_09CC	CCM PLL Control Register (CCM_PLL_CTRL28_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_09D0	CCM PLL Control Register (CCM_PLL_CTRL29)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_09D4	CCM PLL Control Register (CCM_PLL_CTRL29_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_09D8	CCM PLL Control Register (CCM_PLL_CTRL29_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_09DC	CCM PLL Control Register (CCM_PLL_CTRL29_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_09E0	CCM PLL Control Register (CCM_PLL_CTRL30)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_09E4	CCM PLL Control Register (CCM_PLL_CTRL30_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_09E8	CCM PLL Control Register (CCM_PLL_CTRL30_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_09EC	CCM PLL Control Register (CCM_PLL_CTRL30_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_09F0	CCM PLL Control Register (CCM_PLL_CTRL31)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_09F4	CCM PLL Control Register (CCM_PLL_CTRL31_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_09F8	CCM PLL Control Register (CCM_PLL_CTRL31_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_09FC	CCM PLL Control Register (CCM_PLL_CTRL31_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_0A00	CCM PLL Control Register (CCM_PLL_CTRL32)	32	R/W	0000_0002h	5.2.8.2/ 683
3038_0A04	CCM PLL Control Register (CCM_PLL_CTRL32_SET)	32	R/W	0000_0002h	5.2.8.3/ 685
3038_0A08	CCM PLL Control Register (CCM_PLL_CTRL32_CLR)	32	R/W	0000_0002h	5.2.8.4/ 687
3038_0A0C	CCM PLL Control Register (CCM_PLL_CTRL32_TOG)	32	R/W	0000_0002h	5.2.8.5/ 689
3038_4000	CCM Clock Gating Register (CCM_CCGR0)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_4004	CCM Clock Gating Register (CCM_CCGR0_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4008	CCM Clock Gating Register (CCM_CCGR0_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_400C	CCM Clock Gating Register (CCM_CCGR0_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4010	CCM Clock Gating Register (CCM_CCGR1)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4014	CCM Clock Gating Register (CCM_CCGR1_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4018	CCM Clock Gating Register (CCM_CCGR1_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_401C	CCM Clock Gating Register (CCM_CCGR1_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4020	CCM Clock Gating Register (CCM_CCGR2)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4024	CCM Clock Gating Register (CCM_CCGR2_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4028	CCM Clock Gating Register (CCM_CCGR2_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_402C	CCM Clock Gating Register (CCM_CCGR2_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4030	CCM Clock Gating Register (CCM_CCGR3)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4034	CCM Clock Gating Register (CCM_CCGR3_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4038	CCM Clock Gating Register (CCM_CCGR3_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_403C	CCM Clock Gating Register (CCM_CCGR3_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4040	CCM Clock Gating Register (CCM_CCGR4)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4044	CCM Clock Gating Register (CCM_CCGR4_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4048	CCM Clock Gating Register (CCM_CCGR4_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_404C	CCM Clock Gating Register (CCM_CCGR4_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4050	CCM Clock Gating Register (CCM_CCGR5)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4054	CCM Clock Gating Register (CCM_CCGR5_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4058	CCM Clock Gating Register (CCM_CCGR5_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_405C	CCM Clock Gating Register (CCM_CCGR5_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4060	CCM Clock Gating Register (CCM_CCGR6)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4064	CCM Clock Gating Register (CCM_CCGR6_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4068	CCM Clock Gating Register (CCM_CCGR6_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_406C	CCM Clock Gating Register (CCM_CCGR6_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4070	CCM Clock Gating Register (CCM_CCGR7)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4074	CCM Clock Gating Register (CCM_CCGR7_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4078	CCM Clock Gating Register (CCM_CCGR7_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_407C	CCM Clock Gating Register (CCM_CCGR7_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4080	CCM Clock Gating Register (CCM_CCGR8)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4084	CCM Clock Gating Register (CCM_CCGR8_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4088	CCM Clock Gating Register (CCM_CCGR8_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_408C	CCM Clock Gating Register (CCM_CCGR8_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4090	CCM Clock Gating Register (CCM_CCGR9)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4094	CCM Clock Gating Register (CCM_CCGR9_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4098	CCM Clock Gating Register (CCM_CCGR9_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_409C	CCM Clock Gating Register (CCM_CCGR9_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_40A0	CCM Clock Gating Register (CCM_CCGR10)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_40A4	CCM Clock Gating Register (CCM_CCGR10_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_40A8	CCM Clock Gating Register (CCM_CCGR10_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_40AC	CCM Clock Gating Register (CCM_CCGR10_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_40B0	CCM Clock Gating Register (CCM_CCGR11)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_40B4	CCM Clock Gating Register (CCM_CCGR11_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_40B8	CCM Clock Gating Register (CCM_CCGR11_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_40BC	CCM Clock Gating Register (CCM_CCGR11_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_40C0	CCM Clock Gating Register (CCM_CCGR12)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_40C4	CCM Clock Gating Register (CCM_CCGR12_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_40C8	CCM Clock Gating Register (CCM_CCGR12_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_40CC	CCM Clock Gating Register (CCM_CCGR12_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_40D0	CCM Clock Gating Register (CCM_CCGR13)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_40D4	CCM Clock Gating Register (CCM_CCGR13_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_40D8	CCM Clock Gating Register (CCM_CCGR13_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_40DC	CCM Clock Gating Register (CCM_CCGR13_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_40E0	CCM Clock Gating Register (CCM_CCGR14)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_40E4	CCM Clock Gating Register (CCM_CCGR14_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_40E8	CCM Clock Gating Register (CCM_CCGR14_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_40EC	CCM Clock Gating Register (CCM_CCGR14_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_40F0	CCM Clock Gating Register (CCM_CCGR15)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_40F4	CCM Clock Gating Register (CCM_CCGR15_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_40F8	CCM Clock Gating Register (CCM_CCGR15_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_40FC	CCM Clock Gating Register (CCM_CCGR15_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4100	CCM Clock Gating Register (CCM_CCGR16)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4104	CCM Clock Gating Register (CCM_CCGR16_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4108	CCM Clock Gating Register (CCM_CCGR16_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_410C	CCM Clock Gating Register (CCM_CCGR16_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4110	CCM Clock Gating Register (CCM_CCGR17)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4114	CCM Clock Gating Register (CCM_CCGR17_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4118	CCM Clock Gating Register (CCM_CCGR17_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_411C	CCM Clock Gating Register (CCM_CCGR17_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4120	CCM Clock Gating Register (CCM_CCGR18)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4124	CCM Clock Gating Register (CCM_CCGR18_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4128	CCM Clock Gating Register (CCM_CCGR18_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_412C	CCM Clock Gating Register (CCM_CCGR18_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4130	CCM Clock Gating Register (CCM_CCGR19)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4134	CCM Clock Gating Register (CCM_CCGR19_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4138	CCM Clock Gating Register (CCM_CCGR19_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_413C	CCM Clock Gating Register (CCM_CCGR19_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4140	CCM Clock Gating Register (CCM_CCGR20)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4144	CCM Clock Gating Register (CCM_CCGR20_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4148	CCM Clock Gating Register (CCM_CCGR20_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_414C	CCM Clock Gating Register (CCM_CCGR20_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4150	CCM Clock Gating Register (CCM_CCGR21)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4154	CCM Clock Gating Register (CCM_CCGR21_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4158	CCM Clock Gating Register (CCM_CCGR21_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_415C	CCM Clock Gating Register (CCM_CCGR21_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4160	CCM Clock Gating Register (CCM_CCGR22)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_4164	CCM Clock Gating Register (CCM_CCGR22_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4168	CCM Clock Gating Register (CCM_CCGR22_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_416C	CCM Clock Gating Register (CCM_CCGR22_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4170	CCM Clock Gating Register (CCM_CCGR23)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4174	CCM Clock Gating Register (CCM_CCGR23_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4178	CCM Clock Gating Register (CCM_CCGR23_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_417C	CCM Clock Gating Register (CCM_CCGR23_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4180	CCM Clock Gating Register (CCM_CCGR24)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4184	CCM Clock Gating Register (CCM_CCGR24_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4188	CCM Clock Gating Register (CCM_CCGR24_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_418C	CCM Clock Gating Register (CCM_CCGR24_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4190	CCM Clock Gating Register (CCM_CCGR25)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4194	CCM Clock Gating Register (CCM_CCGR25_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4198	CCM Clock Gating Register (CCM_CCGR25_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_419C	CCM Clock Gating Register (CCM_CCGR25_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_41A0	CCM Clock Gating Register (CCM_CCGR26)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_41A4	CCM Clock Gating Register (CCM_CCGR26_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_41A8	CCM Clock Gating Register (CCM_CCGR26_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_41AC	CCM Clock Gating Register (CCM_CCGR26_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_41B0	CCM Clock Gating Register (CCM_CCGR27)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_41B4	CCM Clock Gating Register (CCM_CCGR27_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_41B8	CCM Clock Gating Register (CCM_CCGR27_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_41BC	CCM Clock Gating Register (CCM_CCGR27_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_41C0	CCM Clock Gating Register (CCM_CCGR28)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_41C4	CCM Clock Gating Register (CCM_CCGR28_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_41C8	CCM Clock Gating Register (CCM_CCGR28_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_41CC	CCM Clock Gating Register (CCM_CCGR28_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_41D0	CCM Clock Gating Register (CCM_CCGR29)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_41D4	CCM Clock Gating Register (CCM_CCGR29_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_41D8	CCM Clock Gating Register (CCM_CCGR29_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_41DC	CCM Clock Gating Register (CCM_CCGR29_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_41E0	CCM Clock Gating Register (CCM_CCGR30)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_41E4	CCM Clock Gating Register (CCM_CCGR30_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_41E8	CCM Clock Gating Register (CCM_CCGR30_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_41EC	CCM Clock Gating Register (CCM_CCGR30_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_41F0	CCM Clock Gating Register (CCM_CCGR31)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_41F4	CCM Clock Gating Register (CCM_CCGR31_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_41F8	CCM Clock Gating Register (CCM_CCGR31_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_41FC	CCM Clock Gating Register (CCM_CCGR31_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4200	CCM Clock Gating Register (CCM_CCGR32)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4204	CCM Clock Gating Register (CCM_CCGR32_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4208	CCM Clock Gating Register (CCM_CCGR32_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_420C	CCM Clock Gating Register (CCM_CCGR32_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4210	CCM Clock Gating Register (CCM_CCGR33)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_4214	CCM Clock Gating Register (CCM_CCGR33_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4218	CCM Clock Gating Register (CCM_CCGR33_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_421C	CCM Clock Gating Register (CCM_CCGR33_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4220	CCM Clock Gating Register (CCM_CCGR34)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4224	CCM Clock Gating Register (CCM_CCGR34_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4228	CCM Clock Gating Register (CCM_CCGR34_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_422C	CCM Clock Gating Register (CCM_CCGR34_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4230	CCM Clock Gating Register (CCM_CCGR35)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4234	CCM Clock Gating Register (CCM_CCGR35_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4238	CCM Clock Gating Register (CCM_CCGR35_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_423C	CCM Clock Gating Register (CCM_CCGR35_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4240	CCM Clock Gating Register (CCM_CCGR36)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4244	CCM Clock Gating Register (CCM_CCGR36_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4248	CCM Clock Gating Register (CCM_CCGR36_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_424C	CCM Clock Gating Register (CCM_CCGR36_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4250	CCM Clock Gating Register (CCM_CCGR37)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4254	CCM Clock Gating Register (CCM_CCGR37_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4258	CCM Clock Gating Register (CCM_CCGR37_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_425C	CCM Clock Gating Register (CCM_CCGR37_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4260	CCM Clock Gating Register (CCM_CCGR38)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4264	CCM Clock Gating Register (CCM_CCGR38_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4268	CCM Clock Gating Register (CCM_CCGR38_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_426C	CCM Clock Gating Register (CCM_CCGR38_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4270	CCM Clock Gating Register (CCM_CCGR39)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4274	CCM Clock Gating Register (CCM_CCGR39_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4278	CCM Clock Gating Register (CCM_CCGR39_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_427C	CCM Clock Gating Register (CCM_CCGR39_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4280	CCM Clock Gating Register (CCM_CCGR40)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4284	CCM Clock Gating Register (CCM_CCGR40_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4288	CCM Clock Gating Register (CCM_CCGR40_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_428C	CCM Clock Gating Register (CCM_CCGR40_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4290	CCM Clock Gating Register (CCM_CCGR41)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4294	CCM Clock Gating Register (CCM_CCGR41_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4298	CCM Clock Gating Register (CCM_CCGR41_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_429C	CCM Clock Gating Register (CCM_CCGR41_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_42A0	CCM Clock Gating Register (CCM_CCGR42)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_42A4	CCM Clock Gating Register (CCM_CCGR42_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_42A8	CCM Clock Gating Register (CCM_CCGR42_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_42AC	CCM Clock Gating Register (CCM_CCGR42_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_42B0	CCM Clock Gating Register (CCM_CCGR43)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_42B4	CCM Clock Gating Register (CCM_CCGR43_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_42B8	CCM Clock Gating Register (CCM_CCGR43_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_42BC	CCM Clock Gating Register (CCM_CCGR43_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_42C0	CCM Clock Gating Register (CCM_CCGR44)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_42C4	CCM Clock Gating Register (CCM_CCGR44_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_42C8	CCM Clock Gating Register (CCM_CCGR44_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_42CC	CCM Clock Gating Register (CCM_CCGR44_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_42D0	CCM Clock Gating Register (CCM_CCGR45)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_42D4	CCM Clock Gating Register (CCM_CCGR45_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_42D8	CCM Clock Gating Register (CCM_CCGR45_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_42DC	CCM Clock Gating Register (CCM_CCGR45_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_42E0	CCM Clock Gating Register (CCM_CCGR46)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_42E4	CCM Clock Gating Register (CCM_CCGR46_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_42E8	CCM Clock Gating Register (CCM_CCGR46_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_42EC	CCM Clock Gating Register (CCM_CCGR46_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_42F0	CCM Clock Gating Register (CCM_CCGR47)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_42F4	CCM Clock Gating Register (CCM_CCGR47_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_42F8	CCM Clock Gating Register (CCM_CCGR47_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_42FC	CCM Clock Gating Register (CCM_CCGR47_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4300	CCM Clock Gating Register (CCM_CCGR48)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4304	CCM Clock Gating Register (CCM_CCGR48_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4308	CCM Clock Gating Register (CCM_CCGR48_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_430C	CCM Clock Gating Register (CCM_CCGR48_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4310	CCM Clock Gating Register (CCM_CCGR49)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4314	CCM Clock Gating Register (CCM_CCGR49_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4318	CCM Clock Gating Register (CCM_CCGR49_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_431C	CCM Clock Gating Register (CCM_CCGR49_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4320	CCM Clock Gating Register (CCM_CCGR50)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4324	CCM Clock Gating Register (CCM_CCGR50_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4328	CCM Clock Gating Register (CCM_CCGR50_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_432C	CCM Clock Gating Register (CCM_CCGR50_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4330	CCM Clock Gating Register (CCM_CCGR51)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4334	CCM Clock Gating Register (CCM_CCGR51_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4338	CCM Clock Gating Register (CCM_CCGR51_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_433C	CCM Clock Gating Register (CCM_CCGR51_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4340	CCM Clock Gating Register (CCM_CCGR52)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4344	CCM Clock Gating Register (CCM_CCGR52_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4348	CCM Clock Gating Register (CCM_CCGR52_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_434C	CCM Clock Gating Register (CCM_CCGR52_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4350	CCM Clock Gating Register (CCM_CCGR53)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4354	CCM Clock Gating Register (CCM_CCGR53_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4358	CCM Clock Gating Register (CCM_CCGR53_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_435C	CCM Clock Gating Register (CCM_CCGR53_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4360	CCM Clock Gating Register (CCM_CCGR54)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4364	CCM Clock Gating Register (CCM_CCGR54_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4368	CCM Clock Gating Register (CCM_CCGR54_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_436C	CCM Clock Gating Register (CCM_CCGR54_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4370	CCM Clock Gating Register (CCM_CCGR55)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_4374	CCM Clock Gating Register (CCM_CCGR55_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4378	CCM Clock Gating Register (CCM_CCGR55_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_437C	CCM Clock Gating Register (CCM_CCGR55_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4380	CCM Clock Gating Register (CCM_CCGR56)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4384	CCM Clock Gating Register (CCM_CCGR56_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4388	CCM Clock Gating Register (CCM_CCGR56_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_438C	CCM Clock Gating Register (CCM_CCGR56_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4390	CCM Clock Gating Register (CCM_CCGR57)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4394	CCM Clock Gating Register (CCM_CCGR57_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4398	CCM Clock Gating Register (CCM_CCGR57_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_439C	CCM Clock Gating Register (CCM_CCGR57_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_43A0	CCM Clock Gating Register (CCM_CCGR58)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_43A4	CCM Clock Gating Register (CCM_CCGR58_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_43A8	CCM Clock Gating Register (CCM_CCGR58_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_43AC	CCM Clock Gating Register (CCM_CCGR58_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_43B0	CCM Clock Gating Register (CCM_CCGR59)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_43B4	CCM Clock Gating Register (CCM_CCGR59_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_43B8	CCM Clock Gating Register (CCM_CCGR59_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_43BC	CCM Clock Gating Register (CCM_CCGR59_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_43C0	CCM Clock Gating Register (CCM_CCGR60)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_43C4	CCM Clock Gating Register (CCM_CCGR60_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_43C8	CCM Clock Gating Register (CCM_CCGR60_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_43CC	CCM Clock Gating Register (CCM_CCGR60_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_43D0	CCM Clock Gating Register (CCM_CCGR61)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_43D4	CCM Clock Gating Register (CCM_CCGR61_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_43D8	CCM Clock Gating Register (CCM_CCGR61_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_43DC	CCM Clock Gating Register (CCM_CCGR61_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_43E0	CCM Clock Gating Register (CCM_CCGR62)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_43E4	CCM Clock Gating Register (CCM_CCGR62_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_43E8	CCM Clock Gating Register (CCM_CCGR62_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_43EC	CCM Clock Gating Register (CCM_CCGR62_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_43F0	CCM Clock Gating Register (CCM_CCGR63)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_43F4	CCM Clock Gating Register (CCM_CCGR63_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_43F8	CCM Clock Gating Register (CCM_CCGR63_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_43FC	CCM Clock Gating Register (CCM_CCGR63_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4400	CCM Clock Gating Register (CCM_CCGR64)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4404	CCM Clock Gating Register (CCM_CCGR64_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4408	CCM Clock Gating Register (CCM_CCGR64_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_440C	CCM Clock Gating Register (CCM_CCGR64_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4410	CCM Clock Gating Register (CCM_CCGR65)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4414	CCM Clock Gating Register (CCM_CCGR65_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4418	CCM Clock Gating Register (CCM_CCGR65_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_441C	CCM Clock Gating Register (CCM_CCGR65_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4420	CCM Clock Gating Register (CCM_CCGR66)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_4424	CCM Clock Gating Register (CCM_CCGR66_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4428	CCM Clock Gating Register (CCM_CCGR66_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_442C	CCM Clock Gating Register (CCM_CCGR66_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4430	CCM Clock Gating Register (CCM_CCGR67)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4434	CCM Clock Gating Register (CCM_CCGR67_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4438	CCM Clock Gating Register (CCM_CCGR67_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_443C	CCM Clock Gating Register (CCM_CCGR67_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4440	CCM Clock Gating Register (CCM_CCGR68)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4444	CCM Clock Gating Register (CCM_CCGR68_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4448	CCM Clock Gating Register (CCM_CCGR68_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_444C	CCM Clock Gating Register (CCM_CCGR68_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4450	CCM Clock Gating Register (CCM_CCGR69)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4454	CCM Clock Gating Register (CCM_CCGR69_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4458	CCM Clock Gating Register (CCM_CCGR69_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_445C	CCM Clock Gating Register (CCM_CCGR69_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4460	CCM Clock Gating Register (CCM_CCGR70)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4464	CCM Clock Gating Register (CCM_CCGR70_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4468	CCM Clock Gating Register (CCM_CCGR70_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_446C	CCM Clock Gating Register (CCM_CCGR70_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4470	CCM Clock Gating Register (CCM_CCGR71)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4474	CCM Clock Gating Register (CCM_CCGR71_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4478	CCM Clock Gating Register (CCM_CCGR71_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_447C	CCM Clock Gating Register (CCM_CCGR71_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4480	CCM Clock Gating Register (CCM_CCGR72)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4484	CCM Clock Gating Register (CCM_CCGR72_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4488	CCM Clock Gating Register (CCM_CCGR72_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_448C	CCM Clock Gating Register (CCM_CCGR72_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4490	CCM Clock Gating Register (CCM_CCGR73)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4494	CCM Clock Gating Register (CCM_CCGR73_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4498	CCM Clock Gating Register (CCM_CCGR73_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_449C	CCM Clock Gating Register (CCM_CCGR73_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_44A0	CCM Clock Gating Register (CCM_CCGR74)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_44A4	CCM Clock Gating Register (CCM_CCGR74_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_44A8	CCM Clock Gating Register (CCM_CCGR74_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_44AC	CCM Clock Gating Register (CCM_CCGR74_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_44B0	CCM Clock Gating Register (CCM_CCGR75)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_44B4	CCM Clock Gating Register (CCM_CCGR75_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_44B8	CCM Clock Gating Register (CCM_CCGR75_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_44BC	CCM Clock Gating Register (CCM_CCGR75_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_44C0	CCM Clock Gating Register (CCM_CCGR76)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_44C4	CCM Clock Gating Register (CCM_CCGR76_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_44C8	CCM Clock Gating Register (CCM_CCGR76_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_44CC	CCM Clock Gating Register (CCM_CCGR76_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_44D0	CCM Clock Gating Register (CCM_CCGR77)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_44D4	CCM Clock Gating Register (CCM_CCGR77_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_44D8	CCM Clock Gating Register (CCM_CCGR77_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_44DC	CCM Clock Gating Register (CCM_CCGR77_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_44E0	CCM Clock Gating Register (CCM_CCGR78)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_44E4	CCM Clock Gating Register (CCM_CCGR78_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_44E8	CCM Clock Gating Register (CCM_CCGR78_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_44EC	CCM Clock Gating Register (CCM_CCGR78_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_44F0	CCM Clock Gating Register (CCM_CCGR79)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_44F4	CCM Clock Gating Register (CCM_CCGR79_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_44F8	CCM Clock Gating Register (CCM_CCGR79_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_44FC	CCM Clock Gating Register (CCM_CCGR79_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4500	CCM Clock Gating Register (CCM_CCGR80)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4504	CCM Clock Gating Register (CCM_CCGR80_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4508	CCM Clock Gating Register (CCM_CCGR80_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_450C	CCM Clock Gating Register (CCM_CCGR80_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4510	CCM Clock Gating Register (CCM_CCGR81)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4514	CCM Clock Gating Register (CCM_CCGR81_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4518	CCM Clock Gating Register (CCM_CCGR81_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_451C	CCM Clock Gating Register (CCM_CCGR81_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4520	CCM Clock Gating Register (CCM_CCGR82)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4524	CCM Clock Gating Register (CCM_CCGR82_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4528	CCM Clock Gating Register (CCM_CCGR82_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_452C	CCM Clock Gating Register (CCM_CCGR82_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4530	CCM Clock Gating Register (CCM_CCGR83)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4534	CCM Clock Gating Register (CCM_CCGR83_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4538	CCM Clock Gating Register (CCM_CCGR83_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_453C	CCM Clock Gating Register (CCM_CCGR83_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4540	CCM Clock Gating Register (CCM_CCGR84)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4544	CCM Clock Gating Register (CCM_CCGR84_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4548	CCM Clock Gating Register (CCM_CCGR84_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_454C	CCM Clock Gating Register (CCM_CCGR84_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4550	CCM Clock Gating Register (CCM_CCGR85)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4554	CCM Clock Gating Register (CCM_CCGR85_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4558	CCM Clock Gating Register (CCM_CCGR85_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_455C	CCM Clock Gating Register (CCM_CCGR85_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4560	CCM Clock Gating Register (CCM_CCGR86)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4564	CCM Clock Gating Register (CCM_CCGR86_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4568	CCM Clock Gating Register (CCM_CCGR86_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_456C	CCM Clock Gating Register (CCM_CCGR86_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4570	CCM Clock Gating Register (CCM_CCGR87)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4574	CCM Clock Gating Register (CCM_CCGR87_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4578	CCM Clock Gating Register (CCM_CCGR87_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_457C	CCM Clock Gating Register (CCM_CCGR87_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4580	CCM Clock Gating Register (CCM_CCGR88)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_4584	CCM Clock Gating Register (CCM_CCGR88_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4588	CCM Clock Gating Register (CCM_CCGR88_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_458C	CCM Clock Gating Register (CCM_CCGR88_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4590	CCM Clock Gating Register (CCM_CCGR89)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4594	CCM Clock Gating Register (CCM_CCGR89_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4598	CCM Clock Gating Register (CCM_CCGR89_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_459C	CCM Clock Gating Register (CCM_CCGR89_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_45A0	CCM Clock Gating Register (CCM_CCGR90)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_45A4	CCM Clock Gating Register (CCM_CCGR90_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_45A8	CCM Clock Gating Register (CCM_CCGR90_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_45AC	CCM Clock Gating Register (CCM_CCGR90_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_45B0	CCM Clock Gating Register (CCM_CCGR91)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_45B4	CCM Clock Gating Register (CCM_CCGR91_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_45B8	CCM Clock Gating Register (CCM_CCGR91_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_45BC	CCM Clock Gating Register (CCM_CCGR91_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_45C0	CCM Clock Gating Register (CCM_CCGR92)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_45C4	CCM Clock Gating Register (CCM_CCGR92_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_45C8	CCM Clock Gating Register (CCM_CCGR92_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_45CC	CCM Clock Gating Register (CCM_CCGR92_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_45D0	CCM Clock Gating Register (CCM_CCGR93)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_45D4	CCM Clock Gating Register (CCM_CCGR93_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_45D8	CCM Clock Gating Register (CCM_CCGR93_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_45DC	CCM Clock Gating Register (CCM_CCGR93_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_45E0	CCM Clock Gating Register (CCM_CCGR94)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_45E4	CCM Clock Gating Register (CCM_CCGR94_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_45E8	CCM Clock Gating Register (CCM_CCGR94_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_45EC	CCM Clock Gating Register (CCM_CCGR94_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_45F0	CCM Clock Gating Register (CCM_CCGR95)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_45F4	CCM Clock Gating Register (CCM_CCGR95_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_45F8	CCM Clock Gating Register (CCM_CCGR95_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_45FC	CCM Clock Gating Register (CCM_CCGR95_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4600	CCM Clock Gating Register (CCM_CCGR96)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4604	CCM Clock Gating Register (CCM_CCGR96_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4608	CCM Clock Gating Register (CCM_CCGR96_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_460C	CCM Clock Gating Register (CCM_CCGR96_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4610	CCM Clock Gating Register (CCM_CCGR97)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4614	CCM Clock Gating Register (CCM_CCGR97_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4618	CCM Clock Gating Register (CCM_CCGR97_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_461C	CCM Clock Gating Register (CCM_CCGR97_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4620	CCM Clock Gating Register (CCM_CCGR98)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4624	CCM Clock Gating Register (CCM_CCGR98_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4628	CCM Clock Gating Register (CCM_CCGR98_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_462C	CCM Clock Gating Register (CCM_CCGR98_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4630	CCM Clock Gating Register (CCM_CCGR99)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_4634	CCM Clock Gating Register (CCM_CCGR99_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4638	CCM Clock Gating Register (CCM_CCGR99_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_463C	CCM Clock Gating Register (CCM_CCGR99_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4640	CCM Clock Gating Register (CCM_CCGR100)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4644	CCM Clock Gating Register (CCM_CCGR100_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4648	CCM Clock Gating Register (CCM_CCGR100_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_464C	CCM Clock Gating Register (CCM_CCGR100_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4650	CCM Clock Gating Register (CCM_CCGR101)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4654	CCM Clock Gating Register (CCM_CCGR101_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4658	CCM Clock Gating Register (CCM_CCGR101_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_465C	CCM Clock Gating Register (CCM_CCGR101_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4660	CCM Clock Gating Register (CCM_CCGR102)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4664	CCM Clock Gating Register (CCM_CCGR102_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4668	CCM Clock Gating Register (CCM_CCGR102_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_466C	CCM Clock Gating Register (CCM_CCGR102_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4670	CCM Clock Gating Register (CCM_CCGR103)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4674	CCM Clock Gating Register (CCM_CCGR103_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4678	CCM Clock Gating Register (CCM_CCGR103_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_467C	CCM Clock Gating Register (CCM_CCGR103_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4680	CCM Clock Gating Register (CCM_CCGR104)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4684	CCM Clock Gating Register (CCM_CCGR104_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4688	CCM Clock Gating Register (CCM_CCGR104_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_468C	CCM Clock Gating Register (CCM_CCGR104_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4690	CCM Clock Gating Register (CCM_CCGR105)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4694	CCM Clock Gating Register (CCM_CCGR105_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4698	CCM Clock Gating Register (CCM_CCGR105_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_469C	CCM Clock Gating Register (CCM_CCGR105_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_46A0	CCM Clock Gating Register (CCM_CCGR106)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_46A4	CCM Clock Gating Register (CCM_CCGR106_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_46A8	CCM Clock Gating Register (CCM_CCGR106_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_46AC	CCM Clock Gating Register (CCM_CCGR106_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_46B0	CCM Clock Gating Register (CCM_CCGR107)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_46B4	CCM Clock Gating Register (CCM_CCGR107_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_46B8	CCM Clock Gating Register (CCM_CCGR107_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_46BC	CCM Clock Gating Register (CCM_CCGR107_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_46C0	CCM Clock Gating Register (CCM_CCGR108)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_46C4	CCM Clock Gating Register (CCM_CCGR108_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_46C8	CCM Clock Gating Register (CCM_CCGR108_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_46CC	CCM Clock Gating Register (CCM_CCGR108_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_46D0	CCM Clock Gating Register (CCM_CCGR109)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_46D4	CCM Clock Gating Register (CCM_CCGR109_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_46D8	CCM Clock Gating Register (CCM_CCGR109_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_46DC	CCM Clock Gating Register (CCM_CCGR109_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_46E0	CCM Clock Gating Register (CCM_CCGR110)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_46E4	CCM Clock Gating Register (CCM_CCGR110_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_46E8	CCM Clock Gating Register (CCM_CCGR110_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_46EC	CCM Clock Gating Register (CCM_CCGR110_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_46F0	CCM Clock Gating Register (CCM_CCGR111)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_46F4	CCM Clock Gating Register (CCM_CCGR111_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_46F8	CCM Clock Gating Register (CCM_CCGR111_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_46FC	CCM Clock Gating Register (CCM_CCGR111_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4700	CCM Clock Gating Register (CCM_CCGR112)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4704	CCM Clock Gating Register (CCM_CCGR112_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4708	CCM Clock Gating Register (CCM_CCGR112_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_470C	CCM Clock Gating Register (CCM_CCGR112_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4710	CCM Clock Gating Register (CCM_CCGR113)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4714	CCM Clock Gating Register (CCM_CCGR113_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4718	CCM Clock Gating Register (CCM_CCGR113_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_471C	CCM Clock Gating Register (CCM_CCGR113_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4720	CCM Clock Gating Register (CCM_CCGR114)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4724	CCM Clock Gating Register (CCM_CCGR114_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4728	CCM Clock Gating Register (CCM_CCGR114_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_472C	CCM Clock Gating Register (CCM_CCGR114_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4730	CCM Clock Gating Register (CCM_CCGR115)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4734	CCM Clock Gating Register (CCM_CCGR115_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4738	CCM Clock Gating Register (CCM_CCGR115_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_473C	CCM Clock Gating Register (CCM_CCGR115_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4740	CCM Clock Gating Register (CCM_CCGR116)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4744	CCM Clock Gating Register (CCM_CCGR116_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4748	CCM Clock Gating Register (CCM_CCGR116_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_474C	CCM Clock Gating Register (CCM_CCGR116_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4750	CCM Clock Gating Register (CCM_CCGR117)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4754	CCM Clock Gating Register (CCM_CCGR117_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4758	CCM Clock Gating Register (CCM_CCGR117_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_475C	CCM Clock Gating Register (CCM_CCGR117_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4760	CCM Clock Gating Register (CCM_CCGR118)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4764	CCM Clock Gating Register (CCM_CCGR118_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4768	CCM Clock Gating Register (CCM_CCGR118_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_476C	CCM Clock Gating Register (CCM_CCGR118_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4770	CCM Clock Gating Register (CCM_CCGR119)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4774	CCM Clock Gating Register (CCM_CCGR119_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4778	CCM Clock Gating Register (CCM_CCGR119_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_477C	CCM Clock Gating Register (CCM_CCGR119_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4780	CCM Clock Gating Register (CCM_CCGR120)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4784	CCM Clock Gating Register (CCM_CCGR120_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4788	CCM Clock Gating Register (CCM_CCGR120_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_478C	CCM Clock Gating Register (CCM_CCGR120_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4790	CCM Clock Gating Register (CCM_CCGR121)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_4794	CCM Clock Gating Register (CCM_CCGR121_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4798	CCM Clock Gating Register (CCM_CCGR121_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_479C	CCM Clock Gating Register (CCM_CCGR121_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_47A0	CCM Clock Gating Register (CCM_CCGR122)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_47A4	CCM Clock Gating Register (CCM_CCGR122_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_47A8	CCM Clock Gating Register (CCM_CCGR122_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_47AC	CCM Clock Gating Register (CCM_CCGR122_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_47B0	CCM Clock Gating Register (CCM_CCGR123)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_47B4	CCM Clock Gating Register (CCM_CCGR123_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_47B8	CCM Clock Gating Register (CCM_CCGR123_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_47BC	CCM Clock Gating Register (CCM_CCGR123_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_47C0	CCM Clock Gating Register (CCM_CCGR124)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_47C4	CCM Clock Gating Register (CCM_CCGR124_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_47C8	CCM Clock Gating Register (CCM_CCGR124_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_47CC	CCM Clock Gating Register (CCM_CCGR124_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_47D0	CCM Clock Gating Register (CCM_CCGR125)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_47D4	CCM Clock Gating Register (CCM_CCGR125_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_47D8	CCM Clock Gating Register (CCM_CCGR125_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_47DC	CCM Clock Gating Register (CCM_CCGR125_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_47E0	CCM Clock Gating Register (CCM_CCGR126)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_47E4	CCM Clock Gating Register (CCM_CCGR126_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_47E8	CCM Clock Gating Register (CCM_CCGR126_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_47EC	CCM Clock Gating Register (CCM_CCGR126_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_47F0	CCM Clock Gating Register (CCM_CCGR127)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_47F4	CCM Clock Gating Register (CCM_CCGR127_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_47F8	CCM Clock Gating Register (CCM_CCGR127_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_47FC	CCM Clock Gating Register (CCM_CCGR127_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4800	CCM Clock Gating Register (CCM_CCGR128)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4804	CCM Clock Gating Register (CCM_CCGR128_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4808	CCM Clock Gating Register (CCM_CCGR128_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_480C	CCM Clock Gating Register (CCM_CCGR128_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4810	CCM Clock Gating Register (CCM_CCGR129)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4814	CCM Clock Gating Register (CCM_CCGR129_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4818	CCM Clock Gating Register (CCM_CCGR129_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_481C	CCM Clock Gating Register (CCM_CCGR129_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4820	CCM Clock Gating Register (CCM_CCGR130)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4824	CCM Clock Gating Register (CCM_CCGR130_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4828	CCM Clock Gating Register (CCM_CCGR130_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_482C	CCM Clock Gating Register (CCM_CCGR130_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4830	CCM Clock Gating Register (CCM_CCGR131)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4834	CCM Clock Gating Register (CCM_CCGR131_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4838	CCM Clock Gating Register (CCM_CCGR131_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_483C	CCM Clock Gating Register (CCM_CCGR131_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4840	CCM Clock Gating Register (CCM_CCGR132)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_4844	CCM Clock Gating Register (CCM_CCGR132_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4848	CCM Clock Gating Register (CCM_CCGR132_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_484C	CCM Clock Gating Register (CCM_CCGR132_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4850	CCM Clock Gating Register (CCM_CCGR133)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4854	CCM Clock Gating Register (CCM_CCGR133_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4858	CCM Clock Gating Register (CCM_CCGR133_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_485C	CCM Clock Gating Register (CCM_CCGR133_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4860	CCM Clock Gating Register (CCM_CCGR134)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4864	CCM Clock Gating Register (CCM_CCGR134_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4868	CCM Clock Gating Register (CCM_CCGR134_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_486C	CCM Clock Gating Register (CCM_CCGR134_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4870	CCM Clock Gating Register (CCM_CCGR135)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4874	CCM Clock Gating Register (CCM_CCGR135_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4878	CCM Clock Gating Register (CCM_CCGR135_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_487C	CCM Clock Gating Register (CCM_CCGR135_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4880	CCM Clock Gating Register (CCM_CCGR136)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4884	CCM Clock Gating Register (CCM_CCGR136_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4888	CCM Clock Gating Register (CCM_CCGR136_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_488C	CCM Clock Gating Register (CCM_CCGR136_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4890	CCM Clock Gating Register (CCM_CCGR137)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4894	CCM Clock Gating Register (CCM_CCGR137_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4898	CCM Clock Gating Register (CCM_CCGR137_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_489C	CCM Clock Gating Register (CCM_CCGR137_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_48A0	CCM Clock Gating Register (CCM_CCGR138)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_48A4	CCM Clock Gating Register (CCM_CCGR138_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_48A8	CCM Clock Gating Register (CCM_CCGR138_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_48AC	CCM Clock Gating Register (CCM_CCGR138_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_48B0	CCM Clock Gating Register (CCM_CCGR139)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_48B4	CCM Clock Gating Register (CCM_CCGR139_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_48B8	CCM Clock Gating Register (CCM_CCGR139_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_48BC	CCM Clock Gating Register (CCM_CCGR139_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_48C0	CCM Clock Gating Register (CCM_CCGR140)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_48C4	CCM Clock Gating Register (CCM_CCGR140_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_48C8	CCM Clock Gating Register (CCM_CCGR140_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_48CC	CCM Clock Gating Register (CCM_CCGR140_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_48D0	CCM Clock Gating Register (CCM_CCGR141)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_48D4	CCM Clock Gating Register (CCM_CCGR141_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_48D8	CCM Clock Gating Register (CCM_CCGR141_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_48DC	CCM Clock Gating Register (CCM_CCGR141_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_48E0	CCM Clock Gating Register (CCM_CCGR142)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_48E4	CCM Clock Gating Register (CCM_CCGR142_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_48E8	CCM Clock Gating Register (CCM_CCGR142_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_48EC	CCM Clock Gating Register (CCM_CCGR142_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_48F0	CCM Clock Gating Register (CCM_CCGR143)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_48F4	CCM Clock Gating Register (CCM_CCGR143_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_48F8	CCM Clock Gating Register (CCM_CCGR143_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_48FC	CCM Clock Gating Register (CCM_CCGR143_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4900	CCM Clock Gating Register (CCM_CCGR144)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4904	CCM Clock Gating Register (CCM_CCGR144_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4908	CCM Clock Gating Register (CCM_CCGR144_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_490C	CCM Clock Gating Register (CCM_CCGR144_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4910	CCM Clock Gating Register (CCM_CCGR145)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4914	CCM Clock Gating Register (CCM_CCGR145_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4918	CCM Clock Gating Register (CCM_CCGR145_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_491C	CCM Clock Gating Register (CCM_CCGR145_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4920	CCM Clock Gating Register (CCM_CCGR146)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4924	CCM Clock Gating Register (CCM_CCGR146_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4928	CCM Clock Gating Register (CCM_CCGR146_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_492C	CCM Clock Gating Register (CCM_CCGR146_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4930	CCM Clock Gating Register (CCM_CCGR147)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4934	CCM Clock Gating Register (CCM_CCGR147_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4938	CCM Clock Gating Register (CCM_CCGR147_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_493C	CCM Clock Gating Register (CCM_CCGR147_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4940	CCM Clock Gating Register (CCM_CCGR148)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4944	CCM Clock Gating Register (CCM_CCGR148_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4948	CCM Clock Gating Register (CCM_CCGR148_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_494C	CCM Clock Gating Register (CCM_CCGR148_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4950	CCM Clock Gating Register (CCM_CCGR149)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4954	CCM Clock Gating Register (CCM_CCGR149_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4958	CCM Clock Gating Register (CCM_CCGR149_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_495C	CCM Clock Gating Register (CCM_CCGR149_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4960	CCM Clock Gating Register (CCM_CCGR150)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4964	CCM Clock Gating Register (CCM_CCGR150_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4968	CCM Clock Gating Register (CCM_CCGR150_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_496C	CCM Clock Gating Register (CCM_CCGR150_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4970	CCM Clock Gating Register (CCM_CCGR151)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4974	CCM Clock Gating Register (CCM_CCGR151_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4978	CCM Clock Gating Register (CCM_CCGR151_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_497C	CCM Clock Gating Register (CCM_CCGR151_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4980	CCM Clock Gating Register (CCM_CCGR152)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4984	CCM Clock Gating Register (CCM_CCGR152_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4988	CCM Clock Gating Register (CCM_CCGR152_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_498C	CCM Clock Gating Register (CCM_CCGR152_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4990	CCM Clock Gating Register (CCM_CCGR153)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4994	CCM Clock Gating Register (CCM_CCGR153_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4998	CCM Clock Gating Register (CCM_CCGR153_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_499C	CCM Clock Gating Register (CCM_CCGR153_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_49A0	CCM Clock Gating Register (CCM_CCGR154)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_49A4	CCM Clock Gating Register (CCM_CCGR154_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_49A8	CCM Clock Gating Register (CCM_CCGR154_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_49AC	CCM Clock Gating Register (CCM_CCGR154_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_49B0	CCM Clock Gating Register (CCM_CCGR155)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_49B4	CCM Clock Gating Register (CCM_CCGR155_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_49B8	CCM Clock Gating Register (CCM_CCGR155_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_49BC	CCM Clock Gating Register (CCM_CCGR155_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_49C0	CCM Clock Gating Register (CCM_CCGR156)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_49C4	CCM Clock Gating Register (CCM_CCGR156_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_49C8	CCM Clock Gating Register (CCM_CCGR156_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_49CC	CCM Clock Gating Register (CCM_CCGR156_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_49D0	CCM Clock Gating Register (CCM_CCGR157)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_49D4	CCM Clock Gating Register (CCM_CCGR157_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_49D8	CCM Clock Gating Register (CCM_CCGR157_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_49DC	CCM Clock Gating Register (CCM_CCGR157_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_49E0	CCM Clock Gating Register (CCM_CCGR158)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_49E4	CCM Clock Gating Register (CCM_CCGR158_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_49E8	CCM Clock Gating Register (CCM_CCGR158_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_49EC	CCM Clock Gating Register (CCM_CCGR158_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_49F0	CCM Clock Gating Register (CCM_CCGR159)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_49F4	CCM Clock Gating Register (CCM_CCGR159_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_49F8	CCM Clock Gating Register (CCM_CCGR159_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_49FC	CCM Clock Gating Register (CCM_CCGR159_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4A00	CCM Clock Gating Register (CCM_CCGR160)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4A04	CCM Clock Gating Register (CCM_CCGR160_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4A08	CCM Clock Gating Register (CCM_CCGR160_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4A0C	CCM Clock Gating Register (CCM_CCGR160_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4A10	CCM Clock Gating Register (CCM_CCGR161)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4A14	CCM Clock Gating Register (CCM_CCGR161_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4A18	CCM Clock Gating Register (CCM_CCGR161_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4A1C	CCM Clock Gating Register (CCM_CCGR161_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4A20	CCM Clock Gating Register (CCM_CCGR162)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4A24	CCM Clock Gating Register (CCM_CCGR162_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4A28	CCM Clock Gating Register (CCM_CCGR162_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4A2C	CCM Clock Gating Register (CCM_CCGR162_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4A30	CCM Clock Gating Register (CCM_CCGR163)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4A34	CCM Clock Gating Register (CCM_CCGR163_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4A38	CCM Clock Gating Register (CCM_CCGR163_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4A3C	CCM Clock Gating Register (CCM_CCGR163_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4A40	CCM Clock Gating Register (CCM_CCGR164)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4A44	CCM Clock Gating Register (CCM_CCGR164_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4A48	CCM Clock Gating Register (CCM_CCGR164_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4A4C	CCM Clock Gating Register (CCM_CCGR164_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4A50	CCM Clock Gating Register (CCM_CCGR165)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_4A54	CCM Clock Gating Register (CCM_CCGR165_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4A58	CCM Clock Gating Register (CCM_CCGR165_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4A5C	CCM Clock Gating Register (CCM_CCGR165_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4A60	CCM Clock Gating Register (CCM_CCGR166)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4A64	CCM Clock Gating Register (CCM_CCGR166_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4A68	CCM Clock Gating Register (CCM_CCGR166_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4A6C	CCM Clock Gating Register (CCM_CCGR166_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4A70	CCM Clock Gating Register (CCM_CCGR167)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4A74	CCM Clock Gating Register (CCM_CCGR167_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4A78	CCM Clock Gating Register (CCM_CCGR167_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4A7C	CCM Clock Gating Register (CCM_CCGR167_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4A80	CCM Clock Gating Register (CCM_CCGR168)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4A84	CCM Clock Gating Register (CCM_CCGR168_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4A88	CCM Clock Gating Register (CCM_CCGR168_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4A8C	CCM Clock Gating Register (CCM_CCGR168_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4A90	CCM Clock Gating Register (CCM_CCGR169)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4A94	CCM Clock Gating Register (CCM_CCGR169_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4A98	CCM Clock Gating Register (CCM_CCGR169_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4A9C	CCM Clock Gating Register (CCM_CCGR169_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4AA0	CCM Clock Gating Register (CCM_CCGR170)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4AA4	CCM Clock Gating Register (CCM_CCGR170_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4AA8	CCM Clock Gating Register (CCM_CCGR170_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_4AAC	CCM Clock Gating Register (CCM_CCGR170_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4AB0	CCM Clock Gating Register (CCM_CCGR171)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4AB4	CCM Clock Gating Register (CCM_CCGR171_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4AB8	CCM Clock Gating Register (CCM_CCGR171_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4ABC	CCM Clock Gating Register (CCM_CCGR171_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4AC0	CCM Clock Gating Register (CCM_CCGR172)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4AC4	CCM Clock Gating Register (CCM_CCGR172_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4AC8	CCM Clock Gating Register (CCM_CCGR172_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4ACC	CCM Clock Gating Register (CCM_CCGR172_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4AD0	CCM Clock Gating Register (CCM_CCGR173)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4AD4	CCM Clock Gating Register (CCM_CCGR173_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4AD8	CCM Clock Gating Register (CCM_CCGR173_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4ADC	CCM Clock Gating Register (CCM_CCGR173_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4AE0	CCM Clock Gating Register (CCM_CCGR174)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4AE4	CCM Clock Gating Register (CCM_CCGR174_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4AE8	CCM Clock Gating Register (CCM_CCGR174_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4AEC	CCM Clock Gating Register (CCM_CCGR174_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4AF0	CCM Clock Gating Register (CCM_CCGR175)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4AF4	CCM Clock Gating Register (CCM_CCGR175_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4AF8	CCM Clock Gating Register (CCM_CCGR175_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4AFC	CCM Clock Gating Register (CCM_CCGR175_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4B00	CCM Clock Gating Register (CCM_CCGR176)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_4B04	CCM Clock Gating Register (CCM_CCGR176_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4B08	CCM Clock Gating Register (CCM_CCGR176_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4B0C	CCM Clock Gating Register (CCM_CCGR176_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4B10	CCM Clock Gating Register (CCM_CCGR177)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4B14	CCM Clock Gating Register (CCM_CCGR177_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4B18	CCM Clock Gating Register (CCM_CCGR177_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4B1C	CCM Clock Gating Register (CCM_CCGR177_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4B20	CCM Clock Gating Register (CCM_CCGR178)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4B24	CCM Clock Gating Register (CCM_CCGR178_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4B28	CCM Clock Gating Register (CCM_CCGR178_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4B2C	CCM Clock Gating Register (CCM_CCGR178_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4B30	CCM Clock Gating Register (CCM_CCGR179)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4B34	CCM Clock Gating Register (CCM_CCGR179_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4B38	CCM Clock Gating Register (CCM_CCGR179_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4B3C	CCM Clock Gating Register (CCM_CCGR179_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4B40	CCM Clock Gating Register (CCM_CCGR180)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4B44	CCM Clock Gating Register (CCM_CCGR180_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4B48	CCM Clock Gating Register (CCM_CCGR180_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4B4C	CCM Clock Gating Register (CCM_CCGR180_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4B50	CCM Clock Gating Register (CCM_CCGR181)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4B54	CCM Clock Gating Register (CCM_CCGR181_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4B58	CCM Clock Gating Register (CCM_CCGR181_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_4B5C	CCM Clock Gating Register (CCM_CCGR181_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4B60	CCM Clock Gating Register (CCM_CCGR182)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4B64	CCM Clock Gating Register (CCM_CCGR182_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4B68	CCM Clock Gating Register (CCM_CCGR182_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4B6C	CCM Clock Gating Register (CCM_CCGR182_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4B70	CCM Clock Gating Register (CCM_CCGR183)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4B74	CCM Clock Gating Register (CCM_CCGR183_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4B78	CCM Clock Gating Register (CCM_CCGR183_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4B7C	CCM Clock Gating Register (CCM_CCGR183_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4B80	CCM Clock Gating Register (CCM_CCGR184)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4B84	CCM Clock Gating Register (CCM_CCGR184_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4B88	CCM Clock Gating Register (CCM_CCGR184_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4B8C	CCM Clock Gating Register (CCM_CCGR184_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4B90	CCM Clock Gating Register (CCM_CCGR185)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4B94	CCM Clock Gating Register (CCM_CCGR185_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4B98	CCM Clock Gating Register (CCM_CCGR185_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4B9C	CCM Clock Gating Register (CCM_CCGR185_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4BA0	CCM Clock Gating Register (CCM_CCGR186)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4BA4	CCM Clock Gating Register (CCM_CCGR186_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4BA8	CCM Clock Gating Register (CCM_CCGR186_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4BAC	CCM Clock Gating Register (CCM_CCGR186_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4BB0	CCM Clock Gating Register (CCM_CCGR187)	32	R/W	0000_0002h	5.2.8.6/ 691

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_4BB4	CCM Clock Gating Register (CCM_CCGR187_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4BB8	CCM Clock Gating Register (CCM_CCGR187_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4BBC	CCM Clock Gating Register (CCM_CCGR187_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4BC0	CCM Clock Gating Register (CCM_CCGR188)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4BC4	CCM Clock Gating Register (CCM_CCGR188_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4BC8	CCM Clock Gating Register (CCM_CCGR188_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4BCC	CCM Clock Gating Register (CCM_CCGR188_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4BD0	CCM Clock Gating Register (CCM_CCGR189)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4BD4	CCM Clock Gating Register (CCM_CCGR189_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4BD8	CCM Clock Gating Register (CCM_CCGR189_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4BDC	CCM Clock Gating Register (CCM_CCGR189_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_4BE0	CCM Clock Gating Register (CCM_CCGR190)	32	R/W	0000_0002h	5.2.8.6/ 691
3038_4BE4	CCM Clock Gating Register (CCM_CCGR190_SET)	32	R/W	0000_0002h	5.2.8.7/ 693
3038_4BE8	CCM Clock Gating Register (CCM_CCGR190_CLR)	32	R/W	0000_0002h	5.2.8.8/ 695
3038_4BEC	CCM Clock Gating Register (CCM_CCGR190_TOG)	32	R/W	0000_0002h	5.2.8.9/ 697
3038_8000	Target Register (CCM_TARGET_ROOT0)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8004	Target Register (CCM_TARGET_ROOT0_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8008	Target Register (CCM_TARGET_ROOT0_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_800C	Target Register (CCM_TARGET_ROOT0_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8010	Miscellaneous Register (CCM_MISC0)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8014	Miscellaneous Register (CCM_MISC_ROOT0_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8018	Miscellaneous Register (CCM_MISC_ROOT0_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_801C	Miscellaneous Register (CCM_MISC_ROOT0_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8020	Post Divider Register (CCM_POST0)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8024	Post Divider Register (CCM_POST_ROOT0_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8028	Post Divider Register (CCM_POST_ROOT0_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_802C	Post Divider Register (CCM_POST_ROOT0_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8030	Pre Divider Register (CCM_PRE0)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8034	Pre Divider Register (CCM_PRE_ROOT0_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8038	Pre Divider Register (CCM_PRE_ROOT0_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_803C	Pre Divider Register (CCM_PRE_ROOT0_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8070	Access Control Register (CCM_ACCESS_CTRL0)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8074	Access Control Register (CCM_ACCESS_CTRL_ROOT0_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8078	Access Control Register (CCM_ACCESS_CTRL_ROOT0_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_807C	Access Control Register (CCM_ACCESS_CTRL_ROOT0_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8080	Target Register (CCM_TARGET_ROOT1)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8084	Target Register (CCM_TARGET_ROOT1_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8088	Target Register (CCM_TARGET_ROOT1_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_808C	Target Register (CCM_TARGET_ROOT1_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8090	Miscellaneous Register (CCM_MISC1)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8094	Miscellaneous Register (CCM_MISC_ROOT1_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8098	Miscellaneous Register (CCM_MISC_ROOT1_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_809C	Miscellaneous Register (CCM_MISC_ROOT1_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_80A0	Post Divider Register (CCM_POST1)	32	R/W	0000_0000h	5.2.8.18/ 711

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_80A4	Post Divider Register (CCM_POST_ROOT1_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_80A8	Post Divider Register (CCM_POST_ROOT1_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_80AC	Post Divider Register (CCM_POST_ROOT1_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_80B0	Pre Divider Register (CCM_PRE1)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_80B4	Pre Divider Register (CCM_PRE_ROOT1_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_80B8	Pre Divider Register (CCM_PRE_ROOT1_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_80BC	Pre Divider Register (CCM_PRE_ROOT1_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_80F0	Access Control Register (CCM_ACCESS_CTRL1)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_80F4	Access Control Register (CCM_ACCESS_CTRL_ROOT1_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_80F8	Access Control Register (CCM_ACCESS_CTRL_ROOT1_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_80FC	Access Control Register (CCM_ACCESS_CTRL_ROOT1_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8100	Target Register (CCM_TARGET_ROOT2)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8104	Target Register (CCM_TARGET_ROOT2_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8108	Target Register (CCM_TARGET_ROOT2_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_810C	Target Register (CCM_TARGET_ROOT2_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8110	Miscellaneous Register (CCM_MISC2)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8114	Miscellaneous Register (CCM_MISC_ROOT2_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8118	Miscellaneous Register (CCM_MISC_ROOT2_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_811C	Miscellaneous Register (CCM_MISC_ROOT2_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8120	Post Divider Register (CCM_POST2)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8124	Post Divider Register (CCM_POST_ROOT2_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8128	Post Divider Register (CCM_POST_ROOT2_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_812C	Post Divider Register (CCM_POST_ROOT2_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8130	Pre Divider Register (CCM_PRE2)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8134	Pre Divider Register (CCM_PRE_ROOT2_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8138	Pre Divider Register (CCM_PRE_ROOT2_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_813C	Pre Divider Register (CCM_PRE_ROOT2_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8170	Access Control Register (CCM_ACCESS_CTRL2)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8174	Access Control Register (CCM_ACCESS_CTRL_ROOT2_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8178	Access Control Register (CCM_ACCESS_CTRL_ROOT2_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_817C	Access Control Register (CCM_ACCESS_CTRL_ROOT2_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8180	Target Register (CCM_TARGET_ROOT3)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8184	Target Register (CCM_TARGET_ROOT3_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8188	Target Register (CCM_TARGET_ROOT3_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_818C	Target Register (CCM_TARGET_ROOT3_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8190	Miscellaneous Register (CCM_MISC3)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8194	Miscellaneous Register (CCM_MISC_ROOT3_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8198	Miscellaneous Register (CCM_MISC_ROOT3_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_819C	Miscellaneous Register (CCM_MISC_ROOT3_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_81A0	Post Divider Register (CCM_POST3)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_81A4	Post Divider Register (CCM_POST_ROOT3_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_81A8	Post Divider Register (CCM_POST_ROOT3_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_81AC	Post Divider Register (CCM_POST_ROOT3_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_81B0	Pre Divider Register (CCM_PRE3)	32	R/W	0000_0000h	5.2.8.22/ 723

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_81B4	Pre Divider Register (CCM_PRE_ROOT3_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_81B8	Pre Divider Register (CCM_PRE_ROOT3_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_81BC	Pre Divider Register (CCM_PRE_ROOT3_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_81F0	Access Control Register (CCM_ACCESS_CTRL3)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_81F4	Access Control Register (CCM_ACCESS_CTRL_ROOT3_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_81F8	Access Control Register (CCM_ACCESS_CTRL_ROOT3_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_81FC	Access Control Register (CCM_ACCESS_CTRL_ROOT3_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8200	Target Register (CCM_TARGET_ROOT4)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8204	Target Register (CCM_TARGET_ROOT4_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8208	Target Register (CCM_TARGET_ROOT4_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_820C	Target Register (CCM_TARGET_ROOT4_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8210	Miscellaneous Register (CCM_MISC4)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8214	Miscellaneous Register (CCM_MISC_ROOT4_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8218	Miscellaneous Register (CCM_MISC_ROOT4_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_821C	Miscellaneous Register (CCM_MISC_ROOT4_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8220	Post Divider Register (CCM_POST4)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8224	Post Divider Register (CCM_POST_ROOT4_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8228	Post Divider Register (CCM_POST_ROOT4_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_822C	Post Divider Register (CCM_POST_ROOT4_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8230	Pre Divider Register (CCM_PRE4)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8234	Pre Divider Register (CCM_PRE_ROOT4_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8238	Pre Divider Register (CCM_PRE_ROOT4_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_823C	Pre Divider Register (CCM_PRE_ROOT4_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8270	Access Control Register (CCM_ACCESS_CTRL4)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8274	Access Control Register (CCM_ACCESS_CTRL_ROOT4_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8278	Access Control Register (CCM_ACCESS_CTRL_ROOT4_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_827C	Access Control Register (CCM_ACCESS_CTRL_ROOT4_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8280	Target Register (CCM_TARGET_ROOT5)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8284	Target Register (CCM_TARGET_ROOT5_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8288	Target Register (CCM_TARGET_ROOT5_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_828C	Target Register (CCM_TARGET_ROOT5_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8290	Miscellaneous Register (CCM_MISC5)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8294	Miscellaneous Register (CCM_MISC_ROOT5_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8298	Miscellaneous Register (CCM_MISC_ROOT5_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_829C	Miscellaneous Register (CCM_MISC_ROOT5_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_82A0	Post Divider Register (CCM_POST5)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_82A4	Post Divider Register (CCM_POST_ROOT5_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_82A8	Post Divider Register (CCM_POST_ROOT5_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_82AC	Post Divider Register (CCM_POST_ROOT5_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_82B0	Pre Divider Register (CCM_PRE5)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_82B4	Pre Divider Register (CCM_PRE_ROOT5_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_82B8	Pre Divider Register (CCM_PRE_ROOT5_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_82BC	Pre Divider Register (CCM_PRE_ROOT5_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_82F0	Access Control Register (CCM_ACCESS_CTRL5)	32	R/W	0000_0000h	5.2.8.26/ 735

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_82F4	Access Control Register (CCM_ACCESS_CTRL_ROOT5_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_82F8	Access Control Register (CCM_ACCESS_CTRL_ROOT5_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_82FC	Access Control Register (CCM_ACCESS_CTRL_ROOT5_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8300	Target Register (CCM_TARGET_ROOT6)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8304	Target Register (CCM_TARGET_ROOT6_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8308	Target Register (CCM_TARGET_ROOT6_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_830C	Target Register (CCM_TARGET_ROOT6_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8310	Miscellaneous Register (CCM_MISC6)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8314	Miscellaneous Register (CCM_MISC_ROOT6_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8318	Miscellaneous Register (CCM_MISC_ROOT6_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_831C	Miscellaneous Register (CCM_MISC_ROOT6_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8320	Post Divider Register (CCM_POST6)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8324	Post Divider Register (CCM_POST_ROOT6_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8328	Post Divider Register (CCM_POST_ROOT6_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_832C	Post Divider Register (CCM_POST_ROOT6_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8330	Pre Divider Register (CCM_PRE6)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8334	Pre Divider Register (CCM_PRE_ROOT6_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8338	Pre Divider Register (CCM_PRE_ROOT6_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_833C	Pre Divider Register (CCM_PRE_ROOT6_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8370	Access Control Register (CCM_ACCESS_CTRL6)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8374	Access Control Register (CCM_ACCESS_CTRL_ROOT6_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8378	Access Control Register (CCM_ACCESS_CTRL_ROOT6_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_837C	Access Control Register (CCM_ACCESS_CTRL_ROOT6_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8380	Target Register (CCM_TARGET_ROOT7)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8384	Target Register (CCM_TARGET_ROOT7_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8388	Target Register (CCM_TARGET_ROOT7_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_838C	Target Register (CCM_TARGET_ROOT7_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8390	Miscellaneous Register (CCM_MISC7)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8394	Miscellaneous Register (CCM_MISC_ROOT7_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8398	Miscellaneous Register (CCM_MISC_ROOT7_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_839C	Miscellaneous Register (CCM_MISC_ROOT7_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_83A0	Post Divider Register (CCM_POST7)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_83A4	Post Divider Register (CCM_POST_ROOT7_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_83A8	Post Divider Register (CCM_POST_ROOT7_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_83AC	Post Divider Register (CCM_POST_ROOT7_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_83B0	Pre Divider Register (CCM_PRE7)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_83B4	Pre Divider Register (CCM_PRE_ROOT7_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_83B8	Pre Divider Register (CCM_PRE_ROOT7_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_83BC	Pre Divider Register (CCM_PRE_ROOT7_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_83F0	Access Control Register (CCM_ACCESS_CTRL7)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_83F4	Access Control Register (CCM_ACCESS_CTRL_ROOT7_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_83F8	Access Control Register (CCM_ACCESS_CTRL_ROOT7_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_83FC	Access Control Register (CCM_ACCESS_CTRL_ROOT7_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8400	Target Register (CCM_TARGET_ROOT8)	32	R/W	0000_0000h	5.2.8.10/ 699

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8404	Target Register (CCM_TARGET_ROOT8_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8408	Target Register (CCM_TARGET_ROOT8_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_840C	Target Register (CCM_TARGET_ROOT8_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8410	Miscellaneous Register (CCM_MISC8)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8414	Miscellaneous Register (CCM_MISC_ROOT8_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8418	Miscellaneous Register (CCM_MISC_ROOT8_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_841C	Miscellaneous Register (CCM_MISC_ROOT8_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8420	Post Divider Register (CCM_POST8)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8424	Post Divider Register (CCM_POST_ROOT8_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8428	Post Divider Register (CCM_POST_ROOT8_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_842C	Post Divider Register (CCM_POST_ROOT8_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8430	Pre Divider Register (CCM_PRE8)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8434	Pre Divider Register (CCM_PRE_ROOT8_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8438	Pre Divider Register (CCM_PRE_ROOT8_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_843C	Pre Divider Register (CCM_PRE_ROOT8_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8470	Access Control Register (CCM_ACCESS_CTRL8)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8474	Access Control Register (CCM_ACCESS_CTRL_ROOT8_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8478	Access Control Register (CCM_ACCESS_CTRL_ROOT8_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_847C	Access Control Register (CCM_ACCESS_CTRL_ROOT8_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8480	Target Register (CCM_TARGET_ROOT9)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8484	Target Register (CCM_TARGET_ROOT9_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8488	Target Register (CCM_TARGET_ROOT9_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_848C	Target Register (CCM_TARGET_ROOT9_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8490	Miscellaneous Register (CCM_MISC9)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8494	Miscellaneous Register (CCM_MISC_ROOT9_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8498	Miscellaneous Register (CCM_MISC_ROOT9_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_849C	Miscellaneous Register (CCM_MISC_ROOT9_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_84A0	Post Divider Register (CCM_POST9)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_84A4	Post Divider Register (CCM_POST_ROOT9_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_84A8	Post Divider Register (CCM_POST_ROOT9_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_84AC	Post Divider Register (CCM_POST_ROOT9_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_84B0	Pre Divider Register (CCM_PRE9)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_84B4	Pre Divider Register (CCM_PRE_ROOT9_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_84B8	Pre Divider Register (CCM_PRE_ROOT9_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_84BC	Pre Divider Register (CCM_PRE_ROOT9_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_84F0	Access Control Register (CCM_ACCESS_CTRL9)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_84F4	Access Control Register (CCM_ACCESS_CTRL_ROOT9_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_84F8	Access Control Register (CCM_ACCESS_CTRL_ROOT9_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_84FC	Access Control Register (CCM_ACCESS_CTRL_ROOT9_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8500	Target Register (CCM_TARGET_ROOT10)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8504	Target Register (CCM_TARGET_ROOT10_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8508	Target Register (CCM_TARGET_ROOT10_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_850C	Target Register (CCM_TARGET_ROOT10_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8510	Miscellaneous Register (CCM_MISC10)	32	R/W	0000_0000h	5.2.8.14/ 707

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8514	Miscellaneous Register (CCM_MISC_ROOT10_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8518	Miscellaneous Register (CCM_MISC_ROOT10_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_851C	Miscellaneous Register (CCM_MISC_ROOT10_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8520	Post Divider Register (CCM_POST10)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8524	Post Divider Register (CCM_POST_ROOT10_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8528	Post Divider Register (CCM_POST_ROOT10_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_852C	Post Divider Register (CCM_POST_ROOT10_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8530	Pre Divider Register (CCM_PRE10)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8534	Pre Divider Register (CCM_PRE_ROOT10_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8538	Pre Divider Register (CCM_PRE_ROOT10_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_853C	Pre Divider Register (CCM_PRE_ROOT10_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8570	Access Control Register (CCM_ACCESS_CTRL10)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8574	Access Control Register (CCM_ACCESS_CTRL_ROOT10_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8578	Access Control Register (CCM_ACCESS_CTRL_ROOT10_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_857C	Access Control Register (CCM_ACCESS_CTRL_ROOT10_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8580	Target Register (CCM_TARGET_ROOT11)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8584	Target Register (CCM_TARGET_ROOT11_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8588	Target Register (CCM_TARGET_ROOT11_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_858C	Target Register (CCM_TARGET_ROOT11_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8590	Miscellaneous Register (CCM_MISC11)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8594	Miscellaneous Register (CCM_MISC_ROOT11_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8598	Miscellaneous Register (CCM_MISC_ROOT11_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_859C	Miscellaneous Register (CCM_MISC_ROOT11_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_85A0	Post Divider Register (CCM_POST11)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_85A4	Post Divider Register (CCM_POST_ROOT11_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_85A8	Post Divider Register (CCM_POST_ROOT11_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_85AC	Post Divider Register (CCM_POST_ROOT11_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_85B0	Pre Divider Register (CCM_PRE11)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_85B4	Pre Divider Register (CCM_PRE_ROOT11_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_85B8	Pre Divider Register (CCM_PRE_ROOT11_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_85BC	Pre Divider Register (CCM_PRE_ROOT11_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_85F0	Access Control Register (CCM_ACCESS_CTRL11)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_85F4	Access Control Register (CCM_ACCESS_CTRL_ROOT11_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_85F8	Access Control Register (CCM_ACCESS_CTRL_ROOT11_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_85FC	Access Control Register (CCM_ACCESS_CTRL_ROOT11_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8600	Target Register (CCM_TARGET_ROOT12)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8604	Target Register (CCM_TARGET_ROOT12_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8608	Target Register (CCM_TARGET_ROOT12_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_860C	Target Register (CCM_TARGET_ROOT12_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8610	Miscellaneous Register (CCM_MISC12)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8614	Miscellaneous Register (CCM_MISC_ROOT12_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8618	Miscellaneous Register (CCM_MISC_ROOT12_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_861C	Miscellaneous Register (CCM_MISC_ROOT12_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8620	Post Divider Register (CCM_POST12)	32	R/W	0000_0000h	5.2.8.18/ 711

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8624	Post Divider Register (CCM_POST_ROOT12_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8628	Post Divider Register (CCM_POST_ROOT12_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_862C	Post Divider Register (CCM_POST_ROOT12_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8630	Pre Divider Register (CCM_PRE12)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8634	Pre Divider Register (CCM_PRE_ROOT12_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8638	Pre Divider Register (CCM_PRE_ROOT12_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_863C	Pre Divider Register (CCM_PRE_ROOT12_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8670	Access Control Register (CCM_ACCESS_CTRL12)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8674	Access Control Register (CCM_ACCESS_CTRL_ROOT12_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8678	Access Control Register (CCM_ACCESS_CTRL_ROOT12_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_867C	Access Control Register (CCM_ACCESS_CTRL_ROOT12_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8680	Target Register (CCM_TARGET_ROOT13)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8684	Target Register (CCM_TARGET_ROOT13_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8688	Target Register (CCM_TARGET_ROOT13_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_868C	Target Register (CCM_TARGET_ROOT13_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8690	Miscellaneous Register (CCM_MISC13)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8694	Miscellaneous Register (CCM_MISC_ROOT13_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8698	Miscellaneous Register (CCM_MISC_ROOT13_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_869C	Miscellaneous Register (CCM_MISC_ROOT13_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_86A0	Post Divider Register (CCM_POST13)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_86A4	Post Divider Register (CCM_POST_ROOT13_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_86A8	Post Divider Register (CCM_POST_ROOT13_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_86AC	Post Divider Register (CCM_POST_ROOT13_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_86B0	Pre Divider Register (CCM_PRE13)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_86B4	Pre Divider Register (CCM_PRE_ROOT13_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_86B8	Pre Divider Register (CCM_PRE_ROOT13_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_86BC	Pre Divider Register (CCM_PRE_ROOT13_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_86F0	Access Control Register (CCM_ACCESS_CTRL13)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_86F4	Access Control Register (CCM_ACCESS_CTRL_ROOT13_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_86F8	Access Control Register (CCM_ACCESS_CTRL_ROOT13_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_86FC	Access Control Register (CCM_ACCESS_CTRL_ROOT13_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8700	Target Register (CCM_TARGET_ROOT14)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8704	Target Register (CCM_TARGET_ROOT14_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8708	Target Register (CCM_TARGET_ROOT14_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_870C	Target Register (CCM_TARGET_ROOT14_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8710	Miscellaneous Register (CCM_MISC14)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8714	Miscellaneous Register (CCM_MISC_ROOT14_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8718	Miscellaneous Register (CCM_MISC_ROOT14_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_871C	Miscellaneous Register (CCM_MISC_ROOT14_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8720	Post Divider Register (CCM_POST14)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8724	Post Divider Register (CCM_POST_ROOT14_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8728	Post Divider Register (CCM_POST_ROOT14_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_872C	Post Divider Register (CCM_POST_ROOT14_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8730	Pre Divider Register (CCM_PRE14)	32	R/W	0000_0000h	5.2.8.22/ 723

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8734	Pre Divider Register (CCM_PRE_ROOT14_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8738	Pre Divider Register (CCM_PRE_ROOT14_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_873C	Pre Divider Register (CCM_PRE_ROOT14_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8770	Access Control Register (CCM_ACCESS_CTRL14)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8774	Access Control Register (CCM_ACCESS_CTRL_ROOT14_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8778	Access Control Register (CCM_ACCESS_CTRL_ROOT14_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_877C	Access Control Register (CCM_ACCESS_CTRL_ROOT14_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8780	Target Register (CCM_TARGET_ROOT15)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8784	Target Register (CCM_TARGET_ROOT15_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8788	Target Register (CCM_TARGET_ROOT15_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_878C	Target Register (CCM_TARGET_ROOT15_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8790	Miscellaneous Register (CCM_MISC15)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8794	Miscellaneous Register (CCM_MISC_ROOT15_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8798	Miscellaneous Register (CCM_MISC_ROOT15_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_879C	Miscellaneous Register (CCM_MISC_ROOT15_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_87A0	Post Divider Register (CCM_POST15)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_87A4	Post Divider Register (CCM_POST_ROOT15_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_87A8	Post Divider Register (CCM_POST_ROOT15_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_87AC	Post Divider Register (CCM_POST_ROOT15_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_87B0	Pre Divider Register (CCM_PRE15)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_87B4	Pre Divider Register (CCM_PRE_ROOT15_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_87B8	Pre Divider Register (CCM_PRE_ROOT15_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_87BC	Pre Divider Register (CCM_PRE_ROOT15_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_87F0	Access Control Register (CCM_ACCESS_CTRL15)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_87F4	Access Control Register (CCM_ACCESS_CTRL_ROOT15_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_87F8	Access Control Register (CCM_ACCESS_CTRL_ROOT15_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_87FC	Access Control Register (CCM_ACCESS_CTRL_ROOT15_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8800	Target Register (CCM_TARGET_ROOT16)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8804	Target Register (CCM_TARGET_ROOT16_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8808	Target Register (CCM_TARGET_ROOT16_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_880C	Target Register (CCM_TARGET_ROOT16_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8810	Miscellaneous Register (CCM_MISC16)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8814	Miscellaneous Register (CCM_MISC_ROOT16_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8818	Miscellaneous Register (CCM_MISC_ROOT16_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_881C	Miscellaneous Register (CCM_MISC_ROOT16_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8820	Post Divider Register (CCM_POST16)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8824	Post Divider Register (CCM_POST_ROOT16_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8828	Post Divider Register (CCM_POST_ROOT16_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_882C	Post Divider Register (CCM_POST_ROOT16_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8830	Pre Divider Register (CCM_PRE16)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8834	Pre Divider Register (CCM_PRE_ROOT16_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8838	Pre Divider Register (CCM_PRE_ROOT16_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_883C	Pre Divider Register (CCM_PRE_ROOT16_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8870	Access Control Register (CCM_ACCESS_CTRL16)	32	R/W	0000_0000h	5.2.8.26/ 735

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8874	Access Control Register (CCM_ACCESS_CTRL_ROOT16_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8878	Access Control Register (CCM_ACCESS_CTRL_ROOT16_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_887C	Access Control Register (CCM_ACCESS_CTRL_ROOT16_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8880	Target Register (CCM_TARGET_ROOT17)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8884	Target Register (CCM_TARGET_ROOT17_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8888	Target Register (CCM_TARGET_ROOT17_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_888C	Target Register (CCM_TARGET_ROOT17_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8890	Miscellaneous Register (CCM_MISC17)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8894	Miscellaneous Register (CCM_MISC_ROOT17_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8898	Miscellaneous Register (CCM_MISC_ROOT17_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_889C	Miscellaneous Register (CCM_MISC_ROOT17_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_88A0	Post Divider Register (CCM_POST17)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_88A4	Post Divider Register (CCM_POST_ROOT17_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_88A8	Post Divider Register (CCM_POST_ROOT17_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_88AC	Post Divider Register (CCM_POST_ROOT17_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_88B0	Pre Divider Register (CCM_PRE17)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_88B4	Pre Divider Register (CCM_PRE_ROOT17_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_88B8	Pre Divider Register (CCM_PRE_ROOT17_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_88BC	Pre Divider Register (CCM_PRE_ROOT17_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_88F0	Access Control Register (CCM_ACCESS_CTRL17)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_88F4	Access Control Register (CCM_ACCESS_CTRL_ROOT17_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_88F8	Access Control Register (CCM_ACCESS_CTRL_ROOT17_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_88FC	Access Control Register (CCM_ACCESS_CTRL_ROOT17_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8900	Target Register (CCM_TARGET_ROOT18)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8904	Target Register (CCM_TARGET_ROOT18_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8908	Target Register (CCM_TARGET_ROOT18_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_890C	Target Register (CCM_TARGET_ROOT18_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8910	Miscellaneous Register (CCM_MISC18)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8914	Miscellaneous Register (CCM_MISC_ROOT18_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8918	Miscellaneous Register (CCM_MISC_ROOT18_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_891C	Miscellaneous Register (CCM_MISC_ROOT18_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8920	Post Divider Register (CCM_POST18)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8924	Post Divider Register (CCM_POST_ROOT18_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8928	Post Divider Register (CCM_POST_ROOT18_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_892C	Post Divider Register (CCM_POST_ROOT18_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8930	Pre Divider Register (CCM_PRE18)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8934	Pre Divider Register (CCM_PRE_ROOT18_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8938	Pre Divider Register (CCM_PRE_ROOT18_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_893C	Pre Divider Register (CCM_PRE_ROOT18_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8970	Access Control Register (CCM_ACCESS_CTRL18)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8974	Access Control Register (CCM_ACCESS_CTRL_ROOT18_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8978	Access Control Register (CCM_ACCESS_CTRL_ROOT18_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_897C	Access Control Register (CCM_ACCESS_CTRL_ROOT18_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8980	Target Register (CCM_TARGET_ROOT19)	32	R/W	0000_0000h	5.2.8.10/ 699

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8984	Target Register (CCM_TARGET_ROOT19_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8988	Target Register (CCM_TARGET_ROOT19_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_898C	Target Register (CCM_TARGET_ROOT19_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8990	Miscellaneous Register (CCM_MISC19)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8994	Miscellaneous Register (CCM_MISC_ROOT19_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8998	Miscellaneous Register (CCM_MISC_ROOT19_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_899C	Miscellaneous Register (CCM_MISC_ROOT19_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_89A0	Post Divider Register (CCM_POST19)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_89A4	Post Divider Register (CCM_POST_ROOT19_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_89A8	Post Divider Register (CCM_POST_ROOT19_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_89AC	Post Divider Register (CCM_POST_ROOT19_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_89B0	Pre Divider Register (CCM_PRE19)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_89B4	Pre Divider Register (CCM_PRE_ROOT19_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_89B8	Pre Divider Register (CCM_PRE_ROOT19_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_89BC	Pre Divider Register (CCM_PRE_ROOT19_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_89F0	Access Control Register (CCM_ACCESS_CTRL19)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_89F4	Access Control Register (CCM_ACCESS_CTRL_ROOT19_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_89F8	Access Control Register (CCM_ACCESS_CTRL_ROOT19_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_89FC	Access Control Register (CCM_ACCESS_CTRL_ROOT19_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8A00	Target Register (CCM_TARGET_ROOT20)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8A04	Target Register (CCM_TARGET_ROOT20_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8A08	Target Register (CCM_TARGET_ROOT20_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8A0C	Target Register (CCM_TARGET_ROOT20_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8A10	Miscellaneous Register (CCM_MISC20)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8A14	Miscellaneous Register (CCM_MISC_ROOT20_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8A18	Miscellaneous Register (CCM_MISC_ROOT20_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_8A1C	Miscellaneous Register (CCM_MISC_ROOT20_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8A20	Post Divider Register (CCM_POST20)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8A24	Post Divider Register (CCM_POST_ROOT20_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8A28	Post Divider Register (CCM_POST_ROOT20_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_8A2C	Post Divider Register (CCM_POST_ROOT20_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8A30	Pre Divider Register (CCM_PRE20)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8A34	Pre Divider Register (CCM_PRE_ROOT20_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8A38	Pre Divider Register (CCM_PRE_ROOT20_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_8A3C	Pre Divider Register (CCM_PRE_ROOT20_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8A70	Access Control Register (CCM_ACCESS_CTRL20)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8A74	Access Control Register (CCM_ACCESS_CTRL_ROOT20_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8A78	Access Control Register (CCM_ACCESS_CTRL_ROOT20_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_8A7C	Access Control Register (CCM_ACCESS_CTRL_ROOT20_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8A80	Target Register (CCM_TARGET_ROOT21)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8A84	Target Register (CCM_TARGET_ROOT21_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8A88	Target Register (CCM_TARGET_ROOT21_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_8A8C	Target Register (CCM_TARGET_ROOT21_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8A90	Miscellaneous Register (CCM_MISC21)	32	R/W	0000_0000h	5.2.8.14/ 707

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8A94	Miscellaneous Register (CCM_MISC_ROOT21_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8A98	Miscellaneous Register (CCM_MISC_ROOT21_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_8A9C	Miscellaneous Register (CCM_MISC_ROOT21_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8AA0	Post Divider Register (CCM_POST21)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8AA4	Post Divider Register (CCM_POST_ROOT21_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8AA8	Post Divider Register (CCM_POST_ROOT21_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_8AAC	Post Divider Register (CCM_POST_ROOT21_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8AB0	Pre Divider Register (CCM_PRE21)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8AB4	Pre Divider Register (CCM_PRE_ROOT21_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8AB8	Pre Divider Register (CCM_PRE_ROOT21_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_8ABC	Pre Divider Register (CCM_PRE_ROOT21_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8AF0	Access Control Register (CCM_ACCESS_CTRL21)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8AF4	Access Control Register (CCM_ACCESS_CTRL_ROOT21_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8AF8	Access Control Register (CCM_ACCESS_CTRL_ROOT21_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_8AFC	Access Control Register (CCM_ACCESS_CTRL_ROOT21_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8B00	Target Register (CCM_TARGET_ROOT22)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8B04	Target Register (CCM_TARGET_ROOT22_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8B08	Target Register (CCM_TARGET_ROOT22_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_8B0C	Target Register (CCM_TARGET_ROOT22_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8B10	Miscellaneous Register (CCM_MISC22)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8B14	Miscellaneous Register (CCM_MISC_ROOT22_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8B18	Miscellaneous Register (CCM_MISC_ROOT22_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8B1C	Miscellaneous Register (CCM_MISC_ROOT22_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8B20	Post Divider Register (CCM_POST22)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8B24	Post Divider Register (CCM_POST_ROOT22_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8B28	Post Divider Register (CCM_POST_ROOT22_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_8B2C	Post Divider Register (CCM_POST_ROOT22_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8B30	Pre Divider Register (CCM_PRE22)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8B34	Pre Divider Register (CCM_PRE_ROOT22_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8B38	Pre Divider Register (CCM_PRE_ROOT22_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_8B3C	Pre Divider Register (CCM_PRE_ROOT22_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8B70	Access Control Register (CCM_ACCESS_CTRL22)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8B74	Access Control Register (CCM_ACCESS_CTRL_ROOT22_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8B78	Access Control Register (CCM_ACCESS_CTRL_ROOT22_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_8B7C	Access Control Register (CCM_ACCESS_CTRL_ROOT22_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8B80	Target Register (CCM_TARGET_ROOT23)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8B84	Target Register (CCM_TARGET_ROOT23_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8B88	Target Register (CCM_TARGET_ROOT23_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_8B8C	Target Register (CCM_TARGET_ROOT23_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8B90	Miscellaneous Register (CCM_MISC23)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8B94	Miscellaneous Register (CCM_MISC_ROOT23_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8B98	Miscellaneous Register (CCM_MISC_ROOT23_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_8B9C	Miscellaneous Register (CCM_MISC_ROOT23_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8BA0	Post Divider Register (CCM_POST23)	32	R/W	0000_0000h	5.2.8.18/ 711

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8BA4	Post Divider Register (CCM_POST_ROOT23_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8BA8	Post Divider Register (CCM_POST_ROOT23_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_8BAC	Post Divider Register (CCM_POST_ROOT23_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8BB0	Pre Divider Register (CCM_PRE23)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8BB4	Pre Divider Register (CCM_PRE_ROOT23_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8BB8	Pre Divider Register (CCM_PRE_ROOT23_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_8BBC	Pre Divider Register (CCM_PRE_ROOT23_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8BF0	Access Control Register (CCM_ACCESS_CTRL23)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8BF4	Access Control Register (CCM_ACCESS_CTRL_ROOT23_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8BF8	Access Control Register (CCM_ACCESS_CTRL_ROOT23_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_8BFC	Access Control Register (CCM_ACCESS_CTRL_ROOT23_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8C00	Target Register (CCM_TARGET_ROOT24)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8C04	Target Register (CCM_TARGET_ROOT24_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8C08	Target Register (CCM_TARGET_ROOT24_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_8C0C	Target Register (CCM_TARGET_ROOT24_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8C10	Miscellaneous Register (CCM_MISC24)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8C14	Miscellaneous Register (CCM_MISC_ROOT24_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8C18	Miscellaneous Register (CCM_MISC_ROOT24_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_8C1C	Miscellaneous Register (CCM_MISC_ROOT24_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8C20	Post Divider Register (CCM_POST24)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8C24	Post Divider Register (CCM_POST_ROOT24_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8C28	Post Divider Register (CCM_POST_ROOT24_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8C2C	Post Divider Register (CCM_POST_ROOT24_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8C30	Pre Divider Register (CCM_PRE24)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8C34	Pre Divider Register (CCM_PRE_ROOT24_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8C38	Pre Divider Register (CCM_PRE_ROOT24_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_8C3C	Pre Divider Register (CCM_PRE_ROOT24_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8C70	Access Control Register (CCM_ACCESS_CTRL24)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8C74	Access Control Register (CCM_ACCESS_CTRL_ROOT24_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8C78	Access Control Register (CCM_ACCESS_CTRL_ROOT24_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_8C7C	Access Control Register (CCM_ACCESS_CTRL_ROOT24_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8C80	Target Register (CCM_TARGET_ROOT25)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8C84	Target Register (CCM_TARGET_ROOT25_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8C88	Target Register (CCM_TARGET_ROOT25_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_8C8C	Target Register (CCM_TARGET_ROOT25_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8C90	Miscellaneous Register (CCM_MISC25)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8C94	Miscellaneous Register (CCM_MISC_ROOT25_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8C98	Miscellaneous Register (CCM_MISC_ROOT25_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_8C9C	Miscellaneous Register (CCM_MISC_ROOT25_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8CA0	Post Divider Register (CCM_POST25)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8CA4	Post Divider Register (CCM_POST_ROOT25_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8CA8	Post Divider Register (CCM_POST_ROOT25_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_8CAC	Post Divider Register (CCM_POST_ROOT25_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8CB0	Pre Divider Register (CCM_PRE25)	32	R/W	0000_0000h	5.2.8.22/ 723

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8CB4	Pre Divider Register (CCM_PRE_ROOT25_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8CB8	Pre Divider Register (CCM_PRE_ROOT25_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_8CBC	Pre Divider Register (CCM_PRE_ROOT25_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8CF0	Access Control Register (CCM_ACCESS_CTRL25)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8CF4	Access Control Register (CCM_ACCESS_CTRL_ROOT25_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8CF8	Access Control Register (CCM_ACCESS_CTRL_ROOT25_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_8CFC	Access Control Register (CCM_ACCESS_CTRL_ROOT25_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8D00	Target Register (CCM_TARGET_ROOT26)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8D04	Target Register (CCM_TARGET_ROOT26_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8D08	Target Register (CCM_TARGET_ROOT26_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_8D0C	Target Register (CCM_TARGET_ROOT26_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8D10	Miscellaneous Register (CCM_MISC26)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8D14	Miscellaneous Register (CCM_MISC_ROOT26_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8D18	Miscellaneous Register (CCM_MISC_ROOT26_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_8D1C	Miscellaneous Register (CCM_MISC_ROOT26_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8D20	Post Divider Register (CCM_POST26)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8D24	Post Divider Register (CCM_POST_ROOT26_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8D28	Post Divider Register (CCM_POST_ROOT26_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_8D2C	Post Divider Register (CCM_POST_ROOT26_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8D30	Pre Divider Register (CCM_PRE26)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8D34	Pre Divider Register (CCM_PRE_ROOT26_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8D38	Pre Divider Register (CCM_PRE_ROOT26_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8D3C	Pre Divider Register (CCM_PRE_ROOT26_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8D70	Access Control Register (CCM_ACCESS_CTRL26)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8D74	Access Control Register (CCM_ACCESS_CTRL_ROOT26_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8D78	Access Control Register (CCM_ACCESS_CTRL_ROOT26_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_8D7C	Access Control Register (CCM_ACCESS_CTRL_ROOT26_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8D80	Target Register (CCM_TARGET_ROOT27)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8D84	Target Register (CCM_TARGET_ROOT27_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8D88	Target Register (CCM_TARGET_ROOT27_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_8D8C	Target Register (CCM_TARGET_ROOT27_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8D90	Miscellaneous Register (CCM_MISC27)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8D94	Miscellaneous Register (CCM_MISC_ROOT27_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8D98	Miscellaneous Register (CCM_MISC_ROOT27_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_8D9C	Miscellaneous Register (CCM_MISC_ROOT27_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8DA0	Post Divider Register (CCM_POST27)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8DA4	Post Divider Register (CCM_POST_ROOT27_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8DA8	Post Divider Register (CCM_POST_ROOT27_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_8DAC	Post Divider Register (CCM_POST_ROOT27_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8DB0	Pre Divider Register (CCM_PRE27)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8DB4	Pre Divider Register (CCM_PRE_ROOT27_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8DB8	Pre Divider Register (CCM_PRE_ROOT27_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_8DBC	Pre Divider Register (CCM_PRE_ROOT27_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8DF0	Access Control Register (CCM_ACCESS_CTRL27)	32	R/W	0000_0000h	5.2.8.26/ 735

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8DF4	Access Control Register (CCM_ACCESS_CTRL_ROOT27_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8DF8	Access Control Register (CCM_ACCESS_CTRL_ROOT27_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_8DFC	Access Control Register (CCM_ACCESS_CTRL_ROOT27_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8E00	Target Register (CCM_TARGET_ROOT28)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8E04	Target Register (CCM_TARGET_ROOT28_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8E08	Target Register (CCM_TARGET_ROOT28_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_8E0C	Target Register (CCM_TARGET_ROOT28_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8E10	Miscellaneous Register (CCM_MISC28)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8E14	Miscellaneous Register (CCM_MISC_ROOT28_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8E18	Miscellaneous Register (CCM_MISC_ROOT28_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_8E1C	Miscellaneous Register (CCM_MISC_ROOT28_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8E20	Post Divider Register (CCM_POST28)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8E24	Post Divider Register (CCM_POST_ROOT28_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8E28	Post Divider Register (CCM_POST_ROOT28_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_8E2C	Post Divider Register (CCM_POST_ROOT28_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8E30	Pre Divider Register (CCM_PRE28)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8E34	Pre Divider Register (CCM_PRE_ROOT28_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8E38	Pre Divider Register (CCM_PRE_ROOT28_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_8E3C	Pre Divider Register (CCM_PRE_ROOT28_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8E70	Access Control Register (CCM_ACCESS_CTRL28)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8E74	Access Control Register (CCM_ACCESS_CTRL_ROOT28_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8E78	Access Control Register (CCM_ACCESS_CTRL_ROOT28_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8E7C	Access Control Register (CCM_ACCESS_CTRL_ROOT28_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8E80	Target Register (CCM_TARGET_ROOT29)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8E84	Target Register (CCM_TARGET_ROOT29_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8E88	Target Register (CCM_TARGET_ROOT29_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_8E8C	Target Register (CCM_TARGET_ROOT29_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8E90	Miscellaneous Register (CCM_MISC29)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8E94	Miscellaneous Register (CCM_MISC_ROOT29_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8E98	Miscellaneous Register (CCM_MISC_ROOT29_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_8E9C	Miscellaneous Register (CCM_MISC_ROOT29_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8EA0	Post Divider Register (CCM_POST29)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8EA4	Post Divider Register (CCM_POST_ROOT29_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8EA8	Post Divider Register (CCM_POST_ROOT29_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_8EAC	Post Divider Register (CCM_POST_ROOT29_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8EB0	Pre Divider Register (CCM_PRE29)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8EB4	Pre Divider Register (CCM_PRE_ROOT29_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8EB8	Pre Divider Register (CCM_PRE_ROOT29_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_8EBC	Pre Divider Register (CCM_PRE_ROOT29_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8EF0	Access Control Register (CCM_ACCESS_CTRL29)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8EF4	Access Control Register (CCM_ACCESS_CTRL_ROOT29_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8EF8	Access Control Register (CCM_ACCESS_CTRL_ROOT29_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_8EFC	Access Control Register (CCM_ACCESS_CTRL_ROOT29_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8F00	Target Register (CCM_TARGET_ROOT30)	32	R/W	0000_0000h	5.2.8.10/ 699

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8F04	Target Register (CCM_TARGET_ROOT30_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8F08	Target Register (CCM_TARGET_ROOT30_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_8F0C	Target Register (CCM_TARGET_ROOT30_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8F10	Miscellaneous Register (CCM_MISC30)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8F14	Miscellaneous Register (CCM_MISC_ROOT30_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8F18	Miscellaneous Register (CCM_MISC_ROOT30_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_8F1C	Miscellaneous Register (CCM_MISC_ROOT30_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8F20	Post Divider Register (CCM_POST30)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8F24	Post Divider Register (CCM_POST_ROOT30_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8F28	Post Divider Register (CCM_POST_ROOT30_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_8F2C	Post Divider Register (CCM_POST_ROOT30_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8F30	Pre Divider Register (CCM_PRE30)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8F34	Pre Divider Register (CCM_PRE_ROOT30_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8F38	Pre Divider Register (CCM_PRE_ROOT30_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_8F3C	Pre Divider Register (CCM_PRE_ROOT30_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8F70	Access Control Register (CCM_ACCESS_CTRL30)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8F74	Access Control Register (CCM_ACCESS_CTRL_ROOT30_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8F78	Access Control Register (CCM_ACCESS_CTRL_ROOT30_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_8F7C	Access Control Register (CCM_ACCESS_CTRL_ROOT30_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_8F80	Target Register (CCM_TARGET_ROOT31)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_8F84	Target Register (CCM_TARGET_ROOT31_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_8F88	Target Register (CCM_TARGET_ROOT31_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_8F8C	Target Register (CCM_TARGET_ROOT31_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_8F90	Miscellaneous Register (CCM_MISC31)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_8F94	Miscellaneous Register (CCM_MISC_ROOT31_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_8F98	Miscellaneous Register (CCM_MISC_ROOT31_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_8F9C	Miscellaneous Register (CCM_MISC_ROOT31_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_8FA0	Post Divider Register (CCM_POST31)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_8FA4	Post Divider Register (CCM_POST_ROOT31_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_8FA8	Post Divider Register (CCM_POST_ROOT31_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_8FAC	Post Divider Register (CCM_POST_ROOT31_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_8FB0	Pre Divider Register (CCM_PRE31)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_8FB4	Pre Divider Register (CCM_PRE_ROOT31_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_8FB8	Pre Divider Register (CCM_PRE_ROOT31_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_8FBC	Pre Divider Register (CCM_PRE_ROOT31_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_8FF0	Access Control Register (CCM_ACCESS_CTRL31)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_8FF4	Access Control Register (CCM_ACCESS_CTRL_ROOT31_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_8FF8	Access Control Register (CCM_ACCESS_CTRL_ROOT31_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_8FFC	Access Control Register (CCM_ACCESS_CTRL_ROOT31_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9000	Target Register (CCM_TARGET_ROOT32)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9004	Target Register (CCM_TARGET_ROOT32_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9008	Target Register (CCM_TARGET_ROOT32_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_900C	Target Register (CCM_TARGET_ROOT32_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9010	Miscellaneous Register (CCM_MISC32)	32	R/W	0000_0000h	5.2.8.14/ 707

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9014	Miscellaneous Register (CCM_MISC_ROOT32_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9018	Miscellaneous Register (CCM_MISC_ROOT32_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_901C	Miscellaneous Register (CCM_MISC_ROOT32_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9020	Post Divider Register (CCM_POST32)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9024	Post Divider Register (CCM_POST_ROOT32_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9028	Post Divider Register (CCM_POST_ROOT32_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_902C	Post Divider Register (CCM_POST_ROOT32_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9030	Pre Divider Register (CCM_PRE32)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9034	Pre Divider Register (CCM_PRE_ROOT32_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9038	Pre Divider Register (CCM_PRE_ROOT32_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_903C	Pre Divider Register (CCM_PRE_ROOT32_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9070	Access Control Register (CCM_ACCESS_CTRL32)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9074	Access Control Register (CCM_ACCESS_CTRL_ROOT32_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9078	Access Control Register (CCM_ACCESS_CTRL_ROOT32_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_907C	Access Control Register (CCM_ACCESS_CTRL_ROOT32_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9080	Target Register (CCM_TARGET_ROOT33)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9084	Target Register (CCM_TARGET_ROOT33_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9088	Target Register (CCM_TARGET_ROOT33_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_908C	Target Register (CCM_TARGET_ROOT33_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9090	Miscellaneous Register (CCM_MISC33)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9094	Miscellaneous Register (CCM_MISC_ROOT33_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9098	Miscellaneous Register (CCM_MISC_ROOT33_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_909C	Miscellaneous Register (CCM_MISC_ROOT33_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_90A0	Post Divider Register (CCM_POST33)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_90A4	Post Divider Register (CCM_POST_ROOT33_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_90A8	Post Divider Register (CCM_POST_ROOT33_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_90AC	Post Divider Register (CCM_POST_ROOT33_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_90B0	Pre Divider Register (CCM_PRE33)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_90B4	Pre Divider Register (CCM_PRE_ROOT33_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_90B8	Pre Divider Register (CCM_PRE_ROOT33_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_90BC	Pre Divider Register (CCM_PRE_ROOT33_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_90F0	Access Control Register (CCM_ACCESS_CTRL33)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_90F4	Access Control Register (CCM_ACCESS_CTRL_ROOT33_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_90F8	Access Control Register (CCM_ACCESS_CTRL_ROOT33_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_90FC	Access Control Register (CCM_ACCESS_CTRL_ROOT33_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9100	Target Register (CCM_TARGET_ROOT34)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9104	Target Register (CCM_TARGET_ROOT34_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9108	Target Register (CCM_TARGET_ROOT34_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_910C	Target Register (CCM_TARGET_ROOT34_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9110	Miscellaneous Register (CCM_MISC34)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9114	Miscellaneous Register (CCM_MISC_ROOT34_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9118	Miscellaneous Register (CCM_MISC_ROOT34_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_911C	Miscellaneous Register (CCM_MISC_ROOT34_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9120	Post Divider Register (CCM_POST34)	32	R/W	0000_0000h	5.2.8.18/ 711

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9124	Post Divider Register (CCM_POST_ROOT34_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9128	Post Divider Register (CCM_POST_ROOT34_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_912C	Post Divider Register (CCM_POST_ROOT34_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9130	Pre Divider Register (CCM_PRE34)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9134	Pre Divider Register (CCM_PRE_ROOT34_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9138	Pre Divider Register (CCM_PRE_ROOT34_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_913C	Pre Divider Register (CCM_PRE_ROOT34_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9170	Access Control Register (CCM_ACCESS_CTRL34)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9174	Access Control Register (CCM_ACCESS_CTRL_ROOT34_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9178	Access Control Register (CCM_ACCESS_CTRL_ROOT34_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_917C	Access Control Register (CCM_ACCESS_CTRL_ROOT34_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9180	Target Register (CCM_TARGET_ROOT35)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9184	Target Register (CCM_TARGET_ROOT35_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9188	Target Register (CCM_TARGET_ROOT35_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_918C	Target Register (CCM_TARGET_ROOT35_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9190	Miscellaneous Register (CCM_MISC35)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9194	Miscellaneous Register (CCM_MISC_ROOT35_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9198	Miscellaneous Register (CCM_MISC_ROOT35_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_919C	Miscellaneous Register (CCM_MISC_ROOT35_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_91A0	Post Divider Register (CCM_POST35)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_91A4	Post Divider Register (CCM_POST_ROOT35_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_91A8	Post Divider Register (CCM_POST_ROOT35_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_91AC	Post Divider Register (CCM_POST_ROOT35_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_91B0	Pre Divider Register (CCM_PRE35)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_91B4	Pre Divider Register (CCM_PRE_ROOT35_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_91B8	Pre Divider Register (CCM_PRE_ROOT35_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_91BC	Pre Divider Register (CCM_PRE_ROOT35_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_91F0	Access Control Register (CCM_ACCESS_CTRL35)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_91F4	Access Control Register (CCM_ACCESS_CTRL_ROOT35_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_91F8	Access Control Register (CCM_ACCESS_CTRL_ROOT35_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_91FC	Access Control Register (CCM_ACCESS_CTRL_ROOT35_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9200	Target Register (CCM_TARGET_ROOT36)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9204	Target Register (CCM_TARGET_ROOT36_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9208	Target Register (CCM_TARGET_ROOT36_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_920C	Target Register (CCM_TARGET_ROOT36_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9210	Miscellaneous Register (CCM_MISC36)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9214	Miscellaneous Register (CCM_MISC_ROOT36_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9218	Miscellaneous Register (CCM_MISC_ROOT36_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_921C	Miscellaneous Register (CCM_MISC_ROOT36_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9220	Post Divider Register (CCM_POST36)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9224	Post Divider Register (CCM_POST_ROOT36_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9228	Post Divider Register (CCM_POST_ROOT36_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_922C	Post Divider Register (CCM_POST_ROOT36_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9230	Pre Divider Register (CCM_PRE36)	32	R/W	0000_0000h	5.2.8.22/ 723

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9234	Pre Divider Register (CCM_PRE_ROOT36_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9238	Pre Divider Register (CCM_PRE_ROOT36_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_923C	Pre Divider Register (CCM_PRE_ROOT36_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9270	Access Control Register (CCM_ACCESS_CTRL36)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9274	Access Control Register (CCM_ACCESS_CTRL_ROOT36_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9278	Access Control Register (CCM_ACCESS_CTRL_ROOT36_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_927C	Access Control Register (CCM_ACCESS_CTRL_ROOT36_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9280	Target Register (CCM_TARGET_ROOT37)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9284	Target Register (CCM_TARGET_ROOT37_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9288	Target Register (CCM_TARGET_ROOT37_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_928C	Target Register (CCM_TARGET_ROOT37_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9290	Miscellaneous Register (CCM_MISC37)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9294	Miscellaneous Register (CCM_MISC_ROOT37_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9298	Miscellaneous Register (CCM_MISC_ROOT37_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_929C	Miscellaneous Register (CCM_MISC_ROOT37_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_92A0	Post Divider Register (CCM_POST37)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_92A4	Post Divider Register (CCM_POST_ROOT37_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_92A8	Post Divider Register (CCM_POST_ROOT37_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_92AC	Post Divider Register (CCM_POST_ROOT37_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_92B0	Pre Divider Register (CCM_PRE37)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_92B4	Pre Divider Register (CCM_PRE_ROOT37_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_92B8	Pre Divider Register (CCM_PRE_ROOT37_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_92BC	Pre Divider Register (CCM_PRE_ROOT37_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_92F0	Access Control Register (CCM_ACCESS_CTRL37)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_92F4	Access Control Register (CCM_ACCESS_CTRL_ROOT37_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_92F8	Access Control Register (CCM_ACCESS_CTRL_ROOT37_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_92FC	Access Control Register (CCM_ACCESS_CTRL_ROOT37_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9300	Target Register (CCM_TARGET_ROOT38)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9304	Target Register (CCM_TARGET_ROOT38_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9308	Target Register (CCM_TARGET_ROOT38_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_930C	Target Register (CCM_TARGET_ROOT38_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9310	Miscellaneous Register (CCM_MISC38)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9314	Miscellaneous Register (CCM_MISC_ROOT38_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9318	Miscellaneous Register (CCM_MISC_ROOT38_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_931C	Miscellaneous Register (CCM_MISC_ROOT38_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9320	Post Divider Register (CCM_POST38)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9324	Post Divider Register (CCM_POST_ROOT38_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9328	Post Divider Register (CCM_POST_ROOT38_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_932C	Post Divider Register (CCM_POST_ROOT38_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9330	Pre Divider Register (CCM_PRE38)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9334	Pre Divider Register (CCM_PRE_ROOT38_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9338	Pre Divider Register (CCM_PRE_ROOT38_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_933C	Pre Divider Register (CCM_PRE_ROOT38_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9370	Access Control Register (CCM_ACCESS_CTRL38)	32	R/W	0000_0000h	5.2.8.26/ 735

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9374	Access Control Register (CCM_ACCESS_CTRL_ROOT38_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9378	Access Control Register (CCM_ACCESS_CTRL_ROOT38_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_937C	Access Control Register (CCM_ACCESS_CTRL_ROOT38_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9380	Target Register (CCM_TARGET_ROOT39)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9384	Target Register (CCM_TARGET_ROOT39_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9388	Target Register (CCM_TARGET_ROOT39_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_938C	Target Register (CCM_TARGET_ROOT39_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9390	Miscellaneous Register (CCM_MISC39)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9394	Miscellaneous Register (CCM_MISC_ROOT39_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9398	Miscellaneous Register (CCM_MISC_ROOT39_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_939C	Miscellaneous Register (CCM_MISC_ROOT39_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_93A0	Post Divider Register (CCM_POST39)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_93A4	Post Divider Register (CCM_POST_ROOT39_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_93A8	Post Divider Register (CCM_POST_ROOT39_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_93AC	Post Divider Register (CCM_POST_ROOT39_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_93B0	Pre Divider Register (CCM_PRE39)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_93B4	Pre Divider Register (CCM_PRE_ROOT39_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_93B8	Pre Divider Register (CCM_PRE_ROOT39_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_93BC	Pre Divider Register (CCM_PRE_ROOT39_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_93F0	Access Control Register (CCM_ACCESS_CTRL39)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_93F4	Access Control Register (CCM_ACCESS_CTRL_ROOT39_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_93F8	Access Control Register (CCM_ACCESS_CTRL_ROOT39_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_93FC	Access Control Register (CCM_ACCESS_CTRL_ROOT39_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9400	Target Register (CCM_TARGET_ROOT40)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9404	Target Register (CCM_TARGET_ROOT40_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9408	Target Register (CCM_TARGET_ROOT40_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_940C	Target Register (CCM_TARGET_ROOT40_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9410	Miscellaneous Register (CCM_MISC40)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9414	Miscellaneous Register (CCM_MISC_ROOT40_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9418	Miscellaneous Register (CCM_MISC_ROOT40_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_941C	Miscellaneous Register (CCM_MISC_ROOT40_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9420	Post Divider Register (CCM_POST40)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9424	Post Divider Register (CCM_POST_ROOT40_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9428	Post Divider Register (CCM_POST_ROOT40_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_942C	Post Divider Register (CCM_POST_ROOT40_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9430	Pre Divider Register (CCM_PRE40)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9434	Pre Divider Register (CCM_PRE_ROOT40_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9438	Pre Divider Register (CCM_PRE_ROOT40_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_943C	Pre Divider Register (CCM_PRE_ROOT40_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9470	Access Control Register (CCM_ACCESS_CTRL40)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9474	Access Control Register (CCM_ACCESS_CTRL_ROOT40_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9478	Access Control Register (CCM_ACCESS_CTRL_ROOT40_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_947C	Access Control Register (CCM_ACCESS_CTRL_ROOT40_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9480	Target Register (CCM_TARGET_ROOT41)	32	R/W	0000_0000h	5.2.8.10/ 699

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9484	Target Register (CCM_TARGET_ROOT41_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9488	Target Register (CCM_TARGET_ROOT41_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_948C	Target Register (CCM_TARGET_ROOT41_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9490	Miscellaneous Register (CCM_MISC41)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9494	Miscellaneous Register (CCM_MISC_ROOT41_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9498	Miscellaneous Register (CCM_MISC_ROOT41_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_949C	Miscellaneous Register (CCM_MISC_ROOT41_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_94A0	Post Divider Register (CCM_POST41)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_94A4	Post Divider Register (CCM_POST_ROOT41_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_94A8	Post Divider Register (CCM_POST_ROOT41_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_94AC	Post Divider Register (CCM_POST_ROOT41_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_94B0	Pre Divider Register (CCM_PRE41)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_94B4	Pre Divider Register (CCM_PRE_ROOT41_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_94B8	Pre Divider Register (CCM_PRE_ROOT41_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_94BC	Pre Divider Register (CCM_PRE_ROOT41_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_94F0	Access Control Register (CCM_ACCESS_CTRL41)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_94F4	Access Control Register (CCM_ACCESS_CTRL_ROOT41_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_94F8	Access Control Register (CCM_ACCESS_CTRL_ROOT41_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_94FC	Access Control Register (CCM_ACCESS_CTRL_ROOT41_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9500	Target Register (CCM_TARGET_ROOT42)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9504	Target Register (CCM_TARGET_ROOT42_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9508	Target Register (CCM_TARGET_ROOT42_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_950C	Target Register (CCM_TARGET_ROOT42_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9510	Miscellaneous Register (CCM_MISC42)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9514	Miscellaneous Register (CCM_MISC_ROOT42_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9518	Miscellaneous Register (CCM_MISC_ROOT42_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_951C	Miscellaneous Register (CCM_MISC_ROOT42_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9520	Post Divider Register (CCM_POST42)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9524	Post Divider Register (CCM_POST_ROOT42_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9528	Post Divider Register (CCM_POST_ROOT42_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_952C	Post Divider Register (CCM_POST_ROOT42_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9530	Pre Divider Register (CCM_PRE42)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9534	Pre Divider Register (CCM_PRE_ROOT42_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9538	Pre Divider Register (CCM_PRE_ROOT42_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_953C	Pre Divider Register (CCM_PRE_ROOT42_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9570	Access Control Register (CCM_ACCESS_CTRL42)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9574	Access Control Register (CCM_ACCESS_CTRL_ROOT42_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9578	Access Control Register (CCM_ACCESS_CTRL_ROOT42_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_957C	Access Control Register (CCM_ACCESS_CTRL_ROOT42_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9580	Target Register (CCM_TARGET_ROOT43)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9584	Target Register (CCM_TARGET_ROOT43_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9588	Target Register (CCM_TARGET_ROOT43_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_958C	Target Register (CCM_TARGET_ROOT43_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9590	Miscellaneous Register (CCM_MISC43)	32	R/W	0000_0000h	5.2.8.14/ 707

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9594	Miscellaneous Register (CCM_MISC_ROOT43_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9598	Miscellaneous Register (CCM_MISC_ROOT43_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_959C	Miscellaneous Register (CCM_MISC_ROOT43_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_95A0	Post Divider Register (CCM_POST43)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_95A4	Post Divider Register (CCM_POST_ROOT43_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_95A8	Post Divider Register (CCM_POST_ROOT43_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_95AC	Post Divider Register (CCM_POST_ROOT43_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_95B0	Pre Divider Register (CCM_PRE43)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_95B4	Pre Divider Register (CCM_PRE_ROOT43_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_95B8	Pre Divider Register (CCM_PRE_ROOT43_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_95BC	Pre Divider Register (CCM_PRE_ROOT43_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_95F0	Access Control Register (CCM_ACCESS_CTRL43)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_95F4	Access Control Register (CCM_ACCESS_CTRL_ROOT43_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_95F8	Access Control Register (CCM_ACCESS_CTRL_ROOT43_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_95FC	Access Control Register (CCM_ACCESS_CTRL_ROOT43_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9600	Target Register (CCM_TARGET_ROOT44)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9604	Target Register (CCM_TARGET_ROOT44_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9608	Target Register (CCM_TARGET_ROOT44_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_960C	Target Register (CCM_TARGET_ROOT44_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9610	Miscellaneous Register (CCM_MISC44)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9614	Miscellaneous Register (CCM_MISC_ROOT44_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9618	Miscellaneous Register (CCM_MISC_ROOT44_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_961C	Miscellaneous Register (CCM_MISC_ROOT44_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9620	Post Divider Register (CCM_POST44)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9624	Post Divider Register (CCM_POST_ROOT44_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9628	Post Divider Register (CCM_POST_ROOT44_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_962C	Post Divider Register (CCM_POST_ROOT44_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9630	Pre Divider Register (CCM_PRE44)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9634	Pre Divider Register (CCM_PRE_ROOT44_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9638	Pre Divider Register (CCM_PRE_ROOT44_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_963C	Pre Divider Register (CCM_PRE_ROOT44_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9670	Access Control Register (CCM_ACCESS_CTRL44)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9674	Access Control Register (CCM_ACCESS_CTRL_ROOT44_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9678	Access Control Register (CCM_ACCESS_CTRL_ROOT44_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_967C	Access Control Register (CCM_ACCESS_CTRL_ROOT44_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9680	Target Register (CCM_TARGET_ROOT45)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9684	Target Register (CCM_TARGET_ROOT45_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9688	Target Register (CCM_TARGET_ROOT45_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_968C	Target Register (CCM_TARGET_ROOT45_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9690	Miscellaneous Register (CCM_MISC45)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9694	Miscellaneous Register (CCM_MISC_ROOT45_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9698	Miscellaneous Register (CCM_MISC_ROOT45_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_969C	Miscellaneous Register (CCM_MISC_ROOT45_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_96A0	Post Divider Register (CCM_POST45)	32	R/W	0000_0000h	5.2.8.18/ 711

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_96A4	Post Divider Register (CCM_POST_ROOT45_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_96A8	Post Divider Register (CCM_POST_ROOT45_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_96AC	Post Divider Register (CCM_POST_ROOT45_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_96B0	Pre Divider Register (CCM_PRE45)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_96B4	Pre Divider Register (CCM_PRE_ROOT45_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_96B8	Pre Divider Register (CCM_PRE_ROOT45_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_96BC	Pre Divider Register (CCM_PRE_ROOT45_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_96F0	Access Control Register (CCM_ACCESS_CTRL45)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_96F4	Access Control Register (CCM_ACCESS_CTRL_ROOT45_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_96F8	Access Control Register (CCM_ACCESS_CTRL_ROOT45_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_96FC	Access Control Register (CCM_ACCESS_CTRL_ROOT45_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9700	Target Register (CCM_TARGET_ROOT46)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9704	Target Register (CCM_TARGET_ROOT46_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9708	Target Register (CCM_TARGET_ROOT46_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_970C	Target Register (CCM_TARGET_ROOT46_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9710	Miscellaneous Register (CCM_MISC46)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9714	Miscellaneous Register (CCM_MISC_ROOT46_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9718	Miscellaneous Register (CCM_MISC_ROOT46_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_971C	Miscellaneous Register (CCM_MISC_ROOT46_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9720	Post Divider Register (CCM_POST46)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9724	Post Divider Register (CCM_POST_ROOT46_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9728	Post Divider Register (CCM_POST_ROOT46_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_972C	Post Divider Register (CCM_POST_ROOT46_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9730	Pre Divider Register (CCM_PRE46)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9734	Pre Divider Register (CCM_PRE_ROOT46_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9738	Pre Divider Register (CCM_PRE_ROOT46_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_973C	Pre Divider Register (CCM_PRE_ROOT46_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9770	Access Control Register (CCM_ACCESS_CTRL46)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9774	Access Control Register (CCM_ACCESS_CTRL_ROOT46_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9778	Access Control Register (CCM_ACCESS_CTRL_ROOT46_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_977C	Access Control Register (CCM_ACCESS_CTRL_ROOT46_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9780	Target Register (CCM_TARGET_ROOT47)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9784	Target Register (CCM_TARGET_ROOT47_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9788	Target Register (CCM_TARGET_ROOT47_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_978C	Target Register (CCM_TARGET_ROOT47_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9790	Miscellaneous Register (CCM_MISC47)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9794	Miscellaneous Register (CCM_MISC_ROOT47_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9798	Miscellaneous Register (CCM_MISC_ROOT47_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_979C	Miscellaneous Register (CCM_MISC_ROOT47_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_97A0	Post Divider Register (CCM_POST47)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_97A4	Post Divider Register (CCM_POST_ROOT47_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_97A8	Post Divider Register (CCM_POST_ROOT47_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_97AC	Post Divider Register (CCM_POST_ROOT47_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_97B0	Pre Divider Register (CCM_PRE47)	32	R/W	0000_0000h	5.2.8.22/ 723

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_97B4	Pre Divider Register (CCM_PRE_ROOT47_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_97B8	Pre Divider Register (CCM_PRE_ROOT47_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_97BC	Pre Divider Register (CCM_PRE_ROOT47_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_97F0	Access Control Register (CCM_ACCESS_CTRL47)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_97F4	Access Control Register (CCM_ACCESS_CTRL_ROOT47_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_97F8	Access Control Register (CCM_ACCESS_CTRL_ROOT47_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_97FC	Access Control Register (CCM_ACCESS_CTRL_ROOT47_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9800	Target Register (CCM_TARGET_ROOT48)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9804	Target Register (CCM_TARGET_ROOT48_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9808	Target Register (CCM_TARGET_ROOT48_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_980C	Target Register (CCM_TARGET_ROOT48_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9810	Miscellaneous Register (CCM_MISC48)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9814	Miscellaneous Register (CCM_MISC_ROOT48_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9818	Miscellaneous Register (CCM_MISC_ROOT48_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_981C	Miscellaneous Register (CCM_MISC_ROOT48_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9820	Post Divider Register (CCM_POST48)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9824	Post Divider Register (CCM_POST_ROOT48_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9828	Post Divider Register (CCM_POST_ROOT48_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_982C	Post Divider Register (CCM_POST_ROOT48_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9830	Pre Divider Register (CCM_PRE48)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9834	Pre Divider Register (CCM_PRE_ROOT48_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9838	Pre Divider Register (CCM_PRE_ROOT48_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_983C	Pre Divider Register (CCM_PRE_ROOT48_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9870	Access Control Register (CCM_ACCESS_CTRL48)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9874	Access Control Register (CCM_ACCESS_CTRL_ROOT48_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9878	Access Control Register (CCM_ACCESS_CTRL_ROOT48_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_987C	Access Control Register (CCM_ACCESS_CTRL_ROOT48_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9880	Target Register (CCM_TARGET_ROOT49)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9884	Target Register (CCM_TARGET_ROOT49_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9888	Target Register (CCM_TARGET_ROOT49_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_988C	Target Register (CCM_TARGET_ROOT49_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9890	Miscellaneous Register (CCM_MISC49)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9894	Miscellaneous Register (CCM_MISC_ROOT49_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9898	Miscellaneous Register (CCM_MISC_ROOT49_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_989C	Miscellaneous Register (CCM_MISC_ROOT49_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_98A0	Post Divider Register (CCM_POST49)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_98A4	Post Divider Register (CCM_POST_ROOT49_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_98A8	Post Divider Register (CCM_POST_ROOT49_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_98AC	Post Divider Register (CCM_POST_ROOT49_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_98B0	Pre Divider Register (CCM_PRE49)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_98B4	Pre Divider Register (CCM_PRE_ROOT49_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_98B8	Pre Divider Register (CCM_PRE_ROOT49_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_98BC	Pre Divider Register (CCM_PRE_ROOT49_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_98F0	Access Control Register (CCM_ACCESS_CTRL49)	32	R/W	0000_0000h	5.2.8.26/ 735

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_98F4	Access Control Register (CCM_ACCESS_CTRL_ROOT49_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_98F8	Access Control Register (CCM_ACCESS_CTRL_ROOT49_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_98FC	Access Control Register (CCM_ACCESS_CTRL_ROOT49_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9900	Target Register (CCM_TARGET_ROOT50)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9904	Target Register (CCM_TARGET_ROOT50_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9908	Target Register (CCM_TARGET_ROOT50_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_990C	Target Register (CCM_TARGET_ROOT50_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9910	Miscellaneous Register (CCM_MISC50)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9914	Miscellaneous Register (CCM_MISC_ROOT50_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9918	Miscellaneous Register (CCM_MISC_ROOT50_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_991C	Miscellaneous Register (CCM_MISC_ROOT50_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9920	Post Divider Register (CCM_POST50)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9924	Post Divider Register (CCM_POST_ROOT50_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9928	Post Divider Register (CCM_POST_ROOT50_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_992C	Post Divider Register (CCM_POST_ROOT50_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9930	Pre Divider Register (CCM_PRE50)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9934	Pre Divider Register (CCM_PRE_ROOT50_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9938	Pre Divider Register (CCM_PRE_ROOT50_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_993C	Pre Divider Register (CCM_PRE_ROOT50_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9970	Access Control Register (CCM_ACCESS_CTRL50)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9974	Access Control Register (CCM_ACCESS_CTRL_ROOT50_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9978	Access Control Register (CCM_ACCESS_CTRL_ROOT50_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_997C	Access Control Register (CCM_ACCESS_CTRL_ROOT50_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9980	Target Register (CCM_TARGET_ROOT51)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9984	Target Register (CCM_TARGET_ROOT51_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9988	Target Register (CCM_TARGET_ROOT51_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_998C	Target Register (CCM_TARGET_ROOT51_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9990	Miscellaneous Register (CCM_MISC51)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9994	Miscellaneous Register (CCM_MISC_ROOT51_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9998	Miscellaneous Register (CCM_MISC_ROOT51_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_999C	Miscellaneous Register (CCM_MISC_ROOT51_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_99A0	Post Divider Register (CCM_POST51)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_99A4	Post Divider Register (CCM_POST_ROOT51_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_99A8	Post Divider Register (CCM_POST_ROOT51_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_99AC	Post Divider Register (CCM_POST_ROOT51_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_99B0	Pre Divider Register (CCM_PRE51)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_99B4	Pre Divider Register (CCM_PRE_ROOT51_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_99B8	Pre Divider Register (CCM_PRE_ROOT51_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_99BC	Pre Divider Register (CCM_PRE_ROOT51_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_99F0	Access Control Register (CCM_ACCESS_CTRL51)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_99F4	Access Control Register (CCM_ACCESS_CTRL_ROOT51_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_99F8	Access Control Register (CCM_ACCESS_CTRL_ROOT51_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_99FC	Access Control Register (CCM_ACCESS_CTRL_ROOT51_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9A00	Target Register (CCM_TARGET_ROOT52)	32	R/W	0000_0000h	5.2.8.10/ 699

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9A04	Target Register (CCM_TARGET_ROOT52_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9A08	Target Register (CCM_TARGET_ROOT52_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_9A0C	Target Register (CCM_TARGET_ROOT52_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9A10	Miscellaneous Register (CCM_MISC52)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9A14	Miscellaneous Register (CCM_MISC_ROOT52_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9A18	Miscellaneous Register (CCM_MISC_ROOT52_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_9A1C	Miscellaneous Register (CCM_MISC_ROOT52_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9A20	Post Divider Register (CCM_POST52)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9A24	Post Divider Register (CCM_POST_ROOT52_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9A28	Post Divider Register (CCM_POST_ROOT52_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_9A2C	Post Divider Register (CCM_POST_ROOT52_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9A30	Pre Divider Register (CCM_PRE52)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9A34	Pre Divider Register (CCM_PRE_ROOT52_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9A38	Pre Divider Register (CCM_PRE_ROOT52_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_9A3C	Pre Divider Register (CCM_PRE_ROOT52_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9A70	Access Control Register (CCM_ACCESS_CTRL52)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9A74	Access Control Register (CCM_ACCESS_CTRL_ROOT52_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9A78	Access Control Register (CCM_ACCESS_CTRL_ROOT52_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_9A7C	Access Control Register (CCM_ACCESS_CTRL_ROOT52_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9A80	Target Register (CCM_TARGET_ROOT53)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9A84	Target Register (CCM_TARGET_ROOT53_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9A88	Target Register (CCM_TARGET_ROOT53_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9A8C	Target Register (CCM_TARGET_ROOT53_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9A90	Miscellaneous Register (CCM_MISC53)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9A94	Miscellaneous Register (CCM_MISC_ROOT53_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9A98	Miscellaneous Register (CCM_MISC_ROOT53_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_9A9C	Miscellaneous Register (CCM_MISC_ROOT53_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9AA0	Post Divider Register (CCM_POST53)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9AA4	Post Divider Register (CCM_POST_ROOT53_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9AA8	Post Divider Register (CCM_POST_ROOT53_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_9AAC	Post Divider Register (CCM_POST_ROOT53_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9AB0	Pre Divider Register (CCM_PRE53)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9AB4	Pre Divider Register (CCM_PRE_ROOT53_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9AB8	Pre Divider Register (CCM_PRE_ROOT53_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_9ABC	Pre Divider Register (CCM_PRE_ROOT53_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9AF0	Access Control Register (CCM_ACCESS_CTRL53)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9AF4	Access Control Register (CCM_ACCESS_CTRL_ROOT53_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9AF8	Access Control Register (CCM_ACCESS_CTRL_ROOT53_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_9AFC	Access Control Register (CCM_ACCESS_CTRL_ROOT53_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9B00	Target Register (CCM_TARGET_ROOT54)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9B04	Target Register (CCM_TARGET_ROOT54_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9B08	Target Register (CCM_TARGET_ROOT54_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_9B0C	Target Register (CCM_TARGET_ROOT54_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9B10	Miscellaneous Register (CCM_MISC54)	32	R/W	0000_0000h	5.2.8.14/ 707

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9B14	Miscellaneous Register (CCM_MISC_ROOT54_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9B18	Miscellaneous Register (CCM_MISC_ROOT54_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_9B1C	Miscellaneous Register (CCM_MISC_ROOT54_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9B20	Post Divider Register (CCM_POST54)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9B24	Post Divider Register (CCM_POST_ROOT54_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9B28	Post Divider Register (CCM_POST_ROOT54_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_9B2C	Post Divider Register (CCM_POST_ROOT54_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9B30	Pre Divider Register (CCM_PRE54)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9B34	Pre Divider Register (CCM_PRE_ROOT54_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9B38	Pre Divider Register (CCM_PRE_ROOT54_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_9B3C	Pre Divider Register (CCM_PRE_ROOT54_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9B70	Access Control Register (CCM_ACCESS_CTRL54)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9B74	Access Control Register (CCM_ACCESS_CTRL_ROOT54_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9B78	Access Control Register (CCM_ACCESS_CTRL_ROOT54_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_9B7C	Access Control Register (CCM_ACCESS_CTRL_ROOT54_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9B80	Target Register (CCM_TARGET_ROOT55)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9B84	Target Register (CCM_TARGET_ROOT55_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9B88	Target Register (CCM_TARGET_ROOT55_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_9B8C	Target Register (CCM_TARGET_ROOT55_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9B90	Miscellaneous Register (CCM_MISC55)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9B94	Miscellaneous Register (CCM_MISC_ROOT55_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9B98	Miscellaneous Register (CCM_MISC_ROOT55_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9B9C	Miscellaneous Register (CCM_MISC_ROOT55_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9BA0	Post Divider Register (CCM_POST55)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9BA4	Post Divider Register (CCM_POST_ROOT55_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9BA8	Post Divider Register (CCM_POST_ROOT55_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_9BAC	Post Divider Register (CCM_POST_ROOT55_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9BB0	Pre Divider Register (CCM_PRE55)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9BB4	Pre Divider Register (CCM_PRE_ROOT55_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9BB8	Pre Divider Register (CCM_PRE_ROOT55_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_9BBC	Pre Divider Register (CCM_PRE_ROOT55_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9BF0	Access Control Register (CCM_ACCESS_CTRL55)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9BF4	Access Control Register (CCM_ACCESS_CTRL_ROOT55_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9BF8	Access Control Register (CCM_ACCESS_CTRL_ROOT55_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_9BFC	Access Control Register (CCM_ACCESS_CTRL_ROOT55_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9C00	Target Register (CCM_TARGET_ROOT56)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9C04	Target Register (CCM_TARGET_ROOT56_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9C08	Target Register (CCM_TARGET_ROOT56_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_9C0C	Target Register (CCM_TARGET_ROOT56_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9C10	Miscellaneous Register (CCM_MISC56)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9C14	Miscellaneous Register (CCM_MISC_ROOT56_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9C18	Miscellaneous Register (CCM_MISC_ROOT56_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_9C1C	Miscellaneous Register (CCM_MISC_ROOT56_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9C20	Post Divider Register (CCM_POST56)	32	R/W	0000_0000h	5.2.8.18/ 711

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9C24	Post Divider Register (CCM_POST_ROOT56_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9C28	Post Divider Register (CCM_POST_ROOT56_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_9C2C	Post Divider Register (CCM_POST_ROOT56_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9C30	Pre Divider Register (CCM_PRE56)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9C34	Pre Divider Register (CCM_PRE_ROOT56_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9C38	Pre Divider Register (CCM_PRE_ROOT56_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_9C3C	Pre Divider Register (CCM_PRE_ROOT56_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9C70	Access Control Register (CCM_ACCESS_CTRL56)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9C74	Access Control Register (CCM_ACCESS_CTRL_ROOT56_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9C78	Access Control Register (CCM_ACCESS_CTRL_ROOT56_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_9C7C	Access Control Register (CCM_ACCESS_CTRL_ROOT56_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9C80	Target Register (CCM_TARGET_ROOT57)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9C84	Target Register (CCM_TARGET_ROOT57_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9C88	Target Register (CCM_TARGET_ROOT57_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_9C8C	Target Register (CCM_TARGET_ROOT57_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9C90	Miscellaneous Register (CCM_MISC57)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9C94	Miscellaneous Register (CCM_MISC_ROOT57_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9C98	Miscellaneous Register (CCM_MISC_ROOT57_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_9C9C	Miscellaneous Register (CCM_MISC_ROOT57_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9CA0	Post Divider Register (CCM_POST57)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9CA4	Post Divider Register (CCM_POST_ROOT57_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9CA8	Post Divider Register (CCM_POST_ROOT57_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9CAC	Post Divider Register (CCM_POST_ROOT57_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9CB0	Pre Divider Register (CCM_PRE57)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9CB4	Pre Divider Register (CCM_PRE_ROOT57_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9CB8	Pre Divider Register (CCM_PRE_ROOT57_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_9CBC	Pre Divider Register (CCM_PRE_ROOT57_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9CF0	Access Control Register (CCM_ACCESS_CTRL57)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9CF4	Access Control Register (CCM_ACCESS_CTRL_ROOT57_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9CF8	Access Control Register (CCM_ACCESS_CTRL_ROOT57_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_9CFC	Access Control Register (CCM_ACCESS_CTRL_ROOT57_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9D00	Target Register (CCM_TARGET_ROOT58)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9D04	Target Register (CCM_TARGET_ROOT58_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9D08	Target Register (CCM_TARGET_ROOT58_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_9D0C	Target Register (CCM_TARGET_ROOT58_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9D10	Miscellaneous Register (CCM_MISC58)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9D14	Miscellaneous Register (CCM_MISC_ROOT58_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9D18	Miscellaneous Register (CCM_MISC_ROOT58_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_9D1C	Miscellaneous Register (CCM_MISC_ROOT58_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9D20	Post Divider Register (CCM_POST58)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9D24	Post Divider Register (CCM_POST_ROOT58_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9D28	Post Divider Register (CCM_POST_ROOT58_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_9D2C	Post Divider Register (CCM_POST_ROOT58_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9D30	Pre Divider Register (CCM_PRE58)	32	R/W	0000_0000h	5.2.8.22/ 723

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9D34	Pre Divider Register (CCM_PRE_ROOT58_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9D38	Pre Divider Register (CCM_PRE_ROOT58_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_9D3C	Pre Divider Register (CCM_PRE_ROOT58_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9D70	Access Control Register (CCM_ACCESS_CTRL58)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9D74	Access Control Register (CCM_ACCESS_CTRL_ROOT58_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9D78	Access Control Register (CCM_ACCESS_CTRL_ROOT58_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_9D7C	Access Control Register (CCM_ACCESS_CTRL_ROOT58_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9D80	Target Register (CCM_TARGET_ROOT59)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9D84	Target Register (CCM_TARGET_ROOT59_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9D88	Target Register (CCM_TARGET_ROOT59_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_9D8C	Target Register (CCM_TARGET_ROOT59_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9D90	Miscellaneous Register (CCM_MISC59)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9D94	Miscellaneous Register (CCM_MISC_ROOT59_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9D98	Miscellaneous Register (CCM_MISC_ROOT59_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_9D9C	Miscellaneous Register (CCM_MISC_ROOT59_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9DA0	Post Divider Register (CCM_POST59)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9DA4	Post Divider Register (CCM_POST_ROOT59_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9DA8	Post Divider Register (CCM_POST_ROOT59_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_9DAC	Post Divider Register (CCM_POST_ROOT59_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9DB0	Pre Divider Register (CCM_PRE59)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9DB4	Pre Divider Register (CCM_PRE_ROOT59_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9DB8	Pre Divider Register (CCM_PRE_ROOT59_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9DBC	Pre Divider Register (CCM_PRE_ROOT59_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9DF0	Access Control Register (CCM_ACCESS_CTRL59)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9DF4	Access Control Register (CCM_ACCESS_CTRL_ROOT59_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9DF8	Access Control Register (CCM_ACCESS_CTRL_ROOT59_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_9DFC	Access Control Register (CCM_ACCESS_CTRL_ROOT59_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9E00	Target Register (CCM_TARGET_ROOT60)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9E04	Target Register (CCM_TARGET_ROOT60_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9E08	Target Register (CCM_TARGET_ROOT60_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_9E0C	Target Register (CCM_TARGET_ROOT60_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9E10	Miscellaneous Register (CCM_MISC60)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9E14	Miscellaneous Register (CCM_MISC_ROOT60_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9E18	Miscellaneous Register (CCM_MISC_ROOT60_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_9E1C	Miscellaneous Register (CCM_MISC_ROOT60_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9E20	Post Divider Register (CCM_POST60)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9E24	Post Divider Register (CCM_POST_ROOT60_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9E28	Post Divider Register (CCM_POST_ROOT60_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_9E2C	Post Divider Register (CCM_POST_ROOT60_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9E30	Pre Divider Register (CCM_PRE60)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9E34	Pre Divider Register (CCM_PRE_ROOT60_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9E38	Pre Divider Register (CCM_PRE_ROOT60_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_9E3C	Pre Divider Register (CCM_PRE_ROOT60_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9E70	Access Control Register (CCM_ACCESS_CTRL60)	32	R/W	0000_0000h	5.2.8.26/ 735

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9E74	Access Control Register (CCM_ACCESS_CTRL_ROOT60_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9E78	Access Control Register (CCM_ACCESS_CTRL_ROOT60_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_9E7C	Access Control Register (CCM_ACCESS_CTRL_ROOT60_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9E80	Target Register (CCM_TARGET_ROOT61)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9E84	Target Register (CCM_TARGET_ROOT61_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9E88	Target Register (CCM_TARGET_ROOT61_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_9E8C	Target Register (CCM_TARGET_ROOT61_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9E90	Miscellaneous Register (CCM_MISC61)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9E94	Miscellaneous Register (CCM_MISC_ROOT61_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9E98	Miscellaneous Register (CCM_MISC_ROOT61_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_9E9C	Miscellaneous Register (CCM_MISC_ROOT61_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9EA0	Post Divider Register (CCM_POST61)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9EA4	Post Divider Register (CCM_POST_ROOT61_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9EA8	Post Divider Register (CCM_POST_ROOT61_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_9EAC	Post Divider Register (CCM_POST_ROOT61_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9EB0	Pre Divider Register (CCM_PRE61)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9EB4	Pre Divider Register (CCM_PRE_ROOT61_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9EB8	Pre Divider Register (CCM_PRE_ROOT61_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_9EBC	Pre Divider Register (CCM_PRE_ROOT61_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9EF0	Access Control Register (CCM_ACCESS_CTRL61)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9EF4	Access Control Register (CCM_ACCESS_CTRL_ROOT61_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9EF8	Access Control Register (CCM_ACCESS_CTRL_ROOT61_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9EFC	Access Control Register (CCM_ACCESS_CTRL_ROOT61_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9F00	Target Register (CCM_TARGET_ROOT62)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_9F04	Target Register (CCM_TARGET_ROOT62_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9F08	Target Register (CCM_TARGET_ROOT62_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_9F0C	Target Register (CCM_TARGET_ROOT62_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9F10	Miscellaneous Register (CCM_MISC62)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9F14	Miscellaneous Register (CCM_MISC_ROOT62_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9F18	Miscellaneous Register (CCM_MISC_ROOT62_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_9F1C	Miscellaneous Register (CCM_MISC_ROOT62_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9F20	Post Divider Register (CCM_POST62)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9F24	Post Divider Register (CCM_POST_ROOT62_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9F28	Post Divider Register (CCM_POST_ROOT62_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_9F2C	Post Divider Register (CCM_POST_ROOT62_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9F30	Pre Divider Register (CCM_PRE62)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9F34	Pre Divider Register (CCM_PRE_ROOT62_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9F38	Pre Divider Register (CCM_PRE_ROOT62_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_9F3C	Pre Divider Register (CCM_PRE_ROOT62_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9F70	Access Control Register (CCM_ACCESS_CTRL62)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9F74	Access Control Register (CCM_ACCESS_CTRL_ROOT62_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9F78	Access Control Register (CCM_ACCESS_CTRL_ROOT62_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_9F7C	Access Control Register (CCM_ACCESS_CTRL_ROOT62_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_9F80	Target Register (CCM_TARGET_ROOT63)	32	R/W	0000_0000h	5.2.8.10/ 699

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_9F84	Target Register (CCM_TARGET_ROOT63_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_9F88	Target Register (CCM_TARGET_ROOT63_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_9F8C	Target Register (CCM_TARGET_ROOT63_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_9F90	Miscellaneous Register (CCM_MISC63)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_9F94	Miscellaneous Register (CCM_MISC_ROOT63_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_9F98	Miscellaneous Register (CCM_MISC_ROOT63_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_9F9C	Miscellaneous Register (CCM_MISC_ROOT63_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_9FA0	Post Divider Register (CCM_POST63)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_9FA4	Post Divider Register (CCM_POST_ROOT63_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_9FA8	Post Divider Register (CCM_POST_ROOT63_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_9FAC	Post Divider Register (CCM_POST_ROOT63_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_9FB0	Pre Divider Register (CCM_PRE63)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_9FB4	Pre Divider Register (CCM_PRE_ROOT63_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_9FB8	Pre Divider Register (CCM_PRE_ROOT63_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_9FBC	Pre Divider Register (CCM_PRE_ROOT63_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_9FF0	Access Control Register (CCM_ACCESS_CTRL63)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_9FF4	Access Control Register (CCM_ACCESS_CTRL_ROOT63_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_9FF8	Access Control Register (CCM_ACCESS_CTRL_ROOT63_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_9FFC	Access Control Register (CCM_ACCESS_CTRL_ROOT63_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A000	Target Register (CCM_TARGET_ROOT64)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A004	Target Register (CCM_TARGET_ROOT64_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A008	Target Register (CCM_TARGET_ROOT64_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A00C	Target Register (CCM_TARGET_ROOT64_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A010	Miscellaneous Register (CCM_MISC64)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A014	Miscellaneous Register (CCM_MISC_ROOT64_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A018	Miscellaneous Register (CCM_MISC_ROOT64_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A01C	Miscellaneous Register (CCM_MISC_ROOT64_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A020	Post Divider Register (CCM_POST64)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A024	Post Divider Register (CCM_POST_ROOT64_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A028	Post Divider Register (CCM_POST_ROOT64_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A02C	Post Divider Register (CCM_POST_ROOT64_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A030	Pre Divider Register (CCM_PRE64)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A034	Pre Divider Register (CCM_PRE_ROOT64_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A038	Pre Divider Register (CCM_PRE_ROOT64_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A03C	Pre Divider Register (CCM_PRE_ROOT64_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A070	Access Control Register (CCM_ACCESS_CTRL64)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A074	Access Control Register (CCM_ACCESS_CTRL_ROOT64_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A078	Access Control Register (CCM_ACCESS_CTRL_ROOT64_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A07C	Access Control Register (CCM_ACCESS_CTRL_ROOT64_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A080	Target Register (CCM_TARGET_ROOT65)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A084	Target Register (CCM_TARGET_ROOT65_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A088	Target Register (CCM_TARGET_ROOT65_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A08C	Target Register (CCM_TARGET_ROOT65_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A090	Miscellaneous Register (CCM_MISC65)	32	R/W	0000_0000h	5.2.8.14/ 707

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A094	Miscellaneous Register (CCM_MISC_ROOT65_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A098	Miscellaneous Register (CCM_MISC_ROOT65_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A09C	Miscellaneous Register (CCM_MISC_ROOT65_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A0A0	Post Divider Register (CCM_POST65)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A0A4	Post Divider Register (CCM_POST_ROOT65_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A0A8	Post Divider Register (CCM_POST_ROOT65_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A0AC	Post Divider Register (CCM_POST_ROOT65_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A0B0	Pre Divider Register (CCM_PRE65)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A0B4	Pre Divider Register (CCM_PRE_ROOT65_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A0B8	Pre Divider Register (CCM_PRE_ROOT65_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A0BC	Pre Divider Register (CCM_PRE_ROOT65_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A0F0	Access Control Register (CCM_ACCESS_CTRL65)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A0F4	Access Control Register (CCM_ACCESS_CTRL_ROOT65_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A0F8	Access Control Register (CCM_ACCESS_CTRL_ROOT65_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A0FC	Access Control Register (CCM_ACCESS_CTRL_ROOT65_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A100	Target Register (CCM_TARGET_ROOT66)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A104	Target Register (CCM_TARGET_ROOT66_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A108	Target Register (CCM_TARGET_ROOT66_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A10C	Target Register (CCM_TARGET_ROOT66_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A110	Miscellaneous Register (CCM_MISC66)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A114	Miscellaneous Register (CCM_MISC_ROOT66_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A118	Miscellaneous Register (CCM_MISC_ROOT66_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A11C	Miscellaneous Register (CCM_MISC_ROOT66_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A120	Post Divider Register (CCM_POST66)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A124	Post Divider Register (CCM_POST_ROOT66_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A128	Post Divider Register (CCM_POST_ROOT66_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A12C	Post Divider Register (CCM_POST_ROOT66_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A130	Pre Divider Register (CCM_PRE66)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A134	Pre Divider Register (CCM_PRE_ROOT66_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A138	Pre Divider Register (CCM_PRE_ROOT66_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A13C	Pre Divider Register (CCM_PRE_ROOT66_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A170	Access Control Register (CCM_ACCESS_CTRL66)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A174	Access Control Register (CCM_ACCESS_CTRL_ROOT66_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A178	Access Control Register (CCM_ACCESS_CTRL_ROOT66_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A17C	Access Control Register (CCM_ACCESS_CTRL_ROOT66_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A180	Target Register (CCM_TARGET_ROOT67)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A184	Target Register (CCM_TARGET_ROOT67_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A188	Target Register (CCM_TARGET_ROOT67_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A18C	Target Register (CCM_TARGET_ROOT67_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A190	Miscellaneous Register (CCM_MISC67)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A194	Miscellaneous Register (CCM_MISC_ROOT67_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A198	Miscellaneous Register (CCM_MISC_ROOT67_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A19C	Miscellaneous Register (CCM_MISC_ROOT67_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A1A0	Post Divider Register (CCM_POST67)	32	R/W	0000_0000h	5.2.8.18/ 711

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A1A4	Post Divider Register (CCM_POST_ROOT67_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A1A8	Post Divider Register (CCM_POST_ROOT67_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A1AC	Post Divider Register (CCM_POST_ROOT67_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A1B0	Pre Divider Register (CCM_PRE67)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A1B4	Pre Divider Register (CCM_PRE_ROOT67_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A1B8	Pre Divider Register (CCM_PRE_ROOT67_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A1BC	Pre Divider Register (CCM_PRE_ROOT67_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A1F0	Access Control Register (CCM_ACCESS_CTRL67)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A1F4	Access Control Register (CCM_ACCESS_CTRL_ROOT67_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A1F8	Access Control Register (CCM_ACCESS_CTRL_ROOT67_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A1FC	Access Control Register (CCM_ACCESS_CTRL_ROOT67_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A200	Target Register (CCM_TARGET_ROOT68)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A204	Target Register (CCM_TARGET_ROOT68_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A208	Target Register (CCM_TARGET_ROOT68_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A20C	Target Register (CCM_TARGET_ROOT68_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A210	Miscellaneous Register (CCM_MISC68)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A214	Miscellaneous Register (CCM_MISC_ROOT68_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A218	Miscellaneous Register (CCM_MISC_ROOT68_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A21C	Miscellaneous Register (CCM_MISC_ROOT68_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A220	Post Divider Register (CCM_POST68)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A224	Post Divider Register (CCM_POST_ROOT68_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A228	Post Divider Register (CCM_POST_ROOT68_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A22C	Post Divider Register (CCM_POST_ROOT68_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A230	Pre Divider Register (CCM_PRE68)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A234	Pre Divider Register (CCM_PRE_ROOT68_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A238	Pre Divider Register (CCM_PRE_ROOT68_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A23C	Pre Divider Register (CCM_PRE_ROOT68_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A270	Access Control Register (CCM_ACCESS_CTRL68)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A274	Access Control Register (CCM_ACCESS_CTRL_ROOT68_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A278	Access Control Register (CCM_ACCESS_CTRL_ROOT68_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A27C	Access Control Register (CCM_ACCESS_CTRL_ROOT68_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A280	Target Register (CCM_TARGET_ROOT69)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A284	Target Register (CCM_TARGET_ROOT69_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A288	Target Register (CCM_TARGET_ROOT69_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A28C	Target Register (CCM_TARGET_ROOT69_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A290	Miscellaneous Register (CCM_MISC69)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A294	Miscellaneous Register (CCM_MISC_ROOT69_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A298	Miscellaneous Register (CCM_MISC_ROOT69_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A29C	Miscellaneous Register (CCM_MISC_ROOT69_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A2A0	Post Divider Register (CCM_POST69)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A2A4	Post Divider Register (CCM_POST_ROOT69_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A2A8	Post Divider Register (CCM_POST_ROOT69_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A2AC	Post Divider Register (CCM_POST_ROOT69_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A2B0	Pre Divider Register (CCM_PRE69)	32	R/W	0000_0000h	5.2.8.22/ 723

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A2B4	Pre Divider Register (CCM_PRE_ROOT69_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A2B8	Pre Divider Register (CCM_PRE_ROOT69_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A2BC	Pre Divider Register (CCM_PRE_ROOT69_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A2F0	Access Control Register (CCM_ACCESS_CTRL69)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A2F4	Access Control Register (CCM_ACCESS_CTRL_ROOT69_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A2F8	Access Control Register (CCM_ACCESS_CTRL_ROOT69_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A2FC	Access Control Register (CCM_ACCESS_CTRL_ROOT69_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A300	Target Register (CCM_TARGET_ROOT70)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A304	Target Register (CCM_TARGET_ROOT70_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A308	Target Register (CCM_TARGET_ROOT70_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A30C	Target Register (CCM_TARGET_ROOT70_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A310	Miscellaneous Register (CCM_MISC70)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A314	Miscellaneous Register (CCM_MISC_ROOT70_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A318	Miscellaneous Register (CCM_MISC_ROOT70_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A31C	Miscellaneous Register (CCM_MISC_ROOT70_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A320	Post Divider Register (CCM_POST70)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A324	Post Divider Register (CCM_POST_ROOT70_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A328	Post Divider Register (CCM_POST_ROOT70_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A32C	Post Divider Register (CCM_POST_ROOT70_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A330	Pre Divider Register (CCM_PRE70)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A334	Pre Divider Register (CCM_PRE_ROOT70_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A338	Pre Divider Register (CCM_PRE_ROOT70_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A33C	Pre Divider Register (CCM_PRE_ROOT70_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A370	Access Control Register (CCM_ACCESS_CTRL70)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A374	Access Control Register (CCM_ACCESS_CTRL_ROOT70_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A378	Access Control Register (CCM_ACCESS_CTRL_ROOT70_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A37C	Access Control Register (CCM_ACCESS_CTRL_ROOT70_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A380	Target Register (CCM_TARGET_ROOT71)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A384	Target Register (CCM_TARGET_ROOT71_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A388	Target Register (CCM_TARGET_ROOT71_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A38C	Target Register (CCM_TARGET_ROOT71_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A390	Miscellaneous Register (CCM_MISC71)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A394	Miscellaneous Register (CCM_MISC_ROOT71_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A398	Miscellaneous Register (CCM_MISC_ROOT71_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A39C	Miscellaneous Register (CCM_MISC_ROOT71_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A3A0	Post Divider Register (CCM_POST71)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A3A4	Post Divider Register (CCM_POST_ROOT71_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A3A8	Post Divider Register (CCM_POST_ROOT71_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A3AC	Post Divider Register (CCM_POST_ROOT71_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A3B0	Pre Divider Register (CCM_PRE71)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A3B4	Pre Divider Register (CCM_PRE_ROOT71_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A3B8	Pre Divider Register (CCM_PRE_ROOT71_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A3BC	Pre Divider Register (CCM_PRE_ROOT71_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A3F0	Access Control Register (CCM_ACCESS_CTRL71)	32	R/W	0000_0000h	5.2.8.26/ 735

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A3F4	Access Control Register (CCM_ACCESS_CTRL_ROOT71_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A3F8	Access Control Register (CCM_ACCESS_CTRL_ROOT71_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A3FC	Access Control Register (CCM_ACCESS_CTRL_ROOT71_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A400	Target Register (CCM_TARGET_ROOT72)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A404	Target Register (CCM_TARGET_ROOT72_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A408	Target Register (CCM_TARGET_ROOT72_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A40C	Target Register (CCM_TARGET_ROOT72_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A410	Miscellaneous Register (CCM_MISC72)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A414	Miscellaneous Register (CCM_MISC_ROOT72_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A418	Miscellaneous Register (CCM_MISC_ROOT72_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A41C	Miscellaneous Register (CCM_MISC_ROOT72_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A420	Post Divider Register (CCM_POST72)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A424	Post Divider Register (CCM_POST_ROOT72_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A428	Post Divider Register (CCM_POST_ROOT72_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A42C	Post Divider Register (CCM_POST_ROOT72_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A430	Pre Divider Register (CCM_PRE72)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A434	Pre Divider Register (CCM_PRE_ROOT72_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A438	Pre Divider Register (CCM_PRE_ROOT72_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A43C	Pre Divider Register (CCM_PRE_ROOT72_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A470	Access Control Register (CCM_ACCESS_CTRL72)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A474	Access Control Register (CCM_ACCESS_CTRL_ROOT72_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A478	Access Control Register (CCM_ACCESS_CTRL_ROOT72_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A47C	Access Control Register (CCM_ACCESS_CTRL_ROOT72_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A480	Target Register (CCM_TARGET_ROOT73)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A484	Target Register (CCM_TARGET_ROOT73_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A488	Target Register (CCM_TARGET_ROOT73_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A48C	Target Register (CCM_TARGET_ROOT73_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A490	Miscellaneous Register (CCM_MISC73)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A494	Miscellaneous Register (CCM_MISC_ROOT73_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A498	Miscellaneous Register (CCM_MISC_ROOT73_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A49C	Miscellaneous Register (CCM_MISC_ROOT73_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A4A0	Post Divider Register (CCM_POST73)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A4A4	Post Divider Register (CCM_POST_ROOT73_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A4A8	Post Divider Register (CCM_POST_ROOT73_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A4AC	Post Divider Register (CCM_POST_ROOT73_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A4B0	Pre Divider Register (CCM_PRE73)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A4B4	Pre Divider Register (CCM_PRE_ROOT73_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A4B8	Pre Divider Register (CCM_PRE_ROOT73_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A4BC	Pre Divider Register (CCM_PRE_ROOT73_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A4F0	Access Control Register (CCM_ACCESS_CTRL73)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A4F4	Access Control Register (CCM_ACCESS_CTRL_ROOT73_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A4F8	Access Control Register (CCM_ACCESS_CTRL_ROOT73_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A4FC	Access Control Register (CCM_ACCESS_CTRL_ROOT73_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A500	Target Register (CCM_TARGET_ROOT74)	32	R/W	0000_0000h	5.2.8.10/ 699

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A504	Target Register (CCM_TARGET_ROOT74_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A508	Target Register (CCM_TARGET_ROOT74_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A50C	Target Register (CCM_TARGET_ROOT74_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A510	Miscellaneous Register (CCM_MISC74)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A514	Miscellaneous Register (CCM_MISC_ROOT74_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A518	Miscellaneous Register (CCM_MISC_ROOT74_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A51C	Miscellaneous Register (CCM_MISC_ROOT74_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A520	Post Divider Register (CCM_POST74)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A524	Post Divider Register (CCM_POST_ROOT74_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A528	Post Divider Register (CCM_POST_ROOT74_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A52C	Post Divider Register (CCM_POST_ROOT74_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A530	Pre Divider Register (CCM_PRE74)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A534	Pre Divider Register (CCM_PRE_ROOT74_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A538	Pre Divider Register (CCM_PRE_ROOT74_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A53C	Pre Divider Register (CCM_PRE_ROOT74_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A570	Access Control Register (CCM_ACCESS_CTRL74)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A574	Access Control Register (CCM_ACCESS_CTRL_ROOT74_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A578	Access Control Register (CCM_ACCESS_CTRL_ROOT74_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A57C	Access Control Register (CCM_ACCESS_CTRL_ROOT74_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A580	Target Register (CCM_TARGET_ROOT75)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A584	Target Register (CCM_TARGET_ROOT75_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A588	Target Register (CCM_TARGET_ROOT75_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A58C	Target Register (CCM_TARGET_ROOT75_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A590	Miscellaneous Register (CCM_MISC75)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A594	Miscellaneous Register (CCM_MISC_ROOT75_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A598	Miscellaneous Register (CCM_MISC_ROOT75_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A59C	Miscellaneous Register (CCM_MISC_ROOT75_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A5A0	Post Divider Register (CCM_POST75)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A5A4	Post Divider Register (CCM_POST_ROOT75_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A5A8	Post Divider Register (CCM_POST_ROOT75_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A5AC	Post Divider Register (CCM_POST_ROOT75_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A5B0	Pre Divider Register (CCM_PRE75)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A5B4	Pre Divider Register (CCM_PRE_ROOT75_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A5B8	Pre Divider Register (CCM_PRE_ROOT75_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A5BC	Pre Divider Register (CCM_PRE_ROOT75_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A5F0	Access Control Register (CCM_ACCESS_CTRL75)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A5F4	Access Control Register (CCM_ACCESS_CTRL_ROOT75_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A5F8	Access Control Register (CCM_ACCESS_CTRL_ROOT75_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A5FC	Access Control Register (CCM_ACCESS_CTRL_ROOT75_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A600	Target Register (CCM_TARGET_ROOT76)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A604	Target Register (CCM_TARGET_ROOT76_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A608	Target Register (CCM_TARGET_ROOT76_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A60C	Target Register (CCM_TARGET_ROOT76_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A610	Miscellaneous Register (CCM_MISC76)	32	R/W	0000_0000h	5.2.8.14/ 707

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A614	Miscellaneous Register (CCM_MISC_ROOT76_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A618	Miscellaneous Register (CCM_MISC_ROOT76_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A61C	Miscellaneous Register (CCM_MISC_ROOT76_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A620	Post Divider Register (CCM_POST76)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A624	Post Divider Register (CCM_POST_ROOT76_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A628	Post Divider Register (CCM_POST_ROOT76_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A62C	Post Divider Register (CCM_POST_ROOT76_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A630	Pre Divider Register (CCM_PRE76)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A634	Pre Divider Register (CCM_PRE_ROOT76_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A638	Pre Divider Register (CCM_PRE_ROOT76_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A63C	Pre Divider Register (CCM_PRE_ROOT76_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A670	Access Control Register (CCM_ACCESS_CTRL76)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A674	Access Control Register (CCM_ACCESS_CTRL_ROOT76_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A678	Access Control Register (CCM_ACCESS_CTRL_ROOT76_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A67C	Access Control Register (CCM_ACCESS_CTRL_ROOT76_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A680	Target Register (CCM_TARGET_ROOT77)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A684	Target Register (CCM_TARGET_ROOT77_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A688	Target Register (CCM_TARGET_ROOT77_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A68C	Target Register (CCM_TARGET_ROOT77_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A690	Miscellaneous Register (CCM_MISC77)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A694	Miscellaneous Register (CCM_MISC_ROOT77_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A698	Miscellaneous Register (CCM_MISC_ROOT77_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A69C	Miscellaneous Register (CCM_MISC_ROOT77_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A6A0	Post Divider Register (CCM_POST77)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A6A4	Post Divider Register (CCM_POST_ROOT77_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A6A8	Post Divider Register (CCM_POST_ROOT77_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A6AC	Post Divider Register (CCM_POST_ROOT77_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A6B0	Pre Divider Register (CCM_PRE77)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A6B4	Pre Divider Register (CCM_PRE_ROOT77_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A6B8	Pre Divider Register (CCM_PRE_ROOT77_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A6BC	Pre Divider Register (CCM_PRE_ROOT77_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A6F0	Access Control Register (CCM_ACCESS_CTRL77)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A6F4	Access Control Register (CCM_ACCESS_CTRL_ROOT77_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A6F8	Access Control Register (CCM_ACCESS_CTRL_ROOT77_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A6FC	Access Control Register (CCM_ACCESS_CTRL_ROOT77_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A700	Target Register (CCM_TARGET_ROOT78)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A704	Target Register (CCM_TARGET_ROOT78_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A708	Target Register (CCM_TARGET_ROOT78_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A70C	Target Register (CCM_TARGET_ROOT78_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A710	Miscellaneous Register (CCM_MISC78)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A714	Miscellaneous Register (CCM_MISC_ROOT78_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A718	Miscellaneous Register (CCM_MISC_ROOT78_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A71C	Miscellaneous Register (CCM_MISC_ROOT78_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A720	Post Divider Register (CCM_POST78)	32	R/W	0000_0000h	5.2.8.18/ 711

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A724	Post Divider Register (CCM_POST_ROOT78_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A728	Post Divider Register (CCM_POST_ROOT78_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A72C	Post Divider Register (CCM_POST_ROOT78_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A730	Pre Divider Register (CCM_PRE78)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A734	Pre Divider Register (CCM_PRE_ROOT78_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A738	Pre Divider Register (CCM_PRE_ROOT78_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A73C	Pre Divider Register (CCM_PRE_ROOT78_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A770	Access Control Register (CCM_ACCESS_CTRL78)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A774	Access Control Register (CCM_ACCESS_CTRL_ROOT78_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A778	Access Control Register (CCM_ACCESS_CTRL_ROOT78_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A77C	Access Control Register (CCM_ACCESS_CTRL_ROOT78_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A780	Target Register (CCM_TARGET_ROOT79)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A784	Target Register (CCM_TARGET_ROOT79_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A788	Target Register (CCM_TARGET_ROOT79_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A78C	Target Register (CCM_TARGET_ROOT79_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A790	Miscellaneous Register (CCM_MISC79)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A794	Miscellaneous Register (CCM_MISC_ROOT79_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A798	Miscellaneous Register (CCM_MISC_ROOT79_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A79C	Miscellaneous Register (CCM_MISC_ROOT79_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A7A0	Post Divider Register (CCM_POST79)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A7A4	Post Divider Register (CCM_POST_ROOT79_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A7A8	Post Divider Register (CCM_POST_ROOT79_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A7AC	Post Divider Register (CCM_POST_ROOT79_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A7B0	Pre Divider Register (CCM_PRE79)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A7B4	Pre Divider Register (CCM_PRE_ROOT79_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A7B8	Pre Divider Register (CCM_PRE_ROOT79_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A7BC	Pre Divider Register (CCM_PRE_ROOT79_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A7F0	Access Control Register (CCM_ACCESS_CTRL79)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A7F4	Access Control Register (CCM_ACCESS_CTRL_ROOT79_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A7F8	Access Control Register (CCM_ACCESS_CTRL_ROOT79_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A7FC	Access Control Register (CCM_ACCESS_CTRL_ROOT79_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A800	Target Register (CCM_TARGET_ROOT80)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A804	Target Register (CCM_TARGET_ROOT80_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A808	Target Register (CCM_TARGET_ROOT80_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A80C	Target Register (CCM_TARGET_ROOT80_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A810	Miscellaneous Register (CCM_MISC80)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A814	Miscellaneous Register (CCM_MISC_ROOT80_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A818	Miscellaneous Register (CCM_MISC_ROOT80_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A81C	Miscellaneous Register (CCM_MISC_ROOT80_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A820	Post Divider Register (CCM_POST80)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A824	Post Divider Register (CCM_POST_ROOT80_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A828	Post Divider Register (CCM_POST_ROOT80_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A82C	Post Divider Register (CCM_POST_ROOT80_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A830	Pre Divider Register (CCM_PRE80)	32	R/W	0000_0000h	5.2.8.22/ 723

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A834	Pre Divider Register (CCM_PRE_ROOT80_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A838	Pre Divider Register (CCM_PRE_ROOT80_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A83C	Pre Divider Register (CCM_PRE_ROOT80_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A870	Access Control Register (CCM_ACCESS_CTRL80)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A874	Access Control Register (CCM_ACCESS_CTRL_ROOT80_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A878	Access Control Register (CCM_ACCESS_CTRL_ROOT80_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A87C	Access Control Register (CCM_ACCESS_CTRL_ROOT80_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A880	Target Register (CCM_TARGET_ROOT81)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A884	Target Register (CCM_TARGET_ROOT81_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A888	Target Register (CCM_TARGET_ROOT81_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A88C	Target Register (CCM_TARGET_ROOT81_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A890	Miscellaneous Register (CCM_MISC81)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A894	Miscellaneous Register (CCM_MISC_ROOT81_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A898	Miscellaneous Register (CCM_MISC_ROOT81_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A89C	Miscellaneous Register (CCM_MISC_ROOT81_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A8A0	Post Divider Register (CCM_POST81)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A8A4	Post Divider Register (CCM_POST_ROOT81_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A8A8	Post Divider Register (CCM_POST_ROOT81_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A8AC	Post Divider Register (CCM_POST_ROOT81_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A8B0	Pre Divider Register (CCM_PRE81)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A8B4	Pre Divider Register (CCM_PRE_ROOT81_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A8B8	Pre Divider Register (CCM_PRE_ROOT81_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A8BC	Pre Divider Register (CCM_PRE_ROOT81_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A8F0	Access Control Register (CCM_ACCESS_CTRL81)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A8F4	Access Control Register (CCM_ACCESS_CTRL_ROOT81_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A8F8	Access Control Register (CCM_ACCESS_CTRL_ROOT81_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A8FC	Access Control Register (CCM_ACCESS_CTRL_ROOT81_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A900	Target Register (CCM_TARGET_ROOT82)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A904	Target Register (CCM_TARGET_ROOT82_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A908	Target Register (CCM_TARGET_ROOT82_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A90C	Target Register (CCM_TARGET_ROOT82_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A910	Miscellaneous Register (CCM_MISC82)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A914	Miscellaneous Register (CCM_MISC_ROOT82_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A918	Miscellaneous Register (CCM_MISC_ROOT82_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A91C	Miscellaneous Register (CCM_MISC_ROOT82_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A920	Post Divider Register (CCM_POST82)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A924	Post Divider Register (CCM_POST_ROOT82_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A928	Post Divider Register (CCM_POST_ROOT82_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A92C	Post Divider Register (CCM_POST_ROOT82_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A930	Pre Divider Register (CCM_PRE82)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A934	Pre Divider Register (CCM_PRE_ROOT82_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A938	Pre Divider Register (CCM_PRE_ROOT82_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A93C	Pre Divider Register (CCM_PRE_ROOT82_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A970	Access Control Register (CCM_ACCESS_CTRL82)	32	R/W	0000_0000h	5.2.8.26/ 735

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A974	Access Control Register (CCM_ACCESS_CTRL_ROOT82_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A978	Access Control Register (CCM_ACCESS_CTRL_ROOT82_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_A97C	Access Control Register (CCM_ACCESS_CTRL_ROOT82_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_A980	Target Register (CCM_TARGET_ROOT83)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_A984	Target Register (CCM_TARGET_ROOT83_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_A988	Target Register (CCM_TARGET_ROOT83_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_A98C	Target Register (CCM_TARGET_ROOT83_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_A990	Miscellaneous Register (CCM_MISC83)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_A994	Miscellaneous Register (CCM_MISC_ROOT83_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_A998	Miscellaneous Register (CCM_MISC_ROOT83_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_A99C	Miscellaneous Register (CCM_MISC_ROOT83_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_A9A0	Post Divider Register (CCM_POST83)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_A9A4	Post Divider Register (CCM_POST_ROOT83_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_A9A8	Post Divider Register (CCM_POST_ROOT83_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_A9AC	Post Divider Register (CCM_POST_ROOT83_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_A9B0	Pre Divider Register (CCM_PRE83)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_A9B4	Pre Divider Register (CCM_PRE_ROOT83_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_A9B8	Pre Divider Register (CCM_PRE_ROOT83_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_A9BC	Pre Divider Register (CCM_PRE_ROOT83_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_A9F0	Access Control Register (CCM_ACCESS_CTRL83)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_A9F4	Access Control Register (CCM_ACCESS_CTRL_ROOT83_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_A9F8	Access Control Register (CCM_ACCESS_CTRL_ROOT83_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_A9FC	Access Control Register (CCM_ACCESS_CTRL_ROOT83_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_AA00	Target Register (CCM_TARGET_ROOT84)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_AA04	Target Register (CCM_TARGET_ROOT84_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_AA08	Target Register (CCM_TARGET_ROOT84_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_AA0C	Target Register (CCM_TARGET_ROOT84_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_AA10	Miscellaneous Register (CCM_MISC84)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_AA14	Miscellaneous Register (CCM_MISC_ROOT84_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_AA18	Miscellaneous Register (CCM_MISC_ROOT84_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_AA1C	Miscellaneous Register (CCM_MISC_ROOT84_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_AA20	Post Divider Register (CCM_POST84)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_AA24	Post Divider Register (CCM_POST_ROOT84_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_AA28	Post Divider Register (CCM_POST_ROOT84_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_AA2C	Post Divider Register (CCM_POST_ROOT84_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_AA30	Pre Divider Register (CCM_PRE84)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_AA34	Pre Divider Register (CCM_PRE_ROOT84_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_AA38	Pre Divider Register (CCM_PRE_ROOT84_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_AA3C	Pre Divider Register (CCM_PRE_ROOT84_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_AA70	Access Control Register (CCM_ACCESS_CTRL84)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_AA74	Access Control Register (CCM_ACCESS_CTRL_ROOT84_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_AA78	Access Control Register (CCM_ACCESS_CTRL_ROOT84_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_AA7C	Access Control Register (CCM_ACCESS_CTRL_ROOT84_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_AA80	Target Register (CCM_TARGET_ROOT85)	32	R/W	0000_0000h	5.2.8.10/ 699

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_AA84	Target Register (CCM_TARGET_ROOT85_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_AA88	Target Register (CCM_TARGET_ROOT85_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_AA8C	Target Register (CCM_TARGET_ROOT85_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_AA90	Miscellaneous Register (CCM_MISC85)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_AA94	Miscellaneous Register (CCM_MISC_ROOT85_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_AA98	Miscellaneous Register (CCM_MISC_ROOT85_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_AA9C	Miscellaneous Register (CCM_MISC_ROOT85_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_AAA0	Post Divider Register (CCM_POST85)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_AAA4	Post Divider Register (CCM_POST_ROOT85_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_AAA8	Post Divider Register (CCM_POST_ROOT85_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_AAAC	Post Divider Register (CCM_POST_ROOT85_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_AAB0	Pre Divider Register (CCM_PRE85)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_AAB4	Pre Divider Register (CCM_PRE_ROOT85_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_AAB8	Pre Divider Register (CCM_PRE_ROOT85_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_AABC	Pre Divider Register (CCM_PRE_ROOT85_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_AAF0	Access Control Register (CCM_ACCESS_CTRL85)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_AAF4	Access Control Register (CCM_ACCESS_CTRL_ROOT85_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_AAF8	Access Control Register (CCM_ACCESS_CTRL_ROOT85_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_AAFC	Access Control Register (CCM_ACCESS_CTRL_ROOT85_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_AB00	Target Register (CCM_TARGET_ROOT86)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_AB04	Target Register (CCM_TARGET_ROOT86_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_AB08	Target Register (CCM_TARGET_ROOT86_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_AB0C	Target Register (CCM_TARGET_ROOT86_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_AB10	Miscellaneous Register (CCM_MISC86)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_AB14	Miscellaneous Register (CCM_MISC_ROOT86_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_AB18	Miscellaneous Register (CCM_MISC_ROOT86_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_AB1C	Miscellaneous Register (CCM_MISC_ROOT86_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_AB20	Post Divider Register (CCM_POST86)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_AB24	Post Divider Register (CCM_POST_ROOT86_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_AB28	Post Divider Register (CCM_POST_ROOT86_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_AB2C	Post Divider Register (CCM_POST_ROOT86_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_AB30	Pre Divider Register (CCM_PRE86)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_AB34	Pre Divider Register (CCM_PRE_ROOT86_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_AB38	Pre Divider Register (CCM_PRE_ROOT86_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_AB3C	Pre Divider Register (CCM_PRE_ROOT86_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_AB70	Access Control Register (CCM_ACCESS_CTRL86)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_AB74	Access Control Register (CCM_ACCESS_CTRL_ROOT86_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_AB78	Access Control Register (CCM_ACCESS_CTRL_ROOT86_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_AB7C	Access Control Register (CCM_ACCESS_CTRL_ROOT86_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_AB80	Target Register (CCM_TARGET_ROOT87)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_AB84	Target Register (CCM_TARGET_ROOT87_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_AB88	Target Register (CCM_TARGET_ROOT87_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_AB8C	Target Register (CCM_TARGET_ROOT87_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_AB90	Miscellaneous Register (CCM_MISC87)	32	R/W	0000_0000h	5.2.8.14/ 707

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_AB94	Miscellaneous Register (CCM_MISC_ROOT87_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_AB98	Miscellaneous Register (CCM_MISC_ROOT87_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_AB9C	Miscellaneous Register (CCM_MISC_ROOT87_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_ABA0	Post Divider Register (CCM_POST87)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_ABA4	Post Divider Register (CCM_POST_ROOT87_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_ABA8	Post Divider Register (CCM_POST_ROOT87_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_ABAC	Post Divider Register (CCM_POST_ROOT87_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_ABB0	Pre Divider Register (CCM_PRE87)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_ABB4	Pre Divider Register (CCM_PRE_ROOT87_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_ABB8	Pre Divider Register (CCM_PRE_ROOT87_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_ABBC	Pre Divider Register (CCM_PRE_ROOT87_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_ABF0	Access Control Register (CCM_ACCESS_CTRL87)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_ABF4	Access Control Register (CCM_ACCESS_CTRL_ROOT87_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_ABF8	Access Control Register (CCM_ACCESS_CTRL_ROOT87_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_ABFC	Access Control Register (CCM_ACCESS_CTRL_ROOT87_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_AC00	Target Register (CCM_TARGET_ROOT88)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_AC04	Target Register (CCM_TARGET_ROOT88_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_AC08	Target Register (CCM_TARGET_ROOT88_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_AC0C	Target Register (CCM_TARGET_ROOT88_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_AC10	Miscellaneous Register (CCM_MISC88)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_AC14	Miscellaneous Register (CCM_MISC_ROOT88_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_AC18	Miscellaneous Register (CCM_MISC_ROOT88_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_AC1C	Miscellaneous Register (CCM_MISC_ROOT88_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_AC20	Post Divider Register (CCM_POST88)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_AC24	Post Divider Register (CCM_POST_ROOT88_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_AC28	Post Divider Register (CCM_POST_ROOT88_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_AC2C	Post Divider Register (CCM_POST_ROOT88_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_AC30	Pre Divider Register (CCM_PRE88)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_AC34	Pre Divider Register (CCM_PRE_ROOT88_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_AC38	Pre Divider Register (CCM_PRE_ROOT88_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_AC3C	Pre Divider Register (CCM_PRE_ROOT88_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_AC70	Access Control Register (CCM_ACCESS_CTRL88)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_AC74	Access Control Register (CCM_ACCESS_CTRL_ROOT88_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_AC78	Access Control Register (CCM_ACCESS_CTRL_ROOT88_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_AC7C	Access Control Register (CCM_ACCESS_CTRL_ROOT88_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_AC80	Target Register (CCM_TARGET_ROOT89)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_AC84	Target Register (CCM_TARGET_ROOT89_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_AC88	Target Register (CCM_TARGET_ROOT89_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_AC8C	Target Register (CCM_TARGET_ROOT89_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_AC90	Miscellaneous Register (CCM_MISC89)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_AC94	Miscellaneous Register (CCM_MISC_ROOT89_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_AC98	Miscellaneous Register (CCM_MISC_ROOT89_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_AC9C	Miscellaneous Register (CCM_MISC_ROOT89_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_ACA0	Post Divider Register (CCM_POST89)	32	R/W	0000_0000h	5.2.8.18/ 711

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_ACA4	Post Divider Register (CCM_POST_ROOT89_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_ACA8	Post Divider Register (CCM_POST_ROOT89_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_ACAC	Post Divider Register (CCM_POST_ROOT89_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_ACB0	Pre Divider Register (CCM_PRE89)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_ACB4	Pre Divider Register (CCM_PRE_ROOT89_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_ACB8	Pre Divider Register (CCM_PRE_ROOT89_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_ACBC	Pre Divider Register (CCM_PRE_ROOT89_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_ACF0	Access Control Register (CCM_ACCESS_CTRL89)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_ACF4	Access Control Register (CCM_ACCESS_CTRL_ROOT89_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_ACF8	Access Control Register (CCM_ACCESS_CTRL_ROOT89_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_ACFC	Access Control Register (CCM_ACCESS_CTRL_ROOT89_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_AD00	Target Register (CCM_TARGET_ROOT90)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_AD04	Target Register (CCM_TARGET_ROOT90_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_AD08	Target Register (CCM_TARGET_ROOT90_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_AD0C	Target Register (CCM_TARGET_ROOT90_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_AD10	Miscellaneous Register (CCM_MISC90)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_AD14	Miscellaneous Register (CCM_MISC_ROOT90_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_AD18	Miscellaneous Register (CCM_MISC_ROOT90_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_AD1C	Miscellaneous Register (CCM_MISC_ROOT90_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_AD20	Post Divider Register (CCM_POST90)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_AD24	Post Divider Register (CCM_POST_ROOT90_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_AD28	Post Divider Register (CCM_POST_ROOT90_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_AD2C	Post Divider Register (CCM_POST_ROOT90_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_AD30	Pre Divider Register (CCM_PRE90)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_AD34	Pre Divider Register (CCM_PRE_ROOT90_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_AD38	Pre Divider Register (CCM_PRE_ROOT90_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_AD3C	Pre Divider Register (CCM_PRE_ROOT90_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_AD70	Access Control Register (CCM_ACCESS_CTRL90)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_AD74	Access Control Register (CCM_ACCESS_CTRL_ROOT90_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_AD78	Access Control Register (CCM_ACCESS_CTRL_ROOT90_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_AD7C	Access Control Register (CCM_ACCESS_CTRL_ROOT90_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_AD80	Target Register (CCM_TARGET_ROOT91)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_AD84	Target Register (CCM_TARGET_ROOT91_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_AD88	Target Register (CCM_TARGET_ROOT91_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_AD8C	Target Register (CCM_TARGET_ROOT91_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_AD90	Miscellaneous Register (CCM_MISC91)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_AD94	Miscellaneous Register (CCM_MISC_ROOT91_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_AD98	Miscellaneous Register (CCM_MISC_ROOT91_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_AD9C	Miscellaneous Register (CCM_MISC_ROOT91_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_ADA0	Post Divider Register (CCM_POST91)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_ADA4	Post Divider Register (CCM_POST_ROOT91_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_ADA8	Post Divider Register (CCM_POST_ROOT91_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_ADAC	Post Divider Register (CCM_POST_ROOT91_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_ADB0	Pre Divider Register (CCM_PRE91)	32	R/W	0000_0000h	5.2.8.22/ 723

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_ADB4	Pre Divider Register (CCM_PRE_ROOT91_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_ADB8	Pre Divider Register (CCM_PRE_ROOT91_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_ADBC	Pre Divider Register (CCM_PRE_ROOT91_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_ADF0	Access Control Register (CCM_ACCESS_CTRL91)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_ADF4	Access Control Register (CCM_ACCESS_CTRL_ROOT91_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_ADF8	Access Control Register (CCM_ACCESS_CTRL_ROOT91_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_ADFC	Access Control Register (CCM_ACCESS_CTRL_ROOT91_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_AE00	Target Register (CCM_TARGET_ROOT92)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_AE04	Target Register (CCM_TARGET_ROOT92_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_AE08	Target Register (CCM_TARGET_ROOT92_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_AE0C	Target Register (CCM_TARGET_ROOT92_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_AE10	Miscellaneous Register (CCM_MISC92)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_AE14	Miscellaneous Register (CCM_MISC_ROOT92_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_AE18	Miscellaneous Register (CCM_MISC_ROOT92_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_AE1C	Miscellaneous Register (CCM_MISC_ROOT92_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_AE20	Post Divider Register (CCM_POST92)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_AE24	Post Divider Register (CCM_POST_ROOT92_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_AE28	Post Divider Register (CCM_POST_ROOT92_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_AE2C	Post Divider Register (CCM_POST_ROOT92_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_AE30	Pre Divider Register (CCM_PRE92)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_AE34	Pre Divider Register (CCM_PRE_ROOT92_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_AE38	Pre Divider Register (CCM_PRE_ROOT92_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_AE3C	Pre Divider Register (CCM_PRE_ROOT92_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_AE70	Access Control Register (CCM_ACCESS_CTRL92)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_AE74	Access Control Register (CCM_ACCESS_CTRL_ROOT92_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_AE78	Access Control Register (CCM_ACCESS_CTRL_ROOT92_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_AE7C	Access Control Register (CCM_ACCESS_CTRL_ROOT92_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_AE80	Target Register (CCM_TARGET_ROOT93)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_AE84	Target Register (CCM_TARGET_ROOT93_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_AE88	Target Register (CCM_TARGET_ROOT93_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_AE8C	Target Register (CCM_TARGET_ROOT93_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_AE90	Miscellaneous Register (CCM_MISC93)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_AE94	Miscellaneous Register (CCM_MISC_ROOT93_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_AE98	Miscellaneous Register (CCM_MISC_ROOT93_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_AE9C	Miscellaneous Register (CCM_MISC_ROOT93_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_AEA0	Post Divider Register (CCM_POST93)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_AEA4	Post Divider Register (CCM_POST_ROOT93_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_AEA8	Post Divider Register (CCM_POST_ROOT93_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_AEAC	Post Divider Register (CCM_POST_ROOT93_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_AEB0	Pre Divider Register (CCM_PRE93)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_AEB4	Pre Divider Register (CCM_PRE_ROOT93_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_AEB8	Pre Divider Register (CCM_PRE_ROOT93_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_AEBC	Pre Divider Register (CCM_PRE_ROOT93_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_AEF0	Access Control Register (CCM_ACCESS_CTRL93)	32	R/W	0000_0000h	5.2.8.26/ 735

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_AEF4	Access Control Register (CCM_ACCESS_CTRL_ROOT93_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_AEF8	Access Control Register (CCM_ACCESS_CTRL_ROOT93_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_AEFC	Access Control Register (CCM_ACCESS_CTRL_ROOT93_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_AF00	Target Register (CCM_TARGET_ROOT94)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_AF04	Target Register (CCM_TARGET_ROOT94_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_AF08	Target Register (CCM_TARGET_ROOT94_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_AF0C	Target Register (CCM_TARGET_ROOT94_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_AF10	Miscellaneous Register (CCM_MISC94)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_AF14	Miscellaneous Register (CCM_MISC_ROOT94_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_AF18	Miscellaneous Register (CCM_MISC_ROOT94_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_AF1C	Miscellaneous Register (CCM_MISC_ROOT94_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_AF20	Post Divider Register (CCM_POST94)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_AF24	Post Divider Register (CCM_POST_ROOT94_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_AF28	Post Divider Register (CCM_POST_ROOT94_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_AF2C	Post Divider Register (CCM_POST_ROOT94_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_AF30	Pre Divider Register (CCM_PRE94)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_AF34	Pre Divider Register (CCM_PRE_ROOT94_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_AF38	Pre Divider Register (CCM_PRE_ROOT94_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_AF3C	Pre Divider Register (CCM_PRE_ROOT94_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_AF70	Access Control Register (CCM_ACCESS_CTRL94)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_AF74	Access Control Register (CCM_ACCESS_CTRL_ROOT94_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_AF78	Access Control Register (CCM_ACCESS_CTRL_ROOT94_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_AF7C	Access Control Register (CCM_ACCESS_CTRL_ROOT94_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_AF80	Target Register (CCM_TARGET_ROOT95)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_AF84	Target Register (CCM_TARGET_ROOT95_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_AF88	Target Register (CCM_TARGET_ROOT95_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_AF8C	Target Register (CCM_TARGET_ROOT95_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_AF90	Miscellaneous Register (CCM_MISC95)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_AF94	Miscellaneous Register (CCM_MISC_ROOT95_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_AF98	Miscellaneous Register (CCM_MISC_ROOT95_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_AF9C	Miscellaneous Register (CCM_MISC_ROOT95_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_AFA0	Post Divider Register (CCM_POST95)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_AFA4	Post Divider Register (CCM_POST_ROOT95_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_AFA8	Post Divider Register (CCM_POST_ROOT95_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_AFAC	Post Divider Register (CCM_POST_ROOT95_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_AFB0	Pre Divider Register (CCM_PRE95)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_AFB4	Pre Divider Register (CCM_PRE_ROOT95_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_AFB8	Pre Divider Register (CCM_PRE_ROOT95_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_AFBC	Pre Divider Register (CCM_PRE_ROOT95_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_AFF0	Access Control Register (CCM_ACCESS_CTRL95)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_AFF4	Access Control Register (CCM_ACCESS_CTRL_ROOT95_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_AFF8	Access Control Register (CCM_ACCESS_CTRL_ROOT95_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_AFFC	Access Control Register (CCM_ACCESS_CTRL_ROOT95_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B000	Target Register (CCM_TARGET_ROOT96)	32	R/W	0000_0000h	5.2.8.10/ 699

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B004	Target Register (CCM_TARGET_ROOT96_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B008	Target Register (CCM_TARGET_ROOT96_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B00C	Target Register (CCM_TARGET_ROOT96_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B010	Miscellaneous Register (CCM_MISC96)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B014	Miscellaneous Register (CCM_MISC_ROOT96_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B018	Miscellaneous Register (CCM_MISC_ROOT96_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B01C	Miscellaneous Register (CCM_MISC_ROOT96_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B020	Post Divider Register (CCM_POST96)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B024	Post Divider Register (CCM_POST_ROOT96_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B028	Post Divider Register (CCM_POST_ROOT96_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B02C	Post Divider Register (CCM_POST_ROOT96_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B030	Pre Divider Register (CCM_PRE96)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B034	Pre Divider Register (CCM_PRE_ROOT96_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B038	Pre Divider Register (CCM_PRE_ROOT96_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B03C	Pre Divider Register (CCM_PRE_ROOT96_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B070	Access Control Register (CCM_ACCESS_CTRL96)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B074	Access Control Register (CCM_ACCESS_CTRL_ROOT96_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B078	Access Control Register (CCM_ACCESS_CTRL_ROOT96_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B07C	Access Control Register (CCM_ACCESS_CTRL_ROOT96_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B080	Target Register (CCM_TARGET_ROOT97)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B084	Target Register (CCM_TARGET_ROOT97_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B088	Target Register (CCM_TARGET_ROOT97_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B08C	Target Register (CCM_TARGET_ROOT97_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B090	Miscellaneous Register (CCM_MISC97)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B094	Miscellaneous Register (CCM_MISC_ROOT97_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B098	Miscellaneous Register (CCM_MISC_ROOT97_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B09C	Miscellaneous Register (CCM_MISC_ROOT97_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B0A0	Post Divider Register (CCM_POST97)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B0A4	Post Divider Register (CCM_POST_ROOT97_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B0A8	Post Divider Register (CCM_POST_ROOT97_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B0AC	Post Divider Register (CCM_POST_ROOT97_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B0B0	Pre Divider Register (CCM_PRE97)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B0B4	Pre Divider Register (CCM_PRE_ROOT97_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B0B8	Pre Divider Register (CCM_PRE_ROOT97_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B0BC	Pre Divider Register (CCM_PRE_ROOT97_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B0F0	Access Control Register (CCM_ACCESS_CTRL97)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B0F4	Access Control Register (CCM_ACCESS_CTRL_ROOT97_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B0F8	Access Control Register (CCM_ACCESS_CTRL_ROOT97_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B0FC	Access Control Register (CCM_ACCESS_CTRL_ROOT97_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B100	Target Register (CCM_TARGET_ROOT98)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B104	Target Register (CCM_TARGET_ROOT98_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B108	Target Register (CCM_TARGET_ROOT98_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B10C	Target Register (CCM_TARGET_ROOT98_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B110	Miscellaneous Register (CCM_MISC98)	32	R/W	0000_0000h	5.2.8.14/ 707

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B114	Miscellaneous Register (CCM_MISC_ROOT98_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B118	Miscellaneous Register (CCM_MISC_ROOT98_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B11C	Miscellaneous Register (CCM_MISC_ROOT98_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B120	Post Divider Register (CCM_POST98)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B124	Post Divider Register (CCM_POST_ROOT98_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B128	Post Divider Register (CCM_POST_ROOT98_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B12C	Post Divider Register (CCM_POST_ROOT98_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B130	Pre Divider Register (CCM_PRE98)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B134	Pre Divider Register (CCM_PRE_ROOT98_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B138	Pre Divider Register (CCM_PRE_ROOT98_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B13C	Pre Divider Register (CCM_PRE_ROOT98_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B170	Access Control Register (CCM_ACCESS_CTRL98)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B174	Access Control Register (CCM_ACCESS_CTRL_ROOT98_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B178	Access Control Register (CCM_ACCESS_CTRL_ROOT98_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B17C	Access Control Register (CCM_ACCESS_CTRL_ROOT98_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B180	Target Register (CCM_TARGET_ROOT99)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B184	Target Register (CCM_TARGET_ROOT99_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B188	Target Register (CCM_TARGET_ROOT99_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B18C	Target Register (CCM_TARGET_ROOT99_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B190	Miscellaneous Register (CCM_MISC99)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B194	Miscellaneous Register (CCM_MISC_ROOT99_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B198	Miscellaneous Register (CCM_MISC_ROOT99_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B19C	Miscellaneous Register (CCM_MISC_ROOT99_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B1A0	Post Divider Register (CCM_POST99)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B1A4	Post Divider Register (CCM_POST_ROOT99_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B1A8	Post Divider Register (CCM_POST_ROOT99_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B1AC	Post Divider Register (CCM_POST_ROOT99_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B1B0	Pre Divider Register (CCM_PRE99)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B1B4	Pre Divider Register (CCM_PRE_ROOT99_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B1B8	Pre Divider Register (CCM_PRE_ROOT99_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B1BC	Pre Divider Register (CCM_PRE_ROOT99_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B1F0	Access Control Register (CCM_ACCESS_CTRL99)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B1F4	Access Control Register (CCM_ACCESS_CTRL_ROOT99_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B1F8	Access Control Register (CCM_ACCESS_CTRL_ROOT99_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B1FC	Access Control Register (CCM_ACCESS_CTRL_ROOT99_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B200	Target Register (CCM_TARGET_ROOT100)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B204	Target Register (CCM_TARGET_ROOT100_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B208	Target Register (CCM_TARGET_ROOT100_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B20C	Target Register (CCM_TARGET_ROOT100_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B210	Miscellaneous Register (CCM_MISC100)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B214	Miscellaneous Register (CCM_MISC_ROOT100_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B218	Miscellaneous Register (CCM_MISC_ROOT100_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B21C	Miscellaneous Register (CCM_MISC_ROOT100_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B220	Post Divider Register (CCM_POST100)	32	R/W	0000_0000h	5.2.8.18/ 711

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B224	Post Divider Register (CCM_POST_ROOT100_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B228	Post Divider Register (CCM_POST_ROOT100_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B22C	Post Divider Register (CCM_POST_ROOT100_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B230	Pre Divider Register (CCM_PRE100)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B234	Pre Divider Register (CCM_PRE_ROOT100_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B238	Pre Divider Register (CCM_PRE_ROOT100_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B23C	Pre Divider Register (CCM_PRE_ROOT100_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B270	Access Control Register (CCM_ACCESS_CTRL100)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B274	Access Control Register (CCM_ACCESS_CTRL_ROOT100_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B278	Access Control Register (CCM_ACCESS_CTRL_ROOT100_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B27C	Access Control Register (CCM_ACCESS_CTRL_ROOT100_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B280	Target Register (CCM_TARGET_ROOT101)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B284	Target Register (CCM_TARGET_ROOT101_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B288	Target Register (CCM_TARGET_ROOT101_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B28C	Target Register (CCM_TARGET_ROOT101_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B290	Miscellaneous Register (CCM_MISC101)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B294	Miscellaneous Register (CCM_MISC_ROOT101_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B298	Miscellaneous Register (CCM_MISC_ROOT101_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B29C	Miscellaneous Register (CCM_MISC_ROOT101_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B2A0	Post Divider Register (CCM_POST101)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B2A4	Post Divider Register (CCM_POST_ROOT101_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B2A8	Post Divider Register (CCM_POST_ROOT101_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B2AC	Post Divider Register (CCM_POST_ROOT101_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B2B0	Pre Divider Register (CCM_PRE101)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B2B4	Pre Divider Register (CCM_PRE_ROOT101_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B2B8	Pre Divider Register (CCM_PRE_ROOT101_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B2BC	Pre Divider Register (CCM_PRE_ROOT101_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B2F0	Access Control Register (CCM_ACCESS_CTRL101)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B2F4	Access Control Register (CCM_ACCESS_CTRL_ROOT101_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B2F8	Access Control Register (CCM_ACCESS_CTRL_ROOT101_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B2FC	Access Control Register (CCM_ACCESS_CTRL_ROOT101_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B300	Target Register (CCM_TARGET_ROOT102)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B304	Target Register (CCM_TARGET_ROOT102_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B308	Target Register (CCM_TARGET_ROOT102_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B30C	Target Register (CCM_TARGET_ROOT102_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B310	Miscellaneous Register (CCM_MISC102)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B314	Miscellaneous Register (CCM_MISC_ROOT102_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B318	Miscellaneous Register (CCM_MISC_ROOT102_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B31C	Miscellaneous Register (CCM_MISC_ROOT102_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B320	Post Divider Register (CCM_POST102)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B324	Post Divider Register (CCM_POST_ROOT102_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B328	Post Divider Register (CCM_POST_ROOT102_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B32C	Post Divider Register (CCM_POST_ROOT102_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B330	Pre Divider Register (CCM_PRE102)	32	R/W	0000_0000h	5.2.8.22/ 723

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B334	Pre Divider Register (CCM_PRE_ROOT102_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B338	Pre Divider Register (CCM_PRE_ROOT102_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B33C	Pre Divider Register (CCM_PRE_ROOT102_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B370	Access Control Register (CCM_ACCESS_CTRL102)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B374	Access Control Register (CCM_ACCESS_CTRL_ROOT102_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B378	Access Control Register (CCM_ACCESS_CTRL_ROOT102_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B37C	Access Control Register (CCM_ACCESS_CTRL_ROOT102_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B380	Target Register (CCM_TARGET_ROOT103)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B384	Target Register (CCM_TARGET_ROOT103_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B388	Target Register (CCM_TARGET_ROOT103_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B38C	Target Register (CCM_TARGET_ROOT103_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B390	Miscellaneous Register (CCM_MISC103)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B394	Miscellaneous Register (CCM_MISC_ROOT103_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B398	Miscellaneous Register (CCM_MISC_ROOT103_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B39C	Miscellaneous Register (CCM_MISC_ROOT103_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B3A0	Post Divider Register (CCM_POST103)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B3A4	Post Divider Register (CCM_POST_ROOT103_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B3A8	Post Divider Register (CCM_POST_ROOT103_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B3AC	Post Divider Register (CCM_POST_ROOT103_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B3B0	Pre Divider Register (CCM_PRE103)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B3B4	Pre Divider Register (CCM_PRE_ROOT103_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B3B8	Pre Divider Register (CCM_PRE_ROOT103_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B3BC	Pre Divider Register (CCM_PRE_ROOT103_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B3F0	Access Control Register (CCM_ACCESS_CTRL103)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B3F4	Access Control Register (CCM_ACCESS_CTRL_ROOT103_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B3F8	Access Control Register (CCM_ACCESS_CTRL_ROOT103_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B3FC	Access Control Register (CCM_ACCESS_CTRL_ROOT103_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B400	Target Register (CCM_TARGET_ROOT104)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B404	Target Register (CCM_TARGET_ROOT104_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B408	Target Register (CCM_TARGET_ROOT104_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B40C	Target Register (CCM_TARGET_ROOT104_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B410	Miscellaneous Register (CCM_MISC104)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B414	Miscellaneous Register (CCM_MISC_ROOT104_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B418	Miscellaneous Register (CCM_MISC_ROOT104_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B41C	Miscellaneous Register (CCM_MISC_ROOT104_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B420	Post Divider Register (CCM_POST104)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B424	Post Divider Register (CCM_POST_ROOT104_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B428	Post Divider Register (CCM_POST_ROOT104_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B42C	Post Divider Register (CCM_POST_ROOT104_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B430	Pre Divider Register (CCM_PRE104)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B434	Pre Divider Register (CCM_PRE_ROOT104_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B438	Pre Divider Register (CCM_PRE_ROOT104_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B43C	Pre Divider Register (CCM_PRE_ROOT104_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B470	Access Control Register (CCM_ACCESS_CTRL104)	32	R/W	0000_0000h	5.2.8.26/ 735

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B474	Access Control Register (CCM_ACCESS_CTRL_ROOT104_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B478	Access Control Register (CCM_ACCESS_CTRL_ROOT104_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B47C	Access Control Register (CCM_ACCESS_CTRL_ROOT104_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B480	Target Register (CCM_TARGET_ROOT105)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B484	Target Register (CCM_TARGET_ROOT105_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B488	Target Register (CCM_TARGET_ROOT105_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B48C	Target Register (CCM_TARGET_ROOT105_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B490	Miscellaneous Register (CCM_MISC105)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B494	Miscellaneous Register (CCM_MISC_ROOT105_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B498	Miscellaneous Register (CCM_MISC_ROOT105_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B49C	Miscellaneous Register (CCM_MISC_ROOT105_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B4A0	Post Divider Register (CCM_POST105)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B4A4	Post Divider Register (CCM_POST_ROOT105_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B4A8	Post Divider Register (CCM_POST_ROOT105_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B4AC	Post Divider Register (CCM_POST_ROOT105_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B4B0	Pre Divider Register (CCM_PRE105)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B4B4	Pre Divider Register (CCM_PRE_ROOT105_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B4B8	Pre Divider Register (CCM_PRE_ROOT105_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B4BC	Pre Divider Register (CCM_PRE_ROOT105_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B4F0	Access Control Register (CCM_ACCESS_CTRL105)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B4F4	Access Control Register (CCM_ACCESS_CTRL_ROOT105_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B4F8	Access Control Register (CCM_ACCESS_CTRL_ROOT105_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B4FC	Access Control Register (CCM_ACCESS_CTRL_ROOT105_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B500	Target Register (CCM_TARGET_ROOT106)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B504	Target Register (CCM_TARGET_ROOT106_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B508	Target Register (CCM_TARGET_ROOT106_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B50C	Target Register (CCM_TARGET_ROOT106_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B510	Miscellaneous Register (CCM_MISC106)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B514	Miscellaneous Register (CCM_MISC_ROOT106_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B518	Miscellaneous Register (CCM_MISC_ROOT106_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B51C	Miscellaneous Register (CCM_MISC_ROOT106_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B520	Post Divider Register (CCM_POST106)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B524	Post Divider Register (CCM_POST_ROOT106_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B528	Post Divider Register (CCM_POST_ROOT106_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B52C	Post Divider Register (CCM_POST_ROOT106_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B530	Pre Divider Register (CCM_PRE106)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B534	Pre Divider Register (CCM_PRE_ROOT106_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B538	Pre Divider Register (CCM_PRE_ROOT106_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B53C	Pre Divider Register (CCM_PRE_ROOT106_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B570	Access Control Register (CCM_ACCESS_CTRL106)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B574	Access Control Register (CCM_ACCESS_CTRL_ROOT106_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B578	Access Control Register (CCM_ACCESS_CTRL_ROOT106_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B57C	Access Control Register (CCM_ACCESS_CTRL_ROOT106_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B580	Target Register (CCM_TARGET_ROOT107)	32	R/W	0000_0000h	5.2.8.10/ 699

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B584	Target Register (CCM_TARGET_ROOT107_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B588	Target Register (CCM_TARGET_ROOT107_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B58C	Target Register (CCM_TARGET_ROOT107_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B590	Miscellaneous Register (CCM_MISC107)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B594	Miscellaneous Register (CCM_MISC_ROOT107_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B598	Miscellaneous Register (CCM_MISC_ROOT107_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B59C	Miscellaneous Register (CCM_MISC_ROOT107_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B5A0	Post Divider Register (CCM_POST107)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B5A4	Post Divider Register (CCM_POST_ROOT107_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B5A8	Post Divider Register (CCM_POST_ROOT107_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B5AC	Post Divider Register (CCM_POST_ROOT107_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B5B0	Pre Divider Register (CCM_PRE107)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B5B4	Pre Divider Register (CCM_PRE_ROOT107_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B5B8	Pre Divider Register (CCM_PRE_ROOT107_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B5BC	Pre Divider Register (CCM_PRE_ROOT107_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B5F0	Access Control Register (CCM_ACCESS_CTRL107)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B5F4	Access Control Register (CCM_ACCESS_CTRL_ROOT107_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B5F8	Access Control Register (CCM_ACCESS_CTRL_ROOT107_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B5FC	Access Control Register (CCM_ACCESS_CTRL_ROOT107_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B600	Target Register (CCM_TARGET_ROOT108)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B604	Target Register (CCM_TARGET_ROOT108_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B608	Target Register (CCM_TARGET_ROOT108_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B60C	Target Register (CCM_TARGET_ROOT108_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B610	Miscellaneous Register (CCM_MISC108)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B614	Miscellaneous Register (CCM_MISC_ROOT108_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B618	Miscellaneous Register (CCM_MISC_ROOT108_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B61C	Miscellaneous Register (CCM_MISC_ROOT108_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B620	Post Divider Register (CCM_POST108)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B624	Post Divider Register (CCM_POST_ROOT108_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B628	Post Divider Register (CCM_POST_ROOT108_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B62C	Post Divider Register (CCM_POST_ROOT108_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B630	Pre Divider Register (CCM_PRE108)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B634	Pre Divider Register (CCM_PRE_ROOT108_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B638	Pre Divider Register (CCM_PRE_ROOT108_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B63C	Pre Divider Register (CCM_PRE_ROOT108_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B670	Access Control Register (CCM_ACCESS_CTRL108)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B674	Access Control Register (CCM_ACCESS_CTRL_ROOT108_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B678	Access Control Register (CCM_ACCESS_CTRL_ROOT108_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B67C	Access Control Register (CCM_ACCESS_CTRL_ROOT108_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B680	Target Register (CCM_TARGET_ROOT109)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B684	Target Register (CCM_TARGET_ROOT109_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B688	Target Register (CCM_TARGET_ROOT109_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B68C	Target Register (CCM_TARGET_ROOT109_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B690	Miscellaneous Register (CCM_MISC109)	32	R/W	0000_0000h	5.2.8.14/ 707

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B694	Miscellaneous Register (CCM_MISC_ROOT109_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B698	Miscellaneous Register (CCM_MISC_ROOT109_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B69C	Miscellaneous Register (CCM_MISC_ROOT109_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B6A0	Post Divider Register (CCM_POST109)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B6A4	Post Divider Register (CCM_POST_ROOT109_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B6A8	Post Divider Register (CCM_POST_ROOT109_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B6AC	Post Divider Register (CCM_POST_ROOT109_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B6B0	Pre Divider Register (CCM_PRE109)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B6B4	Pre Divider Register (CCM_PRE_ROOT109_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B6B8	Pre Divider Register (CCM_PRE_ROOT109_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B6BC	Pre Divider Register (CCM_PRE_ROOT109_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B6F0	Access Control Register (CCM_ACCESS_CTRL109)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B6F4	Access Control Register (CCM_ACCESS_CTRL_ROOT109_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B6F8	Access Control Register (CCM_ACCESS_CTRL_ROOT109_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B6FC	Access Control Register (CCM_ACCESS_CTRL_ROOT109_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B700	Target Register (CCM_TARGET_ROOT110)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B704	Target Register (CCM_TARGET_ROOT110_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B708	Target Register (CCM_TARGET_ROOT110_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B70C	Target Register (CCM_TARGET_ROOT110_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B710	Miscellaneous Register (CCM_MISC110)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B714	Miscellaneous Register (CCM_MISC_ROOT110_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B718	Miscellaneous Register (CCM_MISC_ROOT110_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B71C	Miscellaneous Register (CCM_MISC_ROOT110_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B720	Post Divider Register (CCM_POST110)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B724	Post Divider Register (CCM_POST_ROOT110_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B728	Post Divider Register (CCM_POST_ROOT110_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B72C	Post Divider Register (CCM_POST_ROOT110_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B730	Pre Divider Register (CCM_PRE110)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B734	Pre Divider Register (CCM_PRE_ROOT110_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B738	Pre Divider Register (CCM_PRE_ROOT110_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B73C	Pre Divider Register (CCM_PRE_ROOT110_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B770	Access Control Register (CCM_ACCESS_CTRL110)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B774	Access Control Register (CCM_ACCESS_CTRL_ROOT110_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B778	Access Control Register (CCM_ACCESS_CTRL_ROOT110_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B77C	Access Control Register (CCM_ACCESS_CTRL_ROOT110_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B780	Target Register (CCM_TARGET_ROOT111)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B784	Target Register (CCM_TARGET_ROOT111_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B788	Target Register (CCM_TARGET_ROOT111_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B78C	Target Register (CCM_TARGET_ROOT111_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B790	Miscellaneous Register (CCM_MISC111)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B794	Miscellaneous Register (CCM_MISC_ROOT111_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B798	Miscellaneous Register (CCM_MISC_ROOT111_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B79C	Miscellaneous Register (CCM_MISC_ROOT111_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B7A0	Post Divider Register (CCM_POST111)	32	R/W	0000_0000h	5.2.8.18/ 711

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B7A4	Post Divider Register (CCM_POST_ROOT111_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B7A8	Post Divider Register (CCM_POST_ROOT111_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B7AC	Post Divider Register (CCM_POST_ROOT111_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B7B0	Pre Divider Register (CCM_PRE111)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B7B4	Pre Divider Register (CCM_PRE_ROOT111_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B7B8	Pre Divider Register (CCM_PRE_ROOT111_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B7BC	Pre Divider Register (CCM_PRE_ROOT111_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B7F0	Access Control Register (CCM_ACCESS_CTRL111)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B7F4	Access Control Register (CCM_ACCESS_CTRL_ROOT111_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B7F8	Access Control Register (CCM_ACCESS_CTRL_ROOT111_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B7FC	Access Control Register (CCM_ACCESS_CTRL_ROOT111_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B800	Target Register (CCM_TARGET_ROOT112)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B804	Target Register (CCM_TARGET_ROOT112_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B808	Target Register (CCM_TARGET_ROOT112_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B80C	Target Register (CCM_TARGET_ROOT112_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B810	Miscellaneous Register (CCM_MISC112)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B814	Miscellaneous Register (CCM_MISC_ROOT112_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B818	Miscellaneous Register (CCM_MISC_ROOT112_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B81C	Miscellaneous Register (CCM_MISC_ROOT112_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B820	Post Divider Register (CCM_POST112)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B824	Post Divider Register (CCM_POST_ROOT112_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B828	Post Divider Register (CCM_POST_ROOT112_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B82C	Post Divider Register (CCM_POST_ROOT112_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B830	Pre Divider Register (CCM_PRE112)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B834	Pre Divider Register (CCM_PRE_ROOT112_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B838	Pre Divider Register (CCM_PRE_ROOT112_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B83C	Pre Divider Register (CCM_PRE_ROOT112_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B870	Access Control Register (CCM_ACCESS_CTRL112)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B874	Access Control Register (CCM_ACCESS_CTRL_ROOT112_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B878	Access Control Register (CCM_ACCESS_CTRL_ROOT112_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B87C	Access Control Register (CCM_ACCESS_CTRL_ROOT112_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B880	Target Register (CCM_TARGET_ROOT113)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B884	Target Register (CCM_TARGET_ROOT113_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B888	Target Register (CCM_TARGET_ROOT113_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B88C	Target Register (CCM_TARGET_ROOT113_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B890	Miscellaneous Register (CCM_MISC113)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B894	Miscellaneous Register (CCM_MISC_ROOT113_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B898	Miscellaneous Register (CCM_MISC_ROOT113_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B89C	Miscellaneous Register (CCM_MISC_ROOT113_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B8A0	Post Divider Register (CCM_POST113)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B8A4	Post Divider Register (CCM_POST_ROOT113_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B8A8	Post Divider Register (CCM_POST_ROOT113_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B8AC	Post Divider Register (CCM_POST_ROOT113_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B8B0	Pre Divider Register (CCM_PRE113)	32	R/W	0000_0000h	5.2.8.22/ 723

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B8B4	Pre Divider Register (CCM_PRE_ROOT113_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B8B8	Pre Divider Register (CCM_PRE_ROOT113_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B8BC	Pre Divider Register (CCM_PRE_ROOT113_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B8F0	Access Control Register (CCM_ACCESS_CTRL113)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B8F4	Access Control Register (CCM_ACCESS_CTRL_ROOT113_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B8F8	Access Control Register (CCM_ACCESS_CTRL_ROOT113_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B8FC	Access Control Register (CCM_ACCESS_CTRL_ROOT113_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B900	Target Register (CCM_TARGET_ROOT114)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B904	Target Register (CCM_TARGET_ROOT114_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B908	Target Register (CCM_TARGET_ROOT114_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B90C	Target Register (CCM_TARGET_ROOT114_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B910	Miscellaneous Register (CCM_MISC114)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B914	Miscellaneous Register (CCM_MISC_ROOT114_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B918	Miscellaneous Register (CCM_MISC_ROOT114_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B91C	Miscellaneous Register (CCM_MISC_ROOT114_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B920	Post Divider Register (CCM_POST114)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B924	Post Divider Register (CCM_POST_ROOT114_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B928	Post Divider Register (CCM_POST_ROOT114_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B92C	Post Divider Register (CCM_POST_ROOT114_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B930	Pre Divider Register (CCM_PRE114)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B934	Pre Divider Register (CCM_PRE_ROOT114_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B938	Pre Divider Register (CCM_PRE_ROOT114_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B93C	Pre Divider Register (CCM_PRE_ROOT114_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B970	Access Control Register (CCM_ACCESS_CTRL114)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_B974	Access Control Register (CCM_ACCESS_CTRL_ROOT114_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B978	Access Control Register (CCM_ACCESS_CTRL_ROOT114_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B97C	Access Control Register (CCM_ACCESS_CTRL_ROOT114_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_B980	Target Register (CCM_TARGET_ROOT115)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_B984	Target Register (CCM_TARGET_ROOT115_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_B988	Target Register (CCM_TARGET_ROOT115_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_B98C	Target Register (CCM_TARGET_ROOT115_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_B990	Miscellaneous Register (CCM_MISC115)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_B994	Miscellaneous Register (CCM_MISC_ROOT115_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_B998	Miscellaneous Register (CCM_MISC_ROOT115_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_B99C	Miscellaneous Register (CCM_MISC_ROOT115_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_B9A0	Post Divider Register (CCM_POST115)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_B9A4	Post Divider Register (CCM_POST_ROOT115_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_B9A8	Post Divider Register (CCM_POST_ROOT115_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_B9AC	Post Divider Register (CCM_POST_ROOT115_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_B9B0	Pre Divider Register (CCM_PRE115)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_B9B4	Pre Divider Register (CCM_PRE_ROOT115_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_B9B8	Pre Divider Register (CCM_PRE_ROOT115_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_B9BC	Pre Divider Register (CCM_PRE_ROOT115_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_B9F0	Access Control Register (CCM_ACCESS_CTRL115)	32	R/W	0000_0000h	5.2.8.26/ 735

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_B9F4	Access Control Register (CCM_ACCESS_CTRL_ROOT115_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_B9F8	Access Control Register (CCM_ACCESS_CTRL_ROOT115_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_B9FC	Access Control Register (CCM_ACCESS_CTRL_ROOT115_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_BA00	Target Register (CCM_TARGET_ROOT116)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_BA04	Target Register (CCM_TARGET_ROOT116_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_BA08	Target Register (CCM_TARGET_ROOT116_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_BA0C	Target Register (CCM_TARGET_ROOT116_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_BA10	Miscellaneous Register (CCM_MISC116)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_BA14	Miscellaneous Register (CCM_MISC_ROOT116_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_BA18	Miscellaneous Register (CCM_MISC_ROOT116_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_BA1C	Miscellaneous Register (CCM_MISC_ROOT116_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_BA20	Post Divider Register (CCM_POST116)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_BA24	Post Divider Register (CCM_POST_ROOT116_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_BA28	Post Divider Register (CCM_POST_ROOT116_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_BA2C	Post Divider Register (CCM_POST_ROOT116_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_BA30	Pre Divider Register (CCM_PRE116)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_BA34	Pre Divider Register (CCM_PRE_ROOT116_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_BA38	Pre Divider Register (CCM_PRE_ROOT116_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_BA3C	Pre Divider Register (CCM_PRE_ROOT116_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_BA70	Access Control Register (CCM_ACCESS_CTRL116)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_BA74	Access Control Register (CCM_ACCESS_CTRL_ROOT116_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_BA78	Access Control Register (CCM_ACCESS_CTRL_ROOT116_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_BA7C	Access Control Register (CCM_ACCESS_CTRL_ROOT116_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_BA80	Target Register (CCM_TARGET_ROOT117)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_BA84	Target Register (CCM_TARGET_ROOT117_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_BA88	Target Register (CCM_TARGET_ROOT117_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_BA8C	Target Register (CCM_TARGET_ROOT117_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_BA90	Miscellaneous Register (CCM_MISC117)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_BA94	Miscellaneous Register (CCM_MISC_ROOT117_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_BA98	Miscellaneous Register (CCM_MISC_ROOT117_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_BA9C	Miscellaneous Register (CCM_MISC_ROOT117_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_BAA0	Post Divider Register (CCM_POST117)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_BAA4	Post Divider Register (CCM_POST_ROOT117_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_BAA8	Post Divider Register (CCM_POST_ROOT117_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_BAAC	Post Divider Register (CCM_POST_ROOT117_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_BAB0	Pre Divider Register (CCM_PRE117)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_BAB4	Pre Divider Register (CCM_PRE_ROOT117_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_BAB8	Pre Divider Register (CCM_PRE_ROOT117_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_BABC	Pre Divider Register (CCM_PRE_ROOT117_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_BAF0	Access Control Register (CCM_ACCESS_CTRL117)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_BAF4	Access Control Register (CCM_ACCESS_CTRL_ROOT117_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_BAF8	Access Control Register (CCM_ACCESS_CTRL_ROOT117_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_BAFC	Access Control Register (CCM_ACCESS_CTRL_ROOT117_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_BB00	Target Register (CCM_TARGET_ROOT118)	32	R/W	0000_0000h	5.2.8.10/ 699

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_BB04	Target Register (CCM_TARGET_ROOT118_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_BB08	Target Register (CCM_TARGET_ROOT118_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_BB0C	Target Register (CCM_TARGET_ROOT118_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_BB10	Miscellaneous Register (CCM_MISC118)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_BB14	Miscellaneous Register (CCM_MISC_ROOT118_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_BB18	Miscellaneous Register (CCM_MISC_ROOT118_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_BB1C	Miscellaneous Register (CCM_MISC_ROOT118_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_BB20	Post Divider Register (CCM_POST118)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_BB24	Post Divider Register (CCM_POST_ROOT118_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_BB28	Post Divider Register (CCM_POST_ROOT118_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_BB2C	Post Divider Register (CCM_POST_ROOT118_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_BB30	Pre Divider Register (CCM_PRE118)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_BB34	Pre Divider Register (CCM_PRE_ROOT118_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_BB38	Pre Divider Register (CCM_PRE_ROOT118_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_BB3C	Pre Divider Register (CCM_PRE_ROOT118_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_BB70	Access Control Register (CCM_ACCESS_CTRL118)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_BB74	Access Control Register (CCM_ACCESS_CTRL_ROOT118_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_BB78	Access Control Register (CCM_ACCESS_CTRL_ROOT118_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_BB7C	Access Control Register (CCM_ACCESS_CTRL_ROOT118_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_BB80	Target Register (CCM_TARGET_ROOT119)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_BB84	Target Register (CCM_TARGET_ROOT119_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_BB88	Target Register (CCM_TARGET_ROOT119_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_BB8C	Target Register (CCM_TARGET_ROOT119_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_BB90	Miscellaneous Register (CCM_MISC119)	32	R/W	0000_0000h	5.2.8.14/ 707
3038_BB94	Miscellaneous Register (CCM_MISC_ROOT119_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_BB98	Miscellaneous Register (CCM_MISC_ROOT119_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_BB9C	Miscellaneous Register (CCM_MISC_ROOT119_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_BBA0	Post Divider Register (CCM_POST119)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_BBA4	Post Divider Register (CCM_POST_ROOT119_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_BBA8	Post Divider Register (CCM_POST_ROOT119_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_BBAC	Post Divider Register (CCM_POST_ROOT119_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_BBB0	Pre Divider Register (CCM_PRE119)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_BBB4	Pre Divider Register (CCM_PRE_ROOT119_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_BBB8	Pre Divider Register (CCM_PRE_ROOT119_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_BBBC	Pre Divider Register (CCM_PRE_ROOT119_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_BBF0	Access Control Register (CCM_ACCESS_CTRL119)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_BBF4	Access Control Register (CCM_ACCESS_CTRL_ROOT119_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_BBF8	Access Control Register (CCM_ACCESS_CTRL_ROOT119_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_BBFC	Access Control Register (CCM_ACCESS_CTRL_ROOT119_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742
3038_BC00	Target Register (CCM_TARGET_ROOT120)	32	R/W	0000_0000h	5.2.8.10/ 699
3038_BC04	Target Register (CCM_TARGET_ROOT120_SET)	32	R/W	0000_0000h	5.2.8.11/ 701
3038_BC08	Target Register (CCM_TARGET_ROOT120_CLR)	32	R/W	0000_0000h	5.2.8.12/ 703
3038_BC0C	Target Register (CCM_TARGET_ROOT120_TOG)	32	R/W	0000_0000h	5.2.8.13/ 705
3038_BC10	Miscellaneous Register (CCM_MISC120)	32	R/W	0000_0000h	5.2.8.14/ 707

Table continues on the next page...

CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3038_BC14	Miscellaneous Register (CCM_MISC_ROOT120_SET)	32	R/W	0000_0000h	5.2.8.15/ 708
3038_BC18	Miscellaneous Register (CCM_MISC_ROOT120_CLR)	32	R/W	0000_0000h	5.2.8.16/ 709
3038_BC1C	Miscellaneous Register (CCM_MISC_ROOT120_TOG)	32	R/W	0000_0000h	5.2.8.17/ 710
3038_BC20	Post Divider Register (CCM_POST120)	32	R/W	0000_0000h	5.2.8.18/ 711
3038_BC24	Post Divider Register (CCM_POST_ROOT120_SET)	32	R/W	0000_0000h	5.2.8.19/ 714
3038_BC28	Post Divider Register (CCM_POST_ROOT120_CLR)	32	R/W	0000_0000h	5.2.8.20/ 717
3038_BC2C	Post Divider Register (CCM_POST_ROOT120_TOG)	32	R/W	0000_0000h	5.2.8.21/ 720
3038_BC30	Pre Divider Register (CCM_PRE120)	32	R/W	0000_0000h	5.2.8.22/ 723
3038_BC34	Pre Divider Register (CCM_PRE_ROOT120_SET)	32	R/W	0000_0000h	5.2.8.23/ 726
3038_BC38	Pre Divider Register (CCM_PRE_ROOT120_CLR)	32	R/W	0000_0000h	5.2.8.24/ 729
3038_BC3C	Pre Divider Register (CCM_PRE_ROOT120_TOG)	32	R/W	0000_0000h	5.2.8.25/ 732
3038_BC70	Access Control Register (CCM_ACCESS_CTRL120)	32	R/W	0000_0000h	5.2.8.26/ 735
3038_BC74	Access Control Register (CCM_ACCESS_CTRL_ROOT120_SET)	32	R/W	0000_0000h	5.2.8.27/ 737
3038_BC78	Access Control Register (CCM_ACCESS_CTRL_ROOT120_CLR)	32	R/W	0000_0000h	5.2.8.28/ 740
3038_BC7C	Access Control Register (CCM_ACCESS_CTRL_ROOT120_TOG)	32	R/W	0000_0000h	5.2.8.29/ 742

5.2.8.1 General Purpose Register (CCM_GPR0*n*)

GPR0

Address: 3038_0000h base + 0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_GPR0*n* field descriptions

Field	Description
GP0	Timeout cycle count of ipg_clk, when perform read and write.

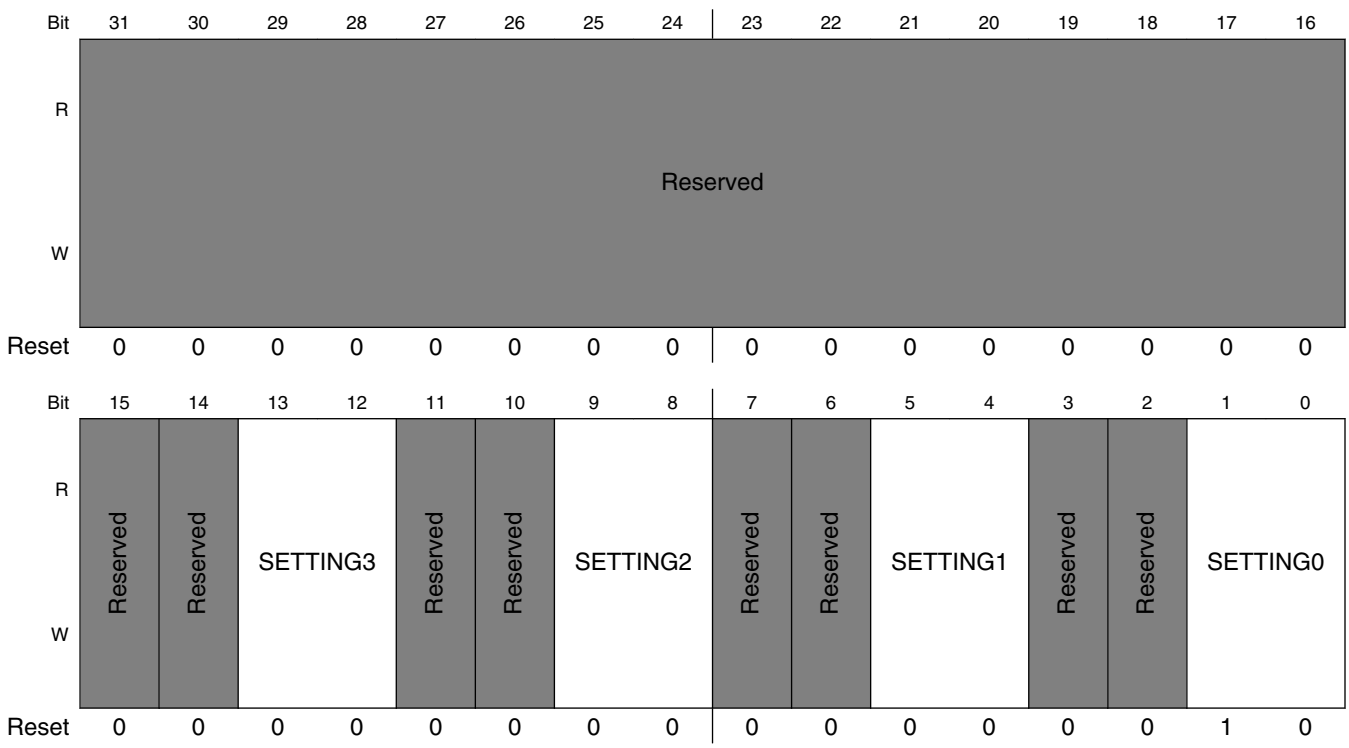
5.2.8.2 CCM PLL Control Register (CCM_PLL_CTRL*n*)

See Input Clocks for PLL control mapping.

NOTE

For the SoC to correctly power up after entering DSM, CCM_PLL_CTRLx must not be set to 0x0 or 0x3 for any domain in use.

Address: 3038_0000h base + 800h offset + (16d × i), where i=0d to 32d



CCM_PLL_CTRL*n* field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved

Table continues on the next page...

CCM_PLL_CTRL*n* field descriptions (continued)

Field	Description
14 -	This field is reserved. Reserved
13–12 SETTING3	Clock gate control setting for domain 3. This field can only be written by domain 3 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
11 -	This field is reserved. Reserved
10 -	This field is reserved. Reserved
9–8 SETTING2	Clock gate control setting for domain 2. This field can only be written by domain 2 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
7 -	This field is reserved. Reserved
6 -	This field is reserved. Reserved
5–4 SETTING1	Clock gate control setting for domain 1. This field can only be written by domain 1. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
3 -	This field is reserved. Reserved
2 -	This field is reserved. Reserved
SETTING0	Clock gate control setting for domain 0. This field can only be written by domain 0. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time

5.2.8.3 CCM PLL Control Register (CCM_PLL_CTRL*n*_SET)

See Input Clocks for PLL control mapping.

NOTE

For the SoC to correctly power up after entering DSM, CCM_PLL_CTRLx must not be set to 0x0 or 0x3 for any domain in use.

Address: 3038_0000h base + 804h offset + (16d × i), where i=0d to 32d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	Reserved	SETTING3		Reserved	Reserved	SETTING2		Reserved	Reserved	SETTING1		Reserved	Reserved	SETTING0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

CCM_PLL_CTRL*n*_SET field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14 -	This field is reserved. Reserved
13–12 SETTING3	Clock gate control setting for domain 3. This field can only be written by domain 3 00 Domain clocks not needed

Table continues on the next page...

CCM_PLL_CTRL*n*_SET field descriptions (continued)

Field	Description
	01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
11 -	This field is reserved. Reserved
10 -	This field is reserved. Reserved
9–8 SETTING2	Clock gate control setting for domain 2. This field can only be written by domain 2 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
7 -	This field is reserved. Reserved
6 -	This field is reserved. Reserved
5–4 SETTING1	Clock gate control setting for domain 1. This field can only be written by domain 1. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
3 -	This field is reserved. Reserved
2 -	This field is reserved. Reserved
SETTING0	Clock gate control setting for domain 0. This field can only be written by domain 0. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time

5.2.8.4 CCM PLL Control Register (CCM_PLL_CTRL*n*_CLR)

See Input Clocks for PLL control mapping.

NOTE

For the SoC to correctly power up after entering DSM, CCM_PLL_CTRLx must not be set to 0x0 or 0x3 for any domain in use.

Address: 3038_0000h base + 808h offset + (16d × i), where i=0d to 32d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved	Reserved	SETTING3		Reserved	Reserved	SETTING2		Reserved	Reserved	SETTING1		Reserved	Reserved	SETTING0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

CCM_PLL_CTRL*n*_CLR field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14 -	This field is reserved. Reserved
13–12 SETTING3	Clock gate control setting for domain 3. This field can only be written by domain 3 00 Domain clocks not needed

Table continues on the next page...

CCM_PLL_CTRL*n*_CLR field descriptions (continued)

Field	Description
	01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
11 -	This field is reserved. Reserved
10 -	This field is reserved. Reserved
9–8 SETTING2	Clock gate control setting for domain 2. This field can only be written by domain 2 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
7 -	This field is reserved. Reserved
6 -	This field is reserved. Reserved
5–4 SETTING1	Clock gate control setting for domain 1. This field can only be written by domain 1. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
3 -	This field is reserved. Reserved
2 -	This field is reserved. Reserved
SETTING0	Clock gate control setting for domain 0. This field can only be written by domain 0. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time

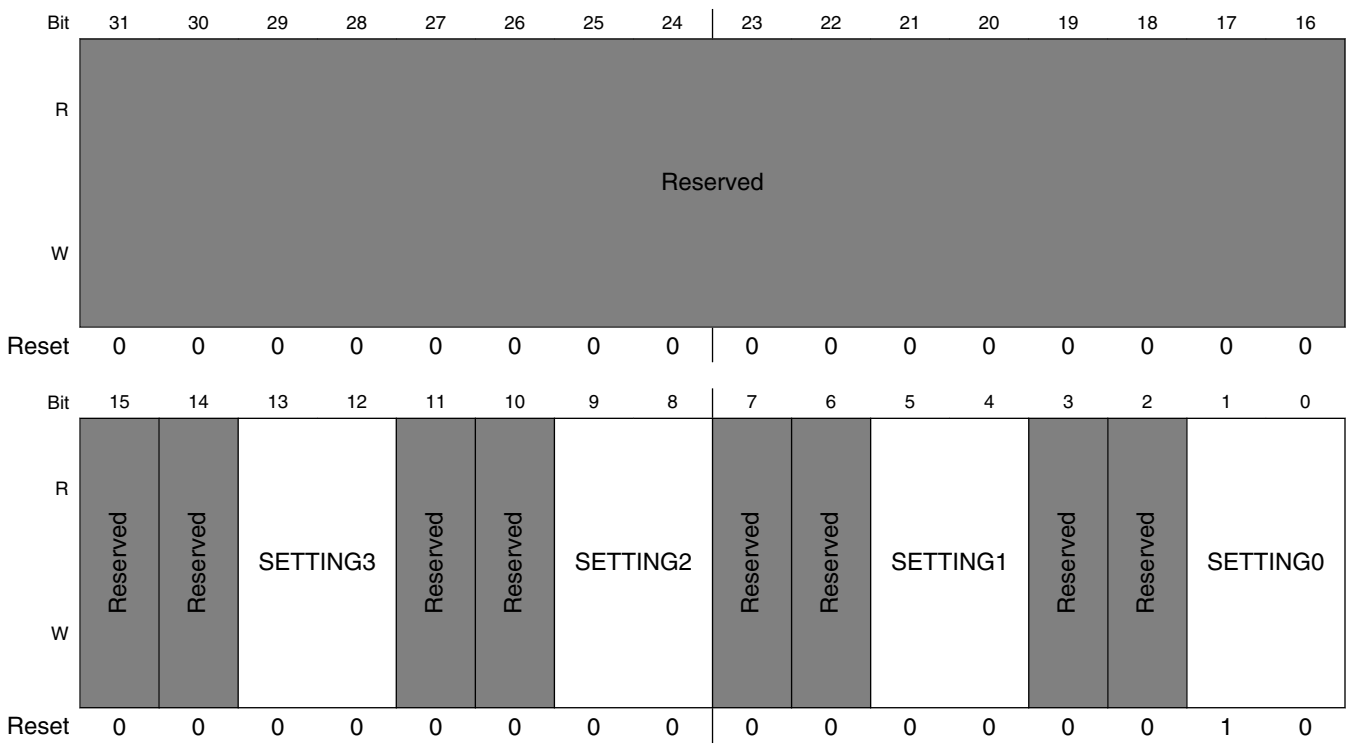
5.2.8.5 CCM PLL Control Register (CCM_PLL_CTRL*n*_TOG)

See Input Clocks for PLL control mapping.

NOTE

For the SoC to correctly power up after entering DSM, CCM_PLL_CTRLx must not be set to 0x0 or 0x3 for any domain in use.

Address: 3038_0000h base + 80Ch offset + (16d × i), where i=0d to 32d



CCM_PLL_CTRL*n*_TOG field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14 -	This field is reserved. Reserved
13–12 SETTING3	Clock gate control setting for domain 3. This field can only be written by domain 3 00 Domain clocks not needed

Table continues on the next page...

CCM_PLL_CTRL*n*_TOG field descriptions (continued)

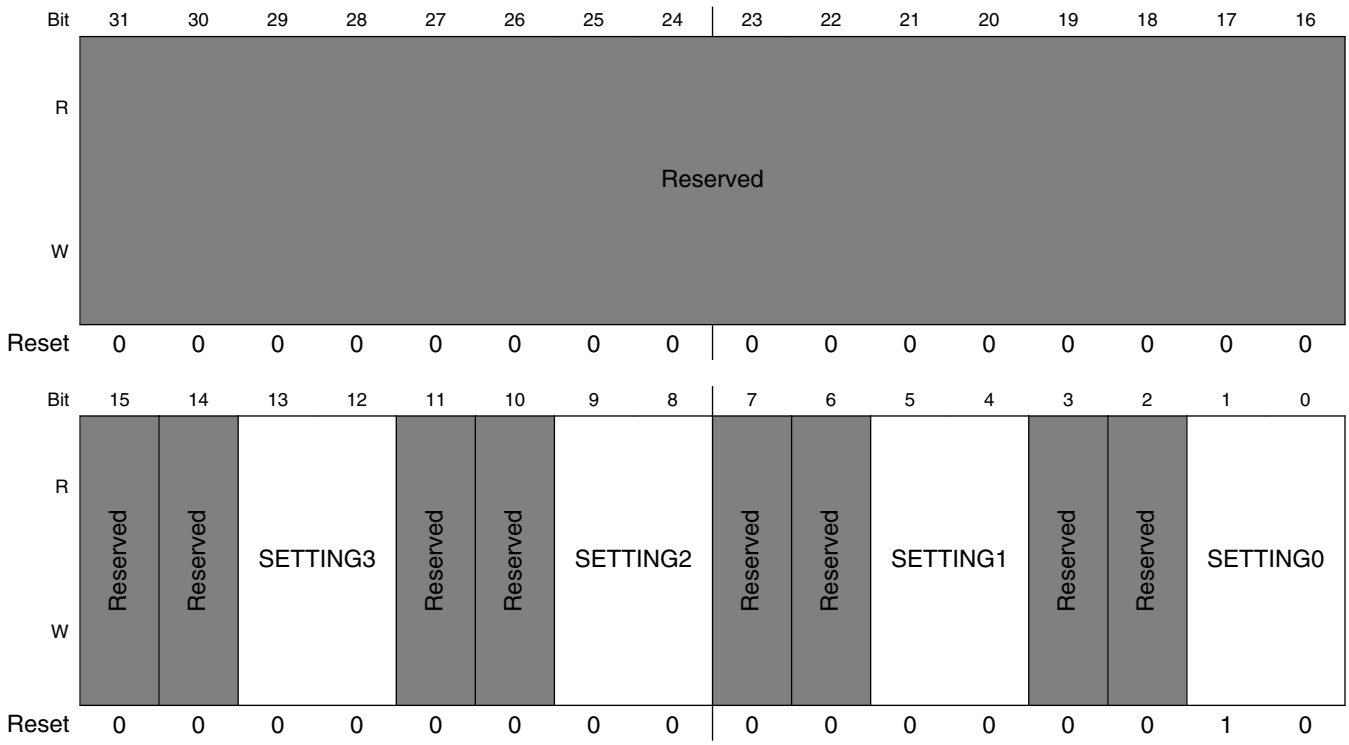
Field	Description
	01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
11 -	This field is reserved. Reserved
10 -	This field is reserved. Reserved
9–8 SETTING2	Clock gate control setting for domain 2. This field can only be written by domain 2 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
7 -	This field is reserved. Reserved
6 -	This field is reserved. Reserved
5–4 SETTING1	Clock gate control setting for domain 1. This field can only be written by domain 1. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
3 -	This field is reserved. Reserved
2 -	This field is reserved. Reserved
SETTING0	Clock gate control setting for domain 0. This field can only be written by domain 0. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time

5.2.8.6 CCM Clock Gating Register (CCM_CCGR*n*)

NOTE

Not all CCGRs are mapped. See CCGR Interface for CCGR mapping and clock gating information.

Address: 3038_0000h base + 4000h offset + (16d × i), where i=0d to 190d



CCM_CCGR*n* field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14 -	This field is reserved. Reserved
13–12 SETTING3	Clock gate control setting for domain 3. This field can only be written by domain 3 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time

Table continues on the next page...

CCM_CCGR*n* field descriptions (continued)

Field	Description
11 -	This field is reserved. Reserved
10 -	This field is reserved. Reserved
9–8 SETTING2	Clock gate control setting for domain 2. This field can only be written by domain 2 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
7 -	This field is reserved. Reserved
6 -	This field is reserved. Reserved
5–4 SETTING1	Clock gate control setting for domain 1. This field can only be written by domain 1. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
3 -	This field is reserved. Reserved
2 -	This field is reserved. Reserved
SETTING0	Clock gate control setting for domain 0. This field can only be written by domain 0. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time

5.2.8.7 CCM Clock Gating Register (CCM_CCGR*n*_SET)

NOTE

Not all CCGRs are mapped. See CCGR Interface for CCGR mapping and clock gating information.

Address: 3038_0000h base + 4004h offset + (16d × i), where i=0d to 190d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved	Reserved	SETTING3		Reserved	Reserved	SETTING2		Reserved	Reserved	SETTING1		Reserved	Reserved	SETTING0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

CCM_CCGR*n*_SET field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14 -	This field is reserved. Reserved
13–12 SETTING3	Clock gate control setting for domain 3. This field can only be written by domain 3 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time

Table continues on the next page...

CCM_CCGR*n*_SET field descriptions (continued)

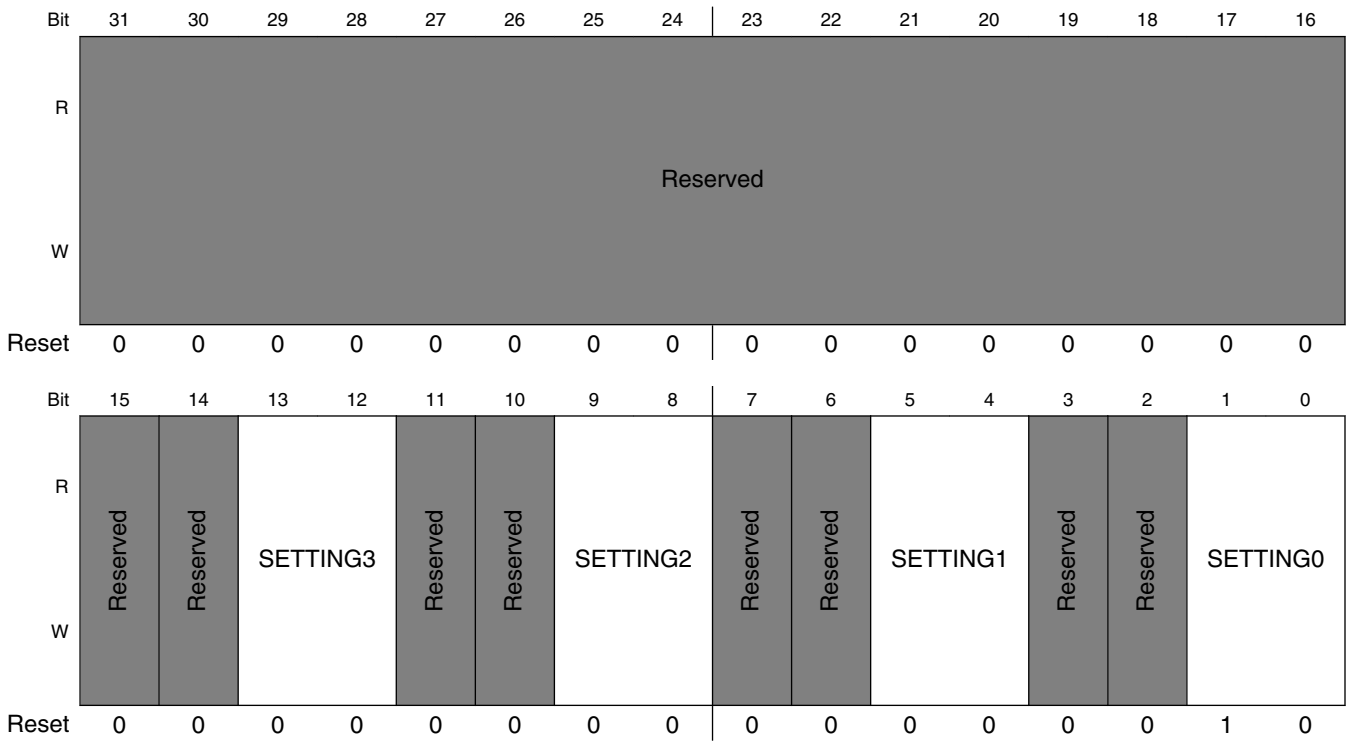
Field	Description
11 -	This field is reserved. Reserved
10 -	This field is reserved. Reserved
9–8 SETTING2	Clock gate control setting for domain 2. This field can only be written by domain 2 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
7 -	This field is reserved. Reserved
6 -	This field is reserved. Reserved
5–4 SETTING1	Clock gate control setting for domain 1. This field can only be written by domain 1. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
3 -	This field is reserved. Reserved
2 -	This field is reserved. Reserved
SETTING0	Clock gate control setting for domain 0. This field can only be written by domain 0. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time

5.2.8.8 CCM Clock Gating Register (CCM_CCGR*n*_CLR)

NOTE

Not all CCGRs are mapped. See CCGR Interface for CCGR mapping and clock gating information.

Address: 3038_0000h base + 4008h offset + (16d × i), where i=0d to 190d



CCM_CCGR*n*_CLR field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14 -	This field is reserved. Reserved
13–12 SETTING3	Clock gate control setting for domain 3. This field can only be written by domain 3 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time

Table continues on the next page...

CCM_CCGR*n*_CLR field descriptions (continued)

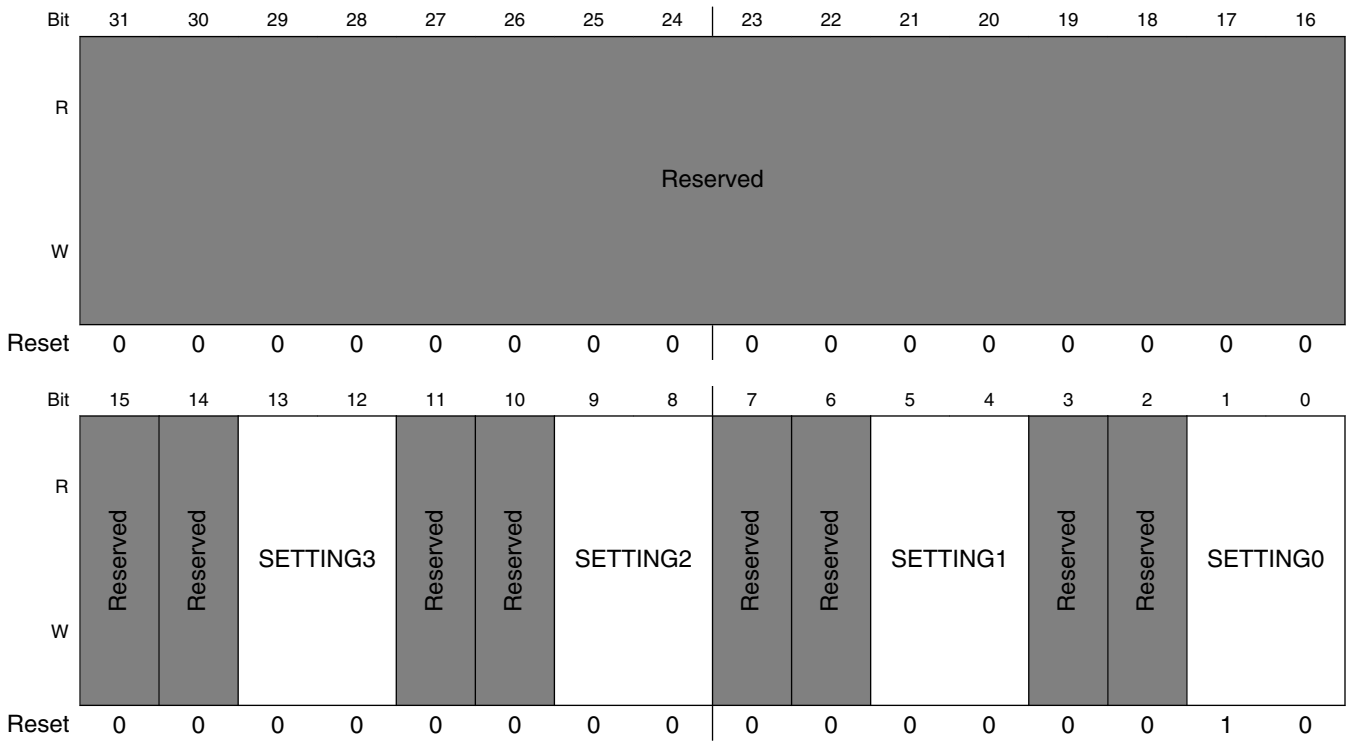
Field	Description
11 -	This field is reserved. Reserved
10 -	This field is reserved. Reserved
9–8 SETTING2	Clock gate control setting for domain 2. This field can only be written by domain 2 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
7 -	This field is reserved. Reserved
6 -	This field is reserved. Reserved
5–4 SETTING1	Clock gate control setting for domain 1. This field can only be written by domain 1. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
3 -	This field is reserved. Reserved
2 -	This field is reserved. Reserved
SETTING0	Clock gate control setting for domain 0. This field can only be written by domain 0. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time

5.2.8.9 CCM Clock Gating Register (CCM_CCGR*n*_TOG)

NOTE

Not all CCGRs are mapped. See CCGR Interface for CCGR mapping and clock gating information.

Address: 3038_0000h base + 400Ch offset + (16d × i), where i=0d to 190d



CCM_CCGR*n*_TOG field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14 -	This field is reserved. Reserved
13–12 SETTING3	Clock gate control setting for domain 3. This field can only be written by domain 3 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time

Table continues on the next page...

CCM_CCGR*n*_TOG field descriptions (continued)

Field	Description
11 -	This field is reserved. Reserved
10 -	This field is reserved. Reserved
9–8 SETTING2	Clock gate control setting for domain 2. This field can only be written by domain 2 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
7 -	This field is reserved. Reserved
6 -	This field is reserved. Reserved
5–4 SETTING1	Clock gate control setting for domain 1. This field can only be written by domain 1. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time
3 -	This field is reserved. Reserved
2 -	This field is reserved. Reserved
SETTING0	Clock gate control setting for domain 0. This field can only be written by domain 0. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time

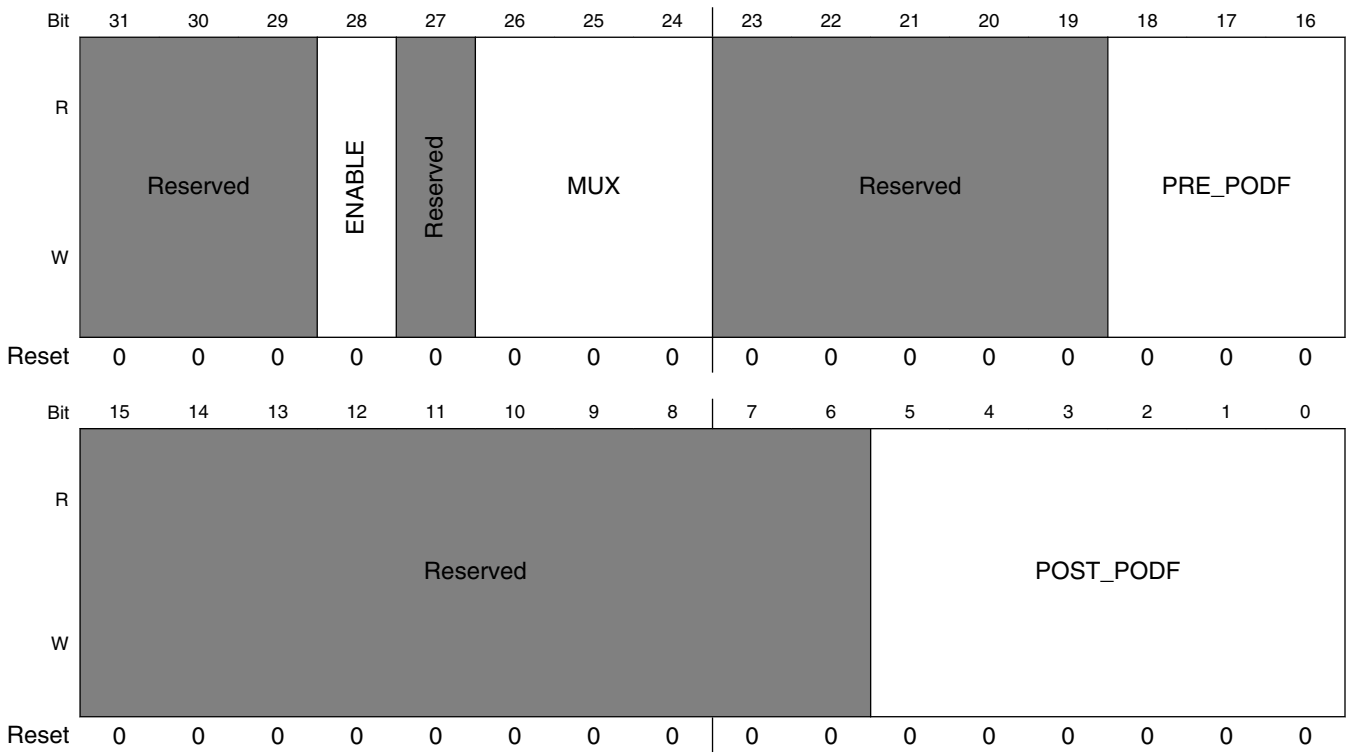
5.2.8.10 Target Register (CCM_TARGET_ROOT*n*)

See Target Interface for more information.

NOTE

See Clock Root Selects for clock root offsets and muxing information.

Address: 3038_0000h base + 8000h offset + (128d × i), where i=0d to 120d



CCM_TARGET_ROOT*n* field descriptions

Field	Description
31–29 -	This field is reserved. Reserved
28 ENABLE	Enable this clock 0 clock root is OFF 1 clock root is ON
27 -	This field is reserved. Reserved
26–24 MUX	Selection of clock sources This field is 1 bit long for DRAM and DRAM_PHYM

Table continues on the next page...

CCM_TARGET_ROOT*n* field descriptions (continued)

Field	Description
23–19 -	This field is reserved. Reserved
18–16 PRE_PODF	Pre divider divide the number Divider value is n+1 This field does not apply for CORE, DRAM, DRAM_PHYM 000 Divide by 1 001 Divide by 2 010 Divide by 3 011 Divide by 4 100 Divide by 5 101 Divide by 6 110 Divide by 7 111 Divide by 8
15–6 -	This field is reserved. Reserved
POST_PODF	Post divider divide number Divider value is n + 1. For CORE, DRAM, this field is 3 bit long. This field does not apply to DRAM_PHYM 000000 Divide by 1 000001 Divide by 2 000010 Divide by 3 000011 Divide by 4 000100 Divide by 5 000101 Divide by 6 : 111111 Divide by 64

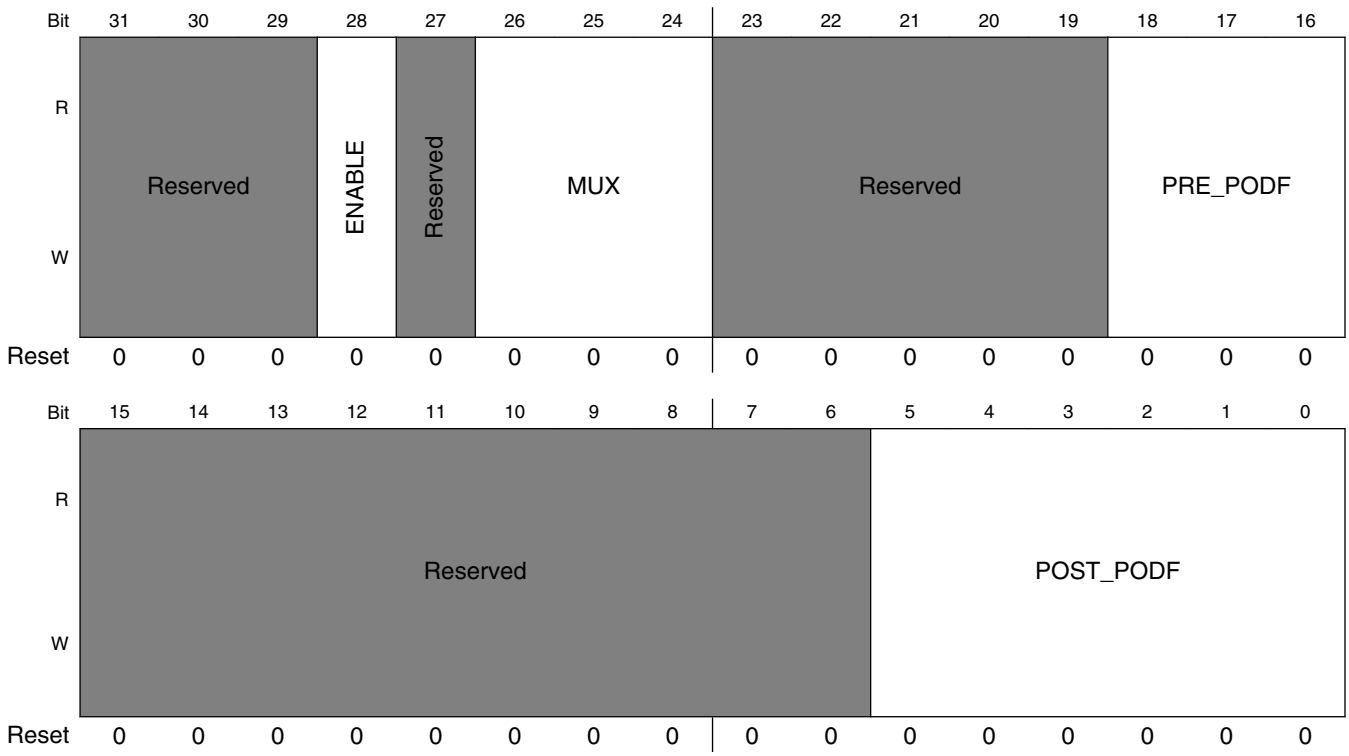
5.2.8.11 Target Register (CCM_TARGET_ROOT*n*_SET)

See Target Interface for more information.

NOTE

See Clock Root Selects for clock root offsets and muxing information.

Address: 3038_0000h base + 8004h offset + (128d × i), where i=0d to 120d



CCM_TARGET_ROOT*n*_SET field descriptions

Field	Description
31–29 -	This field is reserved. Reserved
28 ENABLE	Enable this clock 0 clock root is OFF 1 clock root is ON
27 -	This field is reserved. Reserved
26–24 MUX	Selection of clock sources This field is 1 bit long for DRAM and DRAM_PHYM

Table continues on the next page...

CCM_TARGET_ROOT*n*_SET field descriptions (continued)

Field	Description
23–19 -	This field is reserved. Reserved
18–16 PRE_PODF	Pre divider divide the number Divider value is n+1 This field does not apply for CORE, DRAM, DRAM_PHYM 000 Divide by 1 001 Divide by 2 010 Divide by 3 011 Divide by 4 100 Divide by 5 101 Divide by 6 110 Divide by 7 111 Divide by 8
15–6 -	This field is reserved. Reserved
POST_PODF	Post divider divide the number Divider value is n + 1. For CORE, DRAM, this field is 3 bit long. This field does not apply to DRAM_PHYM 000000 Divide by 1 000001 Divide by 2 000010 Divide by 3 000011 Divide by 4 000100 Divide by 5 000101 Divide by 6 : 111111 Divide by 64

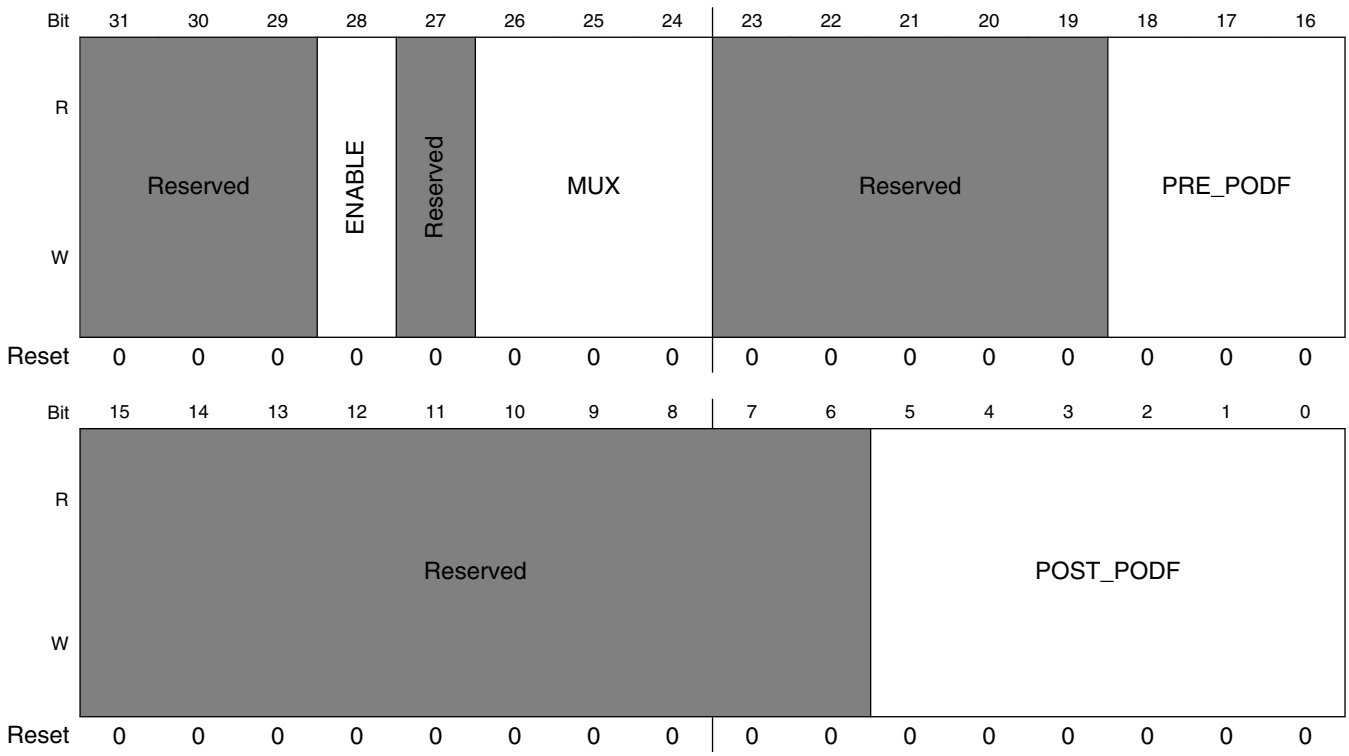
5.2.8.12 Target Register (CCM_TARGET_ROOT*n*_CLR)

See Target Interface for more information.

NOTE

See Clock Root Selects for clock root offsets and muxing information.

Address: 3038_0000h base + 8008h offset + (128d × i), where i=0d to 120d



CCM_TARGET_ROOT*n*_CLR field descriptions

Field	Description
31–29 -	This field is reserved. Reserved
28 ENABLE	Enable this clock 0 clock root is OFF 1 clock root is ON
27 -	This field is reserved. Reserved
26–24 MUX	Selection of clock sources This field is 1 bit long for DRAM and DRAM_PHYM

Table continues on the next page...

CCM_TARGET_ROOT*n*_CLR field descriptions (continued)

Field	Description
23–19 -	This field is reserved. Reserved
18–16 PRE_PODF	Pre divider divide the number Divider value is n+1 This field does not apply for CORE, DRAM, DRAM_PHYM 000 Divide by 1 001 Divide by 2 010 Divide by 3 011 Divide by 4 100 Divide by 5 101 Divide by 6 110 Divide by 7 111 Divide by 8
15–6 -	This field is reserved. Reserved
POST_PODF	Post divider divide the number Divider value is n + 1. For CORE, DRAM, this field is 3 bit long. This field does not apply to DRAM_PHYM 000000 Divide by 1 000001 Divide by 2 000010 Divide by 3 000011 Divide by 4 000100 Divide by 5 000101 Divide by 6 : 111111 Divide by 64

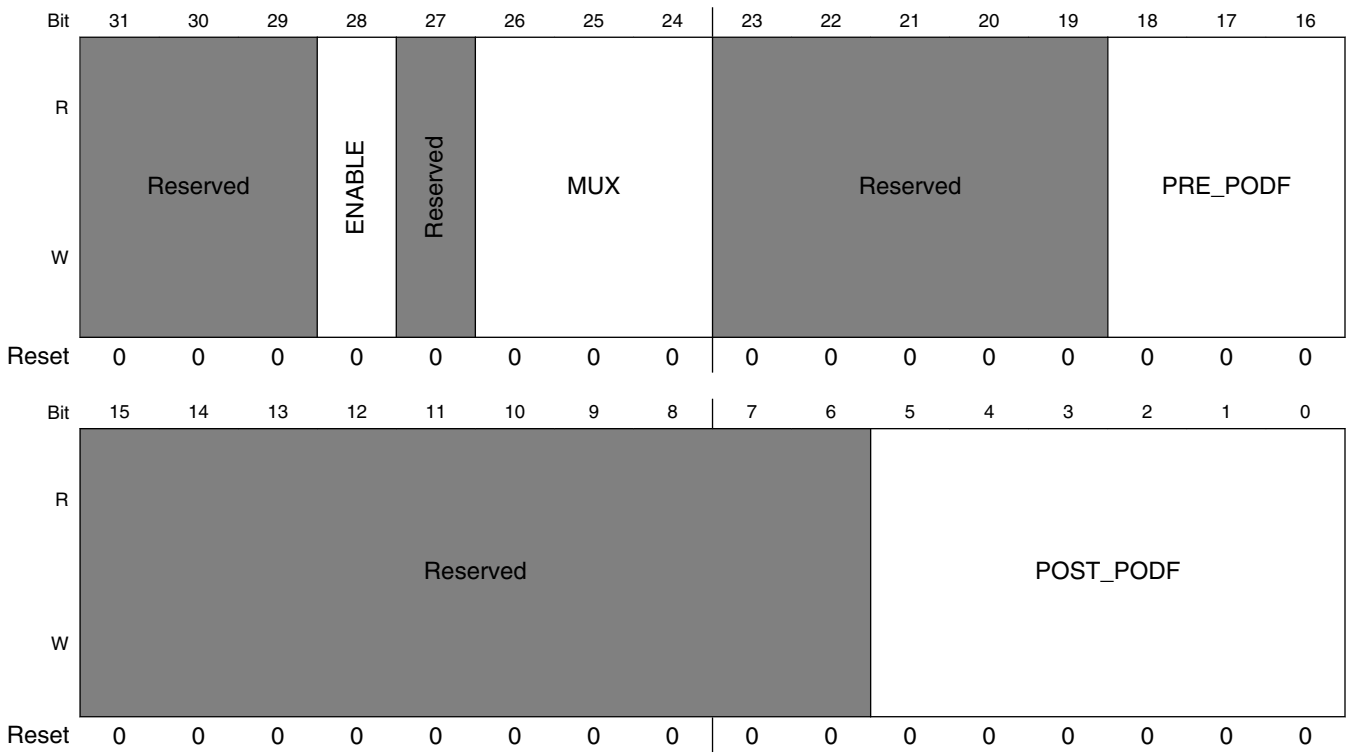
5.2.8.13 Target Register (CCM_TARGET_ROOT*n*_TOG)

See Target Interface for more information.

NOTE

See Clock Root Selects for clock root offsets and muxing information.

Address: 3038_0000h base + 800Ch offset + (128d × i), where i=0d to 120d



CCM_TARGET_ROOT*n*_TOG field descriptions

Field	Description
31–29 -	This field is reserved. Reserved
28 ENABLE	Enable this clock 0 clock root is OFF 1 clock root is ON
27 -	This field is reserved. Reserved
26–24 MUX	Selection of clock sources This field is 1 bit long for DRAM and DRAM_PHYM

Table continues on the next page...

CCM_TARGET_ROOT*n*_TOG field descriptions (continued)

Field	Description
23–19 -	This field is reserved. Reserved
18–16 PRE_PODF	Pre divide divide number Divider value is n+1 This field does not apply for CORE, DRAM, DRAM_PHYM 000 Divide by 1 001 Divide by 2 010 Divide by 3 011 Divide by 4 100 Divide by 5 101 Divide by 6 110 Divide by 7 111 Divide by 8
15–6 -	This field is reserved. Reserved
POST_PODF	Post divider divide number Divider value is n + 1. For CORE, DRAM, this field is 3 bit long. This field does not apply to DRAM_PHYM 000000 Divide by 1 000001 Divide by 2 000010 Divide by 3 000011 Divide by 4 000100 Divide by 5 000101 Divide by 6 : 111111 Divide by 64

5.2.8.14 Miscellaneous Register (CCM_MISC*n*)

MISC

Address: 3038_0000h base + 8010h offset + (128d × i), where i=0d to 120d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved							VIOLATE	Reserved			TIMEOUT	Reserved			AUTHEN_FAIL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

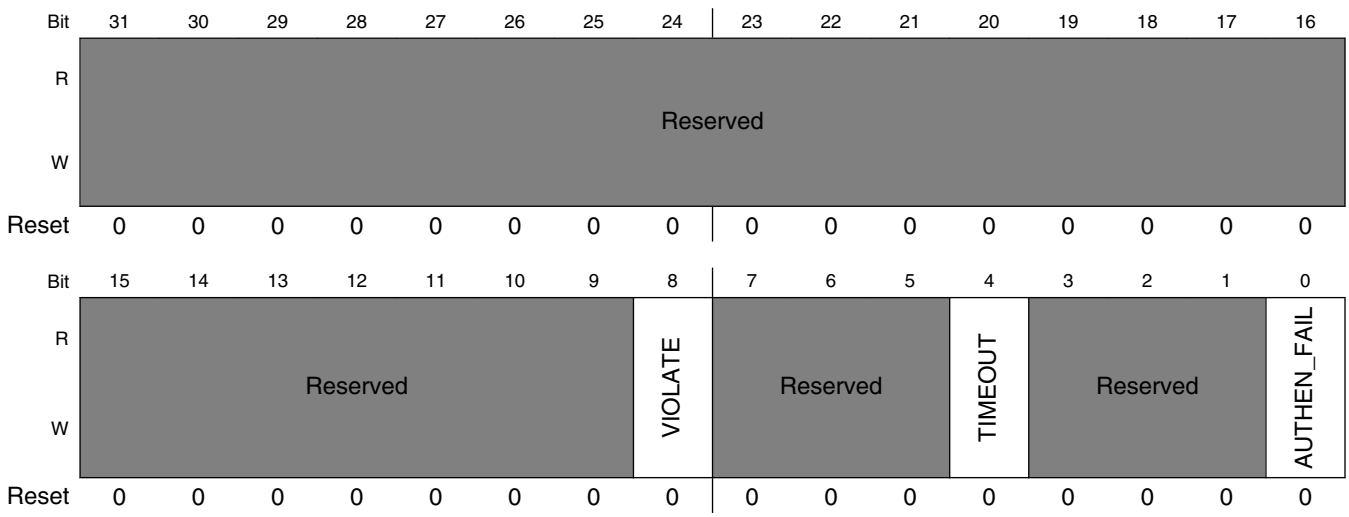
CCM_MISC*n* field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 VIOLATE	This sticky bit reflects access violation in normal interface of this clock. This bit has internal 4 bits, one for each domain. Violation from other domain is not visible or clearable. This file is cleared to 0 while write 1.
7–5 -	This field is reserved. Reserved
4 TIMEOUT	This sticky bit reflects time out happened during accessing this clock. This bit has internal 4 bits, one for each domain. Timeout from other domain is not visible or clearable. This file is cleared to 0 while write 1.
3–1 -	This field is reserved. Reserved
0 AUTHEN_FAIL	This sticky bit reflects access restricted by access control of this clock. This bit has internal 4 bits, one for each domain. Authentic fail from other domain is not visible or clearable. This file is cleared to 0 while write 1

5.2.8.15 Miscellaneous Register (CCM_MISC_ROOT*n*_SET)

Misc

Address: 3038_0000h base + 8014h offset + (128d × i), where i=0d to 120d



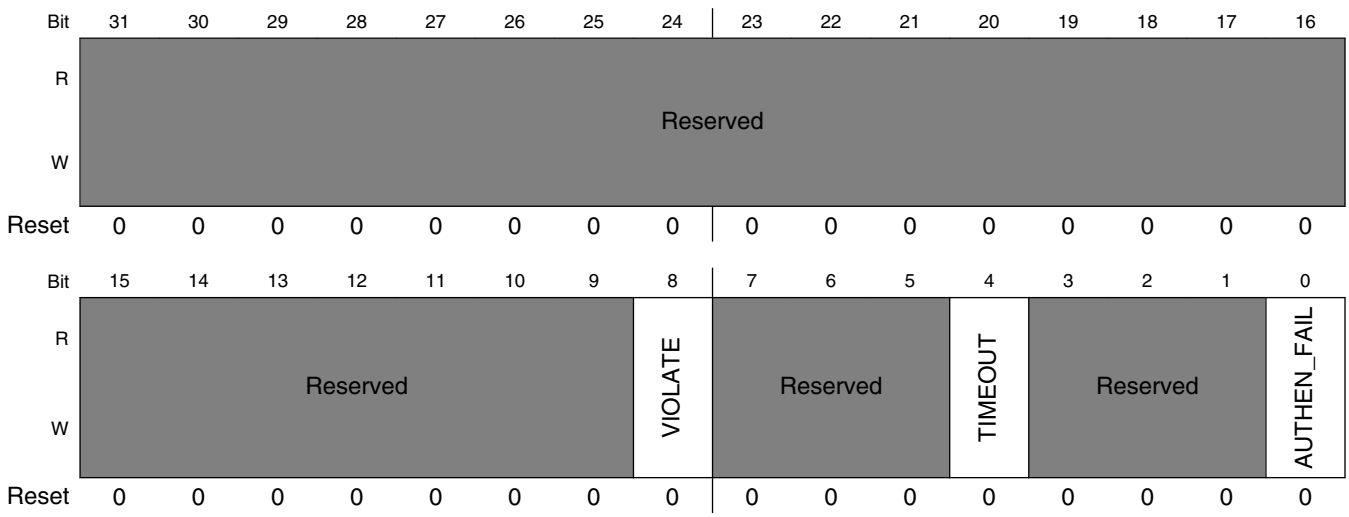
CCM_MISC_ROOT*n*_SET field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 VIOLATE	This sticky bit reflects access violation in normal interface of this clock. This bit has internal 4 bits, one for each domain. Violation from other domain is not visible or clearable. This file is cleared to 0 while write 1.
7–5 -	This field is reserved. Reserved
4 TIMEOUT	This sticky bit reflects time out happened during accessing this clock. This bit has internal 4 bits, one for each domain. Timeout from other domain is not visible or clearable. This file is cleared to 0 while write 1.
3–1 -	This field is reserved. Reserved
0 AUTHEN_FAIL	This sticky bit reflects access restricted by access control of this clock. This bit has internal 4 bits, one for each domain. Authentic fail from other domain is not visible or clearable. This file is cleared to 0 while write 1

5.2.8.16 Miscellaneous Register (CCM_MISC_ROOT*n*_CLR)

MISC

Address: 3038_0000h base + 8018h offset + (128d × i), where i=0d to 120d



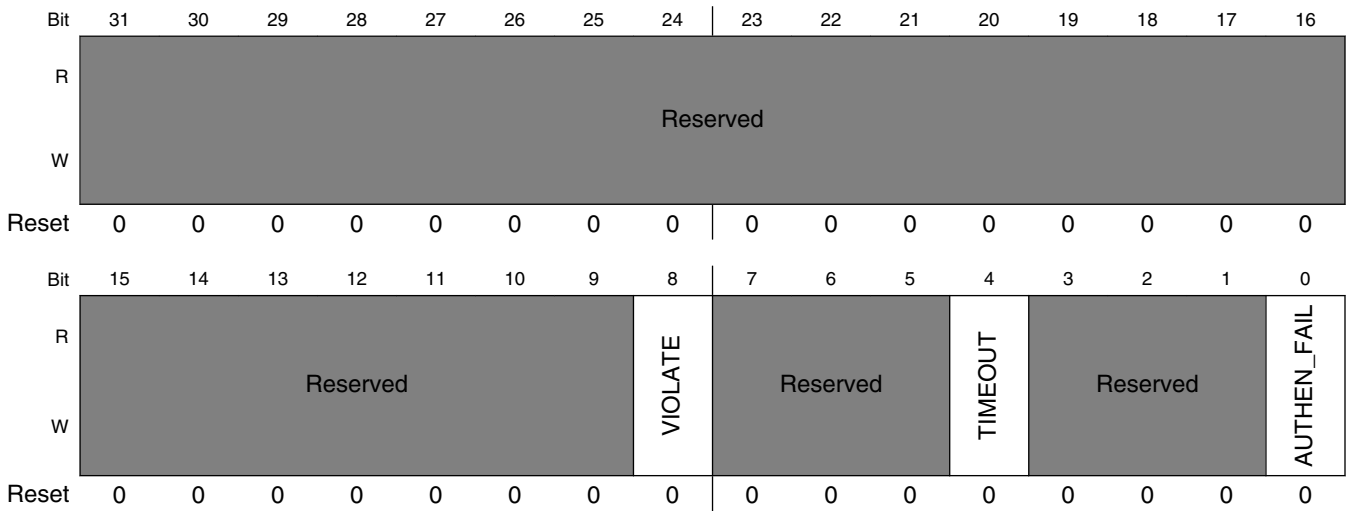
CCM_MISC_ROOT*n*_CLR field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 VIOLATE	This sticky bit reflects access violation in normal interface of this clock. This bit has internal 4 bits, one for each domain. Violation from other domain is not visible or clearable. This file is cleared to 0 while write 1.
7–5 -	This field is reserved. Reserved
4 TIMEOUT	This sticky bit reflects time out happened during accessing this clock. This bit has internal 4 bits, one for each domain. Timeout from other domain is not visible or clearable. This file is cleared to 0 while write 1.
3–1 -	This field is reserved. Reserved
0 AUTHEN_FAIL	This sticky bit reflects access restricted by access control of this clock. This bit has internal 4 bits, one for each domain. Authentic fail from other domain is not visible or clearable. This file is cleared to 0 while write 1

5.2.8.17 Miscellaneous Register (CCM_MISC_ROOT*n*_TOG)

MISC

Address: 3038_0000h base + 801Ch offset + (128d × i), where i=0d to 120d



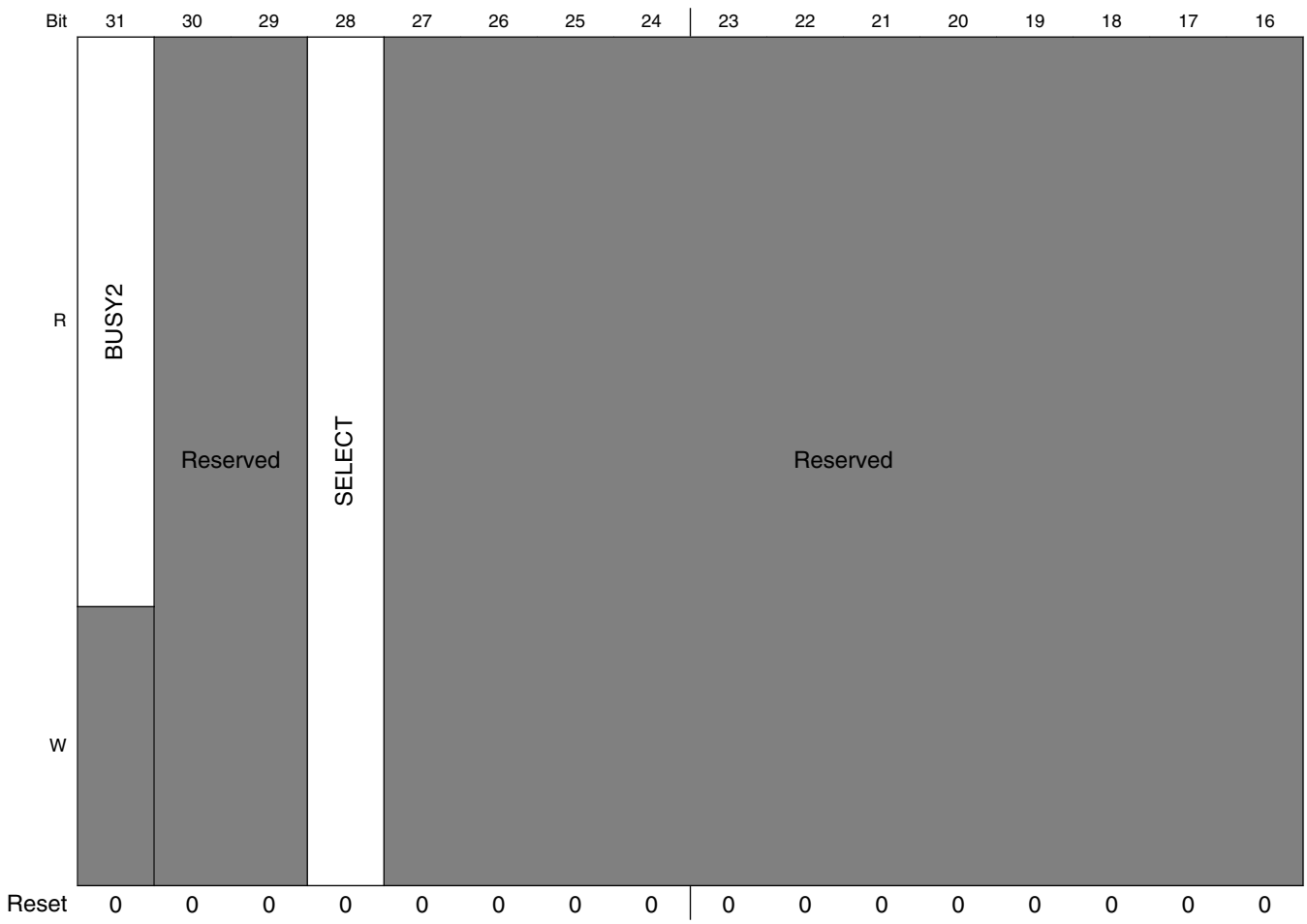
CCM_MISC_ROOT*n*_TOG field descriptions

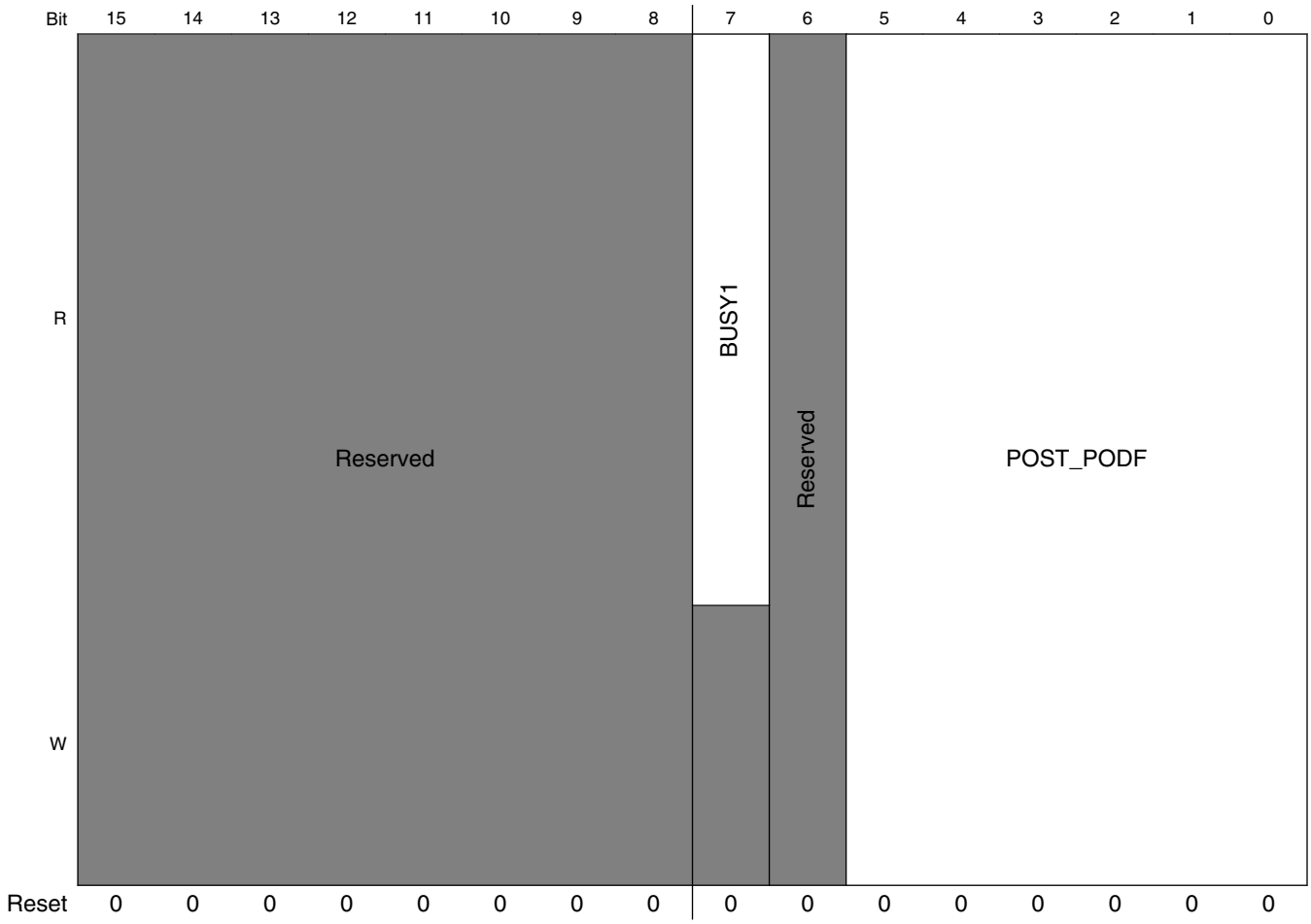
Field	Description
31–9 -	This field is reserved. Reserved
8 VIOLATE	This sticky bit reflects access violation in normal interface of this clock. This bit has internal 4 bits, one for each domain. Violation from other domain is not visible or clearable. This file is cleared to 0 while write 1.
7–5 -	This field is reserved. Reserved
4 TIMEOUT	This sticky bit reflects time out happened during accessing this clock. This bit has internal 4 bits, one for each domain. Timeout from other domain is not visible or clearable. This file is cleared to 0 while write 1.
3–1 -	This field is reserved. Reserved
0 AUTHEN_FAIL	This sticky bit reflects access restricted by access control of this clock. This bit has internal 4 bits, one for each domain. Authentic fail from other domain is not visible or clearable. This file is cleared to 0 while write 1

5.2.8.18 Post Divider Register (CCM_POST*n*)

Post Register

Address: 3038_0000h base + 8020h offset + (128d × i), where i=0d to 120d





CCM_POST*n* field descriptions

Field	Description
31 BUSY2	Safe multiplexer is applying new setting
30–29 -	This field is reserved. Reserved
28 SELECT	Selection of pre clock branches This field is not applied in IP 0 select branch A 1 select branch B
27–8 -	This field is reserved. Reserved
7 BUSY1	Post divider is applying new set value
6 -	This field is reserved. Reserved
POST_PODF	Post divider divide the number Divider value is n + 1

Table continues on the next page...

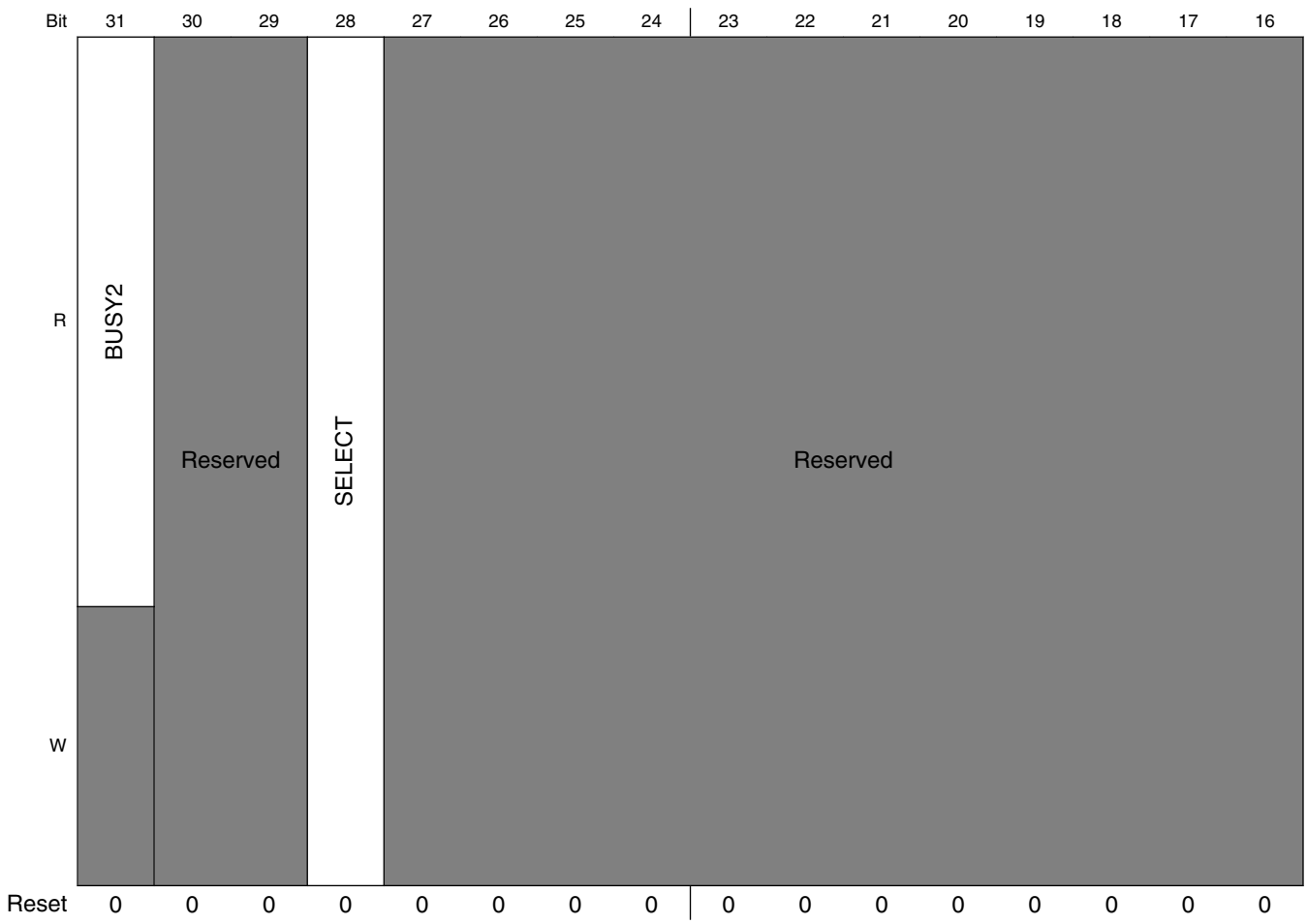
CCM_POST*n* field descriptions (continued)

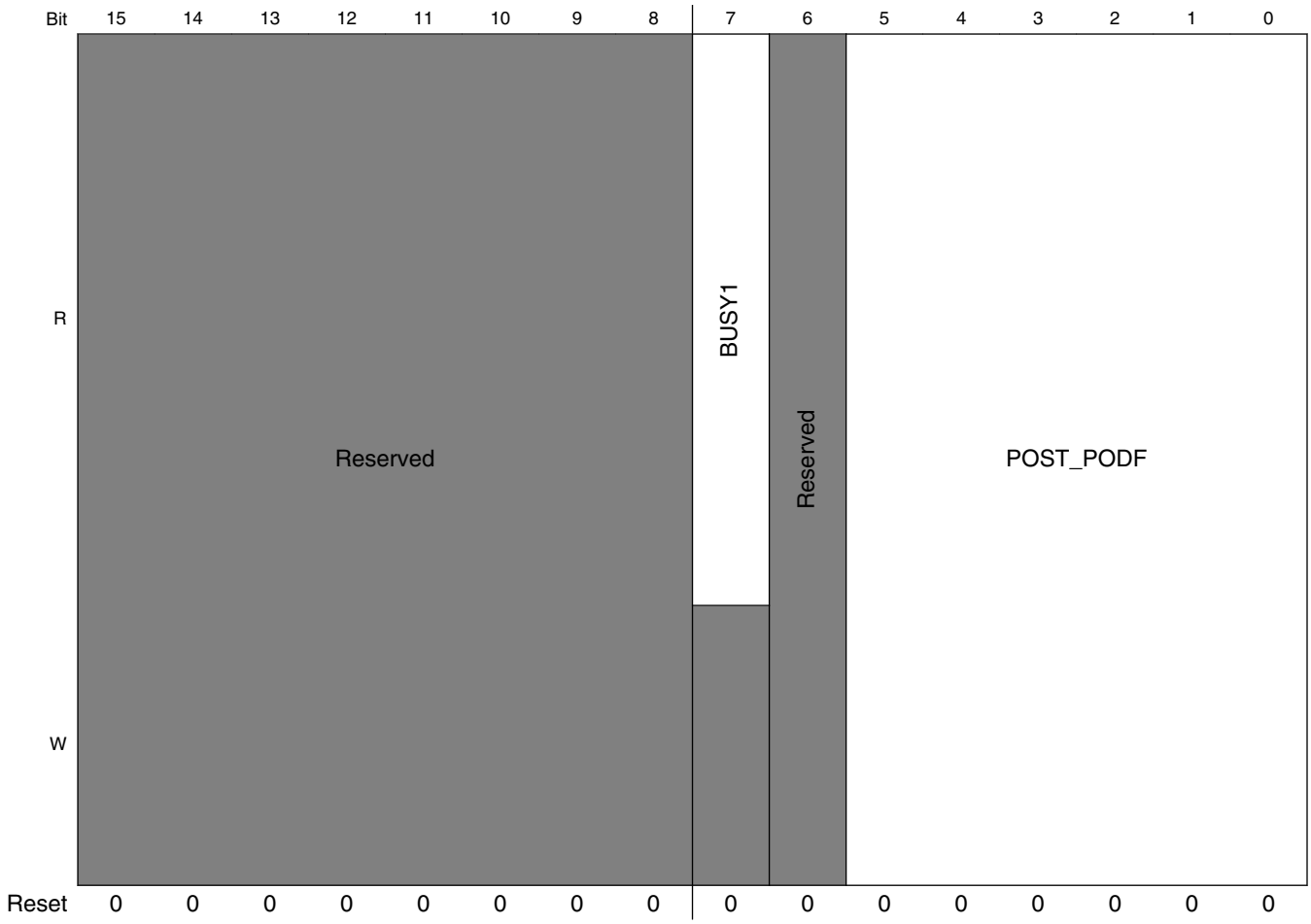
Field	Description
	For CORE, DRAM, this field is 3 bit long. This field does not apply to DRAM_PHYM 000000 Divide by 1 000001 Divide by 2 000010 Divide by 3 000011 Divide by 4 000100 Divide by 5 000101 Divide by 6 : 111111 Divide by 64

5.2.8.19 Post Divider Register (CCM_POST_ROOT*n*_SET)

Post Divider Register

Address: 3038_0000h base + 8024h offset + (128d × i), where i=0d to 120d





CCM_POST_ROOT*n*_SET field descriptions

Field	Description
31 BUSY2	Safe multiplexer is applying new setting
30–29 -	This field is reserved. Reserved
28 SELECT	Selection of pre clock branches This field is not applied in IP 0 select branch A 1 select branch B
27–8 -	This field is reserved. Reserved
7 BUSY1	Post divider is applying new set value
6 -	This field is reserved. Reserved
POST_PODF	Post divider divide number Divider value is n + 1

Table continues on the next page...

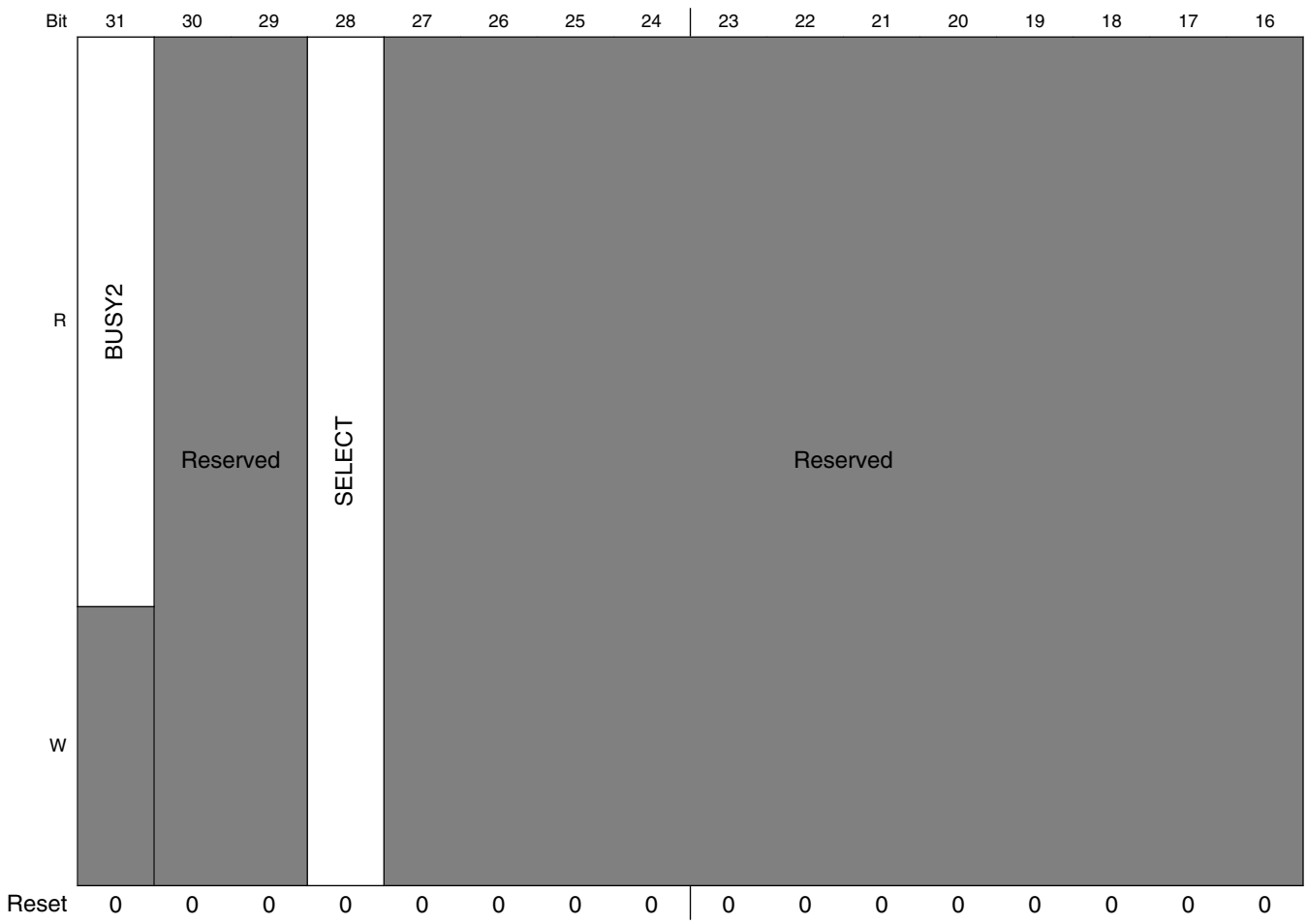
CCM_POST_ROOT*n*_SET field descriptions (continued)

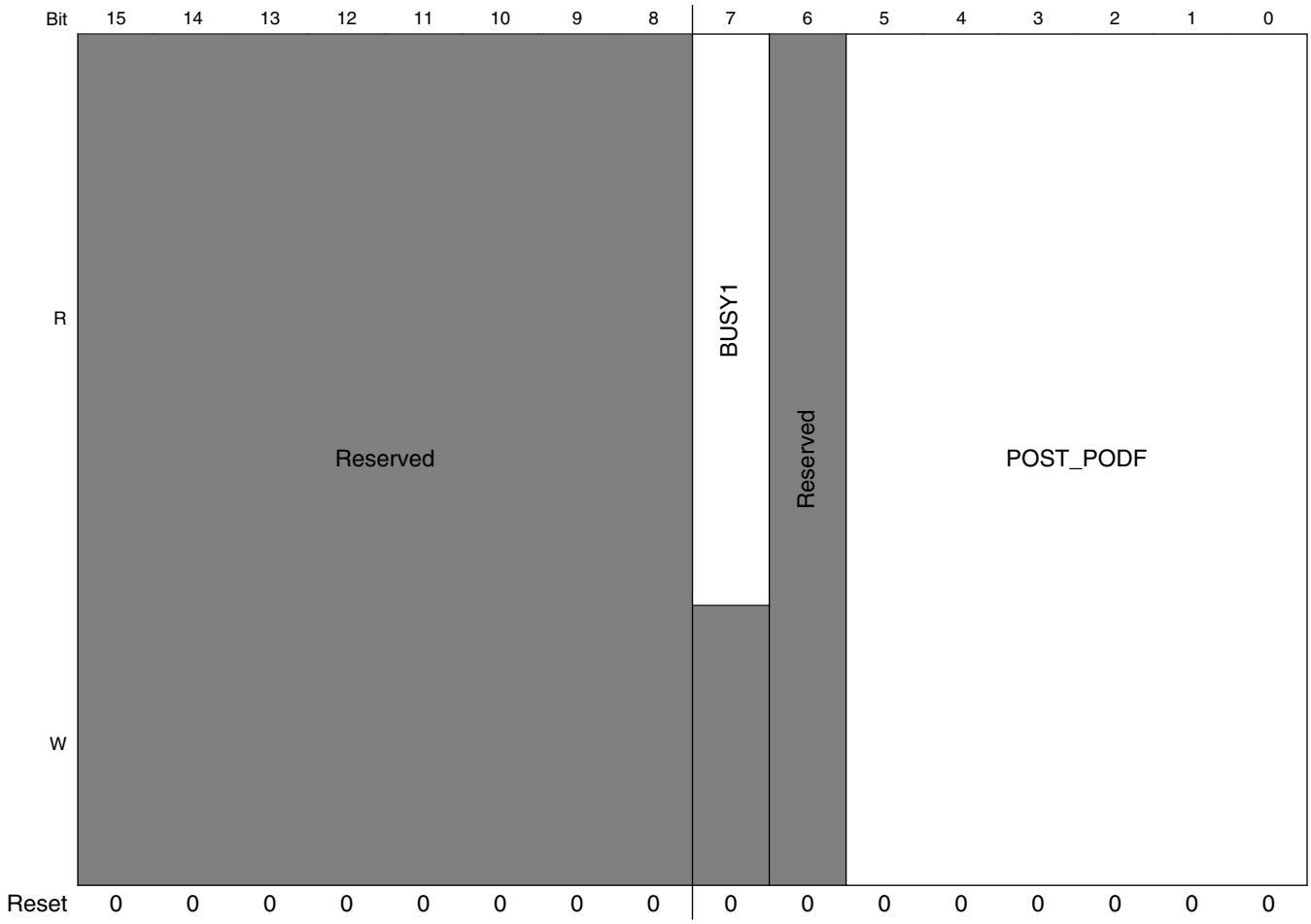
Field	Description
	For CORE, DRAM, this field is 3 bit long. This field does not apply to DRAM_PHYM 000000 Divide by 1 000001 Divide by 2 000010 Divide by 3 000011 Divide by 4 000100 Divide by 5 000101 Divide by 6 : 111111 Divide by 64

5.2.8.20 Post Divider Register (CCM_POST_ROOT*n*_CLR)

Post Root Register

Address: 3038_0000h base + 8028h offset + (128d × i), where i=0d to 120d





CCM_POST_ROOT*n*_CLR field descriptions

Field	Description
31 BUSY2	Safe multiplexer is applying new setting
30–29 -	This field is reserved. Reserved
28 SELECT	Selection of pre clock branches This field is not applied in IP 0 select branch A 1 select branch B
27–8 -	This field is reserved. Reserved
7 BUSY1	Post divider is applying new set value
6 -	This field is reserved. Reserved
POST_PODF	Post divider divide the number Divider value is n + 1

Table continues on the next page...

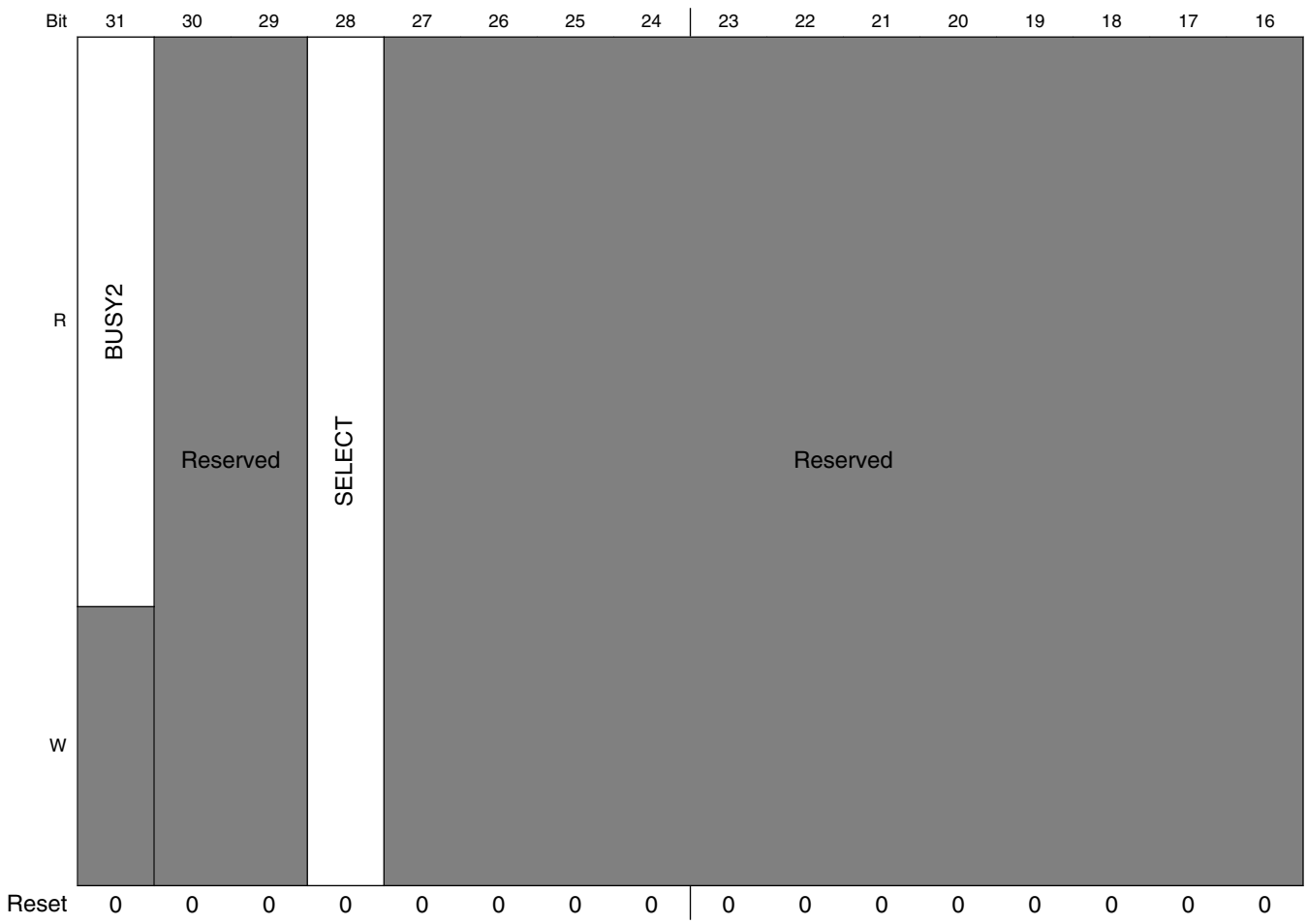
CCM_POST_ROOT*n*_CLR field descriptions (continued)

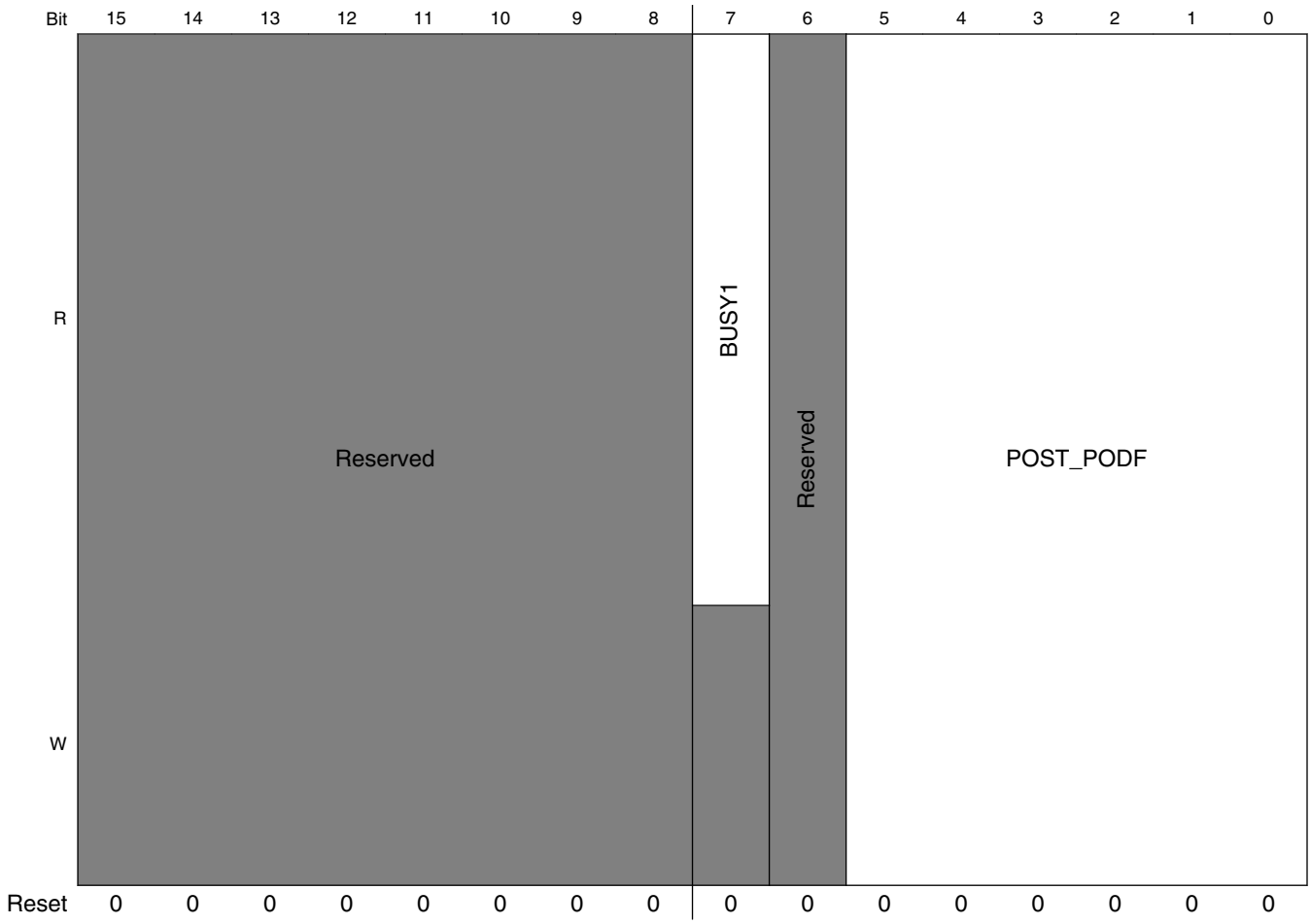
Field	Description
	For CORE, DRAM, this field is 3 bit long. This field does not apply to DRAM_PHYM 000000 Divide by 1 000001 Divide by 2 000010 Divide by 3 000011 Divide by 4 000100 Divide by 5 000101 Divide by 6 : 111111 Divide by 64

5.2.8.21 Post Divider Register (CCM_POST_ROOT*n*_TOG)

Post Root Register

Address: 3038_0000h base + 802Ch offset + (128d × i), where i=0d to 120d





CCM_POST_ROOT*n*_TOG field descriptions

Field	Description
31 BUSY2	Safe multiplexer is applying new setting
30–29 -	This field is reserved. Reserved
28 SELECT	Selection of pre clock branches This field is not applied in IP 0 select branch A 1 select branch B
27–8 -	This field is reserved. Reserved
7 BUSY1	Post divider is applying new set value
6 -	This field is reserved. Reserved
POST_PODF	Post divider divide number Divider value is n + 1

Table continues on the next page...

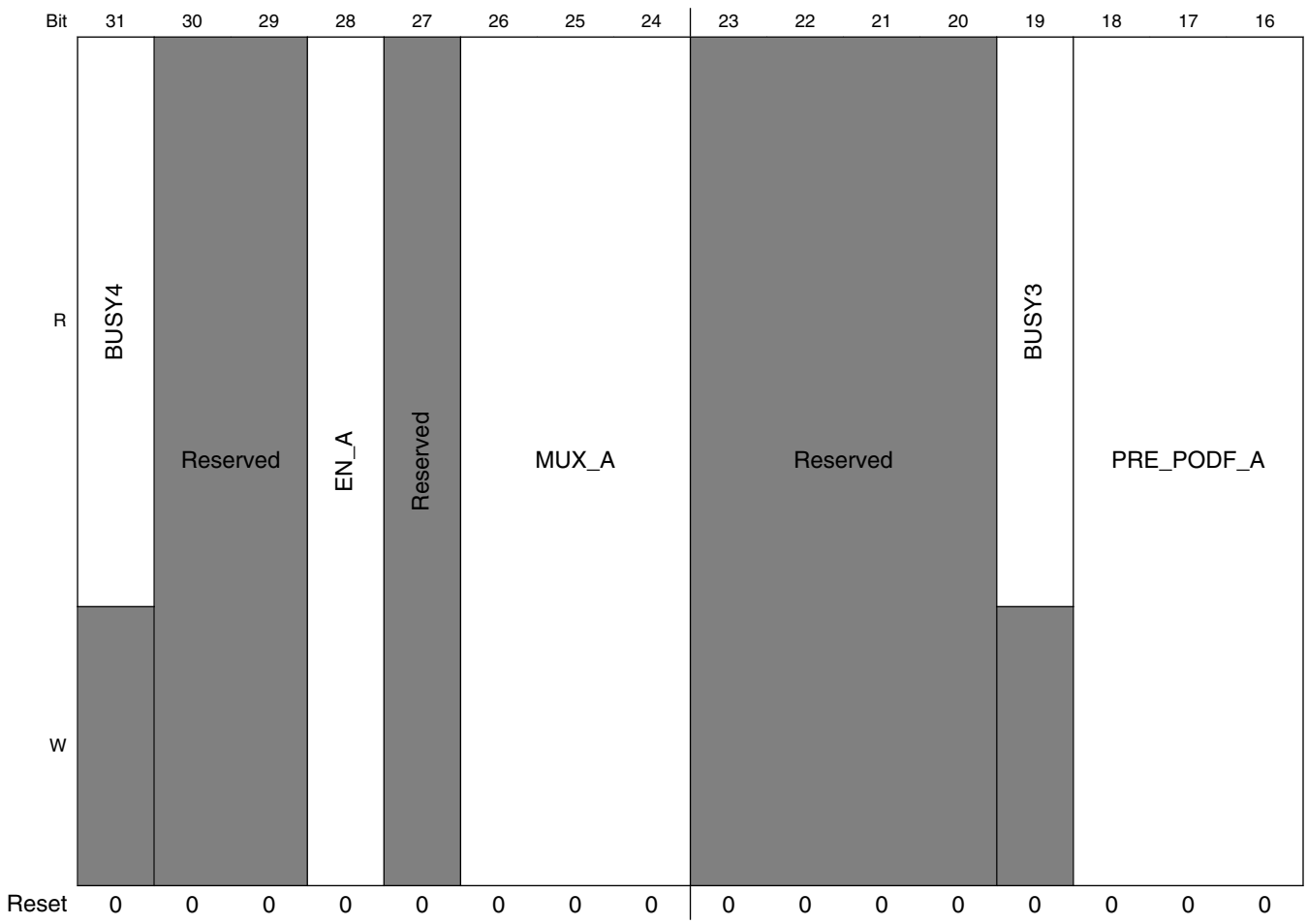
CCM_POST_ROOT*n*_TOG field descriptions (continued)

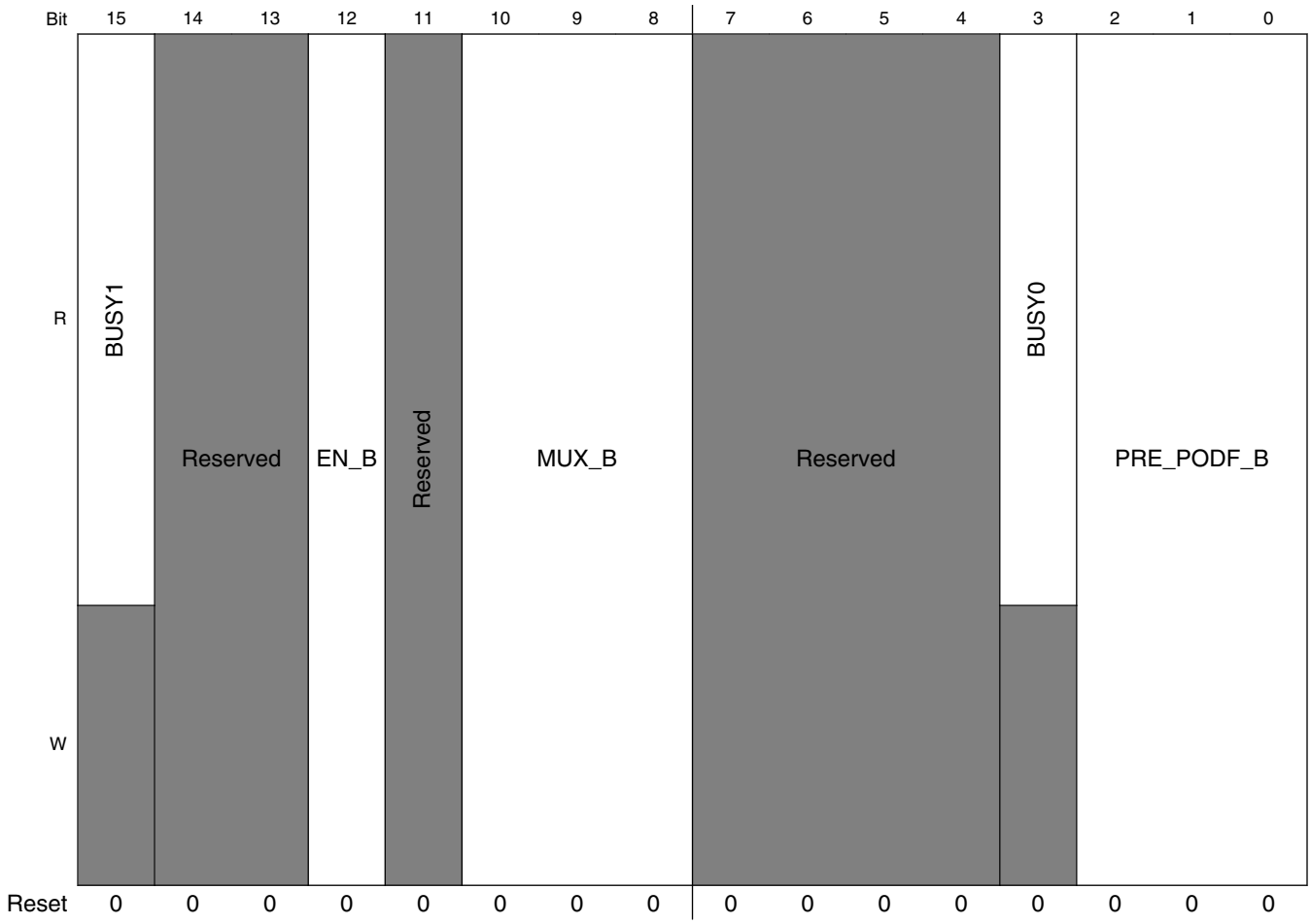
Field	Description
	For CORE, DRAM, this field is 3 bit long. This field does not apply to DRAM_PHYM 000000 Divide by 1 000001 Divide by 2 000010 Divide by 3 000011 Divide by 4 000100 Divide by 5 000101 Divide by 6 : 111111 Divide by 64

5.2.8.22 Pre Divider Register (CCM_PRE*n*)

Pre Register

Address: 3038_0000h base + 8030h offset + (128d × i), where i=0d to 120d





CCM_PRE*n* field descriptions

Field	Description
31 BUSY4	EN_A field is applied to field This field applies to DRAM and DRAM_PHYM
30–29 -	This field is reserved. Reserved
28 EN_A	Branch A clock gate control This field applies to DRAM and DRAM_PHYM 0 Clock shutdown 1 clock ON
27 -	This field is reserved. Reserved
26–24 MUX_A	Selection control of multiplexer of branch A This field applies to DRAM and DRAM_PHYM
23–20 -	This field is reserved. Reserved
19 BUSY3	Pre divider value for branch A is applied This field applies to DRAM and DRAM_PHYM

Table continues on the next page...

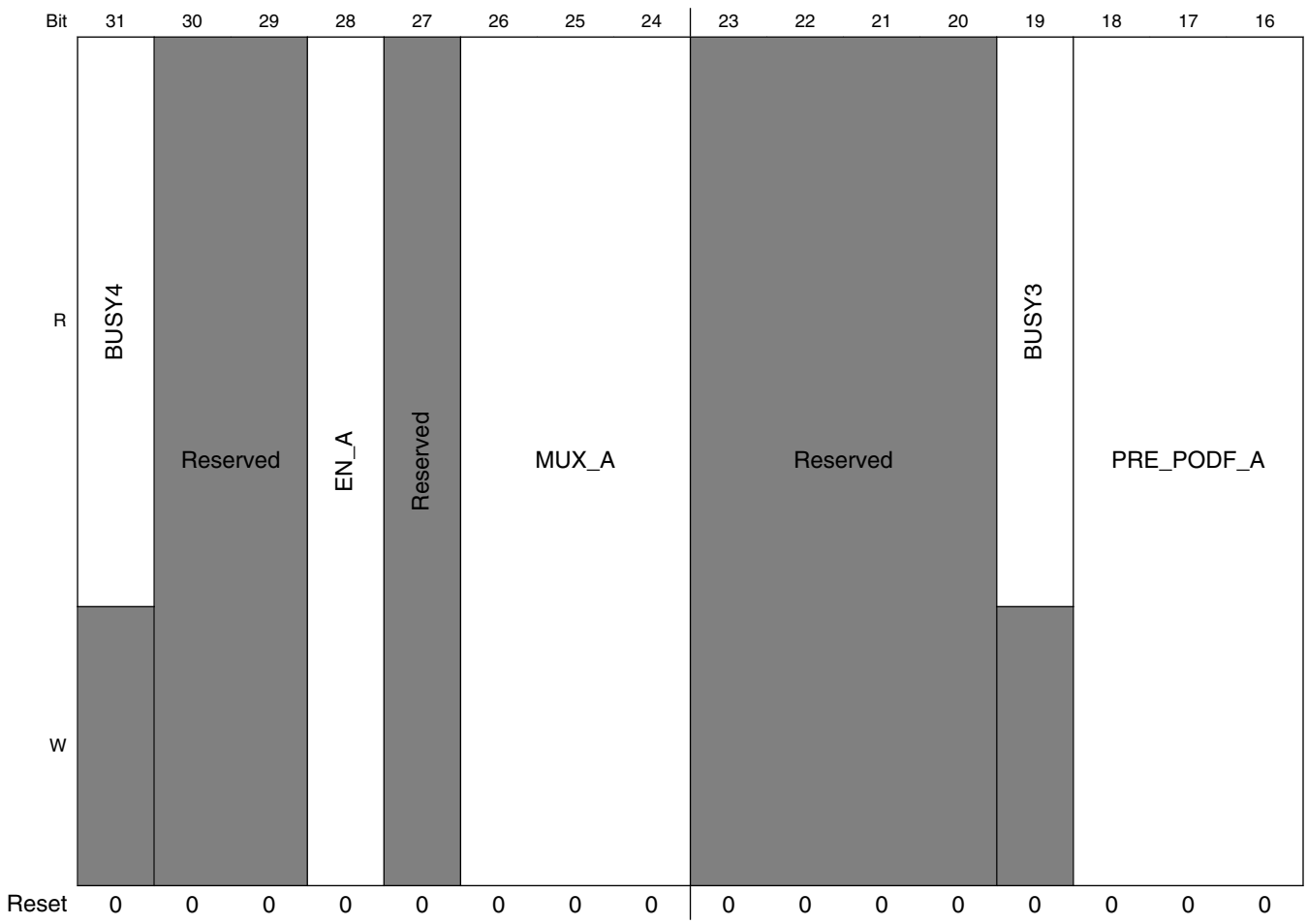
CCM_PRE*n* field descriptions (continued)

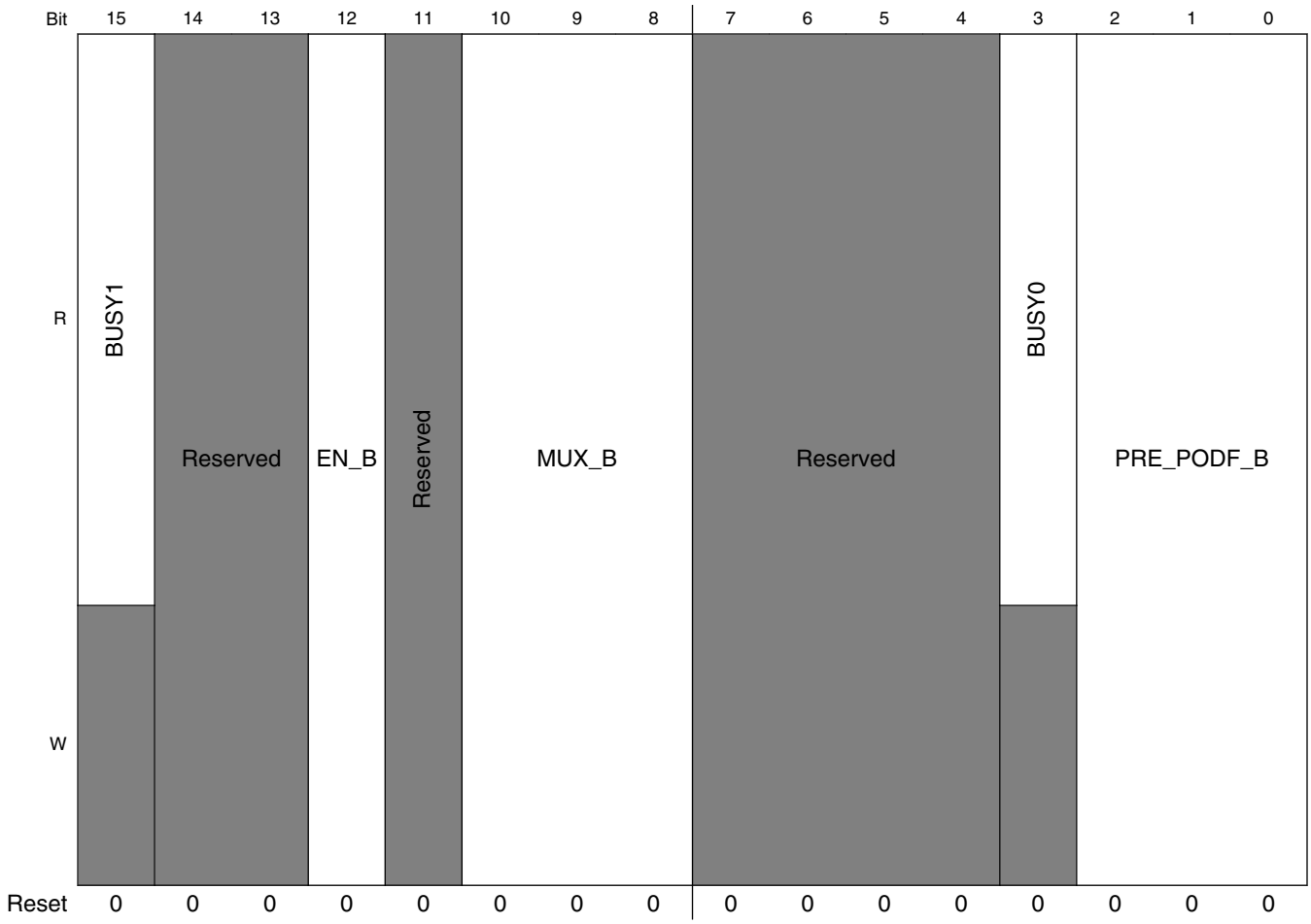
Field	Description
18–16 PRE_PODF_A	Pre divider divide number for branch A Divider value is n + 1. This field does not apply for CORE, DRAM, DRAM_PHYM 000 Divide by 1 001 Divide by 2 010 Divide by 3 011 Divide by 4 100 Divide by 5 101 Divide by 6 110 Divide by 7 111 Divide by 8
15 BUSY1	EN_B is applied to field This field does not apply for CORE, IP,DRAM, DRAM_PHYM
14–13 -	This field is reserved. Reserved
12 EN_B	Branch B clock gate control This field does not apply for CORE, IP,DRAM, DRAM_PHYM 0 Clock shutdown 1 Clock ON
11 -	This field is reserved. Reserved
10–8 MUX_B	Selection control of multiplexer of branch B This field does not apply for CORE, IP,DRAM, DRAM_PHYM
7–4 -	This field is reserved. Reserved
3 BUSY0	Pre divider value for branch a is applying field does not apply for CORE, IP,DRAM, DRAM_PHYM
PRE_PODF_B	Pre divider divide number for branch B Divider value is n + 1. This field does not apply for CORE, IP, DRAM, DRAM_PHYM 000 Divide by 1 001 Divide by 2 010 Divide by 3 011 Divide by 4 100 Divide by 5 101 Divide by 6 110 Divide by 7 111 Divide by 8

5.2.8.23 Pre Divider Register (CCM_PRE_ROOT*n*_SET)

Pre Divider Register

Address: 3038_0000h base + 8034h offset + (128d × i), where i=0d to 120d





CCM_PRE_ROOT*n*_SET field descriptions

Field	Description
31 BUSY4	EN_A field is applied to field This field applies to DRAM and DRAM_PHYM
30–29 -	This field is reserved. Reserved
28 EN_A	Branch A clock gate control This field applies to DRAM and DRAM_PHYM 0 Clock shutdown 1 clock ON
27 -	This field is reserved. Reserved
26–24 MUX_A	Selection control of multiplexer of branch A This field applies to DRAM and DRAM_PHYM
23–20 -	This field is reserved. Reserved
19 BUSY3	Pre divider value for branch A is applied This field applies to DRAM and DRAM_PHYM

Table continues on the next page...

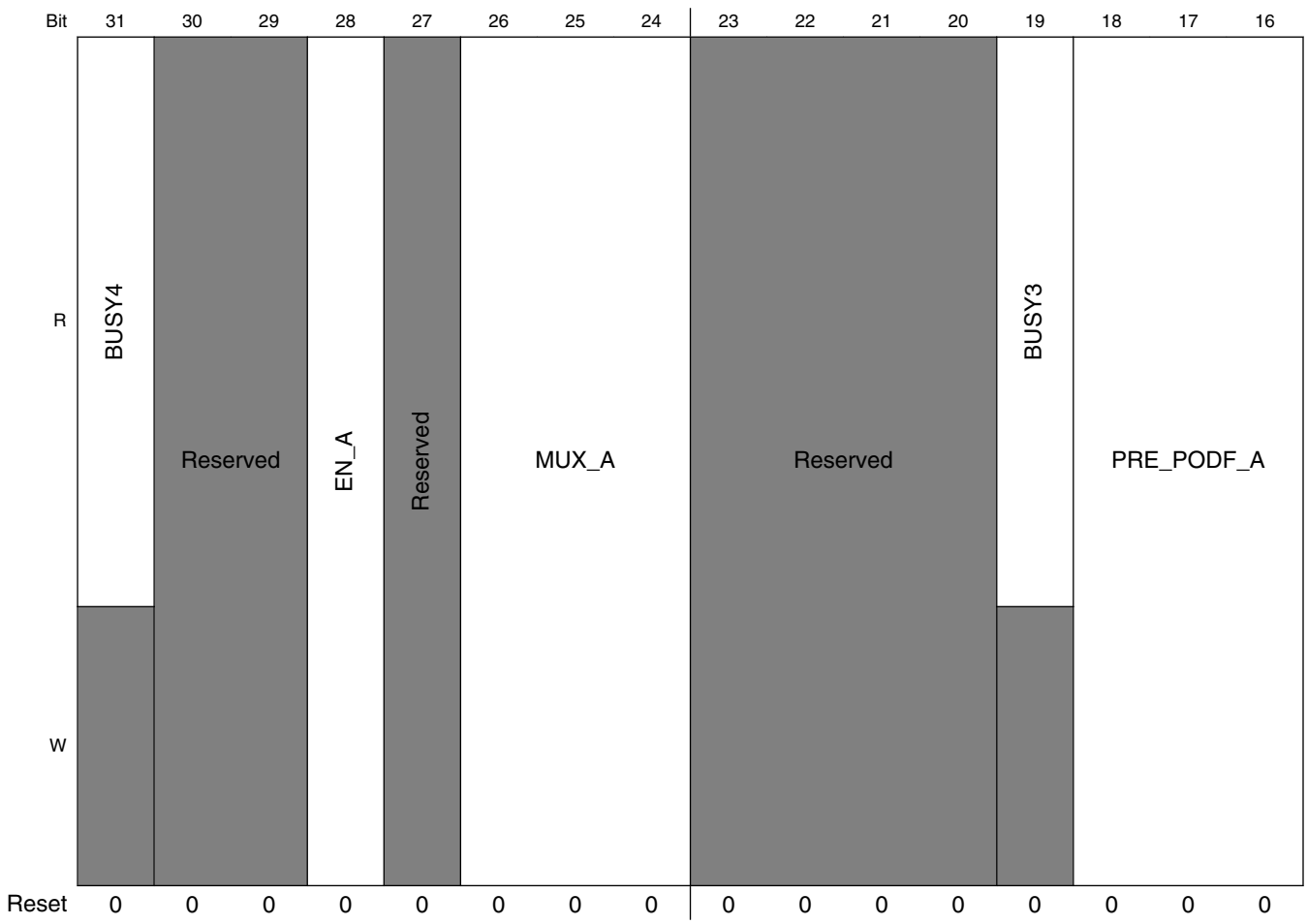
CCM_PRE_ROOT*n*_SET field descriptions (continued)

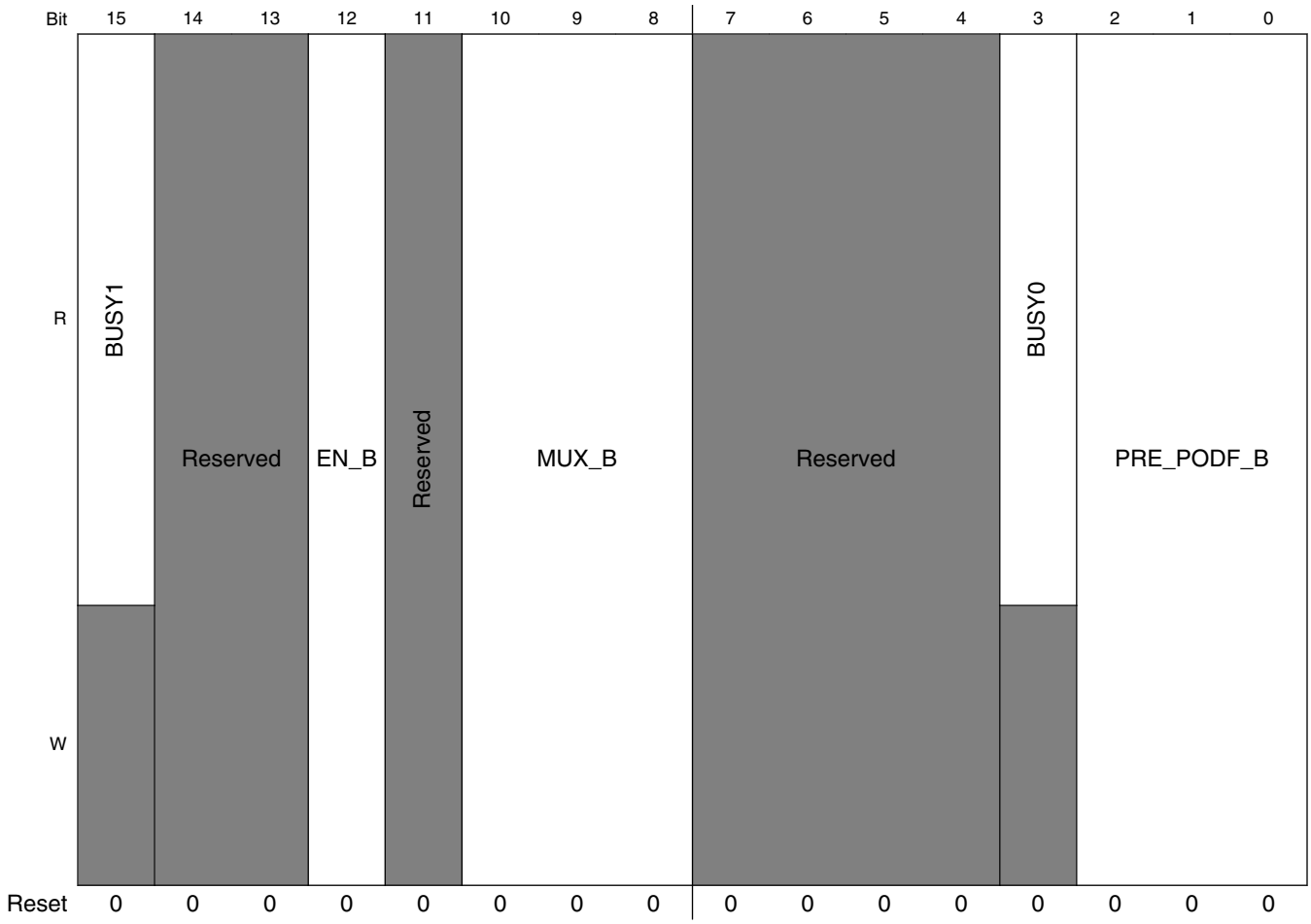
Field	Description
18–16 PRE_PODF_A	Pre divider divide number for branch A Divider value is n + 1. This field does not apply for CORE, DRAM, DRAM_PHYM 000 Divide by 1 001 Divide by 2 010 Divide by 3 011 Divide by 4 100 Divide by 5 101 Divide by 6 110 Divide by 7 111 Divide by 8
15 BUSY1	EN_B is applied to field This field does not apply for CORE, IP,DRAM, DRAM_PHYM
14–13 -	This field is reserved. Reserved
12 EN_B	Branch B clock gate control This field does not apply for CORE, IP,DRAM, DRAM_PHYM 0 Clock shutdown 1 Clock ON
11 -	This field is reserved. Reserved
10–8 MUX_B	Selection control of multiplexer of branch B This field does not apply for CORE, IP,DRAM, DRAM_PHYM
7–4 -	This field is reserved. Reserved
3 BUSY0	Pre divider value for branch A is applying field does not apply for CORE, IP,DRAM, DRAM_PHYM
PRE_PODF_B	Pre divider divide number for branch B Divider value is n + 1. This field does not apply for CORE, IP, DRAM, DRAM_PHYM 000 Divide by 1 001 Divide by 2 010 Divide by 3 011 Divide by 4 100 Divide by 5 101 Divide by 6 110 Divide by 7 111 Divide by 8

5.2.8.24 Pre Divider Register (CCM_PRE_ROOT*n*_CLR)

Pree Root Register

Address: 3038_0000h base + 8038h offset + (128d × i), where i=0d to 120d





CCM_PRE_ROOT*n*_CLR field descriptions

Field	Description
31 BUSY4	EN_A field is applied to field This field applies to DRAM and DRAM_PHYM
30–29 -	This field is reserved. Reserved
28 EN_A	Branch A clock gate control This field applies to DRAM and DRAM_PHYM 0 Clock shutdown 1 clock ON
27 -	This field is reserved. Reserved
26–24 MUX_A	Selection control of multiplexer of branch A This field applies to DRAM and DRAM_PHYM
23–20 -	This field is reserved. Reserved
19 BUSY3	Pre divider value for branch A is applied This field applies to DRAM and DRAM_PHYM

Table continues on the next page...

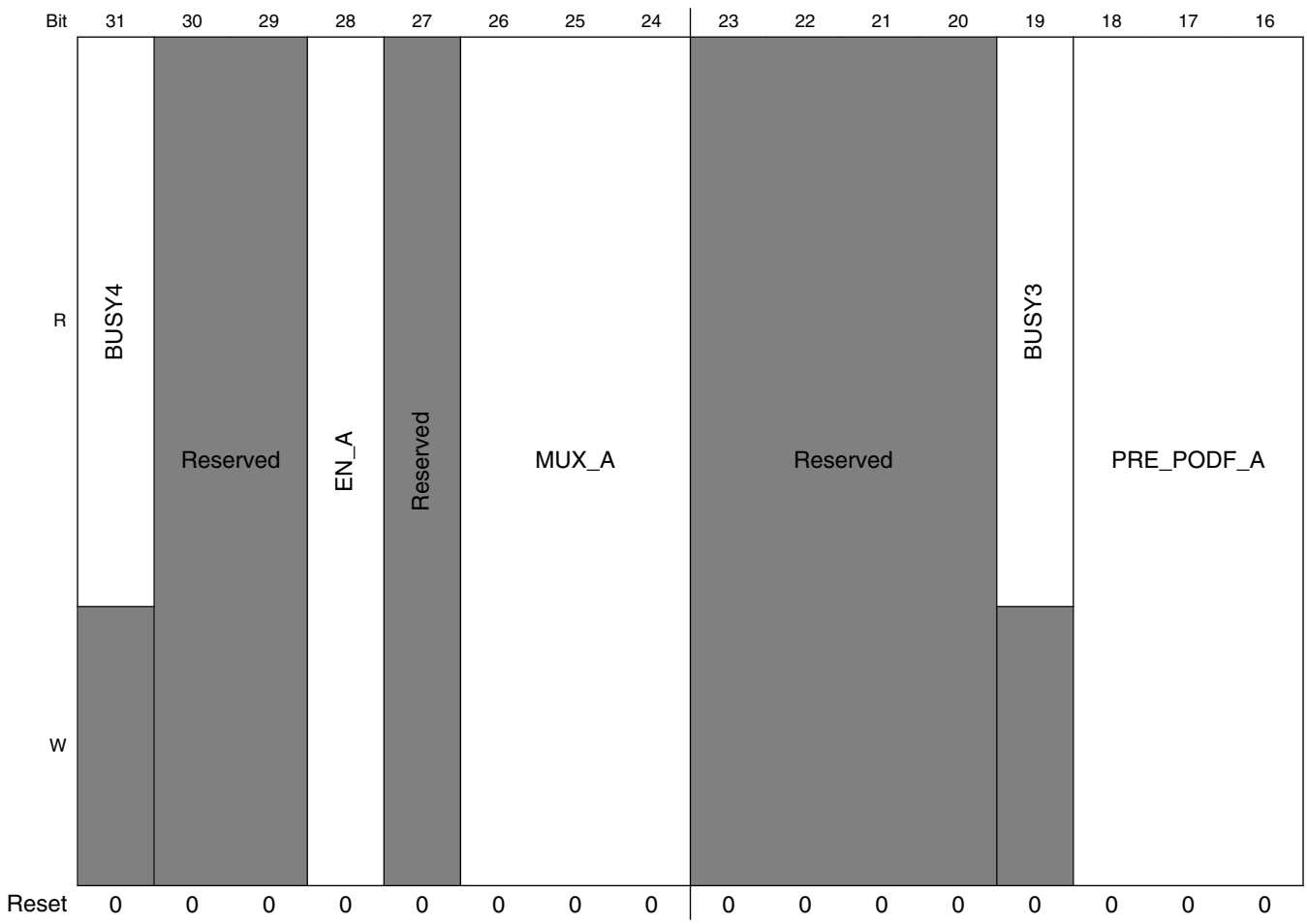
CCM_PRE_ROOT*n*_CLR field descriptions (continued)

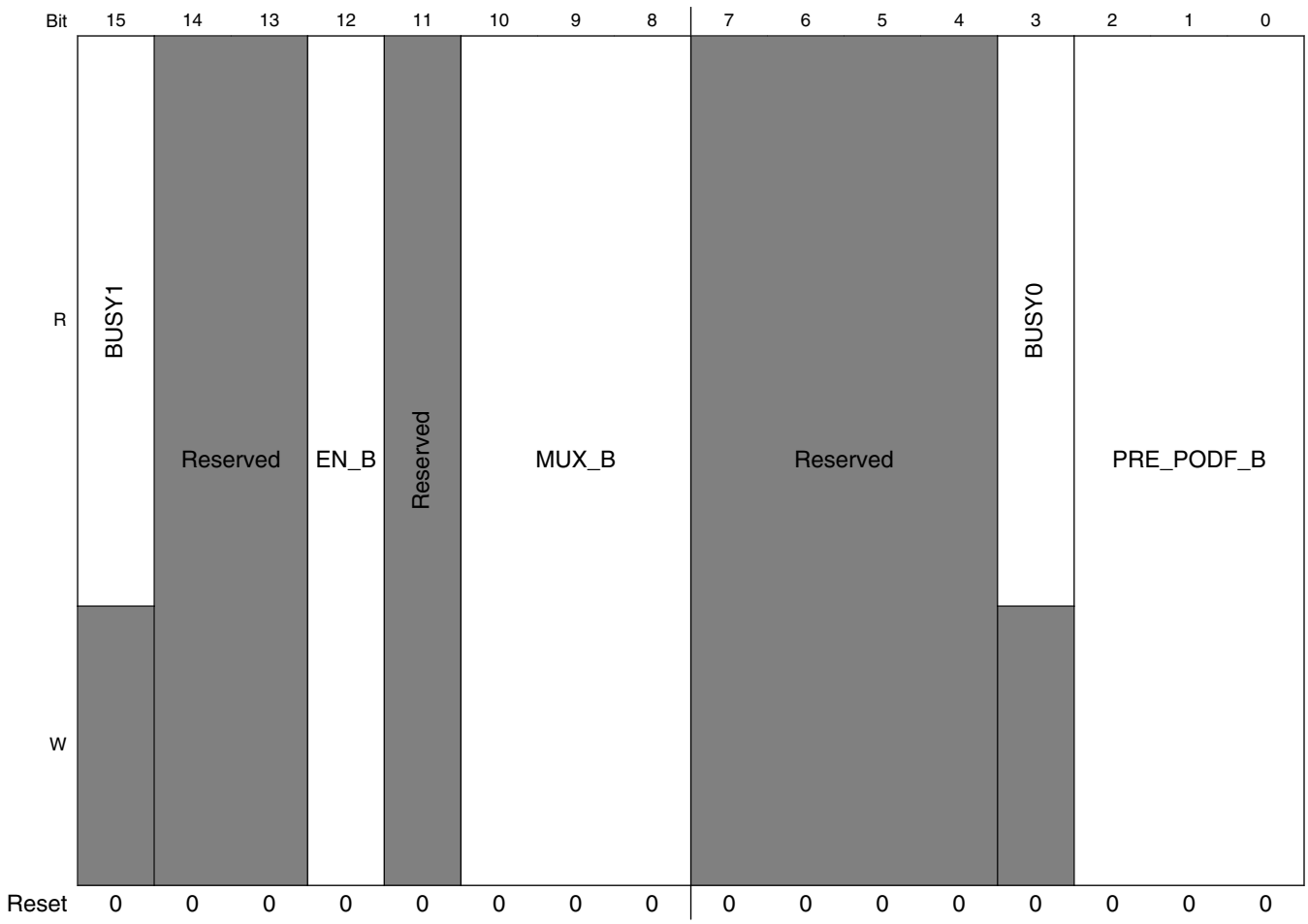
Field	Description
18–16 PRE_PODF_A	Pre divider divide number for branch A Divider value is n + 1. This field does not apply for CORE, DRAM, DRAM_PHYM 000 Divide by 1 001 Divide by 2 010 Divide by 3 011 Divide by 4 100 Divide by 5 101 Divide by 6 110 Divide by 7 111 Divide by 8
15 BUSY1	EN_B is applied to field This field does not apply for CORE, IP,DRAM, DRAM_PHYM
14–13 -	This field is reserved. Reserved
12 EN_B	Branch B clock gate control This field does not apply for CORE, IP,DRAM, DRAM_PHYM 0 Clock shutdown 1 Clock ON
11 -	This field is reserved. Reserved
10–8 MUX_B	Selection control of multiplexer of branch B This field does not apply for CORE, IP,DRAM, DRAM_PHYM
7–4 -	This field is reserved. Reserved
3 BUSY0	Pre divider value for branch A is applied This field does not apply for CORE, IP,DRAM, DRAM_PHYM
PRE_PODF_B	Pre divider divide number for branch B Divider value is n + 1. This field does not apply for CORE, IP, DRAM, DRAM_PHYM 000 Divide by 1 001 Divide by 2 010 Divide by 3 011 Divide by 4 100 Divide by 5 101 Divide by 6 110 Divide by 7 111 Divide by 8

5.2.8.25 Pre Divider Register (CCM_PRE_ROOT*n*_TOG)

Pre Root Register

Address: 3038_0000h base + 803Ch offset + (128d × i), where i=0d to 120d





CCM_PRE_ROOT*n*_TOG field descriptions

Field	Description
31 BUSY4	EN_A field is applied to field This field applies to DRAM and DRAM_PHYM
30–29 -	This field is reserved. Reserved
28 EN_A	Branch A clock gate control This field applies to DRAM and DRAM_PHYM 0 Clock shutdown 1 clock ON
27 -	This field is reserved. Reserved
26–24 MUX_A	Selection control of multiplexer of branch A This field applies to DRAM and DRAM_PHYM
23–20 -	This field is reserved. Reserved
19 BUSY3	Pre divider value for branch A is applied This field applies to DRAM and DRAM_PHYM

Table continues on the next page...

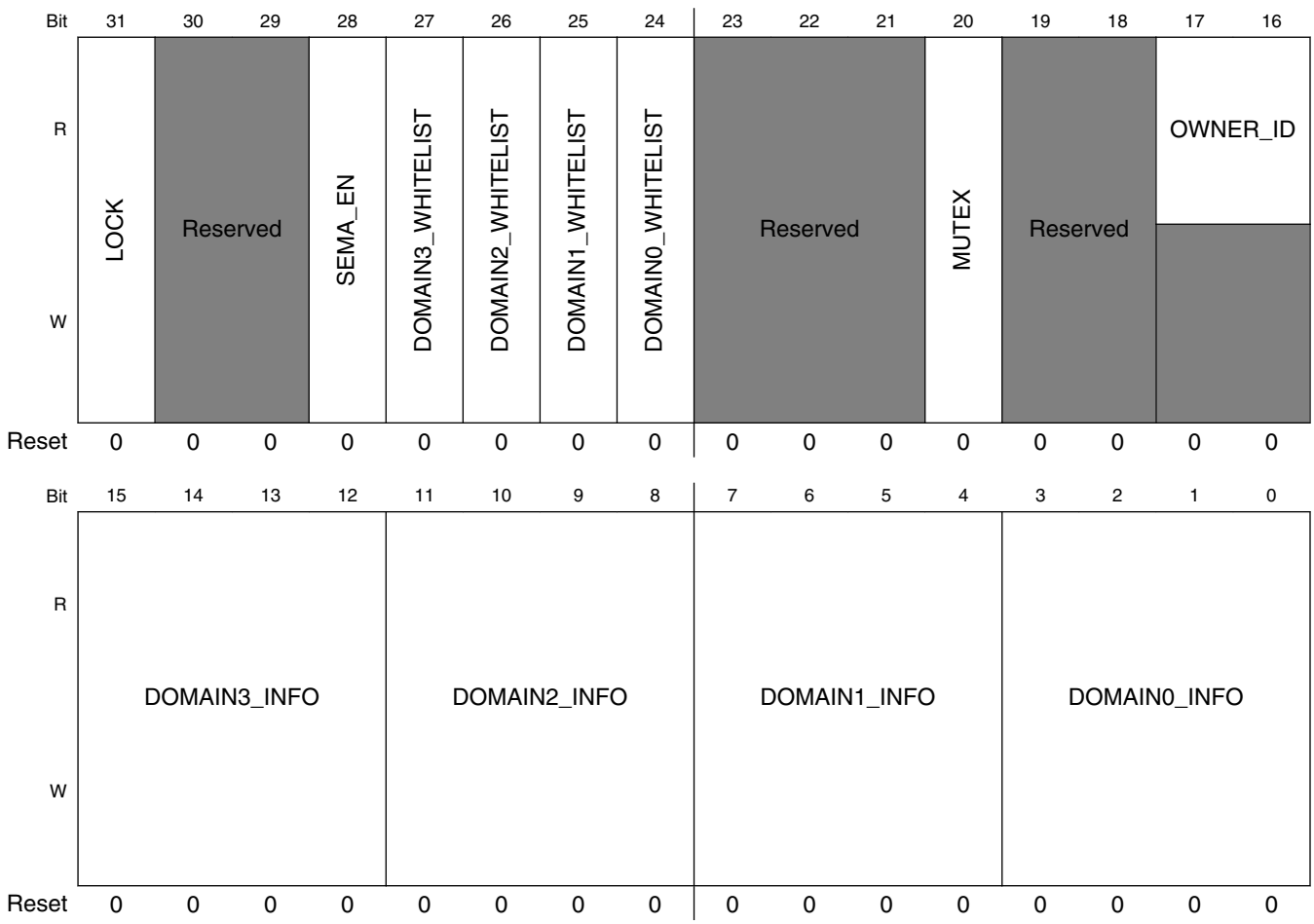
CCM_PRE_ROOT*n*_TOG field descriptions (continued)

Field	Description
18–16 PRE_PODF_A	Pre divider divide number for branch A Divider value is n + 1. This field does not apply for CORE, DRAM, DRAM_PHYM 000 Divide by 1 001 Divide by 2 010 Divide by 3 011 Divide by 4 100 Divide by 5 101 Divide by 6 110 Divide by 7 111 Divide by 8
15 BUSY1	EN_B is applied to field This field does not apply for CORE, IP,DRAM, DRAM_PHYM
14–13 -	This field is reserved. Reserved
12 EN_B	Branch B clock gate control This field does not apply for CORE, IP,DRAM, DRAM_PHYM 0 Clock shutdown 1 Clock ON
11 -	This field is reserved. Reserved
10–8 MUX_B	Selection control of multiplexer of branch B This field does not apply for CORE, IP,DRAM, DRAM_PHYM
7–4 -	This field is reserved. Reserved
3 BUSY0	Pre divider value for branch a is applied field does not apply for CORE, IP,DRAM, DRAM_PHYM
PRE_PODF_B	Pre divider divide number for branch B Divider value is n + 1. This field does not apply for CORE, IP, DRAM, DRAM_PHYM 000 Divide by 1 001 Divide by 2 010 Divide by 3 011 Divide by 4 100 Divide by 5 101 Divide by 6 110 Divide by 7 111 Divide by 8

5.2.8.26 Access Control Register (CCM_ACCESS_CTRL*n*)

Access Control Register

Address: 3038_0000h base + 8070h offset + (128d × i), where i=0d to 120d



CCM_ACCESS_CTRL*n* field descriptions

Field	Description
31 LOCK	Lock this clock root to use access control This bit can be set to 1 by software, and can be cleared only by system reset. 0 Access control inactive 1 Access control active
30–29 -	This field is reserved. Reserved
28 SEMA_EN	Enable internal semaphore This field cannot be changed when lock bit is 1

Table continues on the next page...

CCM_ACCESS_CTRL*n* field descriptions (continued)

Field	Description
	0 Disable 1 Enable
27 DOMAIN3_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
26 DOMAIN2_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
25 DOMAIN1_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
24 DOMAIN0_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
23–21 -	This field is reserved. Reserved
20 MUTEX	Semaphore to control access 0 Semaphore is free to take 1 Semaphore is taken Write 0 Release semaphore Write 1 Acquire semaphore
19–18 -	This field is reserved. Reserved
17–16 OWNER_ID	Current domain that owns semaphore This field is meaningless when MUTEX is 0 0 domaino 1 domain1 2 domain2 3 domain3
15–12 DOMAIN3_INFO	Information from domain 3 to pass to others This field can only be changed by domain 3
11–8 DOMAIN2_INFO	Information from domain 2 to pass to others This field can only be changed by domain 2
7–4 DOMAIN1_INFO	Information from domain 1 to pass to others This field can only be changed by domain 1

Table continues on the next page...

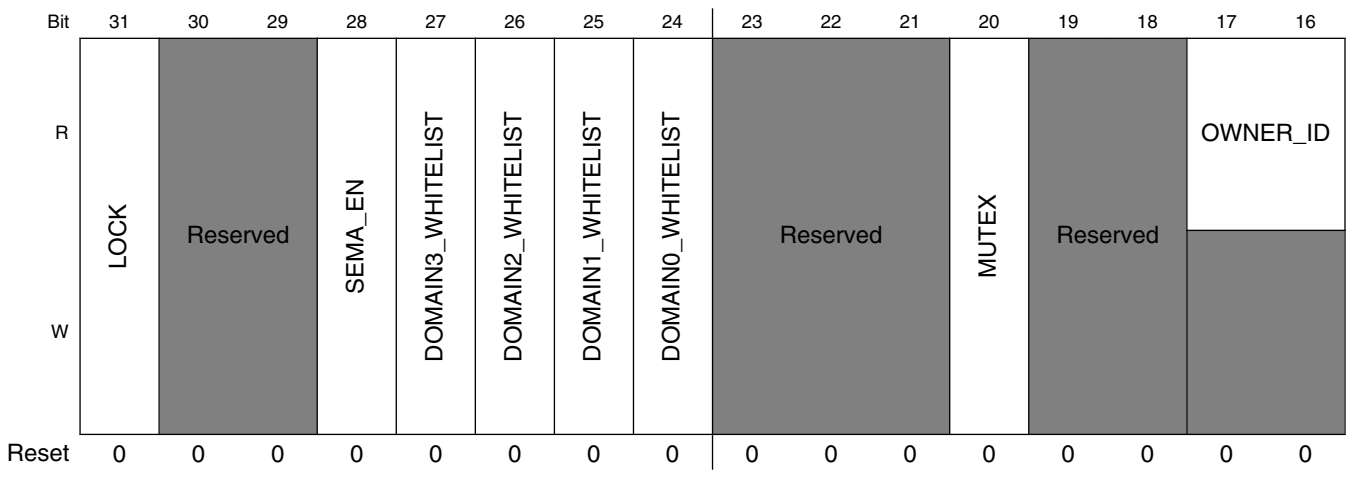
CCM_ACCESS_CTRL*n* field descriptions (continued)

Field	Description
DOMAIN0_INFO	Information from domain 0 to pass to others This field can only be changed by domain 0

5.2.8.27 Access Control Register (CCM_ACCESS_CTRL_ROOT*n*_SET)

Access Control Register

Address: 3038_0000h base + 8074h offset + (128d × i), where i=0d to 120d



CCM_ACCESS_CTRL_ROOT*n*_SET field descriptions

Field	Description
31 LOCK	Lock this clock root to use access control This bit can be set to 1 by software, and can be cleared only by system reset.

Table continues on the next page...

CCM_ACCESS_CTRL_ROOT*n*_SET field descriptions (continued)

Field	Description
	0 Access control inactive 1 Access control active
30–29 -	This field is reserved. Reserved
28 SEMA_EN	Enable internal semaphore This field cannot be changed when lock bit is 1 0 Disable 1 Enable
27 DOMAIN3_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
26 DOMAIN2_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
25 DOMAIN1_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
24 DOMAIN0_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
23–21 -	This field is reserved. Reserved
20 MUTEX	Semaphore to control access 0 Semaphore is free to take 1 Semaphore is taken Write 0 Release semaphore Write 1 Acquire semaphore
19–18 -	This field is reserved. Reserved
17–16 OWNER_ID	Current domain that owns semaphore This field is meaningless when MUTEX is 0 0 domaino 1 domain1 2 domain2 3 domain3

Table continues on the next page...

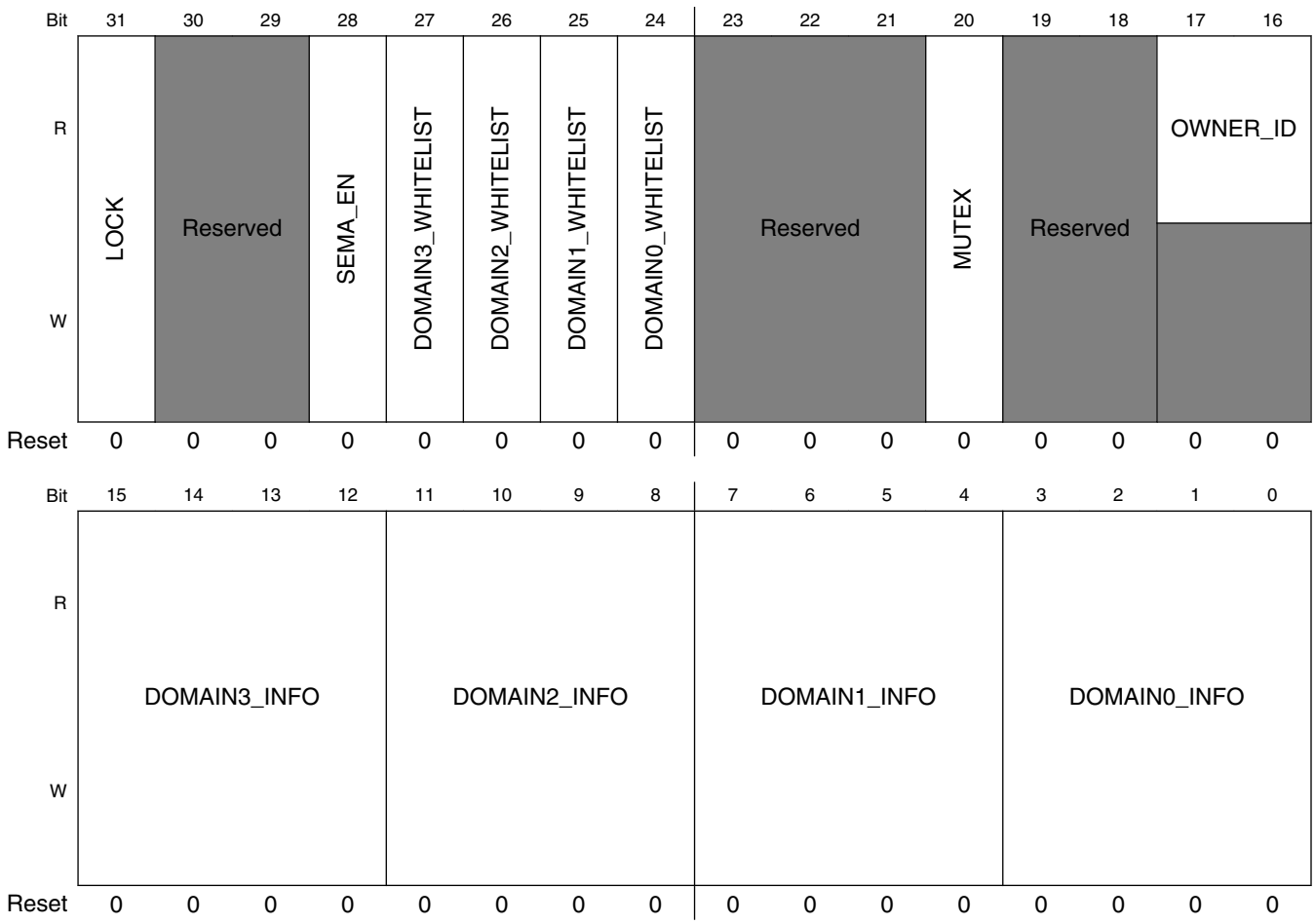
CCM_ACCESS_CTRL_ROOT*n*_SET field descriptions (continued)

Field	Description
15–12 DOMAIN3_INFO	Information from domain 3 to pass to others This field can only be changed by domain 3
11–8 DOMAIN2_INFO	Information from domain 2 to pass to others This field can only be changed by domain 2
7–4 DOMAIN1_INFO	Information from domain 1 to pass to others This field can only be changed by domain 1
DOMAIN0_INFO	Information from domain 0 to pass to others This field can only be changed by domain 0

5.2.8.28 Access Control Register (CCM_ACCESS_CTRL_ROOT*n*_CLR)

Access Control Register

Address: 3038_0000h base + 8078h offset + (128d × i), where i=0d to 120d



CCM_ACCESS_CTRL_ROOT*n*_CLR field descriptions

Field	Description
31 LOCK	Lock this clock root to use access control This bit can be set to 1 by software, and can be cleared only by system reset. 0 Access control inactive 1 Access control active
30–29 -	This field is reserved. Reserved

Table continues on the next page...

CCM_ACCESS_CTRL_ROOT*n*_CLR field descriptions (continued)

Field	Description
28 SEMA_EN	Enable internal semaphore This field cannot be changed when lock bit is 1 0 Disable 1 Enable
27 DOMAIN3_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
26 DOMAIN2_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
25 DOMAIN1_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
24 DOMAIN0_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
23–21 -	This field is reserved. Reserved
20 MUTEX	Semaphore to control access 0 Semaphore is free to take 1 Semaphore is taken Write 0 Release semaphore Write 1 Acquire semaphore
19–18 -	This field is reserved. Reserved
17–16 OWNER_ID	Current domain that owns semaphore This field is meaningless when MUTEX is 0 0 domaino 1 domain1 2 domain2 3 domain3
15–12 DOMAIN3_INFO	Information from domain 3 to pass to others This field can only be changed by domain 3
11–8 DOMAIN2_INFO	Information from domain 2 to pass to others This field can only be changed by domain 2

Table continues on the next page...

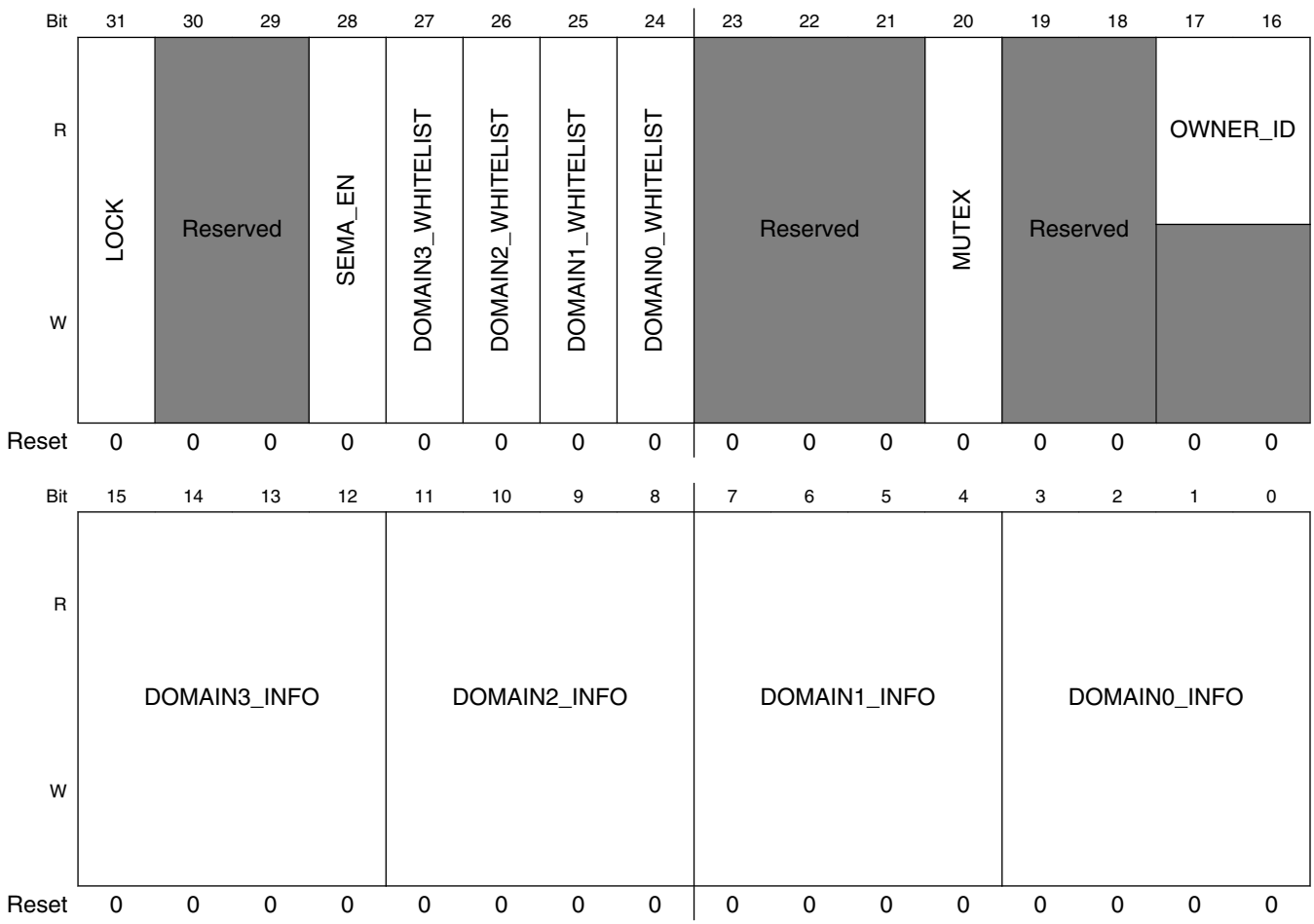
CCM_ACCESS_CTRL_ROOT*n*_CLR field descriptions (continued)

Field	Description
7–4 DOMAIN1_INFO	Information from domain 1 to pass to others This field can only be changed by domain 1
DOMAIN0_INFO	Information from domain 0 to pass to others This field can only be changed by domain 0

5.2.8.29 Access Control Register (CCM_ACCESS_CTRL_ROOT*n*_TOG)

Access Control Register

Address: 3038_0000h base + 807Ch offset + (128d × i), where i=0d to 120d



CCM_ACCESS_CTRL_ROOT*n*_TOG field descriptions

Field	Description
31 LOCK	Lock this clock root to use access control This bit can be set to 1 by software, and can be cleared only by system reset. 0 Access control inactive 1 Access control active
30–29 -	This field is reserved. Reserved
28 SEMA_EN	Enable internal semaphore This field cannot be changed when lock bit is 1 0 Disable 1 Enable
27 DOMAIN3_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
26 DOMAIN2_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
25 DOMAIN1_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
24 DOMAIN0_ WHITELIST	White list of domains that can change setting of this clock root. Each domain has a corresponding bit. 0 Domain cannot change the setting 1 Domain can change the setting
23–21 -	This field is reserved. Reserved
20 MUTEX	Semaphore to control access 0 Semaphore is free to take 1 Semaphore is taken Write 0 Release semaphore Write 1 Acquire semaphore
19–18 -	This field is reserved. Reserved
17–16 OWNER_ID	Current domain that owns semaphore This field is meaningless when MUTEX is 0 0 domaino 1 domain1

Table continues on the next page...

CCM_ACCESS_CTRL_ROOT*n*_TOG field descriptions (continued)

Field	Description
	2 domain2 3 domain3
15–12 DOMAIN3_INFO	Information from domain 3 to pass to others This field can only be changed by domain 3
11–8 DOMAIN2_INFO	Information from domain 2 to pass to others This field can only be changed by domain 2
7–4 DOMAIN1_INFO	Information from domain 1 to pass to others This field can only be changed by domain 1
DOMAIN0_INFO	Information from domain 0 to pass to others This field can only be changed by domain 0

5.2.9 CCM Analog Memory Map/Register Definition

This section describes the registers for the analog PLLs.

CCM_ANALOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3036_0060	Anadig ARM PLL control Register (CCM_ANALOG_PLL_ARM)	32	R/W	0001_3042h	5.2.9.1/ 747
3036_0064	Anadig ARM PLL control Register (CCM_ANALOG_PLL_ARM_SET)	32	R/W	0001_3042h	5.2.9.1/ 747
3036_0068	Anadig ARM PLL control Register (CCM_ANALOG_PLL_ARM_CLR)	32	R/W	0001_3042h	5.2.9.1/ 747
3036_006C	Anadig ARM PLL control Register (CCM_ANALOG_PLL_ARM_TOG)	32	R/W	0001_3042h	5.2.9.1/ 747
3036_0070	Anadig DDR PLL Control Register (CCM_ANALOG_PLL_DDR)	32	R/W	0011_3021h	5.2.9.2/ 749
3036_0074	Anadig DDR PLL Control Register (CCM_ANALOG_PLL_DDR_SET)	32	R/W	0011_3021h	5.2.9.2/ 749
3036_0078	Anadig DDR PLL Control Register (CCM_ANALOG_PLL_DDR_CLR)	32	R/W	0011_3021h	5.2.9.2/ 749
3036_007C	Anadig DDR PLL Control Register (CCM_ANALOG_PLL_DDR_TOG)	32	R/W	0011_3021h	5.2.9.2/ 749
3036_0080	DDR PLL Spread Spectrum Register. (CCM_ANALOG_PLL_DDR_SS)	32	R/W	0000_0000h	5.2.9.3/ 751
3036_0090	Numerator of DDR PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_DDR_NUM)	32	R/W	0000_0000h	5.2.9.4/ 752

Table continues on the next page...

CCM_ANALOG memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3036_00A0	Denominator of DDR PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_DDR_DENOM)	32	R/W	0000_0012h	5.2.9.5/ 752
3036_00B0	Anadig 480MHz PLL Control Register (CCM_ANALOG_PLL_480)	32	R/W	0001_3000h	5.2.9.6/ 754
3036_00B4	Anadig 480MHz PLL Control Register (CCM_ANALOG_PLL_480_SET)	32	R/W	0001_3000h	5.2.9.6/ 754
3036_00B8	Anadig 480MHz PLL Control Register (CCM_ANALOG_PLL_480_CLR)	32	R/W	0001_3000h	5.2.9.6/ 754
3036_00BC	Anadig 480MHz PLL Control Register (CCM_ANALOG_PLL_480_TOG)	32	R/W	0001_3000h	5.2.9.6/ 754
3036_00C0	480MHz Clock Phase Fractional Divider Control Register A (CCM_ANALOG_PFD_480A)	32	R/W	1220_1A16h	5.2.9.7/ 757
3036_00C4	480MHz Clock Phase Fractional Divider Control Register A (CCM_ANALOG_PFD_480A_SET)	32	R/W	1220_1A16h	5.2.9.7/ 757
3036_00C8	480MHz Clock Phase Fractional Divider Control Register A (CCM_ANALOG_PFD_480A_CLR)	32	R/W	1220_1A16h	5.2.9.7/ 757
3036_00CC	480MHz Clock Phase Fractional Divider Control Register A (CCM_ANALOG_PFD_480A_TOG)	32	R/W	1220_1A16h	5.2.9.7/ 757
3036_00D0	480MHz Clock Phase Fractional Divider Control Register B (CCM_ANALOG_PFD_480B)	32	R/W	1212_1212h	5.2.9.8/ 760
3036_00D4	480MHz Clock Phase Fractional Divider Control Register B (CCM_ANALOG_PFD_480B_SET)	32	R/W	1212_1212h	5.2.9.8/ 760
3036_00D8	480MHz Clock Phase Fractional Divider Control Register B (CCM_ANALOG_PFD_480B_CLR)	32	R/W	1212_1212h	5.2.9.8/ 760
3036_00DC	480MHz Clock Phase Fractional Divider Control Register B (CCM_ANALOG_PFD_480B_TOG)	32	R/W	1212_1212h	5.2.9.8/ 760
3036_00E0	Anadig ENET PLL Control Register (CCM_ANALOG_PLL_ENET)	32	R/W	0001_1FE0h	5.2.9.9/ 763
3036_00E4	Anadig ENET PLL Control Register (CCM_ANALOG_PLL_ENET_SET)	32	R/W	0001_1FE0h	5.2.9.9/ 763
3036_00E8	Anadig ENET PLL Control Register (CCM_ANALOG_PLL_ENET_CLR)	32	R/W	0001_1FE0h	5.2.9.9/ 763
3036_00EC	Anadig ENET PLL Control Register (CCM_ANALOG_PLL_ENET_TOG)	32	R/W	0001_1FE0h	5.2.9.9/ 763
3036_00F0	Anadig Audio PLL control Register (CCM_ANALOG_PLL_AUDIO)	32	R/W	0001_301Bh	5.2.9.10/ 766
3036_00F4	Anadig Audio PLL control Register (CCM_ANALOG_PLL_AUDIO_SET)	32	R/W	0001_301Bh	5.2.9.10/ 766
3036_00F8	Anadig Audio PLL control Register (CCM_ANALOG_PLL_AUDIO_CLR)	32	R/W	0001_301Bh	5.2.9.10/ 766
3036_00FC	Anadig Audio PLL control Register (CCM_ANALOG_PLL_AUDIO_TOG)	32	R/W	0001_301Bh	5.2.9.10/ 766
3036_0100	Audio PLL Spread Spectrum Register. (CCM_ANALOG_PLL_AUDIO_SS)	32	R/W	0000_0000h	5.2.9.11/ 768

Table continues on the next page...

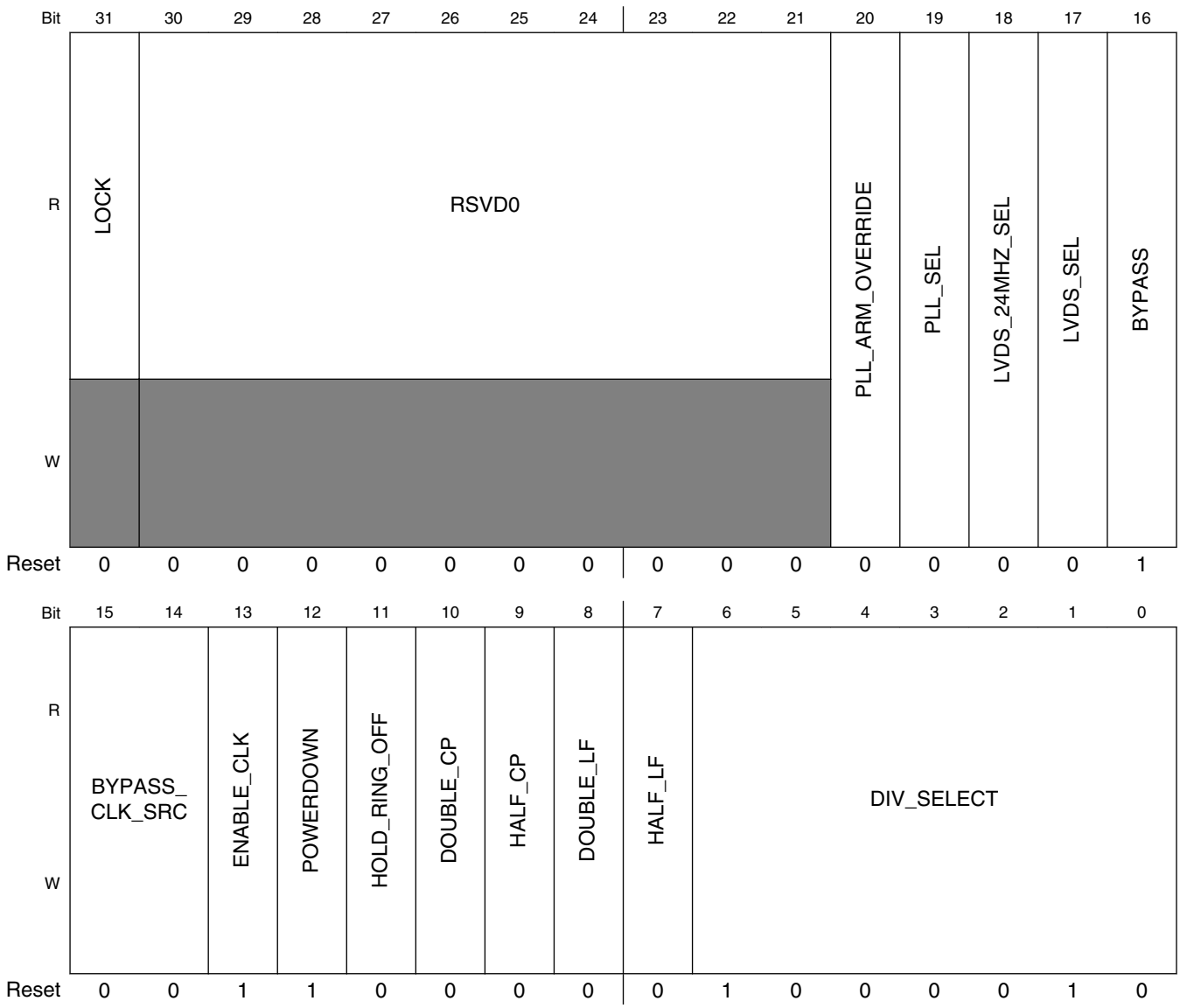
CCM_ANALOG memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3036_0110	Numerator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_NUM)	32	R/W	05F5_E100h	5.2.9.12/ 769
3036_0120	Denominator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_DENOM)	32	R/W	2964_619Ch	5.2.9.13/ 769
3036_0130	Anadig Video PLL control Register (CCM_ANALOG_PLL_VIDEO)	32	R/W	0001_301Bh	5.2.9.14/ 771
3036_0134	Anadig Video PLL control Register (CCM_ANALOG_PLL_VIDEO_SET)	32	R/W	0001_301Bh	5.2.9.14/ 771
3036_0138	Anadig Video PLL control Register (CCM_ANALOG_PLL_VIDEO_CLR)	32	R/W	0001_301Bh	5.2.9.14/ 771
3036_013C	Anadig Video PLL control Register (CCM_ANALOG_PLL_VIDEO_TOG)	32	R/W	0001_301Bh	5.2.9.14/ 771
3036_0140	Video PLL Spread Spectrum Register. (CCM_ANALOG_PLL_VIDEO_SS)	32	R/W	0000_0000h	5.2.9.15/ 773
3036_0150	Numerator of Video PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_VIDEO_NUM)	32	R/W	05F5_E100h	5.2.9.16/ 774
3036_0160	Denominator of Video PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_VIDEO_DENOM)	32	R/W	10A2_4447h	5.2.9.17/ 774
3036_0170	Miscellaneous0 Analog Clock Control and Status Register (CCM_ANALOG_CLK_MISC0)	32	R/W	0000_0000h	5.2.9.18/ 775
3036_0174	Miscellaneous0 Analog Clock Control and Status Register (CCM_ANALOG_CLK_MISC0_SET)	32	R/W	0000_0000h	5.2.9.18/ 775
3036_0178	Miscellaneous0 Analog Clock Control and Status Register (CCM_ANALOG_CLK_MISC0_CLR)	32	R/W	0000_0000h	5.2.9.18/ 775
3036_017C	Miscellaneous0 Analog Clock Control and Status Register (CCM_ANALOG_CLK_MISC0_TOG)	32	R/W	0000_0000h	5.2.9.18/ 775

5.2.9.1 Anadig ARM PLL control Register (CCM_ANALOG_PLL_ARM*n*)

The control register provides control for the system PLL.

Address: 3036_0000h base + 60h offset + (4d × i), where i=0d to 3d



CCM_ANALOG_PLL_ARM*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked; 0 - PLL is not currently locked.

Table continues on the next page...

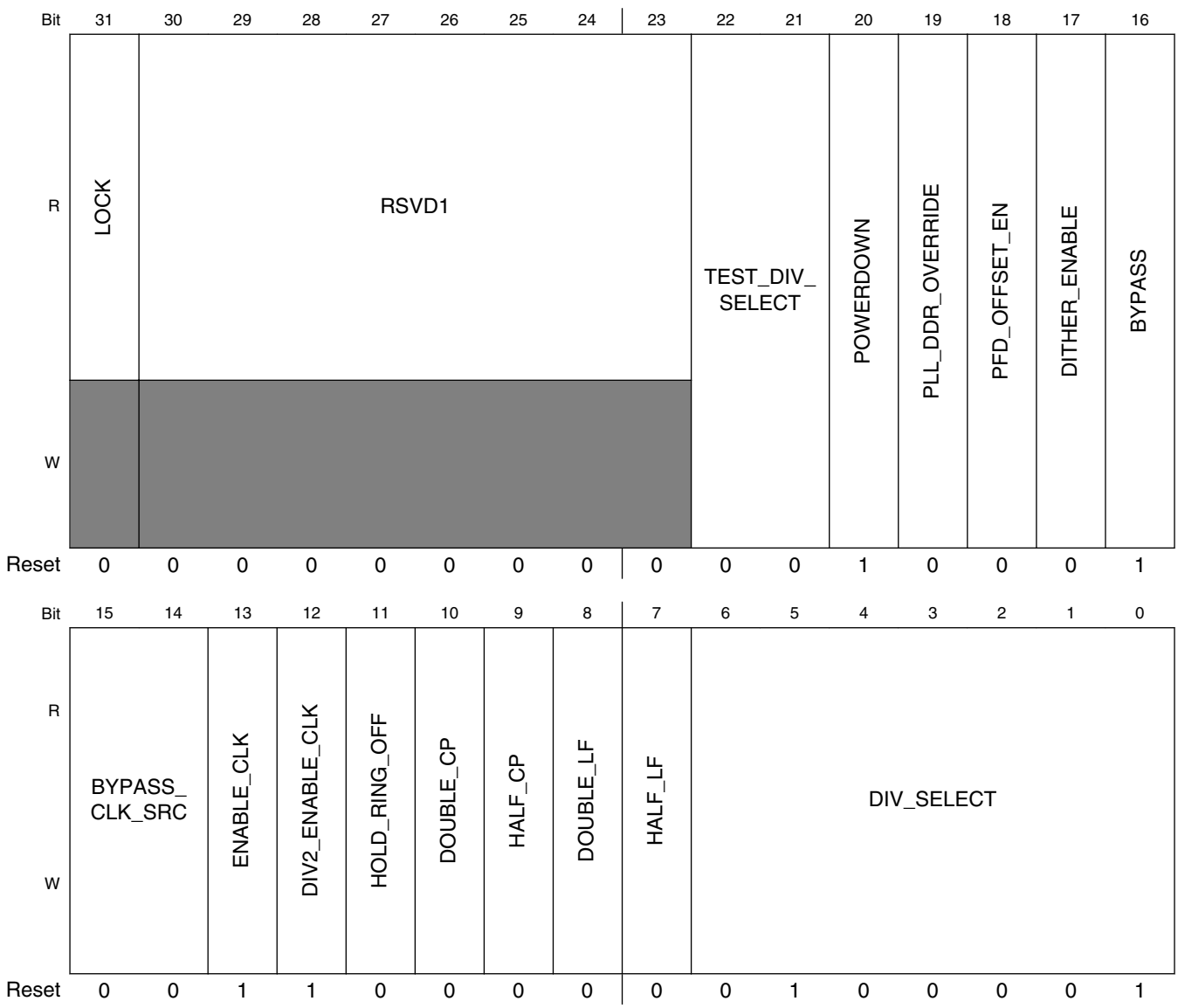
CCM_ANALOG_PLL_ARM*n* field descriptions (continued)

Field	Description
30–21 RSVD0	Always set to zero (0).
20 PLL_ARM_ OVERRIDE	The OVERRIDE bit allows the clock control module to automatically override portions of the register.
19 PLL_SEL	
18 LVDS_24MHZ_ SEL	
17 LVDS_SEL	
16 BYPASS	Bypass the pll.
15–14 BYPASS_CLK_ SRC	Determines the bypass source.
13 ENABLE_CLK	Enable the clock output.
12 POWERDOWN	Powers down the PLL.
11 HOLD_RING_ OFF	Analog debug bit.
10 DOUBLE_CP	Increases the charge pump gain 2x.
9 HALF_CP	Reduces the charge pump gain 2x.
8 DOUBLE_LF	Increases the frequency of the loop filter 2x.
7 HALF_LF	Reduces the frequency of the loop filter 2x.
DIV_SELECT	This field controls the pll loop divider. Valid range for divider value: 54-108. Fout = Fin * div_select/2.0.

5.2.9.2 Anadig DDR PLL Control Register (CCM_ANALOG_PLL_DDR*n*)

The control register provides control for the 480MHz PLL.

Address: 3036_0000h base + 70h offset + (4d × i), where i=0d to 3d



CCM_ANALOG_PLL_DDR*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked; 0 - PLL is not currently locked.

Table continues on the next page...

CCM_ANALOG_PLL_DDR*n* field descriptions (continued)

Field	Description
30–23 RSVD1	Always set to zero (0).
22–21 TEST_DIV_SELECT	Control bits for the post divider for the PLL clk and lvds outputs: 0x0=div-by-4(default), 0x1=div-by-2, 0x2=div-by-1, 0x3=div-by-1
20 POWERDOWN	Powers down the PLL.
19 PLL_DDR_OVERRIDE	The OVERRIDE bit allows the clock control module to automatically override portions of the register.
18 PFD_OFFSET_EN	Enables an offset in the phase frequency detector.
17 DITHER_ENABLE	Enables dither in the fractional modulator calculation.
16 BYPASS	Bypass the pll.
15–14 BYPASS_CLK_SRC	Determines the bypass source.
13 ENABLE_CLK	
12 DIV2_ENABLE_CLK	
11 HOLD_RING_OFF	Status of ana_irq2 input from analog block.
10 DOUBLE_CP	Increases the charge pump gain 2x.
9 HALF_CP	Reduces the charge pump gain 2x.
8 DOUBLE_LF	Increases the frequency of the loop filter 2x.
7 HALF_LF	Reduces the frequency of the loop filter 2x.
DIV_SELECT	This field controls the pll loop divider. Valid values for divider: 33 to 44.

5.2.9.3 DDR PLL Spread Spectrum Register. (CCM_ANALOG_PLL_DDR_SS)

This register contains the DDR PLL spread spectrum controls.

Address: 3036_0000h base + 80h offset = 3036_0080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	STOP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ENABLE	STEP														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_ANALOG_PLL_DDR_SS field descriptions

Field	Description
31–16 STOP	Frequency change = step/B*24MHz.
15 ENABLE	This bit enables the spread spectrum modulation.
STEP	The max frequency change = stop/B*24MHz.

5.2.9.4 Numerator of DDR PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_DDR_NUM)

This register contains the numerator of DDR PLL fractional loop divider.

Address: 3036_0000h base + 90h offset = 3036_0090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0		A													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	A															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_ANALOG_PLL_DDR_NUM field descriptions

Field	Description
31–30 RSVD0	Always set to zero (0).
A	30 bit numerator (A) of fractional loop divider (signed integer).

5.2.9.5 Denominator of DDR PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_DDR_DENOM)

This register contains the Denominator of DDR PLL fractional loop divider.

Address: 3036_0000h base + A0h offset = 3036_00A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0		B													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	B															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

CCM_ANALOG_PLL_DDR_DENOM field descriptions

Field	Description
31–30 RSVD0	Always set to zero (0).
B	30 bit Denominator (B) of fractional loop divider (signed integer).

5.2.9.6 Anadig 480MHz PLL Control Register (CCM_ANALOG_PLL_480*n*)

The control register provides control for the 480MHz PLL.

Address: 3036_0000h base + B0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOCK	RSVD1		PFD2_DIV2_CLKGATE	PFD1_DIV2_CLKGATE	PFD0_DIV2_CLKGATE	PFD7_OVERRIDE	PFD6_OVERRIDE	PFD5_OVERRIDE	PFD4_OVERRIDE	PFD3_OVERRIDE	PFD2_OVERRIDE	PFD1_OVERRIDE	PFD0_OVERRIDE	PLL_480_OVERRIDE	BYPASS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BYPASS_CLK_SRC		ENABLE_CLK	POWERDOWN	HOLD_RING_OFF	DOUBLE_CP	HALF_CP	DOUBLE_LF	HALF_LF	MAIN_DIV4_CLKGATE	MAIN_DIV2_CLKGATE	MAIN_DIV1_CLKGATE	RSVD0			DIV_SELECT
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

CCM_ANALOG_PLL_480*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked; 0 - PLL is not currently locked.
30–29 RSVD1	Always set to zero (0).
28 PFD2_DIV2_ CLKGATE	If set to 1, pll_sys_pfd2_135m_clk is off (power savings). 0: pll_sys_pfd2_135m_clk is enabled.
27 PFD1_DIV2_ CLKGATE	If set to 1, pll_sys_pfd1_166m_clk is off (power savings). 0: pll_sys_pfd1_166m_clk is enabled.
26 PFD0_DIV2_ CLKGATE	If set to 1, pll_sys_pfd0_196m_clk is off (power savings). 0: pll_sys_pfd0_196m_clk is enabled.
25 PFD7_ OVERRIDE	The OVERRIDE bit allows the clock control module to automatically override portions of the register.
24 PFD6_ OVERRIDE	The OVERRIDE bit allows the clock control module to automatically override portions of the register.
23 PFD5_ OVERRIDE	The OVERRIDE bit allows the clock control module to automatically override portions of the register.
22 PFD4_ OVERRIDE	The OVERRIDE bit allows the clock control module to automatically override portions of the register.
21 PFD3_ OVERRIDE	The OVERRIDE bit allows the clock control module to automatically override portions of the register.
20 PFD2_ OVERRIDE	The OVERRIDE bit allows the clock control module to automatically override portions of the register.
19 PFD1_ OVERRIDE	The OVERRIDE bit allows the clock control module to automatically override portions of the register.
18 PFD0_ OVERRIDE	The OVERRIDE bit allows the clock control module to automatically override portions of the register.
17 PLL_480_ OVERRIDE	The OVERRIDE bit allows the clock control module to automatically override portions of the register.
16 BYPASS	Bypass the pll.
15–14 BYPASS_CLK_ SRC	Determines the bypass source.
13 ENABLE_CLK	Enable the clock output.

Table continues on the next page...

CCM_ANALOG_PLL_480*n* field descriptions (continued)

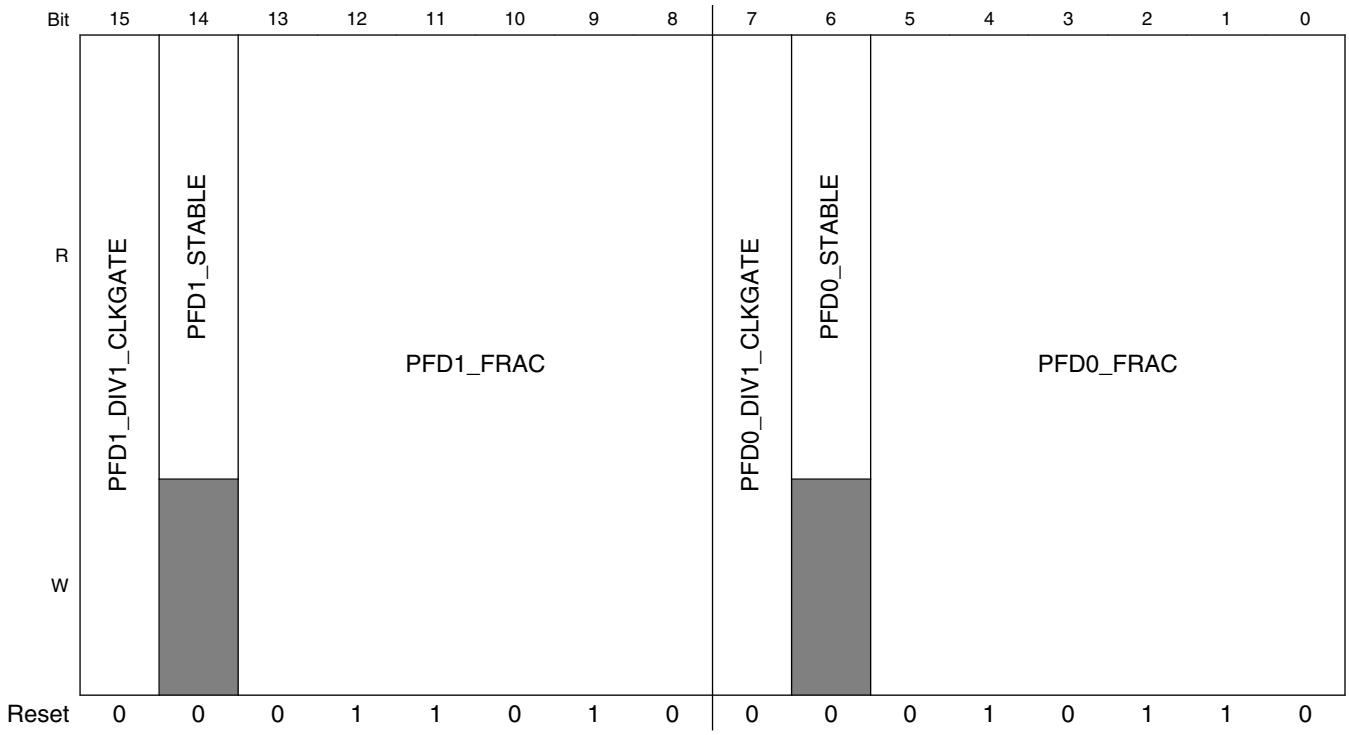
Field	Description
12 POWERDOWN	Powers down the PLL.
11 HOLD_RING_ OFF	Analog debug bit.
10 DOUBLE_CP	Increases the charge pump gain 2x.
9 HALF_CP	Reduces the charge pump gain 2x.
8 DOUBLE_LF	Increases the frequency of the loop filter 2x.
7 HALF_LF	Reduces the frequency of the loop filter 2x.
6 MAIN_DIV4_ CLKGATE	If set to 1, pll_sys_main_120m_clk is off (power savings). 0: pll_sys_main_120m_clk is enabled.
5 MAIN_DIV2_ CLKGATE	If set to 1, pll_sys_main_240m_clk is off (power savings). 0: pll_sys_main_240m_clk is enabled.
4 MAIN_DIV1_ CLKGATE	If set to 1, pll_sys_main_480m_clk is off (power savings). 0: pll_sys_main_480m_clk is enabled.
3–1 RSVD0	Always set to zero (0).
0 DIV_SELECT	This field controls the pll loop divider. 0 - Fout=480MHz; 1 - Fout=528MHz.

5.2.9.7 480MHz Clock Phase Fractional Divider Control Register A (CCM_ANALOG_PFD_480A*n*)

This register controls the 4-phase fractional clock dividers. The fractional clock frequencies are a product of the values in these registers.

Address: 3036_0000h base + C0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PFD3_DIV1_CLKGATE	PFD3_STABLE	PFD3_FRAC						PFD2_DIV1_CLKGATE	PFD2_STABLE	PFD2_FRAC					
W																
Reset	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0



CCM_ANALOG_PFD_480A*n* field descriptions

Field	Description
31 PFD3_DIV1_CLKGATE	IO Clock Gate. If set to 1, the 3rd fractional divider clock (reference ref_pfd3) is off (power savings). 0: ref_pfd3 fractional divider clock is enabled. If the corresponding override bit is set in the 480MHz PLL control register, this field has no effect.
30 PFD3_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
29–24 PFD3_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD3_FRAC where PFD3_FRAC is in the range 12-35.
23 PFD2_DIV1_CLKGATE	IO Clock Gate. If set to 1, the IO fractional divider clock (reference ref_pfd2) is off (power savings). 0: ref_pfd2 fractional divider clock is enabled. If the corresponding override bit is set in the 480MHz PLL control register, this field has no effect.
22 PFD2_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
21–16 PFD2_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD2_FRAC where PFD2_FRAC is in the range 12-35.
15 PFD1_DIV1_CLKGATE	IO Clock Gate. If set to 1, the IO fractional divider clock (reference ref_pfd1) is off (power savings). 0: ref_pfd1 fractional divider clock is enabled. If the corresponding override bit is set in the 480MHz PLL control register, this field has no effect.
14 PFD1_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit,

Table continues on the next page...

CCM_ANALOG_PFD_480A*n* field descriptions (continued)

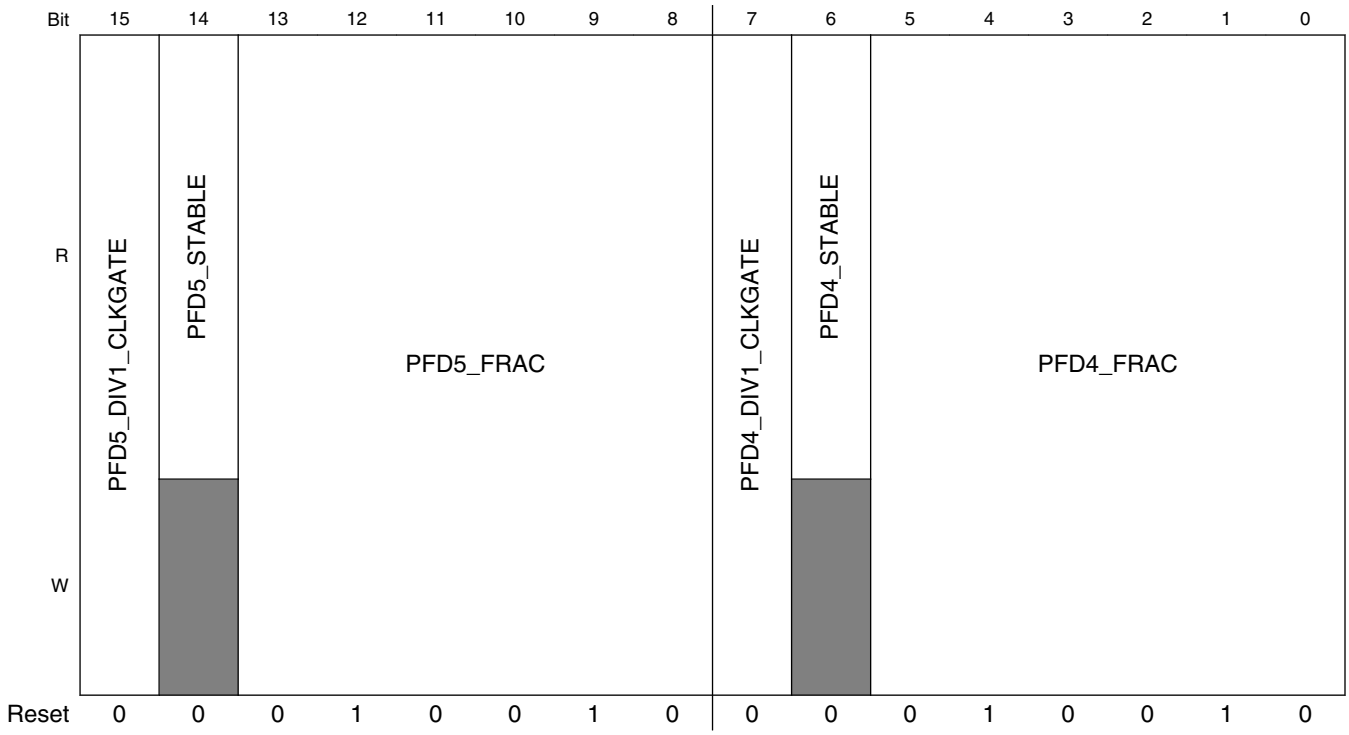
Field	Description
	program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
13–8 PFD1_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD1_FRAC where PFD1_FRAC is in the range 12-35.
7 PFD0_DIV1_ CLKGATE	If set to 1, the IO fractional divider clock (reference ref_pfd0) is off (power savings). 0: ref_pfd0 fractional divider clock is enabled. If the corresponding override bit is set in the 480MHz PLL control register, this field has no effect.
6 PFD0_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
PFD0_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD0_FRAC where PFD0_FRAC is in the range 12-35.

5.2.9.8 480MHz Clock Phase Fractional Divider Control Register B (CCM_ANALOG_PFD_480B*n*)

This register controls the 4-phase fractional clock dividers. The fractional clock frequencies are a product of the values in these registers.

Address: 3036_0000h base + D0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PFD7_DIV1_CLKGATE	PFD7_STABLE	PFD7_FRAC						PFD6_DIV1_CLKGATE	PFD6_STABLE	PFD6_FRAC					
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0



CCM_ANALOG_PFD_480B*n* field descriptions

Field	Description
31 PFD7_DIV1_ CLKGATE	IO Clock Gate. If set to 1, the 3rd fractional divider clock (reference ref_pfd7) is off (power savings). 0: ref_pfd3 fractional divider clock is enabled. If the corresponding override bit is set in the 480MHz PLL control register, this field has no effect.
30 PFD7_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
29–24 PFD7_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD7_FRAC where PFD3_FRAC is in the range 12-35.
23 PFD6_DIV1_ CLKGATE	IO Clock Gate. If set to 1, the IO fractional divider clock (reference ref_pfd2) is off (power savings). 0: ref_pfd2 fractional divider clock is enabled. If the corresponding override bit is set in the 480MHz PLL control register, this field has no effect.
22 PFD6_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
21–16 PFD6_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD6_FRAC where PFD2_FRAC is in the range 12-35.
15 PFD5_DIV1_ CLKGATE	IO Clock Gate. If set to 1, the IO fractional divider clock (reference ref_pfd1) is off (power savings). 0: ref_pfd1 fractional divider clock is enabled. If the corresponding override bit is set in the 480MHz PLL control register, this field has no effect.
14 PFD5_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit,

Table continues on the next page...

CCM_ANALOG_PFD_480B*n* field descriptions (continued)

Field	Description
	program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
13–8 PFD5_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD5_FRAC where PFD1_FRAC is in the range 12-35.
7 PFD4_DIV1_ CLKGATE	If set to 1, the IO fractional divider clock (reference ref_pfd0) is off (power savings). 0: ref_pfd0 fractional divider clock is enabled. If the corresponding override bit is set in the 480MHz PLL control register, this field has no effect.
6 PFD4_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
PFD4_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD4_FRAC where PFD0_FRAC is in the range 12-35.

5.2.9.9 Anadig ENET PLL Control Register (CCM_ANALOG_PLL_ENET*n*)

The control register provides control for the ENET PLL.

Address: 3036_0000h base + E0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOCK	RSVD1												PFD_OFFSET_EN	DITHER_ENABLE	BYPASS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BYPASS_CLK_SRC		PLL_ENET_OVERRIDE	ENABLE_CLK_500MHZ	ENABLE_CLK_250MHZ	ENABLE_CLK_125MHZ	ENABLE_CLK_100MHZ	ENABLE_CLK_50MHZ	ENABLE_CLK_40MHZ	ENABLE_CLK_25MHZ	POWERDOWN	HOLD_RING_OFF	DOUBLE_CP	HALF_CP	DOUBLE_LF	HALF_LF
W																
Reset	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0

CCM_ANALOG_PLL_ENET*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked; 0 - PLL is not currently locked.
30–19 RSVD1	Always set to zero (0).
18 PFD_OFFSET_EN	Enables an offset in the phase frequency detector.
17 DITHER_ENABLE	Enables dither in the fractional modulator calculation.
16 BYPASS	Bypass the pll.
15–14 BYPASS_CLK_SRC	Determines the bypass source.
13 PLL_ENET_OVERRIDE	The OVERRIDE bit allows the clock control module to automatically override portions of the register.
12 ENABLE_CLK_500MHZ	Enables the ethernet 500MHz clock output.
11 ENABLE_CLK_250MHZ	Enables the ethernet 250MHz clock output.
10 ENABLE_CLK_125MHZ	Enables the ethernet 125MHz clock output.
9 ENABLE_CLK_100MHZ	Enables the ethernet 100MHz clock output.
8 ENABLE_CLK_50MHZ	Enables the ethernet 50MHz clock output.
7 ENABLE_CLK_40MHZ	Enables the ethernet 40MHz clock output.
6 ENABLE_CLK_25MHZ	Enables the ethernet 25MHz clock output.
5 POWERDOWN	Powers down the PLL.
4 HOLD_RING_OFF	Status of ana_irq2 input from analog block.
3 DOUBLE_CP	Increases the charge pump gain 2x.
2 HALF_CP	Reduces the charge pump gain 2x.

Table continues on the next page...

CCM_ANALOG_PLL_ENET*n* field descriptions (continued)

Field	Description
1 DOUBLE_LF	Increases the frequency of the loop filter 2x.
0 HALF_LF	Reduces the frequency of the loop filter 2x.

5.2.9.10 Anadig Audio PLL control Register (CCM_ANALOG_PLL_AUDIO*n*)

The control register provides control for the Audio PLL.

Address: 3036_0000h base + F0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOCK	RSVD1						PLL_AUDIO_OVERRIDE	POST_DIV_SEL		RSVD0	TEST_DIV_SELECT		PFD_OFFSET_EN	DITHER_ENABLE	BYPASS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BYPASS_CLK_SRC		ENABLE_CLK	POWERDOWN	HOLD_RING_OFF	DOUBLE_CP	HALF_CP	DOUBLE_LF	HALF_LF	DIV_SELECT						
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	1	1	0	1	1

CCM_ANALOG_PLL_AUDIO*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked; 0 - PLL is not currently locked.
30–25 RSVD1	Always set to zero (0).
24 PLL_AUDIO_ OVERRIDE	The OVERRIDE bit allows the clock control module to automatically override portions of the register.
23–22 POST_DIV_SEL	Post-divider for audio PLL: 0x0=div-by-1(default), 0x1=div-by-2, 0x2=div-by-1, 0x3=div-by-4. The output clock of the audio PLL should be gated prior to changing this divider to prevent glitches.
21 RSVD0	Always set to zero (0).
20–19 TEST_DIV_ SELECT	Control bits for the divider for the PLL clk and lvds outputs: 0x0=div-by-4(default), 0x1=div-by-2, 0x2=div-by-1, 0x3=div-by-1. This divider is placed before the post divider.
18 PFD_OFFSET_ EN	Enables an offset in the phase frequency detector.
17 DITHER_ ENABLE	Enables dither in the fractional modulator calculation.
16 BYPASS	Bypass the pll.
15–14 BYPASS_CLK_ SRC	Determines the bypass source.
13 ENABLE_CLK	
12 POWERDOWN	Powers down the PLL.
11 HOLD_RING_ OFF	Status of ana_irq2 input from analog block.
10 DOUBLE_CP	Increases the charge pump gain 2x.
9 HALF_CP	Reduces the charge pump gain 2x.
8 DOUBLE_LF	Increases the frequency of the loop filter 2x.
7 HALF_LF	Reduces the frequency of the loop filter 2x.
DIV_SELECT	This field controls the pll loop divider. Valid range for DIV_SELECT divider value: 27-54 decimal.

5.2.9.11 Audio PLL Spread Spectrum Register. (CCM_ANALOG_PLL_AUDIO_SS)

This register contains the Audio PLL spread spectrum controls.

Address: 3036_0000h base + 100h offset = 3036_0100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	STOP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ENABLE	STEP														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_ANALOG_PLL_AUDIO_SS field descriptions

Field	Description
31–16 STOP	Frequency change = step/B*24MHz.
15 ENABLE	This bit enables the spread spectrum modulation.
STEP	The max frequency change = stop/B*24MHz.

5.2.9.12 Numerator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_NUM)

This register contains the numerator (A) of Audio PLL fractional loop divider.

Address: 3036_0000h base + 110h offset = 3036_0110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0		A													
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	1	0	1	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	A															
W																
Reset	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0

CCM_ANALOG_PLL_AUDIO_NUM field descriptions

Field	Description
31–30 RSVD0	Always set to zero (0).
A	30 bit numerator of fractional loop divider.

5.2.9.13 Denominator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_DENOM)

This register contains the Denominator (B) of Audio PLL fractional loop divider.

Address: 3036_0000h base + 120h offset = 3036_0120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0		B													
W																
Reset	0	0	1	0	1	0	0	1	0	1	1	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	B															
W																
Reset	0	1	1	0	0	0	0	1	1	0	0	1	1	1	0	0

CCM_ANALOG_PLL_AUDIO_DENOM field descriptions

Field	Description
31–30 RSVD0	Always set to zero (0).
B	30 bit Denominator of fractional loop divider.

5.2.9.14 Anadig Video PLL control Register (CCM_ANALOG_PLL_VIDEO*n*)

The control register provides control for the Video PLL.

Address: 3036_0000h base + 130h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOCK	RSVD1						PLL_VIDEO_OVERRIDE	POST_DIV_SEL		RSVD0	TEST_DIV_SELECT		PFD_OFFSET_EN	DITHER_ENABLE	BYPASS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BYPASS_CLK_SRC		ENABLE_CLK	POWERDOWN	HOLD_RING_OFF	DOUBLE_CP	HALF_CP	DOUBLE_LF	HALF_LF	DIV_SELECT						
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	1	1	0	1	1

CCM_ANALOG_PLL_VIDEO*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked; 0 - PLL is not currently locked.
30–25 RSVD1	Always set to zero (0).
24 PLL_VIDEO_ OVERRIDE	The OVERRIDE bit allows the clock control module to automatically override portions of the register.
23–22 POST_DIV_SEL	Post-divider for video PLL: 0x0=div-by-1(default), 0x1=div-by-2, 0x2=div-by-1, 0x3=div-by-4. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches.
21 RSVD0	Always set to zero (0).
20–19 TEST_DIV_ SELECT	Control bits for the divider for the PLL clk and lvds outputs: 0x0=div-by-4(default), 0x1=div-by-2, 0x2=div-by-1, 0x3=div-by-1. This divider is placed before the post divider.
18 PFD_OFFSET_ EN	Enables an offset in the phase frequency detector.
17 DITHER_ ENABLE	Enables dither in the fractional modulator calculation.
16 BYPASS	Bypass the pll.
15–14 BYPASS_CLK_ SRC	Determines the bypass source.
13 ENABLE_CLK	
12 POWERDOWN	Powers down the PLL.
11 HOLD_RING_ OFF	Status of ana_irq2 input from analog block.
10 DOUBLE_CP	Increases the charge pump gain 2x.
9 HALF_CP	Reduces the charge pump gain 2x.
8 DOUBLE_LF	Increases the frequency of the loop filter 2x.
7 HALF_LF	Reduces the frequency of the loop filter 2x.
DIV_SELECT	This field controls the pll loop divider. Valid range for DIV_SELECT divider value: 27-54 decimal.

5.2.9.15 Video PLL Spread Spectrum Register. (CCM_ANALOG_PLL_VIDEO_SS)

This register contains the Video PLL spread spectrum controls.

Address: 3036_0000h base + 140h offset = 3036_0140h

CCM_ANALOG_PLL_VIDEO_SS field descriptions

Field	Description
31–16 STOP	Frequency change = step/B*24MHz.
15 ENABLE	This bit enables the spread spectrum modulation.
STEP	The max frequency change = stop/B*24MHz.

5.2.9.16 Numerator of Video PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_VIDEO_NUM)

This register contains the numerator (A) of Video PLL fractional loop divider.

Address: 3036_0000h base + 150h offset = 3036_0150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0		A													
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	1	0	1	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	A															
W																
Reset	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0

CCM_ANALOG_PLL_VIDEO_NUM field descriptions

Field	Description
31–30 RSVD0	Always set to zero (0).
A	30 bit numerator of fractional loop divider.

5.2.9.17 Denominator of Video PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_VIDEO_DENOM)

This register contains the Denominator (B) of Video PLL fractional loop divider.

Address: 3036_0000h base + 160h offset = 3036_0160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0		B													
W																
Reset	0	0	0	1	0	0	0	0	1	0	1	0	0	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	B															
W																
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	1

CCM_ANALOG_PLL_VIDEO_DENOM field descriptions

Field	Description
31–30 RSVD0	Always set to zero (0).
B	30 bit Denominator of fractional loop divider.

5.2.9.18 Miscellaneous0 Analog Clock Control and Status Register (CCM_ANALOG_CLK_MISC0*n*)

This register defines the control and status bits for miscellaneous analog blocks. The lvds1 and lvds2 controls control the behavior of the anaclk1/1b and anaclk2/2b lvds IO's.

Address: 3036_0000h base + 170h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0								ACLK2_PREDIV	LVDSCLK1_IBEN	LVDSCLK1_OBEN	LVDS1_CLK_SEL				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

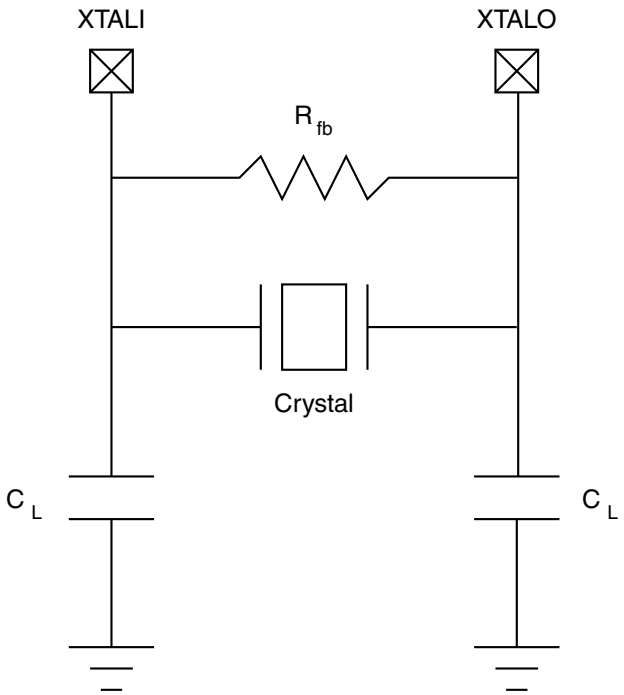
CCM_ANALOG_CLK_MISC0*n* field descriptions

Field	Description
31–8 RSVD0	Always set to zero (0).
7 ACLK2_PREDIV	Predivider ACLK2 source/reference clock of the PLL's. 0 - divide by 1; 1 - divide by 2. Please see PLL programming registers to select this source as the reference clock to the desired PLL
6 LVDSCLK1_ IBEN	This enables the lvds input buffer for anaclk1/1b. Do not enable input and output buffers simutaneously.
5 LVDSCLK1_ OBEN	This enables the lvds output buffer for anaclk1/1b. Do not enable input and output buffers simutaneously.
LVDS1_CLK_ SEL	This field selects the clock to be routed to anaclk1/1b 0x00 - Arm PLL; 0x01 - 480 PLL; 0x02 - pfd0; 0x03 - pfd1; 0x04 - pfd2; 0x05 - pfd3; 0x06 - pfd4; 0x07 - pfd5; 0x08 - pfd6; 0x09 - pfd7; 0x0A - Audio PLL; 0x0B - Video PLL; 0x0C - pll_enet_div2 (500MHz); 0x0D - pll_enet_div4 (250MHz); 0x0E - pll_enet_div8 (125MHz); 0x0F - pll_enet_div10 (100MHz); 0x10 - pll_enet_div20 (50MHz); 0x11 - pll_enet_div25 (40MHz); 0x12 - pll_enet_div40 (25MHz); 0x13 - pll_ddr; 0x14 - RC_OSC24; 0x15 - clk24mhz; 0x16 - anatest ring oscillators; 0x17 - aclk2_loopback; 0x18-0x1F - Reserved.

5.3 Crystal Oscillator (XTALOSC)

5.3.1 Overview

Crystal oscillators are constituted of oscillator I/O (POSC1 and POSCP) together with external board components.



C_L load capacitance

R_{fb} feedback resistor

Figure 5-21. Crystal oscillator block diagram

5.3.2 Functional Description

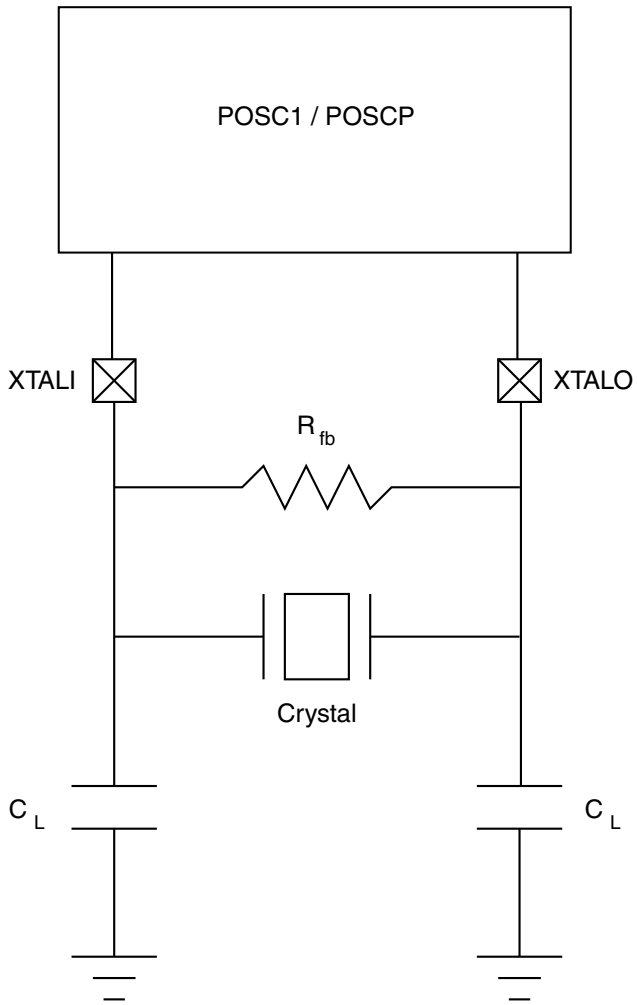


Figure 5-22. Using the crystal oscillator cell

Crystal oscillator provides selection control switches to either select crystal clock or external clock through PADI pin. PADI pin accepts external clock at 1.8V pre-driver supply voltage domain. Any voltage higher than 1.8V supply voltage applied at PADI can damage the oscillator I/O.

Crystal oscillator is required to be placed in a certain order in the I/O ring.

5.3.3 XTALOSC Memory Map/Register Definition

NOTE

The register content is mixed with analog functions not related to the oscillator function. These bits are noted.

XTALOSC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3036_0000	Anadig 24M Oscillator Control Register (XTALOSC_CTRL_24M)	32	R/W	0000_222Ah	5.3.3.1/ 781
3036_0004	Anadig 24M Oscillator Control Register (XTALOSC_CTRL_24M_SET)	32	R/W	0000_222Ah	5.3.3.1/ 781
3036_0008	Anadig 24M Oscillator Control Register (XTALOSC_CTRL_24M_CLR)	32	R/W	0000_222Ah	5.3.3.1/ 781
3036_000C	Anadig 24M Oscillator Control Register (XTALOSC_CTRL_24M_TOG)	32	R/W	0000_222Ah	5.3.3.1/ 781
3036_0010	Anadig 24MHz RC Osc. config0 Register (XTALOSC_RCOSC_CONFIG0)	32	R/W	0002_2930h	5.3.3.2/ 783
3036_0014	Anadig 24MHz RC Osc. config0 Register (XTALOSC_RCOSC_CONFIG0_SET)	32	R/W	0002_2930h	5.3.3.2/ 783
3036_0018	Anadig 24MHz RC Osc. config0 Register (XTALOSC_RCOSC_CONFIG0_CLR)	32	R/W	0002_2930h	5.3.3.2/ 783
3036_001C	Anadig 24MHz RC Osc. config0 Register (XTALOSC_RCOSC_CONFIG0_TOG)	32	R/W	0002_2930h	5.3.3.2/ 783
3036_0020	Anadig 24MHz RC Osc. config1 Register (XTALOSC_RCOSC_CONFIG1)	32	R/W	0000_02FAh	5.3.3.3/ 784
3036_0024	Anadig 24MHz RC Osc. config1 Register (XTALOSC_RCOSC_CONFIG1_SET)	32	R/W	0000_02FAh	5.3.3.3/ 784
3036_0028	Anadig 24MHz RC Osc. config1 Register (XTALOSC_RCOSC_CONFIG1_CLR)	32	R/W	0000_02FAh	5.3.3.3/ 784
3036_002C	Anadig 24MHz RC Osc. config1 Register (XTALOSC_RCOSC_CONFIG1_TOG)	32	R/W	0000_02FAh	5.3.3.3/ 784
3036_0030	Anadig 24MHz RC Osc. config2 Register (XTALOSC_RCOSC_CONFIG2)	32	R/W	0003_02EEh	5.3.3.4/ 785
3036_0034	Anadig 24MHz RC Osc. config2 Register (XTALOSC_RCOSC_CONFIG2_SET)	32	R/W	0003_02EEh	5.3.3.4/ 785
3036_0038	Anadig 24MHz RC Osc. config2 Register (XTALOSC_RCOSC_CONFIG2_CLR)	32	R/W	0003_02EEh	5.3.3.4/ 785
3036_003C	Anadig 24MHz RC Osc. config2 Register (XTALOSC_RCOSC_CONFIG2_TOG)	32	R/W	0003_02EEh	5.3.3.4/ 785
3036_0050	32K Oscillator Control Register (XTALOSC_OSC_32K)	32	R/W	0000_0000h	5.3.3.5/ 786

Table continues on the next page...

XTALOSC memory map (continued)

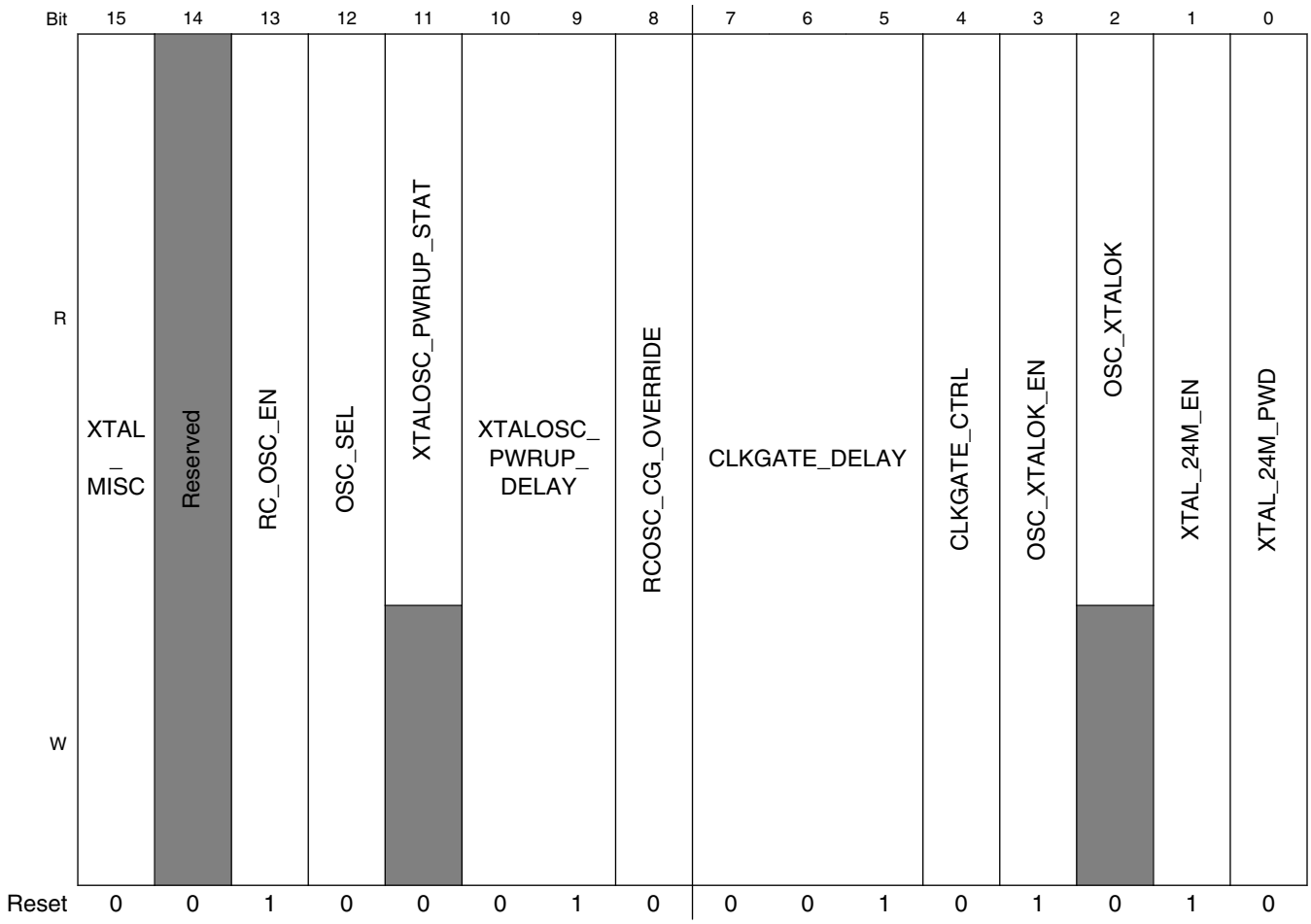
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3036_0054	32K Oscillator Control Register (XTALOSC_OSC_32K_SET)	32	R/W	0000_0000h	5.3.3.5/ 786
3036_0058	32K Oscillator Control Register (XTALOSC_OSC_32K_CLR)	32	R/W	0000_0000h	5.3.3.5/ 786
3036_005C	32K Oscillator Control Register (XTALOSC_OSC_32K_TOG)	32	R/W	0000_0000h	5.3.3.5/ 786

5.3.3.1 Anadig 24M Oscillator Control Register (XTALOSC_CTRL_24M*n*)

The enable bit is useful kill the output clocks while keeping the oscillator working. Naturally, it is good practice to disable the oscillator before powering down.

Address: 3036_0000h base + 0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	XTAL_MISC														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



XTALOSC_CTRL_24M*n* field descriptions

Field	Description
31 -	This field is reserved. Always set to zero (0).
30–15 XTAL_MISC	Misc control bits for 24m xtal osc
14 -	This field is reserved. Always set to zero (0).
13 RC_OSC_EN	RC osc. enable control. 0- turn off the RC osc. 24MHz clock; 1- turn on the RC osc. clock.
12 OSC_SEL	Select the source for the 24MHz clock. 0 - xtal osc.; 1 - RC osc.
11 XTALOSC_ PWRUP_STAT	Status of the 24MHz xtal oscillator. 0 - not stable; 1 - Stable and ready to use.
10–9 XTALOSC_ PWRUP_DELAY	Specifies the time delay between when the 24MHz xtal is powered up until it is stable and ready to use. 0 - 0.25ms; 1 - 0.5ms; 2 - 1ms; 3 - 2ms.

Table continues on the next page...

XTALOSC_CTRL_24M*n* field descriptions (continued)

Field	Description
8 RCOSC_CG_ OVERRIDE	For debug purposes only. This bit effects clock gating of certain digital logic clocked by the 24MHz clk.
7–5 CLKGATE_ DELAY	This field specifies the delay between powering up the xtal 24MHz clock and release the clock to the digital logic inside the analog block. 0x0 - 0.5 ms; 0x1 - 1 ms; 0x2 - 2 ms; 0x3 - 3 ms; 0x4 - 4 ms; 0x5 - 5 ms; 0x6 - 6 ms; 0x7 - 7 ms. Note: Do not change this field during a low power event. This is not a field that the user would normally need to modify.
4 CLKGATE_CTRL	This bit allows disabling the clock gate (always un-gated) for the xtal 24MHz clock that clocks the digital logic in the analog block. 0x0 - Allow the logic to automatically gate the clock when the xtal is powered down. 0x1 - Prevent the logic from ever gating off the clock. Under normal operating conditions, if the RCOsc is sourcing the 24MHz clk, this setting is ignored and the clock gate will not be gated. Note: Do not change this field during a low power event. This control is not a bit that the user would normally need to modify.
3 OSC_XTALOK_ EN	Enable the xtalok detection circuitry.
2 OSC_XTALOK	Status bit which signals that the output of the 24MHz crystal oscillator is stable.
1 XTAL_24M_EN	This field controls the clock gate at the output of the oscillator. 1 = enabled
0 XTAL_24M_PWD	This field powers down the 24M crystal oscillator if set true.

5.3.3.2 Anadig 24MHz RC Osc. config0 Register (XTALOSC_RCOSC_CONFIG0*n*)

This register is used to configure the 24MHz RC oscillator.

Address: 3036_0000h base + 10h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RC_OSC_PROG_CUR								Reserved				HYST_MINUS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HYST_PLUS				RC_OSC_PROG								TUNE_INVERT	TUNE_ BYPASS	TUNE_ ENABLE	TUNE_START
Reset	0	0	1	0	1	0	0	1	0	0	1	1	0	0	0	0

XTALOSC_RCOSC_CONFIG0*n* field descriptions

Field	Description
31–24 RC_OSC_ PROG_CUR	The current tuning value in use.
23–20 -	This field is reserved. Always set to zero (0).
19–16 HYST_MINUS	Negative hysteresis value. Subtracted from target value before comparison. This, along with HYST_PLUS creates a range.
15–12 HYST_PLUS	Positive hysteresis value. Added to target value before comparison. This, along with HYST_MINUS creates a range.
11–4 RC_OSC_PROG	RC osc. tuning values.
3 TUNE_INVERT	Invert the stepping of the calculated RC tuning value.
2 TUNE_BYPASS	Bypasses any calculated RC tuning value and uses the programmed register value.
1 TUNE_ENABLE	Enables the tuning logic to calculate new RC tuning values. Disabling essentially freezes the state of the calculation.
0 TUNE_START	Start/stop bit for the RC tuning calculation logic. If stopped the tuning logic is reset.

5.3.3.3 Anadig 24MHz RC Osc. config1 Register (XTALOSC_RCOSC_CONFIG1*n*)

This register is used to configure the 24MHz RC oscillator.

Address: 3036_0000h base + 20h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNT_RC_CUR												Reserved								COUNT_RC_TRG											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	0	1	0

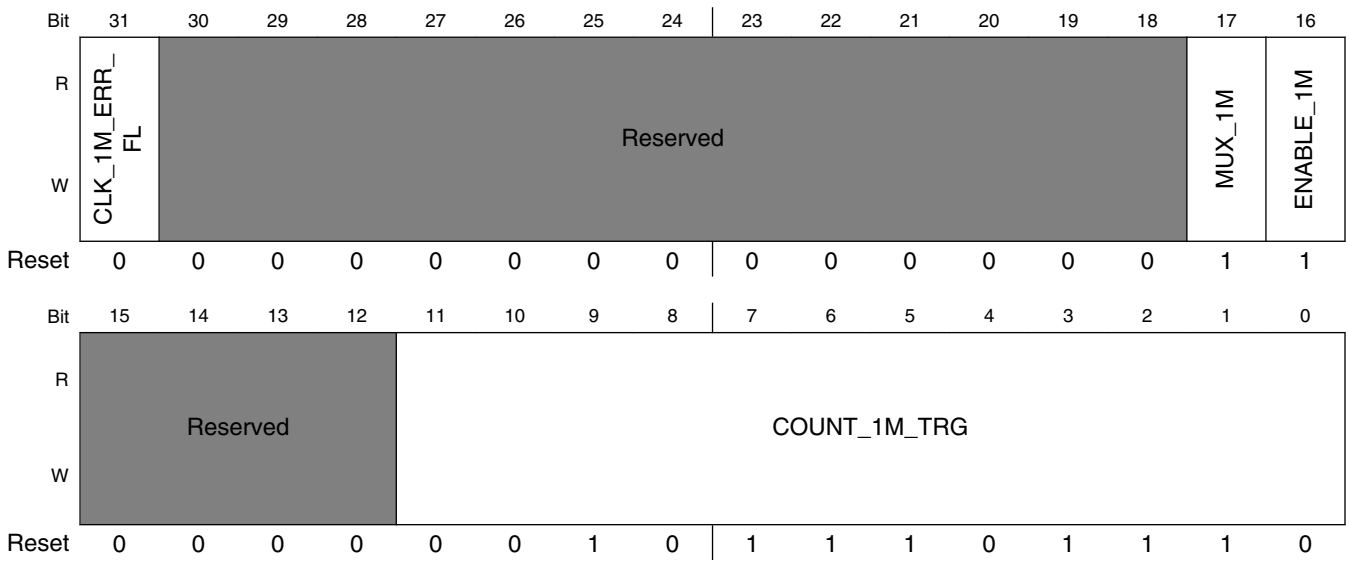
XTALOSC_RCOSC_CONFIG1*n* field descriptions

Field	Description
31–20 COUNT_RC_ CUR	The current tuning value in use.
19–12 -	This field is reserved. Always set to zero (0).
COUNT_RC_ TRG	The target count used to tune the RC OSC frequency. Essentially the number of desired RC Osc clock cycles within 1 32KHz clock cycle.

5.3.3.4 Anadig 24MHz RC Osc. config2 Register (XTALOSC_RCOSC_CONFIG2*n*)

This register is used to configure the 24MHz RC oscillator.

Address: 3036_0000h base + 30h offset + (4d × i), where i=0d to 3d



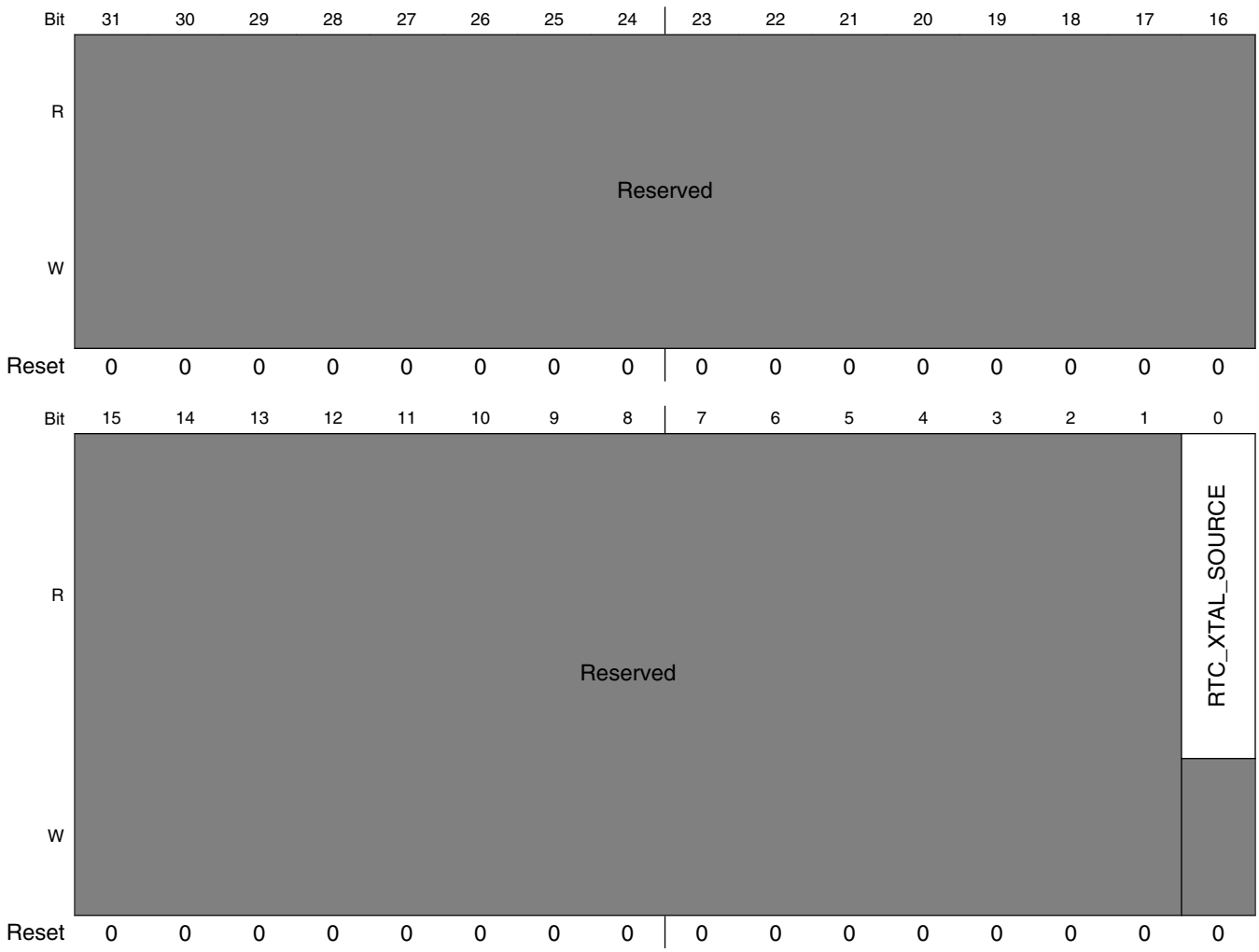
XTALOSC_RCOSC_CONFIG2*n* field descriptions

Field	Description
31 CLK_1M_ERR_FL	Flag indicates that the count_1m count wasn't reached within 1 32KHz period. This is intended as feedback to software that the OSC_CONFIG2_COUNT_1M_TRG value is too high for the RC Osc frequency. Write 1 to this field to clear.
30–18 -	This field is reserved. Always set to zero (0).
17 MUX_1M	Mux the corrected or uncorrected 1MHz clock to the output. 0 - .
16 ENABLE_1M	Enable the 1MHz clock output. 0 - disabled; 1 - enabled.
15–12 -	This field is reserved. Always set to zero (0).
COUNT_1M_TRG	The target count used to tune the RC OSC frequency. Essentially the number of desired RC Osc clock cycles within 1 32KHz clock cycles

5.3.3.5 32K Oscillator Control Register (XTALOSC_OSC_32K*n*)

The 32K low precision oscillator is only enabled when the frequency on the RTC xtal oscillator is determined to be out of range. This detection is done automatically.

Address: 3036_0000h base + 50h offset + (4d × i), where i=0d to 3d



XTALOSC_OSC_32K*n* field descriptions

Field	Description
31–1 -	This field is reserved. Always set to zero (0).
0 RTC_XTAL_ SOURCE	This field indicates which chip source is being used for the rtc clock. 0 - internal ring oscillator, 1 - rtc_xtal.

5.4 Power Management Unit (PMU)

5.4.1 Overview

The power management unit (PMU) is designed to simplify the external power interface. The power system can be split into the input power sources and their characteristics, the integrated power transforming and controlling elements, and the final load interconnection and requirements.

A typical power system uses the PMU is depicted in the following diagram.

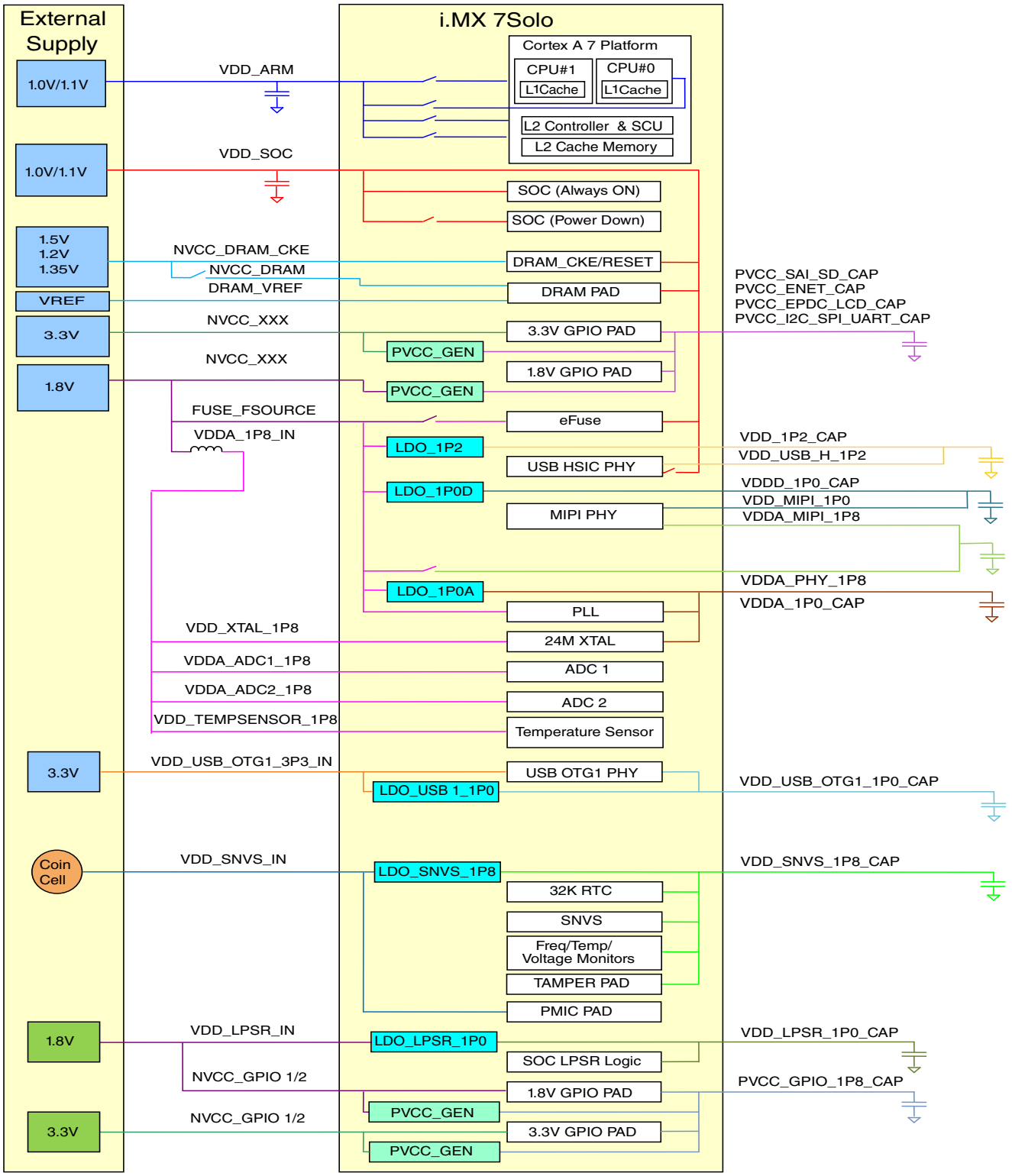


Figure 5-23. Power system overview

Using seven LDO regulators, the number of external supplies is greatly reduced. Not counting the backup coin and USB inputs, the number of external supplies is reduced to two. Missing from this external supply total is the number of necessary external supplies

to power the desired memory interface; that number varies depending on the type of external memory selected. Other supplies may also be necessary to supply the voltage to the different I/O power segments if their I/O voltages have to be different from what is provided above.

5.4.2 LDO Regulators

The PMU has seven LDO regulators, which have two basic modes.

- Power Gate—The regulation FET is switched off fully, limiting the current draw from the supply. The analog part of the regulator is powered down, limiting the power consumption. The output voltage falls to a level at which the residual leakage of the power FET balances with the leakage of the load. (TARG = 0x00)
- Analog regulation mode—The regulation FET is controlled such that the output voltage of the regulator equals the programmed target voltage. The target voltage is fully programmable in 25-mV steps.

5.4.2.1 LDO_1P0A

The LDO_1P0A regulator implements a programmable linear regulator function from the VDDA_1P8_IN supply. Nominal output voltage with default settings is 1.0V. This regulator supplies internal analog functions such as clock generation, ADCs, etc. A programmable brown out detector is included in the regulator that can be used by the system to determine when the load capability of the regulator is being exceeded to take the necessary steps. Current limiting can be enabled to allow for in rush current requirements during start-up, if needed. Active pull down can also be enabled for systems requiring this feature. See the Register PMU_REG_1P0A for the LDO control.

5.4.2.2 LDO_1P0D

The LDO_1P0D regulator implements a programmable linear regulator function from the VDDA_1P8_IN supply. This regulator is disabled out of reset, however, nominal output voltage with default settings is 1.0V once enabled. This regulator is designed to supply the digital portion of the MIPI and PCIe modules. A programmable brown out detector is included in the regulator that can be used by the system to determine when the load capability of the regulator is being exceeded to take the necessary steps. Current limiting can be enabled to allow for in rush current requirements during start-up, if needed. Active pull down can also be enabled for systems requiring this feature. See Register PMU_REG_1P0D for the LDO control.

5.4.2.3 LDO_1P2

The LDO_1P2 regulator implements a programmable linear regulator function from the VDDA_1P8_IN supply. This regulator is disabled out of reset, however, nominal output voltage with default settings is 1.0 V once enabled. This regulator is designed to supply the analog portion of USB HSIC module. A programmable brown out detector is included in the regulator that can be used by the system to determine when the load capability of the regulator is being exceeded to take the necessary steps. Current limiting can be enabled to allow for in rush current requirements during start-up, if needed. Active pull down can also be enabled for systems requiring this feature. See the Register PMU_REG_HSIC_1P2 for the LDO control.

5.4.2.4 LDO_LPSR_1P0

The LDO_LPSR_1P0 regulator implements a fixed low-power regulator function from the VDDA_LPSR_IN supply. This regulator is always on whenever VDD_SNVS_IN and VDDA_LPSR_IN are present and its nominal output voltage is 1.0V. This regulator is designed to provide power to the LPSR logic domain. See the Register PMU_REG_LPSR_1P0 for the LDO control.

5.4.3 LDO_USB1_1P0/LDO_USB2_1P0

The LDO_USB1_1P0/LDO_USB2_1P0 derives a 1V logic supply from the 3V USB input supply. It is utilized by the logic of the USB PHY. There are two instances of this regulator on the die, one for each USB. LDO_USB1_1P0 and LDO_USB2_1P0 are dedicated hardware. No software program interface is available for them.

5.4.4 LDO_SNVS_1P8

The LDO_SNVS_1P8 regulator takes the VDD_SNVS_IN supply and generates a norminal 1.8 V VDD_SNVS_CAP supply, which powers the real time clock and SNVS blocks.

5.4.5 PMU Memory Map/Register Definition

The register definitions that affect the behavior of the digital LDO regulators follow.

NOTE

Some of the registers are collections of bits that affect multiple components on the chip. Those that are not pertinent to this chapter have comments in the related register bitfields.

If a full description is desired, please consult the full register programming reference in the related block.

PMU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3036_0200	Anadig 1.0V A Regulator Control Register (PMU_REG_1P0A)	32	R/W	0000_1073h	5.4.5.1/ 793
3036_0204	Anadig 1.0V A Regulator Control Register (PMU_REG_1P0A_SET)	32	R/W	0000_1073h	5.4.5.1/ 793
3036_0208	Anadig 1.0V A Regulator Control Register (PMU_REG_1P0A_CLR)	32	R/W	0000_1073h	5.4.5.1/ 793
3036_020C	Anadig 1.0V A Regulator Control Register (PMU_REG_1P0A_TOG)	32	R/W	0000_1073h	5.4.5.1/ 793
3036_0210	Anadig 1.0V D Regulator Control Register (PMU_REG_1P0D)	32	R/W	0000_1078h	5.4.5.2/ 795
3036_0214	Anadig 1.0V D Regulator Control Register (PMU_REG_1P0D_SET)	32	R/W	0000_1078h	5.4.5.2/ 795
3036_0218	Anadig 1.0V D Regulator Control Register (PMU_REG_1P0D_CLR)	32	R/W	0000_1078h	5.4.5.2/ 795
3036_021C	Anadig 1.0V D Regulator Control Register (PMU_REG_1P0D_TOG)	32	R/W	0000_1078h	5.4.5.2/ 795
3036_0220	Anadig 1.2V HSIC Regulator Control Register (PMU_REG_HSIC_1P2)	32	R/W	0000_1878h	5.4.5.3/ 797
3036_0224	Anadig 1.2V HSIC Regulator Control Register (PMU_REG_HSIC_1P2_SET)	32	R/W	0000_1878h	5.4.5.3/ 797
3036_0228	Anadig 1.2V HSIC Regulator Control Register (PMU_REG_HSIC_1P2_CLR)	32	R/W	0000_1878h	5.4.5.3/ 797
3036_022C	Anadig 1.2V HSIC Regulator Control Register (PMU_REG_HSIC_1P2_TOG)	32	R/W	0000_1878h	5.4.5.3/ 797
3036_0230	Anadig 1.0V Low Power State Retention Regulator Control Register (PMU_REG_LPSR_1P0)	32	R/W	0000_10FBh	5.4.5.4/ 799
3036_0234	Anadig 1.0V Low Power State Retention Regulator Control Register (PMU_REG_LPSR_1P0_SET)	32	R/W	0000_10FBh	5.4.5.4/ 799

Table continues on the next page...

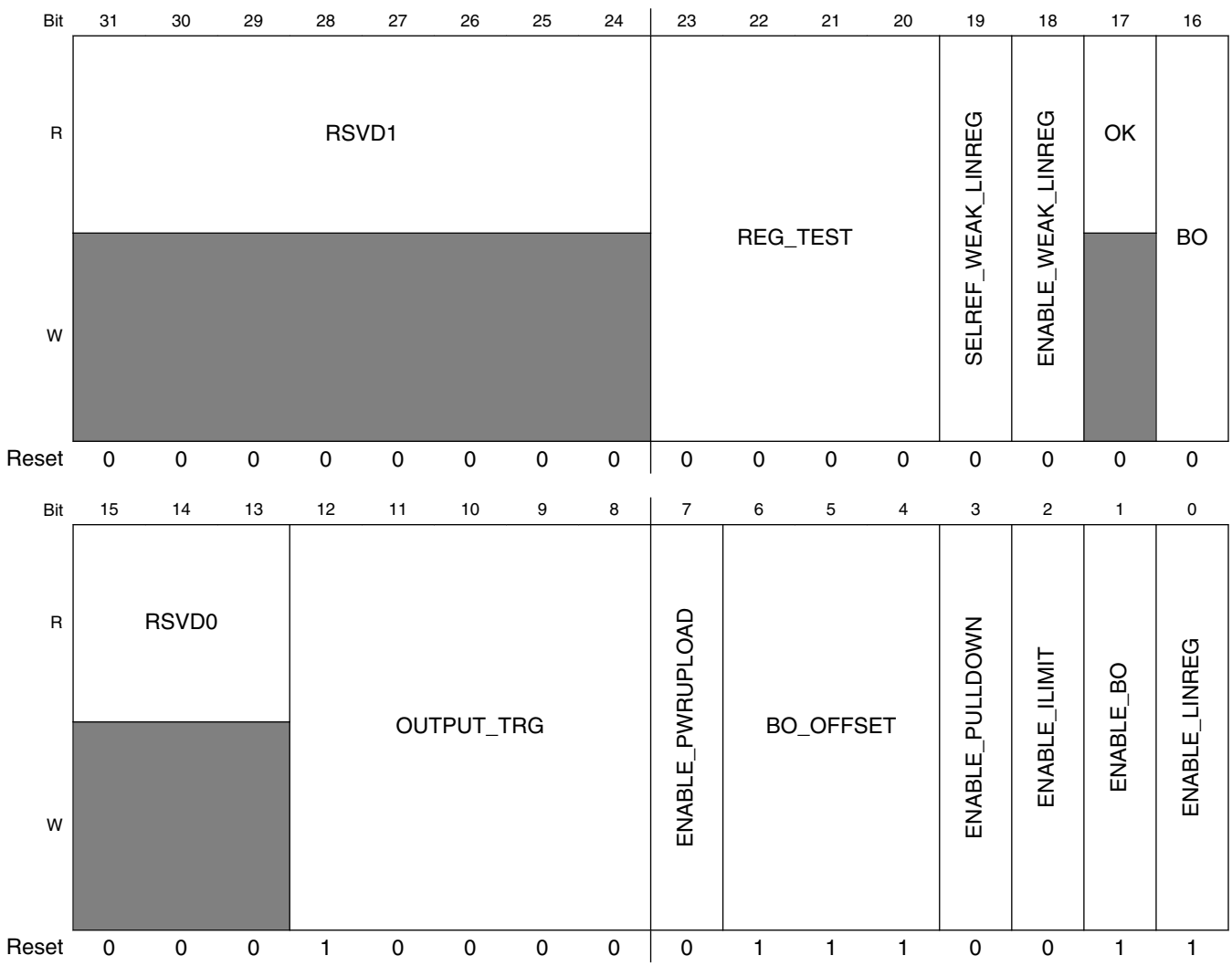
PMU memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3036_0238	Anadig 1.0V Low Power State Retention Regulator Control Register (PMU_REG_LPSR_1P0_CLR)	32	R/W	0000_10FBh	5.4.5.4/ 799
3036_023C	Anadig 1.0V Low Power State Retention Regulator Control Register (PMU_REG_LPSR_1P0_TOG)	32	R/W	0000_10FBh	5.4.5.4/ 799
3036_0270	Anadig Reference Analog Control and Status Register (PMU_REF)	32	R/W	0000_2000h	5.4.5.5/ 801
3036_0274	Anadig Reference Analog Control and Status Register (PMU_REF_SET)	32	R/W	0000_2000h	5.4.5.5/ 801
3036_0278	Anadig Reference Analog Control and Status Register (PMU_REF_CLR)	32	R/W	0000_2000h	5.4.5.5/ 801
3036_027C	Anadig Reference Analog Control and Status Register (PMU_REF_TOG)	32	R/W	0000_2000h	5.4.5.5/ 801
3036_0330	Anadig Low Power Control Register (PMU_LOWPWR_CTRL)	32	R/W	0000_3F00h	5.4.5.6/ 803
3036_0334	Anadig Low Power Control Register (PMU_LOWPWR_CTRL_SET)	32	R/W	0000_3F00h	5.4.5.6/ 803
3036_0338	Anadig Low Power Control Register (PMU_LOWPWR_CTRL_CLR)	32	R/W	0000_3F00h	5.4.5.6/ 803
3036_033C	Anadig Low Power Control Register (PMU_LOWPWR_CTRL_TOG)	32	R/W	0000_3F00h	5.4.5.6/ 803

5.4.5.1 Anadig 1.0V A Regulator Control Register (PMU_REG_1P0A*n*)

This register defines the control and status bits for the 1.1V regulator. This regulator is designed to power the digital portions of the analog cells.

Address: 3036_0000h base + 200h offset + (4d × i), where i=0d to 3d



PMU_REG_1P0A*n* field descriptions

Field	Description
31–24 RSVD1	Always set to zero (0).
23–20 REG_TEST	Test bits to monitor different analog signals though anamux.

Table continues on the next page...

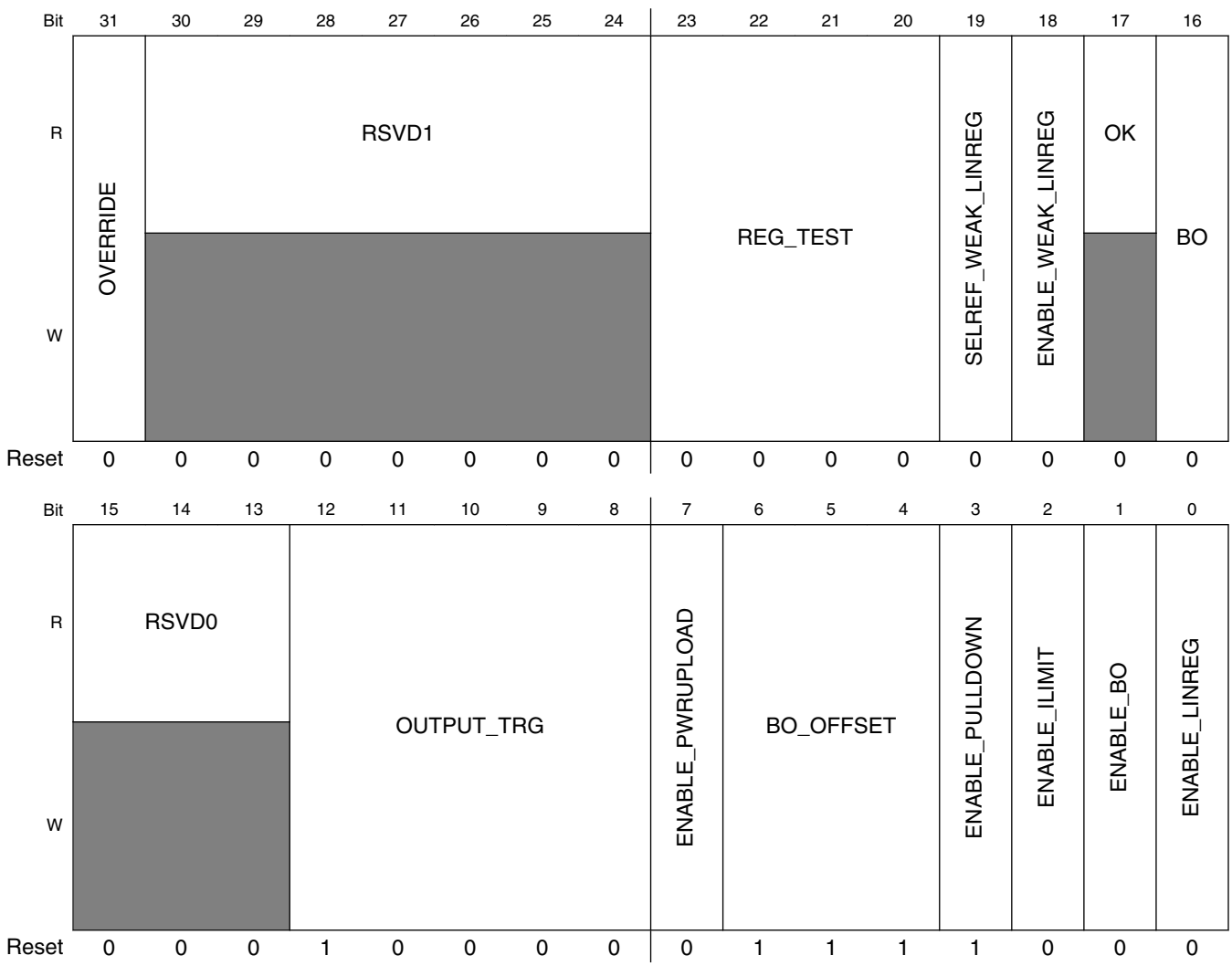
PMU_REG_1P0A*n* field descriptions (continued)

Field	Description
19 SELREF_ WEAK_LINREG	Selects the source for the reference voltage of the weak 1p1 regulator: 0x0=weak-linreg output tracks low-power-bandgap voltage, 0x1=weak-linreg output tracks VDD_SOC_CAP voltage.
18 ENABLE_ WEAK_LINREG	Enables the weak 1p1 regulator. This regulator can be used when the main 1p1 regulator is disabled, under low-power conditions.
17 OK	Status bit that signals when the regulator output is ok.
16 BO	Interrupt/status bit that signals when a brown-out is detected on the regulator output. Sticky bit, write 1 to this field to clear.
15–13 RSVD0	Always set to zero (0).
12–8 OUTPUT_TRG	Control bits to adjust the regulator output voltage in 25mV steps. 0x18 - 1.2V; 0x10 - 1.0V; 0x08 - 0.8V. Note that the programmable output of the regulator excedes the maximum and minimum supply limits for the chip.
7 ENABLE_ PWRUPLOAD	Always set to zero (0).
6–4 BO_OFFSET	Control bits to adjust the regulator brown-out offset voltage in 25mV steps. The brown-out level is relative to the programmed target voltage. The reset brown-out offset is 175mV below the programmed target level.
3 ENABLE_ PULLDOWN	Control bit to enable the pull-down circuitry in the regulator
2 ENABLE_ILIMIT	Control bit to enable the current-limit circuitry in the regulator.
1 ENABLE_BO	Control bit to enable the brown-out circuitry in the regulator.
0 ENABLE_ LINREG	Control bit to enable the regulator output.

5.4.5.2 Anadig 1.0V D Regulator Control Register (PMU_REG_1P0D*n*)

This register defines the control and status bits for the 1.1V regulator. This regulator is designed to power the digital portions of the analog cells.

Address: 3036_0000h base + 210h offset + (4d × i), where i=0d to 3d



PMU_REG_1P0D*n* field descriptions

Field	Description
31 OVERRIDE	The OVERRIDE bit allows the GPC module to automatically override portions of the register.
30–24 RSVD1	Always set to zero (0).

Table continues on the next page...

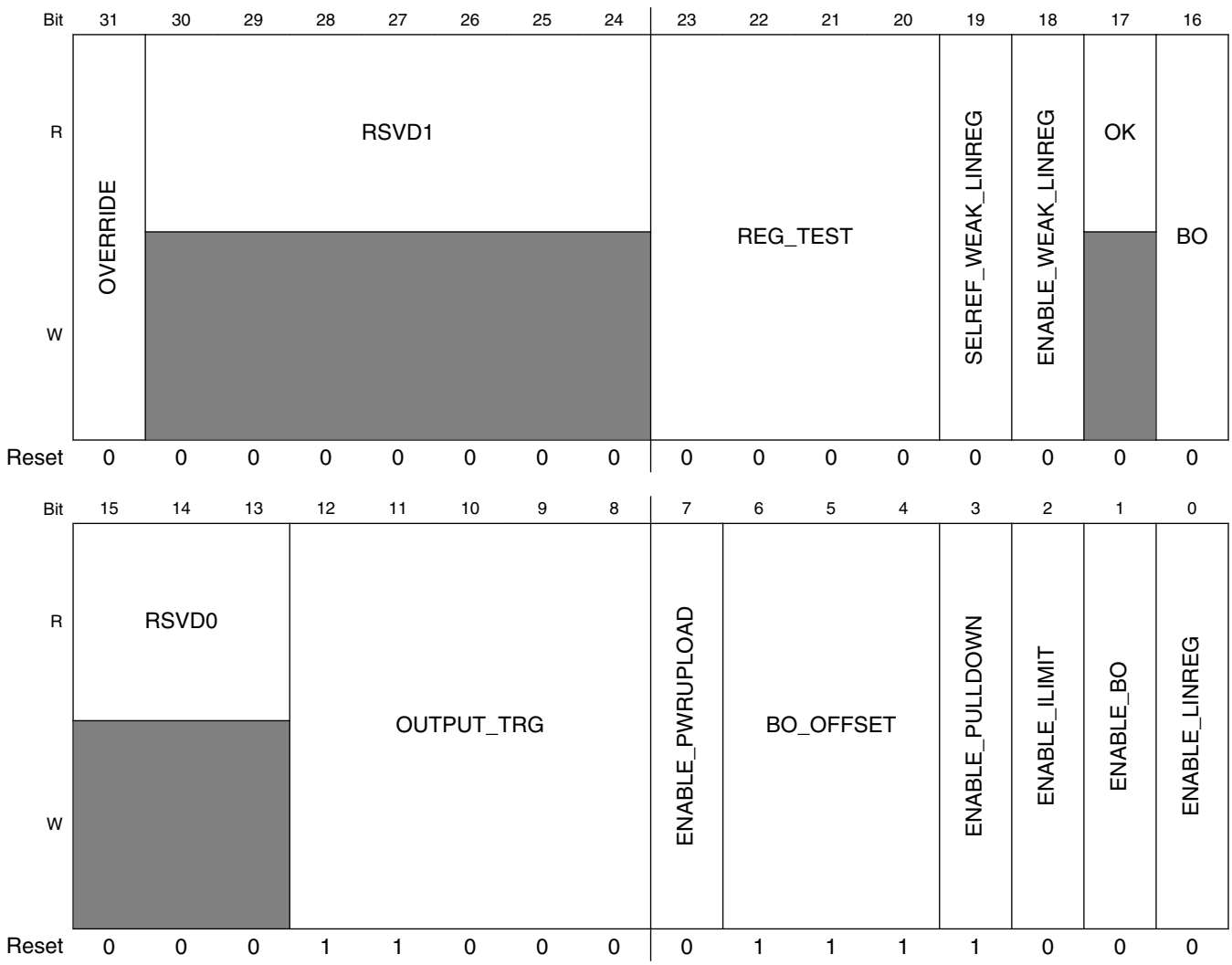
PMU_REG_1P0D*n* field descriptions (continued)

Field	Description
23–20 REG_TEST	Test bits to monitor different analog signals though anamux.
19 SELREF_ WEAK_LINREG	Selects the source for the reference voltage of the weak 1p1 regulator: 0x0=weak-linreg output tracks low-power-bandgap voltage, 0x1=weak-linreg output tracks VDD_SOC_CAP voltage.
18 ENABLE_ WEAK_LINREG	Enables the weak 1p1 regulator. This regulator can be used when the main 1p1 regulator is disabled, under low-power conditions.
17 OK	Status bit that signals when the regulator output is ok.
16 BO	Interrupt/status bit that signals when a brown-out is detected on the regulator output. Sticky bit, write 1 to this field to clear.
15–13 RSVD0	Always set to zero (0).
12–8 OUTPUT_TRG	Control bits to adjust the regulator output voltage in 25mV steps. 0x18 - 1.2V; 0x10 - 1.0V; 0x08 - 0.8V. Note that the programmable output of the regulator excedes the maximum and minimum supply limits for the chip.
7 ENABLE_ PWRUPLOAD	Always set to zero (0).
6–4 BO_OFFSET	Control bits to adjust the regulator brown-out offset voltage in 25mV steps. The brown-out level is relative to the programmed target voltage. The reset brown-out offset is 175mV below the programmed target code.
3 ENABLE_ PULLDOWN	Control bit to enable the pull-down circuitry in the regulator
2 ENABLE_ILIMIT	Control bit to enable the current-limit circuitry in the regulator.
1 ENABLE_BO	Control bit to enable the brown-out circuitry in the regulator.
0 ENABLE_ LINREG	Control bit to enable the regulator output.

5.4.5.3 Anadig 1.2V HSIC Regulator Control Register (PMU_REG_HSIC_1P2*n*)

This register defines the control and status bits for the 1.1V regulator. This regulator is designed to power the digital portions of the analog cells.

Address: 3036_0000h base + 220h offset + (4d × i), where i=0d to 3d



PMU_REG_HSIC_1P2*n* field descriptions

Field	Description
31 OVERRIDE	The OVERRIDE bit allows the GPC module to automatically override portions of the register.
30–24 RSVD1	Always set to zero (0).

Table continues on the next page...

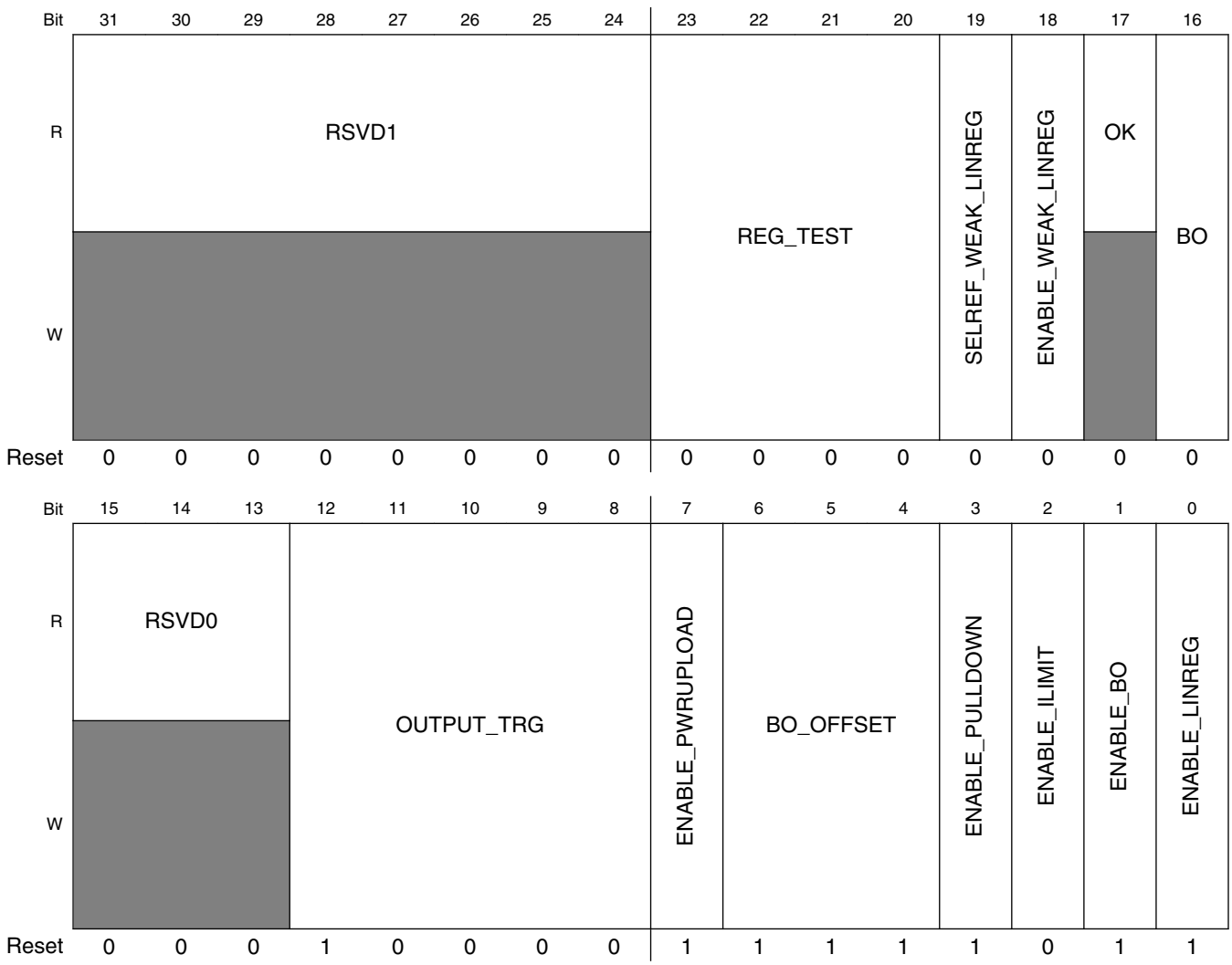
PMU_REG_HSIC_1P2*n* field descriptions (continued)

Field	Description
23–20 REG_TEST	Test bits to monitor different analog signals though anamux.
19 SELREF_ WEAK_LINREG	Selects the source for the reference voltage of the weak 1p1 regulator: 0x0=weak-linreg output tracks low-power-bandgap voltage, 0x1=weak-linreg output tracks VDD_SOC_CAP voltage.
18 ENABLE_ WEAK_LINREG	Enables the weak 1p1 regulator. This regulator can be used when the main 1p1 regulator is disabled, under low-power conditions.
17 OK	Status bit that signals when the regulator output is ok.
16 BO	Interrupt/status bit that signals when a brown-out is detected on the regulator output. Sticky bit, write 1 to this field to clear.
15–13 RSVD0	Always set to zero (0).
12–8 OUTPUT_TRG	Control bits to adjust the regulator output voltage in 25mV steps. 0x1c - 1.3V; 0x18 - 1.2V; 0x14 - 1.1V. Note that the programmable output of the regulator excedes the maximum and minimum supply limits for the chip.
7 ENABLE_ PWRUPLOAD	Always set to zero (0).
6–4 BO_OFFSET	Control bits to adjust the regulator brown-out offset voltage in 25mV steps. The brown-out level is relative to the programmed target voltage. The reset brown-out offset is 175mV below the programmed target code.
3 ENABLE_ PULLDOWN	Control bit to enable the pull-down circuitry in the regulator
2 ENABLE_ILIMIT	Control bit to enable the current-limit circuitry in the regulator.
1 ENABLE_BO	Control bit to enable the brown-out circuitry in the regulator.
0 ENABLE_ LINREG	Control bit to enable the regulator output.

5.4.5.4 Anadig 1.0V Low Power State Retention Regulator Control Register (PMU_REG_LPSR_1P0*n*)

This register defines the control and status bits for the 1.1V regulator. This regulator is designed to power the digital portions of the analog cells.

Address: 3036_0000h base + 230h offset + (4d × i), where i=0d to 3d



PMU_REG_LPSR_1P0*n* field descriptions

Field	Description
31–24 RSVD1	Always set to zero (0).
23–20 REG_TEST	Not used

Table continues on the next page...

PMU_REG_LPSR_1P0*n* field descriptions (continued)

Field	Description
19 SELREF_ WEAK_LINREG	Not used
18 ENABLE_ WEAK_LINREG	Not used, always on.
17 OK	Always 1 after reset
16 BO	Not used
15–13 RSVD0	Always set to zero (0).
12–8 OUTPUT_TRG	Not used.
7 ENABLE_ PWRUPLOAD	Enables a ~100kohm load to ground. Typically software would clear this if not necessary.
6–4 BO_OFFSET	Not used
3 ENABLE_ PULLDOWN	Enables a ~50kohm load to ground. Typically software would clear this if not necessary.
2 ENABLE_ILIMIT	Not used
1 ENABLE_BO	Not used
0 ENABLE_ LINREG	Always enabled, field not used

5.4.5.5 Anadig Reference Analog Control and Status Register (PMU_REF*n*)

This register defines the control and status bits for miscellaneous analog blocks.

Address: 3036_0000h base + 270h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1		REFTOP_LINREGREF_EN	REFTOP_IBIAS_OFF	LPBG_TEST	LPBG_SEL	REFTOP_BIAS_TST		REFTOP_VBGUP	REFTOP_VBGADJ			REFTOP_SELFBIASOFF	REFTOP_LOWPOWER	REFTOP_PWDVBGUP	REFTOP_PWD
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

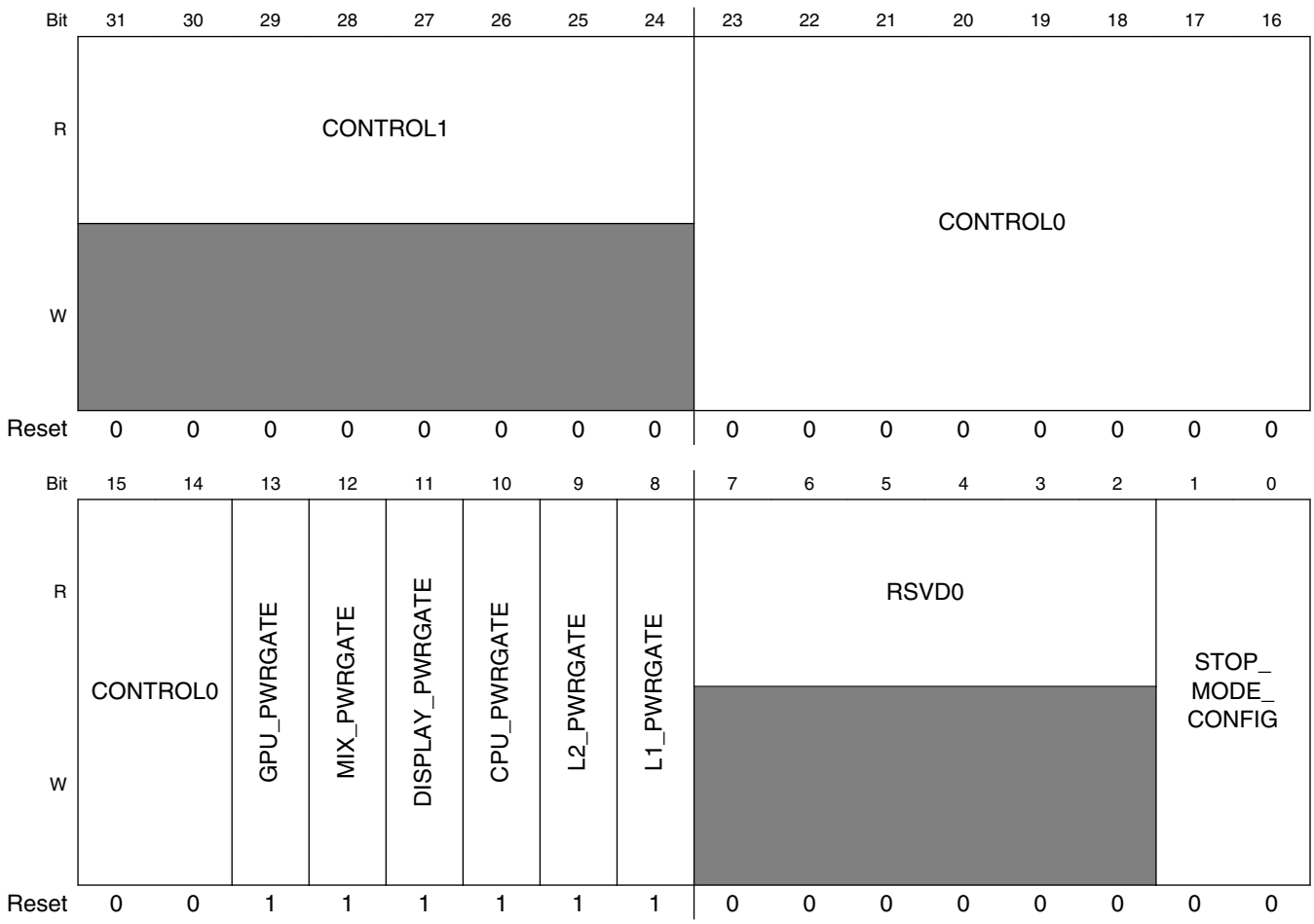
PMU_REF*n* field descriptions

Field	Description
31–14 RSVD1	Always set to zero (0).
13 REFTOP_ LINREGREF_EN	Enables buffer from the main bandgap for use with the linear regulators. This bit is useful to save power when the regulators are configured to use the low power bandgap voltage.
12 REFTOP_IBIAS_ OFF	Low power reftop ibias disable. This bit switches the bias current source from this reference to the low power bandgap.
11 LPBG_TEST	Low power bandgap test bit.. Freescale debug/test purposes only.
10 LPBG_SEL	Bandgap select. 0 - normal power bandgap; 1 - low power bandgap.
9–8 REFTOP_BIAS_ TST	Freescale debug/test bits 0x0 - nom, 0x1 - 90% Ibias, 0x2 - 80% Ibias, 0x3 110% Ibias
7 REFTOP_ VBGUP	Status bit which signals that the analog bandgap voltage is up and stable.
6–4 REFTOP_ VBGADJ	Freescale debug/test bits
3 REFTOP_ SELFBIASOFF	Control bit to disable the self-bias circuit in the analog bandgap. The self-bias circuit is used by the bandgap during startup. This bit should be set after the bandgap has stabilized and is necessary for best noise performance of analog blocks using the outputs of the bandgap.
2 REFTOP_ LOWPOWER	Freescale debug/test only. Control bit to enable the low-power mode in the analog bandgap.
1 REFTOP_ PWDVBGUP	Control bit to power-down the VBG-up detection circuitry in the analog bandgap.
0 REFTOP_PWD	Control bit to power-down the analog bandgap reference circuitry.

5.4.5.6 Anadig Low Power Control Register (PMU_LOWPWR_CTRL*n*)

This register defines fields for low power capabilities of the analog.

Address: 3036_0000h base + 330h offset + (4d × i), where i=0d to 3d



PMU_LOWPWR_CTRL*n* field descriptions

Field	Description
31–24 CONTROL1	Reserved control bits.
23–14 CONTROL0	Reserved control bits.
13 GPU_PWRGATE	GPU power gate control. Used as software mask. Set to zero to force ungated.
12 MIX_PWRGATE	Display power gate control. Used as software mask. Set to zero to force ungated.

Table continues on the next page...

PMU_LOWPWR_CTRL*n* field descriptions (continued)

Field	Description
11 DISPLAY_ PWRGATE	Display power gate control. Used as software mask. Set to zero to force ungated.
10 CPU_PWRGATE	GPU power gate control. Used as software mask. Set to zero to force ungated.
9 L2_PWRGATE	L2 power gate control. Used as software mask. Set to zero to force ungated.
8 L1_PWRGATE	L1 power gate control. Used as software mask. Set to zero to force ungated.
7–2 RSVD0	Always set to zero (0).
STOP_MODE_ CONFIG	Configure the analog behavior in stop mode. 0 All analog except RTC powered down on stop mode assertion. • XtalOsc = on • RCOsc = off 1 Certain analog functions, such as certain regulators left up. • XtalOsc = on • RCOsc = off 2 • XtalOsc = off • RCOsc = on • Old BG = on • New BG = off 3 • XtalOsc = off • RCOsc = on • Old BG = off • New BG = on

5.5 General Power Controller (GPC)

5.5.1 Overview

The General Power Controller (GPC) module controls the following functions:

- Provide low power mode control for A7 and M4 platform
- Provide Power domain management all ARM and SOC power domain
- Provide domain control mechanism based on A7 and M4 CPU domain
- Provide handshake with CCM for clock management in low power mode
- Provide handshake with SRC for power down and power up sequence
- Provide handshake with Analog for Deep Sleep Mode control

5.5.2 Features

The General Power Controller (GPC) module controls the following functions:

- Support programmable feature for WAIT/STOP/DSM low power mode
- Support time slot based power domain control
- Support flexible sleep and wakeup condition
- Support domain control for multi CPU platforms system
- All register accessed by IP bus
- Interface for the following IPs:
    - CCM – clock controller module
    - SRC – system reset controller
    - ANALOG – miscellaneous analog control

5.5.3 Block Diagram

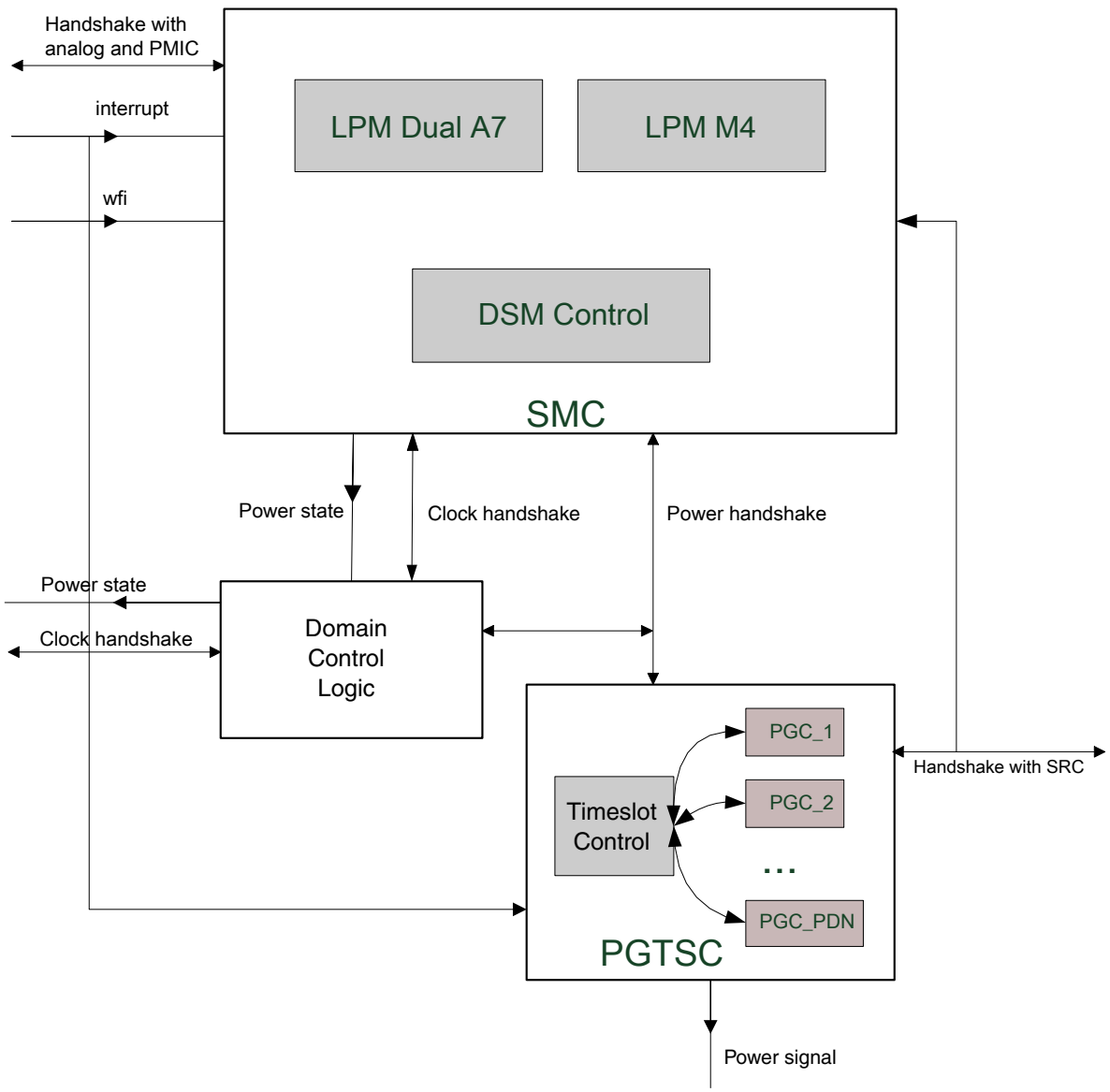


Figure 5-24. GPC Block Diagram

The GPC module contains two sub-modules: System Mode Controller (SMC) and Power Gating Time Slot Control (PGTSC):

- GPC Top: the top level GPC. It also includes the top memory map and registers, domain control information, and memory low power control.
- System Mode Controller (SMC):
  - The SMC supports two low power modes (LPM), WAIT and STOP. Each LPM corresponds to one mode for A7 platform and one mode for M4 platform.

- SMC controls the power sequence in Deep Sleep Mode (DSM)
- SMC support the power up and power down of A7 core0/core1 by IRQ/WFI signals without LPM triggered
- SMC can translate the LPM request for A7 and M4 platform to power up and power down request to PGTSC.
- Power Gating Time Slot Control (PGTSC):
  - The Power Gating Controller (PGC) is a power management component that controls the power-down and power-up sequencing of individual subsystems. For subsystems to be completely powered down in low power modes, a specific sequence of power control signals must be followed. The sequence timing is programmable using the PGC control registers.
  - There are 9 PGCs in the chip, all of them can be power down/up with a software trigger and all of them can be mapped to 10 timing slots and power-up and power down by request from SMC.

5.5.4 Functional Description

5.5.4.1 RUN mode

This is the normal/functional operating mode. In this mode, the CPU runs in its normal operational mode.

5.5.4.2 Low power mode

There are two CPU platforms (each of them represents a CPU domain): Dual core Cortex A7 platform and Cortex M4 platform. Each platform supports two low power modes:WAIT mode and STOP mode.

5.5.4.2.1 WAIT mode

In this low power mode:

- LPCG can be defined to be shut off or not in wait mode for each CPU domain
- PLL can be defined to be shut off or not in wait mode for each CPU domain

NOTE

The PLLs will only been closed in non-fast wake-up mode, relevant bit are

GPC_SLPCR[EN_A7_FASTWUP_WAIT_MODE] and
GPC_SLPCR[EN_M4_FASTWUP_WAIT_MODE]

- CPU clock can be defined been shut off or not in wait for each CPU platform. (GPC_LPCR_A7_BSC[CPU_CLK_ON_LPM] and GPC_LPCR_M4[CPU_CLK_ON_LPM])
- Power of different power domain can be defined be shut off or not in wait mode for each platform domain
- Some peripherals may go to wait mode along with A7 or M4 platform.

5.5.4.2.2 STOP mode

In this low power mode:

- LPCG can be defined been shut off or not in stop mode for each CPU domain
- PLL can be defined been shut off or not in stop mode for each CPU domain

NOTE

The PLLs will only been closed in non-fast wake-up mode, relevant bit are
GPC_SLPCR[EN_A7_FASTWUP_STOP_MODE] and
GPC_SLPCR[EN_M4_FASTWUP_STOP_MODE]

- CPU clock can be defined been shut off or not in stop for each CPU (GPC_LPCR_A7_BSC[CPU_CLK_ON_LPM] and GPC_LPCR_M4[CPU_CLK_ON_LPM])
- Power of different power domain can be defined be shut off or not in stop mode for each platform domain
- Some peripherals may go to stop mode along with A7 or M4 platform.

5.5.4.3 Deep Sleep Mode

The Deep Sleep Mode (DSM) is a system low power mode.

In this mode:

- On-chip OSC can be defined to be shut off or not in DSM (GPC_SLPCR[SBYOS])
- PMIC can be defined to be stand-by mode or not in DSM (GPC_SLPCR[VSTBY])
- Regulator can be defined to be BYPASS mode or not in DSM (GPC_SLPCR[RBC_EN])
- Memory can be defined to go to retention mode or not in DSM(GPC_MLPCR[MEMLP_CTL_DIS])
- A7 platform power (VDD_ARM) can be defined to be shut off or not in DSM.

NOTE

CCM configuration must make sure close all PLLs before system goes to DSM

NOTE

For the SoC to correctly power up after entering DSM, CCM_PLL_CTRLx must not be set to 0x0 or 0x3 for any domain in use.

5.5.4.4 LPM Sleep Process

CPU platform will go to WAIT/STOP under the following conditions:

- LPM registers (GPC_LPCR_A7_BSC[LPM0], GPC_LPCR_A7_BSC[LPM1], GPC_LPCR_M4[LPM]) are set to WAIT or STOP.

NOTE

Since A7 platform has two cores, the so each core has its own LPM register: GPC_LPCR_A7_BSC[LPM0] and GPC_LPCR_A7_BSC[LPM1]. The unified LPM of A7 will be generated with the lower LPM of the two Cores.

- Asserting the WFI signal will trigger CPU sleep process. There are three WFIs that come from A7 platform: WFI_core0, WFI_core1 and WFI_scu. The A7 platform will go to LPM when all three WFIs are asserted. If the GPC_LPCR_A7_AD[EN_C0_WFI_PDN] or GPC_LPCR_A7_AD[EN_C1_WFI_PDN] bit is set, the LPM trigger condition will be a little different, please refer to Power control for A7 Platform for more information. Only one WFI comes from M4 platform. The M4 platform will go to LPM when WFI_m4 asserted.

NOTE

WFI condition can be masked by register bits GPC_LPCR_A7_BSC[MASK_n_WFI] and GPC_LPCR_M4[MASK_M4_WFI]

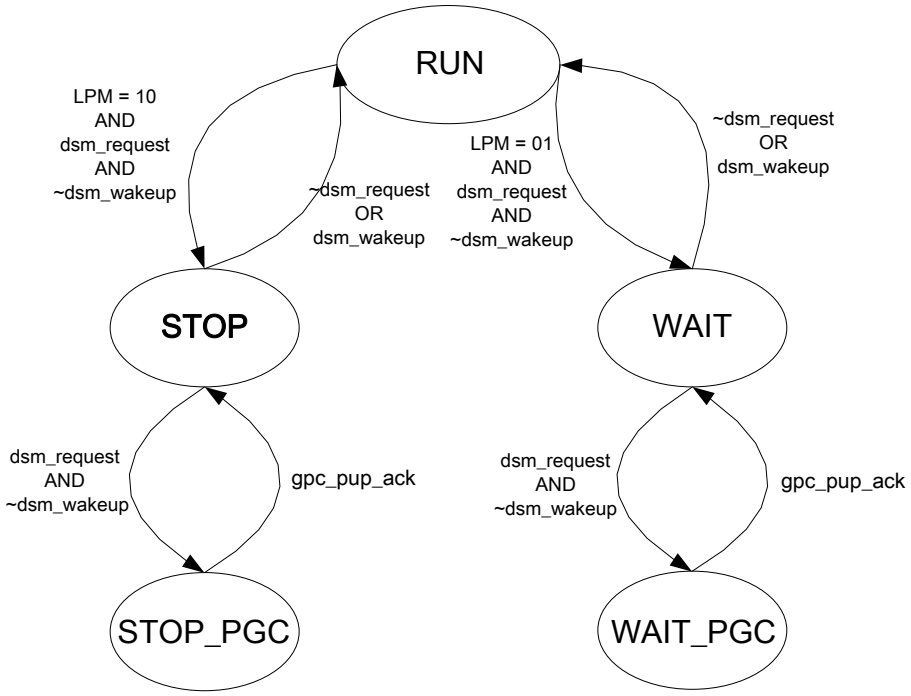


Figure 5-25. LPM transition inside CPU platform

System will go to DSM under the following conditions:

- Both A7 and M4 are STOP mode.
- Both GPC_SLPCR[EN_A7_FASTWUP_STOP_MODE] and GPC_SLPCR[EN_M4_FASTWUP_STOP_MODE] are not set.
- GPC_SLPCR[EN_DSM] is set.

NOTE

If GPC_LPCR_M4[MASK_DSM_TRIGGER] is set, the system will go to DSM when A7 goes STOP mode and GPC_SLPCR[EN_A7_FASTWUP_STOP_MODE] is not set. If GPC_LPCR_A7_BSC[MASK_DSM_TRIGGER] is set, the system will go to DSM when M4 goes to STOP mode and GPC_SLPCR[EN_M4_FASTWUP_STOP_MODE] not set. GPC_LPCR_M4[MASK_DSM_TRIGGER] and GPC_LPCR_A7_BSC[MASK_DSM_TRIGGER] cannot be set at the same time.

5.5.4.5 LPM Wake Up Process

DSM, STOP, WAIT mode will be woken up by interrupts:

- The two CPU platforms share the same IRQ sources in this chip. Software can use GPC_IMRn_CORE0_A7, GPC_IMRn_CORE1_A7, and GPC_IMRn_M4 to separate the 128 bits IRQ sources to A7 core0, A7 core1, and M4 platform.
- The A7 core0 and core1 IRQ can also be from GIC source (defined by GPC_LPCR_A7_BSC[IRQ_SRC_C1] and GPC_LPCR_A7_BSC[IRQ_SRC_C0]) and if it is chosen from GIC source the GPC_IMRn_x_A7 will lose its function (please refer to Power control for A7 Platform for more information).
- Interrupts for both A7 and M4 will cause the system wake up from DSM, A7 interrupt will wake up A7 from LPM, M4 interrupt will wake up M4 from LPM.

5.5.5 Power Gating Controller (PGC) Overview

The Power Gating Controller (PGC) is a power management component that controls the power-down and power-up sequencing of individual subsystems. For subsystems to be completely powered down in low power modes, a specific sequence of power control signals must be followed. The sequence timing is programmable using the PGC control registers.

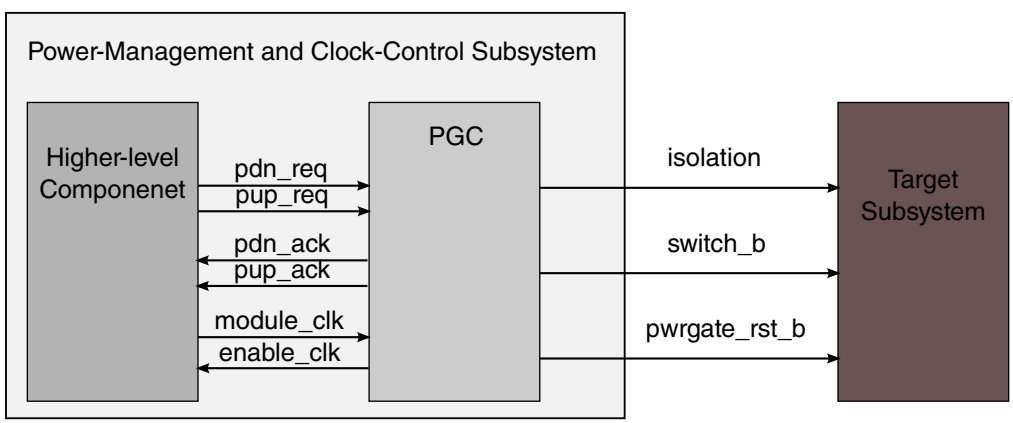


Figure 5-26. Power Gating Controller (PGC)

5.5.5.1 PGC power domains

The following table lists the PGCs in the chip and the corresponding power domain.

PGC type	Power domain
PGC_C0	Core 0 of A7 platform
PGC_C1	Core1 of A7 platform
PGC_SCU	SCU/L2 cache RAM of A7 platform

Table continues on the next page...

PGC type	Power domain
PGC_MF	Fastmix and Megamix (see table below)
PGC_OTG1	USB OTG1 PHY
PGC_MIPI	MIPI PHY
PGC_HSIC	USB HSIC PHY

NOTE

M4 is inside fastmix and doesn't have its own PGC.

Table 5-20. PGC_MF Power Domain

Megamix	Fastmix
QSPI	DDRC
EIM	OCRAM
ENET	TZASC
USB2 (OTG, HSIC)	CSI
uSDHC x3 (1-3)	CM4
NAND (GPMI, BCH)	PXP
APBHDMA	LCDIF
SAI x3 (1-3)	MIPI CSI2
eCSPI x3 (1-3)	MIPI DSI
SIMv2	
MQS	
UART x3 (1-3)	
SDMA	
SPBA	

5.5.5.2 Trigger to PGC: Hardware and Software Requests

All PGCs(except fastmix/megamix PGC) can be power up/down by hardware or software request. Fastmix/megamix PGC can only be power up/down by hardware request.

The LPM controller for each platform can generate hardware power down or power up request. All hardware requests will be mapped to "timeslot controller" before they goes to relevant PGC (see "Time slot control for PGCs" for more information).

The CPU can also generate software power up or power down request to relevant PGCs. The software trigger will not be mapped to timeslot control. If there are PGCs in software PDN/PUP sequence the request from LPM will be masked. The software trigger will also be failed if the timeslot control is in "busy" state.

All power up/down request to PGCs will be mapped to domain control module (see “Domain control for PGCs ”).

PGC_C0 and PGC_C1 can be triggered by its own “WFI/IRQ” without LPM trigger and time slot. (see “Power control for A7 platform” for more information)

5.5.5.3 Time slot control mechanism for PGCs

GPC uses a time slot controller to control the PGC sub-systems, such as the PGC in CPU0, CPU1, MIX, PCIE PHY, etc. We use it for below reasons when system wakes from low power mode.

1. Support flexible power down/up sequence for different sub-system
2. Sub-system can be power up in different slot to avoid large ramping up current in case they start ramping up at the same time

There are a total of 10 time slots used in the chip, one or more PGCs are used for power up or power down in each of these slots. The time slot controller will sample power up/down requests at slot0. If there are power up/down requests from SMC, it will scan from slot0 to slot9. When the scan process comes to one slot which has a defined power up or power down for one or more PGCs (defined by “SLTn_CFG”), the power up/down sequence of relevant PGCs will happen in that slot. Otherwise, the relevant slot will be skipped.

The next slot will not begin until the all PGCs finish their power up or power down process in current time slot. When all the 10 slots are finished, slot controller will jump to IDLE state and monitor new request from SMC.

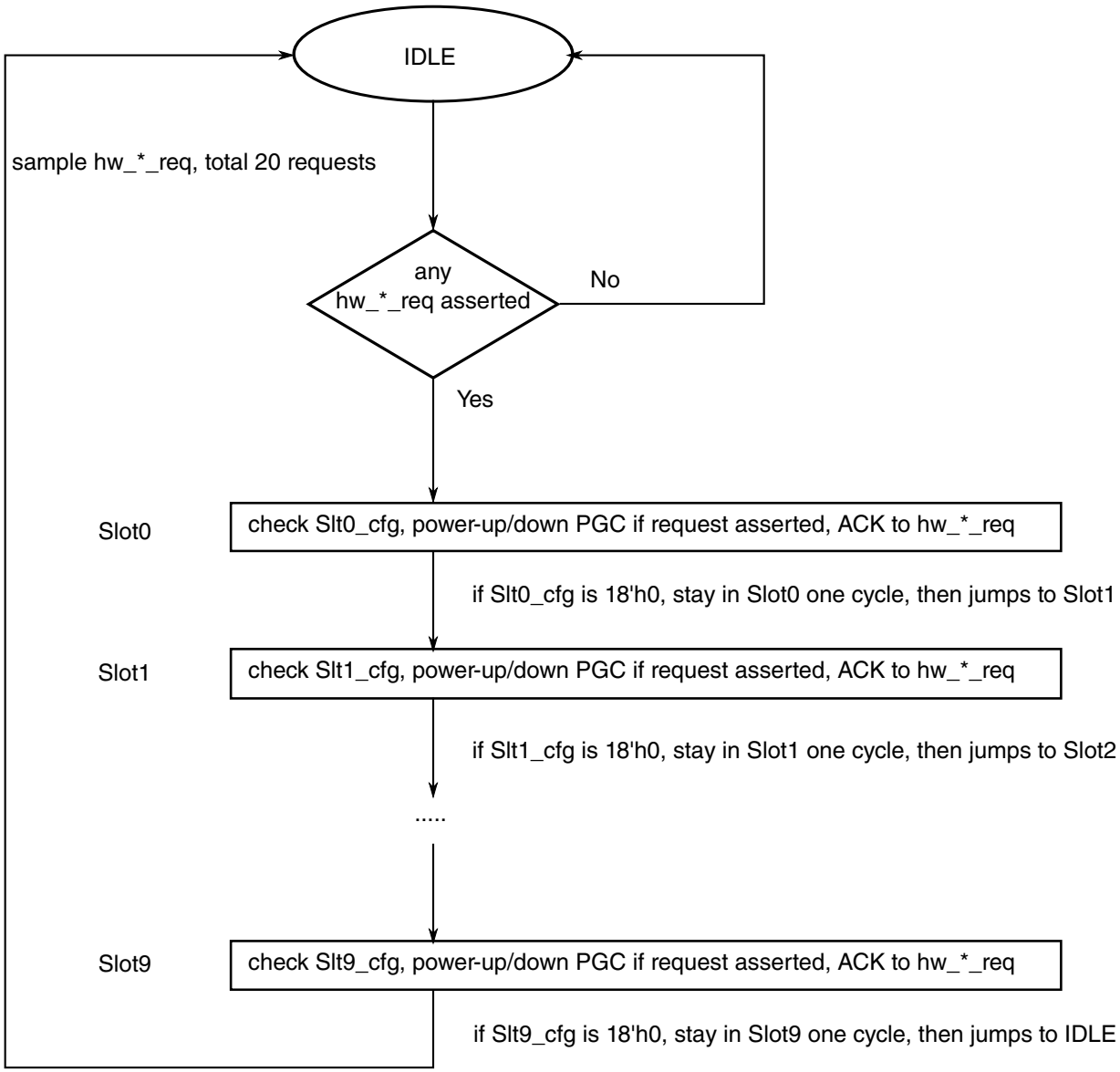


Figure 5-27. Slot controller processing flow

NOTE

PGC_SCU should be “always-on” to PGC_C0 and PGC_C1. This means PGC_SCU should be power up earlier than PGC_C0/PGC_C1 and should be power down later than PGC_C0/PGC_C1 (See example code 1 and 2). If we arrange A7 Cx/A7 SCU power down/up in same slot, special setting is required (See example code 2).

NOTE

When the system enters/exists ALL_OFF or L2_RETENTION mode, PGC_MF should be power up earlier than PGC_C0/

PGC_C1/PGC_SCU. We can arrange MIX PGC power up in earlier slot than A7 Cx/SCU power up slot (See example code 1 and 2).

5.5.5.4 Handshake between LPM controller and time slot controller

The figure below shows an example of A7 “into” LPM sequence. We want to power up fastmix/megamix, A7 SCU and A7 core0 in time slot0 /slot1/slot2 respectively. Request from low power mode controller will be mapped to relevant PGCs according to the rules listed above. We choose acknowledge from PGC_core0 as the acknowledge for A7 LPM power down request (relevant register bits are defined in “PGC_ACK_SEL_A7”). The A7 LPM will regard the three PGCs as a virtual big PGC and it will cancel all power up request when it receive the acknowledge signals from PGC_core0.

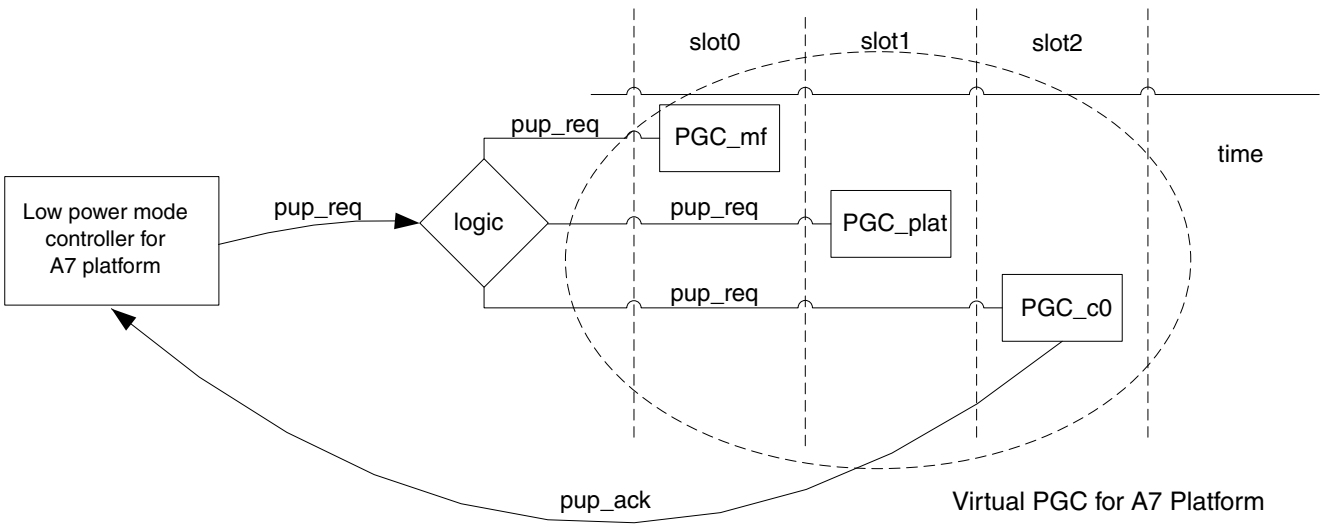


Figure 5-28. A7 into LPM sequence

NOTE

If a PGC is mapped to two CPU domain(refer to “Domain control for PGCs ”for more information), it cannot be selected as the power down acknowledge for both of the CPU platform. “PGC_ACK_SEL_A7” and “PGC_ACK_SEL_M4” are should be chosen for the last PGC in power up or power down sequence in the time slot. If there is no PGC be power up/power down with LPM sequence, the “dummy” acknowledge should be selected. Only one PGC should be selected for power down or power up acknowledge for one CPU platform

5.5.6 Power control for A7 Platform

5.5.6.1 A7 Platform power domains and power modes

There are four power domains inside SEC dual core Cortex A7 platform: Core0, core1, SCU and L2 RAM. There are six power states in A7 platform:

Power state	Core0	Core1	SCU	L2 RAM
ALL_ON	ON	ON	ON	ON
CPU0_OFF	OFF	ON	ON	ON
CPU1_OFF	ON	OFF	ON	ON
CPU01_OFF	OFF	OFF	ON	ON
L2_RETENTION	OFF	OFF	OFF	ON
ALL_OFF	OFF	OFF	OFF	OFF

In all six power states, “ALL_ON”, “CPU0_OFF”, “CPU1_OFF” and “CPU01_OFF” can exist in all RUN, WAIT or STOP mode of A7 platform. “L2_RETENTION” and “ALL_OFF” can only exist in WAIT or STOP mode of A7 platform.

5.5.6.2 Power down process for the A7 Platform

5.5.6.2.1 Power down of Core0 and Core1 in the A7 Platform

The power of core0 and core1 can be shut off along with the LPM process , as show in the figure below:

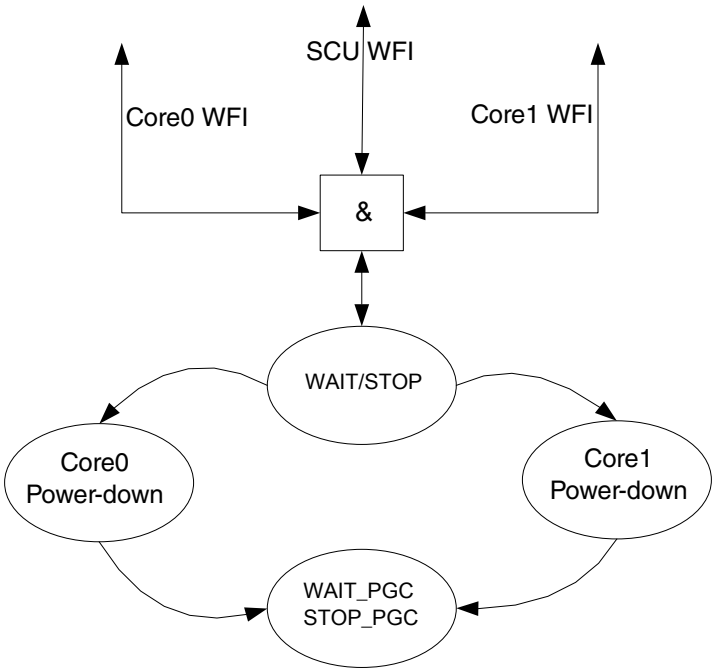


Figure 5-29. Power down of Core0 and Core1

WFIs from A7 platform will trigger the A7 platform LPM and the power of core0 or core1 will be shut off when “lpcr_a7_ad.en_c0_pdn” or “lpcr_a7_ad.en_c1_pdn” enabled in this process. This mode should be used when core0 is used as the leading core of A7 platform.

The power of core0 and core1 can also be shut off in RUN mode: in this mode “LPCR_A7_AD.en_c0_wfi_pdn” and “LPCR_A7_AD.en_c1_wfi_pdn” should be set and the condition to trigger A7 LPM will be some different:

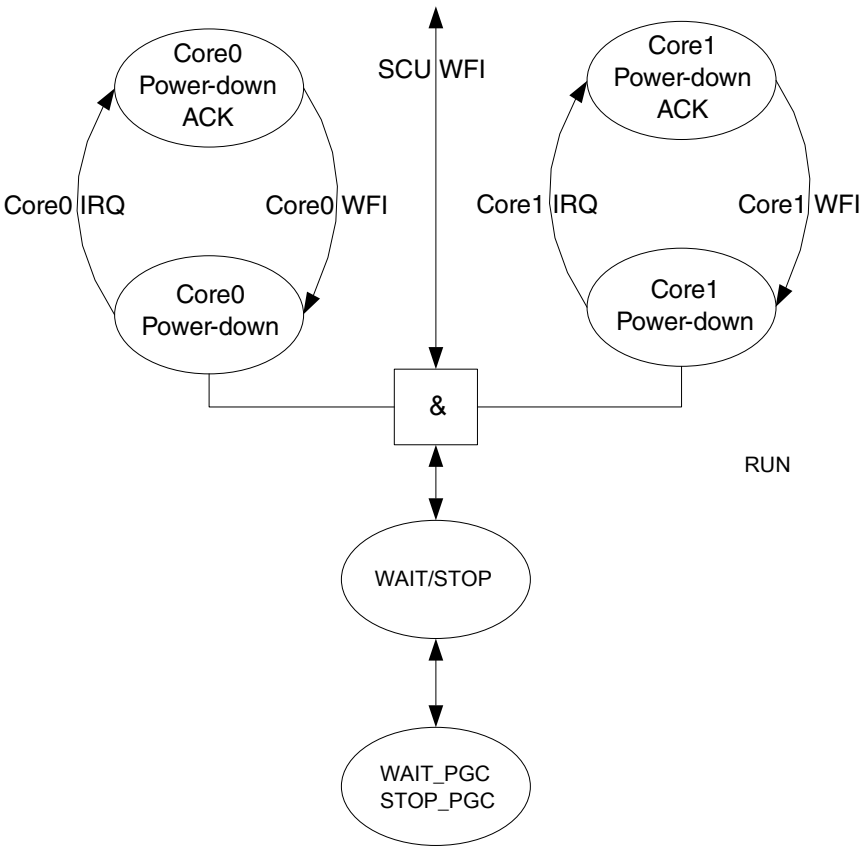


Figure 5-30. Power down of Core0 and Core1 in RUN mode

In this mode, core0/core1 power down process should not be disturbed by IRQs.

5.5.6.2.2 Power down of SCU and L2 Cache RAM

Power domain SCU and L2 cache RAM is controlled by one PGC – “PGC_PLAT” and they can only be power down in LPM process (when “LPCR_A7_AD.en_plat_pdn” is set). There is another bit “LPCR_A7_AD.l2pge” which will decide if L2 cache RAM need to be in retention mode when SCU domain is power down.

5.5.6.3 Power up process for the A7 Platform

- The core0 and core1 can be powered up with the exit of A7 LPM. The relevant bits are “LPCR_A7_AD.en_c0_pup” and “LPCR_A7_AD.en_c1_pup”.
- The core0 and core1 can also be powered up by interrupt signal in RUN mode.
- The relevant bit are “LPCR_A7_AD.en_c0_irq_pup” , “LPCR_A7_AD.en_c1_irq_pup” .

- The interrupt signal can be chosen from GIC or directly from IRQ with mask in GPCv2.
- The relevant select bits are “LPCR_A7_BSC.irq_src_c0”, “LPCR_A7_BSC.irq_src_c1” and “LPCR_A7_BSC.irq_src_a7_wup”. (LPCR_A7_BSC[30:28]).

LPCR_A7_BSC[30:28]	Usage	Restriction
3'b000	Use IRQ trigger A7 LPM and use IRQ to power up core0/core1	None
3'b011	Use GIC trigger A7 LPM and GIC to power up core0/core1	SCU cannot power down in LPM, CPU clock cannot stop in LPM
3'b111	Use IRQ trigger A7 LPM and GIC to power up core0/core1	SCU cannot power down in LPM

As show in the table above, core0/core1 can only be power up by its own interrupt in RUN mode of A7 platform.

There are three combination of “LPCR_A7_BSC[30:28]”:

1. In the first case , “IMRn_CORE0_A7, IMRn_CORE1_A7 “ are used to separate the 128 bits interrupts for core0 and core1 of A7 platform and also used as the interrupt mask for A7 LPM.
2. In the 2nd case, “IMRn_CORE0_A7, IMRn_CORE1_A7” are not used, GIC setting are used to separate interrupts for core0 and core1 of A7 platform and also used as the interrupt mask for A7 LPM.
3. In the 3rd case, “IMRn_CORE0_A7” is used as the mask for interrupt for A7 LPM, GIC setting are used to separate interrupts for core0 and core1.

5.5.7 Power control for the M4 Platform

M4 LPM is same as A7 LPM. M4 platform doesn’t have its own power domain and exist as a part of Megamix. M4 LPM power down request cannot power down the fastmix/ megamix directly. There is a virtual PGC reserved for M4, the virtual PGC will do nothing else except generating an acknowledge signal and this signal can be chosen as the acknowledge signal for M4 platform. Make sure virtual PGC power up/down slot same with fastmix/megamix PGC power up/down slot (See example code 3).

5.5.8 Domain control for PGCs

The following rules are used for PGC power up/down with domain mapping control:

1. For PGCs inside CPU platform (since M4 doesn't have its own power domain, only PGCs in A7 platform are referred): for hardware trigger (including both power up and power down) only the LPM request from its own platform will take effect; for software trigger (including both power up and power down) (the relevant register bit are "CPU_PGC_SW_PUP_REQ" and "CPU_PGC_SW_PDN_REQ"), only the software running in its own platform will take effect.
2. For PGCs outside CPU platform(MIX and PU PGCs), register bits "PGC_CPU_MAPPING" will map MIX and PU PGCs to A7 or M4 CPU domain:
   - One PGC can be mapped to one or both of A7 and M4 domain;
   - For hardware power up request, if a PGC is mapped to any CPU domain, the PGC will be powered up when the corresponding CPU platform sends out its power up request.
   - For software power up, if a PGC is mapped any CPU platform, the PGC can be powered up by software running in corresponding CPU.(the relevant register bits are "MIX_PGC_SW_PUP_REQ" and "PU_PGC_SW_PUP_REQ")
   - For hardware power down, if a power domain is mapped to only one CPU domain, the relevant PGC can be powered down when the corresponding CPU platform sends out its power down request; If a power domain is mapped to both two CPU domain, when one CPU platform sends out its hardware power down request, the relevant PGC will power down with any of the following condition satisfied: the other CPU already in LPM; the other CPU want to power down relevant PGC (the relevant register are "A7_MIX_PDN_FLG, M4_MIX_PDN_FLG, A7_PU_PDN_FLG, M4_PU_PDN_FLG")
   - For software power down, if a power domain is mapped to only one CPU domain, the PGC can be powered down by software running in corresponding CPU. (the relevant register bits are "MIX_PGC_SW_PUP_REQ" and "PU_PGC_SW_PUP_REQ"). If a power domain is mapped to both two CPU domain, when one CPU platform sends out its software power down request, the relevant PGC will power down with any of the following condition satisfied: the other CPU already in LPM; the other CPU want to power down relevant PGC (the relevant register are "A7_MIX_PDN_FLG, M4_MIX_PDN_FLG, A7_PU_PDN_FLG, M4_PU_PDN_FLG")
   - The access of "PGC_CPU_MAPPING" is controlled by domain information from RDC

5.5.9 Example Code

Below are code examples for entering specific power scenarios

5.5.9.1 Example Code 1

```
    //ARM enters into ALL_OFF(STOP) mode and enable DSM :
//after "wfi", MIX/C0/C1 power down in SLOT0, SCU power down in SLOT1 when
//after "GPT1_INT" arrived, MIX power up in SLOT2, SCU power up in SLOT3, C0 power up in
SLOT4

 //IMRx_CORE0_A7
 reg32_write(GPC_IPS_BASE_ADDR + 0x30, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x34, 0xFF7FFFFF);  //[23] : GPT1 used as wakeup source
 reg32_write(GPC_IPS_BASE_ADDR + 0x38, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x3C, 0xFFFFFFFF);

 //IMRx_CORE1_A7
 reg32_write(GPC_IPS_BASE_ADDR + 0x40, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x44, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x48, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x4C, 0xFFFFFFFF);

 //LPCR_A7_BSC
 reg32_write(GPC_IPS_BASE_ADDR,       0x0000000A) ;
        //[30:28] : A7_CO/A7_C1/LPM wakeup from external INT
        //[14]    : CLOCK OFF during STOP mode
        //[3:0]   : STOP mode

 //LPCR_A7_AD
 reg32_write(GPC_IPS_BASE_ADDR + 0x4, 0x00010A1A) ;
        //[16]    : 1(ALL_OFF mode); 0(L2 retention mode)
        //[11]/[9]: A7_C0/A7_C1 power up with A7 LPM PUP REQ
        //[4]     : A7_SCU power down with A7 LPM PDN REQ
        //[3]/[1] : A7_C0/A7_C1 powr down with A7 LPM PDN REQ

 //LPCR_M4
 reg32_write(GPC_IPS_BASE_ADDR + 0x8, 0x80000000) ;
        //[31]    : DSM ingore to check M4 low power state
        //[1:0]   : M4 LPM run mode

 //SLPCR
 reg32_write(GPC_IPS_BASE_ADDR + 0x14,0xe000ffA7) ;
        //[31]    : enable DSM
        //[30]    : eneable regulator bypass
        //[5:3]   : wait 64 ckil clock cycles
        //[2]     : enable PMIC standby
        //[1]     : enable OSC power down
        //[0]     : bypass PMIC ready handshake

 //PGC_ACK_SEL_A7
 reg32_write(GPC_IPS_BASE_ADDR + 0x24,0x00010004) ;
        //[2]     : A7_SCU PGC as LPM power down ack
        //[16]    : A7_C0  PGC as LPM power up ack

//SLT_CFG0
 reg32_write(GPC_IPS_BASE_ADDR + 0xB0,0x00000045) ;
        //[2]/[0] : A7_C0/A7_C1 power down in SLOT0
        //[6]     : fastmix/megamix power down in SLOT0

 //SLT_CFG1
 reg32_write(GPC_IPS_BASE_ADDR + 0xB4,0x00000010) ;
        //[4]     : A7_SCU power down in SLOT1

 //SLT_CFG2
 reg32_write(GPC_IPS_BASE_ADDR + 0xB8,0x00000080) ;
        //[7]     : fastmix/megamix power up in SLOT2

 //SLT_CFG3
 reg32_write(GPC_IPS_BASE_ADDR + 0xBC,0x00000020) ;
        //[5]     : A7_SCU power up in SLOT3
```

```
//SLT_CFG4
reg32_write(GPC_IPS_BASE_ADDR + 0xC0,0x00000002) ;
       //[1]     : A7_C0  power up in SLOT4, A7_C1 not power up

//SLT_CFGx
reg32_write(GPC_IPS_BASE_ADDR + 0xC4,0x00000000) ;   // no action in SLOT5
reg32_write(GPC_IPS_BASE_ADDR + 0xC8,0x00000000) ;   // no action in SLOT6
reg32_write(GPC_IPS_BASE_ADDR + 0xCC,0x00000000) ;   // no action in SLOT7
reg32_write(GPC_IPS_BASE_ADDR + 0xD0,0x00000000) ;   // no action in SLOT8
reg32_write(GPC_IPS_BASE_ADDR + 0xD4,0x00000000) ;   // no action in SLOT9

//PGC_CPU_MAPPING
reg32_write(GPC_IPS_BASE_ADDR + 0xEC,0x00000001) ;
       //[0]     :  MIX PGC mapping to A7 LPM

//A7 PGC
reg32_write(GPC_IPS_BASE_ADDR +0x800, reg32_read(GPC_IPS_BASE_ADDR +0x800) | 0x00000001) ;
       // enable A7_C0  PGC power down
reg32_write(GPC_IPS_BASE_ADDR +0x840, reg32_read(GPC_IPS_BASE_ADDR +0x840) | 0x00000001) ;
       // enable A7_C1  PGC power down
reg32_write(GPC_IPS_BASE_ADDR +0x880, reg32_read(GPC_IPS_BASE_ADDR +0x880) | 0x00000001) ;
       // enable A7_SCU PGC power down

reg32_write(GPC_IPS_BASE_ADDR +0x890, (0x59 &lt;&lt; 10) | 0x5B | (0x51 &lt;&lt; 20) ) ;
       // change nL2retn/mempwr/dftram to meet SCU power up timing

//fastmix/megamix PGC
reg32_write(GPC_IPS_BASE_ADDR +0xA00, reg32_read(GPC_IPS_BASE_ADDR +0xA00) | 0x00000001) ;
       // enable MIX PGC power down
```

5.5.9.2 Example Code 2

```
  //ARM enters into L2_RETENTION(STOP) mode and enable DSM :
//after "wfi", MIX/C0/C1/SCU power down in SLOT0
//after "GPT1_INT" arrived, MIX power up in SLOT1, SCU/C0 power up in SLOT2

 //IMRx_CORE0_A7
 reg32_write(GPC_IPS_BASE_ADDR + 0x30, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x34, 0xFF7FFFFF);
       //[23] : GPT1 used as wakeup source
 reg32_write(GPC_IPS_BASE_ADDR + 0x38, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x3C, 0xFFFFFFFF);

 //IMRx_CORE1_A7
 reg32_write(GPC_IPS_BASE_ADDR + 0x40, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x44, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x48, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x4C, 0xFFFFFFFF);

 //LPCR_A7_BSC
 reg32_write(GPC_IPS_BASE_ADDR,       0x0000000A) ;
       //[30:28] : A7_CO/A7_C1/LPM wakeup from external INT
       //[14]    : CLOCK OFF during STOP mode
       //[3:0]   : STOP mode

 //LPCR_A7_AD
 reg32_write(GPC_IPS_BASE_ADDR + 0x4, 0x00000A1A) ;
       //[16]    : 1(ALL_OFF mode); 0(L2 retention mode)
       //[11]/[9]: A7_C0/A7_C1 power up with A7 LPM PUP REQ
       //[4]     : A7_SCU power down with A7 LPM PDN REQ
       //[3]/[1] : A7_C0/A7_C1 powr down with A7 LPM PDN REQ

 //LPCR_M4
```

```
 reg32_write(GPC_IPS_BASE_ADDR + 0x8, 0x80000000) ;
        //[31]    : DSM ingore to check M4 low power state
        //[1:0]   : M4 LPM run mode

 //SLPCR
 reg32_write(GPC_IPS_BASE_ADDR + 0x14,0xe000ffA7) ;
        //[31]    : enable DSM
        //[30]    : eneable regulator bypass
        //[5:3]   : wait 64 ckil clock cycles
        //[2]     : enable PMIC standby
        //[1]     : enable OSC power down
        //[0]     : bypass PMIC ready handshake

 //PGC_ACK_SEL_A7
 reg32_write(GPC_IPS_BASE_ADDR + 0x24,0x00010004) ;
        //[2]     : A7_SCU PGC as LPM power down ack
        //[16]    : A7_C0  PGC as LPM power up ack

 //SLT_CFG0
 reg32_write(GPC_IPS_BASE_ADDR + 0xB0,0x00000055) ;
        //[2]/[0] : A7_C0/A7_C1 power down in SLOT0
        //[4]     : A7_SCU power down in SLOT0
        //[6]     : fastmix/megamix power down in SLOT0
        //A7_Cx/SCU are power down in same slot. Special setting is required( see below PGC
setting #A )

 //SLT_CFG1
 reg32_write(GPC_IPS_BASE_ADDR + 0xB4,0x00000080) ;
        //[7]     : fastmix/megamix power up in SLOT1

 //SLT_CFG2
 reg32_write(GPC_IPS_BASE_ADDR + 0xB8,0x00000022) ;
        //[5]     : A7_SCU power up in SLOT1
        //[1]     : A7_C0 power up in SLOT1, A7_C1 not power up
        //A7_Cx/SCU are power up in same slot. Special setting is required( see below PGC
setting #B )


 //SLT_CFGx
 reg32_write(GPC_IPS_BASE_ADDR + 0xBC,0x00000000) ;   // no action in SLOT3
 reg32_write(GPC_IPS_BASE_ADDR + 0xC0,0x00000000) ;   // no action in SLOT4
 reg32_write(GPC_IPS_BASE_ADDR + 0xC4,0x00000000) ;   // no action in SLOT5
 reg32_write(GPC_IPS_BASE_ADDR + 0xC8,0x00000000) ;   // no action in SLOT6
 reg32_write(GPC_IPS_BASE_ADDR + 0xCC,0x00000000) ;   // no action in SLOT7
 reg32_write(GPC_IPS_BASE_ADDR + 0xD0,0x00000000) ;   // no action in SLOT8
 reg32_write(GPC_IPS_BASE_ADDR + 0xD4,0x00000000) ;   // no action in SLOT9

 //PGC_CPU_MAPPING
 reg32_write(GPC_IPS_BASE_ADDR + 0xEC,0x00000001) ;
       //[0]     :  MIX PGC mapping to A7 LPM

 //Special PGC setting #A for C0/C1/SCU power down in same slot(SCU should be always ON
comparing to C0/C1)
 reg32_write(GPC_IPS_BASE_ADDR +0x808, (reg32_read(GPC_IPS_BASE_ADDR +0x808) & 0xFFFFC0C0) |
0x0801 ) ;
       // set C0.ISO2SW  = 8  ; C0.ISO  = 1;
 reg32_write(GPC_IPS_BASE_ADDR +0x848, (reg32_read(GPC_IPS_BASE_ADDR +0x848) & 0xFFFFC0C0) |
0x0801 ) ;
       // set C1.ISO2SW  = 8  ; C1.ISO  = 1;
 reg32_write(GPC_IPS_BASE_ADDR +0x888, (reg32_read(GPC_IPS_BASE_ADDR +0x888) & 0xFFFFC0C0) |
0x1001 ) ;
       // set SCU.ISO2SW = 16 ; SCU.ISO = 1;

 //Special PGC setting #B for A7_Cx/SCU power up in same slot(SCU should be always ON
comparing to C0/C1)
 reg32_write(GPC_IPS_BASE_ADDR +0x804, (reg32_read(GPC_IPS_BASE_ADDR +0x804) & 0xFF800040) |
0x11 | (0x20 &amp;lt;&amp;lt; 7) ) ;
       // set C0.SW      = 0x11  ; C0.SW2ISO  = 0x20 ;
 reg32_write(GPC_IPS_BASE_ADDR +0x844, (reg32_read(GPC_IPS_BASE_ADDR +0x844) & 0xFF800040) |
0x11 | (0x20 &amp;lt;&amp;lt; 7) ) ;
```

```
       // set C1.SW      = 0x11  ; C1.SW2ISO  = 0x20 ;
 reg32_write(GPC_IPS_BASE_ADDR +0x884, (reg32_read(GPC_IPS_BASE_ADDR +0x884) & 0xFF800040) |
0x1  | (0x0f &amp;lt;&amp;lt; 7) ) ;
       // set SCU.SW     = 0x1   ; SCU.SW2ISO = 0x0f ;

 //A7 PGC
 reg32_write(GPC_IPS_BASE_ADDR +0x800, reg32_read(GPC_IPS_BASE_ADDR +0x800) | 0x00000001) ;
       // enable A7_C0  PGC power down
 reg32_write(GPC_IPS_BASE_ADDR +0x840, reg32_read(GPC_IPS_BASE_ADDR +0x840) | 0x00000001) ;
       // enable A7_C1  PGC power down
 reg32_write(GPC_IPS_BASE_ADDR +0x880, reg32_read(GPC_IPS_BASE_ADDR +0x880) | 0x00000001) ;
       // enable A7_SCU PGC power down

 reg32_write(GPC_IPS_BASE_ADDR +0x890, (0x59 &amp;lt;&amp;lt; 10) | 0x5B | (0x51 &amp;lt;&amp;lt; 20) ) ;
       // change nL2retn/mempwr/dftram to meet SCU power up timing


 //fastmix/megamix PGC
 reg32_write(GPC_IPS_BASE_ADDR +0xA00, reg32_read(GPC_IPS_BASE_ADDR +0xA00) | 0x00000001) ;
       // enable MIX PGC power down
```

5.5.9.3 Example Code 3

```
  //A7/M4 both enters into low power mode. A7/M4 are in different master domain.
//MIX are mapping to both A7 and M4
//after A7/M4 enters into low power mode, MIX will be also power down.A7/M4 enters into low
power mode any time
//either A7 or M4 exists from low power mode, MIX will be also power up

 //IMRx_CORE0_A7
 reg32_write(GPC_IPS_BASE_ADDR + 0x30, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x34, 0xFF7FFFFF);
       //[23] : GPT1 used as ARM wakeup source
 reg32_write(GPC_IPS_BASE_ADDR + 0x38, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x3C, 0xFFFFFFFF);

 //IMRx_CORE1_A7
 reg32_write(GPC_IPS_BASE_ADDR + 0x40, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x44, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x48, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x4C, 0xFFFFFFFF);

 //IMRx_M4
 reg32_write(GPC_IPS_BASE_ADDR + 0x50, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x54, 0xFFBFFFFF);
       //[22] : GPT2 used as M4 wakeup source
 reg32_write(GPC_IPS_BASE_ADDR + 0x58, 0xFFFFFFFF);
 reg32_write(GPC_IPS_BASE_ADDR + 0x5C, 0xFFFFFFFF);

 //LPCR_A7_BSC
 reg32_write(GPC_IPS_BASE_ADDR,       0x0000000A) ;
       //[30:28] : A7_CO/A7_C1/LPM wakeup from external INT
       //[14]    : CLOCK OFF during STOP mode
       //[3:0]   : STOP mode

 //LPCR_A7_AD
 reg32_write(GPC_IPS_BASE_ADDR + 0x4, 0x00010A1A) ;
       //[16]    : 1(ALL_OFF mode); 0(L2 retention mode)
       //[11]/[9]: A7_C0/A7_C1 power up with A7 LPM PUP REQ
       //[4]     : A7_SCU power down with A7 LPM PDN REQ
       //[3]/[1] : A7_C0/A7_C1 powr down with A7 LPM PDN REQ

 //LPCR_M4
```

```
 reg32_write(GPC_IPS_BASE_ADDR + 0x8, 0x00003FFE) ;
       //[31]    : 0(check M4 low power state to enter into DSM)
       //[14]    : 0(M4 clock OFF during low power mode)
       //[3:2]   : enable M4 virtual PGC power up/down with M4 LPM
       //[1:0]   : M4 LPM STOP mode

 //SLPCR
 reg32_write(GPC_IPS_BASE_ADDR + 0x14,0xe000ffA7) ;
       //[31]    : enable DSM
       //[30]    : eneable regulator bypass
       //[5:3]   : wait 64 ckil clock cycles
       //[2]     : enable PMIC standby
       //[1]     : enable OSC power down
       //[0]     : bypass PMIC ready handshake

 //PGC_ACK_SEL_A7
 reg32_write(GPC_IPS_BASE_ADDR + 0x24,0x00010004) ;
       //[2]     : A7_SCU PGC as LPM power down ack
       //[16]    : A7_C0  PGC as LPM power up ack

 //PGC_ACK_SEL_M4
 reg32_write(GPC_IPS_BASE_ADDR + 0x28,0x00010001) ;
       //[0]     : M4 virtual PGC as M4 LPM power down ack
       //[16]    : M4 virtual PGC as M4 LPM power up ack

 //GPC_MISC
 reg32_write(GPC_IPS_BASE_ADDR + 0x2C,reg32_read(GPC_IPS_BASE_ADDR + 0x2C) | 0x100) ;
       //[8] : not mask M4 power down request to M4 virtual PGC


 //SLT_CFG0
 reg32_write(GPC_IPS_BASE_ADDR + 0xB0,0x00040055) ;
       //[2]/[0] : A7_C0/A7_C1 power down in SLOT0
       //[4]     : A7_SCU power down in SLOT0
       //[6]     : fastmix/megamix power down in SLOT0
       //A7_Cx/SCU are power down in same slot. Special setting is required( see below PGC
setting #A )
       //[18]    : M4 virtual PGC power down in SLOT0, same with fastmix/megamix PGC power
down slot

 //SLT_CFG1
 reg32_write(GPC_IPS_BASE_ADDR + 0xB4,0x00080080) ;
       //[7]     : fastmix/megamix power up in SLOT1
       //[19]    : M4 virtual PGC power up in SLOT1, same with fastmix/megamix PGC power up
slot

 //SLT_CFG2
 reg32_write(GPC_IPS_BASE_ADDR + 0xB8,0x00000022) ;
       //[5]     : A7_SCU power up in SLOT1
       //[1]     : A7_C0 power up in SLOT1, A7_C1 not power up
       //A7_Cx/SCU are power up in same slot. Special setting is required( see below PGC
setting #B )

 //SLT_CFGx
 reg32_write(GPC_IPS_BASE_ADDR + 0xBC,0x00000000) ;   // no action in SLOT3
 reg32_write(GPC_IPS_BASE_ADDR + 0xC0,0x00000000) ;   // no action in SLOT4
 reg32_write(GPC_IPS_BASE_ADDR + 0xC4,0x00000000) ;   // no action in SLOT5
 reg32_write(GPC_IPS_BASE_ADDR + 0xC8,0x00000000) ;   // no action in SLOT6
 reg32_write(GPC_IPS_BASE_ADDR + 0xCC,0x00000000) ;   // no action in SLOT7
 reg32_write(GPC_IPS_BASE_ADDR + 0xD0,0x00000000) ;   // no action in SLOT8
 reg32_write(GPC_IPS_BASE_ADDR + 0xD4,0x00000000) ;   // no action in SLOT9

 //PGC_CPU_MAPPING
 reg32_write(GPC_IPS_BASE_ADDR + 0xEC,0x00000101) ;
       //[0]     :  MIX PGC mapping to A7 LPM
       //[8]     :  MIX PGC mapping to M4 LPM

 //Special PGC setting #A for C0/C1/SCU power down in same slot(SCU should be always ON
comparing to C0/C1)
 reg32_write(GPC_IPS_BASE_ADDR +0x808, (reg32_read(GPC_IPS_BASE_ADDR +0x808) & 0xFFFFC0C0) |
```

```
0x0801 ) ;
       // set C0.ISO2SW  = 8  ; C0.ISO  = 1;
 reg32_write(GPC_IPS_BASE_ADDR +0x848, (reg32_read(GPC_IPS_BASE_ADDR +0x848) & 0xFFFFC0C0) |
0x0801 ) ;
       // set C1.ISO2SW  = 8  ; C1.ISO  = 1;
 reg32_write(GPC_IPS_BASE_ADDR +0x888, (reg32_read(GPC_IPS_BASE_ADDR +0x888) & 0xFFFFC0C0) |
0x1001 ) ;
       // set SCU.ISO2SW = 16 ; SCU.ISO = 1;

 //Special PGC setting #B for A7_Cx/SCU power up in same slot(SCU should be always ON
comparing to C0/C1)
 reg32_write(GPC_IPS_BASE_ADDR +0x804, (reg32_read(GPC_IPS_BASE_ADDR +0x804) & 0xFF800040) |
0x10 | (0x1f &lt;&lt; 7) ) ;
       // set C0.SW      = 0x10  ; C0.SW2ISO  = 0x1f ;
 reg32_write(GPC_IPS_BASE_ADDR +0x844, (reg32_read(GPC_IPS_BASE_ADDR +0x844) & 0xFF800040) |
0x10 | (0x1f &lt;&lt; 7) ) ;
       // set C1.SW      = 0x10  ; C1.SW2ISO  = 0x1f ;
 reg32_write(GPC_IPS_BASE_ADDR +0x884, (reg32_read(GPC_IPS_BASE_ADDR +0x884) & 0xFF800040) |
0x1  | (0x0f &lt;&lt; 7) ) ;
       // set SCU.SW     = 0x1   ; SCU.SW2ISO = 0x0f ;

 //A7 PGC
 reg32_write(GPC_IPS_BASE_ADDR +0x800, reg32_read(GPC_IPS_BASE_ADDR +0x800) | 0x00000001) ;
       // enable A7_C0  PGC power down
 reg32_write(GPC_IPS_BASE_ADDR +0x840, reg32_read(GPC_IPS_BASE_ADDR +0x840) | 0x00000001) ;
       // enable A7_C1  PGC power down
 reg32_write(GPC_IPS_BASE_ADDR +0x880, reg32_read(GPC_IPS_BASE_ADDR +0x880) | 0x00000001) ;
       // enable A7_SCU PGC power down

 reg32_write(GPC_IPS_BASE_ADDR +0x890, (0x59 &lt;&lt; 10) | 0x5B | (0x51 &lt;&lt; 20) ) ;
       // change nL2retn/mempwr/dftram to meet SCU power up timing

 //fastmix/megamix PGC
 reg32_write(GPC_IPS_BASE_ADDR +0xA00, reg32_read(GPC_IPS_BASE_ADDR +0xA00) | 0x00000001) ;
       // enable MIX PGC power down
```

5.5.9.4 Example Code 4

```
  // software power up/down PCIE PHY
 //power up PCIE PHY
 reg32_write ( GPC_IPS_BASE_ADDR + 0xEC,  reg32_read(GPC_IPS_BASE_ADDR + 0x0EC) | 0x8 );
       //map PCIE PGC to A7
 reg32_write ( 0x30360210,  reg32_read(0x30360210) | 0x80000000 );
       //allow GPC to control LDO_1P0D

 reg32_write ( GPC_IPS_BASE_ADDR + 0xF8 ,  reg32_read(GPC_IPS_BASE_ADDR + 0xF8) | 0x2 );
       //trigger sw Power up Request
 while( read(GPC_IPS_BASE_ADDR + 0xF8) & 0x2 );
       //wait software power up request self clear

 //power down PCIE PHY
 reg32_write ( GPC_IPS_BASE_ADDR + 0xEC,  reg32_read(GPC_IPS_BASE_ADDR + 0x0EC) | 0x8 );
       //map PCIE PGC to A7
 reg32_write ( 0x30360210,  reg32_read(0x30360210) | 0x80000000 );
       // allow GPC to control LDO_1P0D
 reg32_write ( GPC_IPS_BASE_ADDR + 0xC40, reg32_read(GPC_IPS_BASE_ADDR + 0xC40) | 0x1 );
       // enable PCIE PGC power down

 reg32_write ( GPC_IPS_BASE_ADDR + 0x104 ,  reg32_read(GPC_IPS_BASE_ADDR + 0x104) | 0x2 );
       //trigger sw Power Down Request
 while( read(GPC_IPS_BASE_ADDR + 0x104) & 0x2 );
```

```
//wait software power down request self clear
```

5.5.10 GPC Memory Map/Register Definition

Detailed descriptions of each register can be found below.

The total GPC memory map is 4KB

Table 5-21. Memory Regions

Address Range(offset)	Region
0x000 - 0x3FF	GPC configuration register
0x400 - 0x7FF	Reserved
0x800 - 0x9FF	CPU and SCU type PGC register base address
0xA00 - 0xBFF	MIX type PGC register base address
0xC00 - 0xDFF	PU type PGC register base address
0xE00 - 0xFFF	Reserved

Each PGC (CPU type, MIX type, PU type) will occupy 64 Bytes address space, the specific base address of each PGC are listed as below.

- 0x800 ~ 0x83F : PGC for A7 core0
- 0x840 ~ 0x87F: PGC for A7 core1
- 0x880 ~ 0x8BF: PGC for A7 SCU
- 0xA00 ~ 0xA3F: PGC for fastmix/megamix
- 0xC00 ~ 0xC3F: PGC for MIPI PHY
- 0xC40 ~ 0xC7F: PGC for PCIE_PHY
- 0xC80 ~ 0xCBF: Reserved
- 0xCC0 ~ 0xCFF: Reserved
- 0xD00 ~ 0xD3F: PGC for USB HSIC PHY

For more specific information about PGC register definition, please see the register definition for each PGC.

GPC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303A_0000	Basic Low power control register of A7 platform (GPC_LPCR_A7_BSC)	32	R/W	0000_3FF0h	5.5.10.1/ 831

Table continues on the next page...

GPC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303A_0004	Advanced Low power control register of A7 platform (GPC_LPCR_A7_AD)	32	R/W	0000_0000h	5.5.10.2/ 833
303A_0008	Low power control register of CPU1 (GPC_LPCR_M4)	32	R/W	0000_3FF0h	5.5.10.3/ 835
303A_0014	System low power control register (GPC_SLPCR)	32	R/W	0000_FF82h	5.5.10.4/ 837
303A_0020	Memory low power control register (GPC_MLPCR)	32	R/W	0048_0100h	5.5.10.5/ 839
303A_0024	PGC acknowledge signal selection of A7 platform (GPC_PGC_ACK_SEL_A7)	32	R/W	0800_0800h	5.5.10.6/ 840
303A_0028	PGC acknowledge signal selection of M4 platform (GPC_PGC_ACK_SEL_M4)	32	R/W	0800_0800h	5.5.10.7/ 843
303A_002C	GPC Miscellaneous register (GPC_MISC)	32	R/W	0000_0002h	5.5.10.8/ 845
303A_0030	IRQ masking register 1 of A7 core0 (GPC_IMR1_CORE0_A7)	32	R/W	0000_0000h	5.5.10.9/ 846
303A_0034	IRQ masking register 2 of A7 core0 (GPC_IMR2_CORE0_A7)	32	R/W	0000_0000h	5.5.10.10/ 846
303A_0038	IRQ masking register 3 of A7 core0 (GPC_IMR3_CORE0_A7)	32	R/W	0000_0000h	5.5.10.11/ 847
303A_003C	IRQ masking register 4 of A7 core0 (GPC_IMR4_CORE0_A7)	32	R/W	0000_0000h	5.5.10.12/ 847
303A_0040	IRQ masking register 1 of A7 core1 (GPC_IMR1_CORE1_A7)	32	R/W	0000_0000h	5.5.10.13/ 848
303A_0044	IRQ masking register 2 of A7 core1 (GPC_IMR2_CORE1_A7)	32	R/W	0000_0000h	5.5.10.14/ 848
303A_0048	IRQ masking register 3 of A7 core1 (GPC_IMR3_CORE1_A7)	32	R/W	0000_0000h	5.5.10.15/ 848
303A_004C	IRQ masking register 4 of A7 core1 (GPC_IMR4_CORE1_A7)	32	R/W	0000_0000h	5.5.10.16/ 849
303A_0050	IRQ masking register 1 of M4 (GPC_IMR1_M4)	32	R/W	0000_0000h	5.5.10.17/ 849
303A_0054	IRQ masking register 2 of M4 (GPC_IMR2_M4)	32	R/W	0000_0000h	5.5.10.18/ 850
303A_0058	IRQ masking register 3 of M4 (GPC_IMR3_M4)	32	R/W	0000_0000h	5.5.10.19/ 850
303A_005C	IRQ masking register 4 of M4 (GPC_IMR4_M4)	32	R/W	0000_0000h	5.5.10.20/ 850
303A_0070	IRQ status register 1 of A7 (GPC_ISR1_A7)	32	R	0000_0000h	5.5.10.21/ 851
303A_0074	IRQ status register 2 of A7 (GPC_ISR2_A7)	32	R	0000_0000h	5.5.10.22/ 851
303A_0078	IRQ status register 3 of A7 (GPC_ISR3_A7)	32	R	0000_0000h	5.5.10.23/ 852

Table continues on the next page...

GPC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303A_007C	IRQ status register 4 of A7 (GPC_ISR4_A7)	32	R	0000_0000h	5.5.10.24/ 852
303A_0080	IRQ status register 1 of M4 (GPC_ISR1_M4)	32	R	0000_0000h	5.5.10.25/ 852
303A_0084	IRQ status register 2 of M4 (GPC_ISR2_M4)	32	R	0000_0000h	5.5.10.26/ 853
303A_0088	IRQ status register 3 of M4 (GPC_ISR3_M4)	32	R	0000_0000h	5.5.10.27/ 853
303A_008C	IRQ status register 4 of M4 (GPC_ISR4_M4)	32	R	0000_0000h	5.5.10.28/ 853
303A_00B0	Slot configure register (GPC_SLT0_CFG)	32	R/W	0000_0000h	5.5.10.29/ 854
303A_00B4	Slot configure register (GPC_SLT1_CFG)	32	R/W	0000_0000h	5.5.10.29/ 854
303A_00B8	Slot configure register (GPC_SLT2_CFG)	32	R/W	0000_0000h	5.5.10.29/ 854
303A_00BC	Slot configure register (GPC_SLT3_CFG)	32	R/W	0000_0000h	5.5.10.29/ 854
303A_00C0	Slot configure register (GPC_SLT4_CFG)	32	R/W	0000_0000h	5.5.10.29/ 854
303A_00C4	Slot configure register (GPC_SLT5_CFG)	32	R/W	0000_0000h	5.5.10.29/ 854
303A_00C8	Slot configure register (GPC_SLT6_CFG)	32	R/W	0000_0000h	5.5.10.29/ 854
303A_00CC	Slot configure register (GPC_SLT7_CFG)	32	R/W	0000_0000h	5.5.10.29/ 854
303A_00D0	Slot configure register (GPC_SLT8_CFG)	32	R/W	0000_0000h	5.5.10.29/ 854
303A_00D4	Slot configure register (GPC_SLT9_CFG)	32	R/W	0000_0000h	5.5.10.29/ 854
303A_00EC	PGC CPU mapping (GPC_PGC_CPU_MAPPING)	32	R/W	0000_0000h	5.5.10.30/ 857
303A_00F0	CPU PGC software up trigger (GPC_CPU_PGC_SW_PUP_REQ)	32	R/W	0000_0000h	5.5.10.31/ 858
303A_00F8	PU PGC software up trigger (GPC_PU_PGC_SW_PUP_REQ)	32	R/W	0000_0000h	5.5.10.32/ 859
303A_00FC	CPU PGC software down trigger (GPC_CPU_PGC_SW_PDN_REQ)	32	R/W	0000_0000h	5.5.10.33/ 860
303A_0104	PU PGC software down trigger (GPC_PU_PGC_SW_PDN_REQ)	32	R/W	0000_0000h	5.5.10.34/ 861
303A_0130	CPU PGC software up trigger status1 (GPC_CPU_PGC_PUP_STATUS1)	32	R	0000_0000h	5.5.10.35/ 862
303A_0134	A7 MIX software up trigger status register (GPC_A7_MIX_PGC_PUP_STATUS0)	32	R	0000_0000h	5.5.10.36/ 864

Table continues on the next page...

GPC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303A_0138	A7 MIX software up trigger status register (GPC_A7_MIX_PGC_PUP_STATUS1)	32	R	0000_0000h	5.5.10.36/ 864
303A_013C	A7 MIX software up trigger status register (GPC_A7_MIX_PGC_PUP_STATUS2)	32	R	0000_0000h	5.5.10.36/ 864
303A_0140	M4 MIX PGC software up trigger status register (GPC_M4_MIX_PGC_PUP_STATUS0)	32	R	0000_0000h	5.5.10.37/ 867
303A_0144	M4 MIX PGC software up trigger status register (GPC_M4_MIX_PGC_PUP_STATUS1)	32	R	0000_0000h	5.5.10.37/ 867
303A_0148	M4 MIX PGC software up trigger status register (GPC_M4_MIX_PGC_PUP_STATUS2)	32	R	0000_0000h	5.5.10.37/ 867
303A_014C	A7 PU software up trigger status register (GPC_A7_PU_PGC_PUP_STATUS0)	32	R	0000_0000h	5.5.10.38/ 870
303A_0150	A7 PU software up trigger status register (GPC_A7_PU_PGC_PUP_STATUS1)	32	R	0000_0000h	5.5.10.38/ 870
303A_0154	A7 PU software up trigger status register (GPC_A7_PU_PGC_PUP_STATUS2)	32	R	0000_0000h	5.5.10.38/ 870
303A_0158	M4 PU PGC software up trigger status register (GPC_M4_PU_PGC_PUP_STATUS0)	32	R	0000_0000h	5.5.10.39/ 873
303A_015C	M4 PU PGC software up trigger status register (GPC_M4_PU_PGC_PUP_STATUS1)	32	R	0000_0000h	5.5.10.39/ 873
303A_0160	M4 PU PGC software up trigger status register (GPC_M4_PU_PGC_PUP_STATUS2)	32	R	0000_0000h	5.5.10.39/ 873
303A_0170	CPU PGC software dn trigger status1 (GPC_CPU_PGC_PDN_STATUS1)	32	R	0000_0000h	5.5.10.40/ 876
303A_018C	A7 PU PGC software down trigger status (GPC_A7_PU_PGC_PDN_STATUS0)	32	R	0000_0000h	5.5.10.41/ 878
303A_0190	A7 PU PGC software down trigger status (GPC_A7_PU_PGC_PDN_STATUS1)	32	R	0000_0000h	5.5.10.41/ 878
303A_0194	A7 PU PGC software down trigger status (GPC_A7_PU_PGC_PDN_STATUS2)	32	R	0000_0000h	5.5.10.41/ 878
303A_0198	M4 PU PGC software down trigger status (GPC_M4_PU_PGC_PDN_STATUS0)	32	R	0000_0000h	5.5.10.42/ 880
303A_019C	M4 PU PGC software down trigger status (GPC_M4_PU_PGC_PDN_STATUS1)	32	R	0000_0000h	5.5.10.42/ 880
303A_01A0	M4 PU PGC software down trigger status (GPC_M4_PU_PGC_PDN_STATUS2)	32	R	0000_0000h	5.5.10.42/ 880
303A_01B0	A7 MIX PDN FLG (GPC_A7_MIX_PDN_FLG)	32	R/W	0000_0000h	5.5.10.43/ 883
303A_01B4	A7 PU PDN FLG (GPC_A7_PU_PDN_FLG)	32	R	0000_0000h	5.5.10.44/ 884
303A_01B8	M4 MIX PDN FLG (GPC_M4_MIX_PDN_FLG)	32	R/W	0000_0000h	5.5.10.45/ 885
303A_01BC	M4 PU PDN FLG (GPC_M4_PU_PDN_FLG)	32	R	0000_0000h	5.5.10.46/ 886

5.5.10.1 Basic Low power control register of A7 platform (GPC_LPCR_A7_BSC)

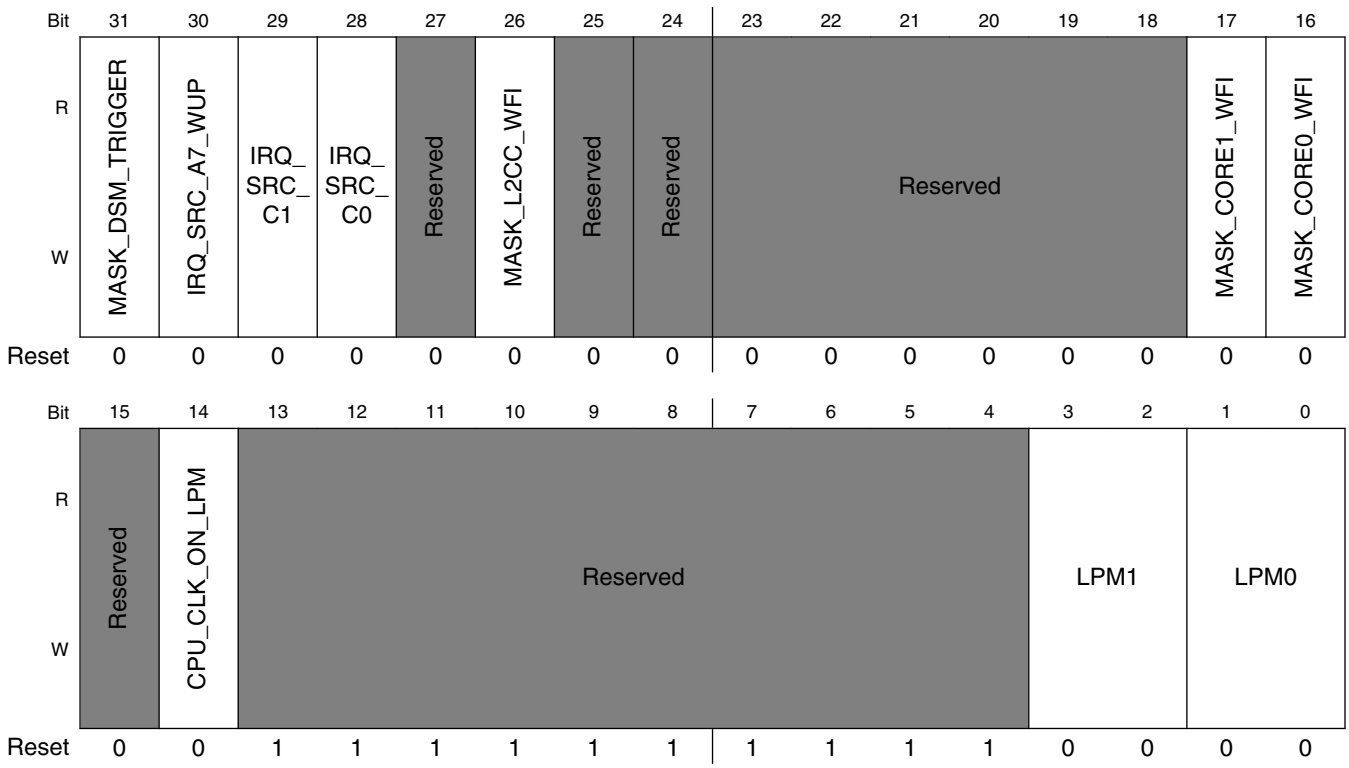
NOTE

LPCR_A7_BSC[CPU_CLK_ON_LPM] should be set 1'b1 when using A7 low power debug feature

NOTE

Always set LPM1/LPM0 with same value

Address: 303A_0000h base + 0h offset = 303A_0000h



GPC_LPCR_A7_BSC field descriptions

Field	Description
31 MASK_DSM_TRIGGER	DSM Trigger Mask 0 DSM trigger of A7 platform will not be masked 1 DSM trigger of A7 platform will be masked
30 IRQ_SRC_A7_WUP	LPCR_A7_BSC[IRQ_SRC_CO], LPCR_A7_BSC[IRQ_SRC_C1], and LPCR_A7_BSC[IRQ_SRC_A7_WUP] work together to decide the wake up source for A7 LPM and core0/core1 power. Please refer to "Power up process for A7 platform" for more specific information.

Table continues on the next page...

GPC_LPCR_A7_BSC field descriptions (continued)

Field	Description
	0 LPM wakeup source be “OR” result of LPCR_A7_BSC[IRQ_SRC_C0]/LPCR_A7_BSC[IRQ_SRC_C1] setting 1 LPM wakeup source from external INT[127:0], masked by IMR0
29 IRQ_SRC_C1	LPCR_A7_BSC[IRQ_SRC_CO], LPCR_A7_BSC[IRQ_SRC_C1], and LPCR_A7_BSC[IRQ_SRC_A7_WUP] work together to decide the wake up source for A7 LPM and core0/core1 power. Please refer to “Power up process for A7 platform” for more specific information. 0 core1 wakeup source from external INT[127:0], masked by IMR1 refer to “Power up process for A7 platform” for more specific information 1 core1 wakeup source from GIC(nFIQ[1]/nIRQ[1]), SCU should not be power down during low power mode when this bit is set to 1’b1
28 IRQ_SRC_C0	LPCR_A7_BSC[IRQ_SRC_CO], LPCR_A7_BSC[IRQ_SRC_C1], and LPCR_A7_BSC[IRQ_SRC_A7_WUP] work together to decide the wake up source for A7 LPM and core0/core1 power. Please refer to “Power up process for A7 platform” for more specific information. 0 core0 wakeup source from external INT[127:0], masked by IMR0 refer to “Power up process for A7 platform” for more specific information 1 core0 wakeup source from GIC(nFIQ[0]/nIRQ[0]), SCU should not be power down during low power mode when this bit is set to 1’b1
27 -	This field is reserved. Reserved
26 MASK_L2CC_WFI	L2 cache controller Wait For Interrupt Mask Register 0 WFI for L2 cache controller is not masked 1 WFI for L2 cache controller is masked
25 -	This field is reserved. Reserved
24 -	This field is reserved. Reserved
23–18 -	This field is reserved. Reserved
17 MASK_CORE1_WFI	CORE1 Wait For Interrupt Mask 0 WFI for CORE1 is not masked 1 WFI for CORE1 is masked
16 MASK_CORE0_WFI	CORE0 Wait For Interrupt Mask 0 WFI for CORE0 is not masked 1 WFI for CORE0 is masked
15 -	This field is reserved. Reserved
14 CPU_CLK_ON_LPM	Define if A7 clocks will be disabled on wait/stop mode. 0 A7 clock disabled on wait/stop mode 1 A7 clock enabled on wait/stop mode
13–4 -	This field is reserved. Reserved
3–2 LPM1	CORE1 Setting the low power mode that system will enter on next assertion of dsm_request signal.

Table continues on the next page...

GPC_LPCR_A7_BSC field descriptions (continued)

Field	Description
	00 Remain in RUN mode 01 Transfer to WAIT mode 10 Transfer to STOP mode 11 Reserved
LPM0	CORE0 Setting the low power mode that system will enter on next assertion of dsm_request signal. 00 Remain in RUN mode 01 Transfer to WAIT mode 10 Transfer to STOP mode 11 Reserved

5.5.10.2 Advanced Low power control register of A7 platform (GPC_LPCR_A7_AD)

Address: 303A_0000h base + 4h offset = 303A_0004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															L2_PGE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				EN_C1_PUP	EN_C1_IRQ_PUP	EN_C0_PUP	EN_C0_IRQ_PUP	Reserved			EN_PLAT_PDN	EN_C1_PDN	EN_C1_WFI_PDN	EN_C0_PDN	EN_C0_WFI_PDN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_LPCR_A7_AD field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 L2_PGE	0 L2 cache RAM will power down with SCU power domain in A7 platform (used for ALL_OFF mode) 1 L2 cache RAM will not power down with SCU power domain in A7 platform (used for L2 retention mode)
15–12 -	This field is reserved. Reserved
11 EN_C1_PUP	0 CORE1 will power up with low power mode request 1 CORE1 will not power up with low power mode request (only used wake up from CPU01_OFF mode)

Table continues on the next page...

GPC_LPCR_A7_AD field descriptions (continued)

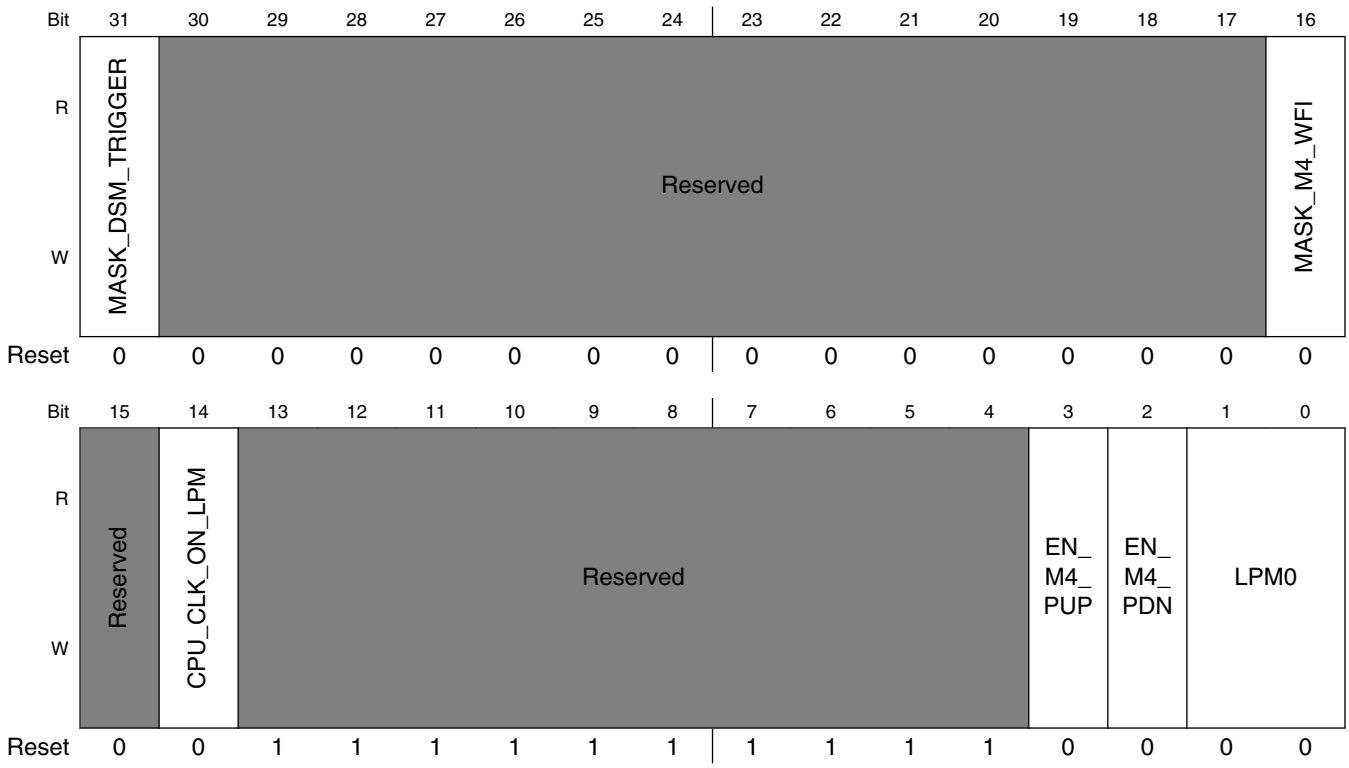
Field	Description
10 EN_C1_IRQ_PUP	0 CORE1 will power up with IRQ request 1 CORE1 will not power up with IRQ request
9 EN_C0_PUP	(only used wake up from CPU01_OFF mode) 0 CORE0 will power up with low power mode request 1 CORE0 will not power up with low power mode request
8 EN_C0_IRQ_PUP	0 CORE0 will power up with IRQ request 1 CORE0 will not power up with IRQ request
7–5 -	This field is reserved. Reserved
4 EN_PLAT_PDN	0 SCU and L2 cache RAM will not be power down with low power mode request 1 SCU and L2 cache RAM will be power down with low power mode request
3 EN_C1_PDN	0 CORE1 will not be power down with low power mode request 1 CORE1 will be power down with low power mode request
2 EN_C1_WFI_PDN	0 CORE1 will not be power down with WFI request 1 CORE1 will be power down with WFI request
1 EN_C0_PDN	0 CORE0 will not be power down with low power mode request 1 CORE0 will be power down with low power mode request
0 EN_C0_WFI_PDN	0 CORE0 will not be power down with WFI request 1 CORE0 will be power down with WFI request

5.5.10.3 Low power control register of CPU1 (GPC_LPCR_M4)

NOTE

LPCR_M4[CPU_CLK_ON_LPM] should be set 1'b0 if M4 goes to LPM without trigger power down of related domains

Address: 303A_0000h base + 8h offset = 303A_0008h



GPC_LPCR_M4 field descriptions

Field	Description
31 MASK_DSM_TRIGGER	M4 WFI Mask 0 DSM trigger of M4 platform will not be masked 1 DSM trigger of M4 platform will be masked
30–17 -	This field is reserved. Reserved
16 MASK_M4_WFI	M4 WFI Mask 0 WFI for M4 is not masked 1 WFI for M4 is masked
15 -	This field is reserved. Reserved

Table continues on the next page...

GPC_LPCR_M4 field descriptions (continued)

Field	Description
14 CPU_CLK_ON_LPM	Define if M4 clocks will be disabled on wait/stop mode. 0 M4 clock disabled on wait/stop mode. 1 M4 clock enabled on wait/stop mode.
13–4 -	This field is reserved. Reserved
3 EN_M4_PUP	Enable m4 virtual PGC power up with LPM enter
2 EN_M4_PDN	Enable m4 virtual PGC power down with LPM enter
LPM0	Setting the low power mode that system will enter on next assertion of dsm_request signal. 00 Remain in RUN mode 01 Transfer to WAIT mode 10 Transfer to STOP mode 11 Reserved

5.5.10.4 System low power control register (GPC_SLPCR)

NOTE

SLPCR[VSTBY] must be set to 1'b1 if SLPCR[RBC_EN] is set to 1'b1; SLPCR[SBYOS] must be set to 1'b1 if SLPCR[VSTBY] is set to 1'b1.

Address: 303A_0000h base + 14h offset = 303A_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EN_DSM	RBC_EN	REG_BYPASS_COUNT						DISABLE_A7_IS_DSM	Reserved			EN_M4_FASTWUP_STOP_MODE	EN_M4_FASTWUP_WAIT_MODE	EN_A7_FASTWUP_STOP_MODE	EN_A7_FASTWUP_WAIT_MODE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OSCCNT								COSC_EN	COSC_PWRDOWN	STBY_COUNT			VSTBY	SBYOS	BYPASS_PMIC_READY
W																
Reset	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0

GPC_SLPCR field descriptions

Field	Description
31 EN_DSM	DSM enable 0 DSM disabled 1 DSM enabled
30 RBC_EN	Enable for REG_BYPASS_COUNTER. If enabled, REG_BYPASS signal will be asserted after REG_BYPASS_COUNT clocks of CKIL, after standby voltage is requested. If standby voltage is not requested REG_BYPASS won't be asserted, even if counter is enabled. 0 REG_BYPASS_COUNTER disabled 1 REG_BYPASS_COUNTER enabled
29–24 REG_BYPASS_COUNT	Counter for REG_BYPASS signal assertion after standby voltage request by PMIC_STBY_REQ. 000000 no delay 000001 1 CKIL clock period delay 111111 63 CKIL clock period delay

Table continues on the next page...

GPC_SLPCR field descriptions (continued)

Field	Description
23 DISABLE_A7_IS_DSM	0 Enable a7 isolation signal in DSM 1 Disable a7 isolation signal in DSM
22–20 -	This field is reserved. Reserved
19 EN_M4_FASTWUP_STOP_MODE	Enable M4 fast wake up stop mode, relevant PLLs will not be closed in this mode.
18 EN_M4_FASTWUP_WAIT_MODE	Enable M4 fast wake up wait mode, relevant PLLs will not be closed in this mode.
17 EN_A7_FASTWUP_STOP_MODE	Enable A7 fast wake up stop mode, relevant PLLs will not be closed in this mode.
16 EN_A7_FASTWUP_WAIT_MODE	Enable A7 fast wake up wait mode, relevant PLLs will not be closed in this mode.
15–8 OSCCNT	Oscillator ready counter value. These bits define value of 32KHz counter, that serve as counter for oscillator lock time. This is used for oscillator lock time. Current estimation is ~5ms. This counter will be used in sequence out of DSM and if sbyos bit was defined. GPC will wait the “OSCCNT” number of cycles before it notify CCM to open the relevant PLLs. 00000000 count 1 ckil 11111111 count 256 ckils
7 COSC_EN	On-chip oscillator enable bit - this bit value is reflected on the output cosc_en. The system will start with on-chip oscillator enabled to supply source for the PLL’s. Software can change this bit if a transition to the bypass PLL clocks was performed for all the PLLs. In cases that this bit is changed from ‘0’ to ‘1’ then GPC will enable the on-chip oscillator and after counting oscnt ckil clock cycles before it notify CCM to open the relevant PLLs . The cosc_en bit should be changed only when on-chip oscillator is not chosen as the clock source. 0 Disable on-chip oscillator 1 Enable on-chip oscillator
6 COSC_PWRDOWN	In run mode, software can manually control powering down of on chip oscillator, i.e. generating ‘1’ on cosc_pwrdown signal. If software manually powered down the on chip oscillator, then sbyos functionality for on-chip oscillator will be bypassed. The manual closing of on-chip oscillator should be performed only in case the reference oscilator is not the source of all the clocks generation. 0 On-chip oscillator will not be powered down, i.e. cosc_pwrdown = 0 1 On-chip oscillator will be powered down, i.e. cosc_pwrdown = 1
5–3 STBY_COUNT	Standby counter definition. These two bits define, in the case of stop exit (if VSTBY bit was set), the amount of time GPC will wait between PMIC_STBY_REQ negation and the check of assertion of PMIC_READY. 000 GPC will wait 4 ckil clock cycles 001 GPC will wait 8 ckil clock cycles

Table continues on the next page...

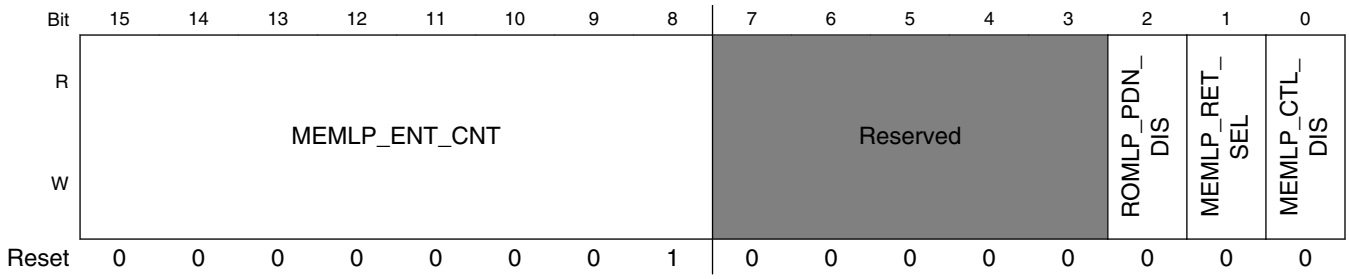
GPC_SLPCR field descriptions (continued)

Field	Description
	010 GPC will wait 16 ckil clock cycles 011 GPC will wait 32 ckil clock cycles 100 GPC will wait 64 ckil clock cycles 101 GPC will wait 128 ckil clock cycles 110 GPC will wait 256 ckil clock cycles 111 GPC will wait 512 ckil clock cycles
2 VSTBY	Voltage standby request bit. This bit defines if PMIC_STBY_REQ pin, which notifies external power management IC to move from functional voltage to standby voltage, will be asserted in stop mode. 0 Voltage will not be changed to standby voltage after next entrance to stop mode. (PMIC_STBY_REQ will remain negated - '0') 1 Voltage will be changed to standby voltage after next entrance to stop mode.
1 SBYOS	Standby clock oscillator bit. This bit defines if cosc_pwrdown, which power down the on chip oscillator, will be asserted in DSM. 0 On chip oscillator will not be powered down, after next entrance to DSM. 1 On chip oscillator will be powered down, after next entrance to DSM. When returning from DSM, external oscillator will be enabled again, on chip oscillator will return to oscillator mode , and after oscnt count GPC will continue with the exit from DSM process.
0 BYPASS_PMIC_READY	By asserting this bit GPC will bypass waiting for PMIC_READY signal when coming out of DSM. This should be used for PMIC's that don't support the PMIC_READY signal. 0 Don't bypass the PMIC_READY signal - GPC will wait for its assertion during exit of low power mode if standby voltage was enabled 1 Bypass the PMIC_READY signal - GPC will wait for its assertion during exit of low power mode if standby voltage was enabled

5.5.10.5 Memory low power control register (GPC_MLPCR)

Address: 303A_0000h base + 20h offset = 303A_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MEMLP_RET_PGEN								MEM_EXT_CNT							
Reset	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0



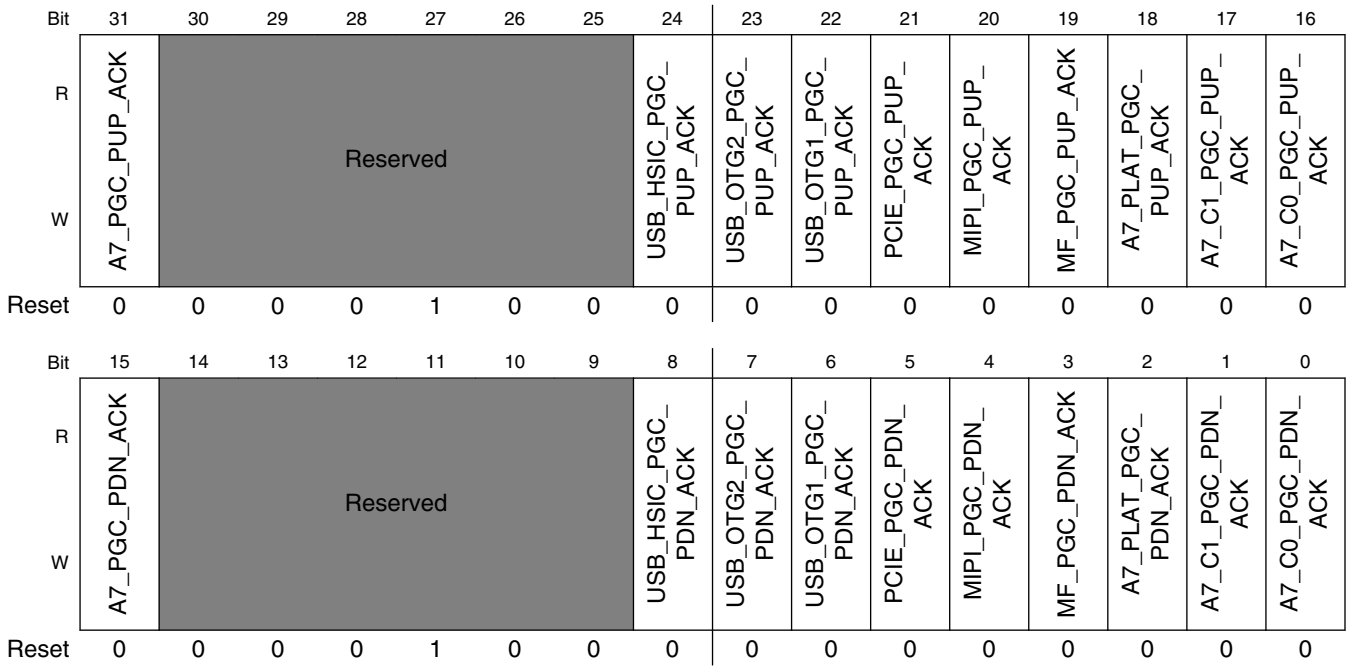
GPC_MLPCR field descriptions

Field	Description
31–24 MEMLP_RET_PGEN	Delay conter for “retnx” and “pgen”
23–16 MEM_EXT_CNT	Delay counter to start existing from memory low power
15–8 MEMLP_ENT_CNT	Delay counter to make sure all clock off after pll_dis_req is issued by smc
7–3 -	This field is reserved. Reserved
2 ROMLP_PDN_DIS	ROM shut down control 0 Enable ROM shut down control(should also enable RAM low power control); 1 Disable ROM shut down control
1 MEMLP_RET_SEL	Retention select 0 retention mode 2 1 retention mode 1
0 MEMLP_CTL_DIS	RAM low-power control 0 Enable RAM low power control 1 Disable RAM low power control

5.5.10.6 PGC acknowledge signal selection of A7 platform (GPC_PGC_ACK_SEL_A7)

The register can only be accessed by A7 platform

Address: 303A_0000h base + 24h offset = 303A_0024h



Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	A7_PGC_PUP_ACK	Reserved						USB_HSIC_PGC_PUP_ACK	USB_OTG2_PGC_PUP_ACK	USB_OTG1_PGC_PUP_ACK	PCIE_PGC_PUP_ACK	MIPI_PGC_PUP_ACK	MF_PGC_PUP_ACK	A7_PLAT_PGC_PUP_ACK	A7_C1_PGC_PUP_ACK	A7_C0_PGC_PUP_ACK
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	A7_PGC_PDN_ACK	Reserved						USB_HSIC_PGC_PDN_ACK	USB_OTG2_PGC_PDN_ACK	USB_OTG1_PGC_PDN_ACK	PCIE_PGC_PDN_ACK	MIPI_PGC_PDN_ACK	MF_PGC_PDN_ACK	A7_PLAT_PGC_PDN_ACK	A7_C1_PGC_PDN_ACK	A7_C0_PGC_PDN_ACK
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

GPC_PGC_ACK_SEL_A7 field descriptions

Field	Description
31 A7_PGC_PUP_ACK	Select power up acknowledge signal of A7 (dummy) PGC as the power up acknowledge for A7 LPM.
30–25 -	This field is reserved. Reserved
24 USB_HSIC_PGC_PUP_ACK	Select power up acknowledge signal of USB_HSIC PGC as the power up acknowledge for A7 LPM.
23 USB_OTG2_PGC_PUP_ACK	Select power up acknowledge signal of USB_OTG2 PGC as the power up acknowledge for A7 LPM.
22 USB_OTG1_PGC_PUP_ACK	Select power up acknowledge signal of USB_OTG1 PGC as the power up acknowledge for A7 LPM.
21 PCIE_PGC_PUP_ACK	Select power up acknowledge signal of PCIE PGC as the power up acknowledge for A7 LPM.
20 MIPI_PGC_PUP_ACK	Select power up acknowledge signal of MIPI PGC as the power up acknowledge for A7 LPM.
19 MF_PGC_PUP_ACK	Select power up acknowledge signal of MF PGC as the power up acknowledge for A7 LPM.
18 A7_PLAT_PGC_PUP_ACK	Select power up acknowledge signal of A7 PLATFORM PGC as the power up acknowledge for A7 LPM.

Table continues on the next page...

GPC_PGC_ACK_SEL_A7 field descriptions (continued)

Field	Description
17 A7_C1_PGC_PUP_ACK	Select power up acknowledge signal of A7 CORE1 PGC as the power up acknowledge for A7 LPM.
16 A7_C0_PGC_PUP_ACK	Select power up acknowledge signal of A7 CORE0 PGC as the power up acknowledge for A7 LPM.
15 A7_PGC_PDN_ACK	Select power down acknowledge signal of A7 (dummy) PGC as the power down acknowledge for A7 LPM.
14–9 -	This field is reserved. Reserved
8 USB_HSIC_PGC_PDN_ACK	Select power down acknowledge signal of USB_HSIC PGC as the power down acknowledge for A7 LPM.
7 USB_OTG2_PGC_PDN_ACK	Select power down acknowledge signal of USB_OTG2 PGC as the power down acknowledge for A7 LPM.
6 USB_OTG1_PGC_PDN_ACK	Select power down acknowledge signal of USB_OTG1 PGC as the power down acknowledge for A7 LPM.
5 PCIE_PGC_PDN_ACK	Select power down acknowledge signal of PCIE PGC as the power down acknowledge for A7 LPM.
4 MIPI_PGC_PDN_ACK	Select power down acknowledge signal of MIPI PGC as the power down acknowledge for A7 LPM.
3 MF_PGC_PDN_ACK	Select power down acknowledge signal of MIX PGC as the power down acknowledge for A7 LPM.
2 A7_PLAT_PGC_PDN_ACK	Select power down acknowledge signal of A7 PLATFORM PGC as the power down acknowledge for A7 LPM.
1 A7_C1_PGC_PDN_ACK	Select power down acknowledge signal of A7 CORE1 PGC as the power down acknowledge for A7 LPM.
0 A7_C0_PGC_PDN_ACK	Select power down acknowledge signal of A7 CORE0 PGC as the power down acknowledge for A7 LPM.

5.5.10.7 PGC acknowledge signal selection of M4 platform (GPC_PGC_ACK_SEL_M4)

This register can only be accessed by the M4 platform.

NOTE

"dummy" PGC cannot be mapped to time slot control. "virtual" PGC can be mapped to time slot control. When virtual PGC is used, below setting is required - GPC_MISC[M4_PDN_REQ_MASK] should be set to 1'b1 and arrange virtual GPC in same slot with MIX. power/up slot (See example code 3). MIX PGC may possibly power down later than A7 platform power down when virtual PGC is used.

Address: 303A_0000h base + 28h offset = 303A_0028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	M4_DUMMY_PGC_PUP_ACK	Reserved						USB_HSIC_PGC_PUP_ACK	USB_OTG2_PGC_PUP_ACK	USB_OTG1_PGC_PUP_ACK	PCIE_PGC_PUP_ACK	MIPI_PGC_PUP_ACK	MF_PGC_PUP_ACK	Reserved		M4_VIRTUAL_PGC_PUP_ACK
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	M4_DUMMY_PGC_PDN_ACK	Reserved						USB_HSIC_PGC_PDN_ACK	USB_OTG2_PGC_PDN_ACK	USB_OTG1_PGC_PDN_ACK	PCIE_PGC_PDN_ACK	MIPI_PGC_PDN_ACK	MF_PGC_PDN_ACK	Reserved		M4_VIRTUAL_PGC_PDN_ACK
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

GPC_PGC_ACK_SEL_M4 field descriptions

Field	Description
31 M4_DUMMY_PGC_PUP_ACK	Select power up acknowledge signal of M4 (dummy) PGC as the power up acknowledge for M4 LPM.
30–25 -	This field is reserved. Reserved

Table continues on the next page...

GPC_PGC_ACK_SEL_M4 field descriptions (continued)

Field	Description
24 USB_HSIC_ PGC_PUP_ACK	Select power up acknowledge signal of USB_HSIC PGC as the power up acknowledge for M4 LPM.
23 USB_OTG2_ PGC_PUP_ACK	Select power up acknowledge signal of USB_OTG2 PGC as the power up acknowledge for M4 LPM.
22 USB_OTG1_ PGC_PUP_ACK	Select power up acknowledge signal of USB_OTG1 PGC as the power up acknowledge for M4 LPM.
21 PCIE_PGC_ PUP_ACK	Select power up acknowledge signal of PCIE PGC as the power up acknowledge for M4 LPM.
20 MIPI_PGC_ PUP_ACK	Select power up acknowledge signal of MIPI PGC as the power up acknowledge for M4 LPM.
19 MF_PGC_PUP_ ACK	Select power up acknowledge signal of MF PGC as the power up acknowledge for M4 LPM.
18–17 -	This field is reserved. Reserved
16 M4_VIRTUAL_ PGC_PUP_ACK	Select power up acknowledge signal of M4 virtual PGC as the power up acknowledge for M4 LPM. M4 virtual PGC only acknowledge power up request in the end of current slot time
15 M4_DUMMY_ PGC_PDN_ACK	Select power down acknowledge signal of M4 (dummy) PGC as the power down acknowledge for M4 LPM.
14–9 -	This field is reserved. Reserved
8 USB_HSIC_ PGC_PDN_ACK	Select power down acknowledge signal of USB_HSIC PGC as the power down acknowledge for M4 LPM.
7 USB_OTG2_ PGC_PDN_ACK	Select power down acknowledge signal of USB_OTG2 PGC as the power down acknowledge for M4 LPM.
6 USB_OTG1_ PGC_PDN_ACK	Select power down acknowledge signal of USB_OTG1 PGC as the power down acknowledge for M4 LPM.
5 PCIE_PGC_ PDN_ACK	Select power down acknowledge signal of PCIE PGC as the power down acknowledge for M4 LPM.
4 MIPI_PGC_ PDN_ACK	Select power down acknowledge signal of MIPI PGC as the power down acknowledge for M4 LPM.
3 MF_PGC_PDN_ ACK	Select power down acknowledge signal of MIX PGC as the power down acknowledge for M4 LPM.
2–1 -	This field is reserved. Reserved

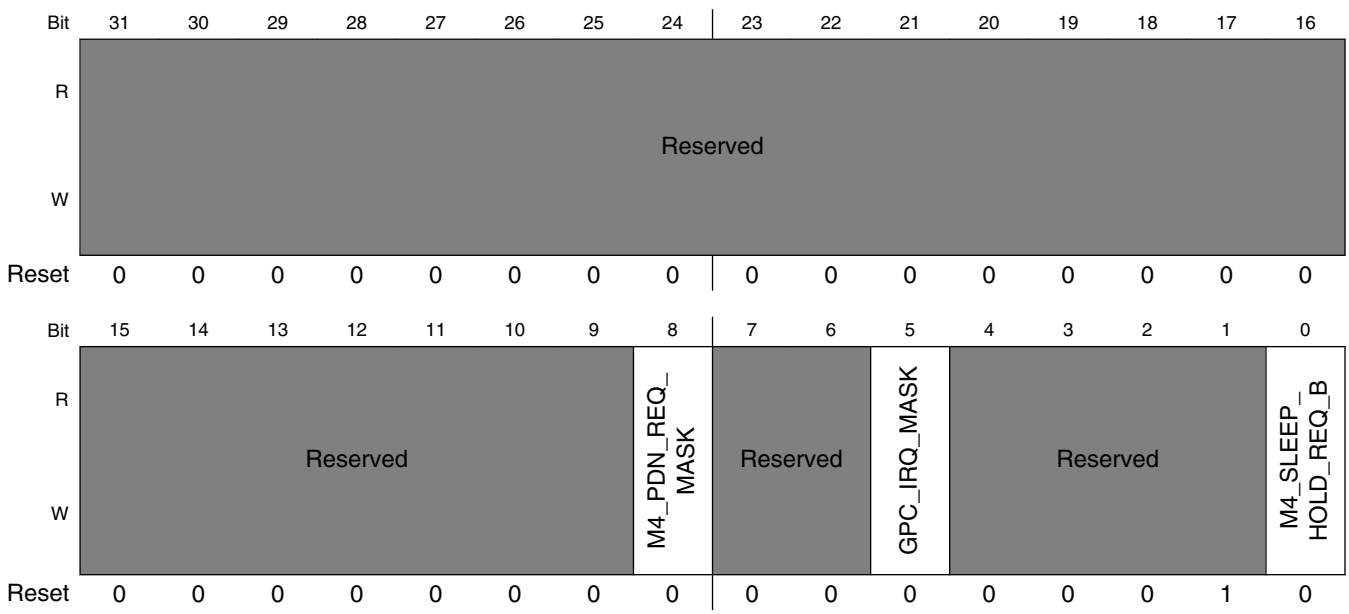
Table continues on the next page...

GPC_PGC_ACK_SEL_M4 field descriptions (continued)

Field	Description
0 M4_VIRTUAL_ PGC_PDN_ACK	Select power down acknowledge signal of M4 virtual PGC as the power down acknowledge for M4 LPM. M4 virtual PGC only acknowledge power down request in the end of current slot time

5.5.10.8 GPC Miscellaneous register (GPC_MISC)

Address: 303A_0000h base + 2Ch offset = 303A_002Ch



GPC_MISC field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 M4_PDN_REQ_ MASK	M4 power-down mask 0 M4 power down request to virtual M4 PGC will be masked. 1 M4 power down request to virtual M4 PGC will not be masked. Set this bit to 1'b1 when M4 virtual PGC is used.
7–6 -	This field is reserved. Reserved
5 GPC_IRQ_MASK	GPC interrupt/event masking 0 Not masked 1 Interrupt / event is masked
4–1 -	This field is reserved. Reserved

Table continues on the next page...

GPC_MISC field descriptions (continued)

Field	Description
0 M4_SLEEP_ HOLD_REQ_B	M4 sleep hold 0 Hold M4 platform in sleep mode. This bit is a software control bit to M4 platform. 1 Don't hold M4 platform in sleep mode.

5.5.10.9 IRQ masking register 1 of A7 core0 (GPC_IMR1_CORE0_A7)

The four IMRn_CORE0_A7 (n = 1,2,3,4) registers are used as interrupt mask for A7 core0.

Address: 303A_0000h base + 30h offset = 303A_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR1_CORE0_A7																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR1_CORE0_A7 field descriptions

Field	Description
IMR1_CORE0_ A7	A7 core0 IRQ[31:0] masking bits: 0 IRQ not masked 1 IRQ masked

5.5.10.10 IRQ masking register 2 of A7 core0 (GPC_IMR2_CORE0_A7)

Address: 303A_0000h base + 34h offset = 303A_0034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR2_CORE0_A7																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR2_CORE0_A7 field descriptions

Field	Description
IMR2_CORE0_ A7	A7 core0 IRQ[63:32] masking bits: 0 IRQ not masked 1 IRQ masked

5.5.10.11 IRQ masking register 3 of A7 core0 (GPC_IMR3_CORE0_A7)

Address: 303A_0000h base + 38h offset = 303A_0038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR3_CORE0_A7																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR3_CORE0_A7 field descriptions

Field	Description
IMR3_CORE0_A7	A7 core0 IRQ[95:64] masking bits: 0 IRQ not masked 1 IRQ masked

5.5.10.12 IRQ masking register 4 of A7 core0 (GPC_IMR4_CORE0_A7)

Address: 303A_0000h base + 3Ch offset = 303A_003Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR4_CORE0_A7																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR4_CORE0_A7 field descriptions

Field	Description
IMR4_CORE0_A7	A7 core0 IRQ[127:96] masking bits: 0 IRQ not masked 1 IRQ masked

5.5.10.13 IRQ masking register 1 of A7 core1 (GPC_IMR1_CORE1_A7)

The four IMRn_CORE1_A7 (n = 1,2,3,4) registers are used as interrupt mask for A7 core1.

Address: 303A_0000h base + 40h offset = 303A_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR1_CORE1_A7																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR1_CORE1_A7 field descriptions

Field	Description
IMR1_CORE1_A7	A7 core1 IRQ[31:0] masking bits: 0 IRQ not masked 1 IRQ masked

5.5.10.14 IRQ masking register 2 of A7 core1 (GPC_IMR2_CORE1_A7)

Address: 303A_0000h base + 44h offset = 303A_0044h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR2_CORE1_A7																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR2_CORE1_A7 field descriptions

Field	Description
IMR2_CORE1_A7	A7 core1 IRQ[63:32] masking bits: 0 IRQ not masked 1 IRQ masked

5.5.10.15 IRQ masking register 3 of A7 core1 (GPC_IMR3_CORE1_A7)

Address: 303A_0000h base + 48h offset = 303A_0048h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR3_CORE1_A7																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR3_CORE1_A7 field descriptions

Field	Description
IMR3_CORE1_A7	A7 core1 IRQ[95:64] masking bits: 0 IRQ not masked 1 IRQ masked

5.5.10.16 IRQ masking register 4 of A7 core1 (GPC_IMR4_CORE1_A7)

Address: 303A_0000h base + 4Ch offset = 303A_004Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR4_CORE1_A7																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR4_CORE1_A7 field descriptions

Field	Description
IMR4_CORE1_A7	A7 core1 IRQ[127:96] masking bits: 0 IRQ not masked 1 IRQ masked

5.5.10.17 IRQ masking register 1 of M4 (GPC_IMR1_M4)

The four IMRn_M4 (n = 1,2,3,4) registers are used as interrupt mask for M4.

Address: 303A_0000h base + 50h offset = 303A_0050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR1_M4																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR1_M4 field descriptions

Field	Description
IMR1_M4	M4 IRQ[31:0] masking bits: 0 IRQ not masked 1 IRQ masked

5.5.10.18 IRQ masking register 2 of M4 (GPC_IMR2_M4)

Address: 303A_0000h base + 54h offset = 303A_0054h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR2_M4																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR2_M4 field descriptions

Field	Description
IMR2_M4	M4 IRQ[63:32] masking bits: 0 IRQ not masked 1 IRQ masked

5.5.10.19 IRQ masking register 3 of M4 (GPC_IMR3_M4)

Address: 303A_0000h base + 58h offset = 303A_0058h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR3_M4																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR3_M4 field descriptions

Field	Description
IMR3_M4	M4 IRQ[95:64] masking bits: 0 IRQ not masked 1 IRQ masked

5.5.10.20 IRQ masking register 4 of M4 (GPC_IMR4_M4)

Address: 303A_0000h base + 5Ch offset = 303A_005Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR4_M4																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR4_M4 field descriptions

Field	Description
IMR4_M4	M4 IRQ[127:96] masking bits: 0 IRQ not masked 1 IRQ masked

5.5.10.21 IRQ status register 1 of A7 (GPC_ISR1_A7)

The four ISRn_A7 (n = 1,2,3,4) registers, all of them are read only register

Address: 303A_0000h base + 70h offset = 303A_0070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISR1_A7																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_ISR1_A7 field descriptions

Field	Description
ISR1_A7	A7 IRQ[31:0] status

5.5.10.22 IRQ status register 2 of A7 (GPC_ISR2_A7)

Address: 303A_0000h base + 74h offset = 303A_0074h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISR2_A7																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_ISR2_A7 field descriptions

Field	Description
ISR2_A7	A7 IRQ[63:32] status

5.5.10.23 IRQ status register 3 of A7 (GPC_ISR3_A7)

Address: 303A_0000h base + 78h offset = 303A_0078h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISR3_A7																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_ISR3_A7 field descriptions

Field	Description
ISR3_A7	A7 IRQ[95:64] status

5.5.10.24 IRQ status register 4 of A7 (GPC_ISR4_A7)

Address: 303A_0000h base + 7Ch offset = 303A_007Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISR4_A7																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_ISR4_A7 field descriptions

Field	Description
ISR4_A7	A7 IRQ[127:96] status

5.5.10.25 IRQ status register 1 of M4 (GPC_ISR1_M4)

The four ISRn_M4 (n = 1,2,3,4) registers, all of them are read only register

Address: 303A_0000h base + 80h offset = 303A_0080h

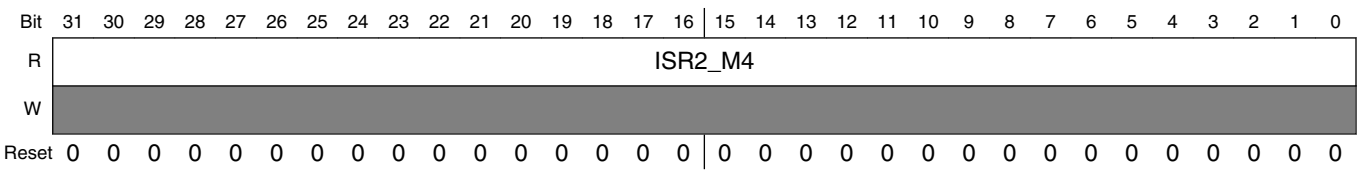
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISR1_M4																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_ISR1_M4 field descriptions

Field	Description
ISR1_M4	M4 IRQ[31:0] status

5.5.10.26 IRQ status register 2 of M4 (GPC_ISR2_M4)

Address: 303A_0000h base + 84h offset = 303A_0084h

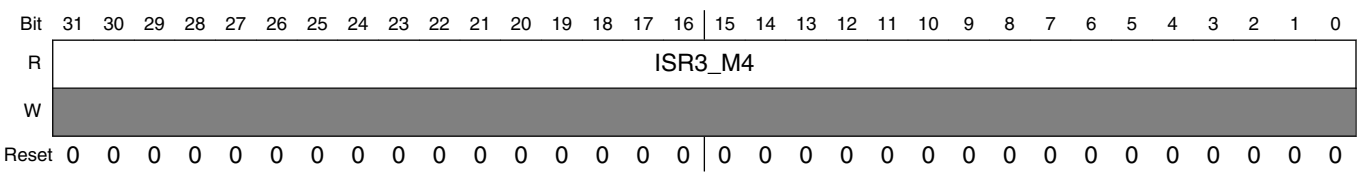


GPC_ISR2_M4 field descriptions

Field	Description
ISR2_M4	M4 IRQ[63:32] status

5.5.10.27 IRQ status register 3 of M4 (GPC_ISR3_M4)

Address: 303A_0000h base + 88h offset = 303A_0088h



GPC_ISR3_M4 field descriptions

Field	Description
ISR3_M4	M4 IRQ[95:64] status

5.5.10.28 IRQ status register 4 of M4 (GPC_ISR4_M4)

Address: 303A_0000h base + 8Ch offset = 303A_008Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISR4_M4																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_ISR4_M4 field descriptions

Field	Description
ISR4_M4	M4 IRQ[127:96] status

5.5.10.29 Slot configure register (GPC_SLT*n*_CFG)

There are 10 slots in each SLTn_CFG(n = 0~9) will define the power up or power down behavior of one or more PGC in that each slot.

In each “SLTn_cfg”, 2 bits (slt_cfg[1:0])are reserved for each PGC:

- 2’b01 (slot controller will power down relevant PGC in corresponding slot if hardware power down request asserted)
- 2’b10 (slot controller will power up relevant PGC in corresponding slot if hardware power up request asserted)
- 2’b00 or 2’b11 (not power down or power up behavior in relevant slot)

The specific bits assignment for each PGC is shown in the table below.

	PGCx	PGCx-1	..	PGC2	PGC1	PGC0
SLT0_CFG	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]
SLT1_CFG	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]
:	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]
SLTn_CFG	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]	slt_cfg[1:0]

Address: 303A_0000h base + B0h offset + (4d × i), where i=0d to 9d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												M4_VIRTUAL_PUP_SLOT_CONTROL	M4_VIRTUAL_PDN_SLOT_CONTROL	USB_HSIC_PUP_SLOT_CONTROL	USB_HSIC_PDN_SLOT_CONTROL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	USB_OTG2_PUP_SLOT_CONTROL	USB_OTG2_PDN_SLOT_CONTROL	USB_OTG1_PUP_SLOT_CONTROL	USB_OTG1_PDN_SLOT_CONTROL	PCIE_PHY_PUP_SLOT_CONTROL	PCIE_PHY_PDN_SLOT_CONTROL	MIPI_PHY_PUP_SLOT_CONTROL	MIPI_PHY_PDN_SLOT_CONTROL	FASTMEGA_PUP_SLOT_CONTROL	FASTMEGA_PDN_SLOT_CONTROL	SCU_PUP_SLOT_CONTROL	SCU_PDN_SLOT_CONTROL	CORE1_A7_PUP_SLOT_CONTROL	CORE1_A7_PDN_SLOT_CONTROL	CORE0_A7_PUP_SLOT_CONTROL	CORE0_A7_PDN_SLOT_CONTROL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_SLT*n*_CFG field descriptions

Field	Description
31–20 -	This field is reserved. Reserved
19 M4_VIRTUAL_PUP_SLOT_CONTROL	M4_VIRTUAL Power-up slot control
18 M4_VIRTUAL_PDN_SLOT_CONTROL	M4_VIRTUAL Power-down slot control
17 USB_HSIC_PUP_SLOT_CONTROL	USB_HSIC Power-up slot control
16 USB_HSIC_PDN_SLOT_CONTROL	USB_HSIC Power-down slot control
15 USB_OTG2_PUP_SLOT_CONTROL	USB_OTG2 Power-up slot control
14 USB_OTG2_PDN_SLOT_CONTROL	USB_OTG2 Power-down slot control
13 USB_OTG1_PUP_SLOT_CONTROL	USB_OTG1 Power-up slot control
12 USB_OTG1_PDN_SLOT_CONTROL	USB_OTG1 Power-down slot control
11 PCIE_PHY_PUP_SLOT_CONTROL	PCIE_PHY Power-up slot control

Table continues on the next page...

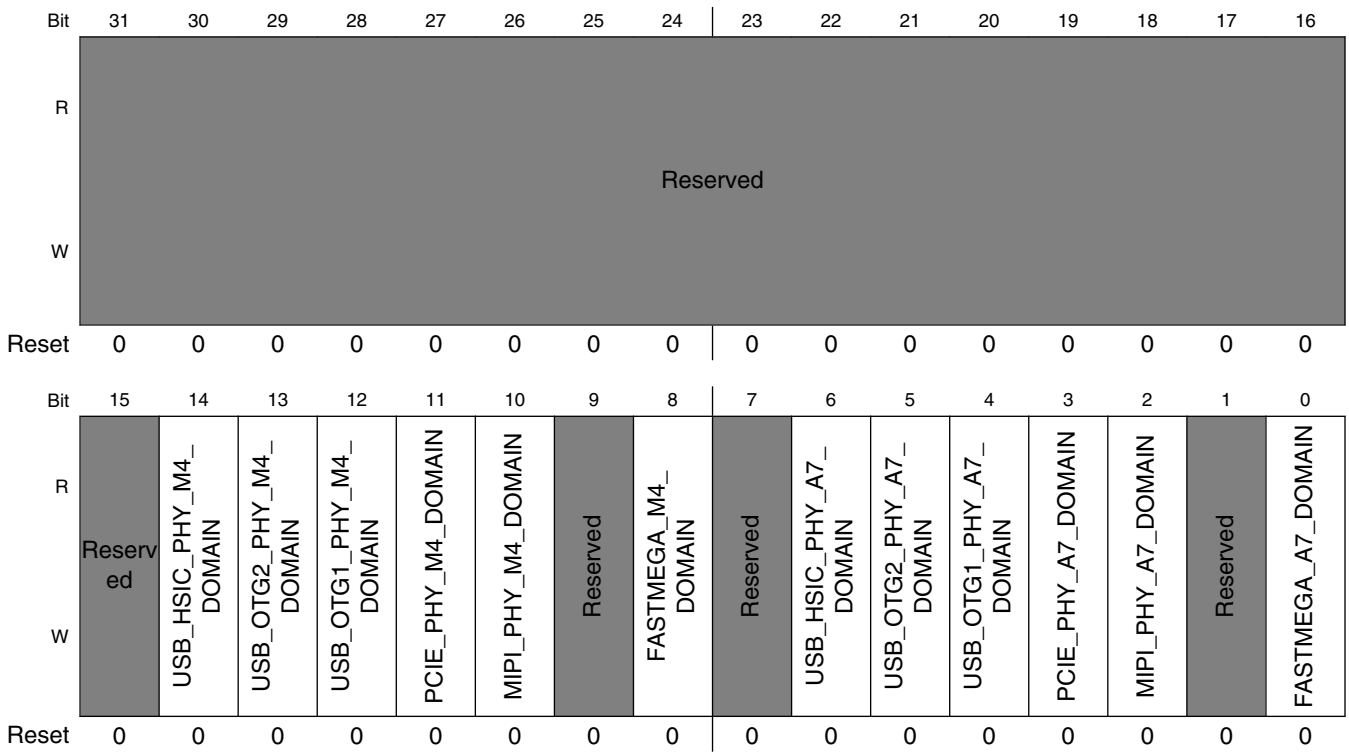
GPC_SLT*n*_CFG field descriptions (continued)

Field	Description
10 PCIE_PHY_ PDN_SLOT_ CONTROL	PCIE_PHY Power-down slot control
9 MIPI_PHY_PUP_ SLOT_ CONTROL	MIPI_PHY Power-up slot control
8 MIPI_PHY_PDN_ SLOT_ CONTROL	MIPI_PHY Power-down slot control
7 FASTMEGA_ PUP_SLOT_ CONTROL	FASTMEGA Power-up slot control
6 FASTMEGA_ PDN_SLOT_ CONTROL	FASTMEGA Power-down slot control
5 SCU_PUP_ SLOT_ CONTROL	SCU Power-up slot control
4 SCU_PDN_ SLOT_ CONTROL	SCU Power-down slot control
3 CORE1_A7_ PUP_SLOT_ CONTROL	CORE1 A7 Power-up slot control
2 CORE1_A7_ PDN_SLOT_ CONTROL	CORE1 A7 Power-down slot control
1 CORE0_A7_ PUP_SLOT_ CONTROL	CORE0 A7 Power-up slot control
0 CORE0_A7_ PDN_SLOT_ CONTROL	CORE0 A7 Power-down slot control

5.5.10.30 PGC CPU mapping (GPC_PGC_CPU_MAPPING)

The byte0 can only be accessed by A7 platform, byte1 can only be accessed by M4 platform.

Address: 303A_0000h base + ECh offset = 303A_00ECh



GPC_PGC_CPU_MAPPING field descriptions

Field	Description
31–15 -	This field is reserved. Reserved
14 USB_HSIC_PHY_M4_DOMAIN	USB_HSIC_PHY mapping 0 Don't map USB_HSIC_PHY to M4 domain 1 Map USB_HSIC_PHY to M4 domain
13 USB_OTG2_PHY_M4_DOMAIN	USB_OTG2_PHY mapping 0 Don't map USB_OTG2_PHY to M4 domain 1 Map USB_OTG2_PHY to M4 domain
12 USB_OTG1_PHY_M4_DOMAIN	USB_OTG1_PHY mapping 0 Don't map USB_OTG1_PHY to M4 domain 1 Map USB_OTG1_PHY to M4 domain

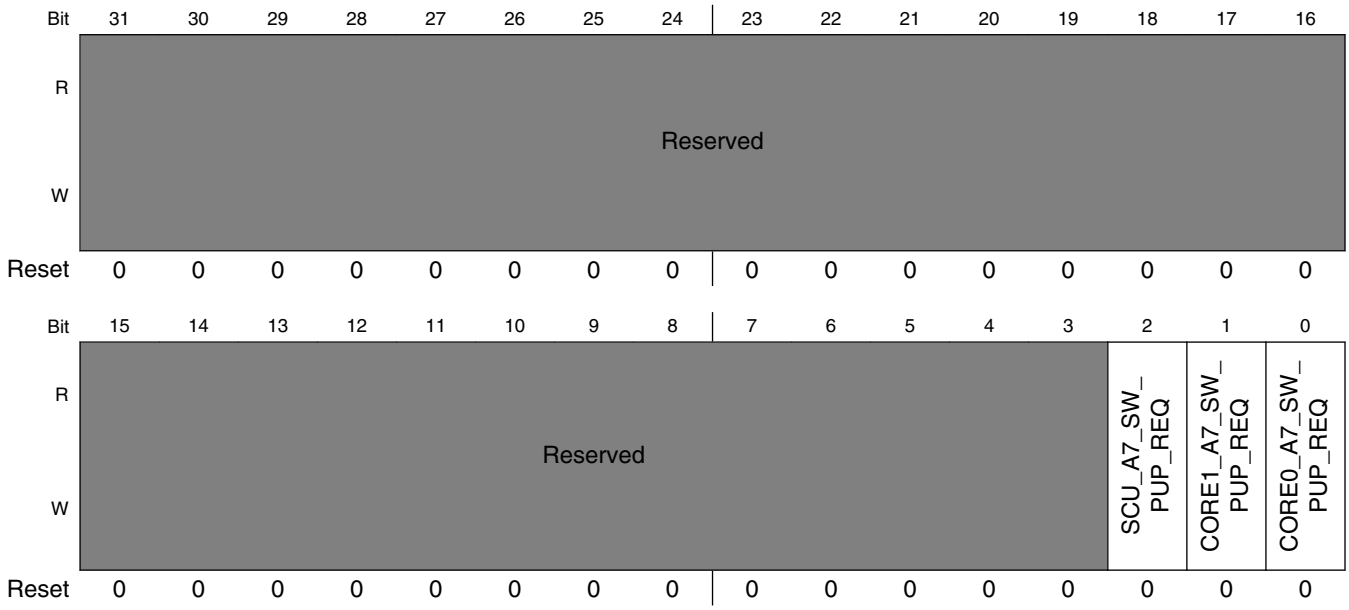
Table continues on the next page...

GPC_PGC_CPU_MAPPING field descriptions (continued)

Field	Description
11 PCIE_PHY_M4_DOMAIN	MIPI_PHY mapping 0 Don't map PCIE_PHY to M4 domain 1 Map PCIE_PHY to M4 domain
10 MIPI_PHY_M4_DOMAIN	MIPI_PHY mapping 0 Don't map MIPI_PHY to M4 domain 1 Map MIPI_PHY to M4 domain
9 -	This field is reserved. Reserved
8 FASTMEGA_M4_DOMAIN	FAST/MEGA mapping 0 Don't map FAST/MEGA to M4 domain 1 Map FAST/MEGA to M4 domain
7 -	This field is reserved. Reserved
6 USB_HSIC_PHY_A7_DOMAIN	USB_HSIC_PHY mapping 0 Don't map USB_HSIC_PHY to A7 domain 1 Map USB_HSIC_PHY to A7 domain
5 USB_OTG2_PHY_A7_DOMAIN	USB_OTG2_PHY mapping 0 Don't map USB_OTG2_PHY to A7 domain 1 Map USB_OTG2_PHY to A7 domain
4 USB_OTG1_PHY_A7_DOMAIN	USB_OTG1_PHY mapping 0 Don't map USB_OTG1_PHY to A7 domain 1 Map USB_OTG1_PHY to A7 domain
3 PCIE_PHY_A7_DOMAIN	MIPI_PHY mapping 0 Don't map PCIE_PHY to A7 domain 1 Map PCIE_PHY to A7 domain
2 MIPI_PHY_A7_DOMAIN	MIPI_PHY mapping 0 Don't map MIPI_PHY to A7 domain 1 Map MIPI_PHY to A7 domain
1 -	This field is reserved. Reserved
0 FASTMEGA_A7_DOMAIN	FAST/MEGA mapping 0 Don't map FAST/MEGA to A7 domain 1 Map FAST/MEGA to A7 domain

5.5.10.31 CPU PGC software up trigger (GPC_CPU_PGC_SW_PUP_REQ)

Address: 303A_0000h base + F0h offset = 303A_00F0h

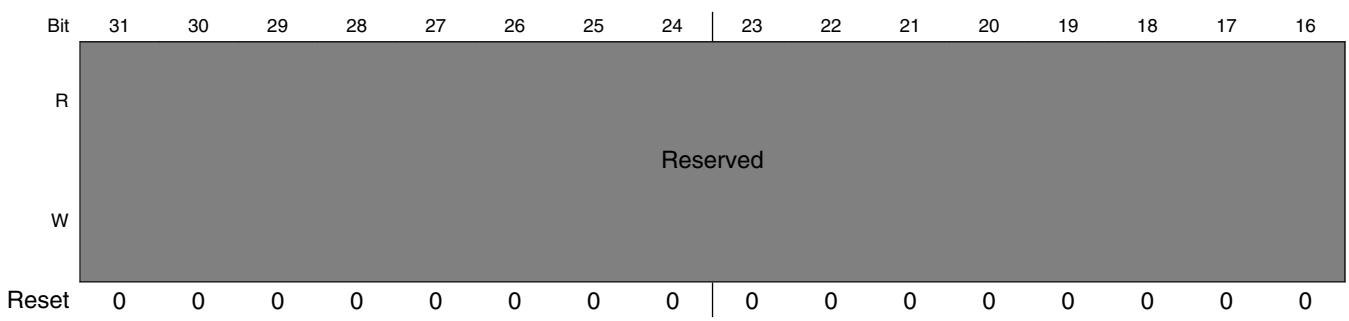


GPC_CPU_PGC_SW_PUP_REQ field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
2 SCU_A7_SW_PUP_REQ	Software power up trigger for SCU A7
1 CORE1_A7_SW_PUP_REQ	Software power up trigger for Core1 A7 PGC
0 CORE0_A7_SW_PUP_REQ	Software power up trigger for Core0 A7 PGC

5.5.10.32 PU PGC software up trigger (GPC_PU_PGC_SW_PUP_REQ)

Address: 303A_0000h base + F8h offset = 303A_00F8h



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved											USB_HSIC_PHY_SW_PUP_REQ	USB_OTG2_PHY_SW_PUP_REQ	USB_OTG1_PHY_SW_PUP_REQ	PCIE_PHY_SW_PUP_REQ	MIPI_PHY_SW_PUP_REQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

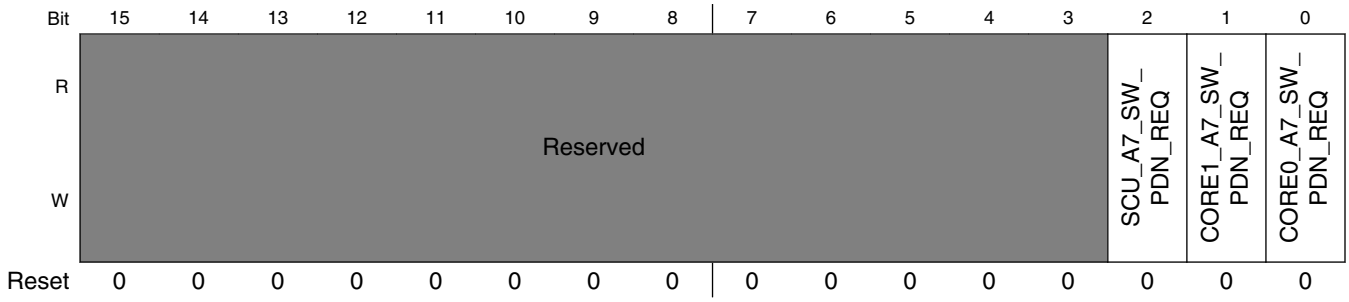
GPC_PU_PGC_SW_PUP_REQ field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 USB_HSIC_PHY_SW_PUP_REQ	Software power up trigger for USB_HSIC_PHY
3 USB_OTG2_PHY_SW_PUP_REQ	Software power up trigger for USB_OTG2_PHY
2 USB_OTG1_PHY_SW_PUP_REQ	Software power up trigger for USB_OTG1_PHY
1 PCIE_PHY_SW_PUP_REQ	Software power up trigger for PCIE_PHY
0 MIPI_PHY_SW_PUP_REQ	Software power up trigger for MIPI_PHY

5.5.10.33 CPU PGC software down trigger (GPC_CPU_PGC_SW_PDN_REQ)

Address: 303A_0000h base + FCh offset = 303A_00FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

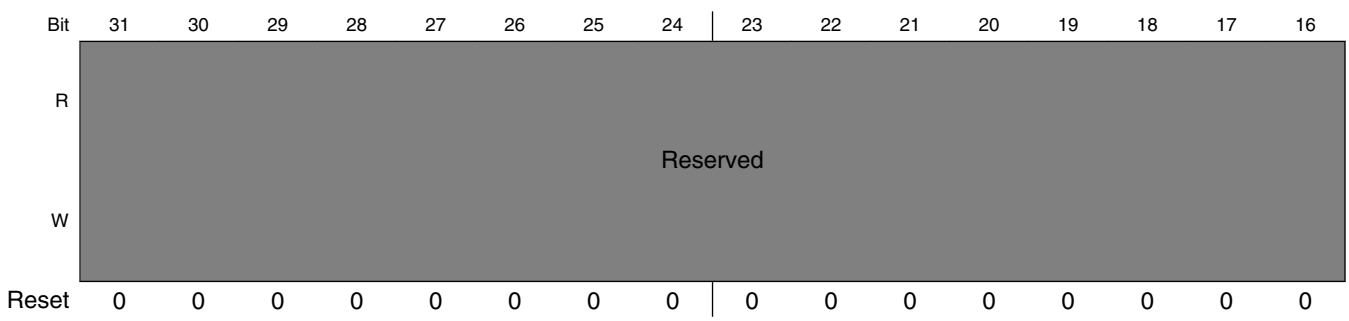


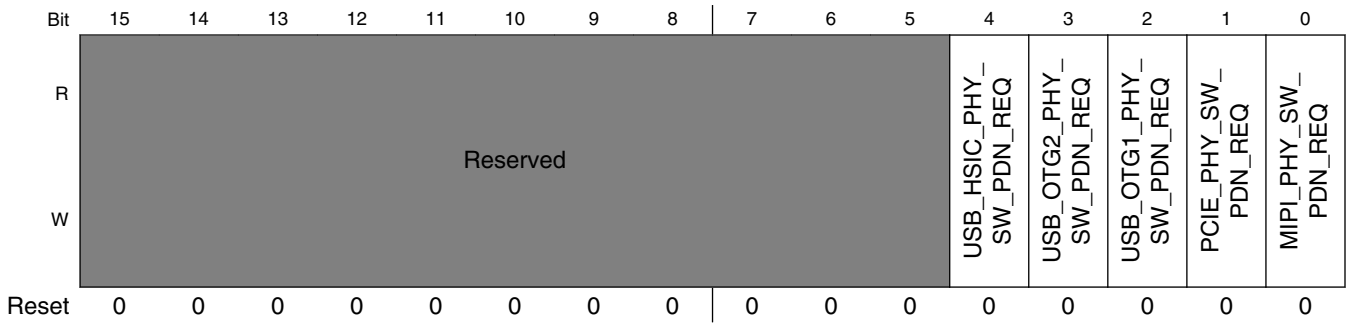
GPC_CPU_PGC_SW_PDN_REQ field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
2 SCU_A7_SW_PDN_REQ	Software power down trigger for SCU A7
1 CORE1_A7_SW_PDN_REQ	Software power down trigger for Core1 A7 PGC
0 CORE0_A7_SW_PDN_REQ	Software power down trigger for Core0 A7 PGC

5.5.10.34 PU PGC software down trigger (GPC_PU_PGC_SW_PDN_REQ)

Address: 303A_0000h base + 104h offset = 303A_0104h





GPC_PU_PGC_SW_PDN_REQ field descriptions

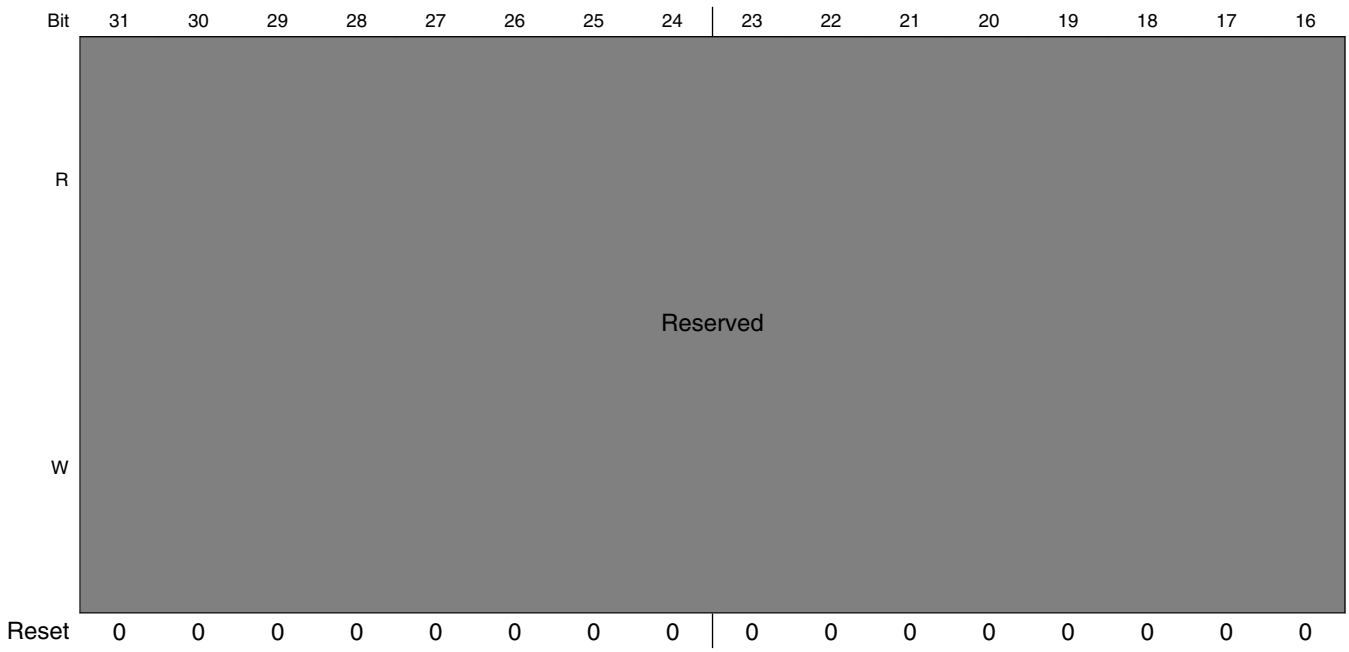
Field	Description
31–5 -	This field is reserved. Reserved
4 USB_HSIC_PHY_SW_PDN_REQ	Software power down trigger for USB_HSIC_PHY
3 USB_OTG2_PHY_SW_PDN_REQ	Software power down trigger for USB_OTG2_PHY
2 USB_OTG1_PHY_SW_PDN_REQ	Software power down trigger for USB_OTG1_PHY
1 PCIE_PHY_SW_PDN_REQ	Software power down trigger for PCIE_PHY
0 MIPI_PHY_SW_PDN_REQ	Software power down trigger for MIPI_PHY

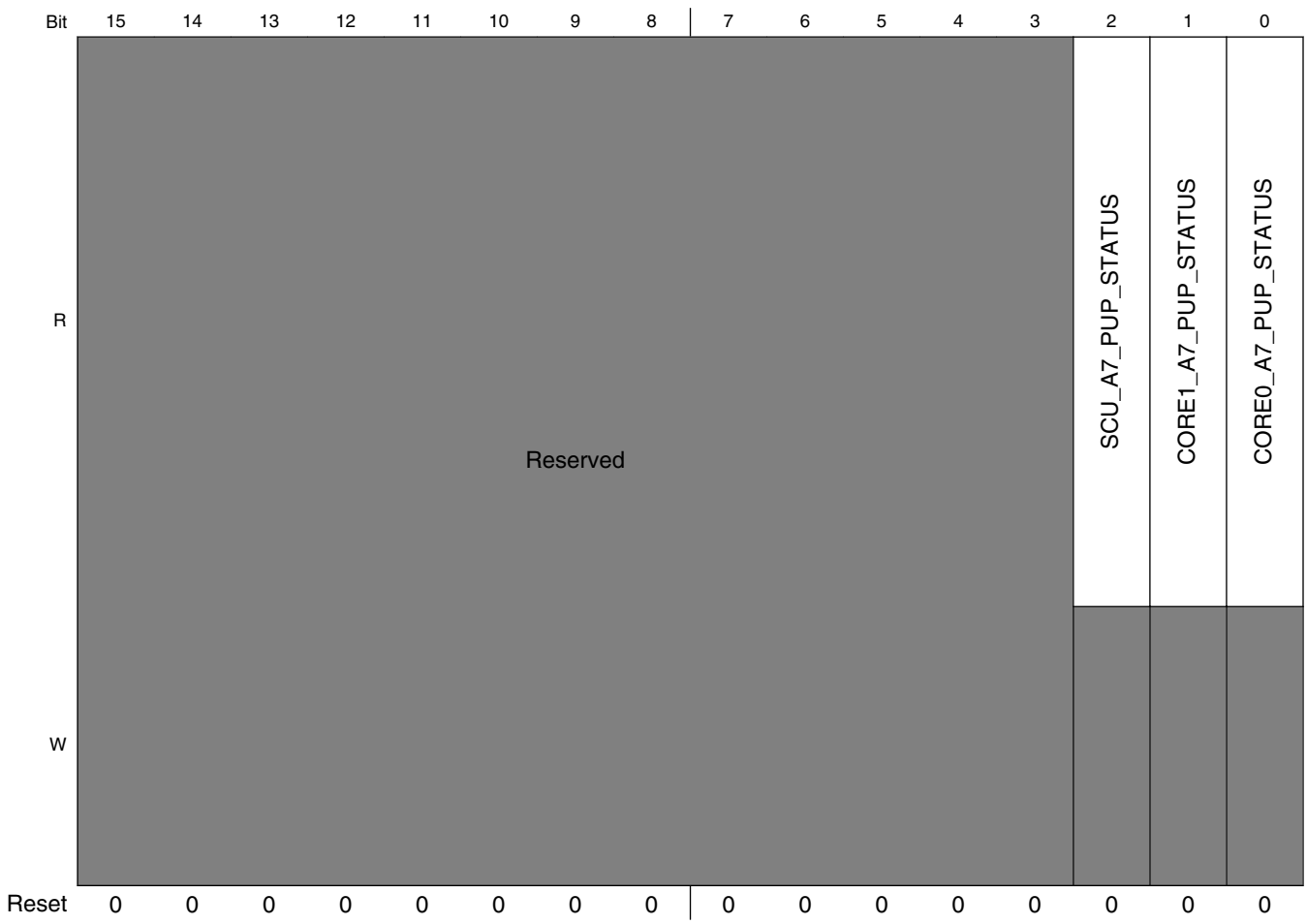
5.5.10.35 CPU PGC software up trigger status1 (GPC_CPU_PGC_PUP_STATUS1)

CPU_PGC_PUP_STATUS1 is a read only register, represents the results for power up software trigger for CPU type PGCs.

The field description is show in table below, the value of “1’b1” represent the software power up trigger failed because the relevant PGC is in a power down process. The relevant bit will be cleared after a success operation of power up software trigger for CPU type PGCs.

Address: 303A_0000h base + 130h offset = 303A_0130h





GPC_CPU_PGC_PUP_STATUS1 field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
2 SCU_A7_PUP_STATUS	
1 CORE1_A7_PUP_STATUS	
0 CORE0_A7_PUP_STATUS	

5.5.10.36 A7 MIX software up trigger status register (GPC_A7_MIX_PGC_PUP_STATUS*n*)

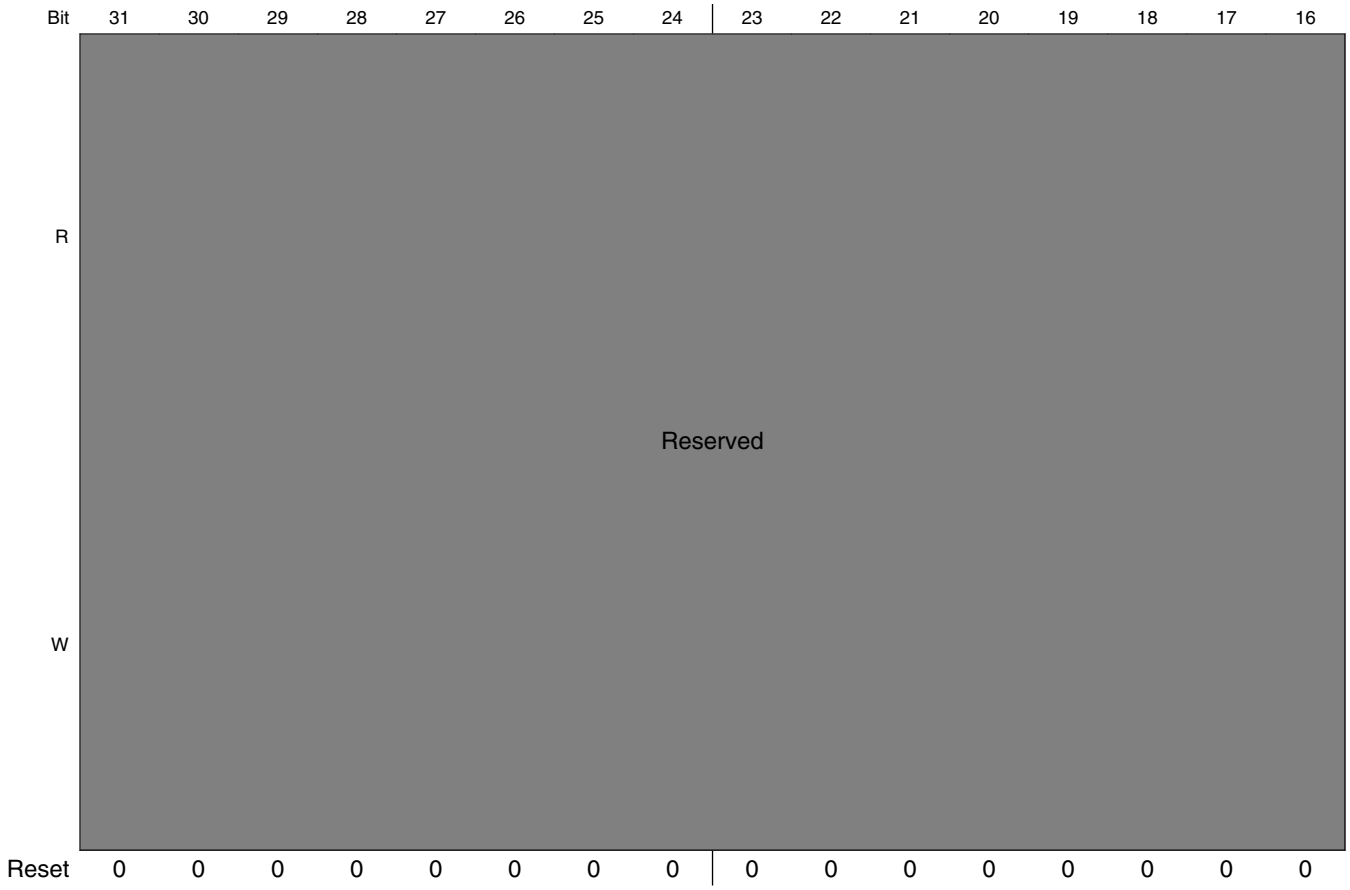
A7_MIX_PGC_PUP_STATUSn (n = 0,1,2) are a read only register, represents the results for power up software trigger from A7 platform to MIX type PGCs.

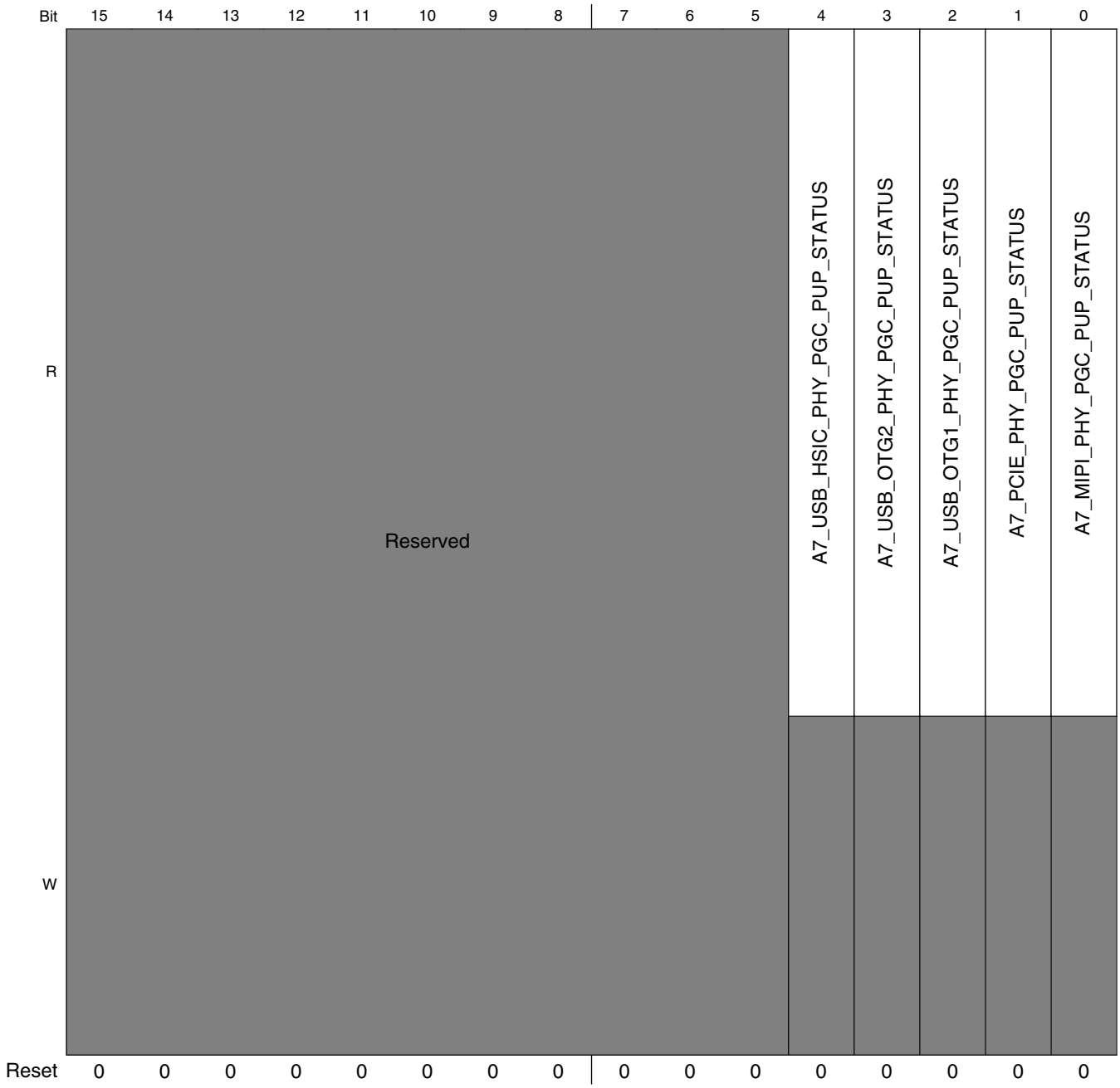
A7_MIX_PGC_PUP_STATUS0: value of “1’b1” represent the software power up trigger failed because domain control condition. The relevant bit will be cleared after a success operation of power up software trigger for MIX type PGCs.

A7_MIX_PGC_PUP_STATUS1: value of “1’b1” represent the software power up trigger failed because the relevant PGC is in a power up process. The relevant bit will be cleared after a success operation of power up software trigger for MIX type PGCs.

A7_MIX_PGC_PUP_STATUS2: value of “1’b1” represent the software power up trigger failed because time slot control is busy. The relevant bit will be cleared after a success operation of power up software trigger for MIX type PGCs.

Address: 303A_0000h base + 134h offset + (4d × i), where i=0d to 2d





GPC_A7_MIX_PGC_PUP_STATUS*n* field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 A7_USB_HSIC_PHY_PGC_PUP_STATUS	

Table continues on the next page...

GPC_A7_MIX_PGC_PUP_STATUS*n* field descriptions (continued)

Field	Description
3 A7_USB_OTG2_PHY_PGC_PUP_STATUS	
2 A7_USB_OTG1_PHY_PGC_PUP_STATUS	
1 A7_PCIE_PHY_PGC_PUP_STATUS	
0 A7_MIPI_PHY_PGC_PUP_STATUS	

5.5.10.37 M4 MIX PGC software up trigger status register (GPC_M4_MIX_PGC_PUP_STATUS*n*)

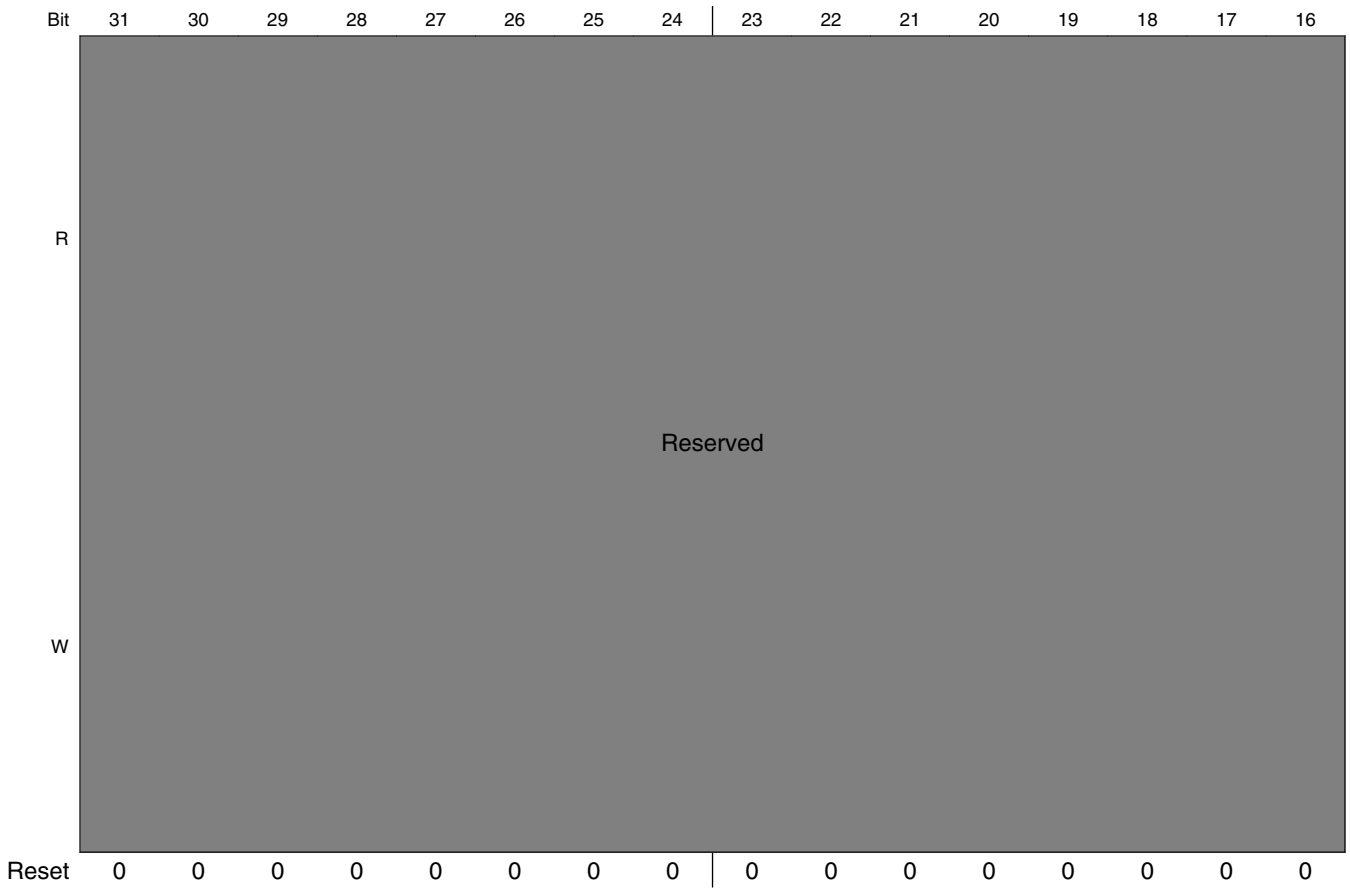
M4_MIX_PGC_PUP_STATUSn (n = 0,1,2) are a read only register, represents the results for power up software trigger from M4 platform to MIX type PGCs.

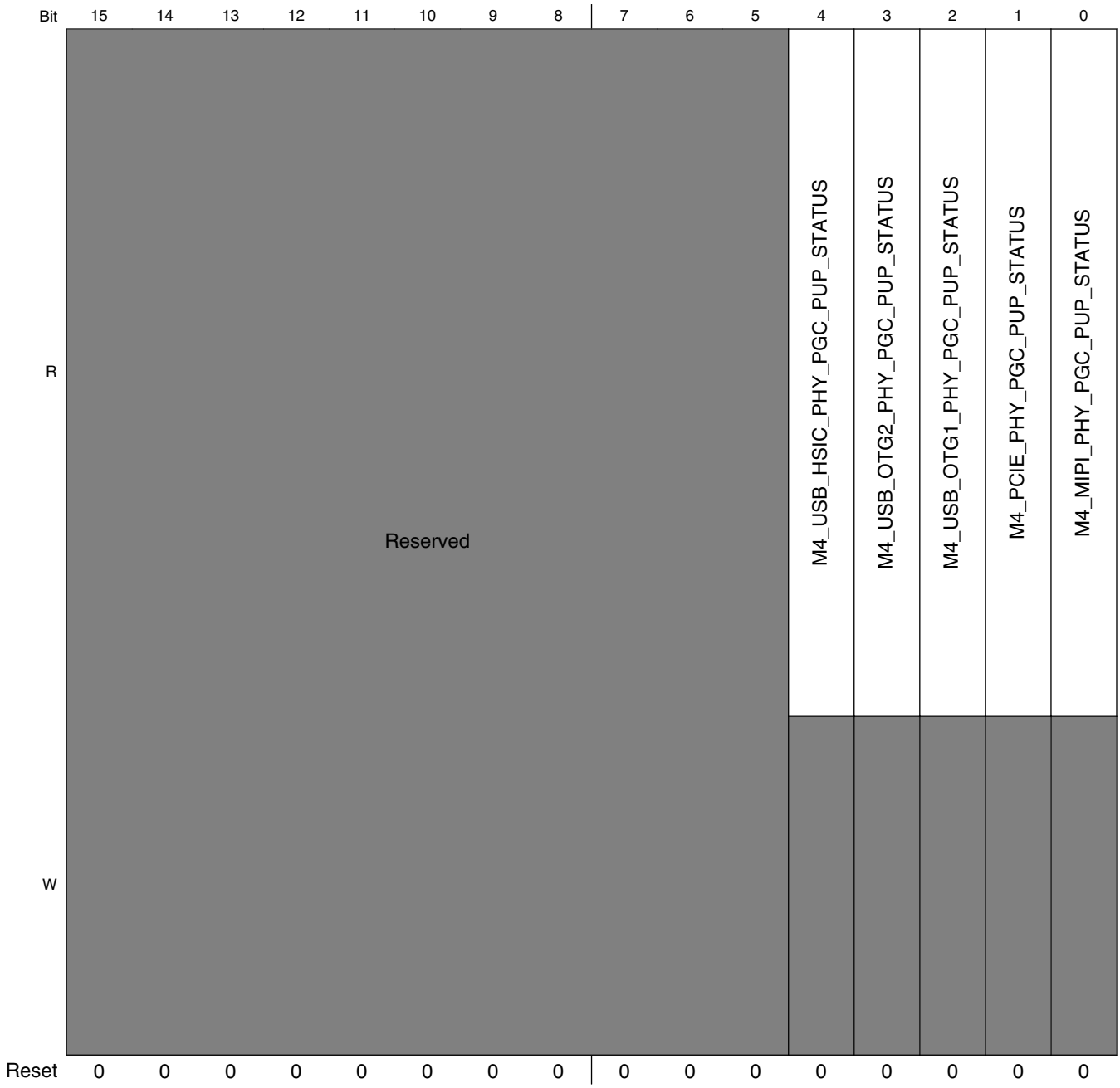
M4_MIX_PGC_PUP_STATUS0: value of “1’b1” represent the software power up trigger failed because domain control condition. The relevant bit will be cleared after a success operation of power up software trigger for MIX type PGCs.

M4_MIX_PGC_PUP_STATUS1: value of “1’b1” represent the software power up trigger failed because the relevant PGC is in a power up process. The relevant bit will be cleared after a success operation of power up software trigger for MIX type PGCs.

M4_MIX_PGC_PUP_STATUS2: value of “1’b1” represent the software power up trigger failed because time slot control is busy. The relevant bit will be cleared after a success operation of power up software trigger for MIX type PGCs.

Address: 303A_0000h base + 140h offset + (4d × i), where i=0d to 2d





GPC_M4_MIX_PGC_PUP_STATUS*n* field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 M4_USB_HSIC_PHY_PGC_PUP_STATUS	

Table continues on the next page...

GPC_M4_MIX_PGC_PUP_STATUS*n* field descriptions (continued)

Field	Description
3 M4_USB_OTG2_PHY_PGC_PUP_STATUS	
2 M4_USB_OTG1_PHY_PGC_PUP_STATUS	
1 M4_PCIE_PHY_PGC_PUP_STATUS	
0 M4_MIPI_PHY_PGC_PUP_STATUS	

5.5.10.38 A7 PU software up trigger status register (GPC_A7_PU_PGC_PUP_STATUS*n*)

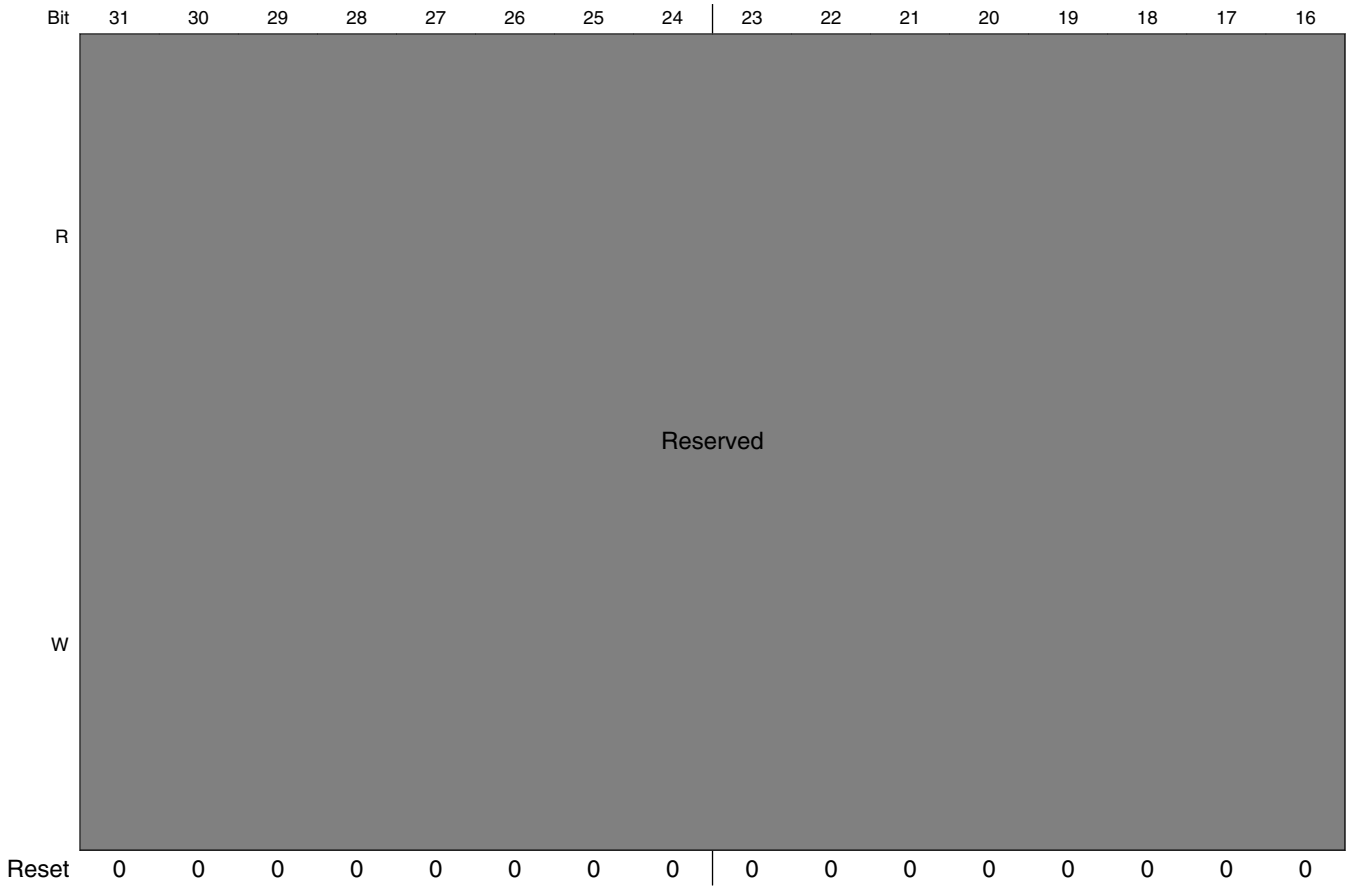
A7_PU_PGC_PUP_STATUSn (n = 0,1,2) are a read only register, represents the results for power up software trigger from A7 platform to PU type PGCs.

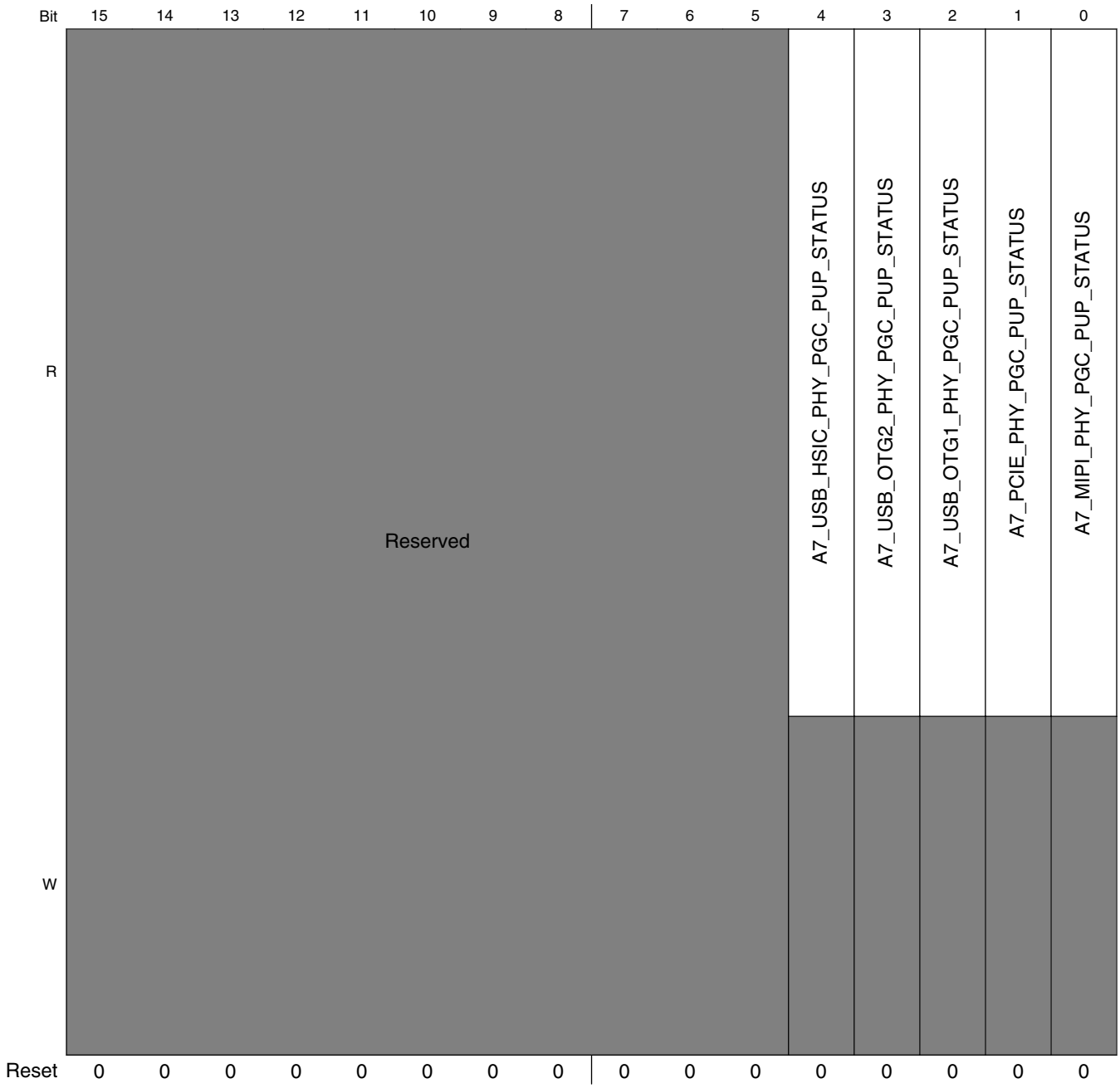
A7_PU_PGC_PUP_STATUS0: value of “1’b1” represent the software power up trigger failed because domain control condition. The relevant bit will be cleared after a success operation of power up software trigger for PU type PGCs.

A7_PU_PGC_PUP_STATUS1: value of “1’b1” represent the software power up trigger failed because the relevant PGC is in a power up process. The relevant bit will be cleared after a success operation of power up software trigger for PU type PGCs.

A7_PU_PGC_PUP_STATUS2: value of “1’b1” represent the software power up trigger failed because time slot control is busy. The relevant bit will be cleared after a success operation of power up software trigger for PU type PGCs.

Address: 303A_0000h base + 14Ch offset + (4d × i), where i=0d to 2d





GPC_A7_PU_PGC_PUP_STATUS*n* field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 A7_USB_HSIC_PHY_PGC_PUP_STATUS	

Table continues on the next page...

GPC_A7_PU_PGC_PUP_STATUS*n* field descriptions (continued)

Field	Description
3 A7_USB_OTG2_PHY_PGC_PUP_STATUS	
2 A7_USB_OTG1_PHY_PGC_PUP_STATUS	
1 A7_PCIE_PHY_PGC_PUP_STATUS	
0 A7_MIPI_PHY_PGC_PUP_STATUS	

5.5.10.39 M4 PU PGC software up trigger status register (GPC_M4_PU_PGC_PUP_STATUS*n*)

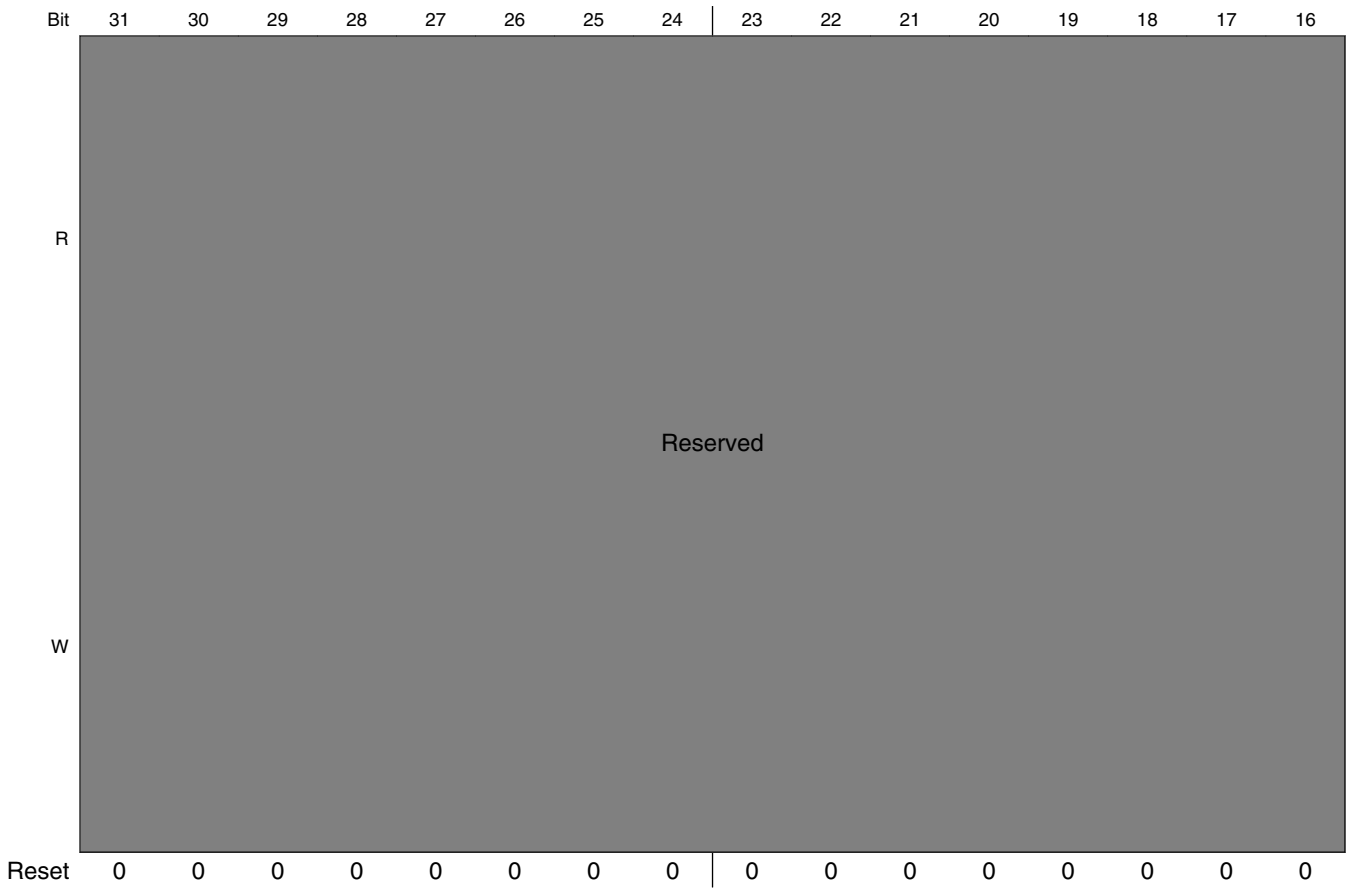
M4_PU_PGC_PUP_STATUSn (n = 0,1,2) are a read only register, represents the results for power up software trigger from M4 platform to PU type PGCs.

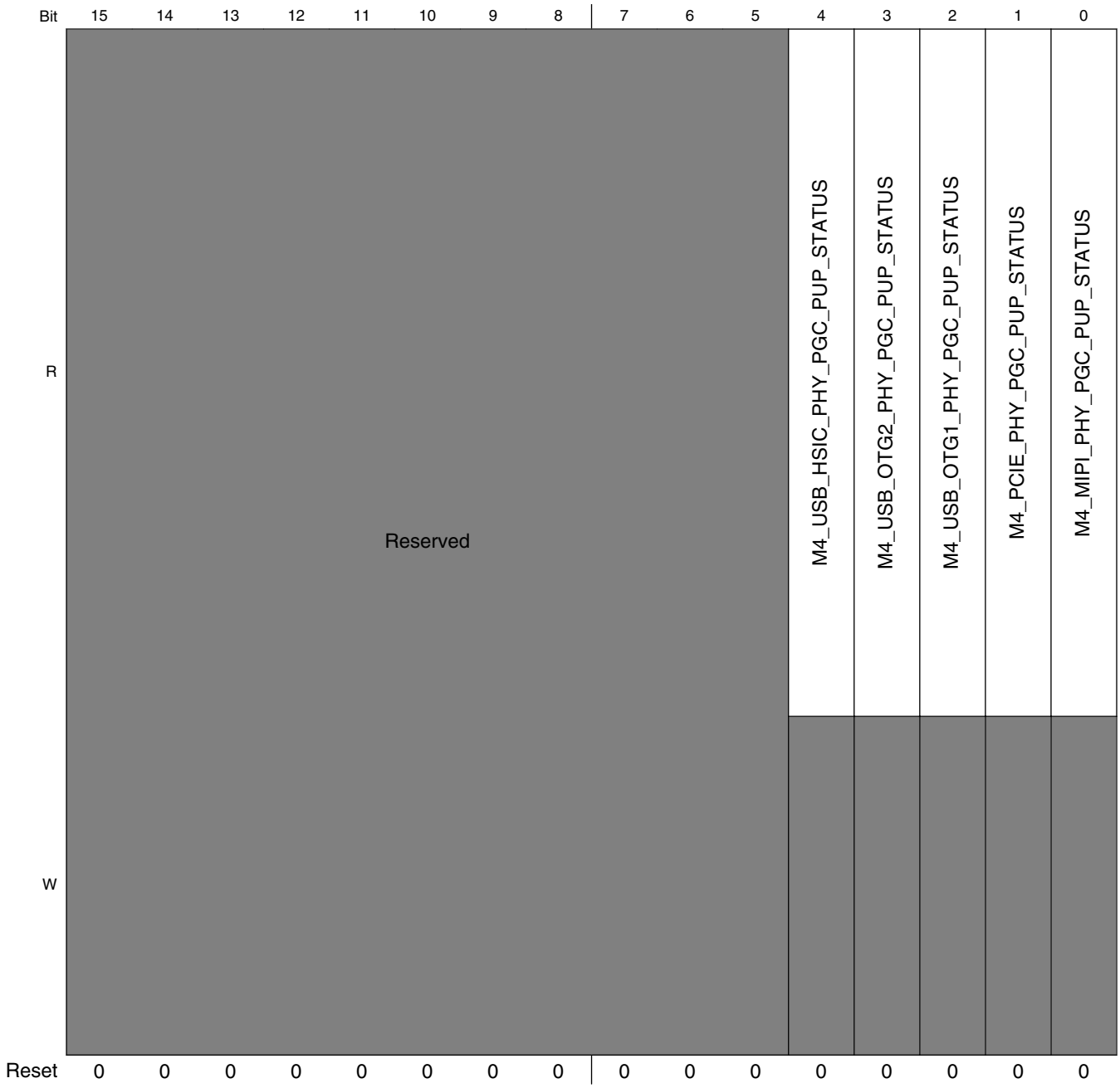
M4_PU_PGC_PUP_STATUS0: value of “1’b1” represent the software power up trigger failed because domain control condition. The relevant bit will be cleared after a success operation of power up software trigger for PU type PGCs.

M4_PU_PGC_PUP_STATUS1: value of “1’b1” represent the software power up trigger failed because the relevant PGC is in a power up process. The relevant bit will be cleared after a success operation of power up software trigger for PU type PGCs.

M4_PU_PGC_PUP_STATUS2: value of “1’b1” represent the software power up trigger failed because time slot control is busy. The relevant bit will be cleared after a success operation of power up software trigger for PU type PGCs.

Address: 303A_0000h base + 158h offset + (4d × i), where i=0d to 2d





GPC_M4_PU_PGC_PUP_STATUS*n* field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 M4_USB_HSIC_PHY_PGC_PUP_STATUS	

Table continues on the next page...

GPC_M4_PU_PGC_PUP_STATUS*n* field descriptions (continued)

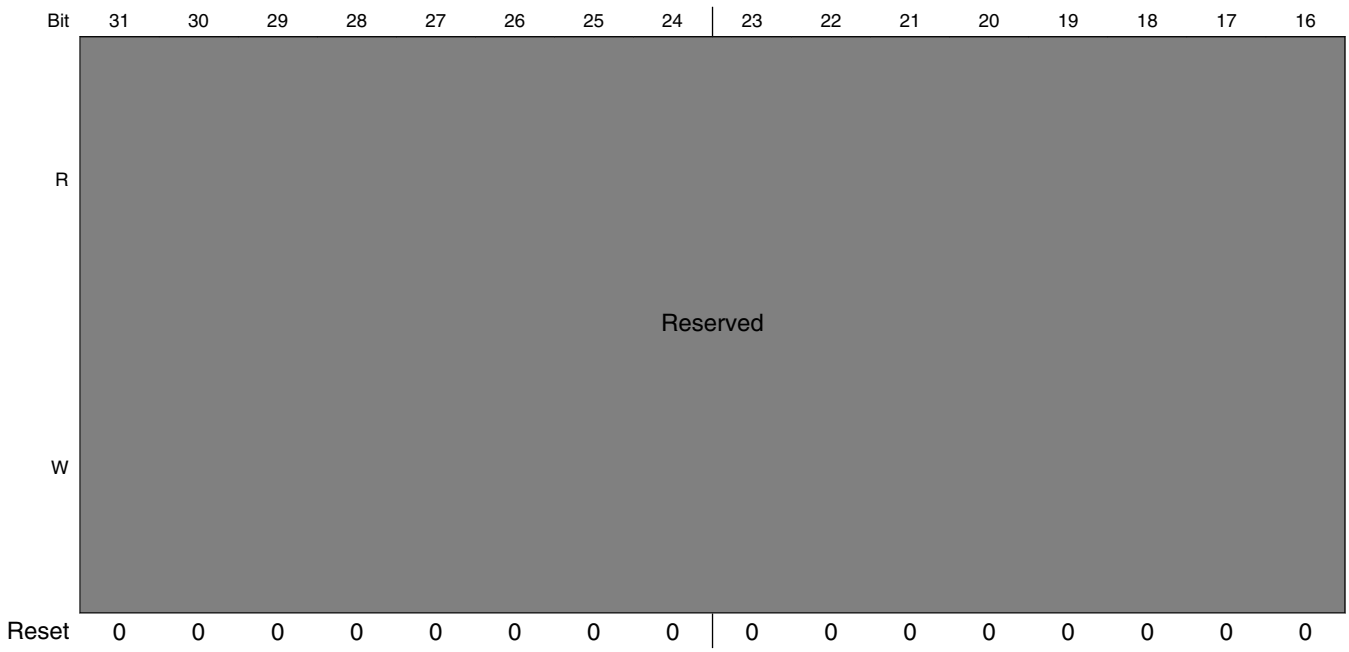
Field	Description
3 M4_USB_OTG2_PHY_PGC_PUP_STATUS	
2 M4_USB_OTG1_PHY_PGC_PUP_STATUS	
1 M4_PCIE_PHY_PGC_PUP_STATUS	
0 M4_MIPI_PHY_PGC_PUP_STATUS	

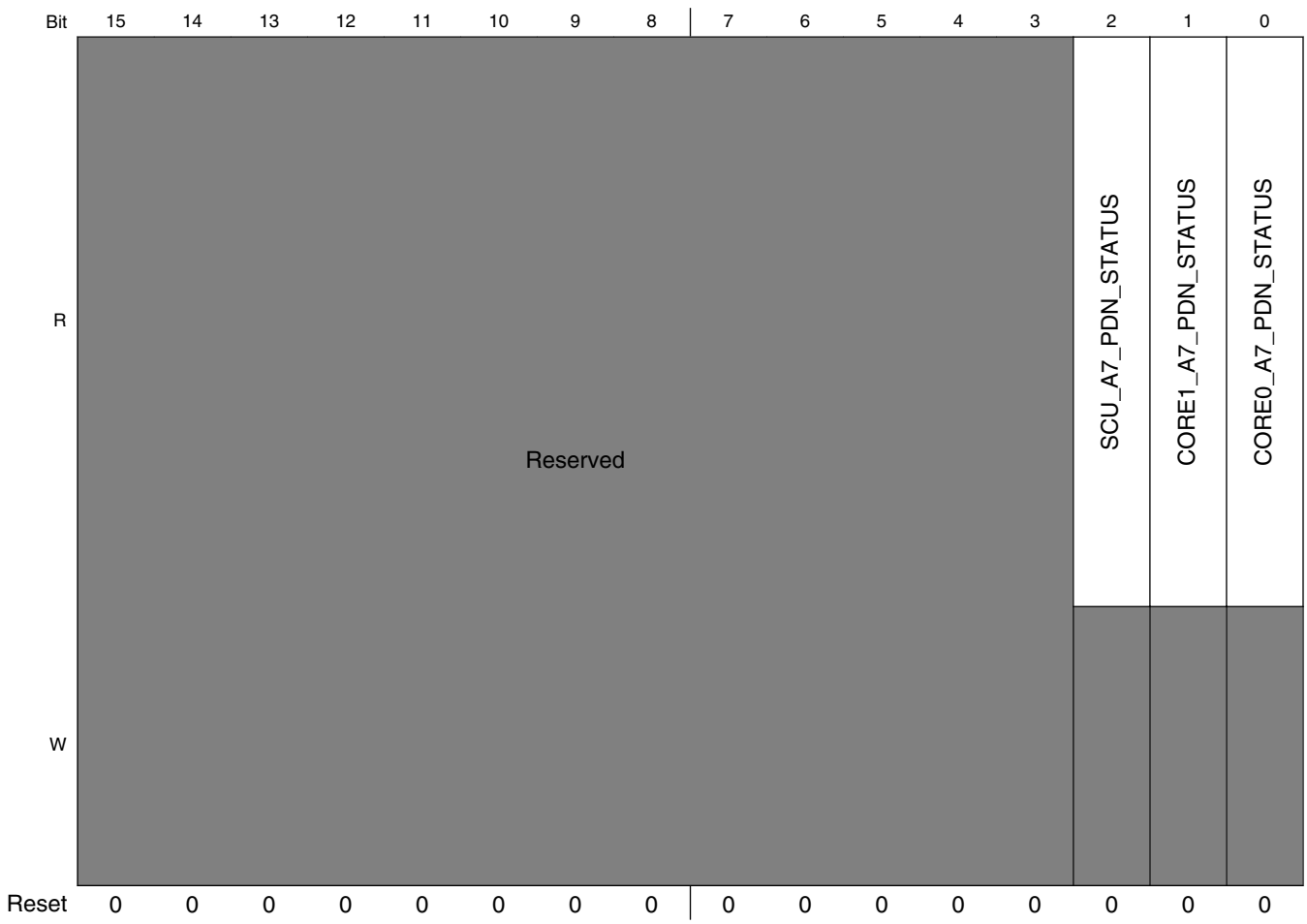
5.5.10.40 CPU PGC software dn trigger status1 (GPC_CPU_PGC_PDN_STATUS1)

CPU_PGC_PDN_STATUS1 is a read only register, represents the results for power DN software trigger for CPU type PGCs.

The field description is show in table below, the value of “1’b1” represent the software power DN trigger failed because the relevant PGC is in a power down process. The relevant bit will be cleared after a success operation of power DN software trigger for CPU type PGCs.

Address: 303A_0000h base + 170h offset = 303A_0170h





GPC_CPU_PGC_PDN_STATUS1 field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
2 SCU_A7_PDN_STATUS	
1 CORE1_A7_PDN_STATUS	
0 CORE0_A7_PDN_STATUS	

5.5.10.41 A7 PU PGC software down trigger status (GPC_A7_PU_PGC_PDN_STATUS*n*)

A7_PU_PGC_PDN_STATUSn (n = 0,1,2) are a read only register, represents the results for power DN software trigger from A7 platform to PU type PGCs.

A7_PU_PGC_PDN_STATUS0: value of “1’b1” represent the software power DN trigger failed because domain control condition. The relevant bit will be cleared after a success operation of power DN software trigger for PU type PGCs.

A7_PU_PGC_PDN_STATUS1: value of “1’b1” represent the software power DN trigger failed because the relevant PGC is in a power DN process. The relevant bit will be cleared after a success operation of power DN software trigger for PU type PGCs.

A7_PU_PGC_PDN_STATUS2: value of “1’b1” represent the software power DN trigger failed because time slot control is busy. The relevant bit will be cleared after a success operation of power DN software trigger for PU type PGCs.

Address: 303A_0000h base + 18Ch offset + (4d × i), where i=0d to 2d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved													SCU_A7_PDN_STATUS	CORE1_A7_PDN_STATUS	CORE0_A7_PDN_STATUS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_A7_PU_PGC_PDN_STATUS*n* field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
2 SCU_A7_PDN_STATUS	
1 CORE1_A7_PDN_STATUS	
0 CORE0_A7_PDN_STATUS	

5.5.10.42 M4 PU PGC software down trigger status (GPC_M4_PU_PGC_PDN_STATUS*n*)

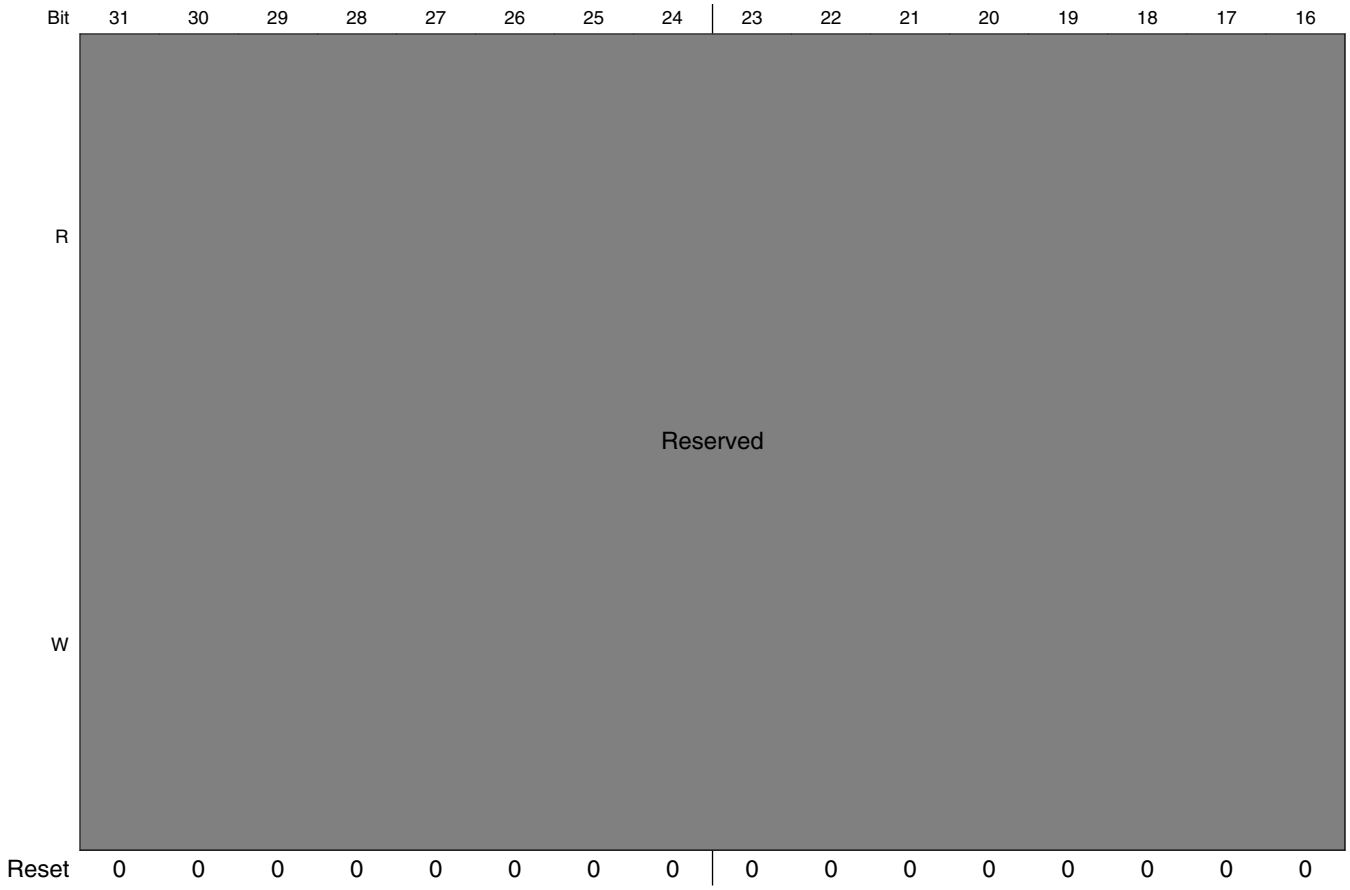
M4_PU_PGC_PDN_STATUSn (n = 0,1,2) are a read only register, represents the results for power DN software trigger from M4 platform to PU type PGCs.

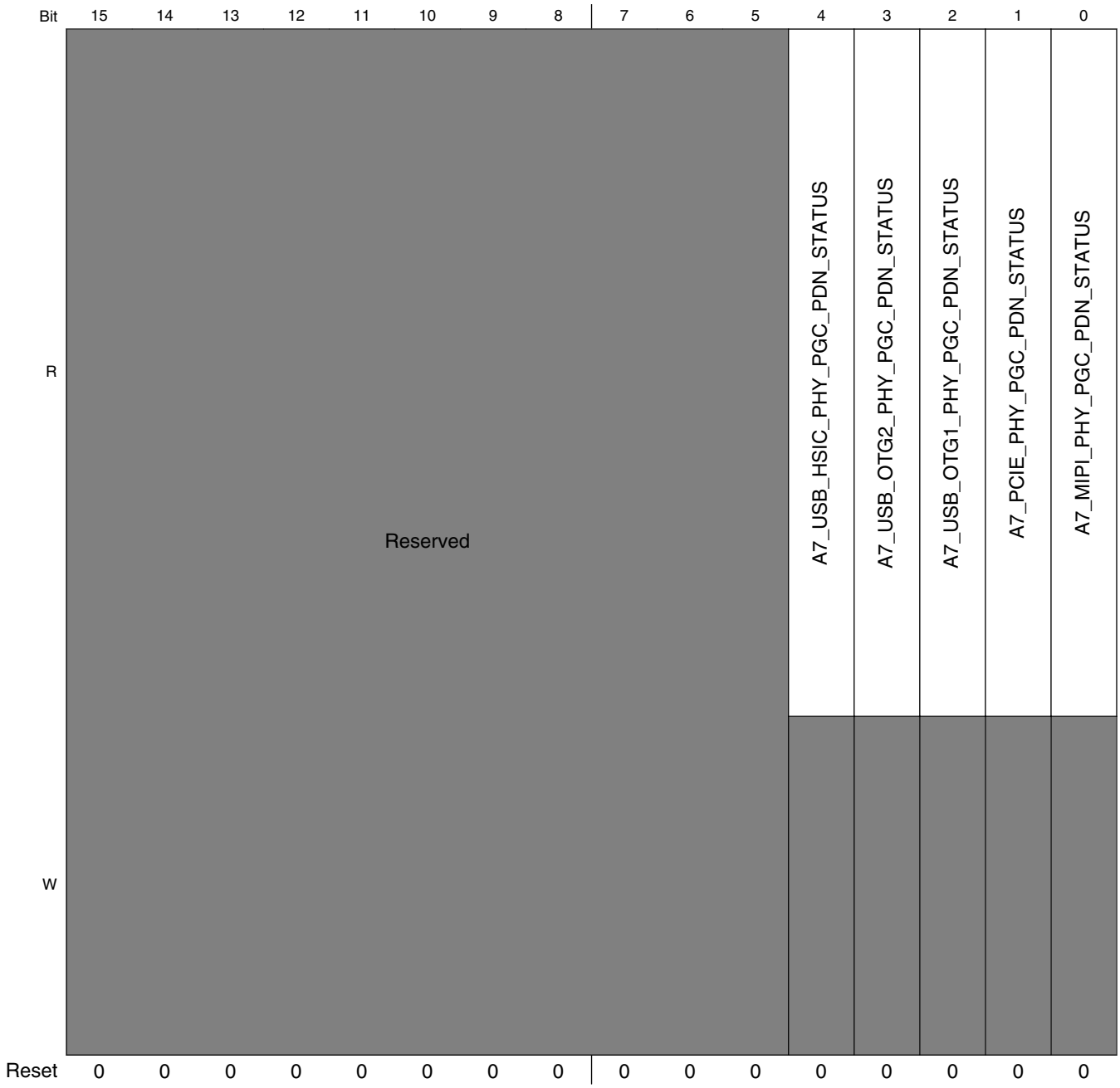
M4_PU_PGC_PDN_STATUS0: value of “1’b1” represent the software power DN trigger failed because domain control condition. The relevant bit will be cleared after a success operation of power DN software trigger for PU type PGCs.

M4_PU_PGC_PDN_STATUS1: value of “1’b1” represent the software power DN trigger failed because the relevant PGC is in a power DN process. The relevant bit will be cleared after a success operation of power DN software trigger for PU type PGCs.

M4_PU_PGC_PDN_STATUS2: value of “1’b1” represent the software power DN trigger failed because time slot control is busy. The relevant bit will be cleared after a success operation of power DN software trigger for PU type PGCs.

Address: 303A_0000h base + 198h offset + (4d × i), where i=0d to 2d





GPC_M4_PU_PGC_PDN_STATUS*n* field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 A7_USB_HSIC_ PHY_PGC_ PDN_STATUS	

Table continues on the next page...

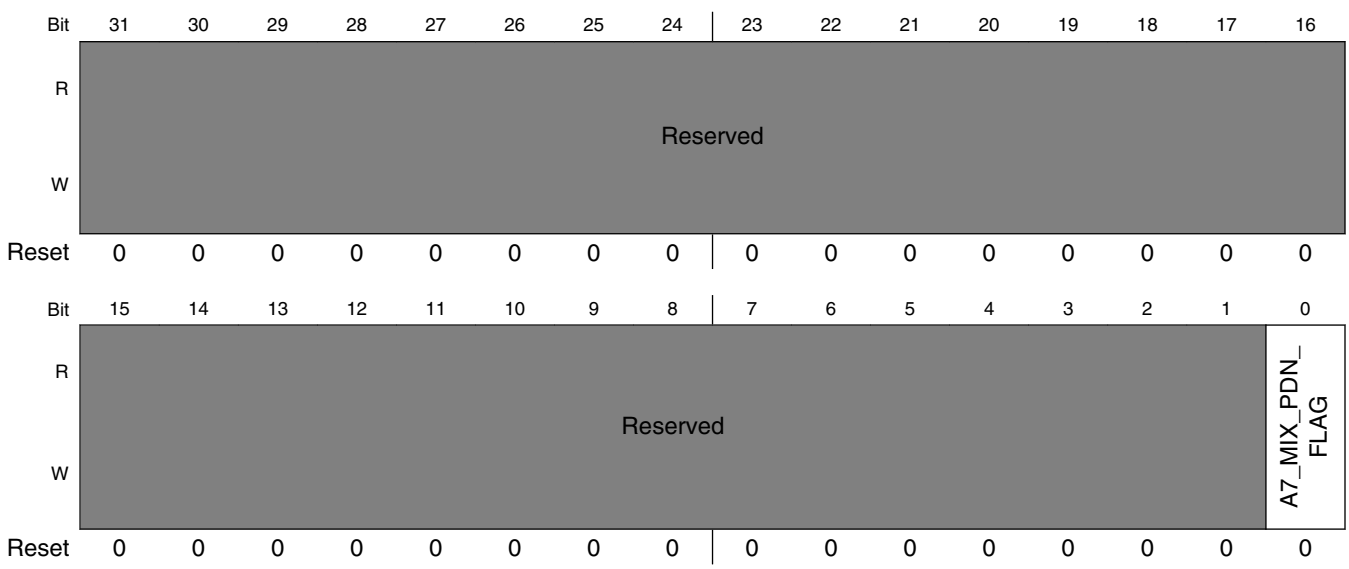
GPC_M4_PU_PGC_PDN_STATUS*n* field descriptions (continued)

Field	Description
3 A7_USB_OTG2_PHY_PGC_PDN_STATUS	
2 A7_USB_OTG1_PHY_PGC_PDN_STATUS	
1 A7_PCIE_PHY_PGC_PDN_STATUS	
0 A7_MIPI_PHY_PGC_PDN_STATUS	

5.5.10.43 A7 MIX PDN FLG (GPC_A7_MIX_PDN_FLG)

This is flag bit relevant domain control, represents A7 CPU platform wants to power down MIX PGC. The register can only be accessed by A7 platform.

Address: 303A_0000h base + 1B0h offset = 303A_01B0h



GPC_A7_MIX_PDN_FLG field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 A7_MIX_PDN_FLAG	A7 MIX power-down flag

5.5.10.44 A7 PU PDN FLG (GPC_A7_PU_PDN_FLG)

The register field is show in the table below. The 1'b1 represents A7 CPU platform wants to power down certain PU PGC. The register is a read only register. The register bits will be set when corresponding A7 software power down trigger happens and will be clear when corresponding A7 software power up trigger happens.

Address: 303A_0000h base + 1B4h offset = 303A_01B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											A7_USB_HSIC_PHY_PGC_PDN_FLG	A7_USB_OTG2_PHY_PGC_PDN_FLG	A7_USB_OTG1_PHY_PGC_PDN_FLG	A7_PCIE_PHY_PGC_PDN_FLG	A7_MIPI_PHY_PGC_PDN_FLG
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_A7_PU_PDN_FLG field descriptions

Field	Description
31–5 -	This field is reserved. Reserved

Table continues on the next page...

GPC_A7_PU_PDN_FLG field descriptions (continued)

Field	Description
4 A7_USB_HSIC_PHY_PGC_PDN_FLG	USB_HSIC PGC power-down flag
3 A7_USB_OTG2_PHY_PGC_PDN_FLG	USB_OTG2 PGC power-down flag
2 A7_USB_OTG1_PHY_PGC_PDN_FLG	USB_OTG1 PGC power-down flag
1 A7_PCIE_PHY_PGC_PDN_FLG	PCIE_PHY PGC power-down flag
0 A7_MIPI_PHY_PGC_PDN_FLG	MIPI_PHY PGC power-down flag

5.5.10.45 M4 MIX PDN FLG (GPC_M4_MIX_PDN_FLG)

This is flag bit relevant domain control, represents M4 CPU platform wants to power down MIX PGC. The register can only be accessed by M4 platform.

Address: 303A_0000h base + 1B8h offset = 303A_01B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															M4_MIX_PDN_FLAG
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_M4_MIX_PDN_FLG field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 M4_MIX_PDN_FLAG	M4_MIX power-down flag

5.5.10.46 M4 PU PDN FLG (GPC_M4_PU_PDN_FLG)

The register field is show in the table below. The 1'b1 represents M4 CPU platform wants to power down certain PU PGC. The register is a read only register. The register bits will be set when corresponding M4 software power down trigger happens and will be clear when corresponding M4 software power up trigger happens.

Address: 303A_0000h base + 1BCh offset = 303A_01BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											M4_USB_HSIC_PHY_PGC_PDN_FLG	M4_USB_OTG2_PHY_PGC_PDN_FLG	M4_USB_OTG1_PHY_PGC_PDN_FLG	M4_PCIE_PHY_PGC_PDN_FLG	M4_MIPI_PHY_PGC_PDN_FLG
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_M4_PU_PDN_FLG field descriptions

Field	Description
31–5 -	This field is reserved. Reserved

Table continues on the next page...

GPC_M4_PU_PDN_FLG field descriptions (continued)

Field	Description
4 M4_USB_HSIC_PHY_PGC_PDN_FLG	USB_HSIC PGC power-down flag
3 M4_USB_OTG2_PHY_PGC_PDN_FLG	USB_OTG2 PGC power-down flag
2 M4_USB_OTG1_PHY_PGC_PDN_FLG	USB_OTG1 PGC power-down flag
1 M4_PCIE_PHY_PGC_PDN_FLG	PCIE_PHY PGC power-down flag
0 M4_MIPI_PHY_PGC_PDN_FLG	MIPI_PHY PGC power-down flag

5.5.11 GPC PGC Memory Map/Register Definition

There are 9 PGC inside GPCv2 with 4 different type : CPU/SCU/MIX/PU. PCIE/MIPI/USB OTGx/USB HSIC PGC belongs to PU type PGC. Each type PGC has 4 different control words PGC_CTRL,PGC_PUPSCR,PGC_PDNSCR and PGC_SR. Different types PGC may have different field definition in these four registers. There is another extra control word PGC_AUXSW which has different field definition for PCIE/MIPI PGC and SCU type PGC.

NOTE

PGCs at offset 0xC80 - 0xCFF are Reserved. Don't use them to power up/down.

The total GPC memory map is 4KB

Table 5-22. Memory Regions

Address Range(offset)	Region
0x000 - 0x3FF	GPC configuration register
0x400 - 0x7FF	Reserved
0x800 - 0x9FF	CPU and SCU type PGC register base address
0xA00 - 0xBFF	MIX type PGC register base address
0xC00 - 0xDFF	PU type PGC register base address
0xE00 - 0xFFF	Reserved

Each PGC (CPU type, MIX type, PU type) will occupy 64 Bytes address space, the specific base address of each PGC are listed as below.

- 0x800 ~ 0x83F : PGC for A7 core0
- 0x840 ~ 0x87F: PGC for A7 core1
- 0x880 ~ 0x8BF: PGC for A7 SCU
- 0xA00 ~ 0xA3F: PGC for fastmix/megamix
- 0xC00 ~ 0xC3F: PGC for MIPI PHY
- 0xC40 ~ 0xC7F: PGC for PCIE_PHY
- 0xC80 ~ 0xCBF: Reserved
- 0xCC0 ~ 0xCFF: Reserved
- 0xD00 ~ 0xD3F: PGC for USB HSIC PHY

GPC_PGC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303A_0800	GPC PGC Control Register (GPC_PGC_A7CORE0_CTRL)	32	R/W	0604_0202h	5.5.11.1/ 890
303A_0804	GPC PGC Up Sequence Control Register (GPC_PGC_A7CORE0_PUPSCR)	32	R/W	0009_97C1h	5.5.11.2/ 891
303A_0808	GPC PGC Down Sequence Control Register (GPC_PGC_A7CORE0_PDNSCR)	32	R/W	2100_0801h	5.5.11.3/ 892
303A_080C	GPC PGC Status Register (GPC_PGC_A7CORE0_SR)	32	R/W	0000_1000h	5.5.11.4/ 894
303A_0840	GPC PGC Control Register (GPC_PGC_A7CORE1_CTRL)	32	R/W	0604_0202h	5.5.11.1/ 890
303A_0844	GPC PGC Up Sequence Control Register (GPC_PGC_A7CORE1_PUPSCR)	32	R/W	0009_97C1h	5.5.11.2/ 891
303A_0848	GPC PGC Down Sequence Control Register (GPC_PGC_A7CORE1_PDNSCR)	32	R/W	2100_0801h	5.5.11.3/ 892
303A_084C	GPC PGC Status Register (GPC_PGC_A7CORE1_SR)	32	R/W	0000_1000h	5.5.11.4/ 894
303A_0880	GPC PGC Control Register (GPC_PGC_A7SCU_CTRL)	32	R/W	0604_0202h	5.5.11.1/ 890
303A_0884	GPC PGC Up Sequence Control Register (GPC_PGC_A7SCU_PUPSCR)	32	R/W	0009_97C1h	5.5.11.2/ 891
303A_0888	GPC PGC Down Sequence Control Register (GPC_PGC_A7SCU_PDNSCR)	32	R/W	2100_0801h	5.5.11.3/ 892
303A_088C	GPC PGC Status Register (GPC_PGC_A7SCU_SR)	32	R/W	0000_1000h	5.5.11.4/ 894
303A_0890	GPC PGC Auxiliary Power Switch SCU Control Register (GPC_PGC_SCU_AUXSW)	32	R/W	0020_240Bh	5.5.11.5/ 897
303A_08C0	GPC PGC Control Register (GPC_PGC_MIX_CTRL)	32	R/W	0604_0202h	5.5.11.1/ 890

Table continues on the next page...

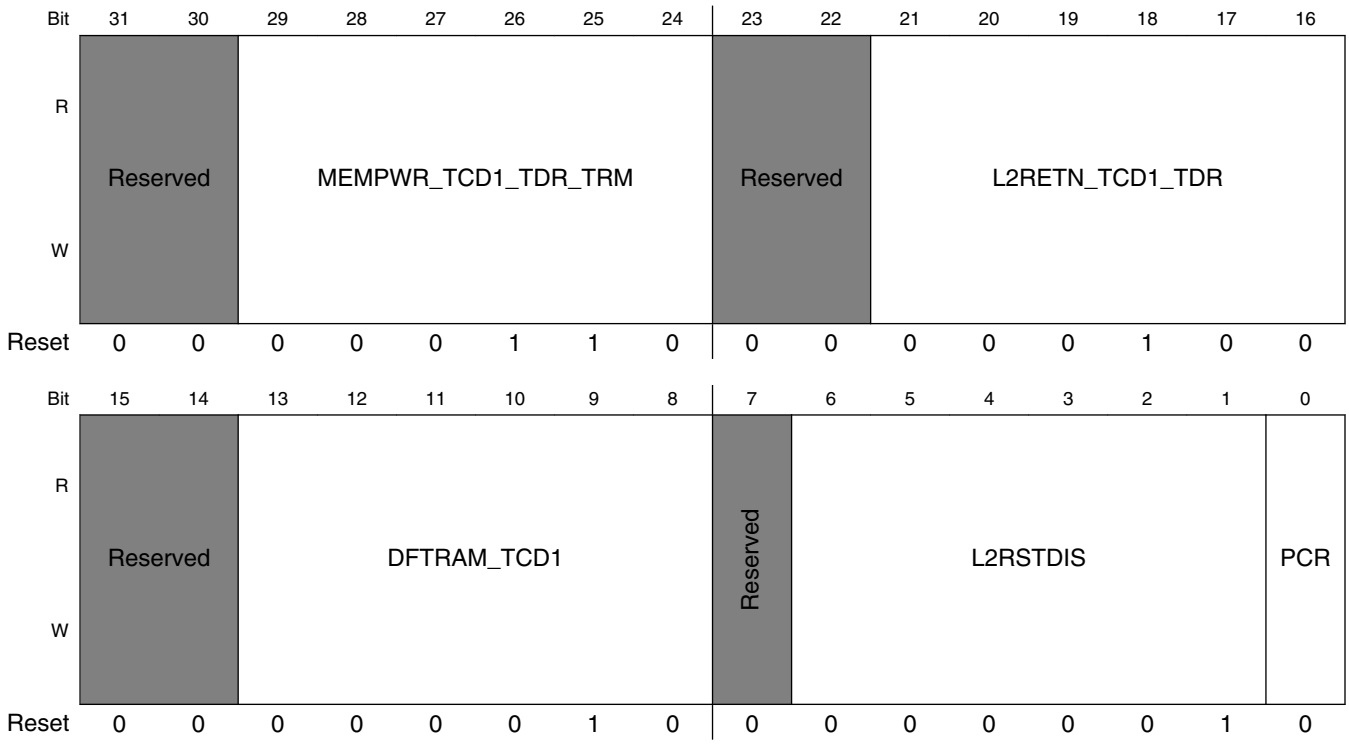
GPC_PGC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
303A_08C4	GPC PGC Up Sequence Control Register (GPC_PGC_MIX_PUPSCR)	32	R/W	0009_97C1h	5.5.11.2/ 891
303A_08C8	GPC PGC Down Sequence Control Register (GPC_PGC_MIX_PDNSCR)	32	R/W	2100_0801h	5.5.11.3/ 892
303A_08CC	GPC PGC Status Register (GPC_PGC_MIX_SR)	32	R/W	0000_1000h	5.5.11.4/ 894
303A_0900	GPC PGC Control Register (GPC_PGC_MIPI_CTRL)	32	R/W	0604_0202h	5.5.11.1/ 890
303A_0904	GPC PGC Up Sequence Control Register (GPC_PGC_MIPI_PUPSCR)	32	R/W	0009_97C1h	5.5.11.2/ 891
303A_0908	GPC PGC Down Sequence Control Register (GPC_PGC_MIPI_PDNSCR)	32	R/W	2100_0801h	5.5.11.3/ 892
303A_090C	GPC PGC Status Register (GPC_PGC_MIPI_SR)	32	R/W	0000_1000h	5.5.11.4/ 894
303A_0940	GPC PGC Control Register (GPC_PGC_PCIE_CTRL)	32	R/W	0604_0202h	5.5.11.1/ 890
303A_0944	GPC PGC Up Sequence Control Register (GPC_PGC_PCIE_PUPSCR)	32	R/W	0009_97C1h	5.5.11.2/ 891
303A_0948	GPC PGC Down Sequence Control Register (GPC_PGC_PCIE_PDNSCR)	32	R/W	2100_0801h	5.5.11.3/ 892
303A_094C	GPC PGC Status Register (GPC_PGC_PCIE_SR)	32	R/W	0000_1000h	5.5.11.4/ 894
303A_0C10	GPC PGC Auxiliary Power Switch Control Register (GPC_PGC_MIPI_AUXSW)	32	R/W	0000_0131h	5.5.11.6/ 898
303A_0C50	GPC PGC Auxiliary Power Switch Control Register (GPC_PGC_PCIE_AUXSW)	32	R/W	0000_0131h	5.5.11.6/ 898
303A_0D00	GPC PGC Control Register (GPC_PGC_HSIC_CTRL)	32	R/W	0604_0202h	5.5.11.7/ 899
303A_0D04	GPC PGC Up Sequence Control Register (GPC_PGC_HSIC_PUPSCR)	32	R/W	0009_97C1h	5.5.11.8/ 901
303A_0D08	GPC PGC Down Sequence Control Register (GPC_PGC_HSIC_PDNSCR)	32	R/W	2100_0801h	5.5.11.9/ 902
303A_0D0C	GPC PGC Status Register (GPC_PGC_HSIC_SR)	32	R/W	0000_1000h	5.5.11.10/ 904

5.5.11.1 GPC PGC Control Register (GPC_PGC_*n*CTRL)

GPC PGC Control Register

Address: 303A_0000h base + 800h offset + (64d × i), where i=0d to 5d



GPC_PGC_*n*CTRL field descriptions

Field	Description
31–30 -	This field is reserved. Reserved
29–24 MEMPWR_ TCD1_TDR_ TRM	After scu pdn_req, count this value to assert A7 mempwr to 1'b1 **NOTE:** Can't be programmed to zero (This register control only for SCU Type PGC)
23–22 -	This field is reserved. Reserved
21–16 L2RETN_TCD1_ TDR	After scu pdn_req, count this value to assert A7 l2retn to 1'b0 **NOTE:** Can't be programmed to zero (This register control only for SCU Type PGC)
15–14 -	This field is reserved. Reserved
13–8 DFTRAM_TCD1	After scu pdn_req, count this value to assert A7 dftram to 1'b1

Table continues on the next page...

GPC_PGC_*n*CTRL field descriptions (continued)

Field	Description
	NOTE: Can't be programmed to zero (This register control only for SCU Type PGC)
7 -	This field is reserved. Reserved
6–1 L2RSTDIS	After scu pdn_req, count this value to assert A7 l2rstdis to 1'b1, it will be clear automatically once any of A7 core0/core1 is wakeup **NOTE:** Can't be programmed to zero (This register control only for SCU Type PGC)
0 PCR	Power Control **NOTE:** PCR must not change from power-down request (pdn_req) assertion until the target subsystem is completely powered up. 0 Do not switch off power even if pdn_req is asserted. 1 Switch off power when pdn_req is asserted.

5.5.11.2 GPC PGC Up Sequence Control Register (GPC_PGC_*n*PUPSCR)

GPC PGC Up Sequence Control Register

Address: 303A_0000h base + 804h offset + (64d × i), where i=0d to 5d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PUP_SCALL_SCALLOUT_CNT									SW2ISO						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SW2ISO								PUP_WAIT_SCALL_OUT	SW						
Reset	1	0	0	1	0	1	1	1	1	1	0	0	0	0	0	1

GPC_PGC_*n*PUPSCR field descriptions

Field	Description
31–23 PUP_SCALL_ SCALLOUT_CNT	After SCALL asserting to 1'b0, count this value to complete switch power up **NOTE:** Only valid when pup_wait_scall_out is set to 1'b0. Can't be programmed to zero (This register control only for MIX Type PGC)
22–7 SW2ISO	After asserting switch_b, the PGC waits a number of clocks equal to the value of SW2ISO before negating isolation.
6 PUP_WAIT_ SCALL_OUT	After SCALL asserting to 1'b0, wait handshake signal SCALL_OUT to return to 1'b0 (This register control only for MIX Type PGC)
SW	After a power-up request (pup_req assertion), the PGC waits a number of clocks equal to the value of SW before asserting switch_b **NOTE:** SW must not be programmed to zero.

5.5.11.3 GPC PGC Down Sequence Control Register (GPC_PGC_*n*PDNSCR)

GPC PGC Down Sequence Control Register

Address: 303A_0000h base + 808h offset + (64d × i), where i=0d to 5d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PUP_SCPRE_SCALL_CNT								PDN_SCALL_SCALLOUT_CNT							
Reset	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		ISO2SW						PDN_WAIT_SCALL_OUT	Reserved	ISO					
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1

GPC_PGC_*n*PDNSCR field descriptions

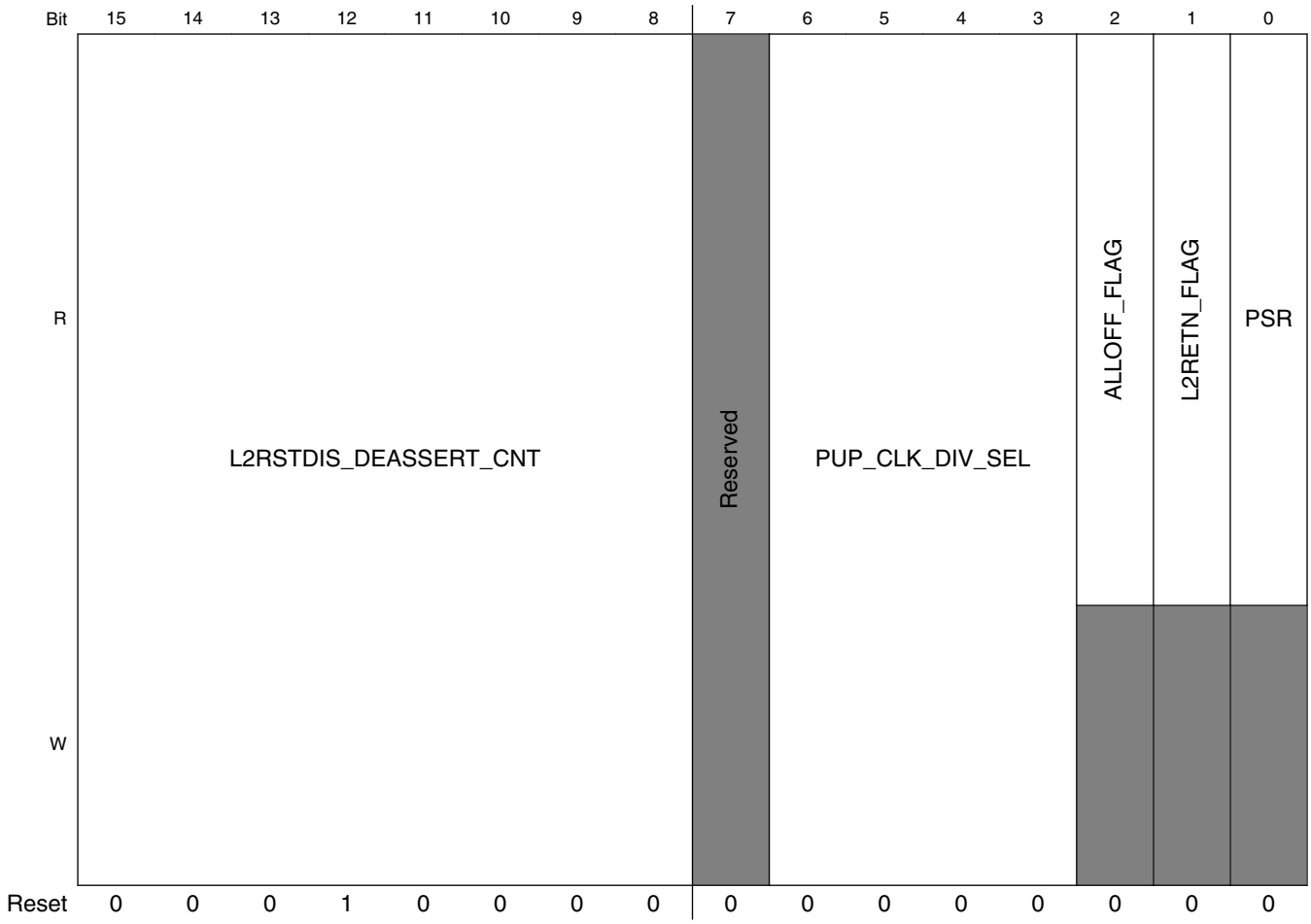
Field	Description
31–24 PUP_SCPRE_ SCALL_CNT	After SCPRE asserting to 1’b0, count this value to assert SCALL to 1’b0 (This register control only for MIX Type PGC)
23–16 PDN_SCALL_ SCALLOUT_CNT	Default 8'h0. After SCALL asserting to 1’b1, count this value to complete switch power down. **NOTE:** This control is only valid when pdn_wait_scall_out is set to 1’b0. It can be programmed to zero. (This register control only for MIX Type PGC)
15–14 -	This field is reserved. Reserved
13–8 ISO2SW	After asserting isolation(by pdn_req assertion), the PGC waits a number of clocks equal to the value of ISO2SW before negating switch_b **NOTE:** ISO2SW must not be programmed to zero.
7 PDN_WAIT_ SCALL_OUT	Default 1'b0 If set to 1'b1, after SCALL asserting to 1'b1, wait handshake signal SCALL_OUT to change to 1'b1. **NOTE:** This register control is only for MIX Type PGC.
6 -	This field is reserved. Reserved
ISO	Aafte a power-down request (pdn_req assertion), the PGC waits a number of clocks equal to the value of ISO before asserting isolation **NOTE:** ISO must not be programmed to zero.

5.5.11.4 GPC PGC Status Register (GPC_PGC_*n*SR)

GPC PGC Status Register

Address: 303A_0000h base + 80Ch offset + (64d × i), where i=0d to 5d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved														L2RSTDIS_DEASSERT_CNT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



GPC_PGC_*n*SR field descriptions

Field	Description
31–18 -	This field is reserved. Reserved
17–8 L2RSTDIS_ DEASSERT_ CNT	Count this value to de-assert L2RSTDISABLE to LOW after CPU0 or CPU1 power up **NOTE:** This value can't be programmed to zero (This register control only for SCU Type PGC)
7 -	This field is reserved. Reserved
6–3 PUP_CLK_DIV_ SEL	Clock divider select for the clock of power up counter(count_clk is 32KHz for CPU/SCU type PGC, ipg_clk(66MHz) for MIX/PU Type PGC) 0000 1 0001 1/2 count_clk 0010 1/4 count_clk 0011 1/8 count_clk 0100 1/16 count_clk 0101 1/32 count_clk 0110 1/64 count_clk 0111 1/128 count_clk

Table continues on the next page...

GPC_PGC_*n*SR field descriptions (continued)

Field	Description
	1000 1/256 count_clk 1001 1/512 count_clk 1010 1/1024 count_clk 1011 1/2056 count_clk 1100 1/4096 count_clk 1101 1/8192 count_clk 1110 1/16384 count_clk 1111 1/32768 count_clk
2 ALLOFF_FLAG	All-off flag. **NOTE:** Software should write “1” to clear this flag after A7 is wakeup from ALL_OFF mode, otherwise, it will always keep to 1 (This register control only for SCU Type PGC) 0 A7 is not wakeup from ALL_OFF mode. 1 A7 is wakeup from ALL_OFF mode.
1 L2RETN_FLAG	L2 Retention Flag **NOTE:** Software should write “1” to clear this flag after A7 is wakeup from L2 retention mode, otherwise it will always keep to 1 (This register control only for SCU Type PGC) 0 A7 is not wakeup from L2 retention mode. 1 A7 is wakeup from L2 retention mode.
0 PSR	Power status. When in functional (or software-controlled debug) mode, PGC hardware sets PSR as soon as any of the power control output changes its state to one. Write one to clear this bit. Software should clear this bit after power up; otherwise, PSR continues to reflect the power status of the initial power down. 0 The target subsystem was not powered down for the previous power-down request. 1 The target subsystem was powered down for the previous power-down request.

5.5.11.5 GPC PGC Auxiliary Power Switch SCU Control Register (GPC_PGC_SCU_AUXSW)

GPC PGC Auxiliary Power Switch Control Register. The register has different field definition for PCIE/MIPI PGC and SCU Type PGC.

NOTE

The value of mempwr_tRC1_tMC/ l2retn_tRC1_tMC_tMR/ dftram_tRC1_tMC_tMR_tCD2 should be programmed to 0x51/0x59/0x5B. That's required for A7 SCU power up timing.

Address: 303A_0000h base + 890h offset = 303A_0890h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		MEMPWR_TRC1_TMC										L2RETN_TRC1_TMC_TMR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	L2RETN_TRC1_TMC_TMR						DFTRAM_TRC1_TMC_TMR_TCD2									
W																
Reset	0	0	1	0	0	1	0	0	0	0	0	0	1	0	1	1

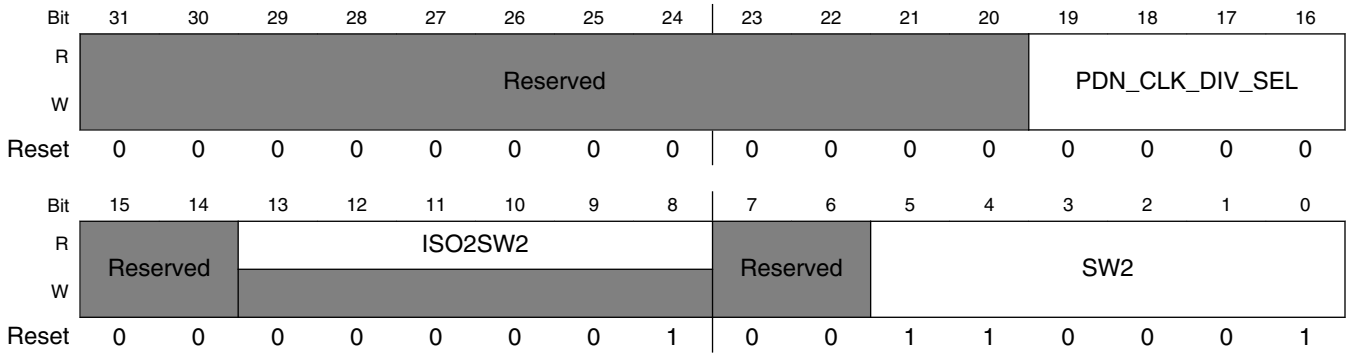
GPC_PGC_SCU_AUXSW field descriptions

Field	Description
31–30 -	This field is reserved. Reserved
29–20 MEMPWR_ TRC1_TMC	After scu start pup reset, count this value to assert A7 mempwr to 1'b0 **NOTE:** Cannot be programmed to zero. 0 A7 is not wakeup from ALL_OFF mode. 1 A7 is wakeup from ALL_OFF mode.
19–10 L2RETN_TRC1_ TMC_TMR	After scu start pup reset, count this value to assert A7 l2retn to 1'b1 **NOTE:** Cannot be programmed to zero.
DFTRAM_TRC1_ TMC_TMR_ TCD2	After scu start pup reset, count this value to assert A7 dftram to 1'b0 **NOTE:** Cannot be programmed to zero.

5.5.11.6 GPC PGC Auxiliary Power Switch Control Register (GPC_PGC_*n*AUXSW)

GPC PGC Auxiliary Power Switch Control Register. The register has different field definition for PCIE/MIPI PGC and SCU Type PGC.

Address: 303A_0000h base + C10h offset + (64d × i), where i=0d to 1d



GPC_PGC_*n*AUXSW field descriptions

Field	Description
31–20 -	This field is reserved. Reserved
19–16 PDN_CLK_DIV_SEL	Clock divider select for the clock of power down counter 0000 1 0001 1/2 count_clk 0010 1/4 count_clk 0011 1/8 count_clk 0100 1/16 count_clk 0101 1/32 count_clk 0110 1/64 count_clk 0111 1/128 count_clk 1000 1/256 count_clk 1001 1/512 count_clk 1010 1/1024 count_clk 1011 1/2056 count_clk 1100 1/4096 count_clk 1101 1/8192 count_clk 1110 1/16384 count_clk 1111 1/32768 count_clk
15–14 -	This field is reserved. Reserved
13–8 ISO2SW2	after asserting isolation by power-down request(pdn_req assertion), the PGC waits a number of clocks equal to the value of ISO2SW2 before negating switch2_b(1P8 Power)

Table continues on the next page...

GPC_PGC_*n*AUXSW field descriptions (continued)

Field	Description
	NOTE: ISO2SW2 must not be programmed to zero. 0 A7 is not wakeup from ALL_OFF mode. 1 A7 is wakeup from ALL_OFF mode.
7–6 -	This field is reserved. Reserved
SW2	After a power-up request (pup_req assertion), the PGC waits a number of clocks equal to the value of SW2 before asserting switch2_b(1P8 Power) **NOTE:** SW2 must not be programmed to zero.

5.5.11.7 GPC PGC Control Register (GPC_PGC_HSIC_CTRL)

GPC PGC Control Register

Address: 303A_0000h base + D00h offset = 303A_0D00h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		MEMPWR_TCD1_TDR_TRM						Reserved		L2RETN_TCD1_TDR					
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		DFTRAM_TCD1						Reserved	L2RSTDIS						PCR
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0

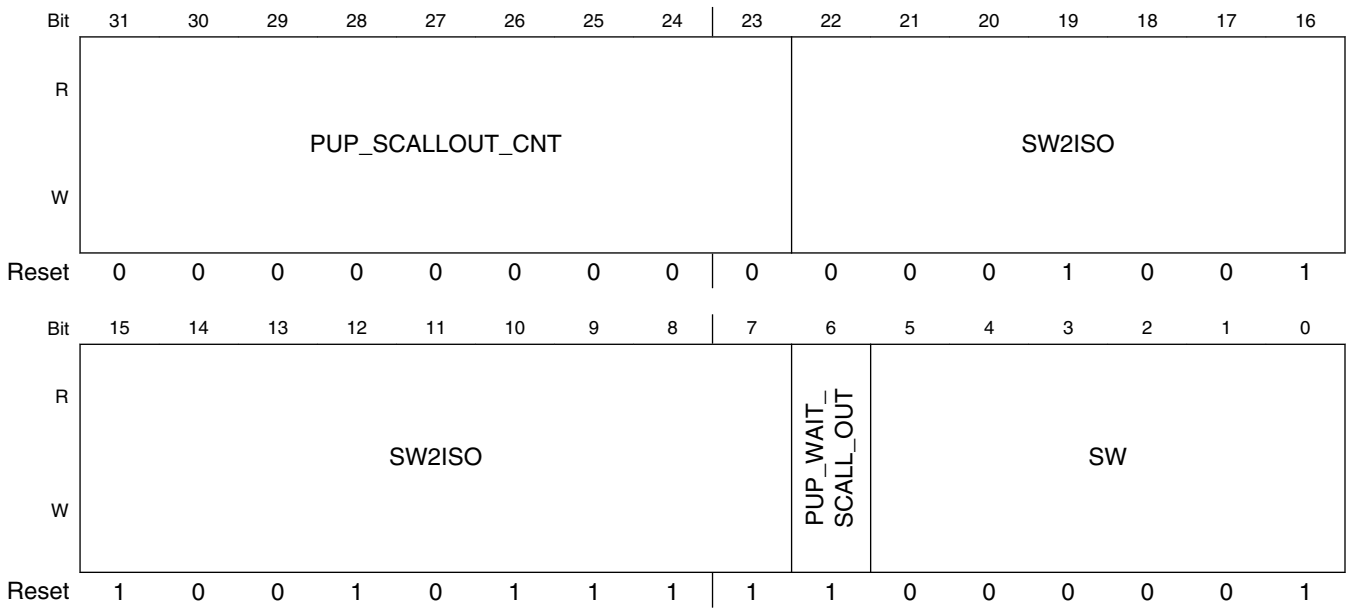
GPC_PGC_HSIC_CTRL field descriptions

Field	Description
31–30 -	This field is reserved. Reserved
29–24 MEMPWR_TCD1_TDR_TRM	After scu pdn_req, count this value to assert A7 mempwr to 1'b1 **NOTE:** Can't be programmed to zero (This register control only for SCU Type PGC)
23–22 -	This field is reserved. Reserved
21–16 L2RETN_TCD1_TDR	After scu pdn_req, count this value to assert A7 l2retn to 1'b0 **NOTE:** Can't be programmed to zero (This register control only for SCU Type PGC)
15–14 -	This field is reserved. Reserved
13–8 DFTRAM_TCD1	After scu pdn_req, count this value to assert A7 dftram to 1'b1 **NOTE:** Can't be programmed to zero (This register control only for SCU Type PGC)
7 -	This field is reserved. Reserved
6–1 L2RSTDIS	After scu pdn_req, count this value to assert A7 l2rstdis to 1'b1, it will be clear automatically once any of A7 core0/core1 is wakeup **NOTE:** Can't be programmed to zero (This register control only for SCU Type PGC)
0 PCR	Power Control **NOTE:** PCR must not change from power-down request (pdn_req) assertion until the target subsystem is completely powered up. 0 Do not switch off power even if pdn_req is asserted. 1 Switch off power when pdn_req is asserted.

5.5.11.8 GPC PGC Up Sequence Control Register (GPC_PGC_HSIC_PUPSCR)

GPC PGC Up Sequence Control Register

Address: 303A_0000h base + D04h offset = 303A_0D04h



GPC_PGC_HSIC_PUPSCR field descriptions

Field	Description
31–23 PUP_ SCALLOUT_CNT	After SCALL asserting to 1'b0, count this value to complete switch power up **NOTE:** Only valid when pup_wait_scall_out is set to 1'b0. Can't be programmed to zero (This register control only for MIX Type PGC)
22–7 SW2ISO	After asserting switch_b, the PGC waits a number of clocks equal to the value of SW2ISO before negating isolation.
6 PUP_WAIT_ SCALL_OUT	After SCALL asserting to 1'b0, wait handshake signal SCALL_OUT to return to 1'b0 (This register control only for MIX Type PGC)
SW	After a power-up request (pup_req assertion), the PGC waits a number of clocks equal to the value of SW before asserting switch_b **NOTE:** SW must not be programmed to zero.

5.5.11.9 GPC PGC Down Sequence Control Register (GPC_PGC_HSIC_PDNSCR)

GPC PGC Down Sequence Control Register

Address: 303A_0000h base + D08h offset = 303A_0D08h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PUP_SCPRE_SCALL_CNT								PUP_SCALLOUT_CNT							
Reset	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		ISO2SW						PUP_WAIT_SCALL_OUT	Reserved	ISO					
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1

GPC_PGC_HSIC_PDNSCR field descriptions

Field	Description
31–24 PUP_SCPRE_SCALL_CNT	After SCPRE asserting to 1'b0, count this value to assert SCALL to 1'b0 (This register control only for MIX Type PGC)
23–16 PUP_SCALLOUT_CNT	After SCALL asserting to 1'b1, count this value to complete switch power down **NOTE:** This control is only valid when pdn_wait_scall_out is set to 1'b0. It can be programmed to zero (This register control only for MIX Type PGC)
15–14 -	This field is reserved. Reserved
13–8 ISO2SW	After asserting isolation(by pdn_req assertion), the PGC waits a number of clocks equal to the value of ISO2SW before negating switch_b **NOTE:** ISO2SW must not be programmed to zero.

Table continues on the next page...

GPC_PGC_HSIC_PDNSCR field descriptions (continued)

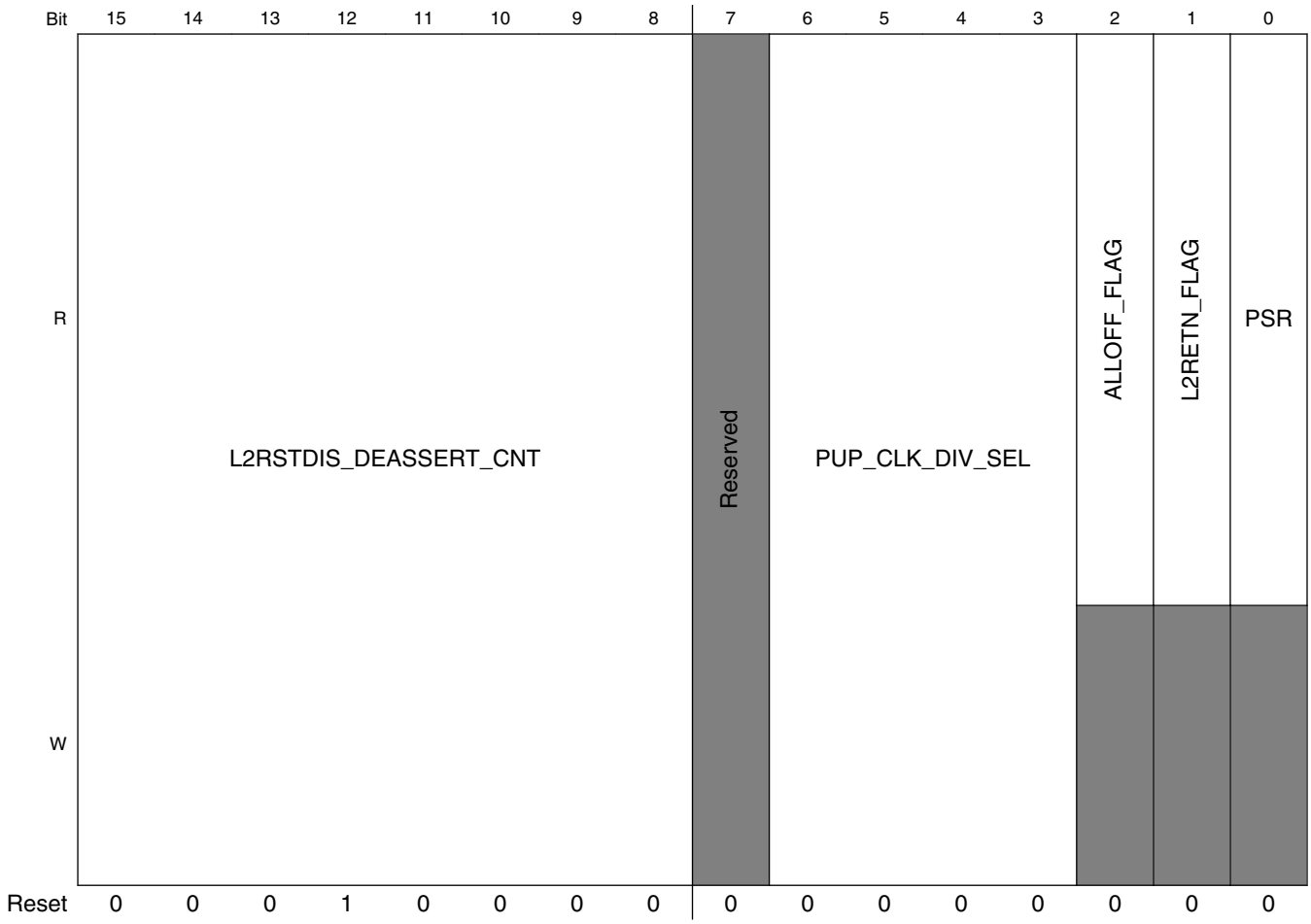
Field	Description
7 PUP_WAIT_ SCALL_OUT	After SCALL asserting to 1'b0, wait handshake signal SCALL_OUT to return to 1'b0 (This register control only for MIX Type PGC)
6 -	This field is reserved. Reserved
ISO	Aafte a power-down request (pdn_req assertion), the PGC waits a number of clocks equal to the value of ISO before asserting isolation **NOTE:** ISO must not be programmed to zero.

5.5.11.10 GPC PGC Status Register (GPC_PGC_HSIC_SR)

GPC PGC Status Register

Address: 303A_0000h base + D0Ch offset = 303A_0D0Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved														L2RSTDIS_DEASSERT_CNT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



GPC_PGC_HSIC_SR field descriptions

Field	Description
31–18 -	This field is reserved. Reserved
17–8 L2RSTDIS_ DEASSERT_ CNT	Count this value to de-assert L2RSTDISABLE to LOW after CPU0 or CPU1 power up **NOTE:** This value can't be programmed to zero (This register control only for SCU Type PGC)
7 -	This field is reserved. Reserved
6–3 PUP_CLK_DIV_ SEL	Clock divider select for the clock of power up counter(count_clk is 32KHz for CPU/SCU type PGC, ipg_clk(66MHz) for MIX/PU Type PGC) 0000 1 0001 1/2 count_clk 0010 1/4 count_clk 0011 1/8 count_clk 0100 1/16 count_clk 0101 1/32 count_clk 0110 1/64 count_clk 0111 1/128 count_clk

Table continues on the next page...

GPC_PGC_HSIC_SR field descriptions (continued)

Field	Description
	1000 1/256 count_clk 1001 1/512 count_clk 1010 1/1024 count_clk 1011 1/2056 count_clk 1100 1/4096 count_clk 1101 1/8192 count_clk 1110 1/16384 count_clk 1111 1/32768 count_clk
2 ALLOFF_FLAG	All-off flag. **NOTE:** Software should write “1” to clear this flag after A7 is wakeup from ALL_OFF mode, otherwise, it will always keep to 1 (This register control only for SCU Type PGC) 0 A7 is not wakeup from ALL_OFF mode. 1 A7 is wakeup from ALL_OFF mode.
1 L2RETN_FLAG	L2 Retention Flag **NOTE:** Software should write “1” to clear this flag after A7 is wakeup from L2 retention mode, otherwise it will always keep to 1 (This register control only for SCU Type PGC) 0 A7 is not wakeup from L2 retention mode. 1 A7 is wakeup from L2 retention mode.
0 PSR	Power status. When in functional (or software-controlled debug) mode, PGC hardware sets PSR as soon as any of the power control output changes its state to one. Write one to clear this bit. Software should clear this bit after power up; otherwise, PSR continues to reflect the power status of the initial power down. 0 The target subsystem was not powered down for the previous power-down request. 1 The target subsystem was powered down for the previous power-down request.

Chapter 6
SNVS, Reset, Fuse and Boot

6.1 Secure Non-Volatile Storage (SNVS)

6.1.1 SNVS overview

The low-power (battery-backed) section incorporates a secure real time counter, a monotonic counter, and a general purpose register. This portion of the block is powered by a battery that maintains the state of the SNVS_LP registers when the chip is powered off.

Figure 6-1. Example SNVS Connectivity

6.1.1.1 SNVS features

The following table summarizes the features:

Table 6-1. SNVS feature summary

Feature	What it does
Non-secure Real time counter (HPRTC)	• The counter is driven by a dedicated clock, which is off when the system power is down • Programmable time alarm interrupt • Periodic interrupt can be generated with different frequencies
Monotonic counter	• The monotonic counter state is nonvolatile. • The counter can only increment. • The counter is a non-rollover counter • The counter value is invalidated in case of security violation.
General-purpose register	• The general-purpose register state is nonvolatile.
Register access protection	• Privileged software access policy • Registers can be programmed only when the system security monitor is in functional state. • Some registers/values can only be programmable once per boot cycle.

6.1.1.2 Modes of operation

The SNVS operates in either the system power-down or system power-up mode of operation.

During system power-down, SNVS_HP is powered-down. SNVS_LP is powered from the backup power supply and is electrically isolated from the rest of the chip. In this mode, SNVS_LP retains the state of its registers .

During system power-up, SNVS_HP and SNVS_LP are both powered-up and all SNVS functions are operational.

6.1.2 SNVS structure

The SNVS block is divided into two major submodules based on power supply: the high power domain (SNVS_HP) and the low power domain (SNVS_LP). They are powered as follows:

- SNVS_LP - dedicated always-powered-on domain
- SNVS_HP - system (chip) power domain

The following figure illustrates the low power and chip power domains of SNVS.

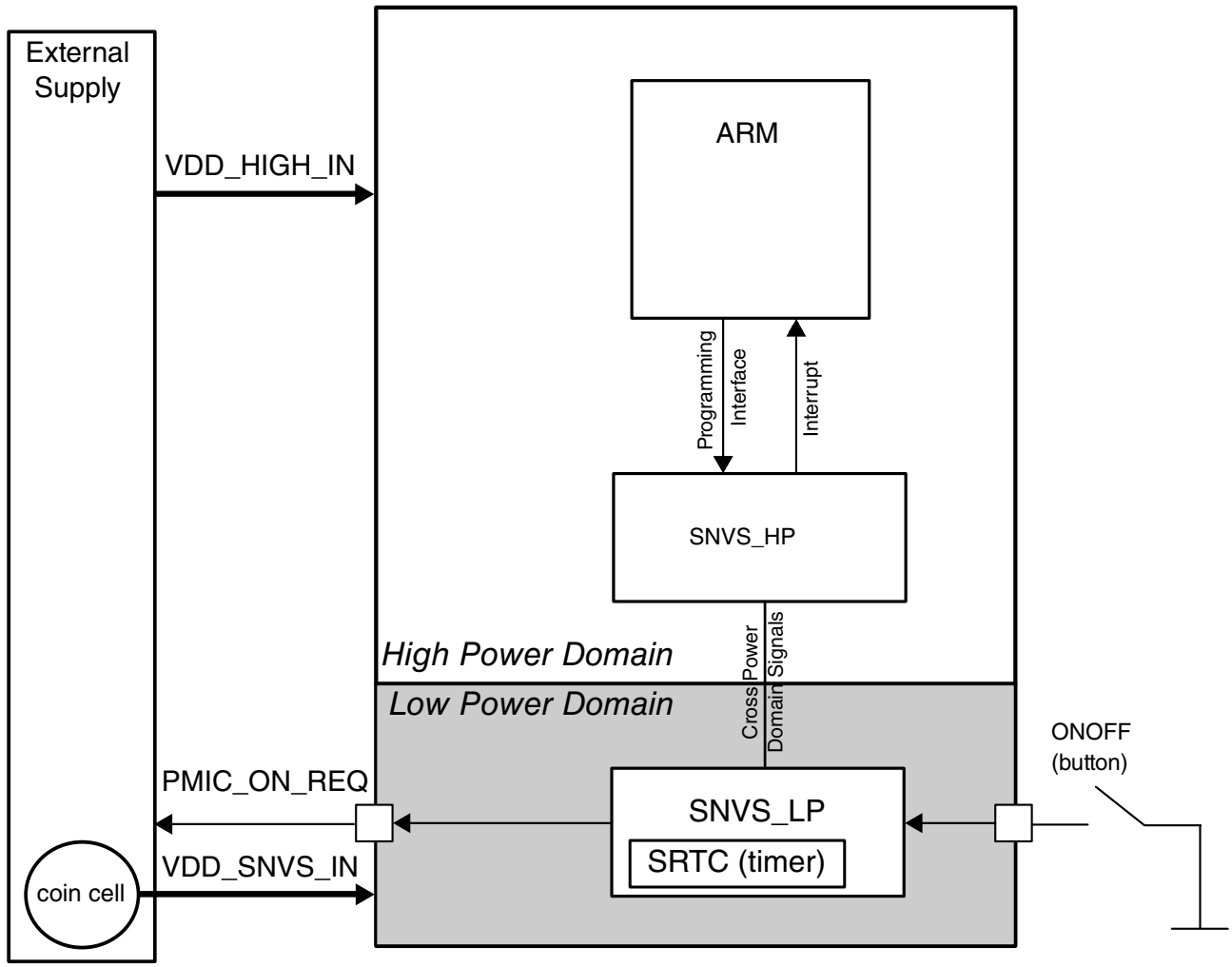


Figure 6-2. SNVS Power Domains

The SNVS_HP section implements all features that enable system communication and provisioning of the SNVS_LP section.

The SNVS_LP section provides hardware that enables secure storage and protection of sensitive data.

6.1.2.1 SNVS_HP (high power domain)

SNVS_HP is partitioned into the following functional units:

- IP bus interface
- SNVS_LP interface
- Real time counter with alarm
- Control and status registers

SNVS_HP is in the chip's power supply domain and thus receives power along with the rest of the chip. SNVS_HP provides an interface between SNVS_LP and the rest of the system; there is no way to access the SNVS_LP registers except through the SNVS_HP. For access to the SNVS_LP registers, SNVS_HP must be powered up. It uses a register access permission policy to determine whether access to particular registers is permitted.

6.1.2.2 Non-secure real time counter

SNVS _HP has an autonomous non-secure real time counter. The counter is not active and is reset when the system is powered down. The HP RTC can be used by any application; it has no privileged software access restrictions. The counter can be synchronized with the SNVS_LP SRTC by writing to a specific bit in the SNVS_HP Control Register.

6.1.2.2.1 Calibrating the time counter

The RTC accuracy may suffer from a drift in the clock, which is used to increment the RTC register. To compensate for this drift, a clock calibration mechanism can adjust the RTC value. It is up to the system processor to decide whether calibration is required or not. If RTC correction is required, enable the mechanism and set the calibration value in the control register. The calibration value is a 5 bit value including the sign bit, which is implemented in 2's complement.

If the calibration mechanism is enabled, the calibration value is added or subtracted from the RTC on a periodic basis, once per 32768 cycles of the RTC clock.

The following table shows the available correction range.

Table 6-2. Time counter calibration settings

Calibration value setting	Correction in counts per 32768 cycles of the counter clock
01111	+15
:	:
00010	+2
00001	+1
00000	0
11111	-1
11110	-2
:	:
10001	-15
10000	-16

6.1.2.2.2 Time counter alarm

The SNVS _HP non-secure RTC has its own time alarm register. Any application can update this register. The SNVS _HP time alarm can generate interrupts to alert the host processor and can wake up the host processor from one of its low-power modes. Note that this alarm cannot wake up the entire system if it is powered off because this alarm would also be powered off.

6.1.2.2.3 Periodic interrupt

The SNVS_HP non-secure RTC incorporates a periodic interrupt. The periodic interrupt is generated when a zero-to-one or one-to-zero transition occurs on the selected bit of the RTC. The periodic interrupt source is chosen from 16 bits of the HP RTC according to the PI_FREQ field setting in the HP Control Register. This bit selection also defines the frequency of the periodic interrupt.

The following figure shows the SNVS_HP RTC and its interrupts.

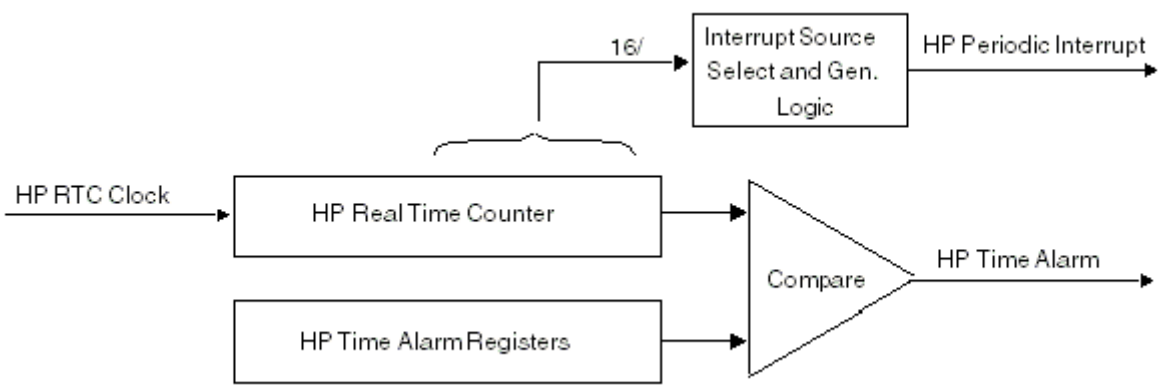


Figure 6-3. SNVS_HP RTC, alarm, and interrupts

6.1.3 SNVS_LP (low power domain)

SNVS_LP has the following functional units:

- Non-rollover monotonic counter
- General purpose register
- Control and status registers

The SNVS_LP is a data storage subsystem. Its purpose is to store and protect system data, regardless of the main system power state.

SNVS _LP is in the always-powered-up domain, which is a separate power domain with its own power supply.

6.1.3.1 Behavior during system power down

When the chip power supply domain loses power, SNVS_LP continues to operate normally, and it ignores all inputs from SNVS_HP.

6.1.3.2 Monotonic counter (MC)

The following figure shows the MC and its rollover security violation.

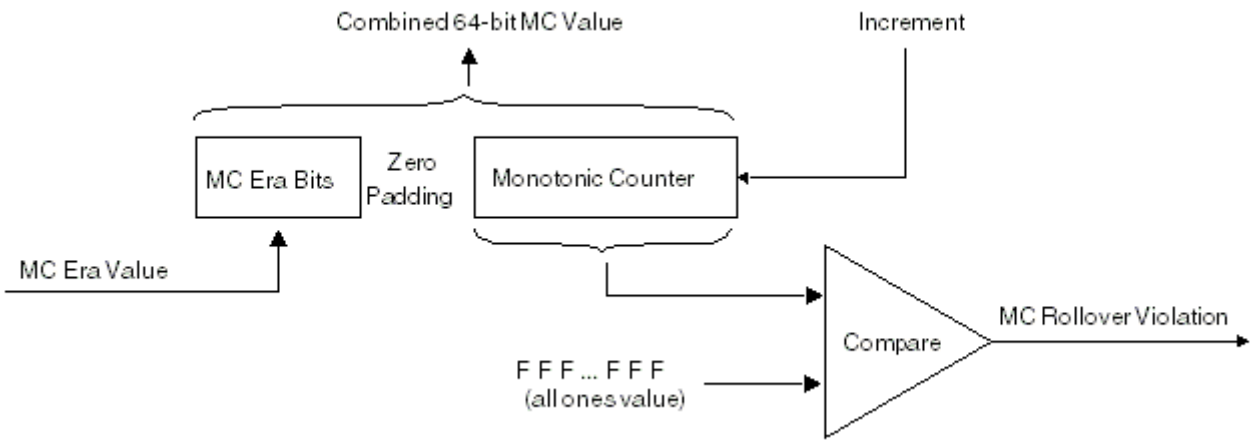


Figure 6-4. SNVS _LP monotonic counter

Some security applications require a monotonic counter (MC) that cannot be exhausted or returned to any previous value during the product's lifetime. Because the MC can never repeat a number, it cannot be reset or cycled back to its starting count. If it reaches its maximum value, it does not rollover. Instead, a monotonic counter rollover indication is generated to the SNVS _LP tamper monitor. This generates an interrupt to the host processor.

The SNVS uses an ERA value derived from the OTP elements as a mechanism for recovery from an MC failure (for example, due to a failure of LP power) where the MC value was compromised or cleared. The ERA value is prepended to the MC to form its most significant bits. Once any of the ERA value bits are set, the MC can count up from any value, including zero. This guarantees that any future value of the combined monotonic counter will be greater than any of its past values.

6.1.4 SNVS reset and system power up

This table describes reset actions for SNVS.

Table 6-3. Reset summary

Reset	Source	Characteristics	Internally resets
HP Hard	ipg_hard_async_reset_b	active-low, asynchronous	All SNVS_HPSNVS_LP registers and flops.
LP Power On Reset (POR)	lp_por_b	active-low, asynchronous	All SNVS_LP registers and flops
LP software Reset	software	active-high, synchronous, 1 cycle	All SNVS_LP registers and flops. LP software Reset can be asserted if not disabled.

6.1.4.1 PMIC Interface

The On/Off logic inside of SNVS_LP allows for connecting directly to a PMIC or other voltage regulator device. The logic takes a button input signal and then outputs a PMIC "ON" Request and a "Power Off" Interrupt. PMIC logic also supports the SNVS_LP tamper logic which will allow waking the system up when a tamper event has happened while in the OFF state. The logic has two different modes of operation (Dumb and Smart mode).

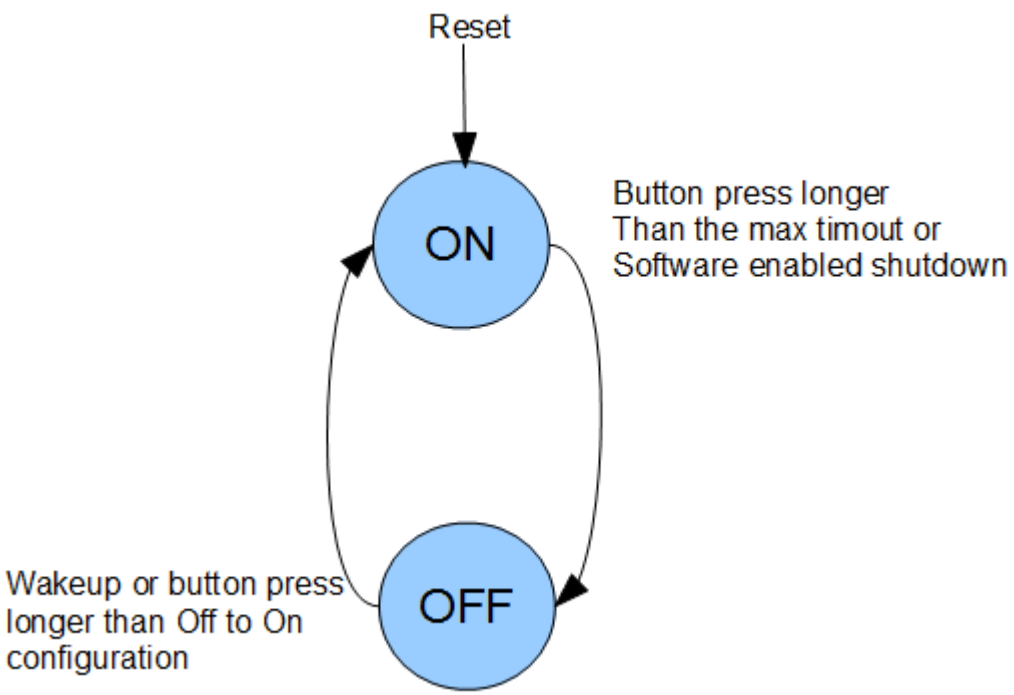
Dumb PMIC Mode:

The Dumb PMIC Mode uses PMIC "ON" Request to issue a level signal for on and off. Dumb pmic mode has many different configuration options which include (debounce, off to on time, and max time out).

- Debounce: The debounce configuration supports 0 msec, 50 msec, 100 msec and 500 msec. The debounce is used to generate the set_pwr_off_irq interrupt. While in the ON state and the button is pressed longer than the debounce time the set_pwr_off_irq is generated.

- Off to On Time: The Off to On configuration supports 0 msec, 50 msec, 100 msec, and 500 msec. This configuration supports the time it takes to request power on after the configured button press time has been reached. Once the button is pressed longer than the configuration time, the state machine will transition from the OFF to the ON state.
- Max Timeout: The max timeout configuration supports 5 secs, 10 secs, 15 secs and disable. This configuration supports the time it takes to request power down after the button has been pressed for the defined time.

The dumb PMIC mode uses a 2 state state machine, as shown below. The output of the pmic_en_b is generated by the state of the state machine.



Smart PMIC Mode:

The smart PMIC mode is meant to connect to another PMIC. The PMIC "ON" Request signal issues a pulse instead of a level signal. The only configuration option available for this mode is the Debounce configuration that is used for the "Power Off" Interrupt.

6.1.5 SNVS interrupts and alarms

SNVS provides the following interrupt and alarm lines:

- Functional interrupt (active low)

- Real-time clock period interrupt
- Power off (button) interrupt

The following table summarizes all SNVS interrupts and alarm sources.

Table 6-4. Interrupts and alarms summary

Interrupt	Source	Default configuration[1]	Configuration options
SNVS functional interrupt	RTC time alarm	Disable	Enable/Disable
	RTC periodic interrupt	Disable	Enable/Disable
SNVS power off (button) interrupt	BTN input signal	50msec debounce	Debounce time

1. Default behavior refers to the setting after LP/HP reset.

6.1.6 Programming Guidelines

This section provides initialization and application information for the SNVS module.

6.1.6.1 RTC control bits setting

All SNVS registers are programmed from the register bus. Therefore, any changes are synchronized with the IP clock. Several registers can also change synchronously with the RTC clock after they are programmed. To avoid IP clock and RTC clock synchronization issues, these values can only be programmed when the corresponding function is disabled. The following table presents the list of these values with the control bit setting required for programming.

Table 6-5. RTC synchronized values list

Function	Value/register	Control bit setting
HP section		
HP Real Time Counter	HPRTCMR and HPRTCLR Registers	RTC_EN = 0 - HPRTCMR/HPRTCLR can be programmed RTC_EN = 1 - HPRTCMR/HPRTCLR cannot be programmed
HP Time Alarm	HPTAMR and HPTALR Registers	HPTA_EN = 0 - HPTAMR/HPTALR can be programmed HPTA_EN = 1 - HPTAMR/HPTALR cannot be programmed
HP Time Calibration Value	HPCALB_VAL Value	HPCALB_EN = 0 - HPCALB_VAL can be programmed HPCALB_EN = 1 - HPCALB_VAL cannot be programmed

Use the following step to program synchronized values:

1. Check the enable bit value. If set, clear it.

2. Verify that the enable bit is cleared.

   There are two reasons to verify the enable bit's setting:

   - Enable bit clearing does not happen immediately; it takes three IPclock cycles and two RTC clock cycles to change the enable bit's value.
   - If the enable bit is locked for programming, it cannot be cleared.
3. Program the desired value.
4. Set the enable bit; it takes three IP clock cycles and two RTC clock cycles for the bit to set.

NOTE

Incrementing the value programmed into RTC registers by two compensates for the two RTC clock cycle delay that is required to enable the counter.

6.1.6.2 RTC value read

There are two scenarios when software can read corrupted values from the RTC (HPRTCMR and HPRTCLR) registers:

- The RTC counters are incremented by the slow 32 kHz clock, which is asynchronous to the system clock. The counter value is synchronized to the system clock before software reads that. The synchronization register may capture the counter value in the middle of the counter update. In this case, it is not guaranteed that all bits are properly sampled by the synchronization register; the value read by software can be wrong.
- The RTC value is longer than the single bus read transaction of 32-bits. Therefore, software reads two registers, each holding a portion of the counter value. After reading one of these registers but before reading the second register, both registers may update their values. In this case, the value combined by software will be incorrect.

To avoid these issues, it is strongly recommended that software perform two consecutive reads of the RTC value:

- If two consecutive reads are similar, the value is correct.
- If two consecutive reads are different, perform two more reads.

The worst case scenario may require three sessions of two consecutive reads.

6.1.6.3 General initialization guidelines

Complete the following steps in order to properly initialize the module:

1. Enable interrupts in SNVScontrol and configuration registers.
2. Program SNVS general functions/configurations.
3. User Specific: Set lock bits.

6.1.7 SNVS Register Descriptions

This section contains detailed register descriptions for the SNVS registers. Each description includes a standard register diagram and register table. The register table provides detailed descriptions of the register bit and field functions, in bit order.

SNVS registers consist of two types:

- Privileged read/write accessible
- Non-privileged read/write accessible

Privileged read/write accessible registers can only be accessed for read/write by privileged software. Unauthorized write accesses are ignored, and unauthorized read accesses return zero. Non-privileged software can access privileged access registers when the non-privileged software access enable bit is set in the SNVS_HP Command Register.

In addition, all privileged access registers (except HPSVSR and LPSR) can be written to only if the system security monitor is in one of the three functional states:

- Non-Secure
- Trusted
- Secure

Certain fields in the SNVS_HP Command Register can be written in non-functional system security monitor states (init, check, soft fail, and hard fail) as follows:

- SW_LPSV, SSM_ST, SW_SV, and SW_FSV can be written in check or soft fail state.
- The HAC_STOP bit can only be set in soft fail state but can be cleared in either soft fail or a functional state.
- HAC_LOAD, HAC_CLEAR bits and HPSVSR, LPSR Registers can be accessed in soft fail or a functional state.

The system security monitor state does not restrict read access to SNVS registers.

Non-privileged read/write accessible registers are read/write accessible by any software.

The LP register values are set only on LP POR and are unaffected by System (HP) POR. The HP registers are set only on System POR and are unaffected by LP POR.

The following table shows the SNVS main memory map.

6.1.7.1 SNVS Memory Map

Offset	Register	Width (In bits)	Access	Reset value
0h	SNVS_HP Lock (HPLR)	32	RW	00000000h
4h	SNVS_HP Command (HPCOMR)	32	RW	00000000h
8h	SNVS_HP Control (HPCR)	32	RW	00000000h
Ch	SNVS_HP Security Interrupt Control (HPSICR)	32	RW	00000000h
10h	SNVS_HP Security Violation Control (HPSVCR)	32	RW	00000000h
14h	SNVS_HP Status (HPSR)	32	W1C	80000000h
18h	SNVS_HP Security Violation Status (HPSVSR)	32	W1C	00000000h
1Ch	SNVS_HP High Assurance Counter IV (HPHACIVR)	32	RW	00000000h
20h	SNVS_HP High Assurance Counter (HPHACR)	32	RO	00000000h
24h	SNVS_HP Real Time Counter MSB (HPRTCMR)	32	RW	00000000h
28h	SNVS_HP Real Time Counter LSB (HPRTCLR)	32	RW	00000000h
2Ch	SNVS_HP Time Alarm MSB (HPTAMR)	32	RW	00000000h
30h	SNVS_HP Time Alarm LSB (HPTALR)	32	RW	00000000h
34h	SNVS_LP Lock (LPLR)	32	RW	00000000h
38h	SNVS_LP Control (LPCR)	32	RW	00000000h
3Ch	SNVS_LP Master Key Control (LPMKCR)	32	RW	00000000h
40h	SNVS_LP Security Violation Control (LPSVCR)	32	RW	00000000h
44h	SNVS_LP Tamper Glitch Filters Configuration (LPTGFCR)	32	RW	00000000h
48h	SNVS_LP Tamper Detectors Configuration (LPTDCR)	32	RW	00000000h
4Ch	SNVS_LP Status (LPSR)	32	W1C	00000008h
50h	SNVS_LP Secure Real Time Counter MSB (LPSRTCMR)	32	RW	00000000h
54h	SNVS_LP Secure Real Time Counter LSB (LPSRTCLR)	32	RW	00000000h
58h	SNVS_LP Time Alarm (LPTAR)	32	RW	00000000h
5Ch	SNVS_LP Secure Monotonic Counter MSB (LPSMCMR)	32	RO	00000000h
60h	SNVS_LP Secure Monotonic Counter LSB (LPSMCLR)	32	RO	00000000h
64h	SNVS_LP Power Glitch Detector (LPPGDR)	32	RW	00000000h
68h	SNVS_LP General Purpose 0 (alias) (LPGPR0_alias)	32	RW	00000000h
6Ch - 8Bh	SNVS_LP Zeroizable Master Key (LPZMKR0 - LPZMKR31)	8	RW	00h
90h - 9Ch	SNVS_LP General Purposes 0 .. 3 (LPGPR0_30 - LPGPR0_33)	32	RW	00000000h
A0h	SNVS_LP Tamper Detectors Config 2 (LPTDC2R)	32	RW	00000000h
A4h	SNVS_LP Tamper Detectors Status (LPTDSR)	32	W1C	00000000h
A8h	SNVS_LP Tamper Glitch Filter 1 Configuration (LPTGF1CR)	32	RW	00000000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
ACh	SNVS_LP Tamper Glitch Filter 2 Configuration (LPTGF2CR)	32	RW	00000000h
C0h	SNVS_LP Active Tamper 1 Configuration (LPAT1CR)	32	WO	00000000h
C4h	SNVS_LP Active Tamper 2 Configuration (LPAT2CR)	32	WO	00000000h
C8h	SNVS_LP Active Tamper 3 Configuration (LPAT3CR)	32	WO	00000000h
CCh	SNVS_LP Active Tamper 4 Configuration (LPAT4CR)	32	WO	00000000h
D0h	SNVS_LP Active Tamper 5 Configuration (LPAT5CR)	32	WO	00000000h
E0h	SNVS_LP Active Tamper Control (LPATCTLR)	32	RW	00000000h
E4h	SNVS_LP Active Tamper Clock Control (LPATCLKR)	32	RW	00000000h
E8h	SNVS_LP Active Tamper Routing Control 1 (LPATRC1R)	32	RW	00000000h
ECh	SNVS_LP Active Tamper Routing Control 2 (LPATRC2R)	32	RW	00000000h
BF8h	SNVS_HP Version ID 1 (HPVIDR1)	32	RO	003E0103h
BFCh	SNVS_HP Version ID 2 (HPVIDR2)	32	RO	05000000h

6.1.7.2 SNVS_HP Lock (HPLR)

6.1.7.2.1 Address

Register	Offset
HPLR	0h

6.1.7.2.2 Function

The SNVS_HP Lock Register contains lock bits for the SNVS registers. This is a privileged write register.

6.1.7.2.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			AT5_SL	AT4_SL	AT3_SL	AT2_SL	AT1_SL	Reserved					HAC_L	HPSICR_L	HPSVCR_L
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						MKS_SL	LPTDCR_SL	LPTGFCR_SL	LPSVCR_SL	GPR_SL	MC_SL	LPCALB_SL	SRTC_SL	ZMK_RSL	ZMK_WSL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.2.4 Fields

Field	Function
31-29 —	
28 AT5_SL	Active Tamper 5 Soft Lock When set, prevents any writes to the Active Tamper 5 registers. Once set, this bit can only be reset by the system reset 0b - 1b -
27 AT4_SL	Active Tamper 4 Soft Lock When set, prevents any writes to the Active Tamper 4 registers. Once set, this bit can only be reset by the system reset. 0b - 1b -
26 AT3_SL	Active Tamper 3 Soft Lock When set, prevents any writes to the Active Tamper 3 registers. Once set, this bit can only be reset by the system reset. 0b - 1b -
25 AT2_SL	Active Tamper 2 Soft Lock When set, prevents any writes to the Active Tamper 2 registers. Once set, this bit can only be reset by the system reset. 0b - 1b -
24 AT1_SL	Active Tamper 1 Soft Lock

Table continues on the next page...

Field	Function
	When set, prevents any writes to the Active Tamper 1 registers. Once set, this bit can only be reset by the system reset. 0b - 1b -
23-19 —	
18 HAC_L	High Assurance Configuration Lock When set, prevents any writes to HPHACIVR, HPHACR, and HAC_EN bit of HPCOMR. Once set, this bit can only be reset by the system reset. 0b - 1b -
17 HPSICR_L	HP Security Interrupt Control Register Lock When set, prevents any writes to the HPSICR. Once set, this bit can only be reset by the system reset. 0b - 1b -
16 HPSVCR_L	HP Security Violation Control Register Lock When set, prevents any writes to the HPSVCR. Once set, this bit can only be reset by the system reset. 0b - 1b -
15-10 —	
9 MKS_SL	Master Key Select Soft Lock When set, prevents any writes to the MASTER_KEY_SEL field of the LPMKCR. Once set, this bit can only be reset by the system reset. 0b - 1b -
8 LPTDCR_SL	LP Tamper Detectors Configuration Register Soft Lock When set, prevents any writes to the LPTDCR. Once set, this bit can only be reset by the system reset. 0b - 1b -
7 LPTGFCR_SL	LP Tamper Glitch Filter Configuration Register Soft Lock When set, prevents any writes to the LPTGFCR. Once set, this bit can only be reset by the system reset. 0b - 1b -
6 LPSVCR_SL	LP Security Violation Control Register Soft Lock When set, prevents any writes to the LPSVCR. Once set, this bit can only be reset by the system reset. 0b - 1b -
5 GPR_SL	General Purpose Register Soft Lock When set, prevents any writes to the GPR. Once set, this bit can only be reset by the system reset.

Table continues on the next page...

Field	Function
	0b - 1b -
4 MC_SL	Monotonic Counter Soft Lock When set, prevents any writes (increments) to the MC Registers and MC_ENV bit. Once set, this bit can only be reset by the system reset. 0b - 1b -
3 LPCALB_SL	LP Calibration Soft Lock When set, prevents any writes to the LP Calibration Value (LPCALB_VAL) and LP Calibration Enable (LPCALB_EN). Once set, this bit can only be reset by the system reset. 0b - 1b -
2 SRTC_SL	Secure Real Time Counter Soft Lock When set, prevents any writes to the SRTC Registers, SRTC_ENV, and SRTC_INV_EN bits. Once set, this bit can only be reset by the system reset. 0b - 1b -
1 ZMK_RSL	Zeroizable Master Key Read Soft Lock When set, prevents any software reads to the ZMK Registers and ZMK_ECC_VALUE field of the LPMKCR. In ZMK hardware programming mode (ZMK_HWP is set), the ZMK and ZMK_ECC_VALUE cannot be read by software. Regardless of the bit setting, hardware can use the ZMK value when ZMK is selected. Once set, this bit can only be reset by the system reset. 0b - 1b -
0 ZMK_WSL	Zeroizable Master Key Write Soft Lock When set, prevents any writes (software and hardware) to the ZMK registers and MASTER_KEY_SEL, ZMK_HWP, ZMK_VAL, and ZMK_ECC_EN fields of the LPMKCR. Once set, this bit can only be cleared by system reset. 0b - 1b -

6.1.7.3 SNVS_HP Command (HPCOMR)

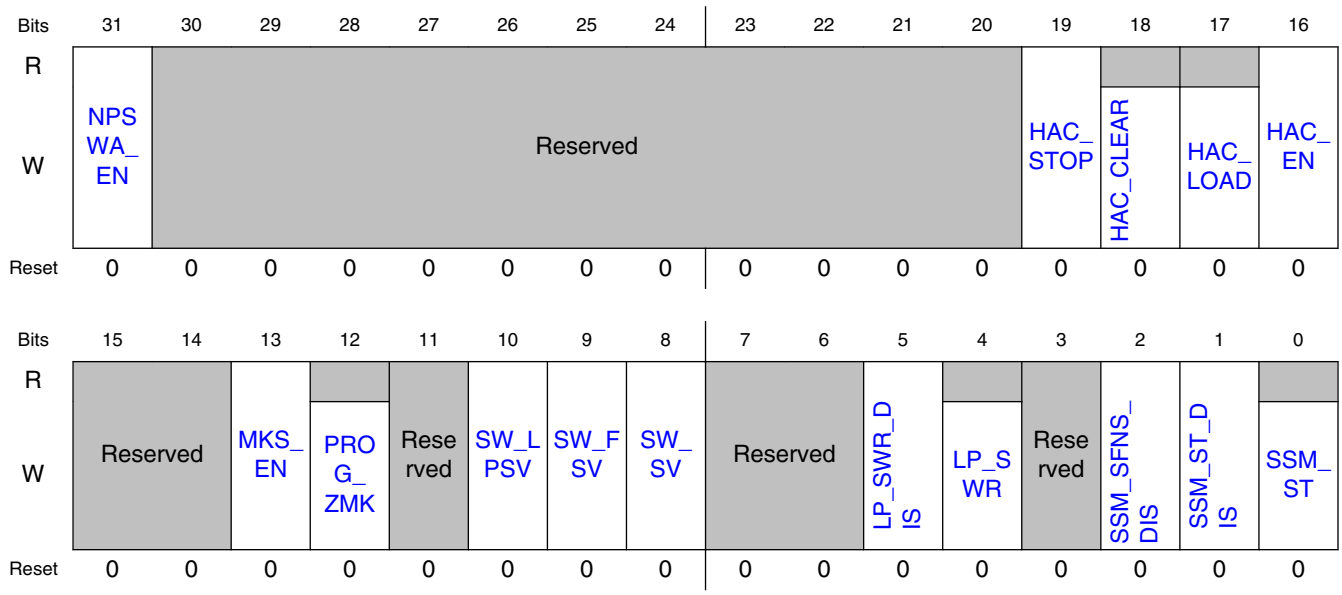
6.1.7.3.1 Address

Register	Offset
HPCOMR	4h

6.1.7.3.2 Function

The SNVS_HP Command Register contains the command, configuration, and control bits for the SNVS block. Some fields of this register can be written to in check and soft fail states in addition to the standard write access in functional states. This is a privileged write register.

6.1.7.3.3 Diagram



6.1.7.3.4 Fields

Field	Function
31 NPSWA_EN	Non-Privileged Software Access Enable When set, allows non-privileged software to access all SNVS registers, including those that are privileged software read/write access only. 0 Only privileged software can access privileged registers 1 Any software can access privileged registers
30-20 —	
19 HAC_STOP	High Assurance Counter Stop This bit can be set only when SSM is in soft fail state. When set, it stops the high assurance counter and prevents transition to the hard fail state. This bit can be cleared in a functional or soft fail state. If the bit is cleared in the soft fail state, the high assurance counter counts down from the place where it was stopped. 0 HAC counter can count down

Table continues on the next page...

Field	Function
	1 HAC counter is stopped
18 HAC_CLEAR	High Assurance Counter Clear When set, it clears the High Assurance Counter Register. It can be cleared in a functional or soft fail state. If the HAC counter is cleared in the soft fail state, the SSM transitions to the hard fail state if high assurance configuration is enabled (HAC_EN is set). This self-clearing bit is always read as zero. 0b - 1b -
17 HAC_LOAD	High Assurance Counter Load When set, it loads the High Assurance Counter Register with the value of the High Assurance Counter Load Register. It can be done in a functional or soft fail state. This self-clearing bit is always read as zero. 0b - 1b -
16 HAC_EN	High Assurance Configuration Enable This bit controls the SSM transition from the soft fail to the hard fail state. When this bit is set and software fails to stop the HAC before it expires, the SSM transitions to the hard fail state. This bit cannot be changed once HAC_L bit is set. 0b - 1b -
15-14 —	
13 MKS_EN	Master Key Select Enable When not set, the one time programmable (OTP) master key is selected by default. When set, the master key is selected according to the setting of the master key select field (MASTER_KEY_SEL) of LPMKCR. Once set, this bit can only be reset by the system reset. 0b - OTP master key is selected as an SNVS master key 1b - SNVS master key is selected according to the setting of the MASTER_KEY_SEL field of LPMKCR
12 PROG_ZMK	Program Zeroizable Master Key This bit activates ZMK hardware programming mechanism. This mechanism is activated only if the ZMK is configured to the hardware programming mode and ZMK in not locked for writes. This self-clearing bit is always read as zero. 0b - 1b -
11 —	
10 SW_LPSV	LP Software Security Violation When set, SNVS_LP treats this bit as a security violation. The LP secure data is zeroized or invalidated according to the configuration. This security violation may result in a system security monitor transition if the LP Security Violation is enabled in the SNVS_HP Security Violation Control Register.
9 SW_FSV	Software Fatal Security Violation When set, the system security monitor treats this bit as a fatal security violation. This security violation has no effect on the LP section. This command results only in the following transitions of the SSM: Check State -> Soft Fail

Table continues on the next page...

Field	Function
	Non-Secure State -> Soft Fail Trusted State -> Soft Fail Secure State -> Soft Fail
8 SW_SV	Software Security Violation When set, the system security monitor treats this bit as a non-fatal security violation. This security violation has no effect on the LP section. This command results only in the following transitions of the SSM: Check -> Non-Secure Trusted -> Soft Fail Secure -> Soft Fail
7-6 —	
5 LP_SWR_DIS	LP Software Reset Disable When set, disables the LP software reset. Once set, this bit can only be reset by the system reset. 0b - 1b -
4 LP_SWR	LP Software Reset When set, it resets the SNVS_LP section. This bit cannot be set when the LP_SWR_DIS bit is set. This self-clearing bit is always read as zero. 0b - 1b -
3 —	
2 SSM_SFNS_DIS	SSM Soft Fail to Non-Secure State Transition Disable When set, it disables the SSM transition from soft fail to non-secure state. Once set after the reset this bit cannot be changed 0b - 1b -
1 SSM_ST_DIS	SSM Secure to Trusted State Transition Disable When set, disables the SSM transition from secure to trusted state. Once set after the reset, this bit cannot be changed. 0b - 1b -
0 SSM_ST	SSM State Transition Transition state of the system security monitor. This self-clearing bit is always read as zero. This command results only in the following transitions of the SSM: Check State -> Non-Secure (when Non-Secure Boot and not in Fab Configuration) Check State -> Trusted (when Secure Boot or in Fab Configuration) Trusted State -> Secure Secure State -> Trusted (if not disabled by SSM_ST_DIS bit)

Field	Function
	Soft Fail -> Non-Secure (if not disabled by SSM_SFNS_DIS bit)

6.1.7.4 SNVS_HP Control (HPCR)

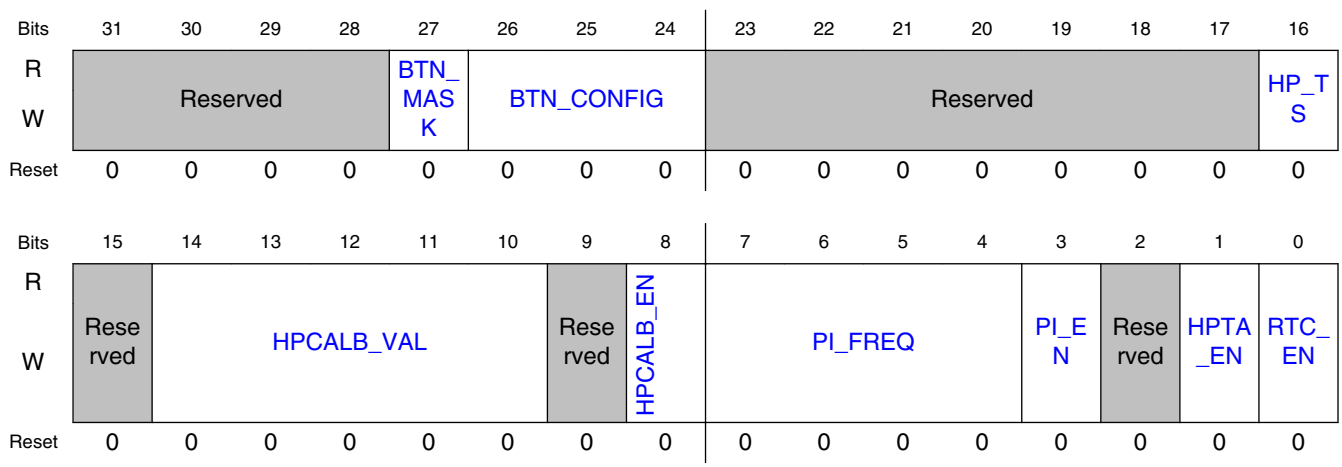
6.1.7.4.1 Address

Register	Offset
HPCR	8h

6.1.7.4.2 Function

The SNVS_HP Control Register contains various control bits of the HP section of SNVS.

6.1.7.4.3 Diagram



6.1.7.4.4 Fields

Field	Function
31-28 —	
27 BTN_MASK	Button interrupt mask. This bit is used to mask the ipi_snvs_btn_int_b (button) interrupt request.

Table continues on the next page...

Field	Function
	0: Interrupt disabled 1: Interrupt enabled
26-24 BTN_CONFIG	Button Configuration. This field is used to configure which feature of the button (BTN) input signal constitutes "active". 000: Button signal is active low 001: Button signal is active high 010: Button signal is active on the rising edge 011: Button signal is active on the falling edge 100: Button signal is active on any edge All other patterns are reserved.
23-17 —	
16 HP_TS	HP Time Synchronize When set, this updates the HP Real Time Counter with the LP Real Time Counter value. This self-clearing bit is always read as zero. 0b - 1b -
15 —	
14-10 HPCALB_VAL	HP Calibration Value Defines signed calibration value for the HP Real Time Counter. This field can be programmed only when RTC Calibration is disabled (HPCALB_EN is not set). This is a 5-bit 2's complement value, hence the allowable calibration values are in the range from -16 to +15 counts per 32768 ticks of the counter. 00000b - 00001b - 00010b - 01111b - 10000b - 10001b - 11110b - 11111b -
9 —	
8 HPCALB_EN	HP Real Time Counter Calibration Enabled Indicates that the time calibration mechanism is enabled. 0b - 1b -
7-4 PI_FREQ	Periodic Interrupt Frequency Defines frequency of the periodic interrupt. The interrupt is generated when a zero-to-one or one-to-zero transition occurs on the selected bit of the HP Real Time Counter and Real Time Counter and Periodic Interrupt are both enabled (RTC_EN and PI_EN are set). It is recommended to program this field when Periodic Interrupt is disabled (PI_EN is not set). The possible frequencies are:

Table continues on the next page...

Field	Function
	0000b - 0001b - 0010b - 0011b - 0100b - 0101b - 0110b - 0111b - 1000b - 1001b - 1010b - 1011b - 1100b - 1101b - 1110b - 1111b -
3 PI_EN	HP Periodic Interrupt Enable The periodic interrupt can be generated only if the HP Real Time Counter is enabled. 0b - 1b -
2 —	
1 HPTA_EN	HP Time Alarm Enable When set, the time alarm interrupt is generated if the value in the HP Time Alarm Registers is equal to the value of the HP Real Time Counter. 0b - 1b -
0 RTC_EN	HP Real Time Counter Enable 0b - 1b -

6.1.7.5 SNVS_HP Security Interrupt Control (HPSICR)

6.1.7.5.1 Address

Register	Offset
HPSICR	Ch

6.1.7.5.2 Function

The HP Security Interrupt Control Register defines the SNVS security interrupt generation policy. This is a privileged write register.

6.1.7.5.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LPSVI_EN	Reserved														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										SVI_EN5	SVI_EN4	SVI_EN3	SVI_EN2	SVI_EN1	SVI_EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.5.4 Fields

Field	Function
31 LPSVI_EN	LP Security Violation Interrupt Enable This bit enables generating of the security interrupt to the host processor upon security violation signal from the LP section. 0b - 1b -
30-6 —	Reserved
5 SVI_EN5	Security Violation Interrupt 5 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 5. 0b - 1b -
4 SVI_EN4	Security Violation Interrupt 4 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 4. 0b - 1b -
3 SVI_EN3	Security Violation Interrupt 3 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 3. 0b - 1b -
2 SVI_EN2	Security Violation Interrupt 2 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 2.

Table continues on the next page...

Field	Function
	0b - 1b -
1 SVI_EN1	Security Violation Interrupt 1 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 1. 0b - 1b -
0 SVI_EN0	Security Violation Interrupt 0 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 0. 0b - 1b -

6.1.7.6 SNVS_HP Security Violation Control (HPSVCR)

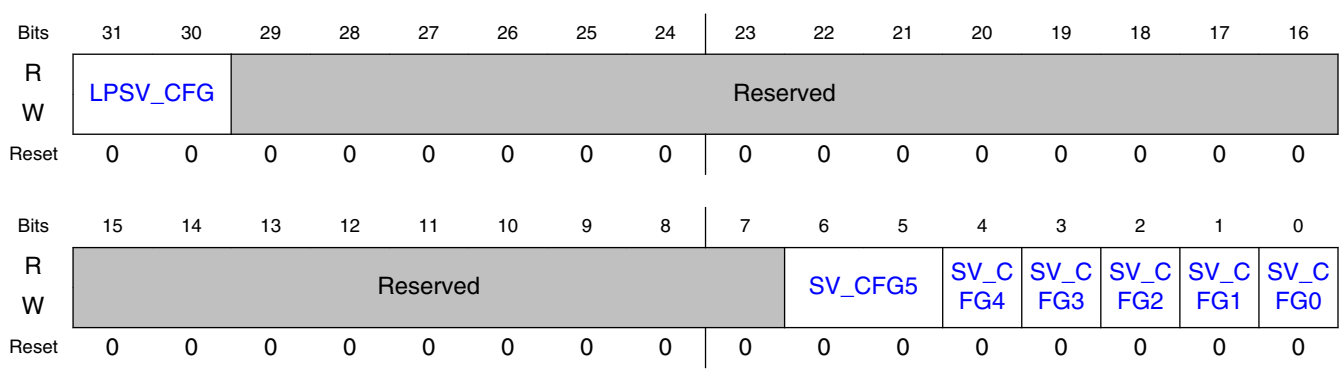
6.1.7.6.1 Address

Register	Offset
HPSVCR	10h

6.1.7.6.2 Function

The HP Security Violation Control Register defines types for each security violation input. This is a privileged write register.

6.1.7.6.3 Diagram



6.1.7.6.4 Fields

Field	Function
31-30 LPSV_CFG	LP Security Violation Configuration This field configures the LP security violation source. 00b - 01b - 1xb -
29-7 —	
6-5 SV_CFG5	Security Violation Input 5 Configuration This field configures the Security Violation Input 5. This setting instructs the SSM how to respond when a security violation on port 5 is detected. 00b - 01b - 1xb -
4 SV_CFG4	Security Violation Input 4 Configuration This field configures the security violation Input 4. This setting instructs the SSM how to respond when a security violation on port 4 is detected. 0b - 1b -
3 SV_CFG3	Security Violation Input 3 Configuration This field configures the security violation input 3. This setting instructs the SSM how to respond when a security violation on port 3 is detected. 0b - 1b -
2 SV_CFG2	Security Violation Input 2 Configuration This field configures the security violation input 2. This setting instructs the SSM how to respond when a security violation on port 2 is detected. 0b - 1b -
1 SV_CFG1	Security Violation Input 1 Configuration This field configures the Security Violation Input 1. This setting instructs the SSM how to respond when a security violation on port 1 is detected. 0b - 1b -
0 SV_CFG0	Security Violation Input 0 Configuration This field configures the security violation input 0. This setting instructs the SSM how to respond when a security violation on port 0 is detected. 0b - 1b -

6.1.7.7 SNVS_HP Status (HPSR)

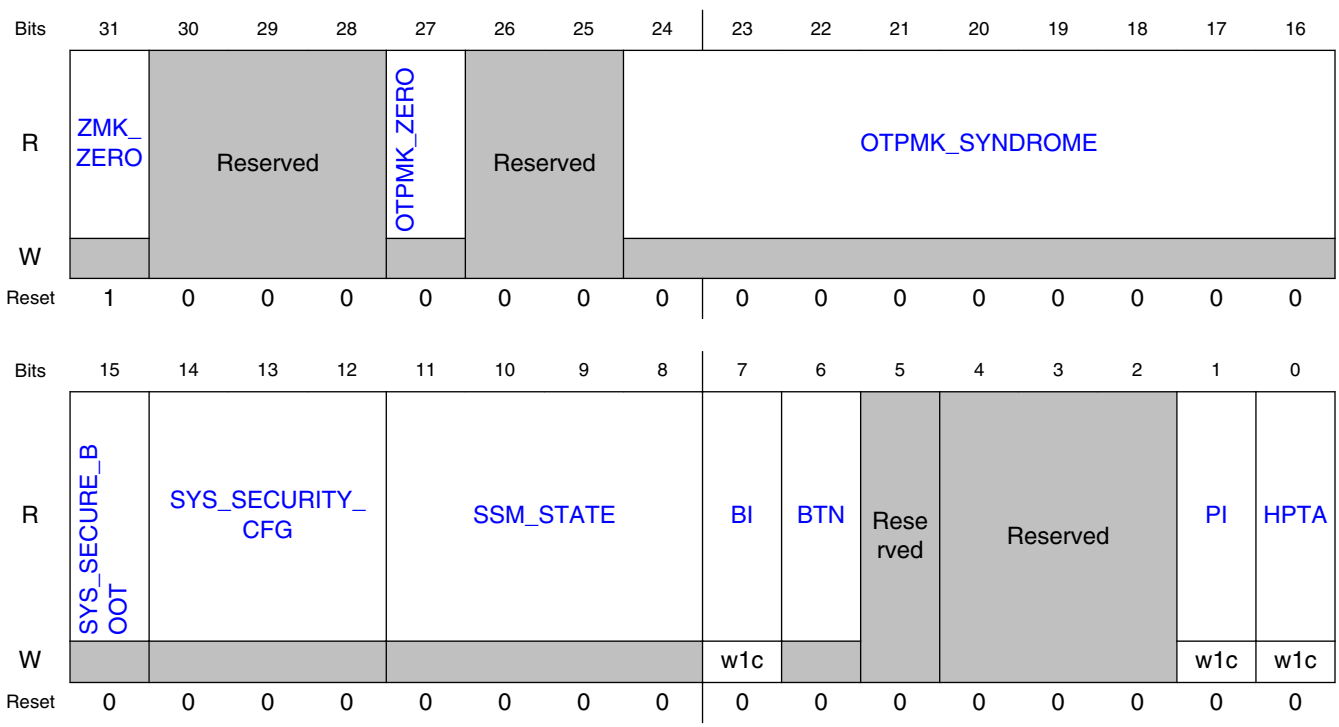
6.1.7.7.1 Address

Register	Offset
HPSR	14h

6.1.7.7.2 Function

The HP Status Register reflects the internal state of the SNVS.

6.1.7.7.3 Diagram



6.1.7.7.4 Fields

Field	Function
31 ZMK_ZERO	Zeroizable Master Key is Equal to Zero. When set, this bit triggers "bad key" violation if the ZMK is selected for use. 0b -

Table continues on the next page...

Field	Function
	1b -
30-28 —	
27 OTPMK_ZERO	One Time Programmable Master Key is Equal to Zero. When set, this bit always triggers "bad key" violation. 0b - 1b -
26-25 —	
24-16 OTPMK_SYNDROME	One Time Programmable Master Key Syndrome In the case of a single-bit error, the eight lower bits of this value indicate the bit number of error location. For example, syndrome word 10010110 indicates that key bit 150 (lsb=0) has an error. The ninth bit of the syndrome word checks parity of the whole key value. This bit is 1 when an odd number of errors has occurred and it is 0 when the number of errors is even. For example, if one of the eight bits indicates a failure and the ninth bit is zero then the number of errors in the one time programmable master key is at least 2 and it cannot be corrected. When one of the syndrome bits is set, the bad key violation is always generated.
15 SYS_SECURE_BOOT	System Secure Boot This bit reflects the value of the sys_secure_boot input signal to SNVS. If this bit is 1, the chip boots from internal ROM.
14-12 SYS_SECURITY_CFG	System Security Configuration This field reflects the value of the sys_security_cfg input signal, which is defined as follows: 000b - 001b - 01xb - 1xxb -
11-8 SSM_STATE	System Security Monitor State This field contains the encoded state of the SSM's state machine. The encoding of the possible states are: 0000b - 0001b - 0011b - 1000b - 1001b - 1011b - 1101b - 1111b -
7 BI	Button Interrupt Signal ipi_snvs_btn_int_b was asserted.
6 BTN	Button Value of the BTN input. This is the external button used for PMIC control. 0: BTN not pressed 1: BTN pressed

Table continues on the next page...

Field	Function
5 —	
4-2 —	
1 PI	Periodic Interrupt Indicates that periodic interrupt has occurred since this bit was last cleared. 0b - 1b -
0 HPTA	HP Time Alarm Indicates that the HP Time Alarm has occurred since this bit was last cleared. 0b - 1b -

6.1.7.8 SNVS_HP Security Violation Status (HPSVSR)

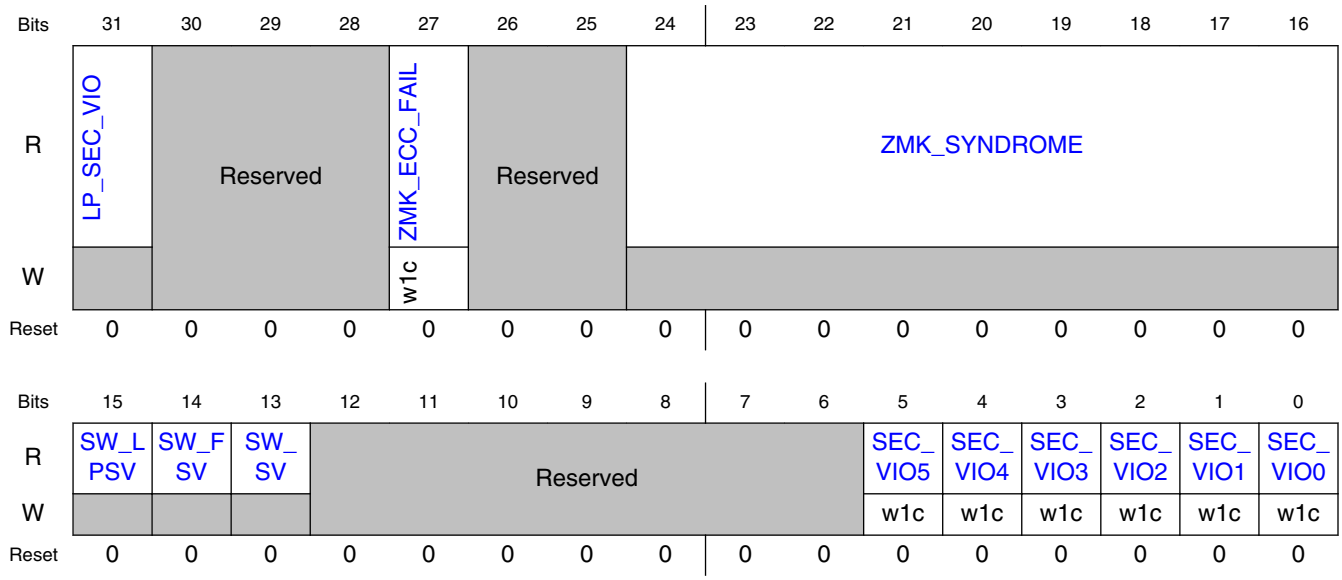
6.1.7.8.1 Address

Register	Offset
HPSVSR	18h

6.1.7.8.2 Function

The HP Security Violation Status Register reflects the HP domain security violation records. Write a 1 to SEC_VIO5-0 to clear the corresponding security violation detection flag. Note that this does not automatically clear the security violation signal that is connected to the input, so the security violation may immediately be detected again

6.1.7.8.3 Diagram



6.1.7.8.4 Fields

Field	Function
31 LP_SEC_VIO	LP Security Violation A security volation was detected in the SNVS low power section.
30-28 —	
27 ZMK_ECC_FAIL	Zeroizable Master Key Error Correcting Code Check Failure When set, this bit triggers a bad key violation to the SSM and a security violation to the SNVS_LP section, which clears security sensitive data. Writing a one to this bit clears the record of this failure and also clears this register's ZMK_SYNDROME field. 0b - 1b -
26-25 —	
24-16 ZMK_SYNDROME	Zeroizable Master Key Syndrome The ZMK syndrome indicates the single-bit error location and parity for the ZMK regsiter. It operates similar to the OTPMK syndrome. This value is set and locked when a ZMK ECC failure is detected. It is cleared by writing one into ZMK_ECC_FAIL bit.
15 SW_LPSV	LP Software Security Violation This bit is a read-only copy of the SW_LPSV bit in the HP Command Register.
14 SW_FSV	Software Fatal Security Violation This bit is a read-only copy of the SW_FSV bit in the HP Command Register.

Table continues on the next page...

Field	Function
13 SW_SV	Software Security Violation This bit is a read-only copy of the SW_SV bit in the HP Command Register.
12-6 —	
5 SEC_VIO5	Security volation on input 5 was detected. 0b - 1b -
4 SEC_VIO4	Security violation on input 4 was detected. 0b - 1b -
3 SEC_VIO3	Security violation on input 3 was detected. 0b - 1b -
2 SEC_VIO2	Security violation on input 2 was detected. 0b - 1b -
1 SEC_VIO1	Security violation on input 1 was detected. 0b - 1b -
0 SEC_VIO0	Security violation on input 0 was detected. 0b - 1b -

6.1.7.9 SNVS_HP High Assurance Counter IV (HPHACIVR)

6.1.7.9.1 Address

Register	Offset
HPHACIVR	1Ch

6.1.7.9.2 Function

The SNVS_HP High Assurance Counter IV Register contains the initial value for the high assurance counter.

6.1.7.9.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HAC_COUNTER_IV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HAC_COUNTER_IV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.9.4 Fields

Field	Function
31-0 HAC_COUNTER_IV	High Assurance Counter Initial Value This register is used to set the starting count value to the high assurance counter. This register cannot be programmed when HAC_L bit is set.

6.1.7.10 SNVS_HP High Assurance Counter (HPHACR)

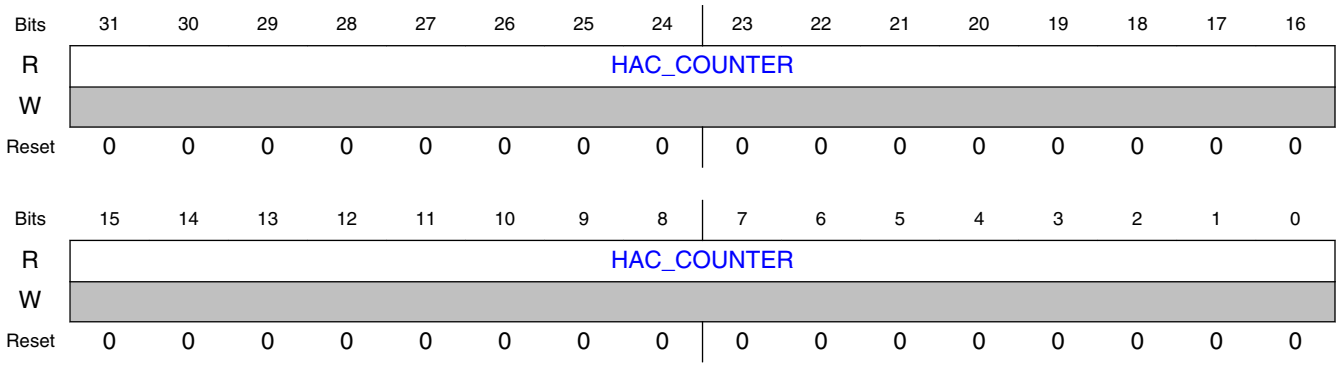
6.1.7.10.1 Address

Register	Offset
HPHACR	20h

6.1.7.10.2 Function

The SNVS_HP High Assurance Counter Register contains the value of the high assurance counter. The high assurance counter is a delay introduced before the system security monitor transitions from soft fail to hard fail state if this transition is enabled.

6.1.7.10.3 Diagram



6.1.7.10.4 Fields

Field	Function
31-0 HAC_COUNTER	High Assurance Counter When the HAC_EN bit is set and the SSM is in the soft fail state, this counter starts to count down with the system clock. When the counter reaches zero, the SSM transitions to the Hard Fail State. • When HAC_STOP bit is set, the HAC Counter is stopped. • When HAC_CLEAR bit is set, the HAC Counter is cleared. • When HAC_LOAD bit is set, the HAC Counter is loaded with the value of the HPHACIVR.

6.1.7.11 SNVS_HP Real Time Counter MSB (HPRTCMR)

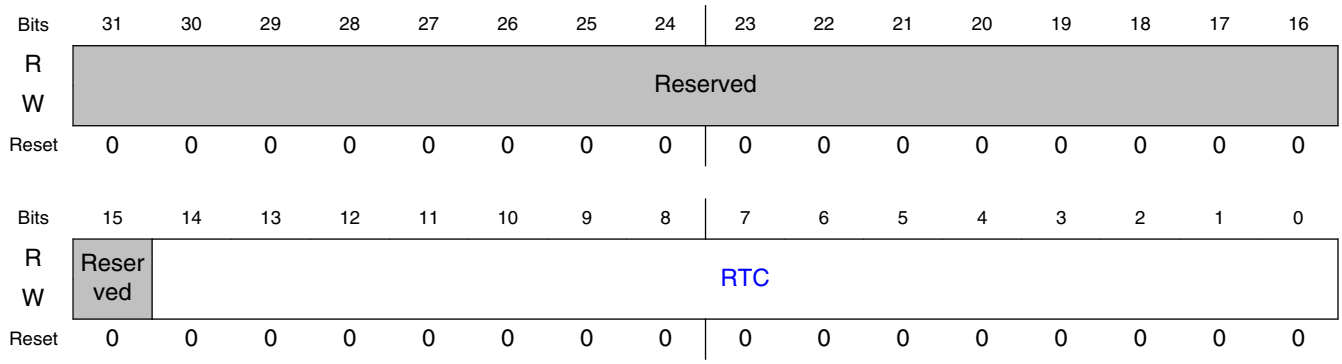
6.1.7.11.1 Address

Register	Offset
HPRTCMR	24h

6.1.7.11.2 Function

The SNVS_HP Real Time Counter MSB register contains the 15 most-significant bits of the HP Real Time Counter.

6.1.7.11.3 Diagram



6.1.7.11.4 Fields

Field	Function
31-15 —	Reserved
14-0 RTC	HP Real Time Counter The most-significant 15 bits of the RTC. This register can be programmed only when RTC is not active (RTC_EN bit is not set).

6.1.7.12 SNVS_HP Real Time Counter LSB (HPRTCLR)

6.1.7.12.1 Address

Register	Offset
HPRTCLR	28h

6.1.7.12.2 Function

The SNVS_HP Real Time Counter LSB register contains the 32 least-significant bits of the HP real time counter.

6.1.7.12.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RTC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RTC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.12.4 Fields

Field	Function
31-0 RTC	HP Real Time Counter least-significant 32 bits. This register can be programmed only when RTC is not active (RTC_EN bit is not set).

6.1.7.13 SNVS_HP Time Alarm MSB (HPTAMR)

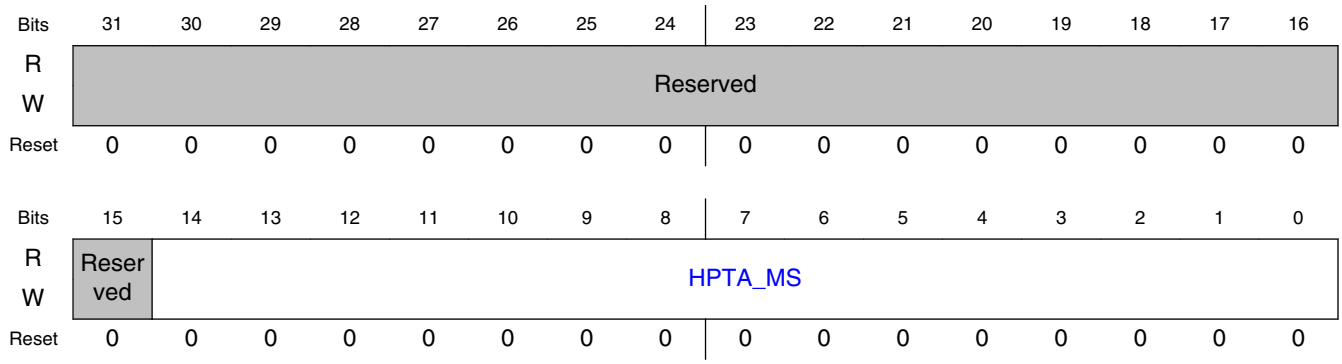
6.1.7.13.1 Address

Register	Offset
HPTAMR	2Ch

6.1.7.13.2 Function

The SNVS_HP Time Alarm MSB register contains the most-significant bits of the SNVS_HP Time Alarm value.

6.1.7.13.3 Diagram



6.1.7.13.4 Fields

Field	Function
31-15 —	
14-0 HPTA_MS	HP Time Alarm, most-significant 15 bits. This register can be programmed only when HP time alarm is disabled (HPTA_EN bit is not set).

6.1.7.14 SNVS_HP Time Alarm LSB (HPTALR)

6.1.7.14.1 Address

Register	Offset
HPTALR	30h

6.1.7.14.2 Function

The SNVS_HP Time Alarm LSB register contains the 32 least-significant bits of the SNVS_HP Time Alarm value.

6.1.7.14.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HPTA_LS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HPTA_LS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.14.4 Fields

Field	Function
31-0 HPTA_LS	HP Time Alarm, 32 least-significant bits. This register can be programmed only when HP time alarm is disabled (HPTA_EN bit is not set).

6.1.7.15 SNVS_LP Lock (LPLR)

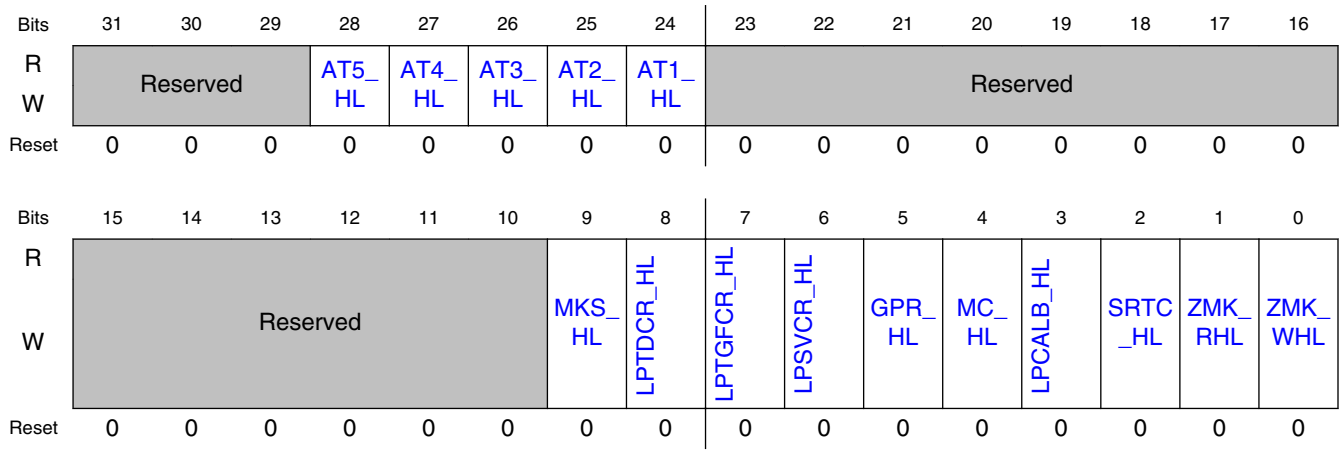
6.1.7.15.1 Address

Register	Offset
LPLR	34h

6.1.7.15.2 Function

The SNVS_LP Lock Register contains lock bits for the SNVS_LP registers.

6.1.7.15.3 Diagram



6.1.7.15.4 Fields

Field	Function
31-29 —	
28 AT5_HL	Active Tamper 5 Hard Lock When set, prevents any writes to the Active Tamper 5 registers. Once set, this bit can only be reset by the LP POR. 0b - 1b -
27 AT4_HL	Active Tamper 4 Hard Lock When set, prevents any writes to the Active Tamper 4 registers. Once set, this bit can only be reset by the LP POR. 0b - 1b -
26 AT3_HL	Active Tamper 3 Hard Lock When set, prevents any writes to the Active Tamper 3 registers. Once set, this bit can only be reset by the LP POR. 0b - 1b -
25 AT2_HL	Active Tamper 2 Hard Lock When set, prevents any writes to the Active Tamper 2 registers. Once set, this bit can only be reset by the LP POR. 0b - 1b -
24 AT1_HL	Active Tamper 1 Hard Lock

Table continues on the next page...

Field	Function
	When set, prevents any writes to the Active Tamper 1 registers. Once set, this bit can only be reset by the LP POR. 0b - 1b -
23-10 —	
9 MKS_HL	Master Key Select Hard Lock When set, prevents any writes to the MASTER_KEY_SEL field of the LP Master Key Control Register. Once set, this bit can only be reset by the LP POR. 0b - 1b -
8 LPTDCR_HL	LP Tamper Detectors Configuration Register Hard Lock When set, prevents any writes to the LPTDCR. Once set, this bit can only be reset by the LP POR. 0b - 1b -
7 LPTGFCR_HL	LP Tamper Glitch Filter Configuration Register Hard Lock When set, prevents any writes to the LPTGFCR. Once set, this bit can only be reset by the LP POR. 0b - 1b -
6 LPSVCR_HL	LP Security Violation Control Register Hard Lock When set, prevents any writes to the LPSVCR. Once set, this bit can only be reset by the LP POR. 0b - 1b -
5 GPR_HL	General Purpose Register Hard Lock When set, prevents any writes to the GPR. Once set, this bit can only be reset by the LP POR. 0b - 1b -
4 MC_HL	Monotonic Counter Hard Lock When set, prevents any writes (increments) to the MC Registers and MC_ENV bit. Once set, this bit can only be reset by the LP POR. 0b - 1b -
3 LPCALB_HL	LP Calibration Hard Lock When set, prevents any writes to the LP Calibration Value (LPCALB_VAL) and LP Calibration Enable (LPCALB_EN). Once set, this bit can only be reset by the LP POR. 0b - 1b -
2 SRTC_HL	Secure Real Time Counter Hard Lock When set, prevents any writes to the SRTC registers, SRTC_ENV, and SRTC_INV_EN bits. Once set, this bit can only be reset by the LP POR. 0b -

Table continues on the next page...

Field	Function
	1b -
1 ZMK_RHL	Zeroizable Master Key Read Hard Lock When set, prevents any software reads to the ZMK registers and ZMK_ECC_VALUE field of the LPMKCR. In ZMK hardware programming mode (ZMK_HWP is set), software cannot read the ZMK or ZMK_ECC_VALUE. Regardless of the setting of this bit, hardware can use the ZMK value when ZMK is selected. Once set, this bit can only be reset by the LP POR. 0b - 1b -
0 ZMK_WHL	Zeroizable Master Key Write Hard Lock When set, prevents any writes (software and hardware) to the ZMK registers and ZMK_HWP, ZMK_VAL, and ZMK_ECC_EN fields of the LPMKCR. Once set, this bit can only be reset by the LP POR. 0b - 1b -

6.1.7.16 SNVS_LP Control (LPCR)

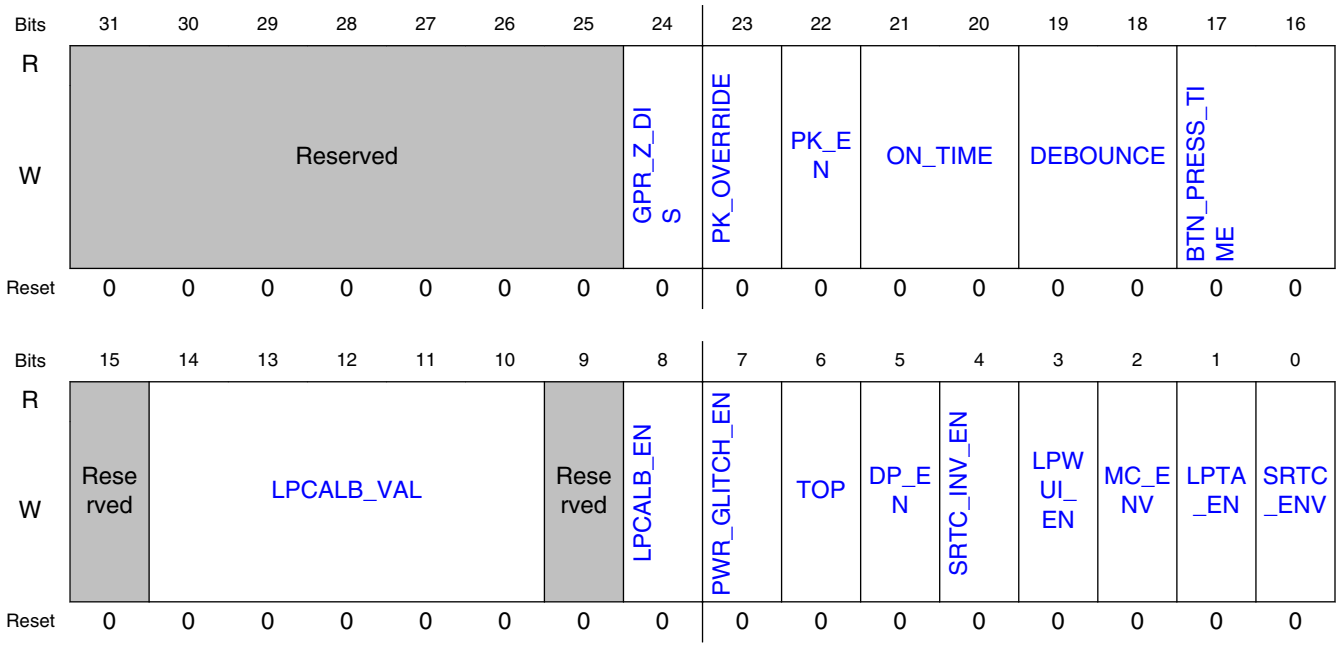
6.1.7.16.1 Address

Register	Offset
LPCR	38h

6.1.7.16.2 Function

The SNVS_LP Control Register contains various control bits of the LP section of SNVS.

6.1.7.16.3 Diagram



6.1.7.16.4 Fields

Field	Function
31-25 —	
24 GPR_Z_DIS	General Purpose Registers Zeroization Disable. 1 = Disable zeroization of the GPR registers when a tamper event occurs. 0 = Zeroize the GPR registers when a tamper event occurs. (Default)
23 PK_OVERRIDE	PMIC On Request Override The value written to PK_OVERRIDE will be asserted on output signal snvs_lp_pk_override. That signal is used to override the IOMUX control for the PMIC I/O pad.
22 PK_EN	PMIC On Request Enable The value written to PK_EN will be asserted on output signal snvs_lp_pk_en. That signal is used to turn off the pullup/pulldown circuitry in the PMIC I/O pad.
21-20 ON_TIME	The ON_TIME field is used to configure the period of time after BTN is asserted before pmic_en_b is asserted to turn on the SoC power. 00: 500msec off->on transition time 01: 50msec off->on transition time 10: 100msec off->on transition time 11: 0msec off->on transition time
19-18	This field configures the amount of debounce time for the BTN input signal.

Table continues on the next page...

Field	Function
DEBOUNCE	00: 50msec debounce 01: 100msec debounce 10: 500msec debounce 11: 0msec debounce
17-16 BTN_PRESS_TIME	This field configures the button press time out values for the PMIC Logic. 00 : 5 secs 01 : 10 secs 10 : 15 secs 11 : long press disabled (pmic_en_b will not be asserted regardlessof how long BTN is asserted)
15 —	
14-10 LPCALB_VAL	LP Calibration Value Defines signed calibration value for SRTC. This field can be programmed only when SRTC calibration is disabled and not locked, i.e. when LPCALB_EN, LPCALB_SL, and LPCALB_HL bits are not set. This is a 5-bit 2's complement value. Hence, the allowable calibration values are in the range from -16 to +15 counts per 32768 ticks of the counter clock 00000b - 00001b - 00010b - 01111b - 10000b - 10001b - 11110b - 11111b -
9 —	
8 LPCALB_EN	LP Calibration Enable When set, enables the SRTC calibration mechanism. This bit cannot be changed once LPCALB_SL or LPCALB_HL bit is set. 0b - 1b -
7 PWR_GLITCH_EN	Power Glitch Enable By default the detection of a power glitch does not cause the pmic_en_b signal to be asserted. Setting the Power Glitch Enable bit to 1 enables the power glitch event for the PMIC. 0 - disabled 1 - enabled
6 TOP	Turn off System Power Asserting this bit causes a signal to be sent to the Power Management IC to turn off the system power. This bit will clear once power is off. This bit is only valid when the Dumb PMIC is enabled. 0b - 1b -
5	Dumb PMIC Enabled

Table continues on the next page...

Field	Function
DP_EN	When set, software can control the system power. When cleared, the system requires a Smart PMIC to automatically turn power off. 0b - 1b -
4 SRTC_INV_EN	If this bit is 1, in the case of a security violation the SRTC stops counting and the SRTC is invalidated (SRTC_ENV bit is cleared). This is intended to allow software to read the time at which the security violation occurred. This field cannot be changed once SRTC_SL or SRTC_HL bit is set. 0b - 1b -
3 LPWUI_EN	LP Wake-Up Interrupt Enable This interrupt line should be connected to the external pin and is intended to inform the external chip about an SNVS_LP event (tamper event, MC rollover,SRTC rollover, or time alarm). This wake-up signal can be asserted only when the chip (HP section) is powered down, and the LP section is isolated. 0 LP wake-up interrupt is disabled. 1 LP wake-up interrupt is enabled.
2 MC_ENV	Monotonic Counter Enabled and Valid When set, the MC can be incremented (by write transaction to the LPSMCMR or LPSMCLR). This bit cannot be changed once MC_SL or MC_HL bit is set. 0b - 1b -
1 LPTA_EN	LP Time Alarm Enable When set, the SNVS functional interrupt is asserted if the LP Time Alarm Register is equal to the 32 MSBs of the secure real time counter. 0b - 1b -
0 SRTC_ENV	Secure Real Time Counter Enabled and Valid When set, the SRTC becomes operational. This bit cannot be changed once SRTC_SL or SRTC_HL bit is set. 0b - 1b -

6.1.7.17 SNVS_LP Master Key Control (LPMKCR)

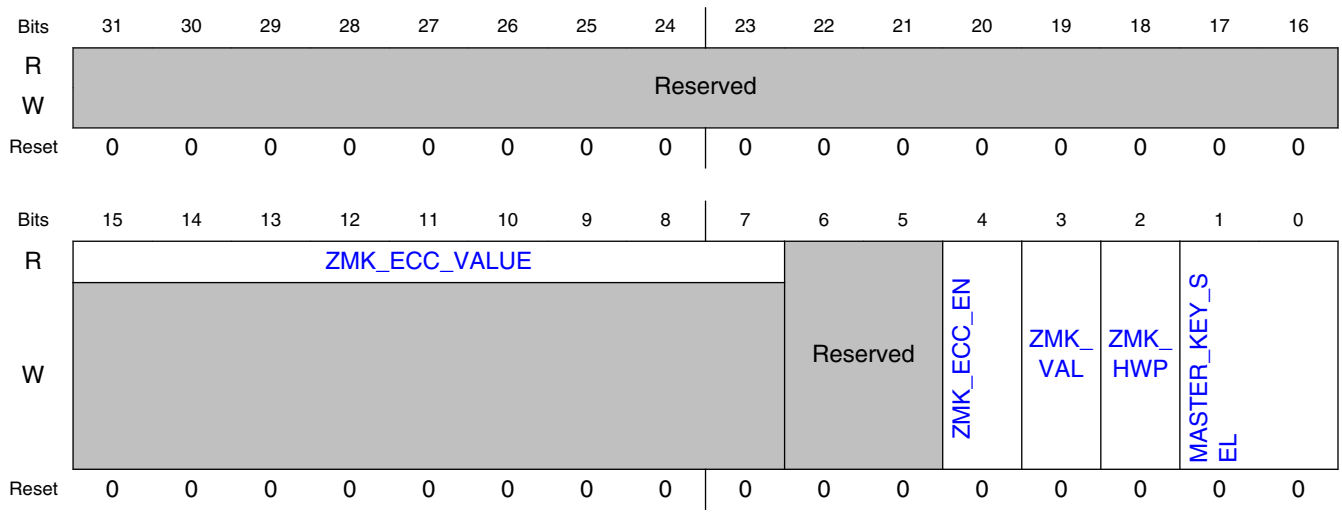
6.1.7.17.1 Address

Register	Offset
LPMKCR	3Ch

6.1.7.17.2 Function

The SNVS_LP Master Key Control Register contains the master keys configuration.

6.1.7.17.3 Diagram



6.1.7.17.4 Fields

Field	Function
31-16 —	
15-7 ZMK_ECC_VALUE	Zeroizable Master Key Error Correcting Code Value This field is automatically calculated and set when one is written into ZMK_ECC_EN bit of this register. This field cannot be programmed by software. It keeps the ECC value of the zeroizable master key, which allows checking that ZMK has not been corrupted/altered with time. Note that this ZMK ECC code is equivalent to the ECC bits encoded into the OTPMK value but for the ZMK, the ECC value is kept separate from the ZMK value. See Error code for the ZMK for details. Read restrictions similar to the ZMK Registers are applied to this field (see).
6-5 —	
4 ZMK_ECC_EN	Zeroizable Master Key Error Correcting Code Check Enable Writing one to this field automatically calculates and sets the ZMK ECC value in the ZMK_ECC_VALUE field of this register. When both ZMK value is valid (ZMK_VAL is set) and ZMK ECC check is enabled (ZMK_ECC_EN is set), the ZMK value is continuously checked for the valid ECC word. If the ZMK ECC word calculated every clock cycle does not match the one recorded in this register, the ZMK ECC Check Fail Violation is generated. This bit cannot be programmed when ZMK_WSL or ZMK_WHL bit is set. 0b - 1b -
3	Zeroizable Master Key Valid

Table continues on the next page...

Field	Function
ZMK_VAL	When set, the ZMK value can be selected by the master key control block for use by cryptographic modules. In hardware programming mode, hardware sets this bit when the ZMK provisioning is complete. In software programming mode, software should set this bit. This bit cannot be programmed when ZMK_WSL or ZMK_WHL bit is set. 0b - 1b -
2 ZMK_HWP	Zeroizable Master Key hardware Programming mode When set, only the hardware key programming mechanism can set the ZMK and software cannot read it. When not set, the ZMK can be programmed only by software. See for details. This bit cannot be programmed when ZMK_WSL or ZMK_WHL bit is set. 0b - 1b -
1-0 MASTER_KEY_SEL	Master Key Select These bits select the SNVS Master Key output when Master Key Select bits are enabled by MKS_EN bit in the HPCOMR . When MKS_EN bit is not set, the one time programmable master key is selected by default. This field cannot be programmed when MKS_SL(or the hard lock) bit is set. 0xb - 10b - Select zeroizable master key when MKS_EN bit is set . 11b - Select combined master key when MKS_EN bit is set .

6.1.7.18 SNVS_LP Security Violation Control (LPSVCR)

6.1.7.18.1 Address

Register	Offset
LPSVCR	40h

6.1.7.18.2 Function

The LP Security Violation Control Register configures security violation inputs. This register cannot be programmed when the LPSVCR Lock bit is set. Note that configurations of the security violation inputs in the HP section (HPSVCR) and LP section (LPSVCR Register) are independent and have different functionality.

6.1.7.18.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										SV_EN5	SV_EN4	SV_EN3	SV_EN2	SV_EN1	SV_EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.18.4 Fields

Field	Function
31-6 —	
5 SV_EN5	Security Violation 5 Enable This bit enables security violation input 5. When set, a security violation 5 causes an LP security violation, which clears LP sensitive data. 0b - 1b -
4 SV_EN4	Security Violation 4 Enable This bit enables security violation input 4. When set, a security violation 4 causes an LP security violation, which clears LP sensitive data. 0b - 1b -
3 SV_EN3	Security Violation 3 Enable This bit enables security violation input 3. When set, a security violation 3 causes an LP security violation, which clears LP sensitive data. 0b - 1b -
2 SV_EN2	Security Violation 2 Enable This bit enables security violation input 2. When set, a security violation 2 causes an LP security violation, which clears LP sensitive data. 0b - 1b -
1 SV_EN1	Security Violation 1 Enable This bit enables security violation input 1. When set, a security violation 1 causes an LP security violation, which clears LP sensitive data. 0b - 1b -

Table continues on the next page...

Field	Function
0 SV_EN0	Security Violation 0 Enable This bit enables security violation input 0. When set, a security violation 0 causes an LP security violation, which clears LP sensitive data. 0b - 1b -

6.1.7.19 SNVS_LP Tamper Glitch Filters Configuration (LPTGFCR)

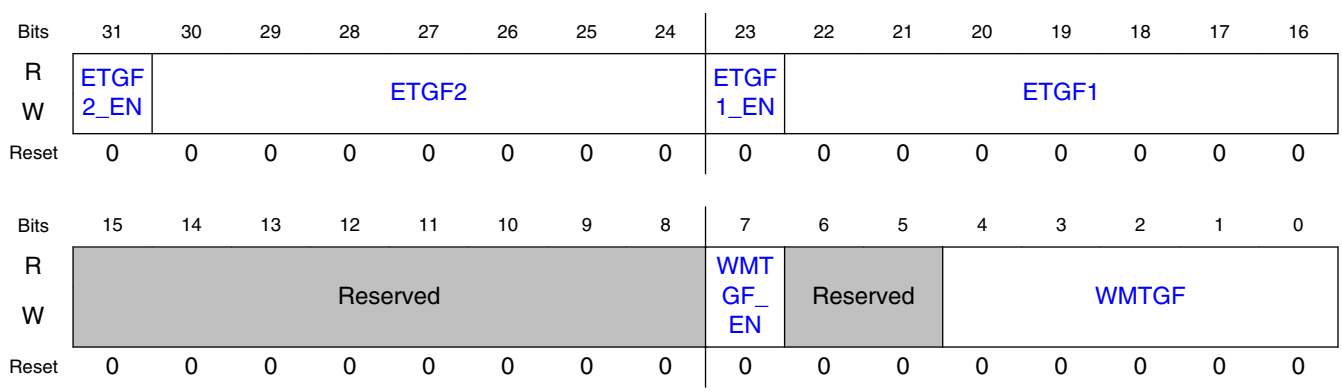
6.1.7.19.1 Address

Register	Offset
LPTGFCR	44h

6.1.7.19.2 Function

The SNVS_LP Tamper Glitch Filters Configuration Register is used to configure the glitch filters for the SNVS_LP tamper inputs. This register cannot be programmed when the LPTGFCR_SL or LPTGFCR_HL bit is set.

6.1.7.19.3 Diagram



6.1.7.19.4 Fields

Field	Function
31	External Tamper Glitch Filter 2 Enable

Table continues on the next page...

Field	Function
ETGF2_EN	When set, enables the external tamper glitch filter 2. 0b - 1b -
30-24 ETGF2	External Tamper Glitch Filter 2 Configures the length of the digital glitch filter for the external tamper 2 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 2 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF2 x 256), where ETGF2 = 0, ... , 127
23 ETGF1_EN	External Tamper Glitch Filter 1 Enable When set, enables the external tamper glitch filter 1. 0b - 1b -
22-16 ETGF1	External Tamper Glitch Filter 1 Configures the length of the digital glitch filter for the external tamper 1 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 1 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF1 x 256), where ETGF1 = 0, ... , 127
15-8 —	
7 WMTGF_EN	Wire-Mesh Tamper Glitch Filter Enable When set, enables the wire-mesh tamper glitch filter. Note that two wire-mesh tamper inputs (1 and 2) share this glitch filter. 0b - 1b -
6-5 —	
4-0 WMTGF	Wire-Mesh Tamper Glitch Filter Configures the length of the digital glitch filter for the wire-mesh tamper 1 and 2 pins between 1 and 63 SRTC clock cycles. Two wire-mesh tamper inputs share this glitch filter. Any assertion on wire-mesh tamper 1 or 2 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 1 + (WMTGF x 2), where WMTGF = 0, ... , 31

6.1.7.20 SNVS_LP Tamper Detectors Configuration (LPTDCR)

6.1.7.20.1 Address

Register	Offset
LPTDCR	48h

6.1.7.20.2 Function

The SNVS_LP Tamper Detectors Configuration Register is used to configure analog and digital tamper detector sources. This register cannot be programmed when LPTDCR is locked for write.

6.1.7.20.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			OSCB	Reserved	VRC			Reserved	HTDC			Reserved	LTDC		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	POR_OBSERV	PFD_OBSERV	Reserved	ET2P	ET1P	ET2_EN	ET1_EN	WMT2_EN	WMT1_EN	VT_EN	TT_EN	CT_EN	Reserved	MCR_EN	SRTCR_EN	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.20.4 Fields

Field	Function
31-29 —	
28 OSCB	Oscillator Bypass When OSCB=1 the osc_bypass signal is asserted. That signal asks SoC logic external to SNVS to bypass the SoC's normal SRTC clock source and drive the SRTC clock from an alternate source. 0b - 1b -
27 —	
26-24 VRC	Voltage Reference Configuration These configuration bits are wired as an output of the module.

Table continues on the next page...

Field	Function
23 —	
22-20 HTDC	High Temperature Detect Configuration These configuration bits are wired as an output of the module.
19 —	
18-16 LTDC	Low Temp Detect Configuration These configuration bits are wired as an output of the module.
15 POR_OBSERV	Power On Reset (POR) Observability Flop The asynchronous reset input of this flop is connected directly to the output of the POR analog circuitry (external to the SNVS. This flop can be used to detect brown-out voltage of the POR circuitry.
14 PFD_OBSERV	System Power Fail Detector (PFD) Observability Flop The asynchronous reset input of this flop is connected directly to the inverted output of the PFD analog circuitry (external to the SNVS block). This flop can be used to detect brown-out voltage of the PFD circuitry.
13 —	
12 ET2P	External Tampering 2 Polarity This bit is used to determine the polarity of external tamper 2. 0b - 1b -
11 ET1P	External Tampering 1 Polarity This bit is used to determine the polarity of external tamper 1. 0b - 1b -
10 ET2_EN	External Tampering 2 Enable When set, external tampering 2 detection generates an LP security violation. 0b - 1b -
9 ET1_EN	External Tampering 1 Enable When set, external tampering 1 detection generates an LP security violation. 0b - 1b -
8 WMT2_EN	Wire-Mesh Tampering 2 Enable When set, wire-mesh tampering 2 detection generates an LP security violation. 0b - 1b -
7 WMT1_EN	Wire-Mesh Tampering 1 Enable

Table continues on the next page...

Field	Function
	When set, wire-mesh tampering 1 detection generates an LP security violation. 0b - 1b -
6 VT_EN	Voltage Tamper Enable **NOTE:** Voltage Tamper Enable should be enabled 500 us after setting SCSC_SOSC_CTR [VOLT_TEMP_TAMPER_EN]. After enabling voltage tamper enable, clear SNVSLPSR [VTD] if set. When set, a voltage monitor tamper generates an LP security violation. 0b - 1b -
5 TT_EN	Temperature Tamper Enable When set, a temperature monitor tamper generates an LP security violation. **NOTE:** Temperature Tamper Enable should be enabled 500 us after setting SCSC_SOSC_CTR [VOLT_TEMP_TAMPER_EN]. After enabling temperature tamper enable, clear SNVS_LPSR [TTD] if set. 0b - 1b -
4 CT_EN	Clock Tamper Enable When set, a clock monitor tamper generates an LP security violation. 0b - 1b -
3 —	Reserved
2 MCR_EN	MC Rollover Enable When set, an MC Rollover event generates an LP security violation. 0b - 1b -
1 SRTCR_EN	SRTC Rollover Enable When set, an SRTC rollover event generates an LP security violation. 0b - 1b -
0 —	

6.1.7.21 SNVS_LP Status (LPSR)

6.1.7.21.1 Address

Register	Offset
LPSR	4Ch

6.1.7.21.2 Function

The SNVS_LP Status Register reflects the internal state and behavior of the SNVS_LP.

6.1.7.21.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LPS	LPNS	Reserved									SED	Reserved	SPO	EO	ESVD
W												w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved					ET2D	ET1D	WMT 2D	WMT 1D	VTD	TTD	CTD	PGD	MCR	SRTC R	LPTA
W						w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

6.1.7.21.4 Fields

Field	Function
31 LPS	LP Section is Secured Indicates that the LP section is provisioned/programmed in the secure or trusted state. The first write to the LP registers in secure or trusted state sets this bit. This bit can never be set together with the LPNS bit. When set the SNVS_LP section cannot be programmed and ZMK cannot be read in the non-secure state of the SSM. 0b - 1b -
30 LPNS	LP Section is Non-Secured Indicates that LP section was provisioned/programmed in the non-secure state. The first successful write to the LP Registers in non-secure state sets this bit. This bit can never be set together with the LPS bit. When set, the entire SNVS_LP section (all LP registers) are cleared upon an SSM transition from check to trusted state. 0b - 1b -
29-21 —	

Table continues on the next page...

Field	Function
20 SED	Scan Exit Detected 0b - 1b -
19 —	
18 SPO	Set Power Off The SPO bit is set when the set_pwr_off_irq interrupt is triggered, which happens when software writes a 1 to the TOP bit in the LPCR or when the power button is pressed longer than the configured debounce time. Writing to the SPO bit will clear the set_pwr_off_irq interrupt. 0b - 1b -
17 EO	Emergency Off This bit is set when a power off is requested. 0b - 1b -
16 ESVD	External Security Violation Detected Indicates that a security violation is detected on one of the HP security violation ports. The record of the port on which the violation has occurred can be found in the HP Security Violation Status Register. 0b - 1b -
15-11 —	
10 ET2D	0b - 1b -
9 ET1D	External Tampering 1 Detected 0b - 1b -
8 WMT2D	Wire-Mesh Tampering 2 Detected 0b - 1b -
7 WMT1D	Wire-Mesh Tampering 1 Detected 0b - 1b -
6 VTD	Voltage Tampering Detected 0b - 1b -
5 TTD	Temperature Tamper Detected 0b - 1b -
4 CTD	Clock Tampering Detected 0b - 1b -

Table continues on the next page...

Field	Function
3 PGD	Power Supply Glitch Detected 0 No power supply glitch. 1 Power supply glitch is detected.
2 MCR	Monotonic Counter Rollover 0b - 1b -
1 SRTCR	Secure Real Time Counter Rollover 0b - 1b -
0 LPTA	LP Time Alarm 0b - 1b -

6.1.7.22 SNVS_LP Secure Real Time Counter MSB (LPSRTCMR)

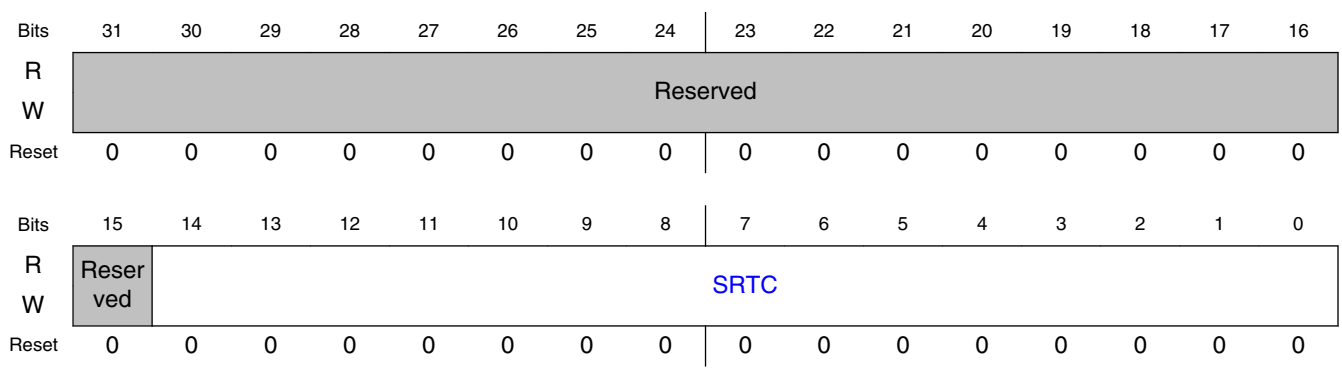
6.1.7.22.1 Address

Register	Offset
LPSRTCMR	50h

6.1.7.22.2 Function

The SNVS_LP Secure Real Time Counter MSB register contains the 15 most-significant bits of the LP Secure Real Time Counter.

6.1.7.22.3 Diagram



6.1.7.22.4 Fields

Field	Function
31-15 —	
14-0 SRTC	LP Secure Real Time Counter The most-significant 15 bits of the SRTC. This register can be programmed only when SRTC is not active and not locked, meaning the SRTC_ENV, SRTC_SL, and SRTC_HL bits are not set.

6.1.7.23 SNVS_LP Secure Real Time Counter LSB (LPSRTCLR)

6.1.7.23.1 Address

Register	Offset
LPSRTCLR	54h

6.1.7.23.2 Function

The SNVS_LP Secure Real Time Counter LSB register contains the 32 least-significant bits of the secure real time counter.

6.1.7.23.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SRTC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SRTC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.23.4 Fields

Field	Function
31-0 SRTC	LP Secure Real Time Counter least-significant 32 bits This register can be programmed only when SRTC is not active and not locked, meaning the SRTC_ENV, SRTC_SL, and SRTC_HL bits are not set.

6.1.7.24 SNVS_LP Time Alarm (LPTAR)

6.1.7.24.1 Address

Register	Offset
LPTAR	58h

6.1.7.24.2 Function

The SNVS_LP Time Alarm register contains the 32-bit LP Time Alarm value.

6.1.7.24.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LPTA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LPTA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.24.4 Fields

Field	Function
31-0 LPTA	LP Time Alarm This register can be programmed only when the LP time alarm is disabled (LPTA_EN bit is not set).

6.1.7.25 SNVS_LP Secure Monotonic Counter MSB (LPSMCMR)

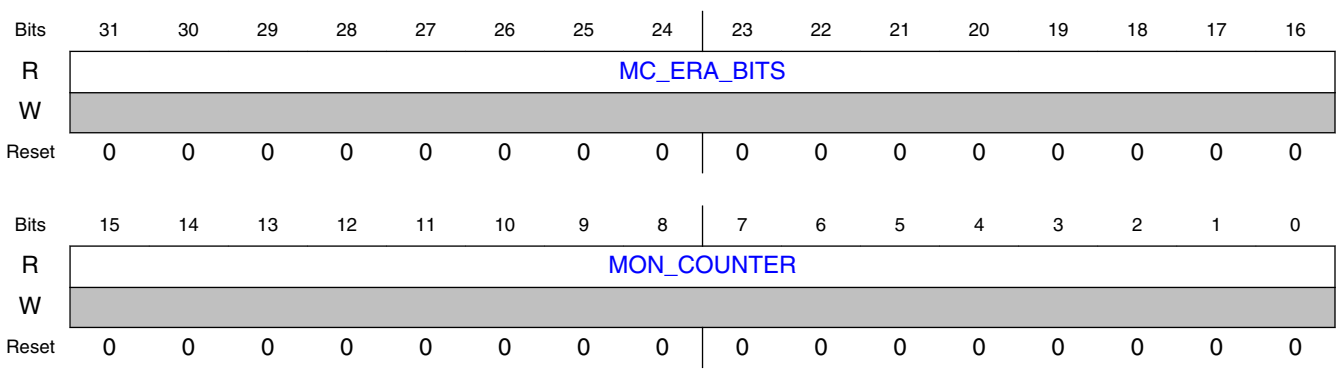
6.1.7.25.1 Address

Register	Offset
LPSMCMR	5Ch

6.1.7.25.2 Function

The SNVS_LP Secure Monotonic Counter MSB Register contains the monotonic counter era bits and the most-significant 16 bits of the monotonic counter. The monotonic counter is incremented by one if there is a write command to the LPSMCMR or LPSMCLR register.

6.1.7.25.3 Diagram



6.1.7.25.4 Fields

Field	Function
31-16 MC_ERA_BITS	Monotonic Counter Era Bits These bits are inputs to the module and typically connect to fuses.
15-0 MON_COUNTER	Monotonic Counter most-significant 16 Bits The MC is incremented by one when: • A write transaction to the LPSMCMR or LPSMCLR register is detected. • The MC_ENV bit is set. • MC_SL and MC_HL bits are not set.

6.1.7.26 SNVS_LP Secure Monotonic Counter LSB (LPSMCLR)

6.1.7.26.1 Address

Register	Offset
LPSMCLR	60h

6.1.7.26.2 Function

The SNVS_LP Secure Monotonic Counter LSB Register contains the 32 least-significant bits of the monotonic counter. The MC is incremented by one if there is a write command to the LPSMCMR or LPSMCLR register.

6.1.7.26.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MON_COUNTER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MON_COUNTER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.26.4 Fields

Field	Function
31-0 MON_COUNTER	Monotonic Counter bits The MC is incremented by one when: • A write transaction to the LPSMCMR or LPSMCLR Register is detected. • The MC_ENV bit is set. • MC_SL and MC_HL bits are not set.

6.1.7.27 SNVS_LP Power Glitch Detector (LPPGDR)

6.1.7.27.1 Address

Register	Offset
LPPGDR	64h

6.1.7.27.2 Function

The SNVS_LP Power Glitch Detector Register is a 32-bit read/write register that is used for storing the power glitch detector value, as described in .

6.1.7.27.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PGD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PGD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.27.4 Fields

Field	Function
31-0 PGD	Power Glitch Detector Value

6.1.7.28 SNVS_LP General Purpose 0 (alias) (LPGPR0_alias)

6.1.7.28.1 Address

Register	Offset
LPGPR0_alias	68h

6.1.7.28.2 Function

The SNVS_LP General Purpose Register is a 128-bit read/write register located in the low power domain, which can be used by any application for retaining data during an SoC power-down mode. The register is accessed as four 32-bit registers located in successive word addresses starting at offset 90h. For backward compatibility with earlier versions of SNVS LPGPR0 is aliased at its original offset of 68h.The GPR will be automatically zeroized when a tamper event occurs, unless GPR zeroization is disabled via the GPR_Z_DIS bit in the LP Control Register.

6.1.7.28.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.28.4 Fields

Field	Function
31-0 GPR	General Purpose Register When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

6.1.7.29 SNVS_LP Zeroizable Master Key (LPZMKRa)

6.1.7.29.1 Address

For a = 0 to 31:

Register	Offset
LPZMKRa	6Ch + (a × 1h)

6.1.7.29.2 Function

The SNVS_LP Zeroizable Master Key Registers contain the 256-bit zeroizable master key value. These registers are programmable as follows:

- When ZMK write lock bit is set, they cannot be programmed.
- When ZMK_HWP is not set, they are in software programming mode and can be programmed only by software.
- When ZMK_HWP is set, they are in hardware programming mode and can be programmed only by hardware.

These registers cannot be read by software when the ZMK_HWP or ZMK read lock bit is set.

6.1.7.29.3 Diagram

Bits	7	6	5	4	3	2	1	0
R W	ZMK							
Reset	0	0	0	0	0	0	0	0

6.1.7.29.4 Fields

Field	Function
7-0 ZMK	Zeroizable Master Key Each of these registers contains part of the 256-bit ZMK value. When accessed via the register bus this should be treated as a byte array with the first byte in offset 6Ch (although the register must be accessed as eight 32-bit words).

6.1.7.30 SNVS_LP General Purposes 0 .. 3 (LPGPR0_3a)

6.1.7.30.1 Address

Register	Offset
LPGPR0_30	90h
LPGPR0_31	94h
LPGPR0_32	98h
LPGPR0_33	9Ch

6.1.7.30.2 Function

The SNVS_LP General Purpose Register is a 128-bit read/write register located in the low power domain, which can be used by any application for retaining data during an SoC power-down mode. The register is accessed as four 32-bit registers located in successive word addresses starting at offset 90h. For backward compatibility with earlier versions of SNVS LPGPR0 is aliased at its original offset of 68h. The GPR will be automatically zeroized when a tamper event occurs, unless GPR zeroization is disabled via the GPR_Z_DIS bit in the LP Control Register.

6.1.7.30.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.30.4 Fields

Field	Function
31-0 GPR	General Purpose Register When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

6.1.7.31 SNVS_LP Tamper Detectors Config 2 (LPTDC2R)

6.1.7.31.1 Address

Register	Offset
LPTDC2R	A0h

6.1.7.31.2 Function

The SNVS_LP Tamper Detectors Configuration 2 Register is used to configure digital external tamper sources. This register cannot be programmed when LPTDCR is locked for write.

6.1.7.31.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								ET10P	ET9P	ET8P	ET7P	ET6P	ET5P	ET4P	ET3P
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								ET10_EN	ET9_EN	ET8_EN	ET7_EN	ET6_EN	ET5_EN	ET4_EN	ET3_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.31.4 Fields

Field	Function
31-24 —	
23 ET10P	External Tampering 10 Polarity This bit is used to determine the polarity of external tamper 10. 0b - 1b -
22 ET9P	External Tampering 9 Polarity This bit is used to determine the polarity of external tamper 9. 0b - 1b -
21 ET8P	External Tampering 8 Polarity This bit is used to determine the polarity of external tamper 8. 0b - 1b -
20 ET7P	External Tampering 7 Polarity This bit is used to determine the polarity of external tamper 7. 0b - 1b -
19	External Tampering 6 Polarity

Table continues on the next page...

Field	Function
ET6P	This bit is used to determine the polarity of external tamper 6. 0b - 1b -
18 ET5P	External Tampering 5 Polarity This bit is used to determine the polarity of external tamper 5. 0b - 1b -
17 ET4P	External Tampering 4 Polarity This bit is used to determine the polarity of external tamper 4. 0b - 1b -
16 ET3P	External Tampering 3 Polarity This bit is used to determine the polarity of external tamper 3. 0b - 1b -
15-8 —	
7 ET10_EN	External Tampering 10 Enable When set, external tampering 10 detection generates an LP security violation. 0b - 1b -
6 ET9_EN	External Tampering 9 Enable When set, external tampering 9 detection generates an LP security violation. 0b - 1b -
5 ET8_EN	External Tampering 8 Enable When set, external tampering 8 detection generates an LP security violation. 0b - 1b -
4 ET7_EN	External Tampering 7 Enable When set, external tampering 7 detection generates an LP security violation. 0b - 1b -
3 ET6_EN	External Tampering 6 Enable When set, external tampering 6 detection generates an LP security violation. 0b - 1b -
2 ET5_EN	External Tampering 5 Enable When set, external tampering 5 detection generates an LP security violation.

Table continues on the next page...

Field	Function
	0b - 1b -
1 ET4_EN	External Tampering 4 Enable When set, external tampering 4 detection generates an LP security violation. 0b - 1b -
0 ET3_EN	External Tampering 3 Enable When set, external tampering 3 detection generates an LP security violation. 0b - 1b -

6.1.7.32 SNVS_LP Tamper Detectors Status (LPTDSR)

6.1.7.32.1 Address

Register	Offset
LPTDSR	A4h

6.1.7.32.2 Function

The SNVS_LP Tamper Detectors Status Register reflects the status of the SNVS_LP external tampers 3-10.

6.1.7.32.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								ET10D	ET9D	ET8D	ET7D	ET6D	ET5D	ET4D	ET3D
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.32.4 Fields

Field	Function
31-8 —	
7 ET10D	External Tampering 10 Detected 0b - 1b -
6 ET9D	External Tampering 9 Enable When set, external tampering 9 detection generates an LP security violation. 0b - 1b -
5 ET8D	External Tampering 8 Detected 0b - 1b -
4 ET7D	External Tampering 7 Detected 0b - 1b -
3 ET6D	External Tampering 6 Detected 0b - 1b -
2 ET5D	External Tampering 5 Detected 0b - 1b -
1 ET4D	External Tampering 4 Detected 0b - 1b -
0 ET3D	External Tampering 3 Detected 0b - 1b -

6.1.7.33 SNVS_LP Tamper Glitch Filter 1 Configuration (LPTGF1CR)

6.1.7.33.1 Address

Register	Offset
LPTGF1CR	A8h

6.1.7.33.2 Function

The SNVS_LP Tamper Glitch Filters Configuration Register is used to configure the glitch filters for the SNVS_LP external tamper inputs 3-6. This register cannot be programmed when the LPTGFCR_SL or LPTGFCR_HL bit is set.

6.1.7.33.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ETGF6_EN	ETGF6							ETGF5_EN	ETGF5						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ETGF4_EN	ETGF4							ETGF3_EN	ETGF3						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.33.4 Fields

Field	Function
31 ETGF6_EN	External Tamper Glitch Filter 6 Enable When set, enables the external tamper glitch filter 6. 0b - 1b -
30-24 ETGF6	External Tamper Glitch Filter 6 Configures the length of the digital glitch filter for the external tamper 6 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 6 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF6 x 256), where ETGF6 = 0, ... , 127
23 ETGF5_EN	External Tamper Glitch Filter 5 Enable When set, enables the external tamper glitch filter 5. 0b - 1b -
22-16 ETGF5	External Tamper Glitch Filter 5 Configures the length of the digital glitch filter for the external tamper 5 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 5 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF5 x 256), where ETGF5 = 0, ... , 127

Table continues on the next page...

Field	Function
15 ETGF4_EN	External Tamper Glitch Filter 4 Enable When set, enables the external tamper glitch filter 4. 0b - 1b -
14-8 ETGF4	External Tamper Glitch Filter 4 Configures the length of the digital glitch filter for the external tamper 4 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 4 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF4 x 256), where ETGF4 = 0, ... , 127
7 ETGF3_EN	External Tamper Glitch Filter 3 Enable When set, enables the external tamper glitch filter 3. 0b - 1b -
6-0 ETGF3	External Tamper Glitch Filter 3 Configures the length of the digital glitch filter for the external tamper 3 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 3 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF3 x 256), where ETGF3 = 0, ... , 127

6.1.7.34 SNVS_LP Tamper Glitch Filter 2 Configuration (LPTGF2CR)

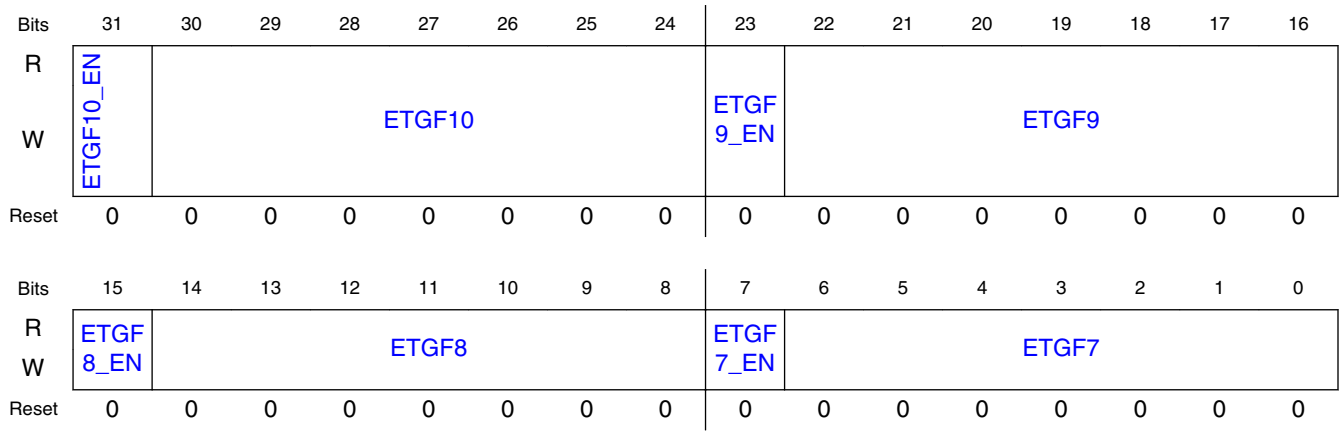
6.1.7.34.1 Address

Register	Offset
LPTGF2CR	ACh

6.1.7.34.2 Function

The SNVS_LP Tamper Glitch Filters Configuration Register is used to configure the glitch filters for the SNVS_LP> external tamper inputs 7-10. This register cannot be programmed when the LPTGFCR_SL or LPTGFCR_HL bit is set.

6.1.7.34.3 Diagram



6.1.7.34.4 Fields

Field	Function
31 ETGF10_EN	External Tamper Glitch Filter 10 Enable When set, enables the external tamper glitch filter 10. 0b - 1b -
30-24 ETGF10	External Tamper Glitch Filter 10 Configures the length of the digital glitch filter for the external tamper 10 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 6 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF10 x 256), where ETGF10 = 0, ... , 127
23 ETGF9_EN	External Tamper Glitch Filter 9 Enable When set, enables the external tamper glitch filter 9. 0b - 1b -
22-16 ETGF9	External Tamper Glitch Filter 9 Configures the length of the digital glitch filter for the external tamper 9 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 5 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF9 x 256), where ETGF9 = 0, ... , 127
15 ETGF8_EN	External Tamper Glitch Filter 8 Enable When set, enables the external tamper glitch filter 8. 0b - 1b -

Table continues on the next page...

Field	Function
14-8 ETGF8	External Tamper Glitch Filter 8 Configures the length of the digital glitch filter for the external tamper 8 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 8 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF8 x 256), where ETGF8 = 0, ... , 127
7 ETGF7_EN	External Tamper Glitch Filter 7 Enable When set, enables the external tamper glitch filter 7. 0b - 1b -
6-0 ETGF7	External Tamper Glitch Filter 7 Configures the length of the digital glitch filter for the external tamper 7 pin between 128 and 32640 SRTC clock cycles. Any assertion on external tamper 7 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF7 x 256), where ETGF7 = 0, ... , 127

6.1.7.35 SNVS_LP Active Tamper 1 Configuration (LPAT1CR)

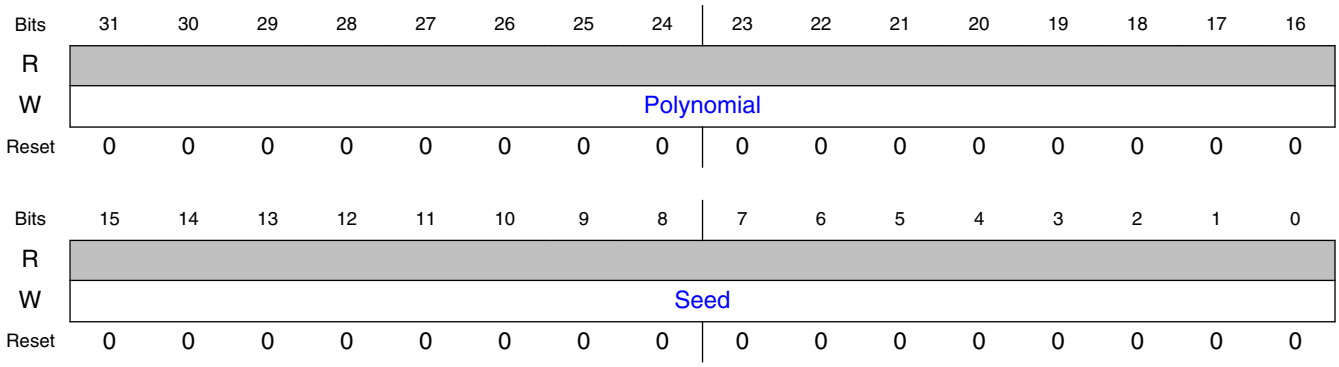
6.1.7.35.1 Address

Register	Offset
LPAT1CR	C0h

6.1.7.35.2 Function

The SNVS_LP Active Tamper 1 Configuration Register is used to configure the LFSR which is used for the SNVS_LP active tamper outputs. This register cannot be programmed when the LPAT1EN bit is set.

6.1.7.35.3 Diagram



6.1.7.35.4 Fields

Field	Function
31-16 Polynomial	Active Tamper 1 Polynomial Default Polynomial is 8400h.
15-0 Seed	Active Tamper 1 Initial Seed Default Seed is 1111h.

6.1.7.36 SNVS_LP Active Tamper 2 Configuration (LPAT2CR)

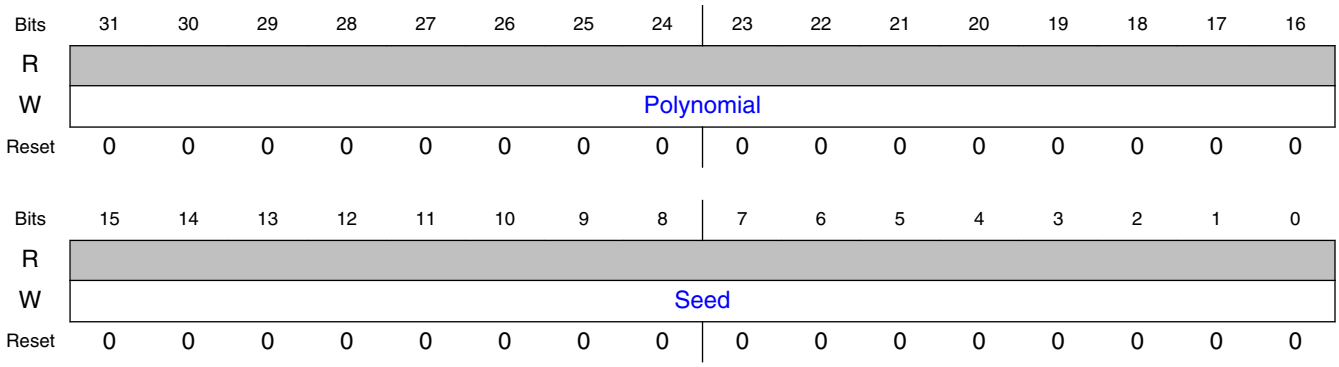
6.1.7.36.1 Address

Register	Offset
LPAT2CR	C4h

6.1.7.36.2 Function

The SNVS_LP Active Tamper 2 Configuration Register is used to configure the LFSR which is used for the SNVS_LP active tamper outputs. This register cannot be programmed when the LPAT2EN bit is set.

6.1.7.36.3 Diagram



6.1.7.36.4 Fields

Field	Function
31-16 Polynomial	Active Tamper 2 Polynomial Default Polynomial is 9C00h.
15-0 Seed	Active Tamper 2 Initial Seed Default Seed is 2222h.

6.1.7.37 SNVS_LP Active Tamper 3 Configuration (LPAT3CR)

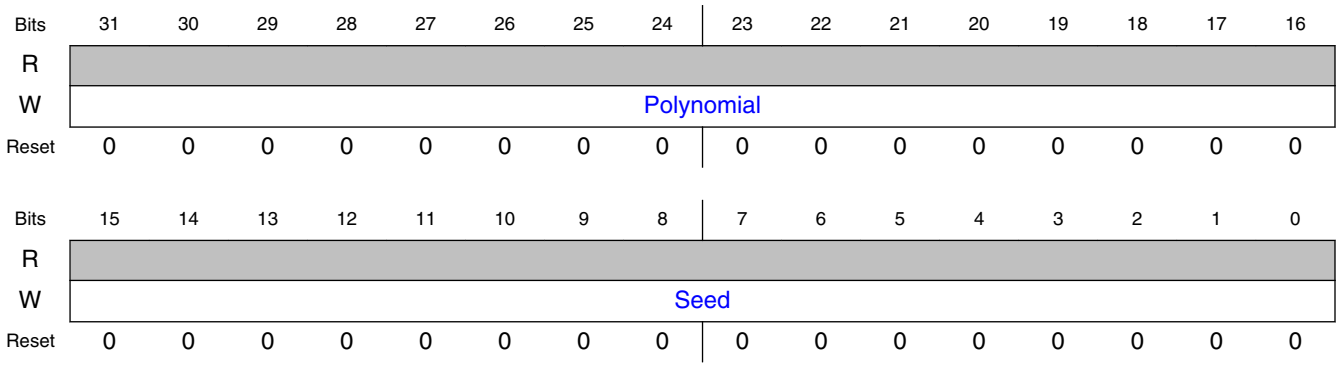
6.1.7.37.1 Address

Register	Offset
LPAT3CR	C8h

6.1.7.37.2 Function

The SNVS_LP Active Tamper 3 Configuration Register is used to configure the LFSR which is used for the SNVS_LP active tamper outputs. This register cannot be programmed when the LPAT3EN bit is set.

6.1.7.37.3 Diagram



6.1.7.37.4 Fields

Field	Function
31-16 Polynomial	Active Tamper 3 Polynomial Default Polynomial is CA00h.
15-0 Seed	Active Tamper 3 Initial Seed Default Seed is 3333h.

6.1.7.38 SNVS_LP Active Tamper 4 Configuration (LPAT4CR)

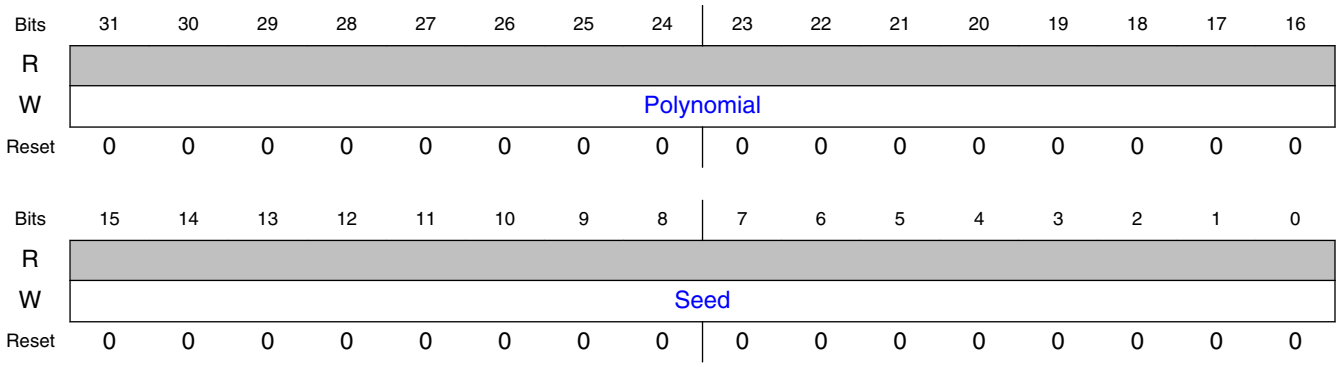
6.1.7.38.1 Address

Register	Offset
LPAT4CR	CCh

6.1.7.38.2 Function

The SNVS_LP Active Tamper 4 Configuration Register is used to configure the LFSR which is used for the SNVS_LP active tamper outputs. This register cannot be programmed when the LPAT4EN bit is set.

6.1.7.38.3 Diagram



6.1.7.38.4 Fields

Field	Function
31-16 Polynomial	Active Tamper 4 Polynomial Default Polynomial is 8580h.
15-0 Seed	Active Tamper 4 Initial Seed Default Seed is 4444h.

6.1.7.39 SNVS_LP Active Tamper 5 Configuration (LPAT5CR)

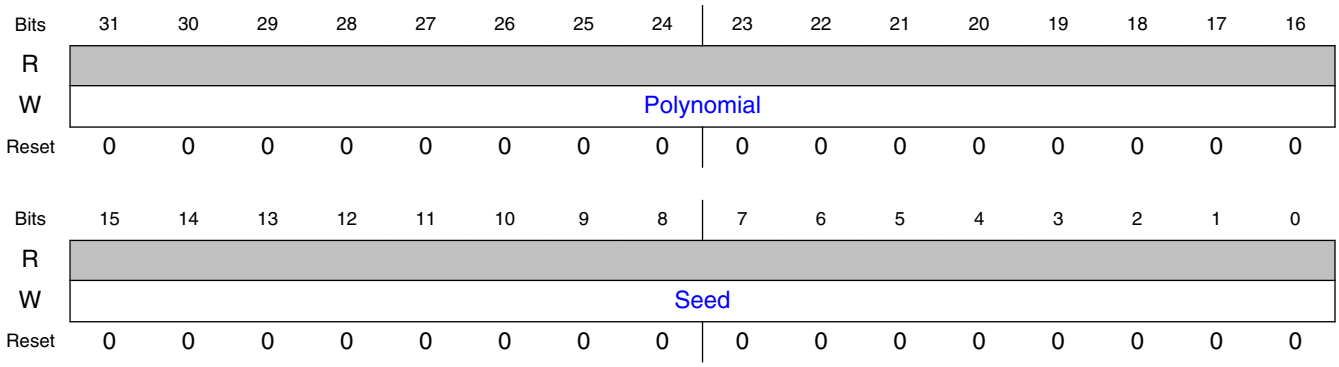
6.1.7.39.1 Address

Register	Offset
LPAT5CR	D0h

6.1.7.39.2 Function

The SNVS_LP Active Tamper 5 Configuration Register is used to configure the LFSR which is used for the SNVS_LP active tamper outputs. This register cannot be programmed when the LPAT5EN bit is set.

6.1.7.39.3 Diagram



6.1.7.39.4 Fields

Field	Function
31-16 Polynomial	Active Tamper 5 Polynomial Default Polynomial is A840h.
15-0 Seed	Active Tamper 5 Initial Seed Default Seed is 5555h.

6.1.7.40 SNVS_LP Active Tamper Control (LPATCTLR)

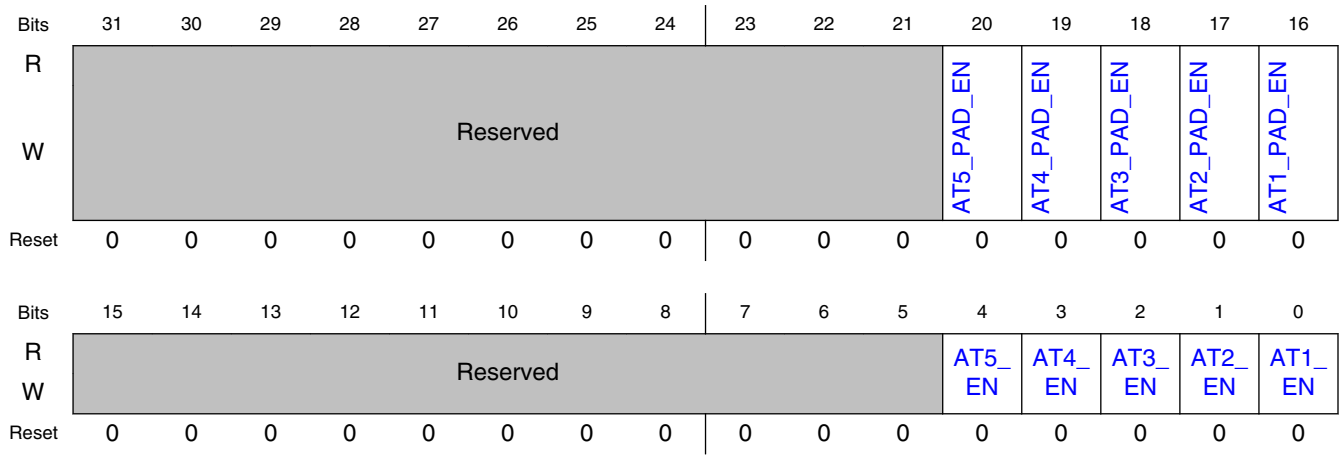
6.1.7.40.1 Address

Register	Offset
LPATCTLR	E0h

6.1.7.40.2 Function

The SNVS_LP Active Tamper Control Register is used to enable the LFSRs which is used for the SNVS_LP active tamper outputs. It is also used to control external pads to enable for input or output.

6.1.7.40.3 Diagram



6.1.7.40.4 Fields

Field	Function
31-21 —	
20 AT5_PAD_EN	Active Tamper 5 Pad Out Enable When set, enables the Active Tamper 5 external pad. 0b - 1b -
19 AT4_PAD_EN	Active Tamper 4 Pad Out Enable When set, enables the Active Tamper 4 external pad. 0b - 1b -
18 AT3_PAD_EN	Active Tamper 3 Pad Out Enable When set, enables the Active Tamper 3 external pad. 0b - 1b -
17 AT2_PAD_EN	Active Tamper 2 Pad Out Enable When set, enables the Active Tamper 2 external pad. 0b - 1b -
16 AT1_PAD_EN	Active Tamper 1 Pad Out Enable When set, enables the Active Tamper 1 external pad. 0b - 1b -
15-5	

Table continues on the next page...

Field	Function
—	
4 AT5_EN	Active Tamper 5 Enable When set, enables the Active Tamper 5 LFSR. 0b - 1b -
3 AT4_EN	Active Tamper 4 Enable When set, enables the Active Tamper 4 LFSR. 0b - 1b -
2 AT3_EN	Active Tamper 3 Enable When set, enables the Active Tamper 3 LFSR. 0b - 1b -
1 AT2_EN	Active Tamper 2 Enable When set, enables the Active Tamper 2 LFSR. 0b - 1b -
0 AT1_EN	Active Tamper 1 Enable When set, enables the Active Tamper 1 LFSR. 0b - 1b -

6.1.7.41 SNVS_LP Active Tamper Clock Control (LPATCLKR)

6.1.7.41.1 Address

Register	Offset
LPATCLKR	E4h

6.1.7.41.2 Function

The SNVS_LP Active Tamper Clock Control Register is used to define what frequency the LFSRs are run at for the SNVS_LP active tamper outputs. The Active Tamper clocks are based on the SRTC. The SRTC must be enabled for Active Tamper to work. The clock control fields are not writeable once an LFSR is enabled.

6.1.7.41.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved														AT5_CLK_CTL	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		AT4_CLK_CTL		Reserved		AT3_CLK_CTL		Reserved		AT2_CLK_CTL		Reserved		AT1_CLK_CTL	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.41.4 Fields

Field	Function
31-18 —	
17-16 AT5_CLK_CTL	Active Tamper 5 Clock Control 00: 16hz 01: 8hz 10: 4hz 11: 2hz
15-14 —	
13-12 AT4_CLK_CTL	Active Tamper 4 Clock Control 00: 16hz 01: 8hz 10: 4hz 11: 2hz
11-10 —	
9-8 AT3_CLK_CTL	Active Tamper 3 Clock Control 00: 16hz 01: 8hz 10: 4hz 11: 2hz
7-6 —	
5-4	Active Tamper 2 Clock Control

Table continues on the next page...

Field	Function
AT2_CLK_CTL	00: 16hz 01: 8hz 10: 4hz 11: 2hz
3-2 —	
1-0 AT1_CLK_CTL	Active Tamper 1 Clock Control 00: 16hz 01: 8hz 10: 4hz 11: 2hz

6.1.7.42 SNVS_LP Active Tamper Routing Control 1 (LPATRC1R)

6.1.7.42.1 Address

Register	Offset
LPATRC1R	E8h

6.1.7.42.2 Function

The SNVS_LP Active Tamper Routing Control Register routes an active tamper compare value to the proper external tamper detector.

6.1.7.42.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved	ET8RCTL			Reserved	ET7RCTL			Reserved	ET6RCTL			Reserved	ET5RCTL		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved	ET4RCTL			Reserved	ET3RCTL			Reserved	ET2RCTL			Reserved	ET1RCTL		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.42.4 Fields

Field	Function
31 —	
30-28 ET8RCTL	External Tamper 8 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
27 —	
26-24 ET7RCTL	External Tamper 7 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
23 —	
22-20 ET6RCTL	External Tamper 6 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
19 —	
18-16 ET5RCTL	External Tamper 5 Routing Control Any undefined selection will be routed to passive. 000: Passive Input

Table continues on the next page...

Field	Function
	001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
15 —	
14-12 ET4RCTL	External Tamper 4 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
11 —	
10-8 ET3RCTL	External Tamper 3 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
7 —	
6-4 ET2RCTL	External Tamper 2 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
3 —	
2-0	External Tamper 1 Routing Control

Field	Function
ET1RCTL	Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5

6.1.7.43 SNVS_LP Active Tamper Routing Control 2 (LPATRC2R)

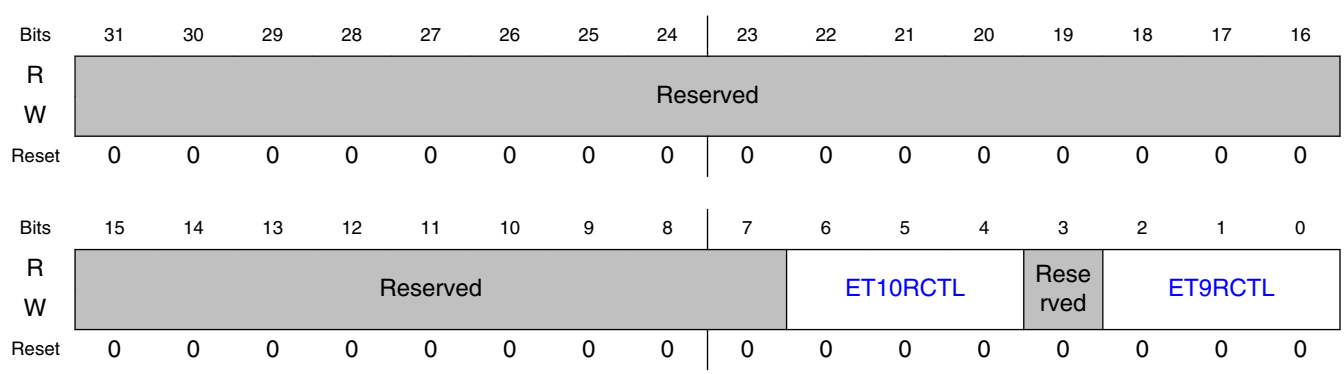
6.1.7.43.1 Address

Register	Offset
LPATRC2R	ECh

6.1.7.43.2 Function

The SNVS_LP Active Tamper Routing Control Register routes an active tamper compare value to the proper external tamper detector.

6.1.7.43.3 Diagram



6.1.7.43.4 Fields

Field	Function
31-7	

Table continues on the next page...

Field	Function
—	
6-4 ET10RCTL	External Tamper 10 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5
3 —	
2-0 ET9RCTL	External Tamper 9 Routing Control Any undefined selection will be routed to passive. 000: Passive Input 001: Active Tamper 1 010: Active Tamper 2 011: Active Tamper 3 100: Active Tamper 4 101: Active Tamper 5

6.1.7.44 SNVS_HP Version ID 1 (HPVIDR1)

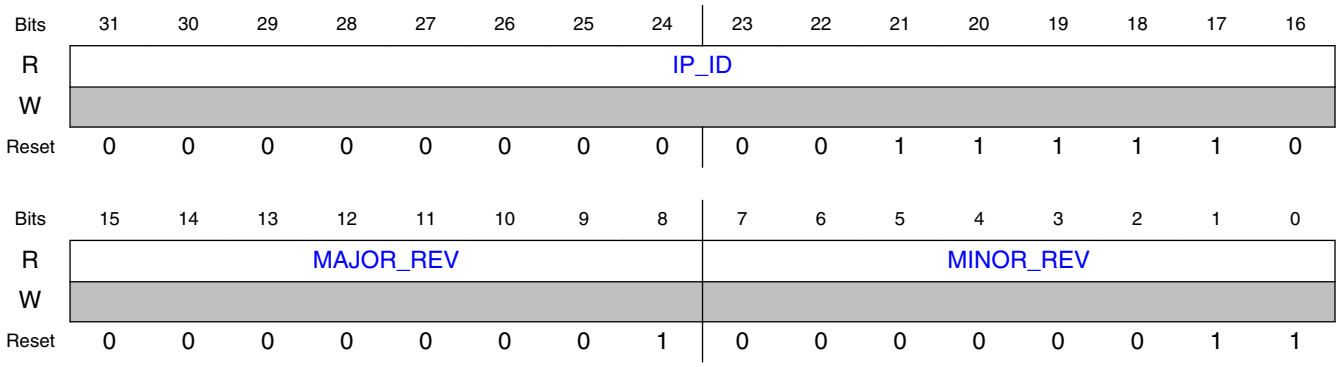
6.1.7.44.1 Address

Register	Offset
HPVIDR1	BF8h

6.1.7.44.2 Function

The SNVS_HP Version ID Register 1 is a read-only register that contains the current version of the SNVS. The version consists of a module ID, a major version number, and a minor version number.

6.1.7.44.3 Diagram



6.1.7.44.4 Fields

Field	Function
31-16 IP_ID	SNVS block ID
15-8 MAJOR_REV	SNVS block major version number
7-0 MINOR_REV	SNVS block minor version number

6.1.7.45 SNVS_HP Version ID 2 (HPVIDR2)

6.1.7.45.1 Address

Register	Offset
HPVIDR2	BFCh

6.1.7.45.2 Function

The SNVS_HP Version ID Register 2 is a read-only register that indicates the current version of the SNVS. Version ID register 2 consists of the following fields: integration options, ECO revision, and configuration options.

6.1.7.45.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IP_ERA								INTG_OPT							
W																
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ECO_REV								CONFIG_OPT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.1.7.45.4 Fields

Field	Function
31-24 IP_ERA	IP Era 00h - Era 1 or 2 03h - Era 3 04h - Era 4 05h - Era 5
23-16 INTG_OPT	SNVS Integration Options
15-8 ECO_REV	SNVS ECO Revision
7-0 CONFIG_OPT	SNVS Configuration Options

6.2 System Reset Controller (SRC)

6.2.1 SRC Overview

The System Reset Controller (SRC) controls the reset and boot operation of the SoC.

It is responsible for the generation of all reset signals and boot decoding.

The reset controller determines the source and the type of reset, such as POR, COLD, and performs the necessary reset qualification and stretching sequences. Based on the type of reset, the reset logic generates the reset sequence for the entire IC. Whenever the chip is powered on, the reset is issued through SRC_ONOFF signal and the entire chip is reset.

6.2.1.1 Features

The SRC includes the following features.

- Receives and handles the resets from all the reset sources
- Resets the appropriate domains based upon the resets sources and the nature of the reset
- Latches the SRC_BOOT_MODE pins and common configuration signals from the internal fuse

6.2.2 External Signals

The following table describes the external signals of SRC.

Table 6-6. SRC External Signals

Signal	Description	Pad	Mode	Direction
SRC_BOOT_CFG00	Boot configuration signal	LCD1_DATA00	ALT6	I
SRC_BOOT_CFG01	Boot configuration signal	LCD1_DATA01	ALT6	I
SRC_BOOT_CFG02	Boot configuration signal	LCD1_DATA02	ALT6	I
SRC_BOOT_CFG03	Boot configuration signal	LCD1_DATA03	ALT6	I
SRC_BOOT_CFG04	Boot configuration signal	LCD1_DATA04	ALT6	I
SRC_BOOT_CFG05	Boot configuration signal	LCD1_DATA05	ALT6	I
SRC_BOOT_CFG06	Boot configuration signal	LCD1_DATA06	ALT6	I
SRC_BOOT_CFG07	Boot configuration signal	LCD1_DATA07	ALT6	I
SRC_BOOT_CFG08	Boot configuration signal	LCD1_DATA08	ALT6	I
SRC_BOOT_CFG09	Boot configuration signal	LCD1_DATA09	ALT6	I
SRC_BOOT_CFG10	Boot configuration signal	LCD1_DATA10	ALT6	I
SRC_BOOT_CFG11	Boot configuration signal	LCD1_DATA11	ALT6	I
SRC_BOOT_CFG12	Boot configuration signal	LCD1_DATA12	ALT6	I
SRC_BOOT_CFG13	Boot configuration signal	LCD1_DATA13	ALT6	I
SRC_BOOT_CFG14	Boot configuration signal	LCD1_DATA14	ALT6	I
SRC_BOOT_CFG15	Boot configuration signal	LCD1_DATA15	ALT6	I
SRC_BOOT_CFG16	Boot configuration signal	LCD1_DATA16	ALT6	I
SRC_BOOT_CFG17	Boot configuration signal	LCD1_DATA17	ALT6	I
SRC_BOOT_CFG18	Boot configuration signal	LCD1_DATA18	ALT6	I
SRC_BOOT_CFG19	Boot configuration signal	LCD1_DATA19	ALT6	I
SRC_BOOT_MODE0	Boot mode signal	BOOT_MODE0	No muxing (ALT0)	I
SRC_BOOT_MODE1	Boot mode signal	BOOT_MODE1	No muxing (ALT0)	I

Table continues on the next page...

Table 6-6. SRC External Signals (continued)

Signal	Description	Pad	Mode	Direction
SRC_CA7_RESET0_B	CA7 Reset signal	SAI1_RXFS	ALT7	
SRC_CA7_RESET1_B	CA7 Reset signal	SAI1_RXC	ALT7	
SRC_ONOFF	ONOFF signal	ONOFF	No muxing	I
SRC_POR_B	Power on reset signal	POR_B	No muxing (ALT0)	I
SRC_SYSTEM_RESET	System reset signal	SAI1_TXD	ALT7	O

6.2.3 Clocks

The table found here describes the clock sources for SRC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 6-7. SRC Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

6.2.4 Top-level resets, power-up sequence and external supply integration

Information found here defines chip resets, power-up sequence, and external supply integration.

6.2.4.1 Reset and Power-up Flow

The chip presumes the following reset and power-up flow:

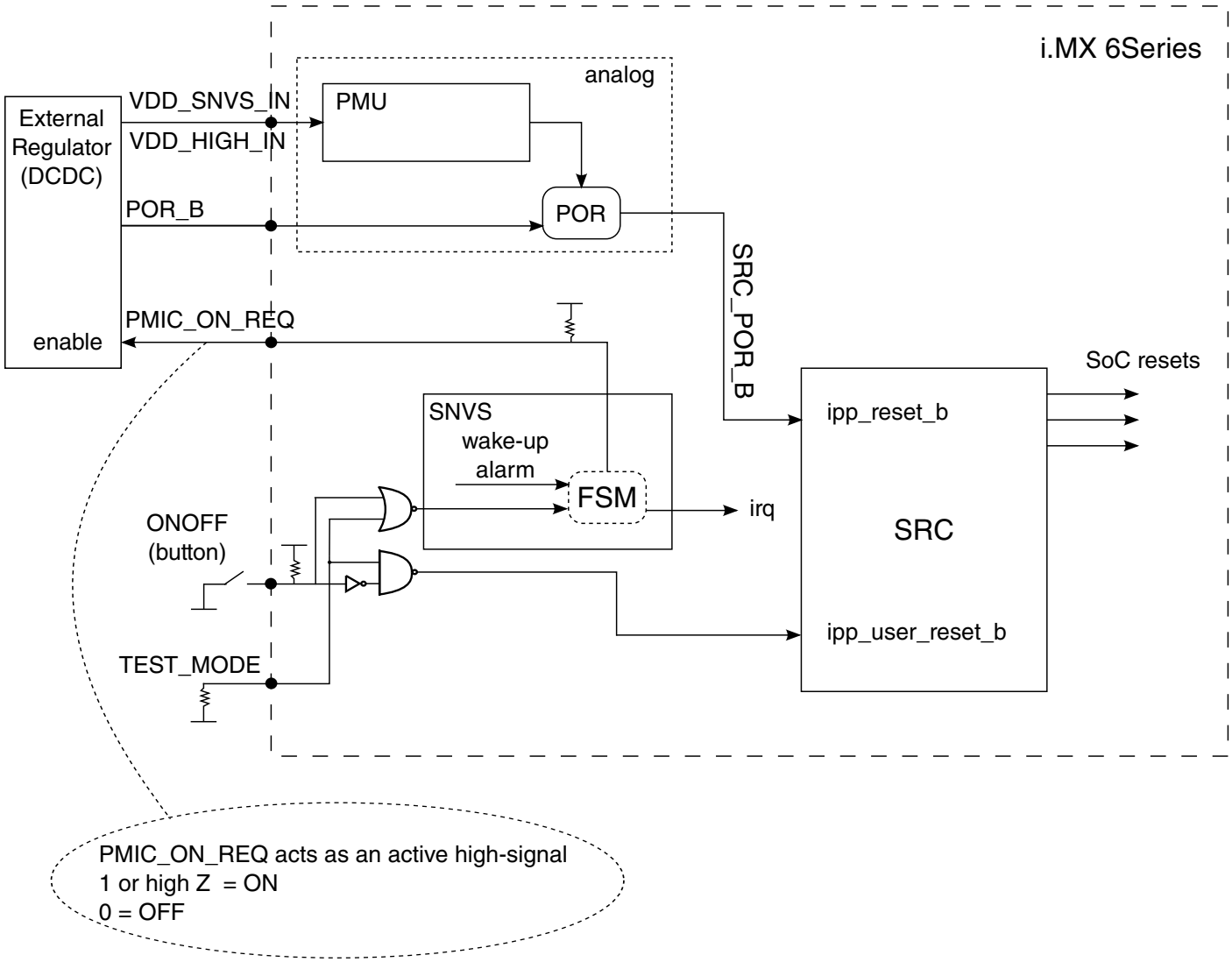


Figure 6-5. Chip reset scheme under PMU control

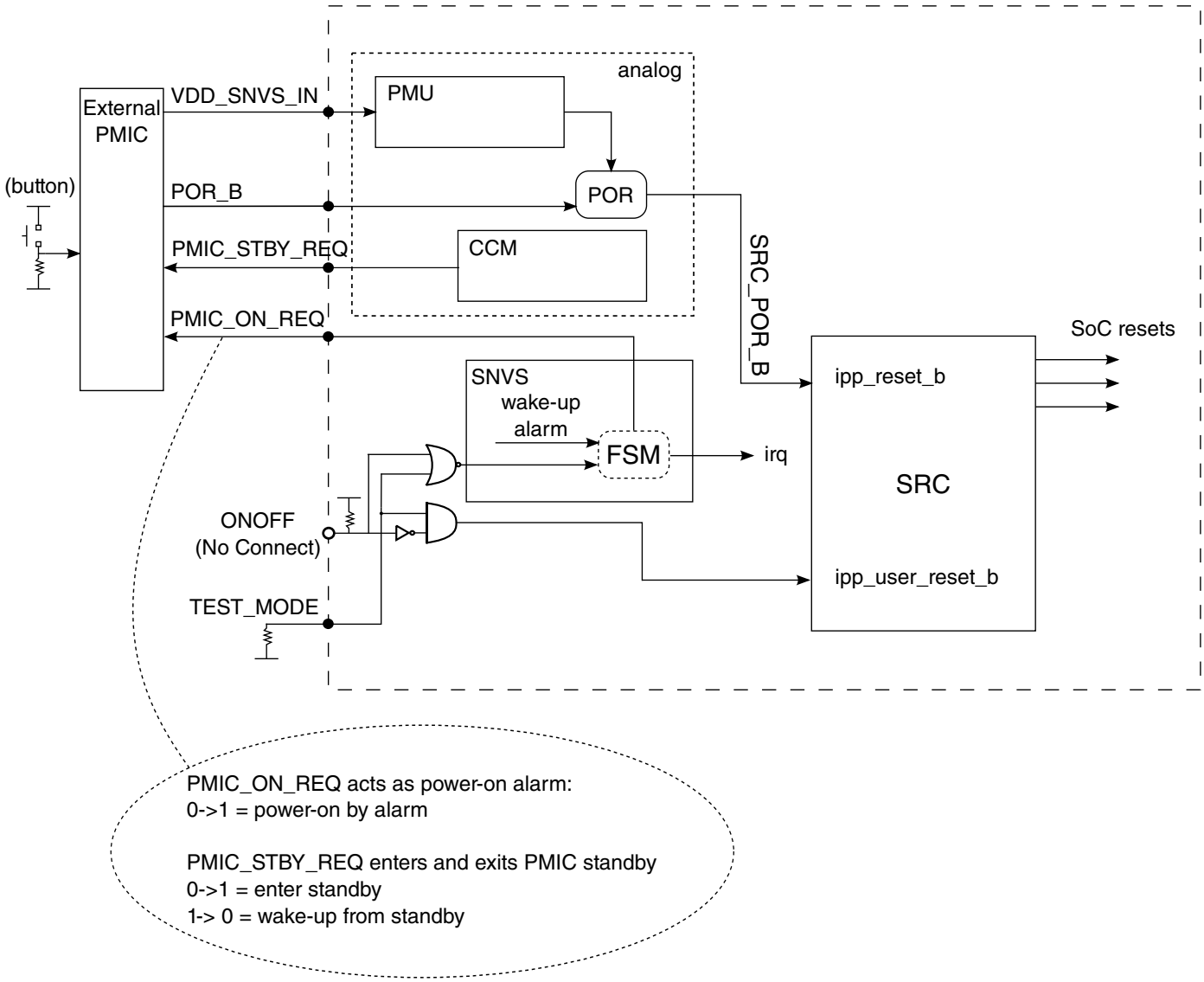


Figure 6-6. Chip reset scheme under external PMIC control

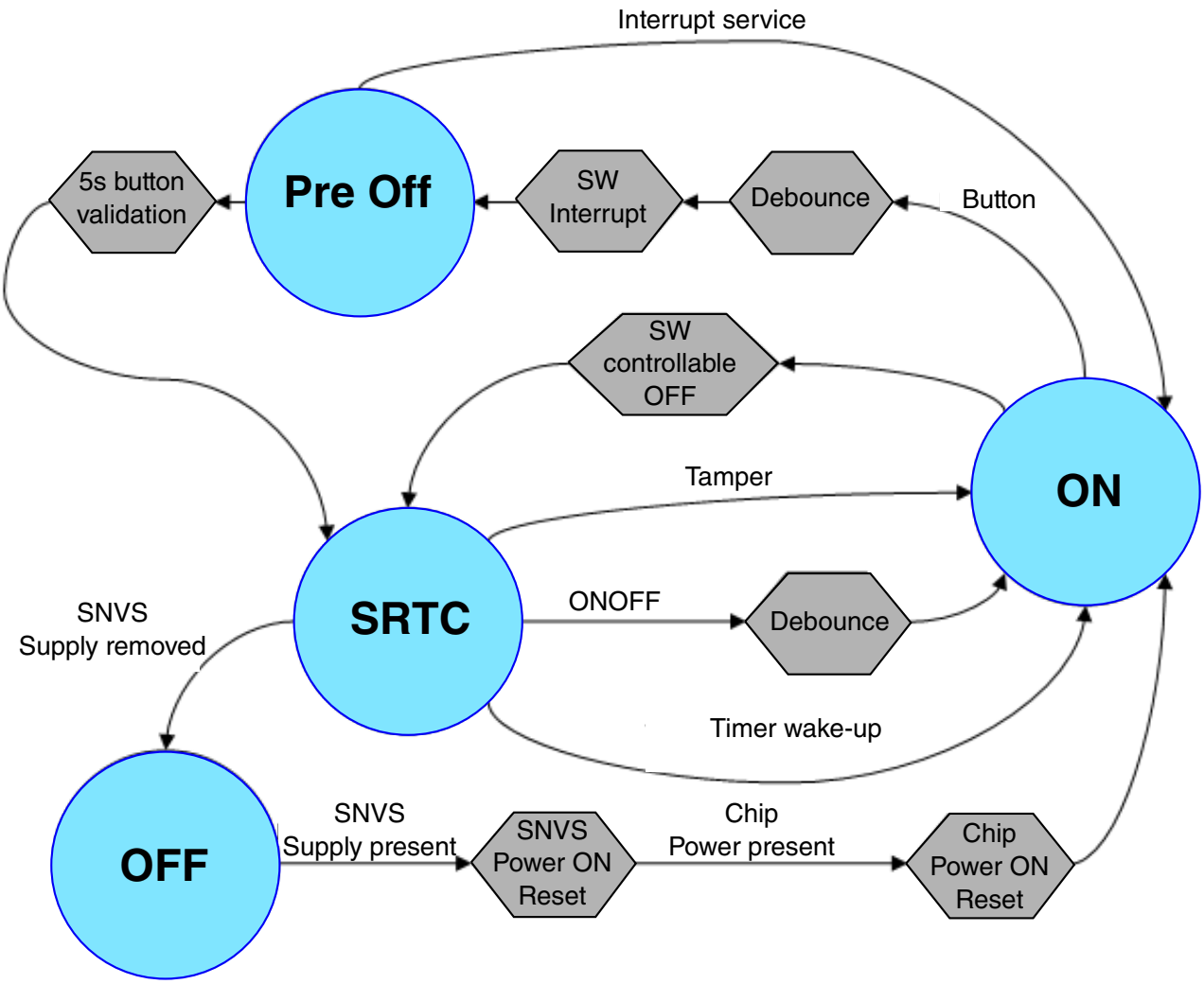


Figure 6-7. Chip on/off state flow diagram

6.2.4.2 Finite-State Machine (FSM)

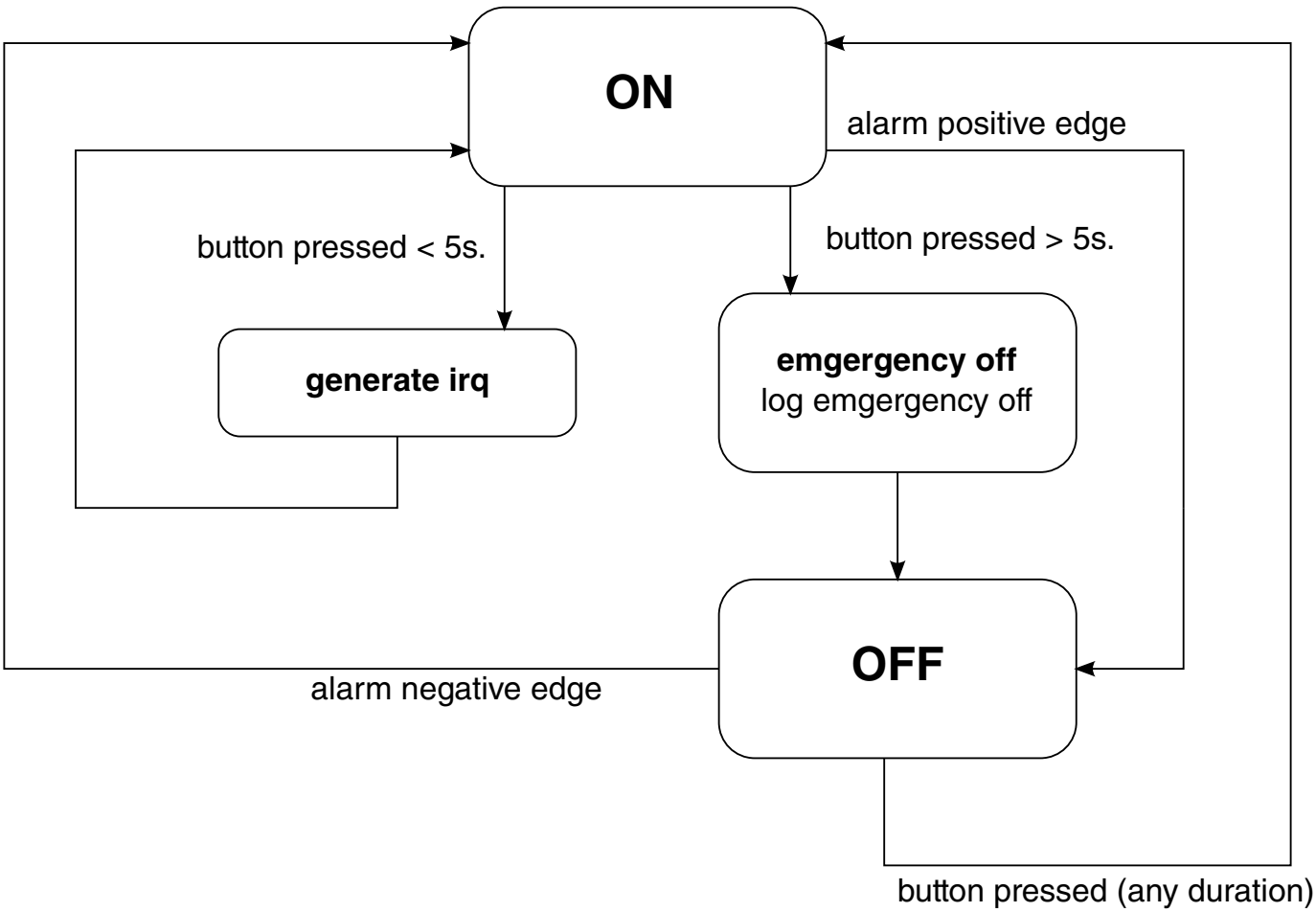


Figure 6-8. FSM

6.2.4.3 Power mode transitions

Table 6-8. Power mode transitions

Power mode	Configuration with external PMIC	Configuration with internal PMIC
ON, first time	1. Either coin cell or SoC power supply is connected to SNVS. 2. When button is pressed, PMIC powers on.	1. Either coin cell or SoC power supply is connected to SNVS. 2. When button is pressed, 'state' goes ON, PMIC_ON_REQ goes '1'. 3. External regulator is enabled.
Normal ON to OFF, by button	1. Button is pressed for a short duration on the external PMIC. 2. Interrupt request (irq) is sent to SoC from external PMIC. 3. SoC is programming PMIC for power off when standby is asserted.	1. SoC button is pressed for a short duration. 2. Interrupt request (irq) is sent to SoC from FSM. 3. Alarm timer is set up by software routine and started. 4. Upon alarm_in assertion to '1', PMIC_ON_REQ goes '0'.

Table continues on the next page...

Table 6-8. Power mode transitions (continued)

Power mode	Configuration with external PMIC	Configuration with internal PMIC
	4. In CCM STOP mode, Standby is asserted, PMIC gates SoC supplies.	5. External regulator goes OFF.
Emergency ON to OFF, by button	1. Button is pressed for an extended time on the external PMIC. 2. PMIC is powering off.	1. Button is pressed for longer than 5 seconds on the SoC. 2. FSM validates button pressed for 5 seconds. 3. Emergency power off is logged, PMIC_ON_REQ goes '0', alarm_mask goes '1'. 4. External regulator goes OFF.
OFF to ON, by button	1. Button is pressed on the external PMIC. 2. PMIC powers ON.	1. Button is pressed on the SoC. 2. PMIC_ON_REQ goes '1', alarm_mask goes '0'. 3. External regulator powers ON.
OFF to ON, by timer alarm	1. Timer alarm in SNVS is programmed by software before SoC goes OFF. 2. SoC enters OFF mode. 3. Upon timer limit, wake up alarm goes '0'. PMIC_ON_REQ goes '1'. 4. PMIC receives assertion of PMIC_ON_REQ and wakes up.	1. Timer alarm in SNVS is programmed by software before SoC goes OFF. 2. SoC enters OFF mode. 3. Upon timer limit, wake up alarm goes '0'. PMIC_ON_REQ goes '1'. 4. External regulator is enabled by PMIC_ON_REQ = 1.

6.2.5 Power-On Reset and power sequencing

This module generates an internal POR_B signal that is logically AND'ed with any externally applied SRC_POR_B signal. The internal POR_B signal will be held low until all of the following conditions are met:

- 4ms after the external power supply VDDHIGH_IN is valid
- 1ms after the VDD_SOC_CAP supply is valid

The 4ms and 1ms delays are derived from counting the 32 kHz RTC clock cycles; the accuracy depends on the accuracy of the RTC. When the RTC crystal is either absent or in the process of powering up, an internal ring oscillator will be the source of RTC, which is not as accurate as the crystal.

6.2.5.1 External POR using SRC_POR_B

If the external SRC_POR_B signal is used to control the processor POR, SRC_POR_B must remain low (asserted) until the VDD_ARM_CAP and VDD_SOC_CAP supplies are stable.

6.2.5.2 Internal POR

If the external SRC_POR_B signal is not used (always held high or left unconnected), the processor defaults to the internal POR function (PMU controls generation of the POR based on the power supplies).

If the internal POR function is used, the following power supply requirements must be met:

- VDD_ARM_IN and VDD_SOC_IN may be supplied from the same source, or
- VDD_SOC_IN can be supplied before VDD_ARM_IN with a maximum delay of 1 ms.

6.2.6 Functional Description

6.2.6.1 Reset Control

This section details the reset control of this device.

6.2.6.1.1 Reset inputs and outputs

The reset control logic receives reset requests from all potential reset sources. All the immediate sources of reset are directly passed to the reset stretching block, whereas the resets requiring qualification are passed on to the reset qualification logic before they are sent to the reset stretching block.

All reset inputs and outputs are described in the following figure:

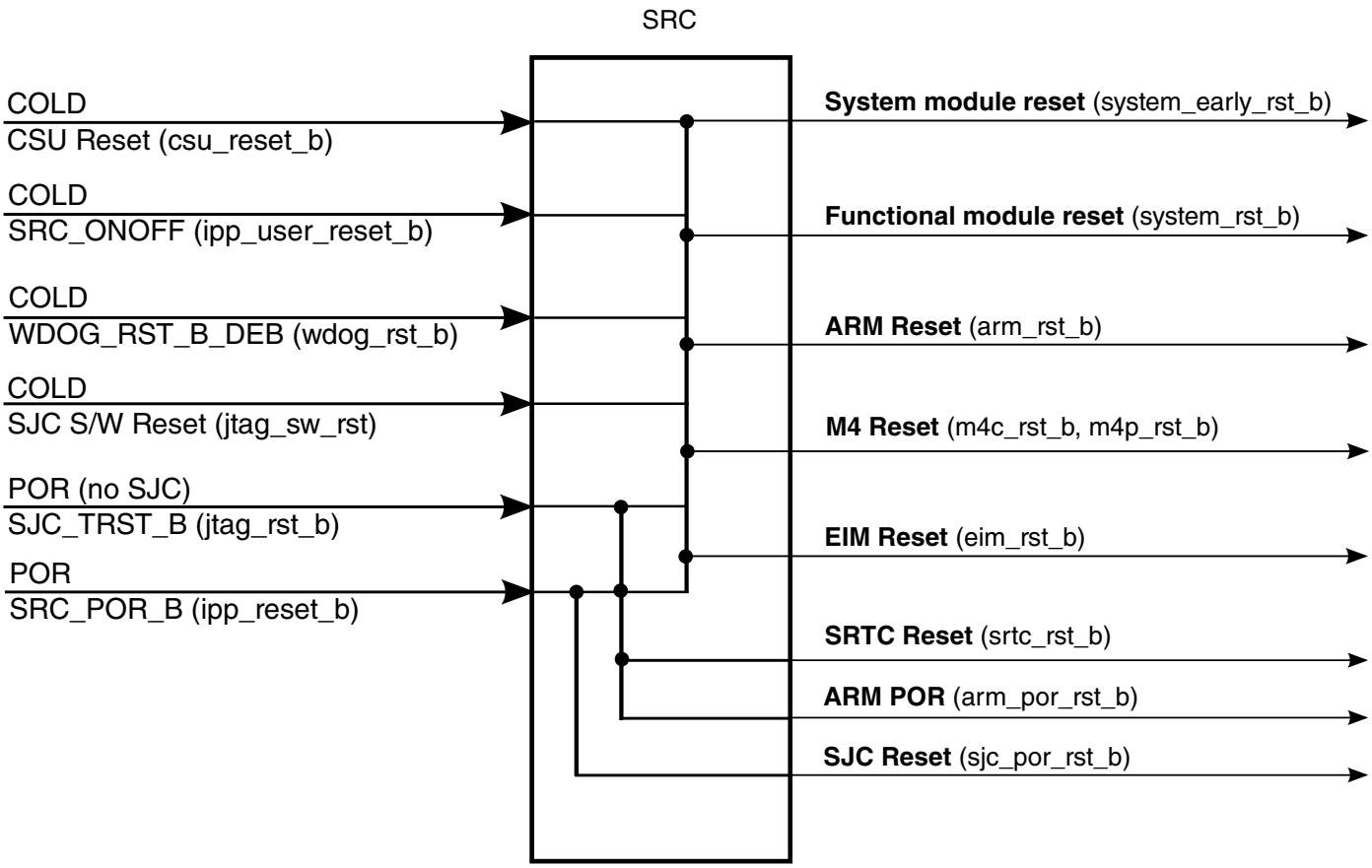


Figure 6-9. SRC inputs and outputs

The reset types and modules they affect are shown in Table 6-9. As there is no chip POR, the POR_B is used to reset the entire chip including test logic and JTAG modules.

NOTE

All resets are expected to be active low except jtag_sw_rst.

Table 6-9. SRC reset functionality

SoC Modules	POR	COLD
System modules (PLLs, fuses, etc)	yes	yes
Functional modules	yes	yes
ARM	yes	yes
ARM SoC	yes	yes
M4 Core	yes	yes
M4 Platform	yes	yes
ARM POR	yes	no
ARM debug	yes	no
SJC	yes	no
SRTC	yes	no

The reset priorities are POR (strongest) and COLD. If a stronger reset is asserted during the sequence of a weaker reset, then the weaker sequence will be overridden, and the stronger reset sequence will commence. There is no priority within a reset type (POR, etc). If a reset is asserted during the reset sequence of the same type, the reset sequence will be interrupted and restarted.

The following lists the functionality of each of these reset outputs:

- system_early_rst_b - Resets the system modules that need to start first as CCM, OCOTP_CTRL, FUSEBOX, etc.
- system_rst_b - Resets functional modules
- arm_rst_b - Resets ARM module (on regular system reset)
- arm_por_rst_b - Resets ARM por input
- arm_soc_rst_b - Reset for ARM SOC
- m4c_rst_b - Reset for M4 core
- m4p_rst_b - Reset for M4 platform
- arm_dbg_rst_b - Reset debug logic of ARM
- test_logic_rst_b - Reset test logic (IOMUXC, DAP)
- sjc_por_rst_b - Reset to SJC
- srtc_rst_b - Resets SRTC

NOTE

It is assumed that each reset source will deassert after its assertion, either due to reset generated to the system from SRC, or by negation of the reset source (if it came from an external source to the chip). In the latter case, the reset source is assumed to be held for at least 2 XTALI clocks so it can be sampled by SRC.

6.2.6.1.2 Reset Handling

6.2.6.1.2.1 Reset Sequence and De-Assertion

The SRC_ONOFF will assert immediately after any reset source is recognized. After all of the reset sources are released, the SRC will start the following set of events depending on the type of reset that occurred.

6.2.6.1.2.2 POR (SRC_POR_B)

SRC_POR_B is an external reset signal. When the chip is powered up, the reset signal is passed through the POR_B pin indicating power-up sequence. The SRC resets the entire chip including the JTAG (SJC) module. All SRC registers will be reset during the POR sequence.

As soon as SRC_POR_B occurs, all resets are asserted and the entire chip is reset by SRC. The SRC_POR_B is stretched for 2 XTALI cycles and the stretching sequence takes place after 2 XTALI clocks of POR_B pin deassertion.

The sjc_por_rst_b and srtc_rst_b signals are deasserted together with SRC_POR_B signal. Those outputs are also deasserted after the stretching of SRC_POR_B has deasserted.

Once the above resets deassert, system_early_rst_b reset is deasserted after 2 XTALI clocks. The system_early_rst_b is used for the CCM and PLL-IPs to start generating PLL clock ouputs and the system root clocks.

When the system root clocks are ready, the CCM will assert system_clk_ready signal. This signal is generated during the start sequence in the CCM and it involves the preparation of the PLLs to generate clock roots for functional operation.

SRC then enables OCOTP_CTRL and fusebox clocks, so that fuses can be loaded to OCOTP_CTRL.

- SRC will prepare the boot information
- After 8 ipg cycles, resets to all modules will be de-asserted
- After 8 ipg cycles, system clocks will be enabled (en_system_clk).

6.2.6.1.2.3 COLD RESET

The sequence is similar to SRC_POR_B except the memory repair operation is not performed.

Once the reset source deasserts, system_early_rst_b reset is deasserted after at least 2 XTALI clocks. The system_early_rst_b is used for the CCM and PLL-IPs to start generating PLL clock outputs and the system root clocks.

Once the system root clocks are ready, the CCM will assert system_clk_ready signal. This signal is generated during the start sequence in the CCM and it involves the preparation of the PLLs to generate clock roots for functional operation. See CCM for more information.

Once system_clk_ready arrives at the SRC, it will enable OCOTP_CTRL and fusebox clocks, so that fuses can be loaded to OCOTP_CTRL. OCOTP_CTRL will notify with iim_ready_flag once the fusebox loading finishes.

- SRC will prepare the boot information
- After 8 ipg cycles resets to all modules will be deasserted
- After 8 ipg cycles, system clocks will be enabled (en_system_clk).

6.2.6.2 Parallel Reset Requests

SRC will follow the following rules in the case of parallel reset requests:

1. The order of strength of resets is POR - strongest, COLD - medium
2. If a stronger reset is asserted during weaker reset sequence, then the stronger reset will take over and the stronger reset process will commence. The following cases fall into this category:
   - POR reset request in the middle of cold reset process - the cold will be stopped and the POR sequence will start.
3. If a weaker reset is asserted during stronger reset sequence, then the stronger reset sequence will continue without interference. If at the end of the stronger reset process the weaker request is still asserted then the weaker sequence will commence. The following cases fall into this category:
   - COLD reset requests in the middle of POR reset process - the POR process will continue without interference.
4. If a similar reset request is asserted during the process of reset handling, then the process of reset handling will start over (with the same process). The following cases fall into this category:
   - POR reset request in the middle of POR reset process - the POR process will start over.
   - COLD reset request in the middle of COLD reset process - the COLD process will start over.

6.2.6.3 Boot Mode Control

6.2.6.3.1 BOOT_MODE Pin Latching

The exact boot sequence is controlled by the values of the BOOT_MODE pins on this device.

The value of the BOOT_MODE pins will be latched after the OCOTP_CTRL asserts the fuse read completion flag. After latching, the values of the BOOT_MODE pins are used to determine the booting options of the core as described in the SRC_SBMRx registers.

The boot mode general purpose bits can be provided to the SRC from either e-fuses or GPIO signals. The gpio_bt_sel e-fuse defines the source to be used to derive the boot information. When gpio_bt_sel is set, e-fuses are used. When cleared, GPIO signals are used.

The boot information is provided in SRC_SBMR1 register. The figure below shows the selection of boot mode information.

NOTE

BOOT_MODE[1:0] inputs of SRC are connected to BOOT_MODE[1:0] pins.

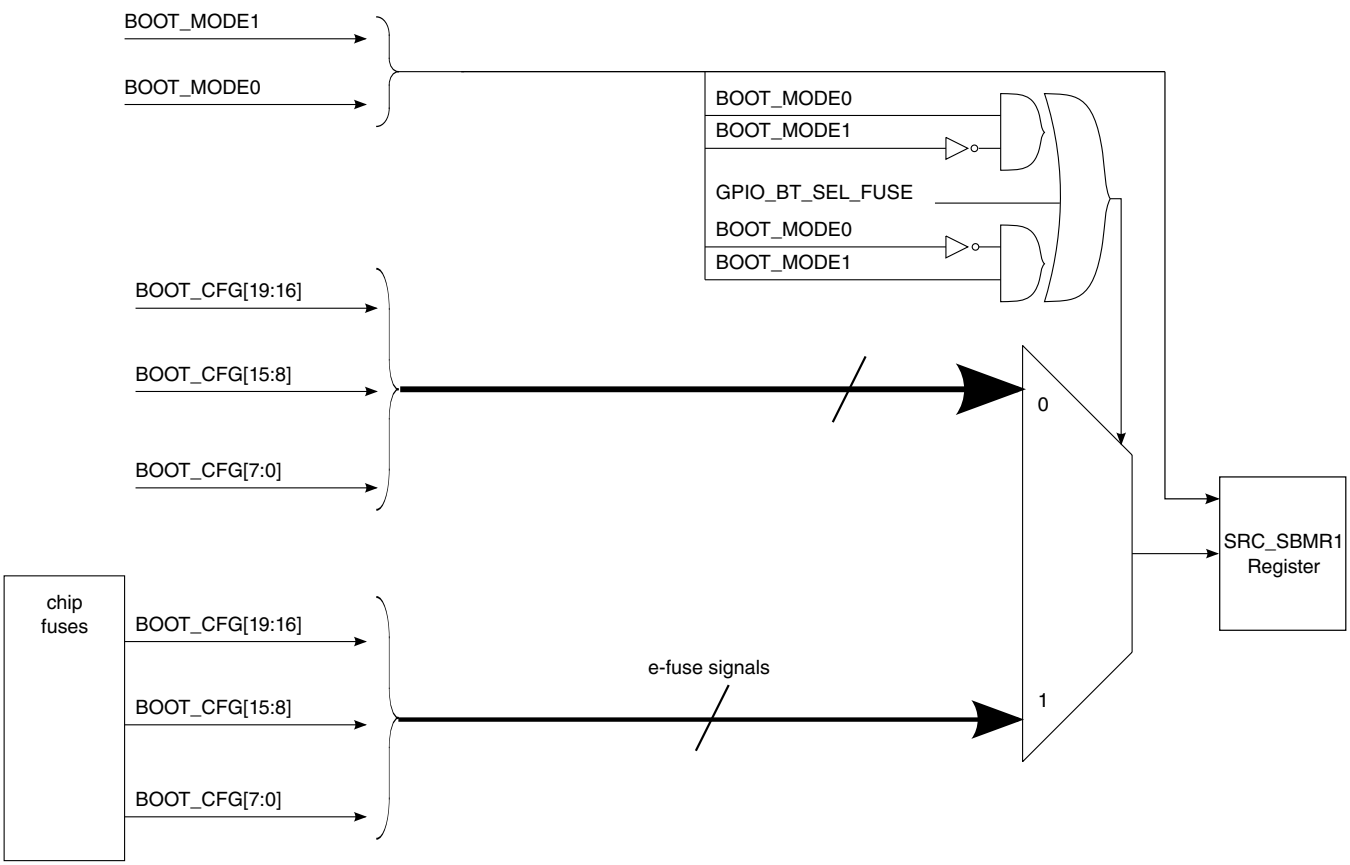


Figure 6-10. Boot mode information

6.2.7 SRC Memory Map/Register Definition

SRC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3039_0000	SRC Reset Control Register (SRC_SCR)	32	R/W	0000_00A0h	6.2.7.1/ 1005
3039_0004	A7 Reset Control Register (SRC_A7RCR0)	32	R/W	000A_0000h	6.2.7.2/ 1007
3039_0008	A7 Reset Control Register (SRC_A7RCR1)	32	R/W	0000_0001h	6.2.7.3/ 1011
3039_000C	M4 Reset Control Register (SRC_M4RCR)	32	R/W	0000_00A8h	6.2.7.4/ 1013
3039_0014	EIM Reset Control Register (SRC_ERCR)	32	R/W	0000_0000h	6.2.7.5/ 1015
3039_001C	HSIC PHY Reset Control Register (SRC_HSICPHY_RCR)	32	R/W	0000_0000h	6.2.7.6/ 1017
3039_0020	USB OTG PHY1 Reset Control Register (SRC_USBOPHY1_RCR)	32	R/W	0000_0000h	6.2.7.7/ 1019
3039_0024	USB OTG PHY2 Reset Control Register (SRC_USBOPHY2_RCR)	32	R/W	0000_0000h	6.2.7.8/ 1021
3039_0028	MIPI PHY Reset Control Register (SRC_MIPIPHY_RCR)	32	R/W	0000_0000h	6.2.7.9/ 1023
3039_002C	PCIE PHY Reset Control Register (SRC_PCIEPHY_RCR)	32	R/W	0000_0006h	6.2.7.10/ 1025
3039_0058	SRC Boot Mode Register 1 (SRC_SBMR1)	32	R	0000_0000h	6.2.7.11/ 1027
3039_005C	SRC Reset Status Register (SRC_SRSR)	32	R/W	0000_0001h	6.2.7.12/ 1028
3039_0068	SRC Interrupt Status Register (SRC_SISR)	32	R	0000_0000h	6.2.7.13/ 1030
3039_006C	SRC Interrupt Mask Register (SRC_SIMR)	32	R/W	0000_001Fh	6.2.7.14/ 1033
3039_0070	SRC Boot Mode Register 2 (SRC_SBMR2)	32	R	0000_0000h	6.2.7.15/ 1036
3039_0074	SRC General Purpose Register 1 (SRC_GPR1)	32	R/W	0000_0000h	6.2.7.16/ 1037
3039_0078	SRC General Purpose Register 2 (SRC_GPR2)	32	R/W	0000_0000h	6.2.7.17/ 1037
3039_007C	SRC General Purpose Register 3 (SRC_GPR3)	32	R/W	0000_0000h	6.2.7.18/ 1038
3039_0080	SRC General Purpose Register 4 (SRC_GPR4)	32	R/W	0000_0000h	6.2.7.19/ 1038
3039_0084	SRC General Purpose Register 5 (SRC_GPR5)	32	R/W	0000_0000h	6.2.7.20/ 1038
3039_0088	SRC General Purpose Register 6 (SRC_GPR6)	32	R/W	0000_0000h	6.2.7.21/ 1039
3039_008C	SRC General Purpose Register 7 (SRC_GPR7)	32	R/W	0000_0000h	6.2.7.22/ 1039

Table continues on the next page...

SRC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3039_0090	SRC General Purpose Register 8 (SRC_GPR8)	32	R/W	0000_0000h	6.2.7.23/ 1039
3039_0094	SRC General Purpose Register 9 (SRC_GPR9)	32	R/W	0000_0000h	6.2.7.24/ 1040
3039_0098	SRC General Purpose Register 10 (SRC_GPR10)	32	R/W	0000_0000h	6.2.7.25/ 1040
3039_1000	SRC DDR Controller Reset Control Register (SRC_DDRC_RCR)	32	R/W	0000_0003h	6.2.7.26/ 1041

6.2.7.1 SRC Reset Control Register (SRC_SCR)

The reset control register (SCR), contains bits that control operation of the reset controller.

Address: 3039_0000h base + 0h offset = 3039_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DOM_EN	LOCK	Reserved		DOMAIN3	DOMAIN2	DOMAIN1	DOMAIN0	Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								MASK_TEMPSENSE_ RESET				Reserved			
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

SRC_SCR field descriptions

Field	Description
31 DOM_EN	Domain Control enable for this register. **NOTE:** [31:30] and [27:24] areas are not controlled by this bit. [31:30] and [27:24] area can be modified by any masters from any domains when Lock bit is not set. 0 Disables domain control. All of this register's bits except [31:30] and [27:24] can be modified by any masters 1 Enables domain control. All of this register's bits except [31:30] and [27:24] can only be modified by the masters from the domains specified in [27:24] area.
30 LOCK	Domain control bits lock

Table continues on the next page...

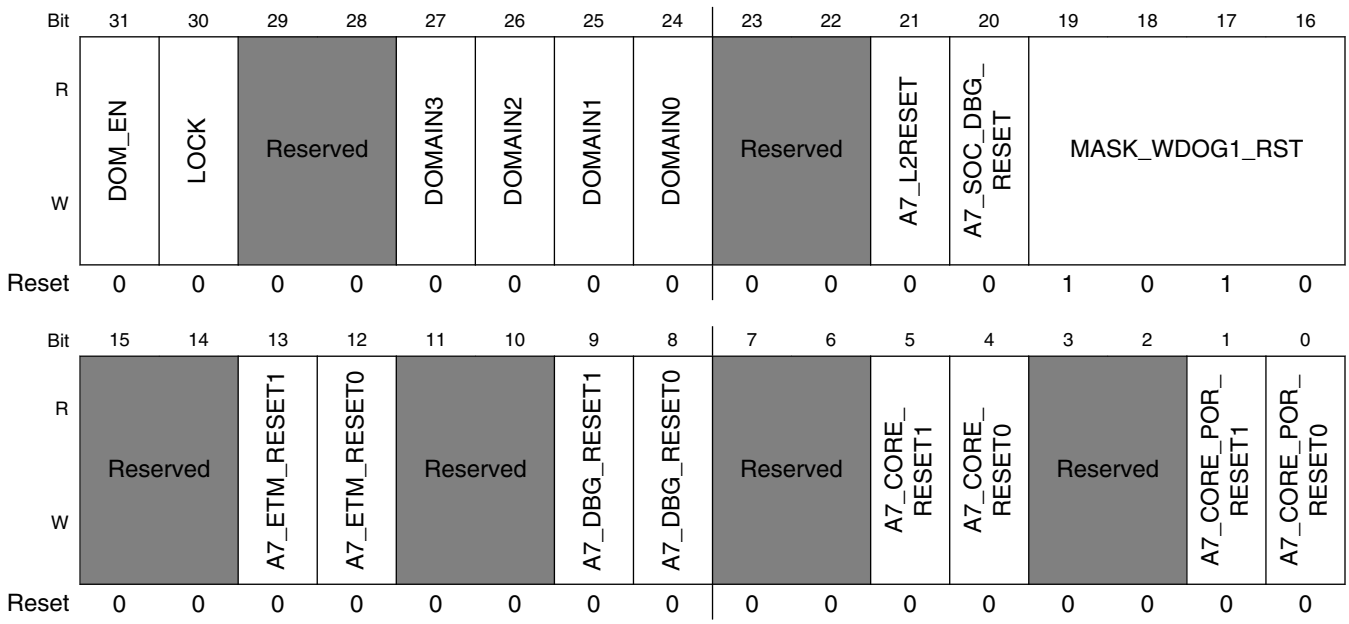
SRC_SCR field descriptions (continued)

Field	Description
	NOTE: Lock bit is a write-once register, once it is set to 1, it can't be write to 0 0 [31] and [27:24] bits can be modified 1 [31] and [27:24] bits cannot be modified
29–28 -	This field is reserved. Reserved
27 DOMAIN3	Domain3 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain3. The master from domain3 cannot write to this register. 1 This register is assigned to domain3. The master from domain3 can write to this register
26 DOMAIN2	Domain2 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain2. The master from domain3 cannot write to this register. 1 This register is assigned to domain2. The master from domain3 can write to this register
25 DOMAIN1	Domain1 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain1. The master from domain3 cannot write to this register. 1 This register is assigned to domain1. The master from domain3 can write to this register
24 DOMAIN0	Domain0 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain0. The master from domain3 cannot write to this register. 1 This register is assigned to domain0. The master from domain3 can write to this register
23–8 -	This field is reserved. Reserved
7–4 MASK_ TEMPSENSE_ RESET	Mask tempsense_reset source. If these 4 bits are coded from A to 5 then, the tempsense_reset input to SRC will be masked and the tempsense_reset will not create a reset to the chip. 0101 tempsense_reset is masked 1010 tempsense_reset is not masked
-	This field is reserved. Reserved

6.2.7.2 A7 Reset Control Register (SRC_A7RCR0)

The A7 Reset Control Register (A7RCR), contains bits that control the A7 reset generation.

Address: 3039_0000h base + 4h offset = 3039_0004h



SRC_A7RCR0 field descriptions

Field	Description
31 DOM_EN	Domain Control enable for this register. **NOTE:** [31:30] and [27:24] areas are not controlled by this bit. [31:30] and [27:24] area can be modified by any masters from any domains when Lock bit is not set. 0 Disables domain control. All of this register's bits except [31:30] and [27:24] can be modified by any masters 1 Enables domain control. All of this register's bits except [31:30] and [27:24] can only be modified by the masters from the domains specified in [27:24] area.
30 LOCK	Domain control bits lock **NOTE:** Lock bit is a write-once register, once it is set to 1, it can't be write to 0 0 [31] and [27:24] bits can be modified 1 [31] and [27:24] bits cannot be modified
29–28 -	This field is reserved. Reserved
27 DOMAIN3	Domain3 assignment control. Effective when dom_en is set to 1.

Table continues on the next page...

SRC_A7RCR0 field descriptions (continued)

Field	Description
	0 This register is not assigned to domain3. The master from domain3 cannot write to this register. 1 This register is assigned to domain3. The master from domain3 can write to this register
26 DOMAIN2	Domain2 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain2. The master from domain3 cannot write to this register. 1 This register is assigned to domain2. The master from domain3 can write to this register
25 DOMAIN1	Domain1 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain1. The master from domain3 cannot write to this register. 1 This register is assigned to domain1. The master from domain3 can write to this register
24 DOMAIN0	Domain0 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain0. The master from domain3 cannot write to this register. 1 This register is assigned to domain0. The master from domain3 can write to this register
23–22 -	This field is reserved. Reserved
21 A7_L2RESET	Software reset for A7 Snoop Control Unit (SCU). **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert SCU reset 1 assert SCU reset
20 A7_SOC_DBG_ RESET	Software reset for system level debug reset. It initializes the shared Debug APB, the CTI, and the CTM. It also causes: • A7_dbgreset[1:0] and A7_etmreset[1:0] to be asserted • debug logic in the processor power domain and in the debug power domain to be reset **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert system level debug reset 1 assert system level debug reset
19–16 MASK_WDOG1_ RST	Mask wdog1_rst_b source. If these 4 bits are coded from A to 5 then, the wdog1_rst_b input to SRC will be masked and the wdog1_rst_b will not create a reset to the chip. **NOTE:** During the time the WDOG event is masked using SRC logic, it is likely that the WDOG Reset Status Register (WRSR) bit 1 (which indicates a WDOG timeout event) will get asserted. software / OS developer must prepare for this case. Re-enabling the WDOG is possible, by un-masking it in SRC, though it must be preceded by servicing the WDOG. However, for the case that the event has been asserted, the status bit (WRSR bit-1) will remain asserted, regardless of servicing the WDOG module. (Hardware reset is the only way to cause the de-assertion of that bit). Any other code will be coded to 1010 i.e. wdog1_rst_b is not masked 0101 wdog1_rst_b is masked 1010 wdog1_rst_b is not masked
15–14 -	This field is reserved. Reserved

Table continues on the next page...

SRC_A7RCR0 field descriptions (continued)

Field	Description
13 A7_ETM_ RESET1	Software reset for core1 ETM only. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core1 ETM reset 1 assert core1 ETM reset
12 A7_ETM_ RESET0	Software reset for core0 ETM only. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core0 ETM reset 1 assert core0 ETM reset
11–10 -	This field is reserved. Reserved
9 A7_DBG_ RESET1	Software reset for core1 debug only. It initialize the debug, and breakpoint and watchpoint logic in the core1 processor power domain. It also reset the debug logic for core1 processor, which is in the debug power domain. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core1 debug reset 1 assert core1 debug reset
8 A7_DBG_ RESET0	Software reset for core0 debug only. It initialize the debug, and breakpoint and watchpoint logic in the core1 processor power domain. It also reset the debug logic for core1 processor, which is in the debug power domain. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core0 debug reset 1 assert core0 debug reset
7–6 -	This field is reserved. Reserved
5 A7_CORE_ RESET1	Software reset for core1 only. It initializes the processor logic in the core1 processor power domains, not including the debug, breakpoint and watchpoint logic. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core1 reset 1 assert core1 reset
4 A7_CORE_ RESET0	Software reset for core0 only. It initializes the processor logic in the core0 processor power domains, not including the debug, breakpoint and watchpoint logic.

Table continues on the next page...

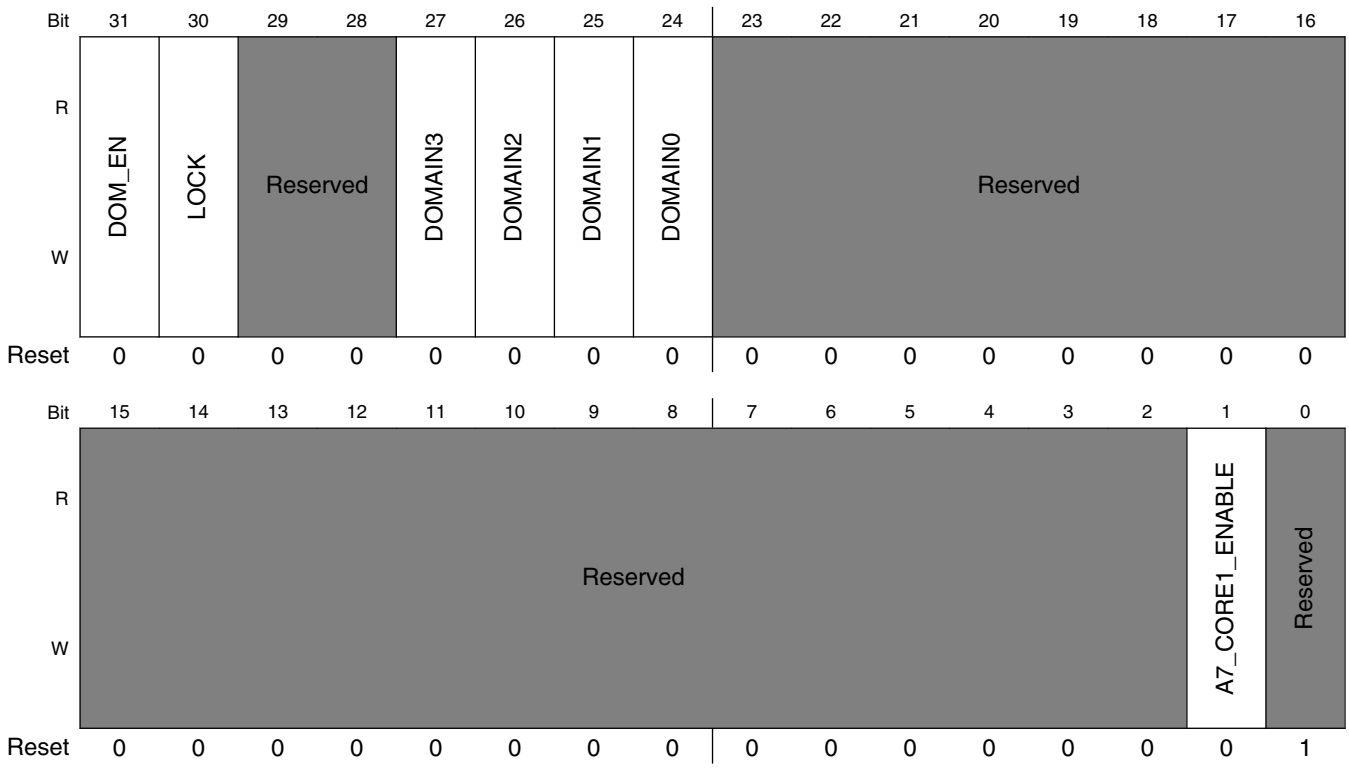
SRC_A7RCR0 field descriptions (continued)

Field	Description
	NOTE: This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core0 reset 1 assert core0 reset
3–2 -	This field is reserved. Reserved
1 A7_CORE_ POR_RESET1	POR reset for A7 core1 only. It initializes all the core1 processor logic, including CPU Debug, and breakpoint and watchpoint logic in the core1 processor power domains **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core1 reset 1 assert core1 reset
0 A7_CORE_ POR_RESET0	POR reset for A7 core0 only. It initializes all the core0 processor logic, including CPU Debug, and breakpoint and watchpoint logic in the core0 processor power domains **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core0 reset 1 assert core0 reset

6.2.7.3 A7 Reset Control Register (SRC_A7RCR1)

The A7 Reset Control Register (A7RCR), contains bits that control the A7 reset generation.

Address: 3039_0000h base + 8h offset = 3039_0008h



SRC_A7RCR1 field descriptions

Field	Description
31 DOM_EN	Domain Control enable for this register. **NOTE:** [31:30] and [27:24] areas are not controlled by this bit. [31:30] and [27:24] area can be modified by any masters from any domains when Lock bit is not set. 0 Disables domain control. All of this register's bits except [31:30] and [27:24] can be modified by any masters 1 Enables domain control. All of this register's bits except [31:30] and [27:24] can only be modified by the masters from the domains specified in [27:24] area.
30 LOCK	Domain control bits lock **NOTE:** Lock bit is a write-once register, once it is set to 1, it can't be write to 0 0 [31] and [27:24] bits can be modified 1 [31] and [27:24] bits cannot be modified

Table continues on the next page...

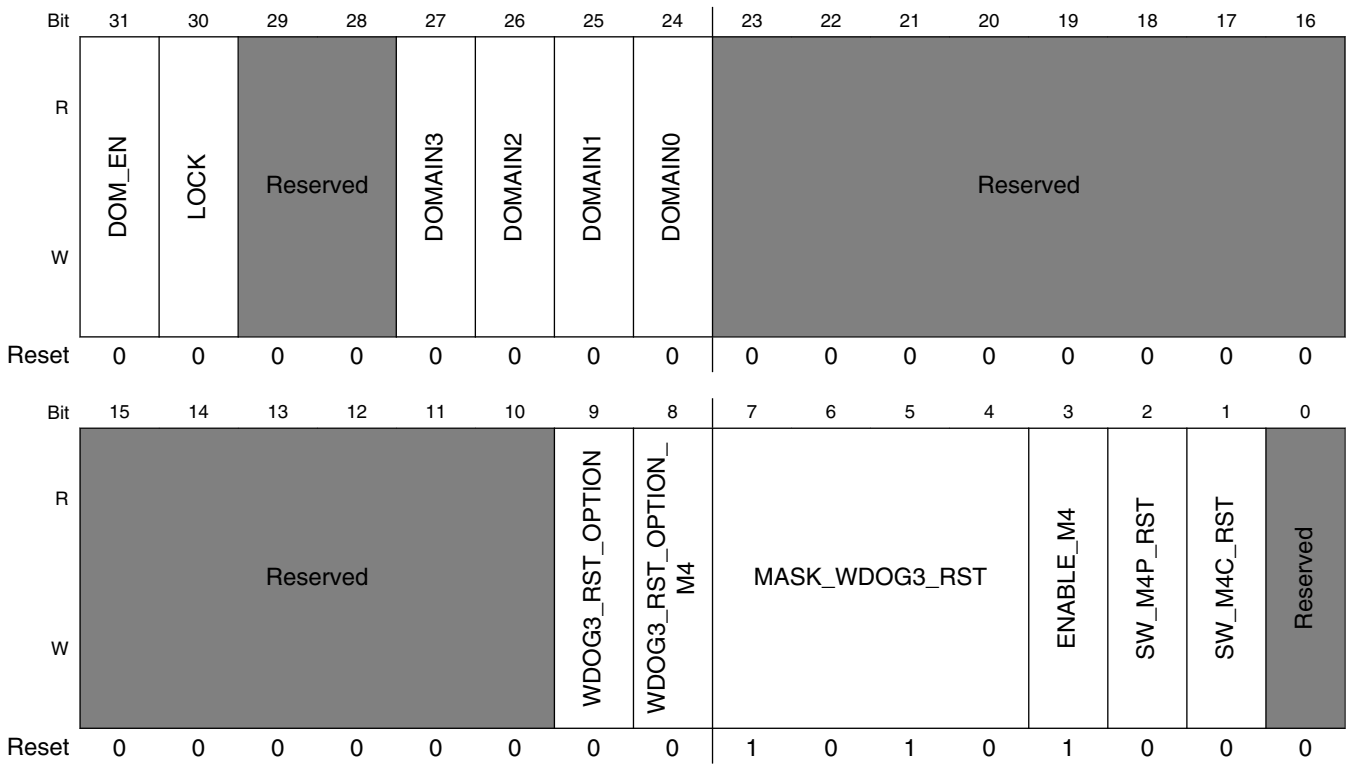
SRC_A7RCR1 field descriptions (continued)

Field	Description
29–28 -	This field is reserved. Reserved
27 DOMAIN3	Domain3 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain3. The master from domain3 cannot write to this register. 1 This register is assigned to domain3. The master from domain3 can write to this register
26 DOMAIN2	Domain2 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain2. The master from domain3 cannot write to this register. 1 This register is assigned to domain2. The master from domain3 can write to this register
25 DOMAIN1	Domain1 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain1. The master from domain3 cannot write to this register. 1 This register is assigned to domain1. The master from domain3 can write to this register
24 DOMAIN0	Domain0 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain0. The master from domain3 cannot write to this register. 1 This register is assigned to domain0. The master from domain3 can write to this register
23–2 -	This field is reserved. Reserved
1 A7_CORE1_ ENABLE	core 1 enable 0 core1 is disabled 1 core1 is enabled
0 -	This field is reserved. Reserved

6.2.7.4 M4 Reset Control Register (SRC_M4RCR)

The M4 Reset Control Register (M4RCR), contains bits that control the M4 reset generation.

Address: 3039_0000h base + Ch offset = 3039_000Ch



SRC_M4RCR field descriptions

Field	Description
31 DOM_EN	Domain Control enable for this register. **NOTE:** [31:30] and [27:24] areas are not controlled by this bit. [31:30] and [27:24] area can be modified by any masters from any domains when Lock bit is not set. 0 Disables domain control. All of this register's bits except [31:30] and [27:24] can be modified by any masters 1 Enables domain control. All of this register's bits except [31:30] and [27:24] can only be modified by the masters from the domains specified in [27:24] area.
30 LOCK	Domain control bits lock **NOTE:** Lock bit is a write-once register, once it is set to 1, it can't be write to 0 0 [31] and [27:24] bits can be modified 1 [31] and [27:24] bits cannot be modified

Table continues on the next page...

SRC_M4RCR field descriptions (continued)

Field	Description
29–28 -	This field is reserved. Reserved
27 DOMAIN3	Domain3 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain3. The master from domain3 cannot write to this register. 1 This register is assigned to domain3. The master from domain3 can write to this register
26 DOMAIN2	Domain2 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain2. The master from domain3 cannot write to this register. 1 This register is assigned to domain2. The master from domain3 can write to this register
25 DOMAIN1	Domain1 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain1. The master from domain3 cannot write to this register. 1 This register is assigned to domain1. The master from domain3 can write to this register
24 DOMAIN0	Domain0 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain0. The master from domain3 cannot write to this register. 1 This register is assigned to domain0. The master from domain3 can write to this register
23–10 -	This field is reserved. Reserved
9 WDOG3_RST_ OPTION	Wdog3_rst_b option 0 Wdog3_rst_b asserts M4 reset 1 Wdog3_rst_b asserts global reset
8 WDOG3_RST_ OPTION_M4	Wdog3_rst_b option for M4. This bit is only effective when wdog3_rst_option is set to 1. 0 wdgo3_rst_b Reset M4 core only 1 Reset both M4 core and platform
7–4 MASK_WDOG3_ RST	Mask wdog3_rst_b source. If these 4 bits are coded from A to 5 then, the wdog3_rst_b input to SRC will be masked and the wdog3_rst_b will not create a reset to the chip. **NOTE:** During the time the WDOG3 event is masked using SRC logic, it is likely that the WDOG3 Reset Status Register (WRSR) bit 1 (which indicates a WDOG3 timeout event) will get asserted. Software / OS developer must prepare for this case. Re-enabling the WDOG3 is possible, by un-masking it in SRC, though it must be preceded by servicing the WDOG3. However, for the case that the event has been asserted, the status bit (WRSR bit-1) will remain asserted, regardless of servicing the WDOG3 module. (Hardware reset is the only way to cause the de-assertion of that bit). Any other code will be coded to 1010 i.e. wdog3_rst_b is not masked 0101 wdog3_rst_b is masked 1010 wdog3_rst_b is not masked
3 ENABLE_M4	Enable M4 0 M4 is disabled 1 M4 is enabled
2 SW_M4P_RST	Self-clearing SW reset for M4 platform

Table continues on the next page...

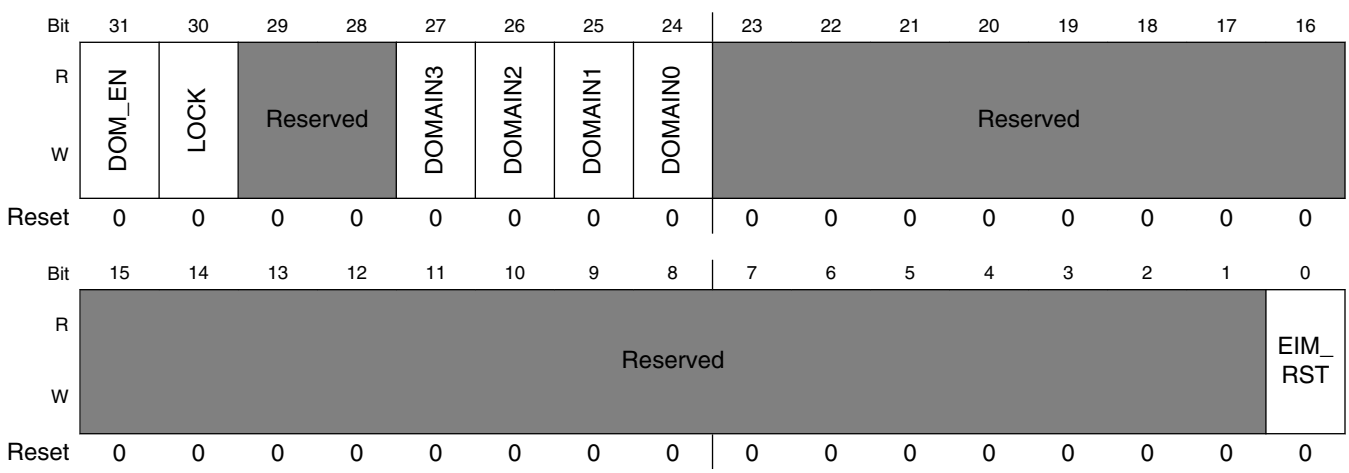
SRC_M4RCR field descriptions (continued)

Field	Description
	NOTE: This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. Software can determine that the reset has finished once this bit is cleared. Software can also configure SRC to generate interrupt once the software has finished. Please refer to SRC_SISR register for details. 0 do not assert M4 platform reset 1 assert M4 platform reset
1 SW_M4C_RST	Self-clearing SW reset for M4 core **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. Software can determine that the reset has finished once this bit is cleared. Software can also configure SRC to generate interrupt once the software has finished. Please refer to SRC_SISR register for details. 0 do not assert M4 core reset 1 assert M4 core reset
0 SW_M4C_NON_SCLR_RST	This field is reserved. Non-self-clearing SW reset for M4 core 0 do not assert M4 core reset 1 assert M4 core reset

6.2.7.5 EIM Reset Control Register (SRC_ERCR)

The EIM Reset Control Register (SRC_ERCR), contains bits that control the EIM Controller reset generation.

Address: 3039_0000h base + 14h offset = 3039_0014h



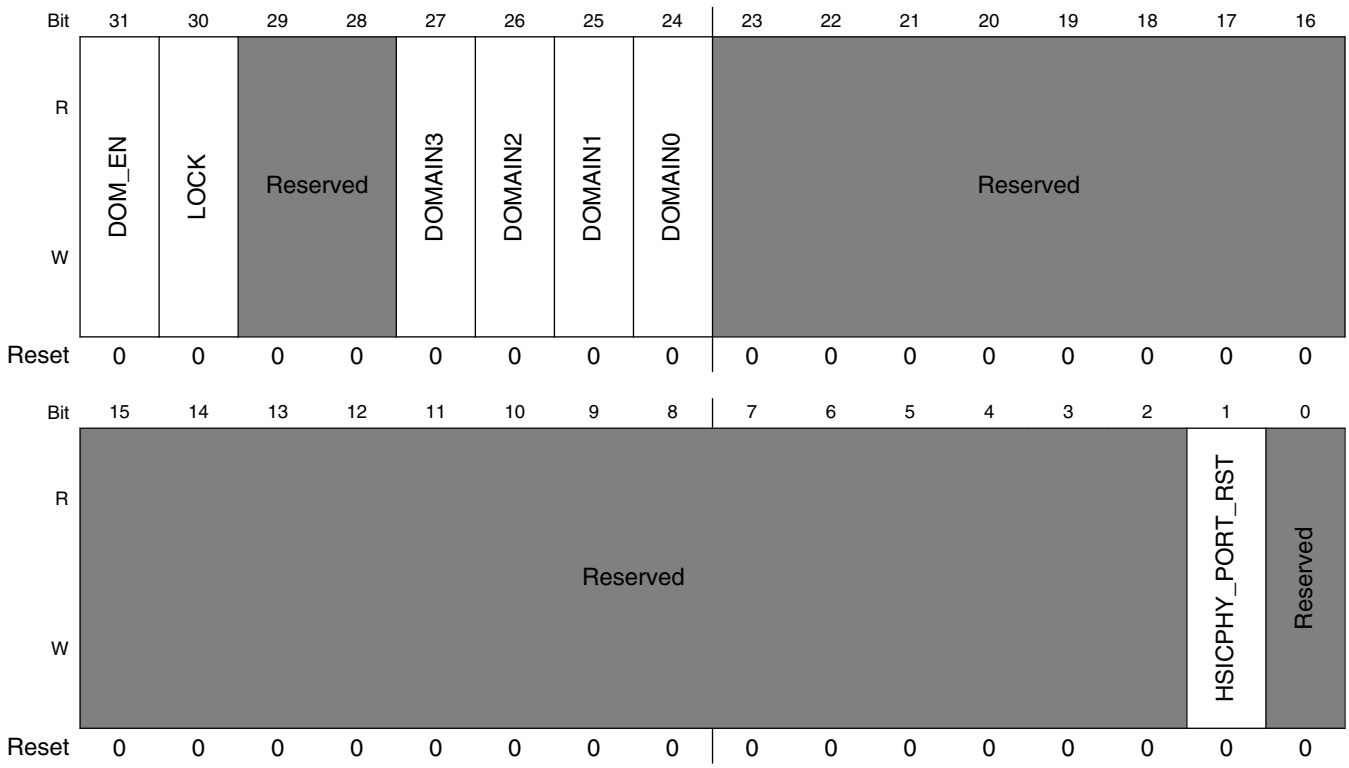
SRC_ERCR field descriptions

Field	Description
31 DOM_EN	Domain Control enable for this register. **NOTE:** [31:30] and [27:24] areas are not controlled by this bit. [31:30] and [27:24] area can be modified by any masters from any domains when Lock bit is not set. 0 Disables domain control. All of this register's bits except [31:30] and [27:24] can be modified by any masters 1 Enables domain control. All of this register's bits except [31:30] and [27:24] can only be modified by the masters from the domains specified in [27:24] area.
30 LOCK	Domain control bits lock **NOTE:** Lock bit is a write-once register, once it is set to 1, it can't be write to 0 0 [31] and [27:24] bits can be modified 1 [31] and [27:24] bits cannot be modified
29–28 -	This field is reserved. Reserved
27 DOMAIN3	Domain3 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain3. The master from domain3 cannot write to this register. 1 This register is assigned to domain3. The master from domain3 can write to this register
26 DOMAIN2	Domain2 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain2. The master from domain3 cannot write to this register. 1 This register is assigned to domain2. The master from domain3 can write to this register
25 DOMAIN1	Domain1 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain1. The master from domain3 cannot write to this register. 1 This register is assigned to domain1. The master from domain3 can write to this register
24 DOMAIN0	Domain0 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain0. The master from domain3 cannot write to this register. 1 This register is assigned to domain0. The master from domain3 can write to this register
23–1 -	This field is reserved. Reserved
0 EIM_RST	EIM reset is needed in order to reconfigure the eim chip select. The software reset bit must de-asserted. The eim chip select configuration should be updated. The software bit must be re-asserted since this is not self-refresh. 0 assert EIM controller reset 1 do not assert EIM controller reset

6.2.7.6 HSIC PHY Reset Control Register (SRC_HSICPHY_RCR)

The HSIC PHY Reset Control Register (SRC_HSICPHY_RCR), contains bits that control the HSIC PHY reset generation.

Address: 3039_0000h base + 1Ch offset = 3039_001Ch



SRC_HSICPHY_RCR field descriptions

Field	Description
31 DOM_EN	Domain Control enable for this register. **NOTE:** [31:30] and [27:24] areas are not controlled by this bit. [31:30] and [27:24] area can be modified by any masters from any domains when Lock bit is not set. 0 Disables domain control. All of this register's bits except [31:30] and [27:24] can be modified by any masters 1 Enables domain control. All of this register's bits except [31:30] and [27:24] can only be modified by the masters from the domains specified in [27:24] area.
30 LOCK	Domain control bits lock **NOTE:** Lock bit is a write-once register, once it is set to 1, it can't be write to 0 0 [31] and [27:24] bits can be modified 1 [31] and [27:24] bits cannot be modified

Table continues on the next page...

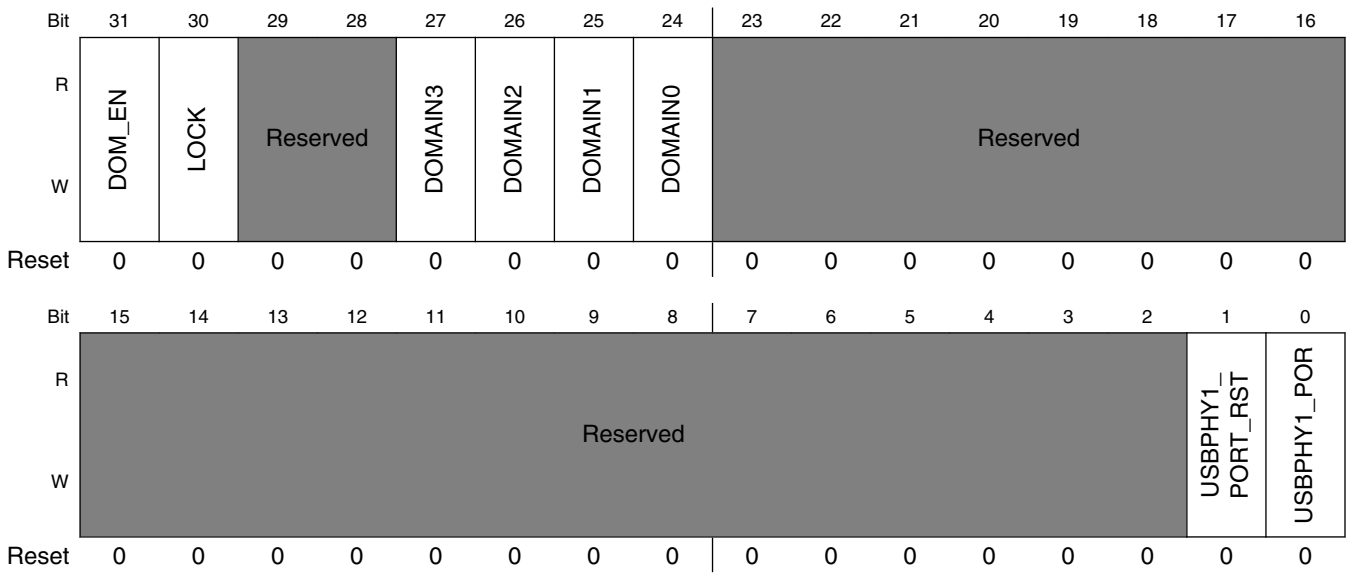
SRC_HSICPHY_RCR field descriptions (continued)

Field	Description
29–28 -	This field is reserved. Reserved
27 DOMAIN3	Domain3 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain3. The master from domain3 cannot write to this register. 1 This register is assigned to domain3. The master from domain3 can write to this register
26 DOMAIN2	Domain2 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain2. The master from domain3 cannot write to this register. 1 This register is assigned to domain2. The master from domain3 can write to this register
25 DOMAIN1	Domain1 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain1. The master from domain3 cannot write to this register. 1 This register is assigned to domain1. The master from domain3 can write to this register
24 DOMAIN0	Domain0 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain0. The master from domain3 cannot write to this register. 1 This register is assigned to domain0. The master from domain3 can write to this register
23–2 -	This field is reserved. Reserved
1 HSICPHY_ PORT_RST	HSIC PHY Port Reset 0 Do not assert HSIC PHY Port Reset 1 Assert HSIC PHY Port Reset
0 HSIC_PHY_POR	This field is reserved. HSIC PHY POR **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. Software can determine that the reset has finished once this bit is cleared. Software can also configure SRC to generate interrupt once the software has finished. Please refer to SRC_SISR register for details. 0 Do not assert HSIC PHY reset 1 Assert HSIC PHY reset

6.2.7.7 USB OTG PHY1 Reset Control Register (SRC_USBOPHY1_RCR)

The USB OTG PHY1 Reset Control Register (SRC_IP_RCR2), contains bits that control the USB OTG PHY1 reset generation.

Address: 3039_0000h base + 20h offset = 3039_0020h



SRC_USBOPHY1_RCR field descriptions

Field	Description
31 DOM_EN	Domain Control enable for this register. **NOTE:** [31:30] and [27:24] areas are not controlled by this bit. [31:30] and [27:24] area can be modified by any masters from any domains when Lock bit is not set. 0 Disables domain control. All of this register's bits except [31:30] and [27:24] can be modified by any masters 1 Enables domain control. All of this register's bits except [31:30] and [27:24] can only be modified by the masters from the domains specified in [27:24] area.
30 LOCK	Domain control bits lock **NOTE:** Lock bit is a write-once register, once it is set to 1, it can't be write to 0 0 [31] and [27:24] bits can be modified 1 [31] and [27:24] bits cannot be modified
29–28 -	This field is reserved. Reserved
27 DOMAIN3	Domain3 assignment control. Effective when dom_en is set to 1.

Table continues on the next page...

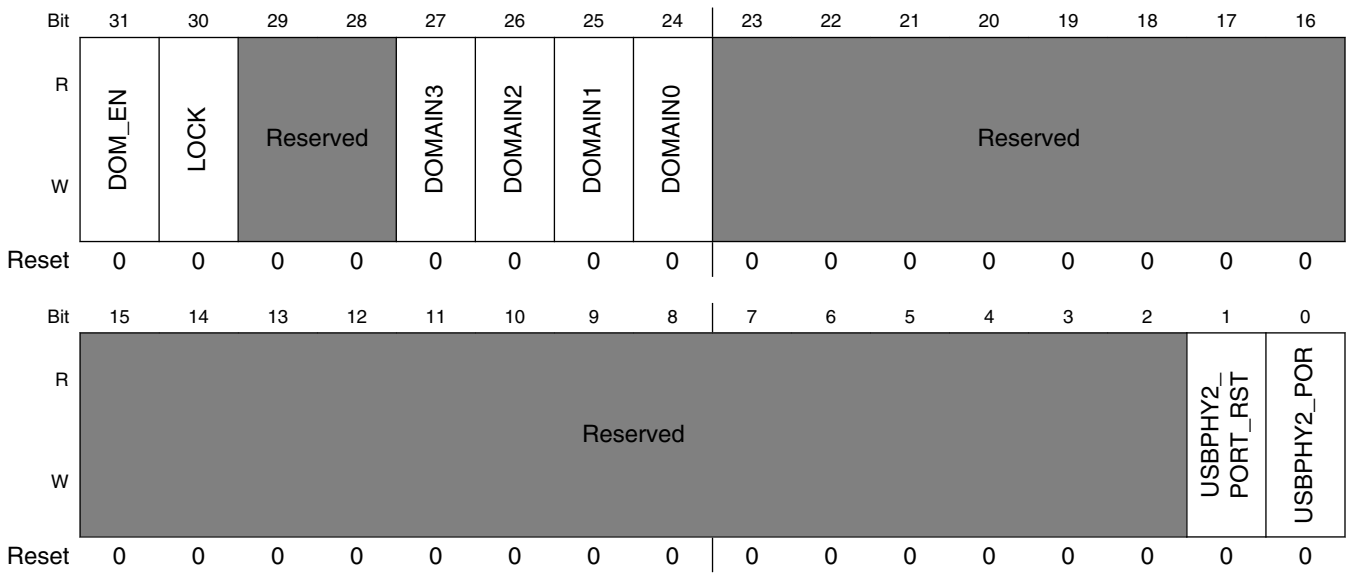
SRC_USBOPHY1_RCR field descriptions (continued)

Field	Description
	0 This register is not assigned to domain3. The master from domain3 cannot write to this register. 1 This register is assigned to domain3. The master from domain3 can write to this register
26 DOMAIN2	Domain2 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain2. The master from domain3 cannot write to this register. 1 This register is assigned to domain2. The master from domain3 can write to this register
25 DOMAIN1	Domain1 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain1. The master from domain3 cannot write to this register. 1 This register is assigned to domain1. The master from domain3 can write to this register
24 DOMAIN0	Domain0 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain0. The master from domain3 cannot write to this register. 1 This register is assigned to domain0. The master from domain3 can write to this register
23–2 -	This field is reserved. Reserved
1 USBPHY1_ PORT_RST	USB OTG PHY1 Port Reset 0 Do not assert USB OTG PHY1 Port Reset 1 Assert USB OTG PHY1 Port Reset
0 USBPHY1_POR	USB OTG PHY 1 POR **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. Software can determine that the reset has finished once this bit is cleared. Software can also configure SRC to generate interrupt once the software has finished. Please refer to SRC_SISR register for details 0 De-assert USB OTG PHY1 reset 1 Assert USB OTG PHY1 reset

6.2.7.8 USB OTG PHY2 Reset Control Register (SRC_USBOPHY2_RCR)

The USB OTG PHY2 Reset Control Register (SRC_IP_RCR2), contains bits that control the USB OTG PHY2 reset generation.

Address: 3039_0000h base + 24h offset = 3039_0024h



SRC_USBOPHY2_RCR field descriptions

Field	Description
31 DOM_EN	Domain Control enable for this register. **NOTE:** [31:30] and [27:24] areas are not controlled by this bit. [31:30] and [27:24] area can be modified by any masters from any domains when Lock bit is not set. 0 Disables domain control. All of this register's bits except [31:30] and [27:24] can be modified by any masters 1 Enables domain control. All of this register's bits except [31:30] and [27:24] can only be modified by the masters from the domains specified in [27:24] area.
30 LOCK	Domain control bits lock **NOTE:** Lock bit is a write-once register, once it is set to 1, it can't be write to 0 0 [31] and [27:24] bits can be modified 1 [31] and [27:24] bits cannot be modified
29–28 -	This field is reserved. Reserved
27 DOMAIN3	Domain3 assignment control. Effective when dom_en is set to 1.

Table continues on the next page...

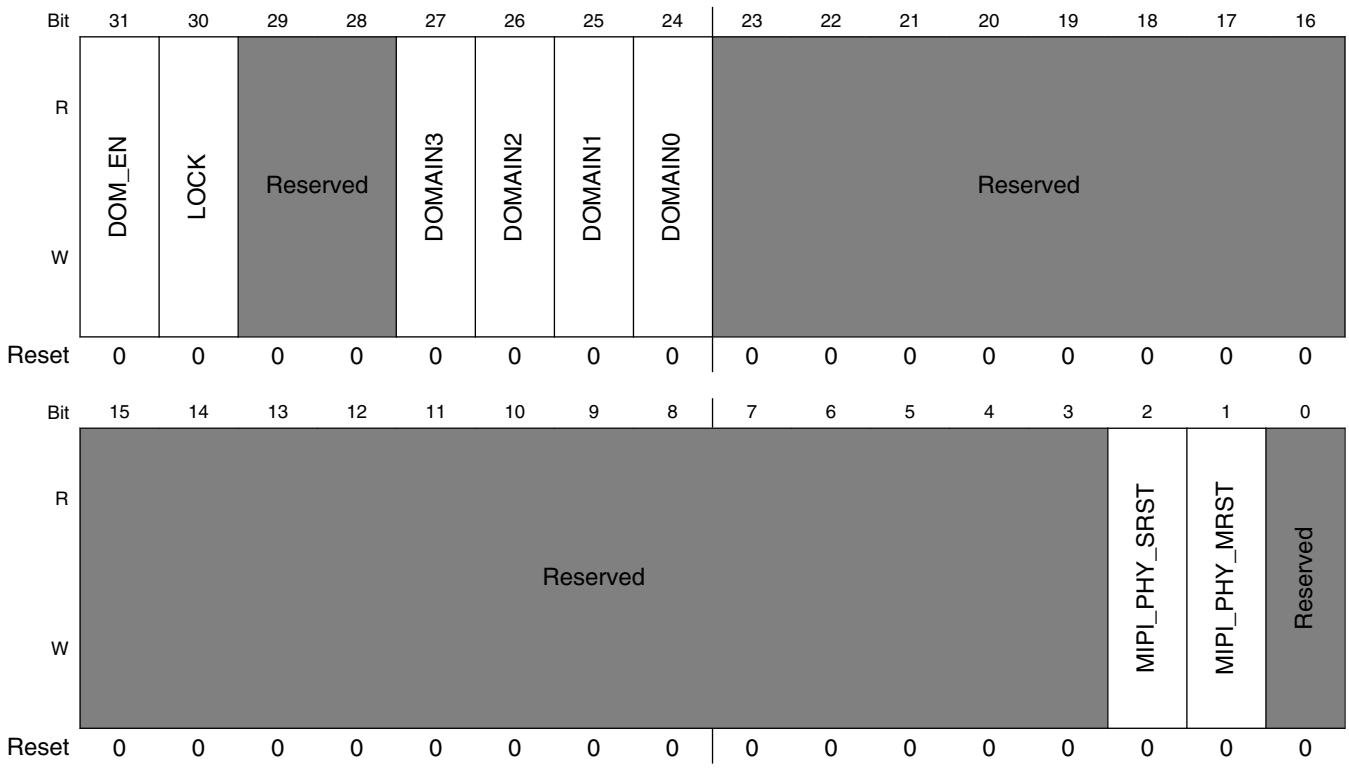
SRC_USBOPHY2_RCR field descriptions (continued)

Field	Description
	0 This register is not assigned to domain3. The master from domain3 cannot write to this register. 1 This register is assigned to domain3. The master from domain3 can write to this register
26 DOMAIN2	Domain2 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain2. The master from domain3 cannot write to this register. 1 This register is assigned to domain2. The master from domain3 can write to this register
25 DOMAIN1	Domain1 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain1. The master from domain3 cannot write to this register. 1 This register is assigned to domain1. The master from domain3 can write to this register
24 DOMAIN0	Domain0 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain0. The master from domain3 cannot write to this register. 1 This register is assigned to domain0. The master from domain3 can write to this register
23–2 -	This field is reserved. Reserved
1 USBPHY2_ PORT_RST	USB OTG PHY2 Port Reset 0 Do not assert USB OTG PHY2 Port Reset 1 Assert USB OTG PHY2 Port Reset
0 USBPHY2_POR	USB OTG PHY 2 POR **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. Software can determine that the reset has finished once this bit is cleared. Software can also configure SRC to generate interrupt once the software has finished. Please refer to SRC_SISR register for details 0 De-assert USB OTG PHY2 reset 1 Assert USB OTG PHY2 reset

6.2.7.9 MIPI PHY Reset Control Register (SRC_MIPIPHY_RCR)

The MIPI PHY Reset Control Register (SRC_MIPIPHY_RCR), contains bits that control the MIPI PHYreset generation.

Address: 3039_0000h base + 28h offset = 3039_0028h



SRC_MIPIPHY_RCR field descriptions

Field	Description
31 DOM_EN	Domain Control enable for this register. **NOTE:** [31:30] and [27:24] areas are not controlled by this bit. [31:30] and [27:24] area can be modified by any masters from any domains when Lock bit is not set. 0 Disables domain control. All of this register's bits except [31:30] and [27:24] can be modified by any masters 1 Enables domain control. All of this register's bits except [31:30] and [27:24] can only be modified by the masters from the domains specified in [27:24] area.
30 LOCK	Domain control bits lock **NOTE:** Lock bit is a write-once register, once it is set to 1, it can't be write to 0 0 [31] and [27:24] bits can be modified 1 [31] and [27:24] bits cannot be modified

Table continues on the next page...

SRC_MIPIPHY_RCR field descriptions (continued)

Field	Description
29–28 -	This field is reserved. Reserved
27 DOMAIN3	Domain3 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain3. The master from domain3 cannot write to this register. 1 This register is assigned to domain3. The master from domain3 can write to this register
26 DOMAIN2	Domain2 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain2. The master from domain3 cannot write to this register. 1 This register is assigned to domain2. The master from domain3 can write to this register
25 DOMAIN1	Domain1 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain1. The master from domain3 cannot write to this register. 1 This register is assigned to domain1. The master from domain3 can write to this register
24 DOMAIN0	Domain0 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain0. The master from domain3 cannot write to this register. 1 This register is assigned to domain0. The master from domain3 can write to this register
23–3 -	This field is reserved. Reserved
2 MIPI_PHY_SRST	MIPI PHY Slave Reset 0 Do not assert MIPI PHY Slave reset 1 Assert MIPI PHY Slave reset
1 MIPI_PHY_ MRST	MIPI PHY Master Reset 0 Do not assert MIPI PHY Master reset 1 Assert MIPI PHY Master reset
0 -	This field is reserved. Reserved

6.2.7.10 PCIE PHY Reset Control Register (SRC_PCIEPHY_RCR)

The PCIE PHY Control Register (SRC_PCIEPHY_RCR), contains bits that control the PCIE PHY reset generation.

Address: 3039_0000h base + 2Ch offset = 3039_002Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DOM_EN	LOCK	Reserved		DOMAIN3	DOMAIN2	DOMAIN1	DOMAIN0	Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved	PCIE_CTRL_SYS_INT	PCIE_CTRL_CFG_L1_MAC	PCIE_CTRL_CFG_L1_AUX	PCIE_CTRL_APPS_TURNOFF	PCIE_CTRL_APPS_PME	PCIE_CTRL_APPS_EXIT	PCIE_CTRL_APPS_ENTER	PCIE_CTRL_APPS_READY	PCIE_CTRL_APPS_EN	PCIE_CTRL_APPS_RST	PCIE_CTRL_APPS_CLK_REQ	PCIEPHY_PERST	PCIEPHY_BTN	PCIEPHY_G_RST	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

SRC_PCIEPHY_RCR field descriptions

Field	Description
31 DOM_EN	Domain Control enable for this register. **NOTE:** [31:30] and [27:24] areas are not controlled by this bit. [31:30] and [27:24] area can be modified by any masters from any domains when Lock bit is not set. 0 Disables domain control. All of this register's bits except [31:30] and [27:24] can be modified by any masters 1 Enables domain control. All of this register's bits except [31:30] and [27:24] can only be modified by the masters from the domains specified in [27:24] area.
30 LOCK	Domain control bits lock **NOTE:** Lock bit is a write-once register, once it is set to 1, it can't be write to 0 0 [31] and [27:24] bits can be modified 1 [31] and [27:24] bits cannot be modified

Table continues on the next page...

SRC_PCIEPHY_RCR field descriptions (continued)

Field	Description
29–28 -	This field is reserved. Reserved
27 DOMAIN3	Domain3 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain3. The master from domain3 cannot write to this register. 1 This register is assigned to domain3. The master from domain3 can write to this register
26 DOMAIN2	Domain2 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain2. The master from domain3 cannot write to this register. 1 This register is assigned to domain2. The master from domain3 can write to this register
25 DOMAIN1	Domain1 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain1. The master from domain3 cannot write to this register. 1 This register is assigned to domain1. The master from domain3 can write to this register
24 DOMAIN0	Domain0 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain0. The master from domain3 cannot write to this register. 1 This register is assigned to domain0. The master from domain3 can write to this register
23–15 -	This field is reserved. Reserved
14 PCIE_CTRL_ SYS_INT	PCIE_CTRL_SYS_INT
13 PCIE_CTRL_ CFG_L1_MAC	Pcie_ctrl_cfg_l1_mac_powerdown_override_to_p2_en
12 PCIE_CTRL_ CFG_L1_AUX	Pcie_ctrl_cfg_l1_aux_clk_switch_core_clk_gate_en
11 PCIE_CTRL_ APPS_ TURNOFF	Pcie_ctrl_apps_pm_xmt_turnoff
10 PCIE_CTRL_ APPS_PME	Pcie_ctrl_apps_pm_xmt_pme
9 PCIE_CTRL_ APPS_EXIT	Pcie_ctrl_app_req_exit_l1
8 PCIE_CTRL_ APPS_ENTER	Pcie_ctrl_app_req_entr_l1
7 PCIE_CTRL_ APPS_READY	Pcie_ctrl_app_ready_entr_l23
6 PCIE_CTRL_ APPS_EN	Pcie_ctrl_app_ltssm_enable

Table continues on the next page...

SRC_PCIEPHY_RCR field descriptions (continued)

Field	Description
5 PCIE_CTRL_ APPS_RST	Pcie_ctrl_app_init_rst
4 PCIE_CTRL_ APPS_CLK_REQ	Pcie_ctrl_app_clk_req_n
3 PCIEPHY_ PERST	Pciephy_perst
2 PCIEPHY_BTN	PCIE PHY button
1 PCIEPHY_G_ RST	PCIE PHY Global Reset
0 -	This field is reserved. Reserved

6.2.7.11 SRC Boot Mode Register 1 (SRC_SBMR1)

The Boot Mode register (SBMR) contains bits that reflect the status of Boot Mode Pins of the chip. The reset value is configuration dependent (depending on boot/fuses/IO pads).

Address: 3039_0000h base + 58h offset = 3039_0058h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BOOT_CFG4[7:0]								BOOT_CFG3[7:0]								BOOT_CFG2[7:0]								BOOT_CFG1[7:0]							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_SBMR1 field descriptions

Field	Description
31–24 BOOT_ CFG4[7:0]	Refer to fusemap.
23–16 BOOT_ CFG3[7:0]	Refer to fusemap.
15–8 BOOT_ CFG2[7:0]	Refer to fusemap.
BOOT_ CFG1[7:0]	Refer to fusemap.

6.2.7.12 SRC Reset Status Register (SRC_SRSR)

The SRSR is a write to one clear register which records the source of the reset events for the chip. The SRC reset status register will capture all the reset sources that have occurred. This register is reset on ipp_reset_b. This is a read-write register.

For bit[9-0] - writing zero does not have any effect. Writing one will clear the corresponding bit. The individual bits can be cleared by writing one to that bit. When the system comes out of reset, this register will have bits set corresponding to all the reset sources that occurred during system reset. Software has to take care to clear this register by writing one after every reset that occurs so that the register will contain the information of recently occurred reset.

Address: 3039_0000h base + 5Ch offset = 3039_005Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						tempsense_rst_b	wdog4_rst_b	wdog3_rst_b	jtag_sw_rst	jtag_rst_b	wdog1_rst_b	ipp_user_reset_b	csu_reset_b	0	ipp_reset_b
W								w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

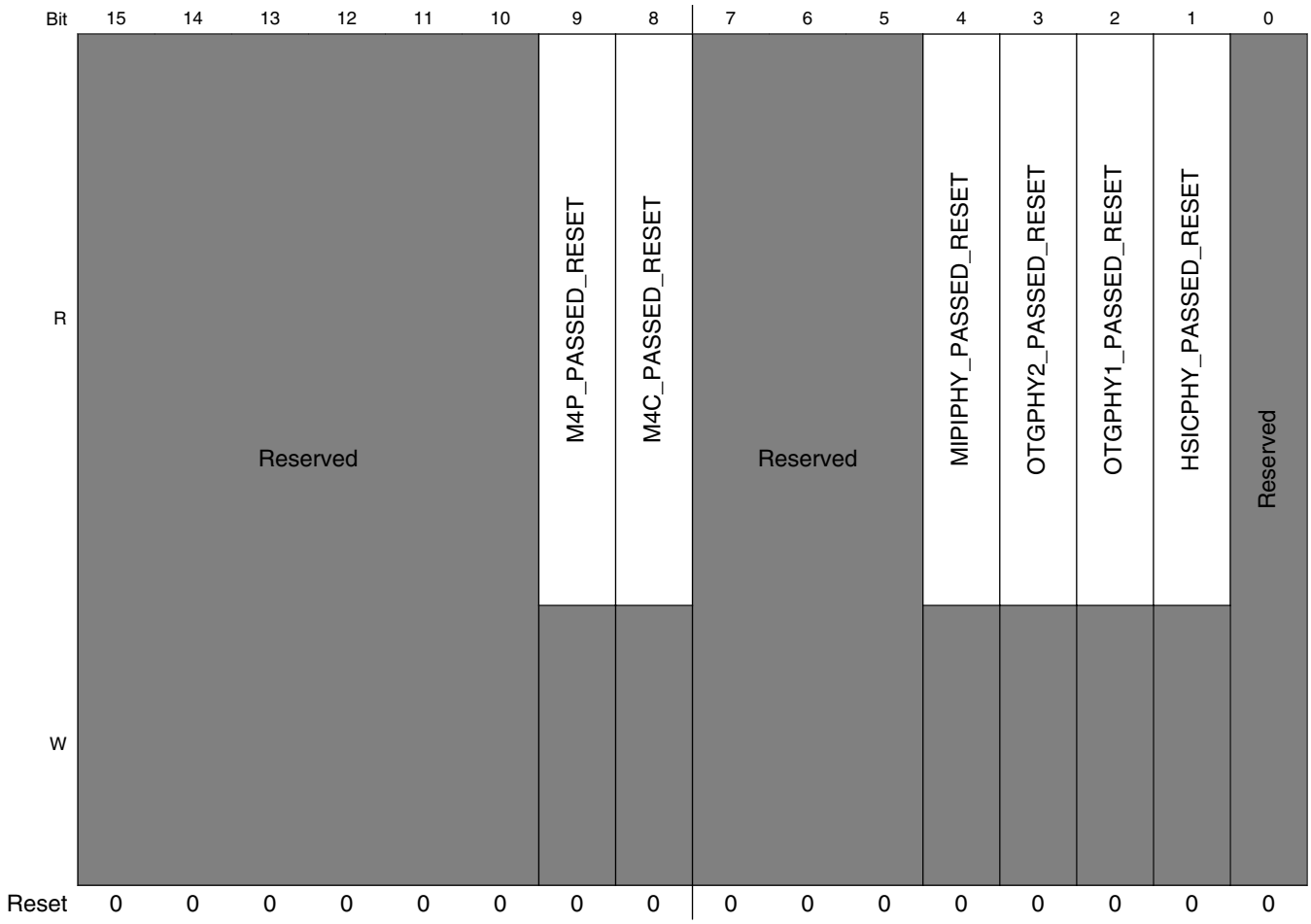
SRC_SRSR field descriptions

Field	Description
31–10 Reserved	This read-only field is reserved and always has the value 0.
9 tempsense_rst_b	Temper Sensor software reset. Indicates whether the reset was the result of software reset from on-chip Temperature Sensor. 0 Reset is not a result of software reset from Temperature Sensor. 1 Reset is a result of software reset from Temperature Sensor.
8 wdog4_rst_b	IC Watchdog4 Time-out reset. Indicates whether the reset was the result of the watchdog4 time-out event. 0 Reset is not a result of the watchdog4 time-out event. 1 Reset is a result of the watchdog4 time-out event.
7 wdog3_rst_b	IC Watchdog3 Time-out reset. Indicates whether the reset was the result of the watchdog3 time-out event. 0 Reset is not a result of the watchdog3 time-out event. 1 Reset is a result of the watchdog3 time-out event.
6 jtag_sw_rst	JTAG software reset. Indicates whether the reset was the result of software reset from JTAG. 0 Reset is not a result of software reset from JTAG. 1 Reset is a result of software reset from JTAG.
5 jtag_rst_b	HIGH - Z JTAG reset. Indicates whether the reset was the result of HIGH-Z reset from JTAG. 0 Reset is not a result of HIGH-Z reset from JTAG. 1 Reset is a result of HIGH-Z reset from JTAG.
4 wdog1_rst_b	IC Watchdog1 Time-out reset. Indicates whether the reset was the result of the watchdog time-out event. 0 Reset is not a result of the watchdog1 time-out event. 1 Reset is a result of the watchdog1 time-out event.
3 ipp_user_reset_b	Indicates whether the reset was the result of the ipp_user_reset_b qualified reset. 0 Reset is not a result of the ipp_user_reset_b qualified as COLD reset event. 1 Reset is a result of the ipp_user_reset_b qualified as COLD reset event.
2 csu_reset_b	Indicates whether the reset was the result of the csu_reset_b input. **NOTE:** If case the csu_reset_b occurred during a WARM reset process, during the phase that ipg_clk is not available yet, then the occurrence of CSU reset will not be reflected in this bit. 0 Reset is not a result of the csu_reset_b event. 1 Reset is a result of the csu_reset_b event.
1 Reserved	This read-only field is reserved and always has the value 0.
0 ipp_reset_b	Indicates whether reset was the result of ipp_reset_b pin (Power-up sequence) 0 Reset is not a result of ipp_reset_b pin. 1 Reset is a result of ipp_reset_b pin.

6.2.7.13 SRC Interrupt Status Register (SRC_SISR)

Address: 3039_0000h base + 68h offset = 3039_0068h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



SRC_SISR field descriptions

Field	Description
31–10 -	This field is reserved. Reserved
9 M4P_PASSED_ RESET	Interrupt generated to indicate that m4 platform passed software reset and is ready to be used 0 interrupt generated not due to m4 platform reset 1 interrupt generated due to m4 platform reset
8 M4C_PASSED_ RESET	Interrupt generated to indicate that m4 core passed software reset and is ready to be used 0 interrupt generated not due to m4 core reset 1 interrupt generated due to m4 core reset
7–5 -	This field is reserved. Reserved.
4 MIPIPHY_ PASSED_ RESET	Interrupt generated to indicate that MIPI PHY passed software reset and is ready to be used 0 Interrupt generated not due to MIPI PHY passed reset 1 Interrupt generated due to MIPI PHY passed reset

Table continues on the next page...

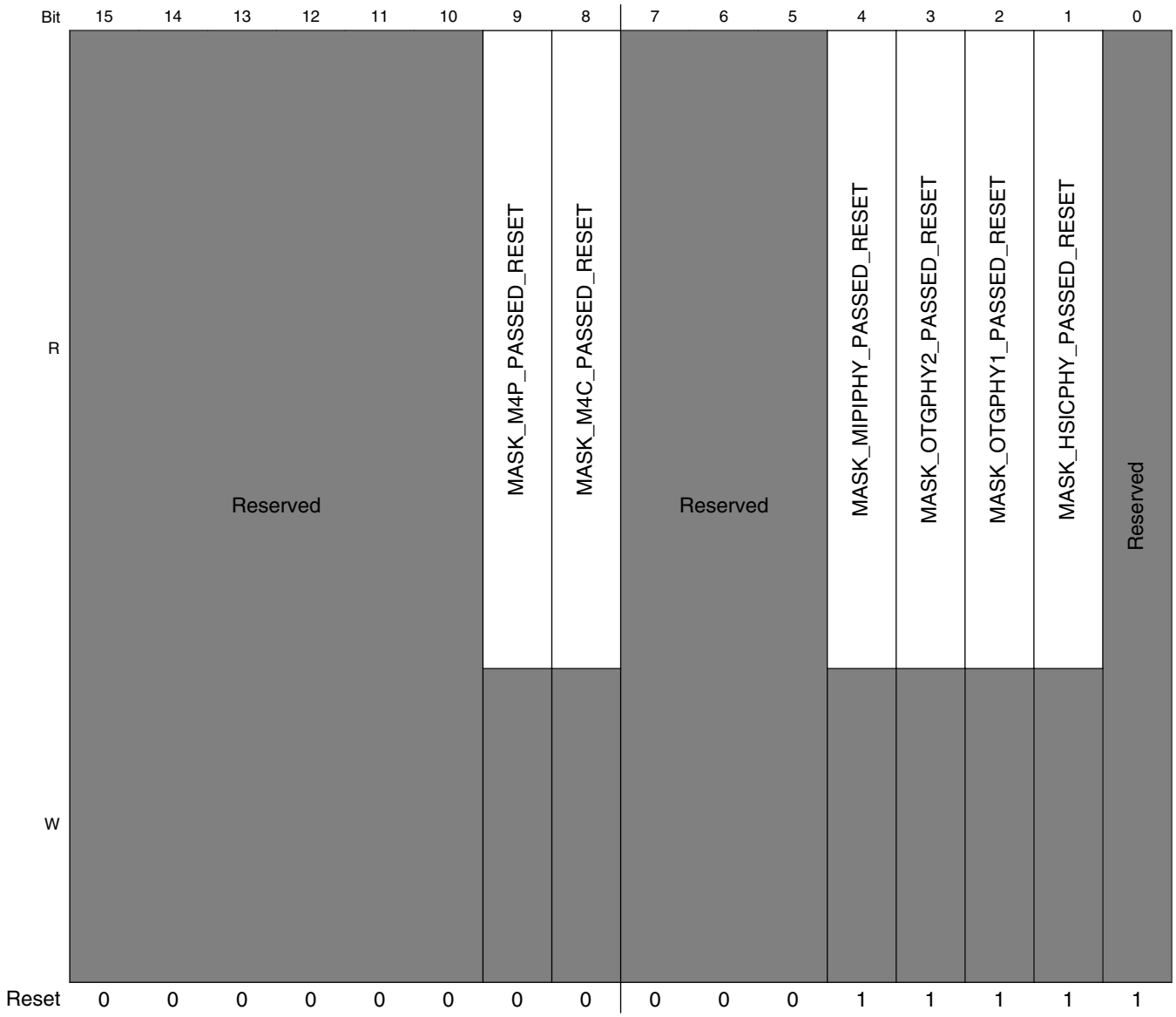
SRC_SISR field descriptions (continued)

Field	Description
3 OTGPHY2_ PASSED_ RESET	Interrupt generated to indicate that OTG PHY2 passed software reset and is ready to be used 0 Interrupt generated not due to OTG PHY2 passed reset 1 Interrupt generated due to OTG PHY2 passed reset
2 OTGPHY1_ PASSED_ RESET	Interrupt generated to indicate that OTG PHY1 passed software reset and is ready to be used 0 Interrupt generated not due to OTG PHY1 passed reset 1 Interrupt generated due to OTG PHY1 passed reset
1 HSICPHY_ PASSED_ RESET	Interrupt generated to indicate that HSIC PHY passed software reset and is ready to be used 0 Interrupt generated not due to HSIC PHY passed reset 1 Interrupt generated due to HSIC PHY passed reset
0 -	This field is reserved. Reserved

6.2.7.14 SRC Interrupt Mask Register (SRC_SIMR)

Address: 3039_0000h base + 6Ch offset = 3039_006Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



SRC_SIMR field descriptions

Field	Description
31–10 -	This field is reserved. Reserved
9 MASK_M4P_ PASSED_ RESET	mask interrupt generation due to m4 platform passed reset 0 do not mask interrupt due to m4 platform passed reset - interrupt will be created 1 mask interrupt due to m4platform passed reset
8 MASK_M4C_ PASSED_ RESET	mask interrupt generation due to m4 core passed reset 0 do not mask interrupt due to m4 core passed reset - interrupt will be created 1 mask interrupt due to m4 core passed reset
7–5 -	This field is reserved. Reserved.

Table continues on the next page...

SRC_SIMR field descriptions (continued)

Field	Description
4 MASK_ MIPIPHY_ PASSED_ RESET	mask interrupt generation due to MIPI PHY passed reset 0 do not mask interrupt due to MIPI PHY passed reset - interrupt will be created 1 mask interrupt due to MIPI PHY passed reset
3 MASK_ OTGPHY2_ PASSED_ RESET	mask interrupt generation due to OTG PHY2 passed reset 0 do not mask interrupt due to OTG PHY2 passed reset - interrupt will be created 1 mask interrupt due to OTG PHY2 passed reset
2 MASK_ OTGPHY1_ PASSED_ RESET	mask interrupt generation due to OTG PHY1 passed reset 0 do not mask interrupt due to OTG PHY1 passed reset - interrupt will be created 1 mask interrupt due to OTG PHY1 passed reset
1 MASK_ HSICPHY_ PASSED_ RESET	mask interrupt generation due to HSIC PHY passed reset 0 do not mask interrupt due to HSIC PHY passed reset - interrupt will be created 1 mask interrupt due to HSIC PHY passed reset
0 -	This field is reserved. Reserved

6.2.7.15 SRC Boot Mode Register 2 (SRC_SBMR2)

The Boot Mode register (SBMR), contains bits that reflect the status of Boot Mode Pins of the chip. The default values for those bits depends on the values of pins/fuses during reset sequence, hence the question mark on their default value.

Address: 3039_0000h base + 70h offset = 3039_0070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0						BMOD[1:0]		0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0											BT_FUSE_SEL	DIR_BT_DIS	0	SEC_CONFIG[1:0]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_SBMR2 field descriptions

Field	Description
31–26 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

SRC_SBMR2 field descriptions (continued)

Field	Description
25–24 BMOD[1:0]	BMOD[1:0] shows the latched state of the BOOT_MODE1 and BOOT_MODE0 signals on the rising edge of POR_B. See the Boot mode pin settings section of System Boot.
23–5 Reserved	This read-only field is reserved and always has the value 0.
4 BT_FUSE_SEL	BT_FUSE_SEL (connected to gpio bt_fuse_sel) shows the state of the BT_FUSE_SEL fuse. See Fusemap for additional information on this fuse.
3 DIR_BT_DIS	DIR_BT_DIS shows the state of the DIR_BT_DIS fuse. See Chapter 5, Fusemap for additional information on this fuse.
2 Reserved	This read-only field is reserved and always has the value 0.
SEC_ CONFIG[1:0]	SECONFIG[1] shows the state of the SECONFIG[1] fuse. See Fusemap for additional information on this fuse. SECONFIG[0] shows the state of the SECONFIG[0] fuse. This fuse is shown as reserved in Fusemap (address 0x440[1]) because it does not have a user-relevant function.

6.2.7.16 SRC General Purpose Register 1 (SRC_GPR1)

Address: 3039_0000h base + 74h offset = 3039_0074h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ENTRY0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR1 field descriptions

Field	Description
PERSISTENT_ ENTRY0	Holds entry function for core0 for waking-up from low power mode. The SRC ensures that the register value will persist across system resets.

6.2.7.17 SRC General Purpose Register 2 (SRC_GPR2)

Address: 3039_0000h base + 78h offset = 3039_0078h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ARG0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR2 field descriptions

Field	Description
PERSISTENT_ ARG0	Holds argument of entry function for core0 for waking-up from low power mode. The SRC ensures that the register value will persist across system resets.

6.2.7.18 SRC General Purpose Register 3 (SRC_GPR3)

Address: 3039_0000h base + 7Ch offset = 3039_007Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ENTRY1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR3 field descriptions

Field	Description
PERSISTENT_ENTRY1	Holds entry function for core1. The SRC ensures that the register value will persist across system resets.

6.2.7.19 SRC General Purpose Register 4 (SRC_GPR4)

Address: 3039_0000h base + 80h offset = 3039_0080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ARG1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR4 field descriptions

Field	Description
PERSISTENT_ARG1	Holds argument of entry function for core1. The SRC ensures that the register value will persist across system resets.

6.2.7.20 SRC General Purpose Register 5 (SRC_GPR5)

Address: 3039_0000h base + 84h offset = 3039_0084h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR5 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

6.2.7.21 SRC General Purpose Register 6 (SRC_GPR6)

Address: 3039_0000h base + 88h offset = 3039_0088h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR6 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

6.2.7.22 SRC General Purpose Register 7 (SRC_GPR7)

Address: 3039_0000h base + 8Ch offset = 3039_008Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR7 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

6.2.7.23 SRC General Purpose Register 8 (SRC_GPR8)

Address: 3039_0000h base + 90h offset = 3039_0090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR8 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

6.2.7.24 SRC General Purpose Register 9 (SRC_GPR9)

Reserved for Internal Use.

Address: 3039_0000h base + 94h offset = 3039_0094h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR9 field descriptions

Field	Description
-	This field is reserved. Reserved.

6.2.7.25 SRC General Purpose Register 10 (SRC_GPR10)

Address: 3039_0000h base + 98h offset = 3039_0098h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-						Reserved		-							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR10 field descriptions

Field	Description
31–26 -	Read/write bits, for general purpose **NOTE:** Reset only by POR
25 -	This field is reserved. Reserved.
-	Read/write bits, for general purpose **NOTE:** Reset only by POR

6.2.7.26 SRC DDR Controller Reset Control Register (SRC_DDRC_RCR)

DDR Controller Reset Control Register

Address: 3039_0000h base + 1000h offset = 3039_1000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DOM_EN	LOCK	Reserved		DOMAIN3	DOMAIN2	DOMAIN1	DOMAIN0	Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DDRC_CORE_RST	DDRC_PRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

SRC_DDRC_RCR field descriptions

Field	Description
31 DOM_EN	Domain Control enable for this register. **NOTE:** [31:30] and [27:24] areas are not controlled by this bit. [31:30] and [27:24] area can be modified by any masters from any domains when Lock bit is not set. 0 Disables domain control. All of this register's bits except [31:30] and [27:24] can be modified by any masters 1 Enables domain control. All of this register's bits except [31:30] and [27:24] can only be modified by the masters from the domains specified in [27:24] area.

Table continues on the next page...

SRC_DDRC_RCR field descriptions (continued)

Field	Description
30 LOCK	Domain control bits lock **NOTE:** Lock bit is a write-once register, once it is set to 1, it can't be write to 0 0 [31] and [27:24] bits can be modified 1 [31] and [27:24] bits cannot be modified
29–28 -	This field is reserved. Reserved
27 DOMAIN3	Domain3 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain3. The master from domain3 cannot write to this register. 1 This register is assigned to domain3. The master from domain3 can write to this register
26 DOMAIN2	Domain2 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain2. The master from domain3 cannot write to this register. 1 This register is assigned to domain2. The master from domain3 can write to this register
25 DOMAIN1	Domain1 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain1. The master from domain3 cannot write to this register. 1 This register is assigned to domain1. The master from domain3 can write to this register
24 DOMAIN0	Domain0 assignment control. Effective when dom_en is set to 1. 0 This register is not assigned to domain0. The master from domain3 cannot write to this register. 1 This register is assigned to domain0. The master from domain3 can write to this register
23–2 -	This field is reserved. Reserved
1 DDRC_CORE_RST	DDR Controller core_ddrc_rstn and aresetn. This bit is set to 1 automatically when fastmix is powered off. 0 De-ssert DDR controller aresetn and core_ddrc_rstn 1 Assert DDR Controller preset and DDR PHY reset
0 DDRC_PRST	DDR Controller preset and DDR PHY reset. This bit is set to 1 automatically when fastmix is powered off. **NOTE:** This reset can only be released when DDR Controller clock inputs are active and stable for 30 cycles 0 De-ssert DDR Controller preset and DDR PHY reset reset 1 Assert DDR Controller preset and DDR PHY reset

6.3 Fusemap

6.3.1 Boot Fusemap

The following section details the various modes and selection of the required boot devices.

A separate map is given for each and every boot device. The device select is specified by BOOT_CFG[15:12] fuses listed in Table 6-10.

Table 6-10. Boot Device Select

Boot Device	BOOT_CFG[15]	BOOT_CFG[14]	BOOT_CFG[13]	BOOT_CFG[12]	BOOT_CFG[11]
SD/eSD	0	0	0	1	x
MMC/eMMC	0	0	1	0	x
NAND	0	0	1	1	x
QSPI	0	1	0	0	x
EIM	0	1	0	1	Memory Type: 0 - NOR Flash 1 - OneNAND
SPI/NOR	0	1	1	0	x

NOTE

Fuses marked as “Reserved” are reserved for Freescale internal (and future) use only. Customers should not attempt to burn these, as the IC behavior may be unpredictable. The reserved fuses can be read as either 0 or 1.

Table 6-11. SD/eSD Boot Fusemap

Addr	7	6	5	4	3	2	1	0
0x470[7:0]	Fast Boot: 0 - Regular 1 - Fast Boot	Reserved		Bus Width: 0 - 1-bit 1 - 4-bit	Speed 000 - Normal/SDR12 001 - High/SDR25 010 - SDR50 011 - SDR104 101 - Reserved for DDR50 Others - Reserved			Reserved

Table 6-12. MMC/eMMC Boot Fusemap

Addr	7	6	5	4	3	2	1	0
0x470[7:0]	Fast Boot: 0 - Regular 1 - Fast Boot	Bus Width: 000 - 1-bit 001 - 4-bit 010 - 8-bit 101 - 4-bit DDR (MMC 4.4) 110 - 8-bit DDR (MMC 4.4) Else - reserved			Speed 00 - Normal 01 - High 10 - Reserved for HS200 11 - Reserved		USDHC1 IO VOLTAGE SELECTION 0 - 3.3V 1 - 1.8V	USDHC2 IO VOLTAGE SELECTION 0 - 3.3V 1 - 1.8V

Table 6-13. NAND Boot Fusemap

Addr	7	6	5	4	3	2	1	0
0x470[7:0]	BT_TOGGLEMODE	BOOT_SEARCH_COUNT 00 - 2 01 - 2 10 - 4 11 - 8		Toggle Mode 33MHz Preamble Delay, Read Latency 000 - 16 GPMICLK cycles 001 - 1 GPMICLK cycles 010 - 2 GPMICLK cycles 011 - 3 GPMICLK cycles 100 - 4 GPMICLK cycles 101 - 5 GPMICLK cycles 110 - 6 GPMICLK cycles 111 - 7 GPMICLK cycles 1111- 15 GPMICLK cycles				Reserved

Table 6-14. QSPI Boot Fusemap

Addr	7	6	5	4	3	2	1	0
0x470[7:0]	HSPHS: Half Speed Phase Selection 0 - select sampling at non-inverted clock 1 - select sampling at inverted clock	HSDLY: Half Speed Delay selection 0 - one clock delay 1 - two clock delay	FSPHS: Full Speed Phase Selection 0 - select sampling at non-inverted clock 1 - select sampling at inverted clock	FSDLY: Full Speed Delay selection 0 - one clock delay 1 - two clock delay	Reserved			

Table 6-15. EIM Boot Fusemap

Addr	7	6	5	4	3	2	1	0
0x470[7:0]	OneNand Page Size 00 - 1KB 01 - 2KB 10 - 4KB 11 - Reserved		Reserved					

Table 6-16. SPI/NOR Boot Fusemap

Addr	7	6	5	4	3	2	1	0
0x470[7:0]	CS select (SPI only) 00 - CS#0 (default)		Reserved					

Table 6-16. SPI/NOR Boot Fusemap

Addr	7	6	5	4	3	2	1	0
	01 - CS#1 10 - CS#2 1 - CS#3							

Table 6-17. Boot Fusemap

Addr	7	6	5	4	3	2	1	0
0x480[7:0]	Reserved							
0x480[15:8]	LPB_BOOT (Core/DDR/ Bus) 00/01 - LPB Disable 10 - Div by2 11 - Div by 4		BT_LPB_POLARITY (GPIO polarity)	L1 I-Cache DISABLE	TZASC_ENABLE	WDOG_ENABLE 0 - Disabled 1 - Enabled	Reserved	
0x480[23:16]	Reserved	WDOG Timeout Select 000 - 64s 001 - 32s 010 - 16s 011 - 8s 100 - 4s Others - Reserved						
0x480[31:24]	Recovery Port Select 000 - eCSPI1 001 - eCSPI2 010 - eCSPI3 011 - eCSPI4			Recovery SPI Addressing 0 - 3-bytes (24-bit) 1 - 2-bytes (16-bit)	Recovery CS select (SPI only) 00 - CS#0 (default) 01 - CS#1 10 - CS#2 11 - CS#3		Reserved	
0x490[7:0]	Reserved for uSDHC future use							
0x490[15:8]	USDHC DLL Select 0 - DLL Slave Mode for 1 - DLL Override Mode	MMC_DLL_DLY[6:0] Delay target for USDHC DLL, it is applied to slave mode target delay or override mode target delay depends on DLL Override fuse bit value.						
0x490[23:16]	Disable SDMMC Manufacture mode 0 - Enable 1 - Disable	USDHC_CMD_OE_PRE_EN (SD/MMC debug)	eMMC 4.4 - RESET TO PRE-IDLE STATE 0 - enable 1 - disable	USDHC_PAD_PULL_DOWN 0 - no action	ENABLE_EMMC_22K_PULLUP 0 - 47K pullup	USDHC_IOMUX_SION_BIT_ENABLE 0 - Disable 1 - Enable	USDHC Override Pad Settings (using PAD_SETTINGS value)	USDHC DLL_ENABLE 0 - Disable DLL for SD/ eMMC

Table continues on the next page...

Table 6-17. Boot Fusemap (continued)

Addr	7	6	5	4	3	2	1	0
				1 - pull down	1 - 22K pullup			1 - Enable DLL for SD/ eMMC
0x490[31:24]	USDHCPAD_SETTINGS[7:0]							
0x4A0[7:0]	SD Calibration Step 00 - 1 TBD		uSDHC Power Cycle Interval 00 - 20ms 01 - 10ms 10 - 5ms 11 - 2.5		uSDHC Power Cycle Delay 0 - 5ms 1 - 2.5ms	uSDHC Power On Polarity 0 - Low 1 - High	USDHC3 IO VOLTAGE SELECTION 0 - 3.3V 1 - 1.8V	Fast Boot Acknowledge Disable 0 - Boot Ack Disabled 1 - Boot Ack Enabled
0x4A0[15:8]	NAND_READ_CMD_CODE1[7:0]							
0x4A0[23:16]	NAND_READ_CMD_CODE2[7:0]							
0x4A0[31:24]	NAND_PAD_SETTINGS[7:0]							
0x4B0[7:0]	Override NAND Pad Settings (using PAD_SETTINGS value)	GPMI Read DDR DLL Target Value 0000 - 7 0001 - 1 0111 - 0 1111 - 15				Reserved for NAND future use		
0x4B0[15:8]	READ_RETRY_SEQ_ID[3:0] 0000 - dont use read retry(RR) sequence embedded in ROM 0001 - Micron 20nm RR sequence 0010 - Toshiba A19nm RR sequence 0011 - Toshiba 19nm RR sequence 0100 - SanDisk 19nm RR sequence 0101 - SanDisk 1ynmRR sequence Others - Reserved				Reserved			
0x4B0[23:16]	SPI SS Active 0 - Active low 1 - Active high	SPI SCLK Active 0 - Active high 1 - Active low	SPI Clock/ Data Phase Control 0 - Tx on falling, Rx on rising 1 - Tx on rising, Rx on falling	Reserved				
0x4B0[31:24]	RNG_TRIM[7:0]							

6.3.2 Lock Fusemap

Table 6-18 describes the functions of various lock fuses.

Table 6-18. Lock Fuses

Addr	7	6	5	4	3	2	1	0
0x400[7:0]	ANALOG_LOCK 1x - OP x1 - WP		MEM_TRIM_LOCK 1x - OP x1 - WP		BOOT_CFG_LOCK 1x - OP x1 - WP		TESTER_LOCK 1x - OP x1 - WP	
0x400[15:8]	MAC_ADDR_LOCK 1x - OP x1 - WP		USB_ID_LOCK 1x - OP x1 - WP		Reserved	SJC_RESP_LOCK WRP,OP,RDP	SRK_LOCK	OTPMK_LOCK 1 - RP, WP, OP
0x400[23:16]	GP2_LOCK 1x - OP x1 - WP		GP1_LOCK 1x - OP x1 - WP		Reserved		ROM_PATCH_LOCK 1 - WP + OP of ROM_PATCH fuses	MANUFACTURE_KEY_LOCK 1 - RP, WP, OP
0x400[31:24]	CRC-GP2_LOCK 1x – WP + Read Protect (CRC locking) x1 – WP + OP (General Purpose use)		CRC-GP1_LOCK 1x – WP + Read Protect (CRC locking) x1 – WP + OP (General Purpose use)		Reserved			

6.3.3 Fusemap Descriptions Table

Table 6-19. Fusemap Descriptions

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Used by
0x400[1:0]	TESTER_LOCK	2			OCOTP
0x400[3:2]	BOOT_CFG_LOCK	2	Perform lock on BOOT related fuses.	0 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) 1 - Lock (The controlled field can be read or sensed only)	OCOTP
0x400[10]	SJC_RESP_LOCK	1			OCOTP
0x400[15:14]	MAC_ADDR_LOCK	2	Lock MAC_ADDR fuses.		OCOTP
0x400[21:20]	GP1_LOCK	2	Lock for General Purpose fuse register #1 (GP1)		OCOTP

Table continues on the next page...

Table 6-19. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Used by
0x400[23:22]	GP2_LOCK	2	Lock for General Purpose fuse register #2 (GP2)		OCOTP
0x400[9]	SRK_LOCK	1	Locking SRK_HASH[255:0]		OCOTP
0x410[31:0]	[31:0] in LOT_NO_ENC[42:0](SJC_CHALL/ UNIQUE_ID[42:0])	32	FSL-wide unique, encoded LOT ID STD II/SJC CHALLENGE/ Unique ID		SJC, SW
0x420[10:0]	[42:32] in LOT_NO_ENC[42:0](SJC_CHALL/ UNIQUE_ID[42:0])	11	FSL-wide unique, encoded LOT ID STD II/SJC CHALLENGE/ Unique ID		SJC, SW
0x420[15:11]	WAFER_NO[4:0] (SJC_CHALL[47:43] / UNIQUE_ID[47:43])	5	The wafer number of the wafer on which the device was fabricated/SJC CHALLENGE/ Unique ID		SJC, SW
0x420[23:16]	DIE-Y-CORDINATE[7:0] (SJC_CHALL[55:48] / UNIQUE_ID[55:48])	8	The Y-coordinate of the die location on the wafer/SJC CHALLENGE/ Unique ID		SJC, SW
0x420[31:24]	DIE-X-CORDINATE[7:0] (SJC_CHALL[63:56] / UNIQUE_ID[63:56])	8	The X-coordinate of the die location on the wafer/SJC CHALLENGE/ Unique ID		SJC, SW
0x440[3:0]	SI_REV[3:0]	4	Silicon Revision number. Can be used in conjunction with HW value set in GPR registers. (gpr4[15:8])		SW
0x440[9:8]	SPEED_GRADING[1:0]	2	Burned by tester program, for indicating IC core speed. (Hot burn may not be used).	FCA[5:4] FHA[3:2] FRAL[1:0] MHz P/N Code xx xx 00 800 08 xx xx 01 500 05 xx xx 10 1000 10 xx xx 11 1200 12	PROD / SW
0x450[0]	NUM_A7_CORES	1	Indicates the type of device : 1x cores or 2x cores.	'0' - 2x A7 Cores '1' - 1x A7 Cores	SRC, SJC, SW
0x450[8]	M4_DISABLE	1	Disable M4 Core.	0 - enabled 1 - disabled	M4
0x450[9]	M4_MPU_DISABLE	1	Disable M4 MPU IP.	0 - enabled 1 - disabled	M4
0x450[10]	M4_FPU_DISABLE	1	Disable M4 FPU IP.	0 - enabled 1 - disabled	M4
0x450[21]	FLEXCAN_DISABLE	1	Disable FlexCAN IP.	0 - enabled 1 - disabled	FLEXCAN
0x450[22]	ADC_DISABLE	1	Disable ADC IP.	0 - enabled 1 - disabled	ADC
0x450[24]	LCDIF_DISABLE	1	Disable LCDIF IP.	0 - enabled 1 - disabled	LCDIF

Table continues on the next page...

Table 6-19. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Used by
0x450[25]	CSI_DISABLE	1	Disable CSI IP.	0 - enabled 1 - disabled	CSI
0x450[26]	PXP_DISABLE	1	Disable PXP IP	0 - enabled 1 - disabled	PXP
0x450[27]	PXP_WFE_DISABLE	1	Disable WFE Module in PXP IP	0 - enabled 1 - disabled	PXP
0x450[29]	ENET1_DISABLE	1	Disable ENET IP 1st instance.	0 - enabled 1 - disabled	ENET 1
0x450[31]	MIPI_DISABLE	1	Disable MIPI IP.	0 - enabled 1 - disabled	MIPI
0x470[19:0]	BOOT_CFG	20	BOOT configuration register, Usage varies, depending on selected boot device.	See "Boot" sheet	SRC SW(ROM)
0x470[25]	SEC_CONFIG[1]	1	Security Configuration (with SEC_CONFIG[0])	00 - FAB (Open) 01 - Open - allows any code to be flashed and executed, even if it has no valid signature. 1x - Closed (Security On)	SW (ROM), SRC, SNVS, TPSMP
0x470[27]	DIR_BT_DIS	1	Direct External Memory Boot Disable	0 - Direct boot from external memory is allowed 1 - Direct boot from external memory is not allowed	SRC SW(ROM)
0x470[28]	BT_FUSE_SEL	1	Determines, whether using fuses for boot configuration, or GPIO /Serial loader.	If boot_mode="00" (Development) 0=Boot mode configuration is taken from GPIOs. 1=Boot mode configuration is taken from fuses. If boot_mode="10" (Production) 0 - Boot using Serila Loader (USB) 1- Boot mode configuration is taken from fuses.	SRC SW(ROM)
0x470[29]	FORCE_COLD_BOOT(SBMR)	1	Force cold boot when A7 core come out of reset. Reflected in SBMR reg of SRC	Fuse Function: 0 – Default behavior equivalent to the rest of the i.MX6 family allowing a fast recovery from low power modes. That is, the ROM is allowed to jump to the address previously programmed in the SRC persistent register. 1 – Fast recovery path in the ROM is not allowed and a cold boot is always performed. Customers wanting a higher level of security should burn this fuse.	SRC SW(ROM)
0x480[31:0]	BOOT_CFG_PARAMETER	32	BOOT configuration parameters, Usage varies, depending on selected boot device.	See Boot Fusemap table for details.	SW (ROM)

Table continues on the next page...

Table 6-19. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Used by
0x490[31:0]	BOOT_CFG_PARAMETER	32	BOOT configuration parameters, Usage varies, depending on selected boot device.	See Boot Fusemap table for details.	SW (ROM)
0x4A0[31:0]	BOOT_CFG_PARAMETER	32	BOOT configuration parameters, Usage varies, depending on selected boot device.	See Boot Fusemap table for details.	SW (ROM)
0x4B0[31:0]	BOOT_CFG_PARAMETER	32	BOOT configuration parameters, Usage varies, depending on selected boot device.	See Boot Fusemap table for details.	SW (ROM)
0x580[31:0]	SRK_HASH[255:0]	256	SRK key, no HW visible lines. NO HW Visible signals available		SW (HAB)
0x600[23:0]	SJC_RESP[55:0]	56	Response reference value for the secure JTAG controller		SJC
0x620[15:0]	USB_VID[31:0]	16	USB VID		SW
0x620[31:16]	USB_PID[31:0]	16	USB PID		SW
0x640[15:0]	MAC1_ADDR[47:0]	48	Reserved for customers/SW		SW
0x650[31:16]	MAC2_ADDR[47:0]	48	Reserved for customers/SW		SW
0x780[31:0]	GP1[63:0]	64	General Purpose fuse register #1		SW
0x7A0[31:0]	GP2[63:0]	64	General Purpose fuse register #2		SW

6.4 On-Chip OTP Controller (OCOTP_CTRL)

6.4.1 Overview

This section contains information describing the requirements for the on-chip eFuse OTP controller along with details about the block functionality and implementation.

In this document, the words "eFuse" and "OTP" are interchangeable. OCOTP refers to the hardware block itself.

6.4.1.1 Features

The OCOTP provides the following features :

- 16 128-bit fuse bank.
- Bank based, restricted program, and read to 4 kbit of eFuse OTP.
- Loading and housing of fuse content into shadow registers.
- Memory-mapped (restricted) access to 4 kbit of shadow registers.
- Generation of HWV_FUSE (hardware visible fuse bus) and the HWV_REG bus which is made of up of volatile PIO register based "fuses". The HWV_REG bits come from the SCS (Software Controllable Signals) register.
- Generation of STICKY_REG which is consist of sticky register bits.
- Provide program-protect and read-protect eFuse.
- Provide override and read protection of shadow register.
- CRC32 test for read-lock fuse content.

6.4.2 Clocks

The table found here describes the clock sources for OCOTP.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 6-20. OCOTP Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock

6.4.3 Top-Level Symbol and Functional Overview

The figure found here shows the OCOTP system level diagram.

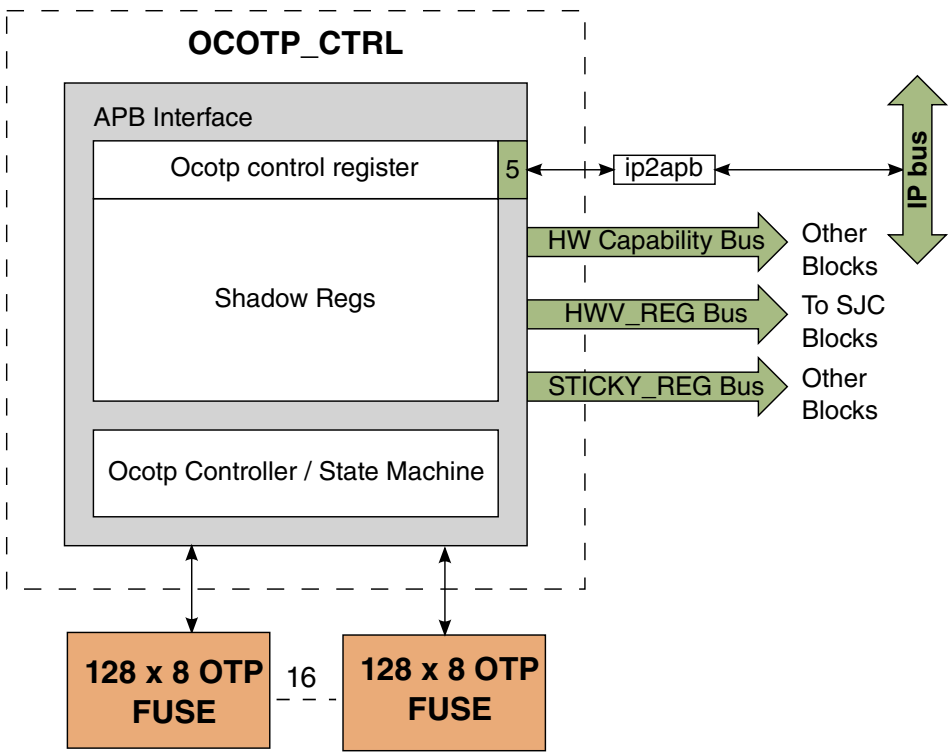


Figure 6-11. OCOTP System Level Diagram

6.4.3.1 Operation

The IP bus interface of the OCOTP provides two functions.

- Configure control registers for programming and reading fuse bank.
- Override and read shadow registers.

4-kbit fuses are physically divided into 16 128-bit fuse banks. Each fuse bank contains four words. OCOTP configuration for program and read are performed on fuse bank. For write, the four program words HW OCOTP DATAx (x = 0, 1, 2, 3) reflects the "write-mask". Bit fields with 0 are not programmed and bit fields with 1 are programmed. OCOTP programs bit field with 1 in the fuse words bit by bit. For reads, OCOTP read 4 fuse words in same fuse bank and put 4 read fuse words into HW_OCOTP_READ_FUSE_DATAx (x = 0, 1, 2, 3).

6.4.3.1.1 Shadow Register Reload

All fuse words in fuse bank are shadowed. Therefore, fuse information is available through memory mapped shadow registers. If fuses are subsequently programmed, the shadow registers should be reloaded to keep them coherent with the fuse bank arrays.

The "reload shadows" feature allows the user to force a reload of the shadow registers (including HW_OCOTP_LOCK) without having to reset the device. To force a reload, complete the following steps:

1. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write , read or reload must be completed before a new access can be requested.
2. Set the HW_OCOTP_CTRL[RELOAD_SHADOWS] bit. OCOTP will read all the fuse banks one by one and put it into corresponding shadow register.
3. Wait for HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[RELOAD_SHADOWS] to be cleared by the controller.

The controller will automatically clear the HW_OCOTP_CTRL[RELOAD_SHADOWS] bit after the successful completion of the operation.

6.4.3.1.2 Fuse and Shadow register read

All shadow registers are always readable through the APB bus except some secret keys regions. When their corresponding fuse lock bits are set, the shadow registers also become read locked. After read locking, reading from these registers will return 0xBADABADA.

In addition HW_OCOTP_CTRL[ERROR] will be set. It must be cleared by software before any new write , read or reload access can be issued. Subsequent reads to unlocked shadow locations will still work successfully however.

To read fuse words directly from fuse bank correctly complete the following steps:

1. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write, read or reload must be completed before a read access can be requested.
2. Write the requested fuse bank number (0 -15) to HW_OCOTP_CTRL[ADDR].
3. Set HW_OCOTP_READ_CTRL[READ_FUSE] to 1. OCOTP read whole 128 bits fuse content in the fuse bank which is assigned by HW_OCOTP_CTRL[ADDR]. Then put read value into HW_OCOTP_READ_FUSE_DATAx (x = 0, 1, 2, 3) registers.
4. If all the 4 fuse words in the fuse bank are read protected or locked, a read request will result in no OTP access and no setting in HW_OCOTP_CTRL[BUSY]. In addition HW_OCOTP_CTRL[ERROR] is set. It must be cleared by software before

any new access issued. If only some fuse words in the fuse bank are read protected or locked, controller read unprotected fuse words and assert HW_OCOTP_CTRL[BUSY]. Once completed, the controller clears HW_OCOTP_CTRL[BUSY].

5. Read HW_OCOTP_READ_FUSE_DATAx (x = 0, 1, 2, 3) registers to acquire the fuse word value. HW_OCOTP_READ_FUSE_DATAx (x = 0, 1, 2, 3) is 0xBADABADA if read fuse lock bit is programmed.

6.4.3.1.3 Fuse and Shadow Register Writes

Shadow register bits can be overridden by software until the corresponding fuse lock bit for the region is set. When the lock shadow bit is set, the shadow registers for that lock region become write locked. The LOCK shadow register also has no shadow or fuse lock bits but it is always read only.

In order to avoid "rogue" code performing erroneous writes to OTP, a special unlocking sequence is required for writes to the fuse banks. To program fuse bank correctly complete the following steps:

1. Program HW_OCOTP_TIMING[STROBE_PROG] and HW_OCOTP_TIMING[FSOURCE fields with timing values to match the current frequency of the ipg_clk. OTP writes will work at maximum bus frequencies as long as the HW_OCOTP_TIMING parameters are set correctly.
2. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write or reload must be completed before a write access can be requested.
3. Write the requested fuse bank number to HW_OCOTP_CTRL[ADDR] and program the unlock code into HW_OCOTP_CTRL[WR_UNLOCK]. This must be programmed for each write access. The lock code is documented in the register description. Both the unlock code and address can be written in the same operation.
4. Write the program data to the HW_OCOTP_DATAx (x = 0, 1, 2, 3) register. The HW_OCOTP_DATA0 must be written at last. After HW_OCOTP_DATA0 is written, OCOTP automatically set HW_OCOTP_CTRL[BUSY] and clears HW_OCOTP_CTRL[WR_UNLOCK]. Bit fields with 1's will result in that OTP bit being programmed. Bit fields with 0's will be ignored. At the same time that the write is accepted, the controller makes an internal copy of HW_OCOTP_CTRL[ADDR] which cannot be updated until the next write sequence is initiated. This copy guarantees that erroneous writes to HW_OCOTP_CTRL[ADDR] will not affect an active write operation.
5. If all the 4 fuse words in the fuse bank are program protected or locked, the write request will result in no OTP access and no setting of HW_OCOTP_CTRL[BUSY]. In addition HW_OCOTP_CTRL[ERROR] will be set. It must be cleared by software

before any new write access issued. If only part of fuse words in the fuse bank are program protected or locked, there are two kinds of cases.

a. The program data for unprotected fuse words are all 0.

   In this case, the write request will result in no OTP access and no setting of HW_OCOTP_CTRL[BUSY]. In addition HW_OCOTP_CTRL[ERROR] will be set. It must be cleared by software before any new write access issued. It is for protecting OTP and reducing unnecessary program operation for OTP.

b. The program data for unprotected fuse words are not 0.

   OCOTP will only program unprotected fuse words and assert HW_OCOTP_CTRL[BUSY] . Once completed, the controller clears HW_OCOTP_CTRL[BUSY].

It should be noted that write latencies to OTP are numbers of 10 micro-seconds. Write latencies is based on amount of bit filed which is 1. For example : program half fuse bits in one word need 10 us x 16.

NOTE

The program times for one fuse bank are limited. It is better to program all the 4 fuse words in one fuse bank at one program operation.

For further details of OTP read/write operations see [eFUSE].

HW_OCOTP_CTRL[ERROR] will be set under the following conditions:

- A write is performed to a shadow register during a shadow reload (essentially, while HW_OCOTP_CTRL[RELOAD_SHADOWS] is set. In addition, the contents of the shadow register shall not be updated.
- A write is performed to a shadow register which has been locked.
- A read is performed to from a shadow register which has been read locked.
- A program is performed to a fuse bank which all the fuse words has been program locked or all the program data for program unlocked region are all 0.
- A read is performed to from a fuse bank which all the fuse words has been read locked.

6.4.3.1.4 Write Postamble

Due to internal electrical characteristics of the OTP during writes, all OTP operations following a write must be separated by 2 us after the clearing of HW_OCOTP_CTRL_BUSY following the write. This guarantees programming voltages on-chip to reach a steady state when exiting a write sequence. This includes reads, shadow reloads, or other writes.

A recommended software sequence to meet the postamble requirements is as follows:

- Issue the write and poll for BUSY (as per Fuse Shadow Memory Footprint).
- Once BUSY is clear, use HW_DIGCTL_MICROSECONDS to wait 2 us.
- Perform the next OTP operation.

6.4.3.2 Fuse Shadow Memory Footprint

The OTP memory footprint shows in the following figure. The registers are grouped by lock region. Their names correspond to the PIO register and fusemap names.

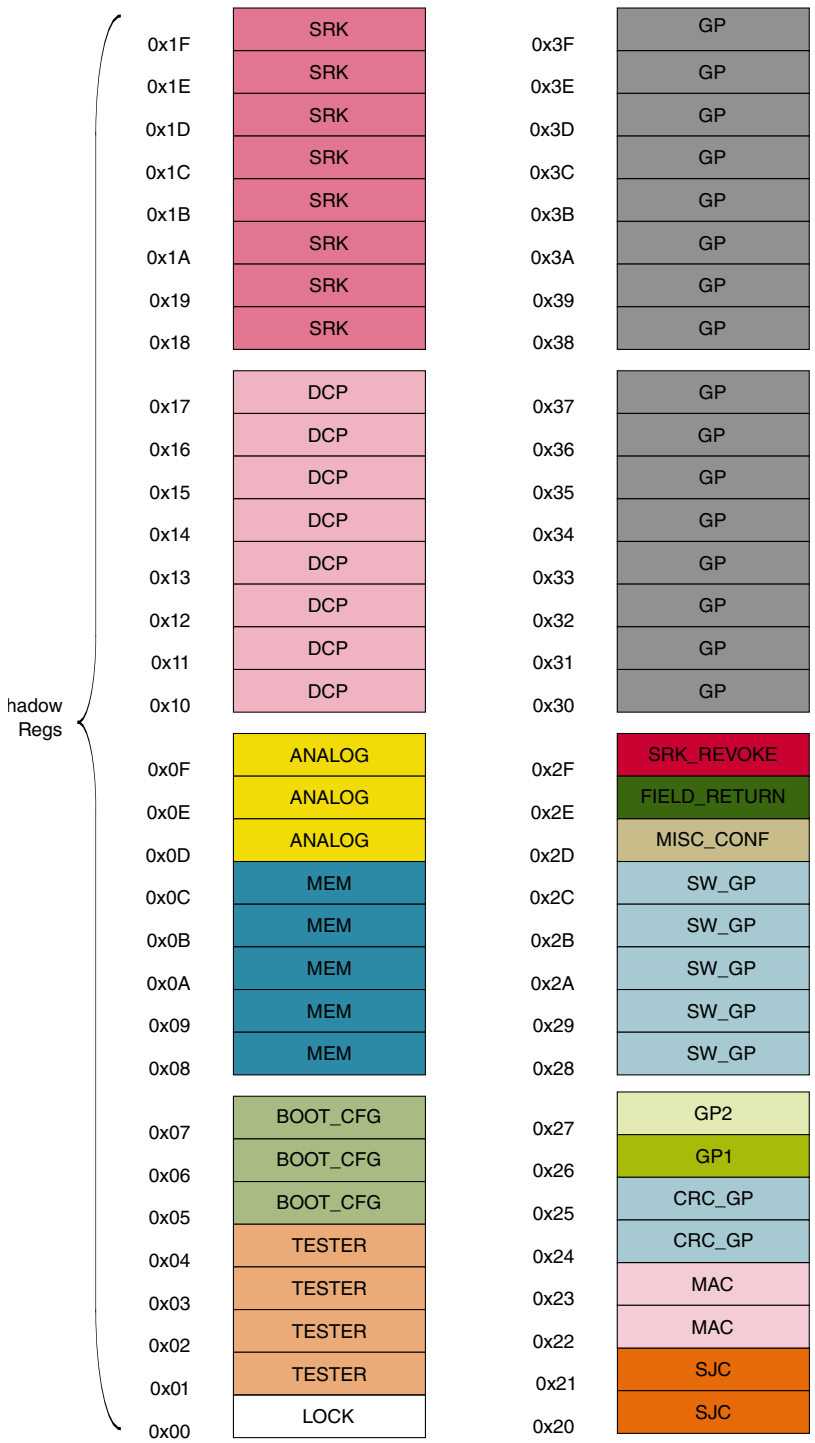


Figure 6-12. OTP Memory Footprint

6.4.3.3 OTP Read/Write Timing Parameters

There are two timing fields contained in the HW_OCOTP_TIMING register that specify counter limit values, which are used to specify the FSOURCE and PROG signal timing.

Both two timing parameters are specified in ipg_clk cycles. Since the ipg_clk frequency can be set to a range of values, these parameters must be adjusted with the clock to yield the appropriate delay.

The HW_OCOTP_TIMING[PROG] field specifies the period of the RPOG signal for fuse writes and given in units of ipg_clk cycles. This value is specified, so that the requirement for the time when the PROG signal is asserted high is met: 9900 ns < tPRW < 10100 ns. Even though a range is given for tPRM, it is recommended that to program [eFUSE] for a value of 10000 ns. Therefore, this field can be set according to the following equation:

tPGM = (HW_OCOTP_TIMING[PROG] - 1) / ipg_frequency = 10000 ns

The HW_OCOTP_TIMING[FSOURCE] field specifies the setup and hold time between FSOURCE signal and PROG signal for fuse writes and given in units of ipg_clk cycles. This field can be set according to the following equation:

tFSRCS = tFSRCH = (HW_OCOTP_TIMING[FSOURCE] - 1) / ipg_frequency > 1000 ns. .

The following figure illustrates the relationship between the PROG and FSOUCE signal in programming mode, as well as the timing PIO register fields that affect it.

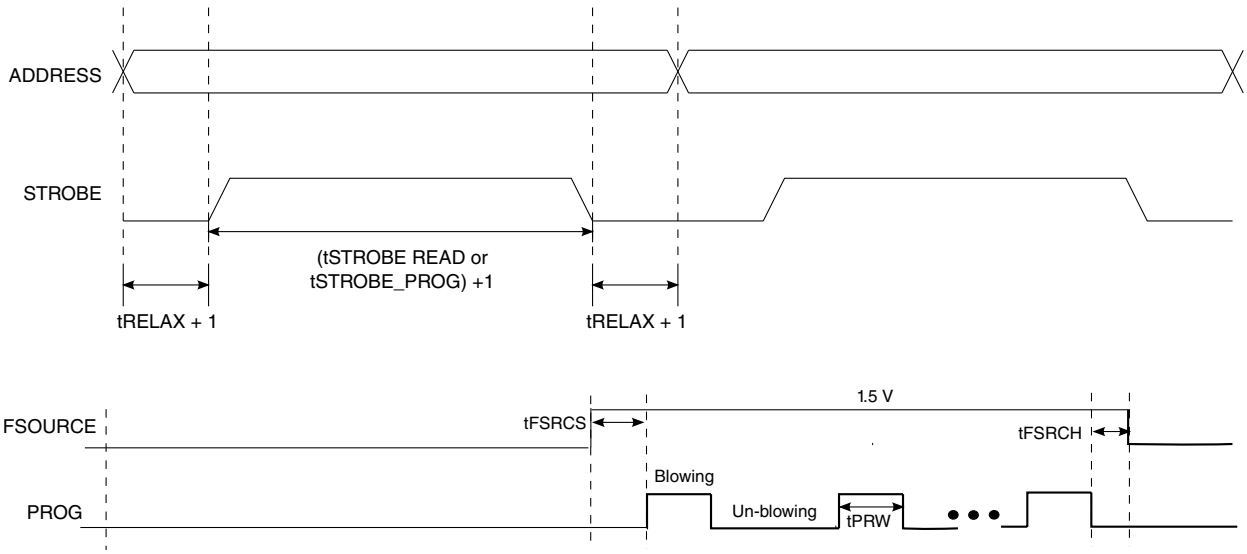


Figure 6-13. FSOURCE and PROG Signal Timing

6.4.3.4 Hardware Visible Fuses

The hwv_fuse bus emanates from the OCOTP block and goes to various other blocks inside the chip. This bus is made up of all the shadow register bits for all the 16 fuse banks.

Only a subset of these fuse bits are currently used by the hardware. The fuse bits are initially copied from the fuse banks after reset is deasserted. When all fuse bits are loaded into their shadow registers, the OCOTP asserts the fuse_latched output signal.

The hwv_reg bus also comes from the OCOTP. Its source is the HW_OCOTP_SCS register. This register has 1 defined bit, the HAB_JDE bit, that is connected to the SJC block. The SCS bits are intended to be used as volatile fuse bits under software control. Additional bits will be defined as needed in future implementations.

The system-wide reset sequence must be coordinated by the system reset controller, so that the hwv_fuse and hwv_reg buses are stable and reflect the values of the fuses before they are used by the rest of the system.

6.4.3.5 Behavior During Reset

The OCOTP is always active. The shadow registers automatically load the appropriate OTP contents after reset is deasserted. During this load-time HW_OCOTP_CTRL_BUSY is set. The load time is similar to that of a "reload shadow" operation.

6.4.3.6 Secure JTAG control

The JTAG control fuses are used to allow or disallow JTAG access to secured resources.

Three JTAG security levels are envisioned, as shown in the table below.

Table 6-21. JTAG Security Level Control Bits

Security Mode	JTAG_SMODE	Description
No Debug	2'b11	The highest security level.
Secure JTAG	2'b01	Limit the JTAG access by using key based authentication mechanism.
JTAG Enable	2'b00	Low Security, all JTAG features are enabled.

6.4.4 Fuse Map

See the Fusemap chapter of this reference manual for more information.

6.4.5 OCOTP Memory Map/Register Definition

OCOTP Hardware Register Format Summary

OCOTP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3035_0000	OTP Controller Control Register (OCOTP_CTRL)	32	R/W	0000_0000h	6.4.5.1/ 1064
3035_0004	OTP Controller Control Register (OCOTP_CTRL_SET)	32	R/W	0000_0000h	6.4.5.1/ 1064
3035_0008	OTP Controller Control Register (OCOTP_CTRL_CLR)	32	R/W	0000_0000h	6.4.5.1/ 1064

Table continues on the next page...

OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3035_000C	OTP Controller Control Register (OCOTP_CTRL_TOG)	32	R/W	0000_0000h	6.4.5.1/ 1064
3035_0010	OTP Controller Timing Register (OCOTP_TIMING)	32	R/W	0146_1299h	6.4.5.2/ 1066
3035_0020	OTP Controller Write Data Register (OCOTP_DATA0)	32	R/W	0000_0000h	6.4.5.3/ 1067
3035_0030	OTP Controller Write Data Register (OCOTP_DATA1)	32	R/W	0000_0000h	6.4.5.4/ 1067
3035_0040	OTP Controller Write Data Register (OCOTP_DATA2)	32	R/W	0000_0000h	6.4.5.5/ 1068
3035_0050	OTP Controller Write Data Register (OCOTP_DATA3)	32	R/W	0000_0000h	6.4.5.6/ 1068
3035_0060	OTP Controller Write Data Register (OCOTP_READ_CTRL)	32	R/W	0000_0000h	6.4.5.7/ 1068
3035_0070	OTP Controller Read Data Register (OCOTP_READ_FUSE_DATA0)	32	R/W	0000_0000h	6.4.5.8/ 1069
3035_0080	OTP Controller Read Data Register (OCOTP_READ_FUSE_DATA1)	32	R/W	0000_0000h	6.4.5.9/ 1070
3035_0090	OTP Controller Read Data Register (OCOTP_READ_FUSE_DATA2)	32	R/W	0000_0000h	6.4.5.10/ 1070
3035_00A0	OTP Controller Read Data Register (OCOTP_READ_FUSE_DATA3)	32	R/W	0000_0000h	6.4.5.11/ 1071
3035_00B0	Sticky bit Register (OCOTP_SW_STICKY)	32	R/W	0000_0000h	6.4.5.12/ 1071
3035_00C0	Software Controllable Signals Register (OCOTP_SCS)	32	R/W	0000_0000h	6.4.5.13/ 1072
3035_00C4	Software Controllable Signals Register (OCOTP_SCS_SET)	32	R/W	0000_0000h	6.4.5.13/ 1072
3035_00C8	Software Controllable Signals Register (OCOTP_SCS_CLR)	32	R/W	0000_0000h	6.4.5.13/ 1072
3035_00CC	Software Controllable Signals Register (OCOTP_SCS_TOG)	32	R/W	0000_0000h	6.4.5.13/ 1072
3035_00D0	OTP Controller CRC test address (OCOTP_CRC_ADDR)	32	R/W	0000_0000h	6.4.5.14/ 1073
3035_00E0	OTP Controller CRC Value Register (OCOTP_CRC_VALUE)	32	R/W	0000_0000h	6.4.5.15/ 1074
3035_00F0	OTP Controller Version Register (OCOTP_VERSION)	32	R/W	0300_0000h	6.4.5.16/ 1075
3035_0400	Value of OTP Bank0 Word0 (Lock controls) (OCOTP_LOCK)	32	R/W	0000_0000h	6.4.5.17/ 1075
3035_0410	Value of OTP Bank0 Word1 (Tester Information) (OCOTP_TESTER0)	32	R/W	0000_0000h	6.4.5.18/ 1077
3035_0420	Value of OTP Bank0 Word2 (Tester Information) (OCOTP_TESTER1)	32	R/W	0000_0000h	6.4.5.19/ 1078

Table continues on the next page...

OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3035_0430	Value of OTP Bank0 Word3 (Tester Information) (OCOTP_TESTER2)	32	R/W	0000_0000h	6.4.5.20/ 1078
3035_0440	Value of OTP Bank1 Word0 (Tester Information) (OCOTP_TESTER3)	32	R/W	0000_0000h	6.4.5.21/ 1079
3035_0450	Value of OTP Bank1 Word1 (Tester Information) (OCOTP_TESTER4)	32	R/W	0000_0000h	6.4.5.22/ 1079
3035_0460	Value of OTP Bank1 Word2 (Tester Information) (OCOTP_TESTER5)	32	R/W	0000_0000h	6.4.5.23/ 1080
3035_0470	Value of OTP Bank1 Word3 (Boot Configuration Information) (OCOTP_BOOT_CFG0)	32	R/W	0000_0000h	6.4.5.24/ 1080
3035_0480	Value of OTP Bank2 Word0 (Boot Configuration Information) (OCOTP_BOOT_CFG1)	32	R/W	0000_0000h	6.4.5.25/ 1081
3035_0490	Value of OTP Bank2 Word1 (Boot Configuration Information) (OCOTP_BOOT_CFG2)	32	R/W	0000_0000h	6.4.5.26/ 1081
3035_04A0	Value of OTP Bank2 Word2 (Boot Configuration Information) (OCOTP_BOOT_CFG3)	32	R/W	0000_0000h	6.4.5.27/ 1082
3035_04B0	Value of OTP Bank2 Word3 (BOOT Configuration Information) (OCOTP_BOOT_CFG4)	32	R/W	0000_0000h	6.4.5.28/ 1082
3035_04C0	Value of OTP Bank3 Word0 (Memory Related Information) (OCOTP_MEM_TRIM0)	32	R/W	0000_0000h	6.4.5.29/ 1083
3035_04D0	Value of OTP Bank3 Word1 (Memory Related Information) (OCOTP_MEM_TRIM1)	32	R/W	0000_0000h	6.4.5.30/ 1083
3035_04E0	Value of OTP Bank3 Word2 (Analog Information) (OCOTP_ANA0)	32	R/W	0000_0000h	6.4.5.31/ 1084
3035_04F0	Value of OTP Bank3 Word3 (Analog Info.) (OCOTP_ANA1)	32	R/W	0000_0000h	6.4.5.32/ 1084
3035_0500	Shadow Register for OTP Bank4 Word0 (OTPMK Key) (OCOTP_OTPMK0)	32	R/W	0000_0000h	6.4.5.33/ 1085
3035_0510	Shadow Register for OTP Bank4 Word1 (OTPMK Key) (OCOTP_OTPMK1)	32	R/W	0000_0000h	6.4.5.34/ 1085
3035_0520	Shadow Register for OTP Bank4 Word2 (OTPMK Key) (OCOTP_OTPMK2)	32	R/W	0000_0000h	6.4.5.35/ 1086
3035_0530	Shadow Register for OTP Bank4 Word3 (OTPMK Key) (OCOTP_OTPMK3)	32	R/W	0000_0000h	6.4.5.36/ 1086
3035_0540	Shadow Register for OTP Bank5 Word0 (OTPMK Key) (OCOTP_OTPMK4)	32	R/W	0000_0000h	6.4.5.37/ 1087
3035_0550	Shadow Register for OTP Bank5 Word1 (OTPMK Key) (OCOTP_OTPMK5)	32	R/W	0000_0000h	6.4.5.38/ 1087
3035_0560	Shadow Register for OTP Bank5 Word2 (OTPMK Key) (OCOTP_OTPMK6)	32	R/W	0000_0000h	6.4.5.39/ 1088
3035_0570	Shadow Register for OTP Bank5 Word3 (OTPMK Key) (OCOTP_OTPMK7)	32	R/W	0000_0000h	6.4.5.40/ 1088
3035_0580	Shadow Register for OTP Bank6 Word0 (SRK Hash) (OCOTP_SRK0)	32	R/W	0000_0000h	6.4.5.41/ 1089

Table continues on the next page...

OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3035_0590	Shadow Register for OTP Bank6 Word1 (SRK Hash) (OCOTP_SRK1)	32	R/W	0000_0000h	6.4.5.42/ 1089
3035_05A0	Shadow Register for OTP Bank6 Word2 (SRK Hash) (OCOTP_SRK2)	32	R/W	0000_0000h	6.4.5.43/ 1090
3035_05B0	Shadow Register for OTP Bank6 Word3 (SRK Hash) (OCOTP_SRK3)	32	R/W	0000_0000h	6.4.5.44/ 1090
3035_05C0	Shadow Register for OTP Bank7 Word0 (SRK Hash) (OCOTP_SRK4)	32	R/W	0000_0000h	6.4.5.45/ 1091
3035_05D0	Shadow Register for OTP Bank7 Word1 (SRK Hash) (OCOTP_SRK5)	32	R/W	0000_0000h	6.4.5.46/ 1091
3035_05E0	Shadow Register for OTP Bank7 Word2 (SRK Hash) (OCOTP_SRK6)	32	R/W	0000_0000h	6.4.5.47/ 1092
3035_05F0	Shadow Register for OTP Bank7 Word3 (SRK Hash) (OCOTP_SRK7)	32	R/W	0000_0000h	6.4.5.48/ 1092
3035_0600	Value of OTP Bank8 Word0 (Secure JTAG Response Field) (OCOTP_SJC_RESP0)	32	R/W	0000_0000h	6.4.5.49/ 1093
3035_0610	Value of OTP Bank8 Word1 (Secure JTAG Response Field) (OCOTP_SJC_RESP1)	32	R/W	0000_0000h	6.4.5.50/ 1093
3035_0620	Value of OTP Bank8 Word2 (USB ID info) (OCOTP_USB_ID)	32	R/W	0000_0000h	6.4.5.51/ 1094
3035_0630	Value of OTP Bank8 Word3 (Field Return) (OCOTP_FIELD_RETURN)	32	R/W	0000_0000h	6.4.5.52/ 1094
3035_0640	Value of OTP Bank9 Word0 (MAC Address) (OCOTP_MAC_ADDR0)	32	R/W	0000_0000h	6.4.5.53/ 1095
3035_0650	Value of OTP Bank9 Word1 (MAC Address) (OCOTP_MAC_ADDR1)	32	R/W	0000_0000h	6.4.5.54/ 1095
3035_0660	Value of OTP Bank9 Word2 (MAC Address) (OCOTP_MAC_ADDR2)	32	R/W	0000_0000h	6.4.5.55/ 1096
3035_0670	Value of OTP Bank9 Word3 (SRK Revoke) (OCOTP_SRK_REVOKE)	32	R/W	0000_0000h	6.4.5.56/ 1096
3035_0680	Shadow Register for OTP Bank10 Word0 (MAU Key) (OCOTP_MAU_KEY0)	32	R/W	0000_0000h	6.4.5.57/ 1097
3035_0690	Shadow Register for OTP Bank10 Word1 (MAU Key) (OCOTP_MAU_KEY1)	32	R/W	0000_0000h	6.4.5.58/ 1097
3035_06A0	Shadow Register for OTP Bank10 Word2 (MAU Key) (OCOTP_MAU_KEY2)	32	R/W	0000_0000h	6.4.5.59/ 1098
3035_06B0	Shadow Register for OTP Bank10 Word3 (MAU Key) (OCOTP_MAU_KEY3)	32	R/W	0000_0000h	6.4.5.60/ 1098
3035_06C0	Shadow Register for OTP Bank11 Word0 (MAU Key) (OCOTP_MAU_KEY4)	32	R/W	0000_0000h	6.4.5.61/ 1099
3035_06D0	Shadow Register for OTP Bank11 Word1 (MAU Key) (OCOTP_MAU_KEY5)	32	R/W	0000_0000h	6.4.5.62/ 1099
3035_06E0	Shadow Register for OTP Bank11 Word2 (MAU Key) (OCOTP_MAU_KEY6)	32	R/W	0000_0000h	6.4.5.63/ 1100

Table continues on the next page...

OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3035_06F0	Shadow Register for OTP Bank11 Word3 (MAU Key) (OCOTP_MAU_KEY7)	32	R/W	0000_0000h	6.4.5.64/ 1100
3035_0780	Value of OTP Bank14 Word0 (OCOTP_GP10)	32	R/W	0000_0000h	6.4.5.65/ 1101
3035_0790	Value of OTP Bank14 Word1 (OCOTP_GP11)	32	R/W	0000_0000h	6.4.5.66/ 1101
3035_07A0	Value of OTP Bank14 Word2 (OCOTP_GP20)	32	R/W	0000_0000h	6.4.5.67/ 1101
3035_07B0	Value of OTP Bank14 Word3 (OCOTP_GP21)	32	R/W	0000_0000h	6.4.5.68/ 1102
3035_07C0	Value of OTP Bank15 Word0 (CRC Key) (OCOTP_CRC_GP10)	32	R/W	0000_0000h	6.4.5.69/ 1102
3035_07D0	Value of OTP Bank15 Word1 (CRC Key) (OCOTP_CRC_GP11)	32	R/W	0000_0000h	6.4.5.70/ 1103
3035_07E0	Value of OTP Bank15 Word2 (CRC Key) (OCOTP_CRC_GP20)	32	R/W	0000_0000h	6.4.5.71/ 1103
3035_07F0	Value of OTP Bank15 Word3 (CRC Key) (OCOTP_CRC_GP21)	32	R/W	0000_0000h	6.4.5.72/ 1104

6.4.5.1 OTP Controller Control Register (OCOTP_CTRL*n*)

The OCOTP Control and Status Register specifies the copy state, as well as the control required for random access of the OTP memory

OCOTP_CTRL: 0x000

OCOTP_CTRL_SET: 0x004

OCOTP_CTRL_CLR: 0x008

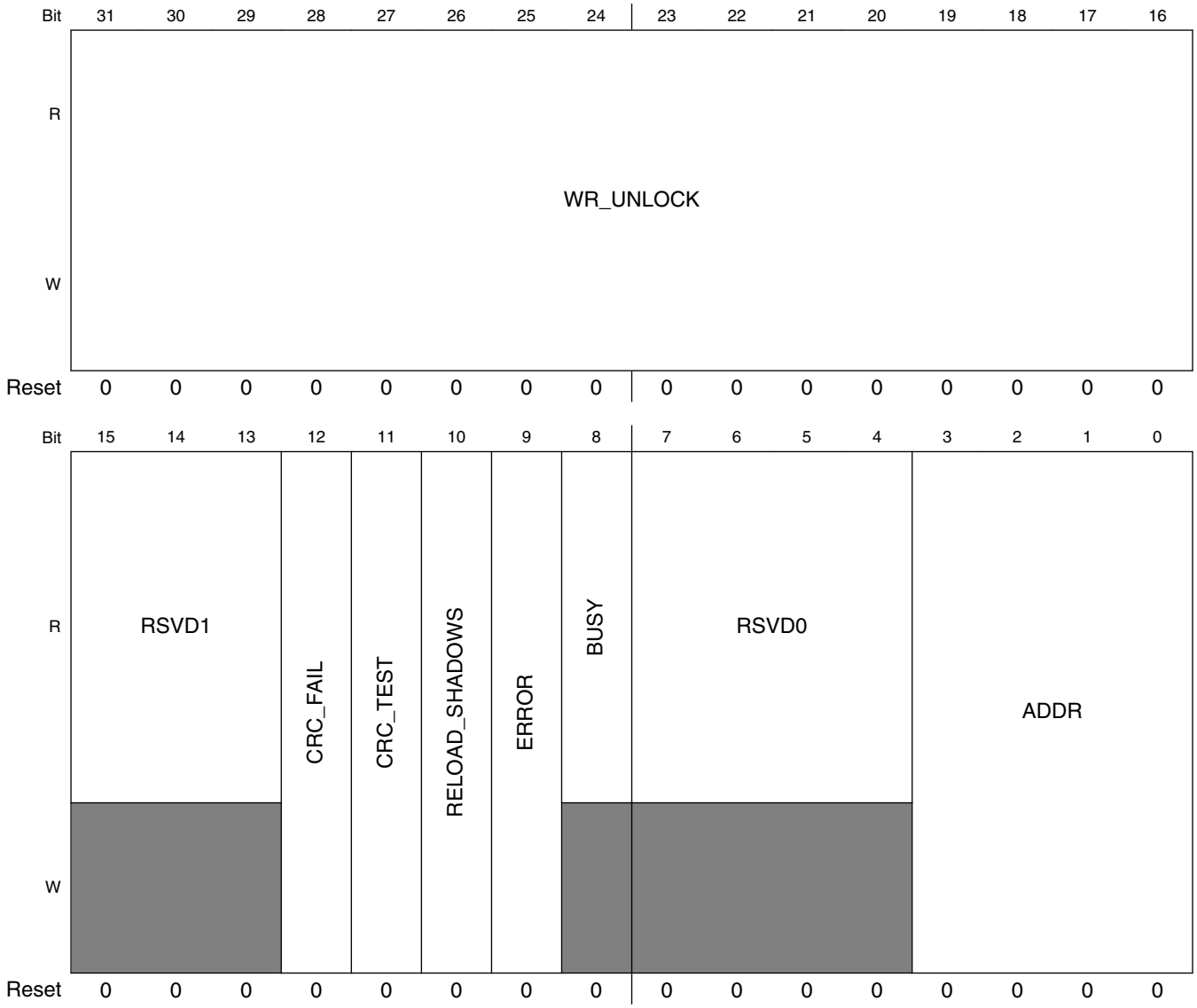
OCOTP_CTRL_TOG: 0x00C

The OCOTP Control and Status Register provides the necessary software interface for performing read and write operations to the On-Chip OTP (One-Time Programmable ROM). The control fields such as WR_UNLOCK, ADDR, and BUSY / ERROR may be used in conjunction with the HW_OCOTP_DATAn (n = 0, 1, 2, 3) register to perform write operations. Read operations to the On-Chip OTP are involving ADDR, BUSY / ERROR bit field and HW_OCOTP_READ_CTRL register. Read value is saved in HW_OCOTP_READ_FUSE_DATAn (n = 0, 1, 2, 3) register.

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 0h offset + (4d × i), where i=0d to 3d



OCOTP_CTRL*n* field descriptions

Field	Description
31–16 WR_UNLOCK	Write 0x3E77 to enable OTP write accesses. **NOTE:** This register must be unlocked on a write-by-write basis (a write is initiated when HW_OCOTP_DATA is written), so the UNLOCK bit must contain the correct key value during all writes to HW_OCOTP_DATA, otherwise a write is not initiated. This field is automatically cleared after a successful write completion (clearing of BUSY). 0x3E77 **KEY** — Key needed to unlock HW_OCOTP_DATA register.
15–13 RSVD1	This field is reserved. This read-only field is reserved and always has the value 0.
12 CRC_FAIL	Set by controller when calculated CRC value is not equal to appointed CRC fuse word.

Table continues on the next page...

OCOTP_CTRL*n* field descriptions (continued)

Field	Description
11 CRC_TEST	Set to calculate CRC32 according to start address and end address in CRC_ADDR register. Then compare with CRC fuse word according CRC address in CRC_ADDR register to generate CRC_FAIL flag. This operation automatically set Busy. Once CRC test is finished, BUSY and CRC_TEST are automatically cleared by the controller.
10 RELOAD_ SHADOWS	Set to force re-loading all the shadow registers (HW / SW capability and LOCK). This operation automatically set BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller.
9 ERROR	Set by the controller when an access to a locked region (OTP or shadow register) is requested. Must be cleared before any further access performed. This bit can only be set by the controller. This bit is also set if the Pin interface is active and software requests an access to the OTP. For instance, the ERROR bit cannot be cleared until the Pin interface access has completed. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
8 BUSY	OTP controller status bit. When active, no new write access or read access to OTP(including RELOAD_SHADOWS) can be performed. Cleared by controller when access complete. After reset (or after setting RELOAD_SHADOWS), this bit is set by the controller until the HW/SW and LOCK registers are successfully copied, after which time it is automatically cleared by the controller.
7–4 RSVD0	This field is reserved. This read-only field is reserved and always has the value 0.
ADDR	OTP write and read access address register. Specifies one of 16 128 bits OTP locations (bank 0x0 ~ 0xF). If a valid access is accepted by the controller, the controller makes an internal copy of this value. This internal copy will not update until the access is complete.

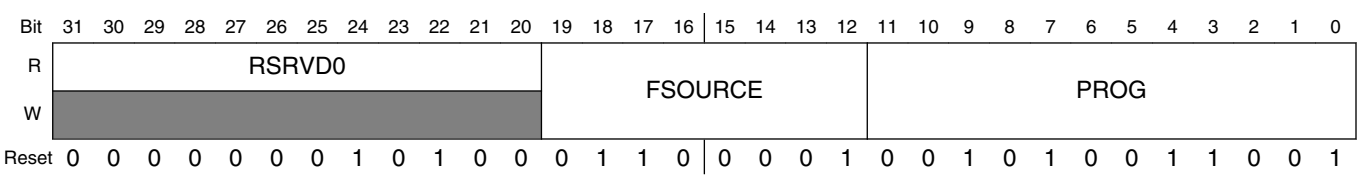
6.4.5.2 OTP Controller Timing Register (OCOTP_TIMING)

This register specifies timing parameters of programming and reading the OCOTP fuse array.

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 10h offset = 3035_0010h



OCOTP_TIMING field descriptions

Field	Description
31–20 RSRVD0	This field is reserved. This read-only field is reserved and always has the value 0.
19–12 FSOURCE	This count value specifies FSOURCE to PROG setup / hold time. tFSRCS = tFSRCH = (FSOURCE + 1) / IPG_CLK_FREQ. It is given in number of IPG_CLK periods.

Table continues on the next page...

OCOTP_TIMING field descriptions (continued)

Field	Description
PROG	This count value specifies the strobe period in one time write OTP. tPRW = (PROG - 1) / IPG_CLK_FREQ. It is given in number of IPG_CLK periods.

6.4.5.3 OTP Controller Write Data Register (OCOTP_DATA0)

This register associates with HW_OCOTP_CTRL to perform one-time writes to the OTP. Please see the "Software Write Sequence" section for operating details.

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 20h offset = 3035_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_DATA0 field descriptions

Field	Description
DATA0	The program data for first fuse word in one 128 bits OTP. It is used to initiate a write to OTP. Please see the "Software Write Sequence" section for operating details.

6.4.5.4 OTP Controller Write Data Register (OCOTP_DATA1)

This register associates with HW_OCOTP_CTRL to perform one-time writes to the OTP. Please see the "Software Write Sequence" section for operating details.

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 30h offset = 3035_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_DATA1 field descriptions

Field	Description
DATA1	The program data for second fuse word in one 128 bits OTP.

6.4.5.5 OTP Controller Write Data Register (OCOTP_DATA2)

This register associates with HW_OCOTP_CTRL to perform one-time writes to the OTP. Please see the "Software Write Sequence" section for operating details.

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 40h offset = 3035_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_DATA2 field descriptions

Field	Description
DATA2	The program data for third fuse word in one 128 bits OTP.

6.4.5.6 OTP Controller Write Data Register (OCOTP_DATA3)

This register associates with HW_OCOTP_CTRL to perform one-time writes to the OTP. Please see the "Software Write Sequence" section for operating details.

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 50h offset = 3035_0050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_DATA3 field descriptions

Field	Description
DATA3	The program data for the last fuse word in one 128 bits OTP.

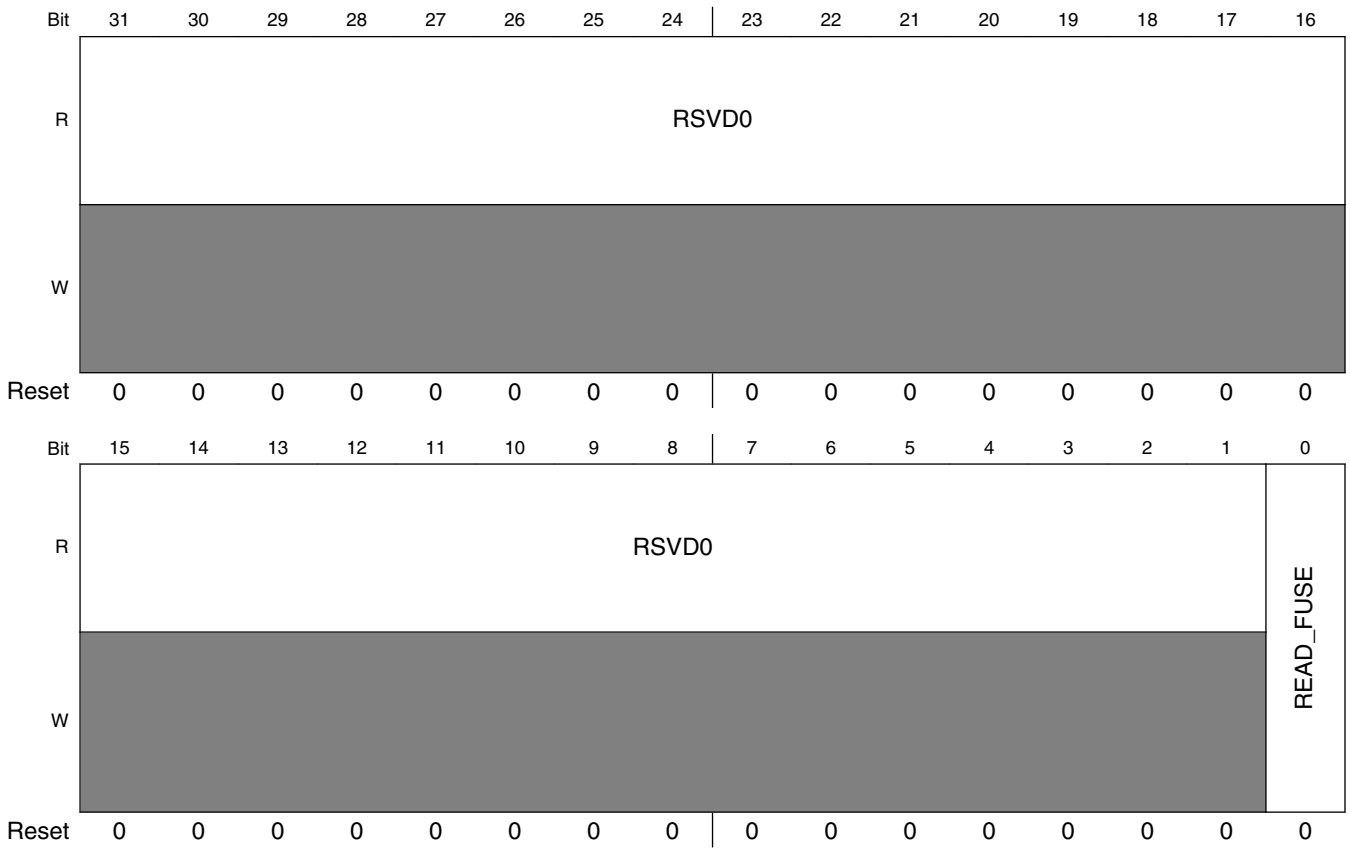
6.4.5.7 OTP Controller Write Data Register (OCOTP_READ_CTRL)

This register associates with HW_OCOTP_CTRL to perform one time read to the OTP. Please see the "Software read Sequence" section for operating details.

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 60h offset = 3035_0060h



OCOTP_READ_CTRL field descriptions

Field	Description
31–1 RSVD0	This field is reserved. This read-only field is reserved and always has the value 0.
0 READ_FUSE	Used to initiate a read to OTP. Please see the "Software read Sequence" section for operating details.

6.4.5.8 OTP Controller Read Data Register (OCOTP_READ_FUSE_DATA0)

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 70h offset = 3035_0070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_READ_FUSE_DATA0 field descriptions

Field	Description
DATA0	The first fuse word data read from one OTP.

6.4.5.9 OTP Controller Read Data Register (OCOTP_READ_FUSE_DATA1)

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 80h offset = 3035_0080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_READ_FUSE_DATA1 field descriptions

Field	Description
DATA1	The second fuse word data read from one OTP.

6.4.5.10 OTP Controller Read Data Register (OCOTP_READ_FUSE_DATA2)

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 90h offset = 3035_0090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_READ_FUSE_DATA2 field descriptions

Field	Description
DATA2	The third fuse word data read from one OTP.

6.4.5.11 OTP Controller Read Data Register (OCOTP_READ_FUSE_DATA3)

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + A0h offset = 3035_00A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_READ_FUSE_DATA3 field descriptions

Field	Description
DATA3	The last fuse word data read from one OTP.

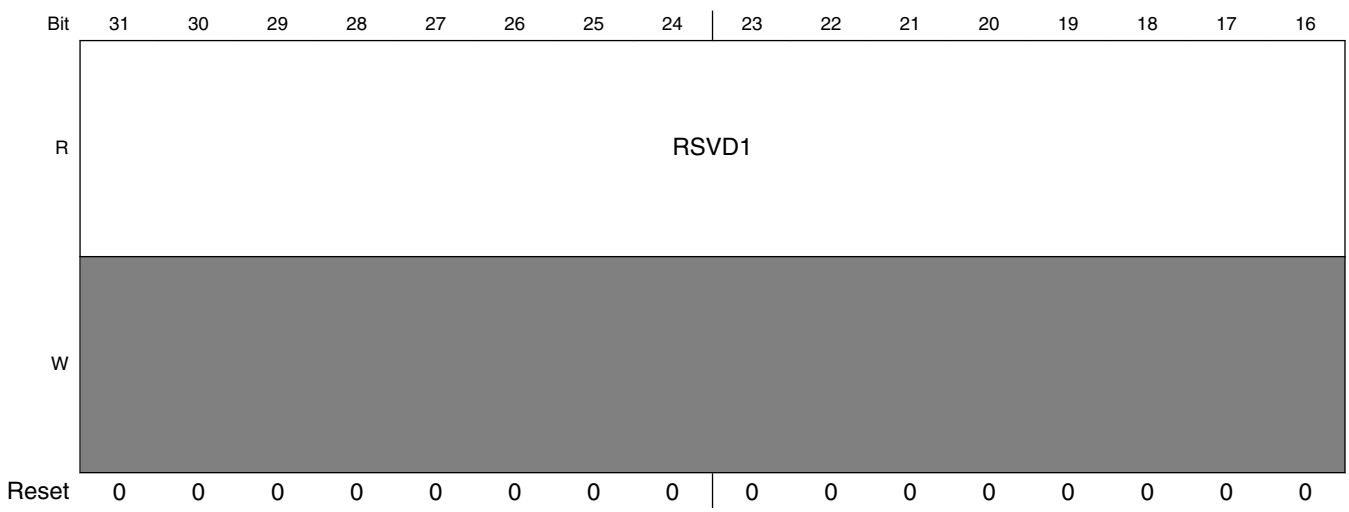
6.4.5.12 Sticky bit Register (OCOTP_SW_STICKY)

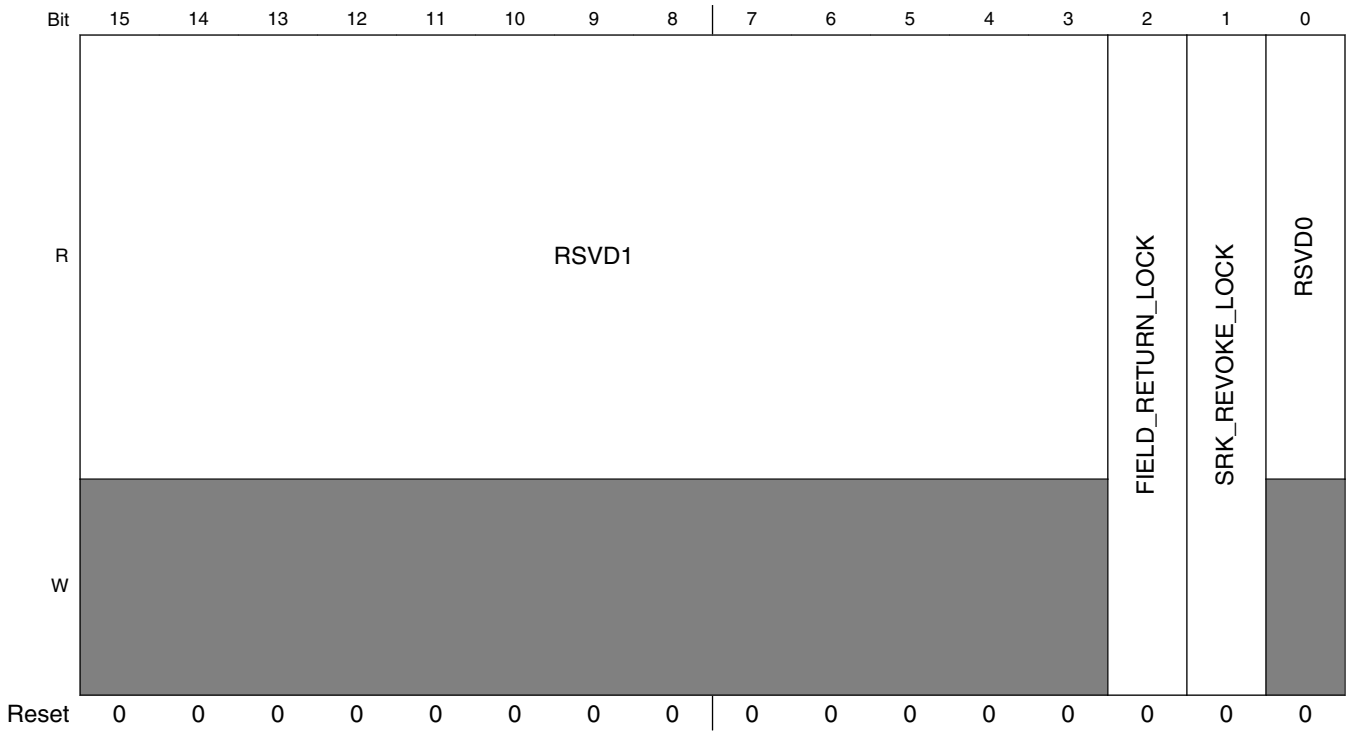
Some sticky bits are used by software to lock some fuse area, shadow registers, and other features.

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + B0h offset = 3035_00B0h





OCOTP_SW_STICKY field descriptions

Field	Description
31–3 RSVD1	This field is reserved. This read-only field is reserved and always has the value 0.
2 FIELD_ RETURN_LOCK	Shadow register write and OTP write lock for FIELD_RETURN region. When set, the writing of this region's shadow register and OTP fuse word are blocked. Once this bit is set, it is always high unless a POR is issued.
1 SRK_REVOKE_ LOCK	Shadow register write and OTP write lock for SRK_REVOKE region. When set, the writing of this region's shadow register and OTP fuse word are blocked. Once this bit is set, it is always high unless a POR is issued.
0 RSVD0	This field is reserved. This read-only field is reserved and always has the value 0.

6.4.5.13 Software Controllable Signals Register (OCOTP_SCS*n*)

OCOTP_SCS: 0x0C0

OCOTP_SCS_SET: 0x0C4

OCOTP_SCS_CLR: 0x0C8

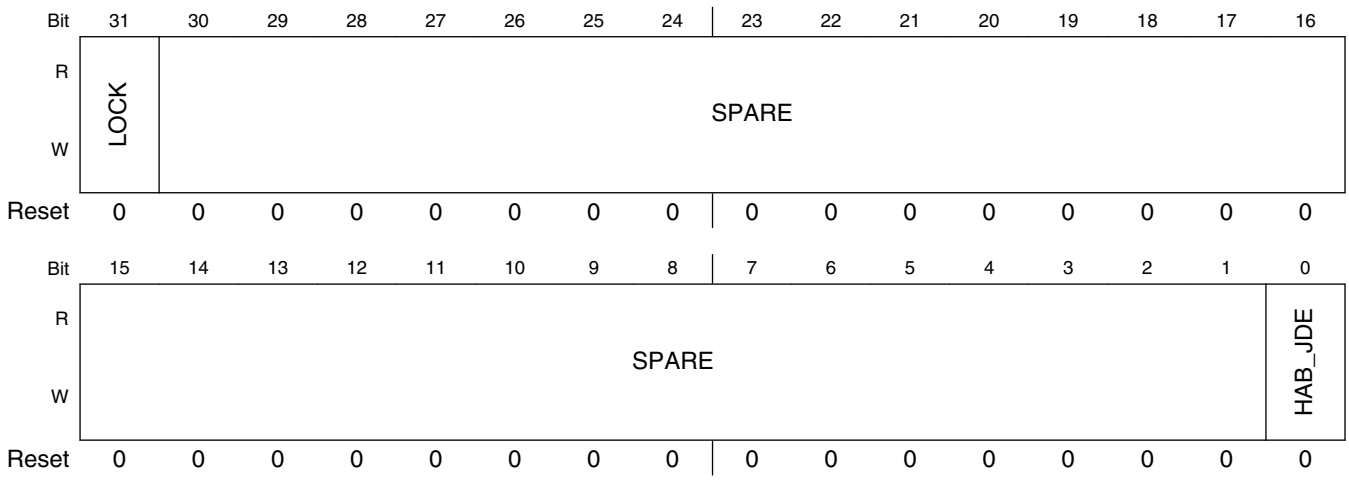
OCOTP_SCS_TOG: 0x0CC

This register holds volatile configuration values that can be set and locked by verified software. All values are back to their default values after POR.

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + C0h offset + (4d × i), where i=0d to 3d



OCOTP_SCS*n* field descriptions

Field	Description
31 LOCK	When set, all of the bits in this register are locked and cannot be changed during SW programming. This bit is only reset after a POR is issued.
30–1 SPARE	Unallocated read / write bits of implementation for specific software use.
0 HAB_JDE	HAB JTAG Debug Enable. This bit is used by the HAB to enable JTAG debugging, assuming that a properly signed command to do it, so it can be found and validated by the HAB. The HAB must lock the register before passing control to the OS depend on whether JTAG debugging has been enabled. Once JTAG is enabled by this bit, it cannot be disabled unless the system is reset by POR. 0 JTAG debugging is not enabled by the HAB (it can be enabled by other mechanisms). 1 JTAG debugging is enabled by the HAB (though this signal may be gated off).

6.4.5.14 OTP Controller CRC test address (OCOTP_CRC_ADDR)

The OCOTP Data Register is used for OTP Read

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + D0h offset = 3035_00D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0														CRC_ADDR	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA_END_ADDR								DATA_START_ADDR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_CRC_ADDR field descriptions

Field	Description
31–18 RSVD0	This field is reserved. This read-only field is reserved and always has the value 0.
17–16 CRC_ADDR	Address of 32-bit CRC result is for comparing.
15–8 DATA_END_ADDR	End address of fuse word location is for CRC calculation.
DATA_START_ADDR	Start address of fuse word location is for CRC calculation.

6.4.5.15 OTP Controller CRC Value Register (OCOTP_CRC_VALUE)

The CRC32 value is based on CRC_ADDR.

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + E0h offset = 3035_00E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_CRC_VALUE field descriptions

Field	Description
DATA	The CRC32 value is based on CRC_ADDR.

6.4.5.16 OTP Controller Version Register (OCOTP_VERSION)

This register always returns a known read value for debug purposes. It indicates the version of the block.

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + F0h offset = 3035_00F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only values reflect the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only values reflect the MINOR field of the RTL version.
STEP	Fixed read-only values reflect the stepping of the RTL version.

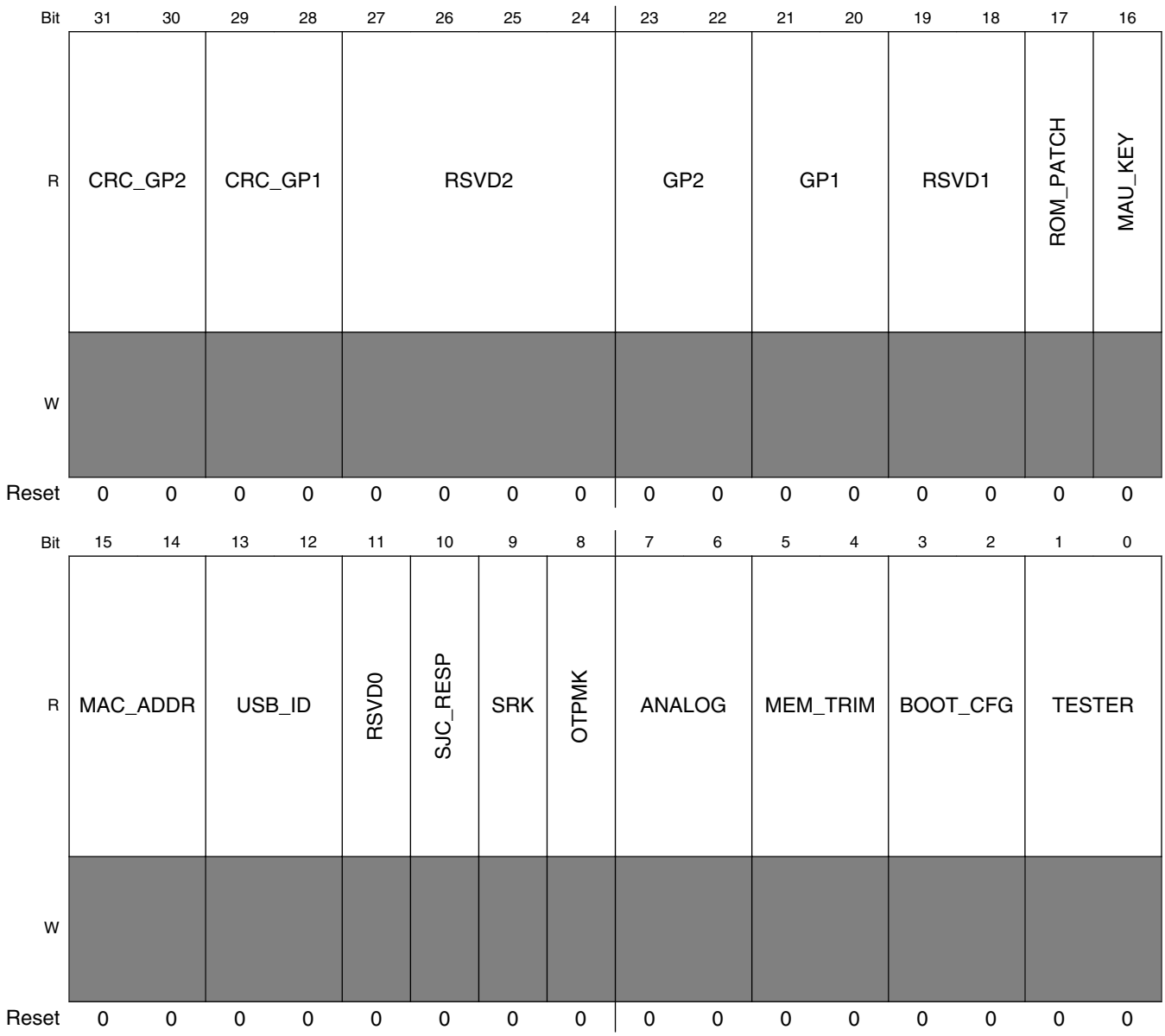
6.4.5.17 Value of OTP Bank0 Word0 (Lock controls) (OCOTP_LOCK)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 400h offset = 3035_0400h



OCOTP_LOCK field descriptions

Field	Description
31–30 CRC_GP2	Status of shadow register write and read, OTP write and read lock for CRC_GP2 region. When bit 1 is set, the reading and writing of this region's OTP fuse and reading of shadow register are blocked. When bit 0 is set, the writing of this region's shadow register and OTP fuse are blocked.
29–28 CRC_GP1	Status of shadow register write and read, OTP write and read lock for CRC_GP1 region. When bit 1 is set, the reading and writing of this region's OTP fuse and reading of shadow register are blocked. When bit 0 is set, the writing of this region's shadow register and OTP fuse are blocked.
27–24 RSVD2	This field is reserved. This read-only field is reserved and always has the value 0.
23–22 GP2	Status of shadow register and OTP write lock for GP2 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.

Table continues on the next page...

OCOTP_LOCK field descriptions (continued)

Field	Description
21–20 GP1	Status of shadow register and OTP write lock for GP1 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
19–18 RSVD1	This field is reserved. This read-only field is reserved and always has the value 0.
17 ROM_PATCH	Status of shadow register and OTP write lock for ROM_PATCH region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
16 MAU_KEY	Status of shadow register read and write, OTP read and write lock for MANUFACTURE_KEY region. When set, the writing of this region's shadow register and OTP fuse word are blocked. The read of this region's shadow register and OTP fuse word are also blocked.
15–14 MAC_ADDR	Status of shadow register and OTP write lock for MAC_ADDR region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
13–12 USB_ID	Status of shadow register and OTP write lock for USB_ID region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
11 RSVD0	This field is reserved. This read-only field is reserved and always has the value 0.
10 SJC_RESP	Status of shadow register read and write, OTP read and write lock for SJC_RESP region. When set, the writing of this region's shadow register and OTP fuse word are blocked. The read of this region's shadow register and OTP fuse word are also blocked.
9 SRK	Status of shadow register and OTP write lock for SRK region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
8 OTPMK	Status of shadow register read and write, OTP read and write lock for OTPMK region. When set, the writing of this region's shadow register and OTP fuse word are blocked. The read of this region's shadow register and OTP fuse word are also blocked.
7–6 ANALOG	Status of shadow register and OTP write lock for analog region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
5–4 MEM_TRIM	Status of shadow register and OTP write lock for MEM_TRIM region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
3–2 BOOT_CFG	Status of shadow register and OTP write lock for BOOT_CFG region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
TESTER	Status of shadow register and OTP write lock for tester region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.

6.4.5.18 Value of OTP Bank0 Word1 (Tester Information) (OCOTP_TESTER0)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 410h offset = 3035_0410h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_TESTER0 field descriptions

Field	Description
BITS	It reflects value of OTP Bank 0, word 1. These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

6.4.5.19 Value of OTP Bank0 Word2 (Tester Information) (OCOTP_TESTER1)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 420h offset = 3035_0420h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_TESTER1 field descriptions

Field	Description
BITS	It reflects value of OTP Bank 0, word 2. These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

6.4.5.20 Value of OTP Bank0 Word3 (Tester Information) (OCOTP_TESTER2)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 430h offset = 3035_0430h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_TESTER2 field descriptions

Field	Description
BITS	It reflects value of OTP Bank 0, word 3. These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

6.4.5.21 Value of OTP Bank1 Word0 (Tester Information) (OCOTP_TESTER3)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 440h offset = 3035_0440h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_TESTER3 field descriptions

Field	Description
BITS	It reflects value of OTP Bank 1, word 0. These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

6.4.5.22 Value of OTP Bank1 Word1 (Tester Information) (OCOTP_TESTER4)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 450h offset = 3035_0450h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_TESTER4 field descriptions

Field	Description
BITS	It reflects value of OTP Bank 1, word 1. These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

6.4.5.23 Value of OTP Bank1 Word2 (Tester Information) (OCOTP_TESTER5)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 460h offset = 3035_0460h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_TESTER5 field descriptions

Field	Description
BITS	It reflects value of OTP Bank 1, word 2. These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

6.4.5.24 Value of OTP Bank1 Word3 (Boot Configuration Information) (OCOTP_BOOT_CFG0)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 470h offset = 3035_0470h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_BOOT_CFG0 field descriptions

Field	Description
BITS	It reflects value of OTP Bank 1, word 3. These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

6.4.5.25 Value of OTP Bank2 Word0 (Boot Configuration Information) (OCOTP_BOOT_CFG1)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 480h offset = 3035_0480h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_BOOT_CFG1 field descriptions

Field	Description
BITS	It reflects value of OTP bank 2, word 0. These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

6.4.5.26 Value of OTP Bank2 Word1 (Boot Configuration Information) (OCOTP_BOOT_CFG2)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 490h offset = 3035_0490h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_BOOT_CFG2 field descriptions

Field	Description
BITS	It reflects value of OTP bank 2, word 1. These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

6.4.5.27 Value of OTP Bank2 Word2 (Boot Configuration Information) (OCOTP_BOOT_CFG3)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 4A0h offset = 3035_04A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_BOOT_CFG3 field descriptions

Field	Description
BITS	It reflects value of OTP bank 2, word 2. These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

6.4.5.28 Value of OTP Bank2 Word3 (BOOT Configuration Information) (OCOTP_BOOT_CFG4)

After reset, it copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 4B0h offset = 3035_04B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_BOOT_CFG4 field descriptions

Field	Description
BITS	It reflects value of OTP bank 2, word 3. These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

6.4.5.29 Value of OTP Bank3 Word0 (Memory Related Information) (OCOTP_MEM_TRIM0)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 4C0h offset = 3035_04C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MEM_TRIM0 field descriptions

Field	Description
BITS	It reflects value of OTP bank 3, word 0. These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

6.4.5.30 Value of OTP Bank3 Word1 (Memory Related Information) (OCOTP_MEM_TRIM1)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 4D0h offset = 3035_04D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MEM_TRIM1 field descriptions

Field	Description
BITS	It reflects value of OTP bank 3, word 1. These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

6.4.5.31 Value of OTP Bank3 Word2 (Analog Information) (OCOTP_ANA0)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 4E0h offset = 3035_04E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_ANA0 field descriptions

Field	Description
BITS	It reflects value of OTP bank 3, word 2. These bits become read-only after the HW_OCOTP_LOCK_ANALOG[1] bit is set.

6.4.5.32 Value of OTP Bank3 Word3 (Analog Info.) (OCOTP_ANA1)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 4F0h offset = 3035_04F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_ANA1 field descriptions

Field	Description
BITS	It reflects value of OTP bank 3, word 3. These bits become read-only after the HW_OCOTP_LOCK_ANALOG[1] bit is set.

6.4.5.33 Shadow Register for OTP Bank4 Word0 (OTPMK Key) (OCOTP_OTPMK0)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 500h offset = 3035_0500h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_OTPMK0 field descriptions

Field	Description
BITS	Shadow register for the OTPMK Key word0 (Copy of OTP Bank 4, word 0). These bits cannot be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.34 Shadow Register for OTP Bank4 Word1 (OTPMK Key) (OCOTP_OTPMK1)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 510h offset = 3035_0510h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_OTPMK1 field descriptions

Field	Description
BITS	Shadow register for the OTPMK Key word1 (Copy of OTP Bank 4, word 1). These bits cannot be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.35 Shadow Register for OTP Bank4 Word2 (OTPMK Key) (OCOTP_OTPMK2)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 520h offset = 3035_0520h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_OTPMK2 field descriptions

Field	Description
BITS	Shadow register for the OTPMK Key word2 (Copy of OTP Bank 4, word 2). These bits cannot be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.36 Shadow Register for OTP Bank4 Word3 (OTPMK Key) (OCOTP_OTPMK3)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 530h offset = 3035_0530h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_OTPMK3 field descriptions

Field	Description
BITS	Shadow register for the OTPMK Key word3 (Copy of OTP Bank 4, word 3). These bits cannot be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.37 Shadow Register for OTP Bank5 Word0 (OTPMK Key) (OCOTP_OTPMK4)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 540h offset = 3035_0540h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_OTPMK4 field descriptions

Field	Description
BITS	Shadow register for the OTPMK Key word4 (Copy of OTP Bank 5, word 0). These bits cannot be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.38 Shadow Register for OTP Bank5 Word1 (OTPMK Key) (OCOTP_OTPMK5)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 550h offset = 3035_0550h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_OTPMK5 field descriptions

Field	Description
BITS	Shadow register for the OTPMK Key word5 (Copy of OTP Bank 5, word 1). These bits cannot be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.39 Shadow Register for OTP Bank5 Word2 (OTPMK Key) (OCOTP_OTPMK6)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 560h offset = 3035_0560h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_OTPMK6 field descriptions

Field	Description
BITS	Shadow register for the OTPMK Key word6 (Copy of OTP Bank 5, word 2). These bits cannot be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.40 Shadow Register for OTP Bank5 Word3 (OTPMK Key) (OCOTP_OTPMK7)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 570h offset = 3035_0570h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_OTPMK7 field descriptions

Field	Description
BITS	Shadow register for the OTPMK Key word7 (Copy of OTP Bank 5, word 3). These bits cannot be read and written after the HW_OCOTP_LOCK_OTPMK bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.41 Shadow Register for OTP Bank6 Word0 (SRK Hash) (OCOTP_SRK0)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 580h offset = 3035_0580h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK0 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word0 (Copy of OTP Bank 6, word 0). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

6.4.5.42 Shadow Register for OTP Bank6 Word1 (SRK Hash) (OCOTP_SRK1)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 590h offset = 3035_0590h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK1 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word1 (Copy of OTP Bank 6, word 1). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

6.4.5.43 Shadow Register for OTP Bank6 Word2 (SRK Hash) (OCOTP_SRK2)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 5A0h offset = 3035_05A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK2 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word2 (Copy of OTP Bank 6, word 2). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

6.4.5.44 Shadow Register for OTP Bank6 Word3 (SRK Hash) (OCOTP_SRK3)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 5B0h offset = 3035_05B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK3 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word3 (Copy of OTP Bank 6, word 3). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

6.4.5.45 Shadow Register for OTP Bank7 Word0 (SRK Hash) (OCOTP_SRK4)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 5C0h offset = 3035_05C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK4 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word4 (Copy of OTP Bank 7, word 0). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

6.4.5.46 Shadow Register for OTP Bank7 Word1 (SRK Hash) (OCOTP_SRK5)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 5D0h offset = 3035_05D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK5 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word5 (Copy of OTP Bank 7, word 1). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

6.4.5.47 Shadow Register for OTP Bank7 Word2 (SRK Hash) (OCOTP_SRK6)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 5E0h offset = 3035_05E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK6 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word6 (Copy of OTP Bank 7, word 2). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

6.4.5.48 Shadow Register for OTP Bank7 Word3 (SRK Hash) (OCOTP_SRK7)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 5F0h offset = 3035_05F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK7 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word7 (Copy of OTP Bank 7, word 3). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

6.4.5.49 Value of OTP Bank8 Word0 (Secure JTAG Response Field) (OCOTP_SJC_RESP0)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 600h offset = 3035_0600h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SJC_RESP0 field descriptions

Field	Description
BITS	Shadow register for the SJC_RESP Key word0 (Copy of OTP Bank 8, word 0). These bits cannot be read and written after the HW_OCOTP_LOCK_SJC_RESP bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.50 Value of OTP Bank8 Word1 (Secure JTAG Response Field) (OCOTP_SJC_RESP1)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 610h offset = 3035_0610h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SJC_RESP1 field descriptions

Field	Description
BITS	Shadow register for the SJC_RESP Key word1 (Copy of OTP Bank 8, word 1). These bits cannot be read and written after the HW_OCOTP_LOCK_SJC_RESP bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.51 Value of OTP Bank8 Word2 (USB ID info) (OCOTP_USB_ID)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 620h offset = 3035_0620h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_USB_ID field descriptions

Field	Description
BITS	It reflects the value of OTP Bank 8, word 2.

6.4.5.52 Value of OTP Bank8 Word3 (Field Return) (OCOTP_FIELD_RETURN)

After reset, it automatically copied from the OTP. It cannot be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 630h offset = 3035_0630h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_FIELD_RETURN field descriptions

Field	Description
BITS	It reflects the value of OTP Bank 8, word 3.

6.4.5.53 Value of OTP Bank9 Word0 (MAC Address) (OCOTP_MAC_ADDR0)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 640h offset = 3035_0640h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MAC_ADDR0 field descriptions

Field	Description
BITS	It reflects the value of OTP Bank 9, word 0.

6.4.5.54 Value of OTP Bank9 Word1 (MAC Address) (OCOTP_MAC_ADDR1)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 650h offset = 3035_0650h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MAC_ADDR1 field descriptions

Field	Description
BITS	It reflects the value of OTP Bank 9, word 1.

6.4.5.55 Value of OTP Bank9 Word2 (MAC Address) (OCOTP_MAC_ADDR2)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 660h offset = 3035_0660h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MAC_ADDR2 field descriptions

Field	Description
BITS	It reflects the value of OTP Bank 9, word 2.

6.4.5.56 Value of OTP Bank9 Word3 (SRK Revoke) (OCOTP_SRK_REVOKE)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 670h offset = 3035_0670h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK_REVOKE field descriptions

Field	Description
BITS	It reflects the value of OTP Bank 9, word 3.

6.4.5.57 Shadow Register for OTP Bank10 Word0 (MAU Key) (OCOTP_MAU_KEY0)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 680h offset = 3035_0680h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MAU_KEY0 field descriptions

Field	Description
BITS	Shadow register for the MAU Key word0 (Copy of OTP Bank 10, word 0). These bits cannot be read and written after the HW_OCOTP_LOCK_MAU_KEY bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.58 Shadow Register for OTP Bank10 Word1 (MAU Key) (OCOTP_MAU_KEY1)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 690h offset = 3035_0690h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MAU_KEY1 field descriptions

Field	Description
BITS	Shadow register for the MAU Key word1 (Copy of OTP Bank 10, word 1). These bits cannot be read and written after the HW_OCOTP_LOCK_MAU_KEY bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.59 Shadow Register for OTP Bank10 Word2 (MAU Key) (OCOTP_MAU_KEY2)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 6A0h offset = 3035_06A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MAU_KEY2 field descriptions

Field	Description
BITS	Shadow register for the MAU Key word2 (Copy of OTP Bank 10, word 2). These bits cannot be read and written after the HW_OCOTP_LOCK_MAU_KEY bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.60 Shadow Register for OTP Bank10 Word3 (MAU Key) (OCOTP_MAU_KEY3)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 6B0h offset = 3035_06B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MAU_KEY3 field descriptions

Field	Description
BITS	Shadow register for the MAU Key word3 (Copy of OTP Bank 10, word 3). These bits cannot be read and written after the HW_OCOTP_LOCK_MAU_KEY bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.61 Shadow Register for OTP Bank11 Word0 (MAU Key) (OCOTP_MAU_KEY4)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 6C0h offset = 3035_06C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MAU_KEY4 field descriptions

Field	Description
BITS	Shadow register for the MAU Key word4 (Copy of OTP Bank 11, word 0). These bits cannot be read and written after the HW_OCOTP_LOCK_MAU_KEY bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.62 Shadow Register for OTP Bank11 Word1 (MAU Key) (OCOTP_MAU_KEY5)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 6D0h offset = 3035_06D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MAU_KEY5 field descriptions

Field	Description
BITS	Shadow register for the MAU Key word5 (Copy of OTP Bank 11, word 1). These bits cannot be read and written after the HW_OCOTP_LOCK_MAU_KEY bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.63 Shadow Register for OTP Bank11 Word2 (MAU Key) (OCOTP_MAU_KEY6)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 6E0h offset = 3035_06E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MAU_KEY6 field descriptions

Field	Description
BITS	Shadow register for the MAU Key word6 (Copy of OTP Bank 11, word 2). These bits cannot be read and written after the HW_OCOTP_LOCK_MAU_KEY bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.64 Shadow Register for OTP Bank11 Word3 (MAU Key) (OCOTP_MAU_KEY7)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 6F0h offset = 3035_06F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MAU_KEY7 field descriptions

Field	Description
BITS	Shadow register for the MAU Key word7 (Copy of OTP Bank 11, word 3). These bits cannot be read and written after the HW_OCOTP_LOCK_MAU_KEY bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

6.4.5.65 Value of OTP Bank14 Word0 (OCOTP_GP10)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 780h offset = 3035_0780h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_GP10 field descriptions

Field	Description
BITS	It reflects the value of OTP Bank 14, word 0.

6.4.5.66 Value of OTP Bank14 Word1 (OCOTP_GP11)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 790h offset = 3035_0790h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_GP11 field descriptions

Field	Description
BITS	It reflects the value of OTP Bank 14, word 1.

6.4.5.67 Value of OTP Bank14 Word2 (OCOTP_GP20)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 7A0h offset = 3035_07A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_GP20 field descriptions

Field	Description
BITS	It reflects the value of OTP Bank 14, word 2.

6.4.5.68 Value of OTP Bank14 Word3 (OCOTP_GP21)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 7B0h offset = 3035_07B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_GP21 field descriptions

Field	Description
BITS	It reflects the value of OTP Bank 14, word 3.

6.4.5.69 Value of OTP Bank15 Word0 (CRC Key) (OCOTP_CRC_GP10)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 7C0h offset = 3035_07C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_CRC_GP10 field descriptions

Field	Description
BITS	It reflects the value of OTP Bank 15, word 0.

6.4.5.70 Value of OTP Bank15 Word1 (CRC Key) (OCOTP_CRC_GP11)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 7D0h offset = 3035_07D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_CRC_GP11 field descriptions

Field	Description
BITS	It reflects the value of OTP Bank 15, word 1.

6.4.5.71 Value of OTP Bank15 Word2 (CRC Key) (OCOTP_CRC_GP20)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 7E0h offset = 3035_07E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_CRC_GP20 field descriptions

Field	Description
BITS	It reflects the value of OTP Bank 15, word 2.

6.4.5.72 Value of OTP Bank15 Word3 (CRC Key) (OCOTP_CRC_GP21)

After reset, it automatically copied from the OTP. It can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

EXAMPLE

```
Empty Example.
```

Address: 3035_0000h base + 7F0h offset = 3035_07F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_CRC_GP21 field descriptions

Field	Description
BITS	It reflects the value of OTP Bank 15, word 3.

6.5 Watchdog Timer (WDOG)

6.5.1 Overview

The Watchdog Timer (WDOG) protects against system failures by providing a method by which to escape from unexpected events or programming errors.

Once the WDOG is activated, it must be serviced by the software on a periodic basis. If servicing does not take place, the timer times out. Upon timeout, the WDOG asserts the internal system reset signal, WDOG_RESET_B_DEB to the System Reset Controller (SRC).

There is also a provision for WDOG signal assertion by timeout counter expiration. There is an option of programmable interrupt generation before the counter actually times out. The time at which the interrupt needs to be generated prior to counter timeout is programmable. There is a power down counter which is enabled out of any reset (POR, Warm/Cold). This counter has a fixed timeout period of 16 seconds, upon which it asserts the WDOG signal.

Flow diagrams for the timeout counter, power down counter and interrupt operations are are shown in Flow Diagrams.

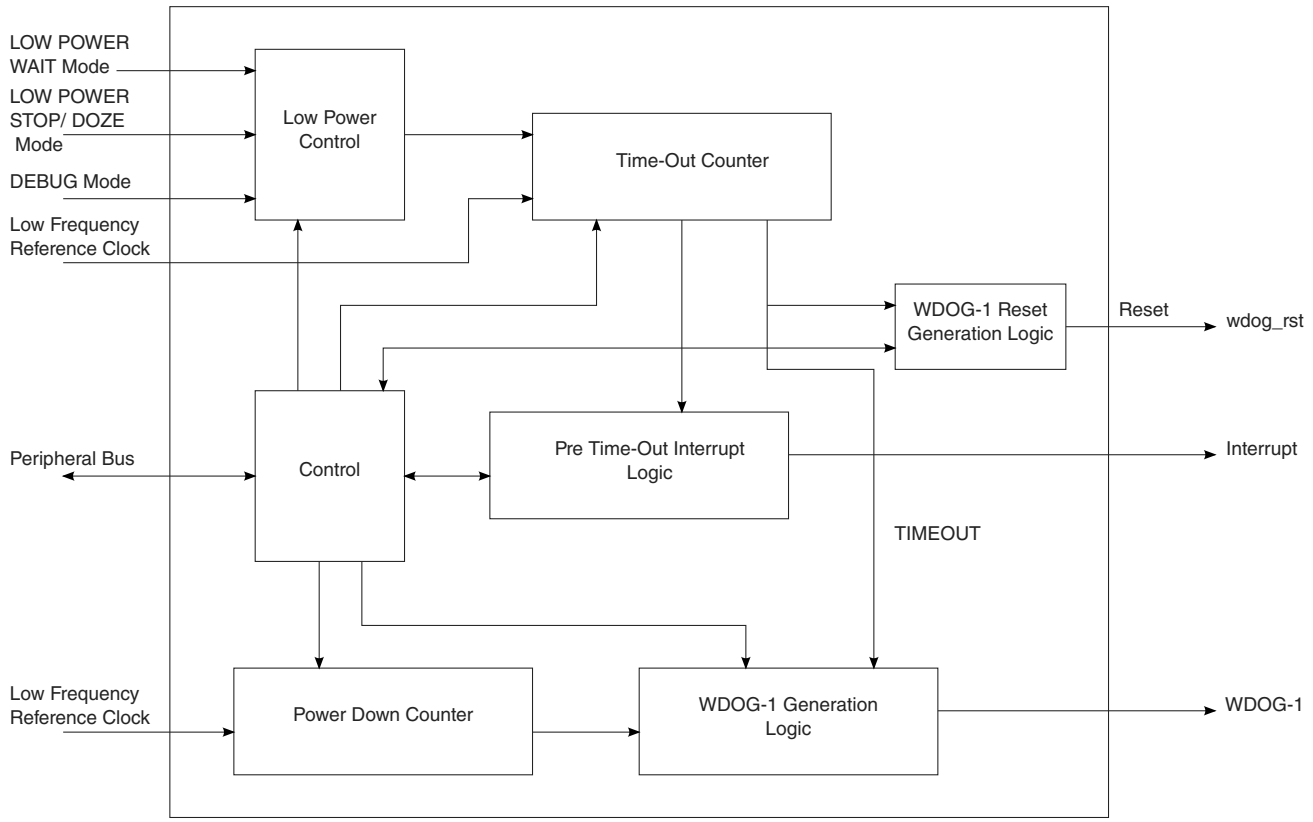


Figure 6-14. WDOG Diagram

6.5.1.1 Features

The WDOG features are listed below:

- Configurable timeout counter with timeout periods from 0.5 to 128 seconds which, after timeout expiration, result in the assertion of WDOG_RESET_B_DEB reset signal .
- Time resolution of 0.5 seconds
- Configurable timeout counter that can be programmed to run or stop during low-power modes
- Configurable timeout counter that can be programmed to run or stop during DEBUG mode
- Programmable interrupt generation prior to timeout

- The duration between interrupt and timeout events can be programmed from 0 to 127.5 seconds in steps of 0.5 seconds.
- Power down counter with fixed timeout period of 16 seconds, which if not disabled after reset will assert WDOG_B signal low
  - Power down counter will be enabled out of any reset (POR, Warm / Cold reset) by default.

6.5.2 External signals

Table 6-22. WDOG External Signals

Signal	Description	Pad	Mode	Direction
WDOG1_ANY	Global WDOG signal	ENET1_COL	ALT1	IO
		GPIO1_IO00	ALT2	
WDOG1_B	This signal will power down the chip.	GPIO1_IO00	ALT3	IO
		GPIO1_IO08	ALT2	
WDOG1_RESET_B_DEB	This signal is a reset source for the chip.	GPIO1_IO00	ALT4	O
WDOG2_B	This signal will power down the chip.	ENET1_RX_CLK	ALT1	IO
WDOG2_RESET_B_DEB	This signal is a reset source for the chip.	ENET1_CRS	ALT1	O
WDOG3_B	This signal will power down the chip.	GPIO1_IO14	ALT4	IO
		I2C2_SCL	ALT2	
WDOG3_RESET_B_DEB	This signal is a reset source for the chip.	I2C2_SDA	ALT2	O
WDOG4_B	This signal will power down the chip.	GPIO1_IO15	ALT4	IO
		I2C4_SCL	ALT2	
WDOG4_RESET_B_DEB	This signal is a reset source for the chip.	I2C4_SDA	ALT2	O

6.5.3 Clocks

This section describes clocks and special clocking requirements of the block.

The WDOG uses the low frequency reference clock for its counter and control operations. The peripheral bus clock is used for register read/write operations.

The following table describes the clock sources for WDOG. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 6-23. WDOG Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	IP Global functional clock. All functionality inside the WDOG module is synchronized to this clock.
ipg_clk_s	ipg_clk_root	IP slave bus clock. This clock is synchronized to ipg_clk and is only used for register read/write operations.
ipg_clk_32k	ckil_sync_clk_root	Low frequency (32.768 kHz) clock that continues to run in low-power mode. It is assumed that the Clock Controller will provide this clock signal synchronized to ipg_clk in the normal mode, and switch to a non-synchronized signal in low-power mode when the ipg_clk is off.

6.5.4 Functional description

This section provides a complete functional description of the block.

6.5.4.1 Timeout event

The WDOG provides timeout periods from 0.5 to 128 seconds with a time resolution of 0.5 seconds.

The user can determine the timeout period by writing to the WDOG timeout field (WT[7:0]) in the Watchdog Control Register (WDOG_WCR). The WDOG must be enabled by setting the WDE bit of Watchdog Control Register (WDOG_WCR) for the timeout counter to start running. After the WDOG is enabled, the counter is activated, loads the timeout value and begins to count down from this programmed value. The timer will time out when the counter reaches zero and the WDOG outputs a system reset signal, WDOG_RESET_B_DEB and asserts WDOG_B (WDT bit should be set in Watchdog Control Register (WDOG_WCR)).

However, the timeout condition can be prevented by reloading the counter with the new timeout value (WT[7:0] of WDOG_WCR) if a service routine (see Servicing WDOG to reload the counter) is performed before the counter reaches zero. If any system errors occur which prevent the software from servicing the Watchdog Service Register (WDOG_WSR), the timeout condition occurs. By performing the service routine, the WDOG reloads its counter to the timeout value indicated by bits WT[7:0] of the Watchdog Control Register (WDOG_WCR) and it restarts the countdown.

A system reset will reset the counter and place it in the idle state at any time during the countdown. The counter flow diagram is shown in Flow Diagrams.

NOTE

The timeout value is reloaded to the counter either at the time WDOG is enabled or after the service routine has been performed.

6.5.4.1.1 Servicing WDOG to reload the counter

To reload a timeout value to the counter the proper service sequence begins by writing 0x_5555 followed by 0x_AAAA to the Watchdog Service Register (WDOG_WSR). Any number of instructions can be executed between the two writes. If the WDOG_WSR is not loaded with 0x_5555 prior to writing 0x_AAAA to the WDOG_WSR, the counter is not reloaded. If any value other than 0x_AAAA is written to the WDOG_WSR after 0x_5555, the counter is not reloaded. This service sequence will reload the counter with the timeout value WT[7:0] of Watchdog Control Register (WDOG_WCR). The timeout value can be changed at any point; it is reloaded when WDOG is serviced by the core.

6.5.4.2 Interrupt event

Prior to timeout, the WDOG can generate an interrupt which can be considered a warning that timeout will occur shortly.

The duration between interrupt event and timeout event can be controlled by writing to the WICT field of Watchdog Interrupt Control Register (WDOG_WICR). It can vary between 0 and 127.5 seconds. If the WDOG is serviced (Servicing WDOG to reload the counter) before the interrupt generation, the counter will be reloaded with the timeout value WT[7:0] of Watchdog Control Register (WDOG_WCR) and the interrupt will not be triggered.

6.5.4.3 Power-down counter event

The power-down counter inside WDOG will be enabled out of reset. This counter has a fixed timeout value of 16 seconds, after which it will drive the WDOG_B signal low.

To prevent this, the software must disable this counter by clearing the PDE bit of Watchdog Miscellaneous Control Register (WDOG_WMCR) within 16 seconds of reset deassertion. Once disabled, this counter can't be enabled again until the next system reset occurs. This feature is intended to prevent the hanging up of cores after reset, as WDOG is not enabled out of reset.

6.5.4.4 Low power modes

6.5.4.4.1 STOP and DOZE mode

If the WDOG timer disable bit for low power STOP and DOZE mode (WDZST) bit in the Watchdog Control Register (WDOG_WCR), is cleared, the WDOG timer continues to operate using the low frequency reference clock. If the low power enable (WDZST) bit is set, the WDOG timer operation will be suspended in low power STOP or DOZE mode. Upon exiting low power STOP or DOZE mode, the WDOG operation returns to what it was prior to entering the STOP or DOZE mode.

6.5.4.4.2 WAIT mode

If the WDOG timer disable bit for low power WAIT mode (WDW) bit in the Watchdog Control Register (WDOG_WCR), is cleared, the WDOG timer continues to operate using the low frequency reference clock . If the low power WAIT enable (WDW) bit is set, the WDOG timer operation will be suspended. Upon exiting low power WAIT mode, the WDOG operation returns to what it was prior to entering the WAIT mode.

NOTE

The WDOG timer won't be able to detect events that happen for periods shorter than one low frequency reference clock cycle. For example, in repeated WAIT mode entry or exit, if the RUN mode time is less than one low frequency reference clock cycle and if the WDW bit is set, the WDOG timer may never time out, even though the system is in RUN mode for a finite duration; WDOG may not see a low frequency reference clock edge during its wake time.

6.5.4.5 Debug mode

The WDOG timer can be configured for continual operation, or for suspension during debug mode. If the WDOG debug enable (WDBG) bit is set in the Watchdog Control Register (WDOG_WCR), the WDOG timer operation is suspended in debug mode. If the WDBG bit is set and the debug mode is entered, WDOG timer operation is suspended after two low frequency reference clocks. Similarly, WDOG timer operation continues after two low frequency reference clocks of debug mode exit. Register read and write accesses in debug mode continue to function normally. Also, while in debug mode, the WDE bit of Watchdog Control Register (WDOG_WCR) can be enabled/disabled directly. If the WDOG debug enable (WDBG) bit is cleared then WDOG timer operation is not suspended. The power-down counter is not affected by debug mode entry/exit.

NOTE

If the WDE bit of Watchdog Control Register (WDOG_WCR) is set/cleared while in debug mode, it remains set/cleared even after exiting debug mode.

6.5.4.6 Operations

6.5.4.6.1 Watchdog reset generation

The WDOG generated reset signal WDOG_RESET_B_DEB is asserted by the following operations:

- A software write to the Software Reset Signal (SRS) bit of the Watchdog Control Register (WDOG_WCR).
- WDOG timeout. See Timeout event.

The $\overline{\text{wdog\_rst}}$ will be asserted for one clock cycle of low frequency reference clock for both a timeout condition and a software write occurrence. It remains asserted for 1 clock cycle of low frequency reference clock even if a system reset is asserted in between. Figure 6-16 shows the timing diagram of this signal due to a timeout condition.

6.5.4.6.2 WDOG_B generation

The WDOG asserts WDOG_B in the following scenarios:

- Software write to WDA bit of Watchdog Control Register (WDOG_WCR). WDOG_B signal remains asserted as long as the WDA bit is "0".
- WDOG timeout condition, WDT bit of Watchdog Control Register (WDOG_WCR) must be set for this scenario. A description of the timeout condition can be found in the Timeout event. WDOG_B signal remains asserted until a power-on reset (POR) occurs. It gets cleared after the POR occurs (not due to any other system reset). Figure 6-17 shows the timing diagram of WDOG_B due to timeout condition.
- WDOG power-down counter timeout, PDE bit of Watchdog Miscellaneous Control Register (WDOG_WMCR) should not be cleared for this scenario. A description of this counter can be found in the Power-down counter event. WDOG_B signal remains asserted for one clock cycle of low frequency reference clock.

Figure 6-15 shows the scenarios under which WDOG_B gets asserted.

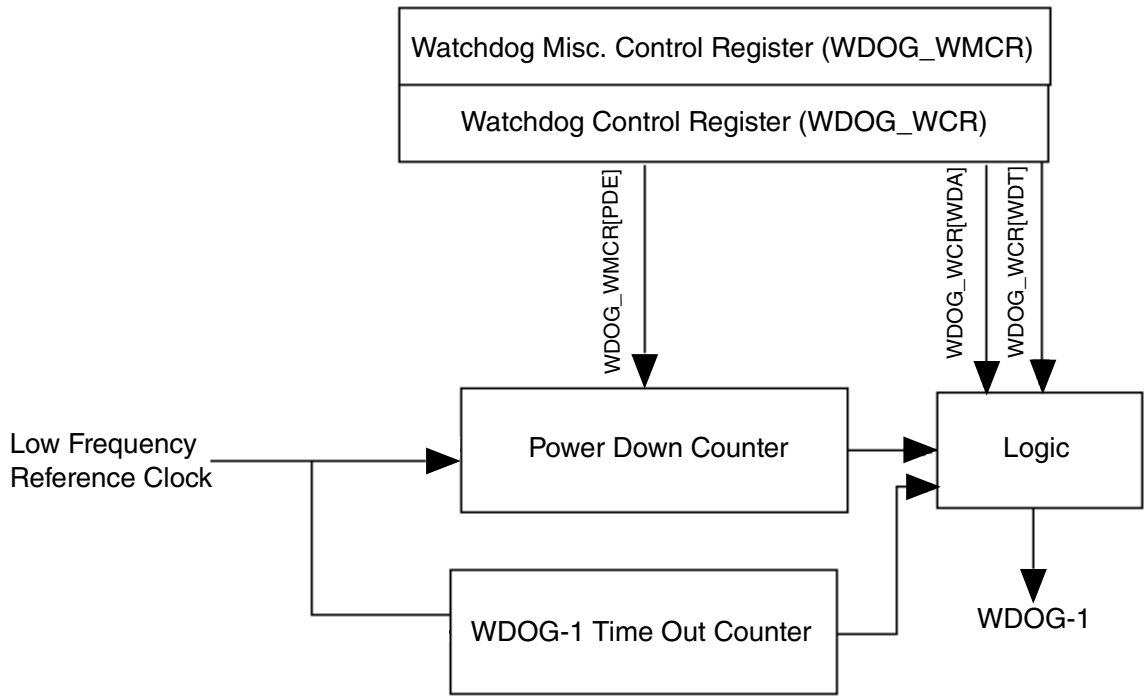


Figure 6-15. WDOG_B generation

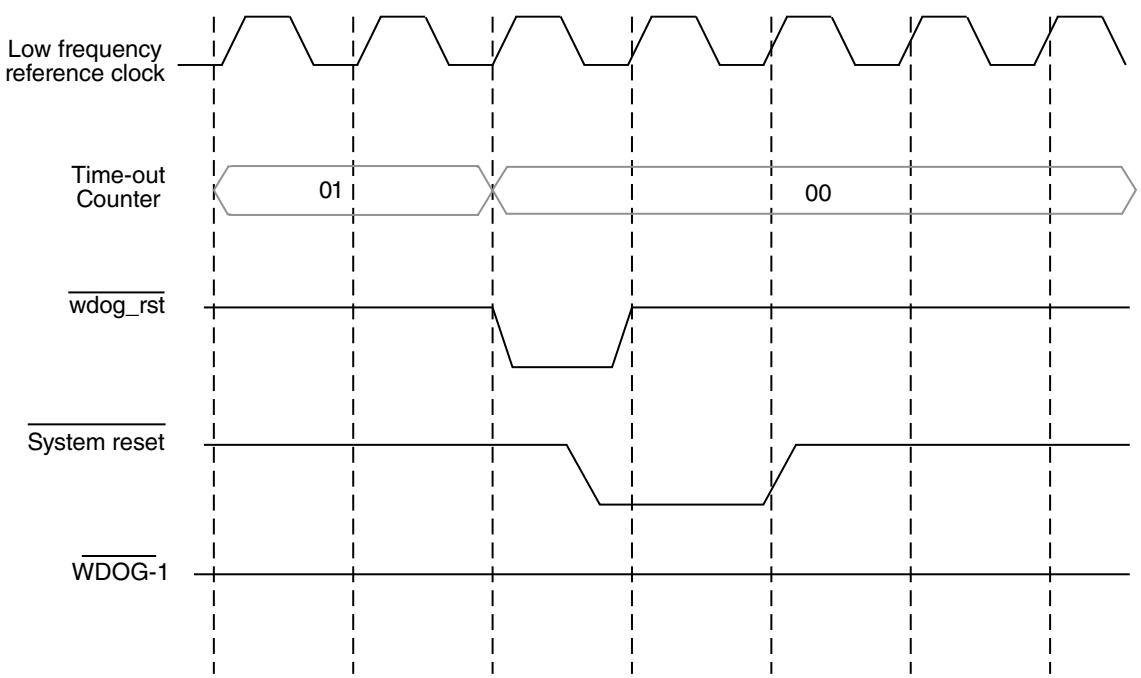


Figure 6-16. WDOG timeout vondition/WDT bit is not set

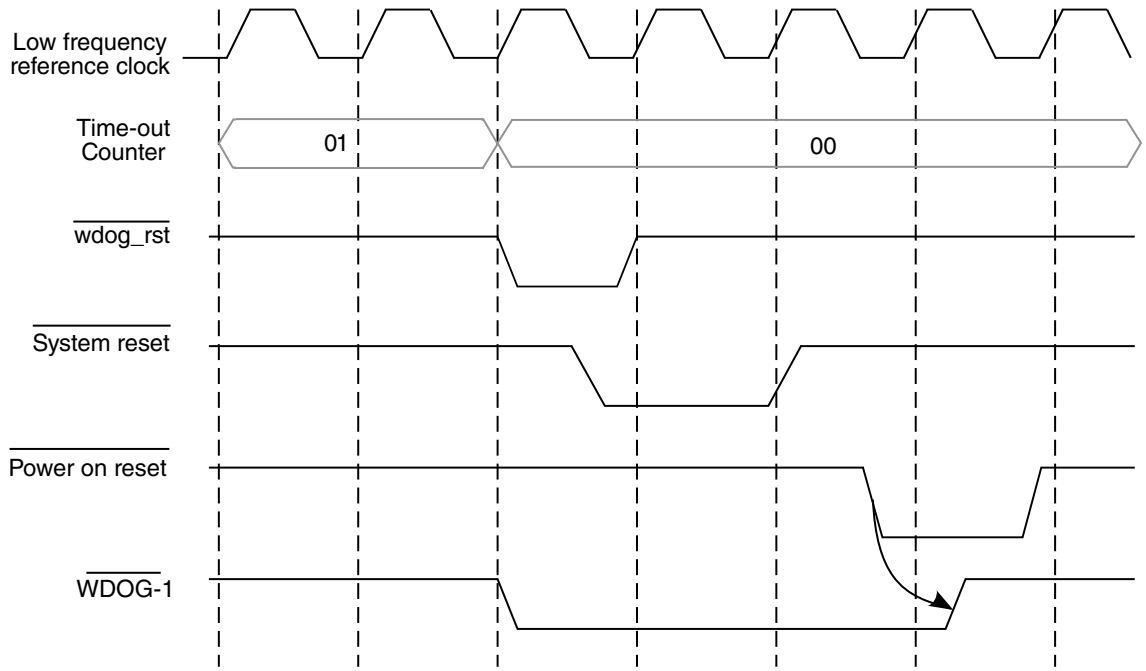


Figure 6-17. WDOG timeout condition/WDT bit is set

6.5.4.7 Reset

The block is reset by a system reset and the WDOG counter will be disabled. The power-down counter is enabled and starts counting.

6.5.4.8 Interrupt

The WDOG has the feature of Interrupt generation before timeout.

The interrupt will be generated only if the WIE bit in Watchdog Interrupt Control Register (WDOG_WICR) is set. The exact time at which the interrupt should occur (prior to timeout) depends on the value of WICT field of Watchdog Interrupt Control Register (WDOG_WICR). For example, if the WICT field has a value 0x04, then the interrupt will be generated two seconds prior to timeout. Once the interrupt is triggered the WTIS bit in Watchdog Interrupt Control Register (WDOG_WICR) will be set. The software needs to clear this bit to deassert the interrupt. If the WDOG is serviced before the interrupt generation then the counter will be reloaded with the timeout value WT[7:0] of Watchdog Control Register (WDOG_WCR) and interrupt would not be triggered.

6.5.4.9 Flow Diagrams

A flow diagram of WDOG operation is shown below.

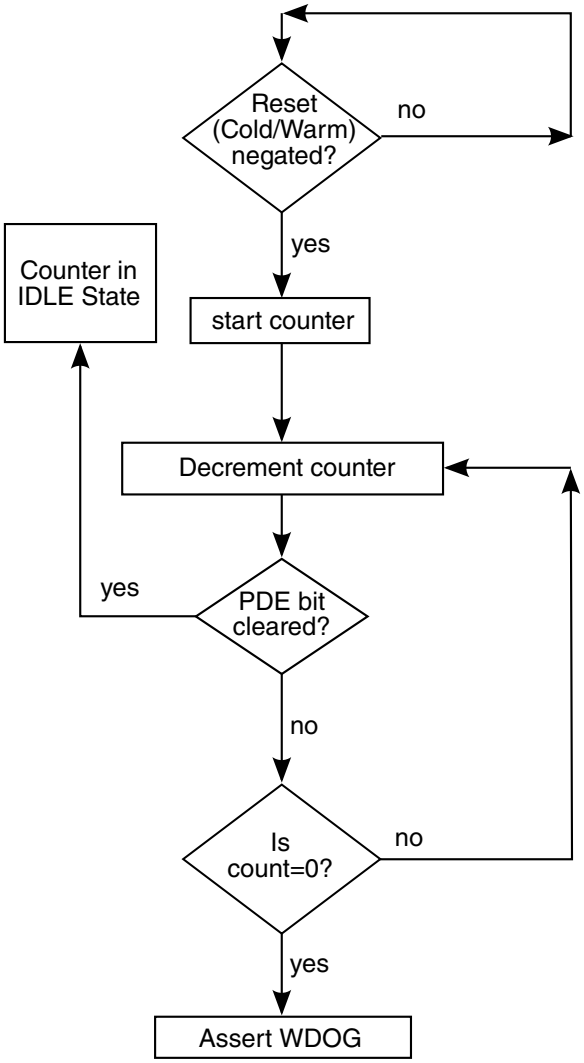


Figure 6-18. Power-Down Counter Flow Diagram

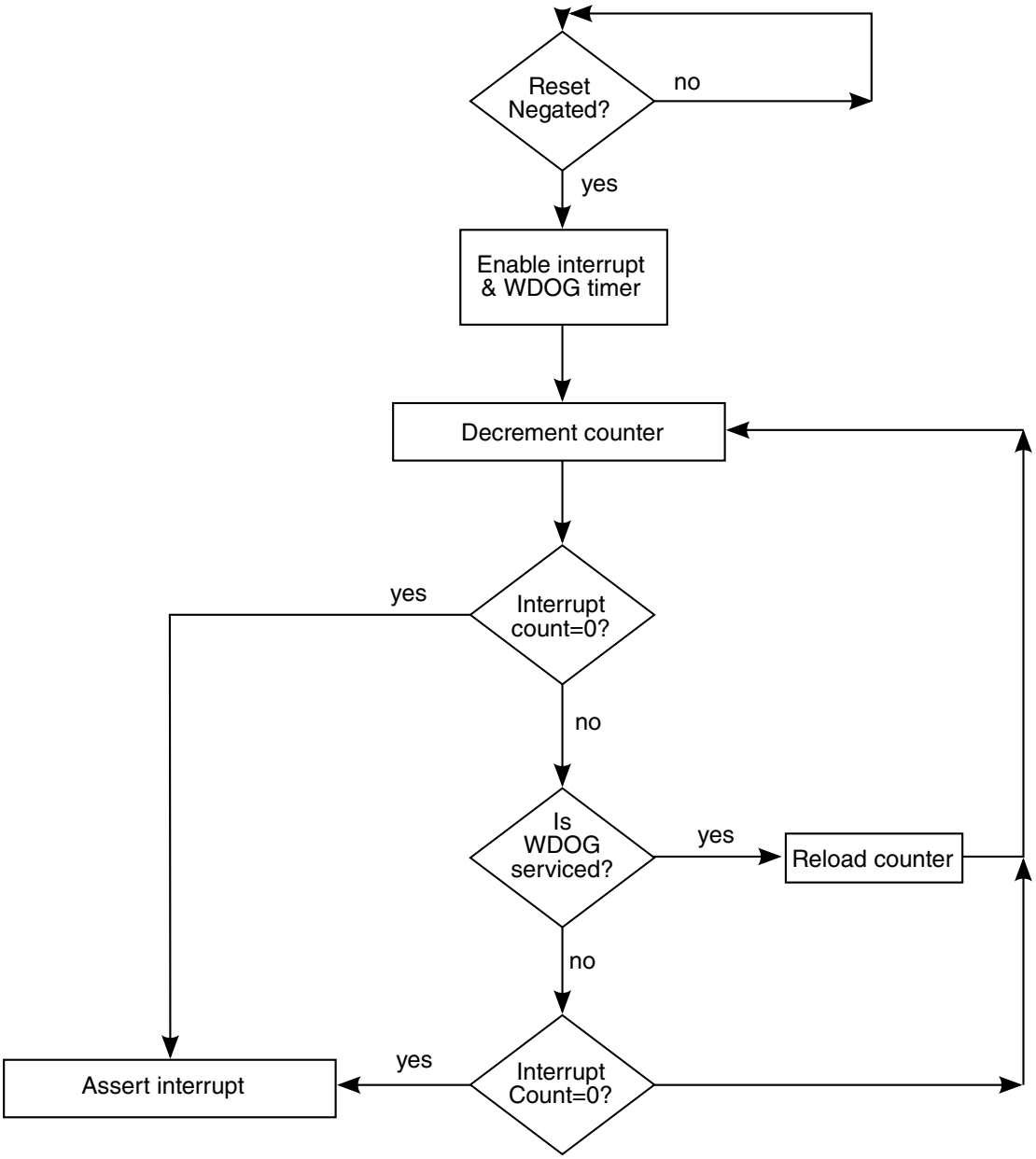


Figure 6-19. Interrupt Generation Flow Diagram

6.5.5 Initialization

The following sequence should be performed for WDOG initialization.

- PDE bit of Watchdog Miscellaneous Control Register (WDOG_WMCR) should be cleared to disable the power down counter.

- WT field of Watchdog Control Register (WDOG_WCR) should be programmed for sufficient timeout value.
- WDOG should be enabled by setting WDE bit of Watchdog Control Register (WDOG_WCR) so that the timeout counter loads the WT field value of Watchdog Control Register (WDOG_WCR) and starts counting.

6.5.6 WDOG Memory Map/Register Definition

The WDOG has user-accessible, 16-bit registers used to configure, operate, and monitor the state of the Watchdog Timer. Byte operations can be performed on these registers. If a 32-bit access is performed,the WDOG will not generate a peripheral bus error but will behave normally, like a 16-Bit access, making read/write possible. A 32-Bit access should be avoided, as the system may go to an unknown state.

WDOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3028_0000	Watchdog Control Register (WDOG1_WCR)	16	R/W	0030h	6.5.6.1/ 1116
3028_0002	Watchdog Service Register (WDOG1_WSR)	16	R/W	0000h	6.5.6.2/ 1118
3028_0004	Watchdog Reset Status Register (WDOG1_WRSR)	16	R	0000h	6.5.6.3/ 1119
3028_0006	Watchdog Interrupt Control Register (WDOG1_WICR)	16	R/W	0004h	6.5.6.4/ 1120
3028_0008	Watchdog Miscellaneous Control Register (WDOG1_WMCR)	16	R/W	0001h	6.5.6.5/ 1121
3029_0000	Watchdog Control Register (WDOG2_WCR)	16	R/W	0030h	6.5.6.1/ 1116
3029_0002	Watchdog Service Register (WDOG2_WSR)	16	R/W	0000h	6.5.6.2/ 1118
3029_0004	Watchdog Reset Status Register (WDOG2_WRSR)	16	R	0000h	6.5.6.3/ 1119
3029_0006	Watchdog Interrupt Control Register (WDOG2_WICR)	16	R/W	0004h	6.5.6.4/ 1120
3029_0008	Watchdog Miscellaneous Control Register (WDOG2_WMCR)	16	R/W	0001h	6.5.6.5/ 1121
302A_0000	Watchdog Control Register (WDOG3_WCR)	16	R/W	0030h	6.5.6.1/ 1116
302A_0002	Watchdog Service Register (WDOG3_WSR)	16	R/W	0000h	6.5.6.2/ 1118

Table continues on the next page...

WDOG memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
302A_0004	Watchdog Reset Status Register (WDOG3_WRSR)	16	R	0000h	6.5.6.3/ 1119
302A_0006	Watchdog Interrupt Control Register (WDOG3_WICR)	16	R/W	0004h	6.5.6.4/ 1120
302A_0008	Watchdog Miscellaneous Control Register (WDOG3_WMCR)	16	R/W	0001h	6.5.6.5/ 1121
302B_0000	Watchdog Control Register (WDOG4_WCR)	16	R/W	0030h	6.5.6.1/ 1116
302B_0002	Watchdog Service Register (WDOG4_WSR)	16	R/W	0000h	6.5.6.2/ 1118
302B_0004	Watchdog Reset Status Register (WDOG4_WRSR)	16	R	0000h	6.5.6.3/ 1119
302B_0006	Watchdog Interrupt Control Register (WDOG4_WICR)	16	R/W	0004h	6.5.6.4/ 1120
302B_0008	Watchdog Miscellaneous Control Register (WDOG4_WMCR)	16	R/W	0001h	6.5.6.5/ 1121

6.5.6.1 Watchdog Control Register (WDOGx_WCR)

The Watchdog Control Register (WDOG_WCR) controls the WDOG operation.

- WDZST, WDBG and WDW are write-once only bits. Once the software does a write access to these bits, they will be locked and cannot be reprogrammed until the next system reset assertion.
- WDE is a write one once only bit. Once software performs a write "1" operation to this bit it cannot be reset/cleared until the next system reset.
- WDT is also a write one once only bit. Once software performs a write "1" operation to this bit it cannot be reset/cleared until the next POR. This bit does not get reset/ cleared due to any system reset.

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8
Read Write	WT							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	WDW	SRE	WDA	SRS	WDT	WDE	WDBG	WDZST
Reset	0	0	1	1	0	0	0	0

WDOGx_WCR field descriptions

Field	Description
15–8 WT	Watchdog Time-out Field. This 8-bit field contains the time-out value that is loaded into the Watchdog counter after the service routine has been performed or after the Watchdog is enabled. After reset, WT[7:0] must have a value written to it before enabling the Watchdog otherwise count value of zero which is 0.5 seconds is loaded into the counter. **NOTE:** The time-out value can be written at any point of time but it is loaded to the counter at the time when WDOG is enabled or after the service routine has been performed. For more information see Timeout event . 0x00 - 0.5 Seconds (Default). 0x01 - 1.0 Seconds. 0x02 - 1.5 Seconds. 0x03 - 2.0 Seconds. 0xff - 128 Seconds.
7 WDW	Watchdog Disable for Wait. This bit determines the operation of WDOG during Low Power WAIT mode. This is a write once only bit. 0 Continue WDOG timer operation (Default). 1 Suspend WDOG timer operation.
6 SRE	software reset extension, an option way to generate software reset adopt a new way to generate a more robust software reset. This bit can be set/clear with IP bus and will be reset with power-on reset . 0 using original way to generate software reset (default) 1 using new way to generate software reset.
5 WDA	WDOG_B assertion. Controls the software assertion of the WDOG_B signal. 0 Assert WDOG_B output. 1 No effect on system (Default).
4 SRS	Software Reset Signal. Controls the software assertion of the WDOG-generated reset signal WDOG_RESET_B_DEB . This bit automatically resets to "1" after it has been asserted to "0". **NOTE:** This bit does not generate the software reset to the block. 0 Assert system reset signal. 1 No effect on the system (Default).
3 WDT	WDOG_B Time-out assertion. Determines if the WDOG_B gets asserted upon a Watchdog Time-out Event. This is a write-one once only bit. **NOTE:** There is no effect on WDOG_RESET_B_DEB (WDOG Reset) upon writing on this bit. WDOG_B gets asserted along with WDOG_RESET_B_DEB if this bit is set. 0 No effect on WDOG_B (Default). 1 Assert WDOG_B upon a Watchdog Time-out event.
2 WDE	Watchdog Enable. Enables or disables the WDOG block. This is a write one once only bit. It is not possible to clear this bit by a software write, once the bit is set. **NOTE:** This bit can be set/reset in debug mode (exception). 0 Disable the Watchdog (Default). 1 Enable the Watchdog.

Table continues on the next page...

WDOGx_WCR field descriptions (continued)

Field	Description
1 WDBG	Watchdog DEBUG Enable. Determines the operation of the WDOG during DEBUG mode. This bit is write once only. 0 Continue WDOG timer operation (Default). 1 Suspend the watchdog timer.
0 WDZST	Watchdog Low Power. Determines the operation of the WDOG during low-power modes. This bit is write once-only. **NOTE:** The WDOG can continue/suspend the timer operation in the low-power modes (STOP and DOZE mode). 0 Continue timer operation (Default). 1 Suspend the watchdog timer.

6.5.6.2 Watchdog Service Register (WDOGx_WSR)

When enabled, the WDOG requires that a service sequence be written to the Watchdog Service Register (WSR) to prevent the timeout condition.

NOTE

Executing the service sequence will reload the WDOG timeout counter.

Address: Base address + 2h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	WSR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WDOGx_WSR field descriptions

Field	Description
WSR	Watchdog Service Register. This 16-bit field contains the Watchdog service sequence. Both writes must occur in the order listed prior to the time-out, but any number of instructions can be executed between the two writes. The service sequence must be performed as follows: 0x5555 Write to the Watchdog Service Register (WDOG_WSR). 0xAAAA Write to the Watchdog Service Register (WDOG_WSR).

6.5.6.3 Watchdog Reset Status Register (WDOGx_WRSR)

The WRSR is a read-only register that records the source of the output reset assertion. It is not cleared by a hard reset. Therefore, only one bit in the WRSR will always be asserted high. The register will always indicate the source of the last reset generated due to WDOG. Read access to this register is with one wait state. Any write performed on this register will generate a Peripheral Bus Error .

A reset can be generated by the following sources, as listed in priority from highest to lowest:

- Watchdog Time-out
- Software Reset

Address: Base address + 4h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0			POR	0		TOUT	SFTW
Write								
Reset	0	0	0	0	0	0	0	0

WDOGx_WRSR field descriptions

Field	Description
15–5 Reserved	This read-only field is reserved and always has the value 0.
4 POR	Power On Reset. Indicates whether the reset is the result of a power on reset. 0 Reset is not the result of a power on reset. 1 Reset is the result of a power on reset.
3–2 Reserved	This read-only field is reserved and always has the value 0.
1 TOUT	Timeout. Indicates whether the reset is the result of a WDOG timeout. 0 Reset is not the result of a WDOG timeout. 1 Reset is the result of a WDOG timeout.
0 SFTW	Software Reset. Indicates whether the reset is the result of a WDOG software reset by asserting SRS bit 0 Reset is not the result of a software reset. 1 Reset is the result of a software reset.

6.5.6.4 Watchdog Interrupt Control Register (WDOGx_WICR)

The WDOG_WICR controls the WDOG interrupt generation.

Address: Base address + 6h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	WIE	WTIS	0						WICT							
Write		w1c														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

WDOGx_WICR field descriptions

Field	Description
15 WIE	Watchdog Timer Interrupt enable bit. Reset value is 0. **NOTE:** This bit is a write once only bit. Once the software does a write access to this bit, it will get locked and cannot be reprogrammed until the next system reset assertion 0 Disable Interrupt (Default). 1 Enable Interrupt.
14 WTIS	Watchdog TImer Interrupt Status bit will reflect the timer interrupt status, whether interrupt has occurred or not. Once the interrupt has been triggered software must clear this bit by writing 1 to it. 0 No interrupt has occurred (Default). 1 Interrupt has occurred
13–8 Reserved	This read-only field is reserved and always has the value 0.
WICT	Watchdog Interrupt Count Time-out (WICT) field determines, how long before the counter time-out must the interrupt occur. The reset value is 0x04 implies interrupt will occur 2 seconds before time-out. The maximum value that can be programmed to WICT field is 127.5 seconds with a resolution of 0.5 seconds. **NOTE:** This field is write once only. Once the software does a write access to this field, it will get locked and cannot be reprogrammed until the next system reset assertion. 0x00 WICT[7:0] = Time duration between interrupt and time-out is 0 seconds. 0x01 WICT[7:0] = Time duration between interrupt and time-out is 0.5 seconds. 0x04 WICT[7:0] = Time duration between interrupt and time-out is 2 seconds (Default). 0xff WICT[7:0] = Time duration between interrupt and time-out is 127.5 seconds.

6.5.6.5 Watchdog Miscellaneous Control Register (WDOGx_WMCR)

WDOG_WMCR Controls the Power Down counter operation.

Address: Base address + 8h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0															PDE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

WDOGx_WMCR field descriptions

Field	Description
15–1 Reserved	This read-only field is reserved and always has the value 0.
0 PDE	Power Down Enable bit. Reset value of this bit is 1, which means the power down counter inside the WDOG is enabled after reset. The software must write 0 to this bit to disable the counter within 16 seconds of reset de-assertion. Once disabled this counter cannot be enabled again. See Power-down counter event for operation of this counter. **NOTE:** This bit is write-one once only bit. Once software sets this bit it cannot be reset until the next system reset. 0 Power Down Counter of WDOG is disabled. 1 Power Down Counter of WDOG is enabled (Default).

6.6 System Boot

6.6.1 Overview

The boot process begins at Power On Reset (POR) where the hardware reset logic forces the ARM core to begin execution starting from the on-chip boot ROM.

Boot ROM code uses the state of the internal register BOOT_MODE[1:0] as well as the state of various eFUSEs and/or GPIO settings to determine the boot flow behavior of the device.

The main features of the ROM include:

- Support for booting from various boot devices
- Serial downloader support (USB OTG)
- Device configuration data (DCD) and plugin

- Digital signature and encryption based High Assurance Boot (HAB)
- Wake-up from low power modes

The boot ROM supports the following boot devices:

- NOR Flash
- NAND Flash
- OneNAND Flash
- SD/MMC
- Serial (I2C/SPI) NOR Flash
- QuadSPI (QSPI) Flash

The Boot ROM uses the state of BOOT_MODE and eFUSEs to determine the boot device. For development purposes, eFUSEs used to determine the boot device may be overridden by using GPIO pin inputs.

Boot ROM code also allows the downloading of programs to be run on the device. An example is a provisioning program that can make further use of the serial connection to provision a boot device with a new image. Typically the provisioning program is downloaded to internal RAM and allows the programming of boot devices, such as an SD/MMC Flash. The ROM Serial Downloader uses high speed USB in a non-stream mode connection.

Boot ROM allows waking up from low-power modes. On reset the ROM checks power gating status register. On waking from low power mode, the core will skip loading an image from the boot device and jump to the address saved in PERSISTENT_ENTRY0.

The device configuration data (DCD) feature allows boot ROM code to obtain SOC configuration data from an external Program Image residing on the boot device. As an example, DCD can be used to program the DDR controller for optimal settings improving the boot performance. DCD is restricted to memory areas and peripheral addresses that are considered essential for boot purposes (see Write Data Command).

A key feature of the boot ROM is the ability to perform a secure boot or High Assurance Boot (HAB). This is supported by the HAB security library which is a subcomponent of the ROM code. HAB uses a combination of hardware and software together with a Public Key Infrastructure (PKI) protocol to protect the system from executing unauthorized programs. Before the HAB allows a user's image to execute, the image must be signed. The signing process is done during the image build process by the private key holder and the signatures are then included as part of the final Program Image. If configured to do so, the ROM verifies the signatures using the public keys included in the Program Image. In addition to supporting digital signature verification to authenticate Program Images, Encrypted boot is also supported. Encrypted boot can be used to prevent cloning of the Program Image directly off the boot device. A secure boot with HAB can be performed

on all boot devices supported on the chip in addition to the Serial Downloader. The HAB library in the boot ROM also provides API functions, allowing additional boot chain components (bootloaders) to extend the secure boot chain. The out-of-fab setting for SEC_CONFIG is the Open configuration in which the ROM/HAB performs image authentication, but all authentication errors are ignored and the image is still allowed to execute.

6.6.2 Boot modes

During reset, the chip checks Power Gating Controller status register.

During boot, the core's behavior is defined by the Boot Mode pins settings as described in Boot mode pin settings. On waking up from low power boot mode, the core skips clock settings. Boot ROM checks that PERSISTENT_ENTRY0 (see Persistent Bits) is a pointer to valid address space (OCRAM, DDR, QSPI, or EIM). If PERSISTENT_ENTRY0 is a pointer to valid range, it starts execution using entry point from PERSISTENT_ENTRY0 register. If PERSISTENT_ENTRY0 is a pointer to invalid range, the core performs system reset.

6.6.2.1 Boot mode pin settings

The device has four boot modes (one is reserved for NXP use). Boot mode is selected based on the binary value stored in the internal BOOT_MODE register.

BOOT_MODE is initialized by sampling the BOOT_MODE0 and BOOT_MODE1 inputs on the rising edge of POR_B. After these inputs are sampled, their subsequent state does not affect the contents of the BOOT_MODE internal register. The state of the internal BOOT_MODE register may be read from the BMOD[1:0] field of the SRC Boot Mode Register (SRC_SBMR2). The available boot modes are: Boot From Fuses, serial boot via USB, and Internal Boot. See the table below for settings.

Table 6-24. Boot MODE Pin Settings

BOOT_MODE[1:0]	Boot Type
00	Boot From Fuses
01	Serial Downloader
10	Internal Boot
11	Reserved

6.6.2.2 High level boot sequence

The figure found here shows the high-level boot ROM code flow.

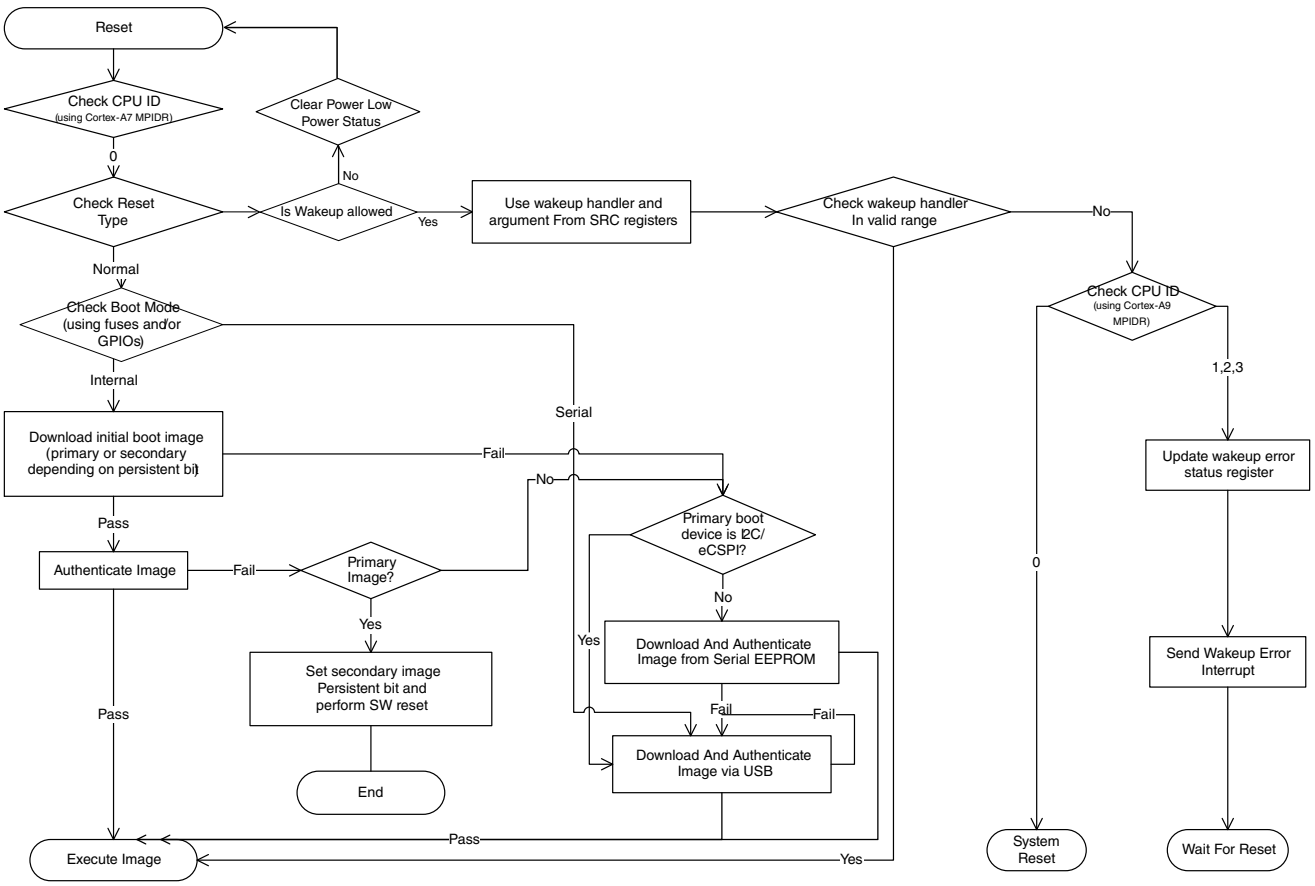


Figure 6-20. Boot Flow

6.6.2.3 Boot From Fuses Mode (BOOT_MODE[1:0] = 00b)

A value of 00b in the BOOT_MODE[1:0] register selects the Boot From Fuses mode.

This mode is similar to the Internal Boot mode described in Internal Boot Mode (BOOT_MODE[1:0] = 0b10) with one difference. In this mode the GPIO boot override pins are ignored. The boot ROM code uses the boot eFUSE settings only. This mode also supports a secure boot using HAB.

If set to Boot From Fuses, the boot flow is controlled by the BT_FUSE_SEL eFUSE value. If BT_FUSE_SEL = 0, indicating that the boot device (for example, Flash, SD/MMC) has not yet been programmed, the boot flow jumps directly to the Serial Downloader. If BT_FUSE_SEL = 1, the normal boot flow is followed, where the ROM attempts to boot from the selected boot device.

The first time a board is used, the default eFUSEs may be configured incorrectly for the hardware on the platform. In such a case, the Boot ROM code may try to boot from a device that does not exist. This may cause an electrical/logic violation on some pads. Using Boot From Fuses mode addresses this problem.

Setting BT_FUSE_SEL=0 forces the ROM code to jump directly to the Serial Downloader. This allows a bootloader to be downloaded which can then provision the boot device with a Program Image and blow the BT_FUSE_SEL and the other boot configuration eFUSEs. After reset, the Boot ROM code determines that BT_FUSE_SEL is blown (BT_FUSE_SEL = 1) and the ROM code performs internal boot according to the new eFUSE settings. This allows a user to set BOOT_MODE[1:0]=00b on a production device and burn fuses on the same device (by forcing entry to the Serial Downloader), without changing the value of BOOT_MODE[1:0] or pullups/pulldowns on the BOOT_MODE pins.

6.6.2.4 Serial Downloader

The Serial Downloader provides a means to download a Program Image to the chip over USB serial connection.

In this mode the ROM programs WDOG1 for a time-out specified by fuse WDOG Time-out Select (See fusemap for details) if WDOG_ENABLE eFuse is 1 and continuously polls for USB connection. If no activity is found on USB OTG1 and the watchdog timer expires, the ARM core is reset.

NOTE

After being loaded, the downloaded image is responsible to manage the watchdog resets properly.

Figure 6-21 shows USB boot flow.

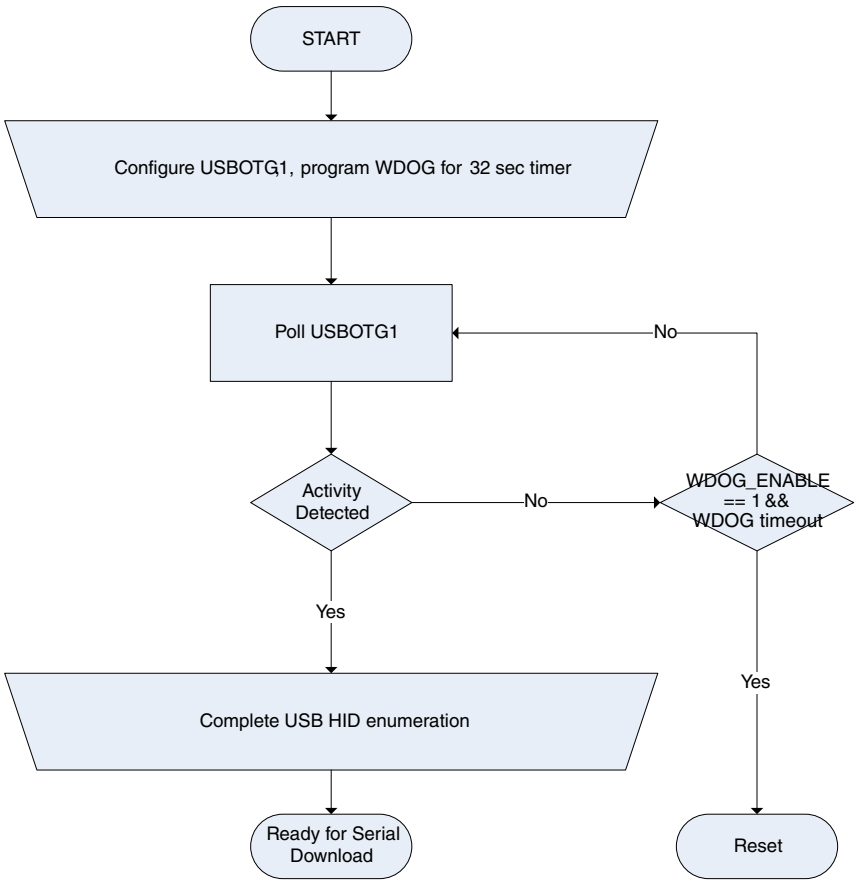


Figure 6-21. Serial Download Boot Flow

6.6.2.5 Internal Boot Mode (BOOT_MODE[1:0] = 0b10)

A value of 0b10 in the BOOT_MODE[1:0] register selects the internal boot mode. In this mode, the processor continues to execute boot code from the internal boot ROM.

The boot code performs hardware initialization, loads the Program Image from the chosen boot device, performs image validation using the HAB library (see Boot security settings), and then jumps to an address derived from the Program Image. If any error occurs during internal boot, the boot code jumps to the Serial Downloader (see Serial Downloader). A secure boot using the HAB is possible in all the three boot modes.

When set to internal boot, the boot flow may be controlled by a combination of eFUSE settings with an option of overriding the fuse settings using General Purpose I/O (GPIO) pins. The GPIO Boot Select FUSE (BT_FUSE_SEL) determines whether the ROM uses GPIO pins for a select number of configuration parameters or eFUSEs in this mode.

- If BT_FUSE_SEL = 1, all boot options are controlled by the eFUSEs described in Table 6-25.
- If BT_FUSE_SEL = 0, specific boot configuration parameters may be set using GPIO pins rather than eFUSEs. The fuses that can be overridden when in this mode are indicated in the GPIO column of Table 6-25. Table 6-26 provides the details on the GPIO pins.

The use of GPIO overrides is intended for development since these pads are used for other purposes in deployed products. NXP recommends controlling the boot configuration by eFUSEs in deployed products and reserving the use of the GPIO mode for development and testing purposes only.

6.6.2.6 Boot security settings

Internal boot modes use one of three security configurations:

- Closed: This level is intended for use with shipping secure products. All HAB functions are executed and security hardware is initialized (the Security Controller, or SNVS, enters Secure state), DCD is processed if present, and the program image is authenticated by HAB prior to its execution. All detected errors will be logged, and the boot flow aborted with control passing to the serial downloader. At this level, execution does not leave the internal ROM unless the target executable image has been authenticated.
- Open: This level is intended for use in non-secure products or during the development phases of a secure product. All HAB functions are executed as for a closed device. Security hardware is initialized (except the SNVS is left in Non-Secure state), DCD is processed if present, and the program image is authenticated by HAB prior to its execution. All detected errors will be logged, but have no influence on the boot flow, which continues as if the errors did not occur. This configuration is useful for secure product development, since the Program Image will run even if the authentication data is missing or incorrect, and the error log can be examined to determine the cause of authentication failure.
- Field Return: This level is intended for parts returned from shipped products.

NOTE

If the DIR_BT_DIS eFuse is not blown, authentication may be bypassed. In this case the system is not secure.

6.6.3 Device Configuration

This section describes the external inputs that control the behavior of the Boot ROM code.

This includes boot device selection (SPI, EIM, NOR, SD, MMC, QSPI, etc.), boot device configuration (SD bus width, speed, etc), and so on. In general, the source for this configuration comes from eFUSEs embedded inside the chip. However, certain configuration parameters can be sourced from GPIO pins allowing further flexibility during the development process.

6.6.3.1 Boot eFUSE Descriptions

The table below is a comprehensive list of the configuration parameters that the ROM uses.

Table 6-25. Boot eFUSE Descriptions

Fuse	Configuration	Definition	GPIO[1]	Shipped Value	Settings[2]
DIR_BT_DIS	OEM	Disables NXP reserved modes. Must be set for secure boot.	NA	0	0 Reserved NXP modes enabled 1 Reserved NXP modes disabled
BT_FUSE_SEL	OEM	In internal Boot mode BOOT_MODE[1:0] = 10, the BT_FUSE_SEL fuse determines whether the boot settings indicated by a Yes in the GPIO column are controlled by GPIO pins or eFUSE settings in the On-Chip OTP Controller (OCOTP). In Boot From Fuse mode BOOT_MODE[1:0] = 00, BT_FUSE_SEL fuse indicates whether bit configuration eFuses have been programmed.	NA	0	If BOOT_MODE[1:0] = 0b10 0 Bits of SBMR are overridden by GPIO pins. 1 Specific bits of SBMR are controlled by eFUSE settings. If BOOT_MODE[1:0] = 0b00 0 BOOT configuration eFuses are not yet programmed. Boot flow jumps to serial downloader. 1 BOOT configuration eFuses have been programmed. Regular boot flow is performed.
SEC_CONFIG[1:0]	SEC_CONFIG[0] - NXP SEC_CONFIG[1] - OEM	Security Configuration as defined in Boot security settings	NA	01	00 Reserved 01 Open (allows any program image, even if authentication fails) 1x Closed (Program image executes only if authenticated)

Table continues on the next page...

Table 6-25. Boot eFUSE Descriptions (continued)

Fuse	Configuration	Definition	GPIO[1]	Shipped Value	Settings[2]
FIELD_RETURN	OEM	Enables NXP reserved modes			0 - NXP reserved modes are enabled/ disabled based on DIR_BT_DIS value 1 - NXP reserved modes are enabled
SRK_HASH[255:0]	OEM	256-bit hash value of super root key (SRK_HASH)	NA	0	Settings vary - used by HAB
UNIQUE_ID[63:0]	NXP	Device Unique ID, 64-bit UID.	NA	Unique ID	Settings vary - used by HAB
BT_MMU_DISABLE (BOOT_CFG16)	OEM	MMU/L1 D Cache/PL310 disable bit used by boot ROM for fast HAB processing	No	0	0 - MMU/L1 D Cache/PL310 is enabled by ROM during the boot 1 - MMU/L1 D Cache/PL310 is disabled by ROM during the boot
BT_MMU_DISABLE (BOOT_CFG16)	OEM	SPI recovery boot enable	Yes	0	0 - Disabled 1 - Enabled
L1 I-Cache DISABLE	OEM	L1 I Cache disable bit used by boot during entire execution	No	0	0 - L1 I Cache is enabled by ROM during the boot 1 - L1 I Cache is disabled by ROM during the boot
BT_CFG[17]	OEM	SPI recovery boot enable	Yes	0	0 - Disabled 1 - Enabled
BT_FREQ (BOOT_CFG18)	OEM	Boot Frequency Selection	Yes	0	0 - ARM-792MHz, DDR-396MHz, AXI-396MHz 1 - ARM-396MHz, DDR-307MHz, AXI-307MHz ARM/DDR/AXI frequency may be lower if LP_BOOT is blown. See Table 6-66 for details.
BOOT_CFG[18:0]	OEM	Boot Configuration	Yes	0	Specific to selected boot mode
BOOT_CFG[19]	OEM	Infinite Loop Enable at start of boot ROM. Used for debugging purposes. Ignored if DIR_BT_DIS is 1 and FIELD_RETURN is 0.	Yes	0	0 - Disabled 1 - Enabled
LPB_BOOT	OEM	USB Low Power Boot	No	0	0x - LPB Disable 10 - Divide by 2 11 - Divide by 4
BT_LPB_POLARITY	OEM	USB Low Power Boot GPIO polarity	No	0	0 - low on GPIO pad indicates low power condition 1 - high on GPIO pad indicates low power condition
WDOG_ENABLE	OEM	Watchdog reset counter enable	No	0	0 - watchdog reset counter is disabled during serial downloader

Table continues on the next page...

Table 6-25. Boot eFUSE Descriptions (continued)

Fuse	Configuration	Definition	GPIO[1]	Shipped Value	Settings[2]
					1 - watchdog reset counter is enabled during serial downloader
MMC_DLL_DLY[6:0]	OEM	uSDHC Delay Line settings	No	0	uSDHC Delay Line settings
SRK_REVOKE[2:0]	OEM	SRK revocation mask	No	0	SRK revocation mask
DISABLE_SDMMC_MFG	OEM	Disable SDMMC manufacture mode	Yes	0	0: enable SD/MMC MFG mode 1: disable SD/MMC MFG mode
USDHC_PAD_SETTINGS NAND_PAD_SETTINGS	OEM	Override values for SD/MMC and NAND boot modes	No	0	Override the following IO PAD settings: [1:0] Driver Strength [2] Slew Rate [3] Hysteresis > [4] Pull/Keeper select [6:5] Pull up/down config .
OVERRIDE_HYS_SDMMC_PADS	OEM	Overrides HYS bit for SD pads	No	0	Override the IO PAD setting HYS to 1 for SD pads
eMMC_4.4_RESET_TO_PRE-IDLE_STATE	OEM	ROM reset the boot device in pre-idle state using eMMC 4.4 feature, CMD0 with argument value 0xf0f0f0f0	No	0	Applicable for booting from eMMC 4.4 spec or greater version devices. The fuse should not be blown for eMMC 4.3 or lesser spec version devices.

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See GPIO Boot Overrides for corresponding GPIO pin.
2. 0 = intact fuse and 1= blown fuse

6.6.3.2 GPIO Boot Overrides

The table below provides a list of GPIO boot overrides.

Table 6-26. GPIO Override Contact Assignments

Package Pin	Direction on reset	eFuse
BOOT_MODE1	Input	Boot Mode Selection
BOOT_MODE0	Input	
LCD1_DATA00	Input	BOOT_CFG[0]
LCD1_DATA01	Input	BOOT_CFG[1]
LCD1_DATA02	Input	BOOT_CFG[2]
LCD1_DATA03	Input	BOOT_CFG[3]

Table continues on the next page...

Table 6-26. GPIO Override Contact Assignments (continued)

Package Pin	Direction on reset	eFuse
LCD1_DATA04	Input	BOOT_CFG[4]
LCD1_DATA05	Input	BOOT_CFG[5]
LCD1_DATA06	Input	BOOT_CFG[6]
LCD1_DATA07	Input	BOOT_CFG[7]
LCD1_DATA08	Input	BOOT_CFG[8]
LCD1_DATA09	Input	BOOT_CFG[9]
LCD1_DATA10	Input	BOOT_CFG[10]
LCD1_DATA11	Input	BOOT_CFG[11]
LCD1_DATA12	Input	BOOT_CFG[12]
LCD1_DATA13	Input	BOOT_CFG[13]
LCD1_DATA14	Input	BOOT_CFG[14]
LCD1_DATA15	Input	BOOT_CFG[15]
LCD1_DATA16	Input	BOOT_CFG[16]
LCD1_DATA17	Input	BOOT_CFG[17]
LCD1_DATA18	Input	BOOT_CFG[18]
LCD1_DATA19	Input	BOOT_CFG[19]

The input pins provided are sampled at boot, and can be used to override corresponding eFUSE values, depending on the setting of the BT_FUSE_SEL fuse.

6.6.3.3 Device Configuration Data (DCD)

DCD is configuration information contained in a Program Image, external to the ROM, that the ROM interprets to configure various on-chip peripherals. See Device Configuration Data (DCD) for more details on Device Configuration Data.

6.6.4 Device Initialization

This section describes the details on the ROM and provides initialization details.

This includes details on:

- The ROM Memory Map
- The RAM Memory Map
- On-chip blocks that the ROM should make use of or change POR register default values
- Clock initialization

- Enabling the MMU/L2 cache
- Exception handling and interrupt handling

6.6.4.1 Internal ROM /RAM memory map

The following figure shows the iROM memory map.

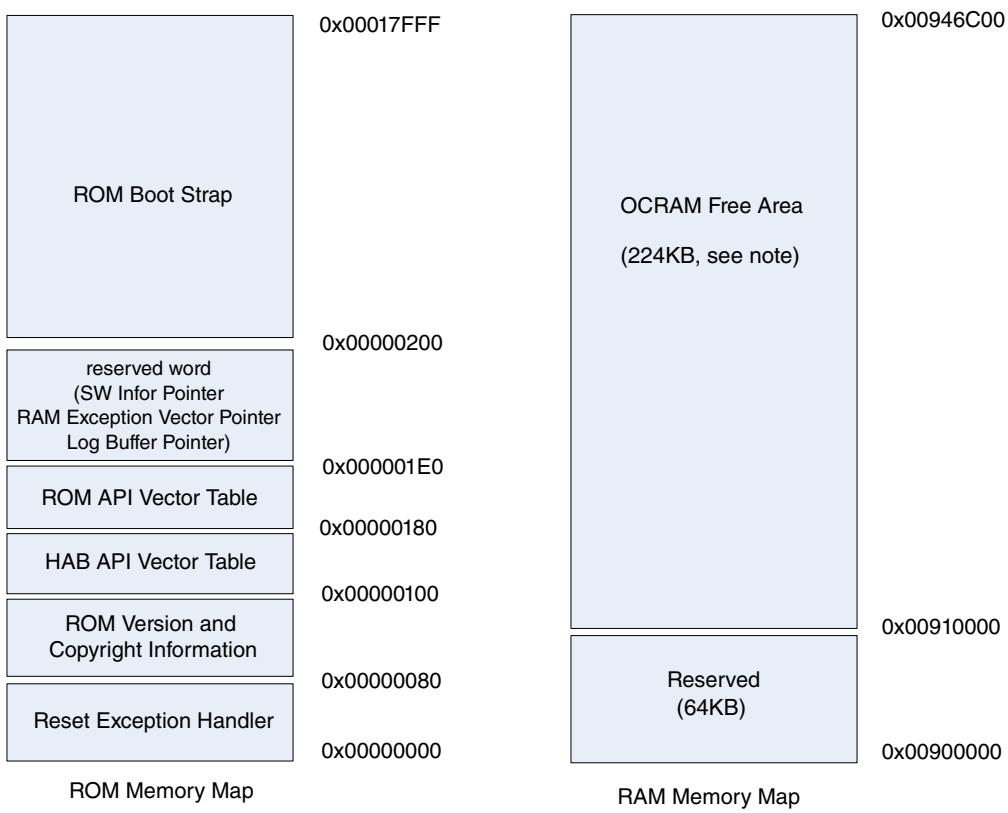


Figure 6-22. Internal ROM and RAM memory map

NOTE

The entire OCRAM region can be used freely post boot. ROM takes OCRAM_PXP(32KB) and OCRAM_GP(128KB) as OCRAM Free Area.

6.6.4.2 Boot Block Activation

The boot ROM affects a number of different hardware blocks which are activated and play a vital role in the boot flow.

The ROM configures and uses the following blocks (listed in alphabetical order) during the boot process. Note that the blocks actually used depend on the boot mode and boot device selection:

- APBH - DMA engine to drive the GPMI module
- BCH - 62-bit error correction hardware engine with AXI bus master and private connection to GPMI
- CCM - Clock Control Module
- ECSPI - Enhanced Configurable Serial Peripheral Interface
- EIM - External Interface Module. Used for NOR and OneNAND devices
- GPMI - NAND controller pin interface
- OCOTP_CTRL - On-Chip OTP Controller. The OCOTP contains the eFUSEs.
- IOMUXC - I/O Multiplexer Control allows GPIO use to override eFUSE boot settings
- IOMUXC GPR - I/O Multiplexer Control General Purpose registers
- CAAM - Cryptographic Acceleration and Assurance Module
- QSPI - QuadSPI Flash
- SNVS - Secure Non-Volatile Storage
- SRC - System Reset Controller
- USB - Used for serial download of a boot device provisioning program
- USDHC - Ultra Secure Digital Host Controller
- WDOG-1 - Watchdog Timer

6.6.4.3 Clocks at Boot Time

The table below shows the various clocks and their sources used by ROM.

Table 6-27. PLL Setting by ROM

PLL Name	Frequency	Comment
ARM_PLL	792MHz	
DDR_PLL	1066MHz	
SYS_PLL	480MHz	All PFDs enabled
ENET_PLL	1000MHz	Enabled when a) NAND boot with sync/toggle mode, and b) NAND working frequency is 50MHz/ 100MHz

NOTE

All other PLLs are in default status

Table 6-28. Clock Root Setting by ROM

Clock Name	Frequency(MHz)	Source	Enable	Comment
ARM_A7_CLK_ROOT	792	PLL_ARM	Yes	Frequency will be lower in Low Power Boot (LPB) mode. See "USB Low Power Boot" for detail.
ARM_M4_CLK_ROOT	240	PLL_SYS_MAIN_240M	Yes	
MAIN_AXI_CLK_ROOT	332	PLL_SYS_PFD1_332M	Yes	Frequency lower in LPB mode.
DISP_AXI_CLK_ROOT	332	PLL_SYS_PFD1_332M	No	
NAND_USDHC_BUS_CLK_ROOT	270	PLL_SYS_PFD2_270M	Yes	
AHB_CLK_ROOT	135	PLL_SYS_PFD2_270M	Yes	
DRAM_PHYM_CLK_ROOT	1066	PLL_DDR	Yes	
DRAM_CLK_ROOT	533	PLL_DDR	Yes	Frequency lower in LPB mode.
USB_HSIC_CLK_ROOT	480	PLL_SYS_MAIN_480M	No	
EIM_CLK_ROOT	120	PLL_SYS_MAIN_120M	Yes	
NAND_CLK_ROOT			Yes	See Table "NAND_CLK_ROOT setting"
QSPI_CLK_ROOT			Yes	See Table "QSPI_CLK_ROOT setting"
USDHC1_CLK_ROOT	196	PLL_SYS_PFD0_392M	Yes	
USDHC2_CLK_ROOT	196	PLL_SYS_PFD0_392M	Yes	
USDHC3_CLK_ROOT	196	PLL_SYS_PFD0_392M	Yes	
ECSPI1_CLK_ROOT	60	PLL_SYS_MAIN_240M	Yes	
ECSPI2_CLK_ROOT	60	PLL_SYS_MAIN_240M	Yes	
ECSPI3_CLK_ROOT	60	PLL_SYS_MAIN_240M	Yes	
ECSPI4_CLK_ROOT	60	PLL_SYS_MAIN_240M	Yes	

NOTE

All other clock roots are in default status.

Table 6-29. NAND_CLK_ROOT Setting

NAND data rate	NAND_CLK_ROOT source	Frequency(MHz)
Async/Legacy NAND	PLL_SYS_MAIN_480M	24
Sync 40M	PLL_SYS_MAIN_480M	40
Toggle/Sync 66M	PLL_SYS_PFD0_392M	66
Toggle 80M	PLL_SYS_MAIN_480M	80

Table continues on the next page...

Table 6-29. NAND_CLK_ROOT Setting (continued)

NAND data rate	NAND_CLK_ROOT source	Frequency(MHz)
Sync 100M	PLL_ENET_MAIN_500M	100
Toggle/Sync 133M	PLL_SYS_PFD0_392M	133
Sync 160M	PLL_SYS_MAIN_480M	160
Toggle/Sync 200M	PLL_SYS_PFD0_392M	198

NOTE

NAND_CLK_ROOT source depends on NAND data rate.

Table 6-30. QSPI_CLK_ROOT Setting

QSPI Serial Clock Frequency	QSPI_CLK_ROOT	Frequency(MHz)
20MHz	PLL_SYS_PFD4	80
49MHz	PLL_SYS_PFD4 configured to 392MHz	196
55MHz	PLL_SYS_PFD4 configured to 432MHz	216
60MHz	PLL_SYS_PFD4	240
66MHz	PLL_SYS_PFD2_270M	270
76MHz	PLL_SYS_PFD4 configured to 320MHz	320
99MHz	PLL_SYS_PFD4 configured to 392MHz	392

NOTE

QSPI_CLK_ROOT source depends on QSPI data rate.

NOTE

QSPI_CLK_ROOT is 4x of the QSPI Serial Clock Frequency.

NOTE

"QSPI Serial Clock Frequency" in QSPI_CLK_ROOT mapping table is from QuadSPI Configuration Parameters; please refer to QuadSPI Configuration Parameters table for details. The real clock frequency may be different.

The ROM code will disable the clocks listed in the following table, except for the boot devices listed in the Enabled for Boot Device column below.

Table 6-31. CCGR setting by ROM

CCGR registers	LPCG Enable	Enabled for Boot Device
CCM_CCGR20	rawnand	RawNAND
CCM_CCGR21	qspi	QSPI NOR flash
CCM_CCGR22	eim	EIM NOR

Table continues on the next page...

Table 6-31. CCGR setting by ROM (continued)

CCGR registers	LPCG Enable	Enabled for Boot Device
CCM_CCGR32	adc	
CCM_CCGR39	mu	
CCM_CCGR40	hs	
CCM_CCGR41	dvfs	
CCM_CCGR64	sema1	
CCM_CCGR65	sema2	
CCM_CCGR68	perfmon1	
CCM_CCGR69	perfmon2	
CCM_CCGR72	sdma	
CCM_CCGR73	csi	
CCM_CCGR75	lcdif	
CCM_CCGR100	mipi_csi	
CCM_CCGR101	mipi_dsi	
CCM_CCGR102	mipi_mem_phy	
CCM_CCGR104	usb_ctrl	USB Serial Download mode
CCM_CCGR105	usb_hsic	
CCM_CCGR108	usdhc1	USDHC1
CCM_CCGR109	usdhc2	USDHC2
CCM_CCGR110	usdhc3	USDHC3
CCM_CCGR112	enet1	
CCM_CCGR116	can1	
CCM_CCGR117	can2	
CCM_CCGR120	ecspi1	ecSPI1
CCM_CCGR121	ecspi2	ecSPI2
CCM_CCGR122	ecspi3	ecSPI3
CCM_CCGR123	ecspi4	ecSPI4
CCM_CCGR126	gpt3	
CCM_CCGR127	gpt4	
CCM_CCGR128	ftm1	
CCM_CCGR129	ftm2	
CCM_CCGR132	pwm1	
CCM_CCGR133	pwm2	
CCM_CCGR134	pwm3	
CCM_CCGR135	pwm4	
CCM_CCGR136	i2c1	
CCM_CCGR137	i2c2	
CCM_CCGR138	i2c3	
CCM_CCGR139	i2c4	
CCM_CCGR140	sai1	

Table continues on the next page...

Table 6-31. CCGR setting by ROM (continued)

CCGR registers	LPCG Enable	Enabled for Boot Device
CCM_CCGR141	sai2	
CCM_CCGR142	sai3	
CCM_CCGR144	sim1	
CCM_CCGR145	sim2	
CCM_CCGR148	uart1	
CCM_CCGR149	uart2	
CCM_CCGR150	uart3	
CCM_CCGR151	uart4	
CCM_CCGR152	uart5	
CCM_CCGR153	uart6	
CCM_CCGR154	uart7	
CCM_CCGR157	wdog2	
CCM_CCGR158	wdog3	
CCM_CCGR159	wdog4	
CCM_CCGR170	kpp	

NOTE

All other CCGR registers are in default status

6.6.4.4 Enabling MMU and Caches

The boot ROM includes a feature of enabling the Memory Management Unit (MMU) and caches to improve boot speed.

L1 instruction cache is enabled at the start of image download. L1 data cache, L2 cache and MMU are enabled during image authentication. Once HAB authentication completes the ROM disables the L1 data cache, L2 cache and MMU.

L1 Instruction cache, L1 data cahce, L2 cache and MMU is controlled by eFuse. By default these features are enabled.

Enabling the MMU when booting non-securely with SEC_CONFIG=Open, and setting the CSF pointer in the Image Vector Table to NULL, has no impact on the boot performance. With this configuration it is recommended to blow BT_MMU_DISABLE fuse.

6.6.4.5 Exception Handling

The exception vectors located at the start of ROM are used to map all the ARM exceptions (except the reset exception) to a duplicate exception vector table in internal RAM.

During the boot phase of CPU0, the RAM vectors point to the serial downloader in ROM.

After boot the program image can overwrite the vectors as required. The code shown below is used to map the ROM exception vector table to the duplicate one in RAM.

Mapping ROM Exception Vector Table

```
;; Define linker area for ROM exception vector table
AREA IROM_VECTORS, CODE, READONLY
LDR     PC, Reset_Addr
LDR     PC, Undefined_Addr
LDR     PC, SWI_Addr
LDR     PC, Prefetch_Addr
LDR     PC, Abort_Addr
NOP                          ; Reserved vector
LDR     PC, IRQ_Addr
LDR     PC, FIQ_Addr

;; Define exception vector table
Reset_Addr      DCD     start_address
Undefined_Addr  DCD     iRAM_Undefined_Handler
SWI_Addr        DCD     iRAM_SWI_Handler
Prefetch_Addr   DCD     iRAM_Prefetch_Handler
Abort_Addr      DCD     iRAM_Abort_Handler
                DCD     0               ; Reserved vector
IRQ_Addr        DCD     iRAM_IRQ_Handler
FIQ_Addr        DCD     iRAM_FIQ_Handler

start_address DCD start ;reset handler vector
```

6.6.4.6 Interrupt Handling During Boot

No special interrupt handling routines are required during the boot process. Interrupts are disabled during boot ROM execution and may be enabled in a later boot stage.

6.6.4.7 Persistent Bits

Some modes of boot ROM require registers that keep their values after warm reset. SRC General Purpose registers are used for this purpose.

See the table below for persistent bits list and description.

Table 6-32. Persistent Bits

Bit Name	Bit Location	Description
PERSIST_SECONDARY_BOOT	SRC_GPR10[30]	This bit identifies which image must be used - primary and secondary. Used only for boot modes that support redundant boot.
PERSIST_BLOCK_REWRITE	SRC_GPR10[29]	This bit is used as warning. It identifies that there are errors in NAND blocks that hold the application image. See NAND Flash for more details.
PERSISTENT_ENTRY0[31:0]	SRC_GPR1[31:0]	Holds entry function for CPU0 for waking-up from low power mode.
PERSISTENT_ARG0[31:0]	SRC_GPR2[31:0]	Holds argument of entry function for CPU0 for waking-up from low power mode.
PERSISTENT_ENTRY1[31:0]	SRC_GPR3[31:0]	Holds entry function for CPU1 for wake-up from low-power mode.
PERSISTENT_ARG1[31:0]	SRC_GPR4[31:0]	Holds argument of entry function for CPU1 for wake-up from low-power mode.

6.6.5 Boot Devices (Internal Boot)

The Chip supports the following boot Flash devices:

- NOR Flash with External Interface Module (EIM), located on CS0, 16-bit bus width
- OneNAND Flash with EIM interface, located on CS0, 16-bits bus width
- Raw NAND (MLC and SLC), and Toggle-mode NAND flash through GPMI-2 interface, located at CS0. Page sizes of 2 Kbyte, 4 Kbyte and 8 Kbyte. Bus widths of 8-bit with 2 through 62-bit BCH Hardware ECC (Error Correction) are supported.
- Quad SPI Flash
- SD/MMC/eSD/SDXC/eMMC4.4 via USDHC interface, supporting high capacity cards
- EEPROM boot via SPI (serial flash)

The selection of external boot device type is controlled by BOOT_CFG[15:12] eFUSEs. See the table below for more details.

Table 6-33. Boot Device Selection

BOOT_CFG[15:12]	Boot Device
0001	SD/eSD/SDXC
0010	MMC/eMMC
0011	Raw NAND

Table continues on the next page...

Table 6-33. Boot Device Selection (continued)

BOOT_CFG[15:12]	Boot Device
0100	QSPI
0101	NOR/OneNAND (EIM)
0110	Serial ROM (I2C/SPI)

6.6.5.1 NOR Flash/OneNAND using EIM Interface

The External Interface Module (EIM) works in the asynchronous mode, and supports either muxed, Address/Data, or non-muxed schemes based on fuse settings.

Table 6-34. EIM Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
BOOT_CFG[15:12]	OEM	Boot Device Selection	Yes	0000	0101 - Boot from EIM Interface
BOOT_CFG[11]	OEM	NOR/OneNAND Selection	Yes	0	0 - NOR 1 - OneNAND
BOOT_CFG[10:9]	OEM	Muxing Scheme	Yes	00	00 - Muxed, 16-bit data (low half) interface 01 - Reserved 10 - Not muxed, 16-bit data (low half) interface 11 - Reserved
BOOT_CFG[7:6]	OEM	OneNAND Page Size	Yes	00	00 - 1K 01 - 2K 10 - 4K 11 - Reserved

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See GPIO Boot Overrides for corresponding GPIO pin.

6.6.5.1.1 NOR Flash Boot Operation

Booting from the NOR Flash is supported via EIM interface. The ROM reads Image Vector Table and Boot Data structures to determine if the image can be executed directly from EIM address space or should be copied to other memory.

The start field of Boot Data Structure specifies the final location of the image (see Image Vector Table and Boot Data).

6.6.5.1.2 OneNAND Flash Boot Operation

At system power-up, the OneNAND device automatically copies an Initial Load Region of 1 Kbyte from the start of the flash array (sector 0 and sector 1, page 0, block 0) to its Boot RAM (OneNAND's internal RAM).

NOTE

The OneNAND boot RAM memory containing the Initial 1K Load Region must contain the IVT, DCD and the Boot Data structures.

Next, the ROM processes the DCD and then proceeds to copy the Program Image contents to the application destination pointer (located in the start entry of Boot Data (see Image Vector Table and Boot Data). The ROM determines the size of the Program Image by the length specified by size entry in Boot Data structure (see Image Vector Table and Boot Data). A failure loading data from the OneNAND device for any reason forces the Chip to enter the Serial Downloader, otherwise the booting from the OneNAND device continues.

The figure below illustrates the layout of the Program Image on a OneNAND boot device.

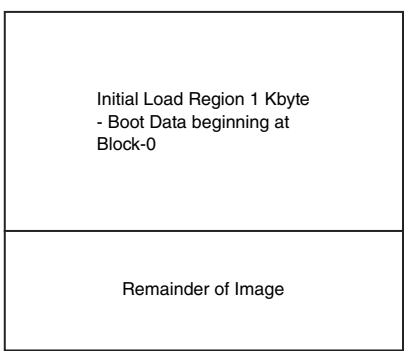


Figure 6-23. Program Image Layout on a OneNAND Flash Device

Prior to accessing the OneNAND device, the Chip waits approximately 500 µs after Power On Reset. This delay is required for the OneNAND device to become ready. After this initial 500 µs delay it can take an addition 70 µs for the OneNAND device to load the Initial Load Region of 1 Kbyte into its boot RAM. The Chip polls the OneNAND device Interrupt Status Register to confirm that the first 1 Kbytes has been loaded to the OneNAND boot RAM before continuing with the boot flow.

6.6.5.1.3 IOMUX Configuration for EIM Devices

The EIM interface uses dedicated contacts on the IC.

The contacts assigned to the data signals used by EIM are shown in the table below.

Table 6-35. EIM IOMUX Pin Configuration

Signal	A/D16 (Muxed, 16-bit data interface)	A+D (Not muxed, 16-bit data interface)
DATA0	EPDC_D0.alt4	LCD_DAT0.alt4
DATA1	EPDC_D1.alt4	LCD_DAT1.alt4
DATA2	EPDC_D2.alt4	LCD_DAT2.alt4
DATA3	EPDC_D3.alt4	LCD_DAT3.alt4
DATA4	EPDC_D4.alt4	LCD_DAT4.alt4
DATA5	EPDC_D5.alt4	LCD_DAT5.alt4
DATA6	EPDC_D6.alt4	LCD_DAT6.alt4
DATA7	EPDC_D7.alt4	LCD_DAT7.alt4
DATA8	EPDC_BDR1.alt4	LCD_DAT8.alt4
DATA9	EPDC_PWRCOM.alt4	LCD_DAT9.alt4
DATA10	EPDC_SDCLK.alt4	LCD_DAT10.alt4
DATA11	EPDC_SDLE.alt4	LCD_DAT11.alt4
DATA12	EPDC_SDOE.alt4	LCD_DAT12.alt4
DATA13	EPDC_SDSHR.alt4	LCD_DAT13.alt4
DATA14	EPDC_SDCE0.alt4	LCD_DAT14.alt4
DATA15	EPDC_SDCE1.alt4	LCD_DAT15.alt4
ADDR0	EPDC_D0.alt4	
ADDR1	EPDC_D1.alt4	
ADDR2	EPDC_D2.alt4	
ADDR3	EPDC_D3.alt4	
ADDR4	EPDC_D4.alt4	
ADDR5	EPDC_D5.alt4	
ADDR6	EPDC_D6.alt4	
ADDR7	EPDC_D7.alt4	
ADDR8	EPDC_BDR1.alt4	
ADDR9	EPDC_PWRCOM.alt4	
ADDR10	EPDC_SDCLK.alt4	
ADDR11	EPDC_SDLE.alt4	
ADDR12	EPDC_SDOE.alt4	
ADDR13	EPDC_SDSHR.alt4	
ADDR14	EPDC_SDCE0.alt4	
ADDR15	EPDC_SDCE1.alt4	
ADDR16	EPDC_SDCE2.alt4	
ADDR17	EPDC_SDCE3.alt4	
ADDR18	EPDC_GDCLK.alt4	
ADDR19	EPDC_GDOE.alt4	
ADDR20	EPDC_GDRL.alt4	
ADDR21	EPDC_GDSP.alt4	

Table continues on the next page...

Table 6-35. EIM IOMUX Pin Configuration (continued)

Signal	A/D16 (Muxed, 16-bit data interface)	A+D (Not muxed, 16-bit data interface)
ADDR22	EPDC_BDR0.alt4	
ADDR23	LCD_DAT20.alt4	
ADDR24	LCD_DAT21.alt4	
ADDR25	LCD_DAT22.alt4	
ADDR26	LCD_DAT23.alt4	

6.6.5.2 NAND Flash

The boot ROM supports a number of MLC/SLC NAND Flash devices from different vendors and LBA NAND Flash devices. The Error Correction and Control (ECC) subblock (BCH) is used to detect the errors.

6.6.5.2.1 NAND eFUSE Configuration

The boot ROM determines the configuration of external NAND flash by parameters, either provided by eFUSE, or sampled on GPIO pins, during boot. See Table 6-36 for parameters details.

NOTE

BOOT_CFGx sampled on GPIO pins depends on BT_FUSE_SEL setting. See Boot Fusemap for details.

NOTE

ROM always boots from CS0_B, while for multiple chip selects (such as some NAND chips), all other chip select pins of such NAND chip except CS0_B, should be pull-up in boot progress

Table 6-36. NAND Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
BOOT_CFG[15:12]	OEM	Boot Device Selection	Yes	0	0011 - Boot from NAND Interface
BOOT_CFG[7]	OEM	BT_TOGGLEMODE	Yes	0	0 - raw NAND 1 - toggle mode NAND
BOOT_CFG[11:10]	OEM	Pages In Block	Yes	0	00 - 128 01- 64 10- 32 11- 256
BOOT_CFG[9:8]	OEM	Row Address Cycles	Yes	00	00 - 3

Table continues on the next page...

Table 6-36. NAND Boot eFUSE Descriptions (continued)

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
					01 - 2 10 - 4 11 - 5
BOOT_CFG[4:1]	OEM	Toggle Mode 33 MHz Preamble Delay, Read Latency	Yes	000	0000 - 16 GPMICLK cycles 0001 - 1 GPMICLK cycles 0010 - 2 GPMICLK cycles 0011 - 3 GPMICLK cycles 0100 - 4 GPMICLK cycles 0101 - 5 GPMICLK cycles 0110 - 6 GPMICLK cycles 0111 - 7 GPMICLK cycles 1000 - 8 GPMICLK cycles 1001 - 9 GPMICLK cycles 1010 - 10 GPMICLK cycles 1011 - 11 GPMICLK cycles 1100 - 12 GPMICLK cycles 1101 - 13 GPMICLK cycles 1110 - 14 GPMICLK cycles 1111 - 15 GPMICLK cycles
BOOT_CFG[6:5]	OEM	Boot Search Count	Yes	00	00 - 2 01 - 2 10 - 4 11 - 8
0x4B0[7]	OEM	Override Pad Settings	No	0	Override NAND Pad Settings 0 - Use default values 1 - Use PAD_SETTINGS value
0x4A0[31:24]	OEM	PAD_SETTINGS[7:0]	No	0	NAND Pad Settings Value
0x4B0[15:12]	OEM	READ_RETRY_SEQ_ID[3:0]	No	0000	0000 - Don't use ROM embedded read-retry sequence 0001 - use Micron 20nm read-retry sequence 0010 - use Toshiba A19nm read-retry sequence 0011 - use Toshiba 19nm read-retry sequence 0100 - use SanDisk 19nm read-retry sequence

Table 6-36. NAND Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
					0101 - use SanDisk 1ynm read-retry sequence 0110 to 1111 - Reserved

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See Table 1 for corresponding GPIO pin.

6.6.5.2.2 NAND Flash Boot Flow and Boot Control Blocks (BCB)

There are two BCB data structures:

- FCB
- DBBT

As part of the NAND media initialization, the ROM driver uses safe NAND timings to search for a Firmware Configuration Block (FCB) that contains the optimum NAND timings, page address of Discovered Bad Block Table (DBBT) Search Area and start page address of primary and secondary firmware.

The hardware ECC level to use is embedded inside FCB block. The FCB data structure is also protected using ECC. Driver reads raw 2112 bytes of first sector and runs through software ECC engine that determines whether FCB data is valid or not.

If the FCB is found, the optimum NAND timings are loaded for further reads. If the ECC fails, or the fingerprints do not match, the Block Search state machine increments page number to Search Stride number of pages to read for the next BCB until SearchCount pages have been read.

If search fails to find a valid FCB, the NAND driver responds with an error and the boot ROM enters into serial download mode.

The FCB contains the page address of DBBT Search Area, and the page address for primary and secondary boot images. DBBT is searched in DBBT Search Area just like how FCB is searched. After the FCB is read, the DBBT is loaded, and the primary or secondary boot image is loaded using starting page address from FCB.

Figure 6-24 shows the state diagram of FCB search.

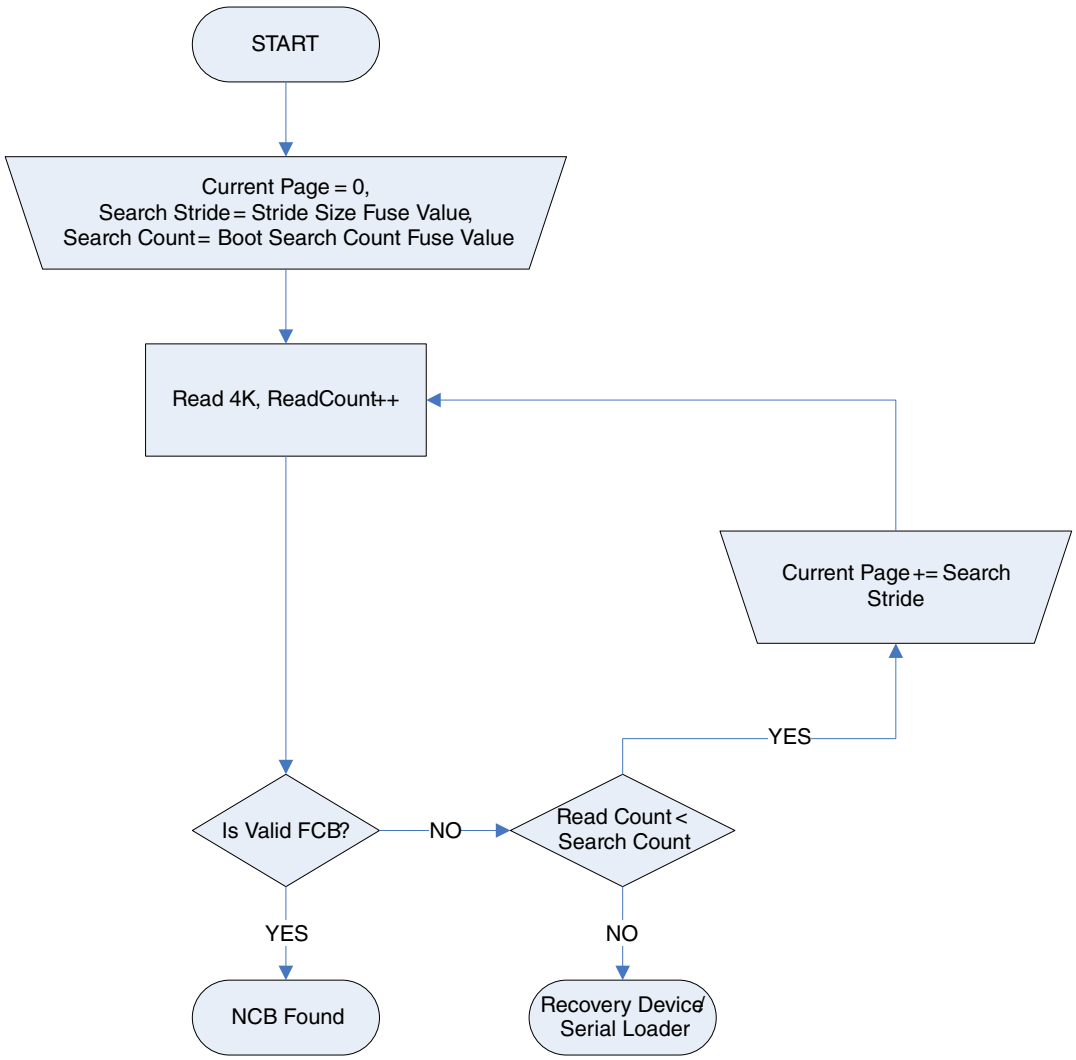


Figure 6-24. FCB Search Flow

Once FCB is found, the boot ROM searches for the Discovered Bad Blocks Table (DBBT). If DBBT Search Area is 0 in FCB, then ROM assumes that there are no bad blocks on NAND device boot area. See Figure 6-25 for the DBBT search flow.

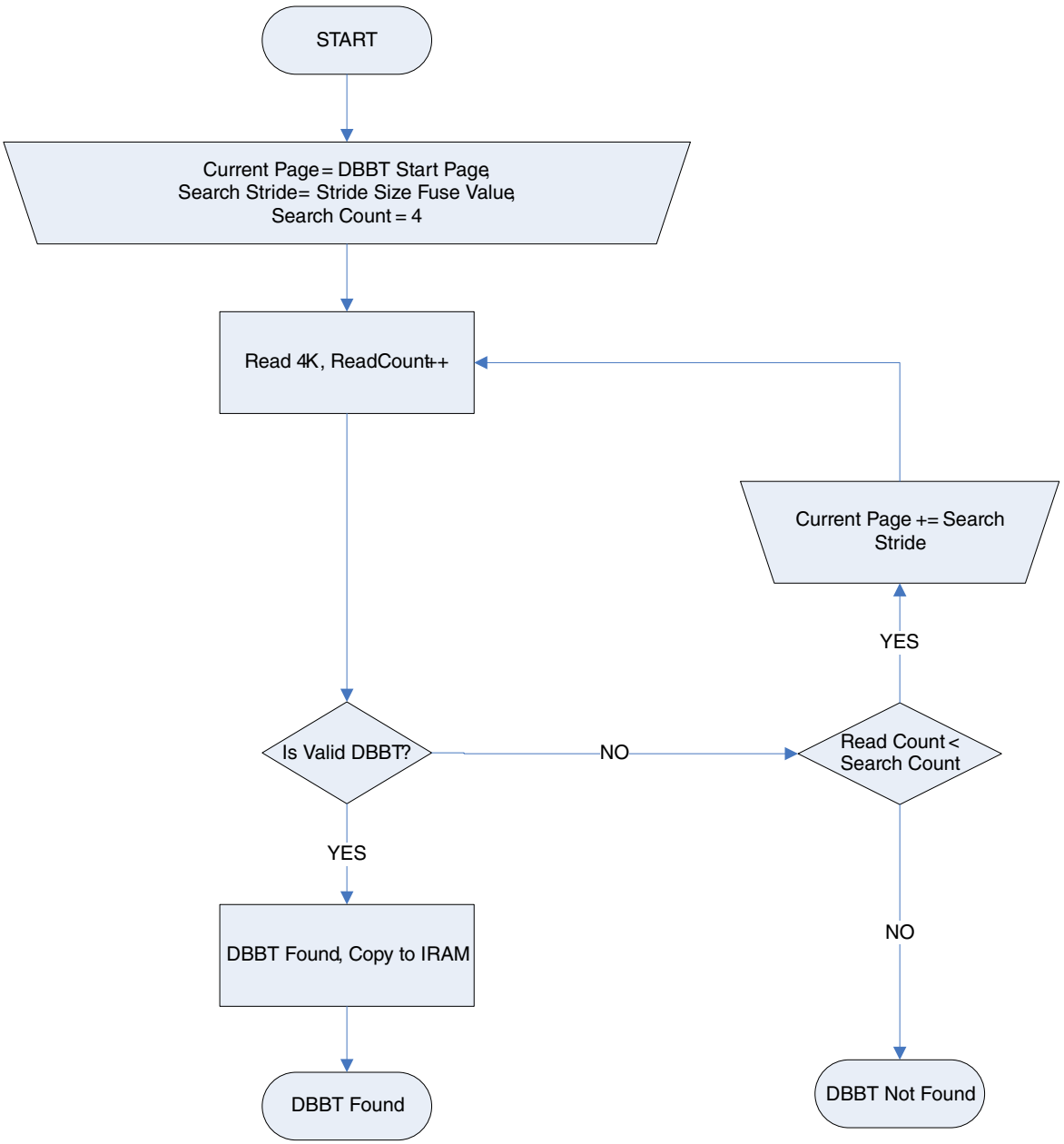


Figure 6-25. DBBT Search Flow

The BCB search and load function also monitors the ECC correction threshold and sets the PERSIST_BLOCK_REWRITE persistent bit if the threshold exceeds the maximum ECC correction ability.

If during primary image read there is a page with a number of errors higher than ECC can correct, the boot ROM will turn on PERSIST_SECONDARY_BOOT bit and perform SW reset (After SW reset, secondary image is used).

If during secondary image read there is a page with number of errors higher than ECC can correct, the boot ROM goes to serial loader.

6.6.5.2.3 Firmware Configuration Block

The FCB is the first sector in the first good block. The FCB should be present at each search stride of the search area.

The search area contains copies of the FCB at each stride distance, so in case the first NAND block becomes corrupted, the ROM will find its copy in the next NAND block. The search area should span over at least two NAND blocks. The location information for DBBT search area, FW1, and FW2 are all specified in the FCB. Table 6-37 shows the Flash Control Block Structure.

Table 6-37. Flash Control Block Structure

Name	Start Byte	Size in Bytes	Description
Reserved	0	4	Reserved for Fingerprint #1(Checksum)
FingerPrint	4	4	32 bit word with a value of 0x46434220, in ascii "FCB"
Version	8	4	32-bit version number; this version of FCB is 0x00000001
m_NANDTiming	12	8	8 bytes of data for 8 NAND Timing Parameters from NAND datasheet. The 8 parameters are: m_NandTiming[0]=data_setup, m_NandTiming[1]=data_hold, m_NandTiming[2]=address_setup, m_NandTiming[3]=dsample_time, m_NandTiming[4]=nand_timing_state, m_NandTiming[5]=REA, m_NandTiming[6]=RLOH, m_NandTiming[7]=RHOH. ROM only uses first 4 parameters but FCB provides space for other 4 parameters to be used by boot-loader or other applications.
PageDataSize	20	4	Number of bytes of data in a page. Typically, this is 2048 bytes for 2112 bytes page size or 4096 bytes for 4314/4224 bytes page size or 8192 for 8568 bytes page size
TotalPageSize	24	4	Total number of bytes in page. Typically, 2112 for 2 KB page or 4224 or 4314 for 4 KB page or 8568 for 8 KB page.
SectorsPerBlock	28	4	Number of pages per block. Typically 64 or 128 or depending on NAND device type.
NumberOfNANDs	32	4	Not used by ROM

Table continues on the next page...

Table 6-37. Flash Control Block Structure (continued)

Name	Start Byte	Size in Bytes	Description
TotalInternalDie	36	4	Not used by ROM
CellType	40	4	Not used by ROM
EccBlockNEccType	44	4	Value from 0 to 31 used to set BCH Error Corrrection level 0, 2, 4, .. or 62 for Block BN of ECC page, used in configuring BCH62 page layout registers
EccBlock0Size	48	4	Size of block B0, used in configuring BCH62 page layout registers
EccBlockNSize	52	4	Size of block BN, used in configuring BCH62 page layout registers
EccBlock0EccType	56	4	Value from 0 to 31 used to set BCH Error Corrrection level 0, 2, 4, .. or 62 for Block BN of ECC page, used in configuring BCH62 page layout registers
MetadataBytes	60	4	Size of metadata bytes used in configuring BCH62 page layout registers
NumEccBlocksPerPage	64	4	Number of ECC blocks BN not including B0. This value is used in configuring BCH62 page layout registers
EccBlockNEccLevelSDK	68	4	Not used by ROM
EccBlock0SizeSDK	72	4	Not used by ROM
EccBlockNSizeSDK	76	4	Not used by ROM
EccBlock0EccLevelSDK	80	4	Not used by ROM
NumEccBlocksPerPageSDK	84	4	Not used by ROM
MetadataBytesSDK	88	4	Not used by ROM
EraseThreshold	92	4	Not used by ROM
Firmware1_startingPage	104	4	Page number address where first copy of bootable firmware is located
Firmware2_startingPage	108	4	Page number address where second copy of bootable firmware is located
PagesInFirmware1	112	4	Size of first copy of firmware in pages
PagesInFirmware2	116	4	Size of second copy of firmware in pages
DBBTSearchAreaStartAddress	120	4	Page address for bad block table search area
BadBlockMarkerByte	124	4	This is an input offset in BCH page for ROM to swap with first byte of metadata after reading a page using BCH62. ROM supports restoration of manufacturer marked bad block markers in the page and this offset is the bad block marker offset location
BadBlockMarkerStartBit	128	4	This is an input bit offset in BadBlockMarkerByte for ROM to use when swapping 8 bits with fiirst byte of metadata.
BBMarkerPhysicalOffset	132	4	This is the offset where manufacturer leaves bad block marker on a page

Table continues on the next page...

Table 6-37. Flash Control Block Structure (continued)

Name	Start Byte	Size in Bytes	Description
BCHType	136	4	0 for BCH20 and 1 for BCH62. The Chip is backward compatible to BCH20 and this field tell ROM to use BCH20 or BCH62 block
TMTiming2_ReadLatency	140	4	Toggle mode NAND timing parameter read latency, ROM use this value to configure timing2 register of GPMI
TMTiming2_PreambleDelay	144	4	Toggle mode NAND timing parameter Preamble Delay. ROM use this value to configure timing2 register of GPMI
TMTiming2_CEDelay	148	4	Toggle mode NAND timing parameter CE Delay. ROM use this value to configure timing2 register of GPMI
TMTiming2_PostambleDelay	152	4	Toggle mode NAND timing parameter Postamble Delay. ROM use this value to configure timing2 register of GPMI
TMTiming2_CmdAddPause	156	4	Toggle mode NAND timing parameter Cmd Add Pause. ROM use this value to configure timing2 register of GPMI
TMTiming2_DataPause	160	4	Toggle mode NAND timing parameter Data Pause. ROM use this value to configure timing2 register of GPMI
TMSpeed	164	4	This is the toggle mode speed for ROM to configure gpmi clock. 0 for 33 MHz, 1 for 40 MHz and 2 for 66 MHz
TMTiming1_BusyTimeout	168	4	Toggle mode NAND timing parameter Busy Timeout. ROM use this value to configure timing1 register of GPMI
DISBBM	172	4	If 0 ROM will swap BadBlockMarkerByte with metadata[0] after reading a page using BCH62. If the value set is 1 then ROM will not do swapping
BBMark_spare_offset	176	4	The offset in mata data place which stores the data in Bad block marker place.
Onfi_sync_enable	180	4	Enable the Onfi nand sync mode support
Onfi_sync_speed	184	4	Speed for onfi nand sync mode: 0 - 24 MHZ, 1 - 33 MHZ, 2 - 40 MHZ, 3 - 50 MHZ, 4 - 66 MHZ, 5 - 80 MHZ, 6 - 100 MHZ, 7 - 133 MHZ, 8 - 160 MHZ, 9 - 200 MHZ
Onfi_syncNANDData	188	28	parameters for onfi nand sync mode timing. They are read latency, ce_delay, preamble_delay, postamble_delay, cmdadd_pause, data_pause, busy_timeout
DISBB_Search	216	4	Disable the badblock search function when reading the firmware, only using DBBT.
Randomizer_Enable	220	4	Enable randomizer support with randomizer type 2. This is for DBBT and firmware.
Reserved	224	60	Reserved for future use

Table continues on the next page...

Table 6-37. Flash Control Block Structure (continued)

Name	Start Byte	Size in Bytes	Description
Read_Retry_Enable	284	4	Enable read retry for DBBT and firmware
Reserved	288	4	Reserved and must be kept as 0

The FCB data structure is protected using 62-bit ECC. The layout of FCB page is illustrated in the figure below.

Figure 6-26. Layout of the FCB Page

The detailed parameters of FCB pages are listed in the table below.

Table 6-38. Parameters Setting for FCB Page

Parameter	Value
TotalPageSize	2048+64=2112
MetadataBytes	32
EccBlock0Size	128
EccBlock0EccType	31
BCHType	0
EccBlockNSize	128
EccBlockNEccType	31
NumEccBlocksPerPage	7

In order to reduce the disturbances caused by a neighboring cell in the FCB page in the NAND chip, a randomizer is enabled when reading FCB page. The randomizer is used to reduce bit errors in FCB. Ensure the randomizer is enabled when burning FCB pages in NAND flash. To control randomizer for the pages except for FCB, a new field called Randomizer_Enable is added into FCB structure. If the Randomizer_Enable field is set as 0, randomizer will be disabled when reading the pages except FCB, being set as a non-zero value will enable randomizer. For detailed randomizer information, please see Randomizer.

6.6.5.2.4 Discovered Bad Block Table (DBBT)

See the table below for DBBT format.

Table 6-39. DBBT Structure

Name	Start Byte	Size in Bytes	Description
reserved	0	4	-
FingerPrint	4	4	32-bit word with a value of 0x44424254,in ascii "DBBT"
Version	8	4	32-bit version number; this version of DBBT is 0x00000001
reserved	12	4	-
DBBT_NUM_OF_PAGES	16	4	Size of DBBT in pages
reserved	20	4*PageSize-20	-
reserved	4*PageSize	4	-
Number of Entries	4*PageSize + 4	4	Number of bad blocks
Bad Block Number	4*PageSize + 8	4	First bad block number
Bad Block Number	4*PageSize + 12	4	Second bad block number
...-	-	-	...next bad block number
...-	-	-	...-
Last bad block number	-	-	last bad block number

6.6.5.2.5 Bad Block Handling in the ROM

During firmware boot, at the block boundary, the Bad Block table is searched for a match to the next block.

If no match is found, the next block can be loaded. If a match is found, the block must be skipped and the next block checked.

If Bad Block table start page is null, check the manufactory made Bad Block marker. The location of Bad Block maker is at the first 3 or last 3 pages in every block of the NAND flash. NAND manufacturers normally use one byte in the spare area of certain pages within a block to mark a block is bad or not. 0xFF means good block, non FF means bad block.

In order to preserve the BI (bad block information), flash updater or gang programmer applications need to swap Bad Block Information (BI) data to byte 0 of metadata area for every page before programming NAND flash. ROM when loading firmware copies back the value at metadata[0] to BI offset in page data. The Figure 6-27 shows how the factory bad block marker is preserved.

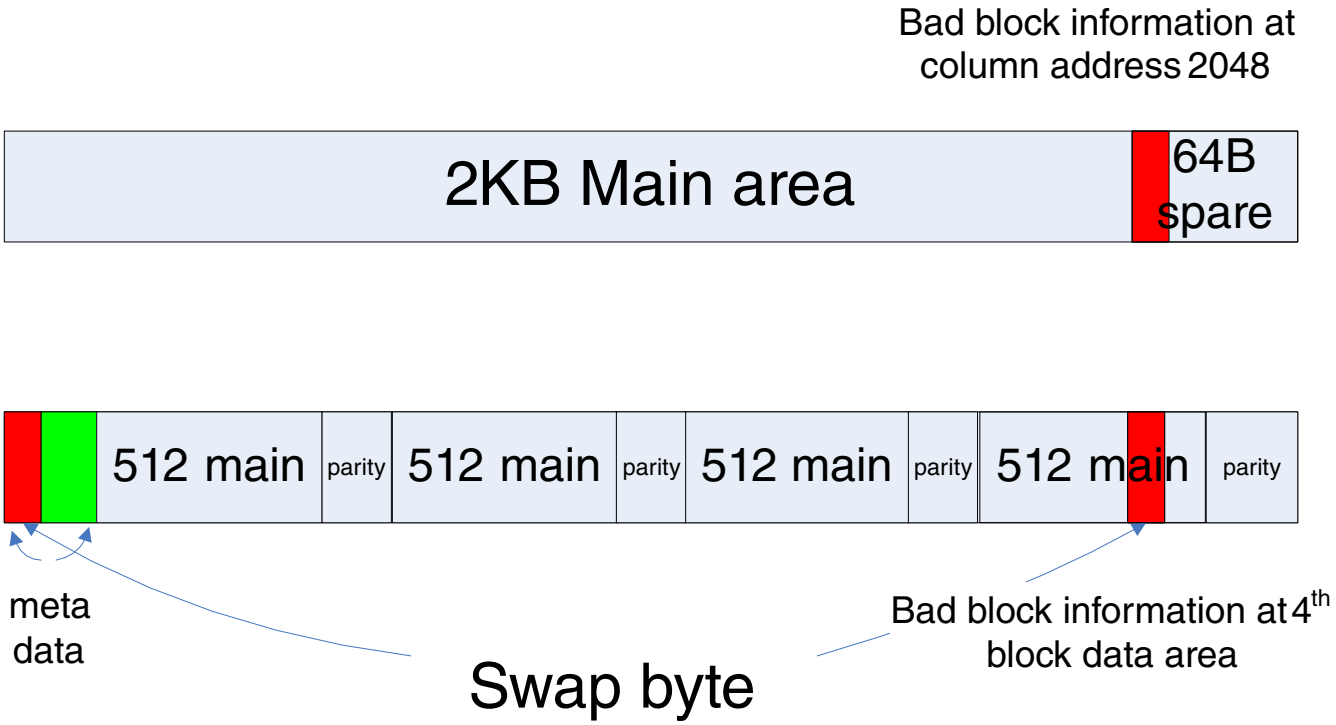


Figure 6-27. Factory Bad Block Marker Preservation

In the FCB structure, there are two elements m_u32BadBlockMarkerByte and m_u32BadBlockMarkerStartBit to indicate the byte and bit place in the page data, that manufacturer marked the bad block marker.

6.6.5.2.6 Read Retry Handling in the ROM

Read retry is used to recover from NAND bit errors beyond the ECC correction threshold. If reading a page failed and the read_retry_enable field in FCB is set as 1, ROM will issue read retry command sequence to shift the read level before reading the page again. If the previous reading failed, ROM continues to shift the read level until reading succeeds or all levels have been tried. The state diagram of read retry is shown in the figure below.

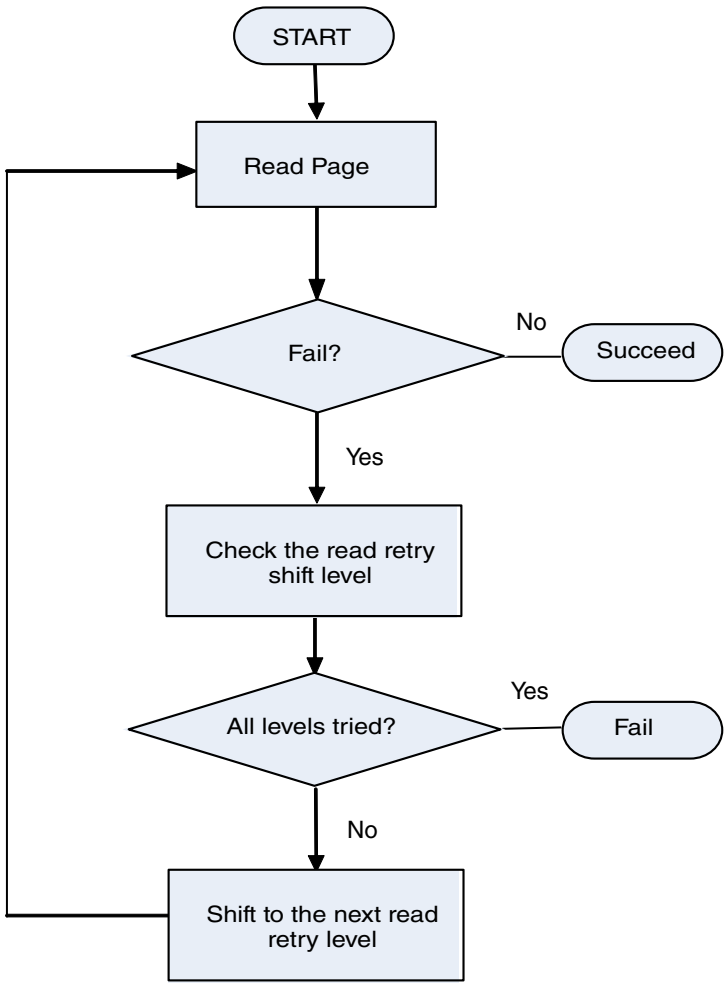


Figure 6-28. Read Retry Flow

Different vendors and different processes may have different read retry sequences. At present, ROM supports five (5) read retry sequences. Blowing the fuse READ_RETRY_SEQ_ID[3:0] can aid in determining which sequence to use.

The read retry sequences are listed in the table below.

Table 6-40. Read retry sequences

Vendor	Process	READ_RETRY_SEQ_ID[3:0]	Comment
Micron	20nm	2'b0001	The detail of this RR sequence is documented in "64Gb, 128Gb, 256Gb, 512Gb Asynchronous/ Synchronous NAND Features(Release: 4/20/12)", please contact Micron for this doc.

Table continues on the next page...

Table 6-40. Read retry sequences (continued)

Vendor	Process	READ_RETRY_SEQ_ID[3:0]	Comment
Toshiba	A19nm	2'b0010	The detail of this RR sequence is documented in "TOSHIBA Technical Information A19nm MLC NAND Retry Read Sequence", please contact Toshiba for this doc.
	19nm	2'b0011	The detail of this RR sequence is documented in "TOSHIBA Technical Information 19nm MLC NAND Read Retry Sequence Rev1.6", please contact Toshiba for this doc.
SanDisk	19nm	2'b0100	The detail of this RR sequence is documented in "App Note 023 (v1.0) 19nm eX2 ABL Dynamic Read Sequence & Parameter Table", please contact SanDisk for this doc.
	1ynm	2'b0101	The detail of this RR sequence is documented in "Application Note 1y_023 19nm eX2 ABL Dynamic Read Sequence & Parameter Table", please contact SanDisk for this doc.

6.6.5.2.7 Toggle Mode DDR NAND Boot

If BT_TOGGLEMODE efuse is blown then ROM does the following to boot from Samsung's toggle mode DDR NAND.

6.6.5.2.7.1 GPMI and BCH Clocks Configuration

ROM sets the clock source and the dividers in CCM registers.

If BOOT_CFG[7] is set(toggle mode), GPMI/BCH CLK source is PLL2PFD4, and running at 66 MHZ, otherwise GPMI/ BCH CLK souce is PLL3, running at 24 MHz. ROM sets default values to timing0, timing1 and timing2 gpmi registers for 24 MHz clock speed. It uses fuse BOOT_CFG[4:1] to configure GPMI timing2 register parameters preamble delay and read latency, the default value for these parameters is 2 when fuses are not blown.

Default timing parameter values used by ROM for toggle-mode device:

- Timing0.ADDRESS_SETUP = 5
- Timing0.DATA_SETUP = 10
- Timing0.DATA_HOLD = 10
- Timing1.DEVICE_BUSY_TIMEOUT = 0 x 500
- Timing2.READ_LATENCY = BOOT_CFG[4:1] if blown, otherwise 2
- Timing2.CE_DELAY = 2
- Timing2.PREAMBLE_DELAY = BOOT_CFG[4:1] if blown, otherwise 2

- Timing2.POSTAMBLE_DELAY = 3
- Timing2.CMDADD_PAUSE = 4
- Timing2.DATA_PAUSE = 6

Default timing parameters can be overriden by TMTiming2_ReadLatency, TMTiming2_PreambleDelay, TMTiming2_CEDelay, TMTiming2_PostambleDelay, TMTiming2_CmdAddPause, TMTiming2_DataPause parameters of FCB.

6.6.5.2.7.2 Setup DMA for DDR Transfers

In DMA descriptors GPMI is configured to read page data at double data rate, the word length is set to 16 and transfer count to half of page size.

6.6.5.2.7.3 Reconfigure Timing and Speed Using Values in FCB

After reading FCB page with GPMI set to default timings and speed 33 MHz, ROM reconfigures CCM dividers to run gpmi/bch clks to desired speed specified in FCB for rest of boot process. The GPMI timing registers are also reconfigured to values specified in FCB.

The GPMI speed can be configured using FCB parameter TMSpeed:

- 0 - 24 MHZ
- 1 - 33 MHZ
- 2 - 40 MHZ
- 3 - 50 MHz
- 4 - 66 MHz
- 5 - 80 MHz
- 6 - 100 MHz
- 7 - 133 MHz
- 8 - 160 MHz
- 9 - 200 MHz

The GPMI timing0 register fields data_setup, data_hold and address_setup are set to values specified for data_setup, data_hold and address_setup in FCB member m_NANDTiming.

The GPMI timing1.DEVICE_BUSY_TIMEOUT is set to value specified in FCB member TMTiming1_BusyTimeout.

The GPMI timing2 register values are set using FCB members TMTiming2.READ_LATENCY, CE_DELAY, PREAMBLE_DELAY, POSTAMBLE_DELAY, CMDADD_PAUSE and DATA_PAUSE.

6.6.5.2.8 Typical NAND Page Organization

6.6.5.2.8.1 BCH ECC Page Organization

The first data block is called block 0 and the rest of the blocks are called block N. Separate ECC level scan be used for block 0 and block N.

The metadata bytes should be located at the beginning of a page, starting at byte 0, followed by data block 0, followed by ECC bytes for data block 0, followed by block 1 and its ECC bytes, and so on until N data blocks. The ECC level for block 0 can be different from the ECC level of rest of the blocks.

For NAND boot, with page size restrictions and data block size restricted to 512 bytes, only few combinations of ECC for block 0 and block N are possible.

The figure below shows the valid layout for 2112 byte sized page.

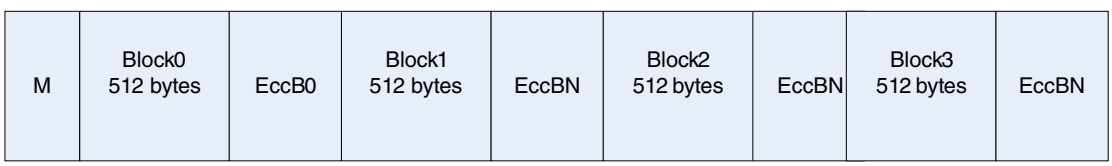


Figure 6-29. Valid Layout for 2112 bytes Sized Page

The example below is for 13 bits of parity(GF13). The number of ECC bits required for a data block is calculated using (ECC_Correction_Level * 13) bits.

In the above layout the ECC size for EccB0 and EccBN should be selected to not exceed a total page size of 2112 bytes. EccB0 and EccBN can be one of 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 bits ECC correction level. The total bytes would then be:

[M + (data_block_size x 4) + ([EccB0 + (EccBN x 3)] x 13) / 8] <= 2112;

M = metadata bytes and data_block_size is 512.

There are 4 data blocks of 512 bytes each in a page of 2k page sized NAND. The values of EccB0 and EccBN should be such that the above calculation would not result in a value greater than 2112 bytes.

M	Block0 512 bytes	EccB0	Block1 512 bytes	EccBN	Block2 512 bytes	EccBN	Block3 512 bytes	EccBN
	Block4 512 bytes	EccBN	Block5 512 bytes	EccBN	Block6 512 bytes	EccBN	Block7 512 bytes	EccBN

Figure 6-30. Valid Layout for 4 Kbytes Sized Page

Different NAND manufacturers have different sizes for a 4K page; 4314 bytes is typical.

[M + (data_block_size x 8) + ([EccB0 + (EccBN x 7)] x 13) / 8] <= 4314;

M= metadata bytes and data_block_size is 512.

There are 8 data blocks of 512 bytes each in a page of a 4k page sized NAND. The values of EccB0 and EccBN should be such that above calculation should not result in a value greater than the size of a page in a 4k page NAND.

6.6.5.2.8.2 Metadata

The number of bytes used for metadata is specified in FCB. Metadata for BCH encoded pages will be placed at the beginning of a page. ROM only cares about the first byte of metadata to swap it with bad block marker byte in page data after each page read; it is important to have at least one byte for the metadata bytes field in FCB data structure.

6.6.5.2.9 IOMUX Configuration for NAND

The table below shows the RawNAND IOMUX pin configuration.

Table 6-41. NAND IOMUX Pin Configuration

Signal	Pad Name
NAND_CLE	SD3_CLK.alt1
NAND_ALE	SD3_CMD.alt1
NAND_WP_B	SAI1_MCLK.alt1
NAND_RE_B	SD3_STROBE.alt1
NAND_WE_B	SD3_RESET_B.alt1
NAND_READY_B	SAI1_TXD.alt1
NAND_DQS	SAI1_TXFS.alt1
NAND_CE0_B	SAI1_TXC.alt1
NAND_DATA00	SD3_DATA0.alt1

Table continues on the next page...

Table 6-41. NAND IOMUX Pin Configuration (continued)

NAND_DATA01	SD3_DATA1.alt1
NAND_DATA02	SD3_DATA2.alt2
NAND_DATA03	SD3_DATA3.alt3
NAND_DATA04	SD3_DATA4.alt4
NAND_DATA05	SD3_DATA5.alt5
NAND_DATA06	SD3_DATA6.alt6
NAND_DATA07	SD3_DATA7.alt7

6.6.5.3 Expansion Device

The ROM supports booting from MMC/eMMC and SD/eSD compliant devices.

6.6.5.3.1 Expansion Device eFUSE Configuration

SD/MMC/eSD/eMMC/SDXC boot can be performed using either USDHC ports, based on setting of the BOOT_CFG[11:10] (Port Select) fuse or it's associated GPIO input value at boot.

All USDHC ports support fast boot. See the table below for details.

Table 6-42. USDHC Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
BOOT_CFG[7]	OEM	Fast Boot Support	Yes	000	0 - Normal Boot 1 - Fast Boot
BOOT_CFG[6:4]	OEM	Bus Width	Yes	000	0 - SD/eSD/SDXC 1 - MMC/eMMC
BOOT_CFG[3:1]	OEM	SD/MMC Speed Mode/ USDHC1 IO Voltage Selection	Yes	000	MMC Speed Selection 00 - Normal 01 - High else - Reserved SD Speed Selection Speed 000 - Normal/SDR12 001 - High/SDR25 010 - SDR50 011 - SDR104 else - Reserved

Table continues on the next page...

Table 6-42. USDHC Boot eFUSE Descriptions (continued)

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
					USDHC1 IO VOLTAGE SELECTION (Only for MMC/eMMC boot) 0 - 3.3V 1 - 1.8V
BOOT_CFG[0]	OEM	USDHC2 IO VOLTAGE	Yes	0	USDHC2 IO VOLTAGE SELECTION (Only for MMC/eMMC boot) 0 - 3.3V 1 - 1.8V
BOOT_CFG[15:12]	OEM	Boot Device Selection	Yes	0000	0001 - Boot from SD/eSD 0010 - Boot from MMC/eMMC
BOOT_CFG[11:10]	OEM	USDHC Port Selection	Yes	00	00 - USDHC-1 01 - USDHC-2 10 - USDHC-3 else - reserved
BOOT_CFG[9]	OEM	SD Power Cycle Enable/ eMMC Reset Enable	Yes	0	SD power cycle/eMMC reset 0 - Disabled 1 - Enabled
BOOT_CFG[8]	OEM	USDHC Loopback Clock Selection	Yes	0	USDHC Loopback Clock Source Selection 0 - Through SD pad 1 - Direct
0x490[14:8]	OEM	SD/MMC DLL DLY Config	No	0	Delay target for USDHC DLL, it is applied to slave mode target delay or override mode target delay depends on DLL Override fuse bit value.
0x490[15]	OEM	USDHC DLL Override Enabled	No	0	0 - Not override 1 - Override
0x490[16]	OEM	USDHC DLL Enabled	No	0	0 - Disable DLL for SD/eMMC 1 - Enable DLL for SD/eMMC
0x490[17]	OEM	USDHC Override Pad Settings Selection			0 - Use default pad settings 1 - Override USDHC pad settings by using PAD_SETTINGS value
0x490[18]	OEM	USDHC_IOMUX_SION_BIT_ENABLE	No	0	0 - Disable 1 - Enable
0x490[19]	OEM	ENABLE_EMMC_5K_PULLUP	No	0	0 - 47K pullup 1 - 5K pullup

Table continues on the next page...

Table 6-42. USDHC Boot eFUSE Descriptions (continued)

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
0x490[20]	OEM	USDHC_PAD_PULL_DOWN	No	0	0 - no action 1 - pull down
0x490[21]	OEM	Issue pre-idle command enabled (for eMMC4.4)	No	0	0 - enable 1 - disable
0x490[23]	OEM	Disable SDMMC Manufacture mode	No	0	0 - Enable 1 - Disable
0X490[31:24]	OEM	USDHC pad setting override	No	0	Override pad settings default if 0X490[17] is set
0x4A0[0]	OEM	Fast Boot Acknowledge Enable	No	0	0 - Boot Ack Disabled 1 - Boot Ack Enabled
0x4A0[1]	OEM	USDHC3 IO Voltage Selection	No	0	0 - 3.3V 1 - 1.8V
0x4A0[2]	OEM	uSDHC Power Off Polarity Selection	No	0	0 - Low 1 - High
0x4A0[3]	OEM	uSDHC Power Cycle Delay Selection	No	0	0 - 5ms 1 - 2.5ms
0x4A0[5:4]	OEM	uSDHC Power Cycle Interval	No	0	00 - 20ms 01 - 10ms 10 - 5ms 11 - 2.5

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See GPIO Boot Overrides for corresponding GPIO pin.

Boot code supports following standards.

- MMCv4.4 or less
- eMMCv4.4 or less
- SDv2.0 or less
- eSDv2.10 rev-0.9, with or without FAST_BOOT.
- SDXCv3.0

MMC/SD/eSD/SDXC/eMMC can be connected to any of the USDHC blocks and can be booted by copying 4Kbyte of data from MMC/SD/eSD/eMMC device to internal RAM. After checking the Image Vector Table header value (0xD1) from Program Image, the ROM code performs a DCD check. After successful DCD extraction, the ROM code extracts from Boot Data Structure the destination pointer and length of image to be copied to RAM device from where code execution occurs.

The maximum image size to load in SD/MMC boot is 32MB. This is due to the limited number of uSDHC ADMA Buffer Descriptors allocated by ROM.

NOTE

The Initial 4Kbyte of Program Image must contain the IVT, DCD and the Boot Data structures.

Table 6-43. SD/MMC Frequencies

	SD	MMC	MMC (DDR Mode)
Identification (KHz)	347.22		
Normal Speed Mode (MHz)	25	20	25
High Speed Mode (MHz)	50	40	50
UHSI SDR50 (MHz)	100		
UHSI SDR104 (MHz)	200		

NOTE

The boot ROM code reads application image length and application destination pointer from image.

6.6.5.3.2 MMC and eMMC Boot

The following table provides MMC and eMMC boot details.

Table 6-44. MMC and eMMC Boot Details

Normal Boot Mode	During initialization (normal boot mode) the MMC frequency is set to 347.22 KHz. When the MMC card enters the identification portion of the initialization, voltage validation is performed and the ROM boot code checks high voltage settings and card capacity. The ROM boot code supports both high capacity and low capacity MMC/eMMC cards. After initialization phase is complete, the ROM boot code switches to a higher frequency (20 MHz in Normal boot mode or 40 MHz in High Speed mode). eMMC is also interfaced via USDHC and follows the same flow as MMC. The boot partition can be selected for an MMC4.x card after the card initialization is complete. The ROM code reads the BOOT_PARTITION_ENABLE field in the Ext_CSD[179] to get the boot partition to be set. If there is no boot partition mentioned in BOOT_PARTITION_ENABLE field or the user partition has been mentioned, ROM boots from the user partition.
eMMC4.3 or eMMC4.4 Device Supporting Special Boot Mode	If using an eMMC4.3 or eMMC4.4 device supporting special boot mode, it can be initiated by pulling the CMD line low. If BOOT ACK is enabled, the eMMC4.3/eMMC4.4 device sends the BOOT ACK via DATA lines and ROM can read the BOOT ACK [S010E] to identify the eMMC4.3/eMMC4.4 device. eMMC4.3/eMMC4.4 device with "Boot mode" feature can only be supported via ESDHCV3-3 and with or without BOOT

Table continues on the next page...

Table 6-44. MMC and eMMC Boot Details (continued)

	ACK. If BOOT ACK is enabled ROM waits 50 ms to get the BOOT ACK and if BOOT ACK is received by ROM. If BOOT ACK is disabled ROM waits 1 second for data. If BOOT ACK or data was received then eMMC4.3/eMMC4.4 is booted in "Boot mode", otherwise eMMC4.3/eMMC4.4 boots as a normal MMC card from the selected boot partition. This boot mode can be selected by BOOT_CFG[7] (Fast Boot) fuse. BOOT ACK is selected by 0x4A0[0].
eMMC4.4 Device	If using eMMC4.4 device, Double Data Rate (DDR) mode can be used. This mode can be selected by BOOT_CFG2[7:5] (Bus Width) fuse.

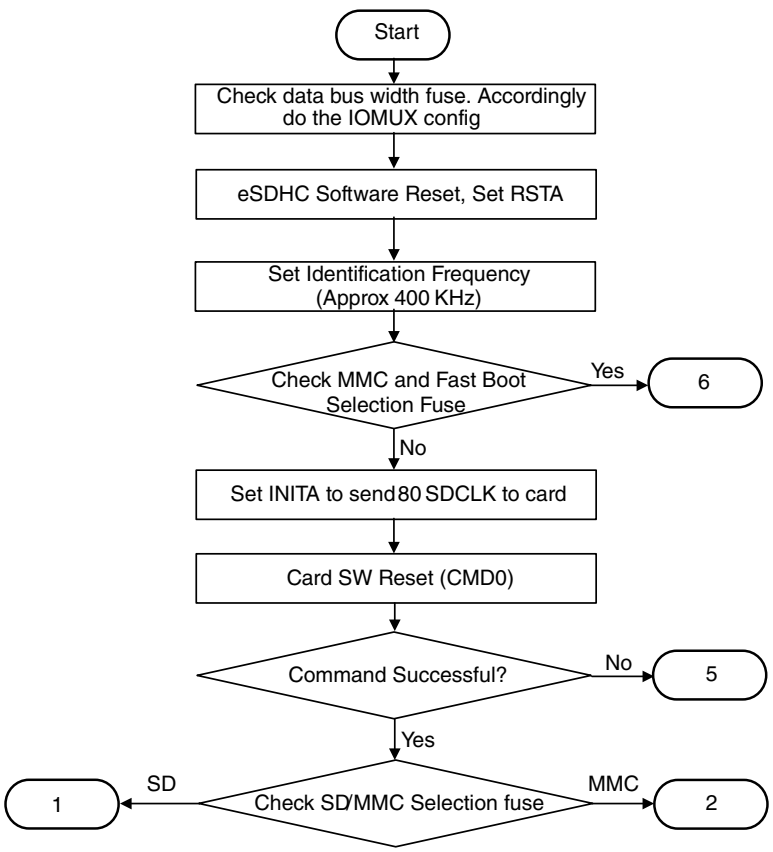


Figure 6-31. Expansion Device Boot Flow (1 of 6)

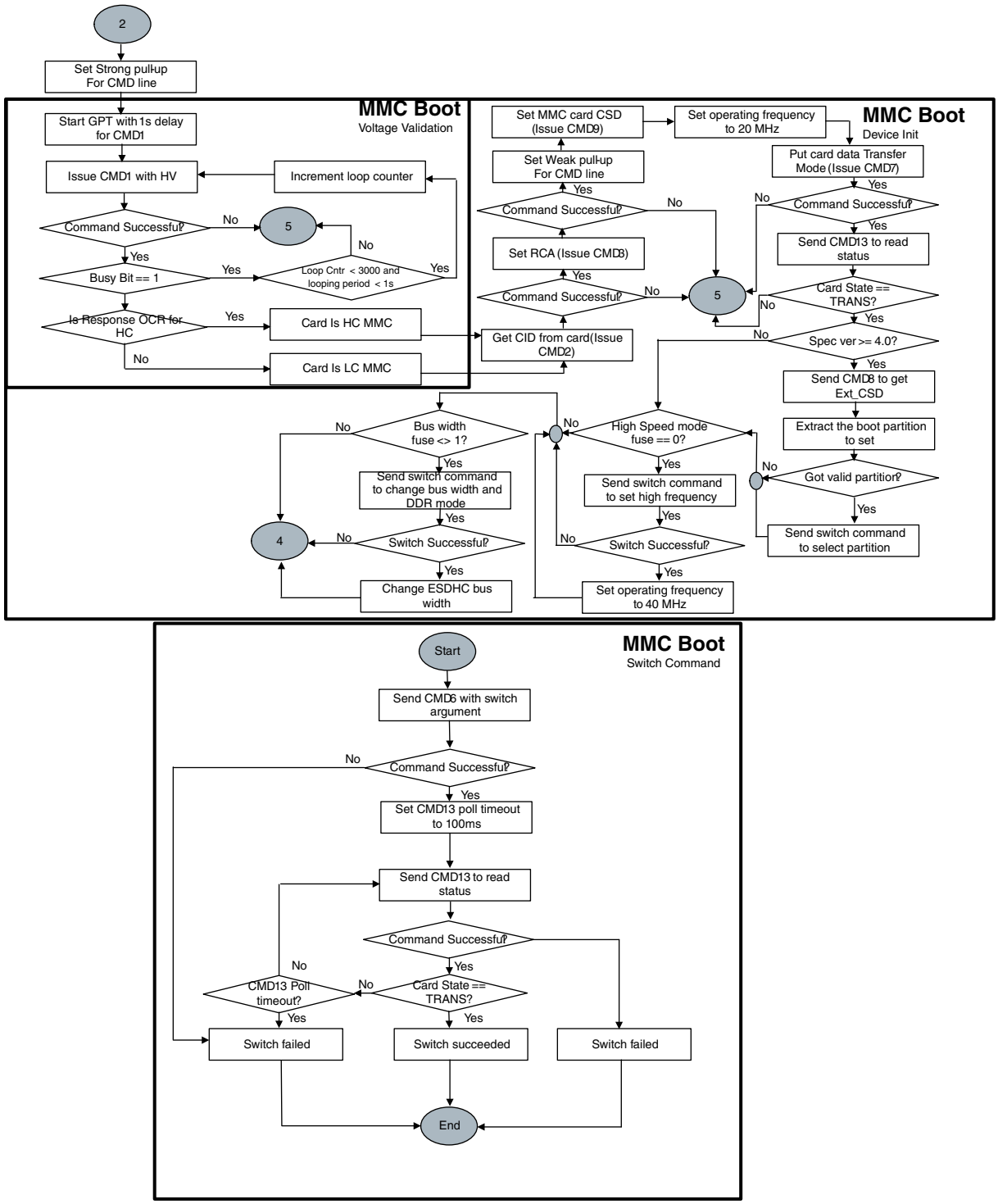


Figure 6-32. Expansion Device (MMC) Boot Flow (2 of 6)

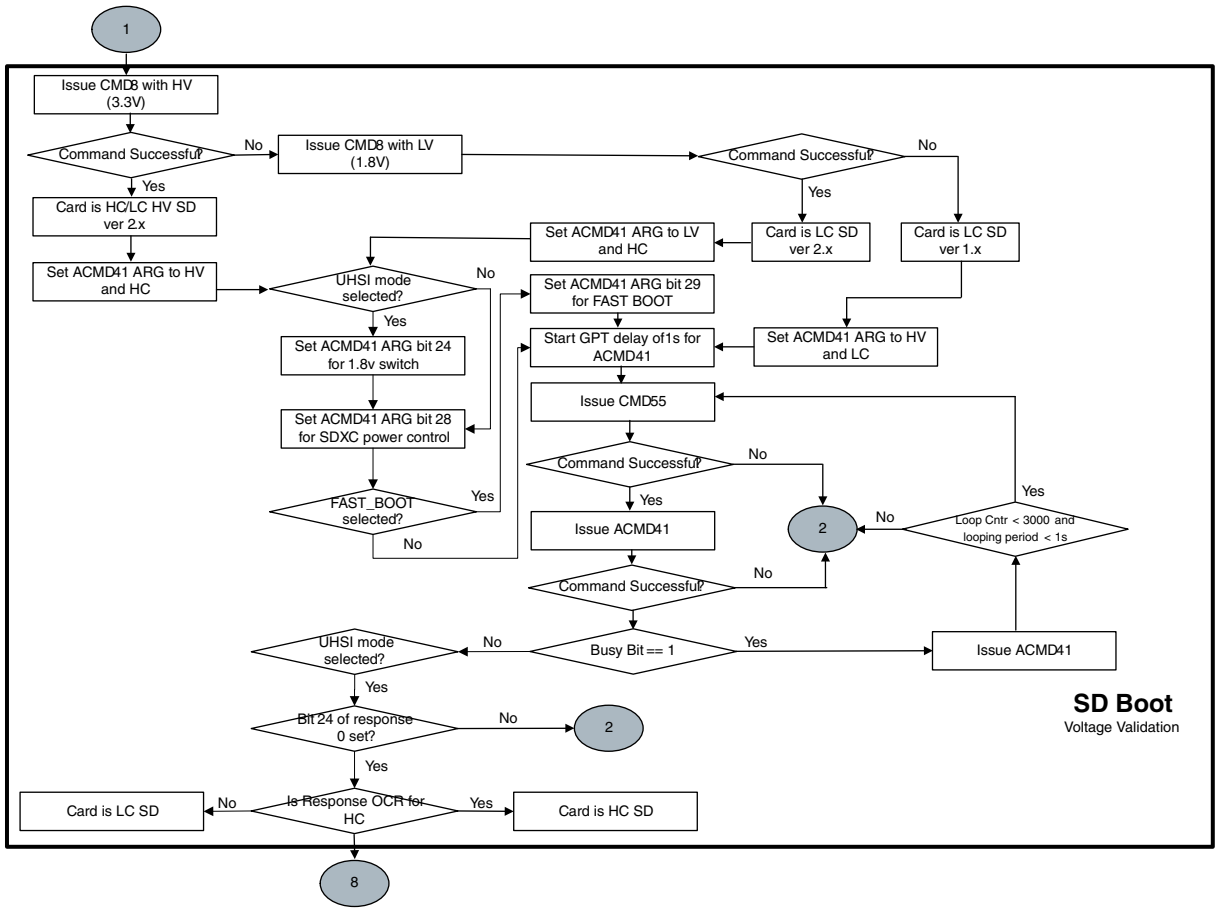


Figure 6-33. Expansion Device (SD/eSD/SDXC) Boot Flow (3 of 6) Part 1

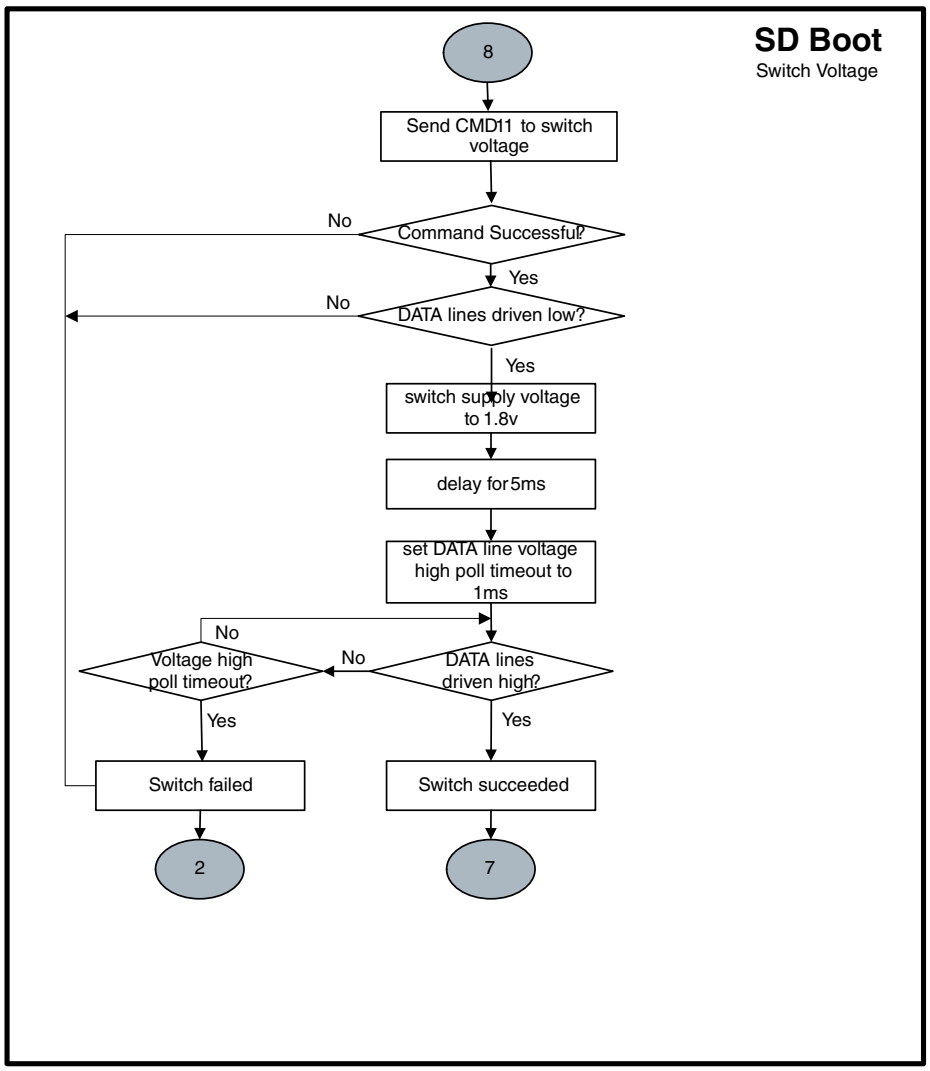
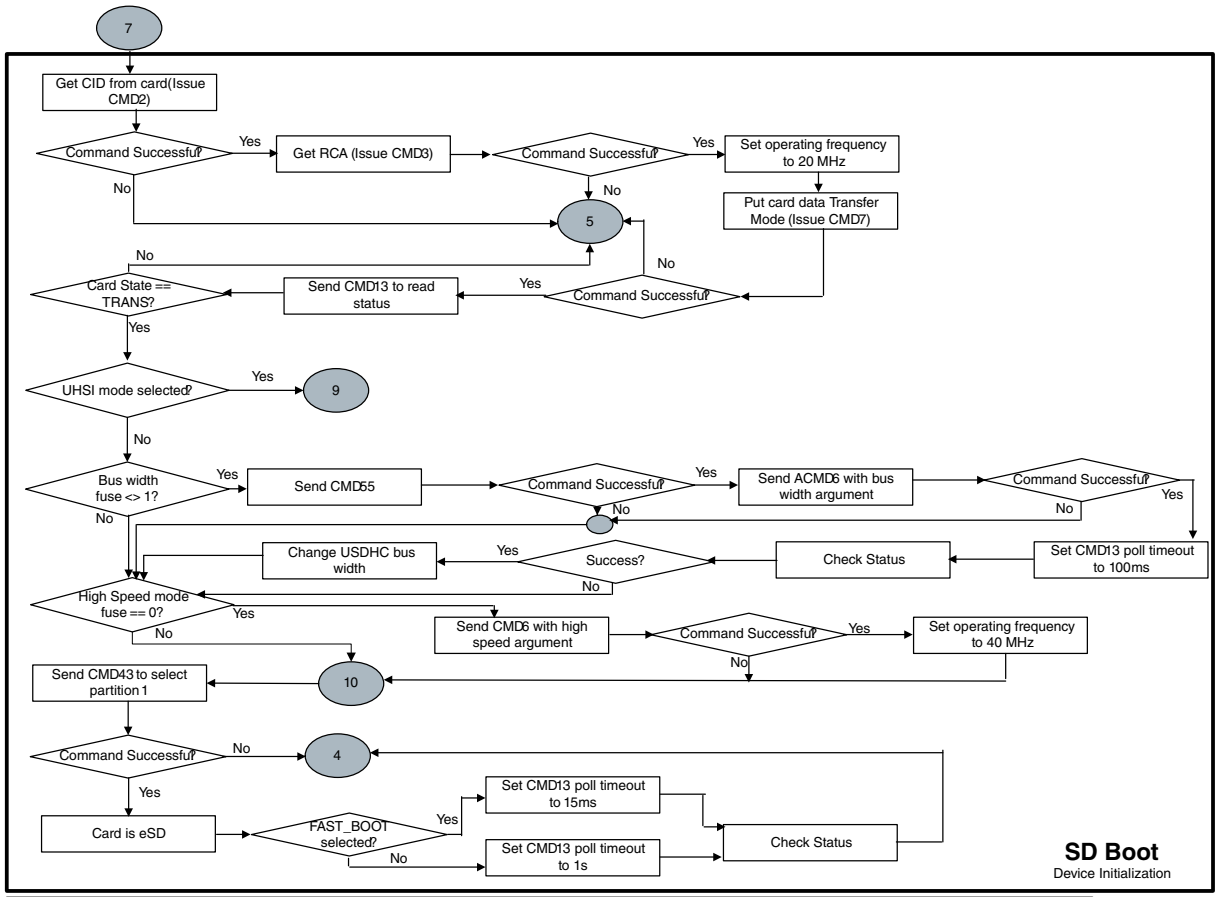


Figure 6-34. Expansion Device (SD/eSD/SDXC) Boot Flow (3 of 6) Part 2



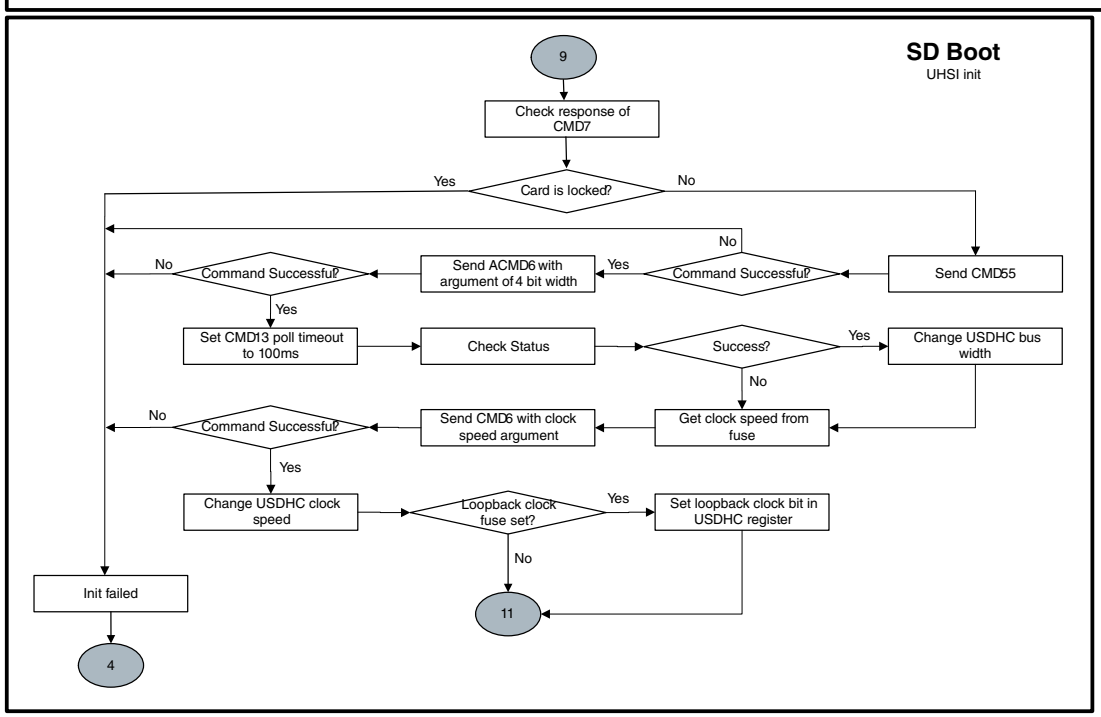
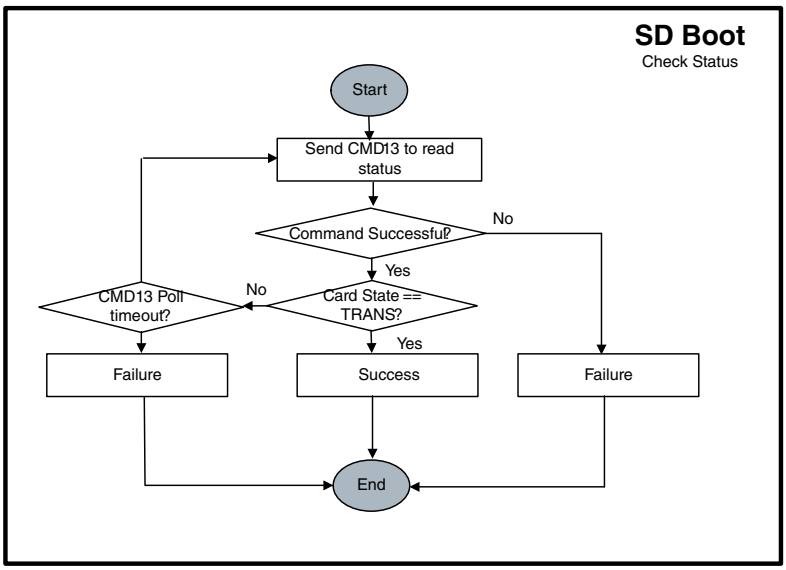
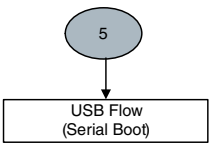


Figure 6-35. Expansion Device (MMCSD/eSD/SDXC) Boot Flow (4 of 6)





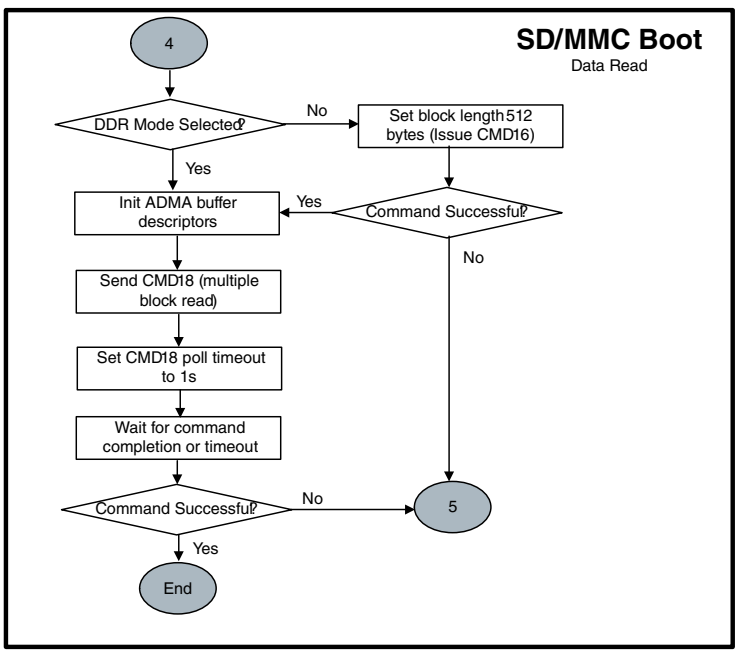
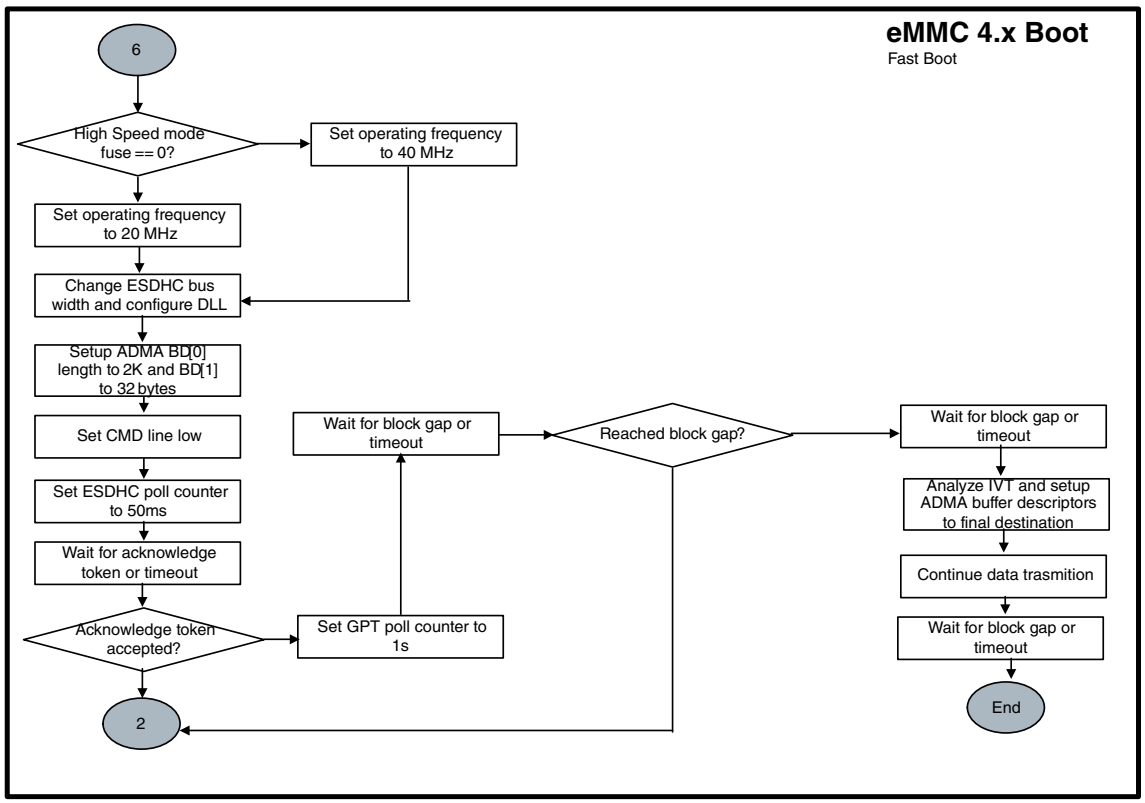


Figure 6-36. Expansion Device (SD/eSD) Boot Flow (5 of 6)



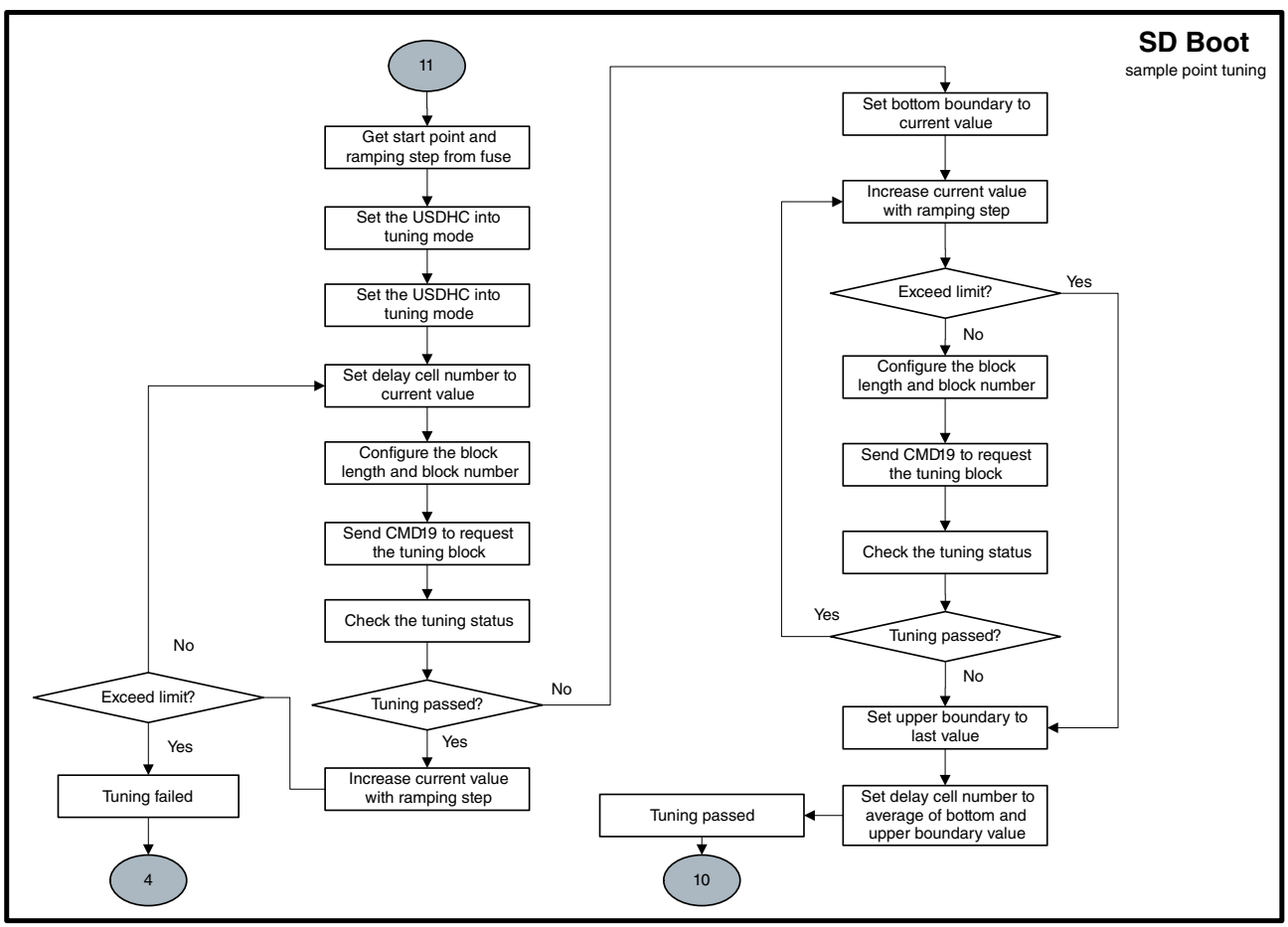


Figure 6-37. Expansion Device Boot Flow (6 of 6)

6.6.5.3.3 SD, eSD and SDXC

After the normal boot mode initialization begins, the SD/eSD/SDXC frequency is set to 347.22 kHz. During the identification phase, SD/eSD/SDXC card voltage validation is performed. During voltage validation, boot code first checks with high voltage settings; if that fails, it checks with low voltage settings.

The capacity of the card is also checked. Boot code supports high capacity and low capacity SD/eSD/SDXC cards after voltage validation card initialization is done.

During card initialization, the ROM boot code attempts to set the boot partition for all SD, eSD, and SDXC devices. If this fails, the boot code assumes the card is a normal SD card or SDXC card. If it does not fail, the boot code assumes it is an eSD card. After the initialization phase is over, boot code switches to a higher frequency (25 MHz in Normal Speed mode or 50 MHz in High Speed Mode). ROM also supports FAST_BOOT mode booting from eSD card. This mode can be selected by BOOT_CFG[4] (Fast Boot) fuse described in Table 6-42.

For UHSI cards, clock speed fuses can be set to SDR50 or SDR104 on USDHC1, USDHC2, USDHC3 and USDHC4 ports. This will enable the voltage switch process to set the signaling voltage to 1.8V during voltage validation. The bus width is fixed at 4 bits wide and a sampling point tuning process is needed to calibrate the number of delay cells. If SD Loopback Clock eFuse is set, the feedback clock will come directly from the loopback SD clock, instead of the card clock (by default). The SD clock speed can be selected by BOOT_CFG[3:2], and the SD Loopback Clock is selected by BOOT_CFG[0].

UHSI calibration start value (MMC_DLL_DLY[6:0]) and step value can be set to optimize the sample point tuning process.

If SD Power Cycle Enable eFuse is 1, ROM will set SD_RST pad low, wait 5ms and then set SD_RST pad high. If SD_RST pad is connected to SD power supply enable logic on board, it enables power cycle of SD card. This may be crucial in case when SD logic is in 1.8V states and must be reset to 3.3V states.

SDR50 and SDR104 boot are not supported on the USDHC1 and USDHC2 ports because there are no reset signals for those ports when connected in the IOMUX.

6.6.5.3.4 IOMUX Configuration for SD/MMC

Table 6-45. SD/MMC IOMUX Pin Configuration

Signal	USDHC1	USDHC2	USDHC3
CLK	SD1_CLK.alt0	SD2_CLK.alt0	SD3_CLK.alt0

Table continues on the next page...

Table 6-45. SD/MMC IOMUX Pin Configuration (continued)

Signal	USDHC1	USDHC2	USDHC3
CMD	SD1_CMD.alt0	SD2_CMD.alt0	SD3_CMD.alt0
DATA0	SD1_DATA0.alt0	SD2_DATA0.alt0	SD3_DATA0.alt0
DATA1	SD1_DATA1.alt0	SD2_DATA1.alt0	SD3_DATA1.alt0
DATA2	SD1_DATA2.alt0	SD2_DATA2.alt0	SD3_DATA2.alt0
DATA3	SD1_DATA3.alt0	SD2_DATA3.alt0	SD3_DATA3.alt0
DATA4	ECSPI2_SCLK.alt2	ECSPI1_SCLK.alt2	SD3_DATA4.alt0
DATA5	ECSPI2_MOSI.alt2	ECSPI1_MOSI.alt2	SD3_DATA5.alt0
DATA6	ECSPI2_MISO.alt2	ECSPI1_MISO.alt2	SD3_DATA6.alt0
DATA7	ECSPI2_SS0.alt2	ECSPI1_SS0.alt2	SD3_DATA7.alt0
VSELECT	GPIO1_IO08.alt1	GPIO1_IO12.alt1	GPIO1_IO13.alt1
RESET_B	SD1_RESET_B.alt5	SD2_RESET_B.alt5	SD3_RESET_B.alt5
CD_B	SD1_CD_B.alt0	-	-

6.6.5.3.5 Redundant Boot Support for Expansion Device

ROM supports redundant boot for expansion device. Primary or Secondary image is selected depending on PERSIST_SECONDARY_BOOT setting (see Table 6-32).

If PERSIST_SECONDARY_BOOT is 0, the boot ROM uses address 0x0 for primary image.

If PERSIST_SECONDARY_BOOT is 1, the boot ROM will read secondary image table from address 0x200 on boot media and will use address specified in the table.

Table 6-46. Secondary Image Table Format

Reserved (chipNum)
Reserved (driveType)
tag
firstSectorNumber
Reserved (sectorCount)

Where:

- tag: used as indication of valid secondary image table. Must be 0x00112233.
- firstSectorNumber is the first 512B sector number of the secondary image.

For secondary image support, the primary image must reserve space for secondary image table. See the figure below for typical structures layout on expansion device.

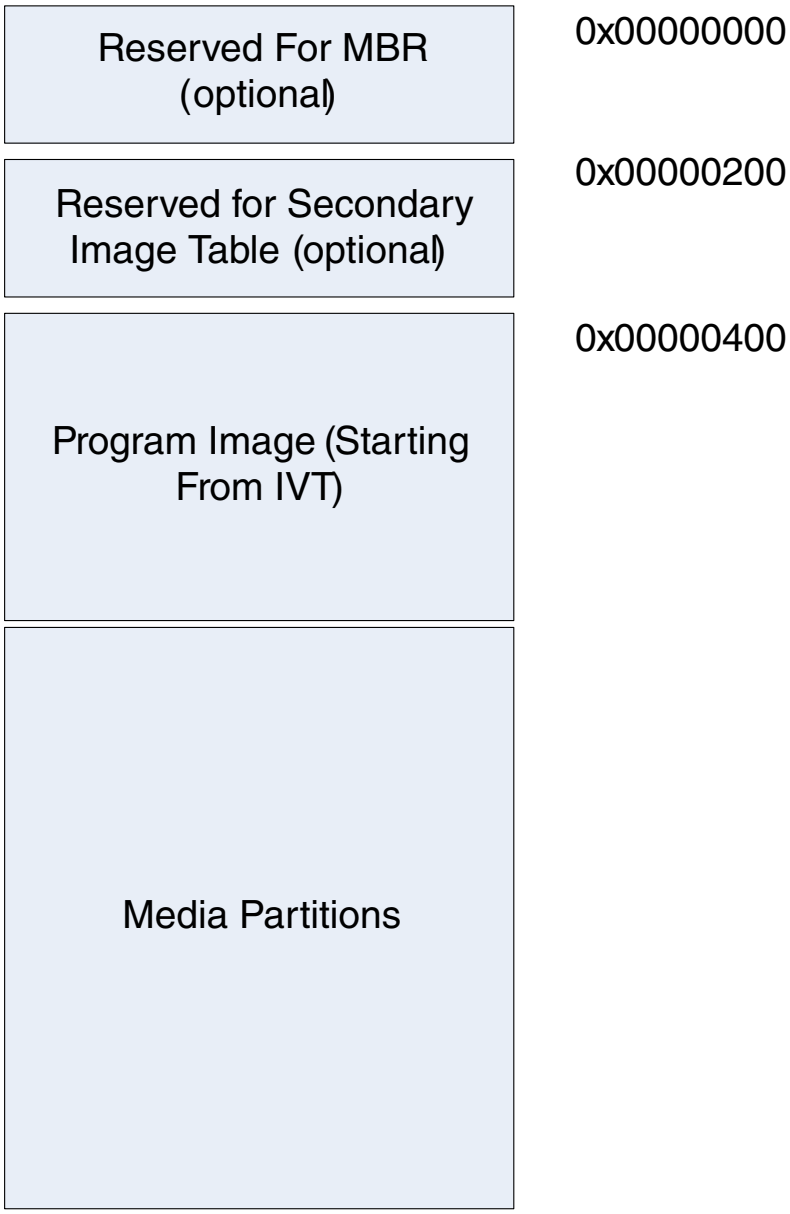


Figure 6-38. Expansion Device Structures Layout

For Closed mode, if there are failures during primary image authentication, the boot ROM will turn on PERSIST_SECONDARY_BOOT bit (see Table 6-32) and perform software reset. (After software reset, secondary image will be used.)

6.6.5.4 Serial ROM through SPI

The chip supports boot from serial memory devices, such as EEPROM and Serial Flash using the SPI.

The following ports are available for serial boot: eCSPI (eCSPI1, eCSPI2, eCSPI3, eCSPI4) interfaces.

6.6.5.4.1 Serial ROM eFUSE Configuration

The boot ROM code determines the type of device using the following parameters, either provided by eFUSE settings or sampled on the I/O pins, during boot.

See the table below for details:

Table 6-47. Serial ROM Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
BOOT_CFG[15:12]	OEM	Boot Device Selection	Yes	0000	0110 - Boot from Serial ROM
BOOT_CFG[11:9]	OEM	ECSPI Port Selection	Yes	000	000 - eCSPI1 001 - eCSPI2 010 - eCSPI3 011 - eCSPI4
BOOT_CFG[8]	OEM	SPI Addressing	Yes	0	0 - 3-bytes (24-bit) 1 - 2-bytes (16-bit)
BOOT_CFG[17]	OEM	Recovery Boot Enable	Yes	0	0 – Disabled 1 – Enabled
BOOT_CFG[7:6]	OEM	CS selection (SPI only)	Yes	00	00 – CS#0 01 – CS#1 10 – CS#2 11 – CS#3
0x480[31:29]	OEM	Recovery Port Selection	No	000	000 - eCSPI1 001 - eCSPI2 010 - eCSPI3 011 - eCSPI4
0x480[28]	OEM	Recovery SPI Addressing	No	0	0 - 3-bytes (24-bit) 1 - 2-bytes (16-bit)
0x480[27:26]	OEM	Recovery CS selection (SPI only)	No	00	00 - CS#0 01 - CS#1 10 - CS#2 11 - CS#3

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See Table 1 for corresponding GPIO pin.

The ECPSI-1/ECPSI-2/ECPSI-3/ECPSI-4 block can be used as boot device using ECSPI interface for serial ROM boot. The SPI interface is configured to operate at 15 MHz for 3-byte addressing device and 3.75 MHz for 2-byte addressing devices.

The boot ROM will copy 4Kbyte of data from Serial ROM device to internal RAM. After checking the Image Vector Table header value (0xD1) from Program Image, the ROM code performs a DCD check. After successful DCD extraction, the ROM code extracts from Boot Data Structure the destination pointer and length of image to be copied to RAM device from where code execution occurs.

NOTE

The Initial 4K of Program Image must contain the IVT, DCD and the Boot Data structures.

6.6.5.4.2 ECSPI Boot

The Enhanced Configurable SPI (ECSPI) interface is configured in master mode and the EEPROM device is connected to ECSPI interface as a slave.

The boot ROM code copies 4 KB data from EEPROM device to the internal RAM. If DCD verification is successful, the ROM code copies the initial 4 KB data, as well as the rest of the image extracted from application image, directly to the application destination. The ECSPI can read data from EEPROM using 2 or 3 byte addressing. Its burst length is 32 bytes.

NOTE

The Serial ROM Chip Select Number is determined by BOOT_CFG4[5:4] (Chip Select) fuse.

When using the SPI as boot device, the Chip supports booting from both Serial EEPROM and Serial Flash devices. The boot code determines which device is being used by reading the appropriate eFUSE/I/O values at boot (see Table 6-47 for details).

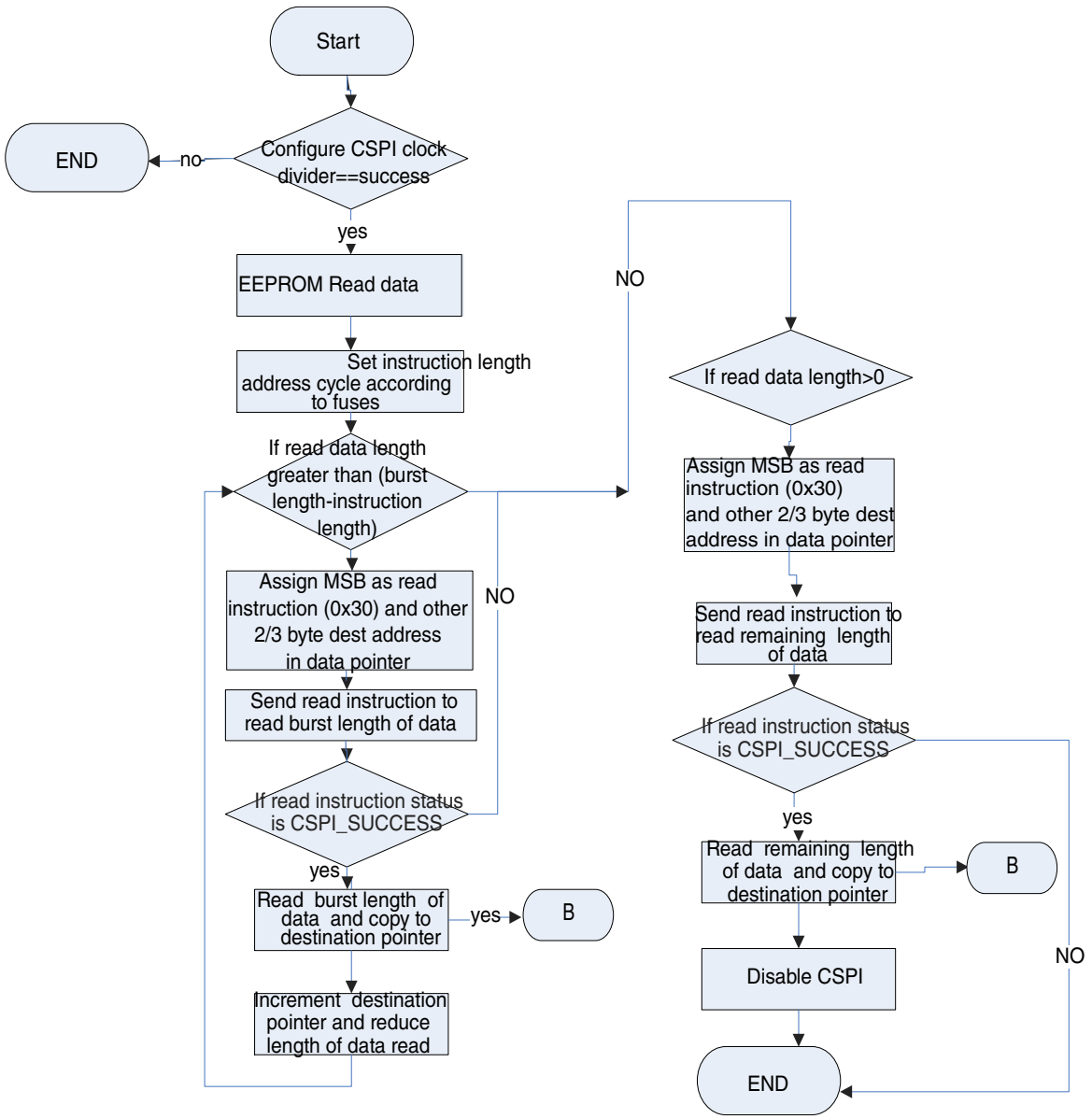


Figure 6-39. CSPI Flow chart

6.6.5.4.2.1 ECSPI IOMUX Pin Configuration

The contacts assigned to the signals used by the CSPI blocks is shown in the table below.

Table 6-48. ECSPI IOMUX Pin Configuration

Signal	eCSPI1	eCSPI2	eCSPI3	eCSPI4
MISO	ECSPI1_MISO.alt0	ECSPI2_MISO.alt0	SAI2_TXFS.alt1	SD1_CD_B.alt3

Table continues on the next page...

Table 6-48. ECSPI IOMUX Pin Configuration (continued)

MOSI	ECSPI1_MOSI.alt0	ECSPI2_MOSI.alt0	SAI2_TXC.alt1	SD1_WP.alt3
SCLK	ECSPI1_SCLK.alt0	ECSPI2_SCLK.alt0	SAI2_RXD.alt1	SD1_RESET_B.alt3
SS0	ECSPI1_SS0.alt0	ECSPI2_SS0.alt0	SAI2_TXD.alt1	SD1_CLK.alt3
SS1	UART1_RXD.alt3	ENET1_RX_CTL.alt2	SD1_DATA3.alt3	SD1_CMD.alt3
SS2	UART1_TXD.alt3	ENET1_RXC.alt2	SD2_CD_B.alt3	SD1_DATA0.alt3
SS3	UART2_RXD.alt3	ENET1_RXC.alt2	SD2_WP.alt3	SD1_DATA1.alt3

6.6.6 QuadSPI Serial Flash Memory Boot

6.6.6.1 QuadSPI eFUSE Configuration

Table 6-49. QSPI Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO	Shipped Value	Settings
BOOT_CFG[15:12]	OEM	Boot Device Selection	Yes	0000	0100 - Boot from QuadSPI
BOOT_CFG[11]	OEM	QuadSPI Interface Selection	Yes	0	0 - QSPI1 1 - Reserved

6.6.6.2 QuadSPI Serial Flash BOOT Operation

The Boot ROM attempts to boot from QuadSPI flash if the BOOT_CFG[15:12] fuses are programmed to 0100 as shown in the QuadSPI eFUSE Configuration table. The ROM will initialize the requested the QuadSPI Interface as selected in Fuse bit BOOT_CFG[11] in the QuadSPI eFUSE Configuration. QuadSPI interface initialization is a two step process.

The ROM expects the QuadSPI configuration parameters as explained in the QuadSPI Configuration Parameters to be present in the Serial Flash memory from offset 0x400 of serial flash of length 512 bytes. The ROM reads these configuration parameters using the default read command configured in the LUT of the QuadSPI interface with SCLOCK operating at 20 MHz.

In the second step, ROM configures the selected QuadSPI interface with the configuration parameters read from the serial flash and starts the boot procedure. Refer to Table 19-12 for details regarding QuadSPI configuration parameters and to the QuadSPI boot flow chart for detailed boot flow chart of QuadSPI.

Both booting an XIP and non XIP image is supported from serial flash. For XIP boot, the image has to be built for QuadSPI address space and for non XIP the image can be built to execute from DDR or OCRAM.

For QUAD mode boot, the Boot ROM expects the Quad Enable bit inside the QSPI Flash to be already set before booting starts. Therefore, the QUAD enable bit must be set in the non-volatile register of the flash at the time of programming.

NOTE

If the SPI flash device requires quad enable command, it can be sent via configuration structure fields: device_quad_mode_en, device_cmd, write_cmd_ipcr, write_enable_ipcr, busy_bit_offset, read_status_ipcr.

6.6.6.3 QuadSPI Configuration Parameters

The QuadSPI Configuration Parameters Table is built in boot image at fixed offset 0x400 from QSPI NOR A1 base address (368 bytes). Table below lists various QuadSPI Configuration Parameters.

Table 6-50. QuadSPI Configuration Parameters

Name	Offset	Size in Bytes	Description
DQS Loopback	0	4	DQS LoopBack Mode to enable Dummy Pad, 0 - Disable, 1 - Enable
Hold Delay	4	4	Hold Delay for QSPI[0,1] A/B Value / QSPI1 B / QSPI1 A 00 / Disable / Disable 01 / Disable / Enable 10 / Enable / Disable 11 / Enable / Enable
Reserved	8	4	Reserved to 0
Reserved	12	4	Reserved to 0
device_quad_mode_en	16	4	Send Quad enable command to SPI device.
device_cmd	20	4	Command to send to SPI device.
write_cmd_ipcr	24	4	IPCR register value for write command
write_enable_ipcr	28	4	IPCR register value for Enable
Chip Select hold time	32	4	This is chip select hold time in terms of Serial clock (For Example 1 serial clock cycle 0-15).
Chip Select setup time	36	4	Chip select setup time in terms of Serial clock (For example 1 serial clock).

Table continues on the next page...

Table 6-50. QuadSPI Configuration Parameters (continued)

<table>
<tr><th>Name</th><th>Offset</th><th>Size in Bytes</th><th>Description</th></tr>
<tr><td>Serial Flash A1 size</td><td>40</td><td>4</td><td>Serial Flash A1 size in units of bytes</td></tr>
<tr><td>Serial Flash A2 size</td><td>44</td><td>4</td><td>Serial Flash A2 size in units of bytes</td></tr>
<tr><td>Serial Flash B1 size</td><td>48</td><td>4</td><td>Serial Flash B1 size in units of bytes</td></tr>
<tr><td>Serial Flash B2</td><td>52</td><td>4</td><td>Serial Flash B2 size in units of bytes</td></tr>
<tr><td>Serial Clock Frequency</td><td>56</td><td>4</td><td>This is serial clock frequency select parameter.
<table><tr><th>Value</th><th>Clock</th></tr><tr><td>00</td><td>18 MHz</td></tr><tr><td>01</td><td>49 MHz</td></tr><tr><td>02</td><td>55 MHz</td></tr><tr><td>03</td><td>60 MHz</td></tr><tr><td>04</td><td>66 MHz</td></tr><tr><td>05</td><td>76 MHz</td></tr><tr><td>06</td><td>99 MHz (only SDR mode)</td></tr></table></td></tr>
<tr><td>busy_bit_offset</td><td>60</td><td>4</td><td>SPI Flash device busy bit offset in its status register, used for enabling Quad mode of SPI device</td></tr>
<tr><td>Mode of operation of serial Flash</td><td>64</td><td>4</td><td>This field describes the mode of operation of Serial flash
<table><tr><th>Value</th><th>Mode</th></tr><tr><td>01</td><td>Single</td></tr><tr><td>02</td><td>Dual</td></tr><tr><td>04</td><td>Quad</td></tr></table></td></tr>
<tr><td>Serial Flash Port B Selection</td><td>68</td><td>4</td><td>Port A is always available. This field informs the device ROM the availability of Port B.
0 – Port B is not used
1 – Port B is used</td></tr>
<tr><td>Dual Data Rate mode enable</td><td>72</td><td>4</td><td>This field enables the device ROM to enable DDR mode.
0 – DDR mode is disabled
1 – DDR mode is enabled</td></tr>
<tr><td>Data Strobe Signal enable in Serial Flash</td><td>76</td><td>4</td><td>This field enables Data Strobe signal in Serial Flash which supports it.
0 – Disable DQS
1 – Enable DQS</td></tr>
<tr><td>Parallel Mode enable</td><td>80</td><td>4</td><td>This field enables parallel mode. Data will be read from serial Flash in parallel mode. Refer to QSP chapter for detail.
0 – Disable Parallel mode in QSPI</td></tr>
</table>

Table continues on the next page...

Table 6-50. QuadSPI Configuration Parameters (continued)

Name	Offset	Size in Bytes	Description
			1 – Enable Parallel Mode in QSPI
CS1 on Port A	84	4	This field enables CS1 on port A 0 – Disable CS1 on Port A 1 – Enable CS1 on Port A
CS1 on Port B	88	4	This field enables CS1 on port B 0 – Disable CS1 on Port B 1 – Enable CS1 on Port B
Full Speed Phase Selection	92	4	Select the edge of the sampling clock valid for full speed commands: 0: Select sampling at non-inverted clock 1: Select sampling at inverted clock This bit is also used to shift the dqs_enable when DQS mode is selected
Full Speed Delay Selection	96	4	Select the delay w.r.t. the reference edge for the sample point valid for full speed commands: 0: One clock cycle delay 1: Two clock cycles delay This bit is also used to shift the dqs_enable when DQS mode is selected
DDR Sampling Point	100	4	Select the sampling point for incoming data when serial flash is in DDR mode. **NOTE:** Valid Values are (b000-b111)
LUT program sequence	104	256	256 Bytes of Look up table program sequence. ROM programs the LUT of QuadSPI with this parameter supplied. It assumes that the optimize read command sequence which will be used to read data from Serial flash and fill the AHB buffer is programmed at index 0.
read_status_ipcr	360	4	IPCR value of Read Status Reg
enable_dqs_phase	364	4	Enable DQS Phase
Reserved	368	36	Not used
dqs_pad_setting_override	404	4	DQS pin pad setting override
sclk_pad_setting_override	408	4	SCLK pin pad setting override
data_pad_setting_override	412	4	DATA pins pad setting override
cs_pad_setting_override	416	4	CS pins pad setting override
dqs_loopback_internal	420	4	0: dqs loopback from pad 1: dqs loopback internally
dqs_phase_sel	424	4	dqs phase selection
dqs_fa_delay_chain_sel	428	4	dqs fa delay chain selection
dqs_fb_delay_chain_sel	432	4	dqs fb delay chain selection

Table continues on the next page...

Table 6-50. QuadSPI Configuration Parameters (continued)

Name	Offset	Size in Bytes	Description
sclk_fa_delay_chain_sel	436	4	sclk fa delay chain selection
sclk_fb_delay_chain_sel	440	4	sclk fb delay chain selection
Reserved	448	60	Reserved
tag	508	4	End flag of QSPI parameters area

6.6.6.4 IOMUX Configuration for QSPI Devices

The QSPI interface uses dedicated contacts on the IC. The contacts assigned to the data signals used by QSPI are shown in the table below.

6.6.6.5 QuadSPI boot flow chart

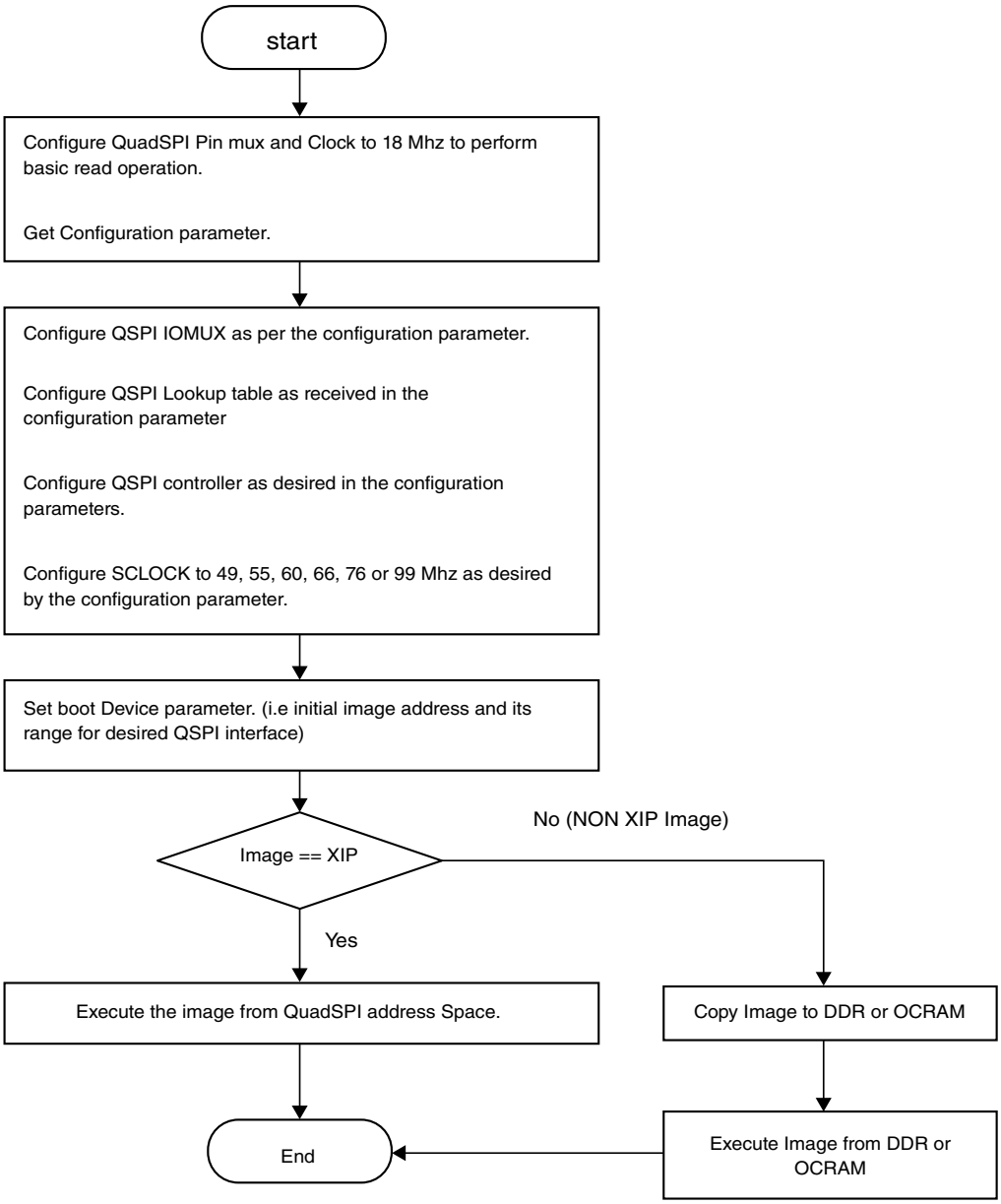


Figure 6-40. QuadSPI boot flow chart

NOTE

If flash is configured for "High performance mode (where command is generated only once)" in LUT program sequence. then external reset should be routed to flash reset to allow rebooting in case of any device reset other than Power On Reset. Also this high performance mode must be exited by application before any Low power mode entry where the device is supposed to reboot from QSPI flash on Low power mode exit. In general, any preserved configuration in external flash will not be understood by device after reset.

6.6.7 Program image

This section describes the data structures that are required to be included in a user's program image. A program image consists of:

- Image vector table—A list of pointers located at a fixed address that the ROM examines to determine where other components of the program image are located
- Boot data—A table indicating the program image location, program image size in bytes, and the plugin flag
- Device configuration data—IC configuration data
- User code and data

6.6.7.1 Image Vector Table and Boot Data

The Image Vector Table (IVT) is the data structure that the ROM reads from the boot device supplying the program image containing the required data components to perform a successful boot.

The IVT includes the program image entry point, a pointer to Device Configuration Data (DCD) and other pointers used by the ROM during the boot process.The ROM locates the IVT at a fixed address that is determined by the boot device connected to the Chip. The IVT offset from the base address and initial load region size for each boot device type is defined in the table below. The location of the IVT is the only fixed requirement by the ROM. The remainder or the image memory map is flexible and is determined by the contents of the IVT.

Table 6-51. Image Vector Table Offset and Initial Load Region Size

Boot Device Type	Image Vector Table Offset	Initial Load Region Size
EIM/QSPI NOR	1 Kbyte = 0x400 bytes	Entire Image Size
NAND	1 Kbyte = 0x400 bytes	4 Kbyte
OneNAND	1 Kbyte = 0x400 bytes	1 Kbyte
SD/MMC/eSD/eMMC/SDXC	1 Kbyte = 0x400 bytes	4 Kbyte
SPI EEPROM	1 Kbyte = 0x400 bytes	4 Kbyte

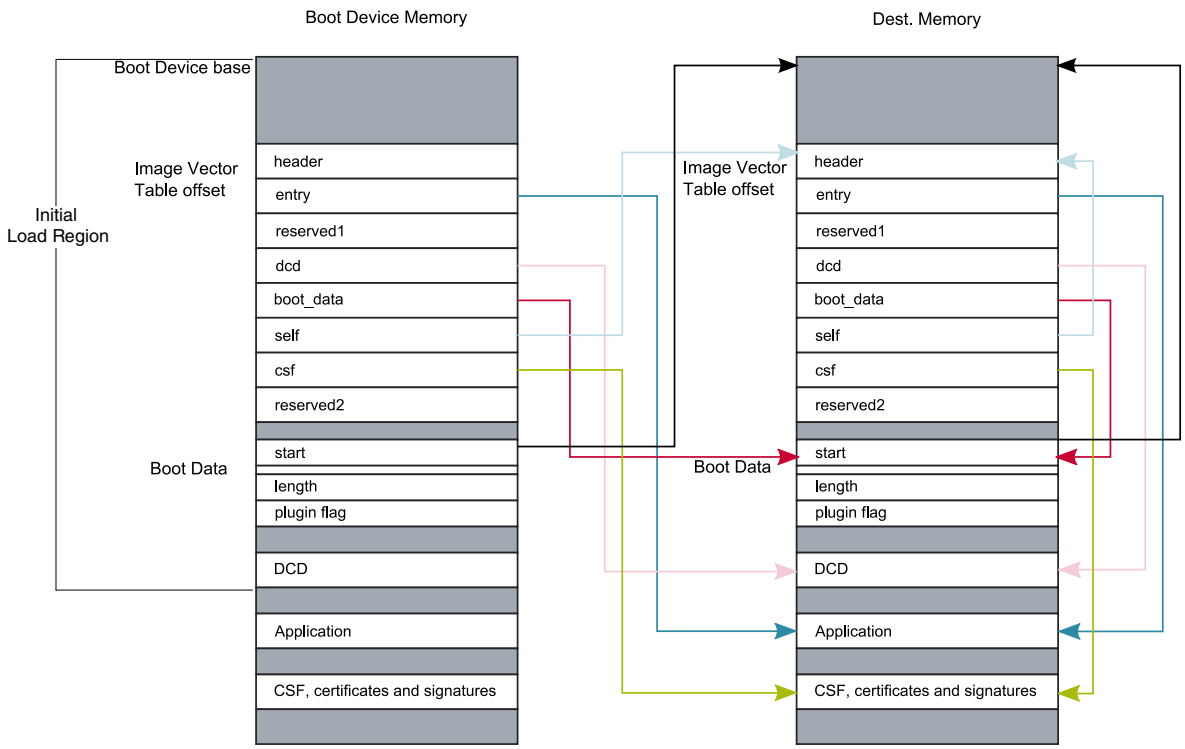


Figure 6-41. Image Vector Table

6.6.7.1.1 Image Vector Table Structure

The IVT has the following format where each entry is a 32 bit word:

Table 6-52. IVT Format

header
entry: Absolute address of the first instruction to execute from the image
reserved1: Reserved and should be zero
dcd: Absolute address of the image DCD. The DCD is optional so this field may be set to NULL if no DCD is required. See Device Configuration Data (DCD) for further details on DCD.
boot data: Absolute address of the Boot Data
self: Absolute address of the IVT. Used internally by the ROM
csf: Absolute address of Command Sequence File (CSF) used by the HAB library. See High Assurance Boot (HAB) for details on secure boot using HAB. This field must be set to NULL when not performing a secure boot
reserved2: Reserved and should be zero

Figure 6-42 shows the IVT header format:

Figure 6-42. IVT Header Format

where:

```
Tag: A single byte field set to 0xD1
Length: a two byte field in big endian format containing the overall length of the IVT,
in bytes, including the header. (the length is fixed and must have a value of
32 bytes)
Version: A single byte field set to 0x40 or 0x41
```

6.6.7.1.2 Boot Data Structure

The Boot Data must follow the format defined in the table found here, each entry is a 32-bit word.

Table 6-53. Boot Data Format

start	Absolute address of the image
length	Size of the program image
plugin	Plugin flag (see Plugin Image)

6.6.7.2 Device Configuration Data (DCD)

Upon reset, the Chip uses the default register values for all peripherals in the system. However, these settings typically are not ideal for achieving optimal system performance and there are even some peripherals that must be configured before they can be used.

The DCD is configuration information contained in a Program Image, external to the ROM, that the ROM interprets to configure various peripherals on the Chip.

For example, the EIM default settings allow the core to interface to a NOR flash device immediately out of reset. This allows the Chip to interface with any NOR flash device, but has the cost of slow performance. Additionally, some components such as DDR require some sequence of register programming as part of configuration before it is ready to be used. The DCD feature can be used to program the EIM registers and MMDC registers to the optimal settings.

The ROM determines the location of the DCD table based on information located in the Image Vector Table (IVT). See Image Vector Table and Boot Data for more details. The DCD table shown below is a big endian byte array of the allowable DCD commands. The maximum size of the DCD limited to 1768 bytes.

Figure 6-43. DCD Data Format

The DCD header is 4 bytes with the following format:

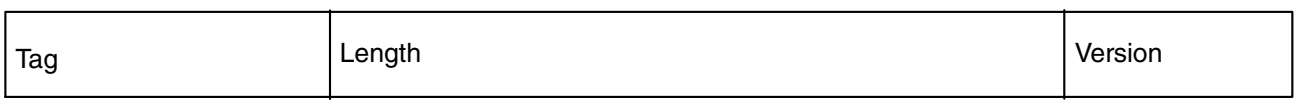


Figure 6-44. DCD Header Format

where:

```
Tag: A single byte field set to 0xD2
Length: a two byte field in big endian format containing the overall length of the DCD,
in bytes, including the header
Version: A single byte field set to 0x41
```

6.6.7.2.1 Write Data Command

The Write Data Command is used to write a list of given 1-, 2- or 4-byte values or bitmasks to a corresponding list of target addresses.

The format of Write Data Command, again a big endian byte array, is shown in the table below.

Table 6-54. Write Data Command Format

Tag	Length	Parameter
Address		
Value/Mask		

Table continues on the next page...

Table 6-54. Write Data Command Format (continued)

[Address]
[Value/Mask]
...
[Address]
[Value/Mask]

where:

```
Tag: A single byte field set to 0xCC
Length: A two byte field in big endian format containing the length of the Write Data
Command, in bytes, including the header
Address: target address to which data should be written
Value/Mask: data value or bitmask to be written to preceding address
```

The Parameter field is a single byte divided into bitfields as follows:

Table 6-55. Write Data Command Parameter field

7	6	5	4	3	2	1	0
flags					bytes		

where

```
bytes: width of target locations in bytes. Either 1, 2 or 4
flags: control flags for command behavior.
Data Mask = bit 3: if set, only specific bits may be overwritten at target address
(otherwise all bits may be overwritten)
Data Set = bit 4: if set, bits at the target address overwritten with this flag
(otherwise it is ignored)
```

One or more target address and value/bitmask pairs can be specified. The same bytes and flags parameters apply to all locations in the command.

When successful, this command writes to each target address in accordance with the flags as follows:

Table 6-56. Interpretation of Write Data Command Flags

"Mask"	"Set"	Action	Interpretation
0	0	*address = val_msk	Write value
0	1	*address = val_msk	Write value
1	0	*address &= ~val_msk	Clear bitmask
1	1	*address |= val_msk	Set bitmask

NOTE

If any of the target addresses does not have the same alignment as the data width indicated in the parameter field, none of the values are written.

If any of the values is larger or any of the bitmasks is wider than permitted by the data width indicated in the parameter field, none of the values are written.

If any of the target addresses do not lie within an allowed region, none of the values are written. The list of allowable blocks and target addresses for the Chip are given below.

Table 6-57. Valid DCD Address Ranges

Address range	Start address	Last Address
DDR controller register set	0x30790000	0x307AFFFF
IOMUXC register set	0x30330000	0x3033FFFF
IOMUXC GPR register set	0x30340000	0x3034FFFF
IOMUXC LPSR register set	0x302C0000	0x302CFFFF
IOMUXC LPSR GPR register set	0x30270000	0x3027FFFF
WEIM register set	0x30BC0000	0x30BCFFFF
CCM register set	0x30380000	0x3038FFFF
Anatop register set	0x30360000	0x3036FFFF
GPT2 register set	0x302E0000	0x302EFFFF
SRC registers for DDRC	0x30391000	0x30391FFF

6.6.7.2.2 Check Data Command

The Check Data Command is used to test for a given -1, 2- or 4-byte bitmasks from a source address.

The Check Data Command is a big endian byte array with format shown in the table below.

Table 6-58. Check Data Command Format

Tag	Length	Parameter
Address		
Mask		
[Count]		

where:

```
Tag: A single byte field set to 0xCF
Length: A two byte field in big endian format containing the length of the Check Data
Command, in bytes, including the header
Address: source address to test
Mask: bit mask to test
Count: optional poll count. If count is not specified this command will poll indefinitely
until the exit condition is met. If count = 0, this command behaves as for NOP.
```

The Parameter field is a single byte divided into bitfields as follows:

Table 6-59. Check Data Command Parameter field

7	6	5	4	3	2	1	0
flags					bytes		

where

```
bytes: width of target locations in bytes. Either 1, 2 or 4
flags: control flags for command behavior.
Data Mask = bit 3: if set, only specific bits may be overwritten at target address
(otherwise all bits may be overwritten)
Data Set = bit 4: if set, bits at the target address overwritten with this flag
(otherwise it is ignored)
```

This command polls the source address until either the exit condition is satisfied, or the poll count is reached. The exit condition is determined by the flags as follows:

Table 6-60. Interpretation of Check Data Command Flags

"Mask"	"Set"	Action	Interpretation
0	0	(*address & mask) == 0	All bits clear
0	1	(*address & mask) == mask	All bits set
1	0	(*address & mask)!= mask	Any bit clear
1	1	(*address & mask)!= 0	Any bit set

NOTE

If the source address does not have the same alignment as the data width indicated in the parameter field, the value is not read.

If the bitmask is wider than permitted by the data width indicated in the parameter field, the value is not read.

6.6.7.2.3 NOP Command

This command has no effect.

The format of NOP Command is a big endian four byte array as shown in the table below.

Table 6-61. NOP Command Format

Tag	Length	Undefined

where:

```
Tag: A single byte field set to 0xC0
Length: A two byte field in big endian containing the length of the NOP Command in bytes.
Fixed to a
value of 4.
Undefined: This byte is ignored and can be set to any value.
```

6.6.7.2.4 Unlock Command

The Unlock Command is used to prevent specific engine features being locked when exiting ROM.

The format of Unlock Command, again a big endian byte array, is shown in the table below.

Table 6-62. Unlock Command Format

Tag	Length	Eng
Value		
Value		
...		
Value		

where:

```
Tag: A single byte field set to 0xB2
Eng: Engine to be left unlocked.
Values: [optional] unlock values required by engine.
```

NOTE

This command may not be used in DCD structure if the SEC_CONFIG is configured as closed.

6.6.8 Plugin Image

The ROM supports a limited number of boot devices. When using other devices as a boot source (for example, Ethernet, CDROM, or USB), then the supported boot device must be used (typically serial ROM) as firmware to provide the missing boot drivers.

Additionally, the plugin can be customized to support boot drivers, which is more flexible when doing device initialization, such as condition judging, delay assertion, or to apply custom settings to the boot device and memory system.

In addition to standard images, the chip also supports plugin images. Plugin images return execution to the ROM whereas a standard image does not.

The boot ROM detects the image type using the plugin flag of the boot data structure (see Boot Data Structure). If the plugin flag is 1, then the ROM uses the image as a plugin function. The function must initialize the boot device and copy the program image to the final location. At the end the plugin function must return with the program image parameters. (See High level boot sequence for details about boot flow).

The boot ROM authenticates the plugin image prior to running the plugin function and then authenticates the program image.

The plugin function must follow the API described below:

typedef BOOLEAN (*plugin_download_f)(void **start, size_t *bytes, UINT32 *ivt_offset);

ARGUMENTS PASSED:

- start - Image load address on exit.
- bytes - Image size on exit.
- ivt_offset - Offset in bytes of the IVT from the image start address on exit.

RETURN VALUE:

- 1 - on success
- 0 - on failure

6.6.9 Serial Downloader

The Serial Downloader provides a means to download a Program Image to the chip over USB serial connection.

In this mode the ROM programs WDOG1 for a time-out specified by fuse WDOG Time-out Select (See fusemap for details) if WDOG_ENABLE eFuse is 1 and continuously polls for USB connection. If no activity is found on USB OTG1 and the watchdog timer expires, the ARM core is reset.

NOTE

After being loaded, the downloaded image is responsible to manage the watchdog resets properly.

Figure 6-21 shows USB boot flow.

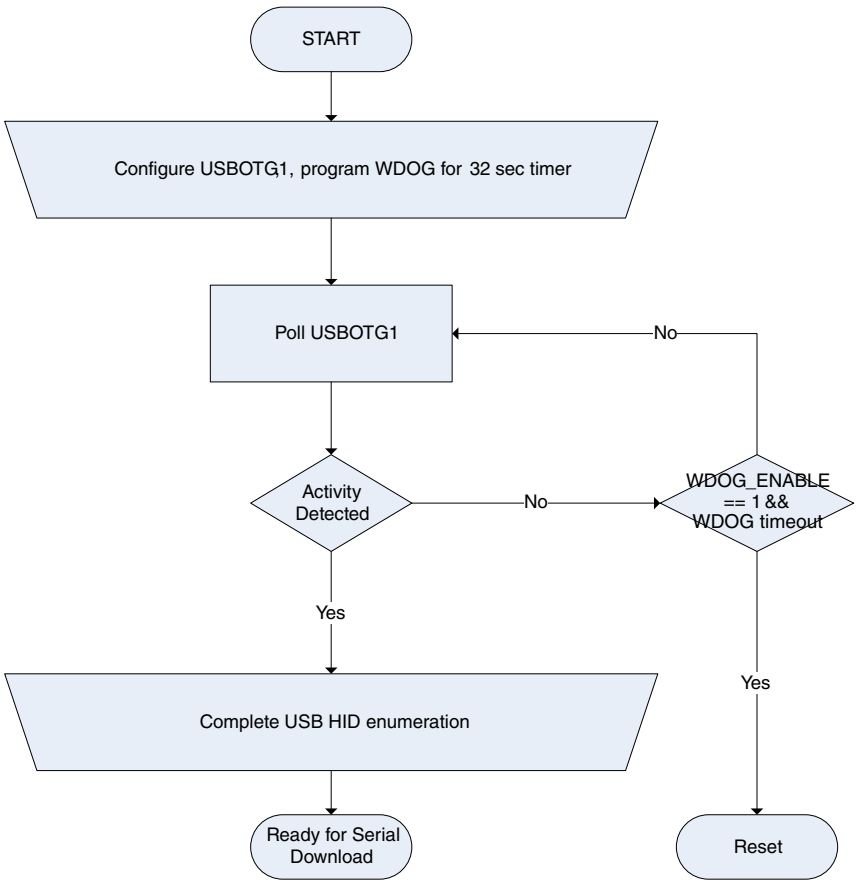


Figure 6-45. Serial Download Boot Flow

6.6.9.1 USB

USB support is composed of the (core controller, compliant with the USB 2.0 specification) and the USBPHY (HS USB transceiver).

The ROM supports the USB OTG port for boot purposes. The other USB ports on the chip are not supported for boot purposes.

The USB Driver is implemented as a USB HID class. A collection of 4 HID reports are used to implement SDP protocol for data transfers as described in Table 6-63.

Table 6-63. USB HID Reports

Report ID (first byte)	Transfer Endpoint	Direction	Length	Description
1	control OUT	Host to device	17 bytes	SDP command from host to device
2	control OUT	Host to device	Up to 1025 bytes	Data associated with report 1 SDP command
3	interrupt	Device to host	5 bytes	HAB security configuration. Device sends 0x12343412 in closed mode and 0x56787856 in open mode.
4	interrupt	Device to host	Up to 65 bytes	Data in response to SDP command in report 1

6.6.9.1.1 USB Configuration Details

The USB OTG function device driver supports a high speed (HS for UTMI) non-stream mode with a maximal packet size of 512 B and a low-level USB OTG function.

The VID/PID and strings for USB device driver are listed in the table below.

Table 6-64. VID/PID and Strings for USB Device Driver

Descriptor	Value
VID	0x15A2 (Freescale vendor ID)
PID[1]	
String Descriptor1 (manufacturer)	Freescale Semiconductor, Inc.
String Descriptor2 (product)	S Blank SE Blank NS Blank
String Descriptor4	Freescale Flash
String Descriptor5	Freescale Flash

1. Allocation based on BPN (Before Part Number)

6.6.9.1.2 IOMUX Configuration for USB

The interface signals of the UTMI PHY are not configured in the IOMUX except for the USB_OTGn_ID pins. The USB ID pin function is configured using USBNC_n_CTRL2[ID_DIG_SEL] and the IOMUXC_USB_OTGn_ID_SELECT_INPUT register. The remaining pins of the UTMI PHY interface use dedicated contacts on the IC. See the chip data sheet for details.

6.6.9.2 Serial Download protocol

The 16 byte SDP command from host to device is sent using HID report 1.

The table below describes 16 byte SDP command data structure:

Table 6-65. 16 Byte SDP Command Data Structure

BYTE Offset	Size	Name	Description
0	2	COMMAND TYPE	The following commands are supported for ROM: • 0x0101 READ_REGISTER • 0x0202 WRITE_REGISTER • 0x0404 WRITE_FILE • 0x0505 ERROR_STATUS • 0x0A0A DCD_WRITE • 0x0B0B JUMP_ADDRESS • 0x0C0C SKIP_DCD_HEADER
2	4	ADDRESS	Only relevant for following commands: READ_REGISTER, WRITE_REGISTER, WRITE_FILE, DCD_WRITE, and JUMP_ADDRESS. For READ_REGISTER and WRITE_REGISTER commands, this field is address to a register. For WRITE_FILE and JUMP_ADDRESS commands, this field is an address to internal or external memory address.
6	1	FORMAT	Format of access, 0x8 for 8-bit access, 0x10 for 16-bit and 0x20 for 32-bit access. Only relevant for READ_REGISTER and WRITE_REGISTER commands.
7	4	DATA COUNT	Size of data to read or write. Only relevant for WRITE_FILE, READ_REGISTER, WRITE_REGISTER and DCD_WRITE commands. For WRITE_FILE and DCD_WRITE commands DATA COUNT is in byte units.
11	4	DATA	Value to write. Only relevant for WRITE_REGISTER command.
15	1	RESERVED	Reserved

6.6.9.2.1 SDP Command

SDP commands are decribed in the following sections.

6.6.9.2.1.1 READ REGISTER

The transaction for command READ_REGISTER consists of following reports: Report1 for command, Report3 for security configuration and Report4 for response or register value.

The register to read is specified in ADDRESS field of SDP command. First device sends Report3 with security configuration followed by Report4 with bytes read at given address. If count is greater than 64 then multiple reports with report id 4 are sent until entire data requested by host is sent. The STATUS is either 0x12343412 for closed parts and 0x56787856 for open or field return parts.

Report1, Command, Host to Device:

1	Valid values for READ_REGISTER COMMAND, ADDRESS, FORMAT, DATA_COUNT

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host: first response report

4	Register Value

ID 4 bytes of data containing register value. If number of bytes requested is less than 4 then remaining bytes should be ignored by host.

Multiple reports of report id 4 are sent until entire data requested is sent

Report4, Response, Device to Host: Last response report

4	Register Value

ID 64 bytes of data containing register value. If number of bytes requested is less than 64 then remaining bytes should be ignored by host.

6.6.9.2.1.2 WRITE REGISTER

The transaction for command WRITE_REGISTER consists of the following reports: Report1 for command, Report3 for security configuration and Report4 for write status.

Host sends Report1 with WRITE_REGISTER command. The register to write is specified in ADDRESS field of SDP command of Report1, with FORMAT field set to data type (number of bits to write 8, 16 or 32) and value to write in DATA field of SDP command. Device writes the DATA to register address and returns WRITE_COMPLETE code using Report4 and security configuration using Report3 to complete the transaction.

Report1, Command, Host to Device:

1	Valid values for WRITE_REGISTER COMMAND, ADDRESS, FORMAT, DATA_COUNT and DATA

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	WRITE_COMPLETE (0x128A8A12) status

ID 64 bytes data with first 4 bytes to indicate write is completed with code 0x128A8A12. On failure device will report HAB error status.

6.6.9.2.1.3 WRITE_FILE

The transaction for command WRITE_FILE consists of following reports: Report1 for command-phase, Report2 for data-phase, Report3 for hab mode and Report4 to indicate data received in full.

The size of each Report2 is limited to 1024 bytes (limitation of USB HID protocol) hence multiple Report2 packets will be sent by host in data phase until entire data is transferred to device. Once entire data (DATA_COUNT bytes) is received then device sends report 3 with hab mode and report 4 with 0x88888888, indicating file download completed.

Report1, Host to Device:

1	Valid values for WRITE_FILE COMMAND, ADDRESS, DATA_COUNT

ID 16 byte SDP Command

==========Optional Begin==========

Host sends ERROR_STATUS command to query if HAB rejected the address

========== Optional End==========

Report2, Host to Device:

2	File data

ID Max 1024 bytes data per report

Report2, Host to Device:

2	File data

ID Max 1024 bytes data per report

Report3, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	COMPLETE (0x88888888) status

ID 64 bytes data with first 4 bytes to indicate file download has completed with code 0x88888888. On failure device will report HAB error status.

6.6.9.2.1.4 ERROR_STATUS

The transaction for SDP command ERROR_STATUS consists of three reports.

Report1 is used by host to send the command; device sends global error status in 4 bytes of Report4 after returning security configuration in Report3. When device receives ERROR_STATUS command it will return global error status that is updated for each command. This command is useful to find out if last command resulted in device error or succeeded.

Report1, Command, Host to Device:

1	ERROR_STATUS COMMAND

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	4 bytes Error status

ID first 4 bytes status in 64 bytes report 4

6.6.9.2.1.5 DCD WRITE

The SDP command DCD_WRITE is used by host to send multiple register writes in one shot. This command is provided to speed up the process of programming register writes such as to configure external RAM device.

The command goes with Report1 from host with COMMAND TYPE set to DCD_WRITE, ADDRESS which is used for temporary location of DCD data and DATA_COUNT to number of bytes sent in data out phase. In data phase host sends data for number of registers using Report2. Device completes the transaction with Report3 indicating security configuration and report 4 with WRITE_COMPLETE code 0x12828212.

Report1, Command, Host to Device:

1	DCD_WRITE COMMAND, ADDRESS, DATA_COUNT

ID 16 byte SDP Command

Report2, Data, Host to Device:

2	DCD binary data

ID Max 1024 bytes per report

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	WRITE_COMPLETE (0x128A8A12) status

ID 64 bytes report with first 4 bytes to indicate write is completed with code 0x128A8A12. On failure device will report HAB error status.

See Device Configuration Data (DCD) for DCD format description.

6.6.9.2.1.6 SKIP_DCD_HEADER

The SDP command SKIP_DCD_HEADER is used by host to inform device skip the DCD configuration within download image.

Normally, if the download image should be run on DDR, the DCD configuration data should be built-in the image. In case that host has issued DCD_WRITE to push DCD configuration data to device for DDR initialization, the image with DCD information will cause ROM initializes DDR twice and may cause initialization processing hung.

The SKIP_DCD_HEADER command inform device to skip the DCD configuration with the download image, and avoid this issue.

This command should typically be sent after JUMP_ADDRESS. The command is sent by host in command-phase of transaction using Report1, there is no data phase for this command. Device completes the transaction with Report3 indicating security configuration and report 4 with OK_ACK code 0x900DD009.

Report1, Command, Host to Device:

1	SKIP_DCD_HEADER

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	OK_ACK (0x900DD009)

6.6.9.2.1.7 JUMP ADDRESS

The SDP command JUMP_ADDRESS will be the last command host can send to the device, after this command device will jump to the address specified in the ADDRESS field of SDP command and start executing.

This command should typically follow after WRITE_FILE command. The command is sent by host in command-phase of transaction using Report1, there is no data phase for this command but device send status report3 to complete the transaction. And if HAB authentication fails then it will also send report 4 with HAB error status.

Report1, Command, Host to Device:

1	JUMP_ADDRESS COMMAND, ADDRESS

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

This report is sent by device only in case of an error jumping to the given address, device reports error in Report4, Response, Device to Host:

4	4 bytes HAB error status

ID 4 bytes status, 64 bytes report length

6.6.10 Recovery Devices

The Chip supports recovery devices. If primary boot device fails, boot ROM will try to boot from recovery device using one of ECSPI ports.

For enabling recovery device BOOT_CFG4[6] fuse must be set. Additionally Serial EEPROM fuses must be set as described in Serial ROM through SPI.

6.6.11 USB Low Power Boot

ROM supports USB Low Power Boot. This feature enables a device with dead or weak battery to power up and boot if the device is connected to a USB upstream port, no matter the upstream port is a USB charger or USB host/hub.

If a USB dedicated charger or host/hub charger are connected, as soon as the device is connected to the upstream port, a stable current (Max.1.5A) can be supplied by charger. If USB host/hub are connected, the maximal 1 00mA current is supplied to the device, the device should be able to power up to boot the image with less than 100mA.

If LPB_BOOT fuses are blown, the Chip will check if there is low power condition via GPIO_3 pad. If there is low power boot condition USB charger detection will be activated. If there is no USB charger, ROM will initialize USB as device and apply division factors on ARM, DDR, AXI and AHB root clocks based on LPB_BOOT fuses value (see the table below). Polarity of low power boot condition on GPIO_3 pad is set by BT _LPB_POLARITY fuse (see the figure below).

Table 6-66. USB Low Power Boot Frequencies

LPB_BOOT	Boot Frequencies=0	Boot Frequencies=1
00	ARM_CLK_ROOT=792MHz MMDC_CLK_ROOT=396MHz FABRIC_CLK_ROOT=396MHz AHB_CLK_ROOT=132MHz	ARM_CLK_ROOT=396MHz MMDC_CLK_ROOT=307MHz FABRIC_CLK_ROOT=307MHz AHB_CLK_ROOT=77MHz
01	ARM_CLK_ROOT=792MHz MMDC_CLK_ROOT=396MHz FABRIC_CLK_ROOT=396MHz AHB_CLK_ROOT=132MHz	ARM_CLK_ROOT=396MHz MMDC_CLK_ROOT=307MHz FABRIC_CLK_ROOT=307MHz AHB_CLK_ROOT=102MHz
10		
11	ARM_CLK_ROOT=198MHz MMDC_CLK_ROOT=99MHz FABRIC_CLK_ROOT=99MHz AHB_CLK_ROOT=49MHz	ARM_CLK_ROOT=99MHz MMDC_CLK_ROOT=76MHz FABRIC_CLK_ROOT=76MHz AHB_CLK_ROOT=38MHz

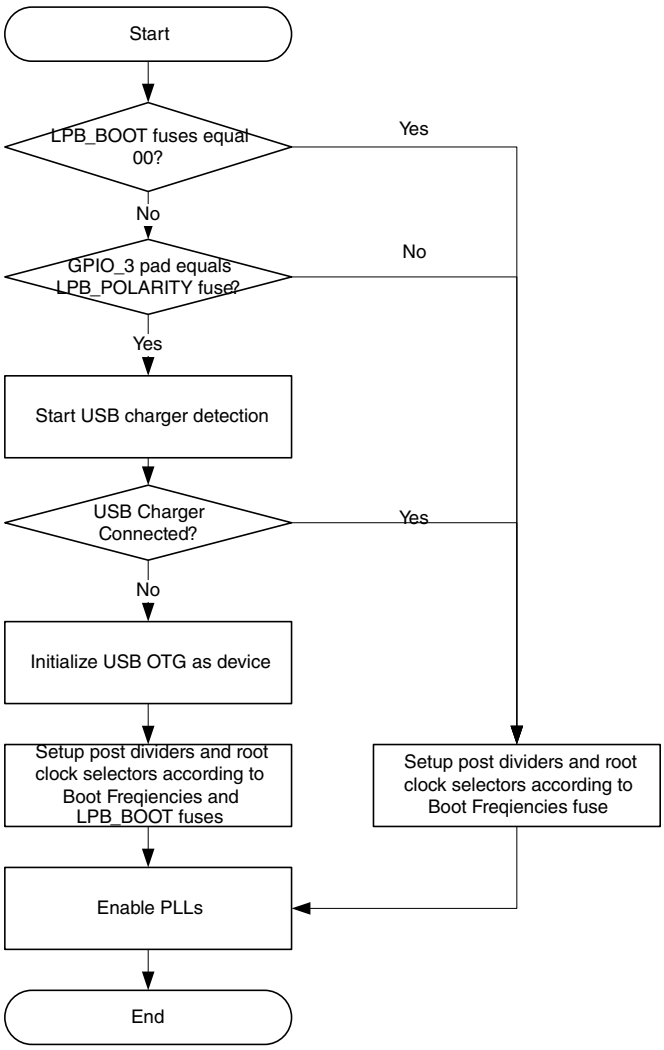


Figure 6-46. USB Low Power Boot Flow

6.6.12 SD/MMC Manufacture Mode

When internal boot and recover boot (if enabled) failed, SDMMC_MFG_DISABLE fuse bit isn't set, and EEPROM Recovery fuse bit is set, boot will go to SD/MMC manufacture mode before serial download mode. In manufacture mode, one bit bus width is used despite of the fuse setting.

In manufacture mode, SD or MMC card will be scanned on uSDHC1. If card is detected and valid boot image is found in card, then boot image will be loaded then executed. Pad of SD1_CD is used to detect whether card is inserted.

By default, SD/MMC manufacture mode is enabled, blow the fuse of DISABLE_SDMMC_MFG to disable it.

NOTE

Secondary boot is not supported on SD/MMC manufacture mode.

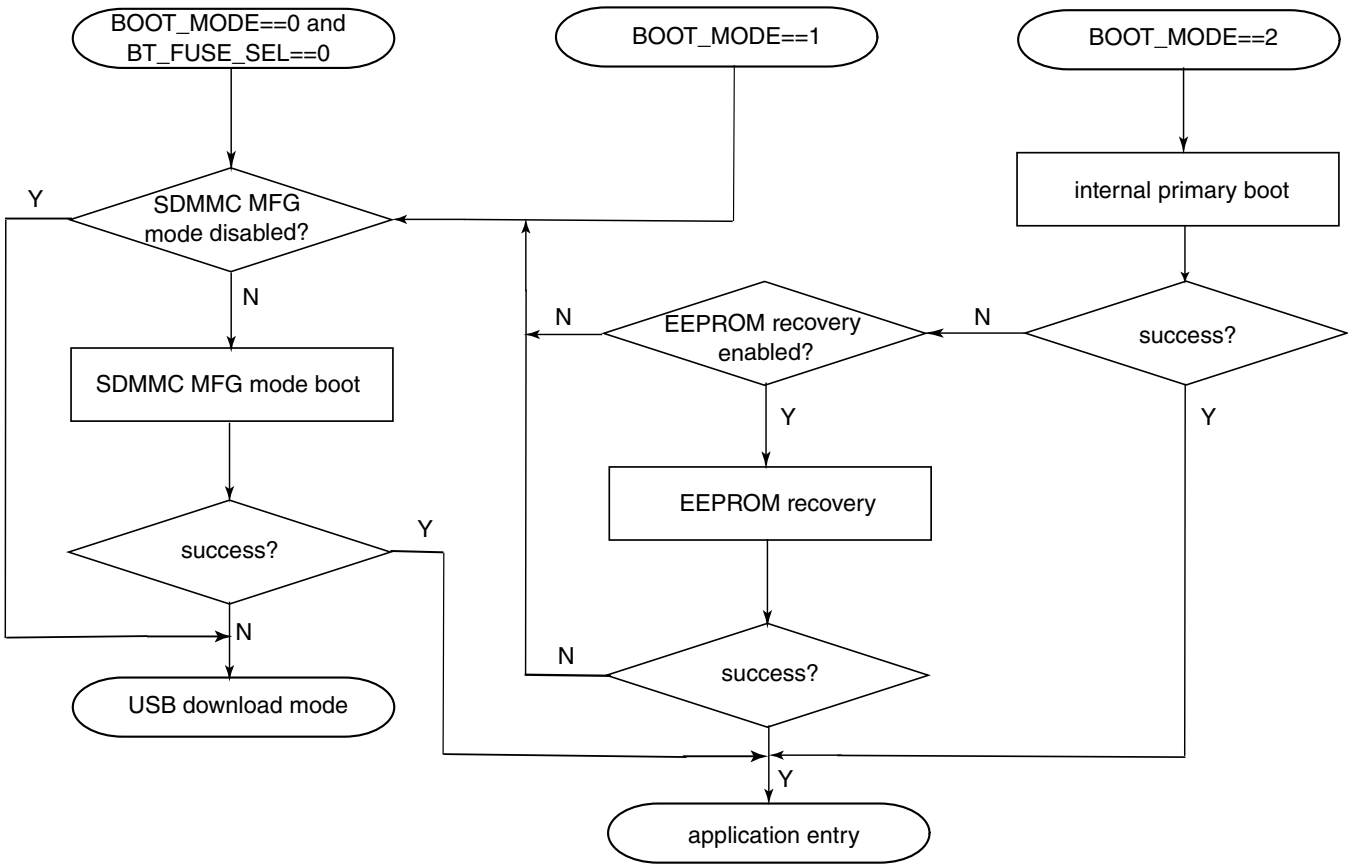


Figure 6-47. SD/MMC Manufacture boot flow

6.6.13 High Assurance Boot (HAB)

The High Assurance Boot (HAB) component of the ROM protects against the potential threat of attackers modifying areas of code or data in programmable memory to make it behave in an incorrect manner. The HAB also prevents attempts to gain access to features which should not be available.

The integration of the HAB feature with the ROM code ensures that Chip does not enter an operational state if the existing hardware security blocks have detected a condition that may be a security threat or areas of memory deemed to be important have been modified. The HAB uses RSA digital signatures to enforce these policies.

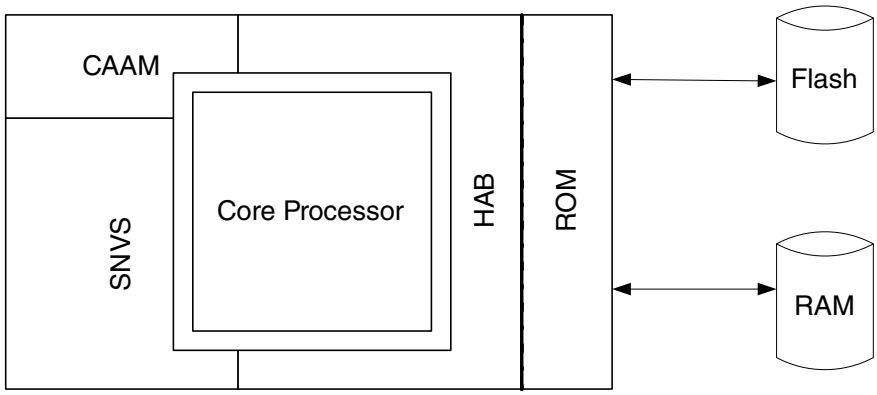


Figure 6-48. Secure Boot Components

The figure above illustrates the components used during a secure boot using HAB. The HAB interfaces with the SNVS to ensure the system security state is as expected. The HAB also makes use of CAAM hardware block to accelerate SHA-256 message digest operations performed during signature verifications and AES-128 operations for encrypted boot operations. The HAB also includes a software implementation of SHA-256 for cases where a hardware accelerator cannot be used. The RSA key sizes supported are 1024, 2048 and 3072 bits. The RSA signature verification operations are performed by a software implementation contained in the HAB library. The main features supported by HAB are:

- X.509 Public key certificate support
- CMS signature format support
- Proprietary encrypted boot support. Note that encrypted boot depends on the CAAM HW module. When CAAM is disabled (i.e. when the EXPORT_CONTROL fuse is blown) then encrypted boot is not available.

NOTE

NXP provides a reference Code Signing Tool (CST) for key generation, certificate generation and code signing for use with the HAB library. The CST can be found by searching for "IMX_CST_TOOL" at http://www.nxp.com.

NOTE

For further details on making use of the secure boot feature using HAB contact your local NXP representative.

6.6.13.1 HAB API Vector Table Addresses

For devices that perform a secure boot, the HAB library may be called by boot stages that execute after ROM code.

The RVT table contains the pointers to the HAB API functions and is located at 0x00000100.

NOTE

For additional information on secure boot including the HAB API, contact your local Freescale representative.

6.6.14 Boot Information for Software

To address the requirement that boot image may need to get the basic boot information when getting out the boot process, Boot Software Information (Boot_SW_Info) is provisioned by ROM.

The base address of Boot_SW_Info, i.e., SW Information Pointer is been recorded at 0x1E8 in ROM. Software should read 0x1E8 to get the base address of Boot_SW_Info, and then parse the Boot_SW_Info content to get the boot information.

Table 6-67. Boot_SW_Info Structure

Offset	Byte3	Byte2	Byte1	Byte0
0x0	Reserved	Boot Device Type	Boot Device Instance	Reserved
0x4	ARM core frequency(in Hz)			
0x8	AXI bus frequency(in Hz)			
0xC	DDR frequency(in Hz)			
0x10	GPT1 input clock frequency(in Hz)			
0x14	Reserved			
0x18				
0x1C				

NOTE

Boot ROM set GPT1 in free-running mode with 32 kHz input clock.

Boot Device Type Mapping:

- 0x1 - SD Card or eSD chip
- 0x2 - MMC Card or eMMC chip
- 0x3 - NAND chip
- 0x4 - QSPI NOR flash

- 0x5 - Parallel NOR flash
- 0x6 - SPI NOR flash

Boot Device Instance: The instance index of the boot device, start from 0.

Chapter 7
Interrupts and DMA Events

7.1 Interrupts and DMA Events

7.1.1 Overview

This chapter discusses the assignments of interrupts from the ARM domain in A7 Interrupts, CM4 interrupts, and from DMA events in SDMA event mapping

7.1.2 A7 Interrupts

The Global Interrupt Controller (GIC) collects up to 128 interrupt requests from all the chip sources and provides an interface to the Cortex A7 CPU.

The first 32 interrupts are used for interrupts that are private to the CPUs interface. These interrupts are not included in the table below. All interrupts besides the private CPU are also hooked up to the GPC in the same order.

Each interrupt can be configured as a normal or a secure interrupt. Software force registers and software priority masking are also supported. The following table describes the A7 interrupt sources. The following table describes the A7 interrupt sources.

Table 7-1. ARM Domain Interrupt Summary

IRQ	Module	Interrupt Description
0	GPR IRQ	Used to notify cores on exception condition while boot
1	DAP	DAP Interrupt
2	SDMA	AND of all 48 SDMA interrupts (events) from all the channels
3	DBGMON	DBGMON Sync Interrupt
4	SNVS WRAPPER	ON-OFF button press shorter than 5 seconds (pulse event)
5	LCDIF	LCDIF Sync Interrupt
6	SIM2	SIM Interrupt

Table continues on the next page...

Table 7-1. ARM Domain Interrupt Summary (continued)

IRQ	Module	Interrupt Description
7	CSI	CSI Interrupt
8	PXP	PXP Interrupt
9	Reserved	Reserved
10	WDOG3	Watchdog Timer reset
11	HS	Hardware Semaphore Interrupt Request
12	APBHDMA	GPMI operation channel 0 description complete interrupt
13	EIM	EIM Interrupt
14	BCH	BCH operation complete interrupt
15	GPMI	GPMI operation TIMEOUT ERROR interrupt
16	UART6	UART-6 ORed interrupt
17	FTM1	Flex Timer1 Fault / Counter / Channel interrupt
18	FTM2	Flex Timer2 Fault / Counter / Channel interrupt
19	SNVS_HP_WRAPPER	SRTC Consolidated Interrupt. Non TZ.
20	SNVS_HP_WRAPPER	SRTC Security Interrupt. TZ.
21	CSU	CSU Interrupt Request. Indicates to the processor that one or more alarm inputs were asserted
22	USDHC1	uSDHC1 Enhanced SDHC Interrupt Request
23	USDHC2	uSDHC2 Enhanced SDHC Interrupt Request
24	USDHC3	uSDHC3 Enhanced SDHC Interrupt Request
25	MIPI_CSI	MIPI CSI interrupt
26	UART1	UART-1 ORed interrupt
27	UART2	UART-2 ORed interrupt
28	UART3	UART-3 ORed interrupt
29	UART4	UART-4 ORed interrupt
30	UART5	UART-5 ORed interrupt
31	eCSPI1	eCSPI1 interrupt request line to the core.
32	eCSPI2	eCSPI2 interrupt request line to the core.
33	eCSPI3	eCSPI3 interrupt request line to the core.
34	eCSPI4	eCSPI4 interrupt request line to the core.
35	I2C1	I2C-1 Interrupt
36	I2C2	I2C-2 Interrupt
37	I2C3	I2C-3 Interrupt
38	I2C4	I2C-4 Interrupt
39	RDC	RDC interrupt
40	USB	USB Host interrupt
41	MIPI_DSI	MIPI CSI Interrupt
42	-	Reserved
43	USB	USB OTG1 core interrupt
44	USB	USB OTG1 wakeup interrupt
45	-	Reserved

Table continues on the next page...

Table 7-1. ARM Domain Interrupt Summary (continued)

IRQ	Module	Interrupt Description
46	PXP	PXP interrupt
47	SCTR	ISO7816IP Interrupt
48	SCTR	ISO7816IP Interrupt
49	Analog	TempSensor (Temperature low alarm).
50	SAI3	SAI3 Receive / Transmit Interrupt
51	Analog	Brown-out event on either analog regulators.
52	GPT4	OR of GPT Rollover interrupt line, Input Capture 1 and 2 lines, Output Compare 1, 2, and 3 Interrupt lines
53	GPT3	OR of GPT Rollover interrupt line, Input Capture 1 and 2 lines, Output Compare 1, 2, and 3 Interrupt lines
54	GPT2	OR of GPT Rollover interrupt line, Input Capture 1 and 2 lines, Output Compare 1, 2, and 3 Interrupt lines
55	GPT1	OR of GPT Rollover interrupt line, Input Capture 1 and 2 lines, Output Compare 1, 2, and 3 Interrupt lines
56	GPIO1	Active HIGH Interrupt from INT7 from GPIO
57	GPIO1	Active HIGH Interrupt from INT6 from GPIO
58	GPIO1	Active HIGH Interrupt from INT5 from GPIO
59	GPIO1	Active HIGH Interrupt from INT4 from GPIO
60	GPIO1	Active HIGH Interrupt from INT3 from GPIO
61	GPIO1	Active HIGH Interrupt from INT2 from GPIO
62	GPIO1	Active HIGH Interrupt from INT1 from GPIO
63	GPIO1	Active HIGH Interrupt from INT0 from GPIO
64	GPIO1	Combined interrupt indication for GPIO1 signal 0 throughout 15
65	GPIO1	Combined interrupt indication for GPIO1 signal 16 throughout 31
66	GPIO2	Combined interrupt indication for GPIO2 signal 0 throughout 15
67	GPIO2	Combined interrupt indication for GPIO2 signals 16 throughout 31
68	GPIO3	Combined interrupt indication for GPIO3 signal 0 throughout 15
69	GPIO3	Combined interrupt indication for GPIO3 signals 16 throughout 31
70	GPIO4	Combined interrupt indication for GPIO4 signal 0 throughout 15
71	GPIO4	Combined interrupt indication for GPIO4 signal 16 throughout 31
72	GPIO5	Combined interrupt indication for GPIO5 signals 0 throughout 15
73	GPIO5	Combined interrupt indication for GPIO5 signals 16 throughout 31
74	GPIO6	Combined interrupt indication for GPIO6 signals 0 throughtout 15
75	GPIO6	Combined interrupt indication for GPIO6 signals 16 throughtout 31
76	GPIO7	Combined interrupt indication for GPIO7 signals 0 throughout 15
77	GPIO7	Combined interrupt indication for GPIO7 signals 16 throughout 31
78	WDOG1	Watchdog Timer reset
79	WDOG2	Watchdog Timer reset
80	KPP	Keypad Interrupt
81	PWM1	Cumulative interrupt line. OR of Rollover Interrupt line, Compare Interrupt line and FIFO Waterlevel crossing interrupt line

Table continues on the next page...

Table 7-1. ARM Domain Interrupt Summary (continued)

IRQ	Module	Interrupt Description
82	PWM2	Cumulative interrupt line. OR of Rollover Interrupt line, Compare Interrupt line and FIFO Waterlevel crossing interrupt line
83	PWM3	Cumulative interrupt line. OR of Rollover Interrupt line, Compare Interrupt line and FIFO Waterlevel crossing interrupt line
84	PWM4	Cumulative interrupt line. OR of Rollover Interrupt line, Compare Interrupt line and FIFO Waterlevel crossing interrupt line
85	CCM	CCM, Interrupt Request 1
86	CCM	CCM, Interrupt Request 2
87	GPC	GPC Interrupt Request 1
88	MU	Interrupt to A7
89	SRC	SRC interrupt request
90	SIM1	Sim Interrupt
91	RTIC	RTIC Interrupt
92	CPU	Performance Unit Interrupts from ARM CPU platform (interrnally: PMUIRQ[0:1])
93	CPU	CTI trigger outputs (internal: nCTIIRQ[0:1])
94	CCM SRC GPC	Combined CPU wdog interrupts (4x) out of SRC.
95	SAI1	SAI1 Receive / Transmit Interrupt
96	SAI2	SAI2 Receive / Transmit Interrupt
97	MU	Interrupt to M4
98	ADC1	ADC-1 Interrupt
99	ADC2	ADC-2 Interrupt
100	-	Reserved
101	-	Reserved
102	-	Reserved
103	-	Reserved
104	TPR IRQ	TPR IRQ
105	CAAM WRAPPER	CAAM interrupt queue for JQ
106	CAAM WRAPPER	CAAM interrupt queue for JQ
107	QSPI	QSPI Interrupt
108	TZASC1	TZASC (PL380) interrupt
109	WDOG4	Watchdog Timer reset
110	FLEXCAN1	FlexCAN1 Interrupt
111	FLEXCAN2	FlexCAN2 Interrupt
112	PERFMON1	General interrupt
113	PERFMON2	General interrupt
114	CAAM_WRAPPER	CAAM interrupt queue for JQ
115	CAAM_WRAPPER	Recoverable error interrupt
116	HS	HS Interrupt Request
117	-	Reserved
118	ENET1	MAC 0 Receive / Trasmit Frame / Buffer Done

Table continues on the next page...

Table 7-1. ARM Domain Interrupt Summary (continued)

IRQ	Module	Interrupt Description
119	ENET1	MAC 0 Receive / Trasmit Frame / Buffer Done
120	ENET1	MAC 0 IRQ
121	ENET1	MAC 0 1588 Timer Interrupt – synchronous
122	-	Reserved
123	-	Reserved
124	-	Reserved
125	-	Reserved
126	UART7	UART-7 ORed interrupt
127	-	Reserved

7.1.3 CM4 interrupts

The Nested Vectored Interrupt Controller (NVIC) collects up to 128 interrupt requests from all chip sources and provides an interface to the Cortex M4 Core.

The following table describes the M4 interrupt sources.

Table 7-2. ARM Domain Interrupt Summary

IRQ	Module	Interrupt Description
0	GPR IRQ	Used to notify cores on exception condition while boot
1	DAP	DAP Interrupt
2	SDMA	AND of all 48 SDMA interrupts (events) from all the channels
3	DBGMON	DBGMON Sync Interrupt
4	SNVS WRAPPER	ON-OFF button press shorter than 5 seconds (pulse event)
5	LCDIF	LCDIF Sync Interrupt
6	SIM2	SIM Interrupt
7	CSI	CSI Interrupt
8	PXP	PXP Interrupt 0
9	Reserved	Reserved
10	WDOG3	Watchdog Timer reset
11	HS	HS Interrupt Request
12	APBHDMA	GPMI operation channel 0 description complete interrupt
13	EIM	EIM Interrupt
14	BCH	BCH operation complete interrupt
15	GPMI	GPMI operation TIMEOUT ERROR interrupt
16	UART6	UART-6 ORed interrupt
17	FTM1	Flex Timer1 Fault / Counter / Channel interrupt
18	FTM2	Flex Timer2 Fault / Counter / Channel interrupt

Table continues on the next page...

Table 7-2. ARM Domain Interrupt Summary (continued)

IRQ	Module	Interrupt Description
19	SNVS_HP_WRAPPER	SRTC Consolidated Interrupt. Non TZ.
20	SNVS_HP_WRAPPER	SRTC Security Interrupt. TZ.
21	CSU	CSU Interrupt Request. Indicates to the processor that one or more alarm inputs were asserted
22	USDHC1	uSDHC1 Enhanced SDHC Interrupt Request
23	USDHC2	uSDHC2 Enhanced SDHC Interrupt Request
24	USDHC3	uSDHC3 Enhanced SDHC Interrupt Request
25	MIPI_CSI	MIPI CSI interrupt
26	UART1	UART-1 ORed interrupt
27	UART2	UART-2 ORed interrupt
28	UART3	UART-3 ORed interrupt
29	UART4	UART-4 ORed interrupt
30	UART5	UART-5 ORed interrupt
31	eCSPI1	eCSPI1 interrupt request line to the core.
32	eCSPI2	eCSPI2 interrupt request line to the core.
33	eCSPI3	eCSPI3 interrupt request line to the core.
34	eCSPI4	eCSPI4 interrupt request line to the core.
35	I2C1	I2C-1 Interrupt
36	I2C2	I2C-2 Interrupt
37	I2C3	I2C-3 Interrupt
38	I2C4	I2C-4 Interrupt
39	RDC	RDC interrupt
40	-	Reserved
41	MIPI_DSI	MIPI CSI Interrupt
42	-	Reserved
43	USB	USB OH2 OTG
44	USB	USB OTG1 Interrupt
45	-	Reserved
46	PXP	PXP interrupt 1
47	SCTR	ISO7816IP Interrupt
48	SCTR	ISO7816IP Interrupt
49	Analog	TempSensor (Temperature low alarm).
50	SAI3	SAI3 Receive / Transmit Interrupt
51	Analog	Brown-out event on either analog regulators.
52	GPT4	OR of GPT Rollover interrupt line, Input Capture 1 and 2 lines, Output Compare 1, 2, and 3 Interrupt lines
53	GPT3	OR of GPT Rollover interrupt line, Input Capture 1 and 2 lines, Output Compare 1, 2, and 3 Interrupt lines
54	GPT2	OR of GPT Rollover interrupt line, Input Capture 1 and 2 lines, Output Compare 1, 2, and 3 Interrupt lines

Table continues on the next page...

Table 7-2. ARM Domain Interrupt Summary (continued)

IRQ	Module	Interrupt Description
55	GPT1	OR of GPT Rollover interrupt line, Input Capture 1 and 2 lines, Output Compare 1, 2, and 3 Interrupt lines
56	GPIO1	Active HIGH Interrupt from INT7 from GPIO
57	GPIO1	Active HIGH Interrupt from INT6 from GPIO
58	GPIO1	Active HIGH Interrupt from INT5 from GPIO
59	GPIO1	Active HIGH Interrupt from INT4 from GPIO
60	GPIO1	Active HIGH Interrupt from INT3 from GPIO
61	GPIO1	Active HIGH Interrupt from INT2 from GPIO
62	GPIO1	Active HIGH Interrupt from INT1 from GPIO
63	GPIO1	Active HIGH Interrupt from INT0 from GPIO
64	GPIO1	Combined interrupt indication for GPIO1 signal 0 throughout 15
65	GPIO1	Combined interrupt indication for GPIO1 signal 16 throughout 31
66	GPIO2	Combined interrupt indication for GPIO2 signal 0 throughout 15
67	GPIO2	Combined interrupt indication for GPIO2 signals 16 throughout 31
68	GPIO3	Combined interrupt indication for GPIO3 signal 0 throughout 15
69	GPIO3	Combined interrupt indication for GPIO3 signals 16 throughout 31
70	GPIO4	Combined interrupt indication for GPIO4 signal 0 throughout 15
71	GPIO4	Combined interrupt indication for GPIO4 signal 16 throughout 31
72	GPIO5	Combined interrupt indication for GPIO5 signals 0 throughout 15
73	GPIO5	Combined interrupt indication for GPIO5 signals 16 throughout 31
74	GPIO6	Combined interrupt indication for GPIO6 signals 0 throughtout 15
75	GPIO6	Combined interrupt indication for GPIO6 signals 16 throughtout 31
76	GPIO7	Combined interrupt indication for GPIO7 signals 0 throughout 15
77	GPIO7	Combined interrupt indication for GPIO7 signals 16 throughout 31
78	WDOG1	Watchdog Timer reset
79	WDOG2	Watchdog Timer reset
80	KPP	Keypad Interrupt
81	PWM1	Cumulative interrupt line. OR of Rollover Interrupt line, Compare Interrupt line and FIFO Waterlevel crossing interrupt line
82	PWM2	Cumulative interrupt line. OR of Rollover Interrupt line, Compare Interrupt line and FIFO Waterlevel crossing interrupt line
83	PWM3	Cumulative interrupt line. OR of Rollover Interrupt line, Compare Interrupt line and FIFO Waterlevel crossing interrupt line
84	PWM4	Cumulative interrupt line. OR of Rollover Interrupt line, Compare Interrupt line and FIFO Waterlevel crossing interrupt line
85	CCM	CCM, Interrupt Request 1
86	CCM	CCM, Interrupt Request 2
87	GPC	GPC Interrupt Request 1
88	MU	Interrupt to A7
89	SRC	SRC interrupt request
90	SIM1	Sim Interrupt

Table continues on the next page...

Table 7-2. ARM Domain Interrupt Summary (continued)

IRQ	Module	Interrupt Description
91	RTIC	RTIC Interrupt
92	CPU	Performance Unit Interrupts from Cheetah (interrnally: PMUIRQ[0])
92	CPU	Performance Unit Interrupts from Cheetah (interrnally: PMUIRQ[1])
93	CPU	CTI trigger outputs (internal: nCTIIRQ[0])
93	CPU	CTI trigger outputs (internal: nCTIIRQ[1])
94	CCM SRC GPC	Combined CPU wdog interrupts (4x) out of SRC.
95	SAI1	SAI1 Receive / Transmit Interrupt
96	SAI2	SAI2 Receive / Transmit Interrupt
97	MU	Interrupt to M4
98	ADC1	ADC-1 Interrupt
99	ADC2	ADC-2 Interrupt
100	-	Reserved
101	-	Reserved
102	-	Reserved
103	-	Reserved
104	TPR IRQ	TPR IRQ
105	CAAM WRAPPER	CAAM interrupt queue for JQ
106	CAAM WRAPPER	CAAM interrupt queue for JQ
107	QSPI	QSPI Interrupt
108	TZASC1	TZASC (PL380) interrupt
109	WDOG4	Watchdog Timer reset
110	FLEXCAN1	FlexCAN1 Interrupt
111	FLEXCAN2	FlexCAN2 Interrupt
112	PERFMON1	General interrupt
113	PERFMON2	General interrupt
114	CAAM_WRAPPER	CAAM interrupt queue for JQ
115	CAAM_WRAPPER	Recoverable error interrupt
116	HS	HS Interrupt Request
117	-	Reserved
118	ENET1	MAC 0 Receive / Trasmit Frame / Buffer Done
119	ENET1	MAC 0 Receive / Trasmit Frame / Buffer Done
120	ENET1	MAC 0 IRQ
121	ENET1	MAC 0 1588 Timer Interrupt – synchronous
122	-	Reserved
123	-	Reserved
124	-	Reserved
125	-	Reserved
126	UART7	UART-7 ORed interrupt
127	-	Reserved

7.1.4 SDMA event mapping

The following table shows the DMA request signals for peripherals in the chip.

Table 7-3. SDMA event mapping

SDMA	Module	Description
0	ECSPI1	eCSPI1 Rx request
1	ECSPI1	eCSPI1 Tx request
2	ECSPI2	eCSPI2 Rx request
3	ECSPI2	eCSPI2 Tx request
4	ECSPI3	eCSPI3 Rx request
5	ECSPI3	eCSPI3 Tx request
6	ECSPI4	eCSPI4 Rx request
7	ECSPI4	eCSPI4 Tx request
8	SAI1	SAI-1 receive 1 DMA request
9	SAI1	SAI-1 receive 1 DMA request
10	SAI2	SAI-3 transmit 1 DMA request
11	SAI2	SAI-3 transmit 1 DMA request
12	SAI3	eCSPI3 Rx request
13	SAI3	eCSPI3 Tx request
14	IOMUX	External DMA from pad through IOMUX #1
15	IOMUX	External DMA from pad through IOMUX #2
16	ADC1	ADC DMA request
17	ADC2	ADC DMA request
18	I2C1 | SIM1	I2C1 DMA event | SIM1 receive DMA request (IOMUXC GPR0[0])
19	I2C2 | SIM1	I2C2 DMA event | SIM1 receive DMA request (IOMUXC GPR0[1])
20	I2C3 | SIM2	I2C3 DMA event | SIM2 receive DMA request (IOMUXC GPR0[2])
21	I2C4 | SIM2	I2C4 DMA event | SIM2 receive DMA request (IOMUXC GPR0[3])
22	UART1	Rx FIFO
23	UART1	Tx FIFO
24	UART2	Rx FIFO
25	UART2	Tx FIFO
26	UART3	Rx FIFO
27	UART3	Tx FIFO
28	UART4	Rx FIFO
29	UART4	Tx FIFO
30	UART5	Rx FIFO
31	UART5	Tx FIFO

Table continues on the next page...

Table 7-3. SDMA event mapping (continued)

SDMA	Module	Description
32	UART6	Rx FIFO
33	UART6	Tx FIFO
34	UART7	Rx FIFO
35	UART7	Tx FIFO
36	QSPI1	QSPI DMA Tx request
37	QSPI1	QSPI DMA Rx request
38	GPT1	GPT1 counter event
39	GPT2	GPT2 counter event
40	GPT3 | FTM1	GPT3 counter event | Flextimer1 8 channel DMA request (IOMUXC GPR0[5])
41	GPT4 | FTM2	GPT4 counter event | Flextimer2 8 channel DMA request (IOMUXC GPR0[6])
42	PXP	PXP DMA Event
43	CSI	CSI DMA Event
44	LCDIF	LCDIF DMA Event
45	ENET1	ENET1 1588 Event0 out
46	-	Reserved
47	ENET1	ENET1 1588 Event0 out

As shown in the table, some of the events are an output of a mux of two signals or triggers. The select of this mux is controlled by the general purpose registers in IOMUXC.

7.2 Smart Direct Memory Access Controller (SDMA)

7.2.1 Overview

The Smart Direct Memory Access (SDMA) controller offers highly-competitive DMA features combined with software-based virtual-DMA flexibility. It enables data transfers between peripheral I/O devices and internal/external memories.

The SDMA controller helps maximize system performance by off-loading the ARM core in dynamic data routing.

7.2.1.1 Block Diagram

The figure below shows a block diagram of the SDMA controller. It includes the custom RISC core along with its RAM, ROM, DMA units, and the scheduler.

The SDMA core executes short routines that perform DMA transfers; these routines are called *scripts*. The SDMA core interfaces to its own memory via the SDMA system bus. The SDMA system bus supports a 32-bit data path and a 16-bit address bus. The system bus datapath is used for both 16-bit instruction (program) memory access and 32-bit data access. DMA units interface to the core via the Functional Unit Bus and use dedicated registers to perform DMA transfers.

The SDMA memory contains a ROM and a RAM. The ROM contains startup scripts (for example, boot code) and other common utilities, which are referenced by the scripts that reside in the RAM. The internal RAM is divided into a context area and a script area (more details about this mapping are available in Instruction Memory Map and Data Memory Map).

Every transfer channel requires one context area to keep the contents of all the core and unit registers while inactive. Channel scripts are downloaded into the internal RAM by the SDMA using a dedicated channel that is started during the boot sequence. Downloads are invoked using commands and pointers provided by the ARM platform. Every channel contains a corresponding channel script located in RAM and/or ROM that can be reconfigured independently as-needed. Channel scripts can be stored in an external memory and downloaded when needed. The SDMA can be configured with any mixture of scripts to enable an endless combination of supported services.

The scheduler monitors and detects DMA requests, mapping them to channels, and mapping individual channels to a pre-configured priority. At any given point, the scheduler presents the highest priority channel that requires service to the SDMA core. A special SDMA core instruction is used to "conditionally yield" the current channel being executed to an eligible channel that requires service. If (and only if) there is an eligible channel pending, will the current channel execution be preempted.

There are two yield instructions that differently determine the eligible channels: In the first version, eligible channels are pending channels with a strictly higher priority than the current channel priority. In the second version (yieldge), eligible channels are pending channels with a priority that is greater or equal to the current channel priority. The scheduler detects devices that need service through its 48 DMA request inputs. After a request is detected, the scheduler determines the channel(s) that is (are) triggered by this request and marks it (them) as pending in the "Channel Pending (EP)" register. The priorities of all the pending channels are continuously evaluated in order to update the highest pending priority. The channel pending flag is cleared by the channel script when the transfer has completed.

The ARM platform control block contains the control registers used to configure the 32 individual channels. There are 48 Channel Enable registers, and every register maps one DMA request to any desired combination of channels. The 32 Priority registers are used to assign a programmable 1-of-7 level priority to every possible channel. This block also contains all other control registers that the ARM platform can access.

The 48 DMA requests that are connected to the scheduler come from a variety of sources. The "receive register full" and "transmit register empty" signals found in the UART and USB ports are typical examples of DMA requests that can be connected to the SDMA. These requests can be used to trigger a specific SDMA channel, or several channels.

There is an OnCE compatible debug port for product development. The OnCE includes support for setting breakpoints, single-step and trace, and register dump capability. In addition, all memory locations are accessible from the debug port.

7.2.1.2 Features

The following are the SDMA features:

- Multi-channel DMA supporting up to 32 time-division multiplexed DMA channels
- Hardware or software driven triggers for each channel
- 48 hardware driven triggers that can be mapped to any channel.
- Memory accesses including linear addressing, FIFO addressing and 2D addressing
- Fast context-switching with two-level, priority-based preemptive multi-tasking
- 16-bit instruction-set micro-RISC engine (the SDMA core)
- Two DMA units with some or all the following features:
  - Auto-flush and prefetch capability
  - Flexible address management (increment, decrement, and no address changes on source and destination address)
  - Misaligned data-transfer support
  - Uni-directional and bi-directional flows (copy mode)
  - Up to eight-word buffers for configurable burst transfers
- Support of byte-swapping
- An available API and library of scripts
- Little-Endian and Big-Endian modes
- Hardware handshakes for low-power entry sequence
- Security support to lock contents of the SDMA script RAM.
- 4-Kbyte ROM containing startup scripts (for example, boot code) and other common utilities that can be referenced by RAM-located scripts
- 8-Kbyte RAM area is divided into a processor context area and a code space area used to store channel scripts that are downloaded from the system memory

- Debug support, including a OnCE port, real-time monitors, and embedded cross-trigger events
- Supported clock frequencies in process:
  - Configurable clock options for the SDMA core and the ARM platform DMA units
    - 1:2 ratio with maximum of SDMA core running at ARM platform Peripheral Bus speed and DMA running at max DMA frequency.
    - 1:1 ratio when both SDMA core and ARM platform DMA clocks are set to the ARM platform Peripheral Bus speed.
- Peripheral bus interface for configuration register programming by the ARM platform
- The SDMA RISC engine (arithmetic and logic operations), which is referred to as the "SDMA core."
- An internal peripheral bus connected to the Shared Peripherals Bus Interface (SPBA) that enables access to up to 14 shared peripherals. SDMA supports 32-bit accesses to word peripherals and 16-bit accesses to half-word peripherals.
- The peripheral DMA unit that is hooked-up to the ARM platform Crossbar Switch to service ARM peripherals
- The burst DMA unit is able to perform burst accesses to the external memory
- All the DMA units are 32-bit AHB masters. They are connected to different buses, thus allowing concurrent accesses.

7.2.2 External Signals

The table found here describes the external signals of SDMA.

7.2.3 Clocks

The following table describes the clock sources for SDMA. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information. For functional information regarding module clocks, see SDMA Clocks and Low Power Modes.

Table 7-4. SDMA Clocks

Clock name	Clock Root	Description
events_sync_clk (clk)	ahb_clk_root	ARM peripheral / events clock
ips_hostctrl_clk	ipg_clk_root	Host control clock
ap_ahb_clk	ahb_clk_root	ARM platform bus clock
core_clk	ipg_clk_root	Module / Core clock
tck	-	JTAG access clock

7.2.4 Functional Description

The figure below shows the SDMA topology, and is composed of the following components:

- SDMA Core (SDMA Core)
- SDMA Scheduler (Scheduler)
- Functional Units:
  - Burst DMA (Burst DMA Unit)
  - Peripheral DMA (Peripheral DMA Unit)
- ARM platform Control for ARM control register access.
- Internal RAM and ROM Memory (SDMA Programming Model)
- OnCE debug Port (The OnCE Controller)

The functional unit bus provides access by the SDMA core to the DMA units. The system bus provides access to SDMA internal memory and also supports up to 14 peripherals.

7.2.4.1 SDMA Core

The SDMA core is a customized RISC-like processor that is specifically developed to control DMA units and perform L1 tasks like byte-stuffing or framing.

The SDMA core incorporates on-chip debug capability using the OnCE.

The SDMA core is based on a 32-bit register architecture with 16-bit instructions. There are eight general purpose 32-bit registers, four flags (T, LM, SF, and DF), and four PCU registers (PC, RPC, SPC, and EPC) that can address 16,384 16-bit instructions.

7.2.4.1.1 SDMA Core Structure

The figure found here shows the structure of the SDMA core. It also shows the different registers, calculation resources, and possible data movements.

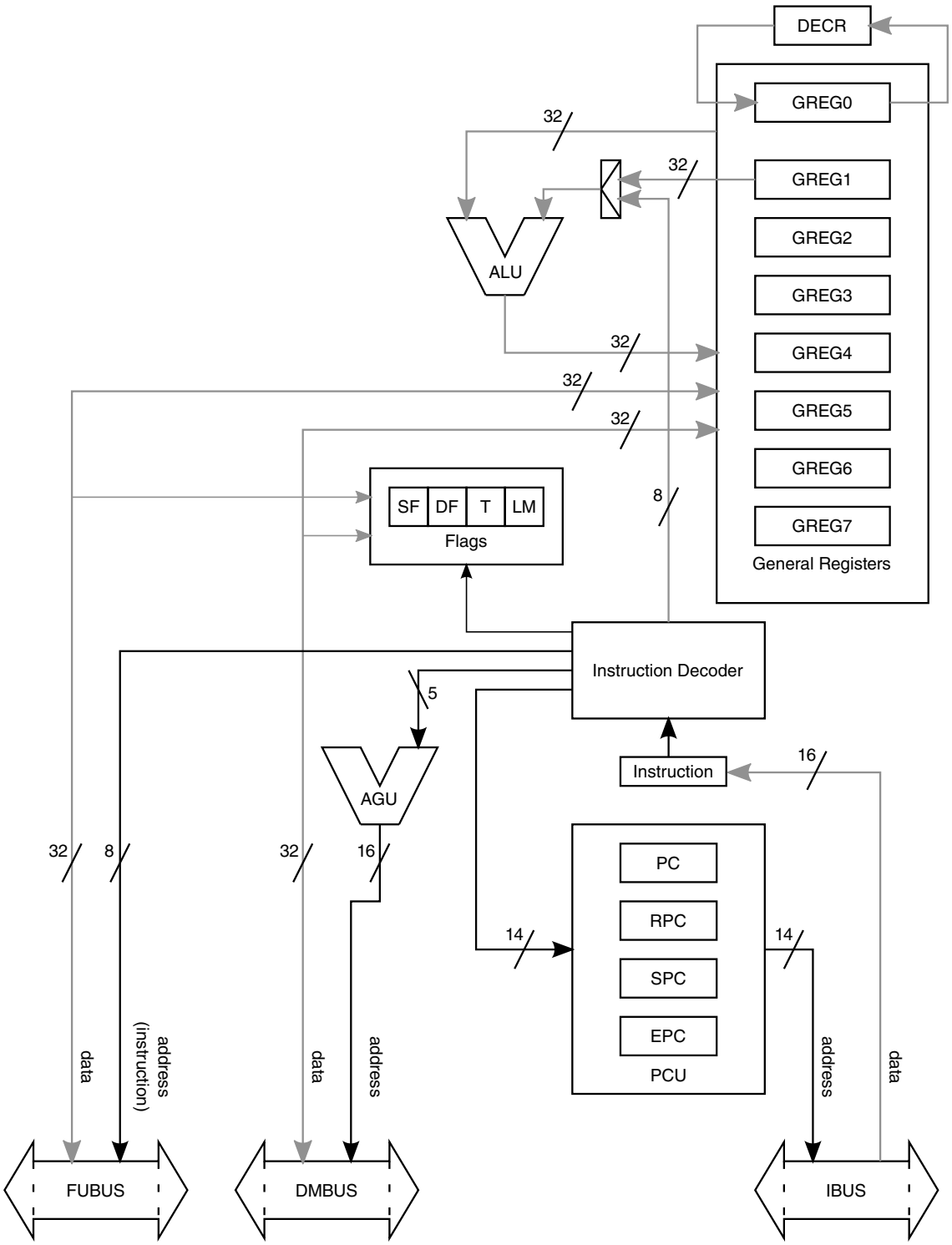


Figure 7-1. SDMA Core

- The Program Control Unit (PCU) is described in Program Control Unit (PCU). It handles the state of the core and generates the instruction fetch addresses. Instructions are retrieved from the Instruction Bus (IBUS) and stored in the SDMA

core instruction register prior to their decoding. The PCU contains the following registers:
    - The Program Counter (PC) contains the address of the current instruction.
    - The Return Program Counter (RPC) contains the address of the instruction that follows a jump to the subroutine.
    - The Start Program Counter (SPC) contains the address of the first instruction of the current hardware loop.
    - End Program Counter (EPC) contains the address of the last instruction of the current hardware loop.
- The other core registers are the general purpose registers (GREGn) and the flags.
    - The general purpose registers can be used to hold data and addresses. They can be loaded with immediate values (for example, 8-bit data that are encoded in the instruction), results of calculations that were performed with the ALU, 32-bit data that comes from the memory or peripherals via the Data Memory Bus (DMBUS), 32-bit data that comes from the DMAs via the Functional Units Bus (FUBUS) or another general purpose register. Their content can be the operands of the ALU, the data to send on either bus (DMBUS or FUBUS), or a pointer to memory (DMBUS address).
    - The general register 0 (GREG0) is also the hardware loop counter. In hardware loops, it cannot be used for any other purpose. This register uses a dedicated decrement unit (DECR) shown in Figure 7-1.
    - The flags reflect the status of operations:
        - SF and DF are set when the last load or store on either bus (FUBUS or DMBUS) received an error response.
        - LM is set when the core is executing instructions inside a hardware loop.
        - T is set when the ALU operation result was 0 or the loop counter reaches 0 (the latter is preponderant when an ALU operation is the last instruction of a hardware loop).
- The ALU has two operands: any general register and either a second general register or an immediate value. The result is always stored into the first general register. A NOP function can be utilized by moving a register's contents into itself (For example, the instruction: mov R0,R0).
- The 16-bit instructions are fetched via the instruction bus (IBUS) whose address is driven by the PC. The SDMA RAM and ROM are visible to the core as 16-bit devices through this interface.
- The memory (RAM and ROM), memory mapped registers, and external peripherals are accessed via the DMBUS. The address is always taken from a general register whose content is added to a 5-bit immediate value. This is the only available addressing mode. The DMBUS is a 32-bit data bus. Except for the peripherals that

are external to the SDMA, the address accuracy is the 32-bit word (for example, adding 1 to an address points to the next word, not the next byte).
- The functional units are accessed via the FUBUS connection. The data is exchanged with any general register, but the address (which in fact is the instruction and the selector of the functional unit) comes from an 8-bit field of the corresponding load or store.

7.2.4.1.2 Program Control Unit (PCU)

This part of the SDMA core is dedicated to the control of the RISC engine, as implied by the instructions that are executed. Its behavior is determined by the instruction type and the inputs of the SDMA.

It contains the PC, RPC, SPC, and EPC registers that are described in SDMA Core Structure.

7.2.4.1.2.1 Instruction Types

The state sequence and the delay of execution vary according to the type of the instruction. There are six possible categories of instructions, as follows:

1. Standard: Most of the instructions belong to this category, and always last 1 cycle.
2. ldf/stf: These are respectively the load and store instructions that access the functional units. They last 1+n cycles where n is the number of wait-states of the targeted functional unit.
3. ld/st: These are the load and store instructions that access the memory and peripherals. They last 1+n cycles where n is the number of wait-states of the targeted device (1 for the ROM, RAM, and memory mapped registers, 1 + the external peripheral wait-states). These instructions always last at least two cycles, but the core is able to handle them in one cycle. The first wait-state is inserted outside the core.
4. Branch: These are all the instructions that cause the Program Counter to point to another instruction other than the following one (for example, one that breaks the sequential flow). There are the absolute jumps, the conditional branches, the jump to the sub-routines, and the return from the sub-routine.
5. Loop,Modified Load or Store: The hardware loop instruction modifies the potential behavior of any load or store inside the loop (for example, when the LM flag is set). A jump may be implied after any such load or store if it received an error. The error causes an early exit of the loop, which means a jump to the instruction that follows the one that is pointed to by EPC. An additional cycle is required by the PCU to perform the jump (+1 to the ld/st/ldf/stf original execution delay). Although there is usually an implicit jump after the last instruction of the loop when the PC goes back to SPC, this is performed at no cycle cost.

6. Done: The done, yield, or yieldge instructions are used to control channel switching. When no channel switching is performed, these instructions last a single cycle. When there is a change of channel or context switch, the delay is variable and depends on many factors (as detailed in Context Switching).

7.2.4.1.2.2 PCU States

The PCU state is visible through outputs of the SDMA (see Real-Time Debug Outputs) or the OnCE status register(see OnCE Status Register (OSTAT)).

The PCU state is a four-bit field that can take the values shown in the following table. Figure 7-2 shows the possible state transitions and the corresponding conditions.

Table 7-5. PCU States

Value	State	Description
0	Program	This is the usual instruction cycle.
1	Data	This state is inserted when there are wait-states during a load or a store on the data bus (ld/st type).
2	Change of Flow	This is the second cycle of any instruction that breaks the sequence of instructions (branch and done types). This state lasts only a single cycle; it is always followed by the Program state.
3	Error in Loop	This state is used when an error causes a hardware loop exit (loop-modified load or store type). This state only lasts a single cycle; it is always followed by the Program state.
4	Debug	The SDMA is stopped in debug mode.
5	Functional Unit	This state is inserted when there are wait-states during a load or a store on the functional units bus (ldf/stf type).
6	Sleep	No script is running: The core is idle after saving the last channel context.
7	Save	The context switch FSM is saving the current channel.
8	Program in Sleep	Same as Program except there is no associated channel, this state is used when instructions are executed after entering debug mode, whereas the core was in either Sleep mode.
9	Data in Sleep	This is the same as Data except there is no associated channel.
10	Change of Flow in Sleep	This is the same as Change of Flow except there is no associated channel. This state only lasts a single cycle, and is always followed by the Program in Sleep state.
11	Error in Loop in Sleep	This is the same as Error in Loop except there is no associated. channel. This state only lasts a single cycle, and is always followed by the Program in Sleep state.
12	Debug in Sleep	This is the same as Debug except the core was put in debug mode when no channel was active.
13	Functional Unit in Sleep	This is the same as Functional Unit except there is no associated channel.
14	Sleep after Reset	This shows that no script is running, and the core is idle after a reset. When a channel becomes active, no context is restored but the core starts its boot program located at address 0 (or the address available in register in Channel 0 Boot Address (SDMAARM_CHN0ADDR)).
15	Restore	The context switch FSM is restoring the next channel context.

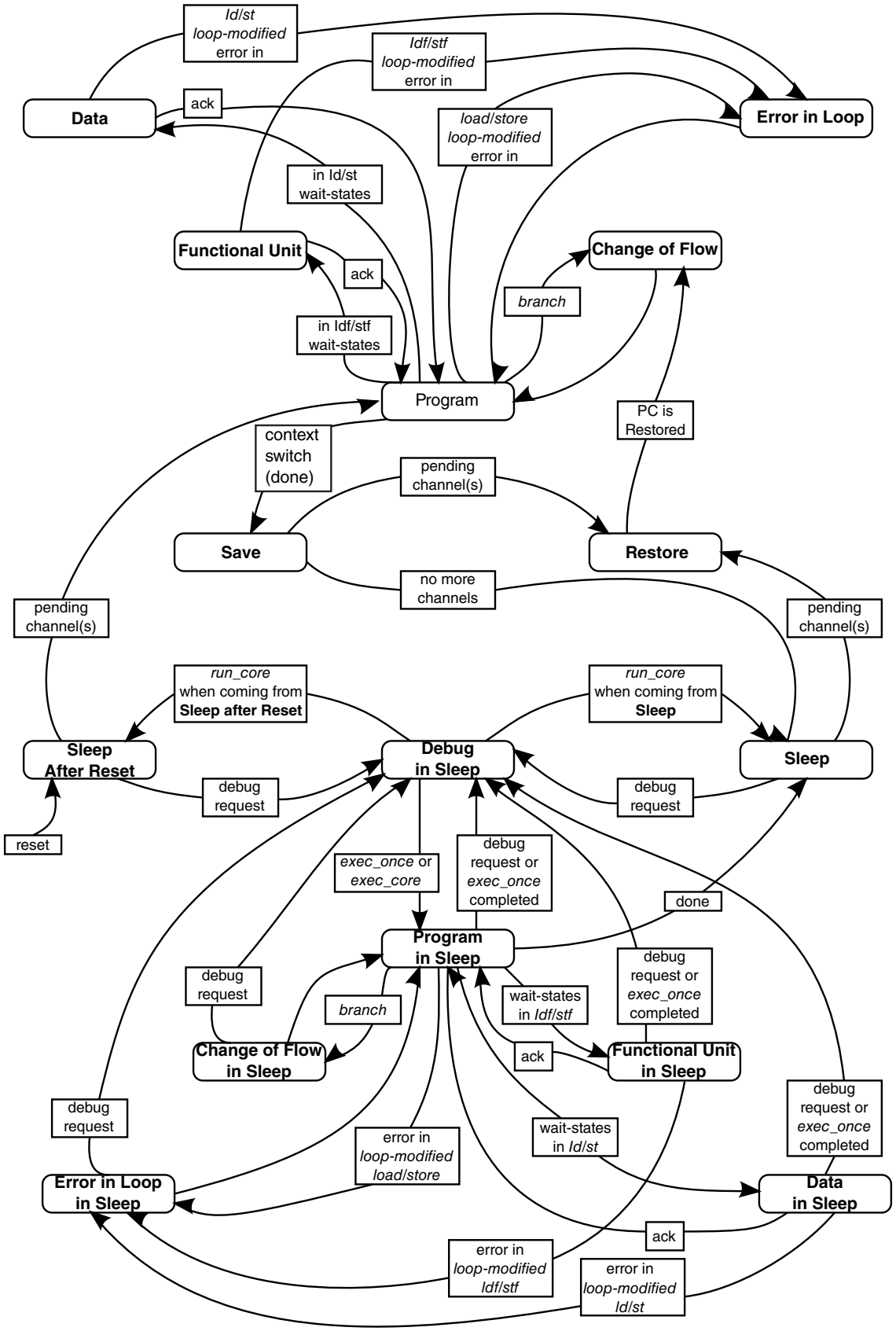


Figure 7-2. PCU State Diagram

7.2.4.1.3 SDMA Core Memory

The SDMA has two memory spaces: one for the instructions and one for the data. As both spaces share the same resources (ROM and RAM devices), the system bus manages possible conflicts when the core accesses the same resource for both an instruction read and a data read or write.

Program and data memory is further described in Address Space.

Instructions of 16-bit width are stored in 32-bit wide devices and can be accessed as data. The mapping is Big Endian: an even instruction address (terminated by 0) accesses the most significant part of the 32-bit data (bits [31:16]), and an odd instruction address (terminated by 1) accesses the least significant part of the 32-bit data (bits [15:0]). Instructions can be fetched out of internal ROM or RAM.

Data can be read from ROM, RAM, memory mapped registers, and external peripherals, and written to the same devices (except the ROM).

The ROM contains bootload scripts, channel scripts, and common subroutines which may be referenced by channel scripts elsewhere in the ROM or RAM.

The RAM is divided into a context area and a code space area which may be used to store channel scripts. The RAM contains undefined values after a hardware reset. Channel scripts and initial context values are downloaded into RAM using channel 0 which is reserved for bootload functions.

7.2.4.2 Scheduler

All channel scheduling hardware is included in the Scheduler.

7.2.4.2.1 Primary Functions

The scheduler is a hardware-based design used to coordinate the timely execution of 32 virtual DMA channels by the SDMA core on the basis of channel status and priority.

The scheduler performs the following functions:

- Monitors, detects, and registers the occurrence of any one of the 48 DMA requests
- Links a specific request to a channel or group of channels (channel mapping)
- Ignores requests that are not mapped to a previously configured channel
- Maintains a list of all the channels that are requesting service
- Assigns a pre-programmed priority level (1 of 7) to every channel requesting service
- Detects and flags overrun/underrun conditions

7.2.4.2.2 Channels and DMA Requests

7.2.4.2.2.1 Channels

A Virtual Channel (hereafter simply called a channel) manages a flow of data through the SDMA. Flows are typically unidirectional.

The SDMA can have up to 32 simultaneously operating channels, numbered from 0 to 31. Channel 0 is usually dedicated to control the SDMA script downloading. All the channels can be assigned by the ARM platform software.

7.2.4.2.2.2 DMA Requests

A DMA request is caused by externally (for example, external to the SDMA) controlled conditions (for example, UART receive FIFO reaches a threshold). The SDMA currently supports up to 48 DMA requests.

7.2.4.2.2.3 Mapping from DMA Requests to Channels and Priorities

A channel can stall waiting on a single DMA request. A single DMA request can awake more than one channel (in fact, any request can awake any combination of channels).

The mapping between DMA requests and channels is program-controlled. There is a storage element assigned for each of the 48 requests that contains a bitmap table of the channels that are awakened by the event.

Every channel also has a three-bit register that indicates its priority.

7.2.4.2.3 Scheduler Functional Description

Scheduler Overview describes the behavior of the SDMA scheduler-from the channel enabling conditions to the highest priority pending channel selection.

7.2.4.2.3.1 Scheduler Overview

The scheduler algorithm is built in hardware. It is provided with possibilities for the ARM platform to control its behavior.

The scheduler processes incoming DMA requests, maps detected requests to 0, one, or several channels, maintains a list of channels that are requesting service (pending channels), identifies the top priority and its associated channel, and selects the next active channel when the current channel yields.

The following figure shows a functional overview.

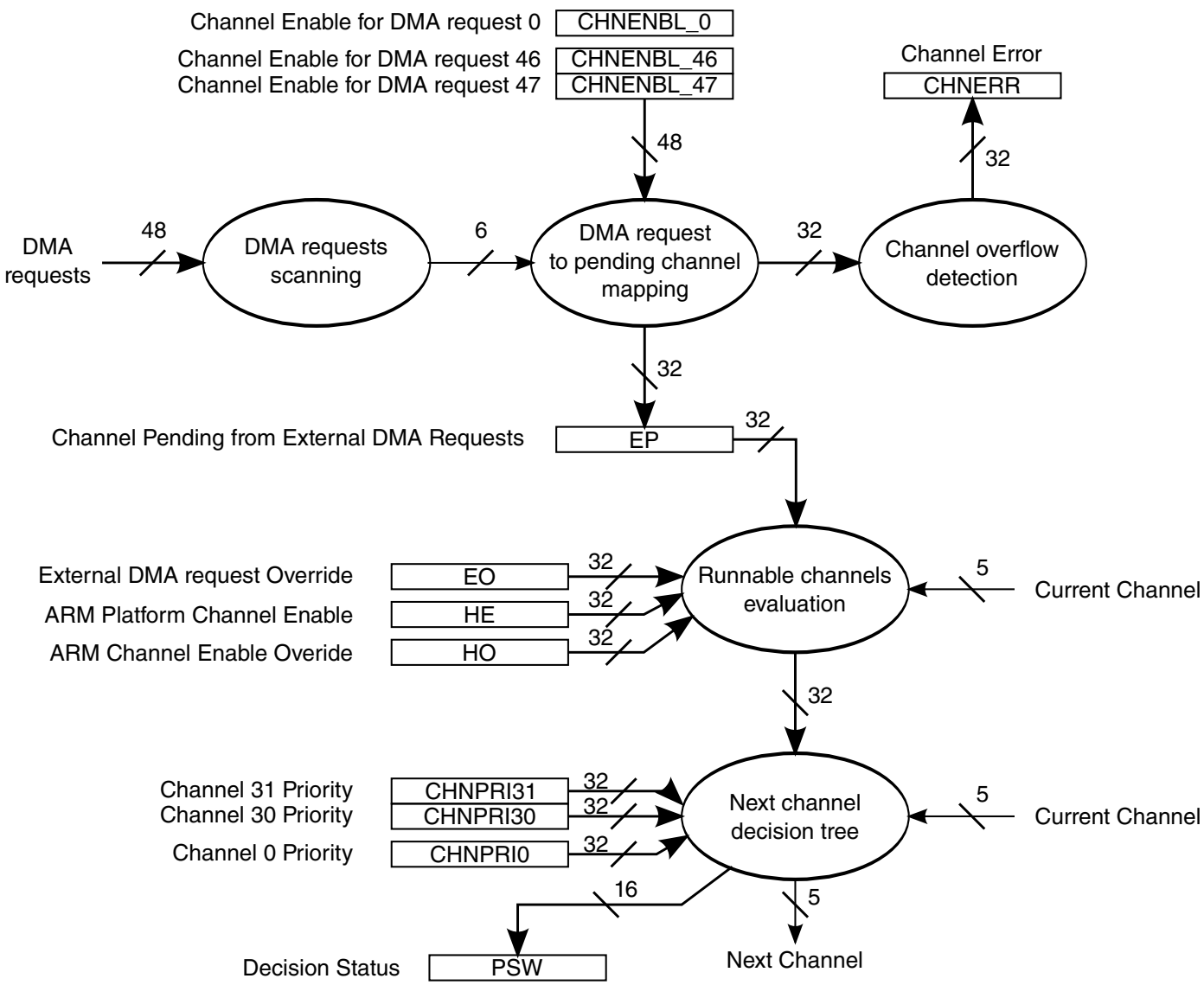


Figure 7-3. SDMA Hardware Scheduler

7.2.4.2.3.2 DMA Requests Scanning

The scheduler contains a 48-bit edge detection device that detects the rising edge of every DMA request and transmits the request number to the next stage.

The DMA requests are assumed to be generated on the same reference clock as the SDMA core clock; they are detected as soon as the signal goes from a 1-to-n-cycles low state to a 1-to-m-cycles high state.

This system is able to detect single-cycle pulses as well as level-based DMA requests such as a FIFO threshold crossing. In this case, the SDMA provides a memory mapped register that can be used by the channel script to monitor the DMA requests lines, and thus determines whether the data transfer is done or not done, and then continues with the transfer or closes the channel.

When several DMA requests are detected at the same time, they are forwarded to the next scheduler stage at the rate of one request per cycle. No request is lost.

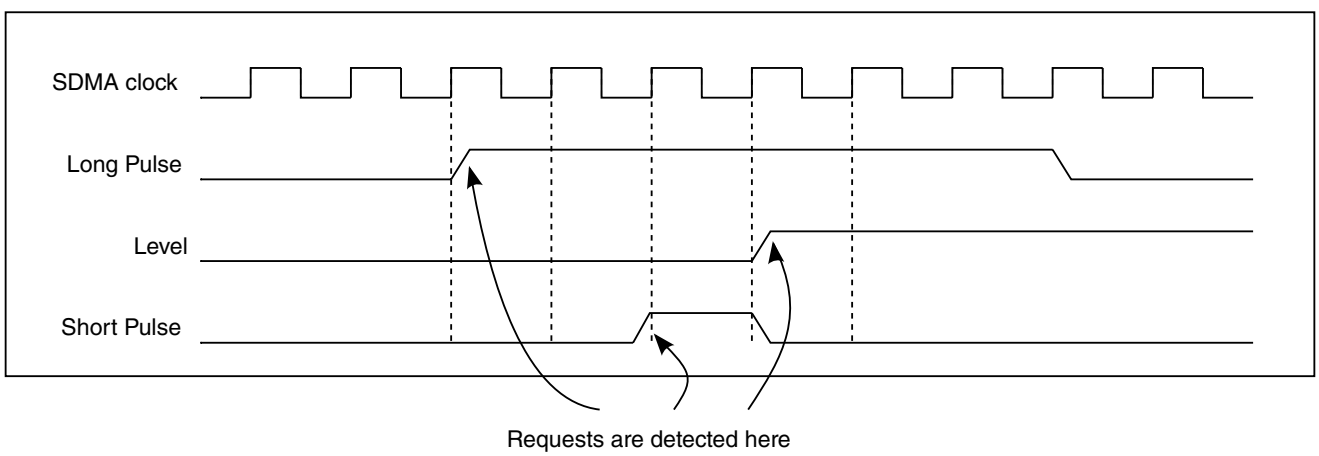


Figure 7-4. Examples of Valid DMA Requests

The DMA request inputs are connected to various sources that depend on the SoC. The exact list of DMA request inputs and their associated number is available in each respective project-specific chapter.

7.2.4.2.3.3 Mapping DMA Requests to Pending Channels

Whenever a DMA request is detected by the first stage, its number is used in the second stage to determine the channels that have to be activated.

This is performed with an array of 48 registers that are 32 bits wide: There are 48 Channel Enable Registers (CHNENBLn), one register per DMA request. The DMA request number selects the Channel Enable Registers, and every bit of this 32-bit register indicates that the corresponding channel must be activated when it is a 1.

This information is passed on the EP register. For every bit of the Channel Enable Register that is set, the corresponding bit of the EP register is also set, and the remaining bits of EP are left unchanged. The transformation of EP is summarized by the following equation:

EP = EP or CHNENBLn

The EP register is used to know which channels require service because they received a DMA request.

Typical contents of the CHNENBLn registers are all 0s, except for a single bit set. For example, a DMA request triggers one channel, but all 0s or several 1s are possible. One DMA request could activate several channels, and the channel execution sequence can be controlled by the channel priorities and numbers, as explained in the next sections. The following table illustrates an example configuration.

NOTE

From the table, the DMA request 0 is programmed to simultaneously trigger channels 0, 1, and 31. Also, DMA requests 30-47 are not used in this example. The remaining channels 2 to 30, are configured to be triggered by DMA requests 29 to 1, respectively.

Table 7-6. Channel Enable RAM Programming Example

DMA Request Number	Channel																																
	31																																0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 7-6. Channel Enable RAM Programming Example (continued)

DMA Request Number	Channel																														
	31																														0
7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 7-6. Channel Enable RAM Programming Example (continued)

DMA Request Number	Channel																														
	31																														0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

7.2.4.2.3.4 Channel Overflow

A channel overflow occurs when a DMA request requires service from channel *n* by setting bit *n* of the register EP, but this bit is already set, meaning channel *n* is already pending. This can come from an overrun/underrun condition.

This detection is possible only when the DMA requests are pulses, because a level-based DMA request stays high until it is serviced, even though an underrun or overrun condition occurs, thus preventing another edge detection of the DMA request.

The channel overflow information is saved in the 32-bit CHNERR register (1 bit per channel). You can configure the SDMA to trigger an interrupt to the ARM platform when there are 1s in CHNERR. Every bit of CHNERR is masked with the corresponding bit of INTRMASK and if it gives a 1, the corresponding bit of INTR is set, triggering the interrupt.

7.2.4.2.3.5 Runnable Channels Evaluation

The EP register is used in conjunction with several other 32-bit registers to determine the channels that are runnable.

Registers EO, DO, HO and HE, are controlled by the ARM platform. EP is controlled by the DMA requests and their mapping to channels.

Several channels may be runnable at any given time. The i^{th} channel is runnable if (and only if) the condition below is true:

(HE[i] or HO[i]) and (DO[i]) and (EP[i] or EO[i])

After reset, the HE[i], HO[i], EP[i], and EO[i] bits are all cleared whereas the DO[i] bits are all set. The functions associated with DO are not available for this device. When DO[i] is set, the scheduler condition becomes:

(HE[i] or HO[i]) and (EP[i] or EO[i])

The registers in these equations are controlled as follows:

- ARM platform (host) channel enable flag HE[i] may be set or cleared by the ARM platform with the HSTART and STOP_STAT registers. It can also be cleared by the i^{th} channel script.

  Typical usage is for the ARM platform to set this flag to activate the channel. The flag is cleared by the SDMA core when the transfer is done.

- Externally triggered channel pending flag EP[i] is set by the scheduler when the channel was activated by a DMA request. It can be cleared by the i^{th} channel script.
- The ARM platform channel override flag HO[i] may be set or cleared by the ARM platform. When set, it enables the i^{th} channel to run without the involvement of the ARM platform.

  Typical usage is for the ARM platform to set this flag for channels that do not need ARM platform supervision such as channels that are controlled by DMA request events (EP).

- DO should always be set to 1 so that the runnable channel evaluation considers only HO, HE, EP, and EO.
- Externally triggered channel override flag EO[i] may be set or cleared by the ARM platform. When set, it prevents the i^{th} channel from stopping and stalling on incoming peripheral DMA requests. This is the case when the channel is not handling data transfers with peripherals (for example, a memory to memory transfer).

The SDMA can clear the HE[i], and EP[i] bits by means of a done or notify instruction. The done instruction causes a reschedule; thus, enabling another channel to preempt the current one, while the notify instruction does not. The done and notify instructions can clear either HE[i] or EP[i] (never more than one at a time).

Table 7-7. Runnable Channel Selection Control

Register	Set by	Cleared By
HO	Write to HOSTOVR register	Write to HOSTOVR register

Table continues on the next page...

Table 7-7. Runnable Channel Selection Control (continued)

Register	Set by	Cleared By
HE	Write to HSTART register	Write to STOP_STAT register or by the channel script with the done or notify instructions.
DO	Write to DSPOVR register	Write to DSPOVR register
EO	Write to EVTOVER register	Write to EVTOVER register
EP	Set by external DMA request event input.	By the channel script with the done or notify instructions

7.2.4.2.3.6 Next Channel Decision Tree

The next channel number is computed from the runnable channels list, the current channel number, and their respective priorities.

It is re-evaluated every cycle, but is only used when the current channel yields or terminates by executing a yield, yieldge, or done instruction.

The decision tree is based on the selection of the runnable channel that has the highest priority.

The highest priority channel is selected according to the following rules:

- Runnable channels are sorted by priority.
- If one of the channels with the highest priority had been preempted by a channel with a higher priority, but did not want to yield to a channel of the same priority (for example, it executed a yield, not a yieldge), it is elected as the next channel.
- The channels that belong to the highest priority group are sorted by their number and the channel that has the highest number in this group becomes the next channel. For example, if priorities are the same, channel 31 will be selected before channel 30.

When the current channel requires a reschedule with a yield(ge) or a done instruction, the context switch decision is based on the instruction parameter, the current channel number and priority, and the next channel number and priority. The possible cases are all listed in the following table. The grayed cells correspond to unusual cases that should not occur with a typical usage of the SDMA.

Table 7-8. Channel Switching Decision with a yield, yield(ge), or done

Instruction	Current Channel	Next Channel	Priorities Comparison	New Running Channel/Comments
yield (done 0)	Runnable	Not runnable	none	Current
	Runnable	Runnable	Current > Next	Current
			Current = Next	Current
			Current < Next	Next [1]

Table continues on the next page...

Table 7-8. Channel Switching Decision with a yield, yield(ge), or done (continued)

Instruction	Current Channel	Next Channel	Priorities Comparison	New Running Channel/Comments
	Not runnable	Not runnable	none	none[2] (occurs when the channel was disabled by the ARM platform)
	Not runnable	Runnable	none	Next[1] (occurs when the channel was disabled by the ARM platform)
yieldge (done 1)	Runnable	Not runnable	none	Current
	Runnable	Runnable	Current > Next	Current
			Current = Next	Next[1]
			Current < Next	Next[1]
	Not runnable	Not runnable	none	none[2] (occurs when the channel was disabled by the ARM platform)
	Not runnable	Runnable	none	Next[1] (occurs when the channel was disabled by the ARM platform)
done (done>1)	Not runnable	Not runnable	none	none[2]
	Runnable	Not runnable	none	Current[3] (occurs when the done instruction does not disable the channel runnable condition)
	Not runnable	Runnable	none	Next[1]
	Runnable	Runnable	none	Current[3] (occurs when the done instruction does not disable the channel runnable condition)

1. Current channel script execution is stopped, its context is saved; the next channel context is restored and its script execution resumes
2. Current channel context is saved and SDMA enters IDLE mode
3. Current channel context is saved, then restored, and the current channel script resumes execution

Finally, when the SDMA is in IDLE mode and a runnable channel is elected as the next channel, its context is immediately restored and the script execution resumes.

The *combinatorial-decision* tree supports dynamic modifications of the EP, EO, HE, HO, and DO flags as well as dynamic modifications of the channel priorities. The propagation times are detailed in Scheduler Pipeline Timing Diagram.

The decision tree status is available in the PSW register, which is continuously updated. It contains the next channel priority, the next channel number, the current channel priority, and the current channel number. When a priority is read as 0, it means the channel is not runnable.

A few examples of decisions are presented below:

- Channel 31 is running with priority 5, channels 13 and 24 are pending with the same priority 5; channel 24 is eligible as the next channel since 24 > 13.
- Channel 31 is running with priority 7, channels 13 and 24 are pending with priority 5; channel 31 is the next channel because its priority is greater than the other pending channels.
- Channels 7, 23, and 29 are pending with the same priority. Channel 7 is active and runs a yieldge; it is preempted by channel 29. After a period of time, channel 29 runs a yieldge, it is then preempted by channel 23 that is the selected channel since channel 29 is the current channel. Later, channel 23 runs a yieldge and is preempted by channel 29. Channels 23 and 29 will go on switching after every yieldge until one of them terminates. It is only at that point that channel 7 becomes eligible again.
- Channel 11 is running with priority 3, and channel 15 is pending with priority 4. When the channel 31 script executes a yield instruction, it gets preempted by channel 15; then channels 6 and 18 with priority 3 become pending. Because channel 11 was preempted after executing a yield and there is no pending channel with a strictly greater priority, it is eligible as the next channel (although its number 11 < 18).

7.2.4.2.3.7 Scheduler State Diagram

The Figure 7-5 summarizes the behavior of the SDMA scheduler with details about the exact mechanism of the priority decision tree. It is important to understand the scheduler is a hardwired pipeline, which means all the stages are performed simultaneously every cycle, but a change on any given stage is reflected on the next stage after the delays presented in Scheduler Pipeline Timing Diagram.

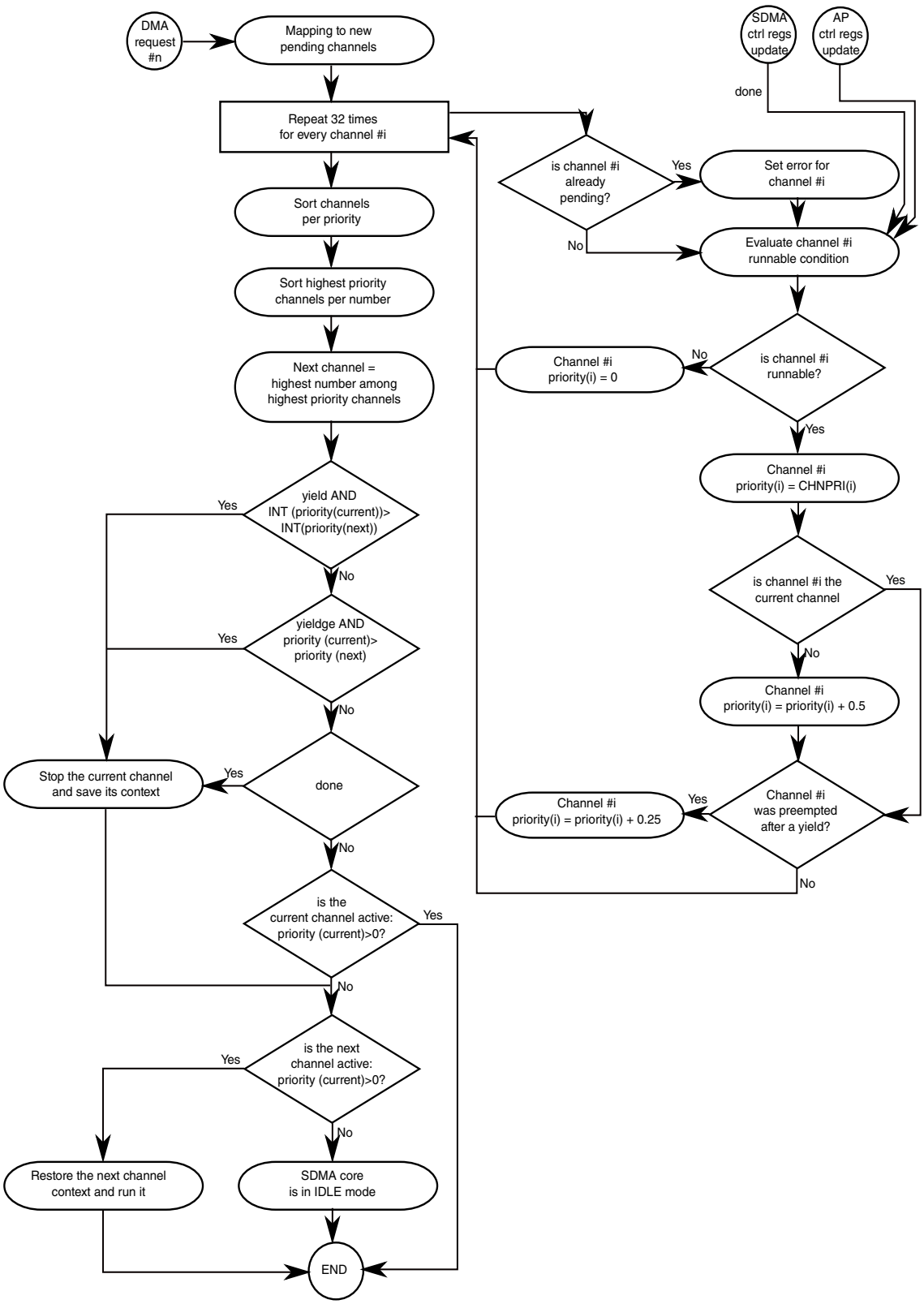


Figure 7-5. Scheduler State Diagram

7.2.4.2.3.8 Scheduler Pipeline Timing Diagram

The SDMA scheduler process of DMA-request and control-register modifications is not immediate.

The figure below shows the exact delays of all the tasks. The reference clock is the SDMA core clock.

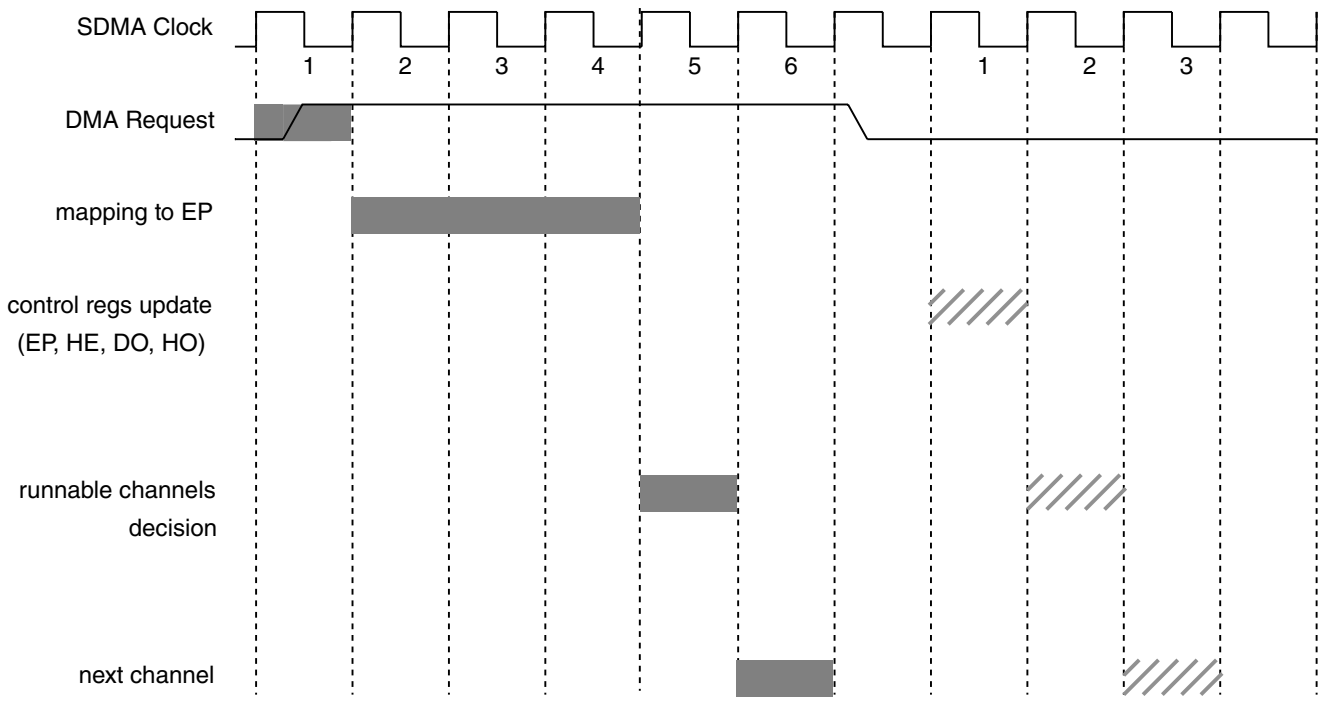


Figure 7-6. Scheduler Timing Diagram

Two numbers can be inferred from this timing diagram. First, it takes six SDMA core clock cycles to update the next channel from a DMA request. Second, it takes three SDMA core clock cycles to update the next channel from a direct modification of the condition registers (EP, DO, HE, or HO) by any processor. The processors that can modify these bits include SDMA with a done instruction or the ARM platform with a write access through the corresponding control port on their respective peripheral bus).

7.2.4.2.3.9 Channel-DMA Request Mapping

The 48 DMA request inputs to the SDMA scheduler are listed in project-specific chapters. Refer to the respective chapters for this information.

7.2.4.2.3.10 Examples: How to Start a Channel

A channel can be started when the following equation is true for channel *i*:

(HE[i] or HO[i]) and (DO[i]) and (EP[i] or EO[i])

Once this equation is true, the scheduler can start this channel according to the priority of all pending channels. Several examples of configuration are listed below:

1. To start a channel triggered by ARM platform software:
   - Initially, configure HO[i]=0, DO[i]=1, and EO[i]=1 using registers indicated in Table 7-7.
   - ARM platform software triggers the channel by writing to the HSTART register to set HE[i]=1, thereby setting the above equation true.
2. To start a channel triggered by DMA request event.
   - Initially, configure HO[i]=1, DO[i]=1, and EO[i]=0 using registers indicated in Table 7-7.
   - The DMA request is asserted to trigger the channel by setting EP[i]=1, which makes the above equation true.

7.2.4.2.4 Context Switching

On execution of a done or yield(ge) instruction, the current channel may be changed either because it has finished (which necessarily happens when the done instruction is executed), or it was preempted by a higher priority channel (which is possible but not systematic when the yield(ge) is executed).

Upon a channel change the SDMA goes through a context switch procedure.

When the current channel yields or ends, the context for that channel is saved into the context RAM locations for that channel. When the next channel starts running, its context is first restored from RAM.

Since context RAM is not yet initialized by reset, there will be no context restore at the beginning of the first channel (bootload channel) run after reset. It is expected that the bootload channel will be used to initialize the context for all other channels. When the bootload channel finishes running or yields, SDMA will enter its SAVE state and save that channel's context into RAM. Then, if the bootload channel is called again later, the context will be restored from RAM when the channel starts again.

The context structure for each channel is defined in Context Switching-Programming and Table 7-13. There will be one context area reserved for each channel. When a channel ends or yields, the SDMA core registers are automatically saved into the context RAM and later restored from the context RAM when the channel is next run. The total RAM

space reserved for 32-channel contexts is either 3K or 4K depending on whether the SMSZ bit is set in the CHN0ADDR register, which enables an additional 8 words of scratch RAM for each context.

7.2.4.2.4.1 Context Switch Modes

The exact procedure to save the context of the old channel, and to restore the context of the new channel depends on the context switch mode selected by the ARM platform in the CONFIG control register.

The following are the context switch modes:

- By default, the "dynamic" context switch is set. This mode provides the most efficient context switch for an average of eight cycles to stop the current channel, save its context, restore the next channel context, and resume its execution. It consists of saving modified registers of the current channel in the background (for example, during the channel execution)-which leaves very few registers to save when the switch is decided-resuming execution of the next channel as soon as possible (for example, when the minimal set of registers is restored), and continuing the restore phase during this execution.
- In "dynamic with no loop" mode, the same principle is followed except the modified registers are only saved in the background when the loop flag is not set. This mode offers almost the same effectiveness as the previous one, but it prevents the system from accessing the RAM during loops to save power. This is the recommended mode for an efficient context-switch when the loop bodies are short.
- In "dynamic power" mode, no background saving is performed, which reduces power consumption to the minimum. The modified registers are only saved when the context switch starts. The restore phase is the same as before. This is the mode that achieves the optimal power consumption at the cost of a slower context-switch.
- In a "static" context switch, all the registers are saved when a context switch is decided, and all the registers are restored before starting the execution of the new channel. This mode enables a predictable behavior of the context switch since all the registers are restored prior to the channel start and all registers are saved after the channel termination.

NOTE

Static context mode should be used for the first channel called after reset to ensure that the all context RAM for that channel is initialized during the context SAVE phase when the channel is done or yields. Subsequent calls to the same channel or different channels may use any of the dynamic context modes. This will ensure that all context locations for the bootload

channel are initialized, and prevent undefined values in context RAM from being loaded during the context restore if the channel is re-started later.

7.2.4.2.4.2 Context Switch Procedure

The Program Control Unit goes into the *save* state, the current context is spilled into memory, and the next channel context is restored according to the context-switch mode that was selected by the ARM platform.

The context switch procedure is as follows:

1. Load the current context's spill base address.
2. Spill the modified registers of the current channel to memory according to the selected context switch mode while the channel is running.

   On a done or yield(ge) that causes the channel preemption, the PCU goes into the *save* state. In *static* mode, all the registers are saved; whereas, in either *dynamic* mode, the registers that were modified but not yet saved are then saved, and the PCU registers and flags are finally saved.

3. Put the SDMA core into *sleep* and wait for new channels to be serviced. This step is skipped if there are pending channels when the current channel is saved.

   As soon as there is at least one pending channel, the PCU goes into its *restore* state to restore the context of the channel that was elected by the scheduler.

   Once a channel is elected, it remains the current channel until its script requests a rescheduling operation with a done or yield(ge) instruction. That means the current channel cannot be modified by the ARM platform, even if it is no more runnable or if its priority is modified.

   The ARM platform can however force a reschedule by writing the corresponding bit in the CONFIG register, which has the same effect as if the script had executed a done instruction. That feature should only be used to stop the SDMA in emergency cases.

4. Load the context base-address of the new channel.

   In "static" mode, all the registers are restored. In either "dynamic" modes, only the PCU registers are restored.

   The new channel is running. In "static" mode, no more activity regarding context restoring or saving is performed. In either "dynamic" modes, the registers are restored in the background every time an access to the context RAM is possible, and

priority is given to restoring the registers that are required by the next instruction to be executed. When a register has not been restored and the next instruction needs it, this instruction gets stalled until the register was restored.

In "dynamic" and "dynamic with no loop" modes, background saving of dirty registers is performed every time an access to the context RAM is possible and allowed by the context switch mode.

NOTE

The contents of a channel context space in the context RAM depends on the selected context switch mode. In "dynamic" and "dynamic with no loop" modes, the contents of the context RAM tend to match the contents of the SDMA registers (except for the PCU registers and flags that are never saved in the background). In "dynamic power" and "static" modes, the contents of the context RAM remain unchanged until the channel terminates with a done or gets preempted.

7.2.4.2.4.3 Context Map in Memory

Refer to Context Switching-Programming.

7.2.4.3 Functional Units

The functional units are small systems that are used by the SDMA core to handle data transfers between the core and a bus domain external to the SDMA.

The SDMA core is able to control and exchange data with these systems by sending instructions and reading or writing data from/to the functional units' registers via the FUBUS. This is done with the ldf and stf instructions.

The following sections provide introductions to the available functional units. Functional Units Programming Model provides descriptions the functional units' behaviors.

7.2.4.3.1 Burst DMA Unit

The burst DMA unit enables the SDMA core to perform data transfers to and from the ARM platform memory.

It is optimized for accessing SDRAM-like devices. It does not provide control to assign a privilege level to the DMA access. The burst DMA unit provides the SDMA with means to do the following:

- Perform up to 8-beat read and write bursts to the ARM platform memory, which optimizes throughput when accessing SDRAM-type devices because of an internal, 36-byte FIFO
- Access the ARM platform memory at once or twice the SDMA core frequency
- Copy data from one ARM platform memory location to another ARM platform memory location at the ARM platform bus speed, which provides a very high throughput
- Control the method for addressing the ARM platform memory (automatic increment of addresses or frozen addresses-the former aimed at accessing RAM-like memory and the latter aimed at accessing single-address FIFOs)
- Enable or disable automatic prefetch when reading data from the ARM platform memory. When the prefetch mode is selected, the burst DMA automatically triggers external bursts to fill its FIFO without waiting for the SDMA core to request the corresponding data, greatly improving throughput.
- Rely on the DMA to automatically flush its FIFO content when there is enough data to generate an 8-beat burst to the ARM platform memory. Or, it forces a flush when a data transfer must terminate.
- In the former case, the SDMA core may only be stalled when it tries writing data and there is not enough room left in the FIFO. In the latter case, the core is stalled until the data is effectively written to the ARM platform memory.

  In automatic flush mode, the core receives an acknowledge that does not reflect the actual error status when the data is effectively written into the ARM platform memory. This error status is retrieved by a later access to the burst DMA. Terminating a write data transfer with a forced flush command guarantees that any bus error to the ARM platform memory is caught.

- Handle address alignment issues between the ARM platform memory map and the SDMA core data. This enables the core to read or write 32-bit data from the burst DMA, whereas the corresponding ARM platform address is not 32-bit aligned. This drastically improves the SDMA scripts' efficiency since the same loop that transfers 32 bits at a time can be used regardless of the start and end addresses in the ARM platform memory space.

This unit structure and registers are described in Burst DMA Structure and Burst DMA Registers.

7.2.4.3.1.1 Burst DMA Structure

The burst DMA is essentially made up of a 36-byte FIFO, address registers, and a controlling state-machine. The 36-byte FIFO enables eight-word buffering with address alignment, and the state-machine manages clock adaptation when required.

The burst DMA is depicted in the figure below.

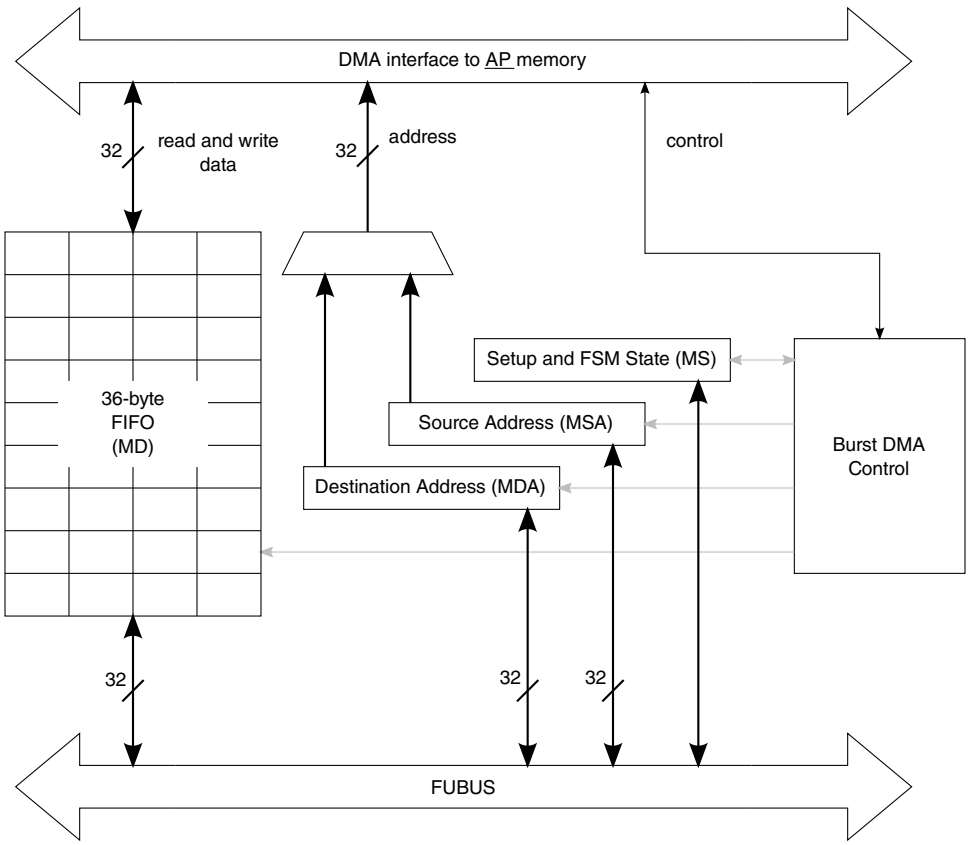


Figure 7-7. Burst DMA Structure

7.2.4.3.1.2 Burst DMA Registers

There are four registers, as follows, that may be accessed from the SDMA core:

- MSA (Memory Source Address) - Holds the source byte address in the ARM platform memory map for reading data from this location. This register is automatically modified every time the core reads new data from the FIFO.
- MDA (Memory Destination Address) - Holds the destination byte address in the ARM platform memory map for writing data to this location. This register is automatically modified every time the core writes new data into the FIFO.
- MD (Memory Data) - Labels the 36-byte FIFO access point: Reading a byte, halfword, or word from MD respectively retrieves the first 1, 2, or 4 bytes of the

FIFO (for example, the bytes that were stored first by the DMA state-machine when transferring data from the ARM platform memory).

- When the FIFO does not hold as many bytes as required by the SDMA core, the core is stalled until the missing bytes are read from the ARM platform memory. In the case of prefetch mode, the DMA controller decides when it should start a burst to ARM platform memory in order to reduce the risk to not have the required data for the future accesses of the core. When there is no prefetching, a burst is triggered when the required data is not available in the FIFO.

  Writing a byte, halfword, or word to MD stores 1, 2, or 4 bytes, respectively, at the end of the FIFO (for example, these bytes are transmitted to the ARM platform memory after all the other bytes that were previously stored in the FIFO). When the FIFO does not have enough room left to hold the written data, the SDMA core is stalled until a sufficient amount of FIFO contents are flushed out to the ARM platform memory. Flushing is decided by the DMA controller when there are enough bytes in the FIFO to perform the largest allowed burst to ARM platform memory (the exact size depends on the burst start address and the AHB 1 Kbyte boundary rule). However, the SDMA core has the ability to force the flushing operation at any time, for example, when at the end of the data transfer, prior to channel closure.

- MS (Memory Setup) - Contains the state of the burst DMA control, the two flags that define whether each address register is incremented after every access to the external memory, and another flag that is set when a bus error occurred.

7.2.4.3.1.3 Burst DMA Data Transfers

Three typical usages have been identified that involve the burst DMA: the data transfer startpoint, the endpoint, or both.

Every case requires a different procedure, as listed in the following sections:

7.2.4.3.1.3.1 Data Retrieval from the ARM platform Memory

The following steps retrieve data from ARM platform memory using the burst DMA unit:

- Set up the MS flags to reflect the mode for the source address (incremented or frozen according to the type of accessed device: memory or peripheral FIFO), then initialize the source address register itself (MSA).
- Read data from the FIFO using the *ldf MD* instruction as many times as needed. If an error occurred during the fetch from ARM platform memory, the DMA control tags the error status on the data and the SDMA core SF flag is set when reading this data from the FIFO.

7.2.4.3.1.3.2 Storing Data Into the ARM platform Memory

The following steps store data from ARM platform memory using the burst DMA unit:

- Set up the MS flags to reflect the mode for the destination address (incremented or frozen according to the type of accessed device: memory or peripheral FIFO), then initialize the destination address register itself (MDA).
- Store data into the FIFO using the *stf MD* instruction as many times as needed.
- When the transfer is finished and if the DMA worked in automatic flush mode, force the flush of the FIFO. This instruction is stalled until all the FIFO data is effectively sent to the ARM platform memory and the error status of the transfer is available in the DF flag.

7.2.4.3.1.3.3 Transferring Data Between Two ARM platform Memory Locations-Burst DMA Unit

The following steps copy data between two ARM platform memory locations using the burst DMA unit:

- Set up the MS flags to reflect the modes for the source and destination addresses (all the combinations are possible), then initialize the source address register (MSA) and the destination address register (MDA). Both addresses must be word-aligned.
- Use as many *stf MD* instructions with the *COPY* flag as needed. Every instruction triggers a burst read of a given number of words from the source address (this number is provided to the burst DMA via the SDMA core general purpose register, which is referenced in the *stf* instruction). Once all the data is loaded into the FIFO, the DMA empties it with a write burst of the same count to the destination address. The DMA acknowledges prior to instruction completion, which frees the SDMA core for other tasks at no delay cost.
- Once the transfer is done, there should be a final access to the burst DMA to check the error status.

7.2.4.3.2 Peripheral DMA Unit

The peripheral DMA unit is the second functional unit that connects the SDMA to the ARM platform memory.

Unlike the burst DMA, it does not support burst transfers and is optimized for accessing peripherals. It does not provide control to assign a privilege level to the DMA access. Its feature list comprises the following:

- Access to the ARM platform peripherals or memory at once or twice the SDMA core frequency

- Data copy from one ARM platform memory location to another ARM platform memory location at memory bus speed, improving throughput
- Control of the method for addressing the ARM platform memory (automatic increment or decrement of addresses or frozen addresses, the first ones aimed at accessing RAM-like memory and the last one aimed at accessing single-address FIFOs)
- Selectable automatic prefetch when reading data from the ARM platform memory. In prefetch mode, the peripheral DMA automatically fetches another data-without waiting for the SDMA core to request it-when its data register is empty, which improves the throughput
- Selectable automatic flush. In this mode, the SDMA core may only be stalled when it tries writing data and the previous write operation is not finished yet; whereas, in forced flush mode, the core is stalled until the data is effectively written to the ARM platform memory.
- In automatic flush mode, the core receives an acknowledge that does not reflect the actual error status when the data is effectively written into the ARM platform memory or the peripheral. This error status is retrieved by a later access to the peripheral DMA. Terminating a write data transfer with a forced flush command guarantees that any bus error to the ARM platform memory has been caught.

  This unit structure and registers are described in Peripheral DMA Structure and Peripheral DMA Registers.

7.2.4.3.2.1 Peripheral DMA Structure

The peripheral DMA is made up of a 32-bit data register, two address registers, and a controlling state-machine. The state-machine manages clock adaptation, when required.

It is shown in the following figure.

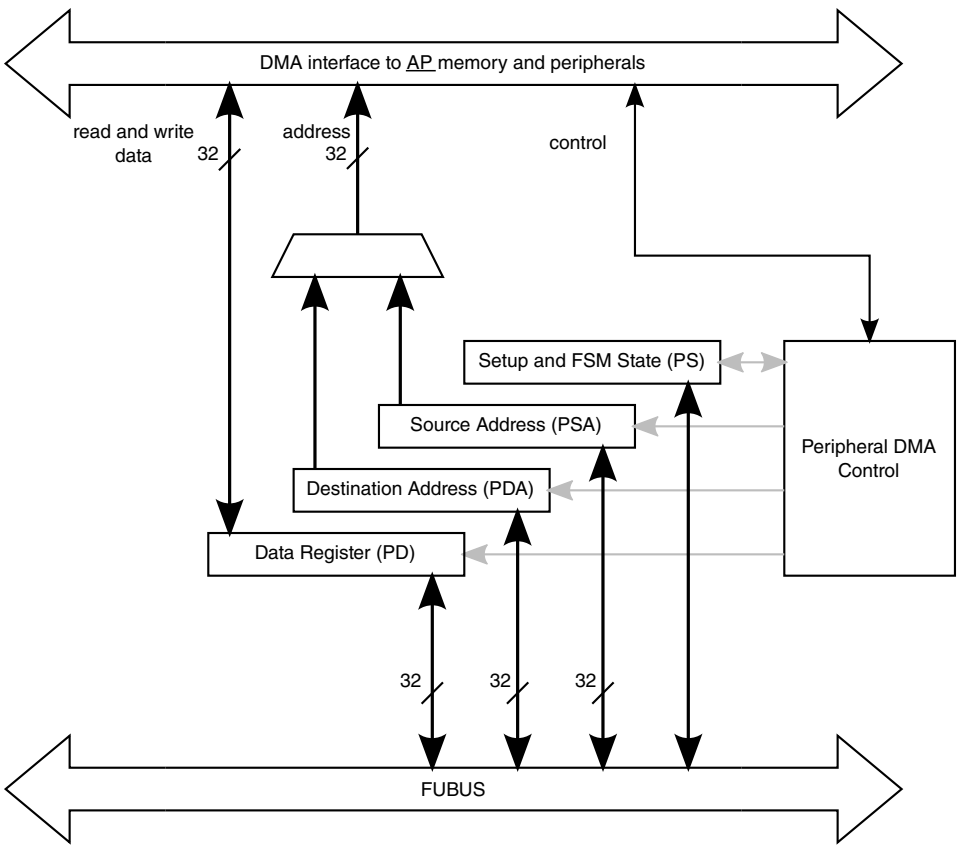


Figure 7-8. Peripheral DMA structure

7.2.4.3.2.2 Peripheral DMA Registers

According to Figure 7-8, the peripheral DMA has four registers that may be read or written by the SDMA core:

- *PD (Peripheral Data)* is the DMA 32-bit data register.
- *PSA (Peripheral Source Address)* holds the source byte address in the ARM platform memory map for reading data from this location. This register is automatically modified every time the core reads a new data from PD.

- *PDA (Peripheral Destination Address)* holds the destination byte address in the ARM platform memory map for writing data to this location. This register is automatically modified every time the core writes a new data into PD.
- *PS (Peripheral Setup)* contains the state of the peripheral DMA control, two configuration fields that define the way address registers are modified after every data access, two additional configuration fields that define the data size to access the source and destination devices, and another field that contains the latest transfer error status.

7.2.4.3.2.3 Peripheral DMA Data Transfers

There are three typical usages that involve the peripheral DMA, whether it is the data transfer start-point, endpoint, or both.

Every case requires a different procedure, as described in Data Retrieval from the ARM platform Memory or Peripheral, Storing Data into the ARM platform Memory or Peripheral, and Transferring Data Between Two ARM platform Memory Locations-Peripheral DMA Unit.

7.2.4.3.2.3.1 Data Retrieval from the ARM platform Memory or Peripheral

The following steps retrieve data from ARM platform memory using the peripheral DMA unit:

- Set up the PS fields to reflect the mode and data size for the source (incremented, decremented, or frozen address register; 8-bit, 16-bit, or 32-bit data transfers), then initialize the source address register itself (PSA) with an address that is aligned to the programmed data size.
- Read data from PD using the ldf PD instruction as many times as needed. If an error occurs during the fetch from the ARM platform memory or peripheral, the DMA control tags the error status on the data and the SDMA core SF flag is set when reading this data from PD.

7.2.4.3.2.3.2 Storing Data into the ARM platform Memory or Peripheral

The following steps store data to ARM platform memory using the peripheral DMA unit:

- Set up the PS fields to reflect the mode and data size for the destination (incremented, decremented, or frozen address register; 8-bit, 16-bit, or 32-bit data transfers), then initialize the destination address register itself (PDA) with an address that is aligned to the programmed data size.

- Store data into PD using the *stf PD* instruction as many times as needed.
- When the transfer is finished and if the peripheral DMA worked in automatic flush mode, force the flush of PD. This instruction is stalled until PD contents are effectively sent to the ARM platform memory or peripheral, and the error status of the transfer is available in the DF flag.

7.2.4.3.2.3.3 Transferring Data Between Two ARM platform Memory Locations-Peripheral DMA Unit

The following steps copy data between two ARM platform memory locations using the peripheral DMA unit:

- Set up the PS fields to reflect the modes and data size for the source and destination addresses (all the combinations of addressing modes are possible, but both data sizes must be identical), then initialize the source address register (PSA) and the destination address register (PDA). Both addresses must be aligned with the programmed data size.
- Use as many *stf PD* instructions with the *COPY* flag as needed. Every instruction triggers a single read from the source address; a single write of the received data immediately follows. The DMA acknowledges prior to instruction completion, which frees the SDMA core for other tasks at no delay cost.
- Once the transfer is done, there should be a final access to the peripheral DMA to check the error status.

7.2.4.4 SDMA Security Support

The SDMA provides support to SDMA software to block unauthorized updates to the scripts in RAM.

SDMA supports the following Security modes:

- Open Mode: has full control to load scripts and context into SDMA RAM. This is the default mode.
- Locked Mode: The ARM platform loads scripts and channel contexts at startup when it is still executing known safe software. When finished, it locks the SDMA to prevent further updates to RAM and selected registers. More details described in Locked Mode.

7.2.4.4.1 Locked Mode

The LOCK bit in the SDMA_LOCK register provides support for SDMA scripts to freeze RAM contents after the initial bootload routine to prevent future unauthorized updates to SDMA RAM.

After initial RAM contents are uploaded, ARM platform software can set the LOCK bit to secure the RAM contents to prevent future updates by an unauthorized. After the LOCK bit is written with a '1', the SDMA is "locked" until reset.

The LOCK bit can be read in the SDMA's internal memory map in the LOCK register (see Section SDMA LOCK (SDMAARM_SDMA_LOCK)). SDMA scripts which load information into RAM can check the value of the LOCK bit to determine if an upload to RAM is allowed. If not allowed, the script can refuse to allow the request to copy data into the RAM to continue. The exact use of the LOCK bit in SDMA scripts for security control will be described in SDMA software documentation (see SDMA Scripts).

While SDMA is locked, attempts to write to the SDMA_LOCK, CHN0ADR, ILLINSTADDR, and ONCE_ENB registers will be ignored. All registers remain readable. Writes to other registers are still allowed.

Once the SDMA is locked, the LOCK bit can only be cleared by a reset. A hardware reset will always clear the LOCK bit. A software reset initiated by writing to the RESET register will only clear the LOCK bit if the SRESET_LOCK_CLR bit in the SDMA_LOCK register is set. Since SDMA_LOCK register cannot be updated if SDMA is locked, the SRESET_LOCK_CLR bit must be configured before setting the LOCK bit. The SREST_LOCK_CLR bit will also be cleared by resets that clear the LOCK bit.

The SDMA RISC core uses the ILLINST and CHN0ADDR registers as pointers to determine where to jump to after an illegal instruction or upon boot after a reset. The LOCK bit prevents updates to these registers to protect against unauthorized changes to these pointers.

While SDMA is locked, the ONCE_ENB register cannot be written to prevent the OnCE under ARM platform control from being used to gain access to SDMA internal memory. If ARM platform control of the OnCE is enabled before setting the LOCK bit, the ARM platform can use the ONCE for debug purpose after LOCK is set.

7.2.4.5 OnCE and PCU Debug States

The SDMA has two different debug modes in which the OnCE performs debug instructions.

Refer to Figure 7-2 for an example of the PCU states in debug. The following are the two debug states:

- When a channel is running (that is, when CCR and CCPRI are different from 0, which can be read in the PSW register), SDMA can execute a SoftBkpt instruction from the channel script or receive a debug request. When either happens, the SDMA enters its "Classical" *Debug* state, which is described in OnCE and Real-Time Debug.
- When a channel is not running, the SDMA can be in *Sleep* state or in *Sleep after Reset* state. If a debug request is sent to the core, it enters its *Debug in Sleep* state. This debug mode works similarly to the "Classical" *Debug* state, except it returns to the original state (*Sleep* or *Sleep after Reset*) when the debug mode is left via the exec_core instruction of the OnCE. From this *Debug in Sleep* state, the SDMA can execute a program whereas no channel is running. If a new debug request is sent to the core or if a SoftBkpt is executed, it comes back to this *Debug in Sleep* state.

The OnCE is provided with several instructions that can be executed when the core is in either debug state. The following table summarizes the behavior of these OnCE debug instructions. There exists other secondary OnCE instructions that are described in OnCE and Real-Time Debug.

Table 7-9. SDMA in Debug Mode

Instruction	Debug	Debug in Sleep
exec_once	exec_once <instruction> SDMA executes the <instruction> and returns to the *Debug* state. The Program Counter (PC) is not incremented. This command must not be used with an instruction that modifies the PC value.	exec_once <instruction> SDMA executes the <instruction> and returns to the *Debug in Sleep* state. The Program Counter (PC) is not incremented. This command must not be used with an instruction that modifies the PC value.
run_core	run_core <instruction> SDMA executes the <instruction>, leaves the *Debug* state and continues executing the channel script from the position where it stopped. This command must not be used with an instruction that modifies the PC value.	run_core <instruction> SDMA executes the <instruction> and returns to its *Sleep* or *Sleep after Reset* initial state. This command must not be used with an instruction that modifies the PC value.
exec_core	exec_core <instruction> It is similar to run_core except it requires an instruction that changes the PC value (jump, branch...): the SDMA jumps to the new PC value, leaves the *Debug* state and starts executing instructions from this new PC value.	exec_core <instruction> If the previous state was *Sleep after Reset*, the SDMA returns to this state, and Chn0Addr value overrides the PC value. Otherwise, the SDMA jumps to the new PC value and starts executing instructions from this new PC.

NOTE

The feature exec_core in *Debug in Sleep* after *Sleep after Reset* was added for the Channel boot (channel 0) to allow the debugger to return to *Sleep after Reset* state with a new PC

value. The SDMA will be ready to boot at the Chn0Addr address.

7.2.4.6 SDMA Clocks and Low Power Modes

The SDMA receives several root clocks from the SoC clock controller block and performs adaptive clock gating to optimize its power consumption. From a user standpoint, clock gating and power mode selection are fully automatized inside the SDMA.

Root clock control is available from the SoC clock controller block.

There are numerous clock sources that are used in the SDMA. They belong to one of two possible clock domains listed in the following table, and have frequency constraints within each domain. Clocks are considered asynchronous between domains.

Within the ARM platform/SDMA clock domain, all clocks must come from the same DPLL. The ARM platform DMA interfaces (peripheral DMA and burst DMA) receive their clock from the ARM platform DMA clock source whose frequency can be once or twice the frequency of the SDMA core clock. The DMA interfaces are designed to work at the ARM platform DMA frequency, but the SDMA core is physically limited to a maximum 104 MHz frequency. Since this is lower than the maximum ARM platform DMA frequency, the SDMA core clock is tied to the ARM platform peripheral clock frequency.

The ARM platform Peripheral Bus Clock source must be an exact sub-frequency of the SDMA Core clock source (any integer value greater or equal to 1).

Table 7-10. Clocking Scheme

Clock Domain	Source Clock	Comments
ARM platform	SDMA core (SDMA main core)	Source clock for the core and all its operations; this clock is thus used by most of the SDMA sub-blocks.
	ARM platform DMA	DMA interface for the peripheral DMA and the burst DMA. It is balanced with the main clock source, and its frequency is either once or twice the main clock frequency.
	ARM platform peripheral	Connection to the ARM platform peripheral bus. It is a sub-frequency of the main clock frequency.
JTAG	TCK	Clock for JTAG access, limited to maximum of 1/8 of the SDMA core clock frequency.

The JTAG clock is sampled by the SDMA main clock to determine its rising edge. This simplifies design and clock management, but it also adds a ratio constraint between those two clocks. It is guaranteed the JTAG interface works properly when the frequency of TCK is lower than $1/8^{th}$ of the frequency of the SDMA main clock (which is about 8 MHz when the SDMA core clock frequency is 66 MHz).

7.2.4.6.1 Clock Gating and Low Power Modes

The SDMA automatically performs power saving without requiring user involvement. It implements two levels of automatic clock gating.

7.2.4.6.1.1 Coarse Clock Gating

Every sub-block clock comes from one of the five available sources, and is gated with the sub-block specific enabling condition.

The following table displays the sub-block clocks and their source. It also indicates the relationships that may exist between different sub-blocks clock enables.

Table 7-11. Sub-blocks Clocks

Sub-block	Source Clocks	Enabling Condition and Comments	Related Enabling Conditions
Core	SDMA Main Core	The core sub-block clock is running when the core is not in one of its sleep states (Sleep or Sleep after Reset) or there is a pending channel. Typically, the core sub-block clock is stopped once all the channels are processed and the core enters its sleep state. A new pending channel awakes the core sub-block clock.	None
Memories	SDMA Main Core	The clock activation only occurs during a core access.	Disabled when Core sub-block clock is disabled or no memory access in progress
Scheduler	SDMA Main Core	Its clock only runs when scheduling is needed: for example, when there are pending channels, upon reception of a DMA request, and anytime the ARM platform modifies the channel running conditions.	None
ARM platform Control	SDMA Main Core & ARM platform peripheral	The ARM platform peripheral clock is solely used to determine the frequency ratio with the SDMA main clock. The control registers' clock is based on *SDMA main clock*; it is active when the ARM platform or the SDMA modifies the contents of one of these registers.	None
Burst DMA	SDMA Main Core & ARM platform DMA	The burst DMA has two clocks: The first clock is derived from the SDMA main core clock and drives registers that are connected to the FUBUS. The second clock is derived from the ARM platform DMA clock and drives registers that are connected to the ARM platform DMA bus outside the SDMA. Both clocks are enabled	Disabled when Core sub-block clock is disabled

Table continues on the next page...

Table 7-11. Sub-blocks Clocks (continued)

Sub-block	Source Clocks	Enabling Condition and Comments	Related Enabling Conditions
		during active phases of data transfers (for example, these clocks are turned off when the burst DMA is not used by the running channel script).	
Peripheral DMA	SDMA Main Core & ARM platform DMA	The peripheral DMA has two clocks: The first clock is derived from SDMA main clock and drives registers that are connected to the FUBUS. The second clock is derived from the ARM platform DMA clock and drives registers that are connected to the ARM platform DMA bus outside the SDMA. Both clocks are enabled during active phases of data transfers (for example, these clocks are turned off when the peripheral DMA is not used by the running channel script).	Disabled when Core sub-block clock is disabled
OnCE	SDMA Main Core	The OnCE clock is derived from main source clock. It is disabled by default. In order to use the OnCE, its clock must be explicitly turned on, either by enabling the OnCE access from the ARM platform peripheral bus (register ONCE_ENB), or by driving the clk_gating_off input pin high. This is a SDMA input whose driver depends on the SoC implementation (typically a JTAG controller). The OnCE also receives the TCK input, which is the JTAG clock. It does not use it as a functional clock; the TCK input is sampled instead. Refer to Synchronization Implementation.	When enabled, all other clocks are systematically on (clock gating is off)

7.2.4.6.1.2 Refined Clock Gating

The SDMA implements a second level of clock gating on a register-per-register basis.

Unlike the first level that covers all the SDMA flip-flops, except the synchronizers (only five flip-flops are always running), the second level is only available for eligible registers, which amounts to about 90% of the SDMA flip-flops.

These gated registers are only clocked when the hardware logic detects a new data loading. This additional gating further reduces dynamic power consumption.

7.2.4.6.1.3 Low Power Modes and User Control

Power savings are automatically managed by the SDMA hardware without any user involvement; however, one can distinguish three different power modes: SLEEP, RUN, and DEBUG.

The following table describes these modes, and shows how to switch from one mode to another.

Table 7-12. Power Modes

Power Mode	Sub-blocks							Comments
	Core	Memories	Scheduler	ARM platform Control	Burst DMA	Peripheral DMA	OnCE	
SLEEP	off[1]	off	wait[2]	wait	off	off	off	Set when the PCU state is either *Sleep* or *Sleep after Reset* and the SDMA is not in DEBUG mode. This is the default mode after reset.
RUN	on[3]	wait	wait	wait	wait	wait	off	Set for the other PCU states that are reachable out of debug: *Program*, *Data*, *Change of Flow*, *Error in Loop*, *Debug*, *Functional Unit*, *Save,* or *Restore*.
DEBUG	on	on	on	on	on	on	on	Set regardless of the PCU state when clock gating is turned off to use the OnCE features (either *clk_gating_off* pin high or ONCE_ENB[0] set).

1. *off*: no clock
2. *wait*: only clocked when accessed or stimulated
3. *on*: clock is always running

It is possible to control the SDMA power mode. The procedures to force the SDMA into either mode are described in SLEEP Mode.

7.2.4.6.1.3.1 SLEEP Mode

This is the default mode after reset; therefore, resetting the SDMA forces this mode.

However, the common procedure is as follows:

- Ensure the *clk_gating_off* pin is low and ONCE_ENB[0] is cleared.
- Disable all channels (via the STOP_STAT control register, and the HO, DO, EO if necessary).
- Wait for the active channels to complete or force a reschedule via the reschedule bit in the RESET register.
- The SDMA is in SLEEP mode making it possible to completely shut off its clock from the chip level clock controller using the procedure described in Stop Mode Response.

7.2.4.6.1.3.2 RUN Mode

This is the default mode when a channel is running:

- Ensure the *clk_gating_off* pin is low and ONCE_ENB[0] is cleared.
- Activate at least one channel (via the HSTART control registers, a DMA request, and/or the HO, DO, EO register bits).

7.2.4.6.1.3.3 DEBUG Mode

The DEBUG mode must be set when one needs to use the debugging facilities of the SDMA.

- Ensure the SDMA clocks are running from the CCM.
- Set the *clk_gating_off* pin high or use the SDMA to set ONCE_ENB[0].

7.2.4.6.1.4 Stop Mode Response

The SDMA receives a stop request from the chip level clock controller. This request may be asserted when the chip enters the stop low power mode.

If the SDMA is running when the request is received, then the SDMA will complete all pending channels before returning to the SLEEP state. The SDMA sends an acknowledgement to the clock controller when the SLEEP state is entered indicating that the SDMAs clocks can be turned off.

7.2.4.6.2 Reset

After reset (either received from the reset block or a software reset required by the ARM platform), the SDMA is in IDLE mode. It will start its boot code located at address 0 once a channel is activated.

Activating a channel can be done by the ARM platform after programming a positive priority and setting the channel bit in the EVTPEND register.

There will not be a context RESTORE for the first channel (bootload channel) called after a reset because the context data in RAM has not been initialized. Static context mode should be used for the first channel called after reset to ensure that the all context RAM for that channel is initialized. Subsequent calls to the same channel or different channels may use any of the dynamic context modes

7.2.4.7 Software Interface

Appendix A fully describes the SDMA Application Programming Interface (API).

7.2.4.8 Initialization Information

This section discusses the following:

- Hardware Reset
- Channel Script Execution
- Initialization and Script Execution Setup Sequence

7.2.4.8.1 Hardware Reset

After reset, the program RAM, context RAM, data RAM, and RAM containing the channel enable registers (CHNENBLn) have unpredictable contents.

The active register set is assigned to channel 0 and the PC is initialized to all zeros. However, since the channel enable register is all zeros, there are no active channels and the SDMA is halted waiting for the boot channel to start.

The ARM platform will have to setup the SDMA in order to boot it. The CONFIG register must be initialized to determine the DMA/core clock ratio (1 or 2). Channel Enable Registers must also be initialized.

To start up the SDMA, the ARM platform first creates some channel control blocks (CCB) and buffer descriptors (BD) in ARM platform memory for the boot channel (channel 0) and then initializes the channel 0 pointer register (SDMA_MC0PTR) to the address of the first control block. Data Structures for Boot Code and Channel Scripts provides an overview of the data structure for the CCB and BD's. The SDMA_HSTART, SDMA_HOSTOVR and SDMA_EVTOVR registers are then configured according to Runnable Channels Evaluation to allow channel 0 to run.

Upon being enabled, the SDMA begins executing the script located at the address indicated by the Channel 0 Boot Address register (SDMA_CHN0ADDR) in the program memory. The reset value of SDMA_CHN0ADDR points to the default bootload script in ROM. This ROM script will read the channel 0 pointer register (SDMA_MC0PTR) to determine the location of the Channel Control Block (SDMA_CCB) in ARM platform memory. The script will then begin fetching by DMA the first channel control block which contains a pointer to the location channel 0 Buffer Descriptor chain which is also fetched via DMA. If the buffer descriptor contains a valid command, the script interprets the command in each buffer descriptor and proceeds to implement the command and move on to the next buffer descriptor control block. The buffer descriptor commands for channel zero are typically set up to load SDMA's program RAM, Data RAM, and initial values for the channel contexts. Some channel scripts expect particular parameters to be passed

There are two ways to make the SDMA boot on a user-defined script. The OnCE (either via its JTAG interface or its ARM platform Control interface) can be used to download any code in the SDMA RAM and force the SDMA to boot on that code. Also, the SDMA_CHN0ADDR register in the ARM platform programming model can be modified to point to user code in RAM which would need to either have been loaded via the ONCE or default bootload routine (ex before a S/W reset).

7.2.4.8.2 Channel Script Execution

The execution of an SDMA script depends on both the instructions that make up the script, the data context upon which it operates, and commands or parameters allowed to the buffer. All these items must be initialized before the script is allowed to execute.

Each of the 32 channels has a separate context, but may share scripts and locations in data RAM.

Channels are initialized by the ARM platform by using channel 0 to download any required scripts and data values and the channels initial context. The context contains all the initial values of the SDMA core registers. This includes the Program Counter (PC) which is set to the start of the desired script in SDMA program memory.

The ARM platform selects which trigger conditions that must occur for the channel to start by configuring the SDMA_CHNENBL, SDMA_HOSTOVR and SDMA_EVTOVR registers. The trigger events include ARM platform setting HE (SDMA_HSTART) or a hardware DMA request asserts an event input to SDMA. The channel can become active according to its priority compared with other runnable channels when the selected trigger(s) cause the condition described in Runnable Channels Evaluation to evaluate as true.

The specific parameters to be passed to each script in the buffer descriptor or context are documented in the software documentation for each script. Please refer to SDMA Scripts for complete script documentation. Buffer Descriptor Format provides an overview of the buffer descriptor format.

7.2.4.8.3 Initialization and Script Execution Setup Sequence

To summarize, the following steps are minimally required to setup SDMA and run channel scripts.

- Perform Hardware Reset. The program RAM, context RAM, data RAM and SDMA_CHNENBLn registers have unpredictable contents after this reset.
- Initialize SDMA_CHNENBLn registers to map DMA request events to desired channels.

- Configure SDMA_CHNPRIn registers to select priority for runnable channels. A non-zero priority is required for the channel to run.
- Configure the SDMA_CONFIG register to select DMA to SDMA core clock ratio .
- Set up channel control blocks and buffer descriptors in ARM platform to specify the loading of SDMA program RAM and channel contexts for each SDMA channel to be used. Reference Data Structures for Boot Code and Channel Scripts.
- Configure SDMA_MC0PTR register with base address of ARM platform Channel Control Block base address.
- Initialize SDMA_CHNENBLn registers to map DMA request events to associated channel. Reference Mapping DMA Requests to Pending Channels.
- Configure SDMA_CHNPRIn registers to set priority for each channel to be run.
- For each channel to be run, configure SDMA_HOSTOVR (HO) and SDMA_EVTOVR (EO) registers to select which events (hardware and/or software trigger events) must occur for the channel to be runnable. Reference Runnable Channels Evaluation.
- Set bit 0 of the SDMA_HSTART register to set HE[0] and allow Channel 0 to run (assumes EO[0] andDO[0] were both set in previous step). This will cause SDMA to load the program RAM and channel contexts configured previously.
- Wait for Channel 0 to finish running. This is indicated by HI[0]=1 in the SDMA_SDMA_INTR register, or by optional interrupt to the ARM platform.
- Set the LOCK bit in the SDMA_SDMA_LOCK register to prevent un-authorized uploads of data to SDMA RAM.
- Additional channel scripts can now be run by enabling the selected software or hardware trigger event according to Runnable Channels Evaluation.

7.2.4.9 SDMA Programming Model

This section describes the programming model for the SDMA RISC engine, including its processor, memory, and internal control registers.

All addresses are related to the internal SDMA memory map, which is completely different from the ARM platform memory maps. The ARM platform processor has no access to any hardware resource described, except when those resources are described in ARM Platform Memory Map and Control Register Summary. .

7.2.4.9.1 State and Registers Per Channel

The SDMA can be seen as a set of 32 identical devices that are able to perform one data transfer channel each. Only one channel can work at a time, but every channel state is available at any time.

This chapter lists the components of every channel state.

7.2.4.9.2 General Purpose Registers

Each channel has eight general purpose registers of 32 bits for use by scripts. General register 0 has a dedicated function for the loop instruction, but otherwise can be used for any purpose.

7.2.4.9.3 Functional Unit State

Each channel context has some state that is part of the functional units.

The specific allocation of this state is part of the functional unit definition that is described in Burst DMA Unit Programming, Peripheral DMA Unit Programming .

This state must be saved/restored on context switches.

7.2.4.9.3.1 Program Counter Register (PC)

The PC is 14 bits. Since instructions are 16 bits in width and all memory in the SDMA is 32 bits in width, the low order bit of the PC selects which half of the 32-bit word contains the current instruction.

A low order bit of zero selects the most significant half of the word.[1]

7.2.4.9.3.2 Flags

Each channel has the following four flags:

- The T bit reflects the status of some arithmetic and test instructions. It is set when the result of an addition or a subtraction is zero and cleared otherwise. It is also the copy of the tested bits. Finally, it can also be set when the loop counter (GReg0) reaches zero. When the last instruction of the hardware loop is an operation that can modify the T flag, its effect on T is discarded and replaced by the GReg0 status.
- Two additional bits, SF and DF, are used to indicate error conditions resulting from loading data sources and storing to destinations, respectively. Access errors set these bits, and successful transactions clear them. They can also be cleared by specific instructions (CLRF and loop). The source fault (SF) is updated by the loads LD and LDF; the destination fault (DF) is updated by the stores ST and STF.

1. For example, big-Endian.

- Access errors are caused by several conditions including writing to the ROM, writing to a read-only memory mapped register, accessing an unmapped address, or any transfer error received by a peripheral when it is accessed.

  The SF and DF flags have a major impact on the behavior of the hardware loop: If SF or DF is set when starting a hardware loop and it is not masked by the loop instruction, the loop body will not be executed. Inside the loop body, if a load or store sets the corresponding SF or DF flag, the loop exits immediately. Testing the status of the T flag at the end of the loop (as well as testing both SF and DF) tells if the loop exited abnormally as any anticipated exit prevents GReg0 from reaching the zero value and thus setting the T flag. This is also valid if the fault occurs at the last instruction of the last loop.

- The last flag is the loop mode flag, LM, which is composed of two bits. The most significant bit indicates when the processor is currently operating in loop mode. It is set by the loop instruction and is cleared after execution of the last instruction of the last loop. The least significant bit is set when the program counter points to the last instruction of a loop on the last path. It is used for a channel that is restored with this configuration to know that the next program counter is EPC. As with the dynamic context switch Greg0, which indicates when the program must get out of the loop, it can be restored only on the last instruction of the loop. This, however, is too late to fetch the next instruction after the loop.

7.2.4.9.3.3 Return Program Counter (RPC)

The RPC is 14 bits. It is set by the jump to the subroutine instructions and used by the return from the subroutine instructions.

Instructions are available to transfer its contents to and from a general register.

7.2.4.9.3.4 Loop Mode Start Program Counter (SPC)

The SPC is 14 bits. It is set by the loop instruction to the location immediately following it.

7.2.4.9.3.5 Loop Mode End Program Counter (EPC)

The EPC is 14 bits. It is set by the loop instruction to the location of the next instruction after the loop.

7.2.4.9.4 Context Switching-Programming

Each channel has a separate context consisting of the eight general purpose registers and additional registers representing the state of the functional units.

The active registers and functional units contain the context of the active channel. The context of inactive channels is stored in SDMA RAM, which is part of the SDMA address space.

In a function of the selected context switching mode (Context Switching), modified registers by the program can be saved in the channel RAM space while the program is going on. In every cycle, a write access to the RAM is possible.

On a done or yield(ge) instruction, SDMA goes into "real" context switching. In one of the dynamic modes, modified registers not previously saved, as well as the PC-Loop registers, are stored into the context area of the channel that will be closed. The new PC-Loop registers are loaded from the context area of the new channel. All other registers are restored while the program is executed, giving priority to registers used by the decoded instruction. Therefore, in the best case, only the PC and Loop registers should be saved and restored during this context-switching phase, which only requires five SDMA cycles.

In static mode, the context switch stores all registers in the old channel RAM space, and restores all registers from the new channel RAM space. It requires 26 SDMA cycles.

The address of the context memory for channel *i* is *CONTEXT_BASE* + 24**i* or *CONTEXT_BASE* + *32*i w*here CONTEXT_BASE equals 0x0800. The table below presents the layout of a channel context in memory:

Table 7-13. Layout of a Channel Context in Memory for SDMA

OFFSET	31	30	29-16	15	14	13-0
0	SF	-	RPC	T	-	PC
1	LM		EPC	DF	-	SPC
2	GR0					
3	GR1					
4	GR2					
5	GR3					
6	GR4					
7	GR5					
8	GR6					
9	GR7					
10	MDA (burst DMA)					
11	MSA (burst DMA)					
12	MS (burst DMA)					
13	MD (burst DMA)					

Table continues on the next page...

Table 7-13. Layout of a Channel Context in Memory for SDMA (continued)

14	PDA (peripheral DMA)
15	PSA (peripheral DMA)
16	PS (peripheral DMA)
17	PD (peripheral DMA)
18	
19	
20	Reserved[1]
21	Reserved[1]
22	Reserved[1]
23	Reserved[1]
24	Scratch RAM (optional)
25	Scratch RAM (optional)
26	Scratch RAM (optional)
27	Scratch RAM (optional)
28	Scratch RAM (optional)
29	Scratch RAM (optional)
30	Scratch RAM (optional)
31	Scratch RAM (optional)

7.2.4.9.5 Address Space

The SDMA has four internal buses which are listed here.

- The Instruction bus reads instructions from the memory. Its address map is described in Instruction Memory Map.
- The Data bus (DMBUS) accesses the same memories as those visible on the Instruction bus, some memory-mapped registers (scheduler status and OnCE registers), and up to 14 peripherals. Its address map is described in Data Memory Map.
- The Functional Units bus (FUBUS) accesses the , Burst DMA, Peripheral DMA . The addressing mechanism is further detailed in Functional Units Programming Model.
- The Context Switch bus reads/writes registers into context-switch RAM space. It is a 64-bit bus dedicated for accessing this RAM space for updating the context of the running channel. While the program is going on, this bus has the lowest priority compared to the Instruction and Data buses, except for restoring a register needed for the decoded instruction to be executed. On the save part of a context switch (when the PCU is in its slave state), this is the only one used. On the restore part, the Instruction bus has the priority to read the next instruction at the restored PC and

otherwise the Context Switch bus is used. It is not possible to control the actual data transfers that occur on this bus.

7.2.4.9.5.1 Instruction Memory Map

The instruction memory map is based on a 14-bit address bus and a 16-bit data (instruction) bus.

Instructions are fetched from either program ROM or program RAM. An SDMA script is able to change the contents of the program RAM, which is also visible from the data bus.

The first two instruction locations (at 0 and 1) are special. Location 0 is where the PC is set on reset. Location 1 is where the PC is set upon the execution of an illegal instruction. It is expected that both of these locations will contain a jmp to handle routines.

Table 7-14. SDMA Instruction Memory Space

Device	SDMA Address (Hex)	Base Address Label	Block Name	WS	Description
ROM	0x0000 ↓ 0x07FF	SDMA_IBUS_ROM_ADDR	-	0	4 Kbyte internal ROM with boot code and standard routines.
RAM	0x1000 ↓ 0x1FFF	SDMA_IBUS_RAM_ADDR	-	0	8 Kbyte internal RAM with channels context and user data/routines.

7.2.4.9.5.2 Data Memory Map

All of the data accessible to SDMA scripts make up the data memory space of the SDMA.

This address space has several components:

- ROM (also visible on the Instruction bus)
- RAM (also visible on the Instruction bus)
- Shared Peripherals Registers
- SDMA Internal Registers (scheduler, OnCE, and registers that are also accessible by the ARM platform)

SDMA scripts can read and write to the context RAM, data RAM, shared peripheral registers, and internal registers.

The address range is 16 bits and the data width is 32 bits. When accessing peripheral registers (USB and so on), the data width may be different. The exact address map for the peripherals depends on the project (as presented in each respective chapter).

Data access is performed with *ld* and *st* instructions that take the address from a general purpose register in the core (GReg*n*). The mapping between the general purpose register contents and the address bus is given in the following table:

Table 7-15. GReg*n* to DMBUS Address Mapping

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
sz	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
address															

Grayed bits are simply discarded but they must be cleared to ensure forward-script compatibility.

- sz (bit 31) indicates the peripheral data width: 0 is used for a 32-bit peripheral and 1 is used for a 16-bit peripheral.
- address (bits 15 down to 0) is the address of the accessed resource (internal memory, internal register, or shared peripheral).

Table 7-16. SDMA Data Memory Space

Device	SDMA Address (Hex)	Size	Description
ROM	0x0000 → 0x03FF	4 Kbyte	4 Kbyte internal ROM with boot code and standard routines
Reserved	0x0400 → 0x07FF	4 Kbyte	4 Kbyte Reserved
RAM	0x0800 → 0x0FFF	8 Kbyte	8 Kbyte internal RAM with channels contexts and user data/routines
per1	0x1000 → 0x1FFF	16 Kbyte	*peripheral 1* memory space (4 Kbyte peripheral's address space)
per2	0x2000 → 0x2FFF	16 Kbyte	*peripheral 2* memory space (4 Kbyte peripheral's address space)
per3	0x3000 → 0x3FFF	16 Kbyte	*peripheral 3* memory space (4 Kbyte peripheral's address space)
per4	0x4000 → 0x4FFF	16 Kbyte	*peripheral 4* memory space (4 Kbyte peripheral's address space)
per5	0x5000 → 0x5FFF	16 Kbyte	*peripheral 5* memory space (4 Kbyte peripheral's address space)
per6	0x6000 → 0x6FFF	16 Kbyte	*peripheral 6* memory space (4 Kbyte peripheral's address space)
Registers	0x7000 → 0x7FFF	16 Kbyte	Memory mapped registers
per7	0x8000 → 0x8FFF	16 Kbyte	*peripheral 7* memory space (4 Kbyte peripheral's address space)
per8	0x9000 → 0x9FFF	16 Kbyte	*peripheral 8* memory space (4 Kbyte peripheral's address space)
per9	0xA000 → 0xAFFF	16 Kbyte	*peripheral 9* memory space (4 Kbyte peripheral's address space)
per10	0xB000 → 0xBFFF	16 Kbyte	*peripheral 10* memory space (4 Kbyte peripheral's address space)
per11	0xC000 → 0xCFFF	16 Kbyte	*peripheral 11* memory space (4 Kbyte peripheral's address space)
per12	0xD000 → 0xDFFF	16 Kbyte	*peripheral 12* memory space (4 Kbyte peripheral's address space)
per13	0xE000 → 0xEFFF	16 Kbyte	*peripheral 13* memory space (4 Kbyte peripheral's address space)
per14	0xF000 → 0xFFFF	16 Kbyte	*peripheral 14* memory space (4 Kbyte peripheral's address space)

7.2.4.10 SDMA Initialization

Appendix A describes the setup of the SDMA . This section provides a quick description of several initialization procedures.

NOTE

There may be differences with the actual implementation in the API.

7.2.4.10.1 Hardware Reset-SDMA

After reset, the RAM that holds contexts, data, scripts, and the DMA request-channels matrix has unpredictable content.

The core registers are all reset to 0, including the PC; the PCU state is *Sleep after Reset*. No channel can be activated because all of the priorities are also reset to 0.

7.2.4.10.2 Standard Boot Sequence

The following is the standard boot sequence:

1. Initialize the CONFIG register-detailed in Configuration Register (SDMAARM_CONFIG)-to determine the ARM platform DMA/core clock ratio (1 or 2)
2. Initialize the DMA request-channels matrix (seeChannel Enable RAM (SDMAARM_CHNENBL*n*)).
3. Program the channel control registers-Channel Event Override (SDMAARM_EVTOVR), Channel BP Override (SDMAARM_DSPOVR), Channel BP Override (SDMA_HOSTOVR), and Channel Event Pending (SDMAARM_EVTPEND)-according to the channel allocation.
4. Perform any necessary setup as required by the standard boot script in ROM (this is described in Appendix A).
5. Trigger channel 0 with the Channel Start (SDMAARM_HSTART) register, which starts the execution of the ROM script starting at address 0. This boot downloads channel scripts and contexts in RAM.

7.2.4.10.3 User-Defined Boot Sequence

The following is a user-defined boot sequence:

1. Initialize the Configuration Register (SDMAARM_CONFIG)Channel Enable RAM (SDMAARM_CHNENBL*n*), Channel Event Override (SDMAARM_EVTOVR), Channel BP Override (SDMAARM_DSPOVR), Channel ARM platform Override (SDMAARM_HOSTOVR), and Channel Event Pending (SDMAARM_EVTPEND).

2. Use the OnCE (either via its JTAG interface or its ARM platform control registers) to download any code in the SDMA RAM. Accessing the Memory describes how to write data to the RAM via the OnCE.
3. Use the OnCE instructions to make the PC default value point to the new boot script start address, or rely on the ROM startup script, which first jumps to the address in Channel 0 Boot Address (SDMAARM_CHN0ADDR). (This register default address points to the standard boot script.)

7.2.4.10.4 Script Loading and Context Initialization

The execution of an SDMA script depends on both the instructions that make up the script and the data context upon which it operates. Both must be initialized before the script is allowed to execute.

Each of the 32 channels has a separate data context, but may share scripts and locations in the data RAM.

The ARM platform manages the space in program RAM and data RAM. It also manages the assignment of SDMA channels to the device drivers that need them. Channels are initialized by the ARM platform via the channel 0 boot script. The boot channel downloads any required scripts with their data and the channels' initial contexts. Every context contains all the initial values of the registers, including the PC. Then the ARM platform can enable any channel that becomes active and begins fetching and executing instructions from its script.

7.2.4.11 Instruction Description

The following sections introduce the instruction of the SDMA.

Instruction set details are available in Instruction Set.

7.2.4.11.1 Scheduling Instructions

The following are scheduling instructions:

- done-The instruction causes certain scheduling or interrupt bits to be set or cleared, which may cause a change in the schedule-ability of the running channel. Then the instruction causes the SDMA to evaluate the current scheduling priorities and to choose the highest priority ready channel. If this channel is not the current channel, a context switch will take place. If there are no runnable channels, the SDMA will enter the stopped mode. The done 5 has a special usage reserved for debug, as explained in Debug Instructions.

- yield-These instructions are special cases of the done instruction. They do not modify the scheduling bits, but allow the highest pending channel (if it exists) to preempt the current channel if the pending channel priority is strictly greater than the current channel priority.
- yieldge-These instructions are special cases of the done instruction. They do not modify the scheduling bits, but allow the highest pending channel (if it exists) to preempt the current channel if the pending channel priority is strictly greater or equal to the current channel priority.
- notify-The notify instruction affects the scheduling bits, but does not cause rescheduling.

7.2.4.11.2 Conditional Branch Instructions

The conditional branch instructions of an 8-bit displacement, which is sign-extended and added to the current PC (which points to the next instruction) if the condition is satisfied.

Otherwise, control passes to the next sequential instruction.

- BF-Branch if False. The branch is taken if the T bit in the processor status is zero (false).
- BT-Branch if True. The branch is taken if the T bit in the processor status is one (true).
- BSF-Branch if Source Fault. The branch is taken if the SF bit in the processor status is one.
- BDF-Branch if Destination Fault. The branch is taken if the DF bit in the processor status is one.

7.2.4.11.3 Unconditional Jump Instructions

There are two varieties of unconditional control transfers: an absolute transfer and a through-register transfer.

Absolute transfers have a 14-bit address field that replaces the current PC.

- JMP-Jump. Causes the processor to jump to an absolute address encoded in the instruction itself.
- JSR-Jump to Subroutine. Causes the processor to jump to a subroutine, the address of which is encoded in the instruction itself.

- JMPR-Jump through Register. Causes the processor to jump to an absolute address contained in a General register. This instruction is meant to be used when more than one level of subroutines are required.
- JSRR-Jump to Subroutine through Register. Causes the processor to jump to a subroutine, the address of which is contained in a General register. This instruction is meant to be used when more than one level of subroutines are required.

7.2.4.11.4 Subroutine Return Instructions

The following are subroutine return instructions:

- RET-Return from Subroutine. The RET restores the contents of RPC to PC.
- LDRPC-Load from RPC to Register. THe LDRPC instruction is meant to be used when more than one level of subroutines are required. It stores the contents of RPC in any General register.

7.2.4.11.5 Loop Instruction

The following is a loop instruction:

LOOP-Enters Loop Mode. Before entering loop mode, the loop instruction can optionally clear the fault flags (SF and/or DF) based on a 2-bit field in the instruction. This feature is linked to the fact that setting SF or DF in loop mode will cause an immediate exit of the loop.

7.2.4.11.6 Miscellaneous Instructions

The following are miscellaneous instructions:

- CLRF-Clear Fault Flags. This instruction clears any combination of SF and DF.
- MOV r,s-This moves data from GReg[s] to GReg[r].
- LDI r,immediate-This loads GReg[r] with a zero-extended immediate value.

7.2.4.11.7 Logic Instructions

The following are logic instructions:

- XORr,s-This performs an exclusive or between GReg[r] and GReg[s], and stores the result in GReg[r].
- XORIr,immediate-This performs an exclusive or between GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].
- ORr,s-This performs an or between GReg[r] and GReg[s], and stores the result in GReg[r].

- ORIr,immediate-This performs an or between GReg[r] and a zero-extended immediate value and, stores the result in GReg[r].
- ANDNr,s-This performs an and between GReg[r] and the negated GReg[s], and stores the result in GReg[r].
- ANDNIr,immediate-This performs an and between GReg[r] and the negated zero-extended immediate value, and stores the result in GReg[r].
- ANDr,s-This performs an and between GReg[r] and GReg[s], and stores the result in GReg[r].
- ANDIr,immediate-This performs an and between GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].

7.2.4.11.8 Arithmetic Instructions

Arithmetic instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if the result is zero, otherwise it is cleared.

- ADD r,s-This performs the addition of GReg[r] and GReg[s], and stores the result in GReg[r].
- ADDI r,immediate-This performs the addition of GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].
- SUB r,s-This performs the subtraction of GReg[s] from GReg[r], and stores the result in GReg[r].
- SUBIr,immediate-This performs the subtraction of a zero-extended immediate value from GReg[r], and stores the result in GReg[r].

7.2.4.11.9 Compare Instructions

Compare instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if the comparison is true, otherwise it is cleared.

NOTE

Only one version of the immediate form is implemented. Non-equality comparisons to immediate values will require two instructions.

- CMPEQ r,s-This sets T when registers GReg[r] and GReg[s] are equal.
- CMPEQIr,immediate-This sets T when register GReg[r] and the zero-extended immediate value are equal.
- CMPLTr,s-This sets T when register GReg[r] is less than and not equal to GReg[s]. The comparison is signed.
- CMPHS r,s-This sets T when register GReg[r] is greater than or equal to GReg[s]. The comparison is signed.

7.2.4.11.10 Test Instructions

Test instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if any bit in the result is one, otherwise it is cleared.

- TSTr,s-This performs an and between GReg[r] and GReg[s], and sets T if the result is not zero.
- TSTIr,immediate-This performs an and between GReg[r] and a zero-extended immediate value, and sets T if the result is not zero.

7.2.4.11.11 Byte Permutation Instructions

These instructions shuffle the bytes in a register. For the purpose of describing these instructions, have the bytes in a register be numbered from the most significant as b_3, b_2, b_1, b_0.

- RORBr-The rotate right byte. The result is b_0, b_3, b_2, b_1.
- REVBr-The reverse bytes in word. The result is b_0, b_1, b_2, b_3.
- REVBLOr-The reverse, two low-order bytes. The result is b_3, b_2, b_0, b_1.

7.2.4.11.12 Bit Shift Instructions

The following are bit shift instructions:

- ROR1r-The rotate right 1 bit. This instruction does a circular right shift of 1 bit.
- LSR1r-The logical shift right 1 bit. This instruction shifts all bits to the right by 1. The high order bit is replaced by a 0.
- ASR1r-The arithmetic shift right 1 bit. This instruction shifts all bits to the right by 1. The high order bit is replaced by itself.
- LSL1r-The logical shift left 1 bit. This instruction shifts all bits to the left by 1. The low order bit is replaced by zero.

7.2.4.11.13 Bit Manipulation Instructions

- BCLRIr,n-The bit clear is immediate; clears bit number *i* in register *r*.
- BSETIr,n-The bit set is immediate; sets bit number *i* in register *r*.
- BTSTIr,n-The bit test is immediate; tests bit number *i* in register *r* (T becomes equal to the selected register bit).

7.2.4.11.14 SDMA Memory Access Instructions

All memory accesses are 32 bits.

Any memory location that is implemented with less than 32 bits (for example, peripheral registers) causes unimplemented bits to be read as 0s.

All memory accesses will cause either the SF or DF flags in the processor status to be set if they cause a fault.

What constitutes a fault, especially when accessing peripheral registers, is a property of the memory location.

- LDr,(b,d)-The load instruction creates an address by adding the displacement field (d) to the contents of the base register (b). The SDMA location at the resulting address is read and placed in the destination register (r).
- STr,(b,d)-The store instruction creates an address in the same manner as the load instruction. The register (r) is stored in the SDMA location at the resulting address.

7.2.4.11.15 Functional Unit Instructions

The functional unit instructions have an 8-bit field that is placed on the functional unit bus.

Some of these bits are used to select which functional unit should be involved in the transfer. The remaining bits are decoded by the selected functional unit so their specific use depends on the functional unit. See Functional Units Programming Model.

There are two functional unit instructions, as follows:

- LDFr,fub-The 8-bit field is placed on the functional unit bus and a read is issued to the selected functional unit. As a result of this instruction, the SF may be set in the processor status.
- STFr,fub-The 8-bit field is placed on the functional unit bus and a write is issued to the selected functional unit. As a result of this instruction, the DF may be set in the processor status.

7.2.4.11.16 Illegal Instructions

All instruction encodings that are illegal cause the following actions:

- The current PC (which points to one beyond the offending instruction) is put in the EPC register.
- The loop mode bit is cleared.
- The PC is set to the value stored in the Illegal Instruction Trap Address (SDMAARM_ILLINSTADDR) register (the default value is 0x0001).

ILLEGAL-Although any instruction other than those indicated in the SDMA specification will trigger the illegal instruction mechanism, the ILLEGAL instruction code is preferred as it will always be kept as *illegal* in the possible future versions of the SDMA core.

7.2.4.11.17 Debug Instructions

The following are debug instructions:

- SOFTBKPT-The software breakpoint instruction causes the core to stop and enter debug mode. The core can then be accessed and started by the OnCE debug block only.
- done 5-This instruction is used for debugging, as it copies the contents of the PCU registers and flags to the context memory. Information on this instruction is described in Saving the Context.
- CpShReg-This instruction copies the context memory into the PCU registers and flags. Modifying the corresponding memory location before executing this instruction enables you to have the channel continue from a new instruction address. This instruction is described in Restoring the Context.

7.2.4.12 Functional Units Programming Model

The functional unit instructions cause an 8-bit code, found in the low eight bits of the instruction, to be asserted on the functional unit control bus.

Some of these bits are used to select one of several functional units. Functional units which can be selected include SDMA registers such as MSA and MSD which are not mapped in the SDMA memory map, and are accessible only through the functional unit bus. These Functional Unit Registers are listed in the following table. In order to establish a programming convention, assume the selection bits are some number of the most significant bits of the 8-bit code. Furthermore, some number of the least significant bits is decoded by a given functional unit to establish the type of operation to perform.

Table 7-17. Functional Unit Registers

Functional Unit	Register	Register Name	Section/Page
Burst DMA Unit Programming	SDMSA	Memory Source Address Register	Memory Source Address Register (MSA)
	MDA	Memory Destination Address Register	Memory Destination Address Register (MDA)
	MD	Memory Data Buffer Register	Memory Data Buffer Register (MD)

Table continues on the next page...

Table 7-17. Functional Unit Registers (continued)

Functional Unit	Register	Register Name	Section/Page
			(Write) Burst DMA Write (stf) (Read) Burst DMA Read (ldf)
	MS	Memory State Register	State Register (MS)
Peripheral DMA Unit Programming	PSA	Peripheral Source Address Register	Peripheral Source Address Register (PSA)
	PDA	Peripheral Destination Address Register	Peripheral Destination Address Register (PDA)
	PD	Peripheral Data Buffer Register	Peripheral Data Register (PD) (Write) Peripheral DMA Write (stf)-Write Mode (Read) Peripheral DMA Read (ldf)-Read Mode
	PS	Peripheral State Register	Peripheral State Register (PS)

More information regarding the functional units can be found in Peripheral DMA Unit, and Burst DMA Unit.

7.2.4.12.1 Burst DMA Unit Programming

The DMA instructions control the DMA state machine and may cause a DMA cycle on the associated memory bus.

There are four registers associated with the burst DMA unit: a Memory Source Address register (MSA), a Memory Destination Address register (MDA), a Memory Data buffer (MD), and a state register (MS). The burst DMA has two different uses:

- A data transfer between External Memory Interface and SDMA general register
- A data transfer in copy mode where blocks of data are transferred from the source address to the destination address

7.2.4.12.1.1 Memory Source Address Register (MSA)

The source address register contains the pointer into EXTMC memory associated with the next read data transfer. It has byte granularity.

Reading the register with the ldf instruction has no side effects, and gives the address value in the EXTMC memory of the next data that is read by the SDMA during an ldf MD instruction.

Writing the source address register has two side effects: If the prefetch bit is set, a DMA read cycle (8-word read access) is issued with the new address. Any data still located in the buffer is lost. If there is valid write data in the buffer, it is necessary to force the DMA to completely flush it out before modifying MSA to guarantee all the data is effectively written to memory.

The MSA register has two modes of programming:

- Frozen-In frozen mode, the MSA register is not modified after DMA accesses.
- Incremented (default mode)-In incremental mode, MSA is incremented by the number of bytes transferred during read cycles.

7.2.4.12.1.2 Memory Destination Address Register (MDA)

The destination address register contains the pointer into EXTMC memory associated with the next write data transfer. It has byte granularity.

Reading the MDA register with the ldf instruction has no side effects. It gives the address value in the EXTMC memory where the next SDMA data (stf r,MD instruction) is stored when MD FIFO is flushed.

Writing the destination address register has one side effect. Any data still located in the buffer is lost. If there is valid write data in the buffer, it is necessary to force the DMA to completely flush it out before modifying MDA to guarantee all the data is effectively written to memory.

The MDA register has two modes of programming:

- Frozen-In frozen mode, the MDA register is not modified after DMA accesses.
- Incremented (default mode)-The MDA register is incremented by the number of bytes transferred during write cycles.

7.2.4.12.1.3 Memory Data Buffer Register (MD)

The data buffer register consists of a bank of 36 bytes that behave like FIFO.

This FIFO stores the eight words received when a read burst is triggered by the DMA (DMA is in read mode).

The MD register is in write mode after a writing in MDA or after an stf MD instruction.

In that case, a burst write access is automatically triggered when there are more than eight words in MD. For bandwidth optimization, any transfers between DMA and the EXTMC controller are based on burst accesses.

An ldf r,MD|SIZE instruction that reads the data buffer may cause a DMA cycle, as follows:

- If there are less bytes in the FIFO than the size parameter of the instruction. For instance, if only two bytes are available in MD and a 4-byte read is requested, a burst read access is executed to complete the two bytes.
- If the prefetch bit is set, and after reading there is enough space in the FIFO to store a full burst, a burst read access is triggered.

An stf r,MD|SIZE instruction that writes to the data buffer may cause a DMA cycle if the number of written bytes in MD is higher than 32 (eight words) or if the flush bit is set.

When DMA is used for data transfer between SDMA and EXTMC (reading or writing), no immediate error is possible because the block manages a data misalignment issue; therefore, it is allowed to read/write a word to/from a half-word address. However, the addresses (source or destination) must belong to the EXTMC memory mapping. The only potential error, in this mode, would be the error sent back by the EXTMC controller when an access to a super-user page is detected. The whole transfer on the DMA associated bus will be considered successful when there are no errors seen on the bus during the transfer. In copy mode, an immediate error could be returned to SDMA as described in Burst DMA Unit Error Management.

7.2.4.12.1.4 State Register (MS)

The state register contains the DMA state-machine value. It can be accessed in case of an error received during a transfer. MS is also accessed to set-up the conditional yielding feature.

The initialization value of this register is 0 and it consists of the following:

Table 7-18. SDMA_MS Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	spriv	stype	0	0	dpriv	dtype
W																
R	0	0	0	0	y	d	e		0	0	n					
W																

Table 7-19. SDMA_MS Field Descriptions

Field	Description
31-22	Reserved

Table continues on the next page...

Table 7-19. SDMA_MS Field Descriptions (continued)

Field	Description
21 spriv	The spriv value is ignored for this device. 0 = valid value 1 = Reserved
20 stype	Source Mode. Indicates if MSA has to be incremented (or not) during accesses. 0 Frozen-MSA is not modified. 1 Incremented-MSA is incremented by the number of transferred bytes during read access.
19-18	Reserved
17 dpriv	The dpriv value is ignored for this device. 0 = valid value 1 = Reserved
16 dtype	Destination Mode. Indicates if MDA has to be incremented (or not) during accesses. 0 Frozen-MDA is not modified. 1 Incremented-MDA is incremented by the number of transferred bytes during write access.
15-12	Reserved
11 y	Conditional Yielding selector. When selected, theyield/yieldge instructions will not switch channels if the Burst DMA is in Write Mode, and it has less than four bytes in its FIFO. This is aimed at reducing the number of inefficient FIFO flushes due to context switches. 0 Always yields 1 Yields conditionally (when there are less than four bytes in the FIFO in write mode)
10 d	Access Direction or DMA Mode. DMA is in write mode when data was written into MD by stf MD instructions, or if a previous DMA cycle on the external bus was a write access. Writing MDA or MSA changes the DMA mode to the respective value. DMA is in read mode when a previous DMA cycle was a read access, and DMA stays in read mode when data is read by SDMA with an ldf MD instruction. Reading MDA or MSA does not change the DMA mode. 0 Read Mode 1 Write Mode
9-8 e	Error. Indicates if the previous access was acknowledged with a bus error. 00 No error was received. 01 *reserved* 10 Error mode 11 error read burst
7-6	Reserved
5-0 n	Number of bytes in the MD FIFO.

7.2.4.12.1.5 Burst DMA Write (stf)

When received from a stf instruction, the function code bits are interpreted as follows, depending on the addressed register:

Table 7-20. STF Code Bits

Register	7	6	5	4	3	2	1	0
MSA	s		p	freeze	r			spriv
MDA								dpriv
MD			f	cpy			sz	
MS								

Table 7-21. STF Code Bit Field Descriptions

Field	Description
7-6 s	Functional Unit selector 00 for Burst DMA
5 p (MSA)	Prefetch Flag 0 No prefetch 1 Prefetch required from new MSA
5 f (MD)	Forced Flush Flag 0 Automatic flush 1 FIFO contents are flushed (including the new written data).
4 freeze (MSA/MDA)	Address Freeze Mode 0 Address is normally incremented. 1 Address is frozen.
4 cpy (MD)	Copy Mode selection 0 Write Mode 1 Copy Mode
3-2 r	Register selection 00 MSA 01 MDA 10 MD 11 MS
1-0 sz (MD/MS)	Transfer Size 00 size 0 (no data stored in the FIFO) 01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
0 spriv (MSA)	The spriv value is ignored for this device. 0 = valid value

Table continues on the next page...

Table 7-21. STF Code Bit Field Descriptions (continued)

Field	Description
	1 = Reserved
0 dpriv (MDA)	The dpriv value is ignored for this device. 0 = valid value 1 = Reserved

The possible write instructions are listed in the table below (unused bits should always be cleared).

Table 7-22. Burst DMA STF Instruction List

Binary	Assembly	Comments
00_0_0_00_00	stf r,MSA	Writes content of the SDMA general register (r) to the source address register. MSA is in incremented mode.
00_0_1_00_00	stf r,MSA|FR	Writes content of the SDMA general register (r) to the source address register. MSA is in frozen mode.
00_1_0_00_00	stf r,MSA|PF	Writes content of the SDMA general register (r) to the source address register, and starts a read burst access. MSA is in incremented mode.
00_1_1_00_00	stf r,MSA|PF|FR	Writes content of the SDMA general register (r) to the source address register, and starts a read burst access.
00_0_0_01_00	stf r,MDA	Writes content of the SDMA general register (r) to the destination address register. MDA is in incremented mode.
00_0_1_01_00	stf r,MDA|FR	Writes content of the SDMA general register (r) to the destination address register. MDA is in frozen mode.
00_1_0_10_00	stf r,MD|SZ0|FL	No data transfers between the SDMA and MD, but all valid written data of the MD is flushed to the memory. An acknowledge or error is sent back to the SDMA core on transfer completion.
00_0_0_10_01	stf r,MD|SZ8	8-bit (byte) transfer to write buffer MD
00_1_0_10_01	stf r,MD|SZ8|FL	8-bit (byte) transfer to write buffer MD and flush after transfer. All valid written data of the MD is flushed to memory.
00_0_0_10_10	stf r,MD|SZ16	16-bit (half-word) transfer to write buffer MD
00_1_0_10_10	stf r,MD|SZ16|FL	16-bit (half-word) transfer to write buffer MD and flush after transfer. All valid written data of the MD is flushed to memory.
00_0_0_10_11	stf r,MD|SZ32	32-bit (word) transfer to write buffer MD
00_1_0_10_11	stf r,MD|SZ32|FL	32-bit (word) transfer to write buffer MD and flush after transfer. All valid written data of MD is flushed to memory.
00_0_1_10_00	stf r,MD|CPY	No data transfer between SDMA and MD but starts a copy transfer whose length is given by the 4 LSB of *r* register. (Maximum burst length is eight words.)
00_0_0_11_11	stf r,MS	32-bit (word) transfer to status register MS
00_0_0_11_00	stf r,MS|SZ0	Clears the error flag (if set). Other MS bits are unchanged; this instruction is also known as clref MS.

NOTE

When a flush bit is set, the SDMA flushes the FIFO including the newly written data. An acknowledge is sent to the core before the flush completes (except if size 0 is used). The goal of this flush bit is to force a flush, but it is recommended to use it only when needed (for example, when finishing a row of pixels during 2D data transfers). Indeed, if this bit is omitted and if there are more than 32 bytes in the FIFO, a burst write access is automatically triggered.

Since all the stf r,MD instructions (including the copy mode) acknowledge the SDMA core before the store is effective (except if size 0 is used), it is recommended to perform an ldf from MS before terminating a channel in order to check the final error status. (The ldf from MS will stall the core until all the data was flushed out and the transfer status is known.)

After every stf MD instruction, the MDA is incremented by the number of bytes that are written in MD, except when it is programmed in frozen mode.

7.2.4.12.1.6 Burst DMA Read (ldf)

When received from an ldf instruction, the function code bits are interpreted as follows, depending on the addressed register:

Table 7-23. LDF Code Bits

Register	7	6	5	4	3	2	1	0
MSA	s				r			
MDA								
MD			p				sz	
MS								

Table 7-24. LDF Code Bit Field Descriptions

Field	Description
7-6 s	Functional Unit selector 00 for Burst DMA
5 p (MD)	Prefetch Flag 0 no prefetch 1 automatic prefetch

Table continues on the next page...

Table 7-24. LDF Code Bit Field Descriptions (continued)

Field	Description
3-2 r	Register selection 00 MSA 01 MDA 10 MD 11 MS
1-0 sz (MD)	Transfer Size 00 reserved 01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)

The table below lists the possible write instructions (unused bits should always be cleared).

Table 7-25. Burst DMA LDF Instruction List

Binary	Assembly	Comments
00_0_0_00_00	ldf r,MSA	Copies the source address register value into an SDMA general register. It gives the memory address of the next data that will be read with an ldf MD instruction.
00_0_0_01_00	ldf r,MDA	Copies the destination address register value into an SDMA general register. It gives the memory address where the next incoming data will be flushed.
00_0_0_10_01	ldf r,MD|SZ8	8-bit (byte) read
00_1_0_10_01	ldf r,MD|SZ8|PF	8-bit (byte) read. If after this reading and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_10_10	ldf r,MD|SZ16	16-bit (half-word) read
00_1_0_10_10	ldf r,MD|SZ16|PF	16-bit (half-word) read. If after this reading, and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_10_11	ldf r,MD|SZ32	32-bit (word) read
00_1_0_10_11	ldf r,MD|SZ32|PF	32-bit (word) read. If after this reading and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_11_00	ldf r,MS	Copy the status register value into an SDMA general register.

NOTE

Read data is 0-extended before writing in the SDMA general registers. When reading the MD register, the DMA takes data from the FIFO if it is available. If part or whole data is not in the FIFO, an external burst read access is performed to provide the missing data. The SDMA is stalled as long as the required read data is not complete.

After every reading, MSA is incremented by the number of read bytes from MD FIFO, except when MSA is programmed in frozen mode.

7.2.4.12.1.7 Prefetch/Flush and Auto-Flush Management-Burst DMA Unit

The prefetch and auto-flush management enables the SDMA RISC machine to go on while a DMA access is performed.

When the RISC core requires a prefetch (p = 1) to the Burst DMA, it will receive an immediate transfer acknowledge before the DMA has finished the external access. This enables the RISC core to do other things like accessing another DMA machine.

The basic principle in prefetch mode is for the DMA to anticipate data reads from the SDMA RISC engine by fetching external bursts of data as soon as there is enough space in the DMA FIFO to store it. If ever the RISC engine required data that is not available in the FIFO, the read acknowledge is delayed until the data is available, but it does not have to wait until the burst completes.

The auto-flush basic principle is similar: An automatic flush is triggered every time there are eight words to be written in the FIFO. If the FIFO is full and the RISC engine requires another write, it is stalled until the burst has started and enough space was freed in the FIFO to store that new data. This means the SDMA RISC engine does not have to wait for the completion of a burst to receive its acknowledge and continue its processing.

In particular, an auto-flush is executed when DMA is in write mode and if the following is true:

- If the FIFO is empty and the first write is to a word-aligned address of any size (ex: the 2 LSB of MDA[1:0]= 0x0), the auto-flush is triggered immediately after the write of the 32'nd byte.
- If the FIFO is empty, and if MDA is an odd byte address (1, 3, 5, 7,...) and an stf MD|SZ8 is executed, the byte is flushed to memory. Once MDA increments to a word aligned address, the auto-flush will be triggered every 32 bytes.
- If the FIFO is empty, and if MDA is a half-word address (2, 6, 0xA,...) and an stf MD|SZ16 is executed, the two bytes of the incoming data are flushed to memory. Once MDA increments to a word aligned address, the auto-flush will be triggered every 32 bytes.
- If the FIFO is empty, and if MDA is not a word-aligned address (ex 1, 2, 3, 5, 6, 7, 9,...), and an stf MD|SZ32 is executed, the first 1 to 3 bytes will be flushed up to the next word aligned address. Afterwards, an auto-flush will be triggered each time the FIFO receives 32-bytes.

- Therefore, if an stf MD|SZ32 is executed with MDA equal to 0x1 and with an empty MD FIFO, the bytes located at addresses 1, 2, and 3 are flushed, and the byte located at address 4 remains in MD FIFO. This solves the misalignment issue. Additionally, the next write instructions (stf) complete the FIFO until it contains eight words; then a burst write is executed by the DMA to empty the FIFO. Protocol on the external bus does not support bursts of different data types (byte, half-word, or word).

  For example, consider the case where data is written using a byte access, stf MD| SZ8. The value of MDA during the very first byte write determines when the auto-flush will occur as follows:

- If MDA=0x0, the flush occurs following the write of byte 32
- If MDA=0x1, the flush occurs following the write of byte 1, byte 3 and byte 35.
- If MDA=0x2, the flush occurs following the write of byte 2 and byte 34.
- If MDA=0x3, the flush occurs following the write of byte 1 and byte 33.
- If MDA=0x4, the flush occurs following the write of byte 32

The flush command forces the DMA to flush all MD valid bytes to the EXTMC controller. An acknowledge is sent immediately to the SDMA, and any potential error is reported on a future access. It is thus essential to conclude a transfer with a last read from MS, which will stall the core until all data was flushed out and returned to the transfer status (acknowledge or error).

NOTE

During this kind of auto-flush (which occurs only at the beginning of a misaligned write transfer) no acknowledge is sent back to the SDMA, which is stalled until a flush is completed.

7.2.4.12.1.8 Data Alignment and Endianness-Burst DMA Unit

7.2.4.12.1.8.1 Burst DMA in Read Mode

For every read access to MD, the data returned to the SDMA core and the new FIFO state depends on the MSA status and the access size.

The FIFO is considered as a stack of 36 bytes: Data is fetched externally on a 32-bit bus, but the valid bytes only are stored in the FIFO and left-aligned (for a transfer of consecutive words, it is only the first word that may be truncated). The following table shows the FIFO byte alignment strategy and the corresponding MSA, the returned data, and the new FIFO state for any access size of an internal read from MD.

Table 7-26. FIFO Read Configuration

Before read		Internal read access size	Read data	After read	
MSA[1:0]	FIFO state			MSA[1:0]	FIFO state
00	x0 x1 x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 and so on...	sz8	00 00 00 x0	01	x1 x2 x3 y0 y1 y2 y3 z0
		sz16	00 00 x0 x1	10	x2 x3 y0 y1 y2 y3 z0 z1
		sz32	x0 x1 x2 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
01	x1 x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 and so on...	sz8	00 00 00 x1	10	x2 x3 y0 y1 y2 y3 z0 z1
		sz16	00 00 x1 x2	11	x3 y0 y1 y2 y3 z0 z1 z2
		sz32	x1 x2 x3 y0	01	y1 y2 y3 z0 z1 z2 z3 t0
10	x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 t1 and so on...	sz8	00 00 00 x2	11	x3 y0 y1 y2 y3 z0 z1 z2
		sz16	00 00 x2 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
		sz32	x2 x3 y0 y1	10	y2 y3 z0 z1 z2 z3 t0 t1
11	x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 t1 t2 and so on...	sz8	00 00 00 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
		sz16	00 00 x3 y0	01	y1 y2 y3 z0 z1 z2 z3 t0
		sz32	x3 y0 y1 y2	11	y3 z0 z1 z2 z3 t0 t1 t2

7.2.4.12.1.8.2 Burst DMA in Write Mode

For every write access to the MD, the new FIFO state depends on the MDA status and the access size.

The FIFO is considered as a stack of 36 bytes: Data is stored in the FIFO according to the internal access size and the former MDA value. The following table shows the FIFO byte alignment strategy corresponding to MDA, as well as the new FIFO state for any access size of an internal write to MD.

Table 7-27. FIFO Write Configuration

Before write		Internal write access size	Written data	After write	
MDA[1:0]	FIFO state			MDA[1:0]	FIFO state
00	tt uu vv ww ?? ?? ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	01	tt uu vv ww x0 ?? ?? ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	10	tt uu vv ww x0 x1 ?? ?? ?? ?? ?? ??
		sz32	x0 x1 x2 x3	00	tt uu vv ww x0 x1 x2 x3 ?? ?? ?? ??
01	tt uu vv ww xx ?? ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	10	tt uu vv ww xx x0 ?? ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	11	tt uu vv ww xx x0 x1 ?? ?? ?? ?? ??
		sz32	x0 x1 x2 x3	01	tt uu vv ww xx x0 x1 x2 x3 ?? ?? ??
10	tt uu vv ww xx yy ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	11	tt uu vv ww xx yy x0 ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	00	tt uu vv ww xx yy x0 x1 ?? ?? ?? ??
		sz32	x0 x1 x2 x3	10	tt uu vv ww xx yy x0 x1 x2 x3 ?? ??
11	tt uu vv ww xx yy zz ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	00	tt uu vv ww xx yy zz x0 ?? ?? ?? ??
		sz16	?? ?? x0 x1	01	tt uu vv ww xx yy zz x0

Table continues on the next page...

Table 7-27. FIFO Write Configuration (continued)

Before write		Internal write access size	Written data	After write	
MDA[1:0]	FIFO state			MDA[1:0]	FIFO state
					x1 ?? ?? ??
		sz32	x0 x1 x2 x3	11	tt uu vv ww xx yy zz x0 x1 x2 x3 ??

NOTE

If the FIFO mode changes from a write to a read mode, all remaining written bytes in MD are lost but no error is returned. Typically, this happens if an ldf MD is executed after stf MD instructions. Before a mode change, it is recommended to force the flush of a potential remaining byte by a stfMD|SZ0|FL instruction. In the same way, if a FIFO mode changes from a read to a write mode, all prefetched data present in the FIFO is lost and no error is returned.

7.2.4.12.1.8.3 Endianness-Burst DMA Unit

Big and Little Endian are supported by the Burst DMA, but data is always stored in MD in Big Endian.

Byte manipulation is performed when data is exchanged with an Burst controller (for example, during read or write burst accesses).

7.2.4.12.1.9 Burst DMA Unit Copy Mode

A mechanism is available to perform fast ARM-to-ARM transfers.

Data does not flow through the SDMA core: It is kept in the DMA FIFO. This mechanism is selected when writing MD with a special option in the instruction code (copy flag).

It is possible to transfer up to eight words in one SDMA instruction (this does not mean in one cycle). In this mode, every time an stf MD|CPY is executed, a read burst is executed and directly followed by a write burst transfer. Burst transfers are limited to eight words. The size of the transfer (in words)-given by the SDMA general register (4 LSB)-is also limited to eight. The following SDMA code shows how 100 bytes could be copied from the MSA address to the MDA address. This is sample code only.

Burst DMA copy mode example

```
          ldi r0,@src
          stf r0,MSA                          // Source address setup
          ldi r1,@dst
          stf r1,MSA                          // Destination address setup
          ldi r0,0x64                         // data transfer counter
          ldi r1,0x8
MAIN_XFER:
          cmphs r0,r1                         // Is r0 >= 0x8
          bf LAST_XFER                      // If not, jump to last transfer label
          stf r1,MD|CPY                        // Copy 8 words from MSA to MDA address.
          subi r0,0x8                         // Decrement counter
          jmp MAIN_XFER                     // return to main transfer loop
LAST_XFER:
          stf r0,MD|CPY
```

The main transfer loop is executed 12 times; then r0 equals 4 and the last transfer loop is run.

In this mode, an acknowledge is transmitted to the core as soon as the read burst can start; thus, a first copy instruction returns an immediate acknowledge and subsequent copy instructions will be acknowledged as soon as the previous copy has finished.

7.2.4.12.1.10 Burst DMA Unit Error Management

Another point to consider is the management of errors.

Because the DMA immediately sends an acknowledge to the RISC core (except for the stf MS|SZ0|FLS instruction), it assumes no error will occur. If an error occurs, it is flagged (transfer error acknowledge) for the following DMA access.

This should not be a problem if the DMA is used properly. The MD accesses are meant to stall the SDMA as little as possible to optimize throughput and hide calculation time. Therefore, final access to MS should be performed before closing a channel. This access waits until any pending operation is finished in the burst DMA and gather any remaining error.

In copy mode, an error could be immediately returned to the SDMA on execution of the ldf copy or stf copy instruction. It happens when MSA or MDA are not word addresses (for example, 0[4]). This is because copy mode must only be used for transferring a large packet of aligned data.

When an error is received during a *read* transfer to the external bus, which may occur during the burst accesses, the MD FIFO contains the valid beats of the burst, and the error flag of MS is set to 2'b11 (error read burst). It is possible to read MS ("n" field) to know how much valid data remains in MD and when MD is empty (after ldf instructions). The next read MD instruction sets the MS error flag to 2'b10 (error mode), and an error is sent back to the SDMA core. In error mode, it is possible to read MSA, which gives the address of the error data. Any attempt to read or write MD, or to modify MDA or MSA in error mode, gives rise to an error; therefore, an error flag must be reset by clearing MS at the end of the SDMA code section responsible for error management.

In "error read burst" mode, writing MDA, MSA, or MD, or starting a copy transfer by a stf MD|COPY instruction will cancel the error mode. The following table shows when an immediate error is sent back according to the executed instruction.

Table 7-28. Possibilities in ERROR READ BURST Mode

DMA Instruction	Immediate Error	Comments
stf rn, MD stf rn, MSA (|U |PF) stf rn, MDA stf rn,MD|COPY	NO	Error mode is reset. MSA, MDA, or MD are updated and a DMA cycle may start. For the stf MD|COPY, a copy loop is executed.
stf rn, MS	NO	MS is updated.
ldf rn, MS ldf rn, MSA ldf rn, MDA	NO	MS, MSA, and MDA could be read in ERROR READ mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, MD	YES/NO	Immediate error if there is no more data available for read in the FIFO.

When an error is received during a *write* transfer, the error is reported to the next DMA access. In this case, an error is sent to the SDMA core and the DMA goes to its error mode. Reading MS gives the number of bytes that remain in MD; reading MDA gives the address of the error data. Any attempt to read or write MD, or to modify MDA or MSA in error mode, give rise to an error; therefore, an error flag must be reset by clearing MS at the end of the SDMA code section responsible for error management.

Table 7-29. Possibilities in ERROR Mode

DMA Instruction	Immediate Error	Comments
stf rn, MD stf rn, MSA stf rn, MDA	Yes	Any attempt to modify MD, MSA, MDA will raise an immediate error and burst DMA remains in error mode. When address registers are write-accessed, an error is returned.
stf rn, MS	No	This is the only way to exit error mode. MS[9:8] must be reset by an stf MS|SZ0 instruction.
ldf rn, MS ldf rn, MSA ldf rn, MDA	No	MS, MSA, and MDA could be read in error mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, MD	Yes	Whatever the DMA direction (read or write), an ldf rn triggers an immediate error.

7.2.4.12.1.11 Conditional Yielding-Burst DMA Unit

The standard SDMA transfer is based upon a hardware loop that has the following structure:

Hardware Loop

```
loop
load Rn,source            // can be ldf or ld
<computation>            // can be done through functional units
store Rn,dest              // can be st or stf
done 0               // yield
```

This structure needs to be kept independent of the functional units' particularities regarding the context switch. However, there can be variations in the context switch's efficiency, which can depend on the number of data received up to that point, and on the data itself.

The DMA, with its 8-word burst capability, has a preferable context switch period when its address register is 8-word aligned: It is the only moment that occurs once every eight loops when the succession of bursts is not broken by the context switch. When this is not the case, a context switch requires the storing (or loading) of less than eight words, which requires separate accesses and is far less efficient. The rest of the 8-word packet is stored (or loaded) after the context restore, and this is done as separate accesses.

The proposed solution is a conditional yielding, which occurs only when the DMA is in an optimum state. It does not require any modification to the scripts. The condition is decided at the DMA level.

The DMA can be programmed in two modes-conditional or always-true-for every channel, which provides complete flexibility. By default, the DMA is not in conditional mode.

The DMA condition is computed from the FIFO fill level and the various modes, as follows:

- When copy mode is selected, regardless of the transfer direction ('read' or 'write'), the condition is always true.
- In read mode, the condition is always true.
- In write mode, the condition is true when there are four bytes or less in the FIFO; it is false when there are more than four bytes. The 4-byte limit comes from the possibility of saving those bytes as MD with absolutely no impact on the bus accesses.

The aim at conditional yielding is to avoid splitting bus accesses (especially bursts).

7.2.4.12.2 Peripheral DMA Unit Programming

The peripheral DMA unit is connected to the Multi-Layer DMA Crossbar Switch of the ARM platform.

Its goal is to perform data transfers between any blocks connected to the DMA bus of this platform. These blocks are either peripherals or memories. The peripheral DMA could be seen as the ARM platform DMA controller.

The DMA performs data transfers in three modes:

- Read mode, where data is read from peripherals or from memory connected to the ARM platform and copied in a SDMA general register.
- Write mode, where data of a general register has to be written in a peripheral or a memory.
- Copy mode, where data is read from a peripheral (or memory) at a source address (PSA) and automatically written to a peripheral (or memory) at a destination address (PDA).

In copy mode, no SDMA general register is involved as transferred data only goes through the data register of the DMA.

The peripheral DMA has three addressing modes: frozen, incremented, and decremented, as follows:

- Frozen mode-When source or destination addresses are frozen, their value is not modified after a transfer. This mode is typically used for addressing peripheral FIFOs located at a fixed address.
- Incremented mode-When source or destination addresses are in incremented mode, after every transfer they are incremented by the number of bytes transferred.
- Decremented mode-In decremented mode, addresses are decremented by the number of bytes transferred.

The peripheral DMA registers are as follows:

- Two, 32-bit address registers (PSA and PDA) that respectively contain the source address for a read access and the destination address for a write access
- A 32-bit status register (PS) that contains information on the peripheral DMA configuration, such as the number of valid bytes in the data register, the error flag, the source and destination address mode, and so on.
- A 32-bit data register (PD) that stores data involved in a data transfer

7.2.4.12.2.1 Peripheral Source Address Register (PSA)

The source address register contains a pointer to a source peripheral or a memory associated with the next read data transfer. It has byte granularity.

It is based on the following:

- A 32-bit register (PSA) to store the address value
- A 2-bit register (stype) to store the source address mode (frozen, incremented, or decremented)
- A 2-bit register (ssize) to store the source target data path size (byte, half-word, or word)

Reading the register with the ldf instruction has no side effects and gives the address value of the next data that will be read by the SDMA during an ldf MD instruction. Writing the source address register may have side effects. If there is valid write data in the data register and the source address is changed, the write data is discarded. If the prefetch bit is set, a DMA read cycle is issued with the new address.

When PSA is to be written, you must specify the source target address mode, providing its size (byte, half-word, or word). This enables omission of the size field in all ldf MD instructions. When DMA performs a read cycle, its size is given by the value of the PSA source size register (ssize). If source is a memory in incremented mode, first programmed in word mode (stf PSA|SZ32|I), and if an SDMA script needs to read bytes from this memory, the size of the source target must be updated before executing new accesses. The source address mode and its size are given by labels added to the stf PSA instruction as described in the write section. The ssize and stype registers are part of the DMA status register (PS).

Writing to PSA may issue an immediate error if the source size is not compatible with the value to be written into the PSA register. For instance, writing a 2 in PSA and specifying that it is memory-accessed in word mode creates an immediate error.

7.2.4.12.2.2 Peripheral Destination Address Register (PDA)

The destination address register contains a pointer to a source peripheral or a memory associated with the next write data transfer. It has byte granularity.

It is based on the following:

- A 32-bit register (PDA) to store the address value
- A 2-bit register (dtype) to store the destination address mode (frozen, incremented, or decremented)
- A 2-bit register (dsize) to store the destination target data path size (byte, half-word, or word)

Reading the register with the ldf instruction has no side effects, and gives the address value of the next data that will be written by SDMA during an stfMD instruction. Writing the destination register has no side effect. Similar to the PSA register, the destination address mode and source are specified in the stf PDA instruction and may also generate an error in case of incorrect programming.

7.2.4.12.2.3 Peripheral Data Register (PD)

The data register of the peripheral DMA is a 32-bit register. When the destination address is correctly set up, any writing to PD will automatically flush the new input data.

The number of SDMA bytes that will be transferred is given by the PDA size register. Unlike other SDMA DMAs, PD is not a FIFO: It is not used to accumulate bytes that from the SDMA and must be packed before being sent to external memories. In read mode, and if the source address is correctly set up, an ldf instruction will empty PD. If a prefetch is required along with the instruction, the DMA will initiate a new read transfer.

Reading PD in prefetch mode only stalls the SDMA when the prefetched data is not yet available. Writing PD only stalls the SDMA if the previous write operation was not completed. As soon as the previous operation is over, the acknowledge is sent back to the SDMA RISC engine.

An error flag-part of PS-is set when an external access fails. The error is thus reported to the next SDMA instruction that involves the peripheral DMA.

7.2.4.12.2.4 Peripheral State Register (PS)

The state register contains the DMA state-machine value. It can be accessed in case of an error received during a transfer.

Although all PS fields can be written by an stf instruction, it is recommended to access only the error bit (to reset it). Modifying other PS fields will provide an un-guaranteed DMA behavior.

The initialization value of PS is 0, and it consists of the following structure:

Table 7-30. PS Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	ssize		stype		dsize		dtype	
W																
R	0	0	0	0	0	d	e		0	0	0	0	0	n		
W																

Table 7-31. PS Field Descriptions

Field	Description
31-24	Reserved
23-22 ssize	Source Target Size. Determines the size of the read transfers on the external bus. It should match the accessed device characteristics. 00 *reserved* 01 Byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
21-20 stype	Source address Mode. Determines whether PSA is incremented, decremented, or kept unmodified after every read from the external bus. 00 Frozen Mode 01 Incremented Mode 10 Decremented Mode 11 *reserved*
19-18 dsize	Destination Target Size. Determines the size of the write transfers on the external bus. It should match the accessed device characteristics. 00 *reserved* 01 Byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
17-16 dtype	Destination address Mode. Determines whether PDA is incremented, decremented, or kept unmodified after every write on the external bus. 00 Frozen Mode 01 Incremented Mode 10 Decremented Mode 11 *reserved*
15-11	Reserved
10 d	Direction Flag or DMA Mode. DMA is in write mode when data was written into PD by stf PD instructions, or if a previous DMA cycle on the external bus was a write access. Writing PDA or PSA does not change the DMA mode. DMA is in read mode when a previous DMA cycle was a read access, and DMA stays in read mode when data is read by the SDMA with an ldf PD instruction. Reading PDA or PSA does not change the DMA mode. 0 Read Mode 1 Write Mode
9-8 e	Error. Indicates if the previous access was acknowledged with a bus error. 00 No error was received. 01 *reserved* 10 Error mode 11 Error read
7-3	Reserved

Table continues on the next page...

Table 7-31. PS Field Descriptions (continued)

Field	Description
2-0 n	number of bytes in PD

NOTE

dtype, dsize, stype, and ssize are updated when PSA and PDA are written.

7.2.4.12.2.5 Peripheral DMA Write (stf)-Write Mode

When written by an stf instruction, the function code bits are interpreted as follows:

Table 7-32. STF Code Bits

Register	7	6	5	4	3	2	1	0
PSA	s		p	ar	am		sz	
PDA								
PD			pdsel					
PS			pssel					

Table 7-33. STF Code Bits Field Descriptions

Field	Description
7-6 s	Functional Unit selector 11 for Peripheral DMA
5 p (PSA)	Prefetch Flag 0 no prefetch 1 automatic prefetch
4 ar (PSA/PDA)	Address Register Selector 0 PSA 1 PDA
3-2 am (PSA/PDA)	Address Mode. Determines how PSA or PDA is modified after every read or write access to the PD. 00 Frozen-Address registers are not modified after the transfer. 01 Incremented-Address registers are incremented by the number of transferred bytes. 10 Decremented-Address registers are decremented by the number of transferred bytes. 11 Updated-PSA and PDA are not modified. Either address mode is not modified, but the width of the data path is updated by the sz field.
1-0 sz	Transfer Size 00 *reserved*

Table continues on the next page...

Table 7-33. STF Code Bits Field Descriptions (continued)

Field	Description
	01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
5-0 pdsel	PD access selector 001000 is the only valid option
5-0 pssel	PS access selector 111111 writes to PS 001100 only clears the error flag in PS

Due to the large number of possible stf instructions, the following table provides only a short list of all the possible write instructions:

Table 7-34. Peripheral DMA STF Instruction List

Binary	Assembly	Comments
11_00_00_01 11_00_00_10 11_00_00_11	stf Rn, PSA|SZ8 |F stf Rn, PSA|SZ16|F stf Rn, PSA|SZ32|F	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is frozen.
11_10_00_01 11_10_00_10 11_10_00_11	stf Rn,PSA|SZ8 |F|PF stf Rn,PSA |SZ16|F|PF stf Rn,PSA |SZ32|F|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • 1, 2, or 4 bytes are *fetched* from the peripheral source. • Source address is frozen.
11_00_01_01 11_00_01_10 11_00_01_11	stf Rn, PSA|SZ8 |I stf Rn, PSA|SZ16|I stf Rn, PSA| SZ32|I	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA + 1,2 or 4 after read PD.
11_10_01_01 11_10_01_10 11_10_01_11	stf Rn, PSA|SZ8 |I|PF stf Rn, PSA|SZ16|I|PF stf Rn, PSA|SZ32|I|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA + 1, 2, or 4 after read PD. • 1, 2, or 4 bytes are *fetched* from the peripheral source.
11_00_10_01 11_00_10_10 11_00_10_11	stf Rn, PSA|SZ8 |D stf Rn, PSA|SZ16|D stf Rn, PSA|SZ32|D	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA-1,2, or 4 after read PD.
11_10_10_01 11_10_10_10 11_10_10_11	stf Rn, PSA|SZ8 |D|PF stf Rn, PSA|SZ16|D|PF stf Rn, PSA|SZ32|D|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA-1,2, or 4 after read PD. • 1, 2, or 4 bytes are *fetched* from the peripheral source.

Table continues on the next page...

Table 7-34. Peripheral DMA STF Instruction List (continued)

Binary	Assembly	Comments
11_00_11_01 11_00_11_10 11_00_11_11	stf Rn, PSA|SZ8 |U stf Rn, PSA|SZ16 |U stf Rn, PSA|SZ32 |U	• *Update* source pointer to memory, which becomes a pointer to a memory accessed in byte, half-word, or word. • PSA value is not modified by Rn. • Bytes present in PD are lost.
11_10_11_01 11_10_11_10 11_10_11_11	stf Rn, PSA|SZ8 |PF|U stf Rn, PSA|SZ16 |PF|U stf Rn, PSA|SZ32 |PF|U	• *Update* source pointer, which becomes a pointer to a target accessed in byte, half-word, or word. • PSA value is not modified by Rn. • Bytes present in PD are lost. • 1, 2, or 4 bytes are *fetched* from the memory source.
11_01_00_01 11_01_00_10 11_01_00_11	stf Rn, PDA|SZ8 |F stf Rn, PDA|SZ16|F stf Rn, PDA|SZ32|F	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is frozen.
11_01_01_01 11_01_01_10 11_01_01_11	stf Rn, PDA|SZ8 |I stf Rn, PDA|SZ16|I stf Rn, PDA|SZ32|I	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is in incremented mode: PDA = PDA + 1, 2, or 4 after write PD.
11_01_10_01 11_01_10_10 11_01_10_11	stf Rn, PDA|SZ8 |D stf Rn, PDA|SZ16|D stf Rn, PDA|SZ32|D	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is in incremented mode: PDA = PDA-1, 2, or 4 after write PD.
11_01_11_01 11_01_11_10 11_01_11_11	stf Rn, PDA|SZ8 |U stf Rn, PDA|SZ16 |U stf Rn, PDA|SZ32 |U	• *Update* destination pointer to memory, which becomes a pointer to a memory accessed in byte, half-word, or word. • PDA value is not modified by Rn • bytes present in PD are lost
11_00_10_00	stf Rn, PD	• Write "dsize" bytes of Rn in PD and automatically flush to destination target
11_11_11_11	stf Rn, PS	• Write status register
11_00_11_00	stf Rn,clrefPS	• Clear error flag if set

NOTE

When writing PD, size information is not important: It is embedded in the dsize field of PDA register. If dsize is 1, 2, or 4, then one, two, or four bytes from Rn is written to the PD register, and automatically flushed out to the destination target.

7.2.4.12.2.6 Peripheral DMA Read (ldf)-Read Mode

When received from an ldf instruction, the function code bits are interpreted as follows.

Table 7-35. LDF Code Bits

Register	7	6	5	4	3	2	1	0
PSA	s			ar	a			
PDA								
PD			p	cpy				
PS			pssel					

Table 7-36. LDF Code Bits Descriptions

Field	Description
7-6 s	Functional Unit selector 11 for Peripheral DMA
5 p (PD)	Prefetch Flag 0 no prefetch 1 automatic prefetch
4 ar (PSA/PDA)	Address Register Selector 0 PSA 1 PDA
4 cpy (PD)	Copy Mode 0 standard access 1 copy mode access
3 a	Register Set selection 0 PSA or PDA 1 PD or PS
5-0 pssel	PS access selector 111111 is the only valid option to read PS

Table 7-37. Peripheral DMA LDF Instruction List

Binary	Assembly	Comments
11_0_0_0_000	ldf Rn, PSA	Reads 32-bit of PSA value
11_0_1_0_000	ldf Rn, PDA	Reads 32-bit of PDA value
11_0_0_1_000	ldf Rn, PD	Reads programmed source size bytes of PD (0-extended)
11_1_0_1_000	ldf Rn, PD|PF	Reads programmed source size bytes of PD (0-extended), and starts a prefetch at PSA address.
11_0_1_1_000	ldf Rn, PD|COPY	Starts a copy transfer from the source target at the PSA address to the destination target at the PDA address. No data transmits through Rn, but Rn contents are lost (Rn is loaded with PD temporary contents that are *not* the copied data).
11_111111	ldf Rn, PS	Reads 32-bit of PS value

NOTE

When reading PD, size information is not important: It is embedded in the ssize field of the PSA register. If ssize is 1, 2, or 4, the one, two, or four bytes is transferred from PD to Rn. Read data is 0-extended.

7.2.4.12.2.7 Peripheral DMA Unit Copy Mode

Like burst DMA, the peripheral DMA unit has a copy mode that is used when data transfers do not involve SDMA general registers.

Data is read from the source target at a PSA address, stored in PD, and then automatically flushed to the destination target at the PDA address. Copy mode is only available for transfers that involve two targets of the same data path width.

Since copy mode is invoked with an ldf instruction, the *loaded* general purpose register loses its previous contents. (However, the new contents are unpredictable as they depend on temporary values that are seen on the external DMA bus.)

7.2.4.12.2.8 Error Management

Peripheral DMA generates two kinds of errors: the immediate error that sanctioned incorrect register programming; and the error triggered by the previous access and stored in the error flag of PS until a DMA instruction is executed.

7.2.4.12.2.8.1 Immediate Errors

The following table lists all incorrect DMA register setups.

Table 7-38. Immediate Errors with Peripheral DMA

Rn[1:0] values	DMA instruction	Comments
0x01 0x11	stf Rn, PSA|SZ16|F stf Rn, PSA|SZ16|I stf Rn, PDA|SZ16|F stf Rn, PDA|SZ16|I	If PSA points to a half-word peripheral or to a half-word address in memory, its value must be 0 modulo 2.
0x01 0x10 0x11	stf Rn, PSA|SZ32|F stf Rn, PSA|SZ32|I stf Rn, PDA|SZ32|F stf Rn, PDA|SZ32|I	If PSA points to a word peripheral or to a word address in memory, its value must be 0 modulo 4.
PSA[1:0]-PDA[1:0]	DMA instruction	Comments

Table continues on the next page...

Table 7-38. Immediate Errors with Peripheral DMA (continued)

Rn[1:0] values	DMA instruction	Comments
0x01 0x10 0x11	stf Rn, PSA|SZ32|U stf Rn, PDA|SZ32|U	When PDA or PSA is updated and becomes a pointer to a word address in memory, its content must be 0 modulo 4.
0x01 0x11	stf Rn, PSA|SZ16|U stf Rn, PDA|SZ16|U	When PDA or PSA is updated and becomes a pointer to a half-word address in memory, its content must be 0 modulo 2.
Read/Write PD instruction	Comments	
stf Rn,PD ldf Rn,PD	If PDA size (dsize) has never been set up before an stf PD instruction (dsize=0) If PSA size (ssize) has never been set up before an ldf PD instruction (ssize=0)	
ldf Rn,PD|CPY	Copy mode is possible only between two targets whose data path width is identical. It is P8↔P8, P16↔P16, or P32↔P32 regardless of the way the address registers are incremented.	

7.2.4.12.2.8.2 Data Transfer Errors

When PSA and PDA are correctly set up, the only error that may arise for an ldf PD or stf PD instruction would be the error of the previous DMA cycle.

Error handling is driven by a single consideration: When an error occurred during a data read on the DMA interface, this error should appear as a transfer error to the core when the core attempts to retrieve the data that was not successfully read from the accessed device (memory or peripheral).

When an error occurred during a write access to the DMA interface, the data is still available in PD and should not be destroyed by subsequent core accesses: The core must be warned about the error issue.

There are three error handling mechanisms for each case: Read Error (First Phase), Write Error and Read Error (Second Phase), and Copy Mode Errors handling.

7.2.4.12.2.8.3 Read Error (First Phase)

If an error occurred during a prefetch command, the peripheral DMA enters its ERROR READ mode (PS[9:8]=11). In this mode, the error is reported on the next ldf PD instruction and writing PSA, PDA, or PD will cancel the error flag.

The block returns no error mode and instructions are normally executed (a DMA cycle may be triggered). Similarly, initiating a copy transfer will reset the error flag and start a copy transfer. The following table details which instructions can be executed in this mode.

Table 7-39. Possibilities in ERROR READ Mode

DMA Instruction	Immediate Error	Comments
stf rn, PD stf rn, PSA (IU IPF) stf rn, PDA ldf rn,PD|COPY	NO	Error mode is reset, PSA or PDA are updated, or a write cycle is started. For the ldf PD|COPY, a copy loop is executed.
stf rn, PS	NO	PS is updated.
ldf rn, PS ldf rn, PSA ldf rn, PDA	NO	PS, PSA, and PDA could be read in ERROR READ mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, PD	YES	Error of the previous read access is reported here and the peripheral DMA enters its ERROR mode.

7.2.4.12.2.8.4 Write Error and Read Error (Second Phase)

The peripheral DMA enters its ERROR mode (PS[9:8]=10) when the previous DMA write cycle failed, or, as explained in Read Error (First Phase), when an ldf PD is executed while the block is in ERROR READ mode. When a DMA cycle failed, address registers (PSA, PDA) are not modified and continue to point to the problematic address. In ERROR mode, stf instructions may raise an immediate error, and ldf instructions will not (as detailed in the table below).

Table 7-40. Possibilities in ERROR Mode

DMA Instruction	Immediate Error	Comments
stf rn, PD stf rn, PSA stf rn, PDA	YES	Any attempt to modify PD, PSA, or PDA will raise an immediate error, and the peripheral DMA stays in ERROR mode. When address registers are write accessed, an error is returned.
stf rn, PS	NO	This is the only way to exit the ERROR mode. PS[3] must be reset by an stf PS instruction.
ldf rn, PS ldf rn, PSA ldf rn, PDA	NO	PS, PSA, and PDA could be read in ERROR mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, PD	YES	Whatever the DMA direction (read or write), an ldf rn, PD instruction will show an immediate error.

7.2.4.12.2.8.5 Copy Mode Errors

Because copy mode is a write access that follows a read access, there are two possible cases of bus error.

When the read access incurs a bus error, the peripheral DMA behaves exactly as described in Read Error (First Phase) and Write Error and Read Error (Second Phase) : It enters its ERROR READ mode, and so on.

When the error occurred during the write access of the copy transfer, the DMA enables the core to retrieve the data that was read because it is assumed the read from the peripheral removed the data from its source device. Therefore, the data to be flushed is still in PD. Any subsequent access to PD triggers an error to the core, which should execute its error handling procedure.

Once the ERROR mode is left (after writing to PS), it is possible for the core to retrieve the data in PD with an ldf instruction or try to flush PD contents once again (for example, when the error was due to a full FIFO and the script waited for the FIFO to be emptied) with another ldf instruction in copy mode. This latter instruction detects that there is valid data in PD, tries to flush it, and thus skips the read phase of the copy instruction. This is a different behavior from the usual stf PD instruction that overwrites PD with the selected General Purpose register contents. The same mechanism can be used any time PD holds data that is not written because of a bus error on the DMA interface; when the data was written via a copy instruction, or via the usual stf PD instruction.

7.2.4.12.2.8.6 Error Check Example

The following code illustrates an example checking for both immediate and data transfer errors on a store to the PD register. The first bdf instruction checks for an immediate error, but if a data transfer error occurred it is reported until the next instruction to access the Peripheral DMA. A second check of the error flags is done after the ldf PS instruction. The value of PS here can be ignored. The act of reading any register in Peripheral DMA while it is in an error mode that returns the error to the core to set either the SF or DF flag. Any error returned on an ldf command sets the SF flag and any error returned on an stf instruction sets the DF flag. This can create a situation as shown in the example where a bus error during a DMA write which would normally be considered as a destination fault is reported as a source fault because the error was reported to the SDMA core during an ldf instruction.

Peripheral DMA Error Check

```
    clrf    0            // Clear SF and DF flags
 stf    R4, PD        // Write data to memory
bdf    error_routine    // Check for immediate error from write to PD.
ldf    r3, PS        // Read PS (PS value in R3 can be ignored)
    bsf    error_routine    // Check for bus error from "stf R4,PD"
           // SF is set because it is a ldf instruction, even though
           // the original error was a destination fault
```

7.2.4.12.2.9 Peripheral DMA Unit Prefetch/Flush Management

There is no flush bit because every time data is stored in PD by a stf PD instruction-assuming PDA is correctly programmed-it is automatically flushed to the destination.

An acknowledge is returned in the cycle of the DMA instruction, and the SDMA is only stalled by an instruction that addresses the peripheral DMA when the previous DMA access is not over.

7.2.4.12.3 OnCE and Real-Time Debug

The On-Chip Emulation block (OnCE) is the debug interface to the SDMA.

It supports the access to all core internal devices (registers, memory, and so on), and provides a set of mechanisms that control the core. The OnCE is accessed by JTAG ports at the chip's board level, or by the host via its peripheral bus.

To reduce the size of the hardware material involved, all tasks supported by the OnCE are performed on the SDMA core. The architecture of the SDMA OnCE is relatively simple and very flexible.

The commands supported by the SDMA OnCE are listed in the following sections.

7.2.4.12.3.1 Memory and Register Access

A set of mechanisms is provided to access SDMA memory and register locations. Both reading and writing are allowed. The access is supported if the processor is in debug mode.

Those registers can also be accessed through the ARM platform Control interface when the OnCE is controlled by the ARM platform, as described in the "Using BP" section.

7.2.4.12.3.2 Hardware Breakpoints

An event detection unit is implemented to support memory breakpoints. The unit watches the data exchanged between the SDMA memory bus and the core.

A debug request is sent to the core when matching conditions occur. The unit supports mixed conditions based on address range, access type, and data value. Event detection unit configuration registers are memory mapped in the SDMA space (see ARM platform Channel 0 Pointer (SDMAARM_MC0PTR)): You can modify them through a regular memory access or the ARM platform control interface.

7.2.4.12.3.3 Watchpoints

One output pin is provided to monitor matching trigger conditions that are defined in the event detection unit.

7.2.4.12.3.4 Software Breakpoints

The SDMA instruction set contains a software breakpoint. Upon executing a software breakpoint instruction, the core suspends normal execution and enters debug mode.

No hardware step execution mode is implemented in the OnCE, but this feature may be implemented at the software level with this instruction.

7.2.4.12.3.5 Core Control

Commands are provided to monitor and control processor activity. You can halt the core, rerun the core from another address location, and get processor status.

Any hardware breakpoint on the instruction bus is not supported, but this feature may be implemented by inserting a software breakpoints program.

7.2.4.13 The OnCE Controller

The OnCE controller receives commands from the ARM platform or from the JTAG controller. Each command is interpreted before being sent to the core.

7.2.4.13.1 OnCE Commands

A small set of commands supports the communication between the OnCE and the external world.

This command set enables you to perform any of the following tasks: control processor activity, save core context, and execute an SDMA instruction from the OnCE. Combined together, these tasks perform more complex commands.

A full OnCE command contains a 4-bit instruction (the OnCE command opcode) and a variable length data field (the OnCE data). During command execution, the OnCE data is transferred in a OnCE internal register before being exchanged with the SDMA. Some data values are also exported. This mechanism creates a link between the processor and the external world. Nine commands are defined: The following table presents their formats.

Table 7-41. OnCE Command Opcode Values

Instruction Opcode	Name	Action	Register	Data Field Size	Mode
0000	rstatus	Reads the OnCE status register	STATUS	16-bit	normal/debug
0001	dmov	Updates general register GReg1	GREG1	32-bit	debug
0010	exec_once	Runs the instruction from the SDMA instruction register	INSTRUCTION	16-bit	debug
0011	run_core	Returns to normal execution	BYPASS	1-bit	debug
0100	exec_core	Returns to normal execution via a jump instruction that specifies the new address	INSTRUCTION	16-bit	debug
0101	debug_rqst	Stops the core after execution of current instruction	BYPASS	1-bit	normal
0110	rbuffer	Reads the real time buffer	RTB	32-bit	normal/debug
0111-1110	reserved	Reserved	BYPASS	1-bit	normal/debug
1111	bypass	Bypasses TARM platform controller	BYPASS	1-bit	normal/debug

Each instruction corresponds to a specific action performed on the OnCE. The nature of the associated data field is clearly identified. The dmov command is followed by a 32-bit data value (which is a data value for the SDMA); the exec_once and the exec_core commands are followed by a 16-bit data value (which is an instruction for the SDMA); the rstatus command is followed by a 16-bit control value (which is the content of the OnCE status register); the rbuffer command is followed by a 32-bit data value. The debug_rqst and the run_core commands are followed by a single bit data field (this is a bypass value). Finally, the bypass instruction enables the SDMA JTAG TAP controller to be daisy-chained with another JTAG TAP controller. This is a JTAG-only feature. The set of commands is simple, but enables you to perform any possible task on the SDMA during a debug process.

7.2.4.13.2 Sending Commands to the OnCE Controller

The JTAG access is the standard access to the OnCE, but sometimes the JTAG is not available to fix some bugs (if the chip is in production for instance), an additional access is then required. Therefore, one ARM platform access to the OnCE is provided.

7.2.4.13.2.1 Using the JTAG Interface

A serial access is performed through the five JTAG pins TCK, TRST, TMS, TDI, and TDO. A Test Access Port controller is provided to decode the TMS control signal.

It produces shift-enable signals (shift_ir and shift_dr), and updates enable signals (update_ir and update_dr). It is fully compliant with the IEEE 1149.1 testability (JTAG) standard.

During the shift_ir state, the command opcode is shifted into the OnCE controller (for example, the signal from the TDI pin is shifted into the command register and the TDO pin receives the signal shifted out). After transferring the four bits of the command, an update_ir signal is asserted and the command is decoded. The target data register is now clearly identified and the corresponding control signal is produced, as follows: bypass enable signal (bp_en), instruction enable signal (inst_en), data enable (data_en), and status enable signal (stat_en).

During the shift_dr state, the TDI signal is shifted into one of the following target registers: bypass register (1 bit), SDMA instruction register (16 bits), SDMA data register (32 bits), or OnCE status register (16 bits). The TDO pin is connected to the output of the selected register to receive the signals shifted out.

The JTAG access is disabled when the ARM platform access is enabled.

7.2.4.13.2.2 Using the ARM platform

The ARM platform access to the OnCE is not the standard access, but it is required if the JTAG is not available.

For example, if the SDMA ROM is out of use on a chip in production, and the ARM platform needs to download new code and restart the SDMA, the OnCE can easily perform this operation. This type of debug operation justifies the use of an ARM platform access to the OnCE.

To drive the OnCE, the ARM platform uses some registers contained in the ARM platform Control block of the SDMA. These registers are accessed through the ARM platform peripheral bus. Most of these registers are connected to another register in the OnCE controller. Thus, accessing one of these registers is equivalent to accessing the associated register in the OnCE controller.

The set of registers in the ARM platform Control block is listed below:

- ONCE_ENB register (1 bit, read/write)-This 1-bit register enables the ARM platform access to the OnCE. When this bit is set, the signals from the JTAG are ignored. When it is cleared, all writing operations to the following registers through the Host Control interface are ignored. This register is reset on a JTAG reset.
- ONCE_CMD register (4 bits, read/write)-This 4-bit register receives the command opcode. It is connected to the command register in the controller. A write access to this register causes the associated command to be executed on the OnCE. For example, after writing "0001" in this register, a dmov command is executed.

NOTE

On the ARM platform side, the rstatus and bypass commands are not supported. This register is reset on a JTAG reset.

- ONCE_DATA register (32 bits, read/write)-This 32-bit register is connected to the SDMA data register. This register is used when executing a dmov or rbuffer command.

NOTE

Before requesting a dmov command, the 32-bit data to transfer must be written in the ONCE_DATA register. At the end of the execution, the register is updated with GReg1 former value. This register is reset on a JTAG reset.

- ONCE_INSTR register (16 bits, read/write)-This 16-bit register is connected to the SDMA instruction register. This register is used when executing an exec_core or an exec_once command.

NOTE

Before requesting an exec_core or an exec_once command, the appropriate instruction must be written in the ONCE_INSTR register. This register is reset on a JTAG reset.

- ONCE_STAT register (16 bits, read only)-A read access to the ONCE_STAT register returns the content of the OnCE status register (OSTAT). This register is read only.
- The bypass register is not useful when the ARM platform controls the OnCE, therefore no register is defined in the ARM platform Control block to access the bypass register.

7.2.4.13.2.3 Conflicts Between the JTAG and the ARM platform Accesses

When ARM platform access to the SDMA OnCE is enabled (that is, when the bit in the ONCE_ENB register is set), the JTAG access is disabled. This guarantees that the block is not accessed at the same time on both sides.

It is possible to check whether the JTAG access to the SDMA OnCE is enabled from the JTAG port. When the JTAG access is disabled, the SDMA TDO always returns 1. The check requires the following steps:

- Execute a dmov command from debug mode (with neither 0xffffffff nor 0x0 as dmov value: 0x5a5a5a5a is good).

- Execute another dmov command (the value here is not important).
- The returned value from the latter dmov command should be the original one if the JTAG access is enabled; if it is 0xffffffff instead of the original input value, this means the JTAG access is disabled.

7.2.4.13.3 Executing a Command from the OnCE

All the commands defined in OnCE Commands can be accessed through the JTAG. The ARM platform can access all these commands except the rstatus command.

On the ARM platform side, the OnCE status is directly accessed by reading the ONCE_STAT register.

7.2.4.13.3.1 Nature of the Commands

Two types of commands may be distinguished. First, there are two commands that do not interact with the core: rstatus and rbuffer. Those commands may be requested at any time: They do not depend on the core status.

NOTE

Each of these commands exports a data value or a status value from the SDMA.

There are also commands that interact with the core: dmov, run_core, exec_core, exec_once, and debug_rqst. These commands are core status dependent, as follows:

- During user mode only the debug_rqst is taken into account.
- During debug mode, all these commands are taken into account except the debug_rqst. For example, an exec_once command requested while not in debug mode has no effect.

7.2.4.13.3.2 Execution Request

The SDMA starts executing a task in debug mode when requested by the OnCE controller. The execution starting time depends on the type of access used to communicate with the OnCE.

If the JTAG is used, the request is send after decoding the update_dr state in the TAP controller. Therefore, always cross this state when sending a command through the JTAG. If the OnCE is driven from the ARM platform side, the request is sent after detecting a write access to the ONCE_CMD register. All the registers involved in this operation must be loaded first.

The following is an example of an exec_core command execution from the ARM platform side: After writing '010' in the ONCE_CMD register, the OnCE controller asks the SDMA to execute the instruction contained in the ONCE_INSTR register. The instruction involved should be available in the ONCE_INSTR register before the beginning of the execution.

7.2.4.13.3.3 Command Execution

The following list shows the commands and details how each command is executed:

- rstatus command execution-The rstatus command exports the content of the OnCE status register (OSR). If the JTAG is used, the status information is captured in the OnCE status register during the capture_dr state, and shifted out after 16 TCK clock cycles in the shift_dr state. The rstatus command is not supported on the ARM platform side, but a status register is provided instead. The rstatus may be performed in both debug and user modes.
- dmov command execution-The dmov command accesses SDMA internal registers. Executing a dmov instruction exchanges the 32-bit data values between the SDMA data register and the general register GReg[1].
- If the JTAG is used, the content of GReg1 is captured in the SDMA data register during the capture_dr state, then it is shifted out after 32 TCK clock cycles in the shift_dr state. During the update_dr state, GReg1 is updated with the new, shifted-in 32-bit data value. If the OnCE is driven from the ARM platform side, the data values contained in GReg1 and the SDMA data register are exchanged after detecting a write access to the ONCE_CMD register. The ONCE_DATA register must therefore be loaded first.

- exec_once command execution-The exec_once command executes the instruction loaded in the SDMA instruction register. The command may only be requested from debug mode. The SDMA returns to debug mode at the end of the execution.
- Change of flow instructions as well as instructions that may cause a context switch are not supported: The comprehensive list comprises done/yield/yiedge (except done 5), BF, BT, BSF, BDF, JMP, JSR, JMPR, JSRR, RET, and LOOP, as well as all the illegal instructions.

  No other command should be requested before the SDMA returns to debug mode. The SDMA status (for example, whether it is in debug mode or not) can be detected by polling with the rstatus OnCE command, monitoring the debug_mode pin, or checking the OnCE Status Register (SDMAARM_ONCE_STAT) register via the ARM platform control interface.

NOTE

Most of the instructions are single-cycle, which omits the step of polling the status. Loads and stores to DMA units are typical instructions that might require this polling.

If the JTAG is used, the 16-bit instruction is shifted in the SDMA instruction register after 16 TCK clock cycles in the shift_dr state. A request is sent to the core when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the request is sent to the SDMA when detecting a write access to the ONCE_CMD register. The ONCE_INSTR register must be therefore be loaded first.

- run_core command execution-The run_core command leaves debug mode and resume normal program execution. The next instruction executed is the last instruction decoded before entering debug mode. Be sure to restore core context before re-running the core. This procedure is detailed in Restoring the Context.
- If the JTAG is used, a 1-bit bypass value is shifted in the bypass register in the shift_dr state. The SDMA is rerun when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the core is rerun when detecting a write access to the ONCE_CMD register.

- exec_core command execution-The exec_core command resumes program execution from any address. The 16-bit instruction provided with the exec_core overwrites the last instruction decoded before entering debug mode. This command is designed to support change of flow instructions, so that a program execution can be restarted from any address. After executing an exec_core command, the SDMA leaves debug mode. The exec_core command is usually used with a jmp instruction.
- If the JTAG is used, the 16-bit branch instruction is shifted in the SDMA instruction register after 16 TCK clock cycles in the shift_dr state. The SDMA is rerun when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the SDMA reruns when detecting a write access to the ONCE_CMD register. The ONCE_INSTR register must therefore be loaded first. For example, to restart the SDMA from the program address 0x100, the instruction loaded should be a jump to address 0x100 instruction.

- debug_rqst command execution-The debug_rqst command puts the SDMA in debug mode. If the JTAG is used, a 1-bit bypass value is shifted in the bypass register during the shift_dr state. A debug request is sent to the SDMA when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the debug request is sent when detecting a write access to the ONCE_CMD register. When the SDMA is already in debug mode, this command is simply ignored.

- rbuffer command execution-The rbuffer command exports the content of the real time buffer (RTB). If the JTAG is used, the content of the real time buffer (RTB) is captured in the SDMA data register during the capture_dr state. The register is completely shifted out after maintaining the shift_dr state during 32 TCK clock cycles. If the OnCE is driven from the ARM platform side, the content of the RTB is captured in the ONCE_DATA register after detecting a write access to the ONCE_CMD register.
- bypass command execution-This command is only available from the JTAG interface. It enables daisy-chaining of the SDMA JTAG TAP controller with other JTAG TAP controllers. This command does not change the SDMA state and can be executed in any mode (run, debug, or sleep). It selects the bypass register of the TAP controller.

7.2.4.13.4 Registers Descriptions

See SDMACORE, and SDMAARM, for detailed information on each register.

7.2.4.13.4.1 Event Cell Counter Register (ECOUNT)

The event cell counter register is a 16-bit register that contains the number of times minus one that an event detection occurs before generating a debug request.

This register should be written before attempting to use the event detection counter during an event detection process. The event cell counter register is cleared on a JTAG reset.

7.2.4.13.4.2 Event Cell Address Registers (EAA or EAB)

The event cell contains two address registers-the event cell address register (a), called EAA, and the event cell address register (b), called EAB. Every address register is a 16-bit register that stores a user-defined address value. This value computes one of the following address conditions: addra_cond or addrb_cond. Every address register is cleared on a JTAG reset.

7.2.4.13.4.3 Event Cell Address Mask Register (EAM)

The event cell address mask register is a 16-bit register that contains a user-defined address mask value. This mask is applied to the address value latched from the memory address bus before comparing addresses.

NOTE

There is a common address mask value for the two address comparators. If bit *i* of this register is set, then bit *i* of the address value latched from the memory bus does not influence the result of the address comparison. The event cell address mask register is cleared on a JTAG reset.

7.2.4.13.4.4 Event Cell Data Register (ED)

The event cell data register is a 32-bit register that contains a user-defined data value. This data value is an input for the data comparator, which generates the data_cond condition.

The event cell data register is cleared on a JTAG reset.

7.2.4.13.4.5 Event Cell Data Mask Register (EDM)

The event cell data mask register is a 32-bit register that contains a user-defined data mask value. This mask is applied to the data value latched from the memory bus before comparing data.

Setting bit *i* of the event cell data mask register means that bit *i* of the data value latched from the address bus does not influence the result of the data comparison. The event cell data mask register is cleared on a JTAG reset.

7.2.4.13.4.6 Real Time Buffer Register (RTB)

The real Time Buffer register is a 32-bit register that stores and retrieves run-time information without putting the SDMA in debug mode.

Refer to Real Time Buffer for more details.

7.2.4.13.4.7 Event Control Register (ECTL)

The event cell control register is a 16-bit register that defines cell event occurrence conditions.

The event cell control register is cleared on a JTAG reset. See also OnCE Event Detection Unit for more details.

7.2.4.13.4.8 Trace Buffer (TB)

The Trace Buffer register retrieves the information in the Trace Buffer.

See Trace Buffer for more details.

7.2.4.13.4.9 OnCE Status Register (OSTAT)

The OnCE status register is a 16-bit register that contains processor and event detection unit status. The OSTAT is a read-only register.

Refer to OnCE Status Register (SDMAARM_ONCE_STAT) for detailed description of the individual fields in the OSTAT register.

The following figure shows the OSTAT structure.

Table 7-42. OnCE Status Register (OnCE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PST[3:0]				RCV	EDR	ODR	SWB	MST					ECDR[2:0]		

Where PST[3:0] is the SDMA core state, RCV is set when the real-time buffer (RTB) is modified. EDR, ODR, and SWB are set, respectively, when the SDMA has entered debug mode because of an external debug request, a OnCE debug_rqst command, or a software breakpoint. MST is set when the OnCE is controlled from the ARM platform control interface, and when ECDR is a three-flag set that shows the event cell condition(s) that put the core in debug mode. The OSTAT never provides more than one reason for entering debug mode.

There are two ways of accessing OSTAT content, as follows:

1. Send an rstatus command to the OnCE controller through the JTAG, or read the ONCE_STAT register through the ARM platform access. Executing the rstatus command through the JTAG can be performed in both user and debug modes.
2. Perform an SDMA read access to the location in the SDMA core memory map (OSTAT register) debug mode using the exec_once command. With this method of access, the SDMA state reflected by the PST (processor status bit) is always DATA.

The register may also be accessed by a running application.

7.2.4.13.5 JTAG Interface Requirements

Because the signals received from the JTAG (running on TCK) are transferred to the OnCE controller (running on the SDMA clock), a synchronization mechanism is required.

7.2.4.13.5.1 TCK Speed Limitation

In the JTAG top-level layer, the TDO signal is always captured on a TCK falling edge. To guarantee a stable TDO signal from the SDMA during this operation, a falling edge detection is performed on TCK.

Before being latched in the *I* flip-flop (see Figure 7-9) on TCK falling edge, the TDO signal must be stable at the input of the flip-flop. This condition is verified if the TCK period is superior to the following delay:

worst-case edge detection delay + negative-edge signal propagation delay + JTAG top-level logic propagation delay

The frequency relationship, TCK < CLK/8, limitation guarantees that all operations are performed as expected.

7.2.4.13.5.2 Synchronization Implementation

The figure found here shows the synchronization mechanism.

Flip-flops tck0, tck1, and tck2 perform falling- and rising-edge detections on TCK. They generate the posedge_detected and negedge_detected nets that are used to sample the TDI and TMS inputs into the respective tdi and tms flip-flops, and update the tdo flip-flop to yield the TDO output. In the design, the only signal that might go metastable is the output of the tck0 flip-flop. This signal is captured in the tck1 flip-flop and no logical operation is performed on it to minimize a metastability propagation risk.

The TDI and TMS flip-flops also cannot go metastable: The propagation time of the rising-edge detection signal through tck0, tck1, and tck2 guarantees that the TDI and TMS inputs are stable when captured in the TDI and TMS flip-flops.

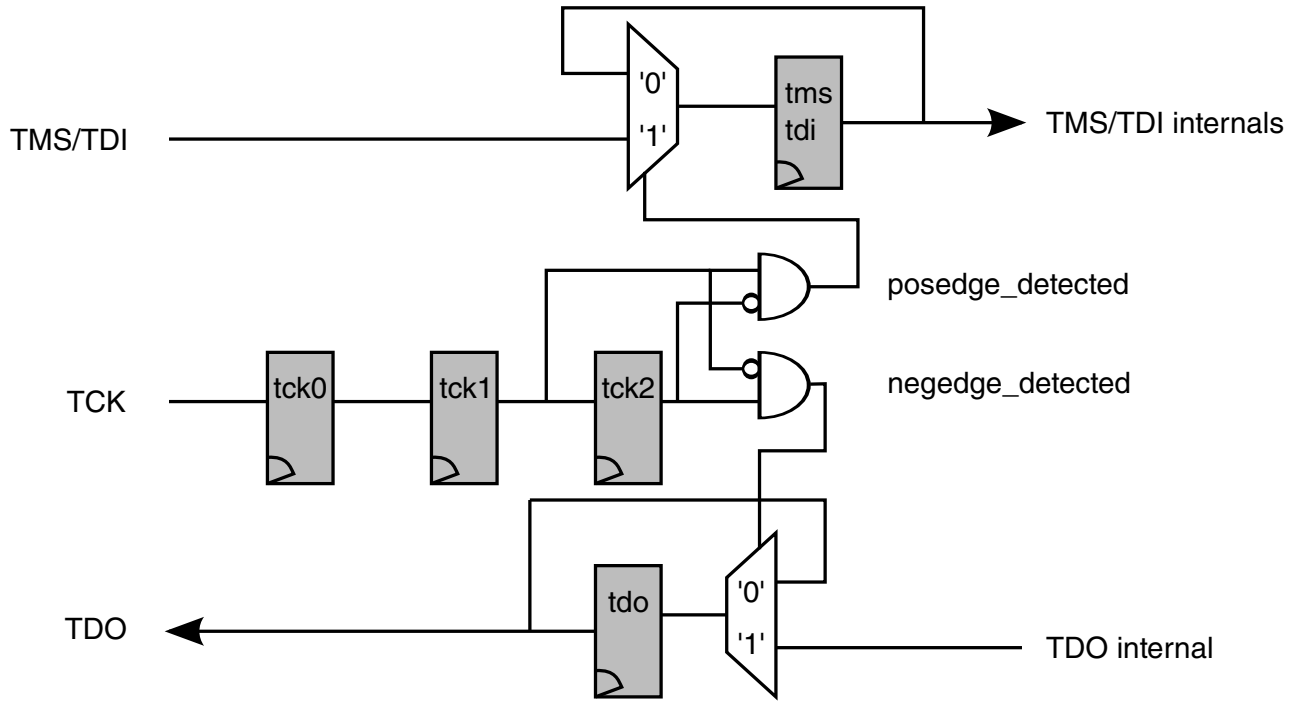


Figure 7-9. OnCE Synchronization Layer

The following figure shows synchronization timings. It takes three CLK clock cycles to synchronize TDI on the SDMA clock.

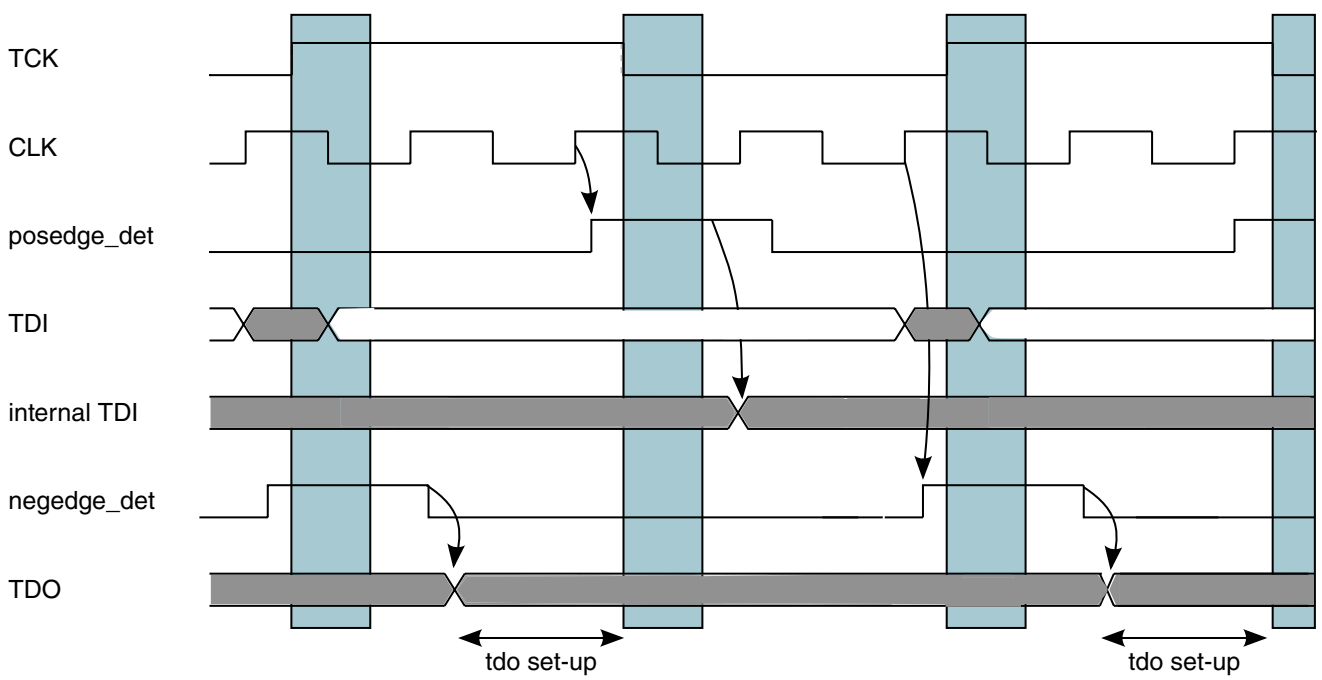


Figure 7-10. Synchronization Timings

7.2.4.13.5.3 JTAG Controller Start-Up Recommended Procedure

To ensure correct TAP controller initialization, it is recommended to use the following procedure:

1. Assert JTAG reset TRSTB (for example, set low).
2. Set TMS low.
3. Wait for 1 TCK clock.
4. Release JTAG reset TRSTB (for example, set high).
5. Wait for a minimum of five TCK cycles.

7.2.4.14 Using the OnCE

This section provides the elements necessary to run the OnCE during a debug process.

In addition to the basic set of commandsdescribed in OnCE Commands, more complex commands can be built to meet users' requirements.

7.2.4.14.1 Activating Clocks in Debug Mode

For power consumption issues, some clocks in the SDMA are disabled when not needed.

This is the case for instances when the SDMA is in sleep mode. Clock gating management depends on the interface used to control the OnCE.

- For the JTAG access, the SDMA clock gating must be turned off via the clk_gating_off input.
- For the ARM platform access, the SDMA clock gating is automatically turned off when the ARM platform access is enabled (see OnCE Enable (SDMAARM_ONCE_ENB)).

7.2.4.14.2 Getting the Current Status

Most of the commands the OnCE supports have an impact on the status of the SDMA.

It is not permissible to request the execution of an instruction on the SDMA from the OnCE while the SDMA is not in debug mode. Such a violation may cause unpredictable behavior, and it might be necessary to reset the SDMA.

Therefore, the value of the PST bits provided in the OnCE status register should always be checked before sending any request to the SDMA.

7.2.4.14.3 Methods of Entering Debug Mode

A debug request may be asserted at any time, but it is not always taken into account immediately. Debug mode cannot be entered in the middle of an instruction, or during the save or restore states of a context switch.

The request is ignored when the core is already in debug mode. Refer to Figure 7-2, which shows all possible transitions to the debug state, as there are several ways to enter debug mode.

7.2.4.14.3.1 External Debug Request During Reset

To enter debug mode after exiting reset, the external debug line has to be maintained high. This line is handled by the JTAG top-level block.

NOTE

The SDMA detects the debug requests only if the SDMA clock is running (see Activating Clocks in Debug Mode). The debug request line should be not be maintained high when the SDMA is in debug mode.

NOTE

The debug_rqst command (from the OnCE command set) is not supported during system reset.

7.2.4.14.3.2 Debug Request During Normal Activity

During normal activity, the SDMA enters debug mode when the following is true:

1. If the debug request line from the JTAG top-level is asserted, or
2. If the OnCE controller receives a debug_rqst command.

   The debug_rqst command can be sent by the JTAG access or by an access on the ARM platform side (if the ARM platform access is enabled).

7.2.4.14.3.3 Software Breakpoint Instruction

The SDMA enters debug mode at the end of the execution of a software breakpoint instruction. This instruction must be inserted in program flow executed by the core.

7.2.4.14.3.4 Event Detection Unit Matching Condition

If the event detection is enabled, a debug request is sent to the core after detecting a matching condition on the SDMA memory bus.

See OnCE Event Detection Unit for more details.

7.2.4.14.4 Executing Instructions in Debug Mode

The OnCE supports a mechanism to execute instructions in debug mode. If the SDMA is in debug mode, then the exec_once command can be used to execute an SDMA instruction from the OnCE controller. The SDMA returns to debug mode at the end of each execution.

Some instructions are not supported by the exec_once command: done/yield/yiedge (except done 5), BF, BT, BSF, BDF, JMP, JSR, JMPR, JSRR, RET, and LOOP, as well as all the illegal instructions are not supported.

NOTE

While instructions are executed in debug mode from the OnCE, the program counter of the SDMA is not incremented.

7.2.4.14.5 Command Sequences Examples

This section provides examples of command sequences that run the SDMA in debug mode. These sequences are available for both the ARM platform and JTAG accesses.

The following presents the syntax used in this section. The data field provided with each command is put in parenthesis with the command name. A '-' is used if the data field provided is a *don't care* value.

```
my_command(data_field);                 // executing my_command with a data field
my_command(-);                        // executing my_command with a don't care data field
```

The value returned by the command (if there is one) is referred by an assignment. In case the value returned by the command is not used, the assignment is omitted. For an ARM platform access, the value returned (it is always a data value) is obtained by reading back into the SDMA data register.

```
data_out = my_command(data_in); // returning a data value
```

To clarify the syntax, the instructions' opcodes are referred to by their names. In practice, use the corresponding 16-bit encoding.

7.2.4.14.5.1 Getting the SDMA Status

NOTE

Before executing any command that affects the SDMA (like dmov or exec_once), check that the SDMA is in debug mode.

Use the following snippet:

```
rstatus();        // read SDMA status until the SDMA is in debug mode
...
rstatus();
```

If the SDMA is not in debug mode, then a debug request must be generated. In this case, the SDMA enters debug mode at the end of the execution of the current instruction. Use this snippet:

```
debug_rqst(-);   // debug request
```

In the following sections, it is assumed that the SDMA was successfully put into debug mode.

7.2.4.14.5.2 Saving the Context

The first debug task is to save the SDMA context, which is the content of the eight general-purpose registers, the loop and PC-related registers, and the flags.

Use the general register GReg[1] as an intermediate register to export the entire context of the SDMA.

The following example shows how to save GReg[0], GReg[1], GReg[2] and GReg[3]. The sequence of commands used to export additional general registers is very similar to this.

Save GReg[0], GReg[1], GReg[2], and GReg[3]

```
GReg1_data = dmov(-);                          // the value exported is the content of
GReg[1]
exec_once("mov GReg1,GReg0");                      // puts the content of GReg[0] into
GReg[1]
GReg0_data = dmov(-);                          // the value exported is the content of
GReg[0]
exec_once("mov GReg1, GReg2");                     // puts the content of GReg[2] into
GReg[1]
GReg2_data = dmov(-);                          // the value exported is the content of
GReg[2]
exec_once("mov GReg1, GReg3");                     // puts the content of GReg[3] into
GReg[1]
GReg3_data = dmov(-);                          // the value exported is the content of
GReg[3]
```

Get the value of the internal flags (SF, DF, T, and LM), of the loop related registers (EPC and SPC), and of the PC-related registers (PC and RPC). Use a done 5, which is the formatting instruction dedicated to the debug. This instruction formats the flags and the

values contained in the registers. It also writes the resulting values into the channel context memory. It should not be used when entering debug from the IDLE state (for example, with no active channel script running on the SDMA), because it will update a channel context that may belong to any channel.

```
exec_once("done 5");                    // formatting the value of flags and registers
```

At this point, the channel context should be up-to-date in memory, and debug operations should now be possible. However, the context can be exported with the following instructions:

Exporting the Context

```
dmov(ctx_base_addr);                                  // loading GReg[1] with the channel
context base address
exec_once("ld GReg0,(GReg1,0)");                          // get RPC-PC into GReg0
exec_once("ld GReg1, (GReg1,1)");                         // get SPC-EPC into GReg1
Loop_data = dmov(-);                                // read back the value of Loop registers
exec_once("mov GReg1, GReg0");                          // puts the PC info into GReg1
PC_data = dmov(-);                                  // reads back the content of the PC registers
```

After this sequence of operations, the entire SDMA context is exported via the OnCE.

7.2.4.14.5.3 Restoring the Context

At this point in the operation, restore the context of the SDMA. It can be different from the original context located in memory, and the content previously saved into the debugging application via the OnCE.

The example found hereshows how it is possible to modify the current channel context.

Modifying the Current Channel Context

```
dmov(Loop_data);                                    // put Loop former value into GReg[1]
exec_once("mov GReg0, GReg1");                          // copy to GReg[0]
dmov(PC_data);                                      // put PC former value into GReg[1]
exec_once("mov GReg2, GReg1");                          // copy to GReg[2]
dmov(ctx_base_addr);                                  // put channel context base address into
GReg[1]
exec_once("st GReg0, (GReg1,1)");                         // restore Loop context
exec_once("st GReg2, (GReg1,0)");                         // restore PC context
```

Once the context in memory is the desired context (with or without applying the previous instruction sequence), it can be restored to the *real* PC and loop registers in the SDMA core:

```
exec_once("cpShReg");                     // restore flags and PC & loop related registers
```

After this command, the SDMA core PC, RPC, SPC, EPC registers, as well as the flags contain the same data as what is stored in the context RAM for the current channel.

The following example shows how to restore the context of general registers GReg[0], GReg[1], GReg[2] and GReg[3].

Restoring the General Register Context

```
dmov(GReg3_data);                    // put GReg[3] restore value in GReg[1]
exec_once("mov GReg3, GReg1"); // restore GReg[3]
dmov(GReg2_data);                    // put GReg[2] restore value in GReg[1]
exec_once("mov GReg2, GReg1"); // restore GReg[2]
dmov(GReg0_data);                    // put GReg[0] restore value in GReg[1]
exec_once("mov GReg0, GReg1"); // restore GReg[0]
dmov(GReg1_data);                    // restore GReg[1]
```

At this point, it is possible to restart the normal program execution.

NOTE

Every SDMA core general register value can be modified by a mov instruction, which makes modification of these registers easy during debug. Unfortunately, there is no such instruction as a mov to directly modify the contents of either PCU register or flag (PC, RPC, SPC, EPC, T, LM, SF, or DF). The cpShReg instruction is meant to provide a means for changing these register contents via the context memory.

7.2.4.14.5.4 Accessing the Memory

In the example shown here, it is assumed that the SDMA context is entirely saved. If true, it is permissible to modify the general purpose registers during debugging activity.

To perform a memory read access, the target address is stored via the OnCE in GReg[1], then the load instruction is executed on the SDMA (the data loaded from the memory overwrites the address contained in GReg[1]), and then the result value is read back via the OnCE.

```
macro READ:            dmov(target_addr);                                  // put the target
address in GReg[1]
                    exec_once("ld GReg1,(GReg1,0)");                        // execute the
load instruction
                    res_data = dmov(-);                                 // exports the result
data value
```

For a memory write access, the target address is written in GReg[0], and the value to store is written in GReg[1]. Then the store instruction is executed on the SDMA.

```
macro WRITE:             dmov(target_addr);                                  // puts the
target address in GReg[1]
                    exec_once("mov GReg0,GReg1");                          // puts the target
address in GReg[0]
                    dmov(target_data);                                  // puts the target
data in GReg[1]
                    exec_once("st GReg1,(GReg0,0)");                            // performs the
store operation
```

This sequence is shown as an example; however, many other sequences are possible.

NOTE

This sequence of commands can also be applied to memory-mapped registers.

7.2.4.14.5.5 Resuming Program Execution

Before resuming program execution, it is assumed that the SDMA context is properly restored. There are two ways to restart the SDMA.

Start by executing the last instruction fetched before entering debug mode, as follows.

```
run_core(-);                          // resume execution from where we stopped before
```

If necessary, restart the execution from a different address. In this case, use the exec_core command. The data field provided with this command must be the encoding of a jump instruction.

```
exec_core("jmp start_addr"); // rerun the SDMA from another address
```

In these two examples, the SDMA exits debug mode and keeps executing the code fetched from the memory.

7.2.4.14.5.6 Single Stepping in RAM

To execute a program step-by-step from the RAM, insert software breakpoints in the program flow at appropriate places so that the SDMA only executes one instruction before returning to debug mode.

First, read the next instruction to execute in the RAM. Then, depending on the value of this instruction, compute the address where a software breakpoint instruction should be inserted. The instruction at the corresponding address must be saved, and, the software breakpoint instruction is inserted. After restarting the SDMA, there is only one instruction executed before meeting the software breakpoint.

The following example shows the macro functions READ and WRITE, which correspond to the sequence of commands (described above) used to access the memory.

NOTE

The data read from the memory are 32-bit values, while the instructions are 16-bit values only. This is why it is best to only use addresses divided by two when accessing the memory.

READ and WRITE Macro Functions

```
next_instr = READ(run_addr/2);                        // read the next instruction to execute
// the tool now has to compute the address where the breakpoint
// instruction should be inserted, this address is the "bkpt_addr"
```

```
instr_save = READ(bkpt_addr/2);                          // save the instruction before
overwriting
STORE("bkpt instruction",bkpt_addr/2);                        // store the bkpt instruction
in memory
exec_core("jmp run_addr");                              // rerun the SDMA
rstatus(-);                                          // wait for the SDMA to enter debug mode
...
rstatus(-);
STORE(instr_save,bpkt_addr/2);                           // restore the instruction
overwritten
```

In case of branched conditional instructions, a breakpoint instruction should be written at the two possible target addresses.

7.2.4.14.5.7 Single Stepping in ROM

No single-step mechanism is supported in ROM. The program code can be loaded in the RAM, where the single-step mechanism can be executed.

7.2.4.14.6 OnCE Event Detection Unit

The event detection unit watches signals from the data memory bus (DMBUS), which the SDMA core uses to access its RAM, ROM, and memory mapped registers.

A debug request is sent to the OnCE controller when user-defined conditions on address and/or data values are true.

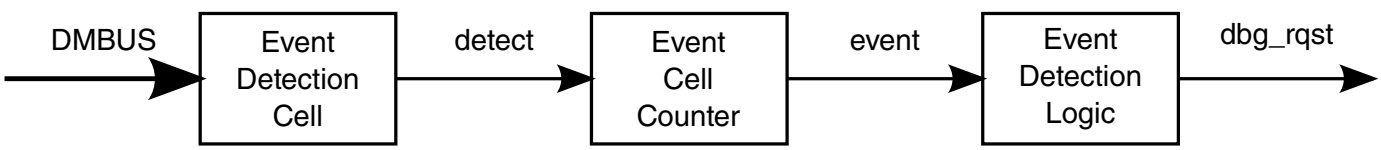


Figure 7-11. Event Detection Unit

A counter, provided with the detection cell, is decreased after an event detection. A debug request is sent to the core only when the counter reaches the value of 0. It is possible to disable the use of the counter if a debug request has to be generated after each event detection.

The event cell is the basic block that supports hardware breakpoints on an address value and/or data values coming from the SDMA memory bus. The trigger condition that generates the debug request is a mixed condition based on those values.

The following figure shows the event cell architecture. The event cell contains the address (stored in the memory address register) and the data (stored in the memory data register) used during the last memory access. There are some user-defined reference

values located in memory mapped registers-the event cell addresses, the event cell address mask, the event cell data, and the event cell data mask. These registers are accessed by standard load/store instructions just like regular memory locations.

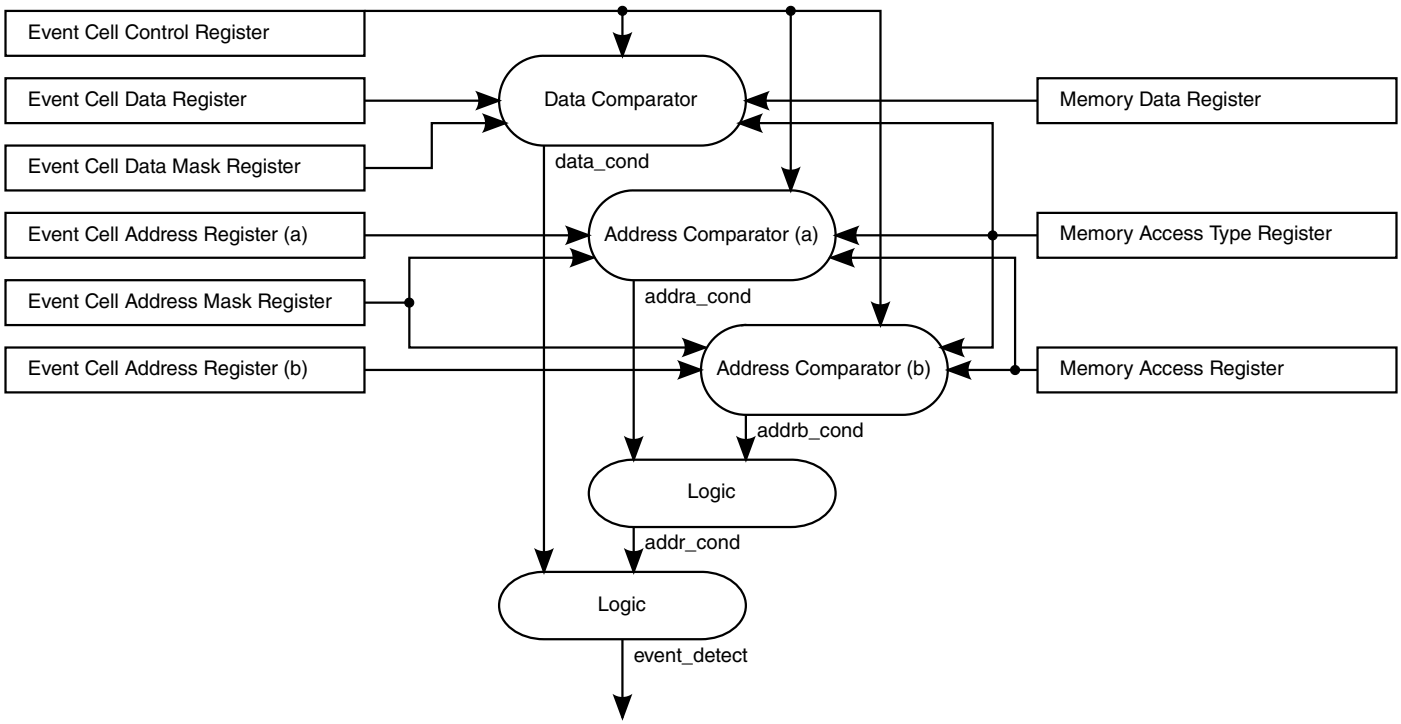


Figure 7-12. Event Cell Architecture

To define a memory breakpoint, three conditions are taken into account: The first two conditions are comparisons of the current memory address with user-defined reference addresses (these conditions are called addressA and addressB). The third condition consists of a comparison between the data received on the DMBUS and a user-defined reference data (this condition is called data). An intermediate address condition is set to express a dependency between addressA and addressB conditions.

7.2.4.14.7 Clock Gating and Reset

This section details how to use the clocks and handle the reset signals.

7.2.4.14.7.1 Clocks

Because the SDMA uses clock gating to save power, it is necessary to disable the clock gating and force the clocks to be enabled when using the OnCE.

When the OnCE is accessed through its JTAG interface, clock gating must be disabled outside the SDMA via a dedicated SDMA input port clk_gating_off. The reason why detection is not performed automatically by the SDMA internal hardware is that it would cost power to monitor activity on the JTAG interface.

When the OnCE is accessed through the ARM platform Control interface, clock gating is automatically turned off. This is done when bit 0 of the ONCE_ENB register (see OnCE Enable (SDMAARM_ONCE_ENB)) is set. A write access to this register is possible even when the OnCE clock is not running. If the ARM platform access is used, the bit in the ONCE_ENB register must be set before any attempt to access any other OnCE register.

7.2.4.14.7.2 Resets

The OnCE reset is different from the SDMA main reset.

Normally, activating the SDMA reset while keeping the OnCE reset inactive (when possible) enables you to reset the core without having to reprogram the OnCE.

7.2.4.14.8 Real Time Features

To rebuild the skeleton of a program execution, it is necessary to store the addresses of the program instructions where jumps are taken: A trace buffer is therefore provided. A real time buffer has also been added to receive data values written during a program execution.

The content of this register may be exported through JTAG ports without stopping the core.

7.2.4.14.8.1 Trace Buffer

The Trace Buffer is a 32-stage buffer that contains appropriate information to identify the 32 last changes of flow detected during a program execution.

The following figure shows an overview of the Trace Buffer.

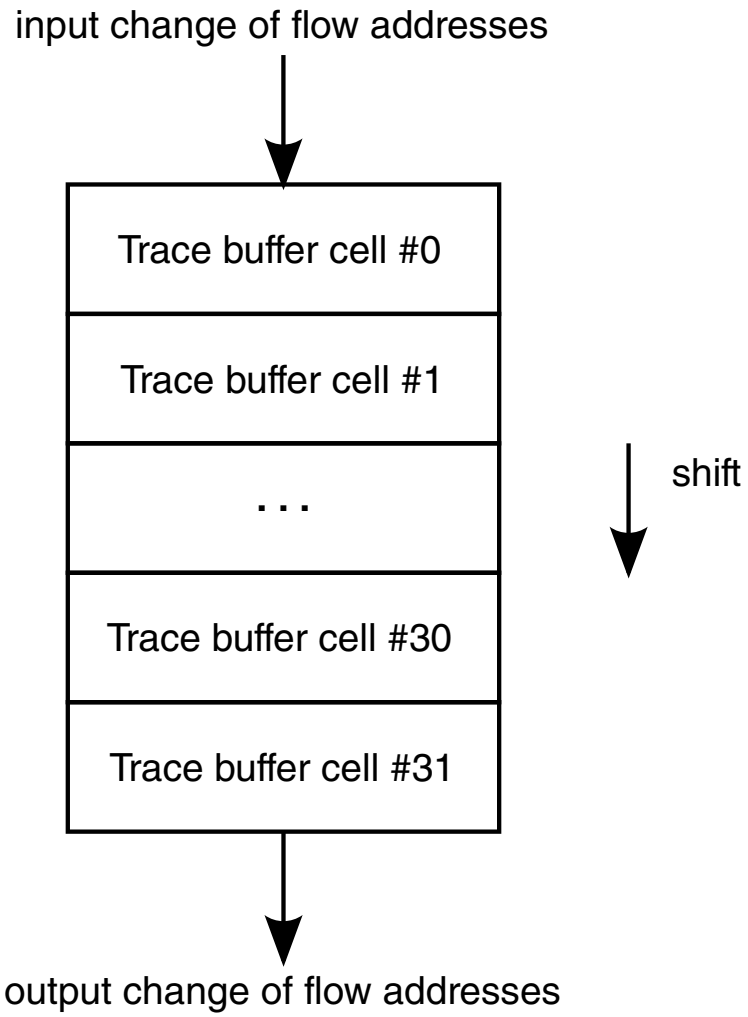


Figure 7-13. Trace Buffer

Each cell of the trace buffer contains two reference addresses and a flag. The flag is set when the addresses stored in the cell correspond to a valid change of flow; otherwise, the flag is cleared. The three most significant bits are unused.

After every change of flow detection, the address of current instruction and the address of the target instruction are stored at the top of the Trace Buffer (cell #0). The flag in the cell is set to indicate that a valid change of flow was detected. Former cell values are shifted one level down. The Trace Buffer contains the 32 last changes of flow. All the flags are reset on a software or a hardware reset, and after each transition from debug mode to user mode.

A memory mapped register of SDMA core, the Trace Buffer register (TB), is provided to read the content of the Trace Buffer. This operation should be done in debug mode. Performing a read access to the Trace Buffer register returns the content of the bottom of the Trace Buffer (cell #31). After every read access, the trace buffer is shifted one level down, and the flag at the top of the trace buffer is cleared.

A typical OnCE command sequence that retrieves the oldest change-of-flow information is a follows:

```
exec_once("mov r1, TB");                              // stores the oldest change-of-flow in
GReg1
dmov(-);                                          // retrieves GReg1 contents
```

This sequence requires the SDMA to be put in debug mode.

7.2.4.14.8.2 Real Time Buffer

The Real Time Buffer register (RTB) is a memory mapped register that can be accessed as a regular memory location by the SDMA core during program execution. This register is located in the OnCE.

Executing ar rbuffer command (see The OnCE Controller for further details) exports the content of this register through JTAG ports.

When a write access is performed at the memory location corresponding to the RTB, the receive flag (for example, the RCV bit) is set in the OnCE Status Register (OSR). This flag is cleared at the end of the execution of a rbuffer command.

NOTE

Every write access to the RTB memory location updates the RTB register even if the RCV flag is set. The RTB is cleared on a JTAG reset.

7.2.4.14.8.3 Emulation Pin

The debug_matched_event emulation pin reflects the matching condition status detected by the Event Detection Unit.

Since it can be necessary to detect conditions without triggering debug requests, it is possible to disable the generation of debug requests by the Event Detection Unit and still have the matching condition available on the emulation pin. This can be done by clearing the EN flag in the ECTL register.

7.2.4.14.8.4 Real-Time Debug Outputs

The table found here shows the debug signals that are available at the SDMA boundaries. Their availability at chip boundaries depends on the project.

Table 7-43. Real-Time Debug Output Pins

Pin	Description
debug_core_state[3:0]	The core_state bits reflect the state of the SDMA core. • The "Program" state is the usual instruction execution cycle. • The "Data" state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The "Change of Flow" state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel switching instructions). • The "Change of Flow in Loop" state is used when an error causes a hardware loop exit. • The "Debug" state means the SDMA is in debug mode. • The "Functional Unit" state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In "Sleep" modes, no script is running (this is the core idle state); the "after Reset" is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 is executed (boot operation). • The "in Sleep" states are the same as above except they do not have any corresponding channel: they are used when entering debug mode after reset; the reason is that it is necessary to return to the "Sleep after Reset" state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Context Switch Saving Channel 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change of Flow in Loop in Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Context Switch Restoring Channel
debug_yield	Pulse that is active when a yield (done 0) or a yieldge (done 1) instruction is executed. 0 - 1 yield/yieldge executed
debug_core_run	Active when the SDMA core is executing instructions. 0 Debug or sleep mode

Table continues on the next page...

Table 7-43. Real-Time Debug Output Pins (continued)

Pin	Description
	1 Run mode
debug_event_channel_sel	Indicates if debug_event_channel displays current channel or last received event 0- debug_event_channel[5:0] gives the number of the current channel 1- debug_event_channel[5:0] gives the number of the last received event
debug_event_channel[5:0]	Gives the number of any DMA request as soon as it is received or the number of the current channel. The value of debug_event_channel_sel indicates if debug_event_channel displays the current channel or last received event. The signal debug_event_channel_sel must be observed to determine what information is provided on debug_event_chanel at any given time.
debug_pc[13:0]	Program Counter value; it has a meaning when the core is in run mode.
debug_mode	Set when the core is in debug. 0 - 1 Core is in debug
debug_bus_error	Set when an error was received during a load or a store (ld, st, ldf, or stf instruction) and registered in SF or DF flag. 0 No error during last load/store 1 Error during last load/store
debug_bus_device[4:0]	Indicates the device or functional unit that is accessed by the current instruction. The debug_bus_device output is always valid when in sleep mode, debug mode, or executing any instruction that does not access the functional units or the memory mapped devices, "no access" is output. 0 No access 1 MSA 2 MDA 3 MD 4 MS 5 PSA 6 PDA 7 PD 8 PS 9 RESERVED 10 RESERVED 11 RESERVED 12 RESERVED 13 CA 14 CS 15 Reserved 16 Memory (RAM or ROM) 17 Memory mapped register

Table continues on the next page...

Table 7-43. Real-Time Debug Output Pins (continued)

Pin	Description
	18 Peripheral #1 19 Peripheral #2 20 Peripheral #3 21 Peripheral #4 22 Peripheral #5 23 Peripheral #6 24 Peripheral #7 25 Peripheral #8 26 Peripheral #9 27 Peripheral #10 28 Peripheral #11 29 Peripheral #12 30 Peripheral #13 31 Peripheral #14
debug_bus_rwb	Indicates the direction of the access given by debug_bus_device 0 Write access (st or stf) 1 Read access (ld or ldf)
debug_matched_dmbus	Pulse indicating the OnCE event detection unit has detected a match on the data bus during an access to memory (RAM or ROM), a memory mapped register or a peripheral that is hooked to the SDMA. 0 - 1 data bus match detected
debug_rtbuffer_write	Pulse indicating when the real-time buffer is written by the core. 0 - 1 RTB was modified
debug_evt_chn_lines[7:0]	Eight lines that generate short pulses when DMA requests are received or channels are (re)started. Every line is controlled through two parameters defined in registers Cross-Trigger Events Configuration Register 1 (SDMAARM_XTRIG_CONF1) (as described in SDMAARM). The following two parameters are available for every line: • CNF-Indicates what is monitored on the line: 0 for a channel start, 1 for a DMA request reception • NUM[5:0]-Gives the number of the DMA request or channel to monitor

The matched_event emulation pin reflects the matching condition status detected by the Event Detection Unit. Because it can be necessary to detect conditions without triggering debug requests, it is possible to disable the generation of debug requests by the Event Detection Unit and still have the matching condition available on the emulation pin. This can be done by clearing the EN flag in the ECTL register.

All real-time debug outputs are disabled by default (for example, they are stuck to 0) to avoid power consumption when they are not used. They are enabled when bit 11 (RTDOBS) of the Configuration Register (SDMAARM_CONFIG) is set. Signals provided to the system JTAG controller for SDMA debug mode status will also be enabled when the *clk_gating_off* input is asserted.

7.2.5 Instruction Set

7.2.5.1 Instruction Encoding

This section presents a short summary of the instruction codes. All context switch instructions are listed for information only; they cannot function properly out of the context switch routine.

```
x...x - don't care

rrr - destination/source general register

sss - additional source general register

bbb - general register used as address base register

ddddd - address displacement

nnnnn - bit number
uuuuuuuu - function unit command bits

pppppppp - branch displacement (signed)

iiiiiiii - 8-bit immediate

jjj - control bit to clear

ff - flag to clear
00000jjj00000000        - done (done,yield,wait)
00000jjj00000001        - notify
00000xxx00000010        - reserved
00000xxx00000011        - reserved
00000xxx00000100        - reserved
0000000000000101     - softBkpt
0000000100000101     - reserved
0000001000000101     - reserved
0000001100000101     - reserved
0000010000000101     - reserved
0000010100000101     - reserved
0000011000000101     - reserved
0000011100000101     - reserved
0000000000000110     - ret
0000000100000110     - reserved
0000001000000110     - reserved
0000001100000110     - reserved
0000010000000110     - reserved
0000010100000110     - reserved
0000011000000110     - reserved
0000011100000110     - reserved
```

```
000000ff00000111       - clrf ff
0000010000000111     - reserved
0000010100000111     - reserved
0000011000000111     - reserved
0000011100000111     - illegal
00000rrr00001000        - jmpr r
00000rrr00001001        - jsrr
00000rrr00001010        - ldrpc r
00000rrr00001011        - reserved
00000rrr000011xx          - reserved
00000rrr00010000        - revb
00000rrr00010001        - revblo
00000rrr00010010        - rorb
00000rrr00010011        - reserved
00000rrr00010100        - ror1
00000rrr00010101        - lsr1
00000rrr00010110        - asr1
00000rrr00010111        - lsl1
00000rrr001nnnnn        - bclri r,n
00000rrr010nnnnn        - bseti r,n
00000rrr011nnnnn        - btsti r,n
00000xxx10000xxx           - reserved
00000rrr10001sss           - mov
00000rrr10010sss           - xor
00000rrr10011sss           - add
00000rrr10100sss           - sub
00000rrr10101sss           - or
00000rrr10110sss           - andn
00000rrr10111sss           - and
00000rrr11000sss           - tst
00000rrr11001sss           - cmpeq
00000rrr11010sss           - cmplt
00000rrr11011sss           - cmphs
0000011011100000     - reserved
0000011011100001     - reserved
0000011011100010     - cpShReg
0000011011100011     - reserved
0000011011100100     - reserved
0000011011100101     - reserved
0000011011100110     - reserved
0000011011100111     - reserved
00000xxx11101xxx           - reserved
00000xxx11110xxx           - reserved
00000xxx11111xxx           - reserved
00001rrriiiiiiii                - ldi r,i
00010rrriiiiiiii                - xori r,i
00011rrriiiiiiii                - addi r,i
00100rrriiiiiiii                - subi r,i
00101rrriiiiiiii                - ori r,i
00110rrriiiiiiii                - andni r,i
00111rrriiiiiiii                - andi r,i
01000rrriiiiiiii                - tsti r,i
01001rrriiiiiiii                - cmpeqi r,i
01010rrrddddbbb         - ld r,(d,b)
01011rrrddddbbb         - st r,u
01100rrruuuuuuuu         - ldf r,u
01101rrruuuuuuuu         - stf r,u
011100xxxxxxxxxx               - reserved
011101xxxxxxxxxx               - reserved
011110ffnnnnnnnn       - Loop ff flags are reset
01111100pppppppp     - bf pc=pc+signed(pppppppp)+1
01111101pppppppp     - bt pc=pc+signed(pppppppp)+1
01111110pppppppp     - bsf pc=pc+signed(pppppppp)+1
01111111pppppppp     - bdf pc=pc+signed(pppppppp)+1
10aaaaaaaaaaaaaa     - jmp absolute
11aaaaaaaaaaaaaa     - jsr absolute
```

7.2.5.2 SDMA Instruction Set

This section describes all the useful instructions from the SDMA set.

Table 7-44. SDMA Instruction List

Instruction	Description	Page
ADD	Addition	ADD (Addition)
ADDI	Add with Immediate Value	ADDI (Add with Immediate Value)
AND	Logical AND	AND (Logical AND)
ANDI	Logical AND with Immediate Value	ANDI (Logical AND with Immediate Value)
ANDN	Logical AND NOT	ANDN (Logical AND NOT)
ANDNI	Logical AND with Negated Immediate Value	ANDNI (Logical AND with Negated Immediate Value)
ASR1	Arithmetic Shift Right by 1 Bit	ASR1 (Arithmetic Shift Right by 1 Bit)
BCLRI	Bit Clear Immediate	BCLRI1 (Bit Clear Immediate)
BDF	Conditional Branch if Destination Fault	BDF (Conditional Branch if Destination Fault)
BF	Conditional Branch if False	Functional Units Programming Model
BSETI	Bit Set Immediate	BSETI (Bit Set Immediate)
BSF	Conditional Branch if Source Fault	BSF (Conditional Branch if Source Fault)
BT	Conditional Branch if True	BT (Conditional Branch if True)
BTSTI	Bit Test immediate	BTSTI (Bit Test immediate)
CLRF	Clear ARM platform flags	CLRF (Clear ARM platform flags)
CMPEQ	Compare for Equal	CMPEQ (Compare for Equal)
CMPEQI	Compare with Immediate for Equal	CMPEQI (Compare with Immediate for Equal)
CMPHS	Compare for Higher or Same	CMPHS (Compare for Higher or Same)
CMPLT	Compare for Less Than	CMPLT (Compare for Less Than)
cpShReg	Update Context of PCU Registers and Flags	cpShReg (Update Context of PCU Registers and Flag)
DONE	DONE, Yield	DONE (DONE, Yield)
ILLEGAL	ILLEGAL Instruction	ILLEGAL (ILLEGAL Instruction)
JMP	Unconditional Jump Immediate	JMP (Unconditional Jump Immediate)
JMPR	Unconditional Jump	JMPR (Unconditional Jump)
JSR	Unconditional Jump to Subroutine Immediate	JSR (Unconditional Jump to Subroutine Immediate)
JSRR	Unconditional Jump to Subroutine	JSRR (Unconditional Jump to Subroutine)
LD	Load Register	LD (Load Register)
LDF	Load Register from Functional Unit	LDF (Load Register from Functional Unit)
LDI	Load Register with Immediate Value	LDI (Load Register with Immediate Value)
LDRPC	Load from RPC to Register	LDRPC (Load from RPC to Register)

Table continues on the next page...

Table 7-44. SDMA Instruction List (continued)

Instruction	Description	Page
LOOP	Hardware Loop	LOOP (Hardware Loop)
LSL1	Logical Shift Left by 1 Bit	LSL1 (Logical Shift Left by 1 Bit)
LSR1	Logical Shift Right by 1 Bit	LSR1 (Logical Shift Right by 1 Bit)
MOV	Logical Move	MOV (Logical Move)
NOTIFY	Notify to ARM platform	NOTIFY (Notify to ARM platform)
OR	Logical OR	OR (Logical OR)
ORI	Logical OR with Immediate Value	ORI (Logical OR with Immediate Value)
RET	Return from Subroutine	RET (Return from Subroutine)
REVB	Reverse Byte Order	REVB (Reverse Byte Order)
REVBLO	Reverse Low Order Bytes	Reverse Low Order Bytes(REVBLO)
ROR1	Rotate Right by 1 Bit	ROR1 (Rotate Right by 1 Bit)
RORB	Rotate Right by 1 Byte	RORB (Rotate Right by 1 Byte)
SOFTBKPT	Software Breakpoint	SOFTBKPT (Software Breakpoint)
ST	Store Register	ST (Store Register)
STF	Store Register in Functional Unit	STF (Store Register in Functional Unit)
SUB	Subtract	SUB (Subtract)
SUBI	Subtract with Immediate	SUBI (Subtract with Immediate)
TST	Test with Zero	TST (Test with Zero)
TSTI	Test Immediate	TSTI (Test Immediate)
XOR	Logical Exclusive OR	XOR (Logical Exclusive OR)
XORI	Exclusive OR with Immediate	XORI (Exclusive OR with Immediate)

7.2.5.2.1 ADD (Addition)

Operation:

```
GReg[r] ← GReg[s] + GReg[r]

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: add r,s

Example: add 0,3
```

ADD GReg[3] and GReg[0] and store the result in GReg[0]

CPU Flags: T

Cycles: 1

Description: Performs the ADDition of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*. The T flag is set if the result of the operation is 0. It is cleared if the result is not 0.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	1	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.2 ADDI (Add with Immediate Value)

Operation:

```
GReg[r] ← GReg[r] + immediate

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: addi r,immediate

Example: add 6,112
```

ADD GReg[6] and decimal value 112 and store the result in GReg[6]

CPU Flags: T

Cycles: 1

Description: Adds a 0-extended immediate value to a general register; stores the result in the general register. The flag T is set when the result of the operation is 0; otherwise, it is cleared. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

 ...

11111110 - 254

11111111 - 255
```

7.2.5.2.3 AND (Logical AND)

Operation:

```
GReg[r] ← GReg[s] & GReg[r]
```

Assembler:

```
Syntax: and r,s

Example: and 1,2
```

AND GReg[1] and GReg[2] and store the result in GReg[1]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the AND of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	1	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.4 ANDI (Logical AND with Immediate Value)

Operation:

```
GReg[r] ← GReg[r] & immediate
```

Assembler:

```
Syntax: andi r,immediate

Example: andi 7,45
```

AND GReg[7] and decimal value 45 and store the result in GReg[7]

CPU Flags: unaffected

Cycles: 1

Description: Performs an AND between a 0-extended immediate value and a general register; stores the result in the general register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

7.2.5.2.5 ANDN (Logical AND NOT)

Operation:

```
GReg[r] ← ~GReg[s] & GReg[r]
```

Assembler:

```
Syntax:andn r,s

Example: andn 3,4
```

AND GReg[3] and NOT GReg[4] (bit inverted) and store the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the AND of the negation of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*.

Instruction Format:

Table 7-45. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	1	0	s	s	s

Instruction Fields:

rrr /sss - destination register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.6 ANDNI (Logical AND with Negated Immediate Value)

Operation:

```
GReg[r] ← GReg[r] & ~immediate
```

Assembler:

```
Syntax: andni r,immediate

Example: andni 0,2
```

AND GReg[0] and decimal value -3 (inverted 32-bit value 2) and store the result in GReg[0]

CPU Flags: unaffected

Cycles: 1

Description: Performs an AND between the negation of a 0-extended 8-bit immediate value and a general register; stores the result in the general register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

7.2.5.2.7 ASR1 (Arithmetic Shift Right by 1 Bit)

Operation:

```
GReg[r]:{b31,b30,...,b1,b0} ← GReg[r]:{b31,b31,b30,...,b1}
```

Assembler:

```
Syntax: asr1 r

Example: asr1 3
```

divide by 2 the signed value of GReg[3] and store the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any general register to the right and keep the same sign: The left bit (bit 31) is kept untouched.

Instruction Format:

Table 7-46. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	1	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.8 BCLRI1 (Bit Clear Immediate)

Operation:

```
GReg[r]:{b31,...,b(i+1),0,b(i-1),...,b0} ← GReg[r]:{b31,...,b(i+1),b(i),b(i-1),...,b0}
```

Assembler:

```
Syntax: bclri r,i

Example: bclri 1,12
```

clear bit 12 in GReg[1]

CPU Flags: Unaffected

Cycles: 1

Description: Clear the bit of register r specified by the 5-bit immediate field

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	1	i	i	i	i	i

rrr - register field:

```
000 - GReg[0]
```

```
001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiii - immediate value:

```
00000 - 0

00001 - 1

...

11110 - 30

11111 - 31
```

7.2.5.2.9 BDF (Conditional Branch if Destination Fault)

Operation:

```
if (DF == 1) PC ← PC + 1 + displacement else PC ← PC + 1
```

Assembler:

```
Syntax:bdf label

Example: bdf LLL
```

Jump to LLL if DF is set, or go to the next instruction if DF is cleared; the displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: If flag DF is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag DF is cleared, no jump is performed: The next instruction is located at the next PC address.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	p	p	p	p	p	p	p	p

Instruction Fields:

pppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110 - 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 - (-2)

11111111 - (-1)
```

7.2.5.2.10 BF (Conditional Branch if False)

Operation:

```
if (T == 0)

PC ← PC + 1 + displacement

else

PC ← PC + 1
```

Assembler:

```
Syntax: bf label

Example: bf LLL
```

Jump to LLL if T is cleared, or go to the next instruction if T is set. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag T is cleared, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag T is set, no jump is performed: The next instruction is located at the next PC address.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	0	0	p	p	p	p	p	p	p	p

Instruction Fields:

ppppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110 - 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 - (-2)

11111111 - (-1)
```

7.2.5.2.11 BSETI (Bit Set Immediate)

Operation:

```
GReg[r]:{b31,...,b(i+1),1,b(i-1),...,b0} ← GReg[r]:{b31,...,b(i+1),b(i),b(i-1),...,b0}
```

Assembler:

```
Syntax: bseti r,i

Example: bseti 6,5
```

Set bit 5 in GReg[6]

CPU Flags: Unaffected

Cycles: 1

Description: Sets bit number i in the selected General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	1	0	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]
```

```
010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiii - bit number field:

```
00000 - 0

00001 - 1

...

11110 - 30

11111 - 31
```

7.2.5.2.12 BSF (Conditional Branch if Source Fault)

Operation:

```
if (SF == 1) PC ← PC + 1 + displacement else PC ← PC + 1
```

Assembler:

```
Syntax: bsf label

Example: bsf LLL
```

Jump to LLL if SF is set, or go to the next instruction if SF is cleared. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag SF is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag SF is cleared, no jump is performed: The next instruction is located at the next PC address.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	0	p	p	p	p	p	p	p	p

Instruction Fields:

pppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110 - 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 - (-2)

11111111 - (-1)
```

7.2.5.2.13 BT (Conditional Branch if True)

Operation

```
if (T == 1)

PC ← PC + 1 + displacement

else

PC ← PC + 1
```

Assembler

```
Syntax: bt label

bt LLL
```

Jump to LLL if T is set, or go to the next instruction if T is cleared. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag T is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag T is cleared, no jump is performed: The next instruction is located at the next PC address.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	0	1	p	p	p	p	p	p	p	p

pppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110- 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 -  (-2)

11111111 - (-1)
```

7.2.5.2.14 BTSTI (Bit Test immediate)

Operation:

```
T ← GReg[r]:b(i)
```

Assembler:

```
Syntax: btsti r,i

Example: btsti 2,29
```

Test bit 29 in GReg[2] and copy its value in flag T

CPU flags: T

Cycles: 1

Description: T is loaded with the value of bit number i from the selected general register.

Instruction Format:

Table 7-47. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	1	1	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]
```

```
010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiii - bit number field:

```
0000 - 0

0001 - 1

...

11110 - 30

11111 - 31
```

7.2.5.2.15 CLRF (Clear ARM platform flags)

Operation:

```
 if (ff%2 == 0)

SF ← 0

if (ff/2 == 0)

DF ← 0
```

Assembler:

```
Syntax: clrf ff

Example: clrf 2
```

Clear flag SF and keep flag DF unchanged

CPU Flags: SF, DF

Cycles: 1

Description: Clears a selection of the ARM platform fault flags: SF, DF, both SF and DF or none can be cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	f	f	0	0	0	0	0	1	1	1

Instruction Fields:

ff - flags field:

```
00 - clear SF and clear DF

01 - clear DF

10 - clear

SF 11 - no clear
```

7.2.5.2.16 CMPEQ (Compare for Equal)

Operation:

```
T ← (GReg[s] == GReg[r])
```

Assembler:

```
Syntax: cmpeq r,s

Example: cmpeq 7,5
```

Compare GReg[7] and GReg[5] and set flag T if they are equal

CPU flags: T

Cycles: 1

Description: Subtracts the destination general register *r* from the source general register *s*, and sets T if the result is 0, clears T if the result is not 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	0	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.17 CMPEQI (Compare with Immediate for Equal)

Operation:

```
T ← (GReg[r] == immediate)
```

Assembler:

```
Syntax: cmpeqi r,immediate

Example: cmpeqi 2,13
```

Compare GReg[2] and decimal value 13 and set flag T if they are equal

CPU Flags: T

Cycles: 1

Description: Subtracts the 0-extended 8-bit immediate value from the general register, and sets T if the result is 0, clears T if the result is not 0. The immediate value is the low-order byte of the instruction.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - destination register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254
```

```
11111111 - 255
```

7.2.5.2.18 CMPHS (Compare for Higher or Same)

Operation:

```
T ← (GReg[r] ≥ GReg[s])
```

Assembler:

```
Syntax: cmphs r,s

Example: cmphs 0,1
```

Compare GReg[0] and GReg[1] and set flag T if GReg[0] is higher than or equal to GReg[1]

CPU Flags: T

Cycles: 1

Description: Compares the destination general register *r* and the source general register *s*, and sets T if the destination general register *r* is higher than or equal to the source general register *s*, clears T otherwise. The comparison is unsigned.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	1	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.19 CMPLT (Compare for Less Than)

Operation:

```
T ← (GReg[r] < GReg[s])
```

Assembler:

```
Syntax: cmplt r,s

Example: cmplt 7,4
```

Compare GReg[7] and GReg[4] and set flag T if GReg[7] is lower than GReg[4]

CPU Flags: T

Cycles: 1

Description: Compares the destination general register *r* and the source general register *s*, and sets T if the destination general register *r* is lower than the source general register *s*, clears T otherwise. The comparison is signed.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	1	0	s	s	s

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.20 cpShReg (Update Context of PCU Registers and Flag)

Assembler:

```
Syntax: cpShReg
```

CPU Flags: none

Cycles: 1

Description: SF, RPC, T, PC,LM, EPC, DF, and SPC registers are updated according to the value of their corresponding bits in the context memory. This instruction must only be used in debug mode via the OnCE. It reverses the done *5* operation.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0	1	1	1	0	0	0	1	0

7.2.5.2.21 DONE (DONE, Yield)

Operation:

```
if (jjj&6 == 2) HE[CCR] ← 0

if (jjj == 3) HI[CCR] ← 1

if (jjj == 4) EP[CCR] ← 0


if ((jjj == 0) && (NCP > CCP)) CCR ← NCR

else if ((jjj == 1) && (NCP >= CCP))

CCR ← NCR

else

CCR ← NCR
```

(CCR stands for Current Channel Register; NCR stands for Next Channel Register)

Assembler:

```
Syntax: done jjj

Example: done 3
```

Clear HE bit for the current channel, send an interrupt to the ARM platform for the current channel and reschedule.

CPU Flags: Unaffected

Cycles: Variable if a context switch is done, 1 otherwise

Description: Clears one of the channel enabling bits (HE or EP for the corresponding channel number) if required. Sends an interrupt to the corresponding ARM platform by setting the appropriate flag, if required (HI for the corresponding channel number). Reschedules according to the mode and the NCP (Next Channel Priority) and CCP (Current Channel Priority) values. According to the scheduling decision, the NCR (Next Channel Register) is copied to the CCR (Current Channel Register) and channel contexts are switched. If several channels with the same highest priority are pending, they are ordered by their number from 31 down to 0. The higher number is selected (for example, channel 26 is selected if channels 3, 12, 14, and 26 with the same highest priority are pending). If no flag is modified, the reschedule can allow the replacement of the current

channel by another channel with a priority strictly greater than the current channel priority (yield). Or, it can allow the replacement of the current channel by another channel with a priority greater than or equal to the current channel priority (yieldge). In the latter case, the selected channel will always be the first one with the same priority, starting from channel number 31 down to channel 0 (the current channel does not belong to the set of selectable channels).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	j	j	j	0	0	0	0	0	0	0	0

jjj - Channel Flags field:

000 - No channel flags affected: Reschedule only if the next channel priority is greater than current channel priority (yield)

001 - No channel flags affected: Reschedule only if the next channel priority is greater than or equal to the current channel priority (yieldge)

010 - Clear HE for the current channel and reschedule 011 - Clear HE, set HI for the current channel and reschedule 100 - Clear EP for the current channel and reschedule

101 - Reserved for debug to copy relevant registers into context memory

110 - RESERVED

111 - RESERVED

For the scheduling rules, refer to Scheduler Functional Description. Every possible done instruction is further described as follows:

- done 0/yield is executed by a channel script when it accepts preemption by a higher priority channel;
- done 1/yieldge is executed by a channel script when it accepts preemption by a higher priority channel and it also accepts a roll-up with other channels that have the same priority;
- done 2 is executed by a channel script that was triggered by a ARM platform start via the Channel Start (SDMAARM_HSTART) register, when its task is completed and it requires termination;
- done 3 is executed by a channel script that was triggered by a ARM platform start via the Channel Start (SDMAARM_HSTART) register, when its task is completed, it requires termination and it needs to trigger an interrupt to the ARM platform upon closure;

- done 4 is executed by a channel script that was triggered by a DMA request, when its task is completed and it requires termination;
- done 5 is used in debug mode only; it copies the PCU registers and flags to the context memory of the current channel;

7.2.5.2.22 ILLEGAL (ILLEGAL Instruction)

Operation:

```
PC ← 0001
```

Assembler:

```
Syntax: illegal
```

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the Illegal instruction routine located at address 0001. All unauthorized instructions result in an Illegal instruction behavior; however, the ILLEGAL instruction must be used to guarantee software compatibility with future versions of the SDMA.

Instruction Format

Table 7-48. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

7.2.5.2.23 JMP (Unconditional Jump Immediate)

Operation:

```
PC ← absolute_address
```

Assembler:

```
Syntax: jmp label

Example: jmp LLL
```

The assembler translates the label to the exact address

CPU Flags:Unaffected

Cycles: 2

Description: Jumps to the absolute address contained the lower 14 bits of the instruction (the PC is a 14-bit register).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	a	a	a	a	a	a	a	a	a	a	a	a	a	a

aaaaaaaaaaaaaa - address field:

```
00000000000000 - 0

00000000000001 - 1

...

11111111111110 - 16382

11111111111111 - 16383
```

7.2.5.2.24 JMPR (Unconditional Jump)

Operation:

```
PC ← GReg[r]
```

Assembler:

```
Syntax: jmpr r

Example: jmpr 0
```

Jump to address stored in GReg[0]

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the absolute address contained in a General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	0	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]
```

```
010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.25 JSR (Unconditional Jump to Subroutine Immediate)

Operation:

```
RPC ← PC + 1

PC ← absolute_address
```

Assembler:

```
Syntax: jsr r

Example:jsr LLL
```

Jumps to subroutine starting at LLL; the assembler translates the label to exact address

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the subroutine located at the absolute address contained the lower 14 bits of the instruction (the PC is a 14-bit register).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	a	a	a	a	a	a	a	a	a	a	a	a	a	a

aaaaaaaaaaaaaa - address field:

```
00000000000000 - 0

00000000000001 - 1

...

11111111111110 - 16382

11111111111111 - 16383
```

7.2.5.2.26 JSRR (Unconditional Jump to Subroutine)

Operation:

```
RPC ← PC + 1

PC ← GReg[r]
```

Assembler:

```
Syntax: jsrr r

Example:jsrr 5
```

Jumps to subroutine located at address stored in GReg[5]

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the subroutine at address contained in a General Register

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	0	1

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.27 LD (Load Register)

Operation:

```
GReg[r] ← [GReg[b] + displacement]

if (transfer_error)

SF ← 1

else

SF ← 0
```

Assembler:

```
Syntax: ld r,(b,displacement)

Example: ld 1,(2,23)
```

Loads data into GReg[1]; the data is located at address obtained by adding decimal value 23 to GReg[2]

CPU Flags: SF

Cycles: 2+n where n is 0 for ROM, RAM or memory mapped registers, and n is the number of wait-states of the peripheral for a peripheral access

Description: Adds a 5-bit 0-extended displacement to a base address in General Register b; the result is the address of the data to fetch on the DM bus. The data received from the bus is stored in the destination General Register r. If an error occurs during the transfer, the flag SF is set, else it is cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	r	r	r	d	d	d	d	d	b	b	b

rrr / bbb - register field:

```
 000 - GReg[0]

 001 - GReg[1]

 ...

 111 - GReg[7]
```

ddddd - displacement value:

```
00000 - 0

00001 - 1

...

11111 - 31
```

7.2.5.2.28 LDF (Load Register from Functional Unit)

Operation:

```
GReg[r] ← [fu_address]

if (transfer_error)

SF ← 1
```

```
else

SF ←0
```

fu_address is an 8-bit field and depends on addressed functional unit

Assembler:

```
Syntax: ldf r,fu_address

Example: ldf 0,13
```

Loads data coming from the Burst DMA register MD into GReg[0]; it is a 32-bit access with no prefetch

CPU Flags: SF

Cycles: 1+n where n is the number of wait-states that may be inserted by the functional unit

Description: Sends an 8-bit address on the Functional Unit Bus (FU bus) and stores the data received from the bus in the destination General Register r. If an error occurs during the transfer, the flag SF is set, else it is cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	r	r	r	f	f	f	f	f	f	f	f

See the following sections for more details of the LDF instruction usage with each functional unit:

- Burst DMA Read (ldf) for Burst DMA
- Peripheral DMA Read (ldf)-Read Mode for Peripheral DMA

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

ffffffff - functional unit source register and action (unspecified values are reserved):

```
00000000 - MSA

00000100 - MDA

00001001 - MD byte

00001010 - MD halfword

00001011 - MD word

00001100 - MS

00101001 - MD byte - prefetch

00101010 - MD halfword - prefetch

00101011 - MD word - prefetch

01000000 - DSA



11000000 - PSA

11001000 - PD

11010000 - PDA

11011000 - PD in copy mode (rrr contents are lost)

11101000 - PD - prefetch next data

11111111 - PS
```

7.2.5.2.29 LDI (Load Register with Immediate Value)

Operation:

```
GReg[r] ← immediate
```

Assembler:

```
Syntax: ldi r,immediate

Example: ldi 6,1
```

loads decimal value 1 into GReg[6]

CPU Flags: Unaffected

Cycles: 1

Description: Stores a 0-extended immediate value in a General Register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

7.2.5.2.30 LDRPC (Load from RPC to Register)

Operation:

```
GReg[r] ← RPC
```

Assembler:

```
Syntax: ldrpc r

Example: ldrpc 3
```

copies RPC to GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Stores the contents of the RPC in a General Register. That instruction may be used to have more than one level of subroutines.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	1	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.31 LOOP (Hardware Loop)

Operation:

```
if (ff%2 == 0)
   SF ← 0
if (ff/2 == 0)
  DF ← 0
if ((GReg[0] == 0) || (SF == 1) || (DF == 1))
   PC ← PC + loop_size + 1
else
   {
      SPC ← PC + 1
      EPC ← PC + loop_size + 1
      LM ← 1
      PC ← PC + 1
   }
```

during every instruction execution in the loop:

```
if ((SF == 1) || (DF == 1))
 {
```

```
    LM ← 0
    PC ← EPC
  }
else if ((PC + 1) == EPC)
 {
    GReg[0] ← GReg[0] - 1
    if (GReg[0] == 0)
      {
         LM ← 0
         PC ← EPC
       }
     else
         PC ← SPC
   }
else
  PC ← nextPC(instruction)
```

after the execution of the last instruction of the loop body:

```
if (GReg[0] == 0)
   T ← 1
else
   T ← 0
```

Assembler:

```
Syntax: loop n{,ff}
Example: loop 3,1
```

Executes GReg[0] times the instructions comprised between PC+1 and PC+3 (included); ff=1 clears the DF flag before starting the loop. When omitted, the ff field is set to 0 (clearing both SF and DF).

CPU Flags: LM[1:0], T

Cycles: 2 when the loop count (GReg[0]) is 0 or SF or DF is set at loop start, 1+1 when the loop starts but exits abnormally (SF or DF set inside the loop which adds 1 cycle to the offending load or store to jump to EPC), 1 when the loop is executed normally

Description: The loop instruction executes a sequence of instructions several times. The number of times is given by the contents of GReg[0], the loop counter. SDMA will jump to the first instruction after the end of the loop if the value in GReg[0] is 0. Otherwise the SDMA enters loop mode. It sets the most significant bit of the LM flag that will only be reset once the last instruction of the last loop is executed. The instructions in the loop are executed GReg[0] times. The management of fault flags (SF and DF) is as follows. When entering the hardware loop, SF and DF can be cleared according to the ff field of the instruction. After that operation, if any flag is still set the loop will not be executed. The SDMA will jump to the first instruction after the end of the loop without entering loop mode. During the execution of the loop, if any fault flag is set by a LD, LDF, ST, or STF instruction, the SDMA will immediately exit loop mode and jump to the first instruction after the end of the loop. In that case, GReg0 is not decremented for that last piece of the loop body execution (even if the SF or DF flag is set at the last instruction of the loop body). The T flag reflects the state of GReg[0] after the end of the loop, which is an indicator of the complete execution of the loop. If the loop exited because of an error (SF or DF set), GReg[0] will not be 0 at the end of the loop, hence T will be cleared. If the loop executes without fault, GReg[0] will be 0 at the end of the loop, hence T will be set. The boundary case when a source or destination fault occurs at the last instruction of the last loop is considered as an anticipated exit of the loop, which causes the T flag to be cleared. If the last instruction executed before leaving the hardware loop also tries to modify the T flag, the flag is updated according to the value of GReg[0], NOT according to the result of the last executed instruction.

Limitations:

1. 1. Jump instructions (JMP, JMPR, JSR, JSRR, BF, BT, BSF, BDF) are not allowed inside the hardware loop.
2. 2. GReg[0] cannot be written to inside the hardware loop (it can be read).
3. 3. The empty loop (0 instruction in the body) is forbidden.
4. 4. If GReg[0] == 0 at the start of the loop, which causes a jump to EPC, the T flag is not updated.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	0	f	f	n	n	n	n	n	n	n	n

Instruction Fields:

```
ff - flags field:

00 - clear SF and clear DF

01 - clear DF

10 - clear SF
```

```
11 - no clear
```

nnnnnnnn - loop size

```
00000000 - empty loop: forbidden value

00000001 - 1 instruction in the loop

00000010 - 2 instructions in the loop

...

11111111 - 255 instructions in the loop
```

7.2.5.2.32 LSL1 (Logical Shift Left by 1 Bit)

Operation:

```
GReg[r]:{b30,...,b1,b0,0} ← GReg[r]:{b31,b30,...,b1,b0}
```

Assembler:

```
Syntax: lsl1 r

Example: lsl1 2
```

multiplies by 2 the value in GReg[2]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any General Register to the left. The right bit (bit 0) is set to 0. No overflow is detected by the hardware.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	1	1

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]
```

111 - GReg[7]

7.2.5.2.33 LSR1 (Logical Shift Right by 1 Bit)

Operation:

GReg[r]:{0,b31,b30,...,b1} ← GReg[r]:{b31,b30,...,b1,b0}

Assembler:

Syntax: lsr1 r

Example: lsr1 4

divides by 2 the unsigned value contained in GReg[4]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any General Register to the right. The left bit (bit 31) is set to 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	0	1

Instruction Fields:

rrr - destination register field:

000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]

7.2.5.2.34 MOV (Logical Move)

Operation:

GReg[r] ← GReg[s]

Assembler:

```
Syntax: mov r,s

Example: mov 4,0
```

copies GReg[0] to GReg[4]

CPU Flags: Unaffected

Cycles: 1

Description: Move the contents of the source General Register s to the destination General Register r.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	0	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.35 NOTIFY (Notify to ARM platform)

Operation:

```
 if (jjj & 4 == 0)

 {

   if (jjj&2 == 2)

      HE[CCR] ← 0

   if (jjj&1== 1)

       HI[CCR] ← 1

  }

else if (jjj == 4)
```

```
   EP[CCR] ← 0

else
```

(CCR stands for Current Channel Register)

Assembler:

```
Syntax: notify jjj

Example: notify 3
```

clears the HE bit for the current channel and sends an interrupt to the Host for the current channel

CPU Flags: Unaffected

Cycles: 1

Description: Clears one of the channel enabling bits (HE or EP for the corresponding channel number) if required, sends an interrupt to the corresponding ARM platform by setting the appropriate flag if required (HI for the corresponding channel number).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	j	j	j	0	0	0	0	0	0	0	1

jjj - Channel Flags field:

```
000 - unused

001 - set HI for the current channel

010 - clear HE for the current channel

011 - clear HE, set HI for the current channel

100 - clear EP for the current channel

101 - RESERVED

110 - RESERVED

111 - RESERVED
```

7.2.5.2.36 OR (Logical OR)

Operation:

```
GReg[r] ← GReg[s] | GReg[r]
```

Assembler:

```
Syntax: or r,s
```

```
Example: or 3,6
```

ORs GReg[3] and GReg[6] and stores the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the OR of the source General Register s and the destination General Register r, and stores the result in the destination General Register r.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	0	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.37 ORI (Logical OR with Immediate Value)

Operation:

```
GReg[r] ← GReg[r] | immediate
```

Assembler:

```
Syntax: ori r,immediate

Example: ori 1,56
```

ORs GReg[1] and the decimal value 56 and stores the result in GReg[1]

CPU Flags: unaffected

Cycles: 1

Description: Performs an OR between a 0-extended 8-bit immediate value and a General Register; stores the result in the General Register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

7.2.5.2.38 RET (Return from Subroutine)

Operation:

```
PC ← RPC
```

Assembler:

```
Syntax: ret
```

CPU Flags: Unaffected

Cycles: 2

Description: Return from subroutine.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

7.2.5.2.39 REVB (Reverse Byte Order)

Operation:

```
GReg[r]:{B3,B2,B1,B0} ← GReg[r]:{B0,B1,B2,B3}
```

Assembler:

```
Syntax: revb r

Example:  revb 5
```

reverses bytes order in GReg[5]

CPU Flags: Unaffected

Cycles: 1

Description: Reverse the byte order of any General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	0	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.40 Reverse Low Order Bytes(REVBLO)

Operation:

```
GReg[r]:{B3,B2,B0,B1} ← GReg[r]:{B3,B2,B1,B0}
```

Assembler:

```
Syntax: revblo r

Example: revblo 0
```

reverses low order bytes in GReg[0]

CPU Flags: Unaffected

Cycles: 1

Description: Reverse both low order bytes of any General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	0	1

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.41 ROR1 (Rotate Right by 1 Bit)

Operation:

```
GReg[r]:{b0,b31,b30,...,b1} ← GReg[r]:{b31,b30,...,b1,b0}
```

Assembler:

```
Syntax: ror1 r

Example: ror1 3
```

rotates bits to the right in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Rotate the bits of any General Register to the right.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	0	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.42 RORB (Rotate Right by 1 Byte)

Operation:

```
GReg[r]:{B0,B3,B2,B1} ← GReg[r]:{B3,B2,B1,B0}
```

Assembler:

```
Syntax: rorb r

Example: rorb 2
```

rotates bytes to the right in GReg[2]

CPU Flags: Unaffected

Cycles: 1

Description: Rotate the bytes of any General Register to the right.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	1	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.43 SOFTBKPT (Software Breakpoint)

Operation:

Stops the current script and enters debug mode

Assembler:

```
softbkpt
```

CPU Flags: Unaffected

Description: When the core executes this instruction, it has the same effect as receiving a debug request from the OnCE or via the external debug request input: the script execution halts, the PCU enters its debug state and waits for the OnCE commands that are described in OnCE and Real-Time Debug.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

7.2.5.2.44 ST (Store Register)

Operation:

```
[GReg[b] + displacement] ← GReg[r]
```

```
if (transfer_error)

 DF ← 1

 else

DF ← 0
```

Assembler:

```
Syntax: st r,(b,displacement)

Example: st 7,(0,9)
```

stores the value from GReg[7] into memory at address obtained by adding decimal value 9 to GReg[0]

CPU Flags: DF

Cycles: 2+n where n is 0 for ROM, RAM or memory mapped registers, and n is the number of wait-states of the peripheral for a peripheral access

Description: Adds a 5-bit 0-extended displacement to a base address in General Register b; the result is the address of the data to store on the DM bus. The data sent on the bus comes from the source General Register r. If an error occurs during the transfer, the flag DF is set, else it is cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	r	r	r	d	d	d	d	d	b	b	b

Instruction Fields:

rrr / bbb - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

ddddd - displacement value:

```
00000 - 0
```

```
00001 - 1

...

11111 - 31
```

7.2.5.2.45 STF (Store Register in Functional Unit)

Operation:

```
[fu_address] ← GReg[r] 0

if (transfer_error) 0

DF ← 1 0

else 0

DF ← 0
```

fu_address is an 8-bit field

Assembler:

```
Syntax: stf r,fu_address

Example: stf 3,0x2B
```

stores the 32-bit contents of GReg[3] to the Burst DMA register MD; waits until the flush to external memory is completed

CPU Flags: DF

Cycles: 1+n where n is the number of wait-states that may be inserted by the functional unit

Description: Sends an 8-bit address on the Functional Unit Bus (FU bus) and sends the contents of the source General Register r on the bus. If an error occurs during the transfer, the flag DF is set, else it is cleared.

Table 7-49. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	r	r	r	f	f	f	f	f	f	f	f

See the following sections for more details of the STF instruction usage with each functional unit:

- Burst DMA Write (stf) for Burst DMA
- Peripheral DMA Write (stf)-Write Mode for Peripheral DMA

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

ffffffff - functional unit destination register and action (unspecified values are reserved):

00000000 - MSA in incremented mode

00000100 - MDA in incremented mode

00001001 - MD byte

00001010 - MD halfword

00001011 - MD word

00001100 - clear MS error flag

00001111 - MS

00010000 - MSA in frozen mode

00010100 - MDA in frozen mode

00011000 - MD in copy mode - number of words in rrr

00100000 - MSA in incremented mode - start prefetch

00101000 - MD no data - flush

00101001 - MD byte - flush

00101010 - MD halfword - flush

00101011 - MD word - flush

00110000 - MSA in frozen mode - start prefetch

11000001 - PSA in frozen mode - 8-bit data width

11000010 - PSA in frozen mode - 16-bit data width

11000011 - PSA in frozen mode - 32-bit data width

11000101 - PSA in incremented mode - 8-bit data width

11000110 - PSA in incremented mode - 16-bit data width

11000111 - PSA in incremented mode - 32-bit data width

11001000 - PD

11001001 - PSA in decremented mode - 8-bit data width

11001010 - PSA in decremented mode - 16-bit data width

11001011 - PSA in decremented mode - 32-bit data width

11001100 - clear PS error flag

11001101 - PSA data width becomes 8-bit

11001110 - PSA data width becomes 16-bit

11001111 - PSA data width becomes 32-bit

11010001 - PDA in frozen mode - 8-bit data width

11010010 - PDA in frozen mode - 16-bit data width

11010011 - PDA in frozen mode - 32-bit data width

11010101 - PDA in incremented mode - 8-bit data width

11010110 - PDA in incremented mode - 16-bit data width

11010111 - PDA in incremented mode - 32-bit data width

11011001 - PDA in decremented mode - 8-bit data width

11011010 - PDA in decremented mode - 16-bit data width

11011011 - PDA in decremented mode - 32-bit data width

11011101 - PDA data width becomes 8-bit

11011110 - PDA data width becomes 16-bit

11011111 - PDA data width becomes 32-bit

11100001 - PSA in frozen mode - 8-bit data width - prefetch data

11100010 - PSA in frozen mode - 16-bit data width - prefetch data

11100011 - PSA in frozen mode - 32-bit data width - prefetch data

11100101 - PSA in incremented mode - 8-bit data width - prefetch data

11100110 - PSA in incremented mode - 16-bit data width - prefetch data

11100111 - PSA in incremented mode - 32-bit data width - prefetch data

11101001 - PSA in decremented mode - 8-bit data width - prefetch data

11101010 - PSA in decremented mode - 16-bit data width - prefetch data

11101011 - PSA in decremented mode - 32-bit data width - prefetch data

11101101 - PSA data width becomes 8-bit - prefetch data

11101110 - PSA data width becomes 16-bit - prefetch data

11101111 - PSA data width becomes 32-bit - prefetch data

11111111- PS

7.2.5.2.46 SUB (Subtract)

Operation:

```
GReg[r] ← GReg[r] - GReg[s]

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: sub r,s

Example: sub 4,7
```

SUBtracts GReg[7] from GReg[4] and stores the result in GReg[4]

CPU Flags: T

Cycles: 1

Description: Subtracts the source General Register s from the destination General Register r, and stores the result in the destination General Register r. The T flag is set if the result of the operation is 0; it is cleared if the result is not 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	0	0	s	s	s

Instruction Fields:

rrr / sss - register fields:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.47 SUBI (Subtract with Immediate)

Operation:

```
GReg[r] ← GReg[r] - immediate

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: sub r,immediate

Example: sub 1,255
```

SUBtracts decimal value 255 from GReg[1] and stores the result in GReg[1]

CPU Flags: T

Cycles: 1

Description: Subtracts a 0-extended 8-bit immediate value from a General Register; stores the result in the General Register. The flag T is set when the result of the operation is 0; otherwise, it is cleared. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

7.2.5.2.48 TST (Test with Zero)

Operation:

```
T ← ((GReg[s] & GReg[r]) != 0)
```

Assembler:

```
Syntax: tst r,s

Example: tst 2,3
```

ANDs GReg[2] and GReg[3] and sets T if the result is non-null

CPU Flags: T

Cycles: 1

Description: Performs the AND of the source General Register s and the destination General Register r, and sets T if the result is not 0, clears T if the result is 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	0	0	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.49 TSTI (Test Immediate)

Operation:

```
T ← ((GReg[r] & immediate) != 0)
```

Assembler:

```
Syntax: tsti r,immediate

Example: tsti 5,13
```

ANDs GReg[5] and decimal value 13 and sets T if the result is non-null

CPU Flags: T

Cycles: 1

Description: Performs the AND of a 0-extended 8-bit immediate value and the destination General Register r, and sets T if the result is not 0, clears T if the result is 0. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - destination register field:

```
000 - GReg[0]

001 - GReg[1]
```

```
010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

7.2.5.2.50 XOR (Logical Exclusive OR)

Operation:

```
GReg[r] ← GReg[s] ^ GReg[r]
```

Assembler:

```
Syntax: xor r,s

Example: xor 0,3
```

XORs GReg[0] and GReg[3] and stores the result in GReg[0]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the eXclusive OR of the source General Register s and the destination General Register r, and stores the result in the destination General Register r.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	1	0	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]
```

```
010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

7.2.5.2.51 XORI (Exclusive OR with Immediate)

Operation:

```
GReg[r] ← GReg[r] ^ immediate
```

Assembler:

```
Syntax: xori r,immediate

Example: xor 7,5
```

XORs GReg[5] and decimal value 5 and stores the result in GReg[7]

CPU Flags: Unaffected

Cycles: 1

Description: Performs an eXclusive OR between a 0-extended 8-bit immediate value and a General Register; stores the result in the General Register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]
```

```
111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

7.2.5.2.52 YIELD, YIELDGE (DONE, Yield)

By default, unsupported assembler syntax. Can be aliased to the corresponding done instructions (yield = done 0; yieldge = done 1). Refer to the done instruction description DONE (DONE, Yield) .

7.2.6 Software Restrictions

7.2.6.1 Unsupported Burst DMA Access Sequence

The SDMA does not support triggering a pre-fetch followed by a flush of the Burst DMA without reading or writing any data. If the flush occurs while the background pre-fetch DMA operation is still in progress, it could result in un-defined behavior.

An example of the sequence which could result in undefined results is shown in the following example:

Instruction sequence not supported

```
stf r1, MSA|PF          ; Update source address, triggers data pre-fetch in the
                        ; background
mov R0,R0               ; Execute multiple assembly instructions, none of which
                        ; read
mov R0,R0               ; or write data to/from MD
stf MD|SZ0|FL           ; Flush FIFO without writing data. If the pre-fetch is still
                        ; in progress when this instruction is executed, there
                        ; could be undefined operation
```

A work-around to avoid any undesirable results is to first read MD to ensure the pre-fetch is complete before the flush is attempted.

Work-Around to previous example

```
stf r1, MSA|PF          ; Update source address, triggers data pre-fetch.
```

```
mov R0,R0              ; Execute multiple assembly instructions, none of which
                       ; read
mov R0,R0              ; or write data to/from MD
ldf r2, MD             ; dummy read of MD to ensure pre-fetch is complete
                       ; before the next instruction
stf MD|SZ0|FL          ; Flush FIFO without writing data
```

7.2.7 Application Notes

7.2.7.1 Data Structures for Boot Code and Channel Scripts

SDMA boot code downloads the different channel contexts and the scripts that will be executed on SDMA channels during the application.

The boot code is run after reset when channel 0 is started by the ARM platform. The boot code is also known as channel 0 script.

The boot code is based on the Channel Control Block (CCB) and Buffer Descriptor (BD) mechanisms that are data structures located into the ARM platform memory space. With these data structures, it is possible to instruct SDMA to download scripts and contexts but also to dump a context or a script to a destination data buffer. Channel scripts also use the CCB and BD data structures to pass instructions and/or pointers to data to be copied.

The format, processing, and field definition of the CCB and BD are defined and performed entirely by the software script rather than the SDMA hardware. An overview of the format and structure is provided here, but for complete details refer to the SDMA software documentation (see SDMA Scripts).

The CCB and BD data structures are accessed by SDMA using DMA and processed by the SDMA scripts. The ROM contains common sub-routines for processing these data structures which may be called by the bootload and channel scripts.

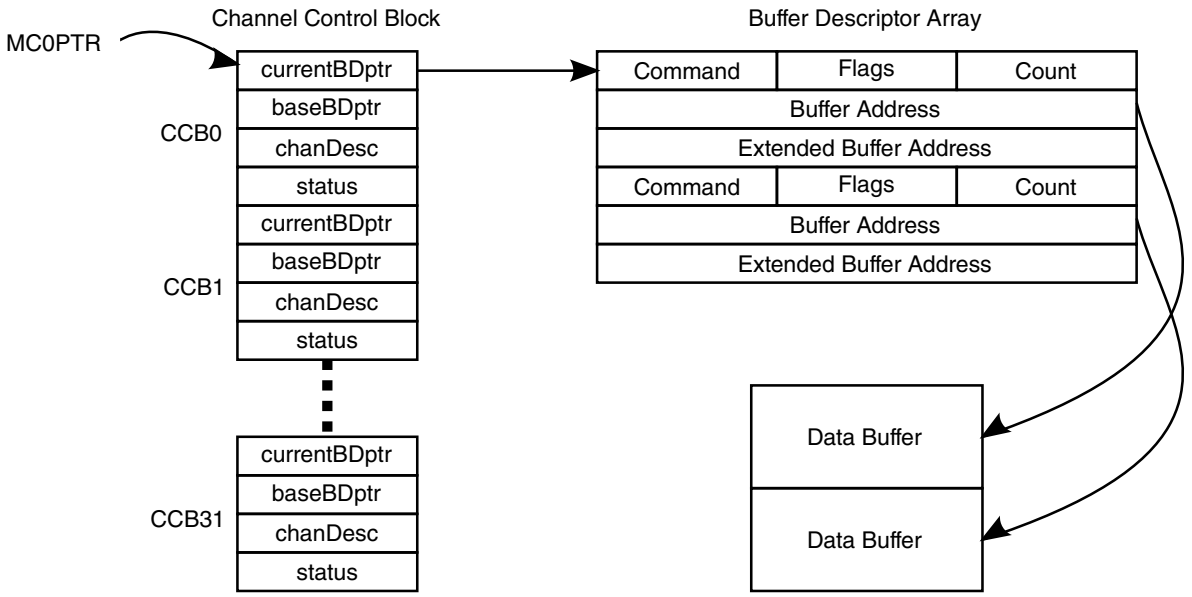


Figure 7-14. Data Structures Layout

The previous figure shows an example how these data structures are linked to pass command and pointers to data buffers. The SDMA's MC0PTR register holds the base address of the Channel 0 Control Block (CCB0). The Channel 0 control block holds a pointer to the array of buffer descriptors. The buffer descriptors are used to tell the channel 0 (boot channel) what to do as described Buffer Descriptor Format.

7.2.7.1.1 Buffer Descriptor Format

Buffer descriptors are three longs (32-bit words) in size as, shown in the figure found here.

A buffer descriptor describes the properties of the data buffer it points to. The buffer descriptors can be used for linear or circular data buffers in the ARM platform processor memory. The CCB contains a pointer to the base BD as well as the current BD.

Table 7-50. Buffer Descriptor

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Command								-	-	L	R	I	C	W	D	Count															
Buffer Address																															
Extended Buffer Address																															

Table 7-51. Buffer Descriptor Field Descriptions

Field	Description
31-24 Command	Command. The command field is used to differentiate operations performed within a script when the script accesses this particular buffer descriptor. The use of this field can be defined by the script. The command values defined for the bootload script are defined in Buffer Descriptor Commands for Bootload scripts. Refer to the individual script definition in script library documents in SDMA Scripts for command field definitions for other scripts.
23	Reserved
22	Reserved
21 L	Last Buffer Descriptor: This bit is set in SDMA IPC scripts to indicate to the receiving Core that the transfer has ended. Whenever the source finishes transferring the count it wanted to transfer, it sets LAST_BIT in the destination BD, to let the destination know that transfer is over.This bit also tells the destination software that when it processes the destination BDs, they need not process any BD after the BD with the LAST_BIT set.For example, when the DSP prepares a single buffer descriptor with count equals to 25 and ARM platform prepares a single buffer descriptor with count equals 100. When 25 bytes have been transferred from DSP to ARM platform, the DSP buffer descriptor is normally closed while the ARM platform buffer descriptor will have the L bit set and the byte count updated to 25.
20 R	erroR. Indicates an error occurred on the channel's buffer descriptor requested command. Some scripts may overwrite the command field with an error code indicating the source of the error. 0 No Error 1 Error
19 I	Interrupt. When SDMA has finished to process data transfer attached to this buffer descriptor, send an interrupt to the ARM platform. 0 No Interrupt 1 Interrupt the processor when BD is complete
18 C	Continuous. This buffer is allowed to receive multiple transmit buffers or is allowed to transmit to multiple receive buffers.The Continuous bit is decoded at the end of the processing of a BD to determine if the SDMA script must open a new BD to potentially continue the data transfer. 0 No further buffer descriptors 1 SDMA should move to the next Buffer descriptor after this one
17 W	Wrap. Indicates if this buffer descriptor is the last one for the channel control block. When encountering this bit set, the SDMA scripts updates the CurrentBD pointer to point to the first Buffer Descriptor of the array. This bit is set if the ARM platform wants to organize the array of BD in a circular way (like a ring). When all BD have been processed and if Wrap bit and CONtinuous bit are set in the last BD, the SDMA script will wrap around and it will try to re-open the first BD. 0 No Error 1 Wrap to first buffer descriptor after this one is processed.
16 D	D - "Done": bit 16: indicates the "ownership" of the buffer descriptor. When D=0 the host owns the buffer descriptor; when D=1 SDMA owns the buffer descriptor. In the case of the channel 0, D=1 indicates the SDMA has not yet processed this buffer, D=0 indicates the SDMA has processed this buffer. 0 ARM platform owns the buffer. 1 SDMA owns the buffer
15-0 Count	Count. the count field (bit 15-0) indicates the size of the data to be transmitted, the size of the data buffer pointed to by the buffer descriptor. The SDMA memory structure is different for program memory (16-bits shorts/half-words) and data memory (32-bits long). For channel 0 buffer descriptors, Count is expressed in 16-bit half-words when PM is addressed and in 32-bit words when DM is addressed. Count is typically expressed in bytes for other channel scripts, but the unit is dependant on the script.
31-0	Buffer address. Address pointer to the data buffer.
31-0	Extended buffer address. Additional pointer or other information required by some scripts.

The buffer descriptors form an array of programmable size. If the last buffer descriptor is marked by the Wrap flag-bit W=1, the array of buffer descriptor is treated as a ring with some logically continuous portion owned by the ARM platform with D=0, and the remainder owned by the SDMA with D=1. The count field of the buffer descriptor indicates how much data has been transmitted.

If ARM platform has prepared 3 buffers to be filled by the SDMA script, it has also prepared 3 BD, one for each buffer. The *Cont* and *Wrap* bits are used to organize the buffers in a circular way. For example, *CONT*inous bit is set to 1 in the 2 first BDs and *Wrap* is set in the 3rd BD. The SDMA script opens and processes BD#1. Since *CONT*inous bit is set for this BD, the SDMA will open the second BD and it will process it. Each time a BD is processed, its *Done* bit is reset by the SDMA. After the 3rd BD, if *CONT*inous is not set but if *Wrap* is set, the SDMA script stops here and the next time the channel will be triggered, the script will open the BD pointed by the currentBDptr pointer of the CCB and it will correspond to the first buffer descriptor.

If the *CONT*inous bit and *Wrap* bits are both set in the 3rd BD, the script will close it and it will try to open the first BD. An error may occur at this point if the BD#1 has already been processed and its *Done* bit is 0. The SDMA script cannot process a BD with a *Done* bit to 0. It means the BD is not ready to be processed. To avoid this situation, the *CONT*inous bit should not be set for the last BD if *Wrap* is set, and the Interrupt flag must set for the last BD. It will warn the owner of the BD that all the BDs have been processed and it has to re-set to 1 the *Done* bit of all the BD's if it desires the SDMA to fill them again. Basically, if the ARM platform expects the SDMA to fill up the buffers in a circular fashion, then it's the responsibility of the ARM platform to set the *Done* bit of a buffer descriptor at an appropriate time.

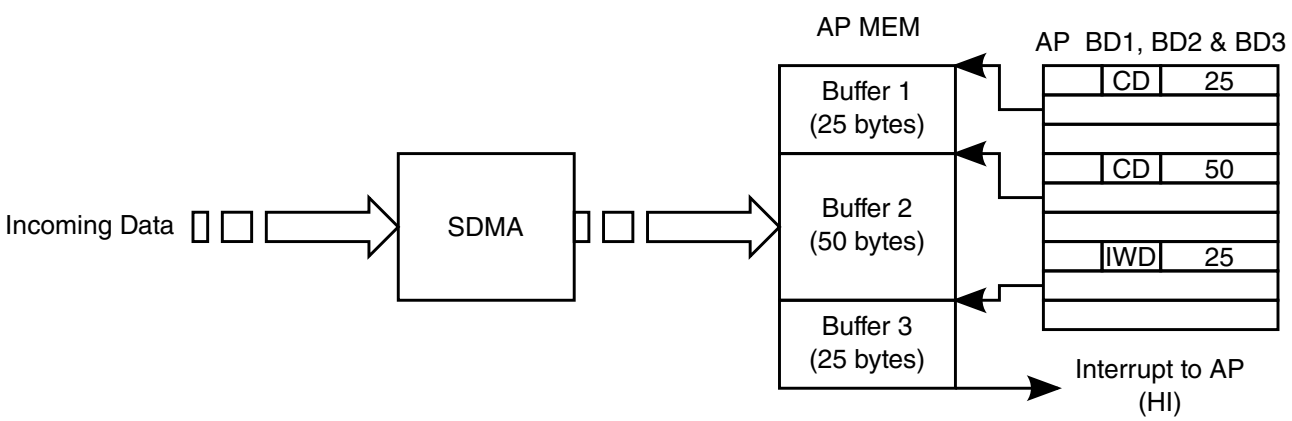


Figure 7-15. Buffer Descriptor Flow

The previous figure shows an example buffer descriptor flow. When the incoming data is stored and fills the first buffer of 25 bytes, the SDMA script opens the second BD because the CONTinuous bit was set. Then next incoming data is put in the second buffer. After receiving 50 bytes, the second buffer descriptor is also closed. The Done bit is reset and the third BD is opened. After receiving another 25 bytes, the third buffer is full and an interrupt is sent to the ARM platform because the Interrupt flag is set in the 3rd BD. The CONTinuous flag is not present the transfer is over. The next time the script will be triggered, the BD to be opened will be the first buffer descriptor since the Wrap flag was set in the 3rd BD. It is the ARM platform responsibility to set the Done bit of all the BD if it wants to use the same buffers.

7.2.7.1.2 Buffer Descriptor Commands for Bootload scripts

The command field of the buffer descriptor is defined separately for each script.

The following table lists the buffer descriptor commands defined for the channel 0 bootload script.

Table 7-52. Channel Zero Buffer Descriptor Commands

Command Field (binary)	Command	Description	Buffer Address	Extended Buffer Address
0000_0001 (0x01)	C0_SET_DM	Load SDMA data memory (RAM) from ARM platform memory buffer	ARM platform memory source address	SDMA memory destination address
0000_0010 (0x02)	C0_GET_DM	Copy SDMA data memory (RAM) to ARM platform memory buffer	ARM platform memory destination address	SDMA memory source address
0000_0100 (0x04)	C0_SET_PM	Load SDMA program memory (RAM) from ARM platform memory buffer	ARM platform memory source address	SDMA memory destination address
0000_0110 (0x06)	C0_GET_PM	Copy SDMA program memory (RAM) to ARM platform memory buffer	ARM platform memory destination address	SDMA memory source address
cccc_c111 (0x07 | CHN)	C0_SETCTX	Load Context for channel cccc into SDMA RAM from ARM platform memory buffer	ARM Platform memory source address	-
cccc_c011 (0x03 | CHN)	C0_GETCTXT	Copy Context for channel ccccc from SDMA RAM to ARM platform memory buffer	ARM platform memory destination address	-

The Channel 0 bootload commands are summarized as follows:

- C0_SET_[PM-DM]: load the buffer descriptor data in the SDMA local memory at the address pointed to by the "extended buffer address" field. The SDMA RAM can be seen as a Program Memory (PM, 16-bit address) or Data Memory (DM 32-bit address). When C0_SET_PM is used, the count field is expressed in "shorts" (16-bit half words), this command can be used to download scripts. When C0_SET_DM is

used, the count field is expressed in "long" (32-bit words), this command can be used to download channel contexts to the context channel area in RAM.

- C0_GET_[PM-DM]: write to the buffer descriptor's data buffer the content of the SDMA local memory from the address pointed to by the "extended buffer address" field for the length defined by the count in the buffer descriptor. C0_GET_PM is used to dump some part of the Program Memory (may be used to dump context of a channel), therefore count is expressed in "shorts"; while C0_GET_DM is used to dump to the buffer descriptor's data buffer, so the count field is in "longs."
- C0_SETCTX: load a context into the SDMA context page area. The handling script decodes the channel number from the 5 MSB of the command field of the buffer descriptor. Using the channel number the script computes the offset of the context data pointer for the channel relative to the context page base to use as the destination address in SDMA memory. Then the C0_SET_DM command explained above is invoked to load SDMA RAM from memory. The counter indicates the size in words of the context structure.
- Command value: (in binary) cccc c111, where ccccc is the channel number (5 bits). For instance, 0x0F means set context for channel 1, 0xFF means set context for channel 31.

- C0_GETCTX: write to the buffer descriptor's data buffer the content of the SDMA context page area. The handling script decodes the channel number from the 5 MSB of the command field of the buffer descriptor. Using this channel number, the script computes the offset of the context data pointer for the channel relative to the context page base to use as the source address for the copy. Then the C0_GET_DM command explained above is invoked to copy the context to memory. The counter indicates the size in words of the context structure.
- Command value: (in binary): cccc c011, where ccccc is the channel number (5 bits). For instance, 0x03 means get context of channel 1, 0xFB means get context of channel 31.

NOTE

To download channel context, C0_SETDM and C0_SETCTXT command can be used but the second one is easier because the channel number is embedded into the command field, whereas with the C0_SETDM, the pointer to the channel context area must be written into the extended buffer address field of the buffer descriptor.

7.2.7.1.3 Example of Buffer Descriptors for Channel 0.

Figure 7-17 illustrates the buffer descriptors that must be set in ARM platform memory space, before execution of boot code, to download contexts and scripts of channels 1, 4, and 10. After boot code execution, SDMA memory will be populated with the different contexts and scripts as presented in the following figure.

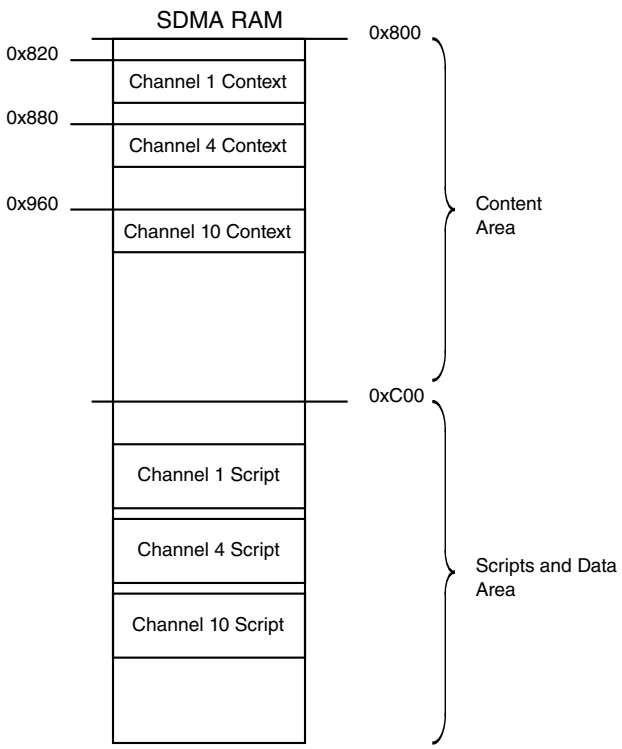
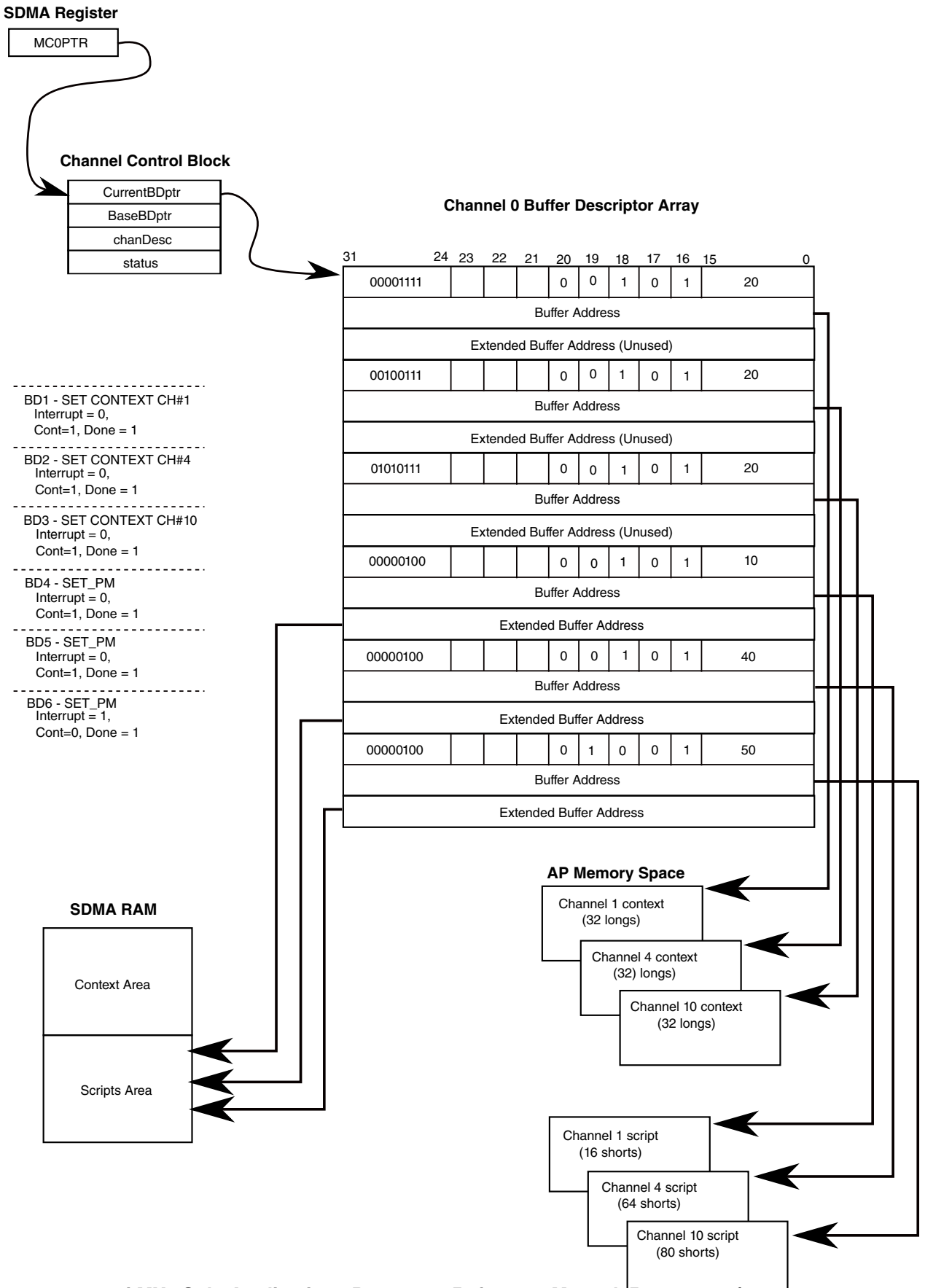


Figure 7-16. Example of SDMA RAM After Boot Session


SDMA Register
MC0PTR
Channel Control Block
CurrentBDptr
BaseBDptr
chanDesc
status
Channel 0 Buffer Descriptor Array
31 24 23 22 21 20 19 18 17 16 15 0
00001111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
00100111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
01010111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
00000100 0 0 1 0 1 10
Buffer Address
Extended Buffer Address
00000100 0 0 1 0 1 40
Buffer Address
Extended Buffer Address
00000100 0 1 0 0 1 50
Buffer Address
Extended Buffer Address
BD1 - SET CONTEXT CH#1
Interrupt = 0,
Cont=1, Done = 1
BD2 - SET CONTEXT CH#4
Interrupt = 0,
Cont=1, Done = 1
BD3 - SET CONTEXT CH#10
Interrupt = 0,
Cont=1, Done = 1
BD4 - SET_PM
Interrupt = 0,
Cont=1, Done = 1
BD5 - SET_PM
Interrupt = 0,
Cont=1, Done = 1
BD6 - SET_PM
Interrupt = 1,
Cont=0, Done = 1
AP Memory Space
Channel 1 context
(32 longs)
Channel 4 context
(32) longs)
Channel 10 context
(32 longs)
SDMA RAM
Context Area
Scripts Area
Channel 1 script
(16 shorts)
Channel 4 script
(64 shorts)
Channel 10 script
(80 shorts)

7.2.7.1.4 Channel Context

There are 32 channel context memory structures pointed to by the local save area pointer. These channel context memory structures are fixed.

The script in the SDMA computes the memory offset for a given channel based on the structure length and channel number. Figure below shows the structure of the channel context as it is saved in the SDMA local memory (RAM).

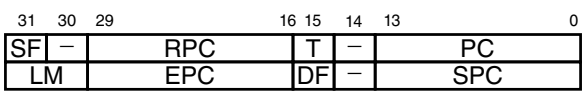
A channel context consists in 24 words, one per register. A total of 32 words are reserved for every channel. The additional 8 words are called scratch ram and they are dedicated to each channel. This memory area is commonly used for stack management.

The structure is divided in 4 areas:

- Channel status registers
- General purpose registers
- Functional units state registers reflecting the state of the ARM platform DMAs (Burst and Peripheral DMA).
- Scratch RAM

The details of the channel context status registers are described in the following figure.

The PC field of the first long register must point to the SDMA RAM address where the script that will be executed on the channel is located and this value equals the one stored in the extended buffer address of the buffer descriptor with C0_SETPM command.



SF: Source fault while loading data
RPC: Return program counter
T: Test bit: status of arithmetic and test instructions
PC: Program counter
LM: Loop mode
EPC: Loop end program counter
DF: Destination fault while storing data
SPC: Loop Start program counter

Figure 7-18. SDMA State Registers (ShPC, ShLoop)

7.2.7.2 Typical Data Transfer Supported by SDMA DMA Units

This section presents a library of SDMA scripts that perform data transfers through the peripheral DMA and the burst DMA units.

The ARM platform memory and peripherals are devices that either the peripheral DMA or the burst DMA can access. The scripts are given for a peripheral DMA whose address registers are programmed in incremented mode when internal memory is involved. See the following table for the summary.

Table 7-53. Typical Data Transfers Summary

Data Transfer	Peripheral DMA	Burst DMA	Comments
ARM platform External Memory ↔ ARM platform External Memory		3	Copy mode Script example, see Burst DMA Unit Copy Mode and External Memory to External Memory.
ARM platform Peripheral ↔ ARM platform Peripheral	3		Copy mode if same data path width Script example, see Peripheral to Peripheral Transfer.
ARM platform External Memory ↔ ARM platform Peripheral	3	3	Data transit through SDMA Script example, see Transfer Between Peripheral and External Memory.
ARM platform External Memory ↔ ARM platform Internal Memory		3	Copy mode Script example, see Transfer Between External Memory and Internal Memory.
ARM platform Internal Memory ↔ ARM platform Internal Memory		3	Copy mode Script example, see Internal Memory to Internal Memory.
ARM platform Internal memory ↔ ARM platform Peripheral	3		Data transit through SDMA Script example, see Transfer Between Peripheral and Internal Memory.

NOTE

These scripts are provided as examples of how to use DMA blocks to perform required data transfers: They are not "official" programs.

7.2.7.2.1 External Memory to External Memory

This section describes the SDMA script that performs data moves in external memory.

For this particular data transfer, only the burst DMA is used. It is programmed in copy mode, so no data transmits through an SDMA general register.

The SDMA core only monitors data transfer status. It is assumed source and destination address values are already present in two SDMA general registers (r1 and r2). For this example, it is also assumed that a 32-bit word-to-move for source-to-destination address is present in r0 and equals 64.

Data Moves in External Memory

```
1         stf r1,MSA                          // Source address setup

2         stf r2,MDA                          // Destination address setup

3         ldi r0,0x64                         // 64 words must be transferred from MSA to
MDA

4         ldi r1,0x8

MAIN_XFER:

5         cmphs r0,r1                         // Is r0 >= 0x8

6         bf LAST_XFER                      // If not, jump to last transfer label

7         stf r1,MD|CPY                        // Copy 8 words from MSA to MDA address.

8         subi r0,0x8                         // Decrement counter

9         jmp MAIN_XFER                     // return to main transfer loop

LAST_XFER:

10        stf r0,MD|CPY                        // perform last transfer
```

All instructions are performed in one cycle (jumps excepted). Instruction 7 triggers a copy transfer: A read burst access of 8-word starts, data is staged in MD and then a write burst of 8 words is executed. Instruction 8, 9, 5, and 6 are executed while the burst access is in progress. If this access is not complete when instruction 7 is executed a second time, SDMA stalls on this instruction as long as the previous copy transfer is not over. In this case, the instruction is no longer a one-cycle instruction.

During the main loop (MAIN_XFER), r1 always equals 8, so burst lengths are 8 words. On the last ldf |CPY instruction (10), r1 equals the reminder of r0 divided by 8; therefore, the length of bursts triggered in copy mode equal r1 value, which is between 1 and 7.

7.2.7.2.2 Peripheral to Peripheral Transfer

For this data transfer, only the peripheral DMA is used.

It is programmed in copy mode, so no data will transmit through the SDMA general register used in the ldf instruction, but the contents of the general register are lost. The SDMA core only monitors the transfer.

7.2.7.2.2.1 Source and Destination Target Have the Same Data Path Width

When the source and destination target have the same data path width, the following is true:

- Source target is a *half-word* (16-bit) peripheral located at address 0x1002.
- Destination is a *half-word* (16-bit) peripheral located at address 0x2006.

It is assumed the address values are already present in two SDMA general registers (r1, r2). The script for a transfer of 10 half-word is as follows:

Same Data Path Width for Source and Destination

```
//SETUP SECTION
1         stf r1, PSA|SZ16|F                    //r1=0x1002 Source address register setup
2         stf r2, PDA|SZ16|F                    //r2=0x2006 Destination address register
setup
3         bdf ERROR_ADDR_SETUP
4         ldi r0,0xa                           //loop counter is 10
//MAIN LOOP TRANFER
copy_loop:

5         loop 2,0
6           ldf r7,PD|CPY                      //Reads 1 half-word from src and writes to
dest.
7           yield
8         bdf ERROR_DURING_XFER
ERROR_ADDR_SETUP:
                    //correction of PSA/PDA setup and jumps to main loop transfer
ERROR_DURING_XFER:
//flag error is set,
//PS can be read to know if error occurs during read or write access.
```

If a data transfer must occur between two word peripherals, only the setup section should be updated. The transfer itself is always performed by the hardware loop instruction.

All instructions are executed in one cycle (change of flow excepted). On instruction 6, a single read access is triggered, read data is staged in PD, and a write-to-destination is executed. When the transfers are in progress, the SDMA can execute he next instructions in parallel. If instruction 6, which performs the copy transfer, is executed while the previous access is not over, SDMA is stalled and instruction ldf is a multi-cycle instruction.

7.2.7.2.2.2 Source and Destination Target Have a Different Data Path Width

When the source and destination target have a different data path width, copy mode cannot be used, and any attempt to initiate a copy transfer immediately raises an error, which is stored in the SF flag.

The following example shows the SDMA code that could transfer 10 words from a *word* (32-bit) peripheral to a *half-word* peripheral whose addresses are preliminary and stored in r1 and r2.

Different Data Path Width for Source and Destination

```
//SETUP SECTION
1         stf r1, PSA|SZ32|F|PF                    //r1=0x1000 and prefetch data
2         stf r2, PDA|SZ16|F                      //r2=0x2006
3         bdf ERROR_ADDR_SETUP
4         ldi r0,0xa                             //loop counter is 10
//MAIN LOOP TRANFER
main_loop_xfer_16_16:
5         loop 6,0
6           ldf r7,PD                           //copy 32-bit of PD in r7
7           stf r7,PD                            //store 16 LSB of r7 in PD and a flush is
executed
8           rorb r7
9           rorb r7                            //16 MSB --> 16 LSB
10          stf r7,PD                            //store 16 LSB of r6 in PD and a flush.
11          yield
```

On instruction 1, when the source address register is programmed and a data prefetch is required, a read access is executed. In parallel, the SDMA executes instructions 2 to 5. On instruction 6, the SDMA tries to read data that was fetched by instruction 1. If data is ready, the ldf will be a one cycle instruction; otherwise, the SDMA is stalled as long as the read access is not finished. Then, the 16 LSB of the read data is stored in PD and automatically flushed to the destination peripheral. In parallel, the SDMA executes the rotation instructions (8, 9), and stores the 16 MSB of the read data into PD. If a previous write access is finished, instruction 10 will be a one-cycle instruction.

The main loop transfer may appear inefficient, but due to wait states imposed to the peripheral DMA each time an external access is performed, a software pipeline is in place. During the time needed to flush PD, the SDMA executes the move and rotation operations. SDMA executes instructions in parallel with DMA accesses.

7.2.7.2.3 Transfer Between Peripheral and External Memory

7.2.7.2.3.1 Peripheral to External Memory Transfer

A transfer from a peripheral to the external memory controller involves the peripheral DMA and the burst DMA.

The code for transferring 100 word from word peripheral to the external memory would be as follows:

Peripheral to External Memory Transfer

```
//SETUP SECTION source and destination addresses are already in r1 and r2
1         stf r1, PSA|SZ16|F|PF                    //r1=0x1000 and prefetch 32-bit data
2         stf r2, MDA                            //r2=0x2000, setup burst DMA destination
address
3         bdf ERROR_ADDR_SETUP
4         ldi r0,0x64                            //loop counter is 100
5
          //MAIN LOOP TRANFER
6         loop 3,0
7           ldf r1,PD|PF                         // read 32 bits of PD and initiate a new read
```

```
access.
8           stf r1,MD|32                    // store 32 bits of r1 in the MD fifo.
9           yield
10        ldf r1,PD                       // last word data is read
11        stf r1,MD|32|FL                   // to flush all remaining bytes of MD
```

On instruction 1, the source address register of the peripheral DMA is programmed and data is fetched. This data is stored in PD and the SDMA reads PD during instruction 7, which is a one-cycle instruction that is read-access finished. On the same instruction (7), a data prefetch is required and a read access to the source peripheral is executed. In parallel, the SDMA stored the previous read data into the data register of MD. When MD (which is an eight-word FIFO) is full, a burst write access is executed to empty the FIFO. As long as the next SDMA instructions do not access the burst DMA, they will be one-cycle instructions. The following figures show how the peripheral DMA and burst DMA work in parallel.

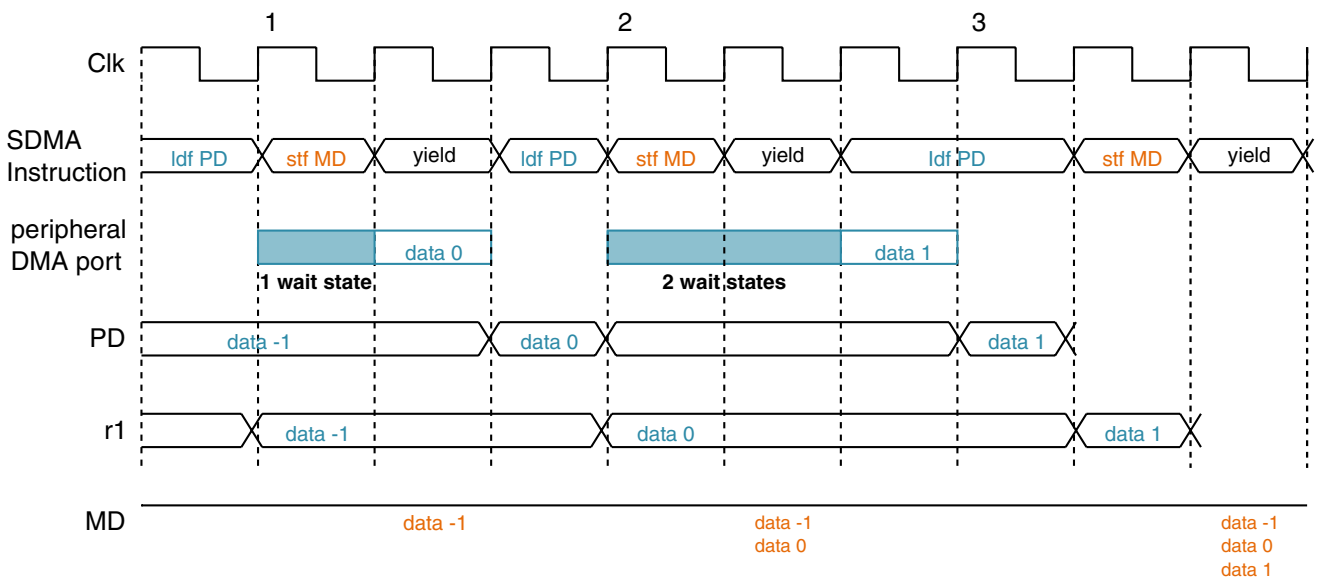


Figure 7-19. Peripheral to External Memory Example (1)

As seen in the figure above, the read access triggered by the ldf PD instruction is symbolized by the blue bar when in progress. After wait states, the read data (data 0, data 1) is stored in PD on the clk rising edge. On edge 2, data 0 is available in PD so it can be transferred to the SDMA general register r1, and then stored in MD FIFO. On edge 3, data 1 is not in PD; therefore, SDMA is stalled on the ldf instruction, which lasts two cycles. The figure below shows an example of when MD FIFO is full with data.

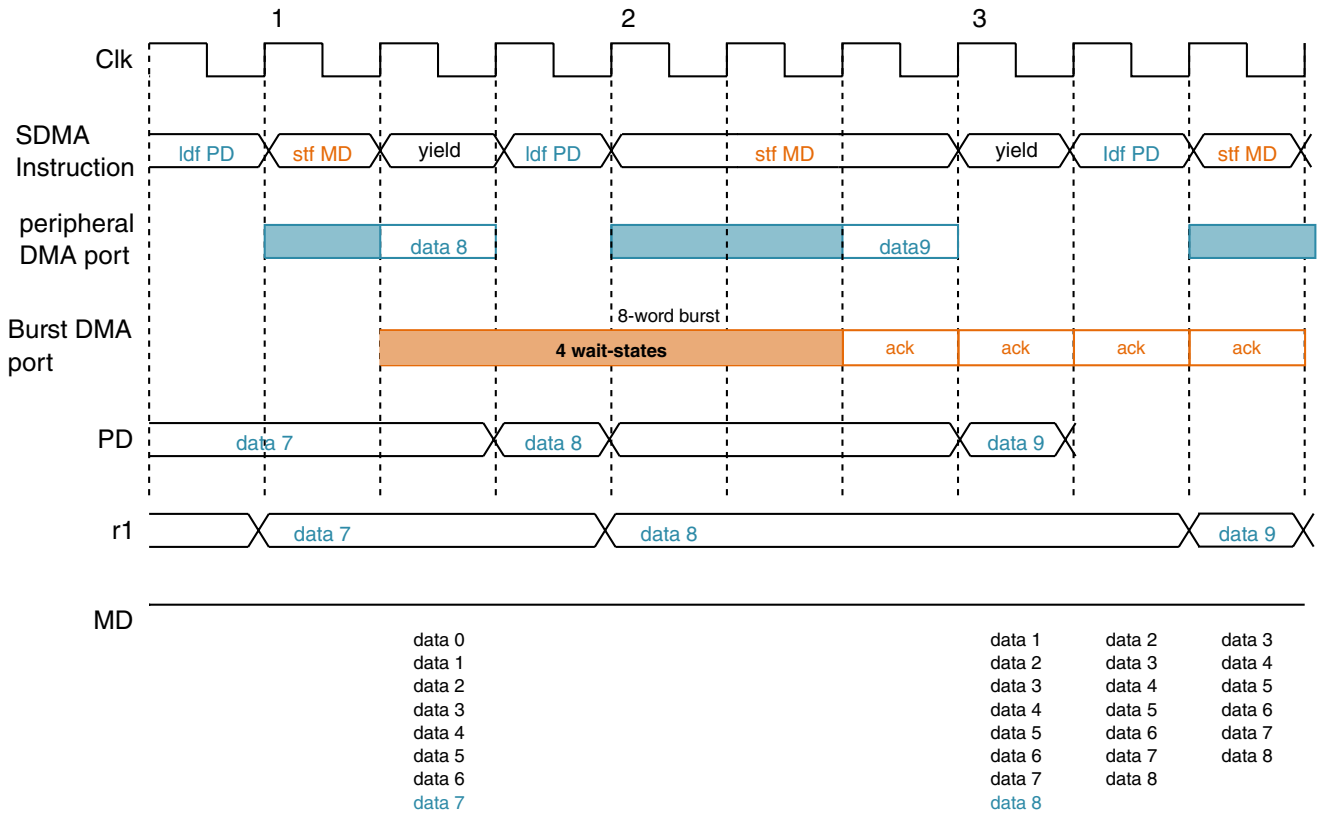


Figure 7-20. Peripheral to External Memory Example (2)

In the previous figure, the write bar means the burst DMA is performing a write burst access. The latency to have the first write acknowledge is four cycles. SDMA is stalled on instruction stf because no acknowledge was received, MD FIFO is full, and there is no empty slot to store data 9. When an acknowledge is sampled by the burst DMA, FIFO is shifted and data 8 is written. As long as there is at least one empty slot in MD FIFO, the stf MD instruction lasts one cycle.

7.2.7.2.3.2 External Memory to Peripheral Transfer

A transfer from the external memory to a peripheral involves the peripheral DMA and the burst DMA.

The code for transferring 100 word from external memory to a word peripheral would be as follows:

External Memory to Peripheral Transfer

```
//SETUP SECTION source and destination addresses are already in r1 and r2
1         stf r1, MSA|PF             //r1=0x1000 and starts a 8-word read burst
2         stf r2, PDA|SZ32|P          //r2=0x2010, setup peripheral DMA destination address
3         bdf ERROR_ADDR_SETUP
4         ldi r0,0x64               //loop counter is 100
//MAIN LOOP TRANFER
```

```
6          loop 3,0
7            ldf r1,MD|32|PF          // read 32 bits of MD and initiate a new read access
                                   // if MD is empty after this reading.
8            stf r1,PD               // store 32 bits of r1 in the PD.
9            yield
           10 ldf r1,MD|32             // last word data is read
11         stf r1,PD                  // last write access
```

On instruction 1, a read burst of 8 words begins. Read data is staged into MD. On instruction 7 (and if data is available in MD), 32 bits are copied into r1. Then instruction 8 writes them into PD and an automatic flush is executed. The SDMA core, peripheral DMA, and burst DMA can work in parallel as long as no SDMA instruction tries to start a new write access on the peripheral DMA while the previous access is still in progress, or as long as there is data in MD when the SDMA tries to read it.

7.2.7.2.4 Transfer Between External Memory and Internal Memory

Since the internal memory (ARM platform RAM) is accessed via the peripheral DMA and the external memory is accessed via the burst DMA, the SDMA scripts that are described in Transfer Between Peripheral and External Memory can be reused. The exception is that the peripheral DMA address registers (PSA or PDA, depending on the script) should be programmed in incremented mode rather than frozen mode.

7.2.7.2.4.1 Internal Memory to Internal Memory

The internal memory can only be accessed via the peripheral DMA, so the script described in Peripheral to Peripheral Transfer can be reused with a different programming of the peripheral DMA address registers.

7.2.7.2.4.2 Transfer Between Peripheral and Internal Memory

For this transfer, the peripheral DMA is also used in copy mode.

The SDMA script is very similar to the one described in Peripheral to Peripheral Transfer, except for the peripheral DMA address registers programming.

7.2.8 ARM Platform Memory Map and Control Register Definitions

The ARM platform controls the SDMA by means of several interface registers. Those registers are described in the current section.

All registers are clocked with the SDMA clock (which means the ARM platform must ensure that the SDMA clock is running when it wants to access any register).

SDMAARM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BD_0000	ARM platform Channel 0 Pointer (SDMAARM_MC0PTR)	32	R/W	0000_0000h	7.2.8.1/ 1409
30BD_0004	Channel Interrupts (SDMAARM_INTR)	32	w1c	0000_0000h	7.2.8.2/ 1409
30BD_0008	Channel Stop/Channel Status (SDMAARM_STOP_STAT)	32	w1c	0000_0000h	7.2.8.3/ 1410
30BD_000C	Channel Start (SDMAARM_HSTART)	32	R/W	0000_0000h	7.2.8.4/ 1410
30BD_0010	Channel Event Override (SDMAARM_EVTOVR)	32	R/W	0000_0000h	7.2.8.5/ 1411
30BD_0014	Channel BP Override (SDMAARM_DSPOVR)	32	R/W	FFFF_FFFFh	7.2.8.6/ 1411
30BD_0018	Channel ARM platform Override (SDMAARM_HOSTOVR)	32	R/W	0000_0000h	7.2.8.7/ 1411
30BD_001C	Channel Event Pending (SDMAARM_EVTPEND)	32	w1c	0000_0000h	7.2.8.8/ 1412
30BD_0024	Reset Register (SDMAARM_RESET)	32	R	0000_0000h	7.2.8.9/ 1413
30BD_0028	DMA Request Error Register (SDMAARM_EVTERR)	32	R	0000_0000h	7.2.8.10/ 1414
30BD_002C	Channel ARM platform Interrupt Mask (SDMAARM_INTRMASK)	32	R/W	0000_0000h	7.2.8.11/ 1414
30BD_0030	Schedule Status (SDMAARM_PSW)	32	R	0000_0000h	7.2.8.12/ 1415
30BD_0034	DMA Request Error Register (SDMAARM_EVTERRDBG)	32	R	0000_0000h	7.2.8.13/ 1415
30BD_0038	Configuration Register (SDMAARM_CONFIG)	32	R/W	0000_0003h	7.2.8.14/ 1416
30BD_003C	SDMA LOCK (SDMAARM_SDMA_LOCK)	32	R/W	0000_0000h	7.2.8.15/ 1417

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BD_0040	OnCE Enable (SDMAARM_ONCE_ENB)	32	R/W	0000_0000h	7.2.8.16/ 1418
30BD_0044	OnCE Data Register (SDMAARM_ONCE_DATA)	32	R/W	0000_0000h	7.2.8.17/ 1418
30BD_0048	OnCE Instruction Register (SDMAARM_ONCE_INSTR)	32	R/W	0000_0000h	7.2.8.18/ 1419
30BD_004C	OnCE Status Register (SDMAARM_ONCE_STAT)	32	R	0000_E000h	7.2.8.19/ 1419
30BD_0050	OnCE Command Register (SDMAARM_ONCE_CMD)	32	R/W	0000_0000h	7.2.8.20/ 1421
30BD_0058	Illegal Instruction Trap Address (SDMAARM_ILLINSTADDR)	32	R/W	0000_0001h	7.2.8.21/ 1421
30BD_005C	Channel 0 Boot Address (SDMAARM_CHN0ADDR)	32	R/W	0000_0050h	7.2.8.22/ 1422
30BD_0060	DMA Requests (SDMAARM_EVT_MIRROR)	32	R	0000_0000h	7.2.8.23/ 1423
30BD_0064	DMA Requests 2 (SDMAARM_EVT_MIRROR2)	32	R	0000_0000h	7.2.8.24/ 1423
30BD_0070	Cross-Trigger Events Configuration Register 1 (SDMAARM_XTRIG_CONF1)	32	R/W	0000_0000h	7.2.8.25/ 1424
30BD_0074	Cross-Trigger Events Configuration Register 2 (SDMAARM_XTRIG_CONF2)	32	R/W	0000_0000h	7.2.8.26/ 1425
30BD_0100	Channel Priority Registers (SDMAARM_SDMA_CHNPRI0)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0104	Channel Priority Registers (SDMAARM_SDMA_CHNPRI1)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0108	Channel Priority Registers (SDMAARM_SDMA_CHNPRI2)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_010C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI3)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0110	Channel Priority Registers (SDMAARM_SDMA_CHNPRI4)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0114	Channel Priority Registers (SDMAARM_SDMA_CHNPRI5)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0118	Channel Priority Registers (SDMAARM_SDMA_CHNPRI6)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_011C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI7)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0120	Channel Priority Registers (SDMAARM_SDMA_CHNPRI8)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0124	Channel Priority Registers (SDMAARM_SDMA_CHNPRI9)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0128	Channel Priority Registers (SDMAARM_SDMA_CHNPRI10)	32	R/W	0000_0000h	7.2.8.27/ 1426

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BD_012C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI11)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0130	Channel Priority Registers (SDMAARM_SDMA_CHNPRI12)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0134	Channel Priority Registers (SDMAARM_SDMA_CHNPRI13)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0138	Channel Priority Registers (SDMAARM_SDMA_CHNPRI14)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_013C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI15)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0140	Channel Priority Registers (SDMAARM_SDMA_CHNPRI16)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0144	Channel Priority Registers (SDMAARM_SDMA_CHNPRI17)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0148	Channel Priority Registers (SDMAARM_SDMA_CHNPRI18)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_014C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI19)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0150	Channel Priority Registers (SDMAARM_SDMA_CHNPRI20)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0154	Channel Priority Registers (SDMAARM_SDMA_CHNPRI21)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0158	Channel Priority Registers (SDMAARM_SDMA_CHNPRI22)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_015C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI23)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0160	Channel Priority Registers (SDMAARM_SDMA_CHNPRI24)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0164	Channel Priority Registers (SDMAARM_SDMA_CHNPRI25)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0168	Channel Priority Registers (SDMAARM_SDMA_CHNPRI26)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_016C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI27)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0170	Channel Priority Registers (SDMAARM_SDMA_CHNPRI28)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0174	Channel Priority Registers (SDMAARM_SDMA_CHNPRI29)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0178	Channel Priority Registers (SDMAARM_SDMA_CHNPRI30)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_017C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI31)	32	R/W	0000_0000h	7.2.8.27/ 1426
30BD_0200	Channel Enable RAM (SDMAARM_CHNENBL0)	32	R/W	0000_0000h	7.2.8.28/ 1427

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BD_0204	Channel Enable RAM (SDMAARM_CHNENBL1)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0208	Channel Enable RAM (SDMAARM_CHNENBL2)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_020C	Channel Enable RAM (SDMAARM_CHNENBL3)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0210	Channel Enable RAM (SDMAARM_CHNENBL4)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0214	Channel Enable RAM (SDMAARM_CHNENBL5)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0218	Channel Enable RAM (SDMAARM_CHNENBL6)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_021C	Channel Enable RAM (SDMAARM_CHNENBL7)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0220	Channel Enable RAM (SDMAARM_CHNENBL8)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0224	Channel Enable RAM (SDMAARM_CHNENBL9)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0228	Channel Enable RAM (SDMAARM_CHNENBL10)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_022C	Channel Enable RAM (SDMAARM_CHNENBL11)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0230	Channel Enable RAM (SDMAARM_CHNENBL12)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0234	Channel Enable RAM (SDMAARM_CHNENBL13)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0238	Channel Enable RAM (SDMAARM_CHNENBL14)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_023C	Channel Enable RAM (SDMAARM_CHNENBL15)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0240	Channel Enable RAM (SDMAARM_CHNENBL16)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0244	Channel Enable RAM (SDMAARM_CHNENBL17)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0248	Channel Enable RAM (SDMAARM_CHNENBL18)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_024C	Channel Enable RAM (SDMAARM_CHNENBL19)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0250	Channel Enable RAM (SDMAARM_CHNENBL20)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0254	Channel Enable RAM (SDMAARM_CHNENBL21)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0258	Channel Enable RAM (SDMAARM_CHNENBL22)	32	R/W	0000_0000h	7.2.8.28/ 1427

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BD_025C	Channel Enable RAM (SDMAARM_CHNENBL23)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0260	Channel Enable RAM (SDMAARM_CHNENBL24)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0264	Channel Enable RAM (SDMAARM_CHNENBL25)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0268	Channel Enable RAM (SDMAARM_CHNENBL26)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_026C	Channel Enable RAM (SDMAARM_CHNENBL27)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0270	Channel Enable RAM (SDMAARM_CHNENBL28)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0274	Channel Enable RAM (SDMAARM_CHNENBL29)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0278	Channel Enable RAM (SDMAARM_CHNENBL30)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_027C	Channel Enable RAM (SDMAARM_CHNENBL31)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0280	Channel Enable RAM (SDMAARM_CHNENBL32)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0284	Channel Enable RAM (SDMAARM_CHNENBL33)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0288	Channel Enable RAM (SDMAARM_CHNENBL34)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_028C	Channel Enable RAM (SDMAARM_CHNENBL35)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0290	Channel Enable RAM (SDMAARM_CHNENBL36)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0294	Channel Enable RAM (SDMAARM_CHNENBL37)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_0298	Channel Enable RAM (SDMAARM_CHNENBL38)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_029C	Channel Enable RAM (SDMAARM_CHNENBL39)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_02A0	Channel Enable RAM (SDMAARM_CHNENBL40)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_02A4	Channel Enable RAM (SDMAARM_CHNENBL41)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_02A8	Channel Enable RAM (SDMAARM_CHNENBL42)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_02AC	Channel Enable RAM (SDMAARM_CHNENBL43)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_02B0	Channel Enable RAM (SDMAARM_CHNENBL44)	32	R/W	0000_0000h	7.2.8.28/ 1427

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BD_02B4	Channel Enable RAM (SDMAARM_CHNENBL45)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_02B8	Channel Enable RAM (SDMAARM_CHNENBL46)	32	R/W	0000_0000h	7.2.8.28/ 1427
30BD_02BC	Channel Enable RAM (SDMAARM_CHNENBL47)	32	R/W	0000_0000h	7.2.8.28/ 1427

7.2.8.1 ARM platform Channel 0 Pointer (SDMAARM_MC0PTR)

Address: 30BD_0000h base + 0h offset = 30BD_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MC0PTR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_MC0PTR field descriptions

Field	Description
MC0PTR	**Channel 0 Pointer** contains the 32-bit address, in ARM platform memory, of channel 0 control block (the boot channel). Appendix A fully describes the SDMA Application Programming Interface (API). The ARM platform has a read/write access and the SDMA has a read-only access.

7.2.8.2 Channel Interrupts (SDMAARM_INTR)

Address: 30BD_0000h base + 4h offset = 30BD_0004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HI[31:0]																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_INTR field descriptions

Field	Description
HI[31:0]	The ARM platform Interrupts register contains the 32 HI[i] bits. If any bit is set, it will cause an interrupt to the ARM platform. This register is a "write-ones" register to the ARM platform. When the ARM platform sets a bit in this register the corresponding HI[i] bit is cleared. The interrupt service routine should clear individual channel bits when their interrupts are serviced, failure to do so will cause continuous interrupts. The SDMA is responsible for setting the HI[i] bit corresponding to the current channel when the corresponding `done` instruction is executed.

7.2.8.3 Channel Stop/Channel Status (SDMAARM_STOP_STAT)

Address: 30BD_0000h base + 8h offset = 30BD_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HE																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_STOP_STAT field descriptions

Field	Description
HE	This 32-bit register gives access to the ARM platform Enable bits. There is one bit for every channel. This register is a "write-ones" register to the ARM platform. When the ARM platform writes 1 in bit i of this register, it clears the HE[i] and HSTART[i] bits. Reading this register yields the current state of the HE[i] bits.

7.2.8.4 Channel Start (SDMAARM_HSTART)

Address: 30BD_0000h base + Ch offset = 30BD_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HSTART_HE																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_HSTART field descriptions

Field	Description
HSTART_HE	The HSTART_HE registers are 32 bits wide with one bit for every channel. When a bit is written to 1, it enables the corresponding channel. Two physical registers are accessed with that address (HSTART and HE), which enables the ARM platform to trigger a channel a second time before the first trigger is processed. • This register is a "write-ones" register to the ARM platform. Neither HSTART[i] bit can be set while the corresponding HE[i] bit is cleared. • When the ARM platform tries to set the HSTART[i] bit by writing a one (if the corresponding HE[i] bit is clear), the bit in the HSTART[i] register will remain cleared and the HE[i] bit will be set. • If the corresponding HE[i] bit was already set, the HSTART[i] bit will be set. The next time the SDMA channel *i* attempts to clear the HE[i] bit by means of a `done` instruction, the bit in the HSTART[i] register will be cleared and the HE[i] bit will take the old value of the HSTART[i] bit. • Reading this register yields the current state of the HSTART[i] bits. This mechanism enables the ARM platform to pipeline two HSTART commands per channel.

7.2.8.5 Channel Event Override (SDMAARM_EVTOVR)

Address: 30BD_0000h base + 10h offset = 30BD_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EO																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVTOVR field descriptions

Field	Description
EO	The Channel Event Override register contains the 32 EO[i] bits. A bit set in this register causes the SDMA to ignore DMA requests when scheduling the corresponding channel.

7.2.8.6 Channel BP Override (SDMAARM_DSPOVR)

Address: 30BD_0000h base + 14h offset = 30BD_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DO																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

SDMAARM_DSPOVR field descriptions

Field	Description
DO	This register is reserved. All DO bits should be set to the reset value of 1. A setting of 0 will prevent SDMA channels from starting according to the condition described in Runnable Channels Evaluation. 0 - Reserved 1 - Reset value.

7.2.8.7 Channel ARM platform Override (SDMAARM_HOSTOVR)

Address: 30BD_0000h base + 18h offset = 30BD_0018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HO																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_HOSTOVR field descriptions

Field	Description
HO	The Channel ARM platform Override register contains the 32 HO[i] bits. A bit set in this register causes the SDMA to ignore the ARM platform enable bit (HE) when scheduling the corresponding channel.

7.2.8.8 Channel Event Pending (SDMAARM_EVTPEND)

Address: 30BD_0000h base + 1Ch offset = 30BD_001Ch

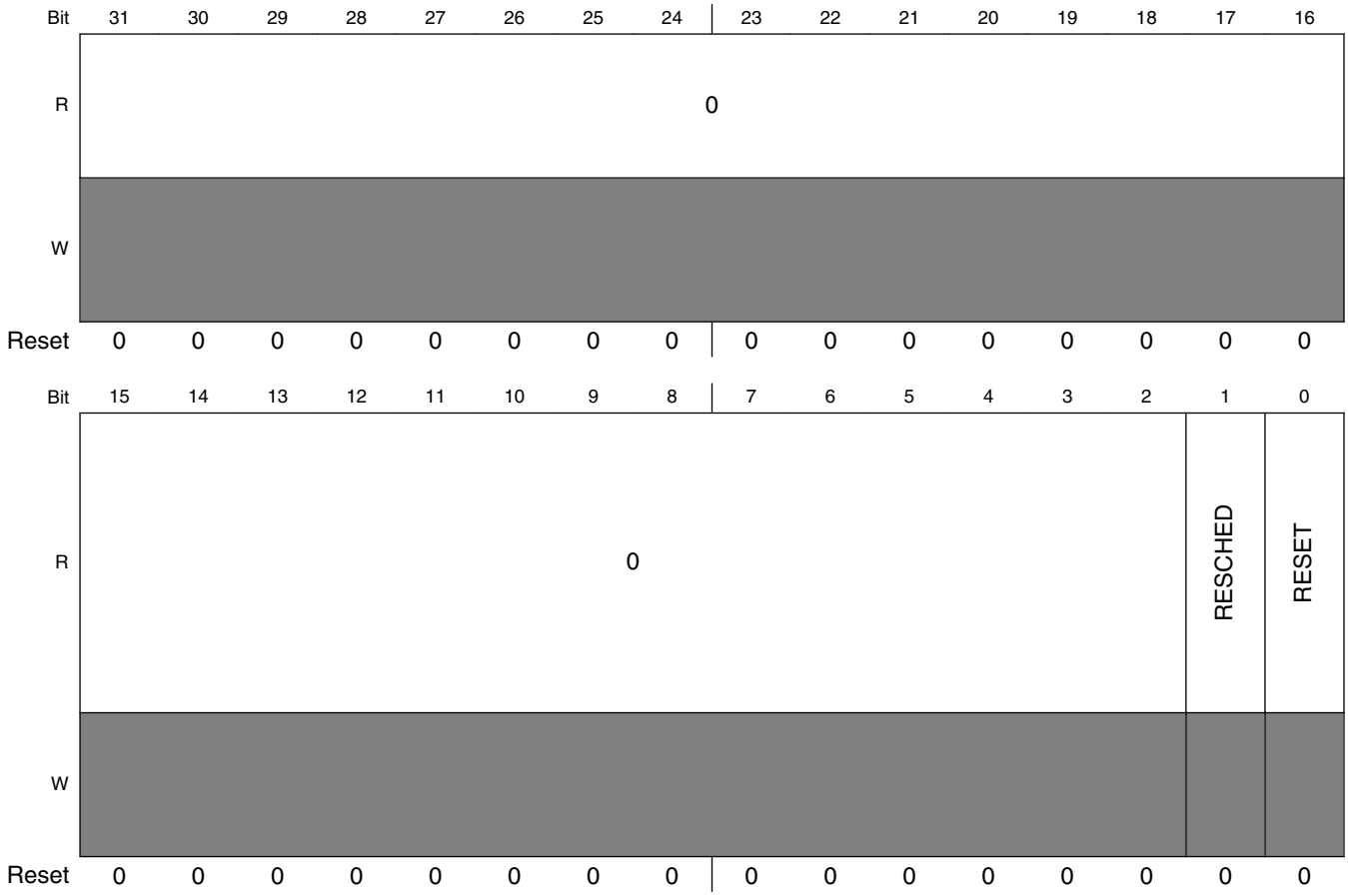
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EP																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVTPEND field descriptions

Field	Description
EP	The Channel Event Pending register contains the 32 EP[i] bits. Reading this register enables the ARM platform to determine what channels are pending after the reception of a DMA request. • Setting a bit in this register causes the SDMA to reevaluate scheduling as if a DMA request mapped on this channel had occurred. This is useful for starting up channels, so that initialization is done before awaiting the first request. The scheduler can also set bits in the EVTPEND register according to the received DMA requests. • The EP[i] bit may be cleared by the `done` instruction when running the channel *i* script. This a "write-ones" mechanism: Writing a '0' does not clear the corresponding bit.

7.2.8.9 Reset Register (SDMAARM_RESET)

Address: 30BD_0000h base + 24h offset = 30BD_0024h



SDMAARM_RESET field descriptions

Field	Description
31–2 Reserved	This read-only field is reserved and always has the value 0.
1 RESCHED	When set, this bit forces the SDMA to reschedule as if a script had executed a `done` instruction. This enables the ARM platform to recover from a runaway script on a channel by clearing its HE[i] bit via the STOP register, and then forcing a reschedule via the RESCHED bit. The RESCHED bit is cleared when the context switch starts.
0 RESET	When set, this bit causes the SDMA to be held in a software reset. The internal reset signal is held low 16 cycles; the RESET bit is automatically cleared when the internal reset signal rises.

7.2.8.10 DMA Request Error Register (SDMAARM_EVTERR)

Address: 30BD_0000h base + 28h offset = 30BD_0028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CHNERR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVTERR field descriptions

Field	Description
CHNERR	This register is used by the SDMA to warn the ARM platform when an incoming DMA request was detected and it triggers a channel that is already pending or being serviced. This probably means there is an overflow of data for that channel. • An interrupt is sent to the ARM platform if the corresponding channel bit is set in the INTRMASK register. • This is a "write-ones" register for the scheduler. It is only able to set the flags. The flags are cleared when the register is read by the ARM platform or during SDMA reset. • The CHNERR[i] bit is set when a DMA request that triggers channel *i* is received through the corresponding input pins and the EP[i] bit is already set; the EVTERR[i] bit is unaffected if the ARM platform tries to set the EP[i] bit, whereas, that EP[i] bit is already set.

7.2.8.11 Channel ARM platform Interrupt Mask (SDMAARM_INTRMASK)

Address: 30BD_0000h base + 2Ch offset = 30BD_002Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HIMASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_INTRMASK field descriptions

Field	Description
HIMASK	The Interrupt Mask Register contains 32 interrupt generation mask bits. If bit HIMASK[i] is set, the HI[i] bit is set and an interrupt is sent to the ARM platform when a DMA request error is detected on channel *i* (for example, EVTERR[i] is set).

7.2.8.12 Schedule Status (SDMAARM_PSW)

Address: 30BD_0000h base + 30h offset = 30BD_0030h

Bit	31–16	15–13	12–8	7–4	3–0
R	0	NCP[2:0]	NCR[4:0]	CCP[2:0]	CCR[4:0]
W					
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0	0 0 0 0 0	0 0 0 0	0 0 0 0

SDMAARM_PSW field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–13 NCP[2:0]	The Next Channel Priority gives the next pending channel priority. When the priority is 0, it means there is no pending channel and the NCR value has no meaning. 0 No running channel 1 Active channel priority
12–8 NCR[4:0]	The Next Channel Register indicates the number of the next scheduled pending channel with the highest priority.
7–4 CCP[2:0]	The Current Channel Priority indicates the priority of the current active channel. When the priority is 0, no channel is running: The SDMA is idle and the CCR value has no meaning. In the case that the SDMA has finished running the channel and has entered sleep state, CCP will indicate the priority of previous running channel. 0 No running channel 1 Active channel priority
CCR[4:0]	The Current Channel Register indicates the number of the channel that is being executed by the SDMA. SDMA. In the case that the SDMA has finished running the channel and has entered sleep state, CCR will indicate the previous running channel.

7.2.8.13 DMA Request Error Register (SDMAARM_EVTERRDBG)

Address: 30BD_0000h base + 34h offset = 30BD_0034h

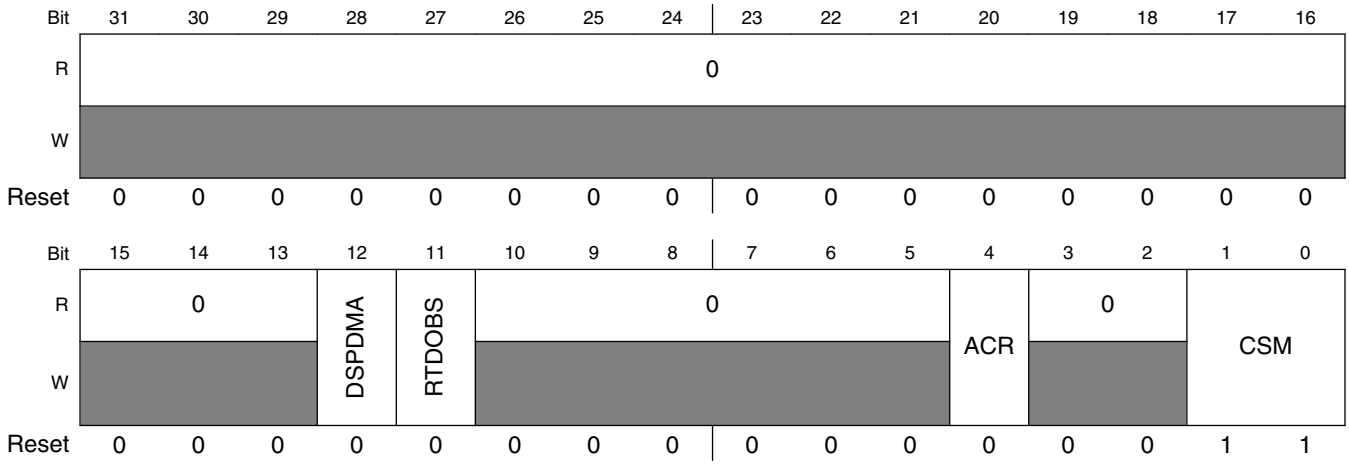
Bit	31–16	15–0
R	CHNERR	
W		
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

SDMAARM_EVTERRDBG field descriptions

Field	Description
CHNERR	This register is the same as EVTERR, except reading it does not clear its contents. This address is meant to be used in debug mode. The ARM platform OnCE may check this register value without modifying it.

7.2.8.14 Configuration Register (SDMAARM_CONFIG)

Address: 30BD_0000h base + 38h offset = 30BD_0038h



SDMAARM_CONFIG field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value 0.
12 DSPDMA	This bit's function is reserved and should be configured as zero. 0 - Reset Value 1 - Reserved
11 RTDOBS	Indicates if Real-Time Debug pins are used: They do not toggle by default in order to reduce power consumption. 0 RTD pins disabled 1 RTD pins enabled
10–5 Reserved	This read-only field is reserved and always has the value 0.
4 ACR	ARM platform DMA / SDMA Core Clock Ratio. Selects the clock ratio between ARM platform DMA interfaces (burst DMA and peripheral DMA) and the internal SDMA core clock. The frequency selection is determined separately by the chip clock controller. This bit has to match the configuration of the chip clock controller that generates the clocks used in the SDMA. 0 ARM platform DMA interface frequency equals twice core frequency 1 ARM platform DMA interface frequency equals core frequency
3–2 Reserved	This read-only field is reserved and always has the value 0.
CSM	Selects the Context Switch Mode. The ARM platform has a read/write access. The SDMA cannot modify that register. The value at reset is 3, which selects the dynamic context switch by default. That register can be modified at anytime but the new context switch configuration will only be taken into account at the start of the next restore phase. NOTE: The first call to SDMA's channel 0 Bootload script after reset should use static context switch mode to ensure the context RAM for channel 0 is initialized in the channel SAVE Phase. After Channel 0 is run once, then any of the dynamic context modes can be used.

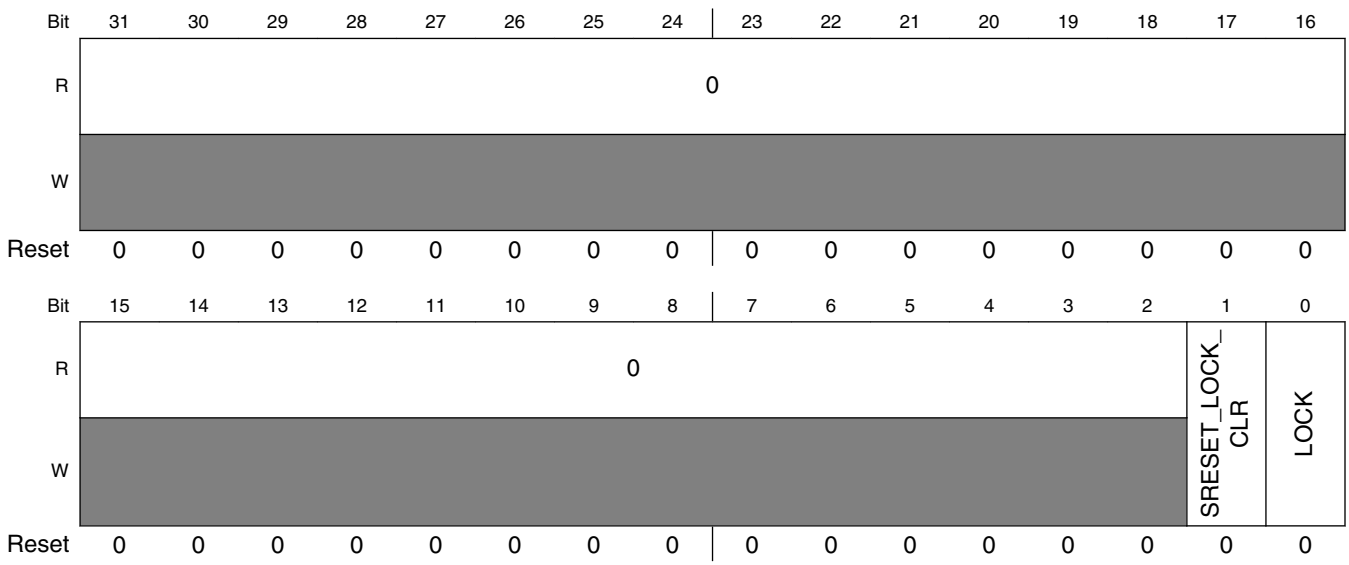
Table continues on the next page...

SDMAARM_CONFIG field descriptions (continued)

Field	Description
	0 static 1 dynamic low power 2 dynamic with no loop 3 dynamic

7.2.8.15 SDMA LOCK (SDMAARM_SDMA_LOCK)

Address: 30BD_0000h base + 3Ch offset = 30BD_003Ch



SDMAARM_SDMA_LOCK field descriptions

Field	Description
31–2 Reserved	This read-only field is reserved and always has the value 0.
1 SRESET_LOCK_CLR	The SRESET_LOCK_CLR bit determine if the LOCK bit is cleared on a software reset triggered by writing to the RESET register. This bit cannot be changed if LOCK=1. SREST_LOCK_CLR is cleared by conditions that clear the LOCK bit. 0 Software Reset does not clear the LOCK bit. 1 Software Reset clears the LOCK bit.
0 LOCK	The LOCK bit is used to restrict access to update SDMA script memory through ROM channel zero scripts and through the OnCE interface under ARM platform control. The LOCK bit is set: • The SDMA_LOCK, ONCE_ENB,CH0ADDR, and ILLINSTADDR registers cannot be written. These registers can be read, but writes are ignored. • SDMA software executing out of ROM or RAM may check the LOCK bit in the LOCK register Lock Status Register (SDMACORE_SDMA_LOCK) to determine if certain operations are allowed, such as up-loading new scripts.

Table continues on the next page...

SDMAARM_SDMA_LOCK field descriptions (continued)

Field	Description
	Once the LOCK bit is set to 1, only a reset can clear it. The LOCK bit is cleared by a hardware reset. LOCK is cleared by a software reset only if SRESET_LOCK_CLR is set. 0 LOCK disengaged. 1 LOCK enabled.

7.2.8.16 OnCE Enable (SDMAARM_ONCE_ENB)

Address: 30BD_0000h base + 40h offset = 30BD_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															ENB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_ONCE_ENB field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 ENB	The OnCE Enable register selects the OnCE control source: When cleared (0), the OnCE registers are accessed through the JTAG interface; when set (1), the OnCE registers may be accessed by the ARM platform through the addresses described, as follows. • After reset, the OnCE registers are accessed through the JTAG interface. • Writing a 1 to ENB enables the ARM platform to access the ONCE_* as any other SDMA control register. • When cleared (0), all the ONCE_xxx registers cannot be written. The value of ENB cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

7.2.8.17 OnCE Data Register (SDMAARM_ONCE_DATA)

Address: 30BD_0000h base + 44h offset = 30BD_0044h

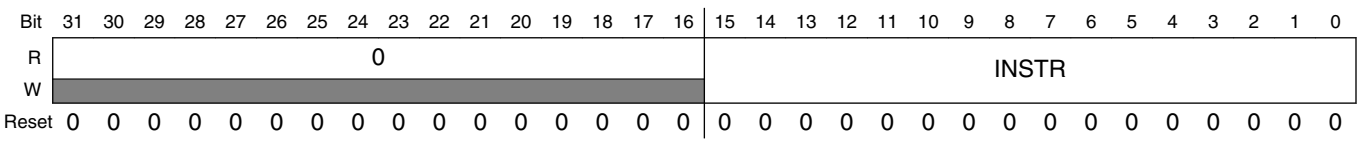
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_ONCE_DATA field descriptions

Field	Description
DATA	Data register of the OnCE JTAG controller. Refer to OnCE and Real-Time Debug for information on this register.

7.2.8.18 OnCE Instruction Register (SDMAARM_ONCE_INSTR)

Address: 30BD_0000h base + 48h offset = 30BD_0048h

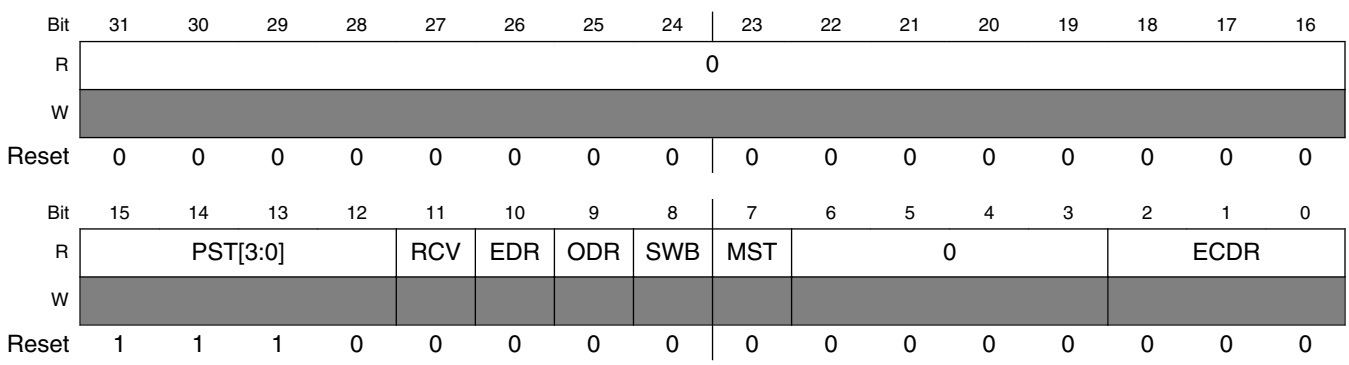


SDMAARM_ONCE_INSTR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
INSTR	Instruction register of the OnCE JTAG controller. Refer to OnCE and Real-Time Debug for information on this register.

7.2.8.19 OnCE Status Register (SDMAARM_ONCE_STAT)

Address: 30BD_0000h base + 4Ch offset = 30BD_004Ch



SDMAARM_ONCE_STAT field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–12 PST[3:0]	The Processor Status bits reflect the state of the SDMA RISC engine. Its states are as follows: • The "Program" state is the usual instruction execution cycle. • The "Data" state is inserted when there are wait-states during a load or a store on the data bus (ld or st).

Table continues on the next page...

SDMAARM_ONCE_STAT field descriptions (continued)

Field	Description
	• The "Change of Flow" state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel switching instructions). • The "Change of Flow in Loop" state is used when an error causes a hardware loop exit. • The "Debug" state means the SDMA is in debug mode. • The "Functional Unit" state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In "Sleep" modes, no script is running (this is the RISC engine idle state). The "after Reset" is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 will be executed (boot operation). • The "in Sleep" states are the same as above except they do not have any corresponding channel: They are used when entering debug mode after reset. The reason is that it is necessary to return to the "Sleep after Reset" state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Save 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change Flow in Loop in Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Restore
11 RCV	After each write access to the real time buffer (RTB), the RCV bit is set. This bit is cleared after execution of an `rbuffer` command and on a JTAG reset.
10 EDR	This flag is raised when the SDMA has entered debug mode after an external debug request.
9 ODR	This flag is raised when the SDMA has entered debug mode after a OnCE debug request.
8 SWB	This flag is raised when the SDMA has entered debug mode after a software breakpoint.
7 MST	This flag is raised when the OnCE is controlled from the ARM platform peripheral interface. 0 The JTAG interface controls the OnCE. 1 The ARM platform peripheral interface controls the OnCE.
6–3 Reserved	This read-only field is reserved and always has the value 0.
ECDR	Event Cell Debug Request. If the debug request comes from the event cell, the reason for entering debug mode is given by the EDR bits. If all three bits of the EDR are reset, then it did not generate any debug request. If the cell did generate a debug request, then at least one of the EDR bits is set (the meaning of the encoding is given below). The encoding of the EDR bits is useful to find out more precisely why the debug request was generated. A debug request from an event cell is generated for a specific combination of the addra_cond, addrb_cond, and data_cond conditions. The value of those fields is given by the EDR bits.

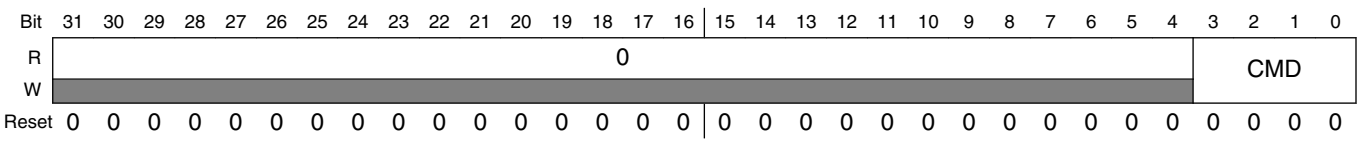
Table continues on the next page...

SDMAARM_ONCE_STAT field descriptions (continued)

Field	Description
	0 1 matched addra_cond 1 1 matched addrb_cond 2 1 matched data_cond

7.2.8.20 OnCE Command Register (SDMAARM_ONCE_CMD)

Address: 30BD_0000h base + 50h offset = 30BD_0050h

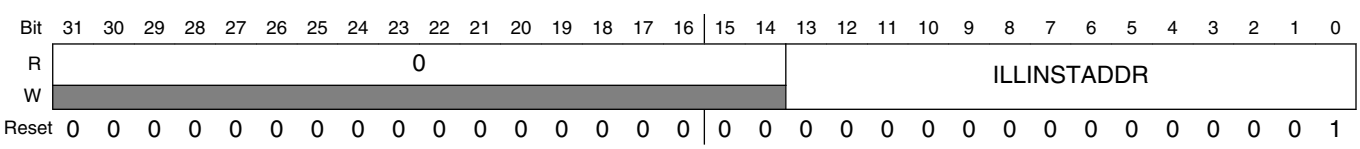


SDMAARM_ONCE_CMD field descriptions

Field	Description
31–4 Reserved	This read-only field is reserved and always has the value 0.
CMD	Writing to this register will cause the OnCE to execute the command that is written. When needed, the ONCE_DATA and ONCE_INSTR registers should be loaded with the correct value before writing the command to that register. For a list of the OnCE commands and their usage, see OnCE and Real-Time Debug. **NOTE:** 7-15 *reserved* 0 rstatus 1 dmov 2 exec_once 3 run_core 4 exec_core 5 debug_rqst 6 rbuffer

7.2.8.21 Illegal Instruction Trap Address (SDMAARM_ILLINSTADDR)

Address: 30BD_0000h base + 58h offset = 30BD_0058h

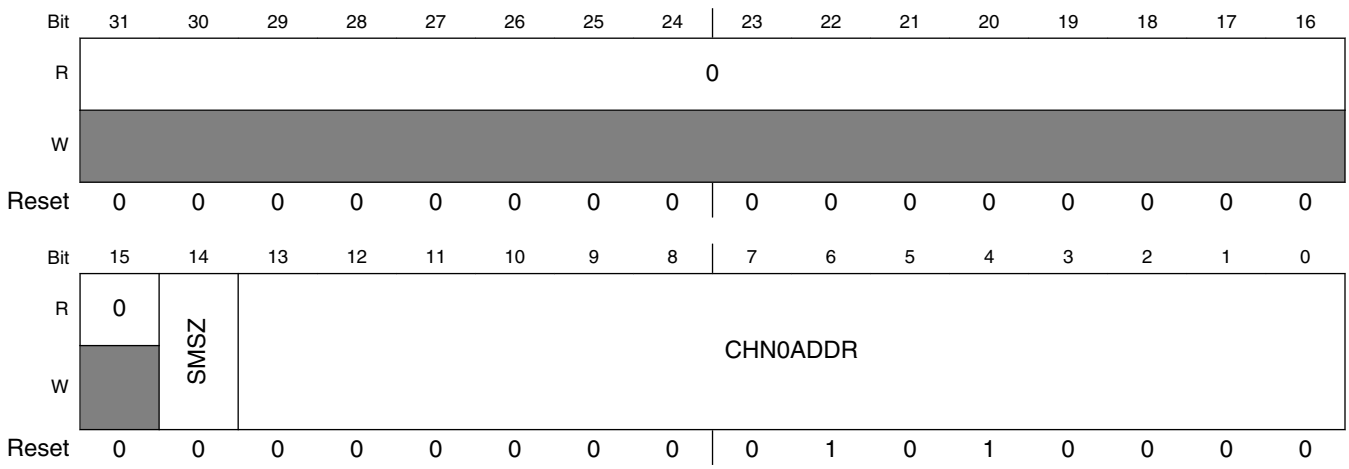


SDMAARM_ILLINSTADDR field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
ILLINSTADDR	The Illegal Instruction Trap Address is the address where the SDMA jumps when an illegal instruction is executed. It is 0x0001 after reset. The value of ILLINSTADDR cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

7.2.8.22 Channel 0 Boot Address (SDMAARM_CHN0ADDR)

Address: 30BD_0000h base + 5Ch offset = 30BD_005Ch



SDMAARM_CHN0ADDR field descriptions

Field	Description
31–15 Reserved	This read-only field is reserved and always has the value 0.
14 SMSZ	The bit 14 (Scratch Memory Size) determines if scratch memory must be available after every channel context. After reset, it is equal to 0, which defines a RAM space of 24 words for each channel. All of this area stores the channel context. By setting this bit, 32 words are reserved for every channel context, which gives eight additional words that can be used by the channel script to store any type of data. Those words are never erased by the context switching mechanism. The value of SMSZ cannot be changed if the LOCK bit in the SDMA_LOCK register is set. 0 24 words per context 1 32 words per context
CHN0ADDR	This 14-bit register is used by the boot code of the SDMA. After reset, it points to the standard boot routine in ROM (channel 0 routine). By changing this address, you can perform a boot sequence with your own routine. The very first instructions of the boot code fetch the contents of this register (it is also mapped in the SDMA memory space) and jump to the given address. The reset value is 0x0050 (decimal 80). The value of CHN0ADDR cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

7.2.8.23 DMA Requests (SDMAARM_EVT_MIRROR)

Address: 30BD_0000h base + 60h offset = 30BD_0060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EVENTS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVT_MIRROR field descriptions

Field	Description
EVENTS	This register reflects the DMA requests received by the SDMA for events 31-0. The ARM platform and the SDMA have a read-only access. There is one bit associated with each of 32 DMA request events. This information may be useful during debug of the blocks that generate the DMA requests. The EVT_MIRROR register is cleared following read access. 0 DMA request event not pending 1 DMA request event pending

7.2.8.24 DMA Requests 2 (SDMAARM_EVT_MIRROR2)

Address: 30BD_0000h base + 64h offset = 30BD_0064h

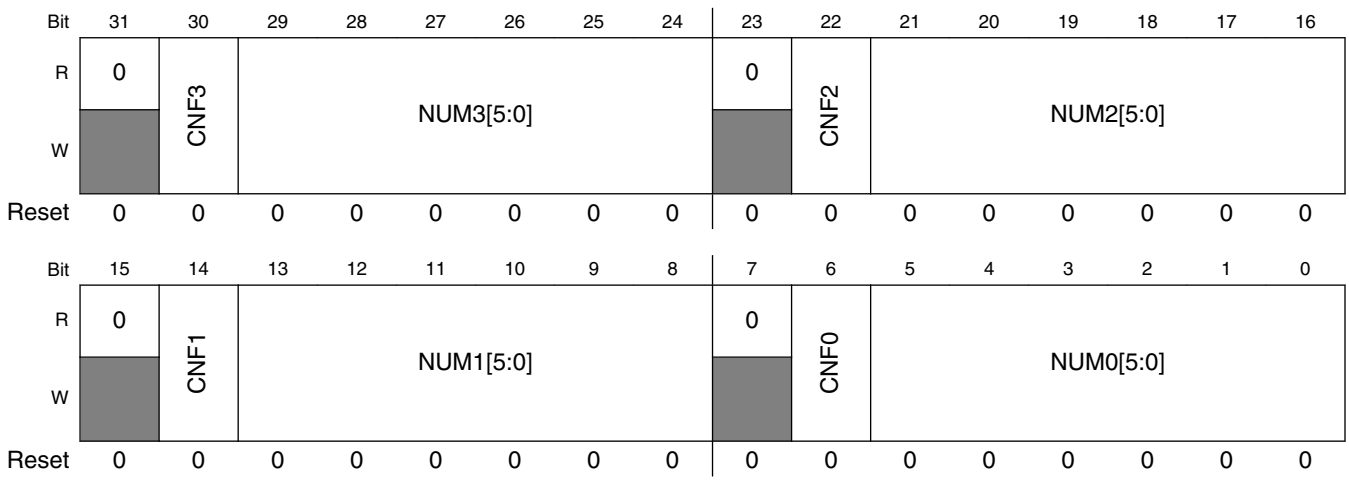
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																EVENTS[47:32]															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVT_MIRROR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EVENTS[47:32]	This register reflects the DMA requests received by the SDMA for events 47-32. The ARM platform and the SDMA have a read-only access. There is one bit associated with each of DMA request events. This information may be useful during debug of the blocks that generate the DMA requests. The EVT_MIRROR2 register is cleared following read access. 0 - DMA request event not pending 1- DMA request event pending

7.2.8.25 Cross-Trigger Events Configuration Register 1 (SDMAARM_XTRIG_CONF1)

Address: 30BD_0000h base + 70h offset = 30BD_0070h



SDMAARM_XTRIG_CONF1 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30 CNF3	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by the reception of a DMA request or by the starting of a channel script execution. 0 channel 1 DMA request
29–24 NUM3[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
23 Reserved	This read-only field is reserved and always has the value 0.
22 CNF2	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
21–16 NUM2[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
15 Reserved	This read-only field is reserved and always has the value 0.
14 CNF1	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution.

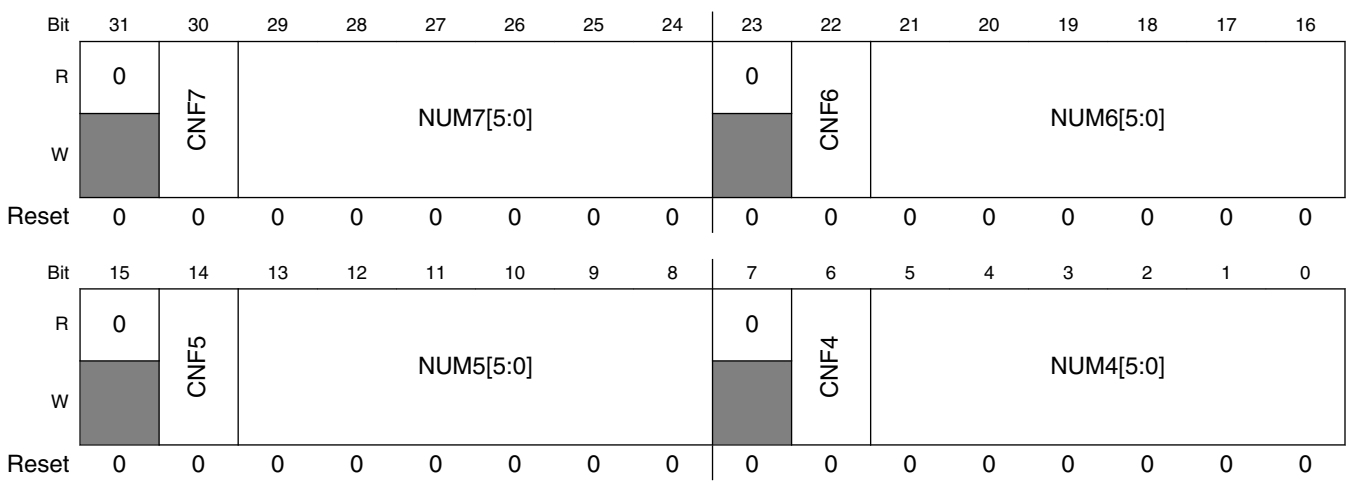
Table continues on the next page...

SDMAARM_XTRIG_CONF1 field descriptions (continued)

Field	Description
	0 channel 1 DMA request
13–8 NUM1[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
7 Reserved	This read-only field is reserved and always has the value 0.
6 CNF0	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
NUM0[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.

7.2.8.26 Cross-Trigger Events Configuration Register 2 (SDMAARM_XTRIG_CONF2)

Address: 30BD_0000h base + 74h offset = 30BD_0074h



SDMAARM_XTRIG_CONF2 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30 CNF7	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request

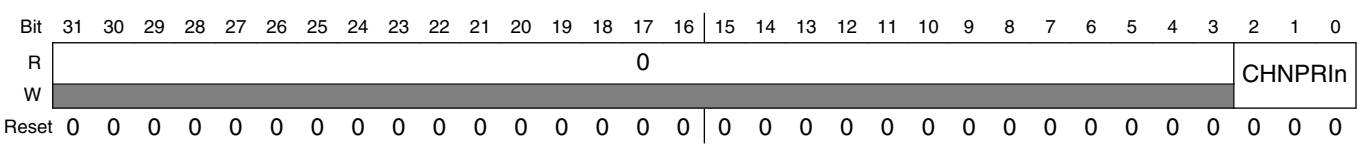
Table continues on the next page...

SDMAARM_XTRIG_CONF2 field descriptions (continued)

Field	Description
29–24 NUM7[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
23 Reserved	This read-only field is reserved and always has the value 0.
22 CNF6	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
21–16 NUM6[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
15 Reserved	This read-only field is reserved and always has the value 0.
14 CNF5	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution 0 channel 1 DMA request
13–8 NUM5[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
7 Reserved	This read-only field is reserved and always has the value 0.
6 CNF4	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
NUM4[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.

7.2.8.27 Channel Priority Registers (SDMAARM_SDMA_CHNPRI*n*)

Address: 30BD_0000h base + 100h offset + (4d × i), where i=0d to 31d



SDMAARM_SDMA_CHNPRI*n* field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
CHNPRIn	This contains the priority of channel number *n*. Useful values are between 1 and 7; 0 is reserved by the SDMA hardware to determine when there is no pending channel. Reset value is 0, which prevents the channels from starting.

7.2.8.28 Channel Enable RAM (SDMAARM_CHNENBL*n*)

Address: 30BD_0000h base + 200h offset + (4d × i), where i=0d to 47d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ENBLn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_CHNENBL*n* field descriptions

Field	Description
ENBLn	This 32-bit value selects the channels that are triggered by the DMA request number *n*. If ENBLn[i] is set to 1, bit EP[i] will be set when the DMA request *n* is received. These 48 32-bit registers are physically located in a RAM, with no known reset value. It is thus essential for the ARM platform to program them before any DMA request is triggered to the SDMA, otherwise an unpredictable combination of channels may be started.

7.2.9 BP Memory Map and Control Register Definitions

The following section describes SDMA control registers available to the BP.

NOTE

These registers are physically implemented in all platforms, but are not accessible when the SDMA BP control port is not connected. Reset values are calculated to allow the system to work when those registers cannot be accessed.

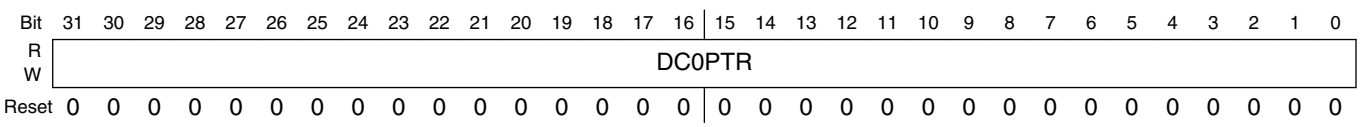
All registers are clocked with the SDMA clock (which means the SDMA clock must be running when the BP wants to access any register).

SDMABP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BD_0000	Channel 0 Pointer (SDMABP_DC0PTR)	32	R/W	0000_0000h	7.2.9.1/ 1428
30BD_0004	Channel Interrupts (SDMABP_INTR)	32	w1c	0000_0000h	7.2.9.2/ 1428
30BD_0008	Channel Stop/Channel Status (SDMABP_STOP_STAT)	32	R/W	0000_0000h	7.2.9.3/ 1429
30BD_000C	Channel Start (SDMABP_DSTART)	32	R	0000_0000h	7.2.9.4/ 1429
30BD_0028	DMA Request Error Register (SDMABP_EVTERR)	32	R	0000_0000h	7.2.9.5/ 1430
30BD_002C	Channel DSP Interrupt Mask (SDMABP_INTRMASK)	32	R/W	0000_0000h	7.2.9.6/ 1430
30BD_0034	DMA Request Error Register (SDMABP_EVTERRDBG)	32	R	0000_0000h	7.2.9.7/ 1431

7.2.9.1 Channel 0 Pointer (SDMABP_DC0PTR)

Address: 30BD_0000h base + 0h offset = 30BD_0000h

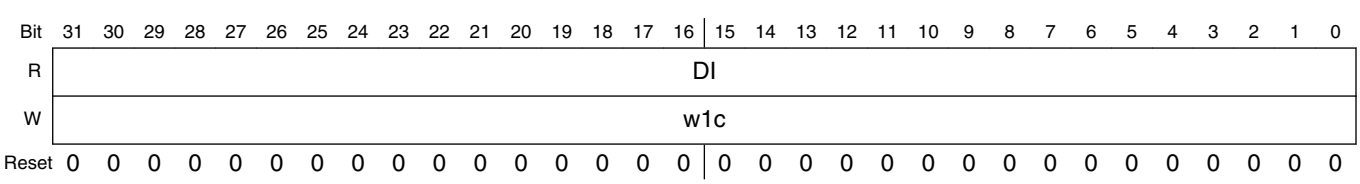


SDMABP_DC0PTR field descriptions

Field	Description
DC0PTR	**Channel 0 Pointer** contains the 32-bit address, in BP memory, of the array of channel control blocks starting with the one for channel 0 (the control channel). This register should be initialized by the BP before it enables a channel (for example, channel 0). See the API document SDMA Scripts User Manual for the use of this register. The BP has a read/write access and the SDMA has a read-only access.

7.2.9.2 Channel Interrupts (SDMABP_INTR)

Address: 30BD_0000h base + 4h offset = 30BD_0004h



SDMABP_INTR field descriptions

Field	Description
DI	The BP Interrupts register contains the 32 DI[i] bits. If any bit is set, it will cause an interrupt to the BP. • This register is a "write-ones" register to the BP. When the BP sets a bit in this register, the corresponding DI[i] bit is cleared. • The interrupt service routine should clear individual channel bits when their interrupts are serviced; failure to do so will cause continuous interrupts. • The SDMA is responsible for setting the DI[i] bit corresponding to the current channel when the corresponding `done` instruction is executed.

7.2.9.3 Channel Stop/Channel Status (SDMABP_STOP_STAT)

Address: 30BD_0000h base + 8h offset = 30BD_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DE																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_STOP_STAT field descriptions

Field	Description
DE	This 32-bit register gives access to the BP (DSP) Enable bits, DE. There is one bit for every channel. • This register is a "write-ones" register to the BP. • When the BP writes 1 in bit *i* of this register, it clears the DE[i] and DSTART[i] bits. • Reading this register yields the current state of the DE[i] bits.

7.2.9.4 Channel Start (SDMABP_DSTART)

Address: 30BD_0000h base + Ch offset = 30BD_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DSTART_DE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_DSTART field descriptions

Field	Description
DSTART_DE	The DSTART_DE registers are 32 bits wide with one bit for every channel. • When a bit is written to 1, it enables the corresponding channel. • Two physical registers are accessed with that address (DSTART and DE), which enables the BP to trigger a channel a second time before the first trigger was processed. • This register is a "write-ones" register to the BP. Neither DSTART[i] bit can be set while the corresponding DE[i] bit is cleared.

SDMABP_DSTART field descriptions (continued)

Field	Description
	• When the BP tries to set the DSTART[i] bit by writing a one (if the corresponding DE[i] bit is clear), the bit in the DSTART[i] register will remain cleared and the DE[i] bit will be set. If the corresponding DE[i] bit was already set, the DSTART[i] bit will be set. • The next time the SDMA channel *i* attempts to clear the DE[i] bit by means of a `done` instruction, the bit in the DSTART[i] register will be cleared and the DE[i] bit will take the old value of the DSTART[i] bit. • Reading this register yields the current state of the DSTART[i] bits. This mechanism enables the BP to pipeline two DSTART commands per channel.

7.2.9.5 DMA Request Error Register (SDMABP_EVTERR)

Address: 30BD_0000h base + 28h offset = 30BD_0028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CHNERR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_EVTERR field descriptions

Field	Description
CHNERR	This register is used by the SDMA to warn the BP when an incoming DMA request was detected; it then triggers a channel that is already pending or being serviced, which may mean there is an overflow of data for that channel. An interrupt is sent to the BP if the corresponding channel bit is set in the INTRMASK register. • This is a "write-ones" register for the scheduler. It is only able to set the flags. The flags are cleared when the register is read by the BP or during an SDMA reset. • The CHNERR[i] bit is set when a DMA request that triggers channel *i* is received through the corresponding input pins and the EP[i] bit is already set. The EVTERR[i] bit is unaffected if the BP tries to set the EP[i] bit when that EP[i] bit is already set.

7.2.9.6 Channel DSP Interrupt Mask (SDMABP_INTRMASK)

Address: 30BD_0000h base + 2Ch offset = 30BD_002Ch

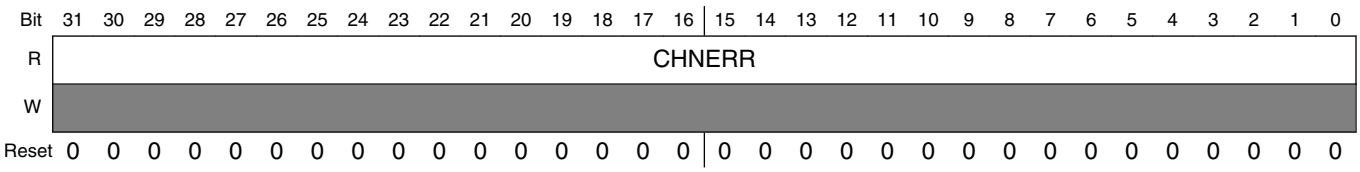
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DIMASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_INTRMASK field descriptions

Field	Description
DIMASK	The Interrupt Mask Register contains 32 interrupt generation mask bits. If bit DIMASK[i] is set, the DI[i] bit is set and an interrupt is sent to the BP when a DMA request error is detected on channel *i* (for example, EVTERR[i] is set).

7.2.9.7 DMA Request Error Register (SDMABP_EVTERRDBG)

Address: 30BD_0000h base + 34h offset = 30BD_0034h



SDMABP_EVTERRDBG field descriptions

Field	Description
CHNERR	This register is the same as EVTERR except reading it does not clear its contents. This address is meant to be used in debug mode. The BP OnCE may check this register value without modifying it.

7.2.10 SDMA Internal (Core) Memory Map and Internal Register Definitions

The actual SDMA memory mapped registers are summarized in the following sections; for peripherals' memory maps, refer to the respective chapters.

The following definitions serve as a key for the SDMA internal register summary.

SDMACORE memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BD_0000	ARM platform Channel 0 Pointer (SDMACORE_MC0PTR)	32	R	0000_0000h	7.2.10.1/ 1432
30BD_0008	Current Channel Pointer (SDMACORE_CCPTR)	32	R	0000_0000h	7.2.10.2/ 1433
30BD_000C	Current Channel Register (SDMACORE_CCR)	32	R	0000_0000h	7.2.10.3/ 1433
30BD_0010	Highest Pending Channel Register (SDMACORE_NCR)	32	R	0000_0000h	7.2.10.4/ 1434
30BD_0014	External DMA Requests Mirror (SDMACORE_EVENTS)	32	R	0000_0000h	7.2.10.5/ 1435
30BD_0018	Current Channel Priority (SDMACORE_CCPRI)	32	R	0000_0000h	7.2.10.6/ 1436
30BD_001C	Next Channel Priority (SDMACORE_NCPRI)	32	R	0000_0000h	7.2.10.7/ 1436

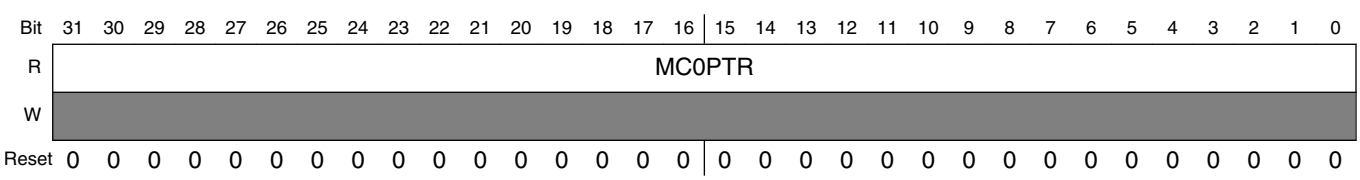
Table continues on the next page...

SDMACORE memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BD_0020	OnCE Event Cell Counter (SDMACORE_ECOUNT)	32	R/W	0000_0000h	7.2.10.8/ 1437
30BD_0024	OnCE Event Cell Control Register (SDMACORE_ECTL)	32	R/W	0000_0000h	7.2.10.9/ 1437
30BD_0028	OnCE Event Address Register A (SDMACORE_EAA)	32	R/W	0000_0000h	7.2.10.10/ 1439
30BD_002C	OnCE Event Cell Address Register B (SDMACORE_EAB)	32	R/W	0000_0000h	7.2.10.11/ 1439
30BD_0030	OnCE Event Cell Address Mask (SDMACORE_EAM)	32	R/W	0000_0000h	7.2.10.12/ 1439
30BD_0034	OnCE Event Cell Data Register (SDMACORE_ED)	32	R/W	0000_0000h	7.2.10.13/ 1440
30BD_0038	OnCE Event Cell Data Mask (SDMACORE_EDM)	32	R/W	0000_0000h	7.2.10.14/ 1440
30BD_003C	OnCE Real-Time Buffer (SDMACORE_RTB)	32	R/W	0000_0000h	7.2.10.15/ 1441
30BD_0040	OnCE Trace Buffer (SDMACORE_TB)	32	R	0000_0000h	7.2.10.16/ 1441
30BD_0044	OnCE Status (SDMACORE_OSTAT)	32	R	0000_0000h	7.2.10.17/ 1442
30BD_0048	Channel 0 Boot Address (SDMACORE_MCHN0ADDR)	32	R	0000_0000h	7.2.10.18/ 1444
30BD_004C	ENDIAN Status Register (SDMACORE_ENDIANNESS)	32	R	0000_0001h	7.2.10.19/ 1445
30BD_0054	Lock Status Register (SDMACORE_SDMA_LOCK)	32	R	0000_0000h	7.2.10.20/ 1446
30BD_0058	External DMA Requests Mirror #2 (SDMACORE_EVENTS2)	32	R	0000_0000h	7.2.10.21/ 1446

7.2.10.1 ARM platform Channel 0 Pointer (SDMACORE_MC0PTR)

Address: 30BD_0000h base + 0h offset = 30BD_0000h



SDMACORE_MC0PTR field descriptions

Field	Description
MC0PTR	Contains the address-in the ARM platform memory space-of the initial SDMA context and scripts that are loaded by the SDMA boot script running on channel 0.

7.2.10.2 Current Channel Pointer (SDMACORE_CCPTR)

Address: 30BD_0000h base + 8h offset = 30BD_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																CCPTR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_CCPTR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
CCPTR	Contains the start address of the context data for the current channel: Its value is *CONTEXT_BASE* + 24* *CCR* or *CONTEXT_BASE* + 32* *CCR* where CONTEXT_BASE = 0x0800. The value 24 or 32 is selected according to the programmed channel scratch RAM size in the register shown in Channel 0 Boot Address (SDMAARM_CHN0ADDR) .

7.2.10.3 Current Channel Register (SDMACORE_CCR)

Address: 30BD_0000h base + Ch offset = 30BD_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																											CCR				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_CCR field descriptions

Field	Description
31–5 Reserved	This read-only field is reserved and always has the value 0.
CCR	Contains the number of the current running channel whose context is installed. In the case that the SDMA has finished running the channel and has entered sleep state, CCR will indicate the previous running channel. The PST bits in the OSTAT register indicate when the SDMA is in sleep state.

7.2.10.4 Highest Pending Channel Register (SDMACORE_NCR)

Address: 30BD_0000h base + 10h offset = 30BD_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																											NCR				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_NCR field descriptions

Field	Description
31–5 Reserved	This read-only field is reserved and always has the value 0.
NCR	Contains the number of the pending channel that the scheduler has selected to run next.

7.2.10.5 External DMA Requests Mirror (SDMACORE_EVENTS)

NOTE

This register is very useful in the case of DMA requests that are active when a peripheral FIFO level is above the programmed watermark. The activation of the DMA request (rising edge) is detected by the SDMA logic and it can enable one or several channels. One of the channels accesses the peripheral and reads or writes a number of data that matches the watermark level (for example, if the watermark is four words, the channel reads or writes four words).

If the channel is effectively executed long after the DMA request was received, reading or writing the watermark number of data may not be sufficient to reset the DMA request (for example, if the FIFO watermark is four and at the channel execution it already contains nine pieces of data). This means no new rising edge may be detected by the SDMA, although there still remains transfers to perform. Therefore, if the channel were terminated at that time, it would not be restarted, causing potential overrun or underrun of the peripheral.

The proposed mechanism is for the channel to check this register after it has performed the "watermark" number of accesses to the peripheral. If the bit for the DMA request that triggers this channel is set, it means there is still another watermark number of data to transfer. This goes on until the bit is cleared. The same script can be used for multiple channels that require this behavior. The script can determine its channel number from the CCR register and infer the corresponding DMA request bit to check. It needs a reference table that is coherent with the request-channel matrix that the ARM platform programmed.

Address: 30BD_0000h base + 14h offset = 30BD_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EVENTS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_EVENTS field descriptions

Field	Description
EVENTS	Reflects the status of the SDMA's external DMA requests. It is meant to allow any channel to monitor the states of these SDMA inputs. This register displays EVENTS 0-31. The EVENTS2 register displays events 32-47.

7.2.10.6 Current Channel Priority (SDMACORE_CCPRI)

Address: 30BD_0000h base + 18h offset = 30BD_0018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																													CCPRI		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_CCPRI field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
CCPRI	Contains the 3-bit priority of the channel whose context is installed. It is 0 when no channel is running. **NOTE:** 1-7 current channel priority 0 no running channel

7.2.10.7 Next Channel Priority (SDMACORE_NCPRI)

Address: 30BD_0000h base + 1Ch offset = 30BD_001Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																													NCPRI		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_NCPRI field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
NCPRI	Contains the 3-bit priority of the channel the scheduler has selected to run next. It is 0 when no other channel is pending.

7.2.10.8 OnCE Event Cell Counter (SDMACORE_ECOUNT)

Address: 30BD_0000h base + 20h offset = 30BD_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																ECOUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_ECOUNT field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
ECOUNT	The event cell counter contains the number of times minus one that an event detection must occur before generating a debug request. • This register should be written before any attempt to use the event detection counter during an event detection process. • The counter is cleared on a JTAG reset.

7.2.10.9 OnCE Event Cell Control Register (SDMACORE_ECTL)

Address: 30BD_0000h base + 24h offset = 30BD_0024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		EN	CNT	ECTC[1:0]		DTC[1:0]		ATC[1:0]		ABTC[1:0]		AATC[1:0]		ATS[1:0]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_ECTL field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
13 EN	Event Cell Enable. If the EN bit is set, the event cell is allowed to generate debug requests (the cell is awakened). If it is cleared, the event detection unit is disabled and no hardware breakpoint is generated, but matching conditions are still reflected on the emulation pin. 0 Cell is disabled. 1 Cell is enabled.
12 CNT	Event Counter Enable. The event counter enable bit determines if the cell counter is used during the event detection. In order to use the event counter during an event detection process, the event cell counter register should be loaded with a value equal to the number of times minus one that an event occurs before a debug request is sent. After every event detection, the counter is decreased. When the counter reaches

Table continues on the next page...

SDMACORE_ECTL field descriptions (continued)

Field	Description
	the value 0, the event detection cell sends a debug request to the core. The event counter register should be written and the EN bit should be set before each new event detection process uses the event counter. 0 Counter is disabled. 1 Counter is enabled.
11–10 ECTC[1:0]	The event cell trigger condition bits select the combination of address and data matching conditions that generate the final address/data condition. During program execution, if this event cell trigger condition goes to 1, a debug request is sent to the SDMA. The EN bit must be set to enable the debug request generation. 00 address ONLY 01 data ONLY 10 address AND data 11 address OR data
9–8 DTC[1:0]	The data trigger condition bits define when data is considered matching after comparison with the data register of the event detection unit. The operations are performed on unsigned values. 00 equal 01 not equal 10 greater than 11 less than
7–6 ATC[1:0]	The address trigger condition bits select how the two address conditions (addressA and addressB) are combined to define the global address matching condition. The supported combinations are described, as follows. 00 addressA ONLY 01 addrA AND addrB 10 addrA OR addrB 11 reserved
5–4 ABTC[1:0]	The Address B Trigger Condition (ABTC) controls the operations performed by address comparator B. All operations are performed on unsigned values. This comparator B outputs the addressB condition. 00 equal 01 not equal 10 greater than 11 less than
3–2 AATC[1:0]	The Address A Trigger Condition (AATC) controls the operations performed by address comparator A. All operations are performed on unsigned values. This comparator A outputs the addressA condition. 00 equal 01 not equal 10 greater than 11 less than
ATS[1:0]	The access type select bits define the memory access type required on the SDMA memory bus. 00 read ONLY 01 write ONLY 10 read or write 11 -

7.2.10.10 OnCE Event Address Register A (SDMACORE_EAA)

Address: 30BD_0000h base + 28h offset = 30BD_0028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																EAA															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_EAA field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EAA	Event Cell Address Register A computes an address A condition. It is cleared on a JTAG reset.

7.2.10.11 OnCE Event Cell Address Register B (SDMACORE_EAB)

Address: 30BD_0000h base + 2Ch offset = 30BD_002Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																EAB															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_EAB field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EAB	Event Cell Address Register B computes an address B condition. It is cleared on a JTAG reset.

7.2.10.12 OnCE Event Cell Address Mask (SDMACORE_EAM)

Address: 30BD_0000h base + 30h offset = 30BD_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																EAM															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_EAM field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EAM	The Event Cell Address Mask contains a user-defined address mask value. This mask is applied to the address value latched from the memory address bus before performing the address comparison. **NOTE:** There is a common address mask value for both address comparators. If bit *i* of this register is set, then bit *i* of the address value latched from the memory bus does not influence the result of the address comparison. The register is cleared on a JTAG reset.

7.2.10.13 OnCE Event Cell Data Register (SDMACORE_ED)

Address: 30BD_0000h base + 34h offset = 30BD_0034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ED																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_ED field descriptions

Field	Description
ED	The event cell data register contains a user defined data value. This data value is an input for the data comparator which generates the data condition. It is cleared on a JTAG reset.

7.2.10.14 OnCE Event Cell Data Mask (SDMACORE_EDM)

Address: 30BD_0000h base + 38h offset = 30BD_0038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EDM																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_EDM field descriptions

Field	Description
EDM	The event cell data mask register contains the user-defined data mask value. • This mask is applied to the data value latched from the memory bus before performing the data comparison. • Setting bit *i* of the event cell data mask register means that bit *i* of the data value latched from the address bus does not influence the result of the data comparison. • The data mask is cleared on a JTAG reset.

7.2.10.15 OnCE Real-Time Buffer (SDMACORE_RTB)

Address: 30BD_0000h base + 3Ch offset = 30BD_003Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RTB																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_RTB field descriptions

Field	Description
RTB	The Real Time Buffer register stores and retrieves run time information without putting the SDMA in debug mode. Writing to that register triggers a pulse on a specific real-time debug pin whose connection depends on the chip implementation. The RTB value can be accessed by the OnCE under ARM platform or JTAG control using the rbuffer command.

7.2.10.16 OnCE Trace Buffer (SDMACORE_TB)

Address: 30BD_0000h base + 40h offset = 30BD_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			TBF	TADDR											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TADDR		CHFADDR													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_TB field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28 TBF	The Trace Buffer Flag is set when the buffer contains the addresses of a valid change of flow. The contents of the buffer should be ignored otherwise. 0 Invalid information 1 Valid information
27–14 TADDR	The target address is the address taken after the execution of the change of flow instruction.
CHFADDR	The change of flow address is the address where the change of flow is taken when executing a change of flow instruction.

7.2.10.17 OnCE Status (SDMACORE_OSTAT)

Address: 30BD_0000h base + 44h offset = 30BD_0044h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PST[3:0]				RCV	EDR	ODR	SWB	MST	0				ECDR[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_OSTAT field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–12 PST[3:0]	The Processor Status bits reflect the state of the SDMA RISC engine. • The "Program" state is the usual instruction execution cycle. • The "Data" state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The "Change of Flow" state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel-switching instructions). • The "Change of Flow in Loop" state is used when an error causes a hardware loop exit. • The "Debug" state means the SDMA is in debug mode. • The "Functional Unit" state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In "Sleep" modes, no script is running (this is the RISC engine idle state). The "after Reset" is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 will be executed (boot operation). • The "in Sleep" states are the same as above except they do not have any corresponding channel. They are used when entering debug mode after reset; the reason is that it is necessary to return to the "Sleep after Reset" state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Save 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change Flow Loop Sleep 12 Debug in Sleep 13 Functional Unit in Sleep

Table continues on the next page...

SDMACORE_OSTAT field descriptions (continued)

Field	Description
	14 Sleep after Reset 15 Restore
11 RCV	After each write access to the real time buffer (RTB), the RCV bit is set. This bit is cleared after execution of an `rbuffer` command and on a JTAG reset.
10 EDR	This flag is raised when the SDMA has entered debug mode after an external debug request.
9 ODR	This flag is raised when the SDMA has entered debug mode after a OnCE debug request.
8 SWB	This flag is raised when the SDMA has entered debug mode after a software breakpoint.
7 MST	This flag is raised when the OnCE is controlled from the ARM platform peripheral interface. 0 JTAG interface controls the OnCE. 1 ARM platform peripheral interface controls the OnCE.
6–3 Reserved	This read-only field is reserved and always has the value 0.
ECDR[2:0]	Event Cell Debug Request. If the debug request comes from the event cell, the reason for entering debug mode is given by the EDR bits. The encoding of the EDR bits is useful to find out more precisely why the debug request was generated. A debug request from an event cell is generated for a specific combination of the addressA, addressB, and data conditions; the value of those fields is given by the EDR bits. If all three bits of the EDR are reset, then it did not generate any debug request. If the cell did generate a debug request, then at least one EDR bit is set; the meaning of the encoding is as follows: 0 1 matched addressA condition 1 1 matched addressB condition 2 1 matched data condition

7.2.10.18 Channel 0 Boot Address (SDMACORE_MCHN0ADDR)

Address: 30BD_0000h base + 48h offset = 30BD_0048h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

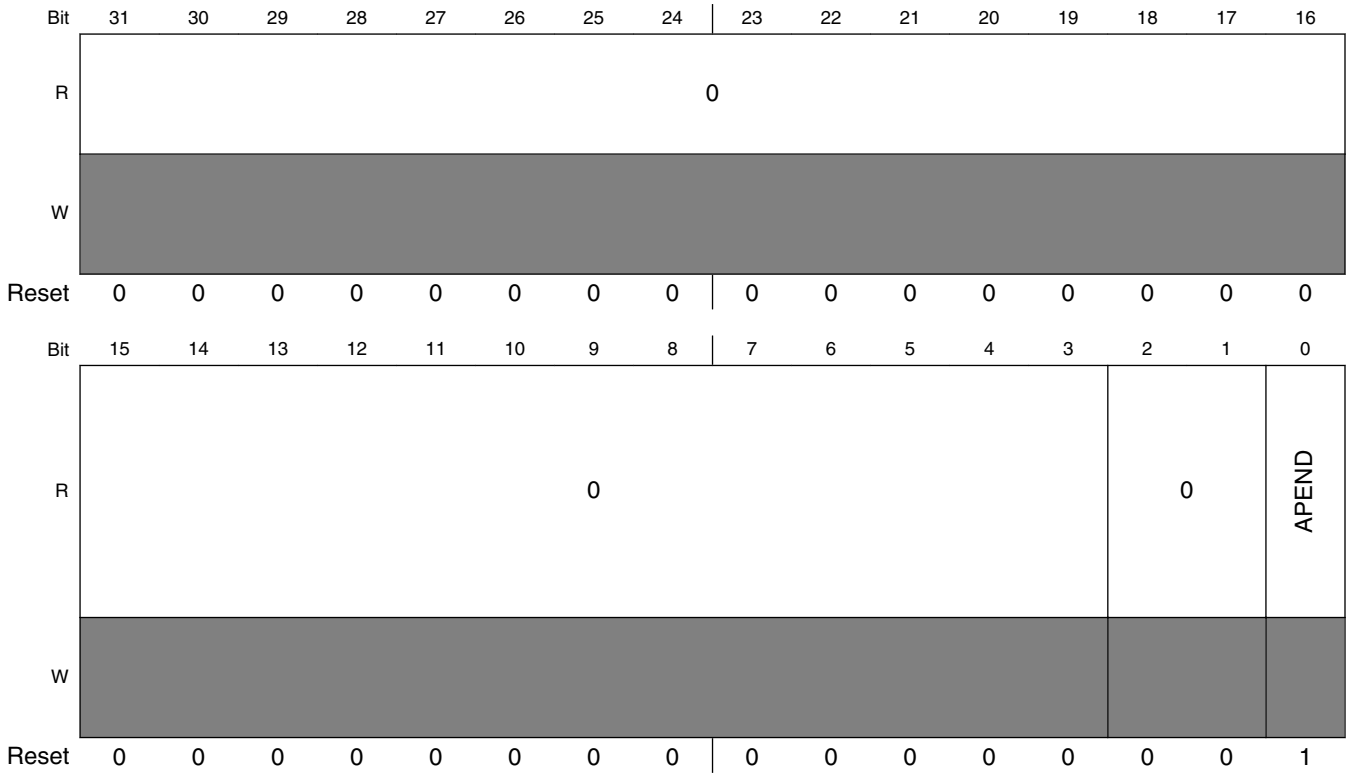
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	SMSZ	CHN0ADDR[13:0]													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_MCHN0ADDR field descriptions

Field	Description
31–15 Reserved	This read-only field is reserved and always has the value 0.
14 SMSZ	The bit 14 (Scratch Memory Size) determines if scratch memory must be available after every channel context. After reset, it is equal to 0, which defines a RAM space of 24 words for each channel. All of this area stores the channel context. By setting this bit, 32 words are reserved for every channel context, which gives eight additional words that can be used by the channel script to store any type of data. Those words are never erased by the context switching mechanism. 0 24 words per context 1 32 words per context
CHN0ADDR[13:0]	Contains the address of the channel 0 routine programmed by the ARM platform; it is loaded into a general register at the very start of the boot and the SDMA jumps to the address it contains. By default, it points to the standard boot routine in ROM.

7.2.10.19 ENDIAN Status Register (SDMACORE_ENDIANNESS)

Address: 30BD_0000h base + 4Ch offset = 30BD_004Ch

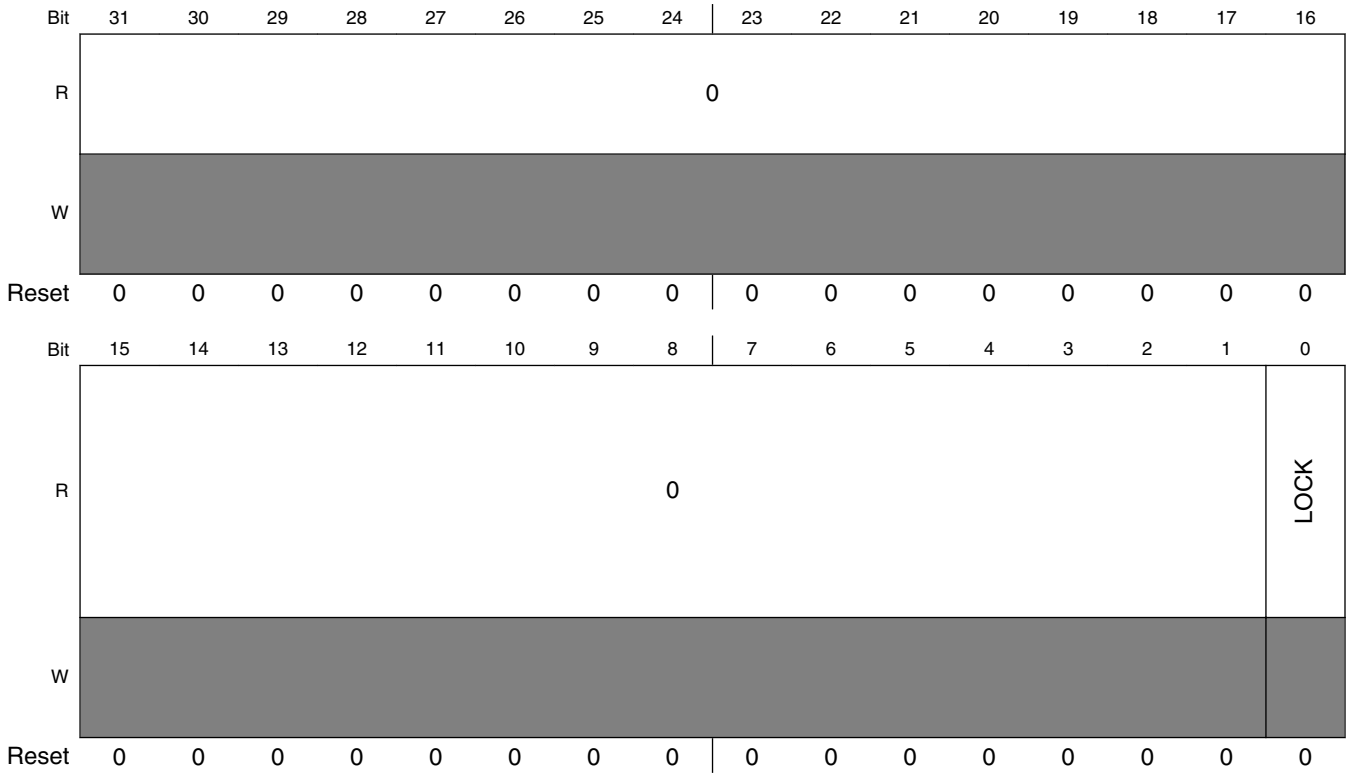


SDMACORE_ENDIANNESS field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
2–1 Reserved	This read-only field is reserved and always has the value 0.
0 APEND	APEND indicates the endian mode of the Peripheral and Burst DMA interfaces. This bit is tied to logic '1' indicating little-endian mode. 0 - ARM platform is in big-endian mode 1 - ARM platform is in little-endian mode

7.2.10.20 Lock Status Register (SDMACORE_SDMA_LOCK)

Address: 30BD_0000h base + 54h offset = 30BD_0054h

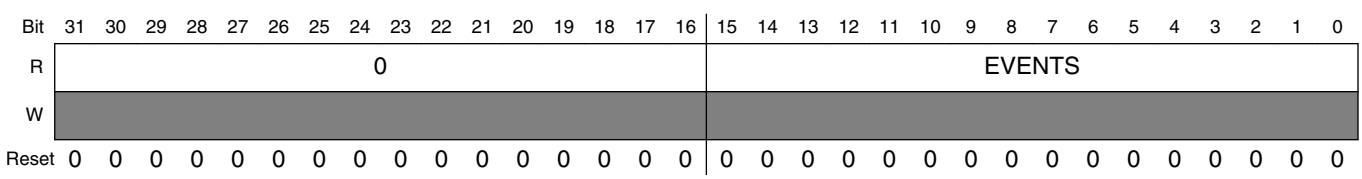


SDMACORE_SDMA_LOCK field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 LOCK	The LOCK bit reports the value of the LOCK bit in the SDMA_LOCK status register. SDMA software may use this value to determine if certain operations such as loading of new scripts is allowed. 0 - LOCK bit clear 1 - LOCK bit set

7.2.10.21 External DMA Requests Mirror #2 (SDMACORE_EVENTS2)

Address: 30BD_0000h base + 58h offset = 30BD_0058h



SDMACORE_EVENTS2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EVENTS	Reflects the status of the SDMA's external DMA requests. It is meant to allow any channel to monitor the states of these SDMA inputs. This register displays EVENTS 32-47. The separate EVENTS register displays events 0-31.

7.2.11 SDMA Peripheral Registers

Refer to the respective peripherals' chapters for more information.

Chapter 8
Chip IO and Pinmux

8.1 External Signals and Pin Multiplexing

8.1.1 Overview

The chip contains a limited number of pins, most of which have multiple signal options. These signal-to-pin and pin-to-signal options are selected by the input-output multiplexer called IOMUX. The IOMUX is also used to configure other pin characteristics, such as voltage level, drive strength, and hysteresis.

The muxing options table lists the external signals grouped by the module instance, the muxing options for each signal, and the registers used to route the signal to the chosen pad.

8.1.1.1 Muxing Options

Instance	Port	Pad	Mode
ADC1	ADC1_IN0	ADC1_IN0	No muxing
	ADC1_IN1	ADC1_IN1	No muxing
	ADC1_IN2	ADC1_IN2	No muxing
	ADC1_IN3	ADC1_IN3	No muxing
	ADC1_VDDA_1P8	ADC1_VDDA_1P8	No muxing
	ADC1_VREFH	ADC1_VREFH	No muxing
	ADC1_VREFL	ADC1_VREFL	No muxing
ADC2	ADC2_IN0	ADC2_IN0	No muxing
	ADC2_IN1	ADC2_IN1	No muxing
	ADC2_IN2	ADC2_IN2	No muxing
	ADC2_IN3	ADC2_IN3	No muxing
	ADC2_VDDA_1P8	ADC2_VDDA_1P8	No muxing

Table continues on the next page...

Instance	Port	Pad	Mode
	ADC2_VREFH	ADC2_VREFH	No muxing
	ADC2_VREFL	ADC2_VREFL	No muxing
ARM PLATFORM	ARM_PLATFORM_EVENTI	LCD1_RESET	ALT2
	ARM_PLATFORM_EVENTO	LCD1_DATA18	ALT2
	ARM_PLATFORM_TRACE_CLK	LCD1_DATA16	ALT2
	ARM_PLATFORM_TRACE_CTL	LCD1_DATA17	ALT2
	ARM_PLATFORM_TRACE00	LCD1_DATA00	ALT2
	ARM_PLATFORM_TRACE01	LCD1_DATA01	ALT2
	ARM_PLATFORM_TRACE02	LCD1_DATA02	ALT2
	ARM_PLATFORM_TRACE03	LCD1_DATA03	ALT2
	ARM_PLATFORM_TRACE04	LCD1_DATA04	ALT2
	ARM_PLATFORM_TRACE05	LCD1_DATA05	ALT2
	ARM_PLATFORM_TRACE06	LCD1_DATA06	ALT2
	ARM_PLATFORM_TRACE07	LCD1_DATA07	ALT2
	ARM_PLATFORM_TRACE08	LCD1_DATA08	ALT2
	ARM_PLATFORM_TRACE09	LCD1_DATA09	ALT2
	ARM_PLATFORM_TRACE10	LCD1_DATA10	ALT2
	ARM_PLATFORM_TRACE11	LCD1_DATA11	ALT2
	ARM_PLATFORM_TRACE12	LCD1_DATA12	ALT2
	ARM_PLATFORM_TRACE13	LCD1_DATA13	ALT2
	ARM_PLATFORM_TRACE14	LCD1_DATA14	ALT2
	ARM_PLATFORM_TRACE15	LCD1_DATA15	ALT2
CCM	CCM_CLK1_N	CCM_CLK1_N	No muxing
	CCM_CLK1_P	CCM_CLK1_P	No muxing
	CCM_CLK2	CCM_CLK2	No muxing
	CCM_PMIC_STBY_REQ	CCM_PMIC_STBY_REQ	No muxing
	CCM_CLKO1	GPIO1_IO02	ALT5
		SD1_CD_B	ALT6
	CCM_CLKO2	GPIO1_IO03	ALT5
		SD1_WP	ALT6
	CCM_ENET1_REF_CLK	ENET1_TX_CLK	ALT1
		GPIO1_IO02	ALT2
		GPIO1_IO12	ALT2
		I2C1_SDA	ALT4
	CCM_ENET2_REF_CLK	EPDC1_BDR0	ALT3
		GPIO1_IO03	ALT2
		GPIO1_IO13	ALT2
		I2C2_SCL	ALT4
	CCM_ENET3_REF_CLK	GPIO1_IO01	ALT2

Table continues on the next page...

Instance	Port	Pad	Mode
		GPIO1_IO09	ALT2
		I2C2_SDA	ALT4
	CCM_EXT_CLK1	ENET1_TX_CLK	ALT6
		GPIO1_IO12	ALT5
		SD1_DATA0	ALT6
	CCM_EXT_CLK2	ENET1_RX_CLK	ALT6
		GPIO1_IO13	ALT5
		SD1_DATA1	ALT6
	CCM_EXT_CLK3	ENET1_CRS	ALT6
		GPIO1_IO14	ALT5
		SD1_DATA2	ALT6
	CCM_EXT_CLK4	ENET1_COL	ALT6
		GPIO1_IO15	ALT5
		SD1_DATA3	ALT6
	CCM_PMIC_READY	GPIO1_IO09	ALT5
		GPIO1_IO13	ALT4
		SAI1_MCLK	ALT3
		UART1_RXD	ALT2
CSI	CSI_DATA00	LCD1_DATA17	ALT3
	CSI_DATA01	LCD1_DATA16	ALT3
	CSI_DATA02	ECSPI1_SCLK	ALT3
		LCD1_DATA15	ALT3
	CSI_DATA03	ECSPI1_MOSI	ALT3
		LCD1_DATA14	ALT3
	CSI_DATA04	ECSPI1_MISO	ALT3
		LCD1_DATA13	ALT3
	CSI_DATA05	ECSPI1_SS0	ALT3
		LCD1_DATA12	ALT3
	CSI_DATA06	ECSPI2_SCLK	ALT3
		LCD1_DATA11	ALT3
	CSI_DATA07	ECSPI2_MOSI	ALT3
		LCD1_DATA10	ALT3
	CSI_DATA08	ECSPI2_MISO	ALT3
		LCD1_DATA09	ALT3
	CSI_DATA09	ECSPI2_SS0	ALT3
		LCD1_DATA08	ALT3
	CSI_DATA10	LCD1_DATA23	ALT3
	CSI_DATA11	LCD1_DATA22	ALT3
	CSI_DATA12	LCD1_DATA21	ALT3
	CSI_DATA13	LCD1_DATA20	ALT3
	CSI_DATA14	LCD1_DATA19	ALT3

Table continues on the next page...

Instance	Port	Pad	Mode
	CSI_DATA15	LCD1_DATA18	ALT3
	CSI_DATA16	LCD1_CLK	ALT3
	CSI_DATA17	LCD1_ENABLE	ALT3
	CSI_DATA18	LCD1_HSYNC	ALT3
	CSI_DATA19	LCD1_VSYNC	ALT3
	CSI_DATA20	LCD1_DATA00	ALT3
	CSI_DATA21	LCD1_DATA01	ALT3
	CSI_DATA22	LCD1_DATA02	ALT3
	CSI_DATA23	LCD1_DATA03	ALT3
	CSI_FIELD	LCD1_RESET	ALT3
	CSI_HSYNC	I2C3_SDA	ALT3
		LCD1_DATA05	ALT3
	CSI_MCLK	I2C4_SDA	ALT3
		LCD1_DATA07	ALT3
	CSI_PIXCLK	I2C4_SCL	ALT3
		LCD1_DATA06	ALT3
	CSI_VSYNC	I2C3_SCL	ALT3
		LCD1_DATA04	ALT3
ECSPI1	ECSPI1_MISO	ECSPI1_MISO	ALT0
		UART3_RXD	ALT3
	ECSPI1_MOSI	ECSPI1_MOSI	ALT0
		UART3_TXD	ALT3
	ECSPI1_RDY	UART2_TXD	ALT3
	ECSPI1_SCLK	ECSPI1_SCLK	ALT0
		UART3_RTS	ALT3
	ECSPI1_SS0	ECSPI1_SS0	ALT0
		UART3_CTS	ALT3
	ECSPI1_SS1	UART1_RXD	ALT3
	ECSPI1_SS2	UART1_TXD	ALT3
	ECSPI1_SS3	UART2_RXD	ALT3
ECSPI2	ECSPI2_MISO	ECSPI2_MISO	ALT0
		ENET1_TDATA2	ALT2
	ECSPI2_MOSI	ECSPI2_MOSI	ALT0
		ENET1_RDATA3	ALT2
	ECSPI2_RDY	ENET1_TDATA1	ALT2
	ECSPI2_SCLK	ECSPI2_SCLK	ALT0
		ENET1_RDATA2	ALT2
	ECSPI2_SS0	ECSPI2_SS0	ALT0
		ENET1_TDATA3	ALT2
	ECSPI2_SS1	ENET1_RX_CTL	ALT2
	ECSPI2_SS2	ENET1_RXC	ALT2

Table continues on the next page...

Instance	Port	Pad	Mode
	ECSPI2_SS3	ENET1_TDATA0	ALT2
ECSPI3	ECSPI3_MISO	I2C1_SCL	ALT3
		SAI2_TXFS	ALT1
	ECSPI3_MOSI	I2C1_SDA	ALT3
		SAI2_TXC	ALT1
	ECSPI3_RDY	SD2_RESET_B	ALT3
	ECSPI3_SCLK	I2C2_SCL	ALT3
		SAI2_RXD	ALT1
	ECSPI3_SS0	I2C2_SDA	ALT3
		SAI2_TXD	ALT1
	ECSPI3_SS1	SD1_DATA3	ALT3
	ECSPI3_SS2	SD2_CD_B	ALT3
	ECSPI3_SS3	SD2_WP	ALT3
ECSPI4	ECSPI4_MISO	LCD1_CLK	ALT1
		SD1_CD_B	ALT3
		SD3_CLK	ALT2
	ECSPI4_MOSI	LCD1_ENABLE	ALT1
		SD1_WP	ALT3
		SD3_CMD	ALT2
	ECSPI4_RDY	SD1_DATA2	ALT3
	ECSPI4_SCLK	LCD1_HSYNC	ALT1
		SD1_RESET_B	ALT3
		SD3_DATA1	ALT2
	ECSPI4_SS0	LCD1_VSYNC	ALT1
		SD1_CLK	ALT3
		SD3_DATA0	ALT2
	ECSPI4_SS1	SD1_CMD	ALT3
	ECSPI4_SS2	SD1_DATA0	ALT3
	ECSPI4_SS3	SD1_DATA1	ALT3
EIM	EIM_ACLK_FREERUN	LCD1_DATA17	ALT4
	EIM_AD00	EPDC1_DATA00	ALT4
	EIM_AD01	EPDC1_DATA01	ALT4
	EIM_AD02	EPDC1_DATA02	ALT4
	EIM_AD03	EPDC1_DATA03	ALT4
	EIM_AD04	EPDC1_DATA04	ALT4
	EIM_AD05	EPDC1_DATA05	ALT4
	EIM_AD06	EPDC1_DATA06	ALT4
	EIM_AD07	EPDC1_DATA07	ALT4
	EIM_AD08	EPDC1_BDR1	ALT4
	EIM_AD09	EPDC1_PWRCOM	ALT4
	EIM_AD10	EPDC1_SDCLK	ALT4

Table continues on the next page...

Instance	Port	Pad	Mode
	EIM_AD11	EPDC1_SDLE	ALT4
	EIM_AD12	EPDC1_SDOE	ALT4
	EIM_AD13	EPDC1_SDSHR	ALT4
	EIM_AD14	EPDC1_SDCE0	ALT4
	EIM_AD15	EPDC1_SDCE1	ALT4
	EIM_ADDR16	EPDC1_SDCE2	ALT4
	EIM_ADDR17	EPDC1_SDCE3	ALT4
	EIM_ADDR18	EPDC1_GDCLK	ALT4
	EIM_ADDR19	EPDC1_GDOE	ALT4
	EIM_ADDR20	EPDC1_GDRL	ALT4
	EIM_ADDR21	EPDC1_GDSP	ALT4
	EIM_ADDR22	EPDC1_BDR0	ALT4
	EIM_ADDR23	LCD1_DATA20	ALT4
	EIM_ADDR24	LCD1_DATA21	ALT4
	EIM_ADDR25	LCD1_DATA22	ALT4
	EIM_ADDR26	LCD1_DATA23	ALT4
	EIM_BCLK	EPDC1_DATA11	ALT4
	EIM_CRE	LCD1_DATA16	ALT4
	EIM_CS0	EPDC1_DATA10	ALT4
	EIM_CS1	EPDC1_DATA15	ALT4
	EIM_CS2	LCD1_DATA18	ALT4
	EIM_CS3	LCD1_DATA19	ALT4
	EIM_DATA00	LCD1_DATA00	ALT4
	EIM_DATA01	LCD1_DATA01	ALT4
	EIM_DATA02	LCD1_DATA02	ALT4
	EIM_DATA03	LCD1_DATA03	ALT4
	EIM_DATA04	LCD1_DATA04	ALT4
	EIM_DATA05	LCD1_DATA05	ALT4
	EIM_DATA06	LCD1_DATA06	ALT4
	EIM_DATA07	LCD1_DATA07	ALT4
	EIM_DATA08	LCD1_DATA08	ALT4
	EIM_DATA09	LCD1_DATA09	ALT4
	EIM_DATA10	LCD1_DATA10	ALT4
	EIM_DATA11	LCD1_DATA11	ALT4
	EIM_DATA12	LCD1_DATA12	ALT4
	EIM_DATA13	LCD1_DATA13	ALT4
	EIM_DATA14	LCD1_DATA14	ALT4
	EIM_DATA15	LCD1_DATA15	ALT4
	EIM_DTACK_B	LCD1_RESET	ALT4
	EIM_EB0	EPDC1_DATA14	ALT4
	EIM_EB1	EPDC1_PWRSTAT	ALT4

Table continues on the next page...

Instance	Port	Pad	Mode
	EIM_LBA_B	EPDC1_DATA12	ALT4
	EIM_OE	EPDC1_DATA08	ALT4
	EIM_RW	EPDC1_DATA09	ALT4
	EIM_WAIT	EPDC1_DATA13	ALT4
ELCDIF	LCD_BUSY	EPDC1_DATA08	ALT7
		EPDC1_GDSP	ALT6
	LCD_CLK	EPDC1_DATA00	ALT7
		EPDC1_SDCLK	ALT6
		LCD1_CLK	ALT0
	LCD_CS	EPDC1_BDR0	ALT6
		EPDC1_DATA13	ALT7
	LCD_DATA00	EPDC1_DATA00	ALT6
		EPDC1_DATA09	ALT7
		LCD1_DATA00	ALT0
	LCD_DATA01	EPDC1_DATA01	ALT6
		EPDC1_DATA11	ALT7
		LCD1_DATA01	ALT0
	LCD_DATA02	EPDC1_DATA02	ALT6
		EPDC1_SDCE3	ALT7
		LCD1_DATA02	ALT0
	LCD_DATA03	EPDC1_DATA03	ALT6
		EPDC1_SDCE2	ALT7
		LCD1_DATA03	ALT0
	LCD_DATA04	EPDC1_DATA04	ALT6
		EPDC1_SDCE1	ALT7
		LCD1_DATA04	ALT0
	LCD_DATA05	EPDC1_DATA05	ALT6
		EPDC1_SDCE0	ALT7
		LCD1_DATA05	ALT0
	LCD_DATA06	EPDC1_BDR1	ALT7
		EPDC1_DATA06	ALT6
		LCD1_DATA06	ALT0
	LCD_DATA07	EPDC1_BDR0	ALT7
		EPDC1_DATA07	ALT6
		LCD1_DATA07	ALT0
	LCD_DATA08	EPDC1_DATA08	ALT6
		EPDC1_SDLE	ALT7
		LCD1_DATA08	ALT0
	LCD_DATA09	EPDC1_DATA09	ALT6
		EPDC1_DATA10	ALT7
		LCD1_DATA09	ALT0

Table continues on the next page...

Instance	Port	Pad	Mode
	LCD_DATA10	EPDC1_DATA10	ALT6
		EPDC1_SDSHR	ALT7
		LCD1_DATA10	ALT0
	LCD_DATA11	EPDC1_DATA11	ALT6
		EPDC1_PWRCOM	ALT7
		LCD1_DATA11	ALT0
	LCD_DATA12	EPDC1_DATA12	ALT6
		EPDC1_PWRSTAT	ALT7
		LCD1_DATA12	ALT0
	LCD_DATA13	ECSPI2_SCLK	ALT4
		EPDC1_DATA13	ALT6
		LCD1_DATA13	ALT0
	LCD_DATA14	ECSPI2_MOSI	ALT4
		EPDC1_DATA14	ALT6
		LCD1_DATA14	ALT0
	LCD_DATA15	ECSPI2_MISO	ALT4
		EPDC1_DATA15	ALT6
		LCD1_DATA15	ALT0
	LCD_DATA16	EPDC1_GDCLK	ALT7
		EPDC1_SDLE	ALT6
		LCD1_DATA16	ALT0
	LCD_DATA17	EPDC1_GDSP	ALT7
		EPDC1_SDOE	ALT6
		LCD1_DATA17	ALT0
	LCD_DATA18	EPDC1_GDOE	ALT7
		EPDC1_SDSHR	ALT6
		LCD1_DATA18	ALT0
	LCD_DATA19	EPDC1_GDRL	ALT7
		EPDC1_SDCE0	ALT6
		LCD1_DATA19	ALT0
	LCD_DATA20	EPDC1_SDCE1	ALT6
		EPDC1_SDCLK	ALT7
		LCD1_DATA20	ALT0
	LCD_DATA21	EPDC1_DATA12	ALT7
		EPDC1_SDCE2	ALT6
		LCD1_DATA21	ALT0
	LCD_DATA22	EPDC1_DATA14	ALT7
		EPDC1_SDCE3	ALT6
		LCD1_DATA22	ALT0
	LCD_DATA23	EPDC1_GDCLK	ALT6
		EPDC1_SDOE	ALT7

Table continues on the next page...

Instance	Port	Pad	Mode
		LCD1_DATA23	ALT0
	LCD_ENABLE	EPDC1_BDR1	ALT6
		EPDC1_DATA01	ALT7
		LCD1_ENABLE	ALT0
	LCD_HSYNC	EPDC1_DATA03	ALT7
		EPDC1_PWRCOM	ALT6
		LCD1_HSYNC	ALT0
	LCD_RD_E	EPDC1_GDRL	ALT6
	LCD_RS	ECSPI2_SS0	ALT4
		LCD1_RESET	ALT0
	LCD_VSYNC	EPDC1_DATA02	ALT7
		EPDC1_PWRSTAT	ALT6
		LCD1_VSYNC	ALT0
	LCD_WR_RWN	EPDC1_DATA15	ALT7
		EPDC1_GDOE	ALT6
ENET1	ENET1_1588_EVENT0_IN	UART3_RXD	ALT4
	ENET1_1588_EVENT0_OUT	UART3_TXD	ALT4
	ENET1_1588_EVENT1_IN	UART3_RTS	ALT4
	ENET1_1588_EVENT1_OUT	UART3_CTS	ALT4
	ENET1_1588_EVENT2_IN	LCD1_CLK	ALT2
	ENET1_1588_EVENT2_OUT	LCD1_DATA20	ALT2
	ENET1_1588_EVENT3_IN	LCD1_ENABLE	ALT2
	ENET1_1588_EVENT3_OUT	LCD1_DATA21	ALT2
	ENET1_COL	ENET1_COL	ALT0
	ENET1_CRS	ENET1_CRS	ALT0
	ENET1_MDC	GPIO1_IO11	ALT2
		SD2_WP	ALT1
		UART1_TXD	ALT6
	ENET1_MDIO	GPIO1_IO10	ALT2
		SD2_CD_B	ALT1
		UART1_RXD	ALT6
	ENET1_RX_CLK	ENET1_RX_CLK	ALT0
	ENET1_RX_ER	ENET1_RXC	ALT1
	ENET1_TX_CLK	ENET1_TX_CLK	ALT0
	ENET1_TX_ER	ENET1_TXC	ALT1
	RGMII1_RD0	ENET1_RDATA0	ALT0
	RGMII1_RD1	ENET1_RDATA1	ALT0
	RGMII1_RD2	ENET1_RDATA2	ALT0
	RGMII1_RD3	ENET1_RDATA3	ALT0
	RGMII1_RX_CTL	ENET1_RX_CTL	ALT0
	RGMII1_RXC	ENET1_RXC	ALT0

Table continues on the next page...

Instance	Port	Pad	Mode
	RGMII1_TD0	ENET1_TDATA0	ALT0
	RGMII1_TD1	ENET1_TDATA1	ALT0
	RGMII1_TD2	ENET1_TDATA2	ALT0
	RGMII1_TD3	ENET1_TDATA3	ALT0
	RGMII1_TX_CTL	ENET1_TX_CTL	ALT0
	RGMII1_TXC	ENET1_TXC	ALT0
FLEXCAN1	FLEXCAN1_RX	ENET1_RDATA2	ALT1
		GPIO1_IO12	ALT3
		I2C1_SCL	ALT2
		SAI1_RXD	ALT3
		SD3_DATA7	ALT4
	FLEXCAN1_TX	ENET1_RDATA3	ALT1
		GPIO1_IO13	ALT3
		I2C1_SDA	ALT2
		SAI1_TXC	ALT3
		SD3_DATA5	ALT4
FLEXCAN2	FLEXCAN2_RX	ENET1_TDATA2	ALT1
		GPIO1_IO14	ALT3
		I2C3_SCL	ALT2
		SAI1_TXFS	ALT3
		SD3_DATA4	ALT4
	FLEXCAN2_TX	ENET1_TDATA3	ALT1
		GPIO1_IO15	ALT3
		I2C3_SDA	ALT2
		SAI1_TXD	ALT3
		SD3_DATA6	ALT4
FTM1	FLEXTIMER1_CH0	EPDC1_SDOE	ALT1
		SD1_CD_B	ALT4
	FLEXTIMER1_CH1	EPDC1_SDSHR	ALT1
		SD1_WP	ALT4
	FLEXTIMER1_CH2	EPDC1_SDCE0	ALT1
		SD1_RESET_B	ALT4
	FLEXTIMER1_CH3	EPDC1_SDCE1	ALT1
		SD1_CLK	ALT4
	FLEXTIMER1_CH4	GPIO1_IO04	ALT2
		LCD1_DATA16	ALT1
	FLEXTIMER1_CH5	GPIO1_IO05	ALT2
		LCD1_DATA17	ALT1
	FLEXTIMER1_CH6	GPIO1_IO06	ALT2
		LCD1_DATA18	ALT1
	FLEXTIMER1_CH7	GPIO1_IO07	ALT2

Table continues on the next page...

Instance	Port	Pad	Mode
		LCD1_DATA19	ALT1
FTM2	FLEXTIMER2_CH0	EPDC1_GDCLK	ALT1
		SD1_CMD	ALT4
	FLEXTIMER2_CH1	EPDC1_GDOE	ALT1
		SD1_DATA0	ALT4
	FLEXTIMER2_CH2	EPDC1_GDRL	ALT1
		SD1_DATA1	ALT4
	FLEXTIMER2_CH3	EPDC1_GDSP	ALT1
		SD1_DATA2	ALT4
	FLEXTIMER2_CH4	LCD1_DATA20	ALT1
		SAI2_TXFS	ALT4
	FLEXTIMER2_CH5	LCD1_DATA21	ALT1
		SAI2_TXC	ALT4
	FLEXTIMER2_CH6	LCD1_DATA22	ALT1
		SAI2_RXD	ALT4
	FLEXTIMER2_CH7	LCD1_DATA23	ALT1
		SAI2_TXD	ALT4
	FLEXTIMER2_PHA	EPDC1_PWRCOM	ALT1
		GPIO1_IO10	ALT5
		SAI1_RXC	ALT4
		SD1_DATA3	ALT4
	FLEXTIMER2_PHB	EPDC1_PWRSTAT	ALT1
		GPIO1_IO11	ALT5
		SAI1_MCLK	ALT4
		SD2_CD_B	ALT4
GPIO1	GPIO1_IO00	GPIO1_IO00	ALT0
	GPIO1_IO01	GPIO1_IO01	ALT0
	GPIO1_IO02	GPIO1_IO02	ALT0
	GPIO1_IO03	GPIO1_IO03	ALT0
	GPIO1_IO04	GPIO1_IO04	ALT0
	GPIO1_IO05	GPIO1_IO05	ALT0
	GPIO1_IO06	GPIO1_IO06	ALT0
	GPIO1_IO07	GPIO1_IO07	ALT0
	GPIO1_IO08	GPIO1_IO08	ALT0
	GPIO1_IO09	GPIO1_IO09	ALT0
	GPIO1_IO10	GPIO1_IO10	ALT0
	GPIO1_IO11	GPIO1_IO11	ALT0
	GPIO1_IO12	GPIO1_IO12	ALT0
	GPIO1_IO13	GPIO1_IO13	ALT0
	GPIO1_IO14	GPIO1_IO14	ALT0
	GPIO1_IO15	GPIO1_IO15	ALT0

Table continues on the next page...

Instance	Port	Pad	Mode
GPIO2	GPIO2_IO00	EPDC1_DATA00	ALT5
	GPIO2_IO01	EPDC1_DATA01	ALT5
	GPIO2_IO02	EPDC1_DATA02	ALT5
	GPIO2_IO03	EPDC1_DATA03	ALT5
	GPIO2_IO04	EPDC1_DATA04	ALT5
	GPIO2_IO05	EPDC1_DATA05	ALT5
	GPIO2_IO06	EPDC1_DATA06	ALT5
	GPIO2_IO07	EPDC1_DATA07	ALT5
	GPIO2_IO08	EPDC1_DATA08	ALT5
	GPIO2_IO09	EPDC1_DATA09	ALT5
	GPIO2_IO10	EPDC1_DATA10	ALT5
	GPIO2_IO11	EPDC1_DATA11	ALT5
	GPIO2_IO12	EPDC1_DATA12	ALT5
	GPIO2_IO13	EPDC1_DATA13	ALT5
	GPIO2_IO14	EPDC1_DATA14	ALT5
	GPIO2_IO15	EPDC1_DATA15	ALT5
	GPIO2_IO16	EPDC1_SDCLK	ALT5
	GPIO2_IO17	EPDC1_SDLE	ALT5
	GPIO2_IO18	EPDC1_SDOE	ALT5
	GPIO2_IO19	EPDC1_SDSHR	ALT5
	GPIO2_IO20	EPDC1_SDCE0	ALT5
	GPIO2_IO21	EPDC1_SDCE1	ALT5
	GPIO2_IO22	EPDC1_SDCE2	ALT5
	GPIO2_IO23	EPDC1_SDCE3	ALT5
	GPIO2_IO24	EPDC1_GDCLK	ALT5
	GPIO2_IO25	EPDC1_GDOE	ALT5
	GPIO2_IO26	EPDC1_GDRL	ALT5
	GPIO2_IO27	EPDC1_GDSP	ALT5
	GPIO2_IO28	EPDC1_BDR0	ALT5
	GPIO2_IO29	EPDC1_BDR1	ALT5
	GPIO2_IO30	EPDC1_PWRCOM	ALT5
	GPIO2_IO31	EPDC1_PWRSTAT	ALT5
GPIO3	GPIO3_IO00	LCD1_CLK	ALT5
	GPIO3_IO01	LCD1_ENABLE	ALT5
	GPIO3_IO02	LCD1_HSYNC	ALT5
	GPIO3_IO03	LCD1_VSYNC	ALT5
	GPIO3_IO04	LCD1_RESET	ALT5
	GPIO3_IO05	LCD1_DATA00	ALT5
	GPIO3_IO06	LCD1_DATA01	ALT5
	GPIO3_IO07	LCD1_DATA02	ALT5
	GPIO3_IO08	LCD1_DATA03	ALT5

Table continues on the next page...

Instance	Port	Pad	Mode
	GPIO3_IO09	LCD1_DATA04	ALT5
	GPIO3_IO10	LCD1_DATA05	ALT5
	GPIO3_IO11	LCD1_DATA06	ALT5
	GPIO3_IO12	LCD1_DATA07	ALT5
	GPIO3_IO13	LCD1_DATA08	ALT5
	GPIO3_IO14	LCD1_DATA09	ALT5
	GPIO3_IO15	LCD1_DATA10	ALT5
	GPIO3_IO16	LCD1_DATA11	ALT5
	GPIO3_IO17	LCD1_DATA12	ALT5
	GPIO3_IO18	LCD1_DATA13	ALT5
	GPIO3_IO19	LCD1_DATA14	ALT5
	GPIO3_IO20	LCD1_DATA15	ALT5
	GPIO3_IO21	LCD1_DATA16	ALT5
	GPIO3_IO22	LCD1_DATA17	ALT5
	GPIO3_IO23	LCD1_DATA18	ALT5
	GPIO3_IO24	LCD1_DATA19	ALT5
	GPIO3_IO25	LCD1_DATA20	ALT5
	GPIO3_IO26	LCD1_DATA21	ALT5
	GPIO3_IO27	LCD1_DATA22	ALT5
	GPIO3_IO28	LCD1_DATA23	ALT5
GPIO4	GPIO4_IO00	UART1_RXD	ALT5
	GPIO4_IO01	UART1_TXD	ALT5
	GPIO4_IO02	UART2_RXD	ALT5
	GPIO4_IO03	UART2_TXD	ALT5
	GPIO4_IO04	UART3_RXD	ALT5
	GPIO4_IO05	UART3_TXD	ALT5
	GPIO4_IO06	UART3_RTS	ALT5
	GPIO4_IO07	UART3_CTS	ALT5
	GPIO4_IO08	I2C1_SCL	ALT5
	GPIO4_IO09	I2C1_SDA	ALT5
	GPIO4_IO10	I2C2_SCL	ALT5
	GPIO4_IO11	I2C2_SDA	ALT5
	GPIO4_IO12	I2C3_SCL	ALT5
	GPIO4_IO13	I2C3_SDA	ALT5
	GPIO4_IO14	I2C4_SCL	ALT5
	GPIO4_IO15	I2C4_SDA	ALT5
	GPIO4_IO16	ECSPI1_SCLK	ALT5
	GPIO4_IO17	ECSPI1_MOSI	ALT5
	GPIO4_IO18	ECSPI1_MISO	ALT5
	GPIO4_IO19	ECSPI1_SS0	ALT5
	GPIO4_IO20	ECSPI2_SCLK	ALT5

Table continues on the next page...

Instance	Port	Pad	Mode
	GPIO4_IO21	ECSPI2_MOSI	ALT5
	GPIO4_IO22	ECSPI2_MISO	ALT5
	GPIO4_IO23	ECSPI2_SS0	ALT5
GPIO5	GPIO5_IO00	SD1_CD_B	ALT5
	GPIO5_IO01	SD1_WP	ALT5
	GPIO5_IO02	SD1_RESET_B	ALT5
	GPIO5_IO03	SD1_CLK	ALT5
	GPIO5_IO04	SD1_CMD	ALT5
	GPIO5_IO05	SD1_DATA0	ALT5
	GPIO5_IO06	SD1_DATA1	ALT5
	GPIO5_IO07	SD1_DATA2	ALT5
	GPIO5_IO08	SD1_DATA3	ALT5
	GPIO5_IO09	SD2_CD_B	ALT5
	GPIO5_IO10	SD2_WP	ALT5
	GPIO5_IO11	SD2_RESET_B	ALT5
	GPIO5_IO12	SD2_CLK	ALT5
	GPIO5_IO13	SD2_CMD	ALT5
	GPIO5_IO14	SD2_DATA0	ALT5
	GPIO5_IO15	SD2_DATA1	ALT5
	GPIO5_IO16	SD2_DATA2	ALT5
	GPIO5_IO17	SD2_DATA3	ALT5
GPIO6	GPIO6_IO00	SD3_CLK	ALT5
	GPIO6_IO01	SD3_CMD	ALT5
	GPIO6_IO02	SD3_DATA0	ALT5
	GPIO6_IO03	SD3_DATA1	ALT5
	GPIO6_IO04	SD3_DATA2	ALT5
	GPIO6_IO05	SD3_DATA3	ALT5
	GPIO6_IO06	SD3_DATA4	ALT5
	GPIO6_IO07	SD3_DATA5	ALT5
	GPIO6_IO08	SD3_DATA6	ALT5
	GPIO6_IO09	SD3_DATA7	ALT5
	GPIO6_IO10	SD3_STROBE	ALT5
	GPIO6_IO11	SD3_RESET_B	ALT5
	GPIO6_IO12	SAI1_RXD	ALT5
	GPIO6_IO13	SAI1_TXC	ALT5
	GPIO6_IO14	SAI1_TXFS	ALT5
	GPIO6_IO15	SAI1_TXD	ALT5
	GPIO6_IO16	SAI1_RXFS	ALT5
	GPIO6_IO17	SAI1_RXC	ALT5
	GPIO6_IO18	SAI1_MCLK	ALT5
	GPIO6_IO19	SAI2_TXFS	ALT5

Table continues on the next page...

Instance	Port	Pad	Mode
	GPIO6_IO20	SAI2_TXC	ALT5
	GPIO6_IO21	SAI2_RXD	ALT5
	GPIO6_IO22	SAI2_TXD	ALT5
GPIO7	GPIO7_IO00	ENET1_RDATA0	ALT5
	GPIO7_IO01	ENET1_RDATA1	ALT5
	GPIO7_IO02	ENET1_RDATA2	ALT5
	GPIO7_IO03	ENET1_RDATA3	ALT5
	GPIO7_IO04	ENET1_RX_CTL	ALT5
	GPIO7_IO05	ENET1_RXC	ALT5
	GPIO7_IO06	ENET1_TDATA0	ALT5
	GPIO7_IO07	ENET1_TDATA1	ALT5
	GPIO7_IO08	ENET1_TDATA2	ALT5
	GPIO7_IO09	ENET1_TDATA3	ALT5
	GPIO7_IO10	ENET1_TX_CTL	ALT5
	GPIO7_IO11	ENET1_TXC	ALT5
	GPIO7_IO12	ENET1_TX_CLK	ALT5
	GPIO7_IO13	ENET1_RX_CLK	ALT5
	GPIO7_IO14	ENET1_CRS	ALT5
	GPIO7_IO15	ENET1_COL	ALT5
GPT1	GPT1_CAPTURE1	LCD1_DATA03	ALT1
	GPT1_CAPTURE2	LCD1_DATA04	ALT1
	GPT1_CLK	LCD1_DATA02	ALT1
	GPT1_COMPARE1	LCD1_RESET	ALT1
	GPT1_COMPARE2	LCD1_DATA00	ALT1
	GPT1_COMPARE3	LCD1_DATA01	ALT1
GPT2	GPT2_CAPTURE1	ENET1_CRS	ALT3
	GPT2_CAPTURE2	ENET1_COL	ALT3
	GPT2_CLK	ENET1_RX_CLK	ALT3
	GPT2_COMPARE1	ENET1_TX_CTL	ALT3
	GPT2_COMPARE2	ENET1_TXC	ALT3
	GPT2_COMPARE3	ENET1_TX_CLK	ALT3
GPT3	GPT3_CAPTURE1	SD3_CMD	ALT4
	GPT3_CAPTURE2	SD3_DATA0	ALT4
	GPT3_CLK	SD3_CLK	ALT4
	GPT3_COMPARE1	SD3_DATA1	ALT4
	GPT3_COMPARE2	SD3_DATA2	ALT4
	GPT3_COMPARE3	SD3_DATA3	ALT4
GPT4	GPT4_CAPTURE1	SD2_CMD	ALT3
	GPT4_CAPTURE2	SD2_DATA0	ALT3
	GPT4_CLK	SD2_CLK	ALT3
	GPT4_COMPARE1	SD2_DATA1	ALT3

Table continues on the next page...

Instance	Port	Pad	Mode
	GPT4_COMPARE2	SD2_DATA2	ALT3
	GPT4_COMPARE3	SD2_DATA3	ALT3
I2C1	I2C1_SCL	GPIO1_IO04	ALT4
		I2C1_SCL	ALT0
		UART1_RXD	ALT1
	I2C1_SDA	GPIO1_IO05	ALT4
		I2C1_SDA	ALT0
		UART1_TXD	ALT1
I2C2	I2C2_SCL	GPIO1_IO06	ALT4
		I2C2_SCL	ALT0
		UART2_RXD	ALT1
	I2C2_SDA	GPIO1_IO07	ALT4
		I2C2_SDA	ALT0
		UART2_TXD	ALT1
I2C3	I2C3_SCL	ENET1_RDATA0	ALT2
		GPIO1_IO08	ALT4
		I2C3_SCL	ALT0
		LCD1_DATA20	ALT6
		SD3_DATA3	ALT2
	I2C3_SDA	ENET1_RDATA1	ALT2
		GPIO1_IO09	ALT4
		I2C3_SDA	ALT0
		LCD1_DATA21	ALT6
		SD3_DATA2	ALT2
I2C4	I2C4_SCL	ENET1_TDATA2	ALT3
		GPIO1_IO10	ALT4
		I2C4_SCL	ALT0
		LCD1_DATA22	ALT6
		SAI1_RXFS	ALT3
	I2C4_SDA	ENET1_TDATA3	ALT3
		GPIO1_IO11	ALT4
		I2C4_SDA	ALT0
		LCD1_DATA23	ALT6
		SAI1_RXC	ALT3
KPP	KEY_COL0	ENET1_TDATA1	ALT6
		EPDC1_DATA07	ALT3
	KEY_COL1	ENET1_RXC	ALT6
		EPDC1_DATA05	ALT3
	KEY_COL2	ENET1_RDATA3	ALT6
		EPDC1_DATA03	ALT3
	KEY_COL3	ENET1_RDATA1	ALT6

Table continues on the next page...

Instance	Port	Pad	Mode
		EPDC1_DATA01	ALT3
	KEY_COL4	EPDC1_SDLE	ALT3
		GPIO1_IO07	ALT6
	KEY_COL5	EPDC1_SDOE	ALT3
		GPIO1_IO08	ALT6
	KEY_COL6	EPDC1_SDCE2	ALT3
		GPIO1_IO10	ALT6
	KEY_COL7	EPDC1_GDCLK	ALT3
		SAI2_RXD	ALT6
	KEY_ROW0	ENET1_TDATA0	ALT6
		EPDC1_DATA06	ALT3
	KEY_ROW1	ENET1_RX_CTL	ALT6
		EPDC1_DATA04	ALT3
	KEY_ROW2	ENET1_RDATA2	ALT6
		EPDC1_DATA02	ALT3
	KEY_ROW3	ENET1_RDATA0	ALT6
		EPDC1_DATA00	ALT3
	KEY_ROW4	EPDC1_SDCLK	ALT3
		GPIO1_IO06	ALT6
	KEY_ROW5	EPDC1_SDSHR	ALT3
		GPIO1_IO09	ALT6
	KEY_ROW6	EPDC1_SDCE3	ALT3
		GPIO1_IO11	ALT6
	KEY_ROW7	EPDC1_GDOE	ALT3
		SAI2_TXD	ALT6
MIPI_CSI	MIPI_CSI_CLK_N	MIPI_CSI_CLK_N	No muxing
	MIPI_CSI_CLK_P	MIPI_CSI_CLK_P	No muxing
	MIPI_CSI_D0_N	MIPI_CSI_D0_N	No muxing
	MIPI_CSI_D0_P	MIPI_CSI_D0_P	No muxing
	MIPI_CSI_D1_N	MIPI_CSI_D1_N	No muxing
	MIPI_CSI_D1_P	MIPI_CSI_D1_P	No muxing
MIPI_DSI	MIPI_DSI_CLK_N	MIPI_DSI_CLK_N	No muxing
	MIPI_DSI_CLK_P	MIPI_DSI_CLK_P	No muxing
	MIPI_DSI_D0_N	MIPI_DSI_D0_N	No muxing
	MIPI_DSI_D0_P	MIPI_DSI_D0_P	No muxing
	MIPI_DSI_D1_N	MIPI_DSI_D1_N	No muxing
	MIPI_DSI_D1_P	MIPI_DSI_D1_P	No muxing
MMDC	DRAM_ADDR00	DRAM_ADDR00	No muxing (ALT0)
	DRAM_ADDR01	DRAM_ADDR01	No muxing (ALT0)
	DRAM_ADDR02	DRAM_ADDR02	No muxing (ALT0)
	DRAM_ADDR03	DRAM_ADDR03	No muxing (ALT0)

Table continues on the next page...

Instance	Port	Pad	Mode
	DRAM_ADDR04	DRAM_ADDR04	No muxing (ALT0)
	DRAM_ADDR05	DRAM_ADDR05	No muxing (ALT0)
	DRAM_ADDR06	DRAM_ADDR06	No muxing (ALT0)
	DRAM_ADDR07	DRAM_ADDR07	No muxing (ALT0)
	DRAM_ADDR08	DRAM_ADDR08	No muxing (ALT0)
	DRAM_ADDR09	DRAM_ADDR09	No muxing (ALT0)
	DRAM_ADDR10	DRAM_ADDR10	No muxing (ALT0)
	DRAM_ADDR11	DRAM_ADDR11	No muxing (ALT0)
	DRAM_ADDR12	DRAM_ADDR12	No muxing (ALT0)
	DRAM_ADDR13	DRAM_ADDR13	No muxing (ALT0)
	DRAM_ADDR14	DRAM_ADDR14	No muxing (ALT0)
	DRAM_ADDR15	DRAM_ADDR15	No muxing (ALT0)
	DRAM_CAS_B	DRAM_CAS_B	No muxing (ALT0)
	DRAM_CS0_B	DRAM_CS0_B	No muxing (ALT0)
	DRAM_CS1_B	DRAM_CS1_B	No muxing (ALT0)
	DRAM_DATA00	DRAM_DATA00	No muxing (ALT0)
	DRAM_DATA01	DRAM_DATA01	No muxing (ALT0)
	DRAM_DATA02	DRAM_DATA02	No muxing (ALT0)
	DRAM_DATA03	DRAM_DATA03	No muxing (ALT0)
	DRAM_DATA04	DRAM_DATA04	No muxing (ALT0)
	DRAM_DATA05	DRAM_DATA05	No muxing (ALT0)
	DRAM_DATA06	DRAM_DATA06	No muxing (ALT0)
	DRAM_DATA07	DRAM_DATA07	No muxing (ALT0)
	DRAM_DATA08	DRAM_DATA08	No muxing (ALT0)
	DRAM_DATA09	DRAM_DATA09	No muxing (ALT0)
	DRAM_DATA10	DRAM_DATA10	No muxing (ALT0)
	DRAM_DATA11	DRAM_DATA11	No muxing (ALT0)
	DRAM_DATA12	DRAM_DATA12	No muxing (ALT0)
	DRAM_DATA13	DRAM_DATA13	No muxing (ALT0)
	DRAM_DATA14	DRAM_DATA14	No muxing (ALT0)
	DRAM_DATA15	DRAM_DATA15	No muxing (ALT0)
	DRAM_DATA16	DRAM_DATA16	No muxing (ALT0)
	DRAM_DATA17	DRAM_DATA17	No muxing (ALT0)
	DRAM_DATA18	DRAM_DATA18	No muxing (ALT0)
	DRAM_DATA19	DRAM_DATA19	No muxing (ALT0)
	DRAM_DATA20	DRAM_DATA20	No muxing (ALT0)
	DRAM_DATA21	DRAM_DATA21	No muxing (ALT0)
	DRAM_DATA22	DRAM_DATA22	No muxing (ALT0)
	DRAM_DATA23	DRAM_DATA23	No muxing (ALT0)
	DRAM_DATA24	DRAM_DATA24	No muxing (ALT0)
	DRAM_DATA25	DRAM_DATA25	No muxing (ALT0)

Table continues on the next page...

Instance	Port	Pad	Mode
	DRAM_DATA26	DRAM_DATA26	No muxing (ALT0)
	DRAM_DATA27	DRAM_DATA27	No muxing (ALT0)
	DRAM_DATA28	DRAM_DATA28	No muxing (ALT0)
	DRAM_DATA29	DRAM_DATA29	No muxing (ALT0)
	DRAM_DATA30	DRAM_DATA30	No muxing (ALT0)
	DRAM_DATA31	DRAM_DATA31	No muxing (ALT0)
	DRAM_DQM0	DRAM_DQM0	No muxing (ALT0)
	DRAM_DQM1	DRAM_DQM1	No muxing (ALT0)
	DRAM_DQM2	DRAM_DQM2	No muxing (ALT0)
	DRAM_DQM3	DRAM_DQM3	No muxing (ALT0)
	DRAM_ODT0	DRAM_ODT0	No muxing (ALT0)
	DRAM_ODT1	DRAM_ODT1	No muxing (ALT0)
	DRAM_RAS_B	DRAM_RAS_B	No muxing (ALT0)
	DRAM_RESET	DRAM_RESET	No muxing (ALT0)
	DRAM_SDBA0	DRAM_SDBA0	No muxing (ALT0)
	DRAM_SDBA1	DRAM_SDBA1	No muxing (ALT0)
	DRAM_SDBA2	DRAM_SDBA2	No muxing (ALT0)
	DRAM_SDCKE0	DRAM_SDCKE0	No muxing (ALT0)
	DRAM_SDCKE1	DRAM_SDCKE1	No muxing (ALT0)
	DRAM_SDCLK0	DRAM_SDCLK0_P	No muxing (ALT0)
	DRAM_SDQS0_P	DRAM_SDQS0_P	No muxing (ALT0)
	DRAM_SDQS1_P	DRAM_SDQS1_P	No muxing (ALT0)
	DRAM_SDQS2_P	DRAM_SDQS2_P	No muxing (ALT0)
	DRAM_SDQS3_P	DRAM_SDQS3_P	No muxing (ALT0)
	DRAM_SDWE_B	DRAM_SDWE_B	No muxing (ALT0)
	DRAM_SDCLK0_N	DRAM_SDCLK0_N	No muxing (ALT0)
	DRAM_SDQS0_N	DRAM_SDQS0_N	No muxing (ALT0)
	DRAM_SDQS1_N	DRAM_SDQS1_N	No muxing (ALT0)
	DRAM_SDQS2_N	DRAM_SDQS2_N	No muxing (ALT0)
	DRAM_SDQS3_N	DRAM_SDQS3_N	No muxing (ALT0)
	DRAM_VREF	DRAM_VREF	No muxing
	DRAM_ZQPAD	DRAM_ZQPAD	No muxing
MQS	MQS_LEFT	SAI1_RXC	ALT6
		SD2_CMD	ALT2
	MQS_RIGHT	SAI1_RXFS	ALT6
		SD2_CLK	ALT2
NAND	NAND_ALE	SD3_CMD	ALT1
	NAND_CE0_B	SAI1_TXC	ALT1
	NAND_CE1_B	SAI1_RXD	ALT1
	NAND_CE2_B	SAI1_RXFS	ALT1
	NAND_CE3_B	SAI1_RXC	ALT1

Table continues on the next page...

Instance	Port	Pad	Mode
	NAND_CLE	SD3_CLK	ALT1
	NAND_DATA00	SD3_DATA0	ALT1
	NAND_DATA01	SD3_DATA1	ALT1
	NAND_DATA02	SD3_DATA2	ALT1
	NAND_DATA03	SD3_DATA3	ALT1
	NAND_DATA04	SD3_DATA4	ALT1
	NAND_DATA05	SD3_DATA5	ALT1
	NAND_DATA06	SD3_DATA6	ALT1
	NAND_DATA07	SD3_DATA7	ALT1
	NAND_DQS	SAI1_TXFS	ALT1
	NAND_RE_B	SD3_STROBE	ALT1
	NAND_READY	SAI1_TXD	ALT1
	NAND_WE_B	SD3_RESET_B	ALT1
	NAND_WP_B	SAI1_MCLK	ALT1
PWM1	PWM1_OUT	ENET1_RDATA0	ALT1
		GPIO1_IO01	ALT1
		GPIO1_IO08	ALT7
PWM2	PWM2_OUT	ENET1_RDATA1	ALT1
		GPIO1_IO02	ALT1
		GPIO1_IO09	ALT7
PWM3	PWM3_OUT	ENET1_TDATA0	ALT1
		GPIO1_IO03	ALT1
		GPIO1_IO10	ALT7
PWM4	PWM4_OUT	ENET1_TDATA1	ALT1
		GPIO1_IO00	ALT1
		GPIO1_IO11	ALT7
QSPI	QSPI_A_DATA0	EPDC1_DATA00	ALT2
	QSPI_A_DATA1	EPDC1_DATA01	ALT2
	QSPI_A_DATA2	EPDC1_DATA02	ALT2
	QSPI_A_DATA3	EPDC1_DATA03	ALT2
	QSPI_A_DQS	EPDC1_DATA04	ALT2
	QSPI_A_SCLK	EPDC1_DATA05	ALT2
	QSPI_A_SS0_B	EPDC1_DATA06	ALT2
	QSPI_A_SS1_B	EPDC1_DATA07	ALT2
	QSPI_B_DATA0	EPDC1_DATA08	ALT2
	QSPI_B_DATA1	EPDC1_DATA09	ALT2
	QSPI_B_DATA2	EPDC1_DATA10	ALT2
	QSPI_B_DATA3	EPDC1_DATA11	ALT2
	QSPI_B_DQS	EPDC1_DATA12	ALT2
	QSPI_B_SCLK	EPDC1_DATA13	ALT2
	QSPI_B_SS0_B	EPDC1_DATA14	ALT2

Table continues on the next page...

Instance	Port	Pad	Mode
	QSPI_B_SS1_B	EPDC1_DATA15	ALT2
SAI1	SAI1_MCLK	GPIO1_IO01	ALT3
		SAI1_MCLK	ALT0
	SAI1_RX_BCLK	ENET1_TXC	ALT2
		SAI1_RXC	ALT0
	SAI1_RX_DATA	ENET1_TX_CLK	ALT2
		SAI1_RXD	ALT0
	SAI1_RX_SYNC	ENET1_TX_CTL	ALT2
		SAI1_RXFS	ALT0
	SAI1_TX_BCLK	ENET1_RX_CLK	ALT2
		SAI1_TXC	ALT0
	SAI1_TX_DATA	ENET1_COL	ALT2
		SAI1_TXD	ALT0
	SAI1_TX_SYNC	ENET1_CRS	ALT2
		SAI1_TXFS	ALT0
SAI2	SAI2_MCLK	GPIO1_IO02	ALT3
		SAI1_MCLK	ALT2
		SD2_RESET_B	ALT1
	SAI2_RX_BCLK	SAI1_RXC	ALT2
		SD2_CMD	ALT1
	SAI2_RX_DATA	SAI2_RXD	ALT0
		SD2_DATA0	ALT1
	SAI2_RX_SYNC	SAI1_RXFS	ALT2
		SD2_CLK	ALT1
	SAI2_TX_BCLK	SAI2_TXC	ALT0
		SD2_DATA1	ALT1
	SAI2_TX_DATA	SAI2_TXD	ALT0
		SD2_DATA3	ALT1
	SAI2_TX_SYNC	SAI2_TXFS	ALT0
		SD2_DATA2	ALT1
SAI3	SAI3_MCLK	GPIO1_IO03	ALT3
		SD1_RESET_B	ALT1
		SD3_RESET_B	ALT3
		UART1_TXD	ALT2
	SAI3_RX_BCLK	SD1_CMD	ALT1
		SD3_CMD	ALT3
		UART2_RXD	ALT2
	SAI3_RX_DATA	SD1_DATA0	ALT1
		SD3_DATA0	ALT3
		UART2_TXD	ALT2
	SAI3_RX_SYNC	SD1_CLK	ALT1

Table continues on the next page...

Instance	Port	Pad	Mode
		SD3_CLK	ALT3
		UART3_RXD	ALT2
	SAI3_TX_BCLK	SD1_DATA1	ALT1
		SD3_DATA1	ALT3
		UART3_TXD	ALT2
	SAI3_TX_DATA	SD1_DATA3	ALT1
		SD3_DATA3	ALT3
		UART3_RTS	ALT2
	SAI3_TX_SYNC	SD1_DATA2	ALT1
		SD3_DATA2	ALT3
		UART3_CTS	ALT2
SDMA	SDMA_EXT_EVENT0	GPIO1_IO14	ALT6
		I2C3_SCL	ALT4
		SD2_CD_B	ALT6
	SDMA_EXT_EVENT1	GPIO1_IO15	ALT6
		I2C3_SDA	ALT4
		SD2_WP	ALT6
SIM1	SIM1_PORT1_CLK	EPDC1_DATA09	ALT1
		SAI1_TXC	ALT4
	SIM1_PORT1_PD	EPDC1_DATA12	ALT1
		SAI1_RXFS	ALT4
	SIM1_PORT1_RST_B	EPDC1_DATA10	ALT1
		SAI1_TXFS	ALT4
	SIM1_PORT1_SVEN	EPDC1_DATA11	ALT1
		SAI1_TXD	ALT4
	SIM1_PORT1_TRXD	EPDC1_DATA08	ALT1
		SAI1_RXD	ALT4
	SIM1_PORT2_CLK	EPDC1_DATA01	ALT1
	SIM1_PORT2_PD	EPDC1_DATA04	ALT1
	SIM1_PORT2_RST_B	EPDC1_DATA02	ALT1
	SIM1_PORT2_SVEN	EPDC1_DATA03	ALT1
	SIM1_PORT2_TRXD	EPDC1_DATA00	ALT1
SIM2	SIM2_PORT1_CLK	EPDC1_DATA14	ALT1
		SD2_DATA0	ALT4
	SIM2_PORT1_PD	EPDC1_SDCE3	ALT1
		SD2_DATA3	ALT4
	SIM2_PORT1_RST_B	EPDC1_DATA15	ALT1
		SD2_DATA1	ALT4
	SIM2_PORT1_SVEN	EPDC1_SDCE2	ALT1
		SD2_DATA2	ALT4
	SIM2_PORT1_TRXD	EPDC1_DATA13	ALT1

Table continues on the next page...

Instance	Port	Pad	Mode
		SD2_CMD	ALT4
	SIM2_PORT2_CLK	EPDC1_DATA06	ALT1
	SIM2_PORT2_PD	EPDC1_SDLE	ALT1
	SIM2_PORT2_RST_B	EPDC1_DATA07	ALT1
	SIM2_PORT2_SVEN	EPDC1_SDCLK	ALT1
	SIM2_PORT2_TRXD	EPDC1_DATA05	ALT1
SJC	JTAG_DE_B	EPDC1_DATA06	ALT7
	JTAG_MOD	JTAG_MOD	No Muxing (ALT0)
	JTAG_TCK	JTAG_TCK	No Muxing (ALT0)
	JTAG_TDI	JTAG_TDI	No Muxing (ALT0)
	JTAG_TDO	JTAG_TDO	No Muxing (ALT0)
	JTAG_TMS	JTAG_TMS	No Muxing (ALT0)
	JTAG_TRST_B	JTAG_TRST_B	No Muxing (ALT0)
SNVS	SNVS_PMIC_ON_REQ	SNVS_PMIC_ON_REQ	No muxing
	SNVS_TAMPER0	SNVS_TAMPER0	No muxing
	SNVS_TAMPER1	SNVS_TAMPER1	No muxing
	SNVS_TAMPER2	SNVS_TAMPER2	No muxing
	SNVS_TAMPER3	SNVS_TAMPER3	No muxing
	SNVS_TAMPER4	SNVS_TAMPER4	No muxing
	SNVS_TAMPER5	SNVS_TAMPER5	No muxing
	SNVS_TAMPER6	SNVS_TAMPER6	No muxing
	SNVS_TAMPER7	SNVS_TAMPER7	No muxing
	SNVS_TAMPER8	SNVS_TAMPER8	No muxing
	SNVS_TAMPER9	SNVS_TAMPER9	No muxing
	SNVS_VIO_5	GPIO1_IO12	ALT6
	SNVS_VIO_5_CTL	GPIO1_IO13	ALT6
SRC	SRC_BOOT_CFG00	LCD1_DATA00	ALT6
	SRC_BOOT_CFG01	LCD1_DATA01	ALT6
	SRC_BOOT_CFG02	LCD1_DATA02	ALT6
	SRC_BOOT_CFG03	LCD1_DATA03	ALT6
	SRC_BOOT_CFG04	LCD1_DATA04	ALT6
	SRC_BOOT_CFG05	LCD1_DATA05	ALT6
	SRC_BOOT_CFG06	LCD1_DATA06	ALT6
	SRC_BOOT_CFG07	LCD1_DATA07	ALT6
	SRC_BOOT_CFG08	LCD1_DATA08	ALT6
	SRC_BOOT_CFG09	LCD1_DATA09	ALT6
	SRC_BOOT_CFG10	LCD1_DATA10	ALT6
	SRC_BOOT_CFG11	LCD1_DATA11	ALT6
	SRC_BOOT_CFG12	LCD1_DATA12	ALT6
	SRC_BOOT_CFG13	LCD1_DATA13	ALT6
	SRC_BOOT_CFG14	LCD1_DATA14	ALT6

Table continues on the next page...

Instance	Port	Pad	Mode
	SRC_BOOT_CFG15	LCD1_DATA15	ALT6
	SRC_BOOT_CFG16	LCD1_DATA16	ALT6
	SRC_BOOT_CFG17	LCD1_DATA17	ALT6
	SRC_BOOT_CFG18	LCD1_DATA18	ALT6
	SRC_BOOT_CFG19	LCD1_DATA19	ALT6
	SRC_BOOT_MODE0	BOOT_MODE0	No Muxing (ALT0)
	SRC_BOOT_MODE1	BOOT_MODE1	No Muxing (ALT0)
	SRC_CA7_RESET0_B	SAI1_RXFS	ALT7
	SRC_CA7_RESET1_B	SAI1_RXC	ALT7
	SRC_POR_B	POR_B	No Muxing (ALT0)
	SRC_SYSTEM_RESET	SAI1_TXD	ALT7
TEMPSENSOR	TEMPSENSOR_REXT	TEMPSENSOR_REXT	No muxing
UART1	UART1_CTS_B	ENET1_RDATA0	ALT3
		SAI2_TXFS	ALT3
	UART1_RTS_B	ENET1_RDATA1	ALT3
		SAI2_TXC	ALT3
	UART1_RX_DATA	ENET1_RDATA2	ALT3
		UART1_RXD	ALT0
	UART1_TX_DATA	ENET1_RDATA3	ALT3
		UART1_TXD	ALT0
UART2	UART2_CTS_B	LCD1_VSYNC	ALT4
		SAI2_RXD	ALT3
	UART2_RTS_B	LCD1_HSYNC	ALT4
		SAI2_TXD	ALT3
	UART2_RX_DATA	LCD1_CLK	ALT4
		UART2_RXD	ALT0
	UART2_TX_DATA	LCD1_ENABLE	ALT4
		UART2_TXD	ALT0
UART3	UART3_CTS_B	GPIO1_IO11	ALT3
		SD3_DATA7	ALT3
		UART3_CTS	ALT0
	UART3_RTS_B	GPIO1_IO10	ALT3
		SD3_DATA6	ALT3
		UART3_RTS	ALT0
	UART3_RX_DATA	GPIO1_IO08	ALT3
		SD3_DATA4	ALT3
		UART3_RXD	ALT0
	UART3_TX_DATA	GPIO1_IO09	ALT3
		SD3_DATA5	ALT3
		UART3_TXD	ALT0
UART4	UART4_CTS_B	I2C1_SCL	ALT1

Table continues on the next page...

Instance	Port	Pad	Mode
		SAI2_RXD	ALT2
		SD2_DATA2	ALT2
	UART4_RTS_B	I2C1_SDA	ALT1
		SAI2_TXD	ALT2
		SD2_DATA3	ALT2
	UART4_RX_DATA	I2C2_SCL	ALT1
		SAI2_TXFS	ALT2
		SD2_DATA0	ALT2
	UART4_TX_DATA	I2C2_SDA	ALT1
		SAI2_TXC	ALT2
		SD2_DATA1	ALT2
UART5	UART5_CTS_B	GPIO1_IO04	ALT3
		I2C3_SCL	ALT1
		SAI1_TXFS	ALT2
	UART5_RTS_B	GPIO1_IO05	ALT3
		I2C3_SDA	ALT1
		SAI1_TXD	ALT2
	UART5_RX_DATA	GPIO1_IO06	ALT3
		I2C4_SCL	ALT1
		SAI1_RXD	ALT2
	UART5_TX_DATA	GPIO1_IO07	ALT3
		I2C4_SDA	ALT1
		SAI1_TXC	ALT2
UART6	UART6_CTS_B	ECSPI1_SS0	ALT1
		EPDC1_DATA11	ALT3
		SD1_CLK	ALT2
	UART6_RTS_B	ECSPI1_MISO	ALT1
		EPDC1_DATA10	ALT3
		SD1_RESET_B	ALT2
	UART6_RX_DATA	ECSPI1_SCLK	ALT1
		EPDC1_DATA08	ALT3
		SD1_CD_B	ALT2
	UART6_TX_DATA	ECSPI1_MOSI	ALT1
		EPDC1_DATA09	ALT3
		SD1_WP	ALT2
UART7	UART7_CTS_B	ECSPI2_SS0	ALT1
		EPDC1_DATA15	ALT3
		SD1_DATA2	ALT2
	UART7_RTS_B	ECSPI2_MISO	ALT1
		EPDC1_DATA14	ALT3
		SD1_DATA3	ALT2

Table continues on the next page...

Instance	Port	Pad	Mode
	UART7_RX_DATA	ECSPI2_SCLK	ALT1
		EPDC1_DATA12	ALT3
		SD1_DATA0	ALT2
	UART7_TX_DATA	ECSPI2_MOSI	ALT1
		EPDC1_DATA13	ALT3
		SD1_DATA1	ALT2
USB	USB_OTG1_ID	GPIO1_IO02	ALT7
		GPIO1_IO12	ALT7
		I2C4_SCL	ALT4
		SD2_WP	ALT4
		USB_OTG1_ID	No Muxing
	USB_OTG1_OC	GPIO1_IO04	ALT1
		UART3_RXD	ALT1
	USB_OTG1_PWR	GPIO1_IO05	ALT1
		UART3_TXD	ALT1
	USB_H_DATA	USB_H_DATA	No muxing
	USB_H_STROBE	USB_H_STROBE	No muxing
	USB_OTG1_CHD_B	USB_OTG1_CHD_B	No muxing
	USB_OTG1_DN	USB_OTG1_DN	No muxing
	USB_OTG1_DP	USB_OTG1_DP	No muxing
	USB_OTG1_REXT	USB_OTG1_REXT	No muxing
USDHC1	SD1_CD_B	SD1_CD_B	ALT0
	SD1_CLK	SD1_CLK	ALT0
	SD1_CMD	SD1_CMD	ALT0
	SD1_DATA0	SD1_DATA0	ALT0
	SD1_DATA1	SD1_DATA1	ALT0
	SD1_DATA2	SD1_DATA2	ALT0
	SD1_DATA3	SD1_DATA3	ALT0
	SD1_DATA4	ECSPI2_SCLK	ALT2
	SD1_DATA5	ECSPI2_MOSI	ALT2
	SD1_DATA6	ECSPI2_MISO	ALT2
	SD1_DATA7	ECSPI2_SS0	ALT2
	SD1_LCTL	GPIO1_IO09	ALT1
		UART3_RXD	ALT6
	SD1_RESET_B	SD1_RESET_B	ALT0
	SD1_VSELECT	GPIO1_IO08	ALT1
		UART3_CTS	ALT6
	SD1_WP	SD1_WP	ALT0
USDHC2	SD2_CD_B	SD2_CD_B	ALT0
	SD2_CLK	SD2_CLK	ALT0
	SD2_CMD	SD2_CMD	ALT0

Table continues on the next page...

Instance	Port	Pad	Mode
	SD2_DATA0	SD2_DATA0	ALT0
	SD2_DATA1	SD2_DATA1	ALT0
	SD2_DATA2	SD2_DATA2	ALT0
	SD2_DATA3	SD2_DATA3	ALT0
	SD2_DATA4	ECSPI1_SCLK	ALT2
	SD2_DATA5	ECSPI1_MOSI	ALT2
	SD2_DATA6	ECSPI1_MISO	ALT2
	SD2_DATA7	ECSPI1_SS0	ALT2
	SD2_LCTL	GPIO1_IO10	ALT1
		UART3_TXD	ALT6
	SD2_RESET_B	SD2_RESET_B	ALT2
		SD2_RESET_B	ALT0
	SD2_VSELECT	GPIO1_IO12	ALT1
		I2C1_SCL	ALT6
	SD2_WP	SD2_WP	ALT0
USDHC3	SD3_CD_B	GPIO1_IO14	ALT1
		I2C2_SCL	ALT6
		SD3_DATA7	ALT2
	SD3_CLK	SD3_CLK	ALT0
	SD3_CMD	SD3_CMD	ALT0
	SD3_DATA0	SD3_DATA0	ALT0
	SD3_DATA1	SD3_DATA1	ALT0
	SD3_DATA2	SD3_DATA2	ALT0
	SD3_DATA3	SD3_DATA3	ALT0
	SD3_DATA4	SD3_DATA4	ALT0
	SD3_DATA5	SD3_DATA5	ALT0
	SD3_DATA6	SD3_DATA6	ALT0
	SD3_DATA7	SD3_DATA7	ALT0
	SD3_LCTL	GPIO1_IO11	ALT1
		UART3_RTS	ALT6
	SD3_RESET_B	SD3_RESET_B	ALT2
		SD3_RESET_B	ALT0
	SD3_STROBE	SD3_STROBE	ALT0
	SD3_VSELECT	GPIO1_IO13	ALT1
		I2C1_SDA	ALT6
	SD3_WP	GPIO1_IO15	ALT1
		I2C2_SDA	ALT6
		SD3_DATA6	ALT2
WDOG1	WDOG1_ANY	ENET1_COL	ALT1
		GPIO1_IO00	ALT2
	WDOG1_B	GPIO1_IO00	ALT3

Table continues on the next page...

Instance	Port	Pad	Mode
		GPIO1_IO08	ALT2
	WDOG1_RESET_B_DEB	GPIO1_IO00	ALT4
WDOG2	WDOG2_B	ENET1_RX_CLK	ALT1
	WDOG2_RESET_B_DEB	ENET1_CRS	ALT1
WDOG3	WDOG3_B	GPIO1_IO14	ALT4
		I2C2_SCL	ALT2
	WDOG3_RESET_B_DEB	I2C2_SDA	ALT2
WDOG4	WDOG4_B	GPIO1_IO15	ALT4
		I2C4_SCL	ALT2
	WDOG4_RESET_B_DEB	I2C4_SDA	ALT2
XTALOSC	ONOFF	ONOFF	No muxing
	REF_CLK_24M	GPIO1_IO01	ALT4
		I2C1_SCL	ALT4
	REF_CLK_32K	GPIO1_IO02	ALT4
	RTC_XTALI	RTC_XTALI	No muxing
	RTC_XTALO	RTC_XTALO	No muxing
	XTALI	XTALI	No muxing
	XTALO	XTALO	No muxing

8.2 IOMUX Controller (IOMUXC)

8.2.1 Overview

The IOMUX Controller (IOMUXC), together with the IOMUX, enables the IC to share one pad to several functional blocks. This sharing is done by multiplexing the pad's input and output signals.

Every module requires a specific pad setting (such as pull up or keeper), and for each pad, there are up to 8 muxing options (called ALT modes). The pad settings parameters are controlled by the IOMUXC.

The IOMUX consists only of combinatorial logic combined from several basic IOMUX cells. Each basic IOMUX cell handles only one pad signal's muxing.

Figure 8-1 illustrates the IOMUX/IOMUXC connectivity in the system.

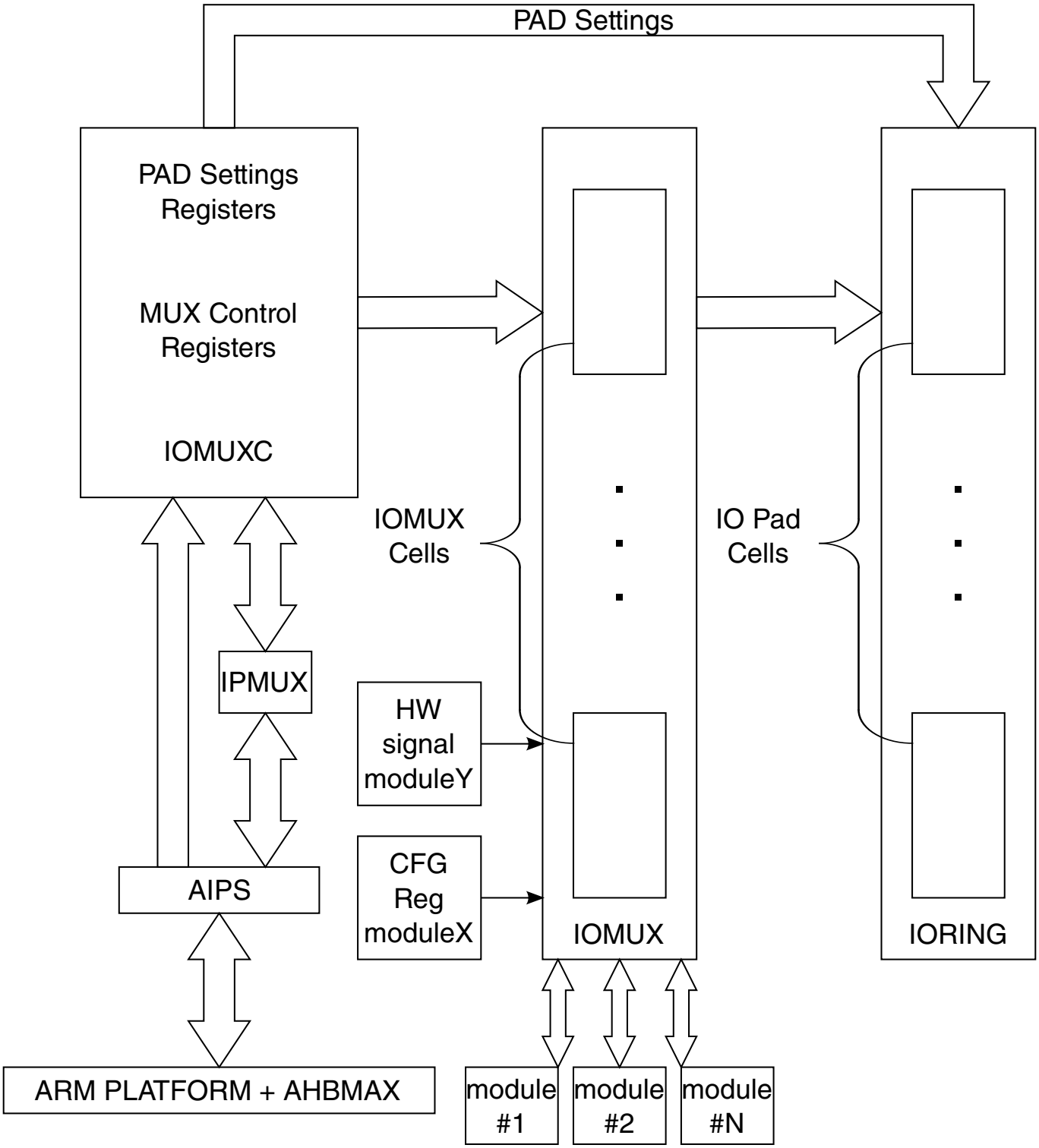


Figure 8-1. IOMUX SoC Level Block Diagram

8.2.1.1 Features

The IOMUXC features are:

- 32-bit software mux control registers (IOMUXC_SW_MUX_CTL_PAD_<PAD NAME> or IOMUXC_SW_MUX_CTL_GRP_<GROUP NAME>) to configure 1 of 8 alternate (ALT) MUX_MODE feilds of each pad or a predefined group of pads and to enable the forcing of an input path of the pad(s) (SION bit).
- 32-bit software pad control registers (IOMUXC_SW_PAD_CTL_PAD_<PAD_NAME> or IOMUXC_SW_PAD_CTL_GRP_<GROUP NAME>) to configure specific pad settings of each pad, or a predefined group of pads.
- 32-bit general purpose registers - 14 (GPR0 to GPR13) 32-bit registers according to SoC requirements for any usage.
- 32-bit input select control registers to control the input path to a module when more than one pad drives this module input.

Each SW MUX/PAD CTL IOMUXC register handles only one pad or one pad's group.

Only the minimum number of registers required by software are implemented by hardware. For example, if only ALT0 and ALT1 modes are used on Pad x then only one bit register will be generated as the MUX_MODE control field in the software mux control register of Pad x.

The software mux control registers may allow the forcing of pads to become input (input path enabled) regardless of the functional direction driven. This may be useful for loopback and GPIO data capture.

8.2.2 Clocks

The table found here describes the clock sources for IOMUXC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 8-1. IOMUXC Clocks

Clock name	Clock Root	Description
ipt_clk_io		IO clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

8.2.3 Functional description

This section provides a complete functional description of the block.

The IOMUXC consists of two sub-blocks:

- IOMUXC_REGISTERS includes all of the IOMUXC registers (see Features).
- IOMUXC_LOGIC includes all of the IOMUXC combinatorial logic (IP interface controls, address decoder, observability muxes).

The IOMUX consists of a number (about the number of pads in the SoC) of basic iomux_cell units. If only one functional mode is required for a specific pad, there is no need for IOMUX and the signals can be connected directly from the module to the I/O. The IOMUX cell is required whenever two or more functional modes are required for a specific pad or when one functional mode and the one test mode are required.

The basic iomux_cell design, which allows two levels of HW signal control (in ALT6 and ALT7 modes - ALT7 gets highest priority) is shown in Figure 8-2.

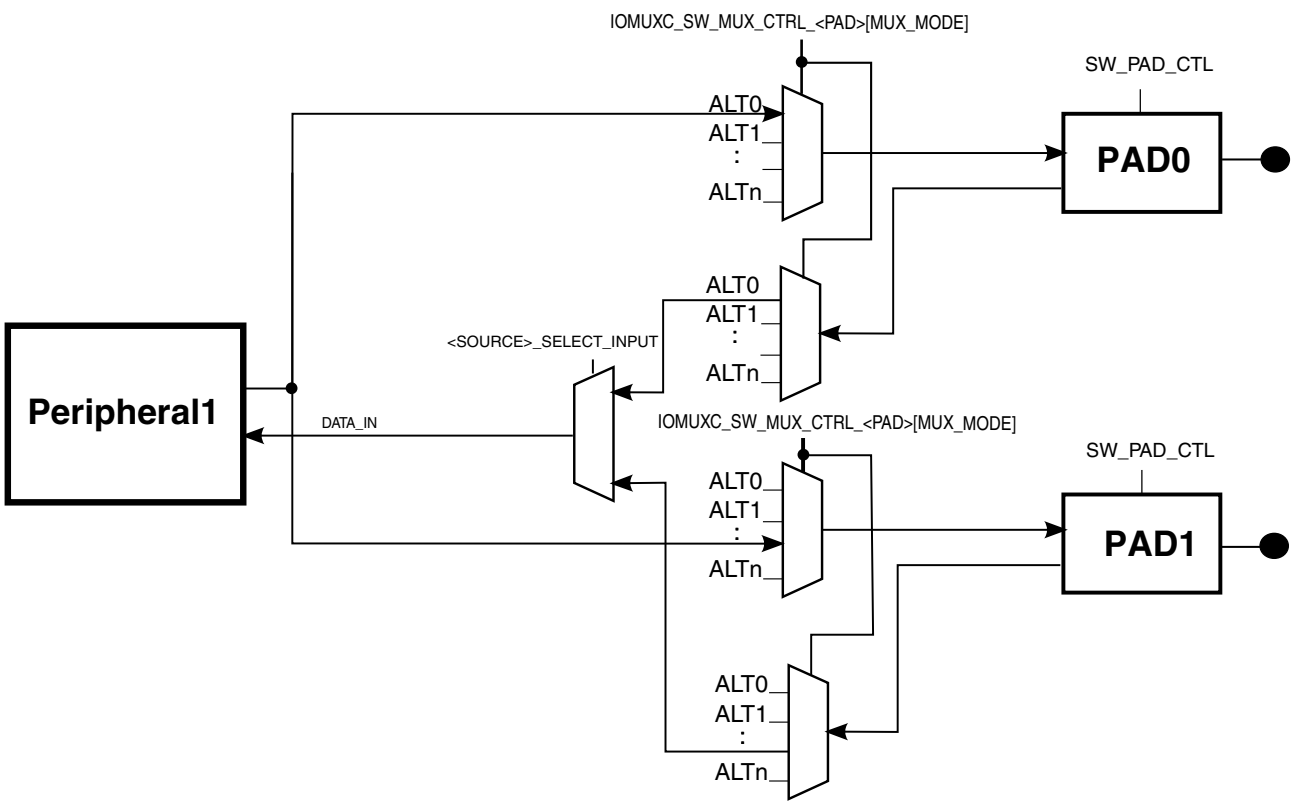


Figure 8-2. IOMUX Cell Block Diagram

8.2.3.1 ALT6 and ALT7 extended muxing modes

The ALT7 and ALT6 extended muxing modes allow any signal in the system (such as fuse, pad input, JTAG, or software register) to override any software configuration and to force the ALT6/ALT7 muxing mode.

It also allows an IOMUX software register to control a group of pads.

8.2.3.2 SW Loopback through SION bit

A limited option exists to override the default pad functionality and force the input path to be active (ipp_ibe==1'b1) regardless of the value driven by the corresponding module. This can be done by setting the SION (Software Input On) bit in the IOMUXC_SW_MUX_CTL register (when available) to "1".

Uses include:

- LoopBack - Module x drives the pad and also receives pad value as an input.
- GPIO Capture - Module x drives the pad and the value is captured by GPIO.

8.2.3.3 Daisy chain - multi pads driving same module input pin

In some cases, more than one pad may drive a single module input pin. Such cases require the addition of one more level of IOMUXing; all of these input signals are muxed, and a dedicated software controlled register controls the mux in order to select the required input path.

A module port involved in "daisy chain" requires two software configuration commands, one for selecting the mode for this pad (programable via the IOMUXC_SW_MUX_CTL_<PAD> registers) and one for defining it as the input path (via the daisy chain registers).

This means that a module port involved in "daisy chain" requires two software configuration commands, one for selecting the mode for this pad (programable via the IOMUXC_SW_MUX_CTL_<PAD> registers) and one for defining it as the input path (via the daisy chain registers). The daisy chain is illustrated in the figure below.

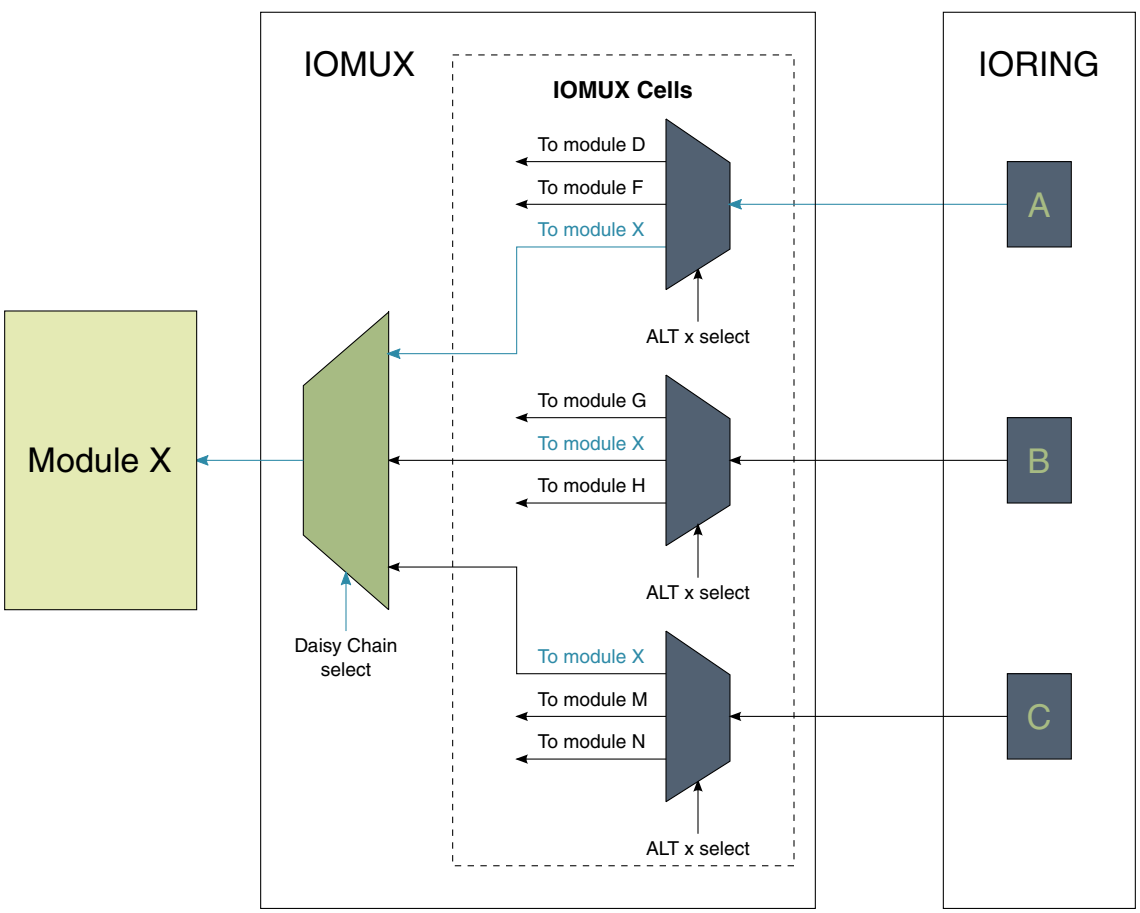


Figure 8-3. Daisy chain illustration

8.2.4 IOMUXC GPR Memory Map/Register Definition

IOMUXC_GPR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3034_0000	GPR0 General Purpose Register (IOMUXC_GPR_GPR0)	32	R/W	0000_0000h	8.2.4.1/ 1483
3034_0004	GPR1 General Purpose Register (IOMUXC_GPR_GPR1)	32	R/W	0F40_0005h	8.2.4.2/ 1485
3034_0008	GPR2 General Purpose Register (IOMUXC_GPR_GPR2)	32	R/W	0000_0000h	8.2.4.3/ 1488
3034_000C	GPR3 General Purpose Register (IOMUXC_GPR_GPR3)	32	R/W	0000_FFFFh	8.2.4.4/ 1490
3034_0010	GPR4 General Purpose Register (IOMUXC_GPR_GPR4)	32	R/W	0000_0000h	8.2.4.5/ 1495
3034_0014	GPR5 General Purpose Register (IOMUXC_GPR_GPR5)	32	R/W	0000_0000h	8.2.4.6/ 1498
3034_0018	GPR6 General Purpose Register (IOMUXC_GPR_GPR6)	32	R/W	0000_0000h	8.2.4.7/ 1500
3034_001C	GPR7 General Purpose Register (IOMUXC_GPR_GPR7)	32	R/W	0000_0000h	8.2.4.8/ 1501
3034_0020	GPR8 General Purpose Register (IOMUXC_GPR_GPR8)	32	R/W	0000_0000h	8.2.4.9/ 1503
3034_0024	GPR9 General Purpose Register (IOMUXC_GPR_GPR9)	32	R/W	0000_0000h	8.2.4.10/ 1504
3034_0028	GPR10 General Purpose Register (IOMUXC_GPR_GPR10)	32	R/W	0000_0003h	8.2.4.11/ 1505
3034_002C	GPR11 General Purpose Register (IOMUXC_GPR_GPR11)	32	R/W	0000_0000h	8.2.4.12/ 1506
3034_0030	GPR12 General Purpose Register (IOMUXC_GPR_GPR12)	32	R/W	3000_0080h	8.2.4.13/ 1507
3034_0034	GPR13 General Purpose Register (IOMUXC_GPR_GPR13)	32	R/W	0000_0000h	8.2.4.14/ 1508
3034_0038	GPR14 General Purpose Register (IOMUXC_GPR_GPR14)	32	R/W	0000_0000h	8.2.4.15/ 1510
3034_003C	GPR15 General Purpose Register (IOMUXC_GPR_GPR15)	32	R/W	0000_0000h	8.2.4.16/ 1511
3034_0040	GPR16 General Purpose Register (IOMUXC_GPR_GPR16)	32	R/W	0000_0000h	8.2.4.17/ 1512
3034_0044	GPR17 General Purpose Register (IOMUXC_GPR_GPR17)	32	R/W	0000_0000h	8.2.4.18/ 1513
3034_0048	GPR18 General Purpose Register (IOMUXC_GPR_GPR18)	32	R/W	0000_0000h	8.2.4.19/ 1514
3034_004C	GPR19 General Purpose Register (IOMUXC_GPR_GPR19)	32	R	0000_0000h	8.2.4.20/ 1516

Table continues on the next page...

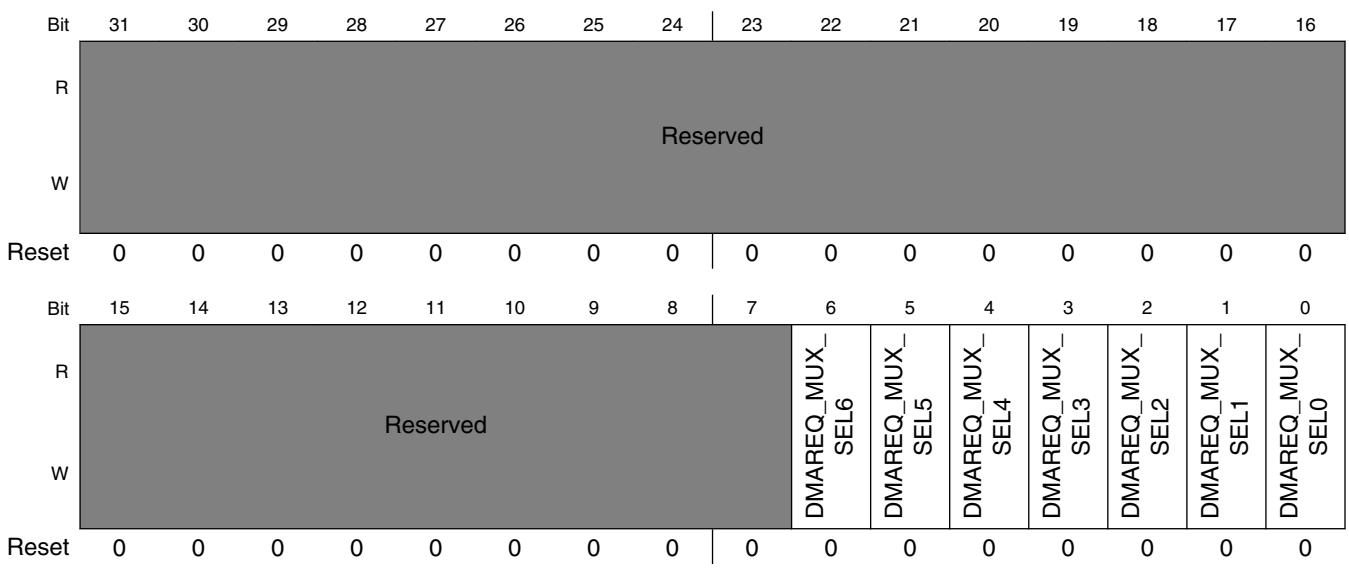
IOMUXC_GPR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3034_0050	GPR20 General Purpose Register (IOMUXC_GPR_GPR20)	32	R/W	0000_0000h	8.2.4.21/ 1518
3034_0054	GPR21 General Purpose Register (IOMUXC_GPR_GPR21)	32	R/W	0000_0000h	8.2.4.22/ 1519
3034_0058	GPR22 General Purpose Register (IOMUXC_GPR_GPR22)	32	R	0000_0000h	8.2.4.23/ 1521

8.2.4.1 GPR0 General Purpose Register (IOMUXC_GPR_GPR0)

GPR Register

Address: 3034_0000h base + 0h offset = 3034_0000h



IOMUXC_GPR_GPR0 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6 DMAREQ_MUX_SEL6	Selects between two possible sources for SDMA_EVENT41 0 GPT4 counter event 1 FTM2 8 channel DMA request
5 DMAREQ_MUX_SEL5	Selects between two possible sources for SDMA_EVENT40 0 GPT3 counter event 1 FTM1 7 channel DMA request

Table continues on the next page...

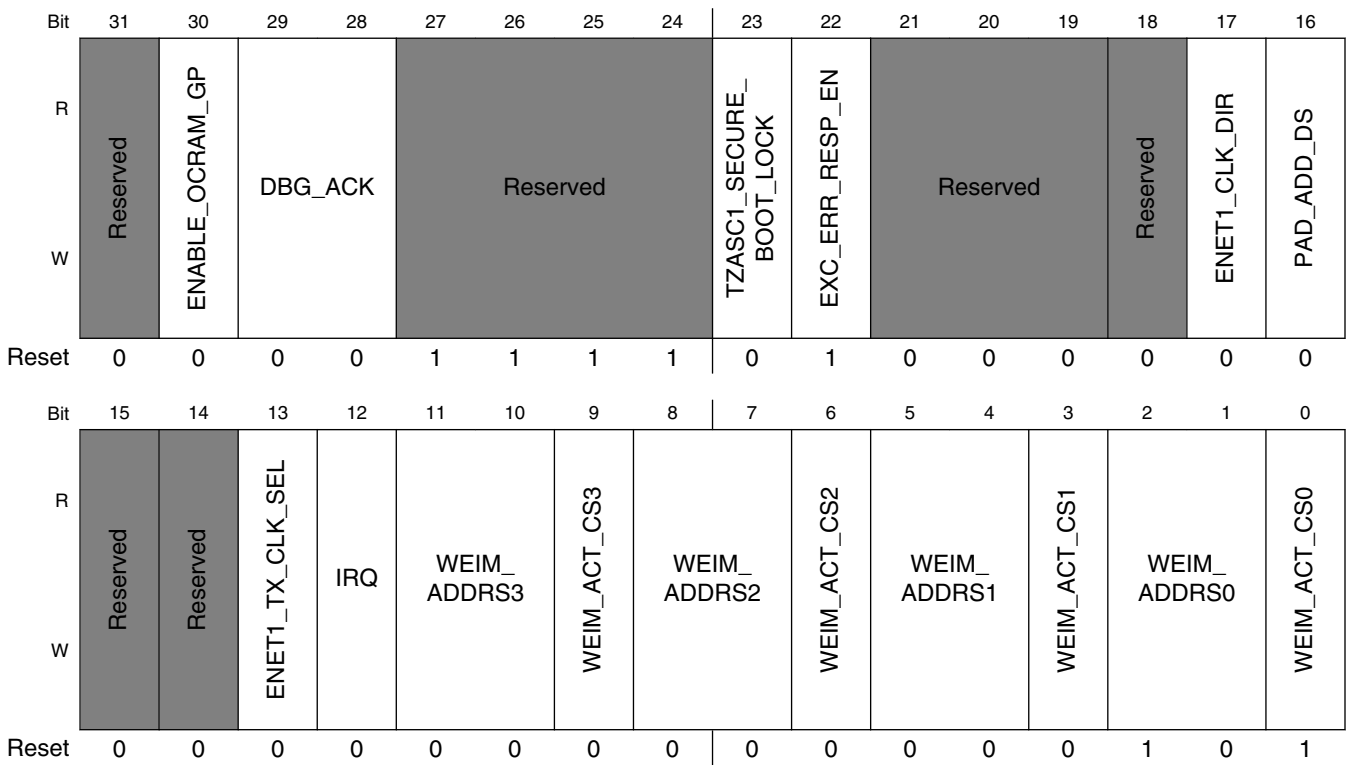
IOMUXC_GPR_GPR0 field descriptions (continued)

Field	Description
4 DMAREQ_MUX_SEL4	Selects between two possible sources for SDMA_EVENT47 0 ENET1 1588 Event1 out 1 Reserved
3 DMAREQ_MUX_SEL3	Selects between two possible sources for SDMA_EVENT21 0 I2C4 DMA event 1 SIM2 transmit DMA request
2 DMAREQ_MUX_SEL2	Selects between two possible sources for SDMA_EVENT20 0 I2C3 DMA event 1 SIM1 receive DMA request
1 DMAREQ_MUX_SEL1	Selects between two possible sources for SDMA_EVENT19 0 I2C2 DMA event 1 SIM1 transmit DMA request
0 DMAREQ_MUX_SEL0	Selects between two possible sources for SDMA_EVENT18 0 I2C1 DMA event 1 SIM1 receive DMA request

8.2.4.2 GPR1 General Purpose Register (IOMUXC_GPR_GPR1)

GPR Register

Address: 3034_0000h base + 4h offset = 3034_0004h



IOMUXC_GPR_GPR1 field descriptions

Field	Description
31 -	This field is reserved. Reserved
30 ENABLE_ OCRAM_GP	Enable On-chip RAM General Purpose 0 Disable On-chip RAM 1 Enable On-chip RAM
29–28 DBG_ACK	Debug Acknowledge, used in sec_wrapper 00 None of the two cores are treated as debug mode by peripherals 01 CA7 platform treated as debug mode when CA7 core0 in debug mode by peripherals 10 CA7 platform treated as debug mode when CA7 core1 in debug mode by peripherals 11 CA7 platform treated as debug mode when any CA7 cores in debug mode by peripherals
27–24 -	This field is reserved. Reserved

Table continues on the next page...

IOMUXC_GPR_GPR1 field descriptions (continued)

Field	Description
23 TZASC1_ SECURE_ BOOT_LOCK	TZASC-1 Secure Boot Lock 0 Secure boot lock is disabled 1 Secure boot lock is enabled
22 EXC_ERR_ RESP_EN	Security exclusive access error response enable for all security gaskets (on both AHB and AXI busses). Enables an ERR response on the AXI vs an OK response for an exclusive access error. 0 OK response on the AXI for an exclusive access error 1 ERR response on the AXI for an exclusive access error
21–19 -	This field is reserved. Reserved
18 -	This field is reserved. Reserved
17 ENET1_CLK_ DIR	ENET1_TX_CLK data direction control when ANATOP ENET_REF_CLK1 is selected (ALT1) 0 ENET1_TX_CLK output driver is disabled when configured for ALT1 1 ENET1_TX_CLK output driver is enabled when configured for ALT1
16 PAD_ADD_DS	Setting ADD_DS to 0 will make the output driver of the SD3 pins ~10% stronger at highest drive strength (DSE=111). This is for use if the I/O buffer operation at WCS and 200MHz is marginal. 0 output driver ~10% stronger 1 output driver is normal
15 -	This field is reserved. Reserved
14 -	This field is reserved. Reserved
13 ENET1_TX_ CLK_SEL	ENET1 reference clock mode select 0 Gets ENET1 TX reference clk. This clock is also output to pins via the IOMUX. ENET_REF_CLK1 function. 1 Gets ENET1 TX reference clk from the ENET1_TX_CLK pin. In this use case, an external OSC provides the clock for both the external PHY and the internal controller
12 IRQ	Interrupt signal which is connected to CPU IRQS[0]. Used to notify cores on exception condition while boot.
11–10 WEIM_ADDRS3	WEIM Address Space 3 Active Chip Select and Address Space. Each of the WEIM_ACT_CSx represents one of the four chip selects of the EIM. When ACT_CSx=1'b1, the corresponding chip select is active and has a valid address space according to its address space configuration determined by WEIM_ADDRSx[1:0] bits. WEIM_ADDRSx[1:0] is setting the space for each chip select which is active. The address space of the first active chip select must be the biggest one, the following active chip select address spaces may be equal or lower. Total address space size is 128 MByte. The supported configurations are: CS0(128M), CS1 (0M), CS2 (0M), CS3(0M) [default configuration]

Table continues on the next page...

IOMUXC_GPR_GPR1 field descriptions (continued)

Field	Description
	CS0(64M), CS1(64M), CS2(0M), CS3(0M) CS0(64M), CS1(32M), CS2(32M), CS3(0M) CS0(32M), CS1(32M), CS2(32M), CS3(32M) Address Space Configuration options (WEIM_ADDRSx[10]):
9 WEIM_ACT_CS3	WEIM Active Chip Select 3 See description for WEIM_ADDRS3
8–7 WEIM_ADDRS2	WEIM Address Space 2 See description for WEIM_ADDRS3
6 WEIM_ACT_CS2	WEIM Active Chip Select 2 See description for WEIM_ADDRS3
5–4 WEIM_ADDRS1	WEIM Address Space 1 See description for WEIM_ADDRS3
3 WEIM_ACT_CS1	WEIM Active Chip Select 1 See description for WEIM_ADDRS3
2–1 WEIM_ADDRS0	WEIM Address Space 0 See description for WEIM_ADDRS3
0 WEIM_ACT_CS0	WEIM Active Chip Select 0 See description for WEIM_ADDRS3

8.2.4.3 GPR2 General Purpose Register (IOMUXC_GPR_GPR2)

GPR Register

Address: 3034_0000h base + 8h offset = 3034_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DRAM_CKE_BYPASS	DRAM_CKE1	DRAM_CKE0	DRAM_RESET	DRAM_RESET_BYPASS	MQS_OVERSAMPLE	MQS_EN	MQS_SW_RST	MQS_CLK_DIV							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MEM_CPU_LS	MEM_CPU_DS	MEM_CPU_SD	MEM_CPU_LOWPOWER	MEM_EPDC_LS	MEM_EPDC_DS	MEM_EPDC_SD	MEM_EPDC_LOWPOWER	MEM_LCDIF_LS	MEM_LCDIF_DS	MEM_LCDIF_SD	MEM_LCDIF_LOWPOWER	MEM_PXP_LS	MEM_PXP_DS	MEM_PXP_SD	MEM_PXP_LOWPOWER
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR2 field descriptions

Field	Description
31 DRAM_CKE_BYPASS	Not used
30 DRAM_CKE1	Not used
29 DRAM_CKE0	Not used
28 DRAM_RESET	Not used
27 DRAM_RESET_BYPASS	Not used
26 MQS_OVERSAMPLE	MQS PWM over-sampling rate option 0 32 1 64
25 MQS_EN	MQS enable

Table continues on the next page...

IOMUXC_GPR_GPR2 field descriptions (continued)

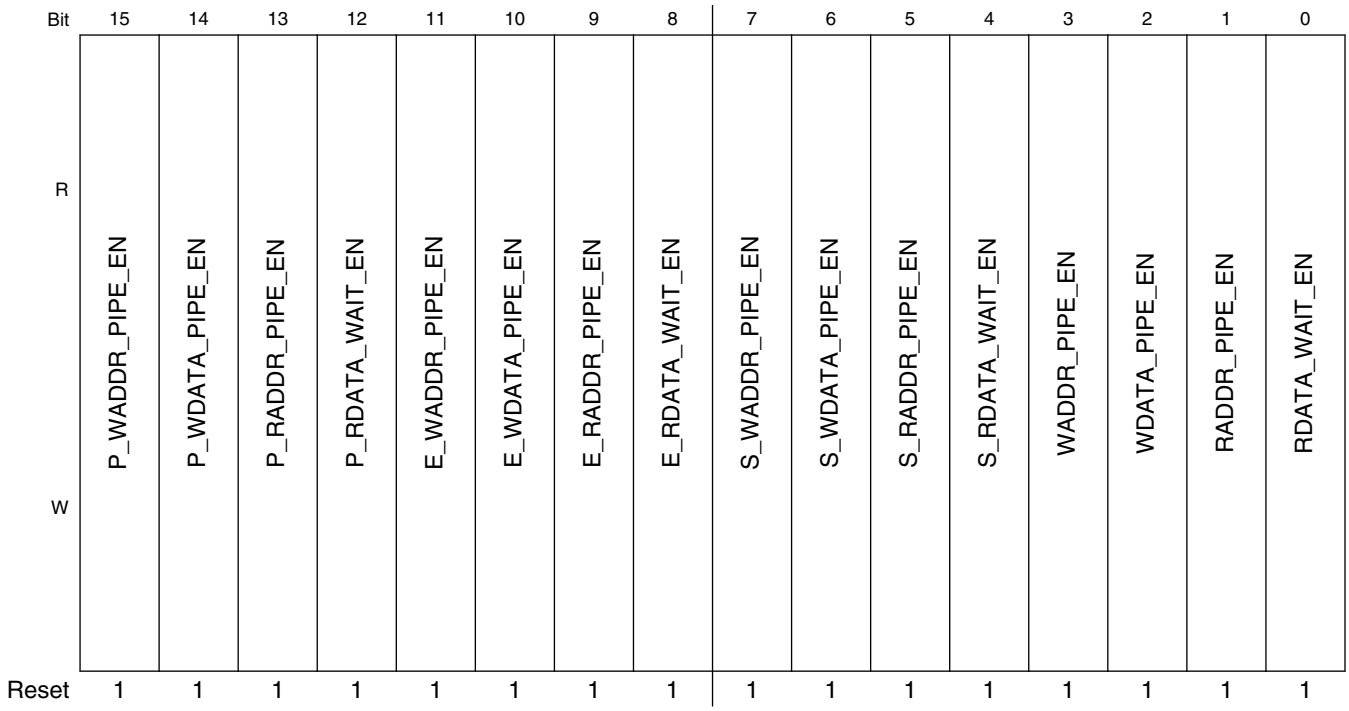
Field	Description
	0 Disable 1 Enable
24 MQS_SW_RST	MQS software reset
23–16 MQS_CLK_DIV	MQS clock divider control
15 MEM_CPU_LS	Not used
14 MEM_CPU_DS	Not used
13 MEM_CPU_SD	Not used
12 MEM_CPU_ LOWPOWER	Not used
11 MEM_EPDC_LS	Not used
10 MEM_EPDC_DS	Not used
9 MEM_EPDC_SD	Not used
8 MEM_EPDC_ LOWPOWER	Not used
7 MEM_LCDIF_LS	Not used
6 MEM_LCDIF_DS	Not used
5 MEM_LCDIF_SD	Not used
4 MEM_LCDIF_ LOWPOWER	Not used
3 MEM_PXP_LS	Not used
2 MEM_PXP_DS	Not used
1 MEM_PXP_SD	Not used
0 MEM_PXP_ LOWPOWER	Not used

8.2.4.4 GPR3 General Purpose Register (IOMUXC_GPR_GPR3)

GPR Register

Address: 3034_0000h base + Ch offset = 3034_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	P_WADDR_PIPE_EN_PNDG	P_WDATA_PIPE_EN_PNDG	P_RADDR_PIPE_EN_PNDG	P_RDATA_WAIT_EN_PNDG	E_WADDR_PIPE_EN_PNDG	E_WDATA_PIPE_EN_PNDG	E_RADDR_PIPE_EN_PNDG	E_RDATA_WAIT_EN_PNDG	S_WADDR_PIPE_EN_PNDG	S_WDATA_PIPE_EN_PNDG	S_RADDR_PIPE_EN_PNDG	S_RDATA_WAIT_EN_PNDG	WADDR_PIPE_EN_PNDG	WDATA_PIPE_EN_PDG	RADDR_PIPE_EN_PDG	RDATA_WAIT_EN_PDG
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



IOMUXC_GPR_GPR3 field descriptions

Field	Description
31 P_WADDR_ PIPE_EN_PNDG	PXP On-chip RAM write address pipeline enable update is pending 0 write address pipeline enable configuration valid 1 write address pipeline enable bit changed
30 P_WDATA_ PIPE_EN_PNDG	PXP On-chip RAM write data pipeline enable update is pending 0 write data pipeline enable configuration valid 1 write data pipeline enable bit changed
29 P_RADDR_ PIPE_EN_PNDG	PXP On-chip RAM read address pipeline enable update is pending 0 read address pipeline enable configuration valid 1 read address pipeline enable bit changed
28 P_RDATA_ WAIT_EN_PNDG	PXP On-chip RAM read data wait state control update is pending 0 read data wait state control configuration valid 1 read data wait state control bit changed
27 E_WADDR_ PIPE_EN_PNDG	On-chip RAM write address pipeline enable update is pending 0 write address pipeline enable configuration valid 1 write address pipeline enable bit changed
26 E_WDATA_ PIPE_EN_PNDG	On-chip RAM write data pipeline enable update is pending 0 write data pipeline enable configuration valid 1 write data pipeline enable bit changed

Table continues on the next page...

IOMUXC_GPR_GPR3 field descriptions (continued)

Field	Description
25 E_RADDR_ PIPE_EN_PNDG	On-chip RAM read address pipeline enable update is pending 0 read address pipeline enable configuration valid 1 read address pipeline enable bit changed
24 E_RDATA_ WAIT_EN_PNDG	On-chip RAM read data wait state control update is pending 0 read data wait state control configuration valid 1 read data wait state control bit changed
23 S_WADDR_ PIPE_EN_PNDG	State Retention On-chip RAM write address pipeline enable update is pending 0 write address pipeline enable configuration valid 1 write address pipeline enable bit changed
22 S_WDATA_ PIPE_EN_PNDG	State Retention On-chip RAM write data pipeline enable update is pending 0 write data pipeline enable configuration valid 1 write data pipeline enable bit changed
21 S_RADDR_ PIPE_EN_PNDG	State Retention On-chip RAM read address pipeline enable update is pending 0 read address pipeline enable configuration valid 1 read address pipeline enable bit changed
20 S_RDATA_ WAIT_EN_PNDG	State Retention On-chip RAM read data wait state control update is pending 0 read data wait state control configuration valid 1 read data wait state control bit changed
19 WADDR_PIPE_ EN_PNDG	On-chip RAM write address pipeline enable update is pending 0 write address pipeline enable configuration valid 1 write address pipeline enable bit changed
18 WDATA_PIPE_ EN_PDG	On-chip RAM write data pipeline enable update is pending 0 write data pipeline enable configuration valid 1 write data pipeline enable bit changed
17 RADDR_PIPE_ EN_PDG	On-chip RAM read address pipeline enable update is pending 0 read address pipeline enable configuration valid 1 read address pipeline enable bit changed
16 RDATA_WAIT_ EN_PDG	On-chip RAM read data wait state control update is pending 0 read data wait state control configuration valid 1 read data wait state control bit changed
15 P_WADDR_ PIPE_EN	PXP On-chip RAM write address pipeline enable For description, please refer to WADDR_PIPE_EN bit
14 P_WDATA_ PIPE_EN	PXP On-chip RAM write data pipeline enable For description, please refer to WDATA_PIPE_EN bit
13 P_RADDR_ PIPE_EN	PXP On-chip RAM read address pipeline enable For description, please refer to RADDR_PIPE_EN bit

Table continues on the next page...

IOMUXC_GPR_GPR3 field descriptions (continued)

Field	Description
12 P_RDATA_WAIT_EN	PXP On-chip RAM read data wait state control For description, please refer to RDATA_WAIT_EN bit
11 E_WADDR_PIPE_EN	On-chip RAM write address pipeline enable For description, please refer to WADDR_PIPE_EN bit
10 E_WDATA_PIPE_EN	On-chip RAM write data pipeline enable For description, please refer to WDATA_PIPE_EN bit
9 E_RADDR_PIPE_EN	On-chip RAM read address pipeline enable For description, please refer to RADDR_PIPE_EN bit
8 E_RDATA_WAIT_EN	On-chip RAM read data wait state control For description, please refer to RDATA_WAIT_EN bit
7 S_WADDR_PIPE_EN	State Retention On-chip RAM write address pipeline enable For description, please refer to WADDR_PIPE_EN bit
6 S_WDATA_PIPE_EN	State Retention On-chip RAM write data pipeline enable For description, please refer to WDATA_PIPE_EN bit
5 S_RADDR_PIPE_EN	State Retention On-chip RAM read address pipeline enable For description, please refer to RADDR_PIPE_EN bit
4 S_RDATA_WAIT_EN	State Retention On-chip RAM read data wait state control For description, please refer to RDATA_WAIT_EN bit
3 WADDR_PIPE_EN	On-chip RAM write address pipeline enable When this feature is enabled, the write address from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the write access on the memory cell at high frequency. Enable this feature would cost at most 1 more clock cycle for each AXI write transaction, i.e., at most 1 more clock cycle for each write burst with multiple beats of data. When this feature is disabled, the write address from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write data is also ready at this cycle). 0 write address pipeline is disabled 0 write address pipeline is disabled 1 write address pipeline is enabled
2 WDATA_PIPE_EN	On-chip RAM write data pipeline enable When this feature is enabled, the write data from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the write access on the memory cell at high frequency. Enable this feature would cost at most 1 more clock cycle for each AXI write transaction, i.e., at most 1 more clock cycle for each write burst with multiple beats of data. When this feature is disabled, the write data from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write address is also ready at this cycle).

Table continues on the next page...

IOMUXC_GPR_GPR3 field descriptions (continued)

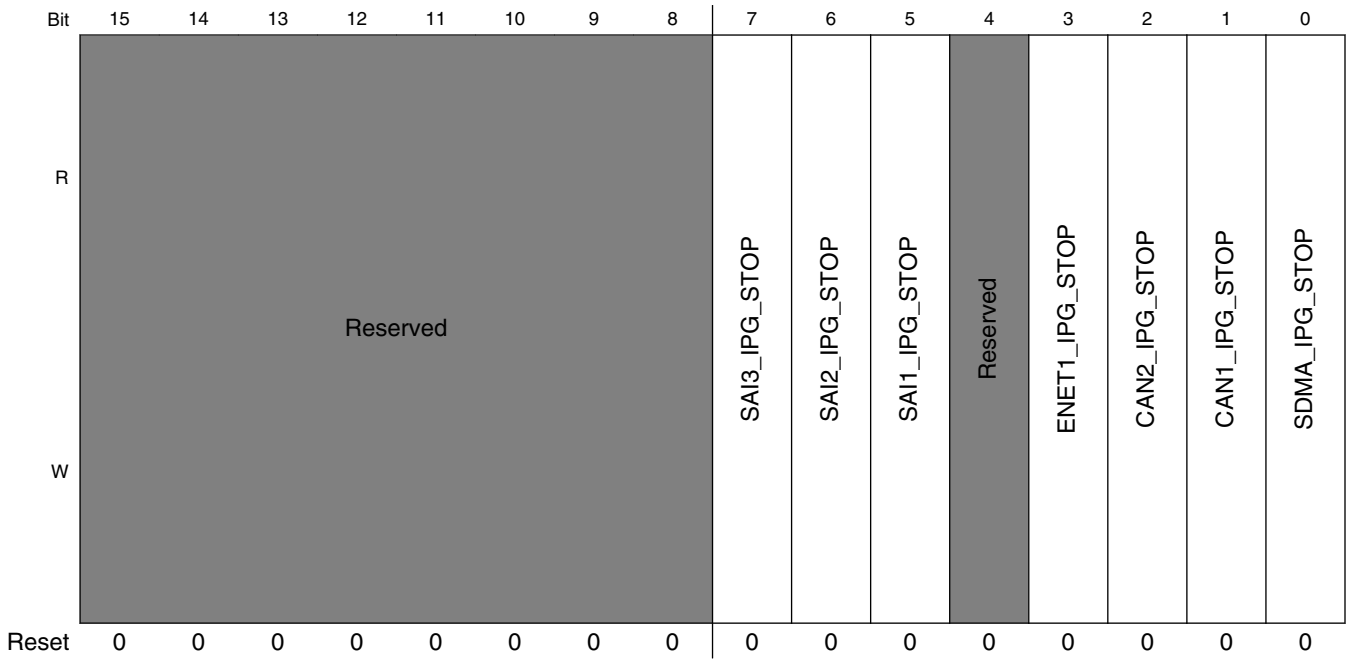
Field	Description
	0 write data pipeline is disabled 1 write data pipeline is enabled
1 RADDR_PIPE_EN	On-chip RAM read address pipeline enable When this feature is enabled, the read address from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the read access on the memory cell at high frequency. Enable this feature would cost at most 1 more clock cycle for each AXI read transaction, i.e., at most 1 more clock cycle for each read burst with multiple beats of data. When this feature is disabled, the read address from the AXI master can be accepted by the on-chip RAM without delay, and data can become ready for master at next clock cycle (if no other access and no read data wait). 0 read address pipeline is disabled 1 read address pipeline is enabled
0 RDATA_WAIT_EN	On-chip RAM read data wait state control When the read data wait state is enabled, it will cost 2 cycles for each read access, (each beat of a read burst). This can avoid the potential timing problem caused by the relatively longer memory access time at higher frequency. When this feature is disabled, it only costs 1 clock cycle to finish a read transaction, i.e., get read data back in the next cycle of read request becomes valid on the bus. 0 read data wait state disabled 1 read data wait state enabled

8.2.4.5 GPR4 General Purpose Register (IOMUXC_GPR_GPR4)

GPR Register

Address: 3034_0000h base + 10h offset = 3034_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			CPU_STANDBYWFE		CPU_STANDBYWFI		0	SAI3_IPG_STOP_ACK	SAI2_IPG_STOP_ACK	SAI1_IPG_STOP_ACK	Reserved	ENET1_IPG_STOP_ACK	CAN2_IPG_STOP_ACK	CAN1_IPG_STOP_ACK	SDMA_IPG_STOP_ACK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



IOMUXC_GPR_GPR4 field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28–27 CPU_ STANDBYWFE	Status of CPU STANDBYWFE low power states MSB: status of A7 core 1 STANDBYWFE low power state LSB: status of A7 core 0 STANDBYWFE low power state
26–25 CPU_ STANDBYWFI	Status of CPU STANDBYWFI low power states MSB: status of A7 core 1 STANDBYWFI low power state LSB: status of A7 core 0 STANDBYWFI low power state
24 Reserved	This read-only field is reserved and always has the value 0.
23 SAI3_IPG_ STOP_ACK	SAI3 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted, peripheral is in STOP mode
22 SAI2_IPG_ STOP_ACK	SAI2 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted, peripheral is in STOP mode
21 SAI1_IPG_ STOP_ACK	SAI1 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted, peripheral is in STOP mode
20 -	This field is reserved. Reserved

Table continues on the next page...

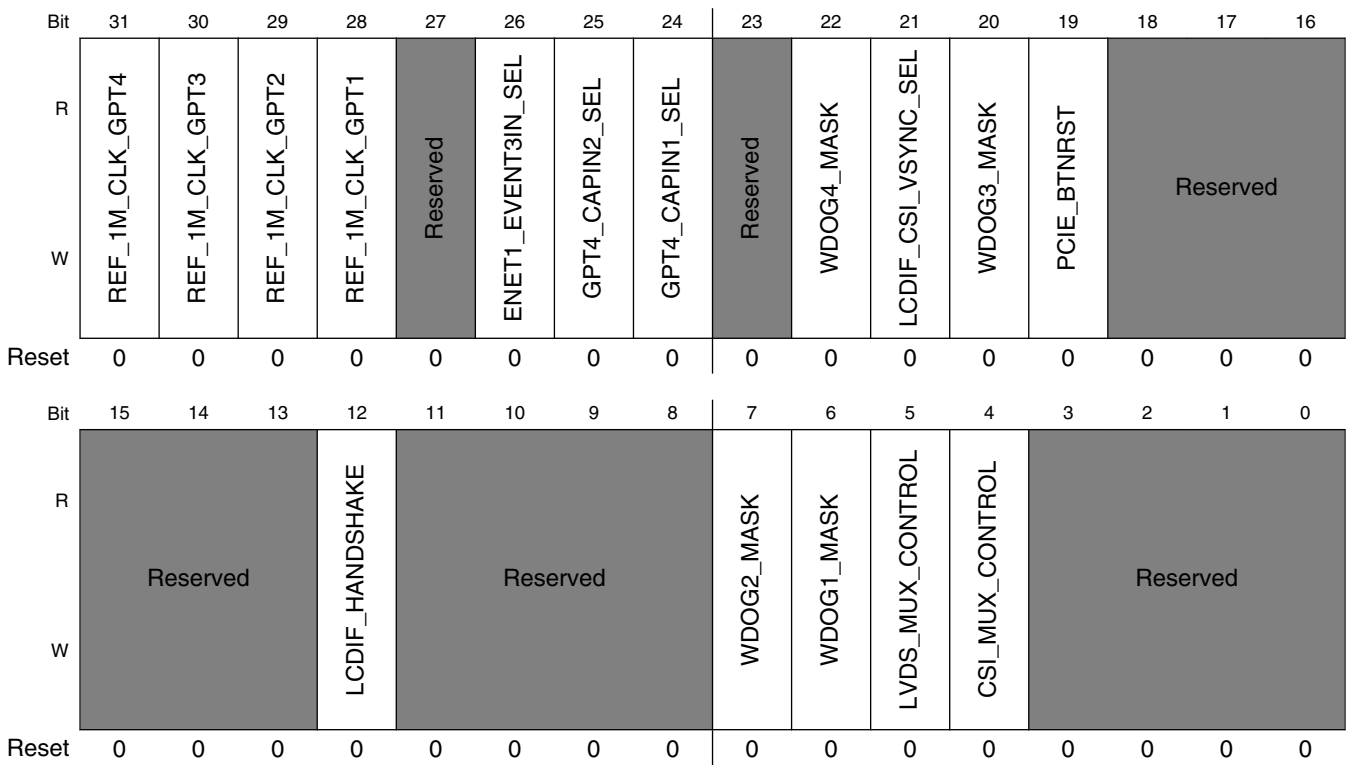
IOMUXC_GPR_GPR4 field descriptions (continued)

Field	Description
19 ENET1_IPG_ STOP_ACK	ENET1 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted, peripheral is in STOP mode
18 CAN2_IPG_ STOP_ACK	CAN2 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted, peripheral is in STOP mode
17 CAN1_IPG_ STOP_ACK	CAN1 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted, peripheral is in STOP mode
16 SDMA_IPG_ STOP_ACK	SDMA stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted, peripheral is in STOP mode
15–8 -	This field is reserved. Reserved
7 SAI3_IPG_STOP	SAI3 stop request 0 stop request off 1 stop request on
6 SAI2_IPG_STOP	SAI2 stop request 0 stop request off 1 stop request on
5 SAI1_IPG_STOP	SAI1 stop request 0 stop request off 1 stop request on
4 -	This field is reserved. Reserved
3 ENET1_IPG_ STOP	ENET1 stop request 0 stop request off 1 stop request on
2 CAN2_IPG_ STOP	CAN2 stop request 0 stop request off 1 stop request on
1 CAN1_IPG_ STOP	CAN1 stop request 0 stop request off 1 stop request on
0 SDMA_IPG_ STOP	SDMA stop request 0 stop request off 1 stop request on

8.2.4.6 GPR5 General Purpose Register (IOMUXC_GPR_GPR5)

GPR Register

Address: 3034_0000h base + 14h offset = 3034_0014h



IOMUXC_GPR_GPR5 field descriptions

Field	Description
31 REF_1M_CLK_GPT4	Reference 1M clock GPT4 0 GPT ipg_clk_highfreq driven by CCM GPT clock 1 GPT ipg_clk_highfreq driven by anatop 1MHz clock
30 REF_1M_CLK_GPT3	GPT3 1MHz clock source select
29 REF_1M_CLK_GPT2	GPT2 1MHz clock source select 0 GPT ipg_clk_highfreq driven by CCM GPT clock 1 GPT ipg_clk_highfreq driven by anatop 1MHz clock
28 REF_1M_CLK_GPT1	GPT1 1MHz clock source select 0 GPT ipg_clk_highfreq driven by CCM GPT clock 1 GPT ipg_clk_highfreq driven by anatop 1MHz clock

Table continues on the next page...

IOMUXC_GPR_GPR5 field descriptions (continued)

Field	Description
27 -	This field is reserved. Reserved
26 ENET1_ EVENT3IN_SEL	ENET1 Mac 0 1588 Timer Event 3 Select 0 Timer event is driven by PAD 1 Timer event is driven by GPT4 Channel 1 Compare Output
25 GPT4_CAPIN2_ SEL	GPT4 Capture Channel 2 Trigger Signal Select 0 Select GPT4 CAPTURE2 signal from PAD 1 Reserved
24 GPT4_CAPIN1_ SEL	GPT4 Capture Channel 1 Trigger Signal Select 0 Select GPT4 CAPTURE1 signal from PAD 1 Select ENET1 1588 Event 3
23 -	This field is reserved. Reserved
22 WDOG4_MASK	WDOG4 Timeout Mask 0 WDOG4 Timeout behaves normally 1 WDOG4 Timeout is masked
21 LCDIF_CSI_ VSYNC_SEL	Not used
20 WDOG3_MASK	WDOG3 Timeout Mask 0 WDOG3 Timeout behaves normally 1 WDOG3 Timeout is masked
19 PCIE_BTNRST	Not used
18–13 -	This field is reserved. Reserved
12 LCDIF_ HANDSHAKE	LCDIF Input Handshake Select 0 LCDIF input handshake signals come from CSI 1 LCDIF input handshake signals come from PXP
11–8 -	This field is reserved. Reserved
7 WDOG2_MASK	WDOG2 Timeout Mask 0 WDOG2 Timeout behaves normally 1 WDOG2 Timeout is masked
6 WDOG1_MASK	WDOG1 Timeout Mask 0 WDOG1 Timeout behaves normally 1 WDOG1 Timeout is masked
5 LVDS_MUX_ CONTROL	Not used

Table continues on the next page...

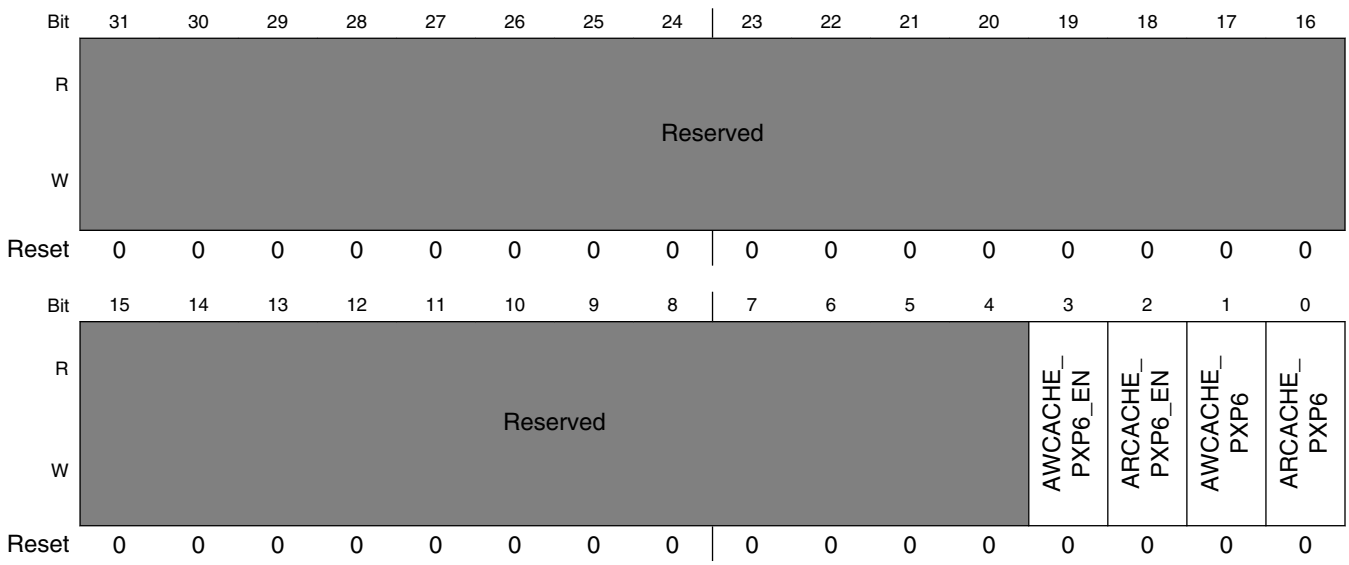
IOMUXC_GPR_GPR5 field descriptions (continued)

Field	Description
4 CSI_MUX_ CONTROL	CSI Input MUX Control 0 CSI input is driven by Parallel CSI 1 CSI input is driven by MIPI CSI
-	This field is reserved. Reserved

8.2.4.7 GPR6 General Purpose Register (IOMUXC_GPR_GPR6)

GPR Register

Address: 3034_0000h base + 18h offset = 3034_0018h



IOMUXC_GPR_GPR6 field descriptions

Field	Description
31–4 -	This field is reserved. Reserved
3 AWCACHE_ PXP6_EN	PXP Secondary AXI Master Port AWCACHE Override Enable 0 PXP Secondary AXI Master Port AWCACHE[1] driven by PXP 1 PXP Secondary AXI Master Port AWCACHE[1] driven to constant value specified by the AWCACHE_PXP6 bit
2 ARCACHE_ PXP6_EN	PXP Secondary AXI Master Port ARCACHE Override Enable

Table continues on the next page...

IOMUXC_GPR_GPR6 field descriptions (continued)

Field	Description
	0 PXP Secondary AXI Master Port ARCACHE[1] driven by PXP 1 PXP Secondary AXI Master Port ARCACHE[1] driven to constant value specified by the ARCACHE_PXP6 bit
1 AWCACHE_PXP6	PXP Secondary AXI Master Port AWCACHE Override Value Note: this bit only takes effect when AWCACHE_PXP6_EN = 1 0 Drive PXP Secondary AXI Master Port AWCACHE[1] to 0 1 Drive PXP Secondary AXI Master Port AWCACHE[1] to 1
0 ARCACHE_PXP6	PXP Secondary AXI Master Port ARCACHE Override Value Note: this bit only takes effect when ARCACHE_PXP6_EN = 1 0 Drive PXP Secondary AXI Master Port ARCACHE[1] to 0 1 Drive PXP Secondary AXI Master Port ARCACHE[1] to 1

8.2.4.8 GPR7 General Purpose Register (IOMUXC_GPR_GPR7)

GPR Register

Address: 3034_0000h base + 1Ch offset = 3034_001Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		-		-		Reserved		-		-		OTG1_LDO_USB_1P0_EN_PWRUPLOAD		-	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

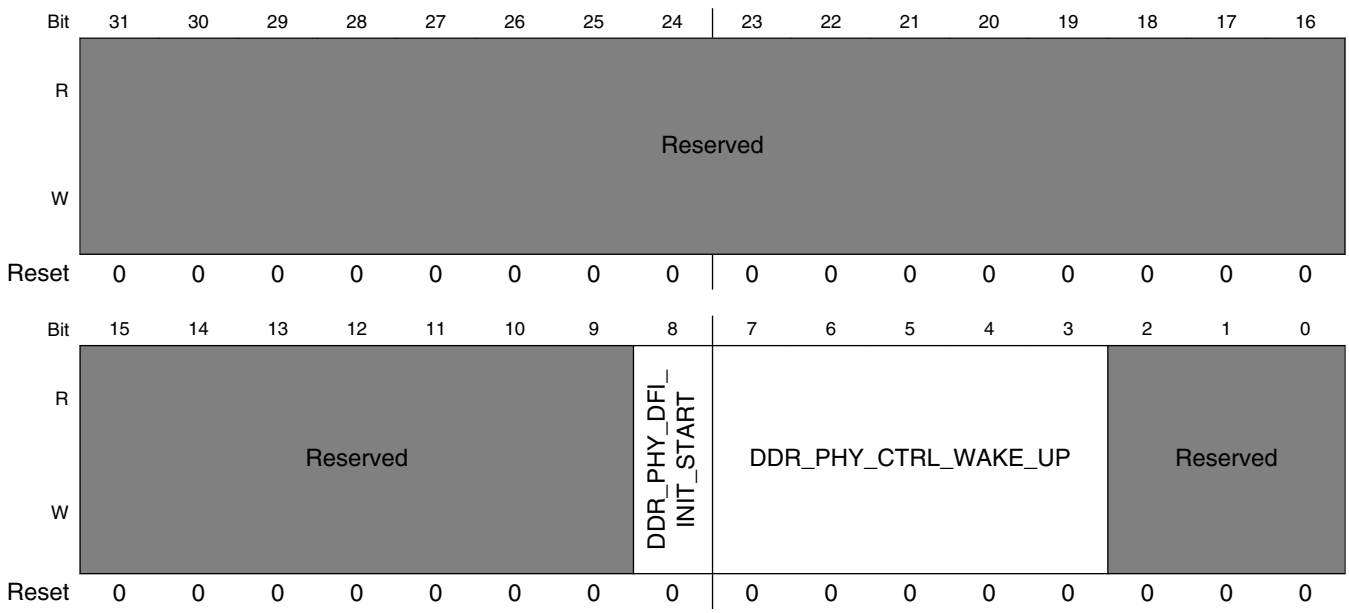
IOMUXC_GPR_GPR7 field descriptions

Field	Description
31–14 -	This field is reserved. Reserved
13–12 -	Reserved. Customers must not change the value of this bit field.
11–10 -	Reserved. Customers must not change the value of this bit field.
9 -	This field is reserved. Reserved
8–7 -	Reserved. Customers must not change the value of this bit field. The value read on this bit field may change after execution of Boot ROM code.
6 -	Reserved. Writes to this bit field will have no effect.
5–4 -	Reserved. Customers must not change the value of this bit field.
3 OTG1_LDO_USB_1P0_EN_PWRUPLOAD	This bit field controls an internal resistive load on the OTG1 PHY's LDO_USB_1P0 output pin, VDD_USB_OTG1_1P0_CAP. Enabling this load pulls the LDO_USB_1P0 supply to ground when no 3.3v USB power is present on the VDD_USB_OTG1_3P3_IN pin. 0 Internal resistive pulldown for LDO_USB_1P0 is disabled. (Default) 1 Internal resistive pulldown for LDO_USB_1P0 is enabled.
2–1 -	Reserved. Customers must not change the value of this bit field. The value read on this bit field may change after execution of Boot ROM code.
0 -	Reserved. Writes to this bit field will have no effect.

8.2.4.9 GPR8 General Purpose Register (IOMUXC_GPR_GPR8)

GPR Register

Address: 3034_0000h base + 20h offset = 3034_0020h



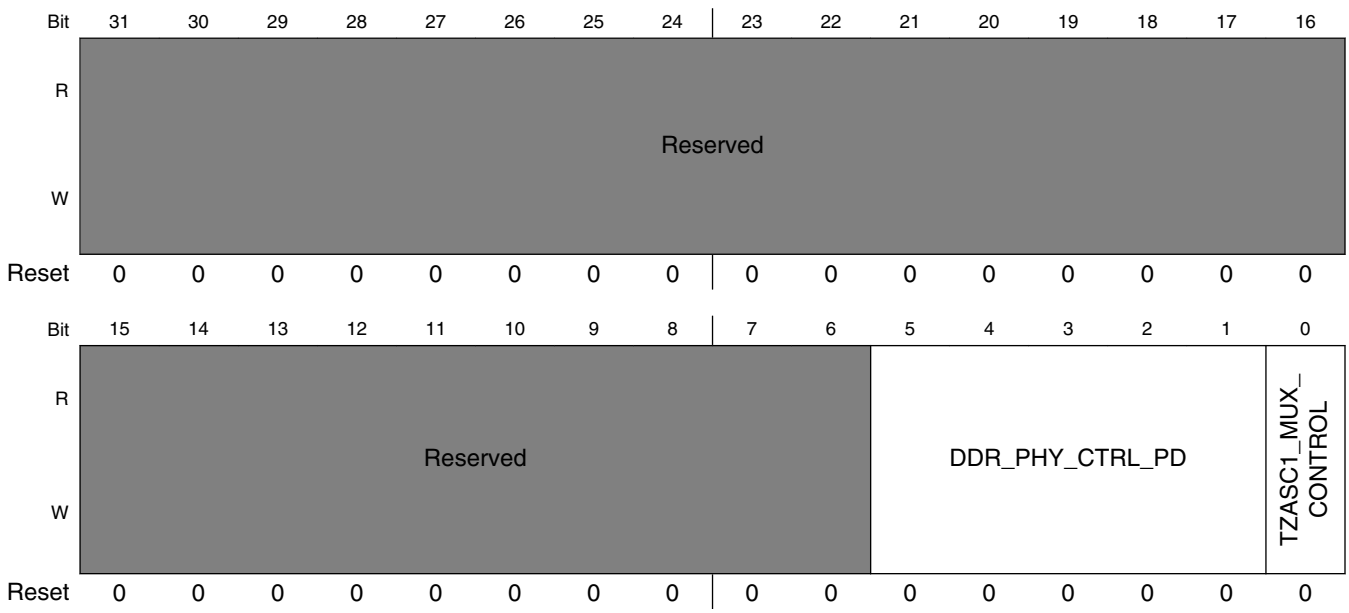
IOMUXC_GPR_GPR8 field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 DDR_PHY_DFI_INIT_START	Drives dfi_init_start input on DDR PHY
7–3 DDR_PHY_CTRL_WAKE_UP	Drives ctrl_wake_up[4:0] inputs on DDR PHY
-	This field is reserved. Reserved

8.2.4.10 GPR9 General Purpose Register (IOMUXC_GPR_GPR9)

GPR Register

Address: 3034_0000h base + 24h offset = 3034_0024h



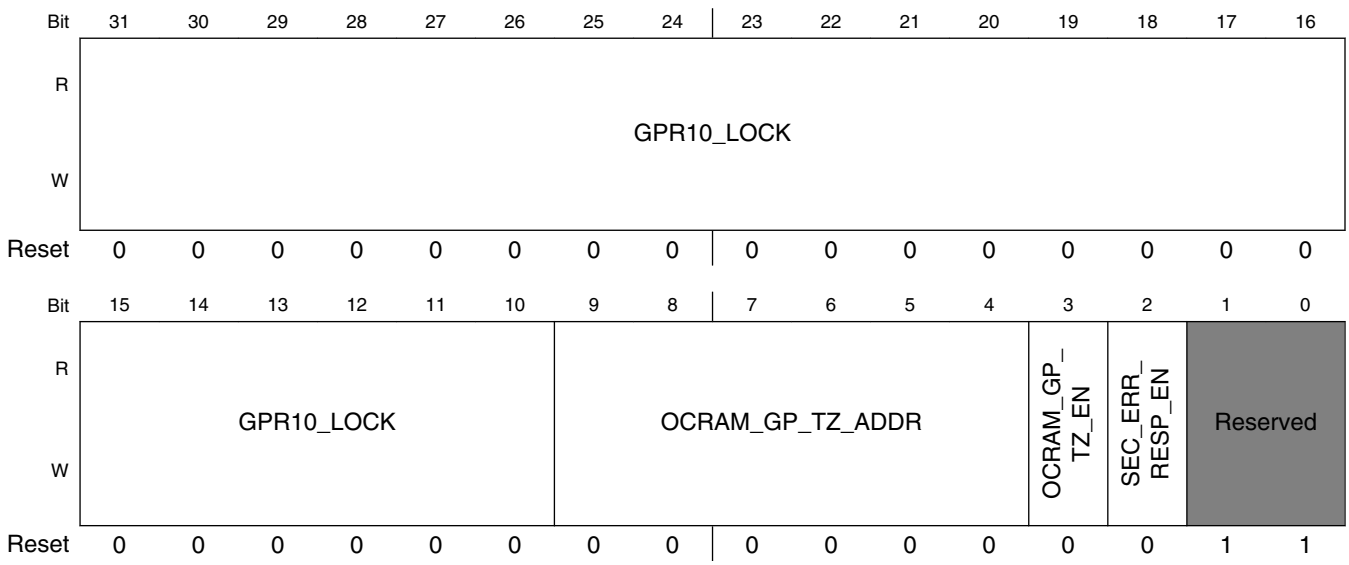
IOMUXC_GPR_GPR9 field descriptions

Field	Description
31–6 -	This field is reserved. Reserved
5–1 DDR_PHY_CTRL_PD	Drives ctrl_pd[4:0] inputs on DDR PHY Each bit is "sticky" to 1. Once a bit is written to 1, it cannot be written back to 0.
0 TZASC1_MUX_CONTROL	TrustZone Address Space Controller Select This bit is "sticky" to 1. Once this bit is written to 1, it cannot be written back to 0. 0 AXI bus connected to DDR Controller bypasses the TrustZone Address Space Controller 1 AXI bus connected to DDR Controller passes through the TrustZone Address Space Controller

8.2.4.11 GPR10 General Purpose Register (IOMUXC_GPR_GPR10)

GPR Register

Address: 3034_0000h base + 28h offset = 3034_0028h



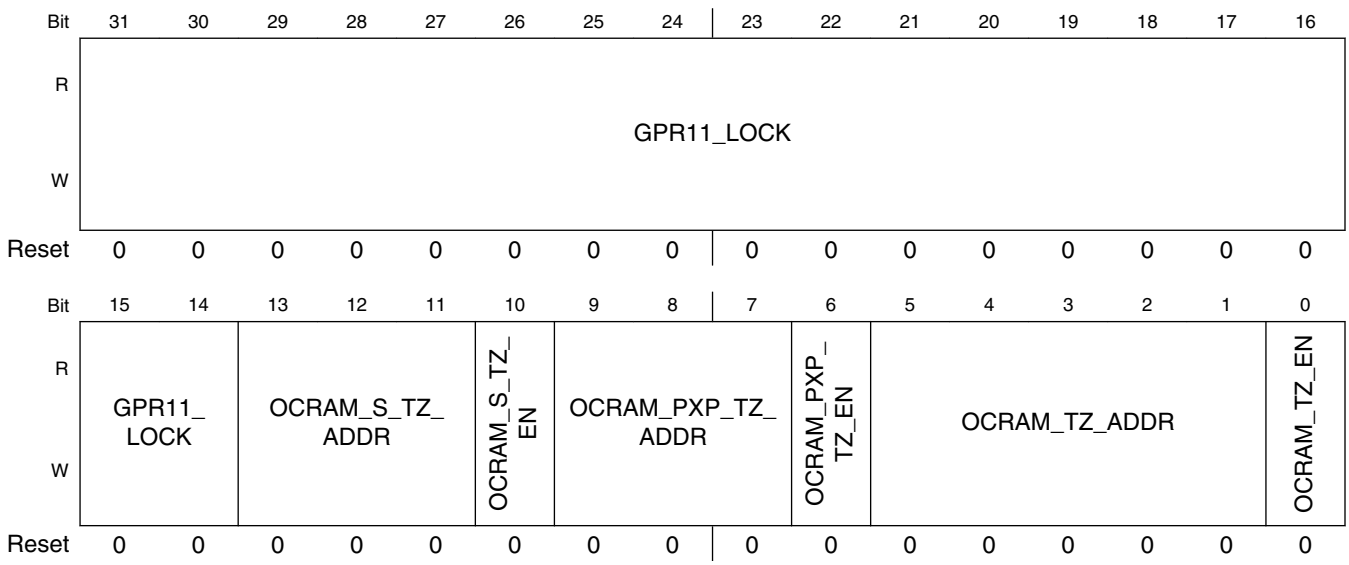
IOMUXC_GPR_GPR10 field descriptions

Field	Description
31–10 GPR10_LOCK	Bits [25:18] are Lock Bits for bits [9:2] Other bits are Reserved
9–4 OCRAM_GP_TZ_ADDR	GP OCRAM TrustZone (TZ) start address. This is the start address of the secure memory region within the GP OCRAM memory space is 4KB granularity. The start address affects the GP OCRAM transactions only if OCRAM_GP_TZ_EN bit is set. The GP OCRAM TZ ENDADDR is not configurable and is set to the end of GP OCRAM memory space. These are "lock" type bits
3 OCRAM_GP_TZ_EN	GP OCRAM TrustZone (TZ) enable This is a "lock" type bit 0 The TrustZone feature is disabled. Entire GP OCRAM space is available for all access types (secure/non-secure/user/supervisor). 1 The TrustZone feature is enabled. Access to address in the range specified by [ENDADDR:STARTADDR] follows the execution mode access policy described in CSU chapter.
2 SEC_ERR_RESP_EN	Security error response enable for all security gaskets (on both AHB and AXI busses) This is a "lock" type bit 0 OKAY response 1 SLVERR response
-	This field is reserved. Reserved

8.2.4.12 GPR11 General Purpose Register (IOMUXC_GPR_GPR11)

GPR Register

Address: 3034_0000h base + 2Ch offset = 3034_002Ch



IOMUXC_GPR_GPR11 field descriptions

Field	Description
31–14 GPR11_LOCK	Bits [29:16] are Lock Bits for bits [13:0] Other bits are Reserved
13–11 OCRAM_S_TZ_ ADDR	State Retention OCRAM TrustZone (TZ) start address. This is the start address of the secure memory region within the State Retention OCRAM memory space is 4KB granularity. The start address affects the State Retention OCRAM transactions only if OCRAM_S_TZ_EN bit is set. The State Retention OCRAM TZ ENDADDR is not configurable and is set to the end of State Retention OCRAM memory space. These are "lock" type bits
10 OCRAM_S_TZ_ EN	State Retention OCRAM TrustZone (TZ) enable This is a "lock" type bit 0 The TrustZone feature is disabled. Entire State Retention OCRAM space is available for all access types (secure/non-secure/user/supervisor). 1 The TrustZone feature is enabled. Access to address in the range specified by [ENDADDR:STARTADDR] follows the execution mode access policy described in CSU chapter.
9–7 OCRAM_PXP_ TZ_ADDR	PXP OCRAM TrustZone (TZ) start address. This is the start address of the secure memory region within the PXP OCRAM memory space is 4KB granularity. The start address affects the PXP OCRAM transactions only if OCRAM_PXP_TZ_EN bit is set. The PXP OCRAM TZ ENDADDR is not configurable and is set to the end of PXP OCRAM memory space. These are "lock" type bits

Table continues on the next page...

IOMUXC_GPR_GPR11 field descriptions (continued)

Field	Description
6 OCRAM_PXP_TZ_EN	PXP OCRAM TrustZone (TZ) enable This is a "lock" type bit 0 The TrustZone feature is disabled. Entire PXP OCRAM space is available for all access types (secure/non-secure/user/supervisor). 1 The TrustZone feature is enabled. Access to address in the range specified by [ENDADDR:STARTADDR] follows the execution mode access policy described in CSU chapter.
5–1 OCRAM_TZ_ADDR	OCRAM TrustZone (TZ) start address. This is the start address of the secure memory region within the OCRAM memory space is 4KB granularity. The start address affects the OCRAM transactions only if OCRAM_TZ_EN bit is set. The OCRAM TZ ENDADDR is not configurable and is set to the end of OCRAM memory space. These are "lock" type bits
0 OCRAM_TZ_EN	OCRAM TrustZone (TZ) enable This is a "lock" type bit 0 The TrustZone feature is disabled. Entire OCRAM space is available for all access types (secure/non-secure/user/supervisor). 1 The TrustZone feature is enabled. Access to address in the range specified by [ENDADDR:STARTADDR] follows the execution mode access policy described in CSU chapter.

8.2.4.13 GPR12 General Purpose Register (IOMUXC_GPR_GPR12)

GPR Register

Address: 3034_0000h base + 30h offset = 3034_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																															
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

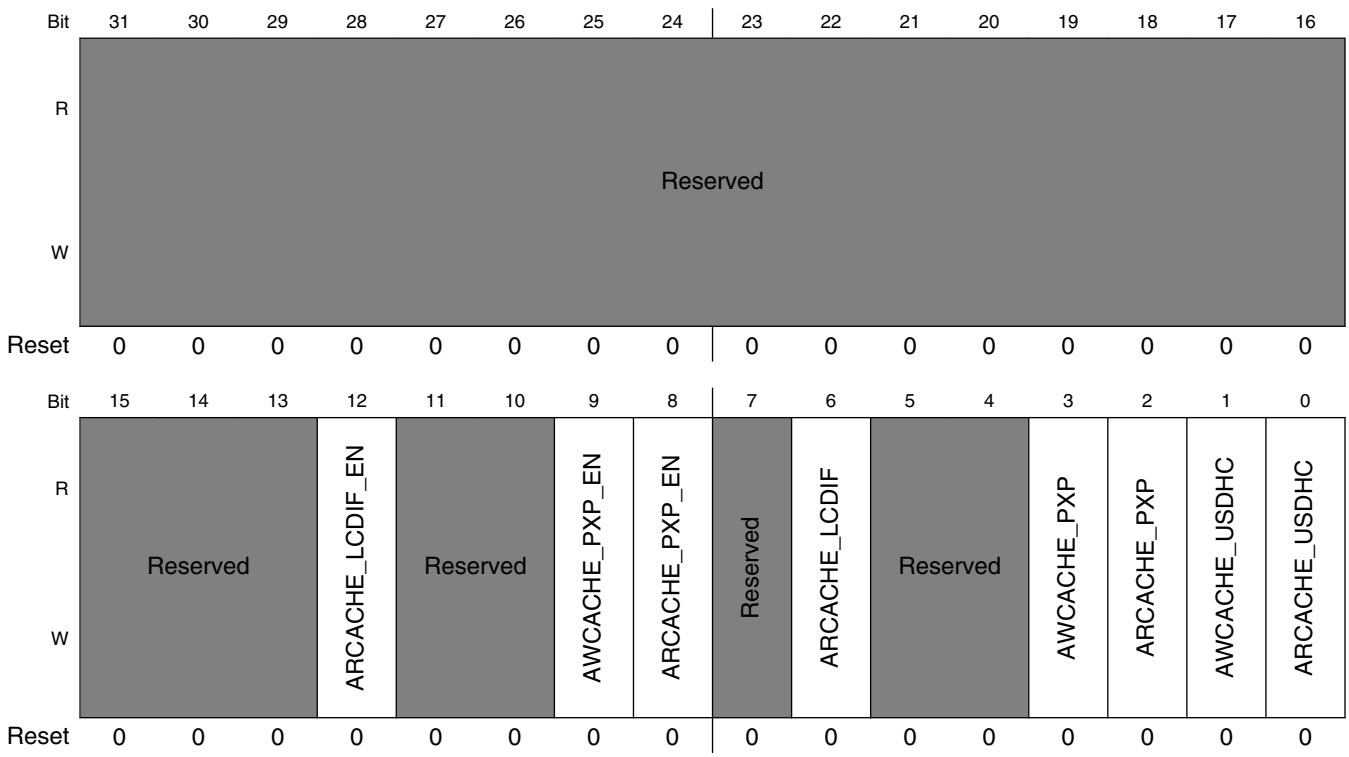
IOMUXC_GPR_GPR12 field descriptions

Field	Description
-	This field is reserved. Reserved

8.2.4.14 GPR13 General Purpose Register (IOMUXC_GPR_GPR13)

GPR Register

Address: 3034_0000h base + 34h offset = 3034_0034h



IOMUXC_GPR_GPR13 field descriptions

Field	Description
31–13 -	This field is reserved. Reserved
12 ARCACHE_ LCDIF_EN	LCDIF AXI Master Port ARCACHE Override Enable 0 LCDIF AXI Master Port ARCACHE[1] driven by LCDIF 1 LCDIF AXI Master Port ARCACHE[1] driven to constant value specified by the ARCACHE_LCDIF bit
11–10 -	This field is reserved. Reserved
9 AWCACHE_ PXP_EN	PXP Primary AXI Master Port AWCACHE Override Enable 0 PXP Primary AXI Master Port AWCACHE[1] driven by PXP 1 PXP Primary AXI Master Port AWCACHE[1] driven to constant value specified by the AWCACHE_PXP bit
8 ARCACHE_ PXP_EN	0 PXP Primary AXI Master Port ARCACHE Override Enable

Table continues on the next page...

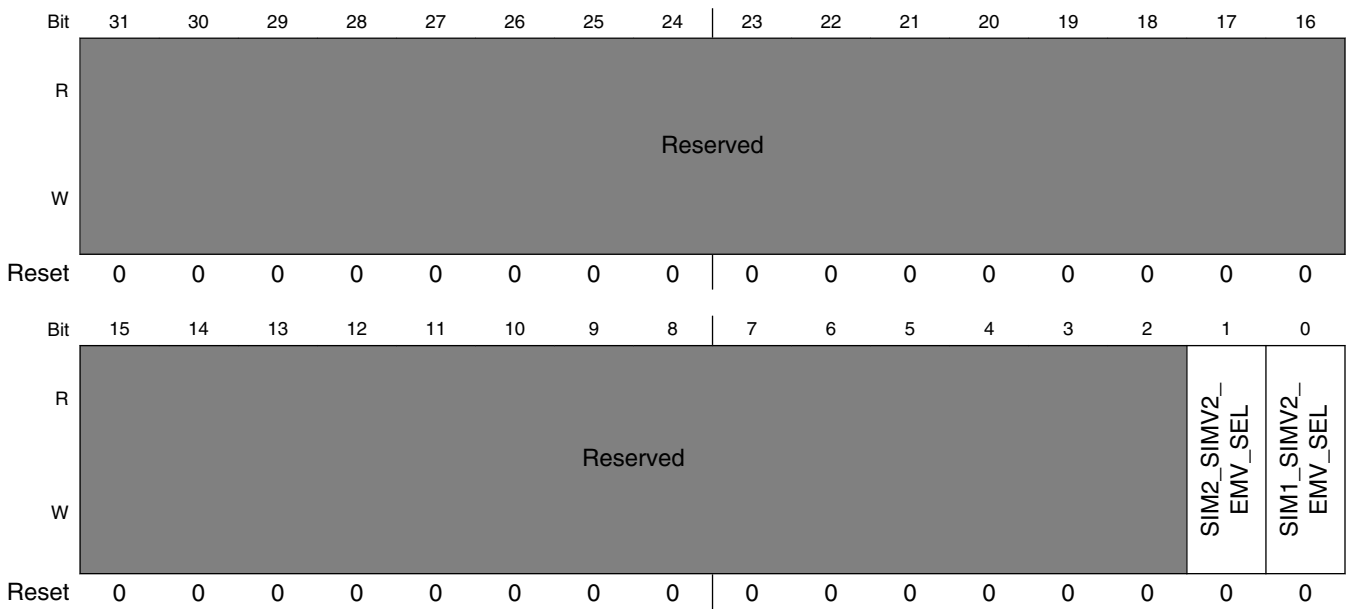
IOMUXC_GPR_GPR13 field descriptions (continued)

Field	Description
	0 PXP Primary AXI Master Port ARCACHE[1] driven by PXP 1 PXP Primary AXI Master Port ARCACHE[1] driven to constant value specified by the ARCACHE_PXP bit
7 -	This field is reserved. Reserved
6 ARCACHE_LCDIF	LCDIF AXI Master Port ARCACHE Override Value Note: this bit only takes effect when ARCACHE_LCDIF_EN = 1 0 Drive LCDIF AXI Master Port ARCACHE[1] to 0 1 Drive LCDIF AXI Master Port ARCACHE[1] to 1
5–4 -	This field is reserved. Reserved
3 AWCACHE_PXP	PXP Primary AXI Master Port AWCACHE Override Value Note: this bit only takes effect when AWCACHE_PXP_EN = 1 0 Drive PXP Primary AXI Master Port AWCACHE[1] to 0 1 Drive PXP Primary AXI Master Port AWCACHE[1] to 1
2 ARCACHE_PXP	PXP Primary AXI Master Port ARCACHE Override Value Note: this bit only takes effect when ARCACHE_PXP_EN = 1 0 Drive PXP Primary AXI Master Port ARCACHE[1] to 0 1 Drive PXP Primary AXI Master Port ARCACHE[1] to 1
1 AWCACHE_USDHC	USDHC 1-3 AXI Master AWCACHE Override Value Note: this bit always overrides 0 Drive USDHC AXI Master AWCACHE[1] to 0 1 Drive USDHC AXI Master AWCACHE[1] to 1
0 ARCACHE_USDHC	USDHC 1-3 AXI Master ARCACHE Override Value Note: this bit always overrides 0 Drive USDHC AXI Master ARCACHE[1] to 0 1 Drive USDHC AXI Master ARCACHE[1] to 1

8.2.4.15 GPR14 General Purpose Register (IOMUXC_GPR_GPR14)

GPR Register

Address: 3034_0000h base + 38h offset = 3034_0038h



IOMUXC_GPR_GPR14 field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
1 SIM2_SIMV2_EMV_SEL	SIM2 EMV Select This bit should be programmed to 0 since EMV not supported 0 Select SIMv2 1 Select EMV
0 SIM1_SIMV2_EMV_SEL	SIM1 EMV Select This bit should be programmed to 0 since EMV not supported 0 Select SIMv2 1 Select EMV

8.2.4.16 GPR15 General Purpose Register (IOMUXC_GPR_GPR15)

GPR Register

Address: 3034_0000h base + 3Ch offset = 3034_003Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved										LVDS_I_LOCK_PPM_SET					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		LVDS_I_DESKEW_CNT_SET												LVDS_I_AUTO_SEL	LVDS_I_VBLK_FLAG
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

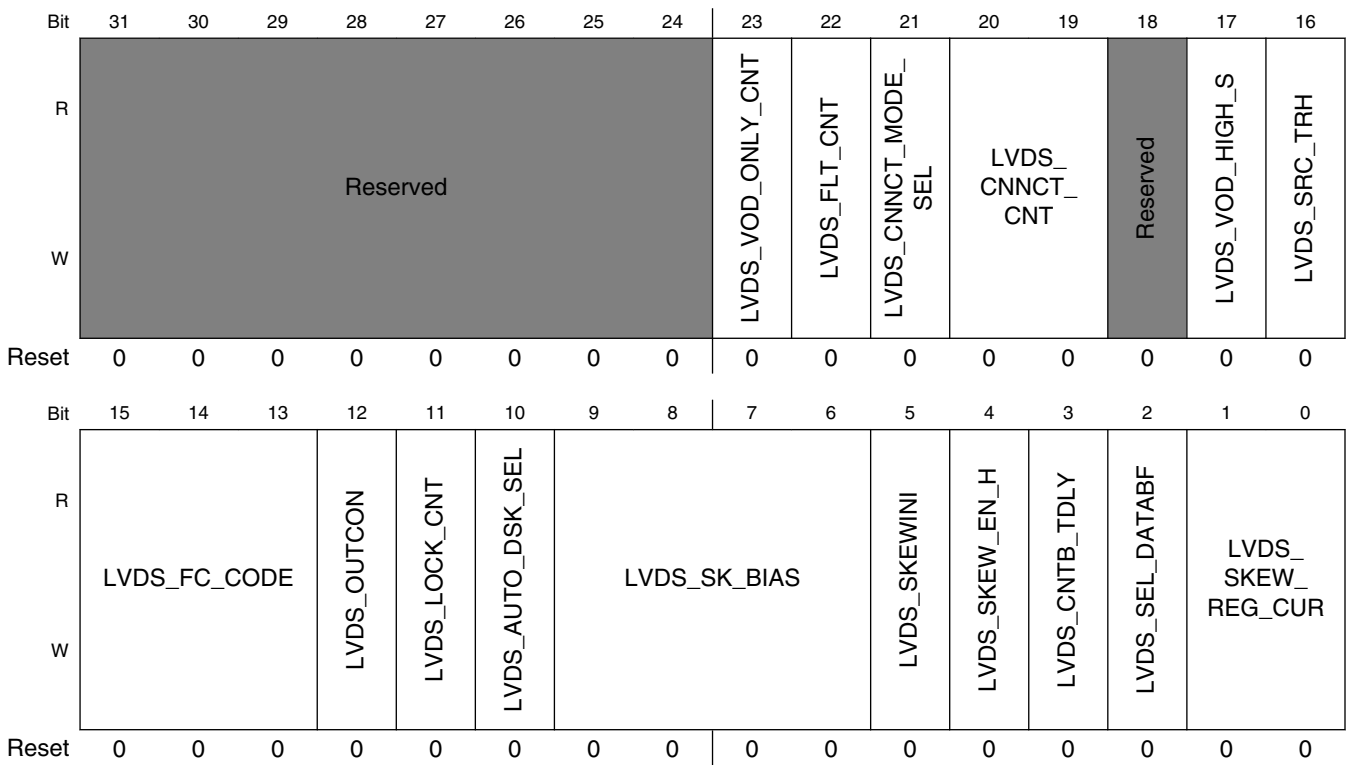
IOMUXC_GPR_GPR15 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–16 LVDS_I_LOCK_PPM_SET	Not Used
15–14 -	This field is reserved. Reserved
13–2 LVDS_I_DESKEW_CNT_SET	Not Used
1 LVDS_I_AUTO_SEL	Not Used
0 LVDS_I_VBLK_FLAG	Not Used

8.2.4.17 GPR16 General Purpose Register (IOMUXC_GPR_GPR16)

GPR Register

Address: 3034_0000h base + 40h offset = 3034_0040h



IOMUXC_GPR_GPR16 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 LVDS_VOD_ONLY_CNT	Not Used
22 LVDS_FLT_CNT	Not Used
21 LVDS_CNNCT_MODE_SEL	Not Used
20–19 LVDS_CNNCT_CNT	Not Used
18 -	This field is reserved. Reserved

Table continues on the next page...

IOMUXC_GPR_GPR16 field descriptions (continued)

Field	Description
17 LVDS_VOD_ HIGH_S	Not Used
16 LVDS_SRC_TRH	Not Used
15–13 LVDS_FC_CODE	Not Used
12 LVDS_OUTCON	Not Used
11 LVDS_LOCK_ CNT	Not Used
10 LVDS_AUTO_ DSK_SEL	Not Used
9–6 LVDS_SK_BIAS	Not Used
5 LVDS_SKEWINI	Not Used
4 LVDS_SKEW_ EN_H	Not Used
3 LVDS_CNTB_ TDLY	Not Used
2 LVDS_SEL_ DATABF	Not Used
LVDS_SKEW_ REG_CUR	Not Used

8.2.4.18 GPR17 General Purpose Register (IOMUXC_GPR_GPR17)

GPR Register

Address: 3034_0000h base + 44h offset = 3034_0044h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																LVDS_CNT_VOD_H								LVDS_CNT_PEN_H							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

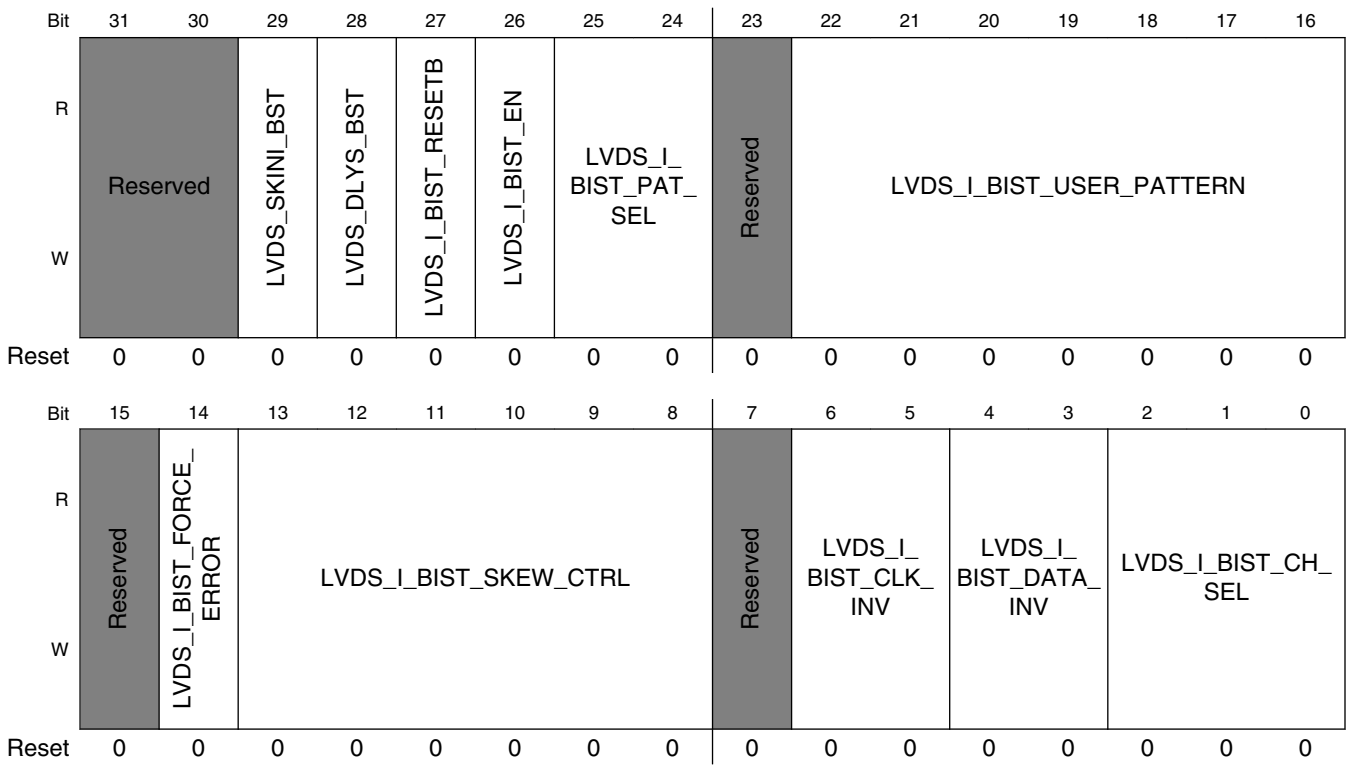
IOMUXC_GPR_GPR17 field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 LVDS_CNT_ VOD_H	Not Used
LVDS_CNT_ PEN_H	Not Used

8.2.4.19 GPR18 General Purpose Register (IOMUXC_GPR_GPR18)

GPR Register

Address: 3034_0000h base + 48h offset = 3034_0048h



IOMUXC_GPR_GPR18 field descriptions

Field	Description
31–30 -	This field is reserved. Reserved

Table continues on the next page...

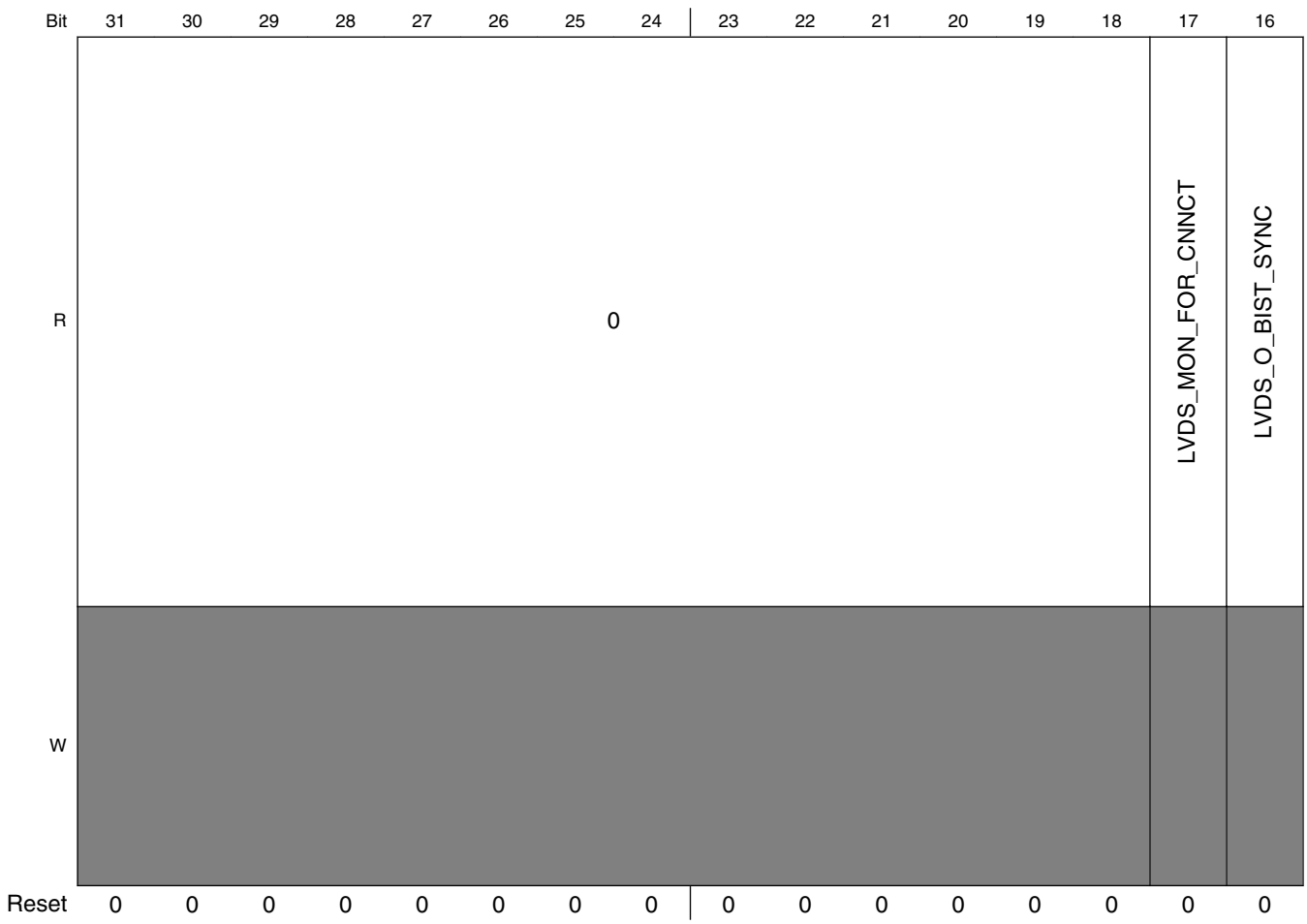
IOMUXC_GPR_GPR18 field descriptions (continued)

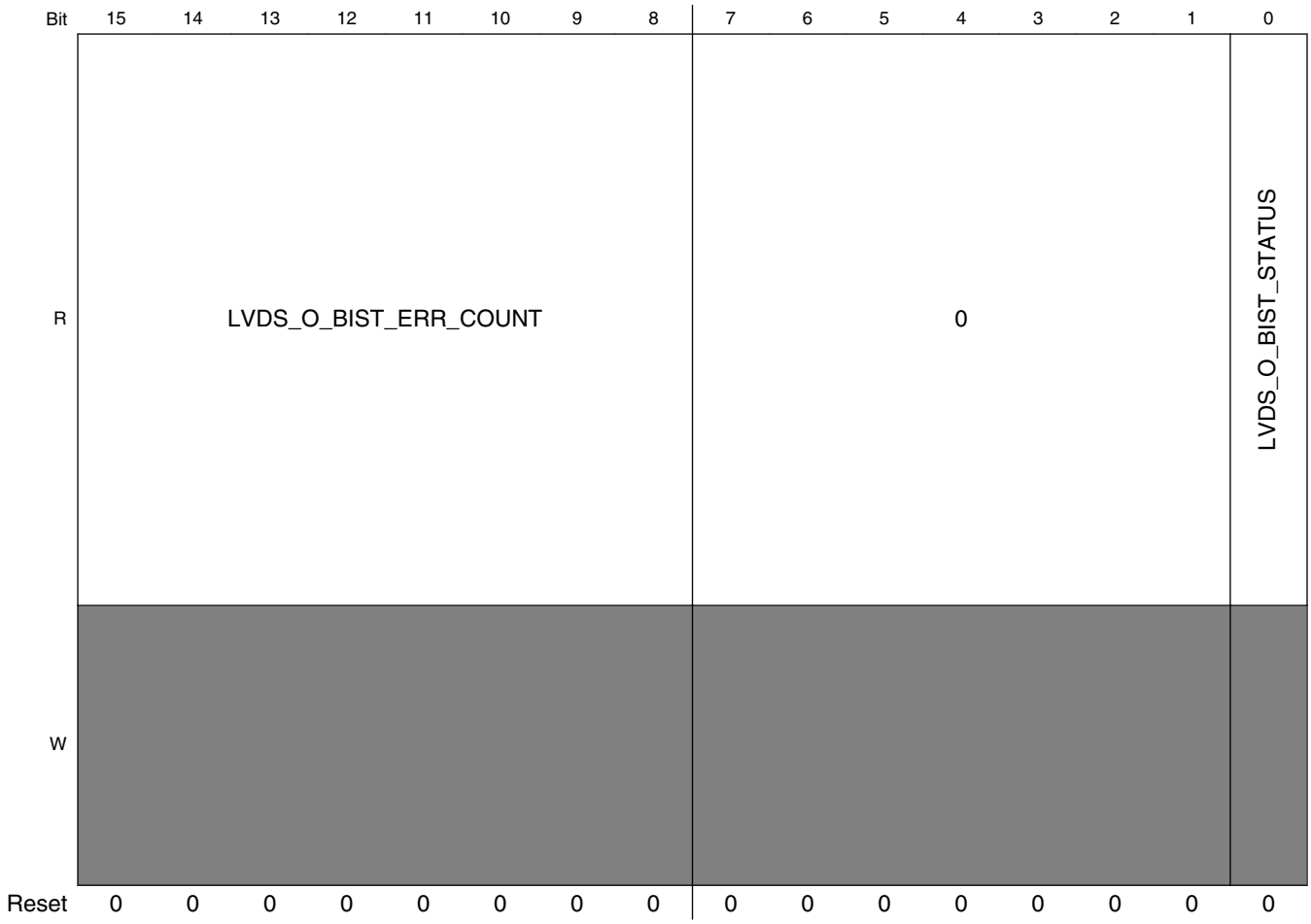
Field	Description
29 LVDS_SKINI_ BST	Not Used
28 LVDS_DLYS_ BST	Not Used
27 LVDS_I_BIST_ RESETB	Not Used
26 LVDS_I_BIST_ EN	Not Used
25–24 LVDS_I_BIST_ PAT_SEL	Not Used
23 -	This field is reserved. Reserved
22–16 LVDS_I_BIST_ USER_ PATTERN	Not Used
15 -	This field is reserved. Reserved
14 LVDS_I_BIST_ FORCE_ERROR	Not Used
13–8 LVDS_I_BIST_ SKEW_CTRL	Not Used
7 -	This field is reserved. Reserved
6–5 LVDS_I_BIST_ CLK_INV	Not Used
4–3 LVDS_I_BIST_ DATA_INV	Not Used
LVDS_I_BIST_ CH_SEL	Not Used

8.2.4.20 GPR19 General Purpose Register (IOMUXC_GPR_GPR19)

GPR Register

Address: 3034_0000h base + 4Ch offset = 3034_004Ch





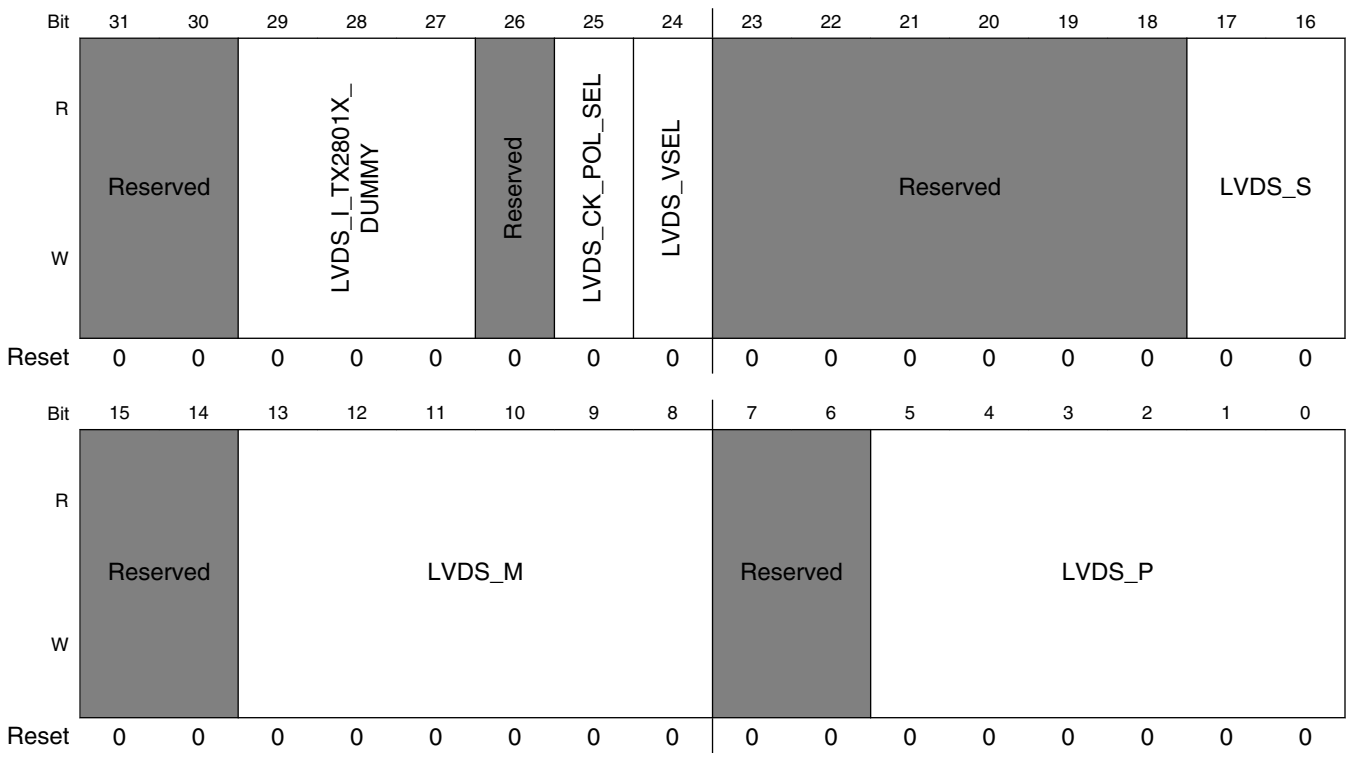
IOMUXC_GPR_GPR19 field descriptions

Field	Description
31–18 Reserved	This read-only field is reserved and always has the value 0.
17 LVDS_MON_ FOR_CNNCT	Not Used
16 LVDS_O_BIST_ SYNC	Not Used
15–8 LVDS_O_BIST_ ERR_COUNT	Not Used
7–1 Reserved	This read-only field is reserved and always has the value 0.
0 LVDS_O_BIST_ STATUS	Not Used

8.2.4.21 GPR20 General Purpose Register (IOMUXC_GPR_GPR20)

GPR Register

Address: 3034_0000h base + 50h offset = 3034_0050h



IOMUXC_GPR_GPR20 field descriptions

Field	Description
31–30 -	This field is reserved. Reserved
29–27 LVDS_I_ TX2801X_ DUMMY	Not Used
26 -	This field is reserved. Reserved
25 LVDS_CK_POL_ SEL	Not Used
24 LVDS_VSEL	Not Used
23–18 -	This field is reserved. Reserved

Table continues on the next page...

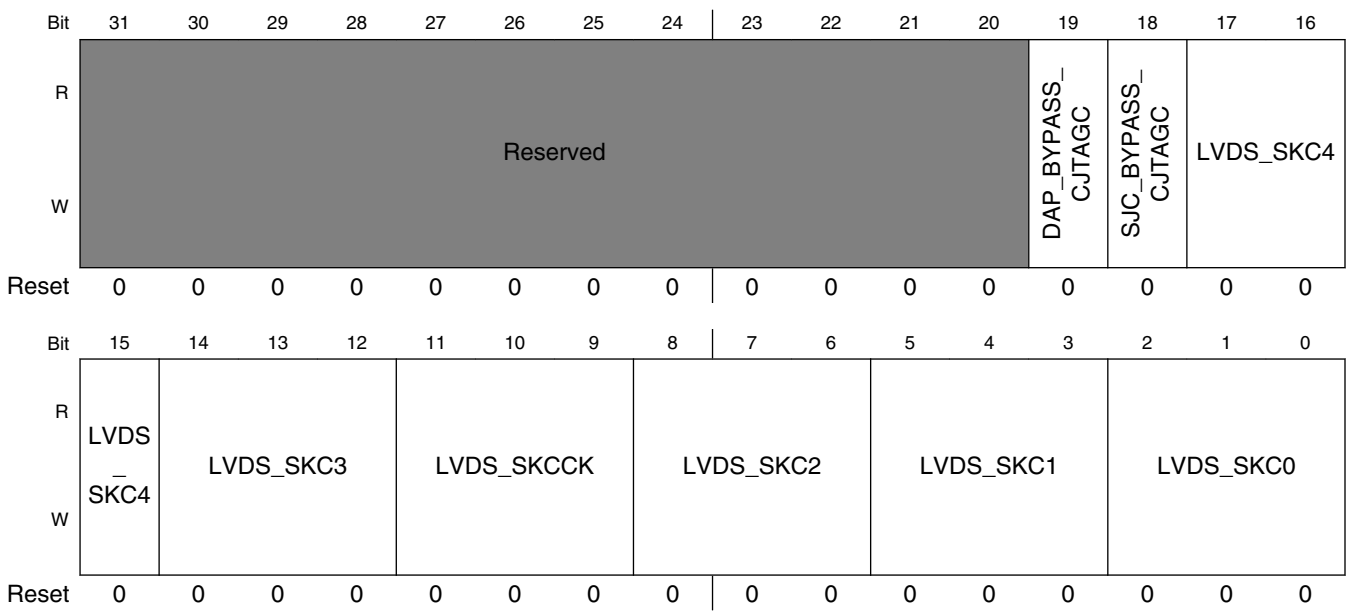
IOMUXC_GPR_GPR20 field descriptions (continued)

Field	Description
17–16 LVDS_S	Not Used
15–14 -	This field is reserved. Reserved
13–8 LVDS_M	Not Used
7–6 -	This field is reserved. Reserved
LVDS_P	Not Used

8.2.4.22 GPR21 General Purpose Register (IOMUXC_GPR_GPR21)

GPR Register

Address: 3034_0000h base + 54h offset = 3034_0054h



IOMUXC_GPR_GPR21 field descriptions

Field	Description
31–20 -	This field is reserved. Reserved
19 DAP_BYPASS_CJTAGC	Not Used

Table continues on the next page...

IOMUXC_GPR_GPR21 field descriptions (continued)

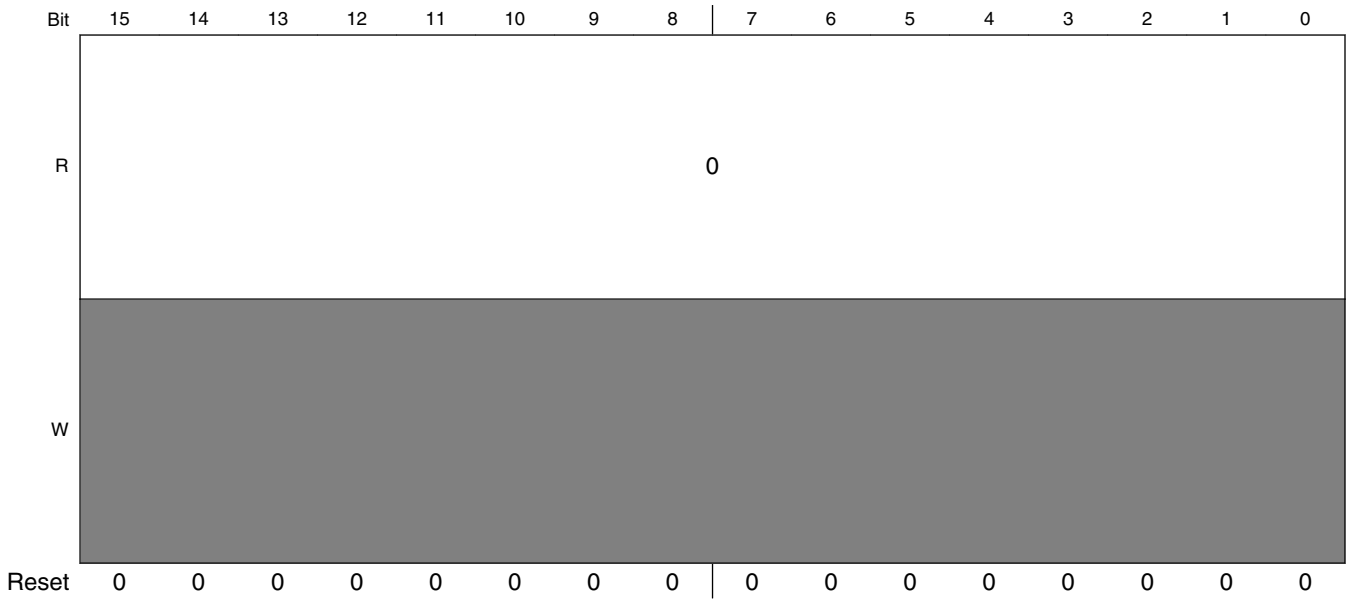
Field	Description
18 SJC_BYPASS_ CJTAGC	Not Used
17–15 LVDS_SKC4	Not Used
14–12 LVDS_SKC3	Not Used
11–9 LVDS_SKCCK	Not Used
8–6 LVDS_SKC2	Not Used
5–3 LVDS_SKC1	Not Used
LVDS_SKC0	Not Used

8.2.4.23 GPR22 General Purpose Register (IOMUXC_GPR_GPR22)

GPR Register

Address: 3034_0000h base + 58h offset = 3034_0058h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	1		CHD1_ISO_ENA_1	CHD1_DVDD_STABLE	Reserved		DFI_INIT_COMPLETE	DDRC_MRR_VALID	DDRC_MRR_DATA							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



IOMUXC_GPR_GPR22 field descriptions

Field	Description
31–30 Reserved	This read-only field is reserved and always has the value 1.
29 CHD1_ISO_ENA_1	USB OTG1 Auxiliary Charge Detector isolation control signal, high when dvdd_usb domain is unpowered and vdd_soc domain is powered. This signal is in the vdd_soc domain.
28 CHD1_DVDD_STABLE	USB OTG1 Auxiliary Charge Detector power stable signal for LDO_USB_1P0 to allow vdd_soc domain to monitor when dvdd_usb domain is valid. This signal is in the vdd_soc domain.
27–26 -	This field is reserved. Reserved
25 DFI_INIT_COMPLETE	DDR PHY initialization complete. This signal indicates that the PHY is able to respond to any proper stimulus on the DFI.
24 DDRC_MRR_VALID	DDR Controller Mode Register Read Data Valid
23–16 DDRC_MRR_DATA	DDR Controller Mode Register Read Data
Reserved	This read-only field is reserved and always has the value 0.

8.2.5 IOMUXC LPSR Memory Map/Register Definition

IOMUXC_LPSR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
302C_0000	SW_MUX_CTL_PAD_GPIO1_IO00 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO00)	32	R/W	0000_0000h	8.2.5.1/ 1524
302C_0004	SW_MUX_CTL_PAD_GPIO1_IO01 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO01)	32	R/W	0000_0000h	8.2.5.2/ 1526
302C_0008	SW_MUX_CTL_PAD_GPIO1_IO02 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO02)	32	R/W	0000_0000h	8.2.5.3/ 1527
302C_000C	SW_MUX_CTL_PAD_GPIO1_IO03 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO03)	32	R/W	0000_0000h	8.2.5.4/ 1528
302C_0010	SW_MUX_CTL_PAD_GPIO1_IO04 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO04)	32	R/W	0000_0000h	8.2.5.5/ 1530
302C_0014	SW_MUX_CTL_PAD_GPIO1_IO05 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO05)	32	R/W	0000_0000h	8.2.5.6/ 1531
302C_0018	SW_MUX_CTL_PAD_GPIO1_IO06 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO06)	32	R/W	0000_0000h	8.2.5.7/ 1532
302C_001C	SW_MUX_CTL_PAD_GPIO1_IO07 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO07)	32	R/W	0000_0000h	8.2.5.8/ 1533
302C_0020	SW_PAD_CTL_PAD_TEST_MODE SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_TEST_MODE)	32	R/W	0000_0014h	8.2.5.9/ 1534
302C_0024	SW_PAD_CTL_PAD_SRC_POR_B SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_SRC_POR_B)	32	R/W	0000_007Ch	8.2.5.10/ 1535
302C_0028	SW_PAD_CTL_PAD_BOOT_MODE0 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_BOOT_MODE0)	32	R/W	0000_001Ch	8.2.5.11/ 1537
302C_002C	SW_PAD_CTL_PAD_BOOT_MODE1 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_BOOT_MODE1)	32	R/W	0000_001Ch	8.2.5.12/ 1538
302C_0030	SW_PAD_CTL_PAD_GPIO1_IO00 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO00)	32	R/W	0000_0074h	8.2.5.13/ 1539
302C_0034	SW_PAD_CTL_PAD_GPIO1_IO01 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO01)	32	R/W	0000_0014h	8.2.5.14/ 1540
302C_0038	SW_PAD_CTL_PAD_GPIO1_IO02 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO02)	32	R/W	0000_0014h	8.2.5.15/ 1541

Table continues on the next page...

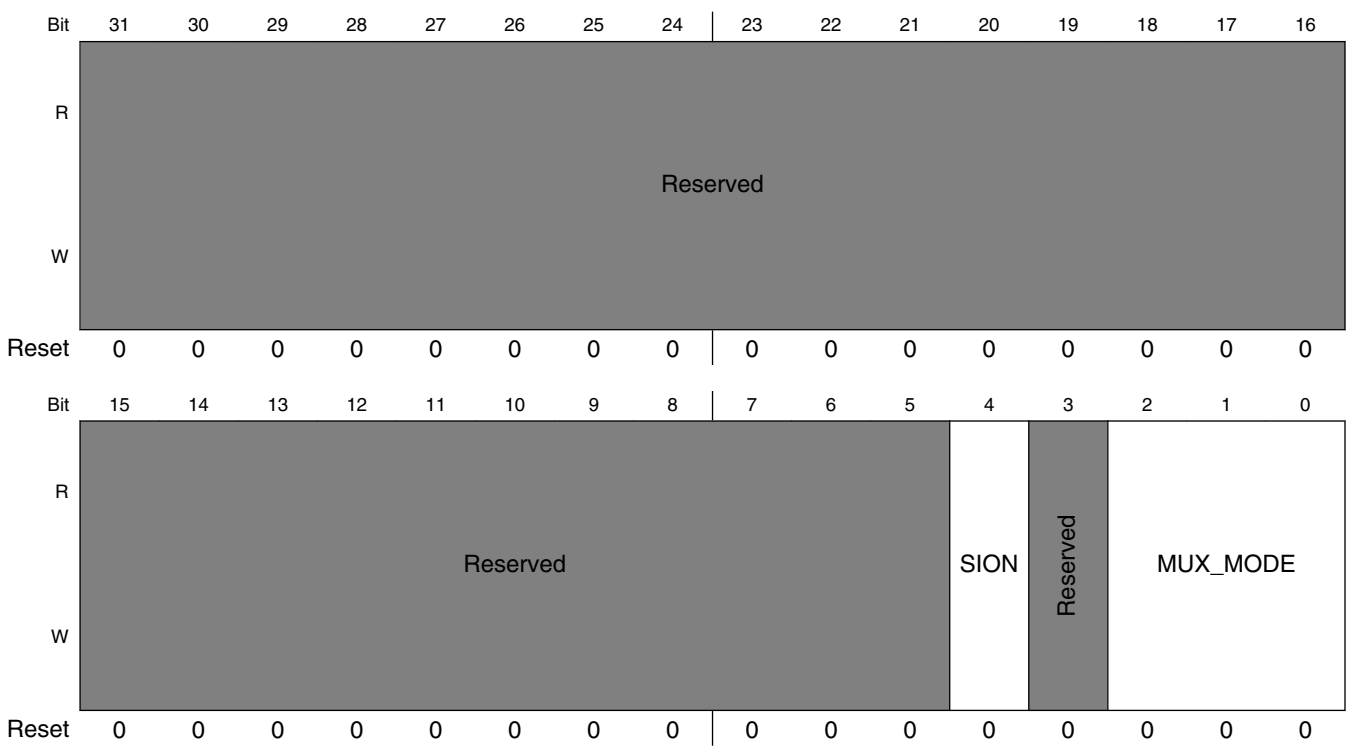
IOMUXC_LPSR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
302C_003C	SW_PAD_CTL_PAD_GPIO1_IO03 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO03)	32	R/W	0000_0014h	8.2.5.16/ 1542
302C_0040	SW_PAD_CTL_PAD_GPIO1_IO04 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO04)	32	R/W	0000_0014h	8.2.5.17/ 1544
302C_0044	SW_PAD_CTL_PAD_GPIO1_IO05 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO05)	32	R/W	0000_0014h	8.2.5.18/ 1545
302C_0048	SW_PAD_CTL_PAD_GPIO1_IO06 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO06)	32	R/W	0000_0014h	8.2.5.19/ 1546
302C_004C	SW_PAD_CTL_PAD_GPIO1_IO07 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO07)	32	R/W	0000_0014h	8.2.5.20/ 1547

8.2.5.1 SW_MUX_CTL_PAD_GPIO1_IO00 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO00)

SW_MUX_CTL Register

Address: 302C_0000h base + 0h offset = 302C_0000h



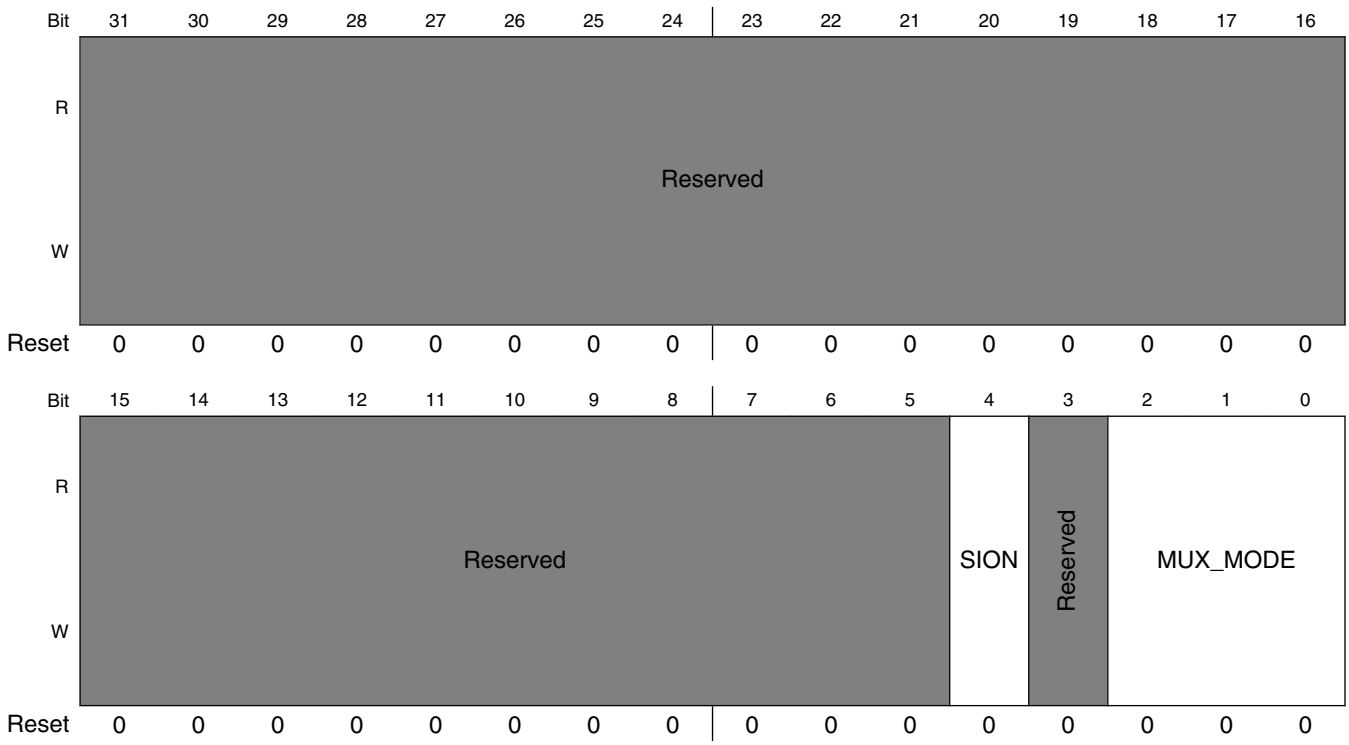
IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO00 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO00 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: GPIO1_IO00. 000 **ALT0_GPIO1_IO0** — Select mux mode: ALT0 mux port: IO0 of instance: GPIO1 001 **ALT1_PWM4_OUT** — Select mux mode: ALT1 mux port: OUT of instance: PWM4 010 **ALT2_WDOG1_WDOG_ANY** — Select mux mode: ALT2 mux port: WDOG_ANY of instance: WDOG1 011 **ALT3_WDOG1_WDOG_B** — Select mux mode: ALT3 mux port: WDOG_B of instance: WDOG1 100 **ALT4_WDOG1_WDOG_RST_B_DEB** — Select mux mode: ALT4 mux port: WDOG_RST_B_DEB of instance: WDOG1

8.2.5.2 SW_MUX_CTL_PAD_GPIO1_IO01 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO01)

SW_MUX_CTL Register

Address: 302C_0000h base + 4h offset = 302C_0004h



IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO01 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO01 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: GPIO1_IO01. 000 **ALT0_GPIO1_IO1** — Select mux mode: ALT0 mux port: IO1 of instance: GPIO1 001 **ALT1_PWM1_OUT** — Select mux mode: ALT1 mux port: OUT of instance: PWM1

Table continues on the next page...

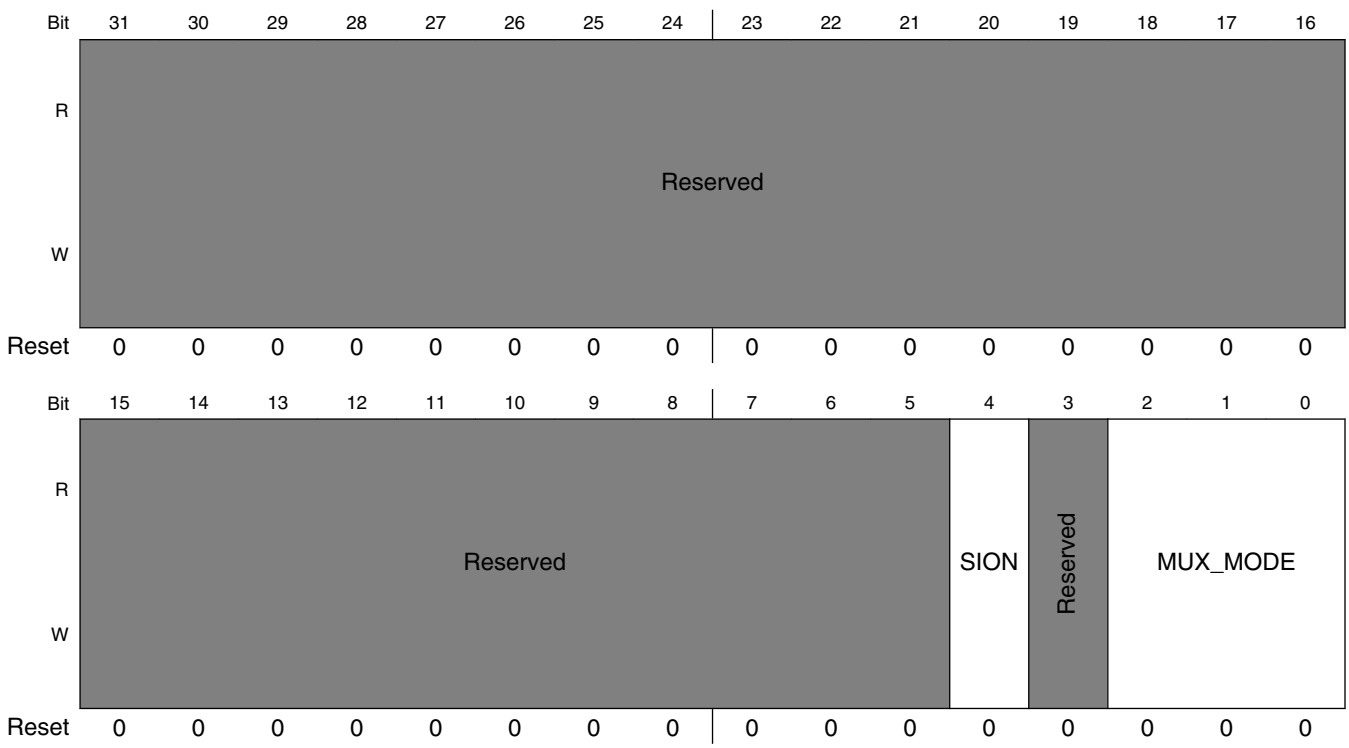
IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO01 field descriptions (continued)

Field	Description
	010 **ALT2_CCM_ENET_REF_CLK3** — Select mux mode: ALT2 mux port: ENET_REF_CLK3 of instance: CCM 011 **ALT3_SAI1_MCLK** — Select mux mode: ALT3 mux port: MCLK of instance: SAI1

8.2.5.3 SW_MUX_CTL_PAD_GPIO1_IO02 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO02)

SW_MUX_CTL Register

Address: 302C_0000h base + 8h offset = 302C_0008h



IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO02 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

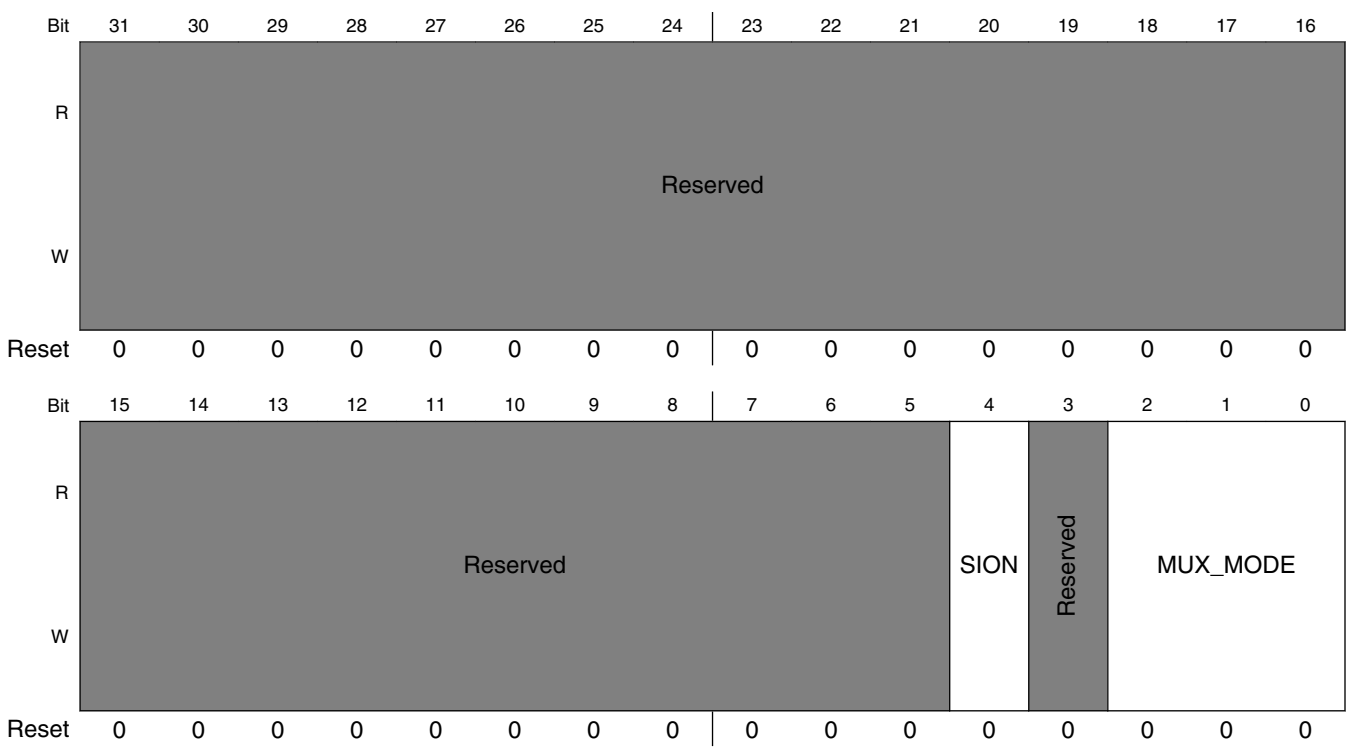
IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO02 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO1_IO02. 000 **ALT0_GPIO1_IO2** — Select mux mode: ALT0 mux port: IO2 of instance: GPIO1 001 **ALT1_PWM2_OUT** — Select mux mode: ALT1 mux port: OUT of instance: PWM2 010 **ALT2_CCM_ENET_REF_CLK1** — Select mux mode: ALT2 mux port: ENET_REF_CLK1 of instance: CCM 011 **ALT3_SAI2_MCLK** — Select mux mode: ALT3 mux port: MCLK of instance: SAI2 101 **ALT5_CCM_CLKO1** — Select mux mode: ALT5 mux port: CLKO1 of instance: CCM 111 **ALT7_USB_OTG1_ID** — Select mux mode: ALT7 mux port: OTG1_ID of instance: USB

8.2.5.4 SW_MUX_CTL_PAD_GPIO1_IO03 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO03)

SW_MUX_CTL Register

Address: 302C_0000h base + Ch offset = 302C_000Ch



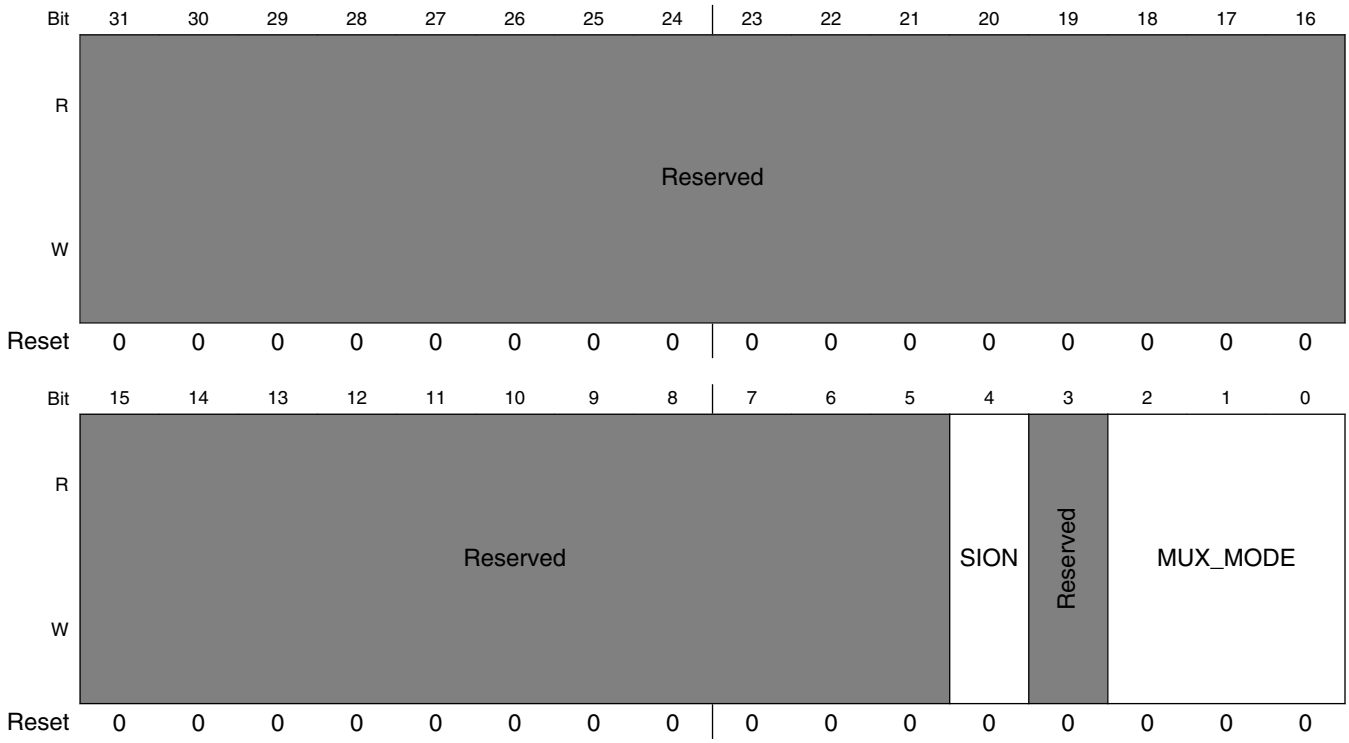
IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO03 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO03 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO1_IO03. 000 **ALT0_GPIO1_IO3** — Select mux mode: ALT0 mux port: IO3 of instance: GPIO1 001 **ALT1_PWM3_OUT** — Select mux mode: ALT1 mux port: OUT of instance: PWM3 010 **ALT2_CCM_ENET_REF_CLK2** — Select mux mode: ALT2 mux port: ENET_REF_CLK2 of instance: CCM 011 **ALT3_SAI3_MCLK** — Select mux mode: ALT3 mux port: MCLK of instance: SAI3 101 **ALT5_CCM_CLKO2** — Select mux mode: ALT5 mux port: CLKO2 of instance: CCM

8.2.5.5 SW_MUX_CTL_PAD_GPIO1_IO04 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO04)

SW_MUX_CTL Register

Address: 302C_0000h base + 10h offset = 302C_0010h



IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO04 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO04 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: GPIO1_IO04. 000 **ALT0_GPIO1_IO4** — Select mux mode: ALT0 mux port: IO4 of instance: GPIO1 001 **ALT1_USB_OTG1_OC** — Select mux mode: ALT1 mux port: OTG1_OC of instance: USB 010 **ALT2_FLEXTIMER1_CH4** — Select mux mode: ALT2 mux port: CH4 of instance: FLEXTIMER1

Table continues on the next page...

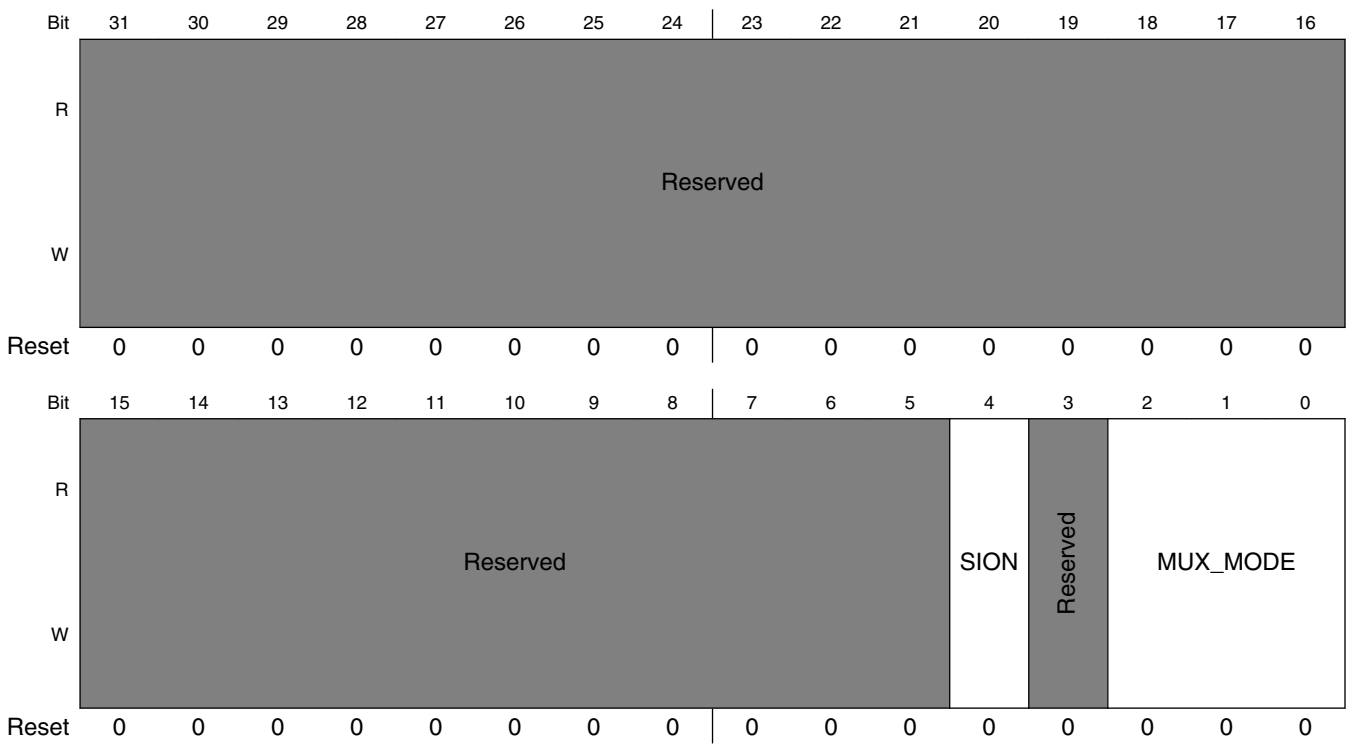
IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO04 field descriptions (continued)

Field	Description
	011 **ALT3_UART5_CTS_B** — Select mux mode: ALT3 mux port: CTS_B of instance: UART5 100 **ALT4_I2C1_SCL** — Select mux mode: ALT4 mux port: SCL of instance: I2C1

8.2.5.6 SW_MUX_CTL_PAD_GPIO1_IO05 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO05)

SW_MUX_CTL Register

Address: 302C_0000h base + 14h offset = 302C_0014h



IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO05 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO05 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved

Table continues on the next page...

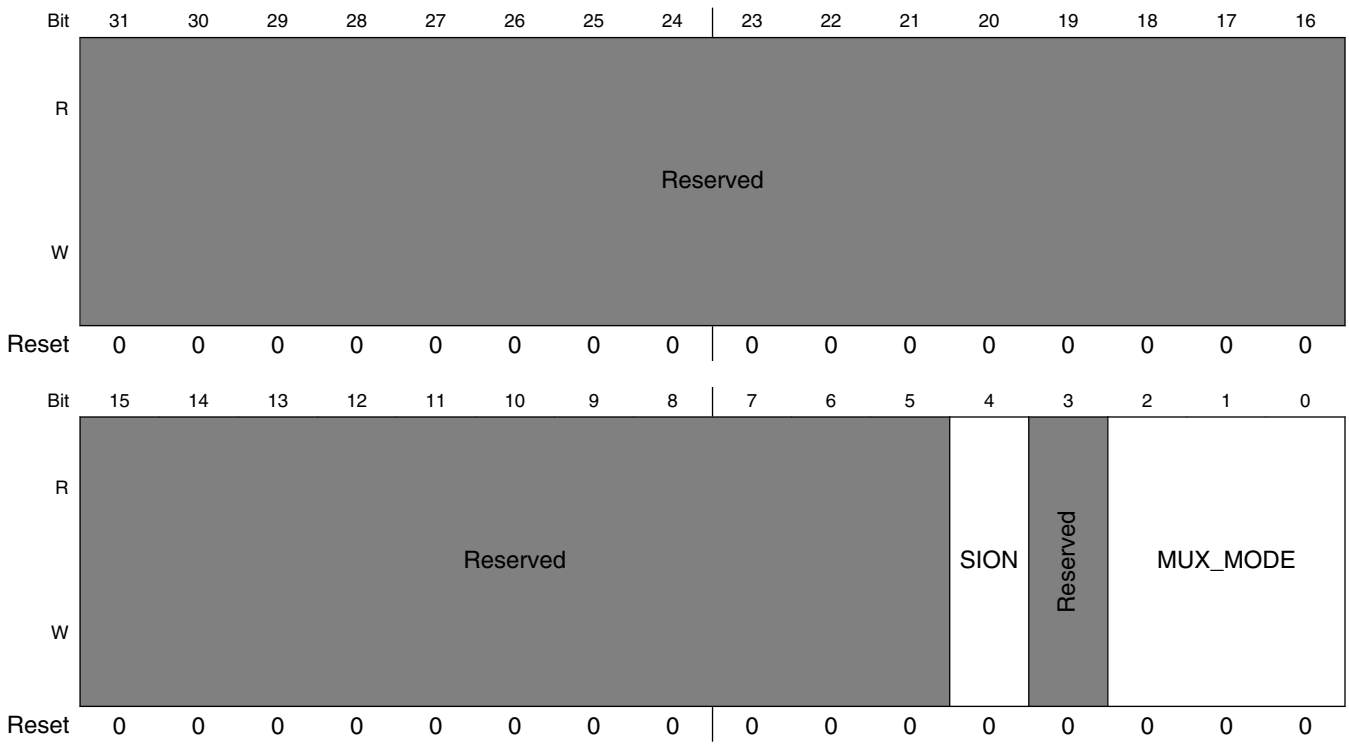
IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO05 field descriptions (continued)

Field	Description
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: GPIO1_IO05. 000 **ALT0_GPIO1_IO5** — Select mux mode: ALT0 mux port: IO5 of instance: GPIO1 001 **ALT1_USB_OTG1_PWR** — Select mux mode: ALT1 mux port: OTG1_PWR of instance: USB 010 **ALT2_FLEXTIMER1_CH5** — Select mux mode: ALT2 mux port: CH5 of instance: FLEXTIMER1 011 **ALT3_UART5_RTS_B** — Select mux mode: ALT3 mux port: RTS_B of instance: UART5 100 **ALT4_I2C1_SDA** — Select mux mode: ALT4 mux port: SDA of instance: I2C1

8.2.5.7 SW_MUX_CTL_PAD_GPIO1_IO06 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO06)

SW_MUX_CTL Register

Address: 302C_0000h base + 18h offset = 302C_0018h



IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO06 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved

Table continues on the next page...

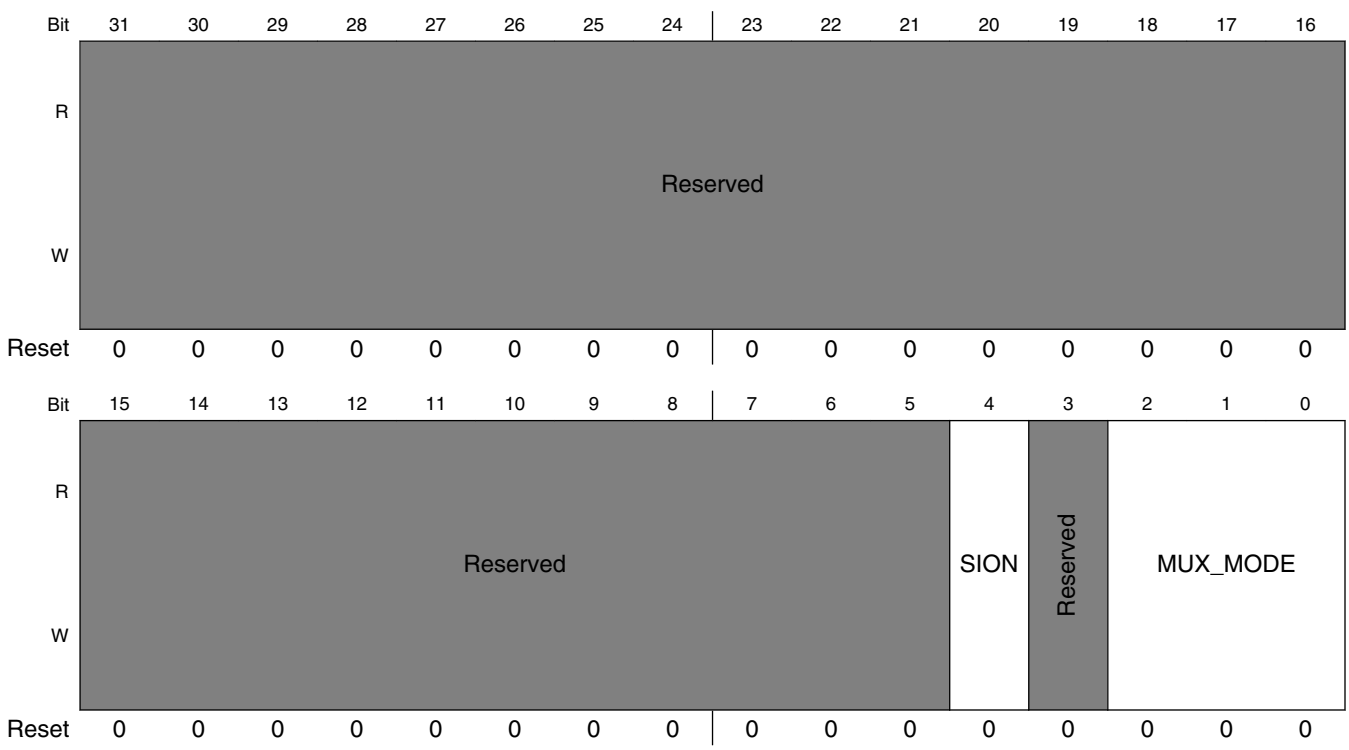
IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO06 field descriptions (continued)

Field	Description
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO06 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO1_IO06. 000 **ALT0_GPIO1_IO6** — Select mux mode: ALT0 mux port: IO6 of instance: GPIO1 010 **ALT2_FLEXTIMER1_CH6** — Select mux mode: ALT2 mux port: CH6 of instance: FLEXTIMER1 011 **ALT3_UART5_RX_DATA** — Select mux mode: ALT3 mux port: RX_DATA of instance: UART5 100 **ALT4_I2C2_SCL** — Select mux mode: ALT4 mux port: SCL of instance: I2C2 101 **ALT5_CCM_WAIT** — Select mux mode: ALT5 mux port: WAIT of instance: CCM 110 **ALT6_KPP_ROW4** — Select mux mode: ALT6 mux port: ROW4 of instance: KPP

8.2.5.8 SW_MUX_CTL_PAD_GPIO1_IO07 SW MUX Control Register (IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO07)

SW_MUX_CTL Register

Address: 302C_0000h base + 1Ch offset = 302C_001Ch



IOMUXC_LPSR_SW_MUX_CTL_PAD_GPIO1_IO07 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO07 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO1_IO07. 000 **ALT0_GPIO1_IO7** — Select mux mode: ALT0 mux port: IO7 of instance: GPIO1 010 **ALT2_FLEXTIMER1_CH7** — Select mux mode: ALT2 mux port: CH7 of instance: FLEXTIMER1 011 **ALT3_UART5_TX_DATA** — Select mux mode: ALT3 mux port: TX_DATA of instance: UART5 100 **ALT4_I2C2_SDA** — Select mux mode: ALT4 mux port: SDA of instance: I2C2 101 **ALT5_CCM_STOP** — Select mux mode: ALT5 mux port: STOP of instance: CCM 110 **ALT6_KPP_COL4** — Select mux mode: ALT6 mux port: COL4 of instance: KPP

8.2.5.9 SW_PAD_CTL_PAD_TEST_MODE SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_TEST_MODE)

SW_PAD_CTL Register

Address: 302C_0000h base + 20h offset = 302C_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_LPSR_SW_PAD_CTL_PAD_TEST_MODE field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field

Table continues on the next page...

IOMUXC_LPSR_SW_PAD_CTL_PAD_TEST_MODE field descriptions (continued)

Field	Description
	Select one out of next values for pad: TEST_MODE 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: TEST_MODE 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Read Only Field 0 **HYS** — Hysteresis Disabled
2 SRE	Slew Rate Field Read Only Field 1 **SRE** — Slow Slew Rate
DSE	Drive Strength Field Read Only Field 00 **DSE** — X1

8.2.5.10 SW_PAD_CTL_PAD_SRC_POR_B SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_SRC_POR_B)

SW_PAD_CTL Register

Address: 302C_0000h base + 24h offset = 302C_0024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0

IOMUXC_LPSR_SW_PAD_CTL_PAD_SRC_POR_B field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SRC_POR_B 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SRC_POR_B 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Read Only Field 1 **HYS** — Hysteresis Enabled
2 SRE	Slew Rate Field Read Only Field 1 **SRE** — Slow Slew Rate
DSE	Drive Strength Field Read Only Field 00 **DSE** — X1

8.2.5.11 SW_PAD_CTL_PAD_BOOT_MODE0 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_BOOT_MODE0)

SW_PAD_CTL Register

Address: 302C_0000h base + 28h offset = 302C_0028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

IOMUXC_LPSR_SW_PAD_CTL_PAD_BOOT_MODE0 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: BOOT_MODE0 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: BOOT_MODE0 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Read Only Field 1 **HYS** — Hysteresis Enabled
2 SRE	Slew Rate Field Read Only Field 1 **SRE** — Slow Slew Rate
DSE	Drive Strength Field

Table continues on the next page...

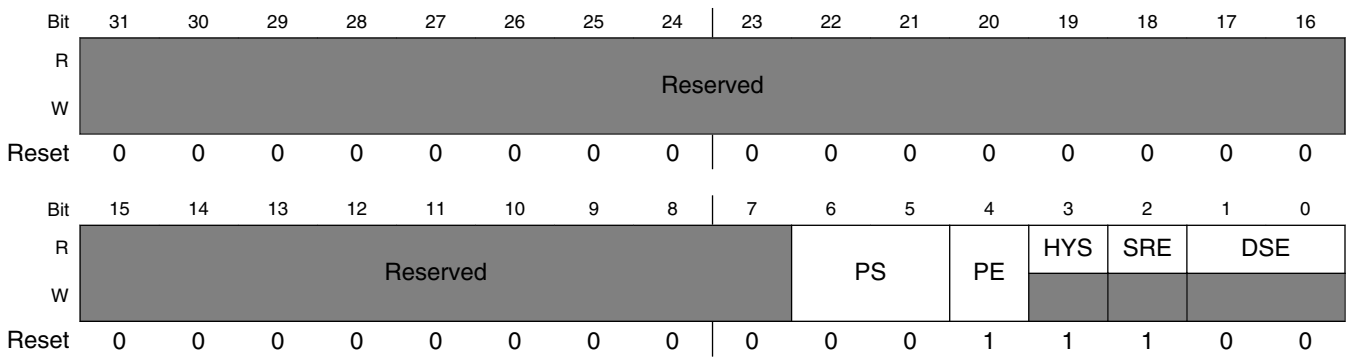
IOMUXC_LPSR_SW_PAD_CTL_PAD_BOOT_MODE0 field descriptions (continued)

Field	Description
	Read Only Field 00 **DSE** — X1

8.2.5.12 SW_PAD_CTL_PAD_BOOT_MODE1 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_BOOT_MODE1)

SW_PAD_CTL Register

Address: 302C_0000h base + 2Ch offset = 302C_002Ch



IOMUXC_LPSR_SW_PAD_CTL_PAD_BOOT_MODE1 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: BOOT_MODE1 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: BOOT_MODE1 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Read Only Field 1 **HYS** — Hysteresis Enabled

Table continues on the next page...

IOMUXC_LPSR_SW_PAD_CTL_PAD_BOOT_MODE1 field descriptions (continued)

Field	Description
2 SRE	Slew Rate Field Read Only Field 1 **SRE** — Slow Slew Rate
DSE	Drive Strength Field Read Only Field 00 **DSE** — X1

8.2.5.13 SW_PAD_CTL_PAD_GPIO1_IO00 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO00)

SW_PAD_CTL Register

Address: 302C_0000h base + 30h offset = 302C_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO00 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO00 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO00 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field

Table continues on the next page...

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO00 field descriptions (continued)

Field	Description
	Select one out of next values for pad: GPIO1_IO00 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO00 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO00 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.5.14 SW_PAD_CTL_PAD_GPIO1_IO01 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO01)

SW_PAD_CTL Register

Address: 302C_0000h base + 34h offset = 302C_0034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO01 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO01 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU

Table continues on the next page...

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO01 field descriptions (continued)

Field	Description
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO01 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO01 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO01 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.5.15 SW_PAD_CTL_PAD_GPIO1_IO02 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO02)

SW_PAD_CTL Register

Address: 302C_0000h base + 38h offset = 302C_0038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO02 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved

Table continues on the next page...

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO02 field descriptions (continued)

Field	Description
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO02 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO02 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO02 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO02 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.5.16 SW_PAD_CTL_PAD_GPIO1_IO03 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO03)

SW_PAD_CTL Register

Address: 302C_0000h base + 3Ch offset = 302C_003Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO03 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO03 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO03 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO03 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO03 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.5.17 SW_PAD_CTL_PAD_GPIO1_IO04 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO04)

SW_PAD_CTL Register

Address: 302C_0000h base + 40h offset = 302C_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO04 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO04 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO04 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO04 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO04 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO04 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.5.18 SW_PAD_CTL_PAD_GPIO1_IO05 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO05)

SW_PAD_CTL Register

Address: 302C_0000h base + 44h offset = 302C_0044h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO05 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO05 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO05 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO05

Table continues on the next page...

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO05 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO05 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.5.19 SW_PAD_CTL_PAD_GPIO1_IO06 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO06)

SW_PAD_CTL Register

Address: 302C_0000h base + 48h offset = 302C_0048h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO06 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO06 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO06 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO06 field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO06 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO06 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO06 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.5.20 SW_PAD_CTL_PAD_GPIO1_IO07 SW PAD Control Register (IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO07)

SW_PAD_CTL Register

Address: 302C_0000h base + 4Ch offset = 302C_004Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO07 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO07 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_LPSR_SW_PAD_CTL_PAD_GPIO1_IO07 field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO07 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO07 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO07 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO07 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.6 IOMUXC LPSR GPR Memory Map/Register Definition

IOMUXC_LPSR_GPR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3027_0000	IOMUXC_LPSR General Purpose Register 0 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR0)	32	R/W	0000_0000h	8.2.6.1/ 1550
3027_0004	IOMUXC_LPSR General Purpose Register 1 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR1)	32	R/W	0F40_0005h	8.2.6.2/ 1550
3027_0008	IOMUXC_LPSR General Purpose Register 2 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR2)	32	R/W	0000_0000h	8.2.6.3/ 1550
3027_000C	IOMUXC_LPSR General Purpose Register 3 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR3)	32	R/W	0000_0FFFh	8.2.6.4/ 1551
3027_0010	IOMUXC_LPSR General Purpose Register 4 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR4)	32	R/W	0000_0000h	8.2.6.5/ 1551

Table continues on the next page...

IOMUXC_LPSR_GPR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3027_0014	IOMUXC_LPSR General Purpose Register 5 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR5)	32	R/W	0000_0000h	8.2.6.6/ 1552
3027_0018	IOMUXC_LPSR General Purpose Register 6 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR6)	32	R/W	0000_0000h	8.2.6.7/ 1552
3027_001C	IOMUXC_LPSR General Purpose Register 7 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR7)	32	R/W	0000_0000h	8.2.6.8/ 1552
3027_0020	IOMUXC_LPSR General Purpose Register 8 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR8)	32	R/W	0000_0000h	8.2.6.9/ 1553
3027_0024	IOMUXC_LPSR General Purpose Register 9 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR9)	32	R/W	0000_0000h	8.2.6.10/ 1553
3027_0028	IOMUXC_LPSR General Purpose Register 10 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR10)	32	R/W	0000_0007h	8.2.6.11/ 1554
3027_002C	IOMUXC_LPSR General Purpose Register 11 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR11)	32	R/W	0000_0000h	8.2.6.12/ 1554
3027_0030	IOMUXC_LPSR General Purpose Register 12 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR12)	32	R/W	3000_0080h	8.2.6.13/ 1555
3027_0034	IOMUXC_LPSR General Purpose Register 13 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR13)	32	R/W	0000_0000h	8.2.6.14/ 1555
3027_0038	IOMUXC_LPSR General Purpose Register 14 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR14)	32	R/W	0000_0000h	8.2.6.15/ 1555
3027_003C	IOMUXC_LPSR General Purpose Register 15 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR15)	32	R/W	0000_0000h	8.2.6.16/ 1556
3027_0040	IOMUXC_LPSR General Purpose Register 16 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR16)	32	R/W	0000_0000h	8.2.6.17/ 1556
3027_0044	IOMUXC_LPSR General Purpose Register 17 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR17)	32	R/W	0000_0000h	8.2.6.18/ 1557
3027_0048	IOMUXC_LPSR General Purpose Register 18 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR18)	32	R/W	0000_0000h	8.2.6.19/ 1557
3027_004C	IOMUXC_LPSR General Purpose Register 19 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR19)	32	R/W	0000_0000h	8.2.6.20/ 1557
3027_0050	IOMUXC_LPSR General Purpose Register 20 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR20)	32	R/W	0000_0000h	8.2.6.21/ 1558
3027_0054	IOMUXC_LPSR General Purpose Register 21 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR21)	32	R/W	0000_0000h	8.2.6.22/ 1561
3027_0058	IOMUXC_LPSR General Purpose Register 22 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR22)	32	R/W	0000_0000h	8.2.6.23/ 1564

8.2.6.1 IOMUXC_LPSR General Purpose Register 0 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR0)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 0h offset = 3027_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR0 field descriptions

Field	Description
GP	Common bits can be R/W freely

8.2.6.2 IOMUXC_LPSR General Purpose Register 1 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR1)

IOMUXC LPSR General Purpose Register 1

Address: 3027_0000h base + 4h offset = 3027_0004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP																															
Reset	0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR1 field descriptions

Field	Description
GP	Common bits can be R/W freely

8.2.6.3 IOMUXC_LPSR General Purpose Register 2 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR2)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 8h offset = 3027_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR2 field descriptions

Field	Description
GP	Common bits can be R/W freely

8.2.6.4 IOMUXC_LPSR General Purpose Register 3 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR3)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + Ch offset = 3027_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RO																GP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR3 field descriptions

Field	Description
31–16 RO	Read only bits, always 16'h0
GP	Common bits can be R/W freely

8.2.6.5 IOMUXC_LPSR General Purpose Register 4 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR4)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 10h offset = 3027_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RO																GP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR4 field descriptions

Field	Description
31–16 RO	Read only bits, always 16'h0
GP	Common bits can be R/W freely

8.2.6.6 IOMUXC_LPSR General Purpose Register 5 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR5)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 14h offset = 3027_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR5 field descriptions

Field	Description
GP	Common bits can be R/W freely

8.2.6.7 IOMUXC_LPSR General Purpose Register 6 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR6)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 18h offset = 3027_0018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR6 field descriptions

Field	Description
GP	Common bits can be R/W freely

8.2.6.8 IOMUXC_LPSR General Purpose Register 7 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR7)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 1Ch offset = 3027_001Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR7 field descriptions

Field	Description
GP	Common bits can be R/W freely

8.2.6.9 IOMUXC_LPSR General Purpose Register 8 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR8)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 20h offset = 3027_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR8 field descriptions

Field	Description
GP	Common bits can be R/W freely

8.2.6.10 IOMUXC_LPSR General Purpose Register 9 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR9)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 24h offset = 3027_0024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STICKY																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR9 field descriptions

Field	Description
STICKY	Each bit is sticky. Once it's written to 1'b1, it cannot be written back to 1'b0.

8.2.6.11 IOMUXC_LPSR General Purpose Register 10 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR10)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 28h offset = 3027_0028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STICKY																LOCK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR10 field descriptions

Field	Description
31–16 STICKY	Bits [31:16] are lock bits for [15:0]. Each of bits [31:16] are sticky. Once it's written to 1'b1, it can not be written back to 1'b0.
LOCK	These are lock type bits. For example, bit [15] can't be written to new value when bit [31] is 1'b1 or new written value of bit [31] is 1'b1.

8.2.6.12 IOMUXC_LPSR General Purpose Register 11 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR11)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 2Ch offset = 3027_002Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STICKY																LOCK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR11 field descriptions

Field	Description
31–16 STICKY	Bits [31:16] are lock bits for [15:0]. Each of bits [31:16] are sticky. Once it's written to 1'b1, it can not be written back to 1'b0.
LOCK	These are lock type bits. For example, bit [15] can't be written to new value when bit [31] is 1'b1 or new written value of bit [31] is 1'b1.

8.2.6.13 IOMUXC_LPSR General Purpose Register 12 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR12)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 30h offset = 3027_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP																															
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR12 field descriptions

Field	Description
GP	Common bits can be R/W freely

8.2.6.14 IOMUXC_LPSR General Purpose Register 13 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR13)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 34h offset = 3027_0034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RO																GP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR13 field descriptions

Field	Description
31–16 RO	Read only bits, always 16'h0
GP	Common bits can be R/W freely

8.2.6.15 IOMUXC_LPSR General Purpose Register 14 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR14)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 38h offset = 3027_0038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR14 field descriptions

Field	Description
GP	Common bits can be R/W freely

8.2.6.16 IOMUXC_LPSR General Purpose Register 15 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR15)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 3Ch offset = 3027_003Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR15 field descriptions

Field	Description
GP	Common bits can be R/W freely

8.2.6.17 IOMUXC_LPSR General Purpose Register 16 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR16)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 40h offset = 3027_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR16 field descriptions

Field	Description
GP	Common bits can be R/W freely

8.2.6.18 IOMUXC_LPSR General Purpose Register 17 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR17)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 44h offset = 3027_0044h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR17 field descriptions

Field	Description
GP	Common bits can be R/W freely

8.2.6.19 IOMUXC_LPSR General Purpose Register 18 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR18)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 48h offset = 3027_0048h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR18 field descriptions

Field	Description
GP	Common bits can be R/W freely

8.2.6.20 IOMUXC_LPSR General Purpose Register 19 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR19)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 4Ch offset = 3027_004Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR19 field descriptions

Field	Description
GP	Common bits can be R/W freely

8.2.6.21 IOMUXC_LPSR General Purpose Register 20 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR20)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 50h offset = 3027_0050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPIO1_IO11_MUX_CTL	GPIO1_IO11_PS		GPIO1_IO11_PE	GPIO1_IO11_HYS	GPIO1_IO11_SRE	GPIO1_IO11_DSE		GPIO1_IO10_MUX_CTL	GPIO1_IO10_PS		GPIO1_IO10_PE	GPIO1_IO10_HYS	GPIO1_IO10_SRE	GPIO1_IO10_DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPIO1_IO09_MUX_CTL	GPIO1_IO09_PS		GPIO1_IO09_PE	GPIO1_IO09_HYS	GPIO1_IO09_SRE	GPIO1_IO09_DSE		GPIO1_IO08_MUX_CTL	GPIO1_IO08_PS		GPIO1_IO08_PE	GPIO1_IO08_HYS	GPIO1_IO08_SRE	GPIO1_IO08_DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR20 field descriptions

Field	Description
31 GPIO1_IO11_MUX_CTL	Mux control for PAD 0 Pad control from SOC IOMUX 1 Pad control from LPSR GPR
30–29 GPIO1_IO11_PS	Pull Select Field Select one out of next values for pad: GPIO1_IO11 00 PS_0_100K_PD — 100K PD 01 PS_1_5K_PU — 5K PU 10 PS_2_47K_PU — 47K PU 11 PS_3_100K_PU — 100K PU
28 GPIO1_IO11_PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO11 0 PE_0_Pull_Disabled — Pull Disabled 1 PE_1_Pull_Enabled — Pull Enabled

Table continues on the next page...

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR20 field descriptions (continued)

Field	Description
27 GPIO1_IO11_ HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO11 0 HYS_0_Hysteresis_Disabled — Hysteresis Disabled 1 HYS_1_Hysteresis_Enabled — Hysteresis Enable
26 GPIO1_IO11_ SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO11 0 SRE_0_Fast_Slew_Rate — Fast Slew Rate 1 SRE_1_Slow_Slew_Rate — Slow Slew Rate
25–24 GPIO1_IO11_ DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO11 00 DSE_0_X1 — X1 01 DSE_1_X1 — X2 10 DSE_2_X1 — X4 11 DSE_3_X1 — X6
23 GPIO1_IO10_ MUX_CTL	Mux control for PAD 0 Pad control from SOC IOMUX 1 Pad control from LPSR GPR
22–21 GPIO1_IO10_PS	Pull Select Field Select one out of next values for pad: GPIO1_IO10 00 PS_0_100K_PD — 100K PD 01 PS_1_5K_PU — 5K PU 10 PS_2_47K_PU — 47K PU 11 PS_3_100K_PU — 100K PU
20 GPIO1_IO10_PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO10 0 PE_0_Pull_Disabled — Pull Disabled 1 PE_1_Pull_Enabled — Pull Enabled
19 GPIO1_IO10_ HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO10 0 HYS_0_Hysteresis_Disabled — Hysteresis Disabled 1 HYS_1_Hysteresis_Enabled — Hysteresis Enable
18 GPIO1_IO10_ SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO10 0 SRE_0_Fast_Slew_Rate — Fast Slew Rate 1 SRE_1_Slow_Slew_Rate — Slow Slew Rate
17–16 GPIO1_IO10_ DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO10 00 DSE_0_X1 — X1 01 DSE_1_X1 — X2 10 DSE_2_X1 — X4 11 DSE_3_X1 — X6
15 GPIO1_IO09_ MUX_CTL	Mux control for PAD 0 Pad control from SOC IOMUX 1 Pad control from LPSR GPR
14–13 GPIO1_IO09_PS	Pull Select Field Select one out of next values for pad: GPIO1_IO09 00 PS_0_100K_PD — 100K PD

Table continues on the next page...

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR20 field descriptions (continued)

Field	Description
	01 PS_1_5K_PU — 5K PU 10 PS_2_47K_PU — 47K PU 11 PS_3_100K_PU — 100K PU
12 GPIO1_IO09_PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO09 0 PE_0_Pull_Disabled — Pull Disabled 1 PE_1_Pull_Enabled — Pull Enabled
11 GPIO1_IO09_ HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO09 0 HYS_0_Hysteresis_Disabled — Hysteresis Disabled 1 HYS_1_Hysteresis_Enabled — Hysteresis Enable
10 GPIO1_IO09_ SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO09 0 SRE_0_Fast_Slew_Rate — Fast Slew Rate 1 SRE_1_Slow_Slew_Rate — Slow Slew Rate
9–8 GPIO1_IO09_ DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO09 00 DSE_0_X1 — X1 01 DSE_1_X1 — X2 10 DSE_2_X1 — X4 11 DSE_3_X1 — X6
7 GPIO1_IO08_ MUX_CTL	Mux control for PAD 0 Pad control from SOC IOMUX 1 Pad control from LPSR GPR
6–5 GPIO1_IO08_PS	Pull Select Field Select one out of next values for pad: GPIO1_IO08 00 PS_0_100K_PD — 100K PD 01 PS_1_5K_PU — 5K PU 10 PS_2_47K_PU — 47K PU 11 PS_3_100K_PU — 100K PU
4 GPIO1_IO08_PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO08 0 PE_0_Pull_Disabled — Pull Disabled 1 PE_1_Pull_Enabled — Pull Enabled
3 GPIO1_IO08_ HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO08 0 HYS_0_Hysteresis_Disabled — Hysteresis Disabled 1 HYS_1_Hysteresis_Enabled — Hysteresis Enable
2 GPIO1_IO08_ SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO08 0 SRE_0_Fast_Slew_Rate — Fast Slew Rate 1 SRE_1_Slow_Slew_Rate — Slow Slew Rate
GPIO1_IO08_ DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO08 00 DSE_0_X1 — X1 01 DSE_1_X1 — X2 10 DSE_2_X1 — X4 11 DSE_3_X1 — X6

8.2.6.22 IOMUXC_LPSR General Purpose Register 21 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR21)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 54h offset = 3027_0054h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPIO1_IO15_MUX_CTL	GPIO1_IO15_PS		GPIO1_IO15_PE	GPIO1_IO15_HYS	GPIO1_IO15_SRE	GPIO1_IO15_DSE		GPIO1_IO14_MUX_CTL	GPIO1_IO14_PS		GPIO1_IO14_PE	GPIO1_IO14_HYS	GPIO1_IO14_SRE	GPIO1_IO14_DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPIO1_IO13_MUX_CTL	GPIO1_IO13_PS		GPIO1_IO13_PE	GPIO1_IO13_HYS	GPIO1_IO13_SRE	GPIO1_IO13_DSE		GPIO1_IO12_MUX_CTL	GPIO1_IO12_PS		GPIO1_IO12_PE	GPIO1_IO12_HYS	GPIO1_IO12_SRE	GPIO1_IO12_DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR21 field descriptions

Field	Description
31 GPIO1_IO15_MUX_CTL	Mux control for PAD 0 Pad control from SOC IOMUX 1 Pad control from LPSR GPR
30–29 GPIO1_IO15_PS	Pull Select Field Select one out of next values for pad: GPIO1_IO15 00 PS_0_100K_PD — 100K PD 01 PS_1_5K_PU — 5K PU 10 PS_2_47K_PU — 47K PU 11 PS_3_100K_PU — 100K PU
28 GPIO1_IO15_PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO15 0 PE_0_Pull_Disabled — Pull Disabled 1 PE_1_Pull_Enabled — Pull Enabled
27 GPIO1_IO15_HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO15 0 HYS_0_Hysteresis_Disabled — Hysteresis Disabled 1 HYS_1_Hysteresis_Enabled — Hysteresis Enable

Table continues on the next page...

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR21 field descriptions (continued)

Field	Description
26 GPIO1_IO15_ SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO15 0 SRE_0_Fast_Slew_Rate — Fast Slew Rate 1 SRE_1_Slow_Slew_Rate — Slow Slew Rate
25–24 GPIO1_IO15_ DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO15 00 DSE_0_X1 — X1 01 DSE_1_X1 — X2 10 DSE_2_X1 — X4 11 DSE_3_X1 — X6
23 GPIO1_IO14_ MUX_CTL	Mux control for PAD 0 Pad control from SOC IOMUX 1 Pad control from LPSR GPR
22–21 GPIO1_IO14_PS	Pull Select Field Select one out of next values for pad: GPIO1_IO14 00 PS_0_100K_PD — 100K PD 01 PS_1_5K_PU — 5K PU 10 PS_2_47K_PU — 47K PU 11 PS_3_100K_PU — 100K PU
20 GPIO1_IO14_PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO14 0 PE_0_Pull_Disabled — Pull Disabled 1 PE_1_Pull_Enabled — Pull Enabled
19 GPIO1_IO14_ HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO14 0 HYS_0_Hysteresis_Disabled — Hysteresis Disabled 1 HYS_1_Hysteresis_Enabled — Hysteresis Enable
18 GPIO1_IO14_ SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO14 0 SRE_0_Fast_Slew_Rate — Fast Slew Rate 1 SRE_1_Slow_Slew_Rate — Slow Slew Rate
17–16 GPIO1_IO14_ DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO14 00 DSE_0_X1 — X1 01 DSE_1_X1 — X2 10 DSE_2_X1 — X4 11 DSE_3_X1 — X6
15 GPIO1_IO13_ MUX_CTL	Mux control for PAD 0 Pad control from SOC IOMUX 1 Pad control from LPSR GPR
14–13 GPIO1_IO13_PS	Pull Select Field Select one out of next values for pad: GPIO1_IO13 00 PS_0_100K_PD — 100K PD 01 PS_1_5K_PU — 5K PU 10 PS_2_47K_PU — 47K PU 11 PS_3_100K_PU — 100K PU

Table continues on the next page...

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR21 field descriptions (continued)

Field	Description
12 GPIO1_IO13_PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO13 0 PE_0_Pull_Disabled — Pull Disabled 1 PE_1_Pull_Enabled — Pull Enabled
11 GPIO1_IO13_ HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO13 0 HYS_0_Hysteresis_Disabled — Hysteresis Disabled 1 HYS_1_Hysteresis_Enabled — Hysteresis Enable
10 GPIO1_IO13_ SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO13 0 SRE_0_Fast_Slew_Rate — Fast Slew Rate 1 SRE_1_Slow_Slew_Rate — Slow Slew Rate
9–8 GPIO1_IO13_ DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO13 00 DSE_0_X1 — X1 01 DSE_1_X1 — X2 10 DSE_2_X1 — X4 11 DSE_3_X1 — X6
7 GPIO1_IO12_ MUX_CTL	Mux control for PAD 0 Pad control from SOC IOMUX 1 Pad control from LPSR GPR
6–5 GPIO1_IO12_PS	Pull Select Field Select one out of next values for pad: GPIO1_IO12 00 PS_0_100K_PD — 100K PD 01 PS_1_5K_PU — 5K PU 10 PS_2_47K_PU — 47K PU 11 PS_3_100K_PU — 100K PU
4 GPIO1_IO12_PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO12 0 PE_0_Pull_Disabled — Pull Disabled 1 PE_1_Pull_Enabled — Pull Enabled
3 GPIO1_IO12_ HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO12 0 HYS_0_Hysteresis_Disabled — Hysteresis Disabled 1 HYS_1_Hysteresis_Enabled — Hysteresis Enable
2 GPIO1_IO12_ SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO12 0 SRE_0_Fast_Slew_Rate — Fast Slew Rate 1 SRE_1_Slow_Slew_Rate — Slow Slew Rate
GPIO1_IO12_ DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO12 00 DSE_0_X1 — X1 01 DSE_1_X1 — X2 10 DSE_2_X1 — X4 11 DSE_3_X1 — X6

8.2.6.23 IOMUXC_LPSR General Purpose Register 22 (IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR22)

IOMUXC LPSR General Purpose Register

Address: 3027_0000h base + 58h offset = 3027_0058h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RO																GP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSR_GPR_IOMUXC_LPSR_GPR22 field descriptions

Field	Description
31–16 RO	Read only bits, always 16'h0
GP	Common bits can be R/W freely

8.2.7 IOMUXC Memory Map/Register Definition

IOMUXC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_0014	SW_MUX_CTL_PAD_GPIO1_IO08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO08)	32	R/W	0000_0000h	8.2.7.1/ 1586
3033_0018	SW_MUX_CTL_PAD_GPIO1_IO09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO09)	32	R/W	0000_0000h	8.2.7.2/ 1588
3033_001C	SW_MUX_CTL_PAD_GPIO1_IO10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO10)	32	R/W	0000_0000h	8.2.7.3/ 1589
3033_0020	SW_MUX_CTL_PAD_GPIO1_IO11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO11)	32	R/W	0000_0000h	8.2.7.4/ 1590
3033_0024	SW_MUX_CTL_PAD_GPIO1_IO12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO12)	32	R/W	0000_0000h	8.2.7.5/ 1592
3033_0028	SW_MUX_CTL_PAD_GPIO1_IO13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO13)	32	R/W	0000_0000h	8.2.7.6/ 1593
3033_002C	SW_MUX_CTL_PAD_GPIO1_IO14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO14)	32	R/W	0000_0000h	8.2.7.7/ 1594
3033_0030	SW_MUX_CTL_PAD_GPIO1_IO15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO15)	32	R/W	0000_0000h	8.2.7.8/ 1596
3033_0034	SW_MUX_CTL_PAD_EPDC_DATA00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA00)	32	R/W	0000_0005h	8.2.7.9/ 1597
3033_0038	SW_MUX_CTL_PAD_EPDC_DATA01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA01)	32	R/W	0000_0005h	8.2.7.10/ 1598

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_003C	SW_MUX_CTL_PAD_EPDC_DATA02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA02)	32	R/W	0000_0005h	8.2.7.11/ 1600
3033_0040	SW_MUX_CTL_PAD_EPDC_DATA03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA03)	32	R/W	0000_0005h	8.2.7.12/ 1601
3033_0044	SW_MUX_CTL_PAD_EPDC_DATA04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA04)	32	R/W	0000_0005h	8.2.7.13/ 1602
3033_0048	SW_MUX_CTL_PAD_EPDC_DATA05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA05)	32	R/W	0000_0005h	8.2.7.14/ 1604
3033_004C	SW_MUX_CTL_PAD_EPDC_DATA06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA06)	32	R/W	0000_0005h	8.2.7.15/ 1605
3033_0050	SW_MUX_CTL_PAD_EPDC_DATA07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA07)	32	R/W	0000_0005h	8.2.7.16/ 1606
3033_0054	SW_MUX_CTL_PAD_EPDC_DATA08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA08)	32	R/W	0000_0005h	8.2.7.17/ 1607
3033_0058	SW_MUX_CTL_PAD_EPDC_DATA09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA09)	32	R/W	0000_0005h	8.2.7.18/ 1608
3033_005C	SW_MUX_CTL_PAD_EPDC_DATA10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA10)	32	R/W	0000_0005h	8.2.7.19/ 1609
3033_0060	SW_MUX_CTL_PAD_EPDC_DATA11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA11)	32	R/W	0000_0005h	8.2.7.20/ 1610
3033_0064	SW_MUX_CTL_PAD_EPDC_DATA12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA12)	32	R/W	0000_0005h	8.2.7.21/ 1611
3033_0068	SW_MUX_CTL_PAD_EPDC_DATA13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA13)	32	R/W	0000_0005h	8.2.7.22/ 1612
3033_006C	SW_MUX_CTL_PAD_EPDC_DATA14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA14)	32	R/W	0000_0005h	8.2.7.23/ 1613
3033_0070	SW_MUX_CTL_PAD_EPDC_DATA15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA15)	32	R/W	0000_0005h	8.2.7.24/ 1614
3033_0074	SW_MUX_CTL_PAD_EPDC_SDCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCLK)	32	R/W	0000_0005h	8.2.7.25/ 1615
3033_0078	SW_MUX_CTL_PAD_EPDC_SDLE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDLE)	32	R/W	0000_0005h	8.2.7.26/ 1616
3033_007C	SW_MUX_CTL_PAD_EPDC_SDOE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDOE)	32	R/W	0000_0005h	8.2.7.27/ 1617
3033_0080	SW_MUX_CTL_PAD_EPDC_SDSHR SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDSHR)	32	R/W	0000_0005h	8.2.7.28/ 1619
3033_0084	SW_MUX_CTL_PAD_EPDC_SDCE0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE0)	32	R/W	0000_0005h	8.2.7.29/ 1620
3033_0088	SW_MUX_CTL_PAD_EPDC_SDCE1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE1)	32	R/W	0000_0005h	8.2.7.30/ 1621
3033_008C	SW_MUX_CTL_PAD_EPDC_SDCE2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE2)	32	R/W	0000_0005h	8.2.7.31/ 1623
3033_0090	SW_MUX_CTL_PAD_EPDC_SDCE3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE3)	32	R/W	0000_0005h	8.2.7.32/ 1624

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_0094	SW_MUX_CTL_PAD_EPDC_GDCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_GDCLK)	32	R/W	0000_0005h	8.2.7.33/ 1625
3033_0098	SW_MUX_CTL_PAD_EPDC_GDOE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_GDOE)	32	R/W	0000_0005h	8.2.7.34/ 1627
3033_009C	SW_MUX_CTL_PAD_EPDC_GDRL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_GDRL)	32	R/W	0000_0005h	8.2.7.35/ 1628
3033_00A0	SW_MUX_CTL_PAD_EPDC_GDSP SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_GDSP)	32	R/W	0000_0005h	8.2.7.36/ 1629
3033_00A4	SW_MUX_CTL_PAD_EPDC_BDR0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_BDR0)	32	R/W	0000_0005h	8.2.7.37/ 1631
3033_00A8	SW_MUX_CTL_PAD_EPDC_BDR1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_BDR1)	32	R/W	0000_0005h	8.2.7.38/ 1632
3033_00AC	SW_MUX_CTL_PAD_EPDC_PWR_COM SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_PWR_COM)	32	R/W	0000_0005h	8.2.7.39/ 1633
3033_00B0	SW_MUX_CTL_PAD_EPDC_PWR_STAT SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_PWR_STAT)	32	R/W	0000_0005h	8.2.7.40/ 1635
3033_00B4	SW_MUX_CTL_PAD_LCD_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_CLK)	32	R/W	0000_0005h	8.2.7.41/ 1636
3033_00B8	SW_MUX_CTL_PAD_LCD_ENABLE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_ENABLE)	32	R/W	0000_0005h	8.2.7.42/ 1637
3033_00BC	SW_MUX_CTL_PAD_LCD_HSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_HSYNC)	32	R/W	0000_0005h	8.2.7.43/ 1639
3033_00C0	SW_MUX_CTL_PAD_LCD_VSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_VSYNC)	32	R/W	0000_0005h	8.2.7.44/ 1640
3033_00C4	SW_MUX_CTL_PAD_LCD_RESET SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_RESET)	32	R/W	0000_0005h	8.2.7.45/ 1641
3033_00C8	SW_MUX_CTL_PAD_LCD_DATA00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA00)	32	R/W	0000_0005h	8.2.7.46/ 1643
3033_00CC	SW_MUX_CTL_PAD_LCD_DATA01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA01)	32	R/W	0000_0005h	8.2.7.47/ 1644
3033_00D0	SW_MUX_CTL_PAD_LCD_DATA02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA02)	32	R/W	0000_0005h	8.2.7.48/ 1645
3033_00D4	SW_MUX_CTL_PAD_LCD_DATA03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA03)	32	R/W	0000_0005h	8.2.7.49/ 1647
3033_00D8	SW_MUX_CTL_PAD_LCD_DATA04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA04)	32	R/W	0000_0005h	8.2.7.50/ 1648
3033_00DC	SW_MUX_CTL_PAD_LCD_DATA05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA05)	32	R/W	0000_0005h	8.2.7.51/ 1649
3033_00E0	SW_MUX_CTL_PAD_LCD_DATA06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA06)	32	R/W	0000_0005h	8.2.7.52/ 1651
3033_00E4	SW_MUX_CTL_PAD_LCD_DATA07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA07)	32	R/W	0000_0005h	8.2.7.53/ 1652

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_00E8	SW_MUX_CTL_PAD_LCD_DATA08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA08)	32	R/W	0000_0005h	8.2.7.54/ 1653
3033_00EC	SW_MUX_CTL_PAD_LCD_DATA09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA09)	32	R/W	0000_0005h	8.2.7.55/ 1654
3033_00F0	SW_MUX_CTL_PAD_LCD_DATA10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA10)	32	R/W	0000_0005h	8.2.7.56/ 1656
3033_00F4	SW_MUX_CTL_PAD_LCD_DATA11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA11)	32	R/W	0000_0005h	8.2.7.57/ 1657
3033_00F8	SW_MUX_CTL_PAD_LCD_DATA12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA12)	32	R/W	0000_0005h	8.2.7.58/ 1658
3033_00FC	SW_MUX_CTL_PAD_LCD_DATA13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA13)	32	R/W	0000_0005h	8.2.7.59/ 1659
3033_0100	SW_MUX_CTL_PAD_LCD_DATA14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA14)	32	R/W	0000_0005h	8.2.7.60/ 1661
3033_0104	SW_MUX_CTL_PAD_LCD_DATA15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA15)	32	R/W	0000_0005h	8.2.7.61/ 1662
3033_0108	SW_MUX_CTL_PAD_LCD_DATA16 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA16)	32	R/W	0000_0005h	8.2.7.62/ 1663
3033_010C	SW_MUX_CTL_PAD_LCD_DATA17 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA17)	32	R/W	0000_0005h	8.2.7.63/ 1664
3033_0110	SW_MUX_CTL_PAD_LCD_DATA18 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA18)	32	R/W	0000_0005h	8.2.7.64/ 1666
3033_0114	SW_MUX_CTL_PAD_LCD_DATA19 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA19)	32	R/W	0000_0005h	8.2.7.65/ 1667
3033_0118	SW_MUX_CTL_PAD_LCD_DATA20 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA20)	32	R/W	0000_0005h	8.2.7.66/ 1668
3033_011C	SW_MUX_CTL_PAD_LCD_DATA21 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA21)	32	R/W	0000_0005h	8.2.7.67/ 1670
3033_0120	SW_MUX_CTL_PAD_LCD_DATA22 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA22)	32	R/W	0000_0005h	8.2.7.68/ 1671
3033_0124	SW_MUX_CTL_PAD_LCD_DATA23 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA23)	32	R/W	0000_0005h	8.2.7.69/ 1672
3033_0128	SW_MUX_CTL_PAD_UART1_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_RX_DATA)	32	R/W	0000_0005h	8.2.7.70/ 1674
3033_012C	SW_MUX_CTL_PAD_UART1_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_TX_DATA)	32	R/W	0000_0005h	8.2.7.71/ 1675
3033_0130	SW_MUX_CTL_PAD_UART2_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_RX_DATA)	32	R/W	0000_0005h	8.2.7.72/ 1676
3033_0134	SW_MUX_CTL_PAD_UART2_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_TX_DATA)	32	R/W	0000_0005h	8.2.7.73/ 1678

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_0138	SW_MUX_CTL_PAD_UART3_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_RX_DATA)	32	R/W	0000_0005h	8.2.7.74/ 1679
3033_013C	SW_MUX_CTL_PAD_UART3_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_TX_DATA)	32	R/W	0000_0005h	8.2.7.75/ 1680
3033_0140	SW_MUX_CTL_PAD_UART3_RTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_RTS_B)	32	R/W	0000_0005h	8.2.7.76/ 1682
3033_0144	SW_MUX_CTL_PAD_UART3_CTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_CTS_B)	32	R/W	0000_0005h	8.2.7.77/ 1683
3033_0148	SW_MUX_CTL_PAD_I2C1_SCL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C1_SCL)	32	R/W	0000_0005h	8.2.7.78/ 1684
3033_014C	SW_MUX_CTL_PAD_I2C1_SDA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C1_SDA)	32	R/W	0000_0005h	8.2.7.79/ 1686
3033_0150	SW_MUX_CTL_PAD_I2C2_SCL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C2_SCL)	32	R/W	0000_0005h	8.2.7.80/ 1687
3033_0154	SW_MUX_CTL_PAD_I2C2_SDA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C2_SDA)	32	R/W	0000_0005h	8.2.7.81/ 1688
3033_0158	SW_MUX_CTL_PAD_I2C3_SCL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C3_SCL)	32	R/W	0000_0005h	8.2.7.82/ 1690
3033_015C	SW_MUX_CTL_PAD_I2C3_SDA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C3_SDA)	32	R/W	0000_0005h	8.2.7.83/ 1691
3033_0160	SW_MUX_CTL_PAD_I2C4_SCL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C4_SCL)	32	R/W	0000_0005h	8.2.7.84/ 1692
3033_0164	SW_MUX_CTL_PAD_I2C4_SDA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C4_SDA)	32	R/W	0000_0005h	8.2.7.85/ 1694
3033_0168	SW_MUX_CTL_PAD_ECSPI1_SCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI1_SCLK)	32	R/W	0000_0005h	8.2.7.86/ 1695
3033_016C	SW_MUX_CTL_PAD_ECSPI1_MOSI SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI1_MOSI)	32	R/W	0000_0005h	8.2.7.87/ 1696
3033_0170	SW_MUX_CTL_PAD_ECSPI1_MISO SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI1_MISO)	32	R/W	0000_0005h	8.2.7.88/ 1698
3033_0174	SW_MUX_CTL_PAD_ECSPI1_SS0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI1_SS0)	32	R/W	0000_0005h	8.2.7.89/ 1699
3033_0178	SW_MUX_CTL_PAD_ECSPI2_SCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI2_SCLK)	32	R/W	0000_0005h	8.2.7.90/ 1700
3033_017C	SW_MUX_CTL_PAD_ECSPI2_MOSI SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI2_MOSI)	32	R/W	0000_0005h	8.2.7.91/ 1702
3033_0180	SW_MUX_CTL_PAD_ECSPI2_MISO SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI2_MISO)	32	R/W	0000_0005h	8.2.7.92/ 1703
3033_0184	SW_MUX_CTL_PAD_ECSPI2_SS0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI2_SS0)	32	R/W	0000_0005h	8.2.7.93/ 1704
3033_0188	SW_MUX_CTL_PAD_SD1_CD_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_CD_B)	32	R/W	0000_0005h	8.2.7.94/ 1706

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_018C	SW_MUX_CTL_PAD_SD1_WP SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_WP)	32	R/W	0000_0005h	8.2.7.95/ 1707
3033_0190	SW_MUX_CTL_PAD_SD1_RESET_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_RESET_B)	32	R/W	0000_0005h	8.2.7.96/ 1708
3033_0194	SW_MUX_CTL_PAD_SD1_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_CLK)	32	R/W	0000_0005h	8.2.7.97/ 1710
3033_0198	SW_MUX_CTL_PAD_SD1_CMD SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_CMD)	32	R/W	0000_0005h	8.2.7.98/ 1711
3033_019C	SW_MUX_CTL_PAD_SD1_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA0)	32	R/W	0000_0005h	8.2.7.99/ 1712
3033_01A0	SW_MUX_CTL_PAD_SD1_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA1)	32	R/W	0000_0005h	8.2.7.100/ 1714
3033_01A4	SW_MUX_CTL_PAD_SD1_DATA2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA2)	32	R/W	0000_0005h	8.2.7.101/ 1715
3033_01A8	SW_MUX_CTL_PAD_SD1_DATA3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA3)	32	R/W	0000_0005h	8.2.7.102/ 1716
3033_01AC	SW_MUX_CTL_PAD_SD2_CD_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_CD_B)	32	R/W	0000_0005h	8.2.7.103/ 1718
3033_01B0	SW_MUX_CTL_PAD_SD2_WP SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_WP)	32	R/W	0000_0005h	8.2.7.104/ 1719
3033_01B4	SW_MUX_CTL_PAD_SD2_RESET_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_RESET_B)	32	R/W	0000_0005h	8.2.7.105/ 1720
3033_01B8	SW_MUX_CTL_PAD_SD2_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_CLK)	32	R/W	0000_0005h	8.2.7.106/ 1722
3033_01BC	SW_MUX_CTL_PAD_SD2_CMD SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_CMD)	32	R/W	0000_0005h	8.2.7.107/ 1723
3033_01C0	SW_MUX_CTL_PAD_SD2_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_DATA0)	32	R/W	0000_0005h	8.2.7.108/ 1724
3033_01C4	SW_MUX_CTL_PAD_SD2_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_DATA1)	32	R/W	0000_0005h	8.2.7.109/ 1725
3033_01C8	SW_MUX_CTL_PAD_SD2_DATA2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_DATA2)	32	R/W	0000_0005h	8.2.7.110/ 1727
3033_01CC	SW_MUX_CTL_PAD_SD2_DATA3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_DATA3)	32	R/W	0000_0005h	8.2.7.111/ 1728
3033_01D0	SW_MUX_CTL_PAD_SD3_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_CLK)	32	R/W	0000_0005h	8.2.7.112/ 1729
3033_01D4	SW_MUX_CTL_PAD_SD3_CMD SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_CMD)	32	R/W	0000_0005h	8.2.7.113/ 1731
3033_01D8	SW_MUX_CTL_PAD_SD3_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA0)	32	R/W	0000_0005h	8.2.7.114/ 1732
3033_01DC	SW_MUX_CTL_PAD_SD3_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA1)	32	R/W	0000_0005h	8.2.7.115/ 1733
3033_01E0	SW_MUX_CTL_PAD_SD3_DATA2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA2)	32	R/W	0000_0005h	8.2.7.116/ 1735

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_01E4	SW_MUX_CTL_PAD_SD3_DATA3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA3)	32	R/W	0000_0005h	8.2.7.117/ 1736
3033_01E8	SW_MUX_CTL_PAD_SD3_DATA4 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA4)	32	R/W	0000_0005h	8.2.7.118/ 1737
3033_01EC	SW_MUX_CTL_PAD_SD3_DATA5 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA5)	32	R/W	0000_0005h	8.2.7.119/ 1739
3033_01F0	SW_MUX_CTL_PAD_SD3_DATA6 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA6)	32	R/W	0000_0005h	8.2.7.120/ 1740
3033_01F4	SW_MUX_CTL_PAD_SD3_DATA7 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA7)	32	R/W	0000_0005h	8.2.7.121/ 1741
3033_01F8	SW_MUX_CTL_PAD_SD3_STROBE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_STROBE)	32	R/W	0000_0005h	8.2.7.122/ 1742
3033_01FC	SW_MUX_CTL_PAD_SD3_RESET_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_RESET_B)	32	R/W	0000_0005h	8.2.7.123/ 1743
3033_0200	SW_MUX_CTL_PAD_SAI1_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI1_RX_DATA)	32	R/W	0000_0005h	8.2.7.124/ 1745
3033_0204	SW_MUX_CTL_PAD_SAI1_TX_BCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI1_TX_BCLK)	32	R/W	0000_0005h	8.2.7.125/ 1746
3033_0208	SW_MUX_CTL_PAD_SAI1_TX_SYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI1_TX_SYNC)	32	R/W	0000_0005h	8.2.7.126/ 1747
3033_020C	SW_MUX_CTL_PAD_SAI1_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI1_TX_DATA)	32	R/W	0000_0005h	8.2.7.127/ 1749
3033_0210	SW_MUX_CTL_PAD_SAI1_RX_SYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI1_RX_SYNC)	32	R/W	0000_0005h	8.2.7.128/ 1750
3033_0214	SW_MUX_CTL_PAD_SAI1_RX_BCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI1_RX_BCLK)	32	R/W	0000_0005h	8.2.7.129/ 1751
3033_0218	SW_MUX_CTL_PAD_SAI1_MCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI1_MCLK)	32	R/W	0000_0005h	8.2.7.130/ 1753
3033_021C	SW_MUX_CTL_PAD_SAI2_TX_SYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI2_TX_SYNC)	32	R/W	0000_0005h	8.2.7.131/ 1754
3033_0220	SW_MUX_CTL_PAD_SAI2_TX_BCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI2_TX_BCLK)	32	R/W	0000_0005h	8.2.7.132/ 1755
3033_0224	SW_MUX_CTL_PAD_SAI2_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI2_RX_DATA)	32	R/W	0000_0005h	8.2.7.133/ 1757
3033_0228	SW_MUX_CTL_PAD_SAI2_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI2_TX_DATA)	32	R/W	0000_0005h	8.2.7.134/ 1758
3033_022C	SW_MUX_CTL_PAD_ENET1_RGMII_RD0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RD0)	32	R/W	0000_0005h	8.2.7.135/ 1759
3033_0230	SW_MUX_CTL_PAD_ENET1_RGMII_RD1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RD1)	32	R/W	0000_0005h	8.2.7.136/ 1761
3033_0234	SW_MUX_CTL_PAD_ENET1_RGMII_RD2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RD2)	32	R/W	0000_0005h	8.2.7.137/ 1762

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_0238	SW_MUX_CTL_PAD_ENET1_RGMII_RD3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RD3)	32	R/W	0000_0005h	8.2.7.138/ 1763
3033_023C	SW_MUX_CTL_PAD_ENET1_RGMII_RX_CTL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RX_CTL)	32	R/W	0000_0005h	8.2.7.139/ 1765
3033_0240	SW_MUX_CTL_PAD_ENET1_RGMII_RXC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RXC)	32	R/W	0000_0005h	8.2.7.140/ 1766
3033_0244	SW_MUX_CTL_PAD_ENET1_RGMII_TD0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TD0)	32	R/W	0000_0005h	8.2.7.141/ 1767
3033_0248	SW_MUX_CTL_PAD_ENET1_RGMII_TD1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TD1)	32	R/W	0000_0005h	8.2.7.142/ 1769
3033_024C	SW_MUX_CTL_PAD_ENET1_RGMII_TD2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TD2)	32	R/W	0000_0005h	8.2.7.143/ 1770
3033_0250	SW_MUX_CTL_PAD_ENET1_RGMII_TD3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TD3)	32	R/W	0000_0005h	8.2.7.144/ 1771
3033_0254	SW_MUX_CTL_PAD_ENET1_RGMII_TX_CTL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TX_CTL)	32	R/W	0000_0005h	8.2.7.145/ 1773
3033_0258	SW_MUX_CTL_PAD_ENET1_RGMII_TXC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TXC)	32	R/W	0000_0005h	8.2.7.146/ 1774
3033_025C	SW_MUX_CTL_PAD_ENET1_TX_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_CLK)	32	R/W	0000_0005h	8.2.7.147/ 1775
3033_0260	SW_MUX_CTL_PAD_ENET1_RX_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_CLK)	32	R/W	0000_0005h	8.2.7.148/ 1777
3033_0264	SW_MUX_CTL_PAD_ENET1_CRS SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_CRS)	32	R/W	0000_0005h	8.2.7.149/ 1778
3033_0268	SW_MUX_CTL_PAD_ENET1_COL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_COL)	32	R/W	0000_0005h	8.2.7.150/ 1779
3033_026C	SW_PAD_CTL_PAD_GPIO1_IO08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO08)	32	R/W	0000_0014h	8.2.7.151/ 1780
3033_0270	SW_PAD_CTL_PAD_GPIO1_IO09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO09)	32	R/W	0000_0014h	8.2.7.152/ 1781
3033_0274	SW_PAD_CTL_PAD_GPIO1_IO10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO10)	32	R/W	0000_0014h	8.2.7.153/ 1783
3033_0278	SW_PAD_CTL_PAD_GPIO1_IO11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO11)	32	R/W	0000_0014h	8.2.7.154/ 1784
3033_027C	SW_PAD_CTL_PAD_GPIO1_IO12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO12)	32	R/W	0000_0014h	8.2.7.155/ 1785

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_0280	SW_PAD_CTL_PAD_GPIO1_IO13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO13)	32	R/W	0000_0014h	8.2.7.156/ 1786
3033_0284	SW_PAD_CTL_PAD_GPIO1_IO14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO14)	32	R/W	0000_0014h	8.2.7.157/ 1787
3033_0288	SW_PAD_CTL_PAD_GPIO1_IO15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO15)	32	R/W	0000_0014h	8.2.7.158/ 1789
3033_028C	SW_PAD_CTL_PAD_JTAG_MOD SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_MOD)	32	R/W	0000_0014h	8.2.7.159/ 1790
3033_0290	SW_PAD_CTL_PAD_JTAG_TCK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TCK)	32	R/W	0000_0054h	8.2.7.160/ 1791
3033_0294	SW_PAD_CTL_PAD_JTAG_TDI SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TDI)	32	R/W	0000_0054h	8.2.7.161/ 1792
3033_0298	SW_PAD_CTL_PAD_JTAG_TDO SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TDO)	32	R/W	0000_0070h	8.2.7.162/ 1793
3033_029C	SW_PAD_CTL_PAD_JTAG_TMS SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TMS)	32	R/W	0000_0054h	8.2.7.163/ 1795
3033_02A0	SW_PAD_CTL_PAD_JTAG_TRST_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TRST_B)	32	R/W	0000_0054h	8.2.7.164/ 1796
3033_02A4	SW_PAD_CTL_PAD_EPDC_DATA00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA00)	32	R/W	0000_0014h	8.2.7.165/ 1797
3033_02A8	SW_PAD_CTL_PAD_EPDC_DATA01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA01)	32	R/W	0000_0014h	8.2.7.166/ 1798
3033_02AC	SW_PAD_CTL_PAD_EPDC_DATA02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA02)	32	R/W	0000_0014h	8.2.7.167/ 1799
3033_02B0	SW_PAD_CTL_PAD_EPDC_DATA03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA03)	32	R/W	0000_0014h	8.2.7.168/ 1800
3033_02B4	SW_PAD_CTL_PAD_EPDC_DATA04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA04)	32	R/W	0000_0014h	8.2.7.169/ 1802
3033_02B8	SW_PAD_CTL_PAD_EPDC_DATA05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA05)	32	R/W	0000_0014h	8.2.7.170/ 1803
3033_02BC	SW_PAD_CTL_PAD_EPDC_DATA06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA06)	32	R/W	0000_0014h	8.2.7.171/ 1804
3033_02C0	SW_PAD_CTL_PAD_EPDC_DATA07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA07)	32	R/W	0000_0014h	8.2.7.172/ 1805
3033_02C4	SW_PAD_CTL_PAD_EPDC_DATA08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA08)	32	R/W	0000_0014h	8.2.7.173/ 1806
3033_02C8	SW_PAD_CTL_PAD_EPDC_DATA09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA09)	32	R/W	0000_0014h	8.2.7.174/ 1808
3033_02CC	SW_PAD_CTL_PAD_EPDC_DATA10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA10)	32	R/W	0000_0014h	8.2.7.175/ 1809
3033_02D0	SW_PAD_CTL_PAD_EPDC_DATA11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA11)	32	R/W	0000_0014h	8.2.7.176/ 1810
3033_02D4	SW_PAD_CTL_PAD_EPDC_DATA12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA12)	32	R/W	0000_0014h	8.2.7.177/ 1811

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_02D8	SW_PAD_CTL_PAD_EPDC_DATA13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA13)	32	R/W	0000_0014h	8.2.7.178/ 1812
3033_02DC	SW_PAD_CTL_PAD_EPDC_DATA14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA14)	32	R/W	0000_0014h	8.2.7.179/ 1814
3033_02E0	SW_PAD_CTL_PAD_EPDC_DATA15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA15)	32	R/W	0000_0014h	8.2.7.180/ 1815
3033_02E4	SW_PAD_CTL_PAD_EPDC_SDCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCLK)	32	R/W	0000_0014h	8.2.7.181/ 1816
3033_02E8	SW_PAD_CTL_PAD_EPDC_SDLE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDLE)	32	R/W	0000_0014h	8.2.7.182/ 1817
3033_02EC	SW_PAD_CTL_PAD_EPDC_SDOE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDOE)	32	R/W	0000_0014h	8.2.7.183/ 1818
3033_02F0	SW_PAD_CTL_PAD_EPDC_SDSHR SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDSHR)	32	R/W	0000_0014h	8.2.7.184/ 1820
3033_02F4	SW_PAD_CTL_PAD_EPDC_SDCE0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE0)	32	R/W	0000_0014h	8.2.7.185/ 1821
3033_02F8	SW_PAD_CTL_PAD_EPDC_SDCE1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE1)	32	R/W	0000_0014h	8.2.7.186/ 1822
3033_02FC	SW_PAD_CTL_PAD_EPDC_SDCE2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE2)	32	R/W	0000_0014h	8.2.7.187/ 1823
3033_0300	SW_PAD_CTL_PAD_EPDC_SDCE3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE3)	32	R/W	0000_0014h	8.2.7.188/ 1824
3033_0304	SW_PAD_CTL_PAD_EPDC_GDCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_GDCLK)	32	R/W	0000_0014h	8.2.7.189/ 1826
3033_0308	SW_PAD_CTL_PAD_EPDC_GDOE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_GDOE)	32	R/W	0000_0014h	8.2.7.190/ 1827
3033_030C	SW_PAD_CTL_PAD_EPDC_GDRL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_GDRL)	32	R/W	0000_0014h	8.2.7.191/ 1828
3033_0310	SW_PAD_CTL_PAD_EPDC_GDSP SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_GDSP)	32	R/W	0000_0014h	8.2.7.192/ 1829
3033_0314	SW_PAD_CTL_PAD_EPDC_BDR0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_BDR0)	32	R/W	0000_0014h	8.2.7.193/ 1830
3033_0318	SW_PAD_CTL_PAD_EPDC_BDR1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_BDR1)	32	R/W	0000_0014h	8.2.7.194/ 1832
3033_031C	SW_PAD_CTL_PAD_EPDC_PWR_COM SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_PWR_COM)	32	R/W	0000_0014h	8.2.7.195/ 1833
3033_0320	SW_PAD_CTL_PAD_EPDC_PWR_STAT SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_PWR_STAT)	32	R/W	0000_0014h	8.2.7.196/ 1834
3033_0324	SW_PAD_CTL_PAD_LCD_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_CLK)	32	R/W	0000_0014h	8.2.7.197/ 1835
3033_0328	SW_PAD_CTL_PAD_LCD_ENABLE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_ENABLE)	32	R/W	0000_0014h	8.2.7.198/ 1836

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_032C	SW_PAD_CTL_PAD_LCD_HSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_HSYNC)	32	R/W	0000_0014h	8.2.7.199/ 1838
3033_0330	SW_PAD_CTL_PAD_LCD_VSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_VSYNC)	32	R/W	0000_0014h	8.2.7.200/ 1839
3033_0334	SW_PAD_CTL_PAD_LCD_RESET SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_RESET)	32	R/W	0000_0014h	8.2.7.201/ 1840
3033_0338	SW_PAD_CTL_PAD_LCD_DATA00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA00)	32	R/W	0000_0014h	8.2.7.202/ 1841
3033_033C	SW_PAD_CTL_PAD_LCD_DATA01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA01)	32	R/W	0000_0014h	8.2.7.203/ 1842
3033_0340	SW_PAD_CTL_PAD_LCD_DATA02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA02)	32	R/W	0000_0014h	8.2.7.204/ 1844
3033_0344	SW_PAD_CTL_PAD_LCD_DATA03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA03)	32	R/W	0000_0014h	8.2.7.205/ 1845
3033_0348	SW_PAD_CTL_PAD_LCD_DATA04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA04)	32	R/W	0000_0014h	8.2.7.206/ 1846
3033_034C	SW_PAD_CTL_PAD_LCD_DATA05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA05)	32	R/W	0000_0014h	8.2.7.207/ 1847
3033_0350	SW_PAD_CTL_PAD_LCD_DATA06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA06)	32	R/W	0000_0014h	8.2.7.208/ 1848
3033_0354	SW_PAD_CTL_PAD_LCD_DATA07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA07)	32	R/W	0000_0014h	8.2.7.209/ 1850
3033_0358	SW_PAD_CTL_PAD_LCD_DATA08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA08)	32	R/W	0000_0014h	8.2.7.210/ 1851
3033_035C	SW_PAD_CTL_PAD_LCD_DATA09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA09)	32	R/W	0000_0014h	8.2.7.211/ 1852
3033_0360	SW_PAD_CTL_PAD_LCD_DATA10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA10)	32	R/W	0000_0014h	8.2.7.212/ 1853
3033_0364	SW_PAD_CTL_PAD_LCD_DATA11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA11)	32	R/W	0000_0014h	8.2.7.213/ 1854
3033_0368	SW_PAD_CTL_PAD_LCD_DATA12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA12)	32	R/W	0000_0014h	8.2.7.214/ 1856
3033_036C	SW_PAD_CTL_PAD_LCD_DATA13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA13)	32	R/W	0000_0014h	8.2.7.215/ 1857
3033_0370	SW_PAD_CTL_PAD_LCD_DATA14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA14)	32	R/W	0000_0014h	8.2.7.216/ 1858
3033_0374	SW_PAD_CTL_PAD_LCD_DATA15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA15)	32	R/W	0000_0014h	8.2.7.217/ 1859
3033_0378	SW_PAD_CTL_PAD_LCD_DATA16 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA16)	32	R/W	0000_0014h	8.2.7.218/ 1860
3033_037C	SW_PAD_CTL_PAD_LCD_DATA17 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA17)	32	R/W	0000_0014h	8.2.7.219/ 1862
3033_0380	SW_PAD_CTL_PAD_LCD_DATA18 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA18)	32	R/W	0000_0014h	8.2.7.220/ 1863

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_0384	SW_PAD_CTL_PAD_LCD_DATA19 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA19)	32	R/W	0000_0014h	8.2.7.221/ 1864
3033_0388	SW_PAD_CTL_PAD_LCD_DATA20 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA20)	32	R/W	0000_0014h	8.2.7.222/ 1865
3033_038C	SW_PAD_CTL_PAD_LCD_DATA21 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA21)	32	R/W	0000_0014h	8.2.7.223/ 1866
3033_0390	SW_PAD_CTL_PAD_LCD_DATA22 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA22)	32	R/W	0000_0014h	8.2.7.224/ 1868
3033_0394	SW_PAD_CTL_PAD_LCD_DATA23 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA23)	32	R/W	0000_0014h	8.2.7.225/ 1869
3033_0398	SW_PAD_CTL_PAD_UART1_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_RX_DATA)	32	R/W	0000_0014h	8.2.7.226/ 1870
3033_039C	SW_PAD_CTL_PAD_UART1_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_TX_DATA)	32	R/W	0000_0014h	8.2.7.227/ 1871
3033_03A0	SW_PAD_CTL_PAD_UART2_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_RX_DATA)	32	R/W	0000_0014h	8.2.7.228/ 1872
3033_03A4	SW_PAD_CTL_PAD_UART2_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_TX_DATA)	32	R/W	0000_0014h	8.2.7.229/ 1874
3033_03A8	SW_PAD_CTL_PAD_UART3_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_RX_DATA)	32	R/W	0000_0014h	8.2.7.230/ 1875
3033_03AC	SW_PAD_CTL_PAD_UART3_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_TX_DATA)	32	R/W	0000_0014h	8.2.7.231/ 1876
3033_03B0	SW_PAD_CTL_PAD_UART3_RTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_RTS_B)	32	R/W	0000_0014h	8.2.7.232/ 1877
3033_03B4	SW_PAD_CTL_PAD_UART3_CTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_CTS_B)	32	R/W	0000_0014h	8.2.7.233/ 1878
3033_03B8	SW_PAD_CTL_PAD_I2C1_SCL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C1_SCL)	32	R/W	0000_0014h	8.2.7.234/ 1880
3033_03BC	SW_PAD_CTL_PAD_I2C1_SDA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C1_SDA)	32	R/W	0000_0014h	8.2.7.235/ 1881
3033_03C0	SW_PAD_CTL_PAD_I2C2_SCL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C2_SCL)	32	R/W	0000_0014h	8.2.7.236/ 1882
3033_03C4	SW_PAD_CTL_PAD_I2C2_SDA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C2_SDA)	32	R/W	0000_0014h	8.2.7.237/ 1883
3033_03C8	SW_PAD_CTL_PAD_I2C3_SCL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C3_SCL)	32	R/W	0000_0014h	8.2.7.238/ 1884
3033_03CC	SW_PAD_CTL_PAD_I2C3_SDA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C3_SDA)	32	R/W	0000_0014h	8.2.7.239/ 1886

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_03D0	SW_PAD_CTL_PAD_I2C4_SCL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C4_SCL)	32	R/W	0000_0014h	8.2.7.240/ 1887
3033_03D4	SW_PAD_CTL_PAD_I2C4_SDA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C4_SDA)	32	R/W	0000_0014h	8.2.7.241/ 1888
3033_03D8	SW_PAD_CTL_PAD_ECSPI1_SCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI1_SCLK)	32	R/W	0000_0014h	8.2.7.242/ 1889
3033_03DC	SW_PAD_CTL_PAD_ECSPI1_MOSI SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI1_MOSI)	32	R/W	0000_0014h	8.2.7.243/ 1890
3033_03E0	SW_PAD_CTL_PAD_ECSPI1_MISO SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI1_MISO)	32	R/W	0000_0014h	8.2.7.244/ 1892
3033_03E4	SW_PAD_CTL_PAD_ECSPI1_SS0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI1_SS0)	32	R/W	0000_0014h	8.2.7.245/ 1893
3033_03E8	SW_PAD_CTL_PAD_ECSPI2_SCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI2_SCLK)	32	R/W	0000_0014h	8.2.7.246/ 1894
3033_03EC	SW_PAD_CTL_PAD_ECSPI2_MOSI SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI2_MOSI)	32	R/W	0000_0014h	8.2.7.247/ 1895
3033_03F0	SW_PAD_CTL_PAD_ECSPI2_MISO SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI2_MISO)	32	R/W	0000_0014h	8.2.7.248/ 1896
3033_03F4	SW_PAD_CTL_PAD_ECSPI2_SS0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI2_SS0)	32	R/W	0000_0014h	8.2.7.249/ 1898
3033_03F8	SW_PAD_CTL_PAD_SD1_CD_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_CD_B)	32	R/W	0000_0014h	8.2.7.250/ 1899
3033_03FC	SW_PAD_CTL_PAD_SD1_WP SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_WP)	32	R/W	0000_0014h	8.2.7.251/ 1900
3033_0400	SW_PAD_CTL_PAD_SD1_RESET_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_RESET_B)	32	R/W	0000_0014h	8.2.7.252/ 1901
3033_0404	SW_PAD_CTL_PAD_SD1_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_CLK)	32	R/W	0000_0014h	8.2.7.253/ 1902
3033_0408	SW_PAD_CTL_PAD_SD1_CMD SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_CMD)	32	R/W	0000_0014h	8.2.7.254/ 1904
3033_040C	SW_PAD_CTL_PAD_SD1_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA0)	32	R/W	0000_0014h	8.2.7.255/ 1905
3033_0410	SW_PAD_CTL_PAD_SD1_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA1)	32	R/W	0000_0014h	8.2.7.256/ 1906
3033_0414	SW_PAD_CTL_PAD_SD1_DATA2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA2)	32	R/W	0000_0014h	8.2.7.257/ 1907
3033_0418	SW_PAD_CTL_PAD_SD1_DATA3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA3)	32	R/W	0000_0014h	8.2.7.258/ 1908
3033_041C	SW_PAD_CTL_PAD_SD2_CD_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_CD_B)	32	R/W	0000_0014h	8.2.7.259/ 1910
3033_0420	SW_PAD_CTL_PAD_SD2_WP SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_WP)	32	R/W	0000_0014h	8.2.7.260/ 1911
3033_0424	SW_PAD_CTL_PAD_SD2_RESET_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_RESET_B)	32	R/W	0000_0014h	8.2.7.261/ 1912

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_0428	SW_PAD_CTL_PAD_SD2_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_CLK)	32	R/W	0000_0014h	8.2.7.262/ 1913
3033_042C	SW_PAD_CTL_PAD_SD2_CMD SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_CMD)	32	R/W	0000_0014h	8.2.7.263/ 1914
3033_0430	SW_PAD_CTL_PAD_SD2_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_DATA0)	32	R/W	0000_0014h	8.2.7.264/ 1916
3033_0434	SW_PAD_CTL_PAD_SD2_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_DATA1)	32	R/W	0000_0014h	8.2.7.265/ 1917
3033_0438	SW_PAD_CTL_PAD_SD2_DATA2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_DATA2)	32	R/W	0000_0014h	8.2.7.266/ 1918
3033_043C	SW_PAD_CTL_PAD_SD2_DATA3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_DATA3)	32	R/W	0000_0014h	8.2.7.267/ 1919
3033_0440	SW_PAD_CTL_PAD_SD3_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_CLK)	32	R/W	0000_0014h	8.2.7.268/ 1920
3033_0444	SW_PAD_CTL_PAD_SD3_CMD SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_CMD)	32	R/W	0000_0014h	8.2.7.269/ 1922
3033_0448	SW_PAD_CTL_PAD_SD3_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA0)	32	R/W	0000_0014h	8.2.7.270/ 1923
3033_044C	SW_PAD_CTL_PAD_SD3_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA1)	32	R/W	0000_0014h	8.2.7.271/ 1924
3033_0450	SW_PAD_CTL_PAD_SD3_DATA2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA2)	32	R/W	0000_0014h	8.2.7.272/ 1925
3033_0454	SW_PAD_CTL_PAD_SD3_DATA3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA3)	32	R/W	0000_0014h	8.2.7.273/ 1926
3033_0458	SW_PAD_CTL_PAD_SD3_DATA4 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA4)	32	R/W	0000_0014h	8.2.7.274/ 1928
3033_045C	SW_PAD_CTL_PAD_SD3_DATA5 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA5)	32	R/W	0000_0014h	8.2.7.275/ 1929
3033_0460	SW_PAD_CTL_PAD_SD3_DATA6 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA6)	32	R/W	0000_0014h	8.2.7.276/ 1930
3033_0464	SW_PAD_CTL_PAD_SD3_DATA7 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA7)	32	R/W	0000_0014h	8.2.7.277/ 1931
3033_0468	SW_PAD_CTL_PAD_SD3_STROBE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_STROBE)	32	R/W	0000_0014h	8.2.7.278/ 1932
3033_046C	SW_PAD_CTL_PAD_SD3_RESET_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_RESET_B)	32	R/W	0000_0014h	8.2.7.279/ 1934
3033_0470	SW_PAD_CTL_PAD_SAI1_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI1_RX_DATA)	32	R/W	0000_0014h	8.2.7.280/ 1935
3033_0474	SW_PAD_CTL_PAD_SAI1_TX_BCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI1_TX_BCLK)	32	R/W	0000_0014h	8.2.7.281/ 1936
3033_0478	SW_PAD_CTL_PAD_SAI1_TX_SYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI1_TX_SYNC)	32	R/W	0000_0014h	8.2.7.282/ 1937
3033_047C	SW_PAD_CTL_PAD_SAI1_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI1_TX_DATA)	32	R/W	0000_0014h	8.2.7.283/ 1938

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_0480	SW_PAD_CTL_PAD_SAI1_RX_SYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI1_RX_SYNC)	32	R/W	0000_0014h	8.2.7.284/ 1940
3033_0484	SW_PAD_CTL_PAD_SAI1_RX_BCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI1_RX_BCLK)	32	R/W	0000_0014h	8.2.7.285/ 1941
3033_0488	SW_PAD_CTL_PAD_SAI1_MCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI1_MCLK)	32	R/W	0000_0014h	8.2.7.286/ 1942
3033_048C	SW_PAD_CTL_PAD_SAI2_TX_SYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI2_TX_SYNC)	32	R/W	0000_0014h	8.2.7.287/ 1943
3033_0490	SW_PAD_CTL_PAD_SAI2_TX_BCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI2_TX_BCLK)	32	R/W	0000_0014h	8.2.7.288/ 1944
3033_0494	SW_PAD_CTL_PAD_SAI2_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI2_RX_DATA)	32	R/W	0000_0014h	8.2.7.289/ 1946
3033_0498	SW_PAD_CTL_PAD_SAI2_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI2_TX_DATA)	32	R/W	0000_0014h	8.2.7.290/ 1947
3033_049C	SW_PAD_CTL_PAD_ENET1_RGMII_RD0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD0)	32	R/W	0000_0014h	8.2.7.291/ 1948
3033_04A0	SW_PAD_CTL_PAD_ENET1_RGMII_RD1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD1)	32	R/W	0000_0014h	8.2.7.292/ 1949
3033_04A4	SW_PAD_CTL_PAD_ENET1_RGMII_RD2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD2)	32	R/W	0000_0014h	8.2.7.293/ 1950
3033_04A8	SW_PAD_CTL_PAD_ENET1_RGMII_RD3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD3)	32	R/W	0000_0014h	8.2.7.294/ 1952
3033_04AC	SW_PAD_CTL_PAD_ENET1_RGMII_RX_CTL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RX_CTL)	32	R/W	0000_0014h	8.2.7.295/ 1953
3033_04B0	SW_PAD_CTL_PAD_ENET1_RGMII_RXC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RXC)	32	R/W	0000_0014h	8.2.7.296/ 1954
3033_04B4	SW_PAD_CTL_PAD_ENET1_RGMII_TD0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD0)	32	R/W	0000_0014h	8.2.7.297/ 1955
3033_04B8	SW_PAD_CTL_PAD_ENET1_RGMII_TD1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD1)	32	R/W	0000_0014h	8.2.7.298/ 1957
3033_04BC	SW_PAD_CTL_PAD_ENET1_RGMII_TD2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD2)	32	R/W	0000_0014h	8.2.7.299/ 1958
3033_04C0	SW_PAD_CTL_PAD_ENET1_RGMII_TD3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD3)	32	R/W	0000_0014h	8.2.7.300/ 1959

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_04C4	SW_PAD_CTL_PAD_ENET1_RGMII_TX_CTL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TX_CTL)	32	R/W	0000_0014h	8.2.7.301/ 1960
3033_04C8	SW_PAD_CTL_PAD_ENET1_RGMII_TXC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TXC)	32	R/W	0000_0014h	8.2.7.302/ 1962
3033_04CC	SW_PAD_CTL_PAD_ENET1_TX_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_CLK)	32	R/W	0000_0014h	8.2.7.303/ 1963
3033_04D0	SW_PAD_CTL_PAD_ENET1_RX_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_CLK)	32	R/W	0000_0014h	8.2.7.304/ 1964
3033_04D4	SW_PAD_CTL_PAD_ENET1_CRS SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_CRS)	32	R/W	0000_0014h	8.2.7.305/ 1965
3033_04D8	SW_PAD_CTL_PAD_ENET1_COL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_COL)	32	R/W	0000_0014h	8.2.7.306/ 1966
3033_04DC	FLEXCAN1_RX_SELECT_INPUT DAISY Register (IOMUXC_FLEXCAN1_RX_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.307/ 1967
3033_04E0	FLEXCAN2_RX_SELECT_INPUT DAISY Register (IOMUXC_FLEXCAN2_RX_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.308/ 1968
3033_04E4	CCM_EXT_CLK_1_SELECT_INPUT DAISY Register (IOMUXC_CCM_EXT_CLK_1_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.309/ 1969
3033_04E8	CCM_EXT_CLK_2_SELECT_INPUT DAISY Register (IOMUXC_CCM_EXT_CLK_2_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.310/ 1969
3033_04EC	CCM_EXT_CLK_3_SELECT_INPUT DAISY Register (IOMUXC_CCM_EXT_CLK_3_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.311/ 1970
3033_04F0	CCM_EXT_CLK_4_SELECT_INPUT DAISY Register (IOMUXC_CCM_EXT_CLK_4_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.312/ 1971
3033_04F4	CCM_PMIC_READY_SELECT_INPUT DAISY Register (IOMUXC_CCM_PMIC_READY_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.313/ 1971
3033_04F8	CSI_DATA2_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA2_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.314/ 1972
3033_04FC	CSI_DATA3_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA3_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.315/ 1973
3033_0500	CSI_DATA4_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA4_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.316/ 1974
3033_0504	CSI_DATA5_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA5_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.317/ 1975
3033_0508	CSI_DATA6_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA6_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.318/ 1976
3033_050C	CSI_DATA7_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA7_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.319/ 1977
3033_0510	CSI_DATA8_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA8_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.320/ 1978
3033_0514	CSI_DATA9_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA9_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.321/ 1979

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_0518	CSI_HSYNC_SELECT_INPUT DAISY Register (IOMUXC_CSI_HSYNC_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.322/ 1980
3033_051C	CSI_PIXCLK_SELECT_INPUT DAISY Register (IOMUXC_CSI_PIXCLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.323/ 1981
3033_0520	CSI_VSYNC_SELECT_INPUT DAISY Register (IOMUXC_CSI_VSYNC_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.324/ 1982
3033_0524	ECSPI1_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_SCLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.325/ 1983
3033_0528	ECSPI1_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_MISO_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.326/ 1984
3033_052C	ECSPI1_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_MOSI_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.327/ 1985
3033_0530	ECSPI1_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_SS0_B_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.328/ 1986
3033_0534	ECSPI2_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_SCLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.329/ 1987
3033_0538	ECSPI2_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_MISO_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.330/ 1988
3033_053C	ECSPI2_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_MOSI_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.331/ 1989
3033_0540	ECSPI2_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_SS0_B_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.332/ 1990
3033_0544	ECSPI3_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_SCLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.333/ 1991
3033_0548	ECSPI3_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_MISO_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.334/ 1992
3033_054C	ECSPI3_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_MOSI_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.335/ 1993
3033_0550	ECSPI3_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_SS0_B_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.336/ 1994
3033_0554	ECSPI4_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_SCLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.337/ 1994
3033_0558	ECSPI4_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_MISO_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.338/ 1995
3033_055C	ECSPI4_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_MOSI_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.339/ 1996
3033_0560	ECSPI4_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_SS0_B_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.340/ 1996
3033_0564	CCM_ENET1_REF_CLK_SELECT_INPUT DAISY Register (IOMUXC_CCM_ENET1_REF_CLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.341/ 1997
3033_0568	ENET1_MDIO_SELECT_INPUT DAISY Register (IOMUXC_ENET1_MDIO_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.342/ 1998
3033_056C	ENET1_RX_CLK_SELECT_INPUT DAISY Register (IOMUXC_ENET1_RX_CLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.343/ 1999

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_0570	CCM_ENET2_REF_CLK_SELECT_INPUT DAISY Register (IOMUXC_CCM_ENET2_REF_CLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.344/ 1999
3033_0574	ENET2_MDIO_SELECT_INPUT DAISY Register (IOMUXC_ENET2_MDIO_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.345/ 2000
3033_0578	ENET2_RX_CLK_SELECT_INPUT DAISY Register (IOMUXC_ENET2_RX_CLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.346/ 2001
3033_057C	EPDC_PWR_IRQ_SELECT_INPUT DAISY Register (IOMUXC_EPDC_PWR_IRQ_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.347/ 2002
3033_0580	EPDC_PWR_STAT_SELECT_INPUT DAISY Register (IOMUXC_EPDC_PWR_STAT_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.348/ 2003
3033_0584	FLEXTIMER1_CH0_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH0_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.349/ 2004
3033_0588	FLEXTIMER1_CH1_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH1_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.350/ 2005
3033_058C	FLEXTIMER1_CH2_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH2_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.351/ 2006
3033_0590	FLEXTIMER1_CH3_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH3_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.352/ 2007
3033_0594	FLEXTIMER1_CH4_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH4_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.353/ 2008
3033_0598	FLEXTIMER1_CH5_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH5_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.354/ 2009
3033_059C	FLEXTIMER1_CH6_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH6_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.355/ 2010
3033_05A0	FLEXTIMER1_CH7_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH7_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.356/ 2011
3033_05A4	FLEXTIMER1_PHA_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_PHA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.357/ 2012
3033_05A8	FLEXTIMER1_PHB_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_PHB_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.358/ 2013
3033_05AC	FLEXTIMER2_CH0_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH0_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.359/ 2014
3033_05B0	FLEXTIMER2_CH1_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH1_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.360/ 2015
3033_05B4	FLEXTIMER2_CH2_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH2_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.361/ 2016
3033_05B8	FLEXTIMER2_CH3_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH3_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.362/ 2017
3033_05BC	FLEXTIMER2_CH4_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH4_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.363/ 2018
3033_05C0	FLEXTIMER2_CH5_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH5_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.364/ 2019
3033_05C4	FLEXTIMER2_CH6_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH6_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.365/ 2020

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_05C8	FLEXTIMER2_CH7_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH7_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.366/ 2021
3033_05CC	FLEXTIMER2_PHA_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_PHA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.367/ 2022
3033_05D0	FLEXTIMER2_PHB_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_PHB_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.368/ 2023
3033_05D4	I2C1_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C1_SCL_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.369/ 2023
3033_05D8	I2C1_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C1_SDA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.370/ 2024
3033_05DC	I2C2_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C2_SCL_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.371/ 2025
3033_05E0	I2C2_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C2_SDA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.372/ 2025
3033_05E4	I2C3_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C3_SCL_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.373/ 2026
3033_05E8	I2C3_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C3_SDA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.374/ 2027
3033_05EC	I2C4_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C4_SCL_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.375/ 2027
3033_05F0	I2C4_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C4_SDA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.376/ 2028
3033_05F4	KPP_COL0_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL0_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.377/ 2029
3033_05F8	KPP_COL1_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL1_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.378/ 2030
3033_05FC	KPP_COL2_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL2_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.379/ 2031
3033_0600	KPP_COL3_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL3_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.380/ 2032
3033_0604	KPP_COL4_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL4_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.381/ 2033
3033_0608	KPP_COL5_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL5_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.382/ 2034
3033_060C	KPP_COL6_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL6_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.383/ 2035
3033_0610	KPP_COL7_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL7_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.384/ 2036
3033_0614	KPP_ROW0_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW0_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.385/ 2037
3033_0618	KPP_ROW1_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW1_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.386/ 2038
3033_061C	KPP_ROW2_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW2_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.387/ 2039

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_0620	KPP_ROW3_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW3_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.388/ 2040
3033_0624	KPP_ROW4_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW4_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.389/ 2041
3033_0628	KPP_ROW5_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW5_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.390/ 2042
3033_062C	KPP_ROW6_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW6_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.391/ 2043
3033_0630	KPP_ROW7_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW7_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.392/ 2044
3033_0634	LCD_BUSY_SELECT_INPUT DAISY Register (IOMUXC_LCD_BUSY_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.393/ 2045
3033_0638	LCD_DATA00_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA00_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.394/ 2045
3033_063C	LCD_DATA01_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA01_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.395/ 2046
3033_0640	LCD_DATA02_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA02_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.396/ 2047
3033_0644	LCD_DATA03_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA03_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.397/ 2047
3033_0648	LCD_DATA04_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA04_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.398/ 2048
3033_064C	LCD_DATA05_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA05_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.399/ 2049
3033_0650	LCD_DATA06_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA06_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.400/ 2049
3033_0654	LCD_DATA07_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA07_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.401/ 2050
3033_0658	LCD_DATA08_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA08_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.402/ 2051
3033_065C	LCD_DATA09_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA09_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.403/ 2051
3033_0660	LCD_DATA10_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA10_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.404/ 2052
3033_0664	LCD_DATA11_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA11_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.405/ 2053
3033_0668	LCD_DATA12_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA12_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.406/ 2053
3033_066C	LCD_DATA13_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA13_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.407/ 2054
3033_0670	LCD_DATA14_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA14_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.408/ 2055
3033_0674	LCD_DATA15_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA15_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.409/ 2055

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_0678	LCD_DATA16_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA16_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.410/ 2056
3033_067C	LCD_DATA17_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA17_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.411/ 2057
3033_0680	LCD_DATA18_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA18_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.412/ 2057
3033_0684	LCD_DATA19_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA19_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.413/ 2058
3033_0688	LCD_DATA20_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA20_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.414/ 2059
3033_068C	LCD_DATA21_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA21_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.415/ 2059
3033_0690	LCD_DATA22_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA22_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.416/ 2060
3033_0694	LCD_DATA23_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA23_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.417/ 2061
3033_0698	LCD_VSYNC_SELECT_INPUT DAISY Register (IOMUXC_LCD_VSYNC_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.418/ 2061
3033_069C	SAI1_RX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_BCLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.419/ 2062
3033_06A0	SAI1_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.420/ 2063
3033_06A4	SAI1_RX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_SYNC_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.421/ 2064
3033_06A8	SAI1_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI1_TX_BCLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.422/ 2065
3033_06AC	SAI1_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI1_TX_SYNC_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.423/ 2066
3033_06B0	SAI2_RX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_BCLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.424/ 2067
3033_06B4	SAI2_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.425/ 2068
3033_06B8	SAI2_RX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_SYNC_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.426/ 2069
3033_06BC	SAI2_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI2_TX_BCLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.427/ 2070
3033_06C0	SAI2_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI2_TX_SYNC_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.428/ 2071
3033_06C4	SAI3_RX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI3_RX_BCLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.429/ 2071
3033_06C8	SAI3_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI3_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.430/ 2072
3033_06CC	SAI3_RX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI3_RX_SYNC_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.431/ 2073

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_06D0	SAI3_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI3_TX_BCLK_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.432/ 2073
3033_06D4	SAI3_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI3_TX_SYNC_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.433/ 2074
3033_06D8	SDMA_EVENTS0_SELECT_INPUT DAISY Register (IOMUXC_SDMA_EVENTS0_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.434/ 2075
3033_06DC	SDMA_EVENTS1_SELECT_INPUT DAISY Register (IOMUXC_SDMA_EVENTS1_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.435/ 2075
3033_06E0	SIM1_PORT1_PD_SELECT_INPUT DAISY Register (IOMUXC_SIM1_PORT1_PD_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.436/ 2076
3033_06E4	SIM1_PORT1_TRXD_SELECT_INPUT DAISY Register (IOMUXC_SIM1_PORT1_TRXD_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.437/ 2077
3033_06E8	SIM2_PORT1_PD_SELECT_INPUT DAISY Register (IOMUXC_SIM2_PORT1_PD_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.438/ 2078
3033_06EC	SIM2_PORT1_TRXD_SELECT_INPUT DAISY Register (IOMUXC_SIM2_PORT1_TRXD_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.439/ 2079
3033_06F0	UART1_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART1_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.440/ 2079
3033_06F4	UART1_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART1_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.441/ 2080
3033_06F8	UART2_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART2_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.442/ 2081
3033_06FC	UART2_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART2_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.443/ 2081
3033_0700	UART3_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART3_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.444/ 2082
3033_0704	UART3_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART3_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.445/ 2083
3033_0708	UART4_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART4_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.446/ 2083
3033_070C	UART4_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART4_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.447/ 2084
3033_0710	UART5_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART5_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.448/ 2084
3033_0714	UART5_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART5_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.449/ 2085
3033_0718	UART6_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART6_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.450/ 2086
3033_071C	UART6_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART6_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.451/ 2086
3033_0720	UART7_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART7_RTS_B_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.452/ 2087
3033_0724	UART7_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART7_RX_DATA_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.453/ 2087

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3033_0728	USB_OTG2_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG2_OC_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.454/ 2088
3033_072C	USB_OTG1_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG1_OC_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.455/ 2089
3033_0730	USB_OTG2_ID_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG2_ID_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.456/ 2089
3033_0734	USB_OTG1_ID_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG1_ID_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.457/ 2090
3033_0738	SD3_CD_B_SELECT_INPUT DAISY Register (IOMUXC_SD3_CD_B_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.458/ 2091
3033_073C	SD3_WP_SELECT_INPUT DAISY Register (IOMUXC_SD3_WP_SELECT_INPUT)	32	R/W	0000_0000h	8.2.7.459/ 2091

8.2.7.1 SW_MUX_CTL_PAD_GPIO1_IO08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO08)

SW_MUX_CTL Register

Address: 3033_0000h base + 14h offset = 3033_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

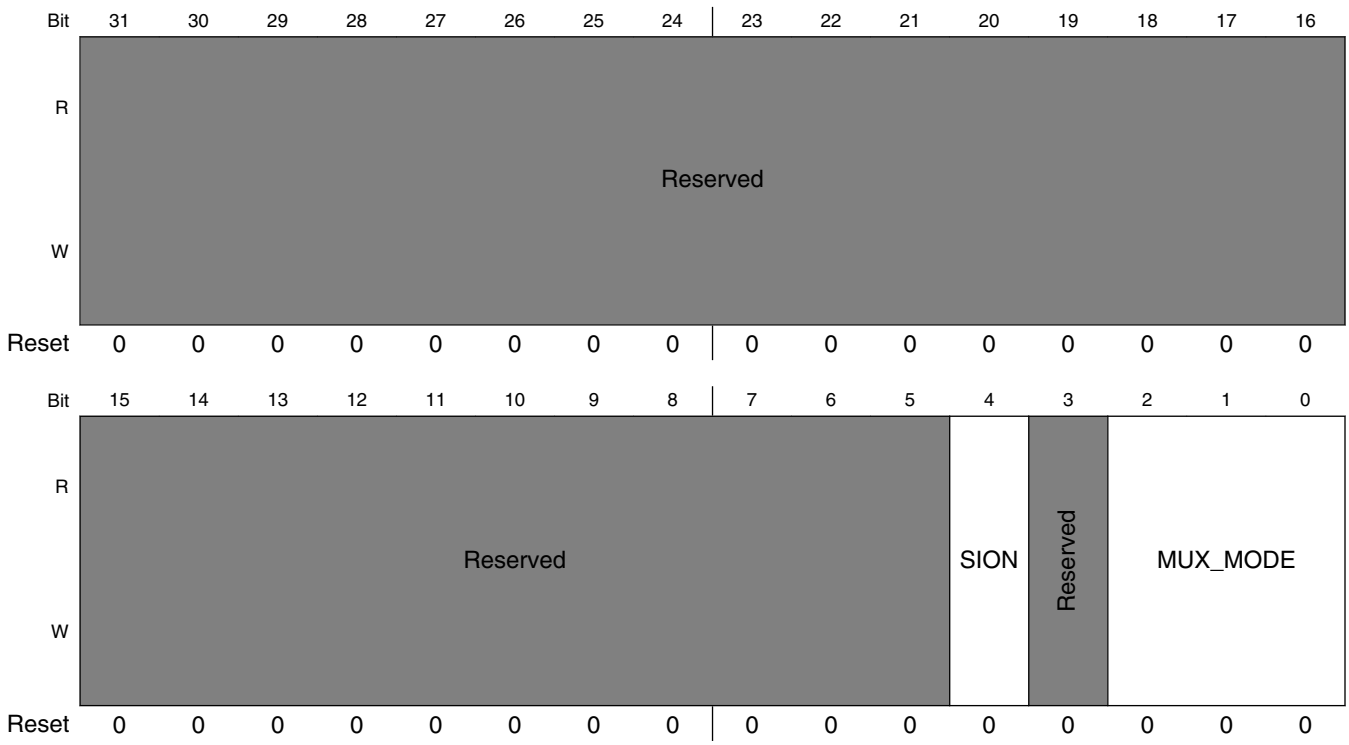
IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO08 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO08 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO1_IO08. 000 **ALT0_GPIO1_IO8** — Select mux mode: ALT0 mux port: IO8 of instance: GPIO1 001 **ALT1_SD1_VSELECT** — Select mux mode: ALT1 mux port: VSELECT of instance: SD1 010 **ALT2_WDOG1_WDOG_B** — Select mux mode: ALT2 mux port: WDOG_B of instance: WDOG1 011 **ALT3_UART3_RX_DATA** — Select mux mode: ALT3 mux port: RX_DATA of instance: UART3 100 **ALT4_I2C3_SCL** — Select mux mode: ALT4 mux port: SCL of instance: I2C3 110 **ALT6_KPP_COL5** — Select mux mode: ALT6 mux port: COL5 of instance: KPP 111 **ALT7_PWM1_OUT** — Select mux mode: ALT7 mux port: OUT of instance: PWM1

8.2.7.2 SW_MUX_CTL_PAD_GPIO1_IO09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO09)

SW_MUX_CTL Register

Address: 3033_0000h base + 18h offset = 3033_0018h



IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO09 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO09 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO1_IO09. 000 **ALT0_GPIO1_IO9** — Select mux mode: ALT0 mux port: IO9 of instance: GPIO1 001 **ALT1_SD1_LCTL** — Select mux mode: ALT1 mux port: LCTL of instance: SD1

Table continues on the next page...

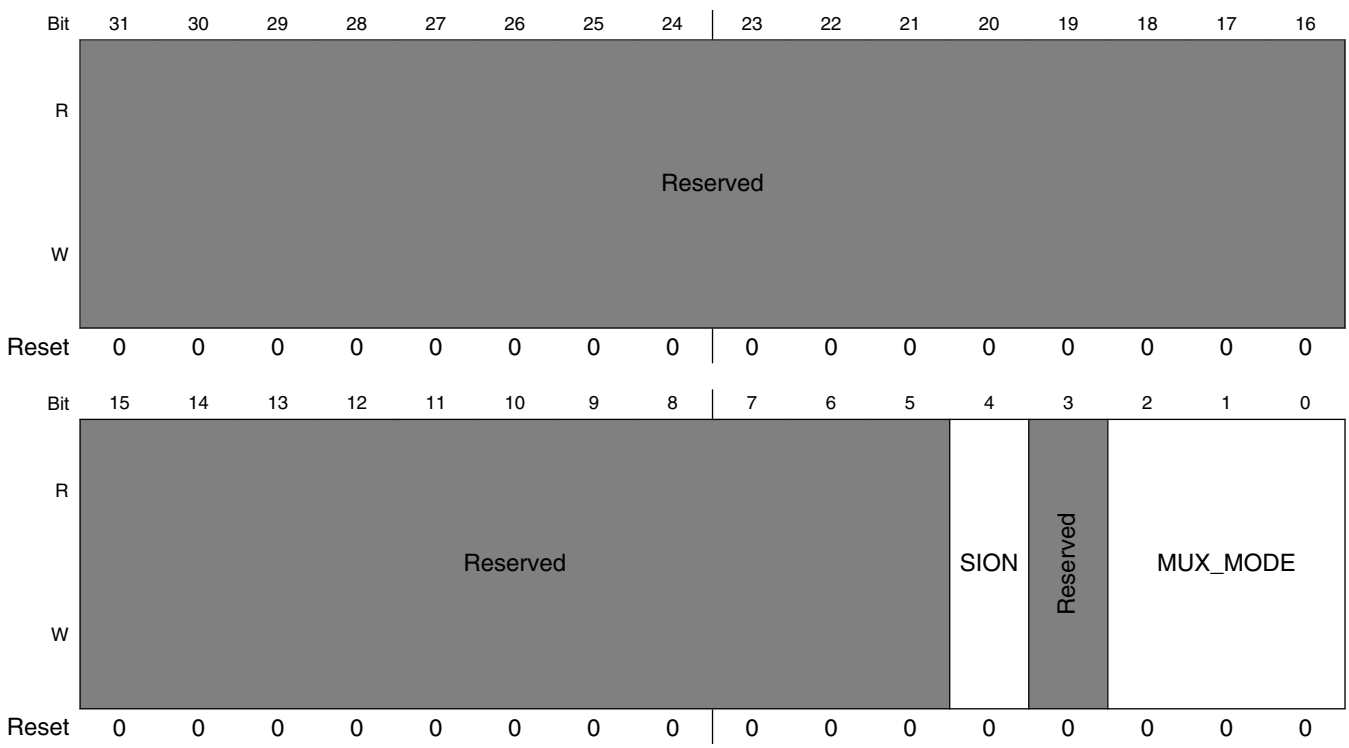
IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO09 field descriptions (continued)

Field	Description
	010 **ALT2_CCM_ENET_PHY_REF_CLK** — Select mux mode: ALT2 mux port: ENET_PHY_REF_CLK of instance: CCM 011 **ALT3_UART3_TX_DATA** — Select mux mode: ALT3 mux port: TX_DATA of instance: UART3 100 **ALT4_I2C3_SDA** — Select mux mode: ALT4 mux port: SDA of instance: I2C3 101 **ALT5_CCM_PMIC_READY** — Select mux mode: ALT5 mux port: CCM_PMIC_READY of instance: CCM 110 **ALT6_KPP_ROW5** — Select mux mode: ALT6 mux port: ROW5 of instance: KPP 111 **ALT7_PWM2_OUT** — Select mux mode: ALT7 mux port: OUT of instance: PWM2

8.2.7.3 SW_MUX_CTL_PAD_GPIO1_IO10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO10)

SW_MUX_CTL Register

Address: 3033_0000h base + 1Ch offset = 3033_001Ch



IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO10 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field.

Table continues on the next page...

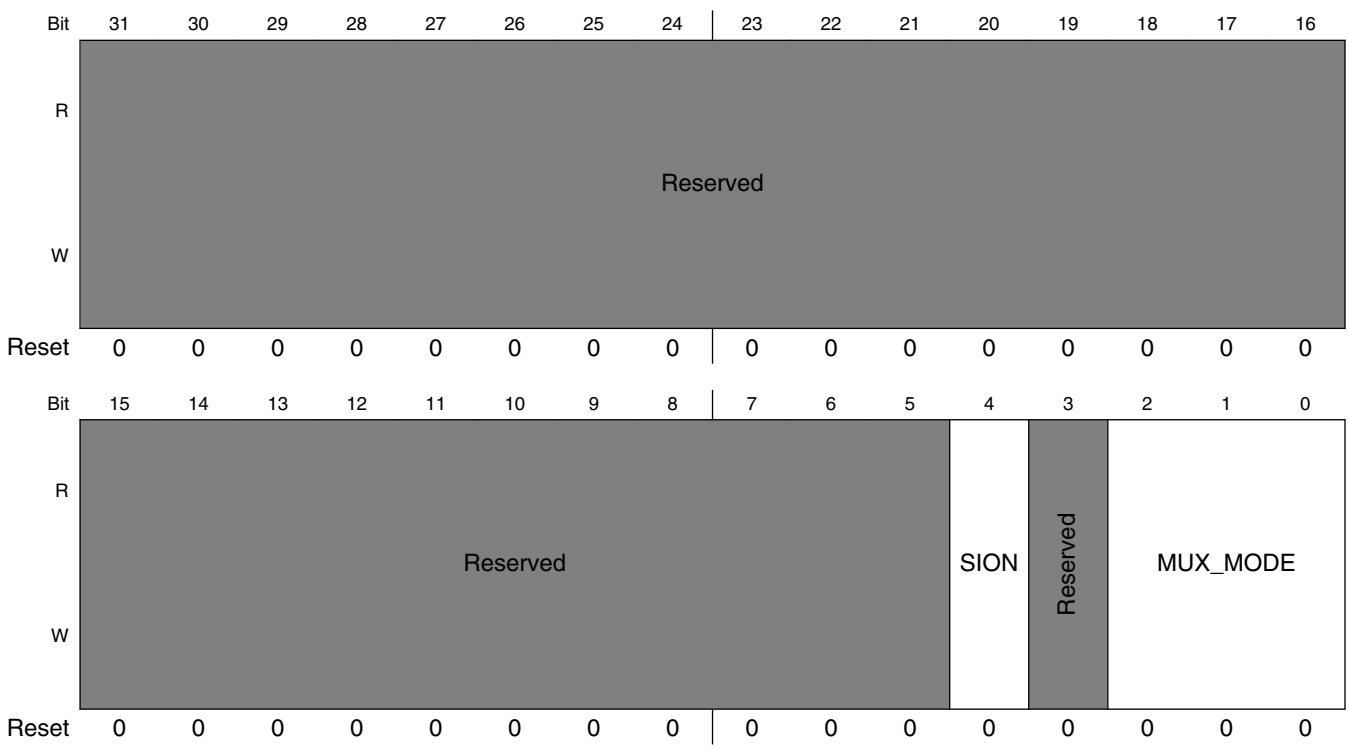
IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO10 field descriptions (continued)

Field	Description
	Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO10 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO1_IO10. 000 **ALT0_GPIO1_IO10** — Select mux mode: ALT0 mux port: IO10 of instance: GPIO1 001 **ALT1_SD2_LCTL** — Select mux mode: ALT1 mux port: LCTL of instance: SD2 010 **ALT2_ENET1_MDIO** — Select mux mode: ALT2 mux port: MDIO of instance: ENET1 011 **ALT3_UART3_RTS_B** — Select mux mode: ALT3 mux port: RTS_B of instance: UART3 100 **ALT4_I2C4_SCL** — Select mux mode: ALT4 mux port: SCL of instance: I2C4 101 **ALT5_FLEXTIMER1_PHA** — Select mux mode: ALT5 mux port: PHA of instance: FLEXTIMER1 110 **ALT6_KPP_COL6** — Select mux mode: ALT6 mux port: COL6 of instance: KPP 111 **ALT7_PWM3_OUT** — Select mux mode: ALT7 mux port: OUT of instance: PWM3

8.2.7.4 SW_MUX_CTL_PAD_GPIO1_IO11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO11)

SW_MUX_CTL Register

Address: 3033_0000h base + 20h offset = 3033_0020h



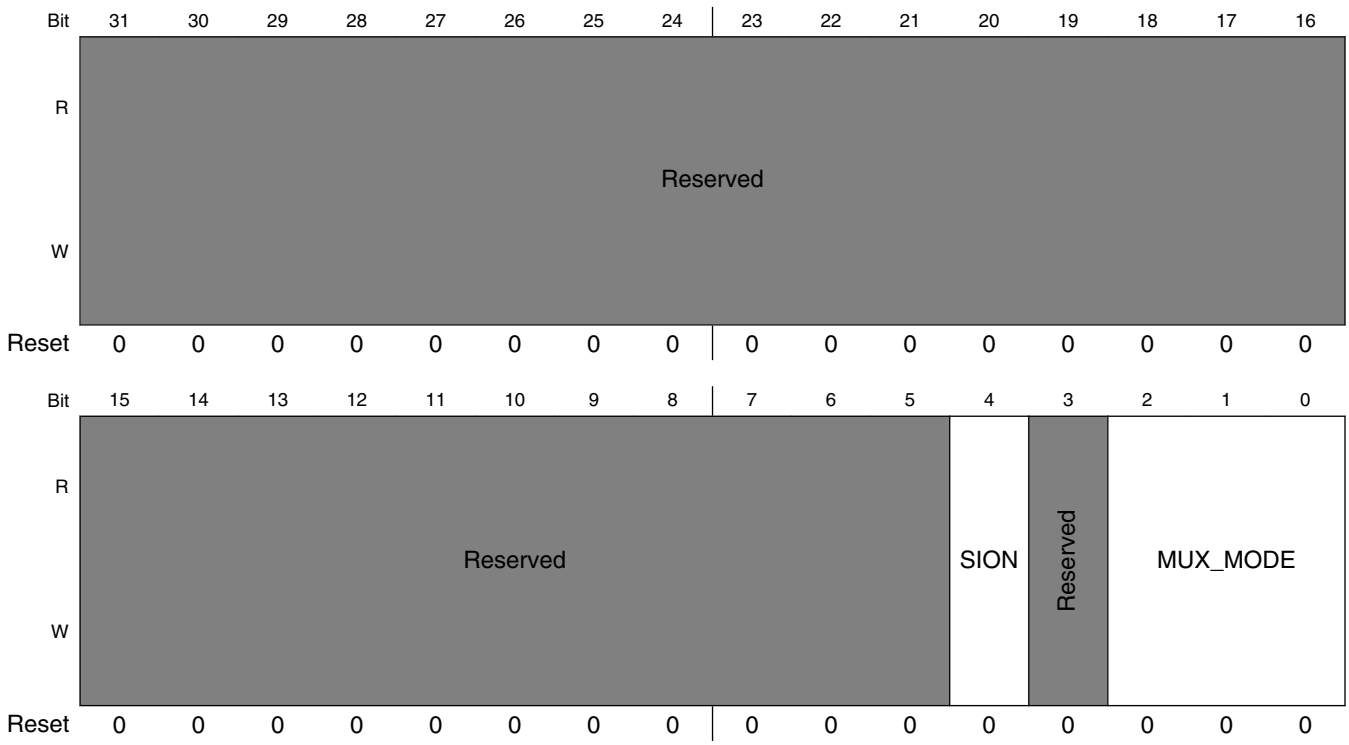
IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO11 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO11 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO1_IO11. 000 **ALT0_GPIO1_IO11** — Select mux mode: ALT0 mux port: IO11 of instance: GPIO1 001 **ALT1_SD3_LCTL** — Select mux mode: ALT1 mux port: LCTL of instance: SD3 010 **ALT2_ENET1_MDC** — Select mux mode: ALT2 mux port: MDC of instance: ENET1 011 **ALT3_UART3_CTS_B** — Select mux mode: ALT3 mux port: CTS_B of instance: UART3 100 **ALT4_I2C4_SDA** — Select mux mode: ALT4 mux port: SDA of instance: I2C4 101 **ALT5_FLEXTIMER1_PHB** — Select mux mode: ALT5 mux port: PHB of instance: FLEXTIMER1 110 **ALT6_KPP_ROW6** — Select mux mode: ALT6 mux port: ROW6 of instance: KPP 111 **ALT7_PWM4_OUT** — Select mux mode: ALT7 mux port: OUT of instance: PWM4

8.2.7.5 SW_MUX_CTL_PAD_GPIO1_IO12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO12)

SW_MUX_CTL Register

Address: 3033_0000h base + 24h offset = 3033_0024h



IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO12 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO12 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO1_IO12. 000 **ALT0_GPIO1_IO12** — Select mux mode: ALT0 mux port: IO12 of instance: GPIO1 001 **ALT1_SD2_VSELECT** — Select mux mode: ALT1 mux port: VSELECT of instance: SD2

Table continues on the next page...

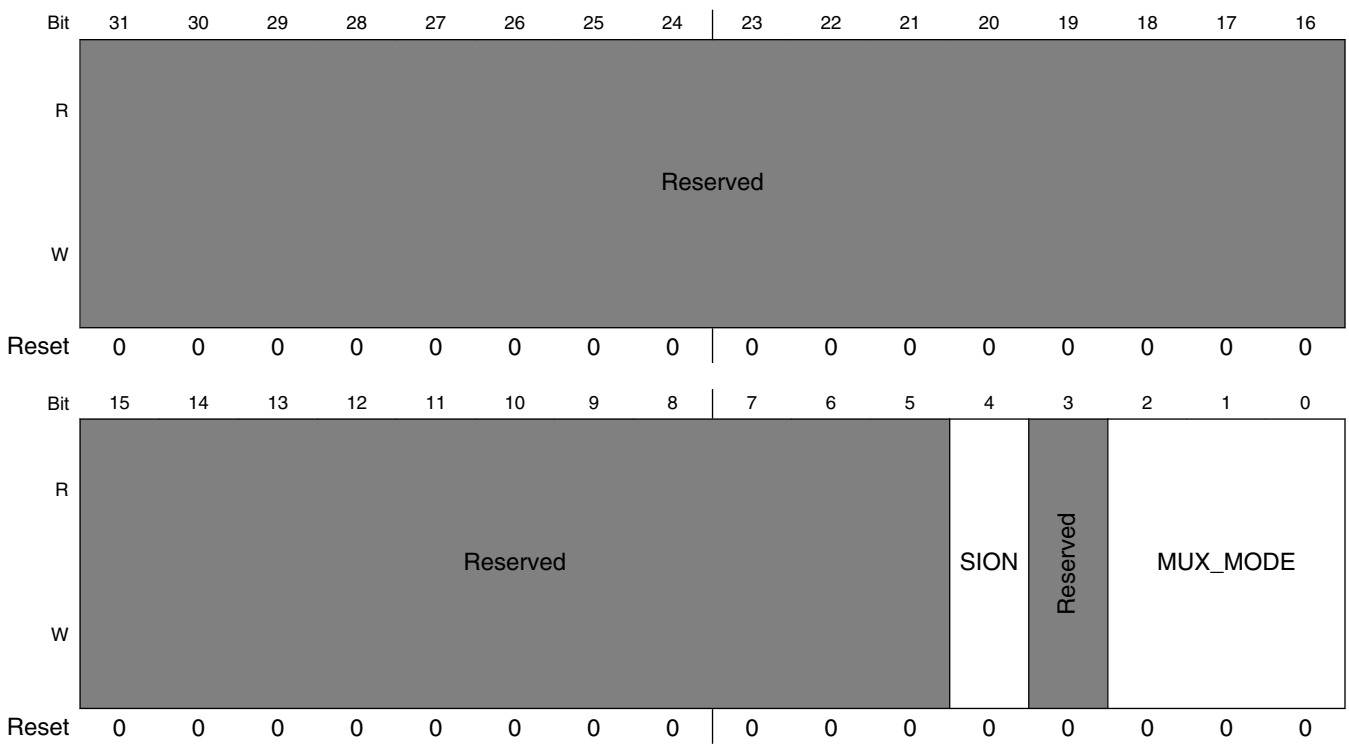
IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO12 field descriptions (continued)

Field	Description
	010 **ALT2_CCM_ENET1_REF_CLK** — Select mux mode: ALT2 mux port: ENET1_REF_CLK of instance: ENET1 011 **ALT3_FLEXCAN1_RX** — Select mux mode: ALT3 mux port: RX of instance: FLEXCAN1 100 **ALT4_CM4_NMI** — Select mux mode: ALT4 mux port: NMI of instance: CM4 101 **ALT5_CCM_EXT_CLK1** — Select mux mode: ALT5 mux port: EXT_CLK1 of instance: CCM 110 **ALT6_SNVS_VIO_5** — Select mux mode: ALT6 mux port: VIO_5 of instance: SNVS 111 **ALT7_USB_OTG1_ID** — Select mux mode: ALT7 mux port: OTG1_ID of instance: USB

8.2.7.6 SW_MUX_CTL_PAD_GPIO1_IO13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO13)

SW_MUX_CTL Register

Address: 3033_0000h base + 28h offset = 3033_0028h



IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO13 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

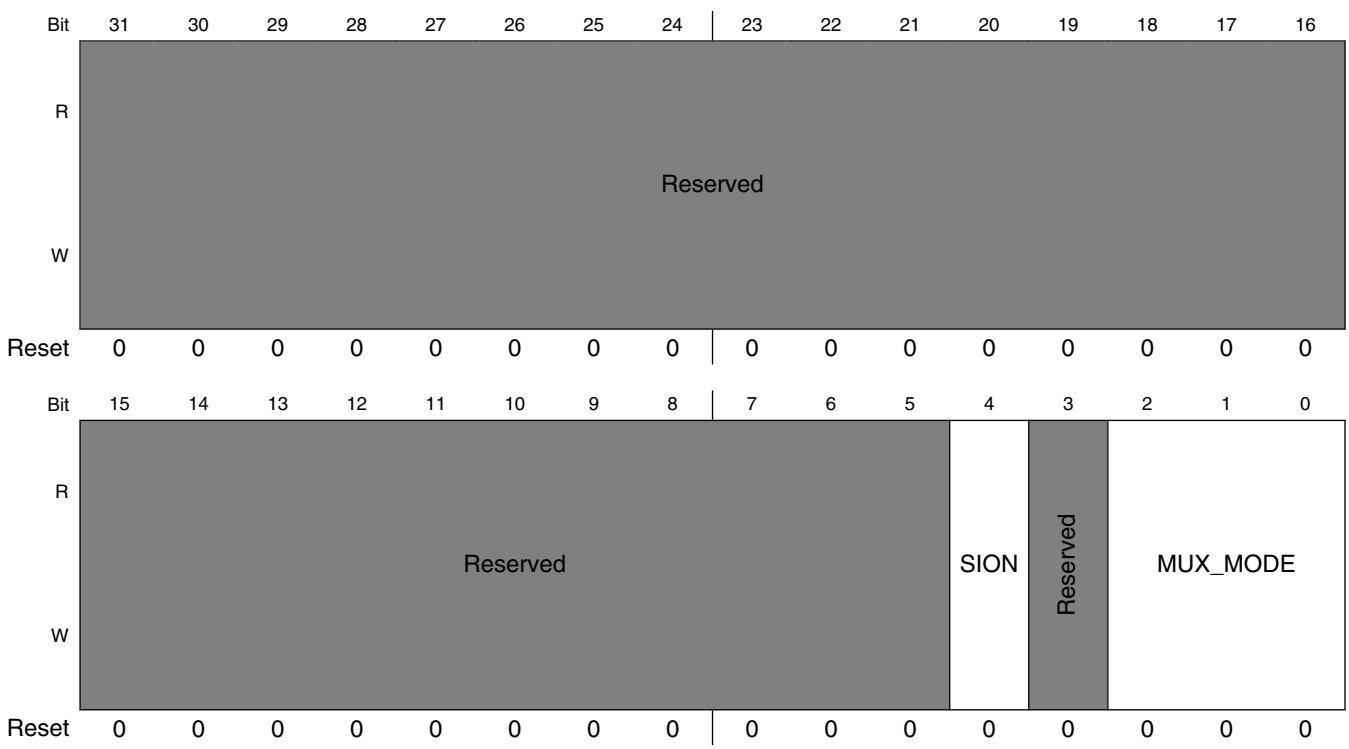
IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO13 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad GPIO1_IO13 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO1_IO13. 000 **ALT0_GPIO1_IO13** — Select mux mode: ALT0 mux port: IO13 of instance: GPIO1 001 **ALT1_SD3_VSELECT** — Select mux mode: ALT1 mux port: VSELECT of instance: SD3 010 **ALT2_CCM_ENET2_REF_CLK** — Select mux mode: ALT2 mux port: ENET2_REF_CLK of instance: ENET2 011 **ALT3_FLEXCAN1_TX** — Select mux mode: ALT3 mux port: TX of instance: FLEXCAN1 100 **ALT4_CCM_PMIC_READY** — Select mux mode: ALT4 mux port: CCM_PMIC_READY of instance: CCM 101 **ALT5_CCM_EXT_CLK2** — Select mux mode: ALT5 mux port: EXT_CLK2 of instance: CCM 110 **ALT6_SNVS_VIO_5_CTL** — Select mux mode: ALT6 mux port: VIO_5_CTL of instance: SNVS 111 **ALT7_USB_OTG2_ID** — Select mux mode: ALT7 mux port: OTG2_ID of instance: USB

8.2.7.7 SW_MUX_CTL_PAD_GPIO1_IO14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO14)

SW_MUX_CTL Register

Address: 3033_0000h base + 2Ch offset = 3033_002Ch



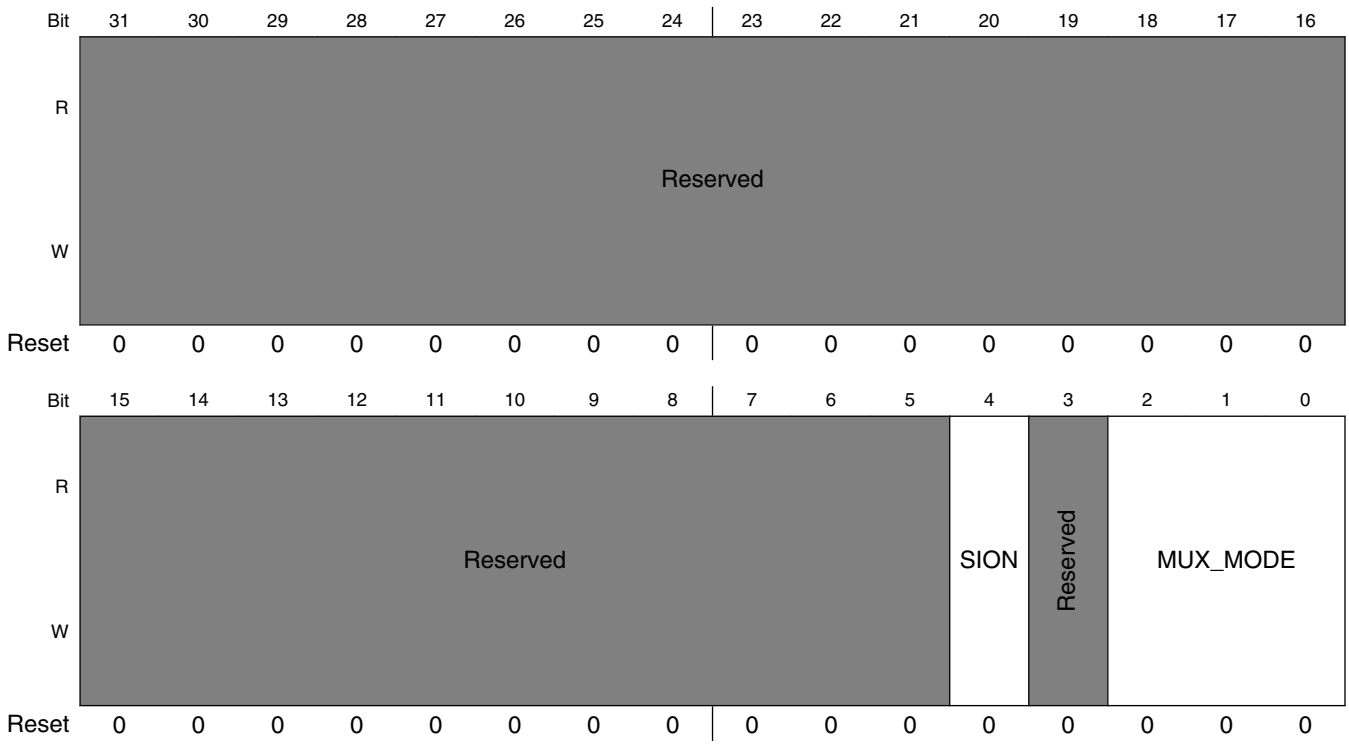
IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO14 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO14 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: GPIO1_IO14. 000 **ALT0_GPIO1_IO14** — Select mux mode: ALT0 mux port: IO14 of instance: GPIO1 001 **ALT1_SD3_CD_B** — Select mux mode: ALT1 mux port: CD_B of instance: SD3 010 **ALT2_ENET2_MDIO** — Select mux mode: ALT2 mux port: MDIO of instance: ENET2 011 **ALT3_FLEXCAN2_RX** — Select mux mode: ALT3 mux port: RX of instance: FLEXCAN2 100 **ALT4_WDOG3_WDOG_B** — Select mux mode: ALT4 mux port: WDOG_B of instance: WDOG3 101 **ALT5_CCM_EXT_CLK3** — Select mux mode: ALT5 mux port: EXT_CLK3 of instance: CCM 110 **ALT6_SDMA_EXT_EVENT0** — Select mux mode: ALT6 mux port: EXT_EVENT0 of instance: SDMA

8.2.7.8 SW_MUX_CTL_PAD_GPIO1_IO15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO15)

SW_MUX_CTL Register

Address: 3033_0000h base + 30h offset = 3033_0030h



IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO15 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO1_IO15 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: GPIO1_IO15. 000 **ALT0_GPIO1_IO15** — Select mux mode: ALT0 mux port: IO15 of instance: GPIO1 001 **ALT1_SD3_WP** — Select mux mode: ALT1 mux port: WP of instance: SD3 010 **ALT2_ENET2_MDC** — Select mux mode: ALT2 mux port: MDC of instance: ENET2

Table continues on the next page...

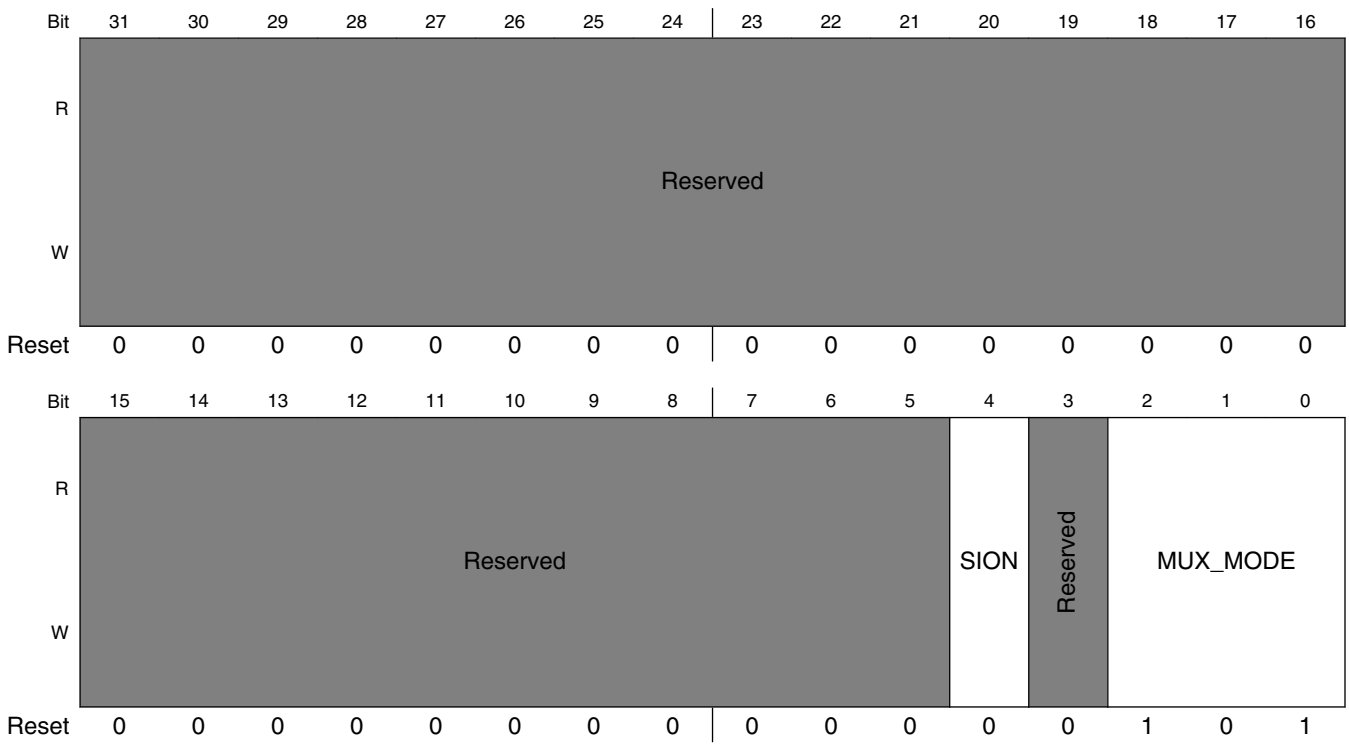
IOMUXC_SW_MUX_CTL_PAD_GPIO1_IO15 field descriptions (continued)

Field	Description
	011 **ALT3_FLEXCAN2_TX** — Select mux mode: ALT3 mux port: TX of instance: FLEXCAN2 100 **ALT4_WDOG4_WDOG_B** — Select mux mode: ALT4 mux port: WDOG_B of instance: WDOG4 101 **ALT5_CCM_EXT_CLK4** — Select mux mode: ALT5 mux port: EXT_CLK4 of instance: CCM 110 **ALT6_SDMA_EXT_EVENT1** — Select mux mode: ALT6 mux port: EXT_EVENT1 of instance: SDMA

8.2.7.9 SW_MUX_CTL_PAD_EPDC_DATA00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA00)

SW_MUX_CTL Register

Address: 3033_0000h base + 34h offset = 3033_0034h



IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA00 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

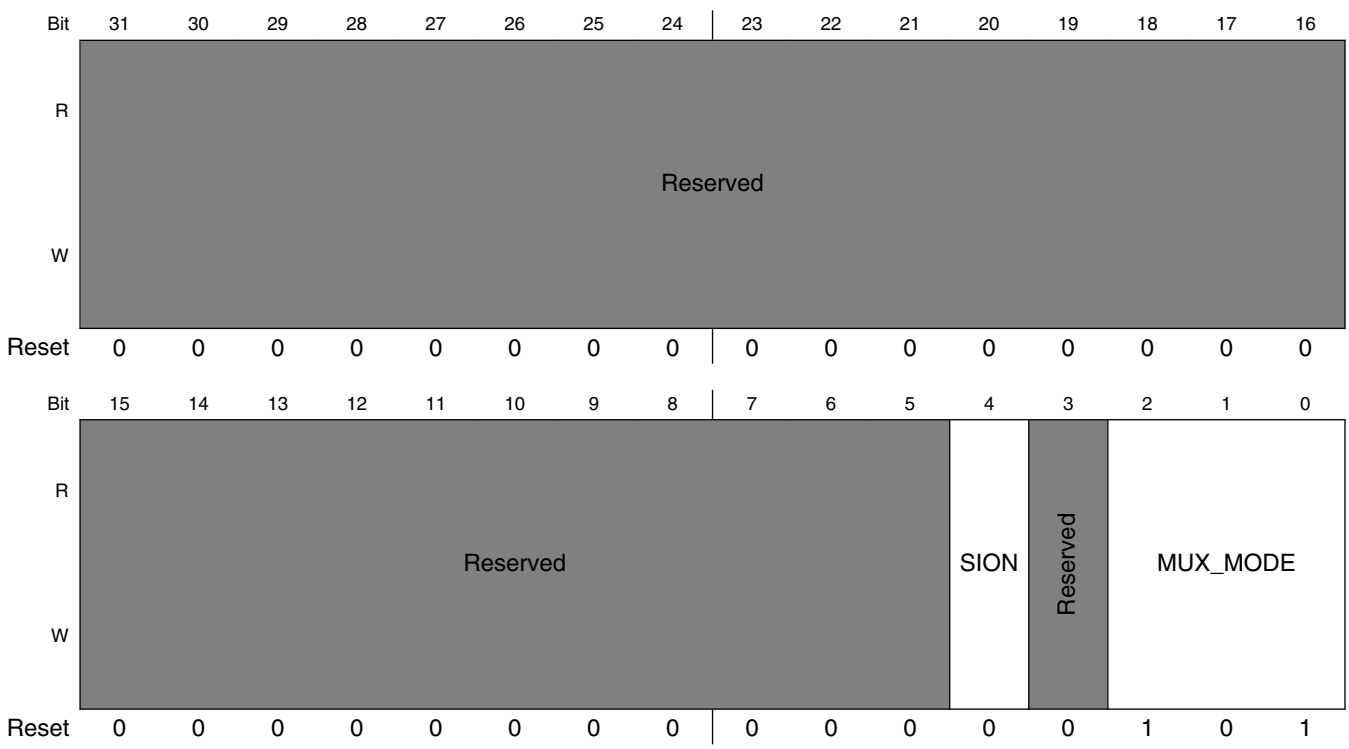
IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA00 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad EPDC_DATA00 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_DATA00. 000 **ALT0_EPDC_DATA0** — Select mux mode: ALT0 mux port: DATA0 of instance: EPDC 001 **ALT1_SIM1_PORT2_TRXD** — Select mux mode: ALT1 mux port: PORT2_TRXD of instance: SIM1 010 **ALT2_QSPI_A_DATA0** — Select mux mode: ALT2 mux port: A_DATA0 of instance: QSPI 011 **ALT3_KPP_ROW3** — Select mux mode: ALT3 mux port: ROW3 of instance: KPP 100 **ALT4_EIM_AD0** — Select mux mode: ALT4 mux port: AD0 of instance: EIM 101 **ALT5_GPIO2_IO0** — Select mux mode: ALT5 mux port: IO0 of instance: GPIO2 110 **ALT6_LCD_DATA0** — Select mux mode: ALT6 mux port: DATA0 of instance: LCD 111 **ALT7_LCD_CLK** — Select mux mode: ALT7 mux port: CLK of instance: LCD

8.2.7.10 SW_MUX_CTL_PAD_EPDC_DATA01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA01)

SW_MUX_CTL Register

Address: 3033_0000h base + 38h offset = 3033_0038h



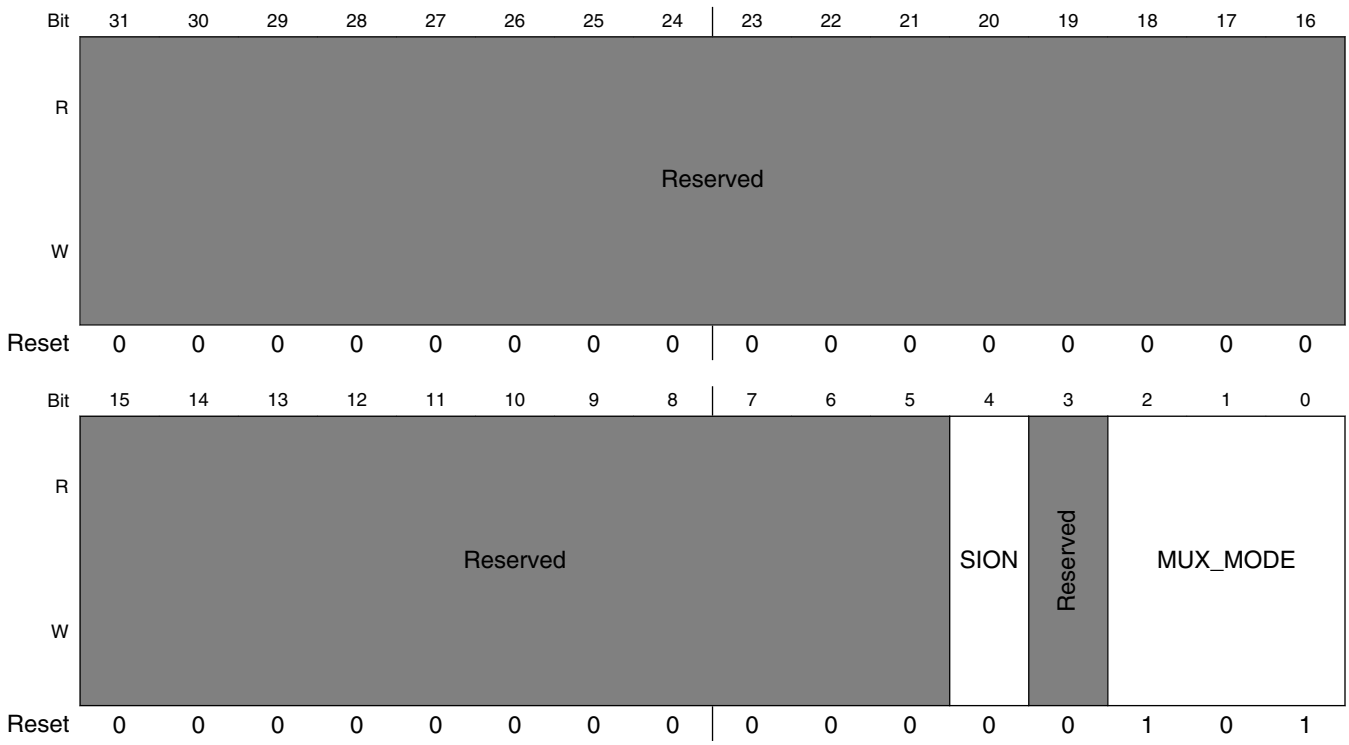
IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA01 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_DATA01 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_DATA01. 000 **ALT0_EPDC_DATA1** — Select mux mode: ALT0 mux port: DATA1 of instance: EPDC 001 **ALT1_SIM1_PORT2_CLK** — Select mux mode: ALT1 mux port: PORT2_CLK of instance: SIM1 010 **ALT2_QSPI_A_DATA1** — Select mux mode: ALT2 mux port: A_DATA1 of instance: QSPI 011 **ALT3_KPP_COL3** — Select mux mode: ALT3 mux port: COL3 of instance: KPP 100 **ALT4_EIM_AD1** — Select mux mode: ALT4 mux port: AD1 of instance: EIM 101 **ALT5_GPIO2_IO1** — Select mux mode: ALT5 mux port: IO1 of instance: GPIO2 110 **ALT6_LCD_DATA1** — Select mux mode: ALT6 mux port: DATA1 of instance: LCD 111 **ALT7_LCD_ENABLE** — Select mux mode: ALT7 mux port: ENABLE of instance: LCD

8.2.7.11 SW_MUX_CTL_PAD_EPDC_DATA02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA02)

SW_MUX_CTL Register

Address: 3033_0000h base + 3Ch offset = 3033_003Ch



IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_DATA02 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_DATA02. 000 **ALT0_EPDC_DATA2** — Select mux mode: ALT0 mux port: DATA2 of instance: EPDC 001 **ALT1_SIM1_PORT2_RST_B** — Select mux mode: ALT1 mux port: PORT2_RST_B of instance: SIM1

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA02 field descriptions (continued)

Field	Description
	010 **ALT2_QSPI_A_DATA2** — Select mux mode: ALT2 mux port: A_DATA2 of instance: QSPI 011 **ALT3_KPP_ROW2** — Select mux mode: ALT3 mux port: ROW2 of instance: KPP 100 **ALT4_EIM_AD2** — Select mux mode: ALT4 mux port: AD2 of instance: EIM 101 **ALT5_GPIO2_IO2** — Select mux mode: ALT5 mux port: IO2 of instance: GPIO2 110 **ALT6_LCD_DATA2** — Select mux mode: ALT6 mux port: DATA2 of instance: LCD 111 **ALT7_LCD_VSYNC** — Select mux mode: ALT7 mux port: VSYNC of instance: LCD

8.2.7.12 SW_MUX_CTL_PAD_EPDC_DATA03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA03)

SW_MUX_CTL Register

Address: 3033_0000h base + 40h offset = 3033_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA03 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

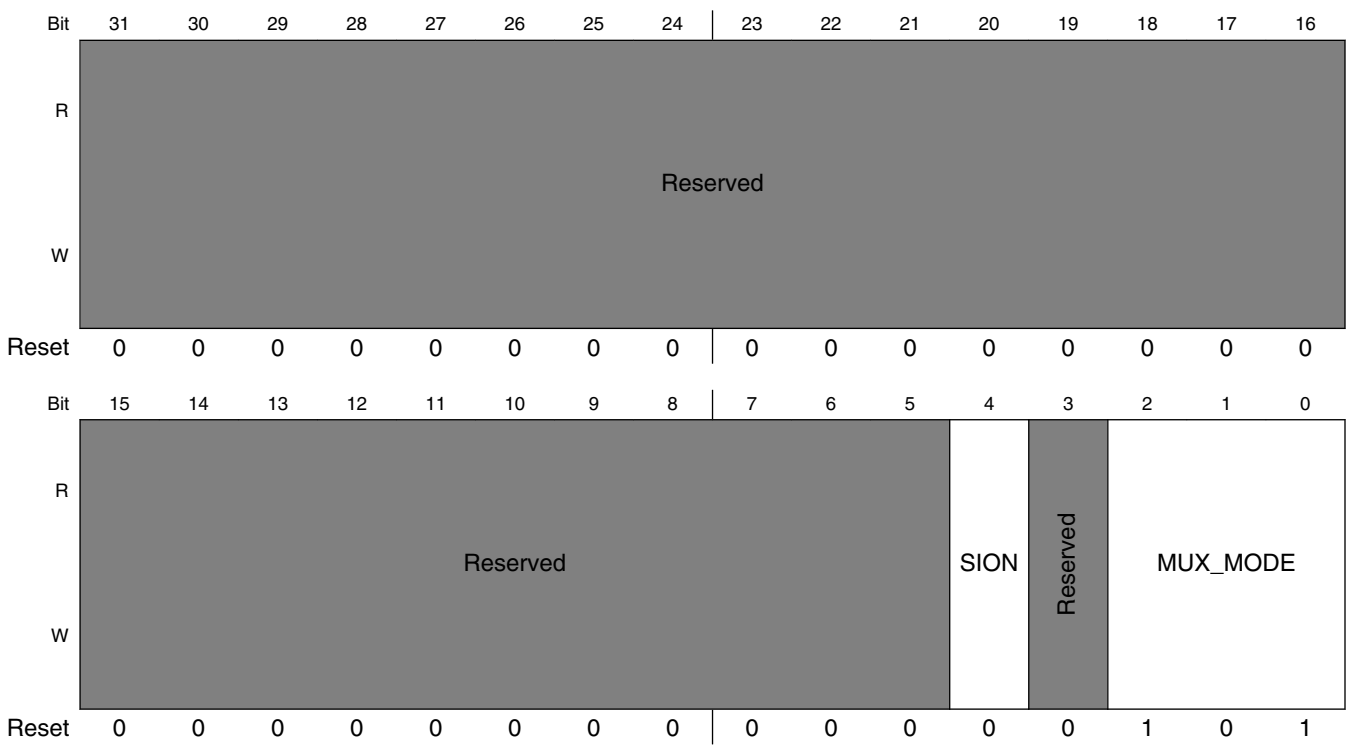
IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA03 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad EPDC_DATA03 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_DATA03. 000 **ALT0_EPDC_DATA3** — Select mux mode: ALT0 mux port: DATA3 of instance: EPDC 001 **ALT1_SIM1_PORT2_SVEN** — Select mux mode: ALT1 mux port: PORT2_SVEN of instance: SIM1 010 **ALT2_QSPI_A_DATA3** — Select mux mode: ALT2 mux port: A_DATA3 of instance: QSPI 011 **ALT3_KPP_COL2** — Select mux mode: ALT3 mux port: COL2 of instance: KPP 100 **ALT4_EIM_AD3** — Select mux mode: ALT4 mux port: AD3 of instance: EIM 101 **ALT5_GPIO2_IO3** — Select mux mode: ALT5 mux port: IO3 of instance: GPIO2 110 **ALT6_LCD_DATA3** — Select mux mode: ALT6 mux port: DATA3 of instance: LCD 111 **ALT7_LCD_HSYNC** — Select mux mode: ALT7 mux port: HSYNC of instance: LCD

8.2.7.13 SW_MUX_CTL_PAD_EPDC_DATA04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA04)

SW_MUX_CTL Register

Address: 3033_0000h base + 44h offset = 3033_0044h



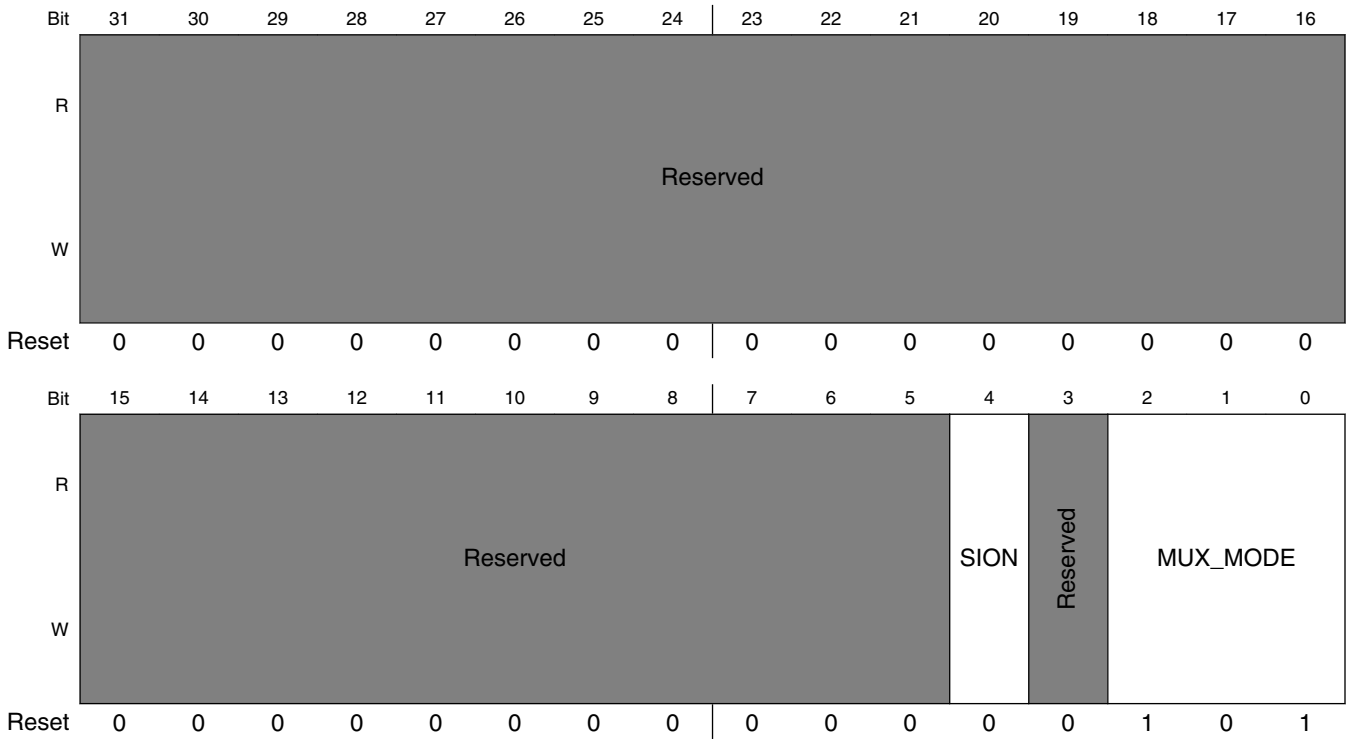
IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA04 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_DATA04 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_DATA04. 000 **ALT0_EPDC_DATA4** — Select mux mode: ALT0 mux port: DATA4 of instance: EPDC 001 **ALT1_SIM1_PORT2_PD** — Select mux mode: ALT1 mux port: PORT2_PD of instance: SIM1 010 **ALT2_QSPI_A_DQS** — Select mux mode: ALT2 mux port: A_DQS of instance: QSPI 011 **ALT3_KPP_ROW1** — Select mux mode: ALT3 mux port: ROW1 of instance: KPP 100 **ALT4_EIM_AD4** — Select mux mode: ALT4 mux port: AD4 of instance: EIM 101 **ALT5_GPIO2_IO4** — Select mux mode: ALT5 mux port: IO4 of instance: GPIO2 110 **ALT6_LCD_DATA4** — Select mux mode: ALT6 mux port: DATA4 of instance: LCD 111 **ALT7_JTAG_FAIL** — Select mux mode: ALT7 mux port: FAIL of instance: JTAG

8.2.7.14 SW_MUX_CTL_PAD_EPDC_DATA05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA05)

SW_MUX_CTL Register

Address: 3033_0000h base + 48h offset = 3033_0048h



IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA05 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_DATA05 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_DATA05. 000 **ALT0_EPDC_DATA5** — Select mux mode: ALT0 mux port: DATA5 of instance: EPDC 001 **ALT1_SIM2_PORT2_TRXD** — Select mux mode: ALT1 mux port: PORT2_TRXD of instance: SIM2 010 **ALT2_QSPI_A_SCLK** — Select mux mode: ALT2 mux port: A_SCLK of instance: QSPI

Table continues on the next page...

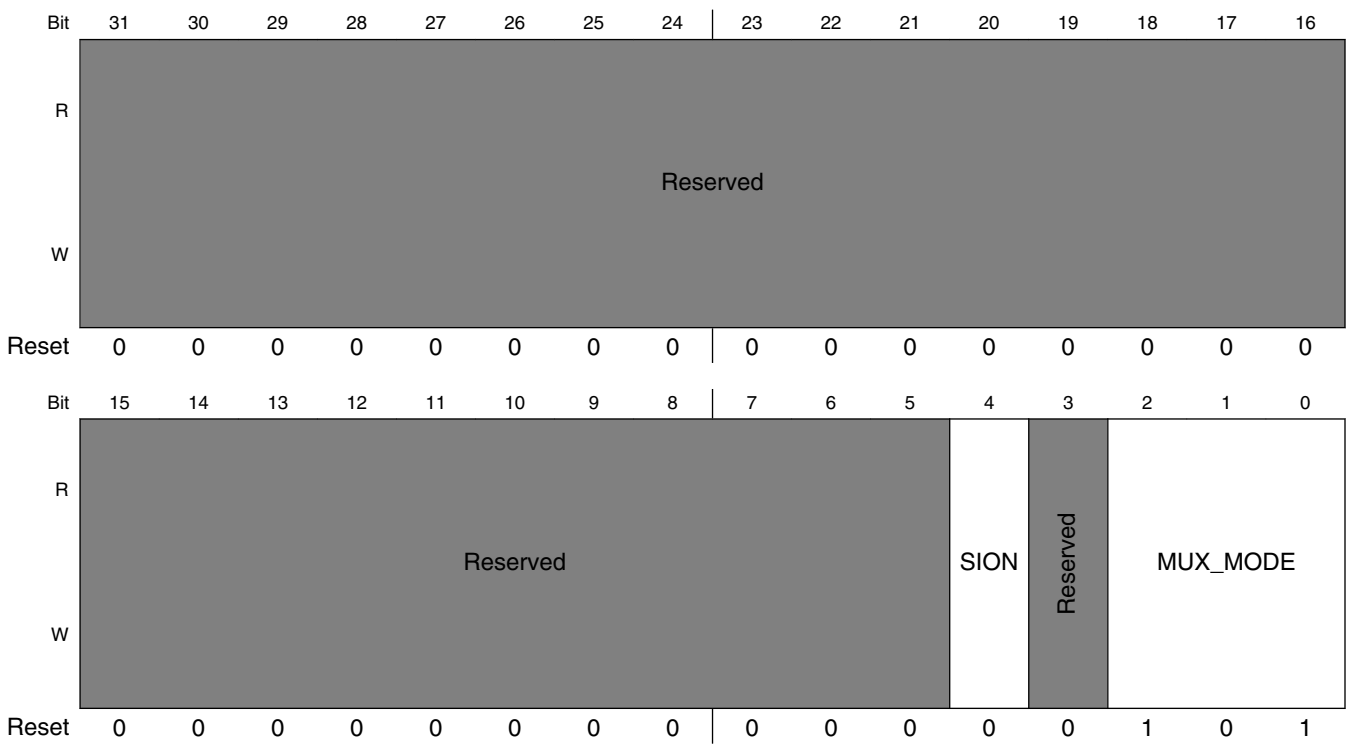
IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA05 field descriptions (continued)

Field	Description
	011 **ALT3_KPP_COL1** — Select mux mode: ALT3 mux port: COL1 of instance: KPP 100 **ALT4_EIM_AD5** — Select mux mode: ALT4 mux port: AD5 of instance: EIM 101 **ALT5_GPIO2_IO5** — Select mux mode: ALT5 mux port: IO5 of instance: GPIO2 110 **ALT6_LCD_DATA5** — Select mux mode: ALT6 mux port: DATA5 of instance: LCD 111 **ALT7_JTAG_ACTIVE** — Select mux mode: ALT7 mux port: ACTIVE of instance: JTAG

8.2.7.15 SW_MUX_CTL_PAD_EPDC_DATA06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA06)

SW_MUX_CTL Register

Address: 3033_0000h base + 4Ch offset = 3033_004Ch



IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA06 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

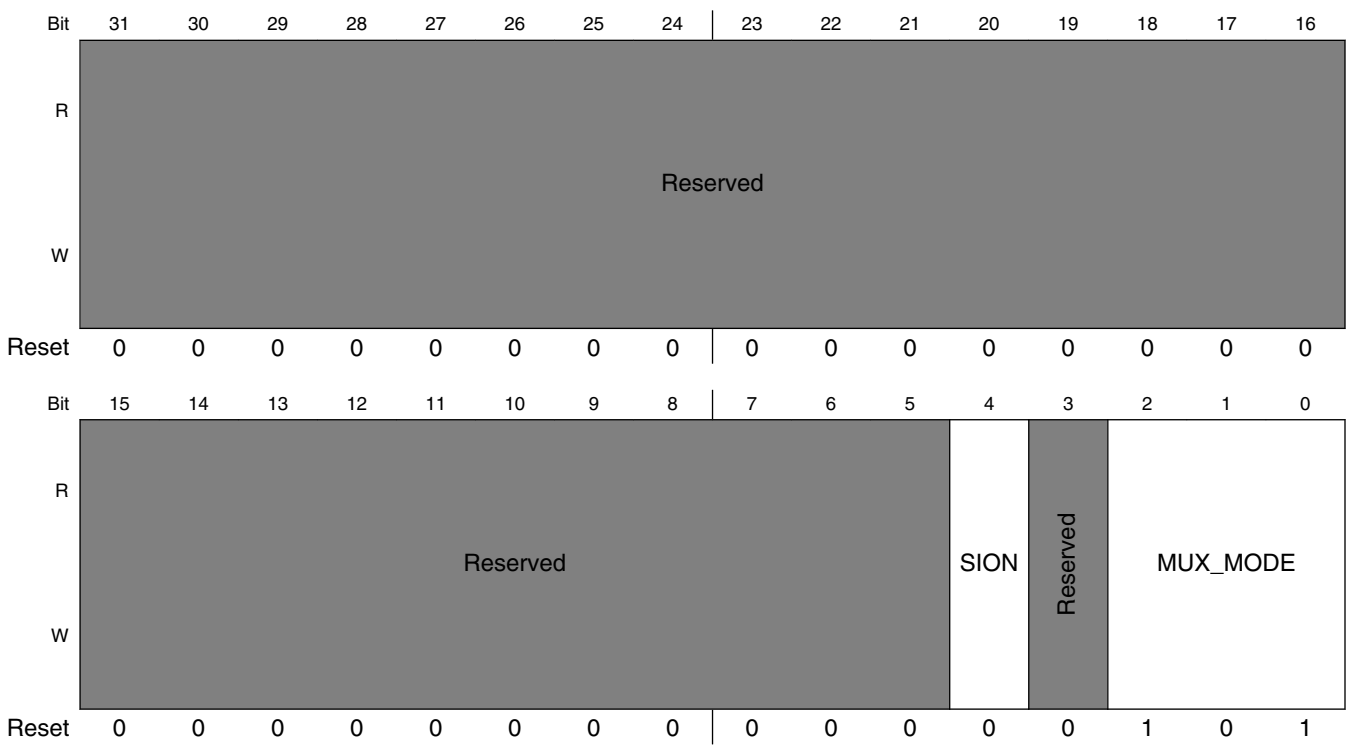
IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA06 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad EPDC_DATA06 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_DATA06. 000 **ALT0_EPDC_DATA6** — Select mux mode: ALT0 mux port: DATA6 of instance: EPDC 001 **ALT1_SIM2_PORT2_CLK** — Select mux mode: ALT1 mux port: PORT2_CLK of instance: SIM2 010 **ALT2_QSPI_A_SS0_B** — Select mux mode: ALT2 mux port: A_SS0_B of instance: QSPI 011 **ALT3_KPP_ROW0** — Select mux mode: ALT3 mux port: ROW0 of instance: KPP 100 **ALT4_EIM_AD6** — Select mux mode: ALT4 mux port: AD6 of instance: EIM 101 **ALT5_GPIO2_IO6** — Select mux mode: ALT5 mux port: IO6 of instance: GPIO2 110 **ALT6_LCD_DATA6** — Select mux mode: ALT6 mux port: DATA6 of instance: LCD 111 **ALT7_JTAG_DE_B** — Select mux mode: ALT7 mux port: DE_B of instance: JTAG

8.2.7.16 SW_MUX_CTL_PAD_EPDC_DATA07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA07)

SW_MUX_CTL Register

Address: 3033_0000h base + 50h offset = 3033_0050h



IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA07 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_DATA07 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_DATA07. 000 **ALT0_EPDC_DATA7** — Select mux mode: ALT0 mux port: DATA7 of instance: EPDC 001 **ALT1_SIM2_PORT2_RST_B** — Select mux mode: ALT1 mux port: PORT2_RST_B of instance: SIM2 010 **ALT2_QSPI_A_SS1_B** — Select mux mode: ALT2 mux port: A_SS1_B of instance: QSPI 011 **ALT3_KPP_COL0** — Select mux mode: ALT3 mux port: COL0 of instance: KPP 100 **ALT4_EIM_AD7** — Select mux mode: ALT4 mux port: AD7 of instance: EIM 101 **ALT5_GPIO2_IO7** — Select mux mode: ALT5 mux port: IO7 of instance: GPIO2 110 **ALT6_LCD_DATA7** — Select mux mode: ALT6 mux port: DATA7 of instance: LCD 111 **ALT7_JTAG_DONE** — Select mux mode: ALT7 mux port: DONE of instance: JTAG

8.2.7.17 SW_MUX_CTL_PAD_EPDC_DATA08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA08)

SW_MUX_CTL Register

Address: 3033_0000h base + 54h offset = 3033_0054h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA08 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA08 field descriptions (continued)

Field	Description
	Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_DATA08 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: EPDC_DATA08. 000 **ALT0_EPDC_DATA8** — Select mux mode: ALT0 mux port: DATA8 of instance: EPDC 001 **ALT1_SIM1_PORT1_TRXD** — Select mux mode: ALT1 mux port: PORT1_TRXD of instance: SIM1 010 **ALT2_QSPI_B_DATA0** — Select mux mode: ALT2 mux port: B_DATA0 of instance: QSPI 011 **ALT3_UART6_RX_DATA** — Select mux mode: ALT3 mux port: RX_DATA of instance: UART6 100 **ALT4_EIM_OE** — Select mux mode: ALT4 mux port: OE of instance: EIM 101 **ALT5_GPIO2_IO8** — Select mux mode: ALT5 mux port: IO8 of instance: GPIO2 110 **ALT6_LCD_DATA8** — Select mux mode: ALT6 mux port: DATA8 of instance: LCD 111 **ALT7_LCD_BUSY** — Select mux mode: ALT7 mux port: BUSY of instance: LCD 1000 **ALT8_EPDC_SDCLK** — Select mux mode: ALT8 mux port: SDCLK of instance: EPDC

8.2.7.18 SW_MUX_CTL_PAD_EPDC_DATA09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA09)

SW_MUX_CTL Register

Address: 3033_0000h base + 58h offset = 3033_0058h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA09 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_DATA09 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA09 field descriptions (continued)

Field	Description
	Select 1 of 9 iomux modes to be used for pad: EPDC_DATA09. 000 **ALT0_EPDC_DATA9** — Select mux mode: ALT0 mux port: DATA9 of instance: EPDC 001 **ALT1_SIM1_PORT1_CLK** — Select mux mode: ALT1 mux port: PORT1_CLK of instance: SIM1 010 **ALT2_QSPI_B_DATA1** — Select mux mode: ALT2 mux port: B_DATA1 of instance: QSPI 011 **ALT3_UART6_TX_DATA** — Select mux mode: ALT3 mux port: TX_DATA of instance: UART6 100 **ALT4_EIM_RW** — Select mux mode: ALT4 mux port: RW of instance: EIM 101 **ALT5_GPIO2_IO9** — Select mux mode: ALT5 mux port: IO9 of instance: GPIO2 110 **ALT6_LCD_DATA9** — Select mux mode: ALT6 mux port: DATA9 of instance: LCD 111 **ALT7_LCD_DATA0** — Select mux mode: ALT7 mux port: DATA0 of instance: LCD 1000 **ALT8_EPDC_SDLE** — Select mux mode: ALT8 mux port: SDLE of instance: EPDC

8.2.7.19 SW_MUX_CTL_PAD_EPDC_DATA10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA10)

SW_MUX_CTL Register

Address: 3033_0000h base + 5Ch offset = 3033_005Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA10 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_DATA10 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: EPDC_DATA10. 000 **ALT0_EPDC_DATA10** — Select mux mode: ALT0 mux port: DATA10 of instance: EPDC 001 **ALT1_SIM1_PORT1_RST_B** — Select mux mode: ALT1 mux port: PORT1_RST_B of instance: SIM1 010 **ALT2_QSPI_B_DATA2** — Select mux mode: ALT2 mux port: B_DATA2 of instance: QSPI

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA10 field descriptions (continued)

Field	Description
	011 **ALT3_UART6_RTS_B** — Select mux mode: ALT3 mux port: RTS_B of instance: UART6 100 **ALT4_EIM_CS0_B** — Select mux mode: ALT4 mux port: CS0_B of instance: EIM 101 **ALT5_GPIO2_IO10** — Select mux mode: ALT5 mux port: IO10 of instance: GPIO2 110 **ALT6_LCD_DATA10** — Select mux mode: ALT6 mux port: DATA10 of instance: LCD 111 **ALT7_LCD_DATA9** — Select mux mode: ALT7 mux port: DATA9 of instance: LCD 1000 **ALT8_EPDC_SDOE** — Select mux mode: ALT8 mux port: SDOE of instance: EPDC

8.2.7.20 SW_MUX_CTL_PAD_EPDC_DATA11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA11)

SW_MUX_CTL Register

Address: 3033_0000h base + 60h offset = 3033_0060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA11 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_DATA11 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: EPDC_DATA11. 000 **ALT0_EPDC_DATA11** — Select mux mode: ALT0 mux port: DATA11 of instance: EPDC 001 **ALT1_SIM1_PORT1_SVEN** — Select mux mode: ALT1 mux port: PORT1_SVEN of instance: SIM1 010 **ALT2_QSPI_B_DATA3** — Select mux mode: ALT2 mux port: B_DATA3 of instance: QSPI 011 **ALT3_UART6_CTS_B** — Select mux mode: ALT3 mux port: CTS_B of instance: UART6 100 **ALT4_EIM_BCLK** — Select mux mode: ALT4 mux port: BCLK of instance: EIM 101 **ALT5_GPIO2_IO11** — Select mux mode: ALT5 mux port: IO11 of instance: GPIO2 110 **ALT6_LCD_DATA11** — Select mux mode: ALT6 mux port: DATA11 of instance: LCD 111 **ALT7_LCD_DATA1** — Select mux mode: ALT7 mux port: DATA1 of instance: LCD 1000 **ALT8_EPDC_SDCE0** — Select mux mode: ALT8 mux port: SDCE0 of instance: EPDC

8.2.7.21 SW_MUX_CTL_PAD_EPDC_DATA12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA12)

SW_MUX_CTL Register

Address: 3033_0000h base + 64h offset = 3033_0064h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA12 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_DATA12 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: EPDC_DATA12. 000 **ALT0_EPDC_DATA12** — Select mux mode: ALT0 mux port: DATA12 of instance: EPDC 001 **ALT1_SIM1_PORT1_PD** — Select mux mode: ALT1 mux port: PORT1_PD of instance: SIM1 010 **ALT2_QSPI_B_DQS** — Select mux mode: ALT2 mux port: B_DQS of instance: QSPI 011 **ALT3_UART7_RX_DATA** — Select mux mode: ALT3 mux port: RX_DATA of instance: UART7 100 **ALT4_EIM_LBA_B** — Select mux mode: ALT4 mux port: LBA_B of instance: EIM 101 **ALT5_GPIO2_IO12** — Select mux mode: ALT5 mux port: IO12 of instance: GPIO2 110 **ALT6_LCD_DATA12** — Select mux mode: ALT6 mux port: DATA12 of instance: LCD 111 **ALT7_LCD_DATA21** — Select mux mode: ALT7 mux port: DATA21 of instance: LCD 1000 **ALT8_EPDC_GDCLK** — Select mux mode: ALT8 mux port: GDCLK of instance: EPDC

8.2.7.22 SW_MUX_CTL_PAD_EPDC_DATA13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA13)

SW_MUX_CTL Register

Address: 3033_0000h base + 68h offset = 3033_0068h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA13 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_DATA13 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: EPDC_DATA13. 000 **ALT0_EPDC_DATA13** — Select mux mode: ALT0 mux port: DATA13 of instance: EPDC 001 **ALT1_SIM2_PORT1_TRXD** — Select mux mode: ALT1 mux port: PORT1_TRXD of instance: SIM2 010 **ALT2_QSPI_B_SCLK** — Select mux mode: ALT2 mux port: B_SCLK of instance: QSPI 011 **ALT3_UART7_TX_DATA** — Select mux mode: ALT3 mux port: TX_DATA of instance: UART7 100 **ALT4_EIM_WAIT** — Select mux mode: ALT4 mux port: WAIT of instance: EIM 101 **ALT5_GPIO2_IO13** — Select mux mode: ALT5 mux port: IO13 of instance: GPIO2 110 **ALT6_LCD_DATA13** — Select mux mode: ALT6 mux port: DATA13 of instance: LCD 111 **ALT7_LCD_CS** — Select mux mode: ALT7 mux port: CS of instance: LCD 1000 **ALT8_EPDC_GDOE** — Select mux mode: ALT8 mux port: GDOE of instance: EPDC

8.2.7.23 SW_MUX_CTL_PAD_EPDC_DATA14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA14)

SW_MUX_CTL Register

Address: 3033_0000h base + 6Ch offset = 3033_006Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA14 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_DATA14 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: EPDC_DATA14. 000 **ALT0_EPDC_DATA14** — Select mux mode: ALT0 mux port: DATA14 of instance: EPDC 001 **ALT1_SIM2_PORT1_CLK** — Select mux mode: ALT1 mux port: PORT1_CLK of instance: SIM2 010 **ALT2_QSPI_B_SS0_B** — Select mux mode: ALT2 mux port: B_SS0_B of instance: QSPI 011 **ALT3_UART7_RTS_B** — Select mux mode: ALT3 mux port: RTS_B of instance: UART7 100 **ALT4_EIM_EB_B0** — Select mux mode: ALT4 mux port: EB_B0 of instance: EIM 101 **ALT5_GPIO2_IO14** — Select mux mode: ALT5 mux port: IO14 of instance: GPIO2 110 **ALT6_LCD_DATA14** — Select mux mode: ALT6 mux port: DATA14 of instance: LCD 111 **ALT7_LCD_DATA22** — Select mux mode: ALT7 mux port: DATA22 of instance: LCD 1000 **ALT8_EPDC_GDSP** — Select mux mode: ALT8 mux port: GDSP of instance: EPDC

8.2.7.24 SW_MUX_CTL_PAD_EPDC_DATA15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA15)

SW_MUX_CTL Register

Address: 3033_0000h base + 70h offset = 3033_0070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

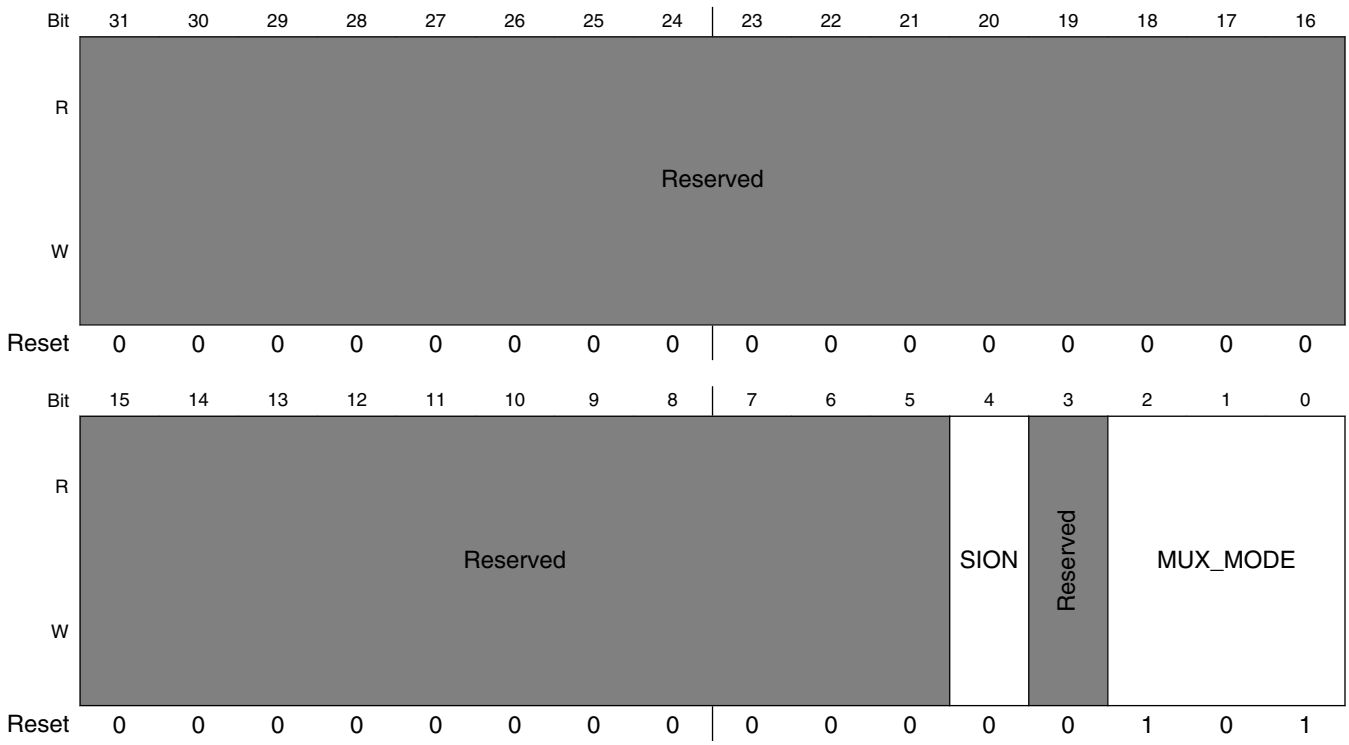
IOMUXC_SW_MUX_CTL_PAD_EPDC_DATA15 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_DATA15 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select 1 of 9 iomux modes to be used for pad: EPDC_DATA15. 000 **ALT0_EPDC_DATA15** — Select mux mode: ALT0 mux port: DATA15 of instance: EPDC 001 **ALT1_SIM2_PORT1_RST_B** — Select mux mode: ALT1 mux port: PORT1_RST_B of instance: SIM2 010 **ALT2_QSPI_B_SS1_B** — Select mux mode: ALT2 mux port: B_SS1_B of instance: QSPI 011 **ALT3_UART7_CTS_B** — Select mux mode: ALT3 mux port: CTS_B of instance: UART7 100 **ALT4_EIM_CS1_B** — Select mux mode: ALT4 mux port: CS1_B of instance: EIM 101 **ALT5_GPIO2_IO15** — Select mux mode: ALT5 mux port: IO15 of instance: GPIO2 110 **ALT6_LCD_DATA15** — Select mux mode: ALT6 mux port: DATA15 of instance: LCD 111 **ALT7_LCD_WR_RWN** — Select mux mode: ALT7 mux port: WR_RWN of instance: LCD 1000 **ALT8_EPDC_PWR_COM** — Select mux mode: ALT8 mux port: PWR_COM of instance: EPDC

8.2.7.25 SW_MUX_CTL_PAD_EPDC_SDCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCLK)

SW_MUX_CTL Register

Address: 3033_0000h base + 74h offset = 3033_0074h



IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_SDCLK 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_SDCLK. 000 **ALT0_EPDC_SDCLK** — Select mux mode: ALT0 mux port: SDCLK of instance: EPDC 001 **ALT1_SIM2_PORT2_SVEN** — Select mux mode: ALT1 mux port: PORT2_SVEN of instance: SIM2 010 **ALT2_ENET2_RGMII_RD0** — Select mux mode: ALT2 mux port: RGMII_RD0 of instance: ENET2

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCLK field descriptions (continued)

Field	Description
	011 **ALT3_KPP_ROW4** — Select mux mode: ALT3 mux port: ROW4 of instance: KPP 100 **ALT4_EIM_AD10** — Select mux mode: ALT4 mux port: AD10 of instance: EIM 101 **ALT5_GPIO2_IO16** — Select mux mode: ALT5 mux port: IO16 of instance: GPIO2 110 **ALT6_LCD_CLK** — Select mux mode: ALT6 mux port: CLK of instance: LCD 111 **ALT7_LCD_DATA20** — Select mux mode: ALT7 mux port: DATA20 of instance: LCD

8.2.7.26 SW_MUX_CTL_PAD_EPDC_SDLE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDLE)

SW_MUX_CTL Register

Address: 3033_0000h base + 78h offset = 3033_0078h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_EPDC_SDLE field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_EPDC_SDLE field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad EPDC_SDLE 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_SDLE. 000 **ALT0_EPDC_SDLE** — Select mux mode: ALT0 mux port: SDLE of instance: EPDC 001 **ALT1_SIM2_PORT2_PD** — Select mux mode: ALT1 mux port: PORT2_PD of instance: SIM2 010 **ALT2_ENET2_RGMII_RD1** — Select mux mode: ALT2 mux port: RGMII_RD1 of instance: ENET2 011 **ALT3_KPP_COL4** — Select mux mode: ALT3 mux port: COL4 of instance: KPP 100 **ALT4_EIM_AD11** — Select mux mode: ALT4 mux port: AD11 of instance: EIM 101 **ALT5_GPIO2_IO17** — Select mux mode: ALT5 mux port: IO17 of instance: GPIO2 110 **ALT6_LCD_DATA16** — Select mux mode: ALT6 mux port: DATA16 of instance: LCD 111 **ALT7_LCD_DATA8** — Select mux mode: ALT7 mux port: DATA8 of instance: LCD

8.2.7.27 SW_MUX_CTL_PAD_EPDC_SDOE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDOE)

SW_MUX_CTL Register

Address: 3033_0000h base + 7Ch offset = 3033_007Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

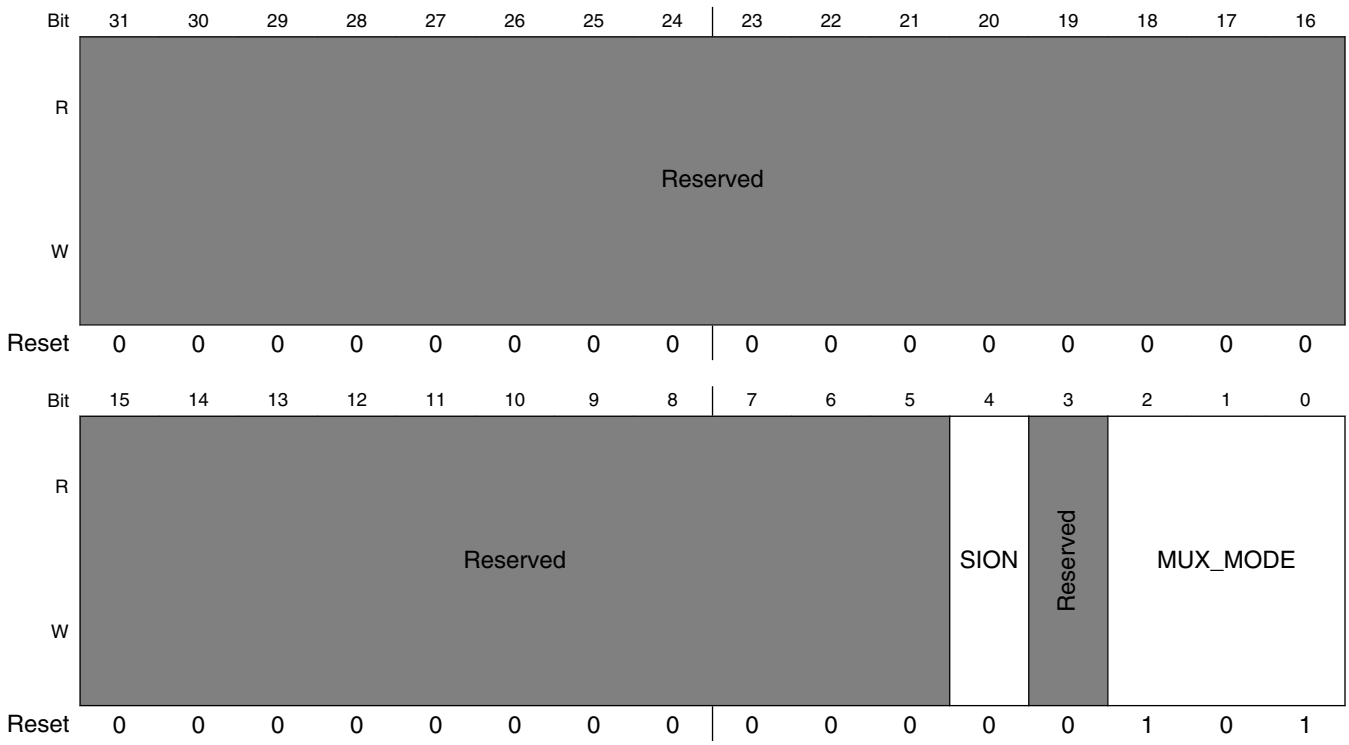
IOMUXC_SW_MUX_CTL_PAD_EPDC_SDOE field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_SDOE 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_SDOE. 000 **ALT0_EPDC_SDOE** — Select mux mode: ALT0 mux port: SDOE of instance: EPDC 001 **ALT1_FLEXTIMER1_CH0** — Select mux mode: ALT1 mux port: CH0 of instance: FLEXTIMER1 010 **ALT2_ENET2_RGMII_RD2** — Select mux mode: ALT2 mux port: RGMII_RD2 of instance: ENET2 011 **ALT3_KPP_COL5** — Select mux mode: ALT3 mux port: COL5 of instance: KPP 100 **ALT4_EIM_AD12** — Select mux mode: ALT4 mux port: AD12 of instance: EIM 101 **ALT5_GPIO2_IO18** — Select mux mode: ALT5 mux port: IO18 of instance: GPIO2 110 **ALT6_LCD_DATA17** — Select mux mode: ALT6 mux port: DATA17 of instance: LCD 111 **ALT7_LCD_DATA23** — Select mux mode: ALT7 mux port: DATA23 of instance: LCD

8.2.7.28 SW_MUX_CTL_PAD_EPDC_SDSHR SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDSHR)

SW_MUX_CTL Register

Address: 3033_0000h base + 80h offset = 3033_0080h



IOMUXC_SW_MUX_CTL_PAD_EPDC_SDSHR field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_SDSHR 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_SDSHR. 000 **ALT0_EPDC_SDSHR** — Select mux mode: ALT0 mux port: SDSHR of instance: EPDC 001 **ALT1_FLEXTIMER1_CH1** — Select mux mode: ALT1 mux port: CH1 of instance: FLEXTIMER1 010 **ALT2_ENET2_RGMII_RD3** — Select mux mode: ALT2 mux port: RGMII_RD3 of instance: ENET2

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_EPDC_SDSHR field descriptions (continued)

Field	Description
	011 **ALT3_KPP_ROW5** — Select mux mode: ALT3 mux port: ROW5 of instance: KPP 100 **ALT4_EIM_AD13** — Select mux mode: ALT4 mux port: AD13 of instance: EIM 101 **ALT5_GPIO2_IO19** — Select mux mode: ALT5 mux port: IO19 of instance: GPIO2 110 **ALT6_LCD_DATA18** — Select mux mode: ALT6 mux port: DATA18 of instance: LCD 111 **ALT7_LCD_DATA10** — Select mux mode: ALT7 mux port: DATA10 of instance: LCD

8.2.7.29 SW_MUX_CTL_PAD_EPDC_SDCE0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE0)

SW_MUX_CTL Register

Address: 3033_0000h base + 84h offset = 3033_0084h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE0 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad EPDC_SDCE0 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_SDCE0. 000 **ALT0_EPDC_SDCE0** — Select mux mode: ALT0 mux port: SDCE0 of instance: EPDC 001 **ALT1_FLEXTIMER1_CH2** — Select mux mode: ALT1 mux port: CH2 of instance: FLEXTIMER1 010 **ALT2_ENET2_RGMII_RX_CTL** — Select mux mode: ALT2 mux port: RGMII_RX_CTL of instance: ENET2 100 **ALT4_EIM_AD14** — Select mux mode: ALT4 mux port: AD14 of instance: EIM 101 **ALT5_GPIO2_IO20** — Select mux mode: ALT5 mux port: IO20 of instance: GPIO2 110 **ALT6_LCD_DATA19** — Select mux mode: ALT6 mux port: DATA19 of instance: LCD 111 **ALT7_LCD_DATA5** — Select mux mode: ALT7 mux port: DATA5 of instance: LCD

8.2.7.30 SW_MUX_CTL_PAD_EPDC_SDCE1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE1)

SW_MUX_CTL Register

Address: 3033_0000h base + 88h offset = 3033_0088h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

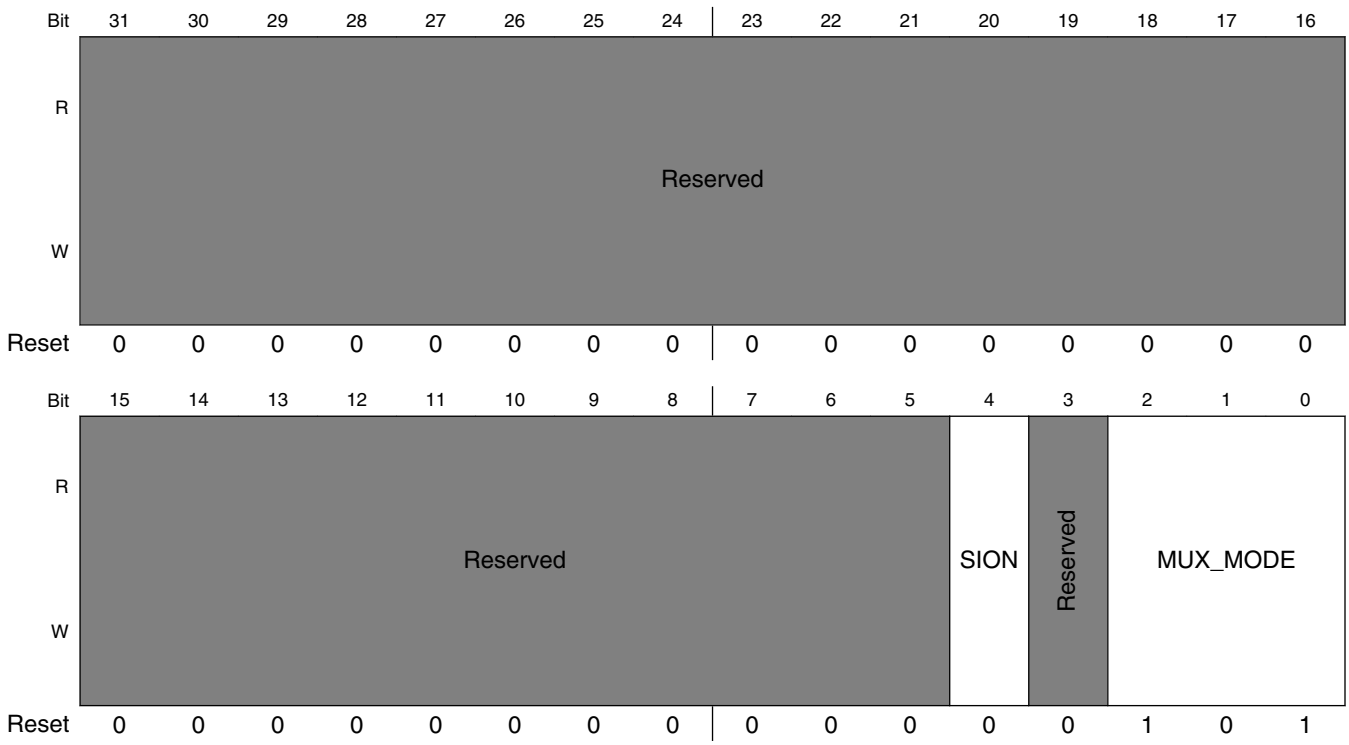
IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE1 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_SDCE1 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_SDCE1. 000 **ALT0_EPDC_SDCE1** — Select mux mode: ALT0 mux port: SDCE1 of instance: EPDC 001 **ALT1_FLEXTIMER1_CH3** — Select mux mode: ALT1 mux port: CH3 of instance: FLEXTIMER1 010 **ALT2_ENET2_RGMII_RXC** — Select mux mode: ALT2 mux port: RGMII_RXC of instance: ENET2 011 **ALT3_ENET2_RX_ER** — Select mux mode: ALT3 mux port: RX_ER of instance: ENET2 100 **ALT4_EIM_AD15** — Select mux mode: ALT4 mux port: AD15 of instance: EIM 101 **ALT5_GPIO2_IO21** — Select mux mode: ALT5 mux port: IO21 of instance: GPIO2 110 **ALT6_LCD_DATA20** — Select mux mode: ALT6 mux port: DATA20 of instance: LCD 111 **ALT7_LCD_DATA4** — Select mux mode: ALT7 mux port: DATA4 of instance: LCD

8.2.7.31 SW_MUX_CTL_PAD_EPDC_SDCE2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE2)

SW_MUX_CTL Register

Address: 3033_0000h base + 8Ch offset = 3033_008Ch



IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE2 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_SDCE2 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_SDCE2. 000 **ALT0_EPDC_SDCE2** — Select mux mode: ALT0 mux port: SDCE2 of instance: EPDC 001 **ALT1_SIM2_PORT1_SVEN** — Select mux mode: ALT1 mux port: PORT1_SVEN of instance: SIM2 010 **ALT2_ENET2_RGMII_TD0** — Select mux mode: ALT2 mux port: RGMII_TD0 of instance: ENET2

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE2 field descriptions (continued)

Field	Description
	011 **ALT3_KPP_COL6** — Select mux mode: ALT3 mux port: COL6 of instance: KPP 100 **ALT4_EIM_ADDR16** — Select mux mode: ALT4 mux port: ADDR16 of instance: EIM 101 **ALT5_GPIO2_IO22** — Select mux mode: ALT5 mux port: IO22 of instance: GPIO2 110 **ALT6_LCD_DATA21** — Select mux mode: ALT6 mux port: DATA21 of instance: LCD 111 **ALT7_LCD_DATA3** — Select mux mode: ALT7 mux port: DATA3 of instance: LCD

8.2.7.32 SW_MUX_CTL_PAD_EPDC_SDCE3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE3)

SW_MUX_CTL Register

Address: 3033_0000h base + 90h offset = 3033_0090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE3 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

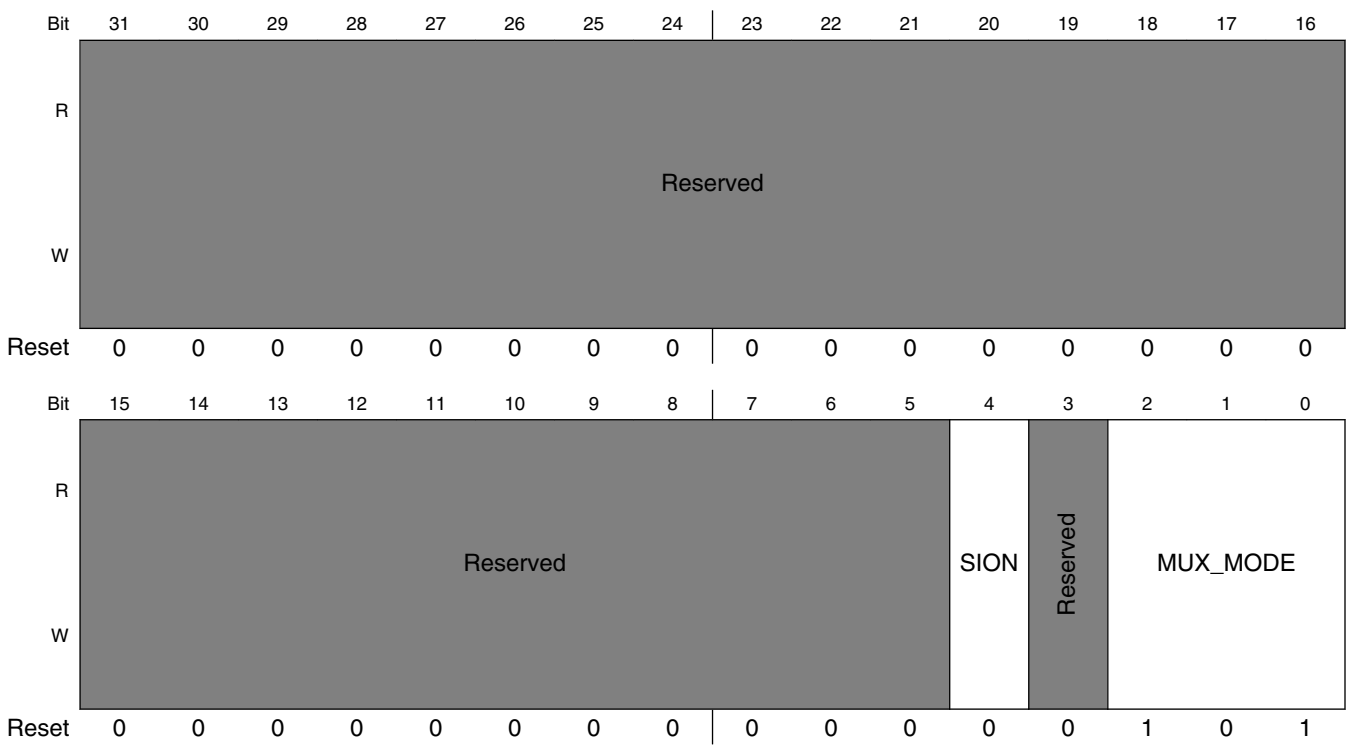
IOMUXC_SW_MUX_CTL_PAD_EPDC_SDCE3 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad EPDC_SDCE3 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_SDCE3. 000 **ALT0_EPDC_SDCE3** — Select mux mode: ALT0 mux port: SDCE3 of instance: EPDC 001 **ALT1_SIM2_PORT1_PD** — Select mux mode: ALT1 mux port: PORT1_PD of instance: SIM2 010 **ALT2_ENET2_RGMII_TD1** — Select mux mode: ALT2 mux port: RGMII_TD1 of instance: ENET2 011 **ALT3_KPP_ROW6** — Select mux mode: ALT3 mux port: ROW6 of instance: KPP 100 **ALT4_EIM_ADDR17** — Select mux mode: ALT4 mux port: ADDR17 of instance: EIM 101 **ALT5_GPIO2_IO23** — Select mux mode: ALT5 mux port: IO23 of instance: GPIO2 110 **ALT6_LCD_DATA22** — Select mux mode: ALT6 mux port: DATA22 of instance: LCD 111 **ALT7_LCD_DATA2** — Select mux mode: ALT7 mux port: DATA2 of instance: LCD

8.2.7.33 SW_MUX_CTL_PAD_EPDC_GDCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_GDCLK)

SW_MUX_CTL Register

Address: 3033_0000h base + 94h offset = 3033_0094h



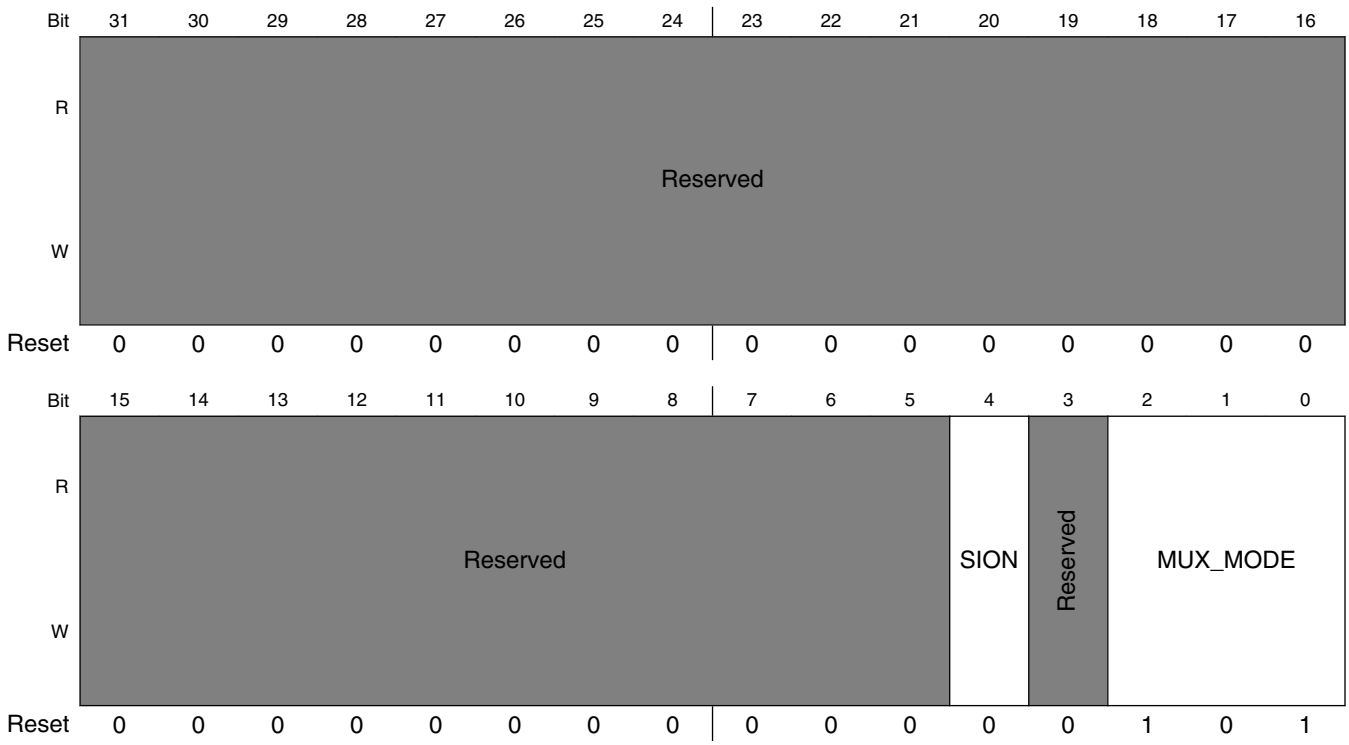
IOMUXC_SW_MUX_CTL_PAD_EPDC_GDCLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_GDCLK 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_GDCLK. 000 **ALT0_EPDC_GDCLK** — Select mux mode: ALT0 mux port: GDCLK of instance: EPDC 001 **ALT1_FLEXTIMER2_CH0** — Select mux mode: ALT1 mux port: CH0 of instance: FLEXTIMER2 010 **ALT2_ENET2_RGMII_TD2** — Select mux mode: ALT2 mux port: RGMII_TD2 of instance: ENET2 011 **ALT3_KPP_COL7** — Select mux mode: ALT3 mux port: COL7 of instance: KPP 100 **ALT4_EIM_ADDR18** — Select mux mode: ALT4 mux port: ADDR18 of instance: EIM 101 **ALT5_GPIO2_IO24** — Select mux mode: ALT5 mux port: IO24 of instance: GPIO2 110 **ALT6_LCD_DATA23** — Select mux mode: ALT6 mux port: DATA23 of instance: LCD 111 **ALT7_LCD_DATA16** — Select mux mode: ALT7 mux port: DATA16 of instance: LCD

8.2.7.34 SW_MUX_CTL_PAD_EPDC_GDOE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_GDOE)

SW_MUX_CTL Register

Address: 3033_0000h base + 98h offset = 3033_0098h



IOMUXC_SW_MUX_CTL_PAD_EPDC_GDOE field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_GDOE 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_GDOE. 000 **ALT0_EPDC_GDOE** — Select mux mode: ALT0 mux port: GDOE of instance: EPDC 001 **ALT1_FLEXTIMER2_CH1** — Select mux mode: ALT1 mux port: CH1 of instance: FLEXTIMER2 010 **ALT2_ENET2_RGMII_TD3** — Select mux mode: ALT2 mux port: RGMII_TD3 of instance: ENET2

Table continues on the next page...

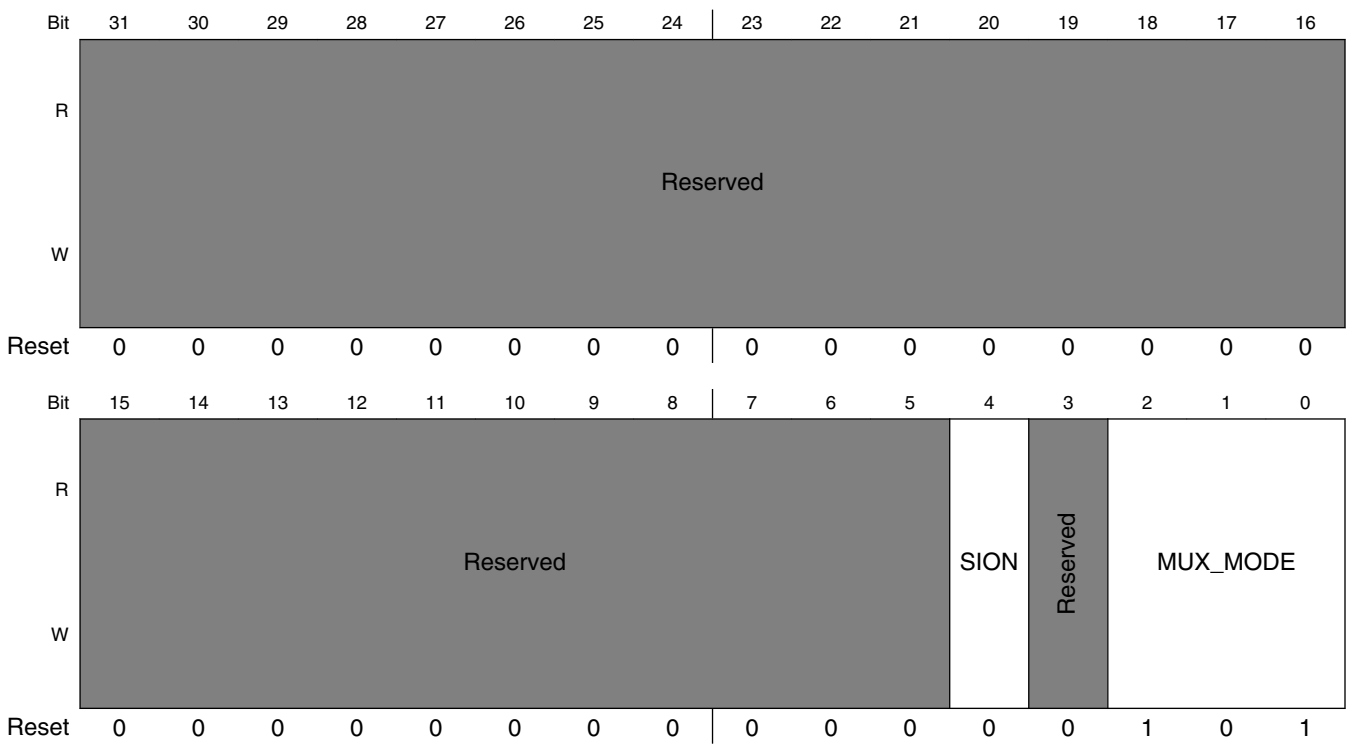
IOMUXC_SW_MUX_CTL_PAD_EPDC_GDOE field descriptions (continued)

Field	Description
	011 **ALT3_KPP_ROW7** — Select mux mode: ALT3 mux port: ROW7 of instance: KPP 100 **ALT4_EIM_ADDR19** — Select mux mode: ALT4 mux port: ADDR19 of instance: EIM 101 **ALT5_GPIO2_IO25** — Select mux mode: ALT5 mux port: IO25 of instance: GPIO2 110 **ALT6_LCD_WR_RWN** — Select mux mode: ALT6 mux port: WR_RWN of instance: LCD 111 **ALT7_LCD_DATA18** — Select mux mode: ALT7 mux port: DATA18 of instance: LCD

8.2.7.35 SW_MUX_CTL_PAD_EPDC_GDRL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_GDRL)

SW_MUX_CTL Register

Address: 3033_0000h base + 9Ch offset = 3033_009Ch



IOMUXC_SW_MUX_CTL_PAD_EPDC_GDRL field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

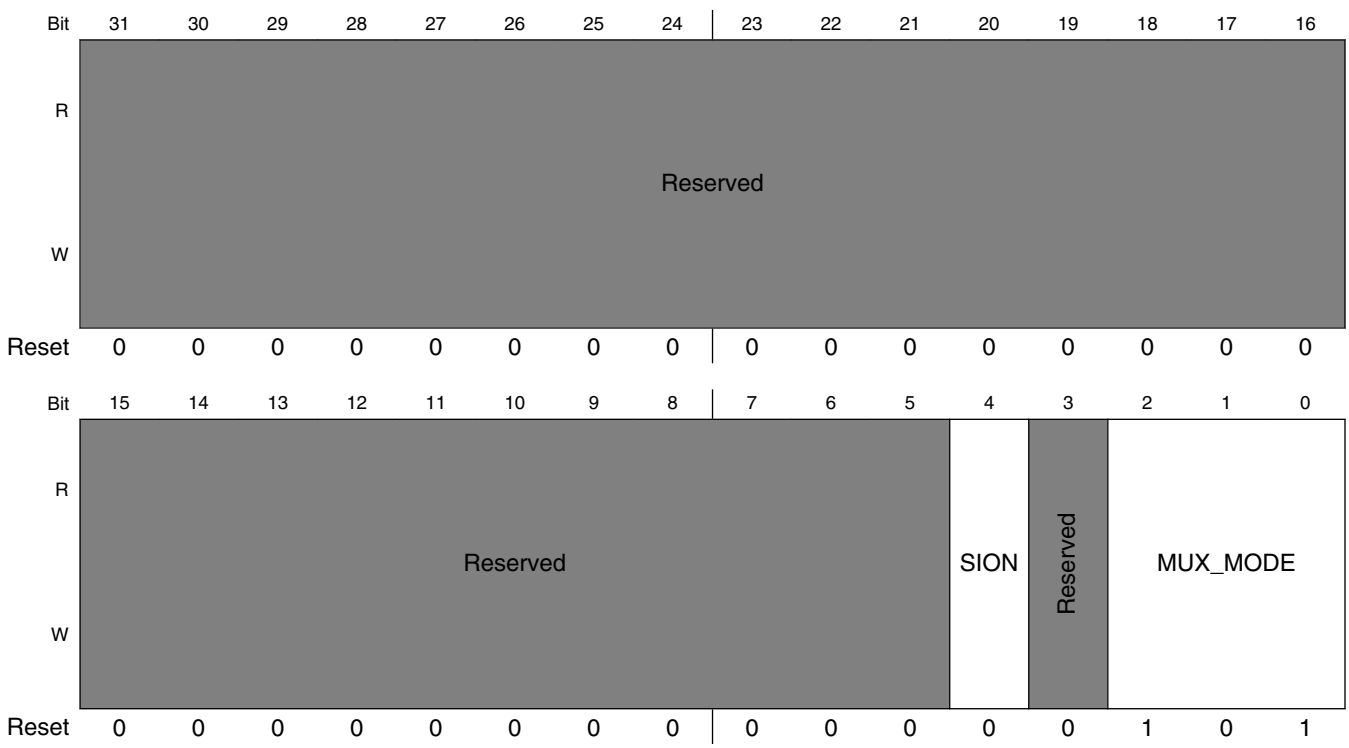
IOMUXC_SW_MUX_CTL_PAD_EPDC_GDRL field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad EPDC_GDRL 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_GDRL. 000 **ALT0_EPDC_GDRL** — Select mux mode: ALT0 mux port: GDRL of instance: EPDC 001 **ALT1_FLEXTIMER2_CH2** — Select mux mode: ALT1 mux port: CH2 of instance: FLEXTIMER2 010 **ALT2_ENET2_RGMII_TX_CTL** — Select mux mode: ALT2 mux port: RGMII_TX_CTL of instance: ENET2 100 **ALT4_EIM_ADDR20** — Select mux mode: ALT4 mux port: ADDR20 of instance: EIM 101 **ALT5_GPIO2_IO26** — Select mux mode: ALT5 mux port: IO26 of instance: GPIO2 110 **ALT6_LCD_RD_E** — Select mux mode: ALT6 mux port: RD_E of instance: LCD 111 **ALT7_LCD_DATA19** — Select mux mode: ALT7 mux port: DATA19 of instance: LCD

8.2.7.36 SW_MUX_CTL_PAD_EPDC_GDSP SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_GDSP)

SW_MUX_CTL Register

Address: 3033_0000h base + A0h offset = 3033_00A0h



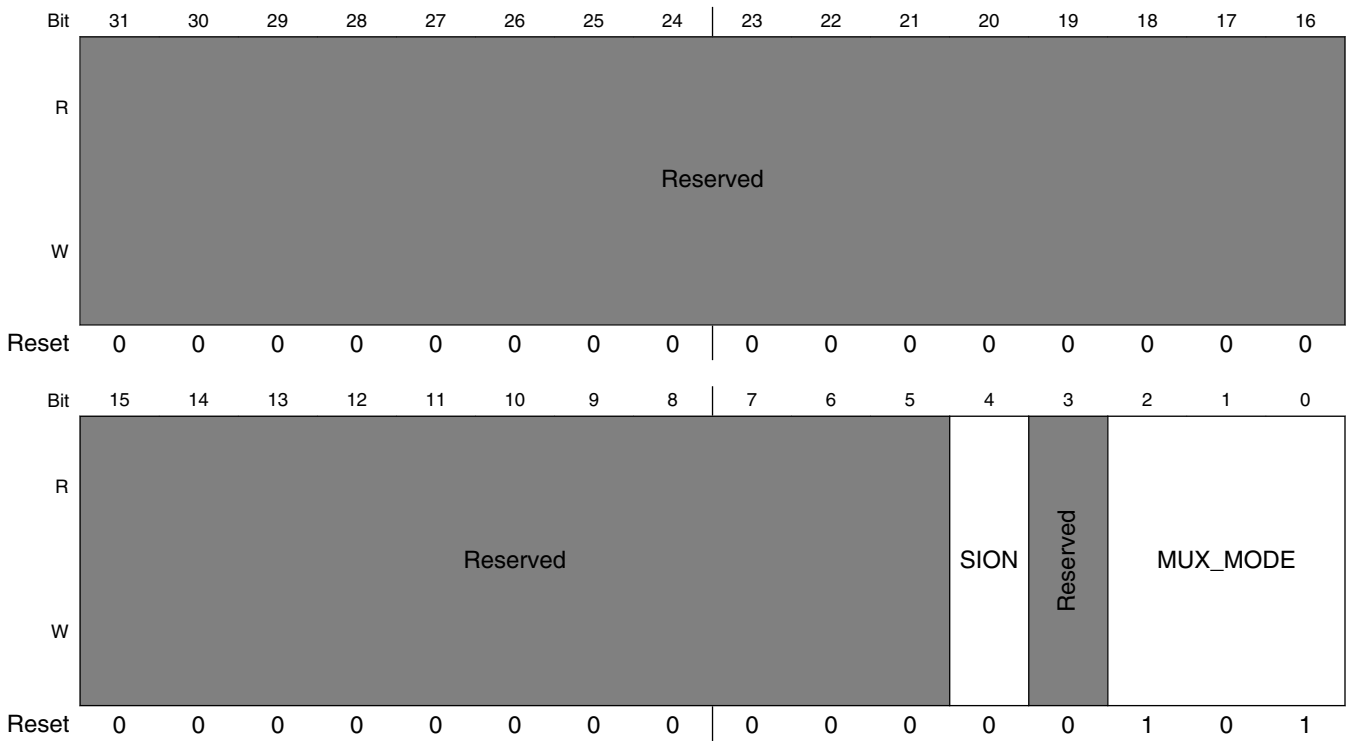
IOMUXC_SW_MUX_CTL_PAD_EPDC_GDSP field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_GDSP 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPDC_GDSP. 000 **ALT0_EPDC_GDSP** — Select mux mode: ALT0 mux port: GDSP of instance: EPDC 001 **ALT1_FLEXTIMER2_CH3** — Select mux mode: ALT1 mux port: CH3 of instance: FLEXTIMER2 010 **ALT2_ENET2_RGMII_TXC** — Select mux mode: ALT2 mux port: RGMII_TXC of instance: ENET2 011 **ALT3_ENET2_TX_ER** — Select mux mode: ALT3 mux port: TX_ER of instance: ENET2 100 **ALT4_EIM_ADDR21** — Select mux mode: ALT4 mux port: ADDR21 of instance: EIM 101 **ALT5_GPIO2_IO27** — Select mux mode: ALT5 mux port: IO27 of instance: GPIO2 110 **ALT6_LCD_BUSY** — Select mux mode: ALT6 mux port: BUSY of instance: LCD 111 **ALT7_LCD_DATA17** — Select mux mode: ALT7 mux port: DATA17 of instance: LCD

8.2.7.37 SW_MUX_CTL_PAD_EPDC_BDR0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_BDR0)

SW_MUX_CTL Register

Address: 3033_0000h base + A4h offset = 3033_00A4h



IOMUXC_SW_MUX_CTL_PAD_EPDC_BDR0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_BDR0 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_BDR0. 000 **ALT0_EPDC_BDR0** — Select mux mode: ALT0 mux port: BDR0 of instance: EPDC 010 **ALT2_ENET2_TX_CLK** — Select mux mode: ALT2 mux port: TX_CLK of instance: ENET2

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_EPDC_BDR0 field descriptions (continued)

Field	Description
	011 **ALT3_CCM_ENET2_REF_CLK** — Select mux mode: ALT3 mux port: ENET2_REF_CLK of instance: ENET2 100 **ALT4_EIM_ADDR22** — Select mux mode: ALT4 mux port: ADDR22 of instance: EIM 101 **ALT5_GPIO2_IO28** — Select mux mode: ALT5 mux port: IO28 of instance: GPIO2 110 **ALT6_LCD_CS** — Select mux mode: ALT6 mux port: CS of instance: LCD 111 **ALT7_LCD_DATA7** — Select mux mode: ALT7 mux port: DATA7 of instance: LCD

8.2.7.38 SW_MUX_CTL_PAD_EPDC_BDR1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_BDR1)

SW_MUX_CTL Register

Address: 3033_0000h base + A8h offset = 3033_00A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_EPDC_BDR1 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_EPDC_BDR1 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad EPDC_BDR1 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_BDR1. 000 **ALT0_EPDC_BDR1** — Select mux mode: ALT0 mux port: BDR1 of instance: EPDC 001 **ALT1_EPDC_SDCLKN** — Select mux mode: ALT1 mux port: SDCLKN of instance: EPDC 010 **ALT2_ENET2_RX_CLK** — Select mux mode: ALT2 mux port: RX_CLK of instance: ENET2 100 **ALT4_EIM_AD8** — Select mux mode: ALT4 mux port: AD8 of instance: EIM 101 **ALT5_GPIO2_IO29** — Select mux mode: ALT5 mux port: IO29 of instance: GPIO2 110 **ALT6_LCD_ENABLE** — Select mux mode: ALT6 mux port: ENABLE of instance: LCD 111 **ALT7_LCD_DATA6** — Select mux mode: ALT7 mux port: DATA6 of instance: LCD

8.2.7.39 SW_MUX_CTL_PAD_EPDC_PWR_COM SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_PWR_COM)

SW_MUX_CTL Register

Address: 3033_0000h base + ACh offset = 3033_00ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

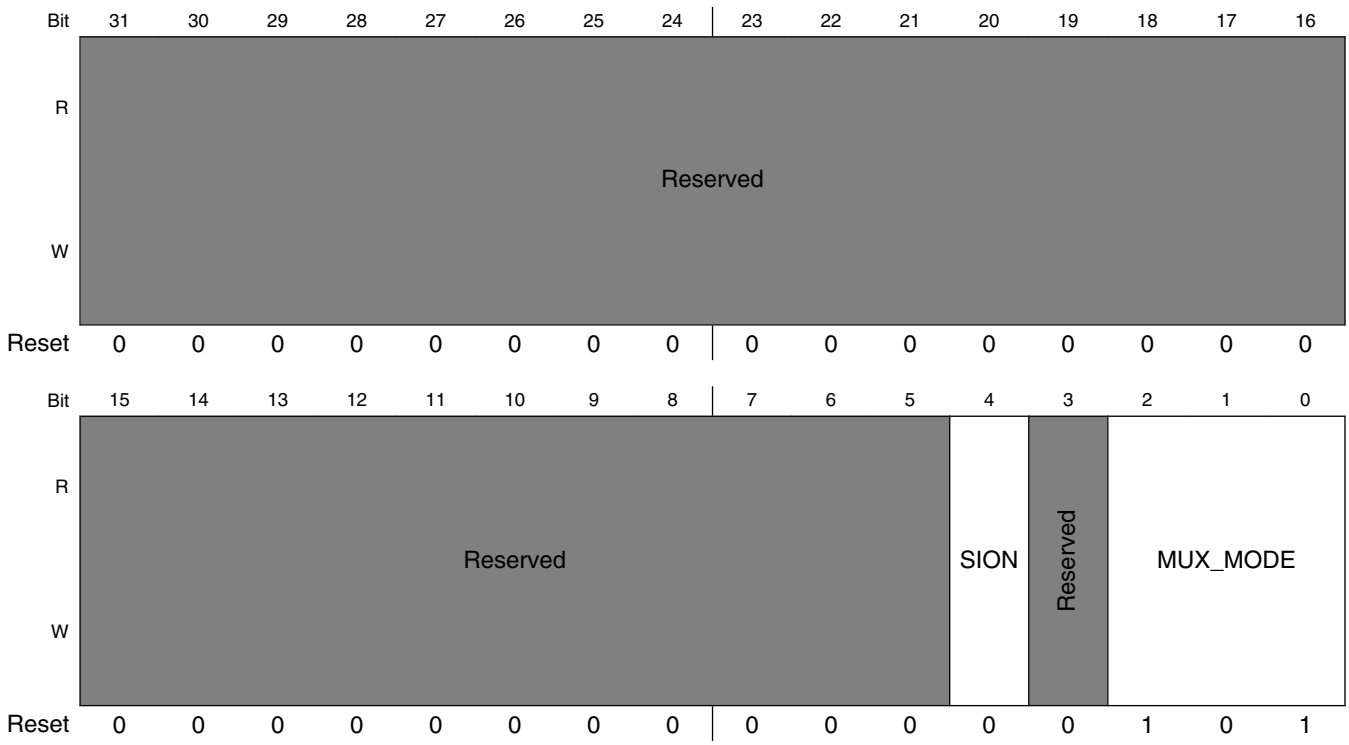
IOMUXC_SW_MUX_CTL_PAD_EPDC_PWR_COM field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_PWR_COM 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_PWR_COM. 000 **ALT0_EPDC_PWR_COM** — Select mux mode: ALT0 mux port: PWR_COM of instance: EPDC 001 **ALT1_FLEXTIMER2_PHA** — Select mux mode: ALT1 mux port: PHA of instance: FLEXTIMER2 010 **ALT2_ENET2_CRS** — Select mux mode: ALT2 mux port: CRS of instance: ENET2 100 **ALT4_EIM_AD9** — Select mux mode: ALT4 mux port: AD9 of instance: EIM 101 **ALT5_GPIO2_IO30** — Select mux mode: ALT5 mux port: IO30 of instance: GPIO2 110 **ALT6_LCD_HSYNC** — Select mux mode: ALT6 mux port: HSYNC of instance: LCD 111 **ALT7_LCD_DATA11** — Select mux mode: ALT7 mux port: DATA11 of instance: LCD

8.2.7.40 SW_MUX_CTL_PAD_EPDC_PWR_STAT SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_EPDC_PWR_STAT)

SW_MUX_CTL Register

Address: 3033_0000h base + B0h offset = 3033_00B0h



IOMUXC_SW_MUX_CTL_PAD_EPDC_PWR_STAT field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad EPDC_PWR_STAT 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_PWR_STAT. 000 **ALT0_EPDC_PWR_STAT** — Select mux mode: ALT0 mux port: PWR_STAT of instance: EPDC 001 **ALT1_FLEXTIMER2_PHB** — Select mux mode: ALT1 mux port: PHB of instance: FLEXTIMER2 010 **ALT2_ENET2_COL** — Select mux mode: ALT2 mux port: COL of instance: ENET2

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_EPDC_PWR_STAT field descriptions (continued)

Field	Description
	100 **ALT4_EIM_EB_B1** — Select mux mode: ALT4 mux port: EB_B1 of instance: EIM 101 **ALT5_GPIO2_IO31** — Select mux mode: ALT5 mux port: IO31 of instance: GPIO2 110 **ALT6_LCD_VSYNC** — Select mux mode: ALT6 mux port: VSYNC of instance: LCD 111 **ALT7_LCD_DATA12** — Select mux mode: ALT7 mux port: DATA12 of instance: LCD

8.2.7.41 SW_MUX_CTL_PAD_LCD_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_CLK)

SW_MUX_CTL Register

Address: 3033_0000h base + B4h offset = 3033_00B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved											SION	Reserved	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_LCD_CLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_CLK 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_LCD_CLK field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_CLK. 000 **ALT0_LCD_CLK** — Select mux mode: ALT0 mux port: CLK of instance: LCD 001 **ALT1_ECSPI4_MISO** — Select mux mode: ALT1 mux port: MISO of instance: ECSPI4 010 **ALT2_ENET1_1588_EVENT2_IN** — Select mux mode: ALT2 mux port: 1588_EVENT2_IN of instance: ENET1 011 **ALT3_CSI_DATA16** — Select mux mode: ALT3 mux port: DATA16 of instance: CSI 100 **ALT4_UART2_RX_DATA** — Select mux mode: ALT4 mux port: RX_DATA of instance: UART2 101 **ALT5_GPIO3_IO0** — Select mux mode: ALT5 mux port: IO0 of instance: GPIO3

8.2.7.42 SW_MUX_CTL_PAD_LCD_ENABLE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_ENABLE)

SW_MUX_CTL Register

Address: 3033_0000h base + B8h offset = 3033_00B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

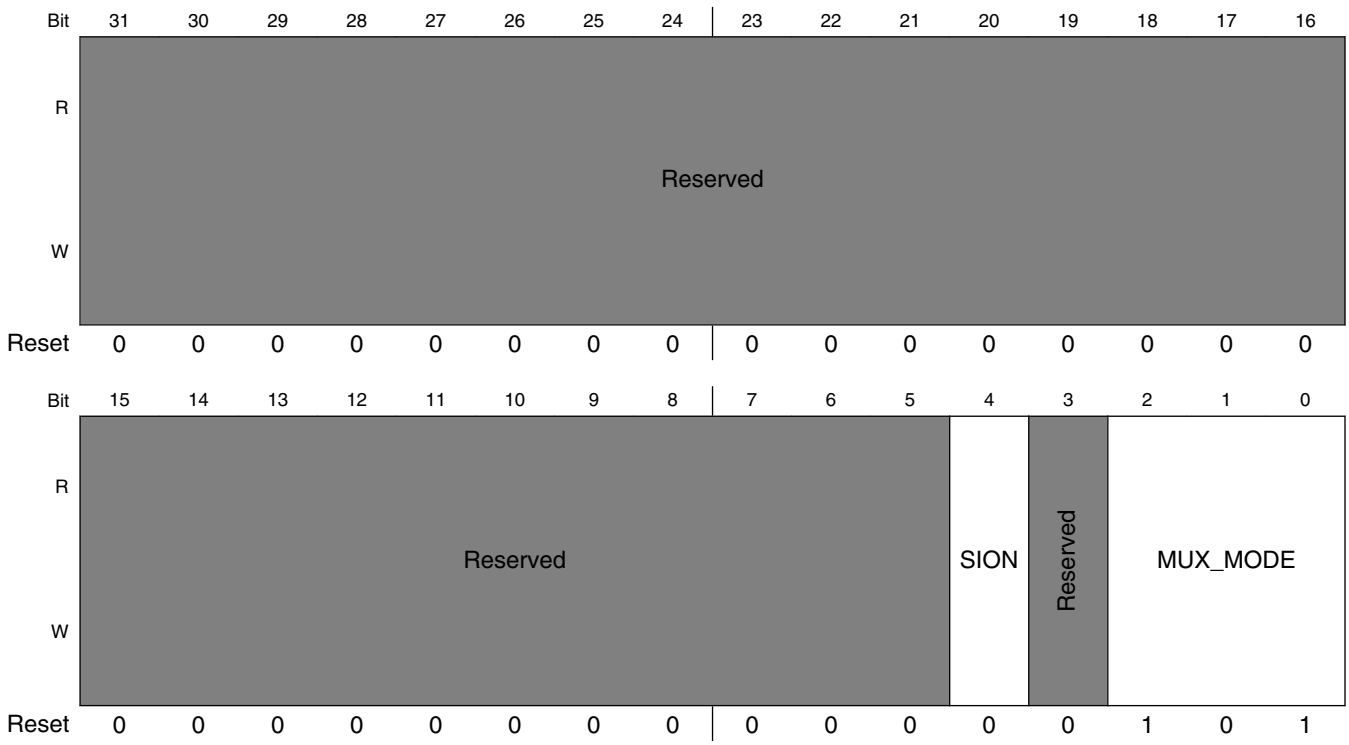
IOMUXC_SW_MUX_CTL_PAD_LCD_ENABLE field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_ENABLE 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_ENABLE. 000 **ALT0_LCD_ENABLE** — Select mux mode: ALT0 mux port: ENABLE of instance: LCD 001 **ALT1_ECSPI4_MOSI** — Select mux mode: ALT1 mux port: MOSI of instance: ECSPI4 010 **ALT2_ENET1_1588_EVENT3_IN** — Select mux mode: ALT2 mux port: 1588_EVENT3_IN of instance: ENET1 011 **ALT3_CSI_DATA17** — Select mux mode: ALT3 mux port: DATA17 of instance: CSI 100 **ALT4_UART2_TX_DATA** — Select mux mode: ALT4 mux port: TX_DATA of instance: UART2 101 **ALT5_GPIO3_IO1** — Select mux mode: ALT5 mux port: IO1 of instance: GPIO3

8.2.7.43 SW_MUX_CTL_PAD_LCD_HSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_HSYNC)

SW_MUX_CTL Register

Address: 3033_0000h base + BCh offset = 3033_00BCh



IOMUXC_SW_MUX_CTL_PAD_LCD_HSYNC field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_HSYNC 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_HSYNC. 000 **ALT0_LCD_HSYNC** — Select mux mode: ALT0 mux port: HSYNC of instance: LCD 001 **ALT1_ECSPI4_SCLK** — Select mux mode: ALT1 mux port: SCLK of instance: ECSPI4

Table continues on the next page...

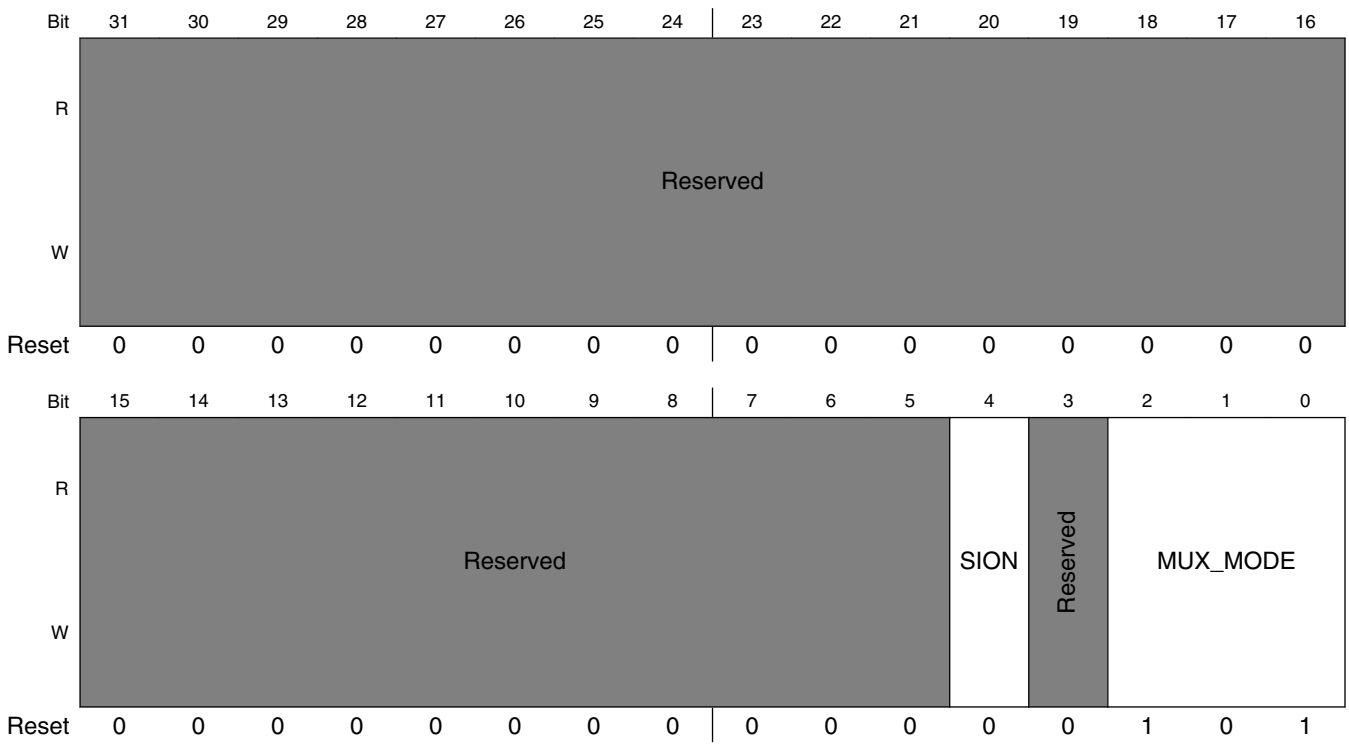
IOMUXC_SW_MUX_CTL_PAD_LCD_HSYNC field descriptions (continued)

Field	Description
	010 **ALT2_ENET2_1588_EVENT2_IN** — Select mux mode: ALT2 mux port: 1588_EVENT2_IN of instance: ENET2 011 **ALT3_CSI_DATA18** — Select mux mode: ALT3 mux port: DATA18 of instance: CSI 100 **ALT4_UART2_RTS_B** — Select mux mode: ALT4 mux port: RTS_B of instance: UART2 101 **ALT5_GPIO3_IO2** — Select mux mode: ALT5 mux port: IO2 of instance: GPIO3

8.2.7.44 SW_MUX_CTL_PAD_LCD_VSYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_VSYNC)

SW_MUX_CTL Register

Address: 3033_0000h base + C0h offset = 3033_00C0h



IOMUXC_SW_MUX_CTL_PAD_LCD_VSYNC field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_LCD_VSYNC field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad LCD_VSYNC 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_VSYNC. 000 **ALT0_LCD_VSYNC** — Select mux mode: ALT0 mux port: VSYNC of instance: LCD 001 **ALT1_ECSPI4_SS0** — Select mux mode: ALT1 mux port: SS0 of instance: ECSPI4 010 **ALT2_ENET2_1588_EVENT3_IN** — Select mux mode: ALT2 mux port: 1588_EVENT3_IN of instance: ENET2 011 **ALT3_CSI_DATA19** — Select mux mode: ALT3 mux port: DATA19 of instance: CSI 100 **ALT4_UART2_CTS_B** — Select mux mode: ALT4 mux port: CTS_B of instance: UART2 101 **ALT5_GPIO3_IO3** — Select mux mode: ALT5 mux port: IO3 of instance: GPIO3

8.2.7.45 SW_MUX_CTL_PAD_LCD_RESET SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_RESET)

SW_MUX_CTL Register

Address: 3033_0000h base + C4h offset = 3033_00C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

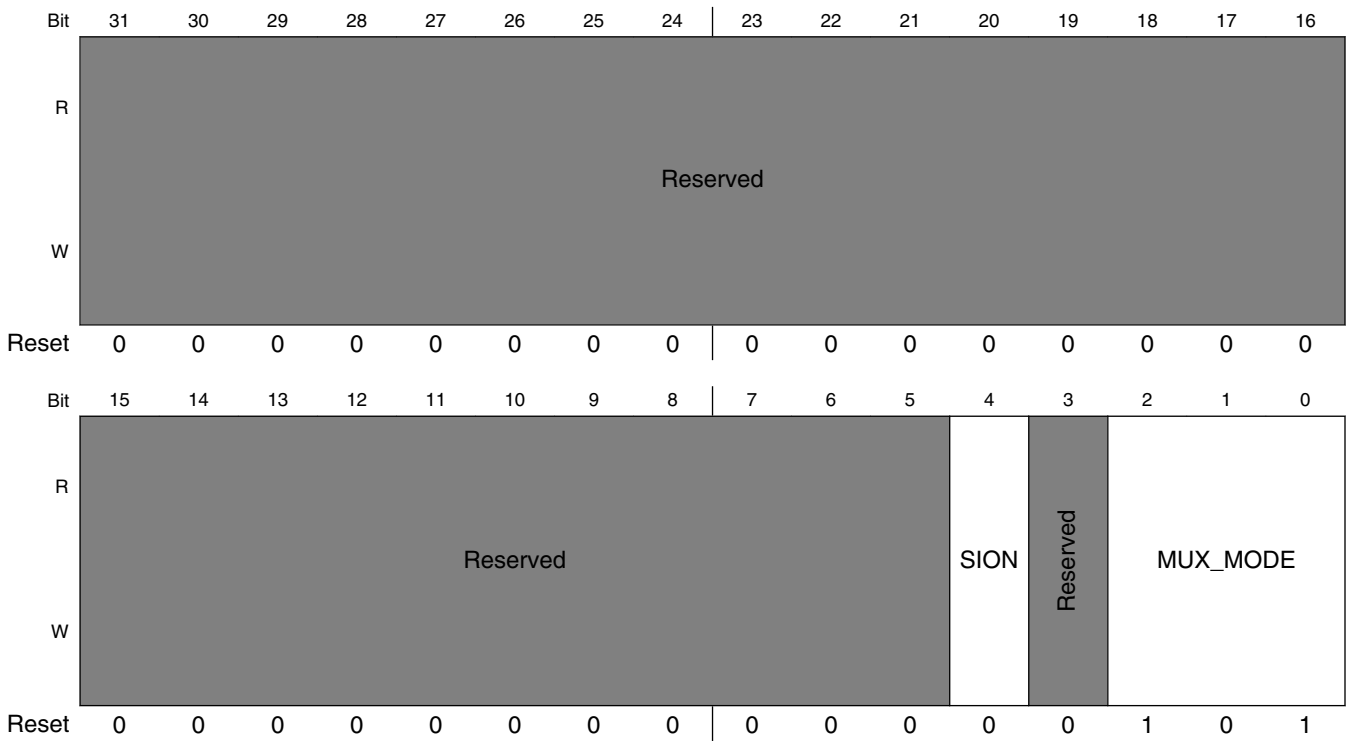
IOMUXC_SW_MUX_CTL_PAD_LCD_RESET field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_RESET 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_RESET. 000 **ALT0_LCD_RESET** — Select mux mode: ALT0 mux port: RESET of instance: LCD 001 **ALT1_GPT1_COMPARE1** — Select mux mode: ALT1 mux port: COMPARE1 of instance: GPT1 010 **ALT2_ARM_PLATFORM_EVENTI** — Select mux mode: ALT2 mux port: EVENTI of instance: ARM_PLATFORM 011 **ALT3_CSI_FIELD** — Select mux mode: ALT3 mux port: FIELD of instance: CSI 100 **ALT4_EIM_DTACK_B** — Select mux mode: ALT4 mux port: DTACK_B of instance: EIM 101 **ALT5_GPIO3_IO4** — Select mux mode: ALT5 mux port: IO4 of instance: GPIO3

8.2.7.46 SW_MUX_CTL_PAD_LCD_DATA00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA00)

SW_MUX_CTL Register

Address: 3033_0000h base + C8h offset = 3033_00C8h



IOMUXC_SW_MUX_CTL_PAD_LCD_DATA00 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA00 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LCD_DATA00. 000 **ALT0_LCD_DATA0** — Select mux mode: ALT0 mux port: DATA0 of instance: LCD 001 **ALT1_GPT1_COMPARE2** — Select mux mode: ALT1 mux port: COMPARE2 of instance: GPT1 011 **ALT3_CSI_DATA20** — Select mux mode: ALT3 mux port: DATA20 of instance: CSI

Table continues on the next page...

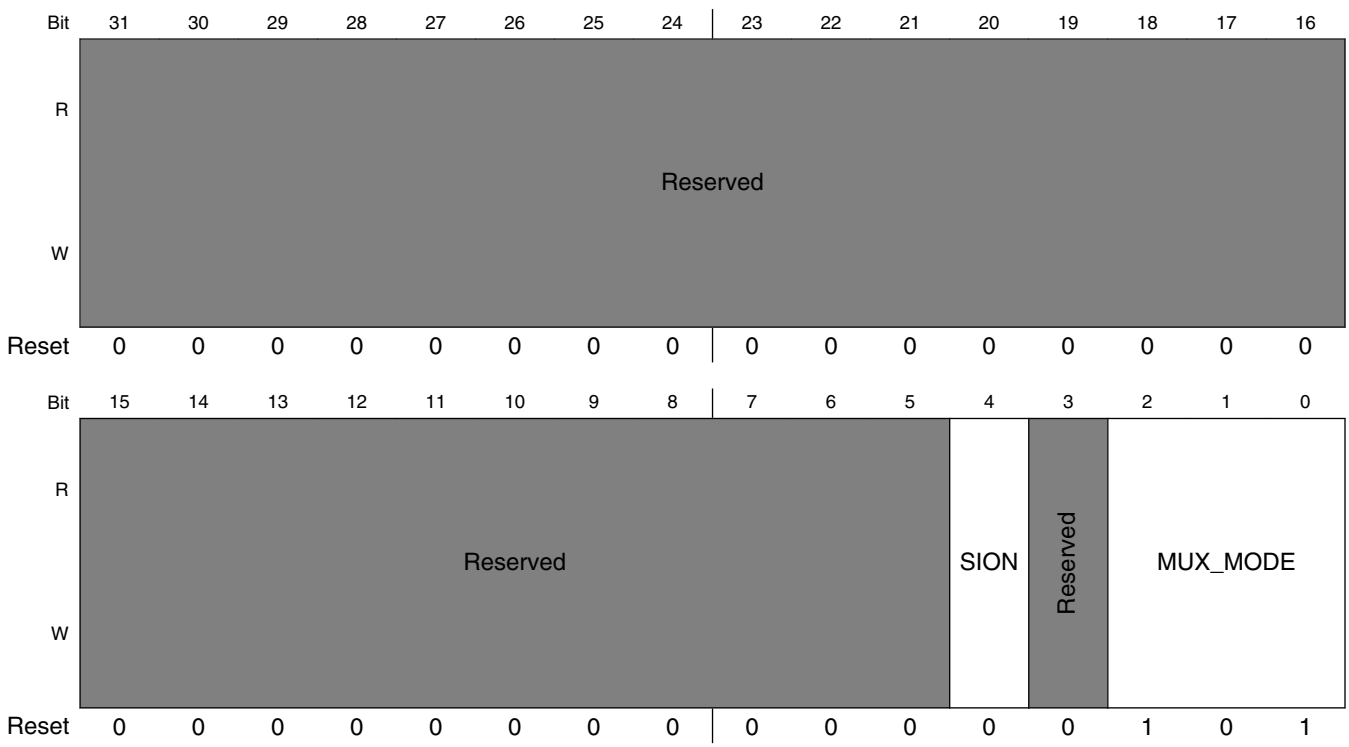
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA00 field descriptions (continued)

Field	Description
	100 **ALT4_EIM_DATA0** — Select mux mode: ALT4 mux port: DATA0 of instance: EIM 101 **ALT5_GPIO3_IO5** — Select mux mode: ALT5 mux port: IO5 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG0** — Select mux mode: ALT6 mux port: BOOT_CFG0 of instance: SRC

8.2.7.47 SW_MUX_CTL_PAD_LCD_DATA01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA01)

SW_MUX_CTL Register

Address: 3033_0000h base + CCh offset = 3033_00CCh



IOMUXC_SW_MUX_CTL_PAD_LCD_DATA01 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA01 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA01 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LCD_DATA01. 000 **ALT0_LCD_DATA1** — Select mux mode: ALT0 mux port: DATA1 of instance: LCD 001 **ALT1_GPT1_COMPARE3** — Select mux mode: ALT1 mux port: COMPARE3 of instance: GPT1 011 **ALT3_CSI_DATA21** — Select mux mode: ALT3 mux port: DATA21 of instance: CSI 100 **ALT4_EIM_DATA1** — Select mux mode: ALT4 mux port: DATA1 of instance: EIM 101 **ALT5_GPIO3_IO6** — Select mux mode: ALT5 mux port: IO6 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG1** — Select mux mode: ALT6 mux port: BOOT_CFG1 of instance: SRC

8.2.7.48 SW_MUX_CTL_PAD_LCD_DATA02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA02)

SW_MUX_CTL Register

Address: 3033_0000h base + D0h offset = 3033_00D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

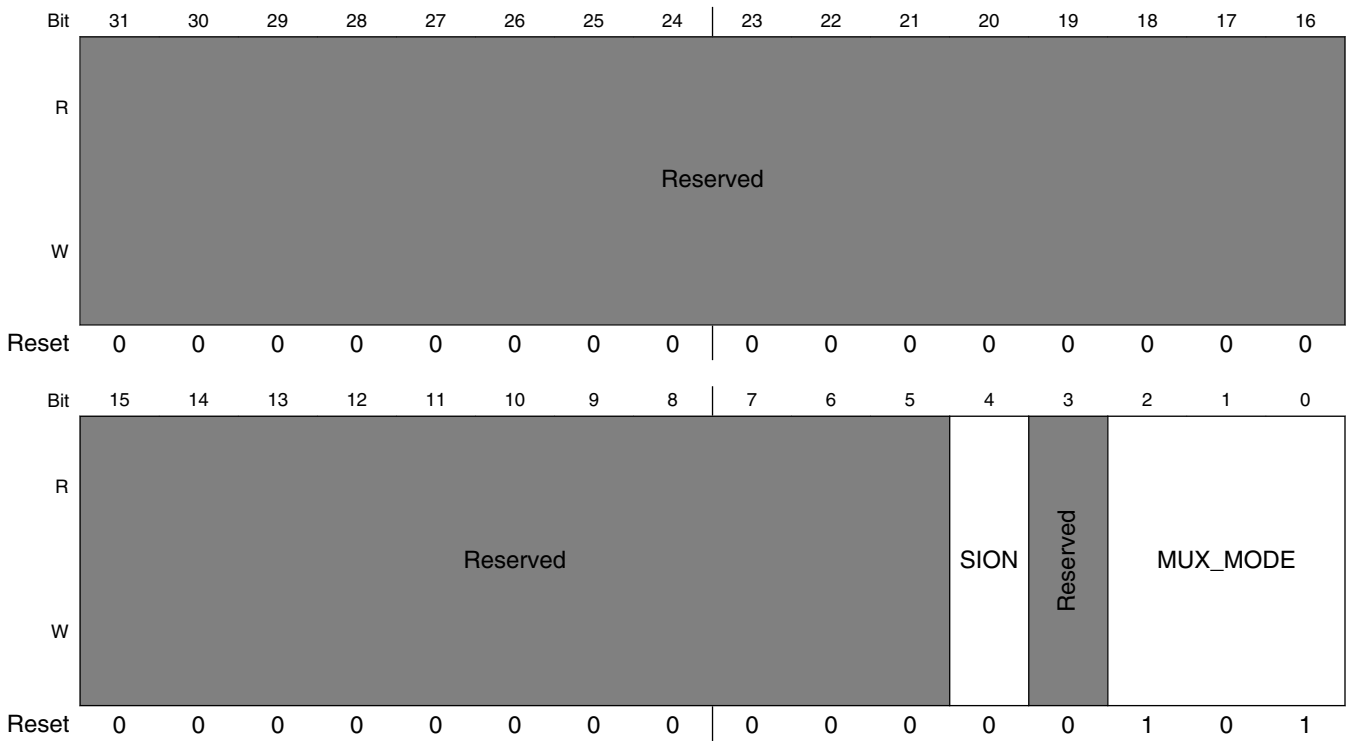
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA02 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LCD_DATA02. 000 **ALT0_LCD_DATA2** — Select mux mode: ALT0 mux port: DATA2 of instance: LCD 001 **ALT1_GPT1_CLK** — Select mux mode: ALT1 mux port: CLK of instance: GPT1 011 **ALT3_CSI_DATA22** — Select mux mode: ALT3 mux port: DATA22 of instance: CSI 100 **ALT4_EIM_DATA2** — Select mux mode: ALT4 mux port: DATA2 of instance: EIM 101 **ALT5_GPIO3_IO7** — Select mux mode: ALT5 mux port: IO7 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG2** — Select mux mode: ALT6 mux port: BOOT_CFG2 of instance: SRC

8.2.7.49 SW_MUX_CTL_PAD_LCD_DATA03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA03)

SW_MUX_CTL Register

Address: 3033_0000h base + D4h offset = 3033_00D4h



IOMUXC_SW_MUX_CTL_PAD_LCD_DATA03 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA03 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LCD_DATA03. 000 **ALT0_LCD_DATA3** — Select mux mode: ALT0 mux port: DATA3 of instance: LCD 001 **ALT1_GPT1_CAPTURE1** — Select mux mode: ALT1 mux port: CAPTURE1 of instance: GPT1 011 **ALT3_CSI_DATA23** — Select mux mode: ALT3 mux port: DATA23 of instance: CSI

Table continues on the next page...

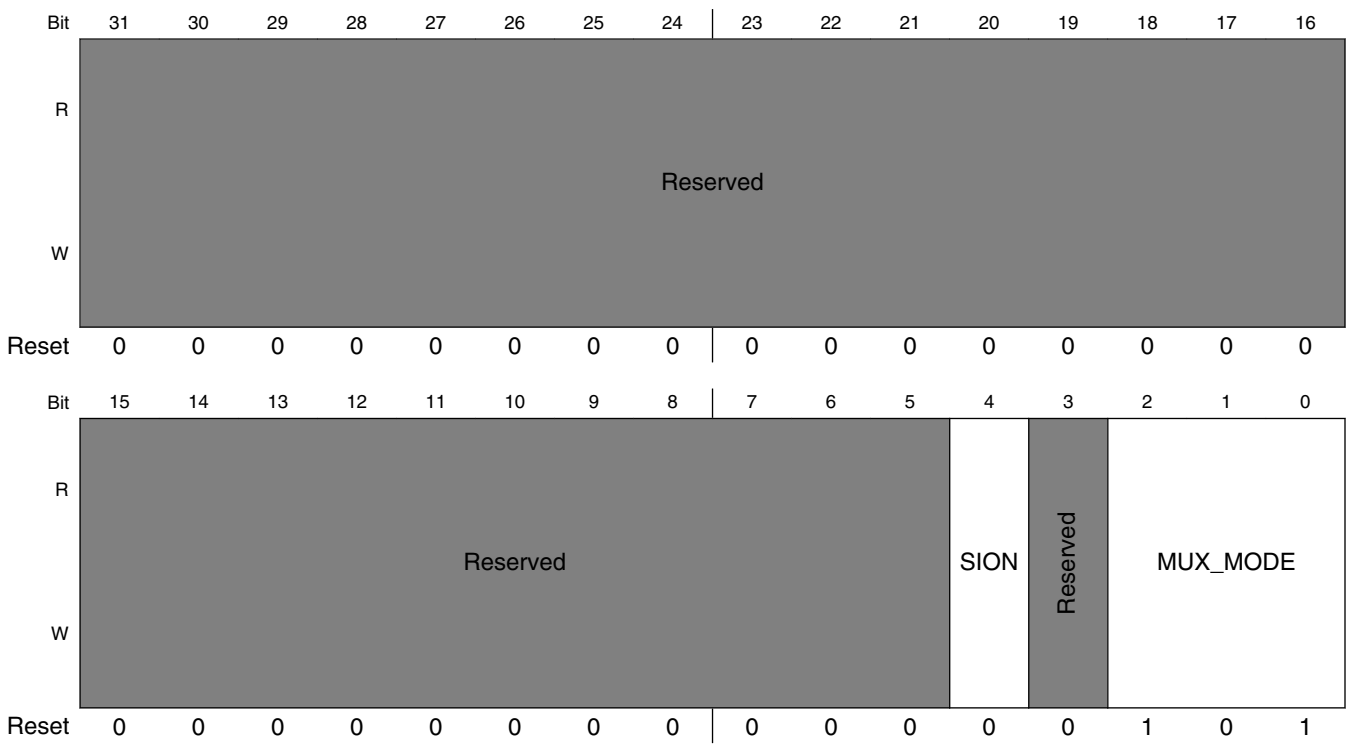
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA03 field descriptions (continued)

Field	Description
	100 **ALT4_EIM_DATA3** — Select mux mode: ALT4 mux port: DATA3 of instance: EIM 101 **ALT5_GPIO3_IO8** — Select mux mode: ALT5 mux port: IO8 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG3** — Select mux mode: ALT6 mux port: BOOT_CFG3 of instance: SRC

8.2.7.50 SW_MUX_CTL_PAD_LCD_DATA04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA04)

SW_MUX_CTL Register

Address: 3033_0000h base + D8h offset = 3033_00D8h



IOMUXC_SW_MUX_CTL_PAD_LCD_DATA04 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA04 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA04 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LCD_DATA04. 000 **ALT0_LCD_DATA4** — Select mux mode: ALT0 mux port: DATA4 of instance: LCD 001 **ALT1_GPT1_CAPTURE2** — Select mux mode: ALT1 mux port: CAPTURE2 of instance: GPT1 011 **ALT3_CSI_VSYNC** — Select mux mode: ALT3 mux port: VSYNC of instance: CSI 100 **ALT4_EIM_DATA4** — Select mux mode: ALT4 mux port: DATA4 of instance: EIM 101 **ALT5_GPIO3_IO9** — Select mux mode: ALT5 mux port: IO9 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG4** — Select mux mode: ALT6 mux port: BOOT_CFG4 of instance: SRC

8.2.7.51 SW_MUX_CTL_PAD_LCD_DATA05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA05)

SW_MUX_CTL Register

Address: 3033_0000h base + DCh offset = 3033_00DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

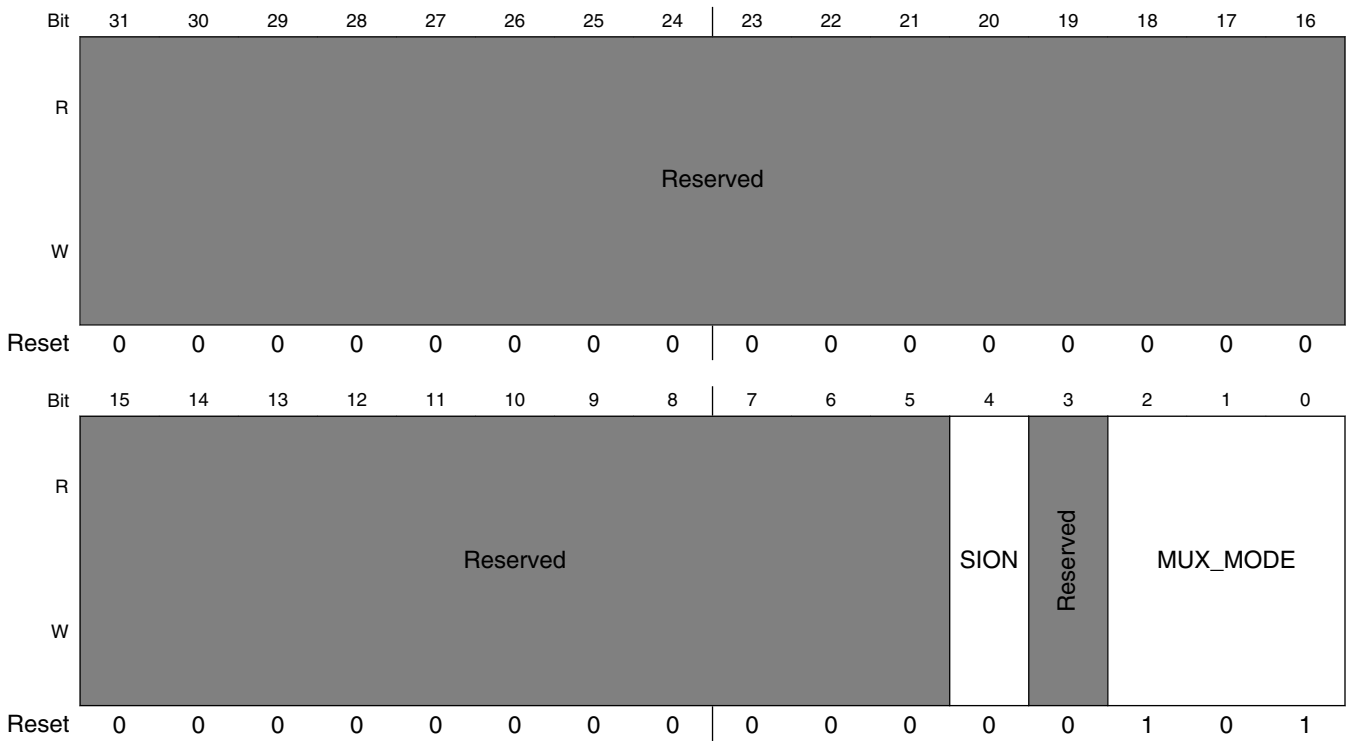
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA05 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA05 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_DATA05. 000 **ALT0_LCD_DATA5** — Select mux mode: ALT0 mux port: DATA5 of instance: LCD 011 **ALT3_CSI_HSYNC** — Select mux mode: ALT3 mux port: HSYNC of instance: CSI 100 **ALT4_EIM_DATA5** — Select mux mode: ALT4 mux port: DATA5 of instance: EIM 101 **ALT5_GPIO3_IO10** — Select mux mode: ALT5 mux port: IO10 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG5** — Select mux mode: ALT6 mux port: BOOT_CFG5 of instance: SRC

8.2.7.52 SW_MUX_CTL_PAD_LCD_DATA06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA06)

SW_MUX_CTL Register

Address: 3033_0000h base + E0h offset = 3033_00E0h



IOMUXC_SW_MUX_CTL_PAD_LCD_DATA06 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA06 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_DATA06. 000 **ALT0_LCD_DATA6** — Select mux mode: ALT0 mux port: DATA6 of instance: LCD 011 **ALT3_CSI_PIXCLK** — Select mux mode: ALT3 mux port: PIXCLK of instance: CSI 100 **ALT4_EIM_DATA6** — Select mux mode: ALT4 mux port: DATA6 of instance: EIM

Table continues on the next page...

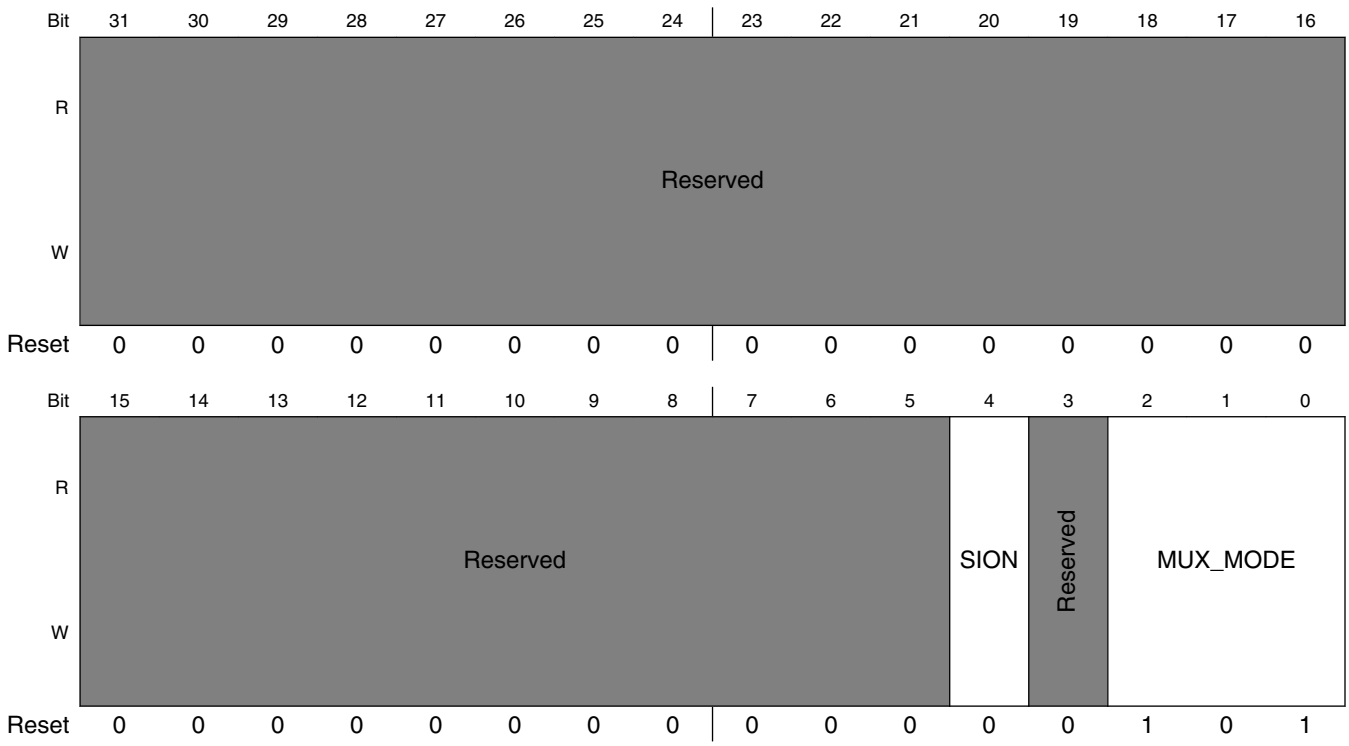
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA06 field descriptions (continued)

Field	Description
	101 **ALT5_GPIO3_IO11** — Select mux mode: ALT5 mux port: IO11 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG6** — Select mux mode: ALT6 mux port: BOOT_CFG6 of instance: SRC

8.2.7.53 SW_MUX_CTL_PAD_LCD_DATA07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA07)

SW_MUX_CTL Register

Address: 3033_0000h base + E4h offset = 3033_00E4h



IOMUXC_SW_MUX_CTL_PAD_LCD_DATA07 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA07 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA07 field descriptions (continued)

Field	Description
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_DATA07. 000 **ALT0_LCD_DATA7** — Select mux mode: ALT0 mux port: DATA7 of instance: LCD 011 **ALT3_CSI_MCLK** — Select mux mode: ALT3 mux port: MCLK of instance: CSI 100 **ALT4_EIM_DATA7** — Select mux mode: ALT4 mux port: DATA7 of instance: EIM 101 **ALT5_GPIO3_IO12** — Select mux mode: ALT5 mux port: IO12 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG7** — Select mux mode: ALT6 mux port: BOOT_CFG7 of instance: SRC

8.2.7.54 SW_MUX_CTL_PAD_LCD_DATA08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA08)

SW_MUX_CTL Register

Address: 3033_0000h base + E8h offset = 3033_00E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA08 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved

Table continues on the next page...

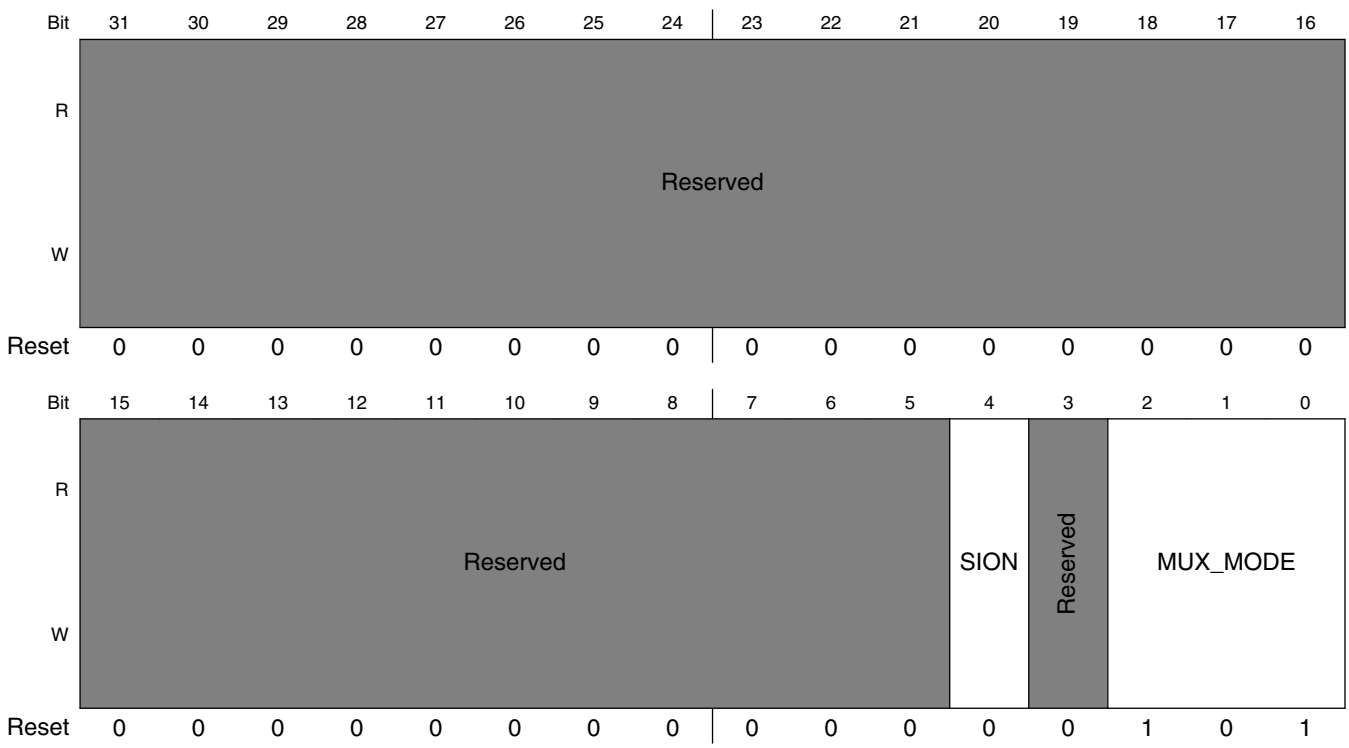
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA08 field descriptions (continued)

Field	Description
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA08 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_DATA08. 000 **ALT0_LCD_DATA8** — Select mux mode: ALT0 mux port: DATA8 of instance: LCD 011 **ALT3_CSI_DATA9** — Select mux mode: ALT3 mux port: DATA9 of instance: CSI 100 **ALT4_EIM_DATA8** — Select mux mode: ALT4 mux port: DATA8 of instance: EIM 101 **ALT5_GPIO3_IO13** — Select mux mode: ALT5 mux port: IO13 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG8** — Select mux mode: ALT6 mux port: BOOT_CFG8 of instance: SRC

8.2.7.55 SW_MUX_CTL_PAD_LCD_DATA09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA09)

SW_MUX_CTL Register

Address: 3033_0000h base + ECh offset = 3033_00ECh



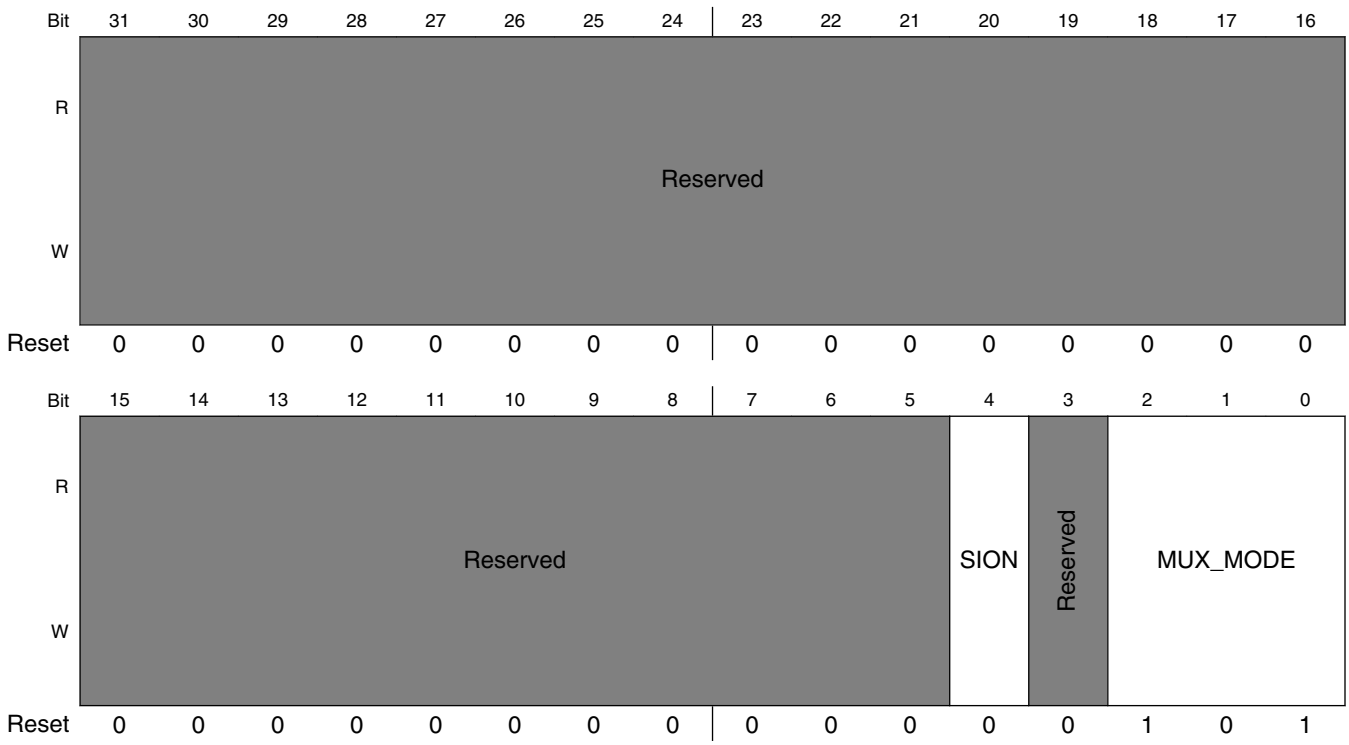
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA09 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA09 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_DATA09. 000 **ALT0_LCD_DATA9** — Select mux mode: ALT0 mux port: DATA9 of instance: LCD 011 **ALT3_CSI_DATA8** — Select mux mode: ALT3 mux port: DATA8 of instance: CSI 100 **ALT4_EIM_DATA9** — Select mux mode: ALT4 mux port: DATA9 of instance: EIM 101 **ALT5_GPIO3_IO14** — Select mux mode: ALT5 mux port: IO14 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG9** — Select mux mode: ALT6 mux port: BOOT_CFG9 of instance: SRC

8.2.7.56 SW_MUX_CTL_PAD_LCD_DATA10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA10)

SW_MUX_CTL Register

Address: 3033_0000h base + F0h offset = 3033_00F0h



IOMUXC_SW_MUX_CTL_PAD_LCD_DATA10 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA10 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_DATA10. 000 **ALT0_LCD_DATA10** — Select mux mode: ALT0 mux port: DATA10 of instance: LCD 011 **ALT3_CSI_DATA7** — Select mux mode: ALT3 mux port: DATA7 of instance: CSI 100 **ALT4_EIM_DATA10** — Select mux mode: ALT4 mux port: DATA10 of instance: EIM

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA10 field descriptions (continued)

Field	Description
	101 **ALT5_GPIO3_IO15** — Select mux mode: ALT5 mux port: IO15 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG10** — Select mux mode: ALT6 mux port: BOOT_CFG10 of instance: SRC

8.2.7.57 SW_MUX_CTL_PAD_LCD_DATA11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA11)

SW_MUX_CTL Register

Address: 3033_0000h base + F4h offset = 3033_00F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA11 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA11 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA11 field descriptions (continued)

Field	Description
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_DATA11. 000 **ALT0_LCD_DATA11** — Select mux mode: ALT0 mux port: DATA11 of instance: LCD 011 **ALT3_CSI_DATA6** — Select mux mode: ALT3 mux port: DATA6 of instance: CSI 100 **ALT4_EIM_DATA11** — Select mux mode: ALT4 mux port: DATA11 of instance: EIM 101 **ALT5_GPIO3_IO16** — Select mux mode: ALT5 mux port: IO16 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG11** — Select mux mode: ALT6 mux port: BOOT_CFG11 of instance: SRC

8.2.7.58 SW_MUX_CTL_PAD_LCD_DATA12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA12)

SW_MUX_CTL Register

Address: 3033_0000h base + F8h offset = 3033_00F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved											SION	Reserved	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA12 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA12 field descriptions (continued)

Field	Description
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA12 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_DATA12. 000 **ALT0_LCD_DATA12** — Select mux mode: ALT0 mux port: DATA12 of instance: LCD 011 **ALT3_CSI_DATA5** — Select mux mode: ALT3 mux port: DATA5 of instance: CSI 100 **ALT4_EIM_DATA12** — Select mux mode: ALT4 mux port: DATA12 of instance: EIM 101 **ALT5_GPIO3_IO17** — Select mux mode: ALT5 mux port: IO17 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG12** — Select mux mode: ALT6 mux port: BOOT_CFG12 of instance: SRC

8.2.7.59 SW_MUX_CTL_PAD_LCD_DATA13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA13)

SW_MUX_CTL Register

Address: 3033_0000h base + FCh offset = 3033_00FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

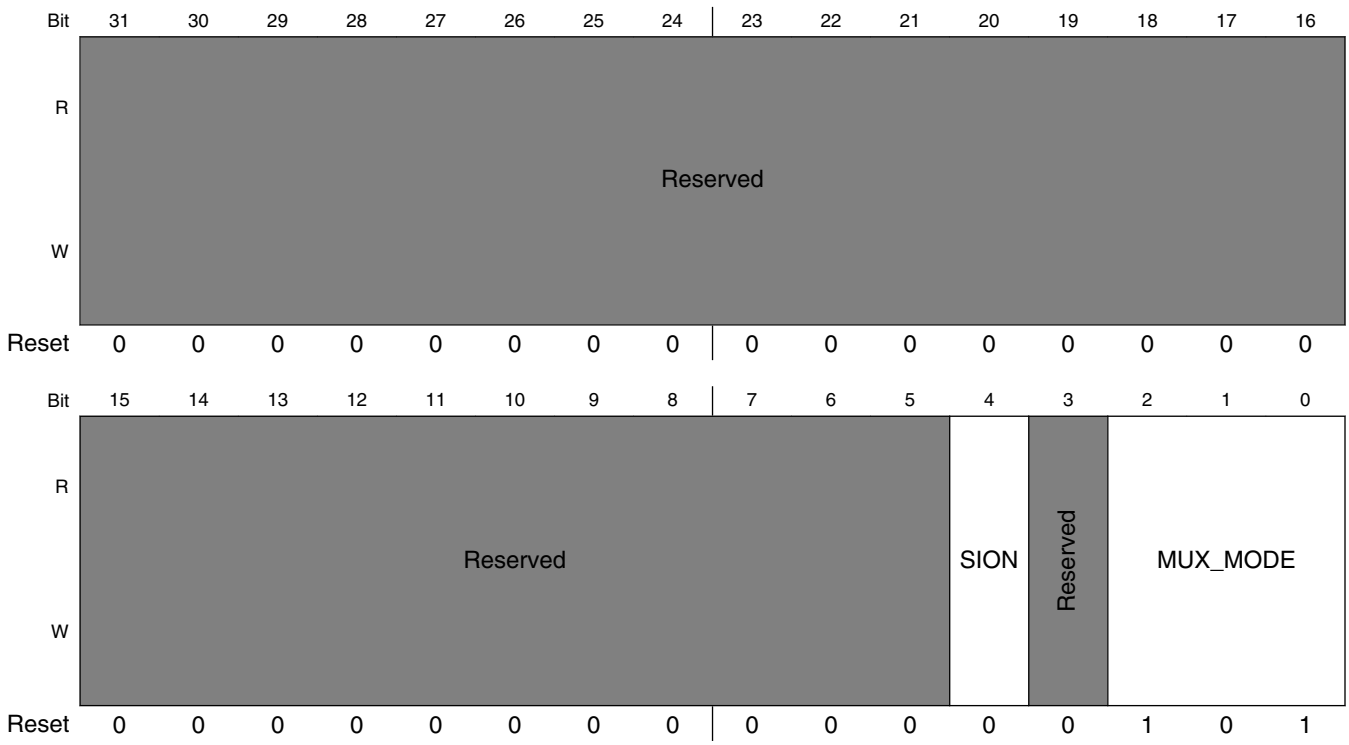
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA13 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA13 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_DATA13. 000 **ALT0_LCD_DATA13** — Select mux mode: ALT0 mux port: DATA13 of instance: LCD 011 **ALT3_CSI_DATA4** — Select mux mode: ALT3 mux port: DATA4 of instance: CSI 100 **ALT4_EIM_DATA13** — Select mux mode: ALT4 mux port: DATA13 of instance: EIM 101 **ALT5_GPIO3_IO18** — Select mux mode: ALT5 mux port: IO18 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG13** — Select mux mode: ALT6 mux port: BOOT_CFG13 of instance: SRC

8.2.7.60 SW_MUX_CTL_PAD_LCD_DATA14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA14)

SW_MUX_CTL Register

Address: 3033_0000h base + 100h offset = 3033_0100h



IOMUXC_SW_MUX_CTL_PAD_LCD_DATA14 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA14 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_DATA14. 000 **ALT0_LCD_DATA14** — Select mux mode: ALT0 mux port: DATA14 of instance: LCD 011 **ALT3_CSI_DATA3** — Select mux mode: ALT3 mux port: DATA3 of instance: CSI 100 **ALT4_EIM_DATA14** — Select mux mode: ALT4 mux port: DATA14 of instance: EIM

Table continues on the next page...

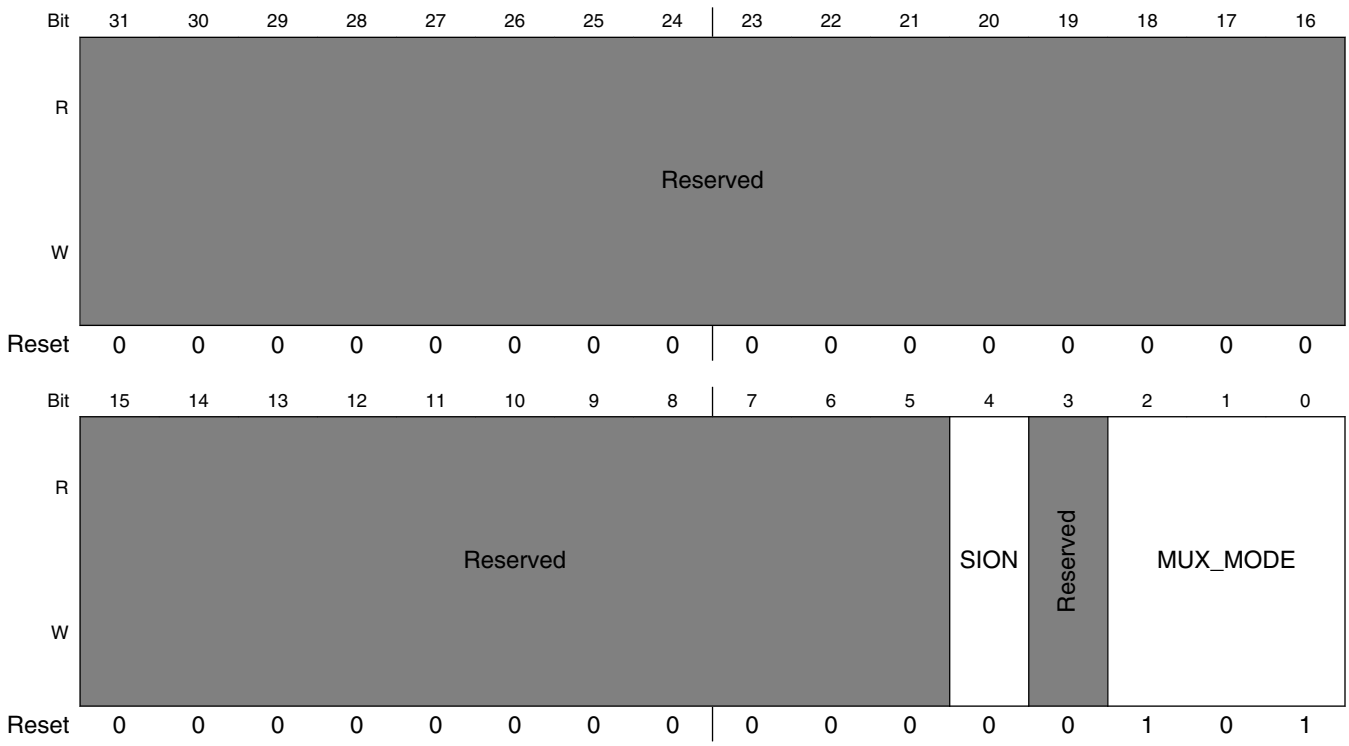
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA14 field descriptions (continued)

Field	Description
	101 **ALT5_GPIO3_IO19** — Select mux mode: ALT5 mux port: IO19 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG14** — Select mux mode: ALT6 mux port: BOOT_CFG14 of instance: SRC

8.2.7.61 SW_MUX_CTL_PAD_LCD_DATA15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA15)

SW_MUX_CTL Register

Address: 3033_0000h base + 104h offset = 3033_0104h



IOMUXC_SW_MUX_CTL_PAD_LCD_DATA15 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA15 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved

Table continues on the next page...

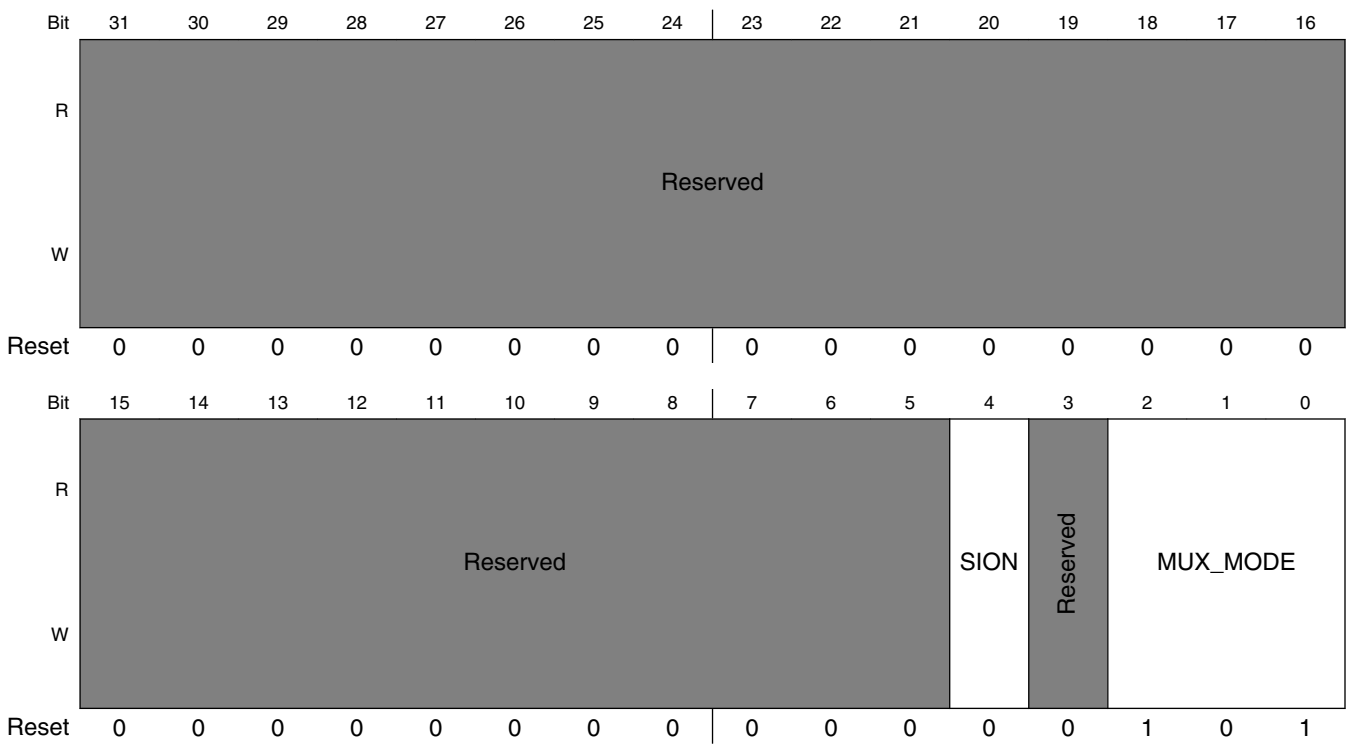
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA15 field descriptions (continued)

Field	Description
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_DATA15. 000 **ALT0_LCD_DATA15** — Select mux mode: ALT0 mux port: DATA15 of instance: LCD 011 **ALT3_CSI_DATA2** — Select mux mode: ALT3 mux port: DATA2 of instance: CSI 100 **ALT4_EIM_DATA15** — Select mux mode: ALT4 mux port: DATA15 of instance: EIM 101 **ALT5_GPIO3_IO20** — Select mux mode: ALT5 mux port: IO20 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG15** — Select mux mode: ALT6 mux port: BOOT_CFG15 of instance: SRC

8.2.7.62 SW_MUX_CTL_PAD_LCD_DATA16 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA16)

SW_MUX_CTL Register

Address: 3033_0000h base + 108h offset = 3033_0108h



IOMUXC_SW_MUX_CTL_PAD_LCD_DATA16 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved

Table continues on the next page...

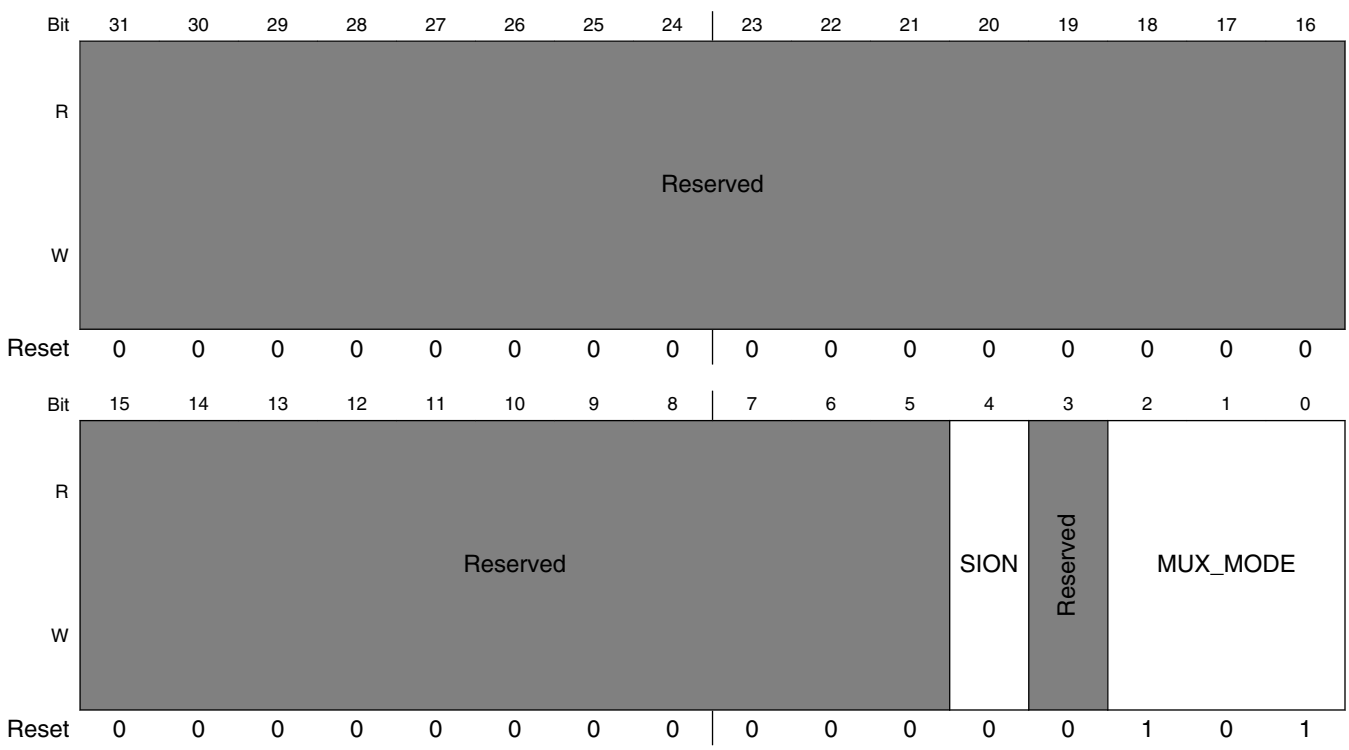
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA16 field descriptions (continued)

Field	Description
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA16 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LCD_DATA16. 000 **ALT0_LCD_DATA16** — Select mux mode: ALT0 mux port: DATA16 of instance: LCD 001 **ALT1_FLEXTIMER1_CH4** — Select mux mode: ALT1 mux port: CH4 of instance: FLEXTIMER1 011 **ALT3_CSI_DATA1** — Select mux mode: ALT3 mux port: DATA1 of instance: CSI 100 **ALT4_EIM_CRE** — Select mux mode: ALT4 mux port: CRE of instance: EIM 101 **ALT5_GPIO3_IO21** — Select mux mode: ALT5 mux port: IO21 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG16** — Select mux mode: ALT6 mux port: BOOT_CFG16 of instance: SRC

8.2.7.63 SW_MUX_CTL_PAD_LCD_DATA17 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA17)

SW_MUX_CTL Register

Address: 3033_0000h base + 10Ch offset = 3033_010Ch



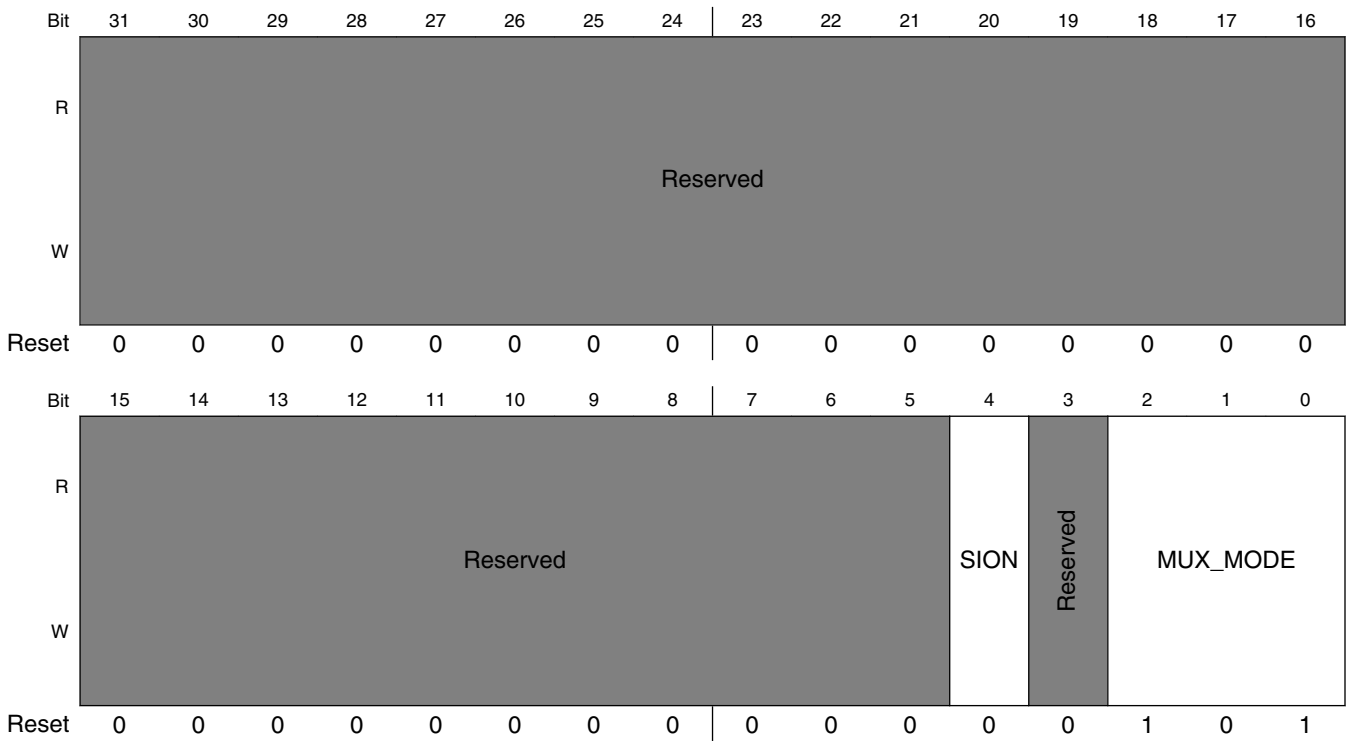
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA17 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA17 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LCD_DATA17. 000 **ALT0_LCD_DATA17** — Select mux mode: ALT0 mux port: DATA17 of instance: LCD 001 **ALT1_FLEXTIMER1_CH5** — Select mux mode: ALT1 mux port: CH5 of instance: FLEXTIMER1 011 **ALT3_CSI_DATA0** — Select mux mode: ALT3 mux port: DATA0 of instance: CSI 100 **ALT4_EIM_ACLK_FREERUN** — Select mux mode: ALT4 mux port: ACLK_FREERUN of instance: EIM 101 **ALT5_GPIO3_IO22** — Select mux mode: ALT5 mux port: IO22 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG17** — Select mux mode: ALT6 mux port: BOOT_CFG17 of instance: SRC

8.2.7.64 SW_MUX_CTL_PAD_LCD_DATA18 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA18)

SW_MUX_CTL Register

Address: 3033_0000h base + 110h offset = 3033_0110h



IOMUXC_SW_MUX_CTL_PAD_LCD_DATA18 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA18 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LCD_DATA18. 000 **ALT0_LCD_DATA18** — Select mux mode: ALT0 mux port: DATA18 of instance: LCD 001 **ALT1_FLEXTIMER1_CH6** — Select mux mode: ALT1 mux port: CH6 of instance: FLEXTIMER1

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA18 field descriptions (continued)

Field	Description
	010 **ALT2_ARM_PLATFORM_EVENTO** — Select mux mode: ALT2 mux port: EVENTO of instance: ARM_PLATFORM 011 **ALT3_CSI_DATA15** — Select mux mode: ALT3 mux port: DATA15 of instance: CSI 100 **ALT4_EIM_CS2_B** — Select mux mode: ALT4 mux port: CS2_B of instance: EIM 101 **ALT5_GPIO3_IO23** — Select mux mode: ALT5 mux port: IO23 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG18** — Select mux mode: ALT6 mux port: BOOT_CFG18 of instance: SRC

8.2.7.65 SW_MUX_CTL_PAD_LCD_DATA19 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA19)

SW_MUX_CTL Register

Address: 3033_0000h base + 114h offset = 3033_0114h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA19 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA19 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad LCD_DATA19 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LCD_DATA19. 000 **ALT0_LCD_DATA19** — Select mux mode: ALT0 mux port: DATA19 of instance: LCD 001 **ALT1_FLEXTIMER1_CH7** — Select mux mode: ALT1 mux port: CH7 of instance: FLEXTIMER1 011 **ALT3_CSI_DATA14** — Select mux mode: ALT3 mux port: DATA14 of instance: CSI 100 **ALT4_EIM_CS3_B** — Select mux mode: ALT4 mux port: CS3_B of instance: EIM 101 **ALT5_GPIO3_IO24** — Select mux mode: ALT5 mux port: IO24 of instance: GPIO3 110 **ALT6_SRC_BOOT_CFG19** — Select mux mode: ALT6 mux port: BOOT_CFG19 of instance: SRC

8.2.7.66 SW_MUX_CTL_PAD_LCD_DATA20 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA20)

SW_MUX_CTL Register

Address: 3033_0000h base + 118h offset = 3033_0118h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

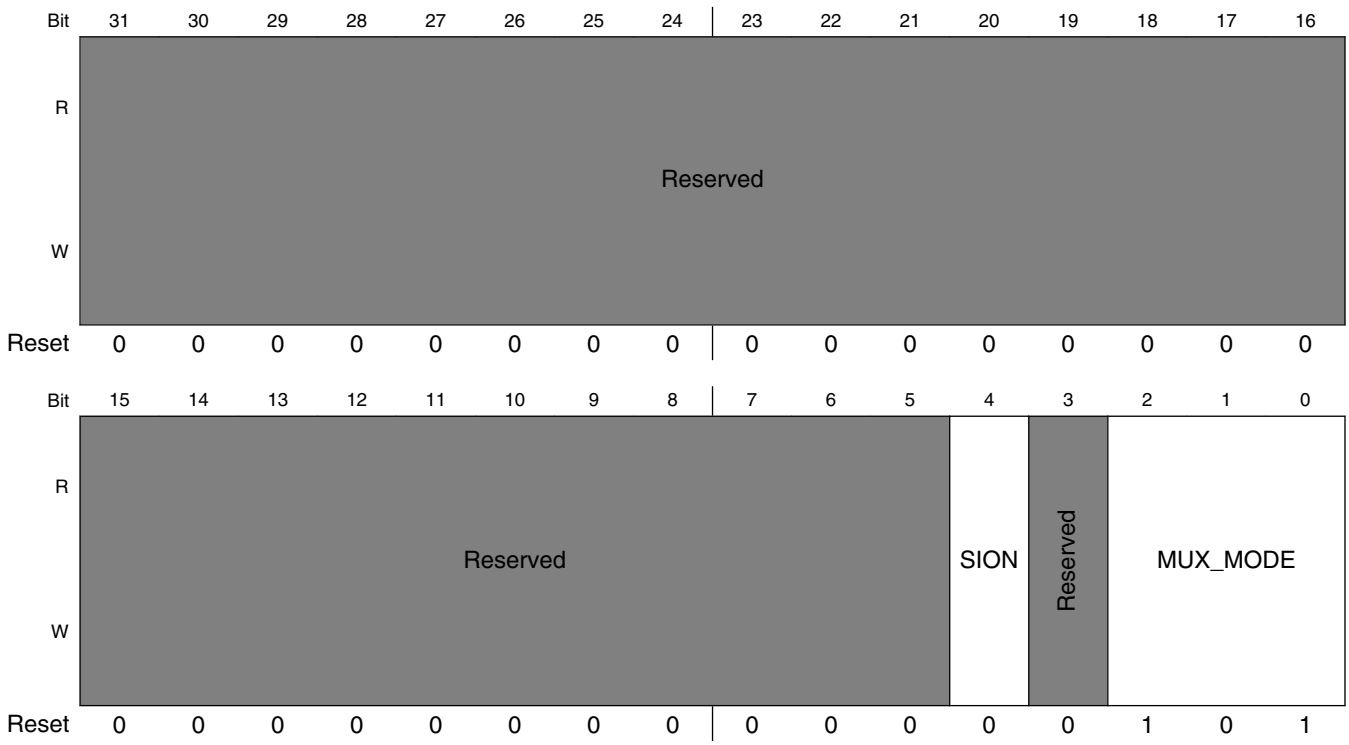
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA20 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA20 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LCD_DATA20. 000 **ALT0_LCD_DATA20** — Select mux mode: ALT0 mux port: DATA20 of instance: LCD 001 **ALT1_FLEXTIMER2_CH4** — Select mux mode: ALT1 mux port: CH4 of instance: FLEXTIMER2 010 **ALT2_ENET1_1588_EVENT2_OUT** — Select mux mode: ALT2 mux port: 1588_EVENT2_OUT of instance: ENET1 011 **ALT3_CSI_DATA13** — Select mux mode: ALT3 mux port: DATA13 of instance: CSI 100 **ALT4_EIM_ADDR23** — Select mux mode: ALT4 mux port: ADDR23 of instance: EIM 101 **ALT5_GPIO3_IO25** — Select mux mode: ALT5 mux port: IO25 of instance: GPIO3 110 **ALT6_I2C3_SCL** — Select mux mode: ALT6 mux port: SCL of instance: I2C3

8.2.7.67 SW_MUX_CTL_PAD_LCD_DATA21 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA21)

SW_MUX_CTL Register

Address: 3033_0000h base + 11Ch offset = 3033_011Ch



IOMUXC_SW_MUX_CTL_PAD_LCD_DATA21 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA21 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LCD_DATA21. 000 **ALT0_LCD_DATA21** — Select mux mode: ALT0 mux port: DATA21 of instance: LCD 001 **ALT1_FLEXTIMER2_CH5** — Select mux mode: ALT1 mux port: CH5 of instance: FLEXTIMER2

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA21 field descriptions (continued)

Field	Description
	010 **ALT2_ENET1_1588_EVENT3_OUT** — Select mux mode: ALT2 mux port: 1588_EVENT3_OUT of instance: ENET1 011 **ALT3_CSI_DATA12** — Select mux mode: ALT3 mux port: DATA12 of instance: CSI 100 **ALT4_EIM_ADDR24** — Select mux mode: ALT4 mux port: ADDR24 of instance: EIM 101 **ALT5_GPIO3_IO26** — Select mux mode: ALT5 mux port: IO26 of instance: GPIO3 110 **ALT6_I2C3_SDA** — Select mux mode: ALT6 mux port: SDA of instance: I2C3

8.2.7.68 SW_MUX_CTL_PAD_LCD_DATA22 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA22)

SW_MUX_CTL Register

Address: 3033_0000h base + 120h offset = 3033_0120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved											SION	Reserved	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_LCD_DATA22 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

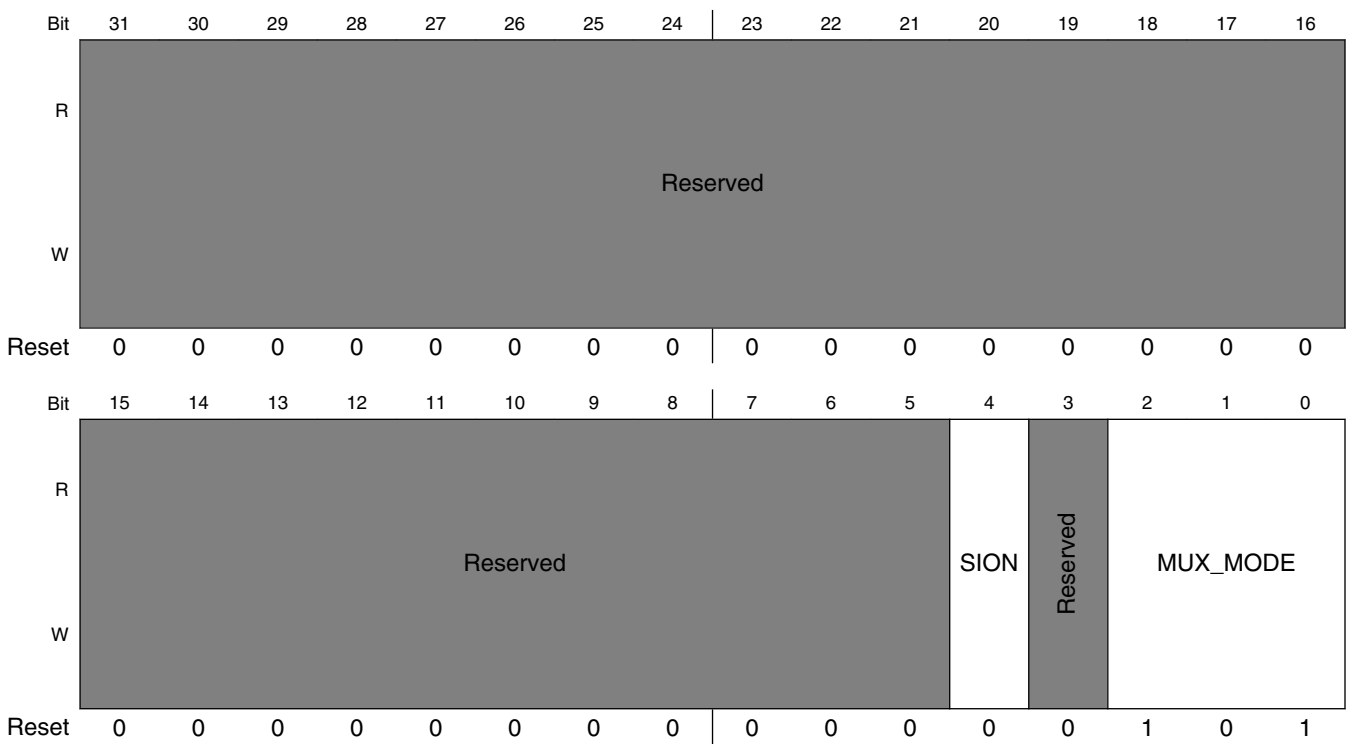
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA22 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad LCD_DATA22 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LCD_DATA22. 000 **ALT0_LCD_DATA22** — Select mux mode: ALT0 mux port: DATA22 of instance: LCD 001 **ALT1_FLEXTIMER2_CH6** — Select mux mode: ALT1 mux port: CH6 of instance: FLEXTIMER2 010 **ALT2_ENET2_1588_EVENT2_OUT** — Select mux mode: ALT2 mux port: 1588_EVENT2_OUT of instance: ENET2 011 **ALT3_CSI_DATA11** — Select mux mode: ALT3 mux port: DATA11 of instance: CSI 100 **ALT4_EIM_ADDR25** — Select mux mode: ALT4 mux port: ADDR25 of instance: EIM 101 **ALT5_GPIO3_IO27** — Select mux mode: ALT5 mux port: IO27 of instance: GPIO3 110 **ALT6_I2C4_SCL** — Select mux mode: ALT6 mux port: SCL of instance: I2C4

8.2.7.69 SW_MUX_CTL_PAD_LCD_DATA23 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_LCD_DATA23)

SW_MUX_CTL Register

Address: 3033_0000h base + 124h offset = 3033_0124h



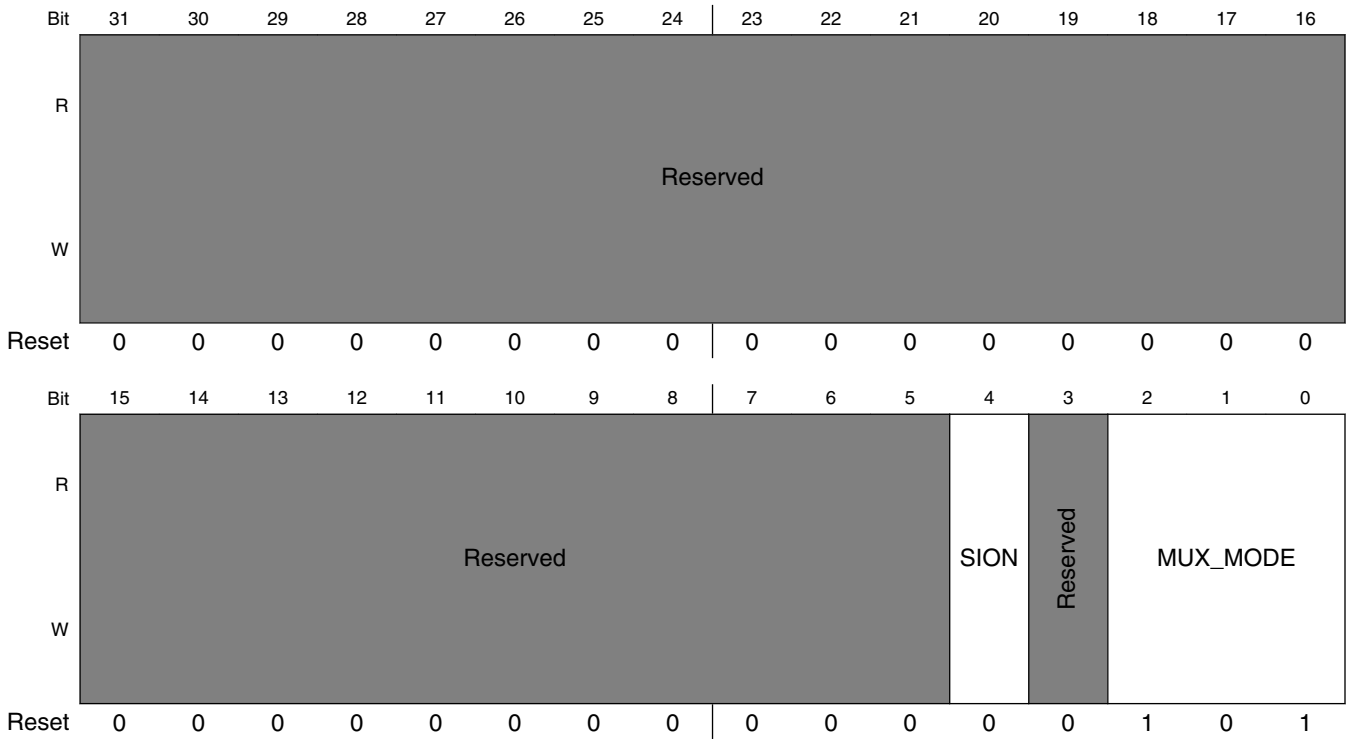
IOMUXC_SW_MUX_CTL_PAD_LCD_DATA23 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad LCD_DATA23 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LCD_DATA23. 000 **ALT0_LCD_DATA23** — Select mux mode: ALT0 mux port: DATA23 of instance: LCD 001 **ALT1_FLEXTIMER2_CH7** — Select mux mode: ALT1 mux port: CH7 of instance: FLEXTIMER2 010 **ALT2_ENET2_1588_EVENT3_OUT** — Select mux mode: ALT2 mux port: 1588_EVENT3_OUT of instance: ENET2 011 **ALT3_CSI_DATA10** — Select mux mode: ALT3 mux port: DATA10 of instance: CSI 100 **ALT4_EIM_ADDR26** — Select mux mode: ALT4 mux port: ADDR26 of instance: EIM 101 **ALT5_GPIO3_IO28** — Select mux mode: ALT5 mux port: IO28 of instance: GPIO3 110 **ALT6_I2C4_SDA** — Select mux mode: ALT6 mux port: SDA of instance: I2C4

8.2.7.70 SW_MUX_CTL_PAD_UART1_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_RX_DATA)

SW_MUX_CTL Register

Address: 3033_0000h base + 128h offset = 3033_0128h



IOMUXC_SW_MUX_CTL_PAD_UART1_RX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART1_RX_DATA 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: UART1_RX_DATA. 000 **ALT0_UART1_RX_DATA** — Select mux mode: ALT0 mux port: RX_DATA of instance: UART1 001 **ALT1_I2C1_SCL** — Select mux mode: ALT1 mux port: SCL of instance: I2C1

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_UART1_RX_DATA field descriptions (continued)

Field	Description
	010 **ALT2_CCM_PMIC_READY** — Select mux mode: ALT2 mux port: CCM_PMIC_READY of instance: CCM 011 **ALT3_ECSPI1_SS1** — Select mux mode: ALT3 mux port: SS1 of instance: ECSPI1 100 **ALT4_ENET2_1588_EVENT0_IN** — Select mux mode: ALT4 mux port: 1588_EVENT0_IN of instance: ENET2 101 **ALT5_GPIO4_IO0** — Select mux mode: ALT5 mux port: IO0 of instance: GPIO4 110 **ALT6_ENET1_MDIO** — Select mux mode: ALT6 mux port: MDIO of instance: ENET1

8.2.7.71 SW_MUX_CTL_PAD_UART1_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART1_TX_DATA)

SW_MUX_CTL Register

Address: 3033_0000h base + 12Ch offset = 3033_012Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_UART1_TX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

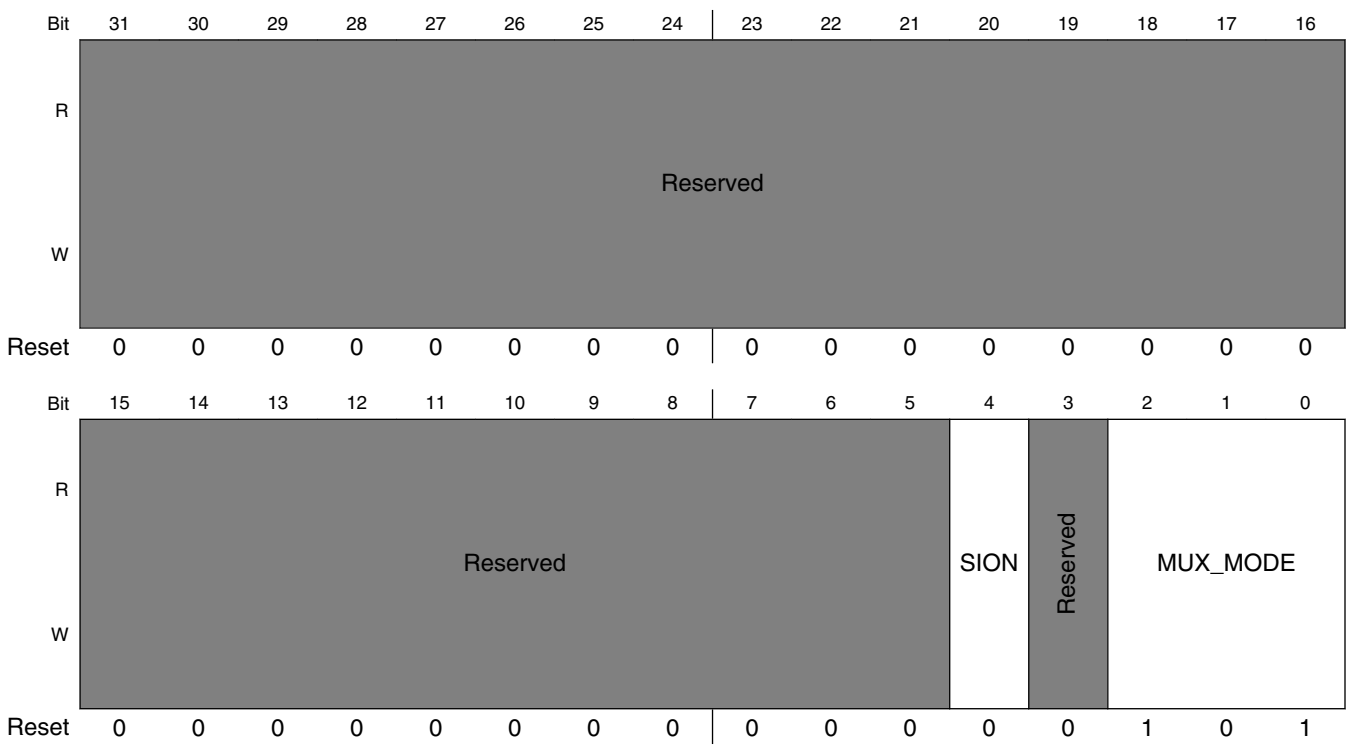
IOMUXC_SW_MUX_CTL_PAD_UART1_TX_DATA field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad UART1_TX_DATA 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: UART1_TX_DATA. 000 **ALT0_UART1_TX_DATA** — Select mux mode: ALT0 mux port: TX_DATA of instance: UART1 001 **ALT1_I2C1_SDA** — Select mux mode: ALT1 mux port: SDA of instance: I2C1 010 **ALT2_SAI3_MCLK** — Select mux mode: ALT2 mux port: MCLK of instance: SAI3 011 **ALT3_ECSPI1_SS2** — Select mux mode: ALT3 mux port: SS2 of instance: ECSPI1 100 **ALT4_ENET2_1588_EVENT0_OUT** — Select mux mode: ALT4 mux port: 1588_EVENT0_OUT of instance: ENET2 101 **ALT5_GPIO4_IO1** — Select mux mode: ALT5 mux port: IO1 of instance: GPIO4 110 **ALT6_ENET1_MDC** — Select mux mode: ALT6 mux port: MDC of instance: ENET1

8.2.7.72 SW_MUX_CTL_PAD_UART2_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_RX_DATA)

SW_MUX_CTL Register

Address: 3033_0000h base + 130h offset = 3033_0130h



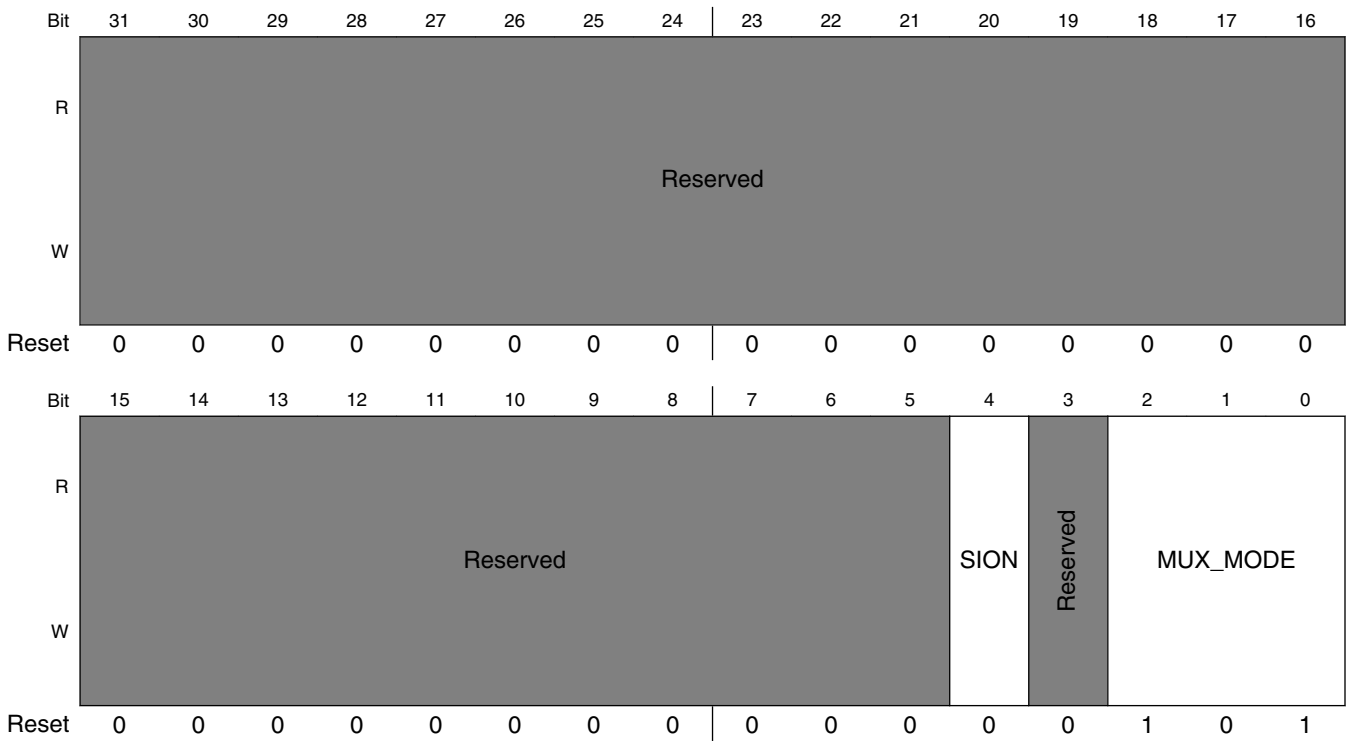
IOMUXC_SW_MUX_CTL_PAD_UART2_RX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART2_RX_DATA 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: UART2_RX_DATA. 000 **ALT0_UART2_RX_DATA** — Select mux mode: ALT0 mux port: RX_DATA of instance: UART2 001 **ALT1_I2C2_SCL** — Select mux mode: ALT1 mux port: SCL of instance: I2C2 010 **ALT2_SAI3_RX_BCLK** — Select mux mode: ALT2 mux port: RX_BCLK of instance: SAI3 011 **ALT3_ECSPI1_SS3** — Select mux mode: ALT3 mux port: SS3 of instance: ECSPI1 100 **ALT4_ENET2_1588_EVENT1_IN** — Select mux mode: ALT4 mux port: 1588_EVENT1_IN of instance: ENET2 101 **ALT5_GPIO4_IO2** — Select mux mode: ALT5 mux port: IO2 of instance: GPIO4 110 **ALT6_ENET2_MDIO** — Select mux mode: ALT6 mux port: MDIO of instance: ENET2

8.2.7.73 SW_MUX_CTL_PAD_UART2_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART2_TX_DATA)

SW_MUX_CTL Register

Address: 3033_0000h base + 134h offset = 3033_0134h



IOMUXC_SW_MUX_CTL_PAD_UART2_TX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART2_TX_DATA 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: UART2_TX_DATA. 000 **ALT0_UART2_TX_DATA** — Select mux mode: ALT0 mux port: TX_DATA of instance: UART2 001 **ALT1_I2C2_SDA** — Select mux mode: ALT1 mux port: SDA of instance: I2C2 010 **ALT2_SAI3_RX_DATA0** — Select mux mode: ALT2 mux port: RX_DATA0 of instance: SAI3

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_UART2_TX_DATA field descriptions (continued)

Field	Description
	011 **ALT3_ECSPI1_RDY** — Select mux mode: ALT3 mux port: RDY of instance: ECSPI1 100 **ALT4_ENET2_1588_EVENT1_OUT** — Select mux mode: ALT4 mux port: 1588_EVENT1_OUT of instance: ENET2 101 **ALT5_GPIO4_IO3** — Select mux mode: ALT5 mux port: IO3 of instance: GPIO4 110 **ALT6_ENET2_MDC** — Select mux mode: ALT6 mux port: MDC of instance: ENET2

8.2.7.74 SW_MUX_CTL_PAD_UART3_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_RX_DATA)

SW_MUX_CTL Register

Address: 3033_0000h base + 138h offset = 3033_0138h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_UART3_RX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_UART3_RX_DATA field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad UART3_RX_DATA 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: UART3_RX_DATA. 000 **ALT0_UART3_RX_DATA** — Select mux mode: ALT0 mux port: RX_DATA of instance: UART3 001 **ALT1_USB_OTG1_OC** — Select mux mode: ALT1 mux port: OTG1_OC of instance: USB 010 **ALT2_SAI3_RX_SYNC** — Select mux mode: ALT2 mux port: RX_SYNC of instance: SAI3 011 **ALT3_ECSPI1_MISO** — Select mux mode: ALT3 mux port: MISO of instance: ECSPI1 100 **ALT4_ENET1_1588_EVENT0_IN** — Select mux mode: ALT4 mux port: 1588_EVENT0_IN of instance: ENET1 101 **ALT5_GPIO4_IO4** — Select mux mode: ALT5 mux port: IO4 of instance: GPIO4 110 **ALT6_SD1_LCTL** — Select mux mode: ALT6 mux port: LCTL of instance: SD1

8.2.7.75 SW_MUX_CTL_PAD_UART3_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_TX_DATA)

SW_MUX_CTL Register

Address: 3033_0000h base + 13Ch offset = 3033_013Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

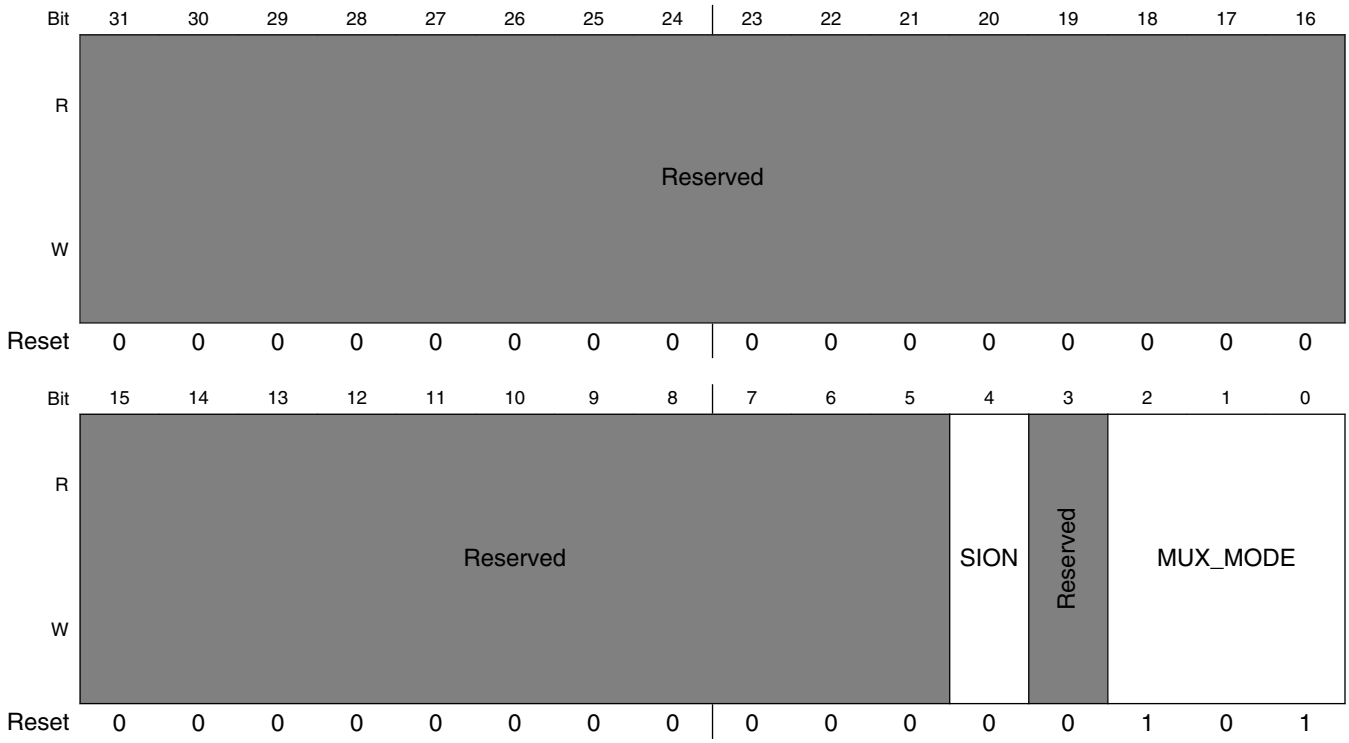
IOMUXC_SW_MUX_CTL_PAD_UART3_TX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART3_TX_DATA 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: UART3_TX_DATA. 000 **ALT0_UART3_TX_DATA** — Select mux mode: ALT0 mux port: TX_DATA of instance: UART3 001 **ALT1_USB_OTG1_PWR** — Select mux mode: ALT1 mux port: OTG1_PWR of instance: USB 010 **ALT2_SAI3_TX_BCLK** — Select mux mode: ALT2 mux port: TX_BCLK of instance: SAI3 011 **ALT3_ECSPI1_MOSI** — Select mux mode: ALT3 mux port: MOSI of instance: ECSPI1 100 **ALT4_ENET1_1588_EVENT0_OUT** — Select mux mode: ALT4 mux port: 1588_EVENT0_OUT of instance: ENET1 101 **ALT5_GPIO4_IO5** — Select mux mode: ALT5 mux port: IO5 of instance: GPIO4 110 **ALT6_SD2_LCTL** — Select mux mode: ALT6 mux port: LCTL of instance: SD2

8.2.7.76 SW_MUX_CTL_PAD_UART3_RTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_RTS_B)

SW_MUX_CTL Register

Address: 3033_0000h base + 140h offset = 3033_0140h



IOMUXC_SW_MUX_CTL_PAD_UART3_RTS_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad UART3_RTS_B 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: UART3_RTS_B. 000 **ALT0_UART3_RTS_B** — Select mux mode: ALT0 mux port: RTS_B of instance: UART3 001 **ALT1_USB_OTG2_OC** — Select mux mode: ALT1 mux port: OTG2_OC of instance: USB 010 **ALT2_SAI3_TX_DATA0** — Select mux mode: ALT2 mux port: TX_DATA0 of instance: SAI3

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_UART3_RTS_B field descriptions (continued)

Field	Description
	011 **ALT3_ECSPI1_SCLK** — Select mux mode: ALT3 mux port: SCLK of instance: ECSPI1 100 **ALT4_ENET1_1588_EVENT1_IN** — Select mux mode: ALT4 mux port: 1588_EVENT1_IN of instance: ENET1 101 **ALT5_GPIO4_IO6** — Select mux mode: ALT5 mux port: IO6 of instance: GPIO4 110 **ALT6_SD3_LCTL** — Select mux mode: ALT6 mux port: LCTL of instance: SD3

8.2.7.77 SW_MUX_CTL_PAD_UART3_CTS_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_UART3_CTS_B)

SW_MUX_CTL Register

Address: 3033_0000h base + 144h offset = 3033_0144h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_UART3_CTS_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

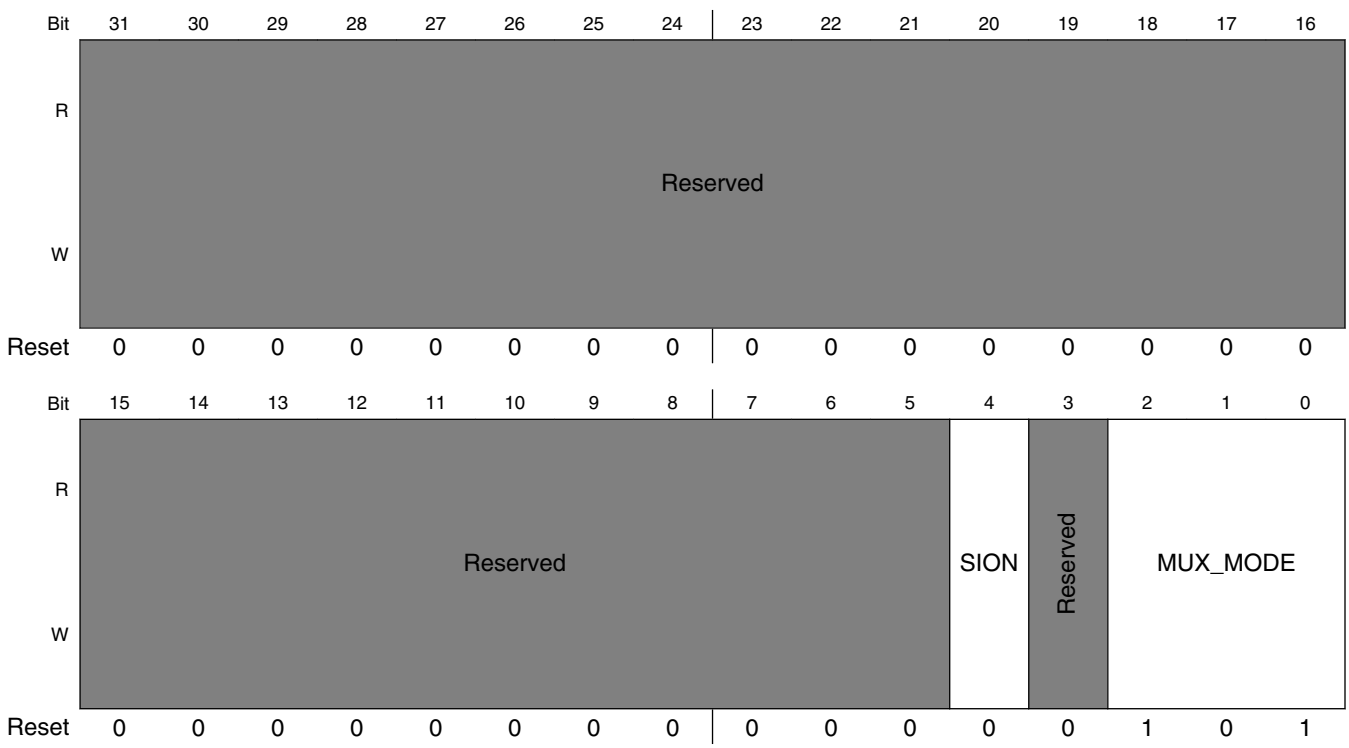
IOMUXC_SW_MUX_CTL_PAD_UART3_CTS_B field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad UART3_CTS_B 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: UART3_CTS_B. 000 **ALT0_UART3_CTS_B** — Select mux mode: ALT0 mux port: CTS_B of instance: UART3 001 **ALT1_USB_OTG2_PWR** — Select mux mode: ALT1 mux port: OTG2_PWR of instance: USB 010 **ALT2_SAI3_TX_SYNC** — Select mux mode: ALT2 mux port: TX_SYNC of instance: SAI3 011 **ALT3_ECSPI1_SS0** — Select mux mode: ALT3 mux port: SS0 of instance: ECSPI1 100 **ALT4_ENET1_1588_EVENT1_OUT** — Select mux mode: ALT4 mux port: 1588_EVENT1_OUT of instance: ENET1 101 **ALT5_GPIO4_IO7** — Select mux mode: ALT5 mux port: IO7 of instance: GPIO4 110 **ALT6_SD1_VSELECT** — Select mux mode: ALT6 mux port: VSELECT of instance: SD1

8.2.7.78 SW_MUX_CTL_PAD_I2C1_SCL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C1_SCL)

SW_MUX_CTL Register

Address: 3033_0000h base + 148h offset = 3033_0148h



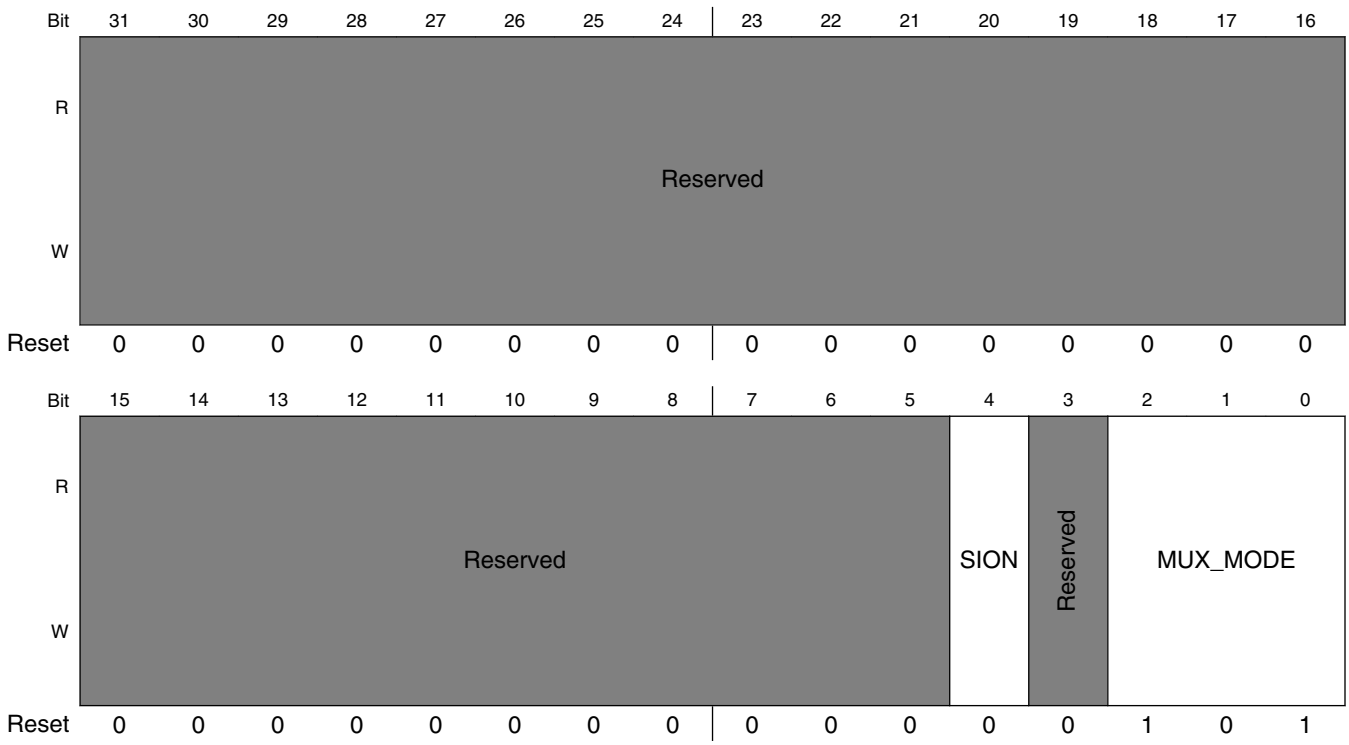
IOMUXC_SW_MUX_CTL_PAD_I2C1_SCL field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad I2C1_SCL 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: I2C1_SCL. 000 **ALT0_I2C1_SCL** — Select mux mode: ALT0 mux port: SCL of instance: I2C1 001 **ALT1_UART4_CTS_B** — Select mux mode: ALT1 mux port: CTS_B of instance: UART4 010 **ALT2_FLEXCAN1_RX** — Select mux mode: ALT2 mux port: RX of instance: FLEXCAN1 011 **ALT3_ECSPI3_MISO** — Select mux mode: ALT3 mux port: MISO of instance: ECSPI3 101 **ALT5_GPIO4_IO8** — Select mux mode: ALT5 mux port: IO8 of instance: GPIO4 110 **ALT6_SD2_VSELECT** — Select mux mode: ALT6 mux port: VSELECT of instance: SD2

8.2.7.79 SW_MUX_CTL_PAD_I2C1_SDA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C1_SDA)

SW_MUX_CTL Register

Address: 3033_0000h base + 14Ch offset = 3033_014Ch



IOMUXC_SW_MUX_CTL_PAD_I2C1_SDA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad I2C1_SDA 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: I2C1_SDA. 000 **ALT0_I2C1_SDA** — Select mux mode: ALT0 mux port: SDA of instance: I2C1 001 **ALT1_UART4_RTS_B** — Select mux mode: ALT1 mux port: RTS_B of instance: UART4 010 **ALT2_FLEXCAN1_TX** — Select mux mode: ALT2 mux port: TX of instance: FLEXCAN1

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_I2C1_SDA field descriptions (continued)

Field	Description
	011 **ALT3_ECSPI3_MOSI** — Select mux mode: ALT3 mux port: MOSI of instance: ECSPI3 100 **ALT4_CCM_ENET1_REF_CLK** — Select mux mode: ALT4 mux port: ENET1_REF_CLK of instance: ENET1 101 **ALT5_GPIO4_IO9** — Select mux mode: ALT5 mux port: IO9 of instance: GPIO4 110 **ALT6_SD3_VSELECT** — Select mux mode: ALT6 mux port: VSELECT of instance: SD3

8.2.7.80 SW_MUX_CTL_PAD_I2C2_SCL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C2_SCL)

SW_MUX_CTL Register

Address: 3033_0000h base + 150h offset = 3033_0150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_I2C2_SCL field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

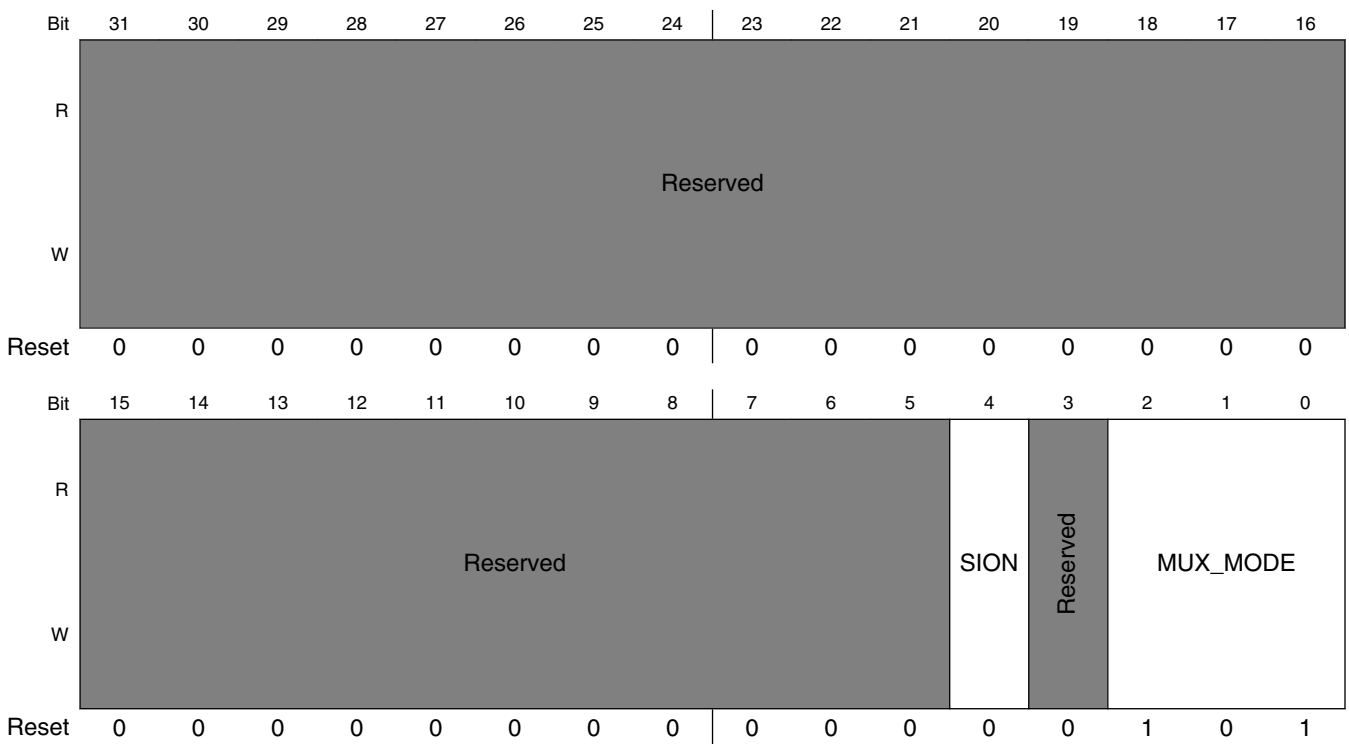
IOMUXC_SW_MUX_CTL_PAD_I2C2_SCL field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad I2C2_SCL 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: I2C2_SCL. 000 **ALT0_I2C2_SCL** — Select mux mode: ALT0 mux port: SCL of instance: I2C2 001 **ALT1_UART4_RX_DATA** — Select mux mode: ALT1 mux port: RX_DATA of instance: UART4 010 **ALT2_WDOG3_WDOG_B** — Select mux mode: ALT2 mux port: WDOG_B of instance: WDOG3 011 **ALT3_ECSPI3_SCLK** — Select mux mode: ALT3 mux port: SCLK of instance: ECSPI3 100 **ALT4_CCM_ENET2_REF_CLK** — Select mux mode: ALT4 mux port: ENET2_REF_CLK of instance: ENET2 101 **ALT5_GPIO4_IO10** — Select mux mode: ALT5 mux port: IO10 of instance: GPIO4 110 **ALT6_SD3_CD_B** — Select mux mode: ALT6 mux port: CD_B of instance: SD3

8.2.7.81 SW_MUX_CTL_PAD_I2C2_SDA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C2_SDA)

SW_MUX_CTL Register

Address: 3033_0000h base + 154h offset = 3033_0154h



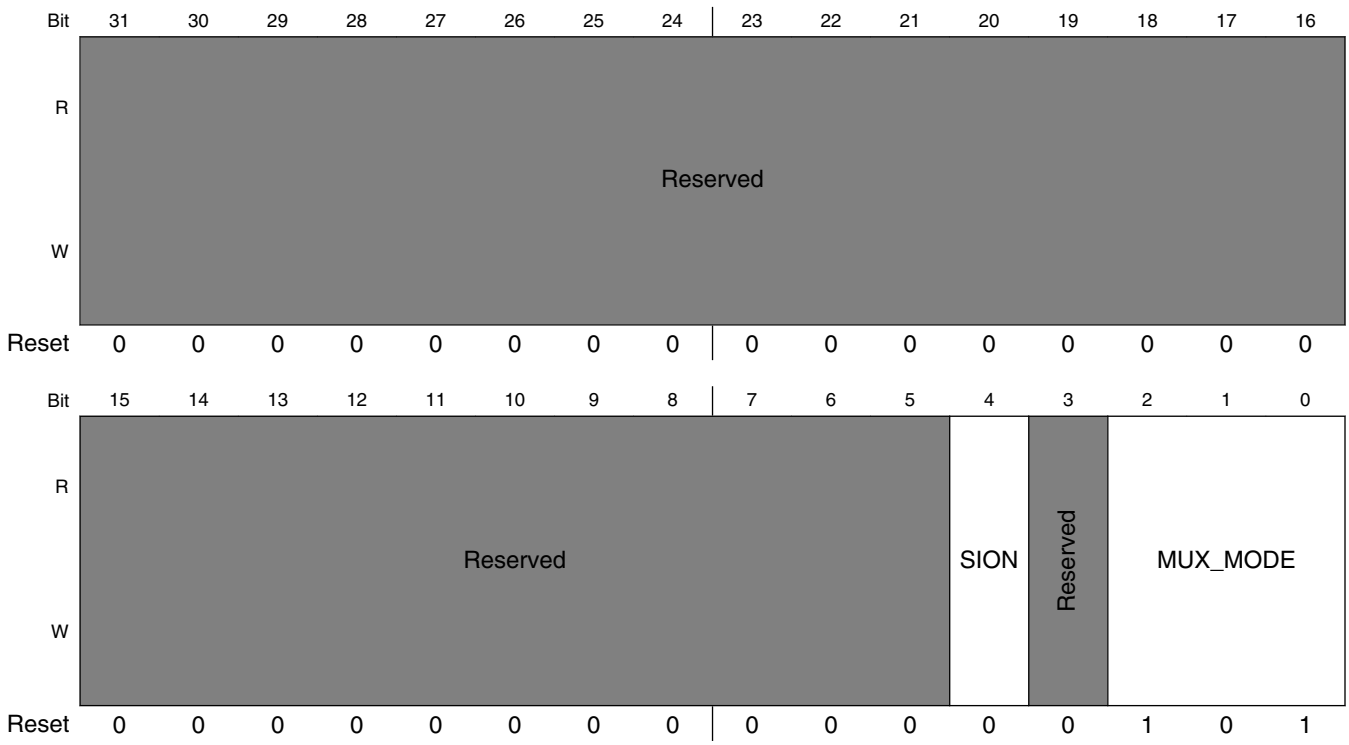
IOMUXC_SW_MUX_CTL_PAD_I2C2_SDA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad I2C2_SDA 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: I2C2_SDA. 000 **ALT0_I2C2_SDA** — Select mux mode: ALT0 mux port: SDA of instance: I2C2 001 **ALT1_UART4_TX_DATA** — Select mux mode: ALT1 mux port: TX_DATA of instance: UART4 010 **ALT2_WDOG3_WDOG_RST_B_DEB** — Select mux mode: ALT2 mux port: WDOG_RST_B_DEB of instance: WDOG3 011 **ALT3_ECSPI3_SS0** — Select mux mode: ALT3 mux port: SS0 of instance: ECSPI3 100 **ALT4_CCM_ENET_PHY_REF_CLK** — Select mux mode: ALT4 mux port: ENET_PHY_REF_CLK of instance: CCM 101 **ALT5_GPIO4_IO11** — Select mux mode: ALT5 mux port: IO11 of instance: GPIO4 110 **ALT6_SD3_WP** — Select mux mode: ALT6 mux port: WP of instance: SD3

8.2.7.82 SW_MUX_CTL_PAD_I2C3_SCL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C3_SCL)

SW_MUX_CTL Register

Address: 3033_0000h base + 158h offset = 3033_0158h



IOMUXC_SW_MUX_CTL_PAD_I2C3_SCL field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad I2C3_SCL 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: I2C3_SCL. 000 **ALT0_I2C3_SCL** — Select mux mode: ALT0 mux port: SCL of instance: I2C3 001 **ALT1_UART5_CTS_B** — Select mux mode: ALT1 mux port: CTS_B of instance: UART5 010 **ALT2_FLEXCAN2_RX** — Select mux mode: ALT2 mux port: RX of instance: FLEXCAN2

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_I2C3_SCL field descriptions (continued)

Field	Description
	011 **ALT3_CSI_VSYNC** — Select mux mode: ALT3 mux port: VSYNC of instance: CSI 100 **ALT4_SDMA_EXT_EVENT0** — Select mux mode: ALT4 mux port: EXT_EVENT0 of instance: SDMA 101 **ALT5_GPIO4_IO12** — Select mux mode: ALT5 mux port: IO12 of instance: GPIO4 110 **ALT6_EPDC_BDR0** — Select mux mode: ALT6 mux port: BDR0 of instance: EPDC

8.2.7.83 SW_MUX_CTL_PAD_I2C3_SDA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C3_SDA)

SW_MUX_CTL Register

Address: 3033_0000h base + 15Ch offset = 3033_015Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_I2C3_SDA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

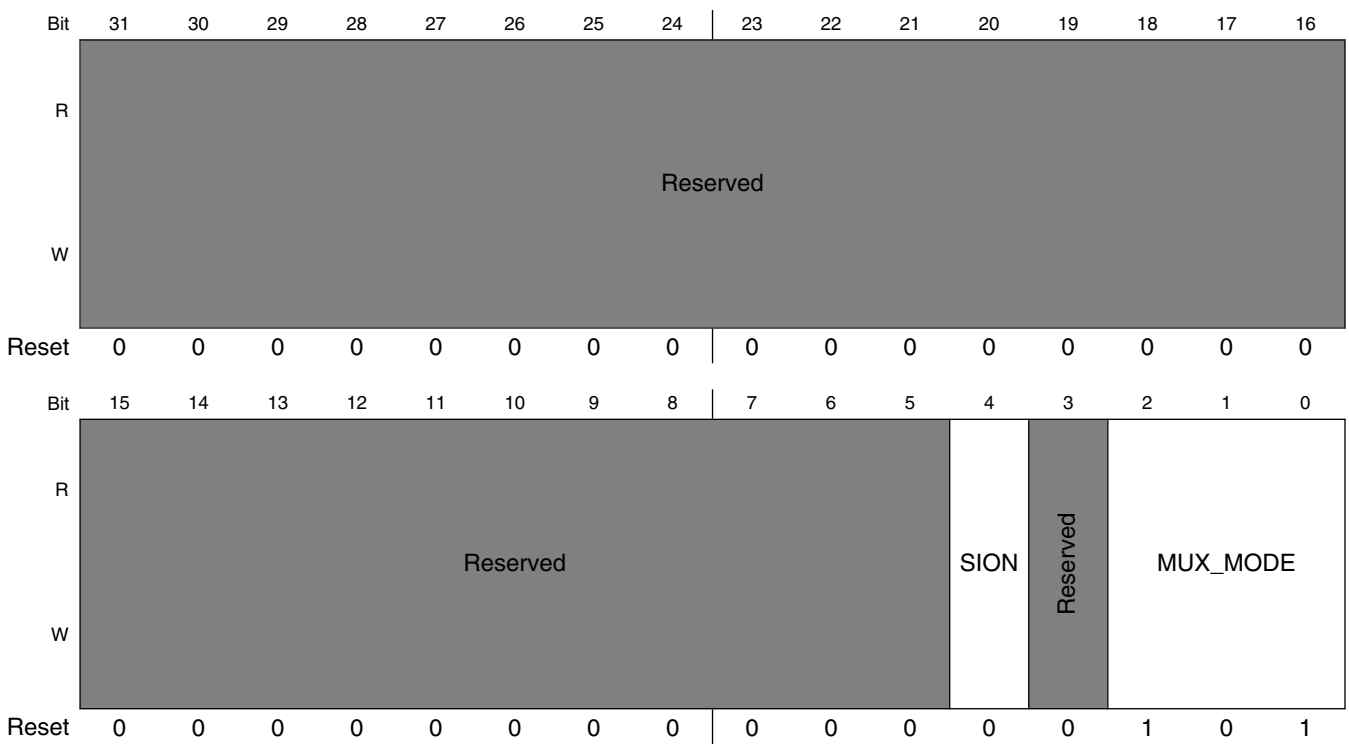
IOMUXC_SW_MUX_CTL_PAD_I2C3_SDA field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad I2C3_SDA 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: I2C3_SDA. 000 **ALT0_I2C3_SDA** — Select mux mode: ALT0 mux port: SDA of instance: I2C3 001 **ALT1_UART5_RTS_B** — Select mux mode: ALT1 mux port: RTS_B of instance: UART5 010 **ALT2_FLEXCAN2_TX** — Select mux mode: ALT2 mux port: TX of instance: FLEXCAN2 011 **ALT3_CSI_HSYNC** — Select mux mode: ALT3 mux port: HSYNC of instance: CSI 100 **ALT4_SDMA_EXT_EVENT1** — Select mux mode: ALT4 mux port: EXT_EVENT1 of instance: SDMA 101 **ALT5_GPIO4_IO13** — Select mux mode: ALT5 mux port: IO13 of instance: GPIO4 110 **ALT6_EPDC_BDR1** — Select mux mode: ALT6 mux port: BDR1 of instance: EPDC

8.2.7.84 SW_MUX_CTL_PAD_I2C4_SCL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C4_SCL)

SW_MUX_CTL Register

Address: 3033_0000h base + 160h offset = 3033_0160h



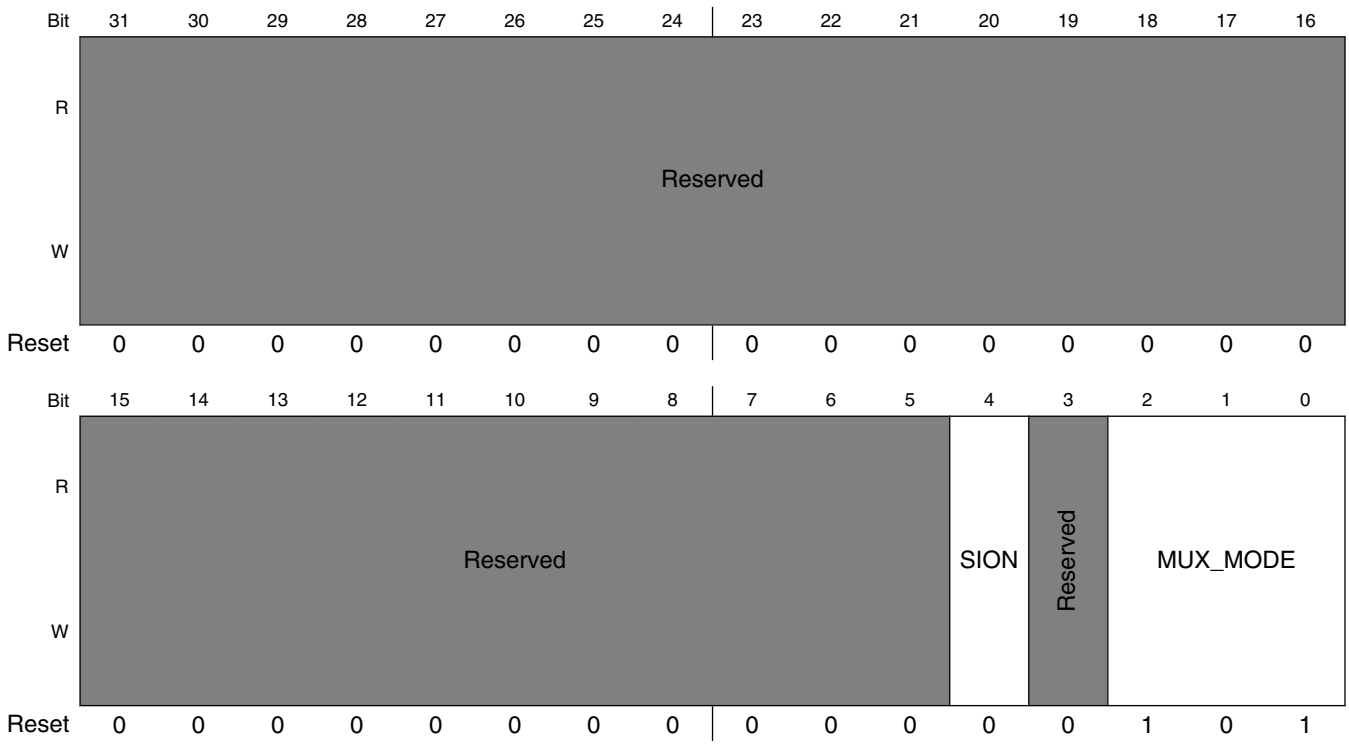
IOMUXC_SW_MUX_CTL_PAD_I2C4_SCL field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad I2C4_SCL 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: I2C4_SCL. 000 **ALT0_I2C4_SCL** — Select mux mode: ALT0 mux port: SCL of instance: I2C4 001 **ALT1_UART5_RX_DATA** — Select mux mode: ALT1 mux port: RX_DATA of instance: UART5 010 **ALT2_WDOG4_WDOG_B** — Select mux mode: ALT2 mux port: WDOG_B of instance: WDOG4 011 **ALT3_CSI_PIXCLK** — Select mux mode: ALT3 mux port: PIXCLK of instance: CSI 100 **ALT4_USB_OTG1_ID** — Select mux mode: ALT4 mux port: OTG1_ID of instance: USB 101 **ALT5_GPIO4_IO14** — Select mux mode: ALT5 mux port: IO14 of instance: GPIO4 110 **ALT6_EPDC_VCOM0** — Select mux mode: ALT6 mux port: VCOM0 of instance: EPDC

8.2.7.85 SW_MUX_CTL_PAD_I2C4_SDA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_I2C4_SDA)

SW_MUX_CTL Register

Address: 3033_0000h base + 164h offset = 3033_0164h



IOMUXC_SW_MUX_CTL_PAD_I2C4_SDA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad I2C4_SDA 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: I2C4_SDA. 000 **ALT0_I2C4_SDA** — Select mux mode: ALT0 mux port: SDA of instance: I2C4 001 **ALT1_UART5_TX_DATA** — Select mux mode: ALT1 mux port: TX_DATA of instance: UART5

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_I2C4_SDA field descriptions (continued)

Field	Description
	010 **ALT2_WDOG4_WDOG_RST_B_DEB** — Select mux mode: ALT2 mux port: WDOG_RST_B_DEB of instance: WDOG4 011 **ALT3_CSI_MCLK** — Select mux mode: ALT3 mux port: MCLK of instance: CSI 100 **ALT4_USB_OTG2_ID** — Select mux mode: ALT4 mux port: OTG2_ID of instance: USB 101 **ALT5_GPIO4_IO15** — Select mux mode: ALT5 mux port: IO15 of instance: GPIO4 110 **ALT6_EPDC_VCOM1** — Select mux mode: ALT6 mux port: VCOM1 of instance: EPDC

8.2.7.86 SW_MUX_CTL_PAD_ECSPI1_SCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI1_SCLK)

SW_MUX_CTL Register

Address: 3033_0000h base + 168h offset = 3033_0168h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved											SION	Reserved	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_ECSPI1_SCLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_ECSPI1_SCLK field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad ECSPI1_SCLK 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: ECSPI1_SCLK. 000 **ALT0_ECSPI1_SCLK** — Select mux mode: ALT0 mux port: SCLK of instance: ECSPI1 001 **ALT1_UART6_RX_DATA** — Select mux mode: ALT1 mux port: RX_DATA of instance: UART6 010 **ALT2_SD2_DATA4** — Select mux mode: ALT2 mux port: DATA4 of instance: SD2 011 **ALT3_CSI_DATA2** — Select mux mode: ALT3 mux port: DATA2 of instance: CSI 101 **ALT5_GPIO4_IO16** — Select mux mode: ALT5 mux port: IO16 of instance: GPIO4 110 **ALT6_EPDC_PWR_COM** — Select mux mode: ALT6 mux port: PWR_COM of instance: EPDC

8.2.7.87 SW_MUX_CTL_PAD_ECSPI1_MOSI SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI1_MOSI)

SW_MUX_CTL Register

Address: 3033_0000h base + 16Ch offset = 3033_016Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

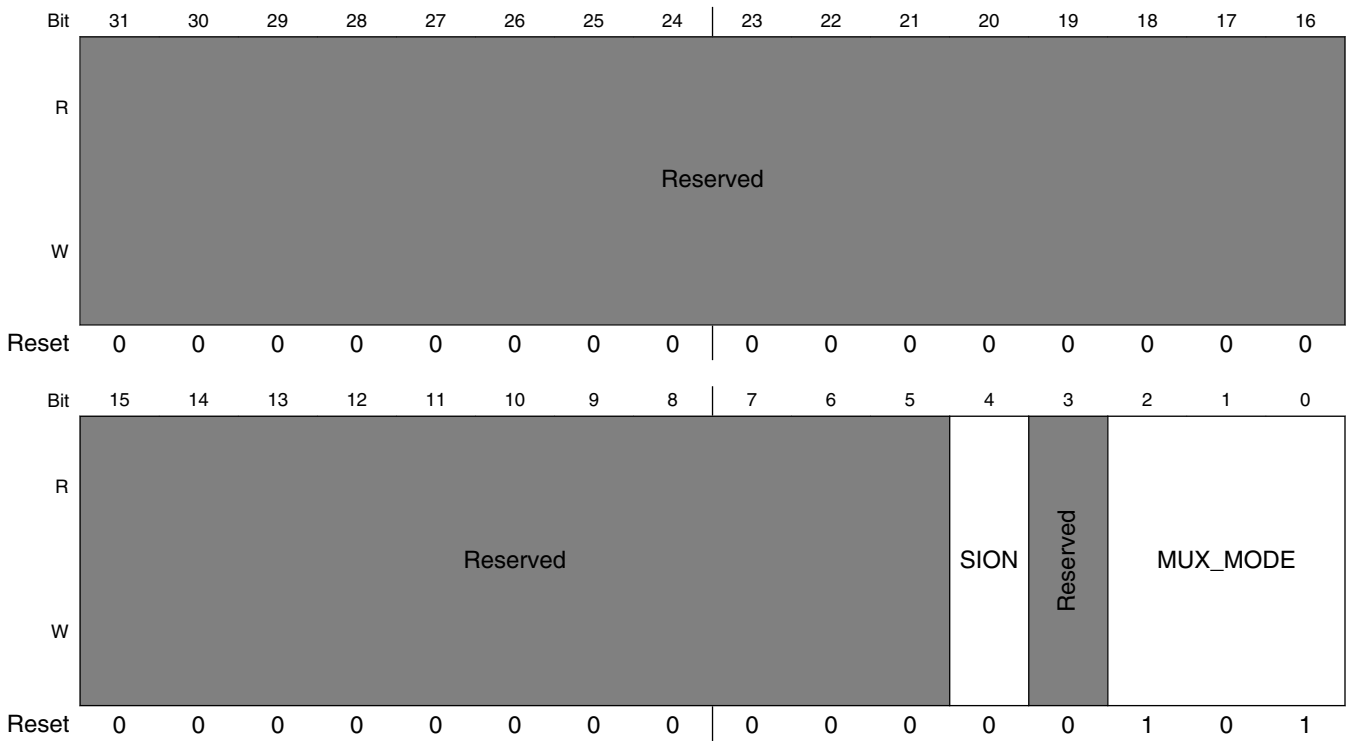
IOMUXC_SW_MUX_CTL_PAD_ECSPI1_MOSI field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ECSPI1_MOSI 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ECSPI1_MOSI. 000 **ALT0_ECSPI1_MOSI** — Select mux mode: ALT0 mux port: MOSI of instance: ECSPI1 001 **ALT1_UART6_TX_DATA** — Select mux mode: ALT1 mux port: TX_DATA of instance: UART6 010 **ALT2_SD2_DATA5** — Select mux mode: ALT2 mux port: DATA5 of instance: SD2 011 **ALT3_CSI_DATA3** — Select mux mode: ALT3 mux port: DATA3 of instance: CSI 101 **ALT5_GPIO4_IO17** — Select mux mode: ALT5 mux port: IO17 of instance: GPIO4 110 **ALT6_EPDC_PWR_STAT** — Select mux mode: ALT6 mux port: PWR_STAT of instance: EPDC

8.2.7.88 SW_MUX_CTL_PAD_ECSPI1_MISO SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI1_MISO)

SW_MUX_CTL Register

Address: 3033_0000h base + 170h offset = 3033_0170h



IOMUXC_SW_MUX_CTL_PAD_ECSPI1_MISO field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ECSPI1_MISO 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ECSPI1_MISO. 000 **ALT0_ECSPI1_MISO** — Select mux mode: ALT0 mux port: MISO of instance: ECSPI1 001 **ALT1_UART6_RTS_B** — Select mux mode: ALT1 mux port: RTS_B of instance: UART6 010 **ALT2_SD2_DATA6** — Select mux mode: ALT2 mux port: DATA6 of instance: SD2

Table continues on the next page...

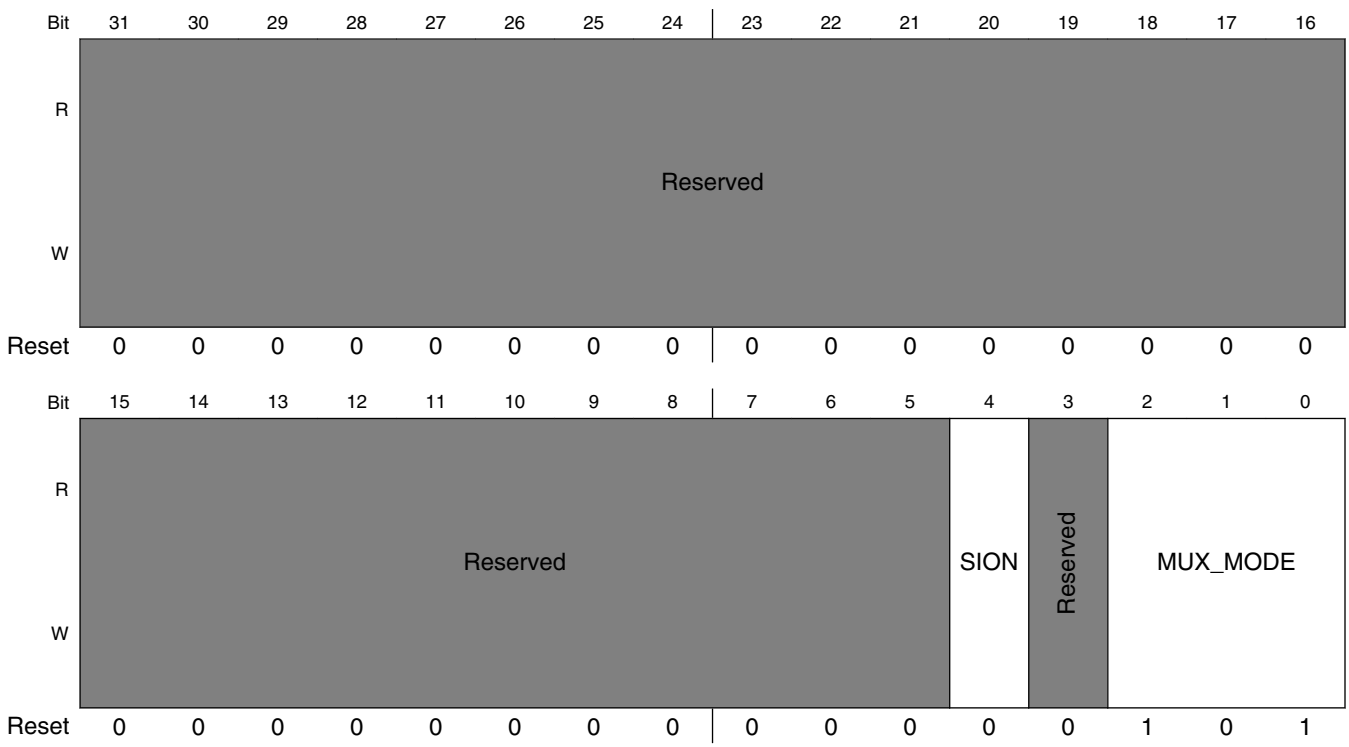
IOMUXC_SW_MUX_CTL_PAD_ECSPI1_MISO field descriptions (continued)

Field	Description
	011 **ALT3_CSI_DATA4** — Select mux mode: ALT3 mux port: DATA4 of instance: CSI 101 **ALT5_GPIO4_IO18** — Select mux mode: ALT5 mux port: IO18 of instance: GPIO4 110 **ALT6_EPDC_PWR_IRQ** — Select mux mode: ALT6 mux port: PWR_IRQ of instance: EPDC

8.2.7.89 SW_MUX_CTL_PAD_ECSPI1_SS0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI1_SS0)

SW_MUX_CTL Register

Address: 3033_0000h base + 174h offset = 3033_0174h



IOMUXC_SW_MUX_CTL_PAD_ECSPI1_SS0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ECSPI1_SS0 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_ECSPI1_SS0 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ECSPI1_SS0. 000 **ALT0_ECSPI1_SS0** — Select mux mode: ALT0 mux port: SS0 of instance: ECSPI1 001 **ALT1_UART6_CTS_B** — Select mux mode: ALT1 mux port: CTS_B of instance: UART6 010 **ALT2_SD2_DATA7** — Select mux mode: ALT2 mux port: DATA7 of instance: SD2 011 **ALT3_CSI_DATA5** — Select mux mode: ALT3 mux port: DATA5 of instance: CSI 101 **ALT5_GPIO4_IO19** — Select mux mode: ALT5 mux port: IO19 of instance: GPIO4 110 **ALT6_EPDC_PWR_CTRL3** — Select mux mode: ALT6 mux port: PWR_CTRL3 of instance: EPDC

8.2.7.90 SW_MUX_CTL_PAD_ECSPI2_SCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI2_SCLK)

SW_MUX_CTL Register

Address: 3033_0000h base + 178h offset = 3033_0178h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

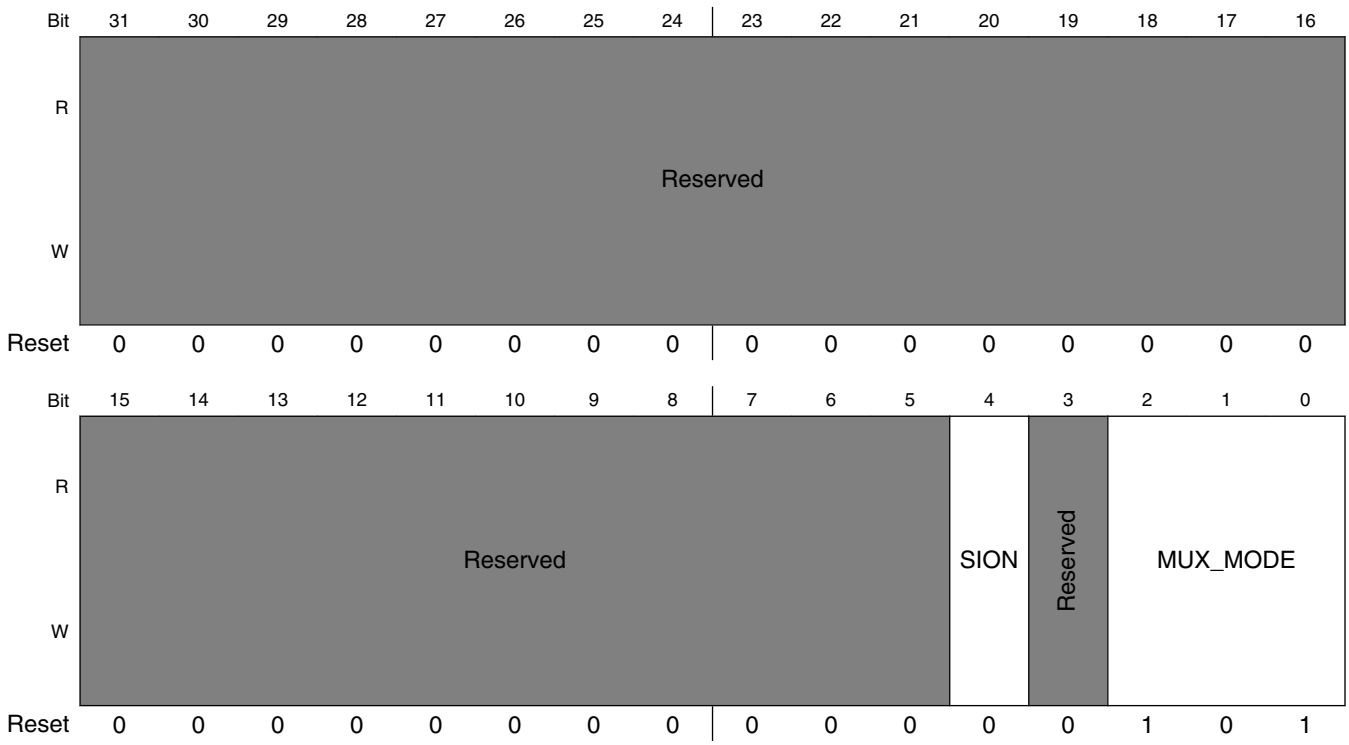
IOMUXC_SW_MUX_CTL_PAD_ECSPI2_SCLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ECSPI2_SCLK 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ECSPI2_SCLK. 000 **ALT0_ECSPI2_SCLK** — Select mux mode: ALT0 mux port: SCLK of instance: ECSPI2 001 **ALT1_UART7_RX_DATA** — Select mux mode: ALT1 mux port: RX_DATA of instance: UART7 010 **ALT2_SD1_DATA4** — Select mux mode: ALT2 mux port: DATA4 of instance: SD1 011 **ALT3_CSI_DATA6** — Select mux mode: ALT3 mux port: DATA6 of instance: CSI 100 **ALT4_LCD_DATA13** — Select mux mode: ALT4 mux port: DATA13 of instance: LCD 101 **ALT5_GPIO4_IO20** — Select mux mode: ALT5 mux port: IO20 of instance: GPIO4 110 **ALT6_EPDC_PWR_CTRL0** — Select mux mode: ALT6 mux port: PWR_CTRL0 of instance: EPDC

8.2.7.91 SW_MUX_CTL_PAD_ECSPI2_MOSI SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI2_MOSI)

SW_MUX_CTL Register

Address: 3033_0000h base + 17Ch offset = 3033_017Ch



IOMUXC_SW_MUX_CTL_PAD_ECSPI2_MOSI field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ECSPI2_MOSI 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ECSPI2_MOSI. 000 **ALT0_ECSPI2_MOSI** — Select mux mode: ALT0 mux port: MOSI of instance: ECSPI2 001 **ALT1_UART7_TX_DATA** — Select mux mode: ALT1 mux port: TX_DATA of instance: UART7 010 **ALT2_SD1_DATA5** — Select mux mode: ALT2 mux port: DATA5 of instance: SD1

Table continues on the next page...

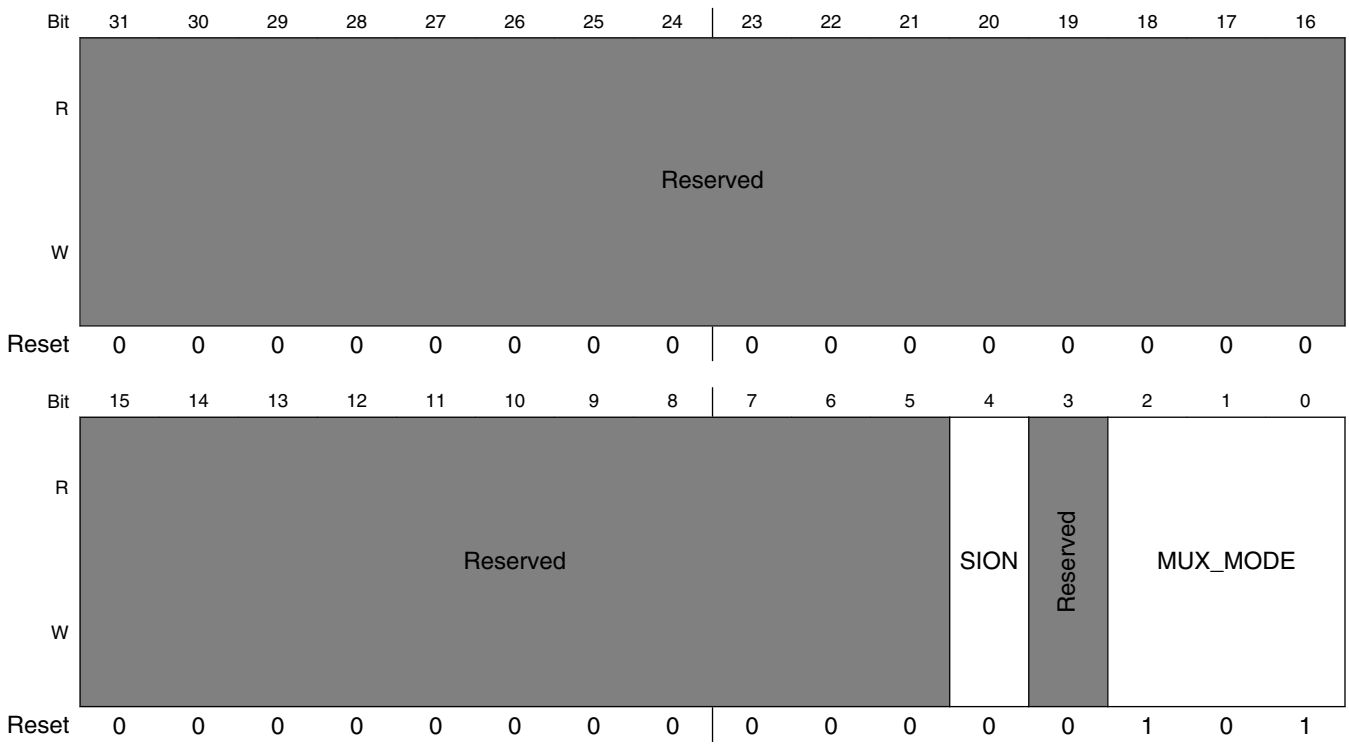
IOMUXC_SW_MUX_CTL_PAD_ECSPI2_MOSI field descriptions (continued)

Field	Description
	011 **ALT3_CSI_DATA7** — Select mux mode: ALT3 mux port: DATA7 of instance: CSI 100 **ALT4_LCD_DATA14** — Select mux mode: ALT4 mux port: DATA14 of instance: LCD 101 **ALT5_GPIO4_IO21** — Select mux mode: ALT5 mux port: IO21 of instance: GPIO4 110 **ALT6_EPDC_PWR_CTRL1** — Select mux mode: ALT6 mux port: PWR_CTRL1 of instance: EPDC

8.2.7.92 SW_MUX_CTL_PAD_ECSPI2_MISO SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI2_MISO)

SW_MUX_CTL Register

Address: 3033_0000h base + 180h offset = 3033_0180h



IOMUXC_SW_MUX_CTL_PAD_ECSPI2_MISO field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ECSPI2_MISO 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_ECSPI2_MISO field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ECSPI2_MISO. 101 **ALT5_GPIO4_IO22** — Select mux mode: ALT5 mux port: IO22 of instance: GPIO4 110 **ALT6_EPDC_PWR_CTRL2** — Select mux mode: ALT6 mux port: PWR_CTRL2 of instance: EPDC 000 **ALT0_ECSPI2_MISO** — Select mux mode: ALT0 mux port: MISO of instance: ECSPI2 001 **ALT1_UART7_RTS_B** — Select mux mode: ALT1 mux port: RTS_B of instance: UART7 010 **ALT2_SD1_DATA6** — Select mux mode: ALT2 mux port: DATA6 of instance: SD1 011 **ALT3_CSI_DATA8** — Select mux mode: ALT3 mux port: DATA8 of instance: CSI 100 **ALT4_LCD_DATA15** — Select mux mode: ALT4 mux port: DATA15 of instance: LCD

8.2.7.93 SW_MUX_CTL_PAD_ECSPI2_SS0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ECSPI2_SS0)

SW_MUX_CTL Register

Address: 3033_0000h base + 184h offset = 3033_0184h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

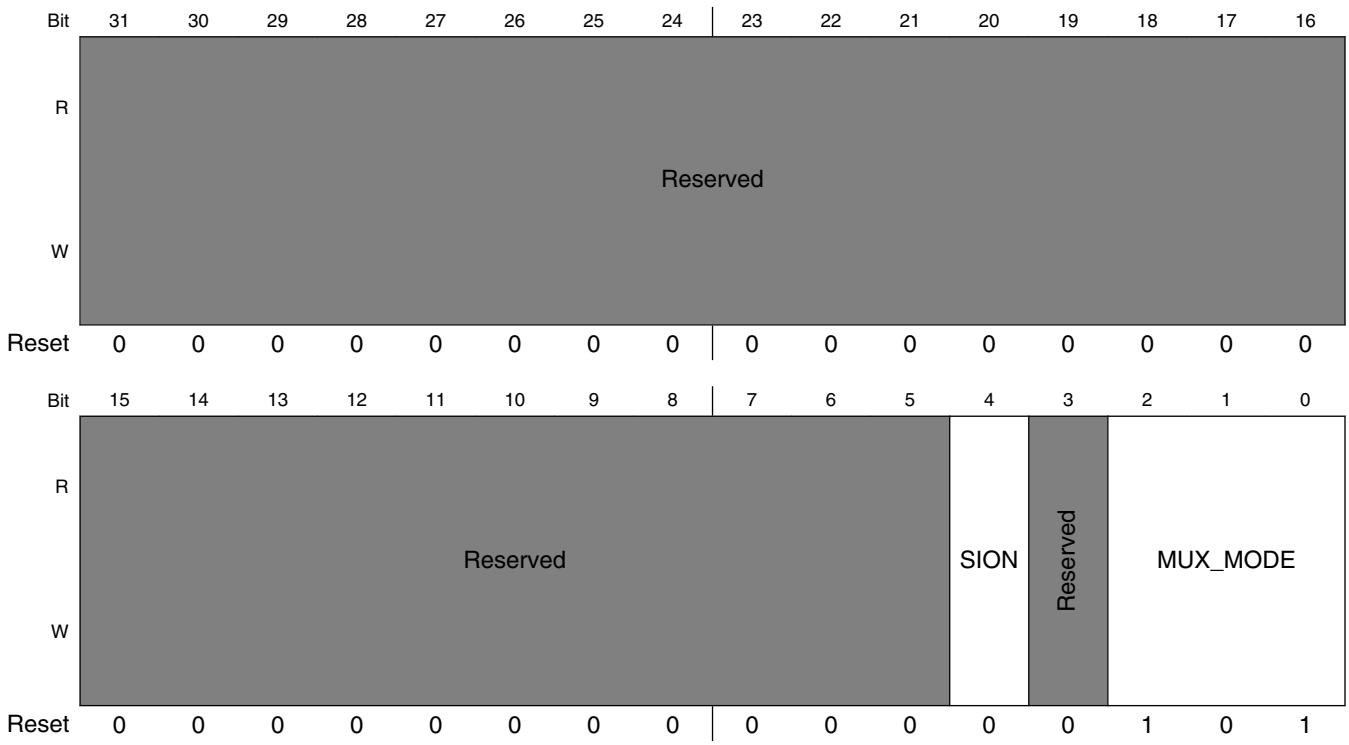
IOMUXC_SW_MUX_CTL_PAD_ECSPI2_SS0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ECSPI2_SS0 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ECSPI2_SS0. 000 **ALT0_ECSPI2_SS0** — Select mux mode: ALT0 mux port: SS0 of instance: ECSPI2 001 **ALT1_UART7_CTS_B** — Select mux mode: ALT1 mux port: CTS_B of instance: UART7 010 **ALT2_SD1_DATA7** — Select mux mode: ALT2 mux port: DATA7 of instance: SD1 011 **ALT3_CSI_DATA9** — Select mux mode: ALT3 mux port: DATA9 of instance: CSI 100 **ALT4_LCD_RESET** — Select mux mode: ALT4 mux port: RESET of instance: LCD 101 **ALT5_GPIO4_IO23** — Select mux mode: ALT5 mux port: IO23 of instance: GPIO4 110 **ALT6_EPDC_PWR_WAKE** — Select mux mode: ALT6 mux port: PWR_WAKE of instance: EPDC

8.2.7.94 SW_MUX_CTL_PAD_SD1_CD_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_CD_B)

SW_MUX_CTL Register

Address: 3033_0000h base + 188h offset = 3033_0188h



IOMUXC_SW_MUX_CTL_PAD_SD1_CD_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_CD_B 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD1_CD_B. 000 **ALT0_SD1_CD_B** — Select mux mode: ALT0 mux port: CD_B of instance: SD1 010 **ALT2_UART6_RX_DATA** — Select mux mode: ALT2 mux port: RX_DATA of instance: UART6 011 **ALT3_ECSPI4_MISO** — Select mux mode: ALT3 mux port: MISO of instance: ECSPI4

Table continues on the next page...

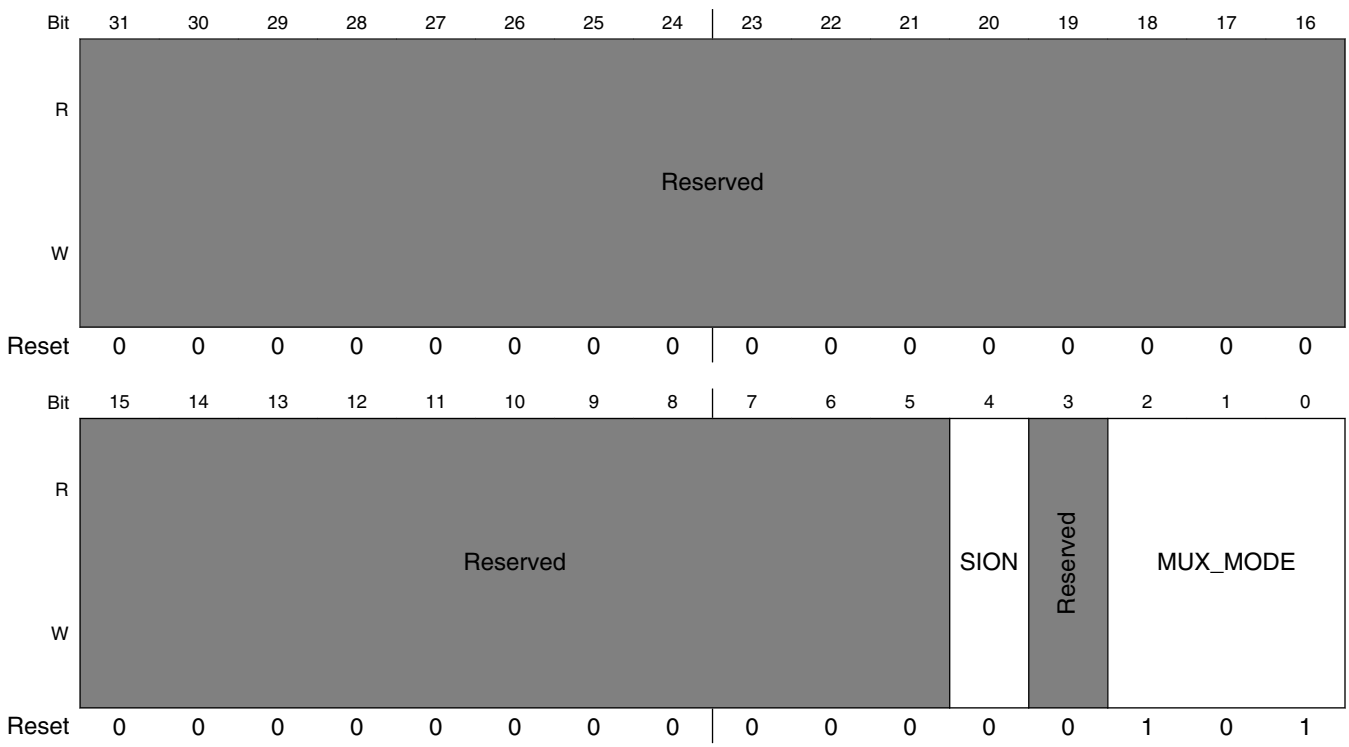
IOMUXC_SW_MUX_CTL_PAD_SD1_CD_B field descriptions (continued)

Field	Description
	100 **ALT4_FLEXTIMER1_CH0** — Select mux mode: ALT4 mux port: CH0 of instance: FLEXTIMER1 101 **ALT5_GPIO5_IO0** — Select mux mode: ALT5 mux port: IO0 of instance: GPIO5 110 **ALT6_CCM_CLKO1** — Select mux mode: ALT6 mux port: CLKO1 of instance: CCM

8.2.7.95 SW_MUX_CTL_PAD_SD1_WP SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_WP)

SW_MUX_CTL Register

Address: 3033_0000h base + 18Ch offset = 3033_018Ch



IOMUXC_SW_MUX_CTL_PAD_SD1_WP field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_WP 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

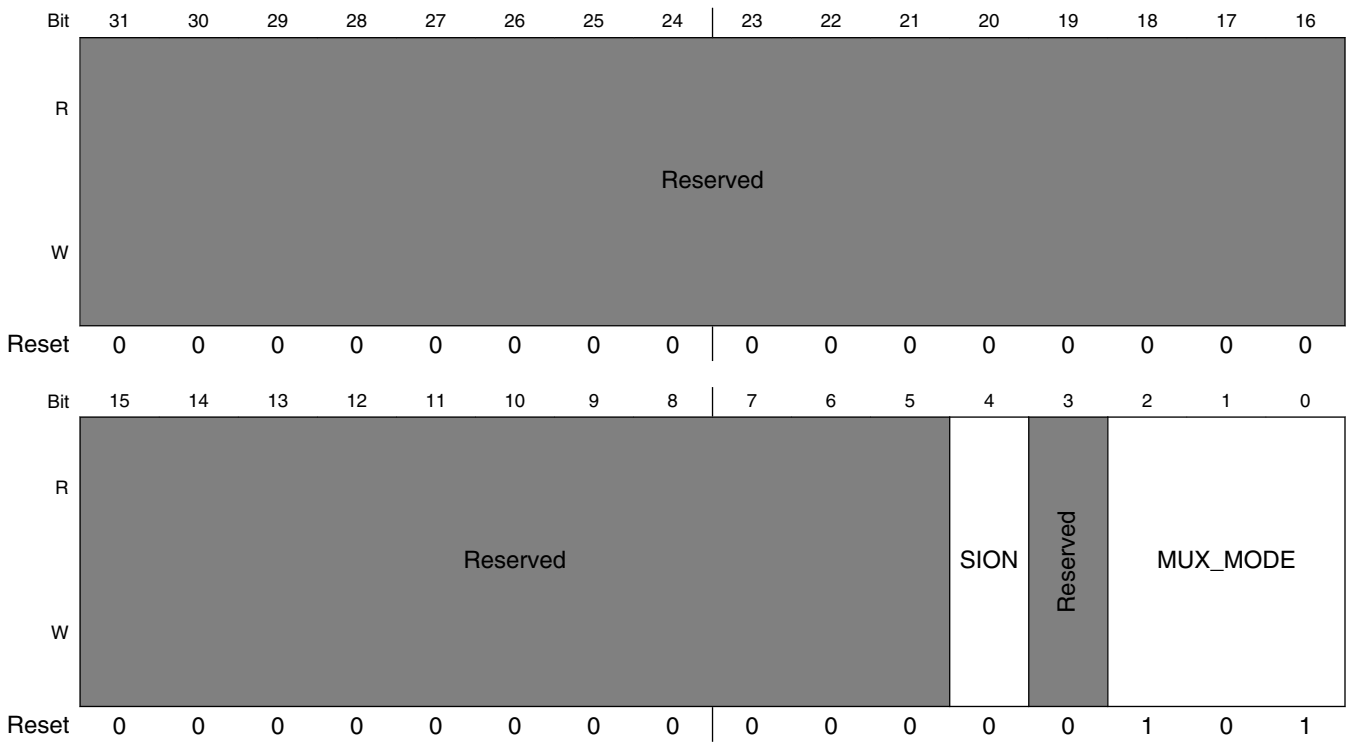
IOMUXC_SW_MUX_CTL_PAD_SD1_WP field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD1_WP. 000 **ALT0_SD1_WP** — Select mux mode: ALT0 mux port: WP of instance: SD1 010 **ALT2_UART6_TX_DATA** — Select mux mode: ALT2 mux port: TX_DATA of instance: UART6 011 **ALT3_ECSPI4_MOSI** — Select mux mode: ALT3 mux port: MOSI of instance: ECSPI4 100 **ALT4_FLEXTIMER1_CH1** — Select mux mode: ALT4 mux port: CH1 of instance: FLEXTIMER1 101 **ALT5_GPIO5_IO1** — Select mux mode: ALT5 mux port: IO1 of instance: GPIO5 110 **ALT6_CCM_CLKO2** — Select mux mode: ALT6 mux port: CLKO2 of instance: CCM

8.2.7.96 SW_MUX_CTL_PAD_SD1_RESET_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_RESET_B)

SW_MUX_CTL Register

Address: 3033_0000h base + 190h offset = 3033_0190h



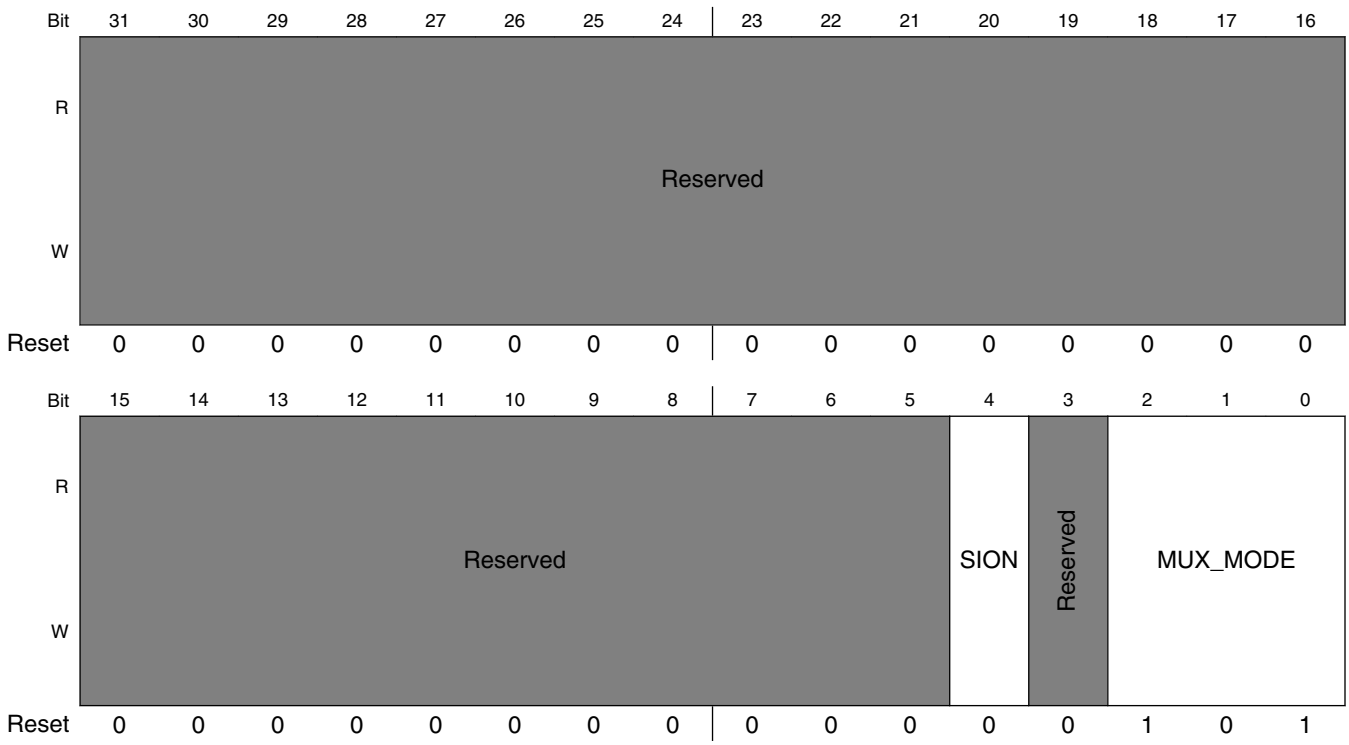
IOMUXC_SW_MUX_CTL_PAD_SD1_RESET_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_RESET_B 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SD1_RESET_B. 000 **ALT0_SD1_RESET_B** — Select mux mode: ALT0 mux port: RESET_B of instance: SD1 001 **ALT1_SAI3_MCLK** — Select mux mode: ALT1 mux port: MCLK of instance: SAI3 010 **ALT2_UART6_RTS_B** — Select mux mode: ALT2 mux port: RTS_B of instance: UART6 011 **ALT3_ECSPI4_SCLK** — Select mux mode: ALT3 mux port: SCLK of instance: ECSPI4 100 **ALT4_FLEXTIMER1_CH2** — Select mux mode: ALT4 mux port: CH2 of instance: FLEXTIMER1 101 **ALT5_GPIO5_IO2** — Select mux mode: ALT5 mux port: IO2 of instance: GPIO5

8.2.7.97 SW_MUX_CTL_PAD_SD1_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_CLK)

SW_MUX_CTL Register

Address: 3033_0000h base + 194h offset = 3033_0194h



IOMUXC_SW_MUX_CTL_PAD_SD1_CLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_CLK 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SD1_CLK. 000 **ALT0_SD1_CLK** — Select mux mode: ALT0 mux port: CLK of instance: SD1 001 **ALT1_SAI3_RX_SYNC** — Select mux mode: ALT1 mux port: RX_SYNC of instance: SAI3 010 **ALT2_UART6_CTS_B** — Select mux mode: ALT2 mux port: CTS_B of instance: UART6

Table continues on the next page...

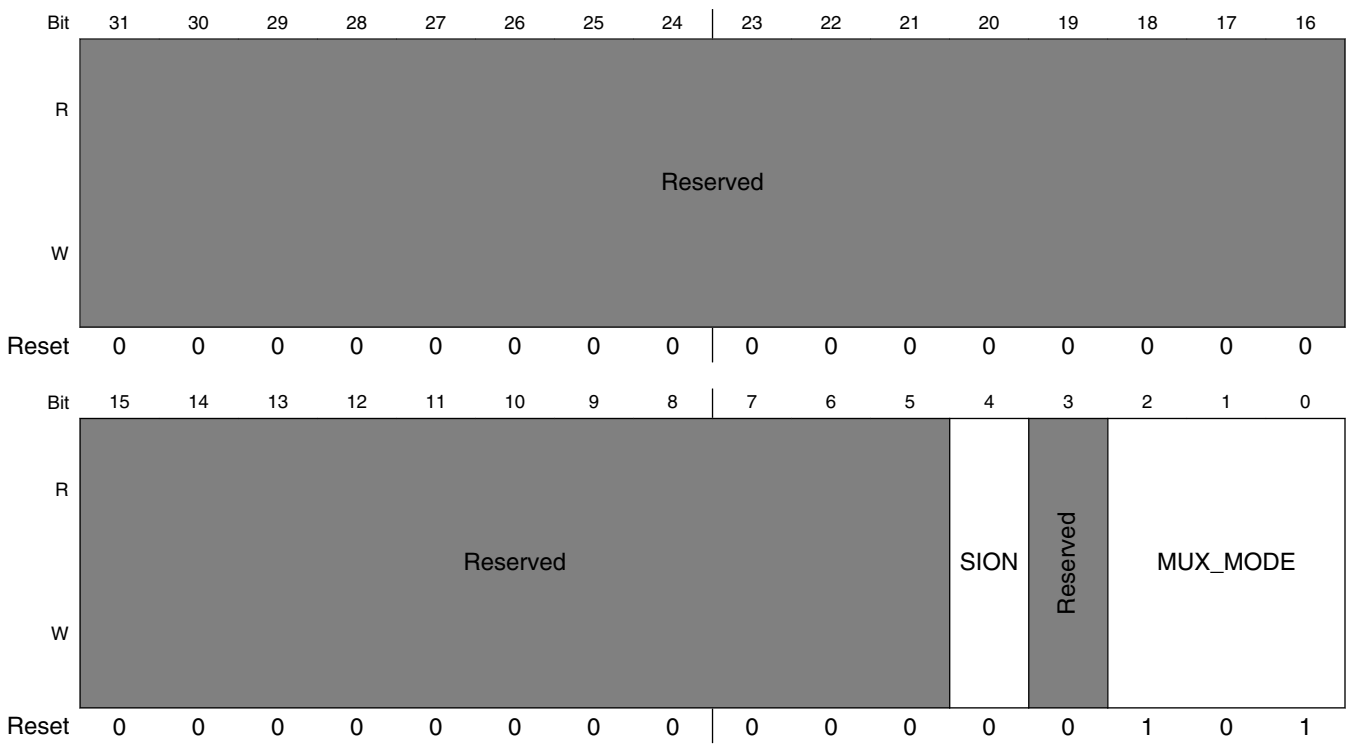
IOMUXC_SW_MUX_CTL_PAD_SD1_CLK field descriptions (continued)

Field	Description
	011 **ALT3_ECSPI4_SS0** — Select mux mode: ALT3 mux port: SS0 of instance: ECSPI4 100 **ALT4_FLEXTIMER1_CH3** — Select mux mode: ALT4 mux port: CH3 of instance: FLEXTIMER1 101 **ALT5_GPIO5_IO3** — Select mux mode: ALT5 mux port: IO3 of instance: GPIO5

8.2.7.98 SW_MUX_CTL_PAD_SD1_CMD SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_CMD)

SW_MUX_CTL Register

Address: 3033_0000h base + 198h offset = 3033_0198h



IOMUXC_SW_MUX_CTL_PAD_SD1_CMD field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_CMD 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

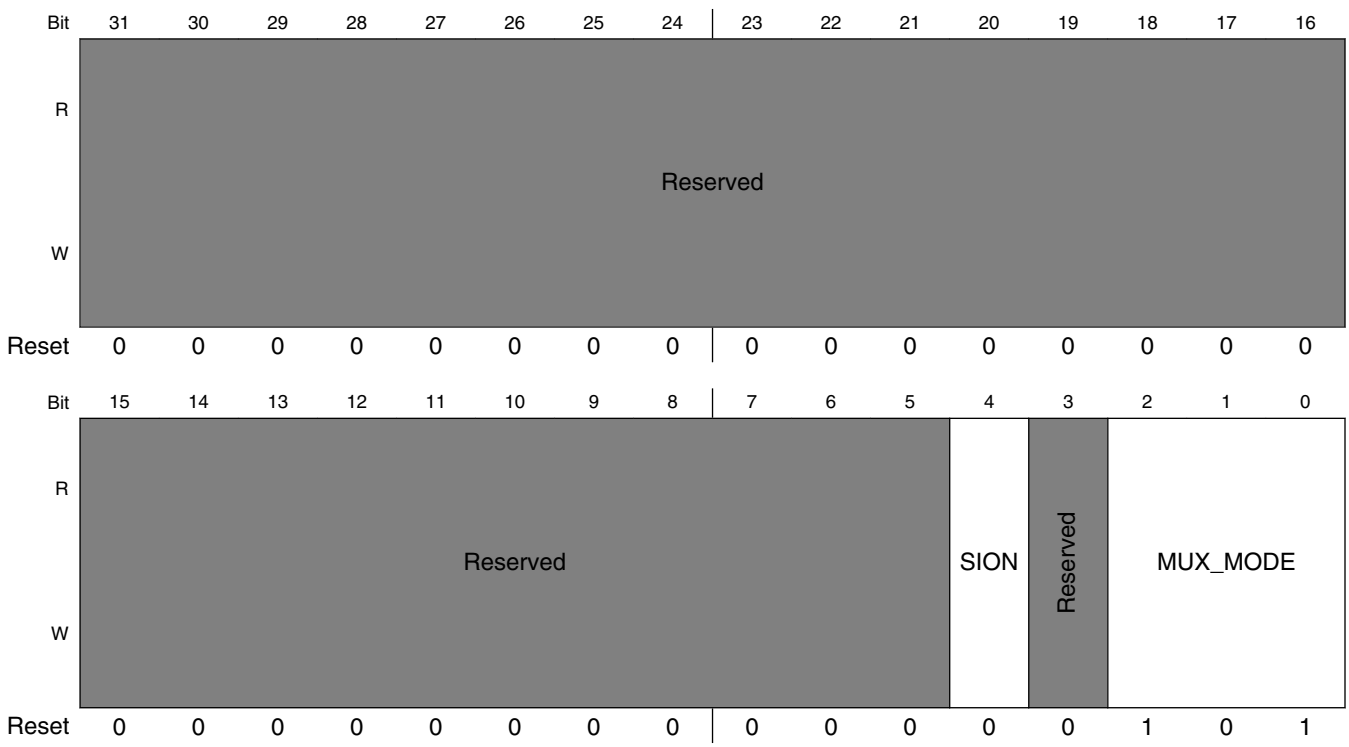
IOMUXC_SW_MUX_CTL_PAD_SD1_CMD field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD1_CMD. 000 **ALT0_SD1_CMD** — Select mux mode: ALT0 mux port: CMD of instance: SD1 001 **ALT1_SAI3_RX_BCLK** — Select mux mode: ALT1 mux port: RX_BCLK of instance: SAI3 011 **ALT3_ECSPI4_SS1** — Select mux mode: ALT3 mux port: SS1 of instance: ECSPI4 100 **ALT4_FLEXTIMER2_CH0** — Select mux mode: ALT4 mux port: CH0 of instance: FLEXTIMER2 101 **ALT5_GPIO5_IO4** — Select mux mode: ALT5 mux port: IO4 of instance: GPIO5

8.2.7.99 SW_MUX_CTL_PAD_SD1_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA0)

SW_MUX_CTL Register

Address: 3033_0000h base + 19Ch offset = 3033_019Ch



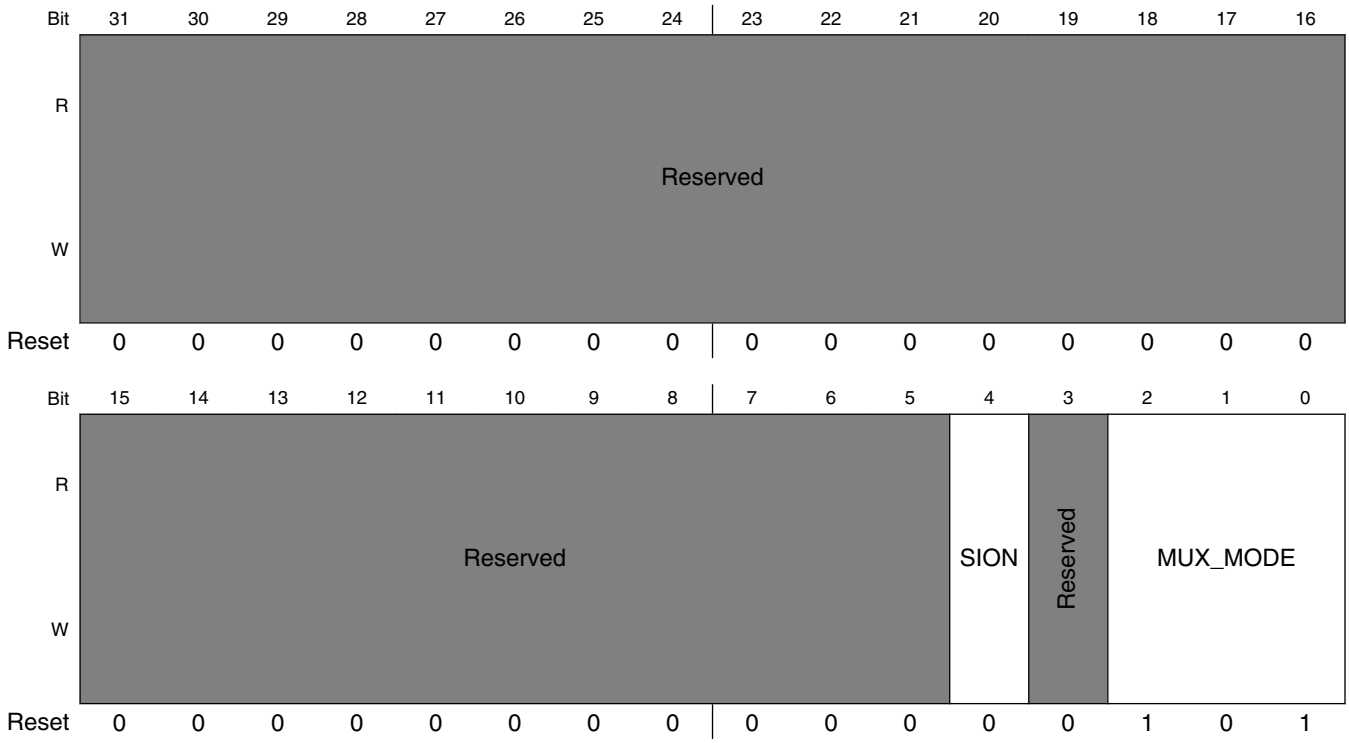
IOMUXC_SW_MUX_CTL_PAD_SD1_DATA0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_DATA0 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SD1_DATA0. 000 **ALT0_SD1_DATA0** — Select mux mode: ALT0 mux port: DATA0 of instance: SD1 001 **ALT1_SAI3_RX_DATA0** — Select mux mode: ALT1 mux port: RX_DATA0 of instance: SAI3 010 **ALT2_UART7_RX_DATA** — Select mux mode: ALT2 mux port: RX_DATA of instance: UART7 011 **ALT3_ECSPI4_SS2** — Select mux mode: ALT3 mux port: SS2 of instance: ECSPI4 100 **ALT4_FLEXTIMER2_CH1** — Select mux mode: ALT4 mux port: CH1 of instance: FLEXTIMER2 101 **ALT5_GPIO5_IO5** — Select mux mode: ALT5 mux port: IO5 of instance: GPIO5 110 **ALT6_CCM_EXT_CLK1** — Select mux mode: ALT6 mux port: EXT_CLK1 of instance: CCM

8.2.7.100 SW_MUX_CTL_PAD_SD1_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA1)

SW_MUX_CTL Register

Address: 3033_0000h base + 1A0h offset = 3033_01A0h



IOMUXC_SW_MUX_CTL_PAD_SD1_DATA1 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_DATA1 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SD1_DATA1. 000 **ALT0_SD1_DATA1** — Select mux mode: ALT0 mux port: DATA1 of instance: SD1 001 **ALT1_SAI3_TX_BCLK** — Select mux mode: ALT1 mux port: TX_BCLK of instance: SAI3 010 **ALT2_UART7_TX_DATA** — Select mux mode: ALT2 mux port: TX_DATA of instance: UART7

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_SD1_DATA1 field descriptions (continued)

Field	Description
	011 **ALT3_ECSPI4_SS3** — Select mux mode: ALT3 mux port: SS3 of instance: ECSPI4 100 **ALT4_FLEXTIMER2_CH2** — Select mux mode: ALT4 mux port: CH2 of instance: FLEXTIMER2 101 **ALT5_GPIO5_IO6** — Select mux mode: ALT5 mux port: IO6 of instance: GPIO5 110 **ALT6_CCM_EXT_CLK2** — Select mux mode: ALT6 mux port: EXT_CLK2 of instance: CCM

8.2.7.101 SW_MUX_CTL_PAD_SD1_DATA2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA2)

SW_MUX_CTL Register

Address: 3033_0000h base + 1A4h offset = 3033_01A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_SD1_DATA2 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_DATA2 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_SD1_DATA2 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SD1_DATA2. 000 **ALT0_SD1_DATA2** — Select mux mode: ALT0 mux port: DATA2 of instance: SD1 001 **ALT1_SAI3_TX_SYNC** — Select mux mode: ALT1 mux port: TX_SYNC of instance: SAI3 010 **ALT2_UART7_CTS_B** — Select mux mode: ALT2 mux port: CTS_B of instance: UART7 011 **ALT3_ECSPI4_RDY** — Select mux mode: ALT3 mux port: RDY of instance: ECSPI4 100 **ALT4_FLEXTIMER2_CH3** — Select mux mode: ALT4 mux port: CH3 of instance: FLEXTIMER2 101 **ALT5_GPIO5_IO7** — Select mux mode: ALT5 mux port: IO7 of instance: GPIO5 110 **ALT6_CCM_EXT_CLK3** — Select mux mode: ALT6 mux port: EXT_CLK3 of instance: CCM

8.2.7.102 SW_MUX_CTL_PAD_SD1_DATA3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA3)

SW_MUX_CTL Register

Address: 3033_0000h base + 1A8h offset = 3033_01A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

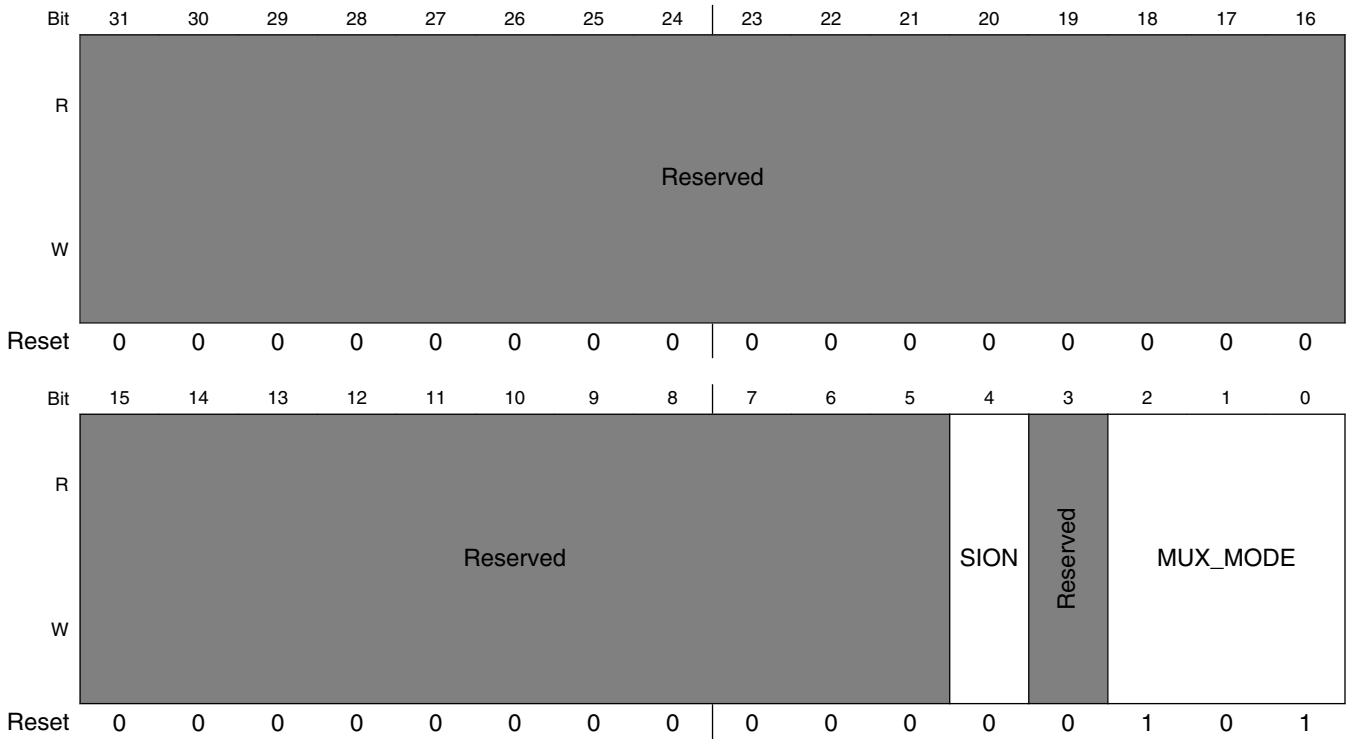
IOMUXC_SW_MUX_CTL_PAD_SD1_DATA3 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD1_DATA3 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SD1_DATA3. 000 **ALT0_SD1_DATA3** — Select mux mode: ALT0 mux port: DATA3 of instance: SD1 001 **ALT1_SAI3_TX_DATA0** — Select mux mode: ALT1 mux port: TX_DATA0 of instance: SAI3 010 **ALT2_UART7_RTS_B** — Select mux mode: ALT2 mux port: RTS_B of instance: UART7 011 **ALT3_ECSPI3_SS1** — Select mux mode: ALT3 mux port: SS1 of instance: ECSPI3 100 **ALT4_FLEXTIMER1_PHA** — Select mux mode: ALT4 mux port: PHA of instance: FLEXTIMER1 101 **ALT5_GPIO5_IO8** — Select mux mode: ALT5 mux port: IO8 of instance: GPIO5 110 **ALT6_CCM_EXT_CLK4** — Select mux mode: ALT6 mux port: EXT_CLK4 of instance: CCM

8.2.7.103 SW_MUX_CTL_PAD_SD2_CD_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_CD_B)

SW_MUX_CTL Register

Address: 3033_0000h base + 1ACh offset = 3033_01ACh



IOMUXC_SW_MUX_CTL_PAD_SD2_CD_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD2_CD_B 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SD2_CD_B. 000 **ALT0_SD2_CD_B** — Select mux mode: ALT0 mux port: CD_B of instance: SD2 001 **ALT1_ENET1_MDIO** — Select mux mode: ALT1 mux port: MDIO of instance: ENET1 010 **ALT2_ENET2_MDIO** — Select mux mode: ALT2 mux port: MDIO of instance: ENET2

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_SD2_CD_B field descriptions (continued)

Field	Description
	011 **ALT3_ECSPI3_SS2** — Select mux mode: ALT3 mux port: SS2 of instance: ECSPI3 100 **ALT4_FLEXTIMER1_PHB** — Select mux mode: ALT4 mux port: PHB of instance: FLEXTIMER1 101 **ALT5_GPIO5_IO9** — Select mux mode: ALT5 mux port: IO9 of instance: GPIO5 110 **ALT6_SDMA_EXT_EVENT0** — Select mux mode: ALT6 mux port: EXT_EVENT0 of instance: SDMA

8.2.7.104 SW_MUX_CTL_PAD_SD2_WP SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_WP)

SW_MUX_CTL Register

Address: 3033_0000h base + 1B0h offset = 3033_01B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_SD2_WP field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_SD2_WP field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad SD2_WP 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SD2_WP. 000 **ALT0_SD2_WP** — Select mux mode: ALT0 mux port: WP of instance: SD2 001 **ALT1_ENET1_MDC** — Select mux mode: ALT1 mux port: MDC of instance: ENET1 010 **ALT2_ENET2_MDC** — Select mux mode: ALT2 mux port: MDC of instance: ENET2 011 **ALT3_ECSPI3_SS3** — Select mux mode: ALT3 mux port: SS3 of instance: ECSPI3 100 **ALT4_USB_OTG1_ID** — Select mux mode: ALT4 mux port: OTG1_ID of instance: USB 101 **ALT5_GPIO5_IO10** — Select mux mode: ALT5 mux port: IO10 of instance: GPIO5 110 **ALT6_SDMA_EXT_EVENT1** — Select mux mode: ALT6 mux port: EXT_EVENT1 of instance: SDMA

8.2.7.105 SW_MUX_CTL_PAD_SD2_RESET_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_RESET_B)

SW_MUX_CTL Register

Address: 3033_0000h base + 1B4h offset = 3033_01B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

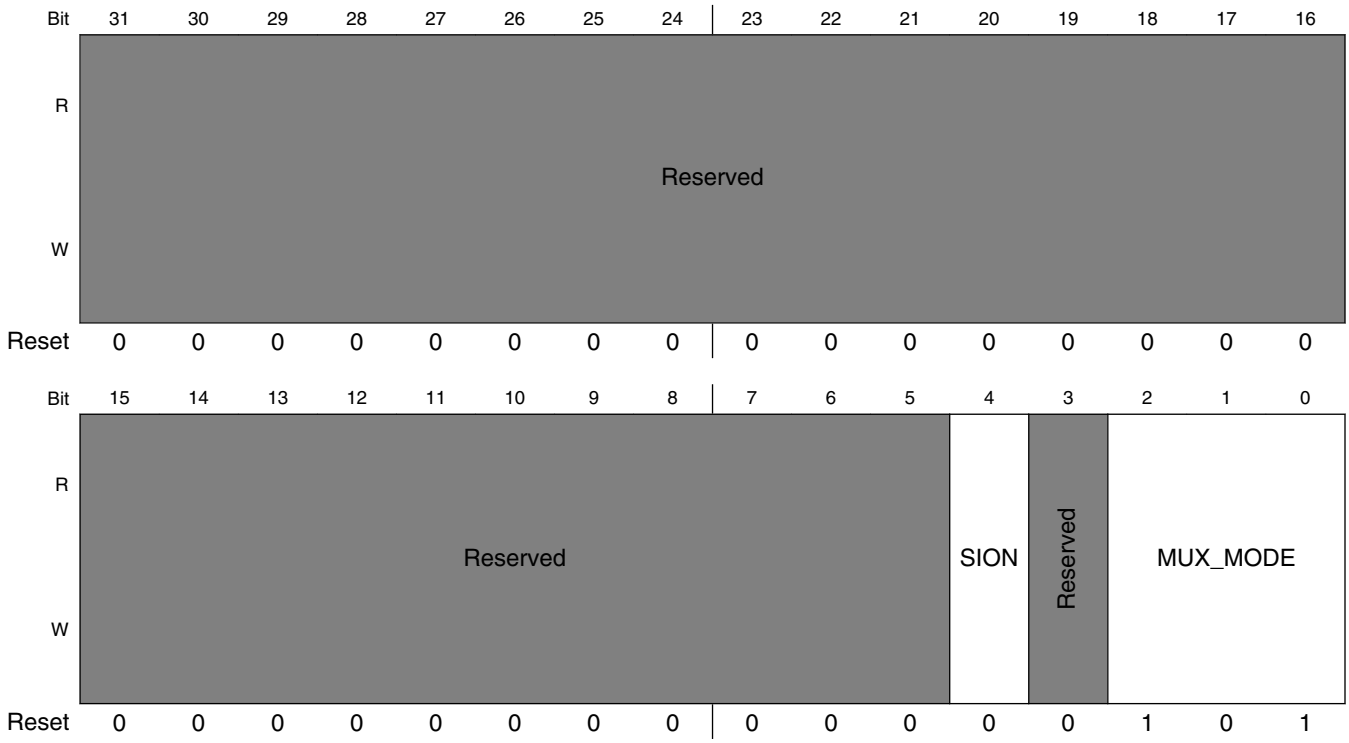
IOMUXC_SW_MUX_CTL_PAD_SD2_RESET_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD2_RESET_B 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD2_RESET_B. 000 **ALT0_SD2_RESET_B** — Select mux mode: ALT0 mux port: RESET_B of instance: SD2 001 **ALT1_SAI2_MCLK** — Select mux mode: ALT1 mux port: MCLK of instance: SAI2 010 **ALT2_SD2_RESET** — Select mux mode: ALT2 mux port: RESET of instance: SD2 011 **ALT3_ECSPI3_RDY** — Select mux mode: ALT3 mux port: RDY of instance: ECSPI3 100 **ALT4_USB_OTG2_ID** — Select mux mode: ALT4 mux port: OTG2_ID of instance: USB 101 **ALT5_GPIO5_IO11** — Select mux mode: ALT5 mux port: IO11 of instance: GPIO5

8.2.7.106 SW_MUX_CTL_PAD_SD2_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_CLK)

SW_MUX_CTL Register

Address: 3033_0000h base + 1B8h offset = 3033_01B8h



IOMUXC_SW_MUX_CTL_PAD_SD2_CLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD2_CLK 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: SD2_CLK. 000 **ALT0_SD2_CLK** — Select mux mode: ALT0 mux port: CLK of instance: SD2 001 **ALT1_SAI2_RX_SYNC** — Select mux mode: ALT1 mux port: RX_SYNC of instance: SAI2 010 **ALT2_MQS_RIGHT** — Select mux mode: ALT2 mux port: RIGHT of instance: MQS

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_SD2_CLK field descriptions (continued)

Field	Description
	011 **ALT3_GPT4_CLK** — Select mux mode: ALT3 mux port: CLK of instance: GPT4 101 **ALT5_GPIO5_IO12** — Select mux mode: ALT5 mux port: IO12 of instance: GPIO5

8.2.7.107 SW_MUX_CTL_PAD_SD2_CMD SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_CMD)

SW_MUX_CTL Register

Address: 3033_0000h base + 1BCh offset = 3033_01BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_SD2_CMD field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD2_CMD 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved

Table continues on the next page...

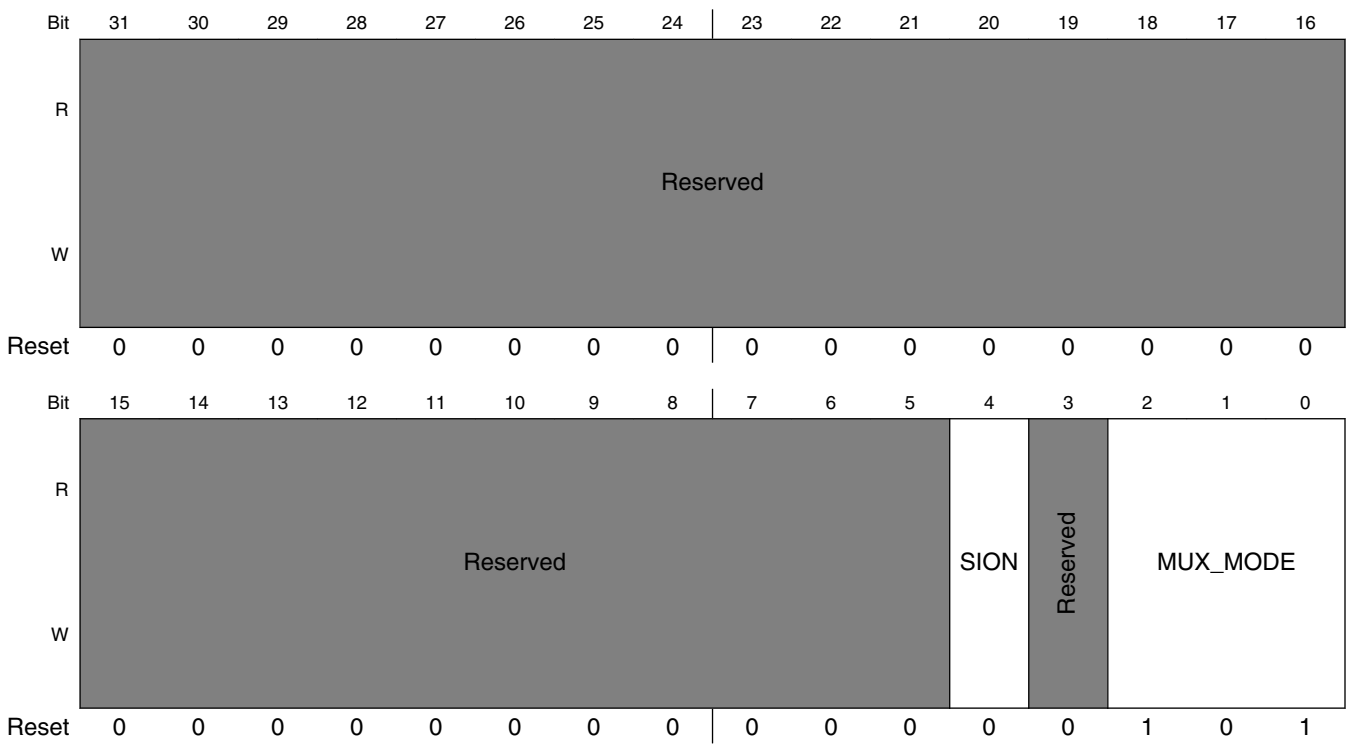
IOMUXC_SW_MUX_CTL_PAD_SD2_CMD field descriptions (continued)

Field	Description
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD2_CMD. 000 **ALT0_SD2_CMD** — Select mux mode: ALT0 mux port: CMD of instance: SD2 001 **ALT1_SAI2_RX_BCLK** — Select mux mode: ALT1 mux port: RX_BCLK of instance: SAI2 010 **ALT2_MQS_LEFT** — Select mux mode: ALT2 mux port: LEFT of instance: MQS 011 **ALT3_GPT4_CAPTURE1** — Select mux mode: ALT3 mux port: CAPTURE1 of instance: GPT4 100 **ALT4_SIM2_PORT1_TRXD** — Select mux mode: ALT4 mux port: PORT1_TRXD of instance: SIM2 101 **ALT5_GPIO5_IO13** — Select mux mode: ALT5 mux port: IO13 of instance: GPIO5

8.2.7.108 SW_MUX_CTL_PAD_SD2_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_DATA0)

SW_MUX_CTL Register

Address: 3033_0000h base + 1C0h offset = 3033_01C0h



IOMUXC_SW_MUX_CTL_PAD_SD2_DATA0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved

Table continues on the next page...

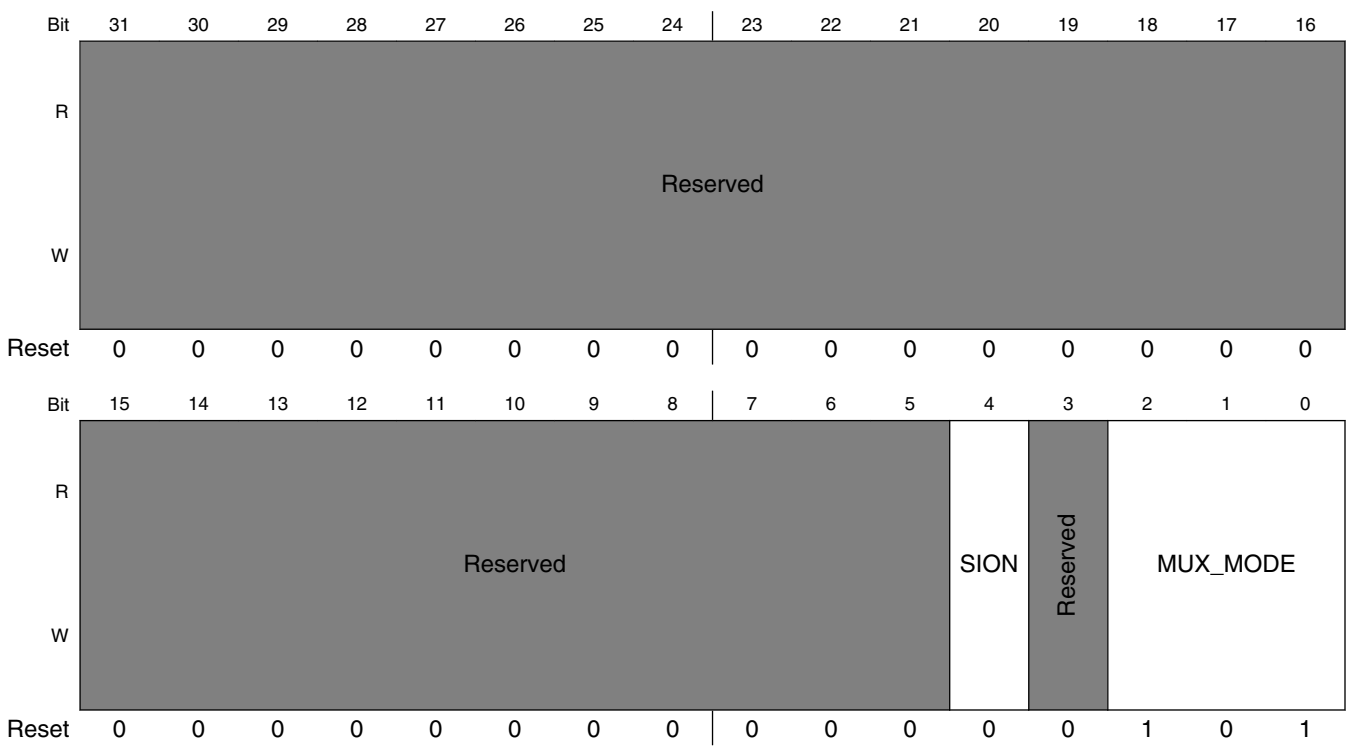
IOMUXC_SW_MUX_CTL_PAD_SD2_DATA0 field descriptions (continued)

Field	Description
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD2_DATA0 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD2_DATA0. 000 **ALT0_SD2_DATA0** — Select mux mode: ALT0 mux port: DATA0 of instance: SD2 001 **ALT1_SAI2_RX_DATA0** — Select mux mode: ALT1 mux port: RX_DATA0 of instance: SAI2 010 **ALT2_UART4_RX_DATA** — Select mux mode: ALT2 mux port: RX_DATA of instance: UART4 011 **ALT3_GPT4_CAPTURE2** — Select mux mode: ALT3 mux port: CAPTURE2 of instance: GPT4 100 **ALT4_SIM2_PORT1_CLK** — Select mux mode: ALT4 mux port: PORT1_CLK of instance: SIM2 101 **ALT5_GPIO5_IO14** — Select mux mode: ALT5 mux port: IO14 of instance: GPIO5

8.2.7.109 SW_MUX_CTL_PAD_SD2_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_DATA1)

SW_MUX_CTL Register

Address: 3033_0000h base + 1C4h offset = 3033_01C4h



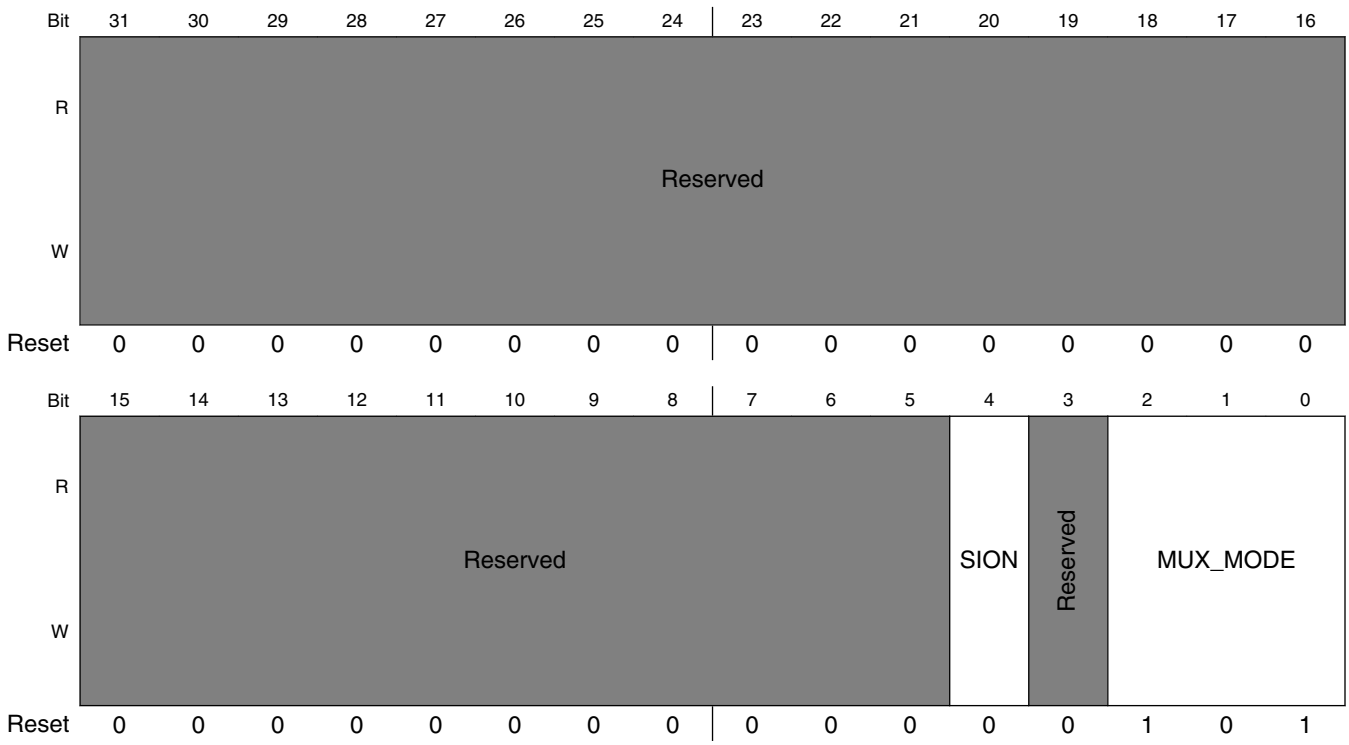
IOMUXC_SW_MUX_CTL_PAD_SD2_DATA1 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD2_DATA1 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD2_DATA1. 000 **ALT0_SD2_DATA1** — Select mux mode: ALT0 mux port: DATA1 of instance: SD2 001 **ALT1_SAI2_TX_BCLK** — Select mux mode: ALT1 mux port: TX_BCLK of instance: SAI2 010 **ALT2_UART4_TX_DATA** — Select mux mode: ALT2 mux port: TX_DATA of instance: UART4 011 **ALT3_GPT4_COMPARE1** — Select mux mode: ALT3 mux port: COMPARE1 of instance: GPT4 100 **ALT4_SIM2_PORT1_RST_B** — Select mux mode: ALT4 mux port: PORT1_RST_B of instance: SIM2 101 **ALT5_GPIO5_IO15** — Select mux mode: ALT5 mux port: IO15 of instance: GPIO5

8.2.7.110 SW_MUX_CTL_PAD_SD2_DATA2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_DATA2)

SW_MUX_CTL Register

Address: 3033_0000h base + 1C8h offset = 3033_01C8h



IOMUXC_SW_MUX_CTL_PAD_SD2_DATA2 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD2_DATA2 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD2_DATA2. 000 **ALT0_SD2_DATA2** — Select mux mode: ALT0 mux port: DATA2 of instance: SD2 001 **ALT1_SAI2_TX_SYNC** — Select mux mode: ALT1 mux port: TX_SYNC of instance: SAI2 010 **ALT2_UART4_CTS_B** — Select mux mode: ALT2 mux port: CTS_B of instance: UART4

Table continues on the next page...

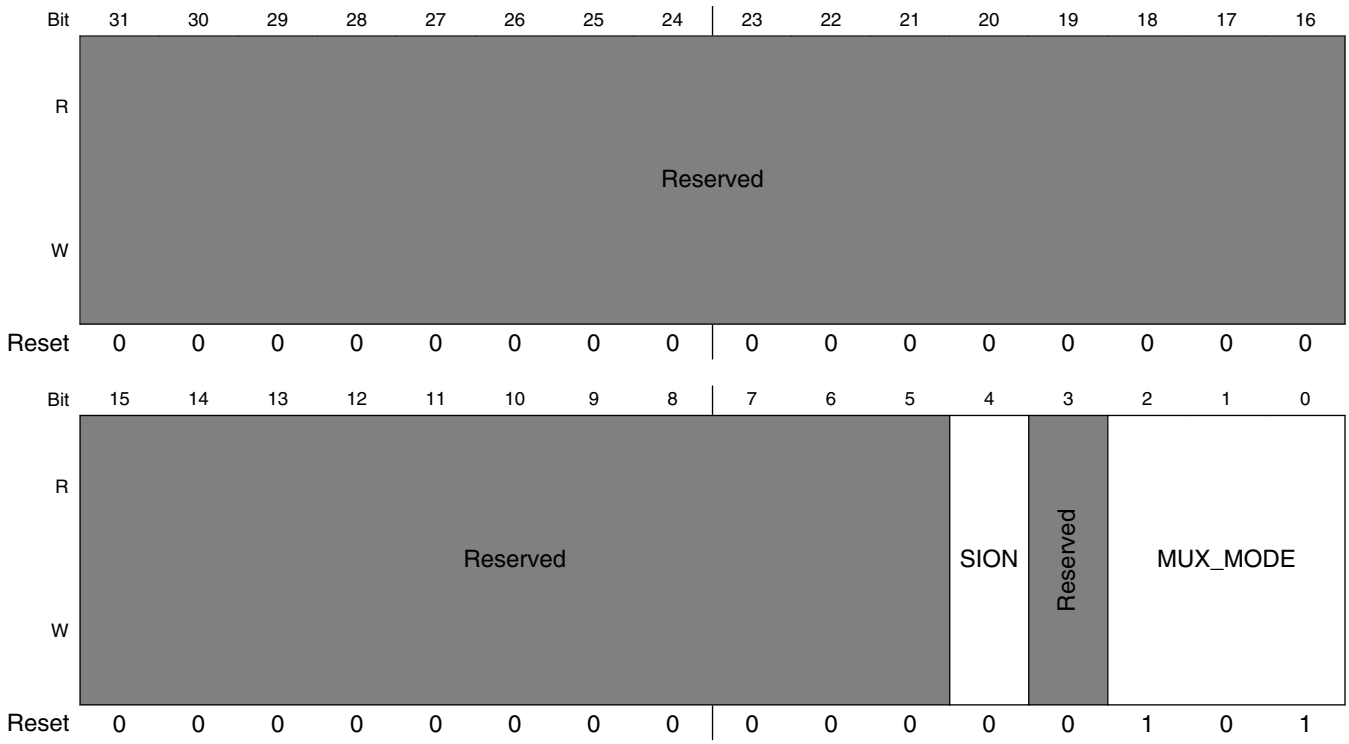
IOMUXC_SW_MUX_CTL_PAD_SD2_DATA2 field descriptions (continued)

Field	Description
	011 **ALT3_GPT4_COMPARE2** — Select mux mode: ALT3 mux port: COMPARE2 of instance: GPT4 100 **ALT4_SIM2_PORT1_SVEN** — Select mux mode: ALT4 mux port: PORT1_SVEN of instance: SIM2 101 **ALT5_GPIO5_IO16** — Select mux mode: ALT5 mux port: IO16 of instance: GPIO5

8.2.7.111 SW_MUX_CTL_PAD_SD2_DATA3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD2_DATA3)

SW_MUX_CTL Register

Address: 3033_0000h base + 1CCh offset = 3033_01CCh



IOMUXC_SW_MUX_CTL_PAD_SD2_DATA3 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD2_DATA3 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_SD2_DATA3 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD2_DATA3. 000 **ALT0_SD2_DATA3** — Select mux mode: ALT0 mux port: DATA3 of instance: SD2 001 **ALT1_SAI2_TX_DATA0** — Select mux mode: ALT1 mux port: TX_DATA0 of instance: SAI2 010 **ALT2_UART4_RTS_B** — Select mux mode: ALT2 mux port: RTS_B of instance: UART4 011 **ALT3_GPT4_COMPARE3** — Select mux mode: ALT3 mux port: COMPARE3 of instance: GPT4 100 **ALT4_SIM2_PORT1_PD** — Select mux mode: ALT4 mux port: PORT1_PD of instance: SIM2 101 **ALT5_GPIO5_IO17** — Select mux mode: ALT5 mux port: IO17 of instance: GPIO5

8.2.7.112 SW_MUX_CTL_PAD_SD3_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_CLK)

SW_MUX_CTL Register

Address: 3033_0000h base + 1D0h offset = 3033_01D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

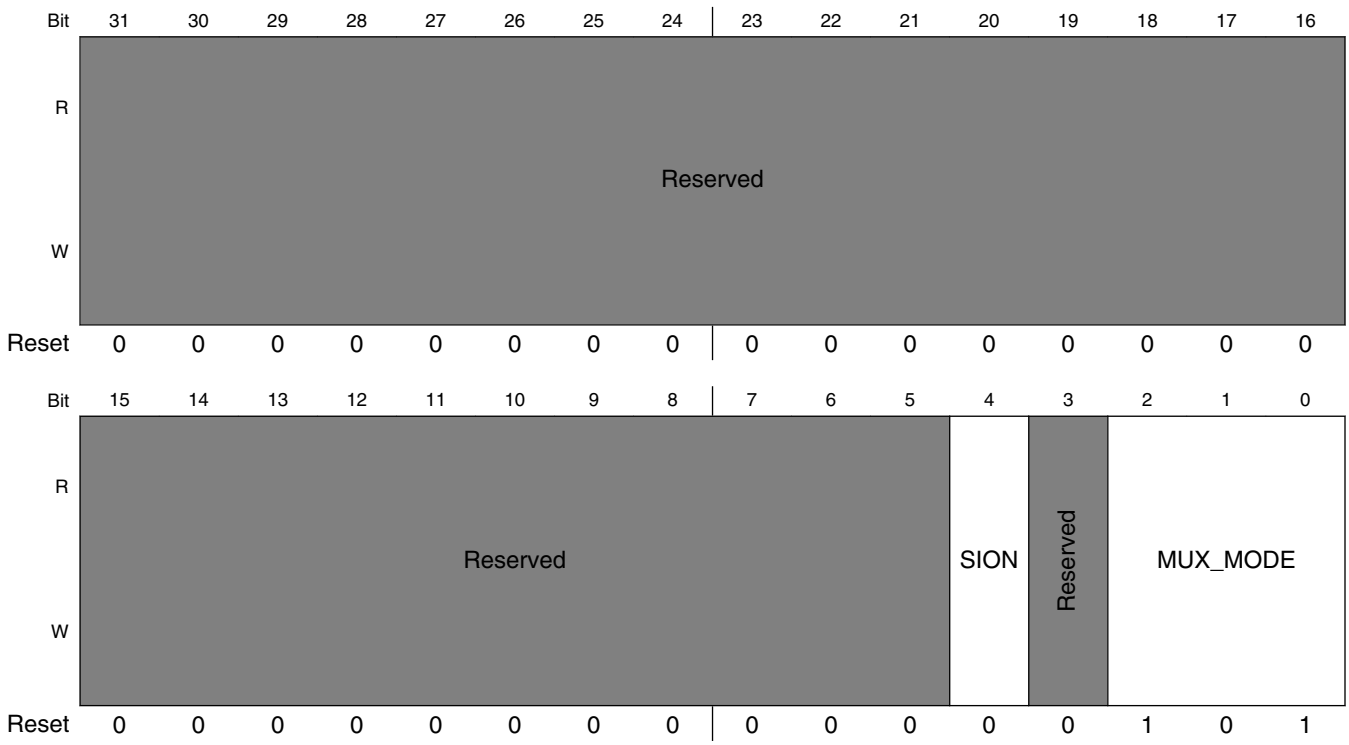
IOMUXC_SW_MUX_CTL_PAD_SD3_CLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD3_CLK 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD3_CLK. 000 **ALT0_SD3_CLK** — Select mux mode: ALT0 mux port: CLK of instance: SD3 001 **ALT1_NAND_CLE** — Select mux mode: ALT1 mux port: CLE of instance: NAND 010 **ALT2_ECSPI4_MISO** — Select mux mode: ALT2 mux port: MISO of instance: ECSPI4 011 **ALT3_SAI3_RX_SYNC** — Select mux mode: ALT3 mux port: RX_SYNC of instance: SAI3 100 **ALT4_GPT3_CLK** — Select mux mode: ALT4 mux port: CLK of instance: GPT3 101 **ALT5_GPIO6_IO0** — Select mux mode: ALT5 mux port: IO0 of instance: GPIO6

8.2.7.113 SW_MUX_CTL_PAD_SD3_CMD SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_CMD)

SW_MUX_CTL Register

Address: 3033_0000h base + 1D4h offset = 3033_01D4h



IOMUXC_SW_MUX_CTL_PAD_SD3_CMD field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD3_CMD 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD3_CMD. 000 **ALT0_SD3_CMD** — Select mux mode: ALT0 mux port: CMD of instance: SD3 001 **ALT1_NAND_ALE** — Select mux mode: ALT1 mux port: ALE of instance: NAND 010 **ALT2_ECSPI4_MOSI** — Select mux mode: ALT2 mux port: MOSI of instance: ECSPI4

Table continues on the next page...

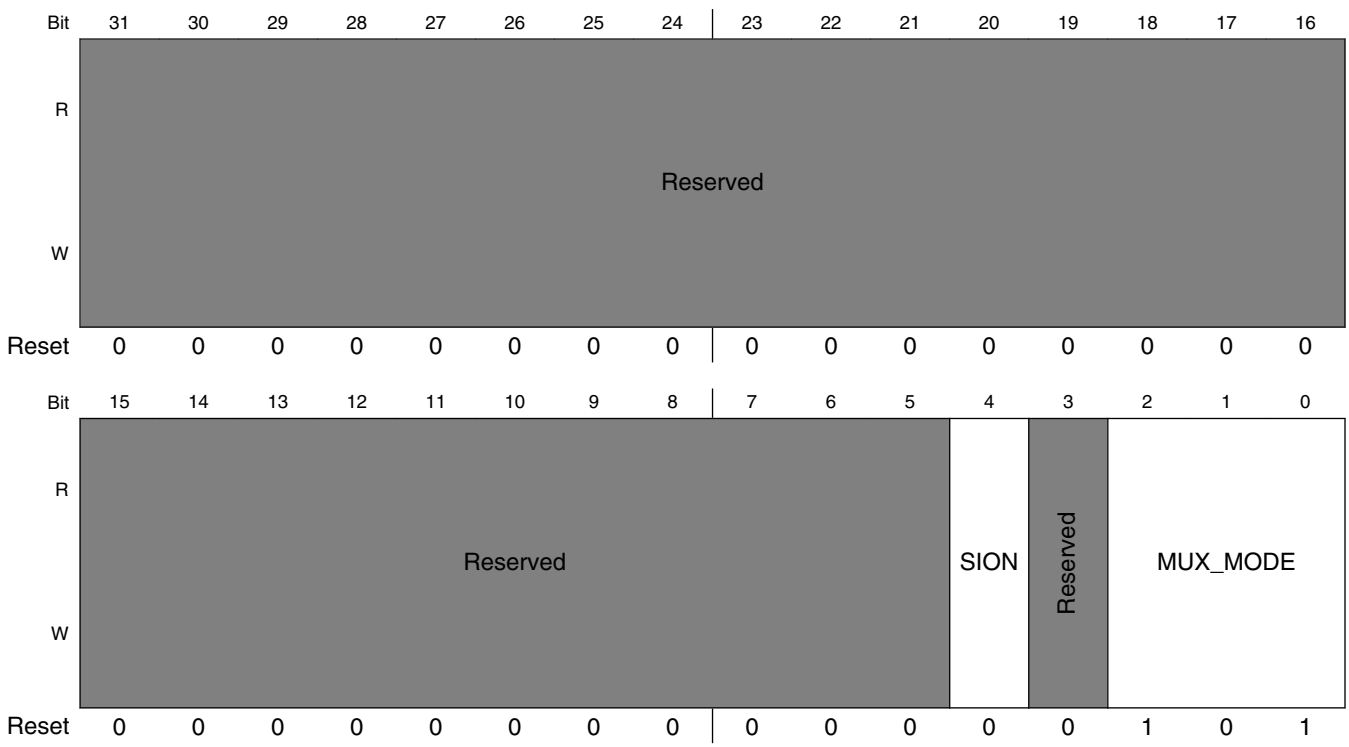
IOMUXC_SW_MUX_CTL_PAD_SD3_CMD field descriptions (continued)

Field	Description
	011 **ALT3_SAI3_RX_BCLK** — Select mux mode: ALT3 mux port: RX_BCLK of instance: SAI3 100 **ALT4_GPT3_CAPTURE1** — Select mux mode: ALT4 mux port: CAPTURE1 of instance: GPT3 101 **ALT5_GPIO6_IO1** — Select mux mode: ALT5 mux port: IO1 of instance: GPIO6

8.2.7.114 SW_MUX_CTL_PAD_SD3_DATA0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA0)

SW_MUX_CTL Register

Address: 3033_0000h base + 1D8h offset = 3033_01D8h



IOMUXC_SW_MUX_CTL_PAD_SD3_DATA0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD3_DATA0 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

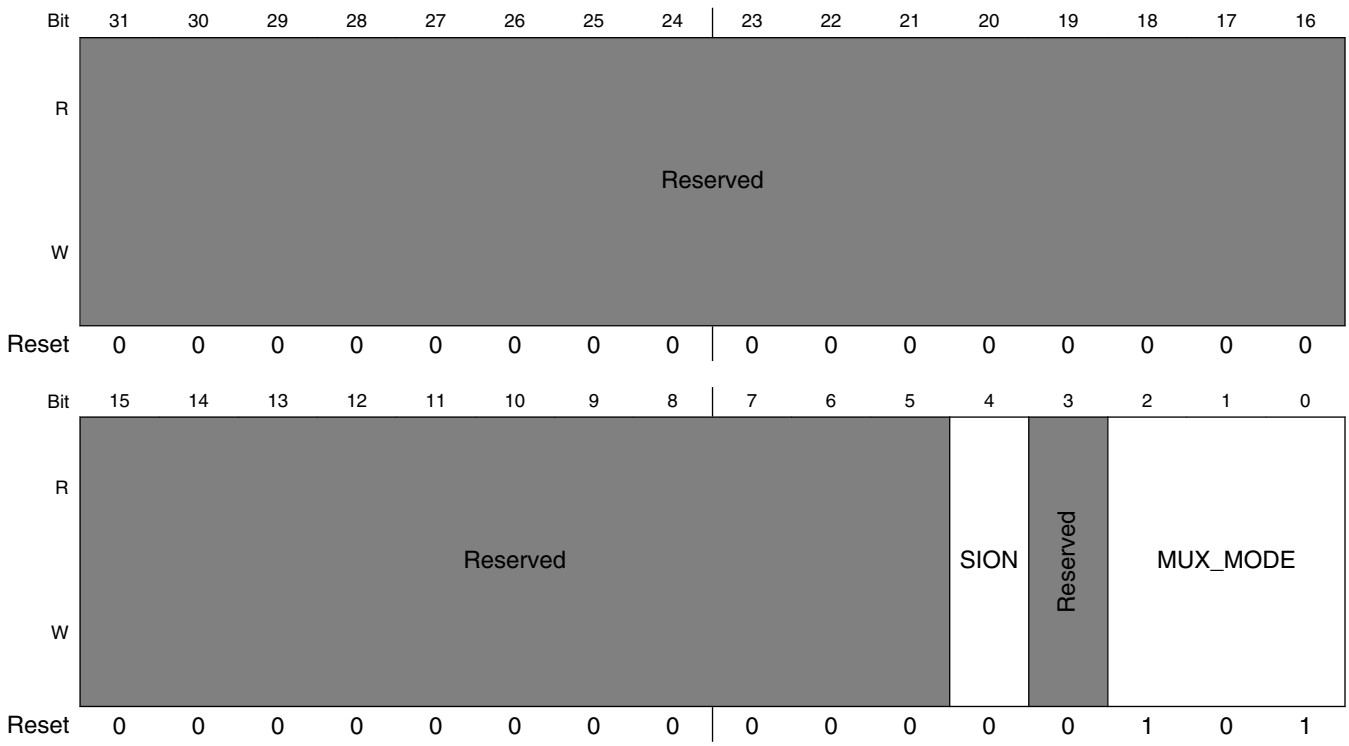
IOMUXC_SW_MUX_CTL_PAD_SD3_DATA0 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD3_DATA0. 000 **ALT0_SD3_DATA0** — Select mux mode: ALT0 mux port: DATA0 of instance: SD3 001 **ALT1_NAND_DATA00** — Select mux mode: ALT1 mux port: DATA00 of instance: NAND 010 **ALT2_ECSPI4_SS0** — Select mux mode: ALT2 mux port: SS0 of instance: ECSPI4 011 **ALT3_SAI3_RX_DATA0** — Select mux mode: ALT3 mux port: RX_DATA0 of instance: SAI3 100 **ALT4_GPT3_CAPTURE2** — Select mux mode: ALT4 mux port: CAPTURE2 of instance: GPT3 101 **ALT5_GPIO6_IO2** — Select mux mode: ALT5 mux port: IO2 of instance: GPIO6

8.2.7.115 SW_MUX_CTL_PAD_SD3_DATA1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA1)

SW_MUX_CTL Register

Address: 3033_0000h base + 1DCh offset = 3033_01DCh



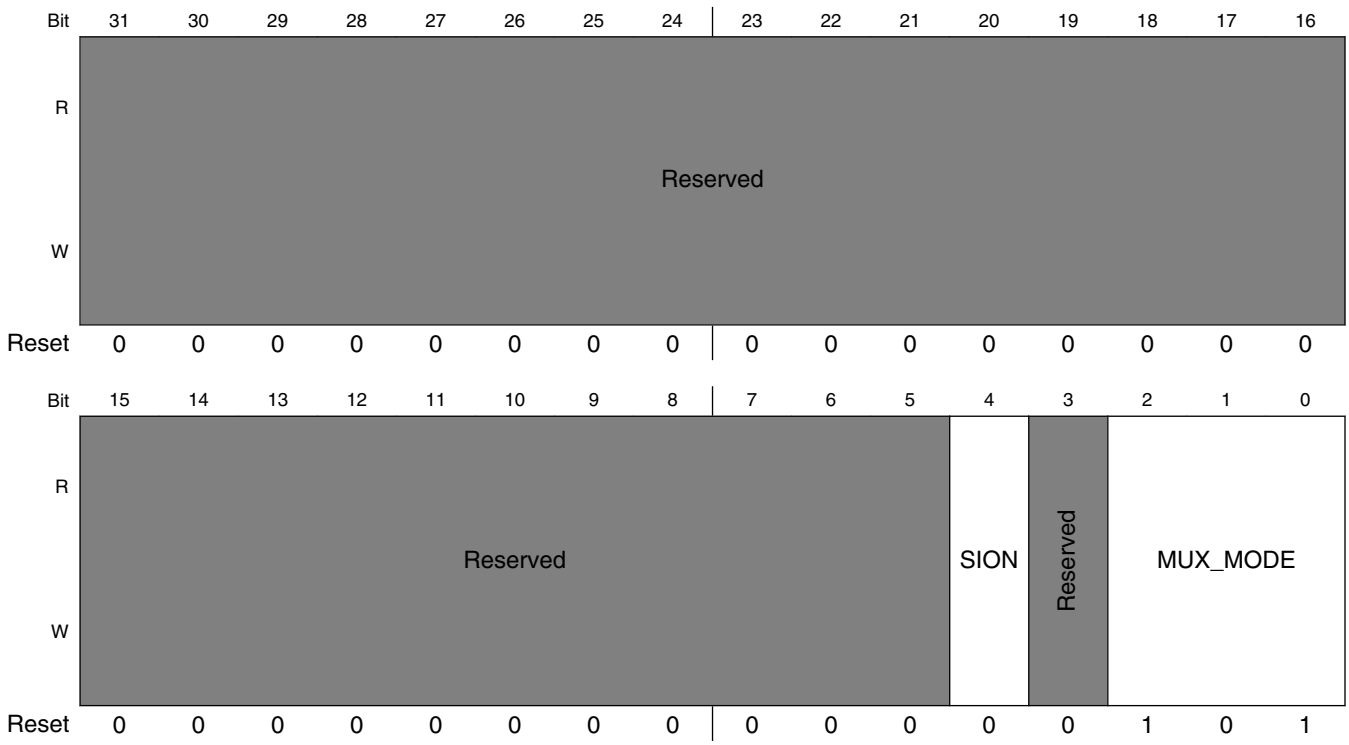
IOMUXC_SW_MUX_CTL_PAD_SD3_DATA1 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD3_DATA1 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD3_DATA1. 000 **ALT0_SD3_DATA1** — Select mux mode: ALT0 mux port: DATA1 of instance: SD3 001 **ALT1_NAND_DATA01** — Select mux mode: ALT1 mux port: DATA01 of instance: NAND 010 **ALT2_ECSPI4_SCLK** — Select mux mode: ALT2 mux port: SCLK of instance: ECSPI4 011 **ALT3_SAI3_TX_BCLK** — Select mux mode: ALT3 mux port: TX_BCLK of instance: SAI3 100 **ALT4_GPT3_COMPARE1** — Select mux mode: ALT4 mux port: COMPARE1 of instance: GPT3 101 **ALT5_GPIO6_IO3** — Select mux mode: ALT5 mux port: IO3 of instance: GPIO6

8.2.7.116 SW_MUX_CTL_PAD_SD3_DATA2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA2)

SW_MUX_CTL Register

Address: 3033_0000h base + 1E0h offset = 3033_01E0h



IOMUXC_SW_MUX_CTL_PAD_SD3_DATA2 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD3_DATA2 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD3_DATA2. 000 **ALT0_SD3_DATA2** — Select mux mode: ALT0 mux port: DATA2 of instance: SD3 001 **ALT1_NAND_DATA02** — Select mux mode: ALT1 mux port: DATA02 of instance: NAND 010 **ALT2_I2C3_SDA** — Select mux mode: ALT2 mux port: SDA of instance: I2C3

Table continues on the next page...

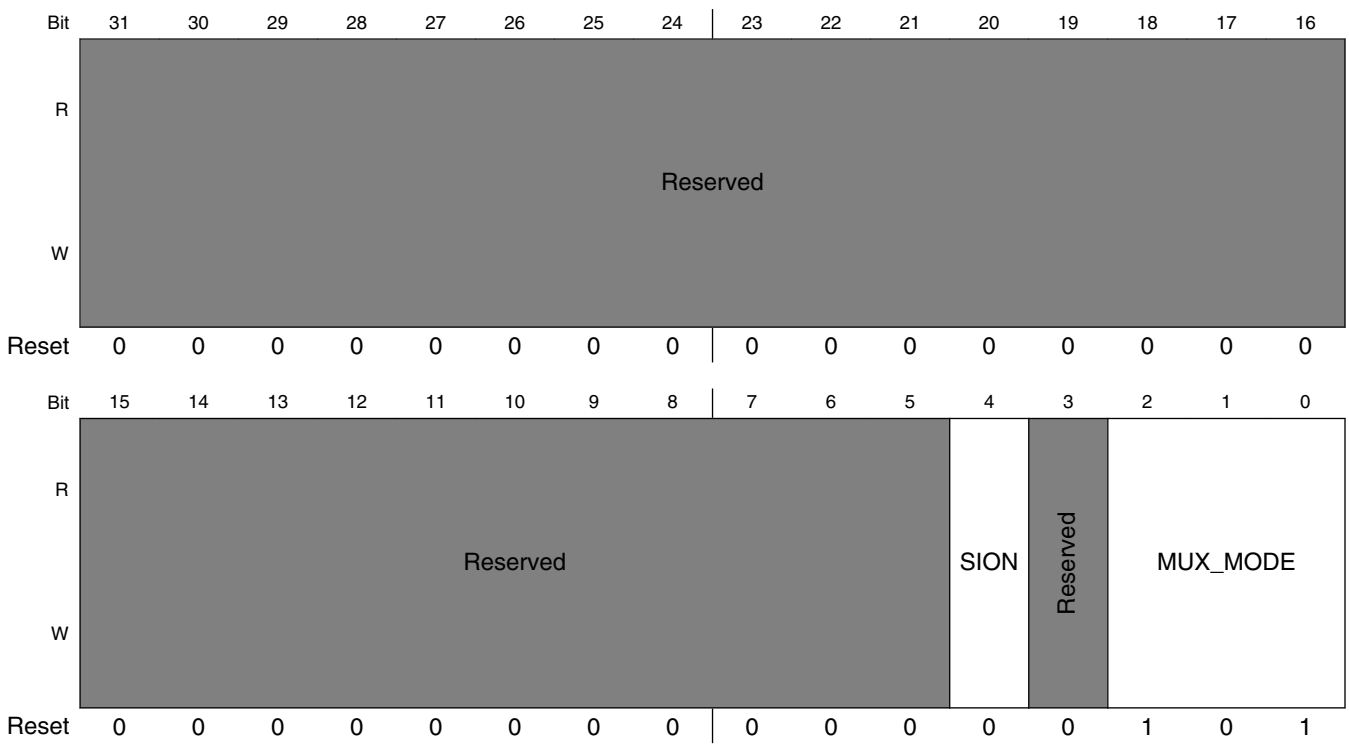
IOMUXC_SW_MUX_CTL_PAD_SD3_DATA2 field descriptions (continued)

Field	Description
	011 **ALT3_SAI3_TX_SYNC** — Select mux mode: ALT3 mux port: TX_SYNC of instance: SAI3 100 **ALT4_GPT3_COMPARE2** — Select mux mode: ALT4 mux port: COMPARE2 of instance: GPT3 101 **ALT5_GPIO6_IO4** — Select mux mode: ALT5 mux port: IO4 of instance: GPIO6

8.2.7.117 SW_MUX_CTL_PAD_SD3_DATA3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA3)

SW_MUX_CTL Register

Address: 3033_0000h base + 1E4h offset = 3033_01E4h



IOMUXC_SW_MUX_CTL_PAD_SD3_DATA3 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD3_DATA3 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_SD3_DATA3 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD3_DATA3. 000 **ALT0_SD3_DATA3** — Select mux mode: ALT0 mux port: DATA3 of instance: SD3 001 **ALT1_NAND_DATA03** — Select mux mode: ALT1 mux port: DATA03 of instance: NAND 010 **ALT2_I2C3_SCL** — Select mux mode: ALT2 mux port: SCL of instance: I2C3 011 **ALT3_SAI3_TX_DATA0** — Select mux mode: ALT3 mux port: TX_DATA0 of instance: SAI3 100 **ALT4_GPT3_COMPARE3** — Select mux mode: ALT4 mux port: COMPARE3 of instance: GPT3 101 **ALT5_GPIO6_IO5** — Select mux mode: ALT5 mux port: IO5 of instance: GPIO6

8.2.7.118 SW_MUX_CTL_PAD_SD3_DATA4 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA4)

SW_MUX_CTL Register

Address: 3033_0000h base + 1E8h offset = 3033_01E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

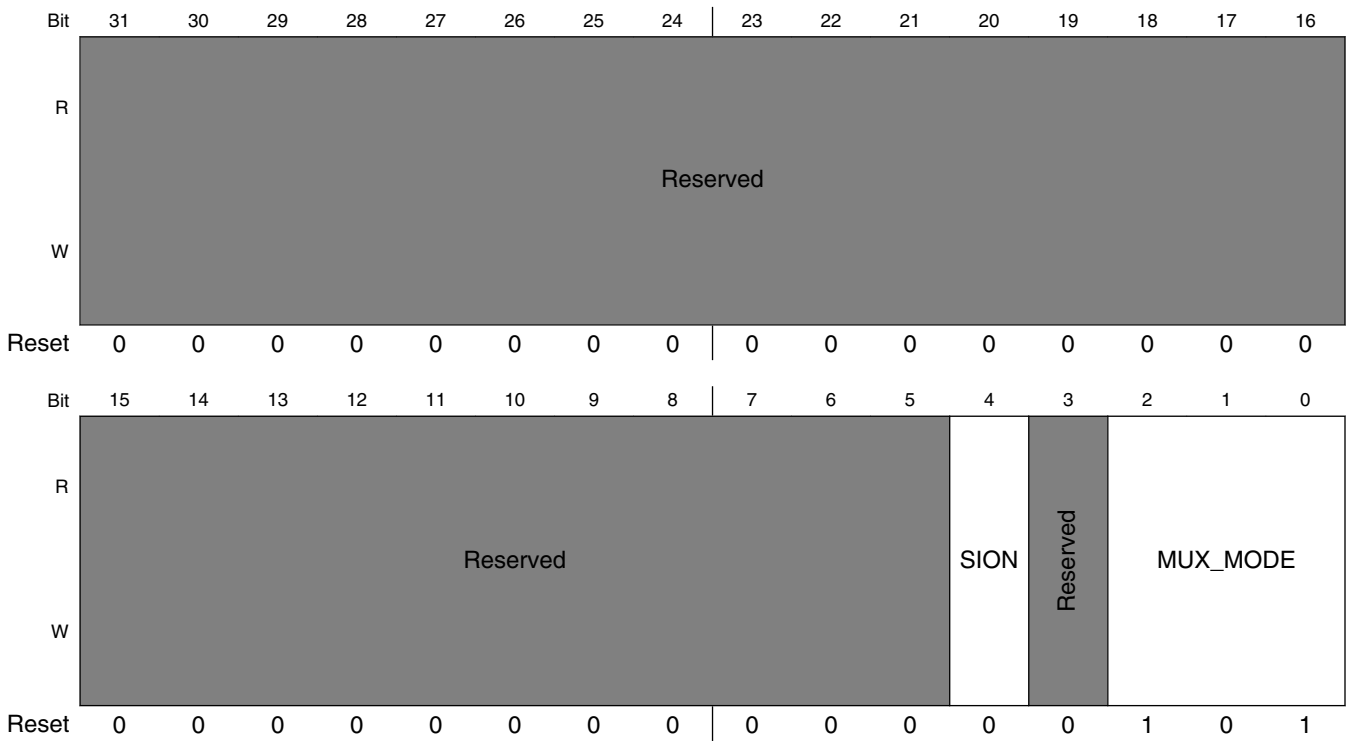
IOMUXC_SW_MUX_CTL_PAD_SD3_DATA4 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD3_DATA4 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: SD3_DATA4. 000 **ALT0_SD3_DATA4** — Select mux mode: ALT0 mux port: DATA4 of instance: SD3 001 **ALT1_NAND_DATA04** — Select mux mode: ALT1 mux port: DATA04 of instance: NAND 011 **ALT3_UART3_RX_DATA** — Select mux mode: ALT3 mux port: RX_DATA of instance: UART3 100 **ALT4_FLEXCAN2_RX** — Select mux mode: ALT4 mux port: RX of instance: FLEXCAN2 101 **ALT5_GPIO6_IO6** — Select mux mode: ALT5 mux port: IO6 of instance: GPIO6

8.2.7.119 SW_MUX_CTL_PAD_SD3_DATA5 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA5)

SW_MUX_CTL Register

Address: 3033_0000h base + 1ECh offset = 3033_01ECh



IOMUXC_SW_MUX_CTL_PAD_SD3_DATA5 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD3_DATA5 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: SD3_DATA5. 000 **ALT0_SD3_DATA5** — Select mux mode: ALT0 mux port: DATA5 of instance: SD3 001 **ALT1_NAND_DATA05** — Select mux mode: ALT1 mux port: DATA05 of instance: NAND 011 **ALT3_UART3_TX_DATA** — Select mux mode: ALT3 mux port: TX_DATA of instance: UART3

Table continues on the next page...

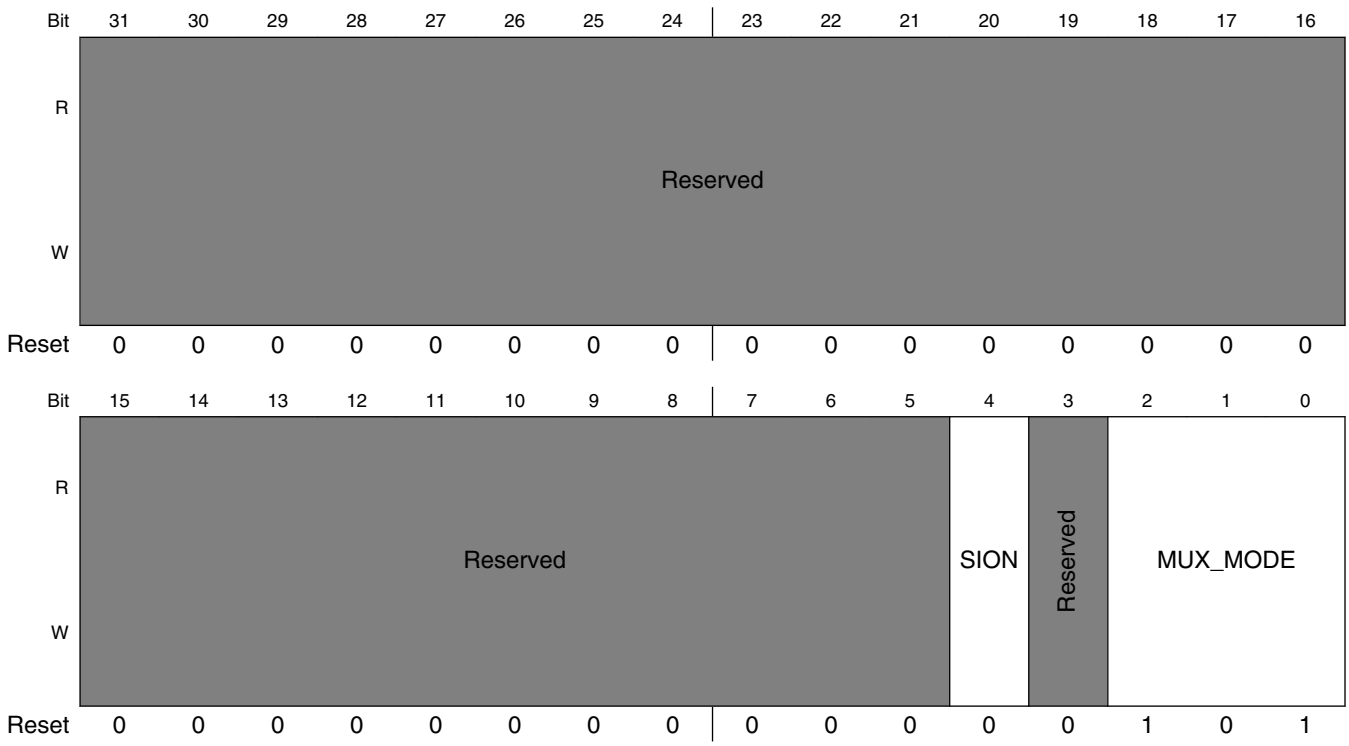
IOMUXC_SW_MUX_CTL_PAD_SD3_DATA5 field descriptions (continued)

Field	Description
	100 **ALT4_FLEXCAN1_TX** — Select mux mode: ALT4 mux port: TX of instance: FLEXCAN1 101 **ALT5_GPIO6_IO7** — Select mux mode: ALT5 mux port: IO7 of instance: GPIO6

8.2.7.120 SW_MUX_CTL_PAD_SD3_DATA6 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA6)

SW_MUX_CTL Register

Address: 3033_0000h base + 1F0h offset = 3033_01F0h



IOMUXC_SW_MUX_CTL_PAD_SD3_DATA6 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD3_DATA6 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_SD3_DATA6 field descriptions (continued)

Field	Description
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD3_DATA6. 000 **ALT0_SD3_DATA6** — Select mux mode: ALT0 mux port: DATA6 of instance: SD3 001 **ALT1_NAND_DATA06** — Select mux mode: ALT1 mux port: DATA06 of instance: NAND 010 **ALT2_SD3_WP** — Select mux mode: ALT2 mux port: WP of instance: SD3 011 **ALT3_UART3_RTS_B** — Select mux mode: ALT3 mux port: RTS_B of instance: UART3 100 **ALT4_FLEXCAN2_TX** — Select mux mode: ALT4 mux port: TX of instance: FLEXCAN2 101 **ALT5_GPIO6_IO8** — Select mux mode: ALT5 mux port: IO8 of instance: GPIO6

8.2.7.121 SW_MUX_CTL_PAD_SD3_DATA7 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_DATA7)

SW_MUX_CTL Register

Address: 3033_0000h base + 1F4h offset = 3033_01F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_SD3_DATA7 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved

Table continues on the next page...

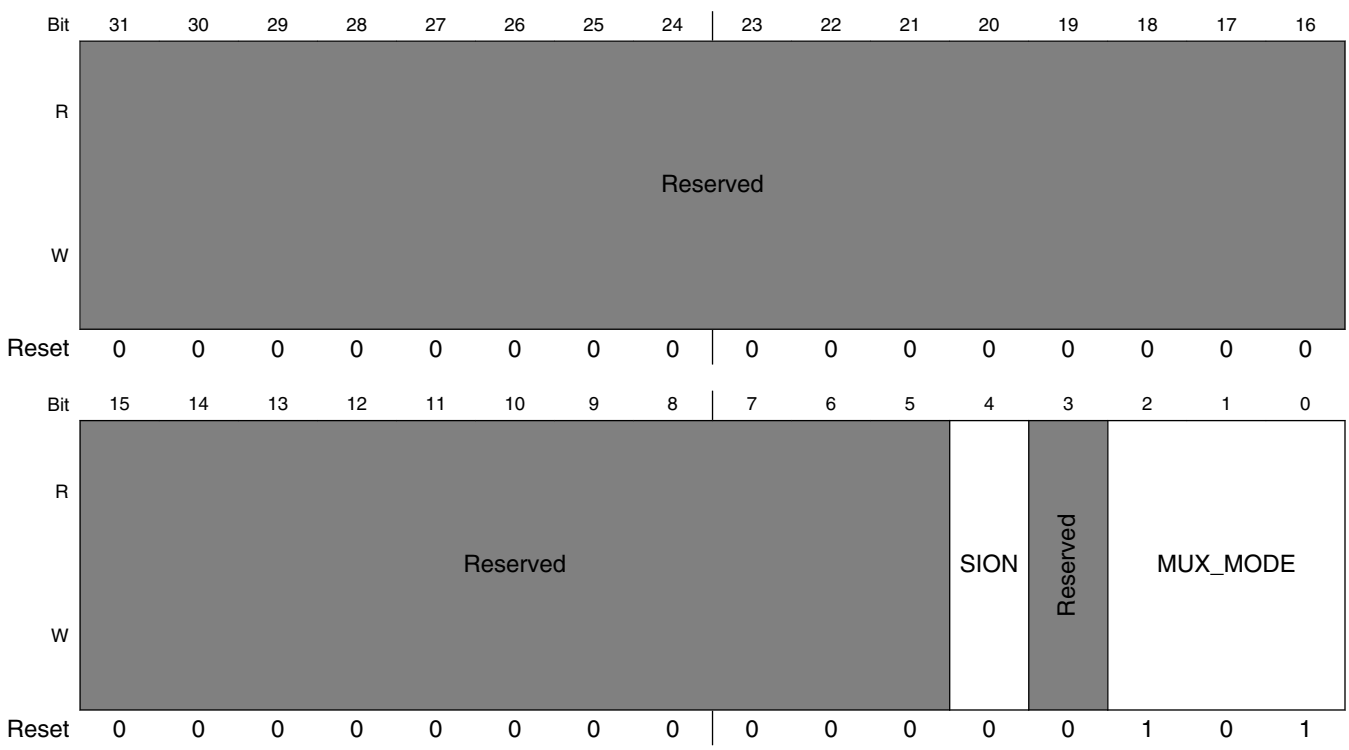
IOMUXC_SW_MUX_CTL_PAD_SD3_DATA7 field descriptions (continued)

Field	Description
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD3_DATA7 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD3_DATA7. 000 **ALT0_SD3_DATA7** — Select mux mode: ALT0 mux port: DATA7 of instance: SD3 001 **ALT1_NAND_DATA07** — Select mux mode: ALT1 mux port: DATA07 of instance: NAND 010 **ALT2_SD3_CD_B** — Select mux mode: ALT2 mux port: CD_B of instance: SD3 011 **ALT3_UART3_CTS_B** — Select mux mode: ALT3 mux port: CTS_B of instance: UART3 100 **ALT4_FLEXCAN1_RX** — Select mux mode: ALT4 mux port: RX of instance: FLEXCAN1 101 **ALT5_GPIO6_IO9** — Select mux mode: ALT5 mux port: IO9 of instance: GPIO6

8.2.7.122 SW_MUX_CTL_PAD_SD3_STROBE SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_STROBE)

SW_MUX_CTL Register

Address: 3033_0000h base + 1F8h offset = 3033_01F8h



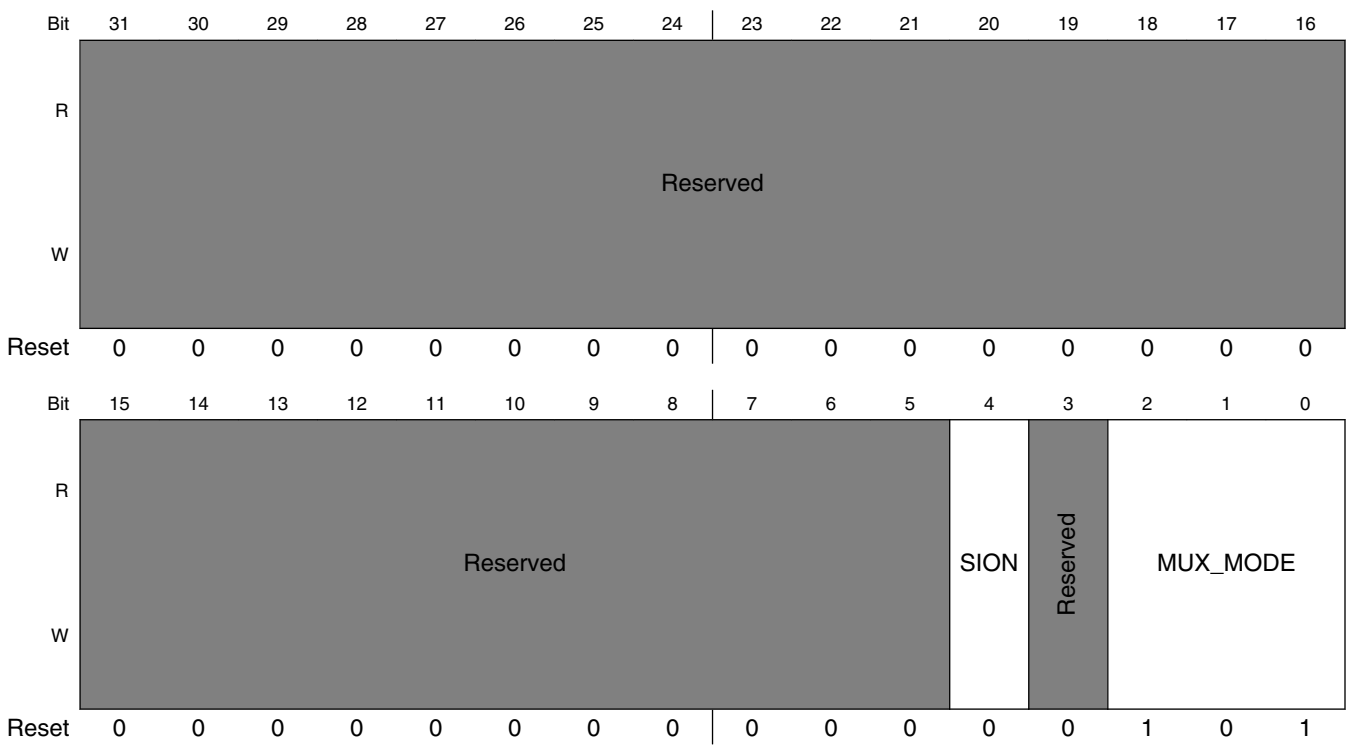
IOMUXC_SW_MUX_CTL_PAD_SD3_STROBE field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD3_STROBE 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SD3_STROBE. 000 **ALT0_SD3_STROBE** — Select mux mode: ALT0 mux port: STROBE of instance: SD3 001 **ALT1_NAND_RE_B** — Select mux mode: ALT1 mux port: RE_B of instance: NAND 101 **ALT5_GPIO6_IO10** — Select mux mode: ALT5 mux port: IO10 of instance: GPIO6

8.2.7.123 SW_MUX_CTL_PAD_SD3_RESET_B SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SD3_RESET_B)

SW_MUX_CTL Register

Address: 3033_0000h base + 1FCh offset = 3033_01FCh



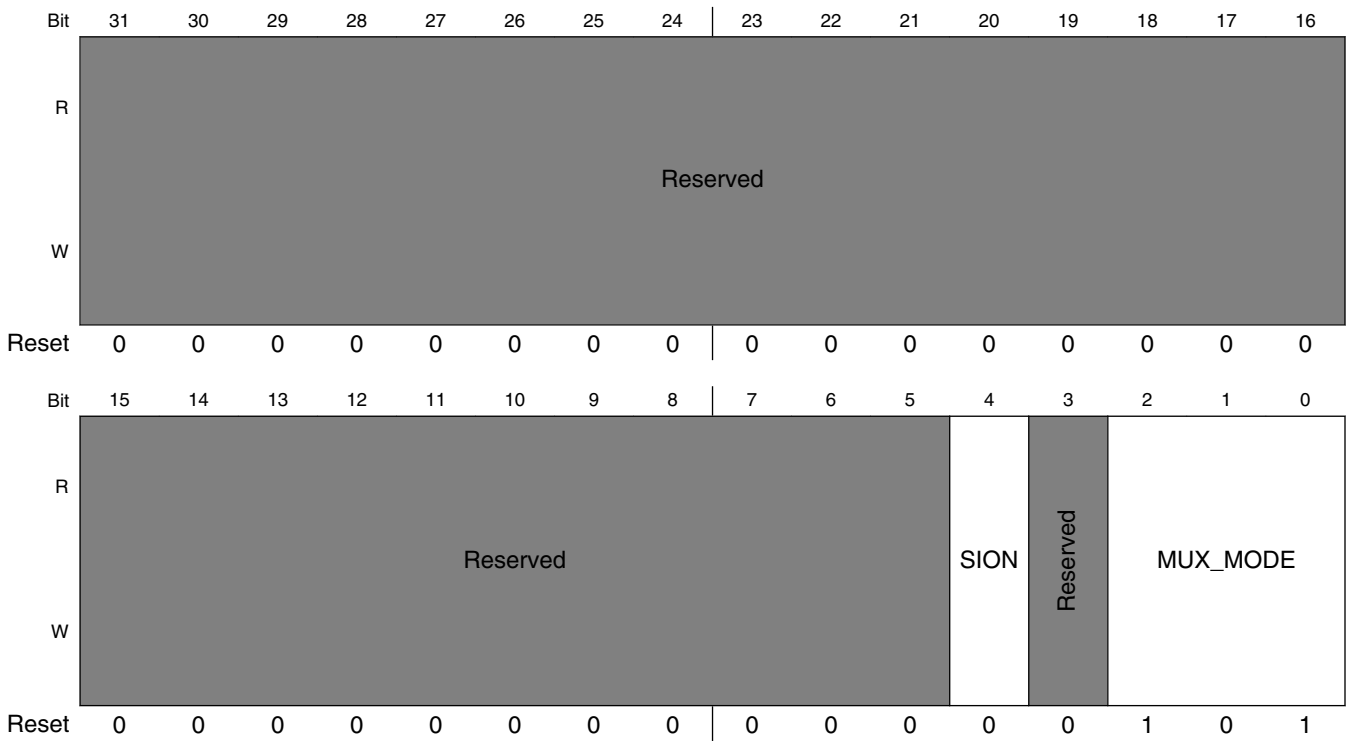
IOMUXC_SW_MUX_CTL_PAD_SD3_RESET_B field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SD3_RESET_B 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: SD3_RESET_B. 000 **ALT0_SD3_RESET_B** — Select mux mode: ALT0 mux port: RESET_B of instance: SD3 001 **ALT1_NAND_WE_B** — Select mux mode: ALT1 mux port: WE_B of instance: NAND 010 **ALT2_SD3_RESET** — Select mux mode: ALT2 mux port: RESET of instance: SD3 011 **ALT3_SAI3_MCLK** — Select mux mode: ALT3 mux port: MCLK of instance: SAI3 101 **ALT5_GPIO6_IO11** — Select mux mode: ALT5 mux port: IO11 of instance: GPIO6

8.2.7.124 SW_MUX_CTL_PAD_SAI1_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI1_RX_DATA)

SW_MUX_CTL Register

Address: 3033_0000h base + 200h offset = 3033_0200h



IOMUXC_SW_MUX_CTL_PAD_SAI1_RX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SAI1_RX_DATA 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SAI1_RX_DATA. 000 **ALT0_SAI1_RX_DATA0** — Select mux mode: ALT0 mux port: RX_DATA0 of instance: SAI1 001 **ALT1_NAND_CE1_B** — Select mux mode: ALT1 mux port: CE1_B of instance: NAND 010 **ALT2_UART5_RX_DATA** — Select mux mode: ALT2 mux port: RX_DATA of instance: UART5

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_SAI1_RX_DATA field descriptions (continued)

Field	Description
	011 **ALT3_FLEXCAN1_RX** — Select mux mode: ALT3 mux port: RX of instance: FLEXCAN1 100 **ALT4_SIM1_PORT1_TRXD** — Select mux mode: ALT4 mux port: PORT1_TRXD of instance: SIM1 101 **ALT5_GPIO6_IO12** — Select mux mode: ALT5 mux port: IO12 of instance: GPIO6 111 **ALT7_SRC_ANY_PU_RESET** — Select mux mode: ALT7 mux port: ANY_PU_RESET of instance: SRC

8.2.7.125 SW_MUX_CTL_PAD_SAI1_TX_BCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI1_TX_BCLK)

SW_MUX_CTL Register

Address: 3033_0000h base + 204h offset = 3033_0204h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_SAI1_TX_BCLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

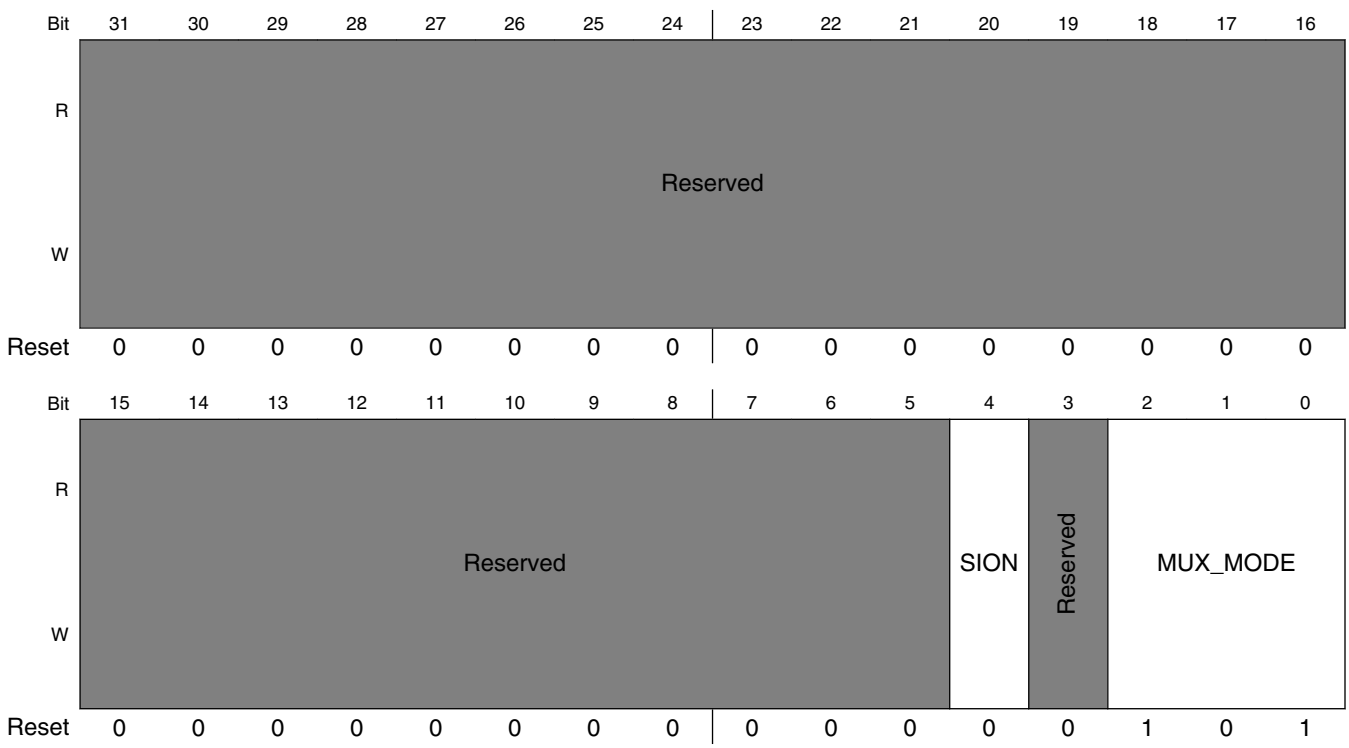
IOMUXC_SW_MUX_CTL_PAD_SAI1_TX_BCLK field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad SAI1_TX_BCLK 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SAI1_TX_BCLK. 000 **ALT0_SAI1_TX_BCLK** — Select mux mode: ALT0 mux port: TX_BCLK of instance: SAI1 001 **ALT1_NAND_CE0_B** — Select mux mode: ALT1 mux port: CE0_B of instance: NAND 010 **ALT2_UART5_TX_DATA** — Select mux mode: ALT2 mux port: TX_DATA of instance: UART5 011 **ALT3_FLEXCAN1_TX** — Select mux mode: ALT3 mux port: TX of instance: FLEXCAN1 100 **ALT4_SIM1_PORT1_CLK** — Select mux mode: ALT4 mux port: PORT1_CLK of instance: SIM1 101 **ALT5_GPIO6_IO13** — Select mux mode: ALT5 mux port: IO13 of instance: GPIO6 111 **ALT7_SRC_EARLY_RESET** — Select mux mode: ALT7 mux port: EARLY_RESET of instance: SRC

8.2.7.126 SW_MUX_CTL_PAD_SAI1_TX_SYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI1_TX_SYNC)

SW_MUX_CTL Register

Address: 3033_0000h base + 208h offset = 3033_0208h



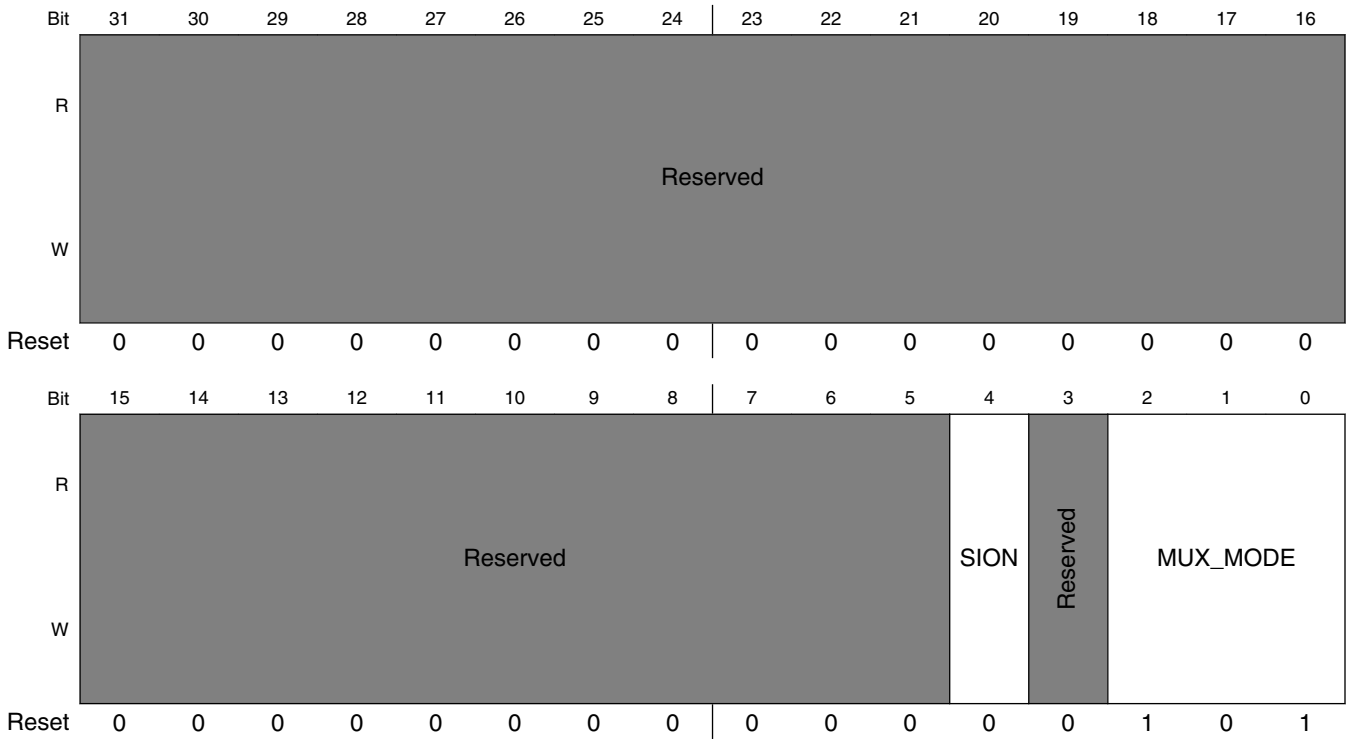
IOMUXC_SW_MUX_CTL_PAD_SAI1_TX_SYNC field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SAI1_TX_SYNC 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SAI1_TX_SYNC. 000 **ALT0_SAI1_TX_SYNC** — Select mux mode: ALT0 mux port: TX_SYNC of instance: SAI1 001 **ALT1_NAND_DQS** — Select mux mode: ALT1 mux port: DQS of instance: NAND 010 **ALT2_UART5_CTS_B** — Select mux mode: ALT2 mux port: CTS_B of instance: UART5 011 **ALT3_FLEXCAN2_RX** — Select mux mode: ALT3 mux port: RX of instance: FLEXCAN2 100 **ALT4_SIM1_PORT1_RST_B** — Select mux mode: ALT4 mux port: PORT1_RST_B of instance: SIM1 101 **ALT5_GPIO6_IO14** — Select mux mode: ALT5 mux port: IO14 of instance: GPIO6 111 **ALT7_SRC_INT_BOOT** — Select mux mode: ALT7 mux port: INT_BOOT of instance: SRC

8.2.7.127 SW_MUX_CTL_PAD_SAI1_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI1_TX_DATA)

SW_MUX_CTL Register

Address: 3033_0000h base + 20Ch offset = 3033_020Ch



IOMUXC_SW_MUX_CTL_PAD_SAI1_TX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SAI1_TX_DATA 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SAI1_TX_DATA. 000 **ALT0_SAI1_TX_DATA0** — Select mux mode: ALT0 mux port: TX_DATA0 of instance: SAI1 001 **ALT1_NAND_READY_B** — Select mux mode: ALT1 mux port: READY_B of instance: NAND 010 **ALT2_UART5_RTS_B** — Select mux mode: ALT2 mux port: RTS_B of instance: UART5

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_SAI1_TX_DATA field descriptions (continued)

Field	Description
	011 **ALT3_FLEXCAN2_TX** — Select mux mode: ALT3 mux port: TX of instance: FLEXCAN2 100 **ALT4_SIM1_PORT1_SVEN** — Select mux mode: ALT4 mux port: PORT1_SVEN of instance: SIM1 101 **ALT5_GPIO6_IO15** — Select mux mode: ALT5 mux port: IO15 of instance: GPIO6 111 **ALT7_SRC_SYSTEM_RESET** — Select mux mode: ALT7 mux port: SYSTEM_RESET of instance: SRC

8.2.7.128 SW_MUX_CTL_PAD_SAI1_RX_SYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI1_RX_SYNC)

SW_MUX_CTL Register

Address: 3033_0000h base + 210h offset = 3033_0210h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved											SION	Reserved	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_SAI1_RX_SYNC field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

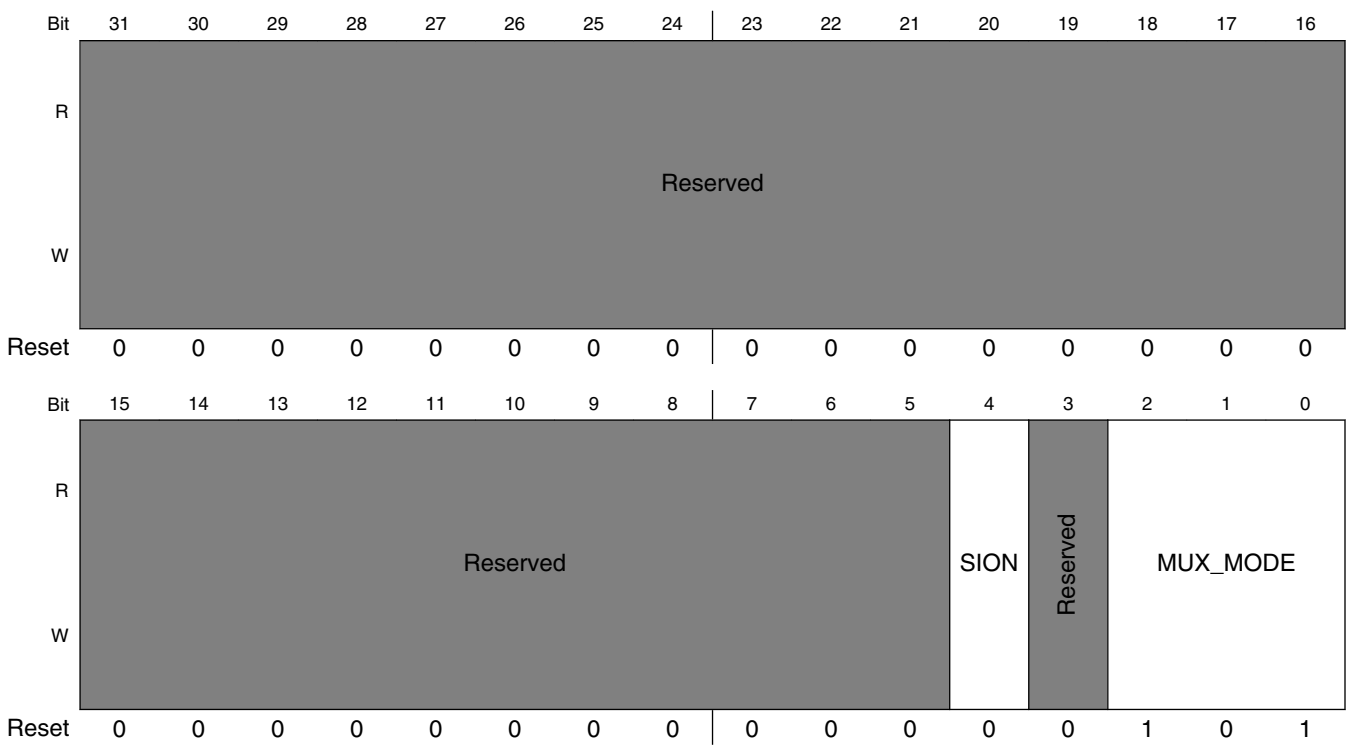
IOMUXC_SW_MUX_CTL_PAD_SAI1_RX_SYNC field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad SAI1_RX_SYNC 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SAI1_RX_SYNC. 000 **ALT0_SAI1_RX_SYNC** — Select mux mode: ALT0 mux port: RX_SYNC of instance: SAI1 001 **ALT1_NAND_CE2_B** — Select mux mode: ALT1 mux port: CE2_B of instance: NAND 010 **ALT2_SAI2_RX_SYNC** — Select mux mode: ALT2 mux port: RX_SYNC of instance: SAI2 011 **ALT3_I2C4_SCL** — Select mux mode: ALT3 mux port: SCL of instance: I2C4 100 **ALT4_SIM1_PORT1_PD** — Select mux mode: ALT4 mux port: PORT1_PD of instance: SIM1 101 **ALT5_GPIO6_IO16** — Select mux mode: ALT5 mux port: IO16 of instance: GPIO6 110 **ALT6_MQS_RIGHT** — Select mux mode: ALT6 mux port: RIGHT of instance: MQS 111 **ALT7_SRC_CA7_RESET_B0** — Select mux mode: ALT7 mux port: CA7_RESET_B0 of instance: SRC

8.2.7.129 SW_MUX_CTL_PAD_SAI1_RX_BCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI1_RX_BCLK)

SW_MUX_CTL Register

Address: 3033_0000h base + 214h offset = 3033_0214h



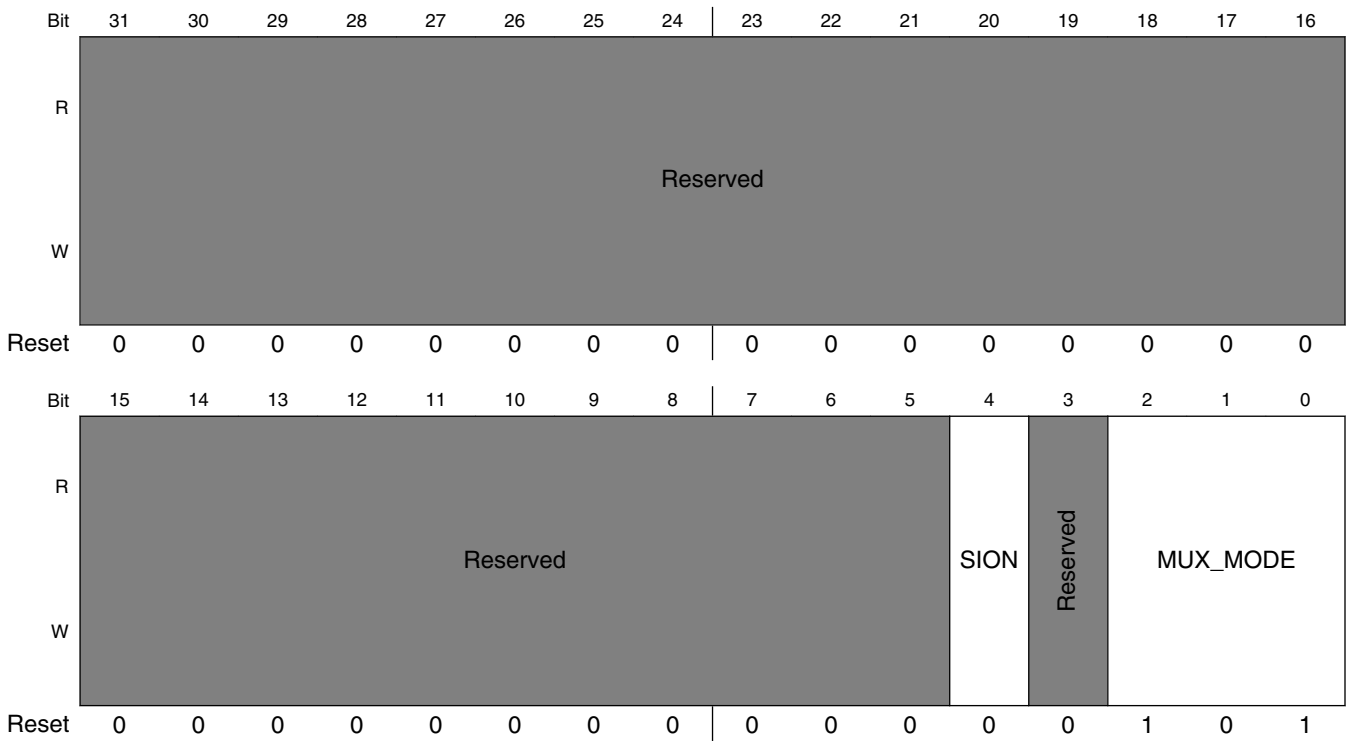
IOMUXC_SW_MUX_CTL_PAD_SAI1_RX_BCLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SAI1_RX_BCLK 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SAI1_RX_BCLK. 000 **ALT0_SAI1_RX_BCLK** — Select mux mode: ALT0 mux port: RX_BCLK of instance: SAI1 001 **ALT1_NAND_CE3_B** — Select mux mode: ALT1 mux port: CE3_B of instance: NAND 010 **ALT2_SAI2_RX_BCLK** — Select mux mode: ALT2 mux port: RX_BCLK of instance: SAI2 011 **ALT3_I2C4_SDA** — Select mux mode: ALT3 mux port: SDA of instance: I2C4 100 **ALT4_FLEXTIMER2_PHA** — Select mux mode: ALT4 mux port: PHA of instance: FLEXTIMER2 101 **ALT5_GPIO6_IO17** — Select mux mode: ALT5 mux port: IO17 of instance: GPIO6 110 **ALT6_MQS_LEFT** — Select mux mode: ALT6 mux port: LEFT of instance: MQS 111 **ALT7_SRC_CA7_RESET_B1** — Select mux mode: ALT7 mux port: CA7_RESET_B1 of instance: SRC

8.2.7.130 SW_MUX_CTL_PAD_SAI1_MCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI1_MCLK)

SW_MUX_CTL Register

Address: 3033_0000h base + 218h offset = 3033_0218h



IOMUXC_SW_MUX_CTL_PAD_SAI1_MCLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SAI1_MCLK 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SAI1_MCLK. 000 **ALT0_SAI1_MCLK** — Select mux mode: ALT0 mux port: MCLK of instance: SAI1 001 **ALT1_NAND_WP_B** — Select mux mode: ALT1 mux port: WP_B of instance: NAND 010 **ALT2_SAI2_MCLK** — Select mux mode: ALT2 mux port: MCLK of instance: SAI2

Table continues on the next page...

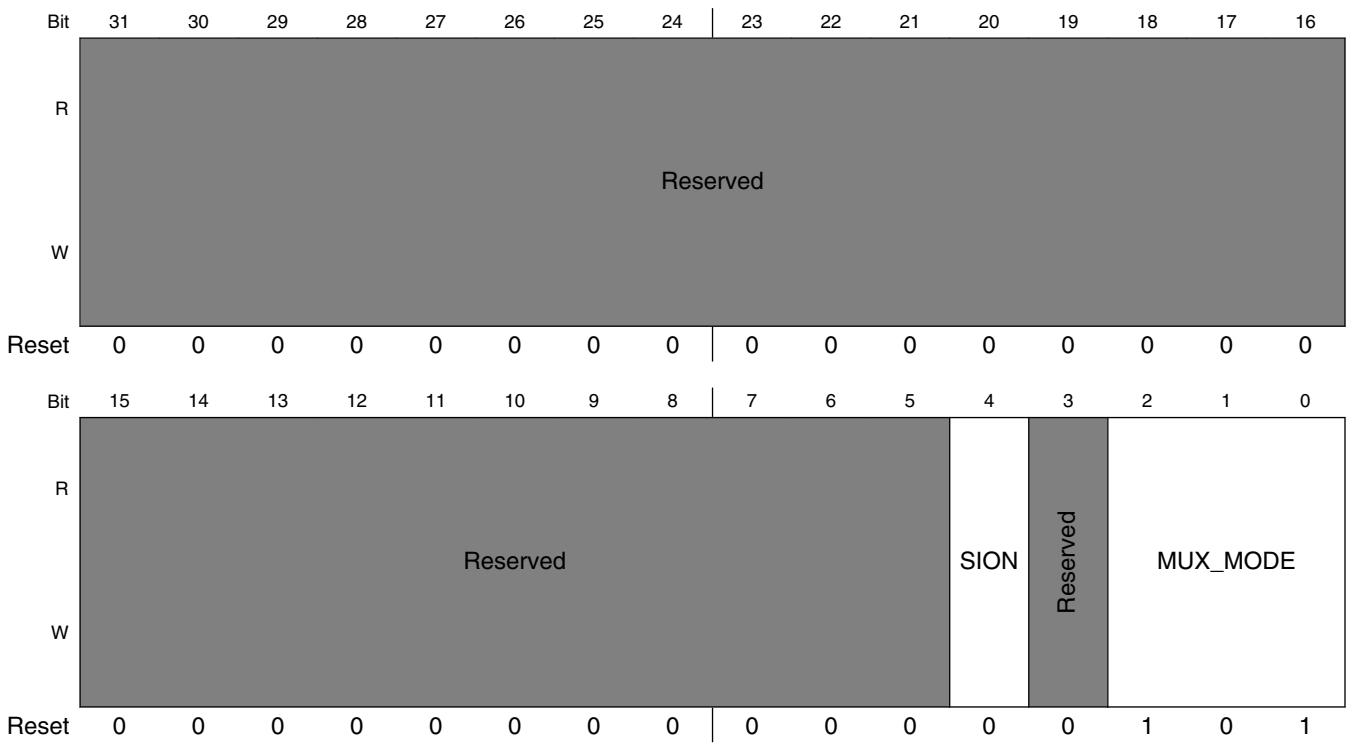
IOMUXC_SW_MUX_CTL_PAD_SAI1_MCLK field descriptions (continued)

Field	Description
	011 **ALT3_CCM_PMIC_READY** — Select mux mode: ALT3 mux port: CCM_PMIC_READY of instance: CCM 100 **ALT4_FLEXTIMER2_PHB** — Select mux mode: ALT4 mux port: PHB of instance: FLEXTIMER2 101 **ALT5_GPIO6_IO18** — Select mux mode: ALT5 mux port: IO18 of instance: GPIO6 111 **ALT7_SRC_TESTER_ACK** — Select mux mode: ALT7 mux port: TESTER_ACK of instance: SRC

8.2.7.131 SW_MUX_CTL_PAD_SAI2_TX_SYNC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI2_TX_SYNC)

SW_MUX_CTL Register

Address: 3033_0000h base + 21Ch offset = 3033_021Ch



IOMUXC_SW_MUX_CTL_PAD_SAI2_TX_SYNC field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_SAI2_TX_SYNC field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad SAI2_TX_SYNC 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SAI2_TX_SYNC. 000 **ALT0_SAI2_TX_SYNC** — Select mux mode: ALT0 mux port: TX_SYNC of instance: SAI2 001 **ALT1_ECSPI3_MISO** — Select mux mode: ALT1 mux port: MISO of instance: ECSPI3 010 **ALT2_UART4_RX_DATA** — Select mux mode: ALT2 mux port: RX_DATA of instance: UART4 011 **ALT3_UART1_CTS_B** — Select mux mode: ALT3 mux port: CTS_B of instance: UART1 100 **ALT4_FLEXTIMER2_CH4** — Select mux mode: ALT4 mux port: CH4 of instance: FLEXTIMER2 101 **ALT5_GPIO6_IO19** — Select mux mode: ALT5 mux port: IO19 of instance: GPIO6

8.2.7.132 SW_MUX_CTL_PAD_SAI2_TX_BCLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI2_TX_BCLK)

SW_MUX_CTL Register

Address: 3033_0000h base + 220h offset = 3033_0220h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

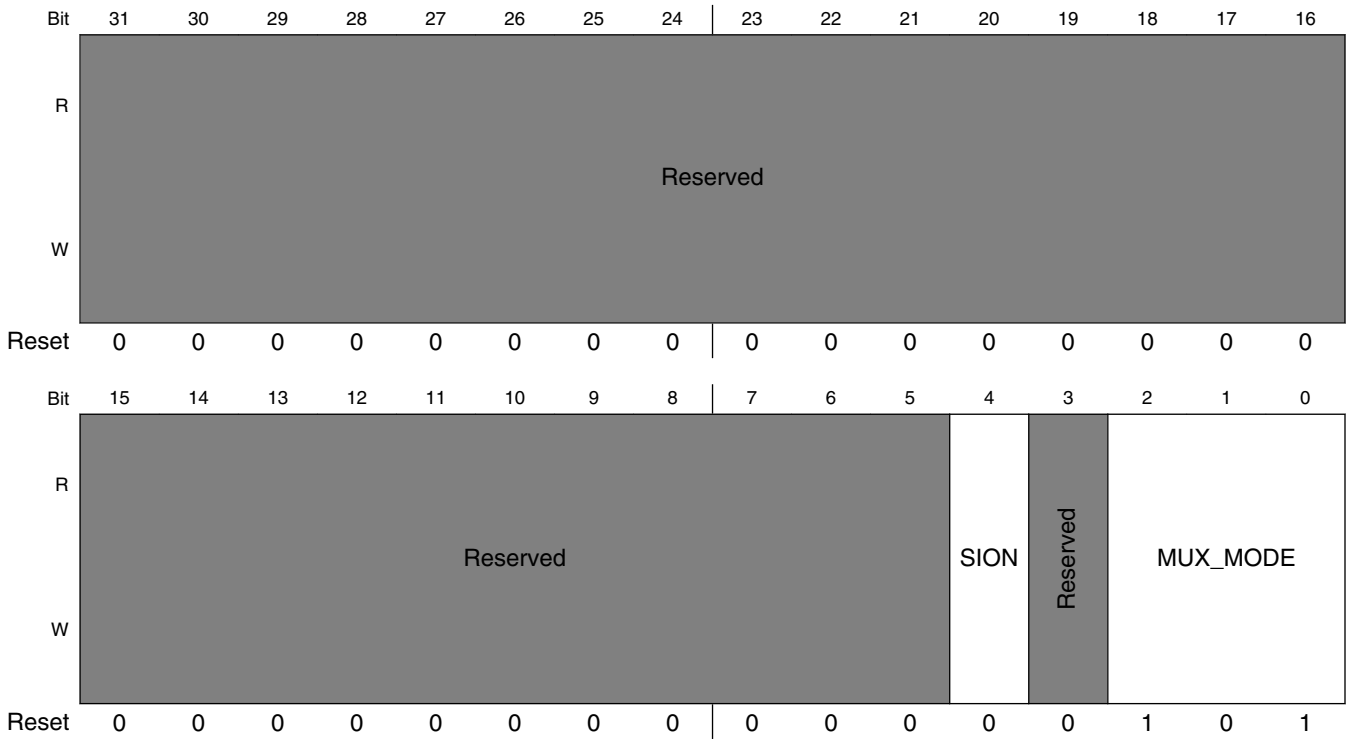
IOMUXC_SW_MUX_CTL_PAD_SAI2_TX_BCLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SAI2_TX_BCLK 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SAI2_TX_BCLK. 000 **ALT0_SAI2_TX_BCLK** — Select mux mode: ALT0 mux port: TX_BCLK of instance: SAI2 001 **ALT1_ECSPI3_MOSI** — Select mux mode: ALT1 mux port: MOSI of instance: ECSPI3 010 **ALT2_UART4_TX_DATA** — Select mux mode: ALT2 mux port: TX_DATA of instance: UART4 011 **ALT3_UART1_RTS_B** — Select mux mode: ALT3 mux port: RTS_B of instance: UART1 100 **ALT4_FLEXTIMER2_CH5** — Select mux mode: ALT4 mux port: CH5 of instance: FLEXTIMER2 101 **ALT5_GPIO6_IO20** — Select mux mode: ALT5 mux port: IO20 of instance: GPIO6

8.2.7.133 SW_MUX_CTL_PAD_SAI2_RX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI2_RX_DATA)

SW_MUX_CTL Register

Address: 3033_0000h base + 224h offset = 3033_0224h



IOMUXC_SW_MUX_CTL_PAD_SAI2_RX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SAI2_RX_DATA 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SAI2_RX_DATA. 000 **ALT0_SAI2_RX_DATA0** — Select mux mode: ALT0 mux port: RX_DATA0 of instance: SAI2 001 **ALT1_ECSPI3_SCLK** — Select mux mode: ALT1 mux port: SCLK of instance: ECSPI3 010 **ALT2_UART4_CTS_B** — Select mux mode: ALT2 mux port: CTS_B of instance: UART4

Table continues on the next page...

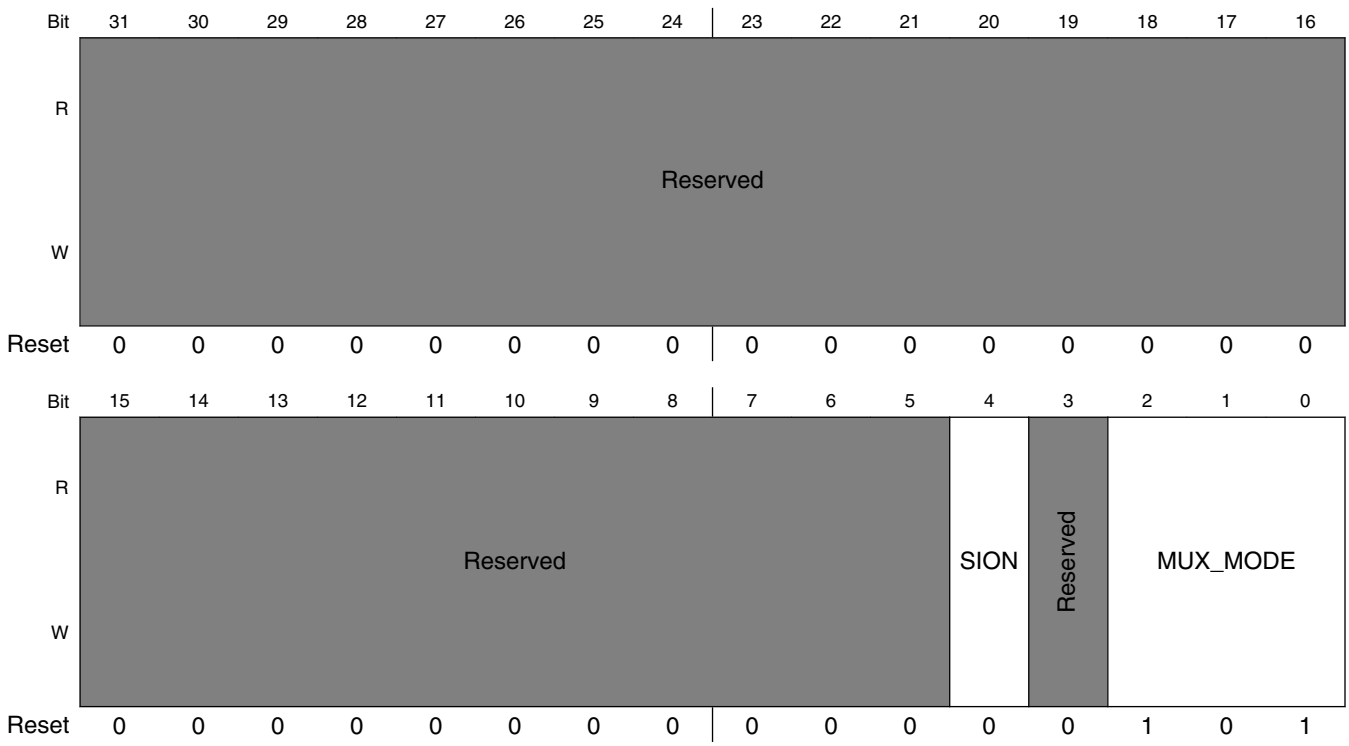
IOMUXC_SW_MUX_CTL_PAD_SAI2_RX_DATA field descriptions (continued)

Field	Description
	011 **ALT3_UART2_CTS_B** — Select mux mode: ALT3 mux port: CTS_B of instance: UART2 100 **ALT4_FLEXTIMER2_CH6** — Select mux mode: ALT4 mux port: CH6 of instance: FLEXTIMER2 101 **ALT5_GPIO6_IO21** — Select mux mode: ALT5 mux port: IO21 of instance: GPIO6 110 **ALT6_KPP_COL7** — Select mux mode: ALT6 mux port: COL7 of instance: KPP

8.2.7.134 SW_MUX_CTL_PAD_SAI2_TX_DATA SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_SAI2_TX_DATA)

SW_MUX_CTL Register

Address: 3033_0000h base + 228h offset = 3033_0228h



IOMUXC_SW_MUX_CTL_PAD_SAI2_TX_DATA field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad SAI2_TX_DATA 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_SAI2_TX_DATA field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SAI2_TX_DATA. 000 **ALT0_SAI2_TX_DATA0** — Select mux mode: ALT0 mux port: TX_DATA0 of instance: SAI2 001 **ALT1_ECSPI3_SS0** — Select mux mode: ALT1 mux port: SS0 of instance: ECSPI3 010 **ALT2_UART4_RTS_B** — Select mux mode: ALT2 mux port: RTS_B of instance: UART4 011 **ALT3_UART2_RTS_B** — Select mux mode: ALT3 mux port: RTS_B of instance: UART2 100 **ALT4_FLEXTIMER2_CH7** — Select mux mode: ALT4 mux port: CH7 of instance: FLEXTIMER2 101 **ALT5_GPIO6_IO22** — Select mux mode: ALT5 mux port: IO22 of instance: GPIO6 110 **ALT6_KPP_ROW7** — Select mux mode: ALT6 mux port: ROW7 of instance: KPP

8.2.7.135 SW_MUX_CTL_PAD_ENET1_RGMII_RD0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RD0)

SW_MUX_CTL Register

Address: 3033_0000h base + 22Ch offset = 3033_022Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

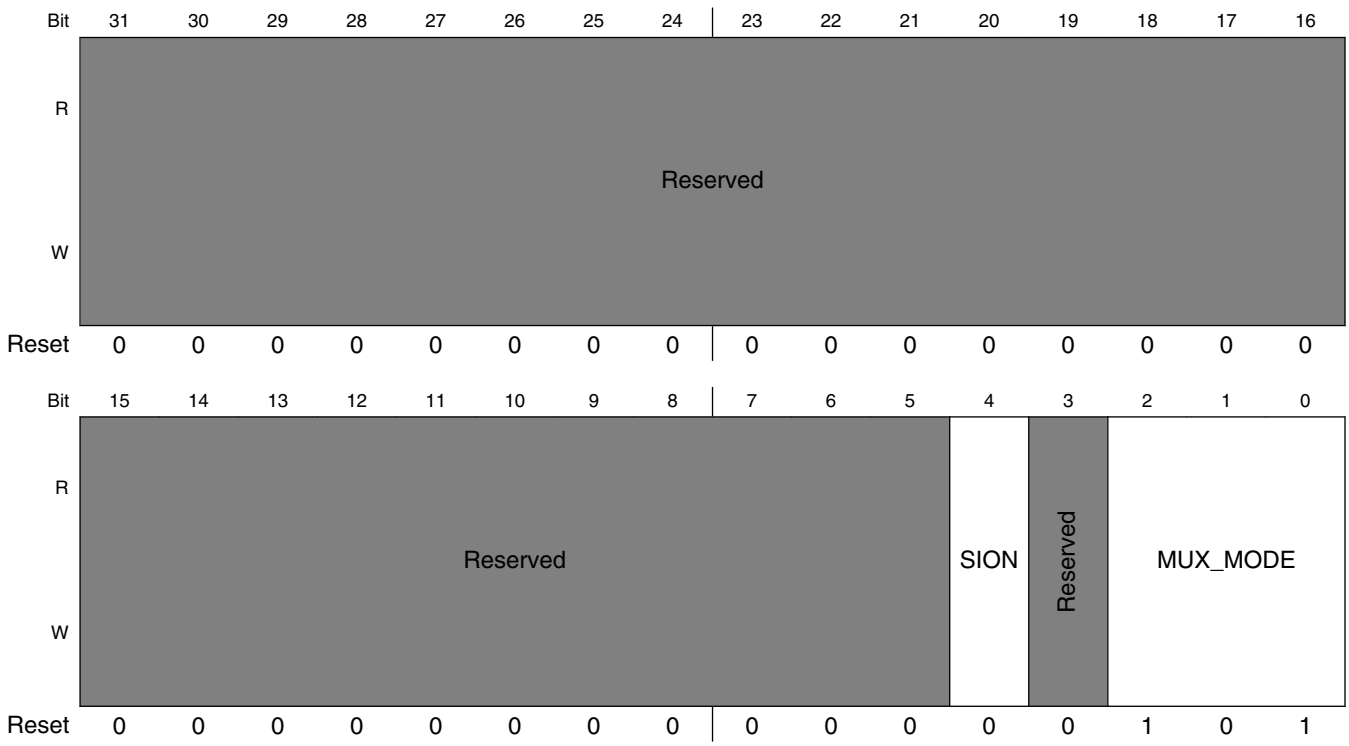
IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RD0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_RGMII_RD0 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ENET1_RGMII_RD0. 000 **ALT0_ENET1_RGMII_RD0** — Select mux mode: ALT0 mux port: RGMII_RD0 of instance: ENET1 001 **ALT1_PWM1_OUT** — Select mux mode: ALT1 mux port: OUT of instance: PWM1 010 **ALT2_I2C3_SCL** — Select mux mode: ALT2 mux port: SCL of instance: I2C3 011 **ALT3_UART1_CTS_B** — Select mux mode: ALT3 mux port: CTS_B of instance: UART1 100 **ALT4_EPDC_VCOM0** — Select mux mode: ALT4 mux port: VCOM0 of instance: EPDC 101 **ALT5_GPIO7_IO0** — Select mux mode: ALT5 mux port: IO0 of instance: GPIO7 110 **ALT6_KPP_ROW3** — Select mux mode: ALT6 mux port: ROW3 of instance: KPP

8.2.7.136 SW_MUX_CTL_PAD_ENET1_RGMII_RD1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RD1)

SW_MUX_CTL Register

Address: 3033_0000h base + 230h offset = 3033_0230h



IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RD1 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_RGMII_RD1 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ENET1_RGMII_RD1. 000 **ALT0_ENET1_RGMII_RD1** — Select mux mode: ALT0 mux port: RGMII_RD1 of instance: ENET1 001 **ALT1_PWM2_OUT** — Select mux mode: ALT1 mux port: OUT of instance: PWM2

Table continues on the next page...

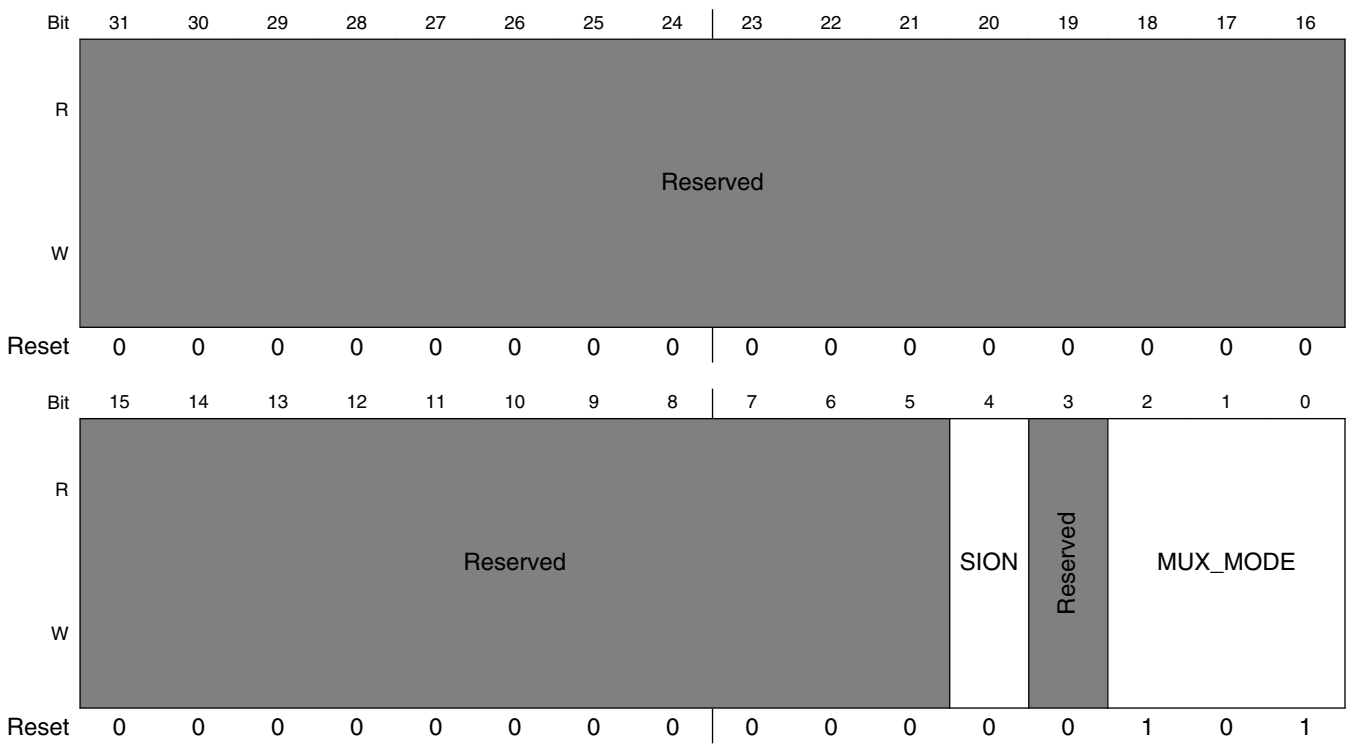
IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RD1 field descriptions (continued)

Field	Description
	010 **ALT2_I2C3_SDA** — Select mux mode: ALT2 mux port: SDA of instance: I2C3 011 **ALT3_UART1_RTS_B** — Select mux mode: ALT3 mux port: RTS_B of instance: UART1 100 **ALT4_EPDC_VCOM1** — Select mux mode: ALT4 mux port: VCOM1 of instance: EPDC 101 **ALT5_GPIO7_IO1** — Select mux mode: ALT5 mux port: IO1 of instance: GPIO7 110 **ALT6_KPP_COL3** — Select mux mode: ALT6 mux port: COL3 of instance: KPP

8.2.7.137 SW_MUX_CTL_PAD_ENET1_RGMII_RD2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RD2)

SW_MUX_CTL Register

Address: 3033_0000h base + 234h offset = 3033_0234h



IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RD2 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RD2 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad ENET1_RGMII_RD2 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ENET1_RGMII_RD2. 000 **ALT0_ENET1_RGMII_RD2** — Select mux mode: ALT0 mux port: RGMII_RD2 of instance: ENET1 001 **ALT1_FLEXCAN1_RX** — Select mux mode: ALT1 mux port: RX of instance: FLEXCAN1 010 **ALT2_ECSPI2_SCLK** — Select mux mode: ALT2 mux port: SCLK of instance: ECSPI2 011 **ALT3_UART1_RX_DATA** — Select mux mode: ALT3 mux port: RX_DATA of instance: UART1 100 **ALT4_EPDC_SDCE4** — Select mux mode: ALT4 mux port: SDCE4 of instance: EPDC 101 **ALT5_GPIO7_IO2** — Select mux mode: ALT5 mux port: IO2 of instance: GPIO7 110 **ALT6_KPP_ROW2** — Select mux mode: ALT6 mux port: ROW2 of instance: KPP

8.2.7.138 SW_MUX_CTL_PAD_ENET1_RGMII_RD3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RD3)

SW_MUX_CTL Register

Address: 3033_0000h base + 238h offset = 3033_0238h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

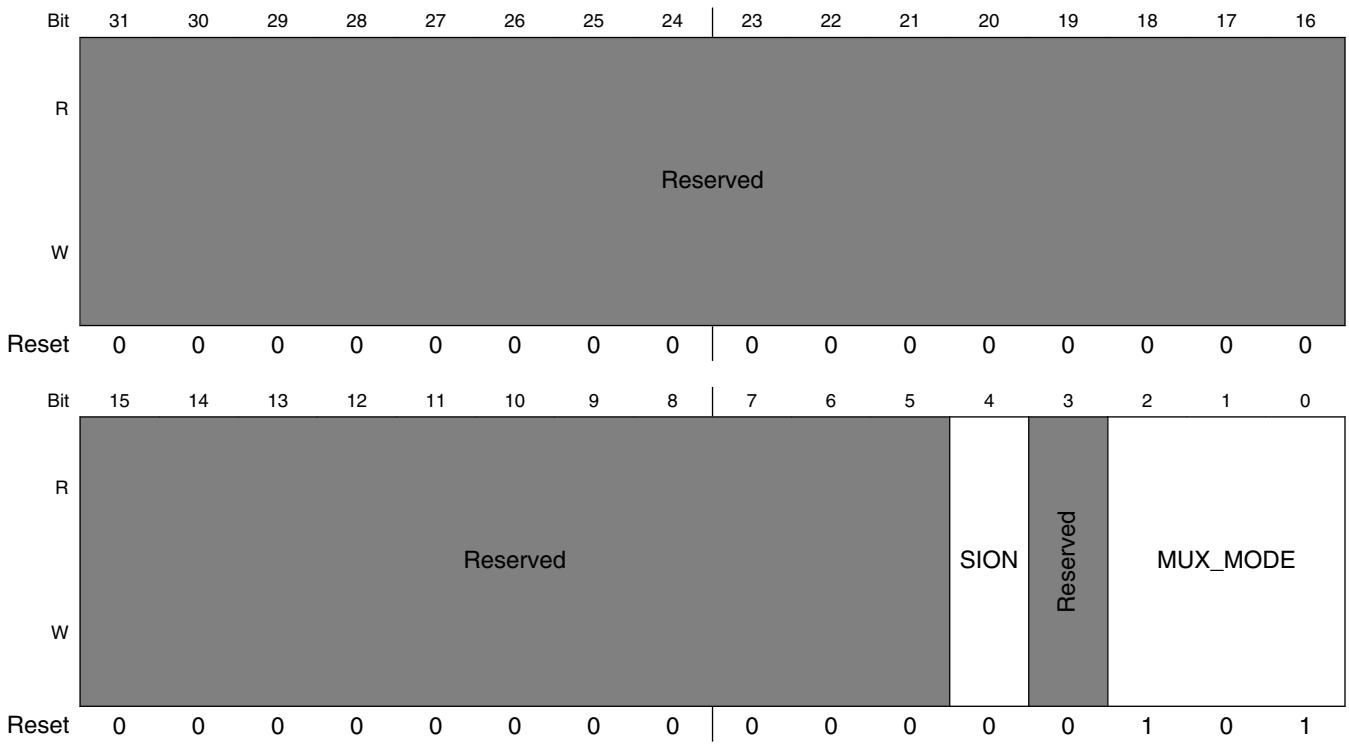
IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RD3 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_RGMII_RD3 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ENET1_RGMII_RD3. 000 **ALT0_ENET1_RGMII_RD3** — Select mux mode: ALT0 mux port: RGMII_RD3 of instance: ENET1 001 **ALT1_FLEXCAN1_TX** — Select mux mode: ALT1 mux port: TX of instance: FLEXCAN1 010 **ALT2_ECSPI2_MOSI** — Select mux mode: ALT2 mux port: MOSI of instance: ECSPI2 011 **ALT3_UART1_TX_DATA** — Select mux mode: ALT3 mux port: TX_DATA of instance: UART1 100 **ALT4_EPDC_SDCE5** — Select mux mode: ALT4 mux port: SDCE5 of instance: EPDC 101 **ALT5_GPIO7_IO3** — Select mux mode: ALT5 mux port: IO3 of instance: GPIO7 110 **ALT6_KPP_COL2** — Select mux mode: ALT6 mux port: COL2 of instance: KPP

8.2.7.139 SW_MUX_CTL_PAD_ENET1_RGMII_RX_CTL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RX_CTL)

SW_MUX_CTL Register

Address: 3033_0000h base + 23Ch offset = 3033_023Ch



IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RX_CTL field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_RGMII_RX_CTL 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: ENET1_RGMII_RX_CTL. 000 **ALT0_ENET1_RGMII_RX_CTL** — Select mux mode: ALT0 mux port: RGMII_RX_CTL of instance: ENET1

Table continues on the next page...

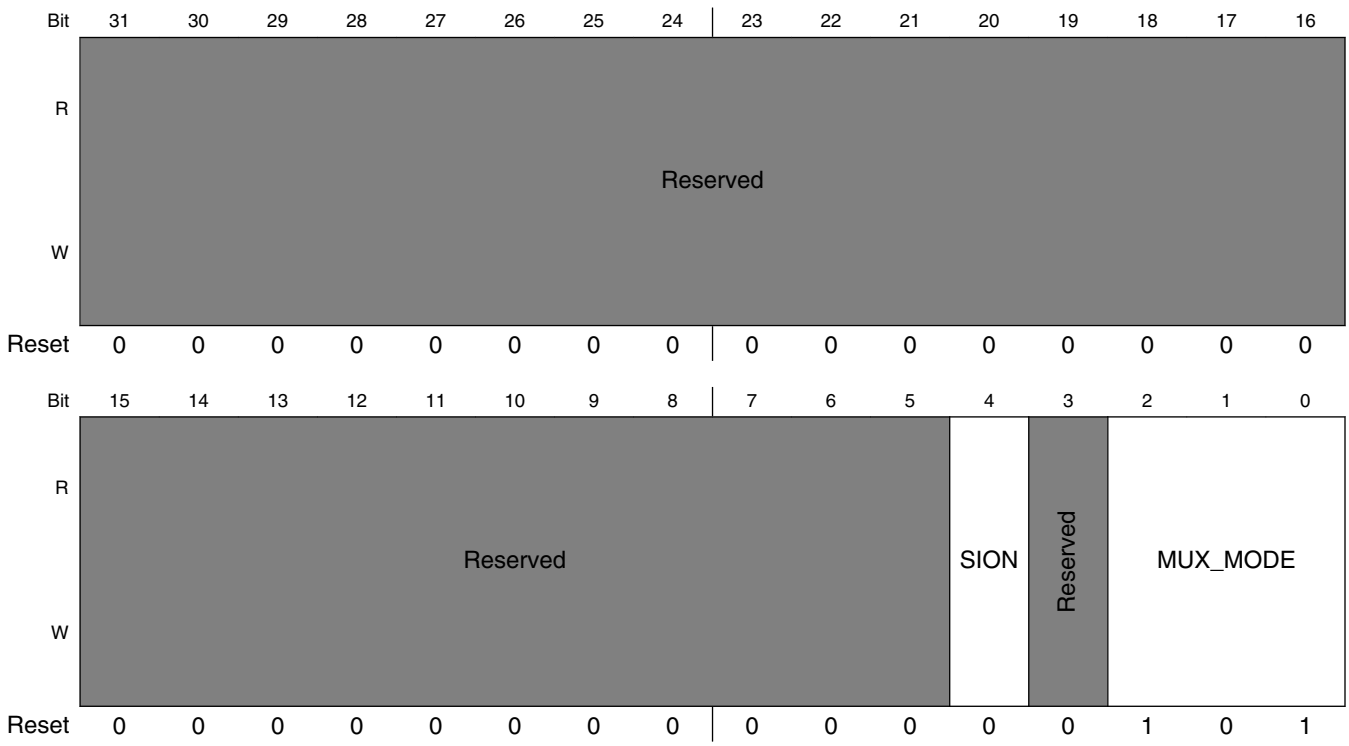
IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RX_CTL field descriptions (continued)

Field	Description
	010 **ALT2_ECSPI2_SS1** — Select mux mode: ALT2 mux port: SS1 of instance: ECSPI2 100 **ALT4_EPDC_SDCE6** — Select mux mode: ALT4 mux port: SDCE6 of instance: EPDC 101 **ALT5_GPIO7_IO4** — Select mux mode: ALT5 mux port: IO4 of instance: GPIO7 110 **ALT6_KPP_ROW1** — Select mux mode: ALT6 mux port: ROW1 of instance: KPP

8.2.7.140 SW_MUX_CTL_PAD_ENET1_RGMII_RXC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RXC)

SW_MUX_CTL Register

Address: 3033_0000h base + 240h offset = 3033_0240h



IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RXC field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

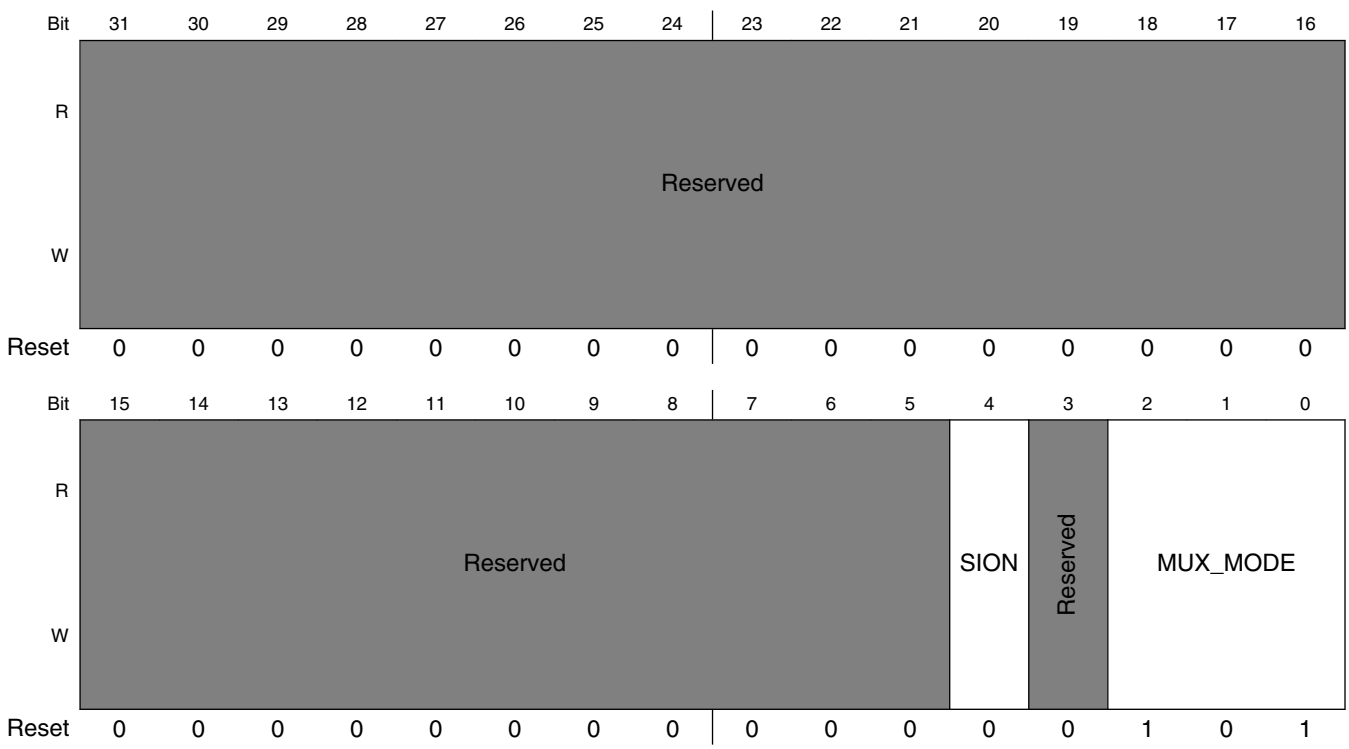
IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_RXC field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad ENET1_RGMII_RXC 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: ENET1_RGMII_RXC. 000 **ALT0_ENET1_RGMII_RXC** — Select mux mode: ALT0 mux port: RGMII_RXC of instance: ENET1 001 **ALT1_ENET1_RX_ER** — Select mux mode: ALT1 mux port: RX_ER of instance: ENET1 010 **ALT2_ECSPI2_SS2** — Select mux mode: ALT2 mux port: SS2 of instance: ECSPI2 100 **ALT4_EPDC_SDCE7** — Select mux mode: ALT4 mux port: SDCE7 of instance: EPDC 101 **ALT5_GPIO7_IO5** — Select mux mode: ALT5 mux port: IO5 of instance: GPIO7 110 **ALT6_KPP_COL1** — Select mux mode: ALT6 mux port: COL1 of instance: KPP

8.2.7.141 SW_MUX_CTL_PAD_ENET1_RGMII_TD0 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TD0)

SW_MUX_CTL Register

Address: 3033_0000h base + 244h offset = 3033_0244h



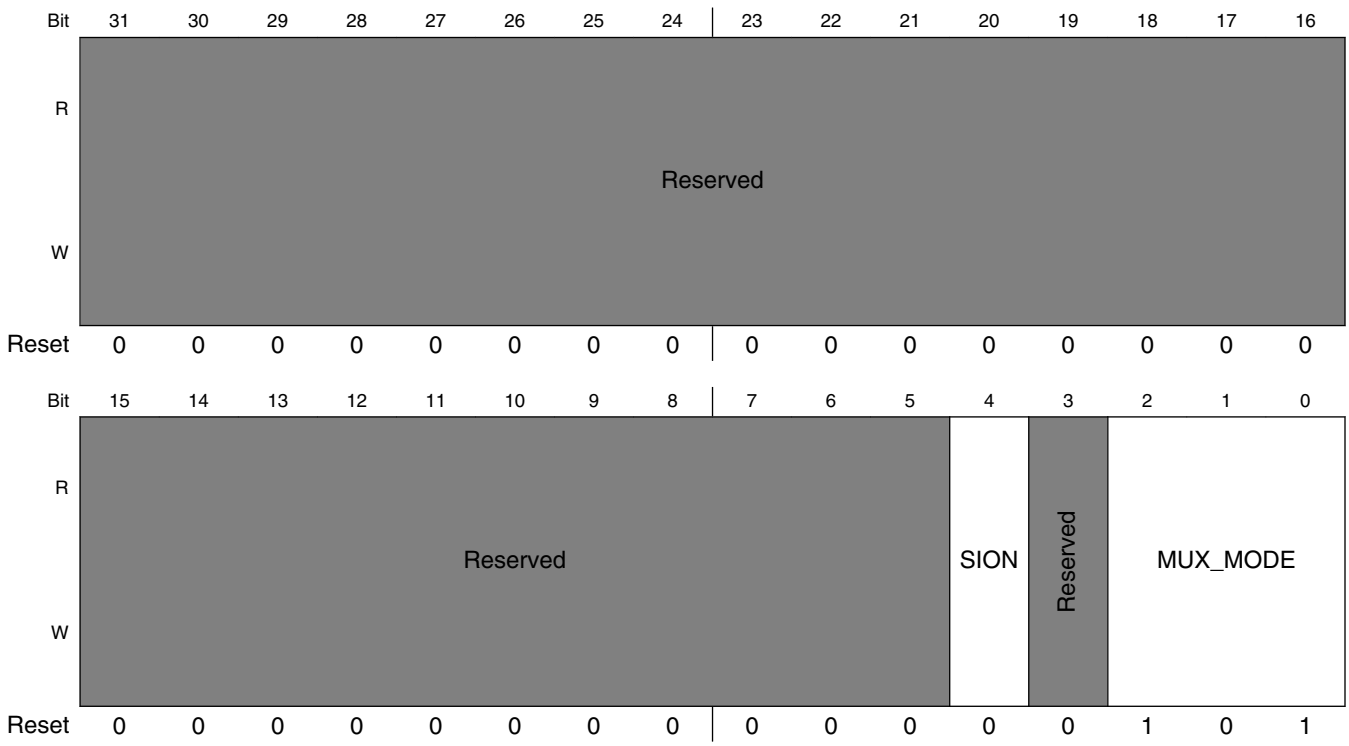
IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TD0 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_RGMII_TD0 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: ENET1_RGMII_TD0. 000 **ALT0_ENET1_RGMII_TD0** — Select mux mode: ALT0 mux port: RGMII_TD0 of instance: ENET1 001 **ALT1_PWM3_OUT** — Select mux mode: ALT1 mux port: OUT of instance: PWM3 010 **ALT2_ECSPI2_SS3** — Select mux mode: ALT2 mux port: SS3 of instance: ECSPI2 100 **ALT4_EPDC_SDCE8** — Select mux mode: ALT4 mux port: SDCE8 of instance: EPDC 101 **ALT5_GPIO7_IO6** — Select mux mode: ALT5 mux port: IO6 of instance: GPIO7 110 **ALT6_KPP_ROW0** — Select mux mode: ALT6 mux port: ROW0 of instance: KPP

8.2.7.142 SW_MUX_CTL_PAD_ENET1_RGMII_TD1 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TD1)

SW_MUX_CTL Register

Address: 3033_0000h base + 248h offset = 3033_0248h



IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TD1 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_RGMII_TD1 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: ENET1_RGMII_TD1. 000 **ALT0_ENET1_RGMII_TD1** — Select mux mode: ALT0 mux port: RGMII_TD1 of instance: ENET1 001 **ALT1_PWM4_OUT** — Select mux mode: ALT1 mux port: OUT of instance: PWM4

Table continues on the next page...

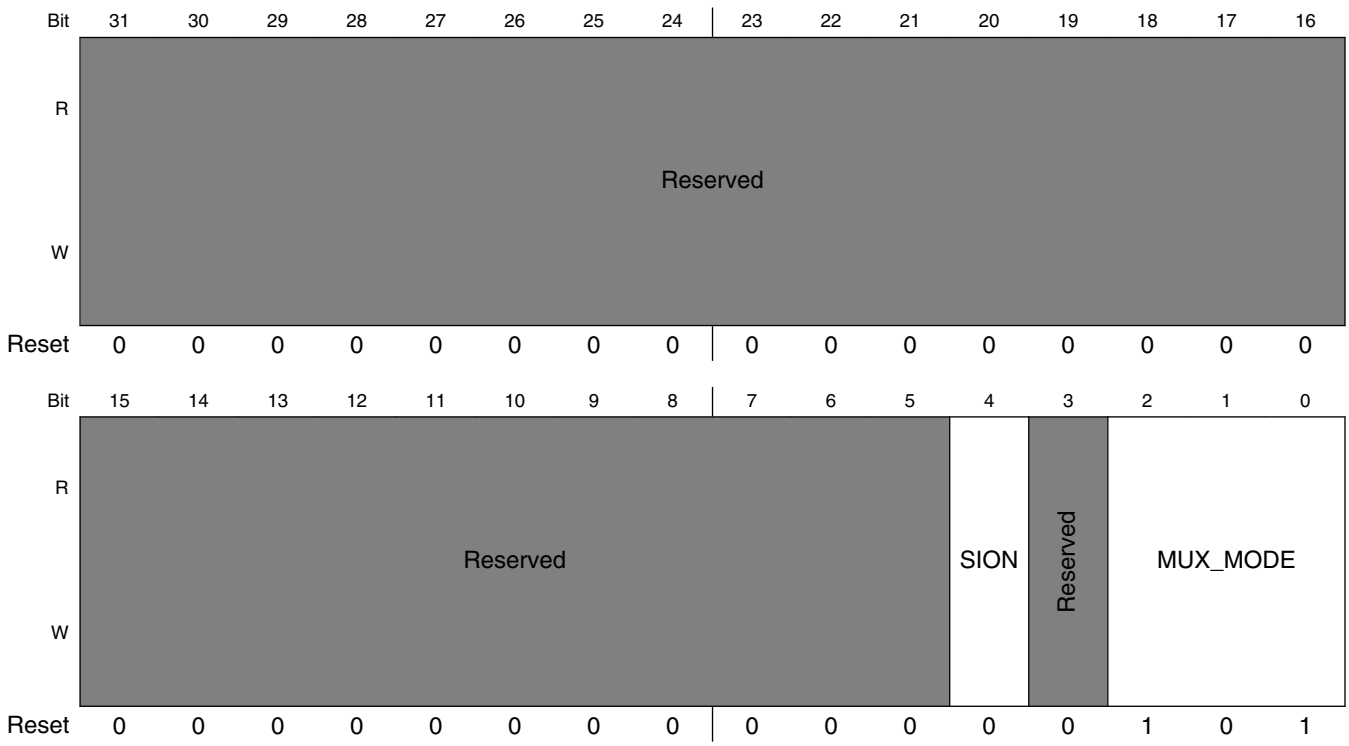
IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TD1 field descriptions (continued)

Field	Description
	010 **ALT2_ECSPI2_RDY** — Select mux mode: ALT2 mux port: RDY of instance: ECSPI2 100 **ALT4_EPDC_SDCE9** — Select mux mode: ALT4 mux port: SDCE9 of instance: EPDC 101 **ALT5_GPIO7_IO7** — Select mux mode: ALT5 mux port: IO7 of instance: GPIO7 110 **ALT6_KPP_COL0** — Select mux mode: ALT6 mux port: COL0 of instance: KPP

8.2.7.143 SW_MUX_CTL_PAD_ENET1_RGMII_TD2 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TD2)

SW_MUX_CTL Register

Address: 3033_0000h base + 24Ch offset = 3033_024Ch



IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TD2 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

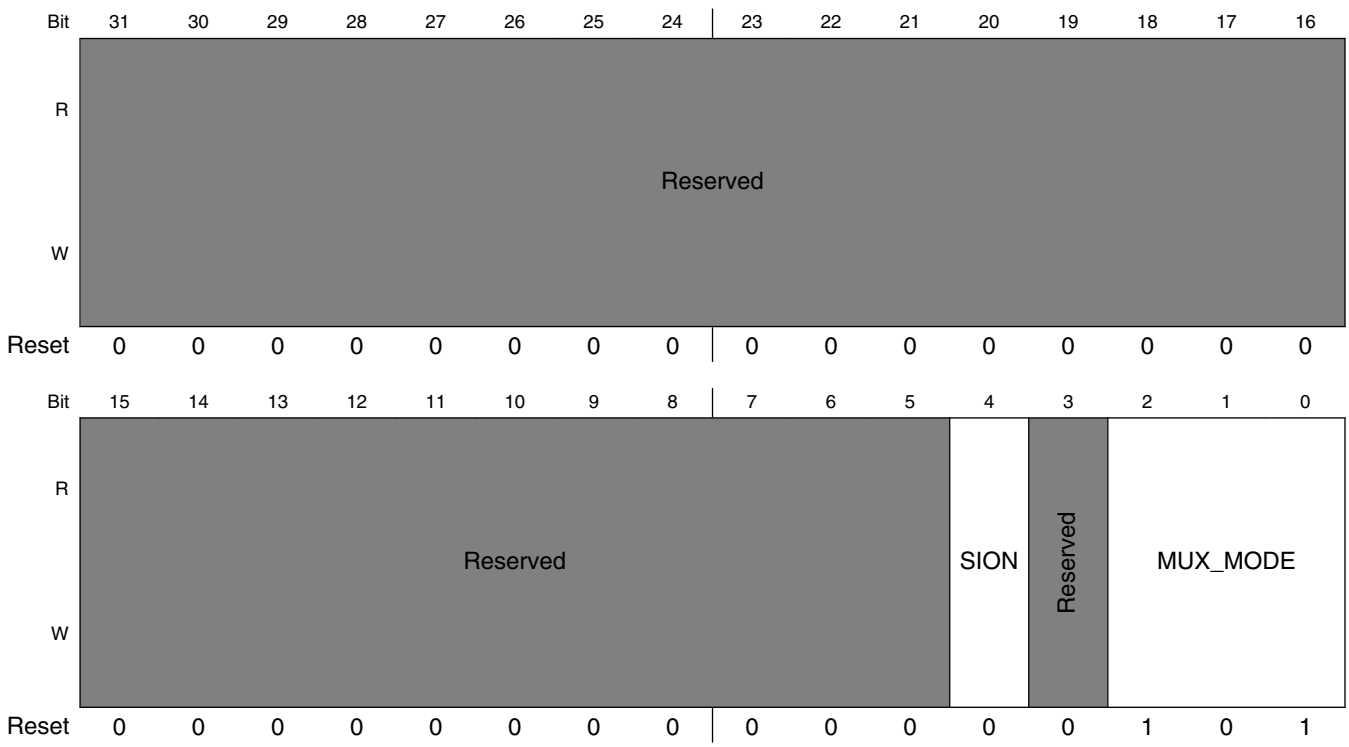
IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TD2 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad ENET1_RGMII_TD2 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: ENET1_RGMII_TD2. 000 **ALT0_ENET1_RGMII_TD2** — Select mux mode: ALT0 mux port: RGMII_TD2 of instance: ENET1 001 **ALT1_FLEXCAN2_RX** — Select mux mode: ALT1 mux port: RX of instance: FLEXCAN2 010 **ALT2_ECSPI2_MISO** — Select mux mode: ALT2 mux port: MISO of instance: ECSPI2 011 **ALT3_I2C4_SCL** — Select mux mode: ALT3 mux port: SCL of instance: I2C4 100 **ALT4_EPDC_SDOED** — Select mux mode: ALT4 mux port: SDOED of instance: EPDC 101 **ALT5_GPIO7_IO8** — Select mux mode: ALT5 mux port: IO8 of instance: GPIO7

8.2.7.144 SW_MUX_CTL_PAD_ENET1_RGMII_TD3 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TD3)

SW_MUX_CTL Register

Address: 3033_0000h base + 250h offset = 3033_0250h



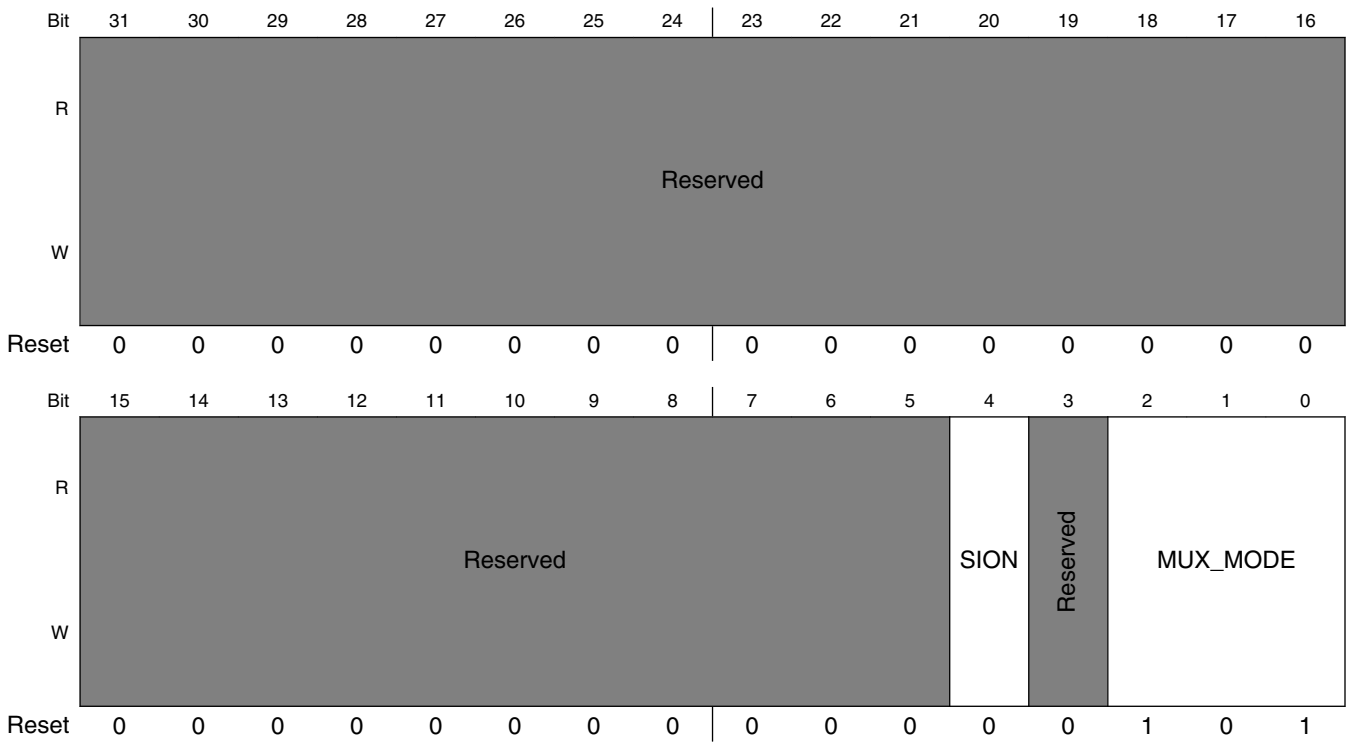
IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TD3 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_RGMII_TD3 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: ENET1_RGMII_TD3. 000 **ALT0_ENET1_RGMII_TD3** — Select mux mode: ALT0 mux port: RGMII_TD3 of instance: ENET1 001 **ALT1_FLEXCAN2_TX** — Select mux mode: ALT1 mux port: TX of instance: FLEXCAN2 010 **ALT2_ECSPI2_SS0** — Select mux mode: ALT2 mux port: SS0 of instance: ECSPI2 011 **ALT3_I2C4_SDA** — Select mux mode: ALT3 mux port: SDA of instance: I2C4 100 **ALT4_EPDC_SDOEZ** — Select mux mode: ALT4 mux port: SDOEZ of instance: EPDC 101 **ALT5_GPIO7_IO9** — Select mux mode: ALT5 mux port: IO9 of instance: GPIO7 111 **ALT7_CAAM_RNG_OSC_OBS** — Select mux mode: ALT7 mux port: RNG_OSC_OBS of instance: CAAM

8.2.7.145 SW_MUX_CTL_PAD_ENET1_RGMII_TX_CTL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TX_CTL)

SW_MUX_CTL Register

Address: 3033_0000h base + 254h offset = 3033_0254h



IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TX_CTL field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_RGMII_TX_CTL 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: ENET1_RGMII_TX_CTL. 000 **ALT0_ENET1_RGMII_TX_CTL** — Select mux mode: ALT0 mux port: RGMII_TX_CTL of instance: ENET1

Table continues on the next page...

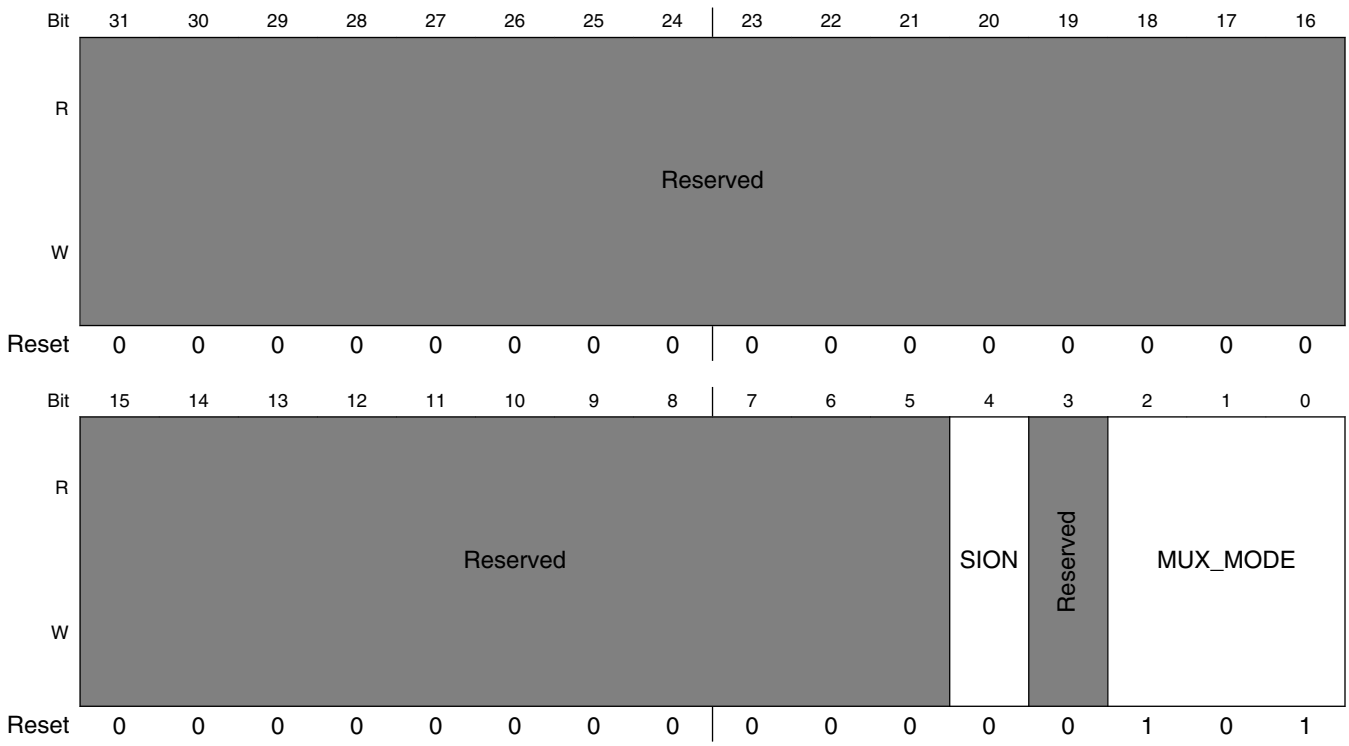
IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TX_CTL field descriptions (continued)

Field	Description
	010 **ALT2_SAI1_RX_SYNC** — Select mux mode: ALT2 mux port: RX_SYNC of instance: SAI1 011 **ALT3_GPT2_COMPARE1** — Select mux mode: ALT3 mux port: COMPARE1 of instance: GPT2 100 **ALT4_EPDC_PWR_CTRL2** — Select mux mode: ALT4 mux port: PWR_CTRL2 of instance: EPDC 101 **ALT5_GPIO7_IO10** — Select mux mode: ALT5 mux port: IO10 of instance: GPIO7

8.2.7.146 SW_MUX_CTL_PAD_ENET1_RGMII_TXC SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TXC)

SW_MUX_CTL Register

Address: 3033_0000h base + 258h offset = 3033_0258h



IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TXC field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_ENET1_RGMII_TXC field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad ENET1_RGMII_TXC 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: ENET1_RGMII_TXC. 000 **ALT0_ENET1_RGMII_TXC** — Select mux mode: ALT0 mux port: RGMII_TXC of instance: ENET1 001 **ALT1_ENET1_TX_ER** — Select mux mode: ALT1 mux port: TX_ER of instance: ENET1 010 **ALT2_SAI1_RX_BCLK** — Select mux mode: ALT2 mux port: RX_BCLK of instance: SAI1 011 **ALT3_GPT2_COMPARE2** — Select mux mode: ALT3 mux port: COMPARE2 of instance: GPT2 100 **ALT4_EPDC_PWR_CTRL3** — Select mux mode: ALT4 mux port: PWR_CTRL3 of instance: EPDC 101 **ALT5_GPIO7_IO11** — Select mux mode: ALT5 mux port: IO11 of instance: GPIO7

8.2.7.147 SW_MUX_CTL_PAD_ENET1_TX_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_CLK)

SW_MUX_CTL Register

Address: 3033_0000h base + 25Ch offset = 3033_025Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

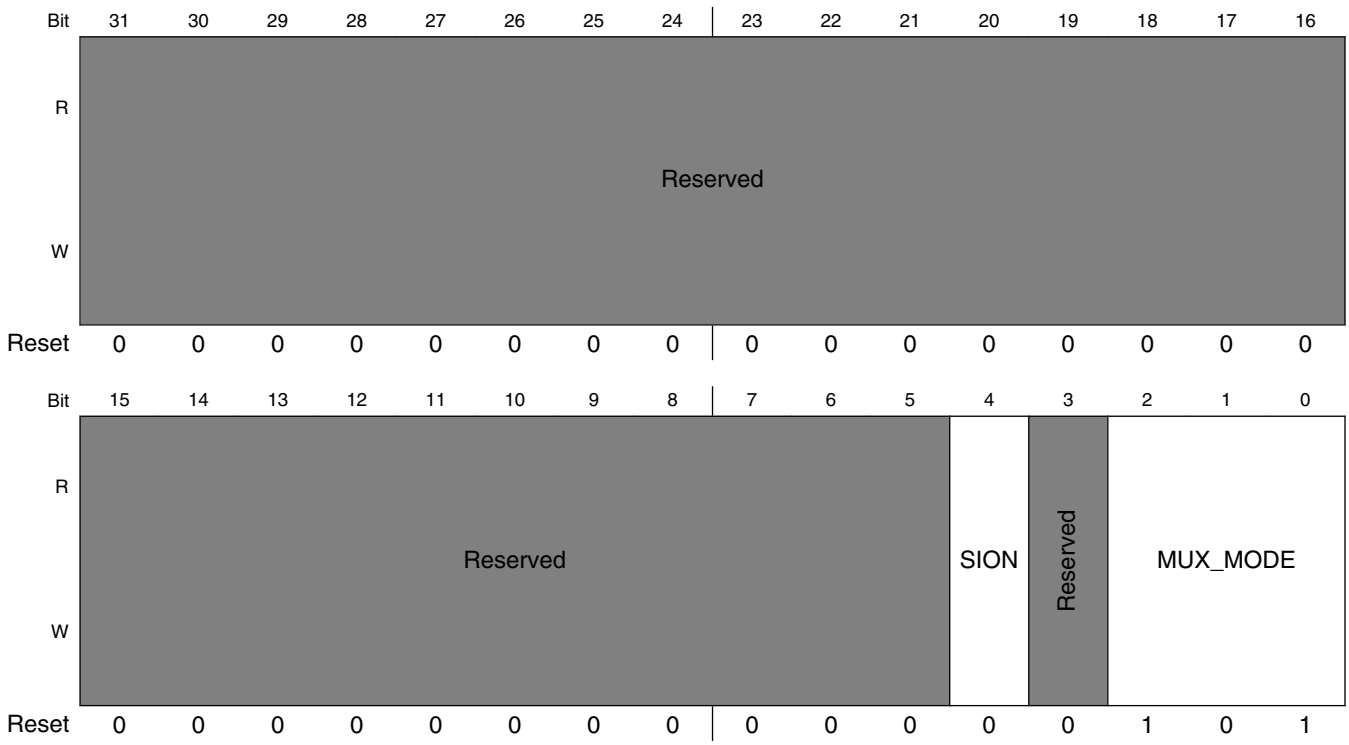
IOMUXC_SW_MUX_CTL_PAD_ENET1_TX_CLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_TX_CLK 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ENET1_TX_CLK. 000 **ALT0_ENET1_TX_CLK** — Select mux mode: ALT0 mux port: TX_CLK of instance: ENET1 001 **ALT1_CCM_ENET1_REF_CLK** — Select mux mode: ALT1 mux port: ENET1_REF_CLK of instance: ENET1 010 **ALT2_SAI1_RX_DATA0** — Select mux mode: ALT2 mux port: RX_DATA0 of instance: SAI1 011 **ALT3_GPT2_COMPARE3** — Select mux mode: ALT3 mux port: COMPARE3 of instance: GPT2 100 **ALT4_EPDC_PWR_IRQ** — Select mux mode: ALT4 mux port: PWR_IRQ of instance: EPDC 101 **ALT5_GPIO7_IO12** — Select mux mode: ALT5 mux port: IO12 of instance: GPIO7 110 **ALT6_CCM_EXT_CLK1** — Select mux mode: ALT6 mux port: EXT_CLK1 of instance: CCM 111 **ALT7_CSU_ALARM_AUT0** — Select mux mode: ALT7 mux port: CSU_ALARM_AUT0 of instance: CSU

8.2.7.148 SW_MUX_CTL_PAD_ENET1_RX_CLK SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_CLK)

SW_MUX_CTL Register

Address: 3033_0000h base + 260h offset = 3033_0260h



IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_CLK field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_RX_CLK 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ENET1_RX_CLK. 000 **ALT0_ENET1_RX_CLK** — Select mux mode: ALT0 mux port: RX_CLK of instance: ENET1 001 **ALT1_WDOG2_WDOG_B** — Select mux mode: ALT1 mux port: WDOG_B of instance: WDOG2 010 **ALT2_SAI1_TX_BCLK** — Select mux mode: ALT2 mux port: TX_BCLK of instance: SAI1

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_ENET1_RX_CLK field descriptions (continued)

Field	Description
	011 **ALT3_GPT2_CLK** — Select mux mode: ALT3 mux port: CLK of instance: GPT2 100 **ALT4_EPDC_PWR_WAKE** — Select mux mode: ALT4 mux port: PWR_WAKE of instance: EPDC 101 **ALT5_GPIO7_IO13** — Select mux mode: ALT5 mux port: IO13 of instance: GPIO7 110 **ALT6_CCM_EXT_CLK2** — Select mux mode: ALT6 mux port: EXT_CLK2 of instance: CCM 111 **ALT7_CSU_ALARM_AUT1** — Select mux mode: ALT7 mux port: CSU_ALARM_AUT1 of instance: CSU

8.2.7.149 SW_MUX_CTL_PAD_ENET1_CRS SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_CRS)

SW_MUX_CTL Register

Address: 3033_0000h base + 264h offset = 3033_0264h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_ENET1_CRS field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

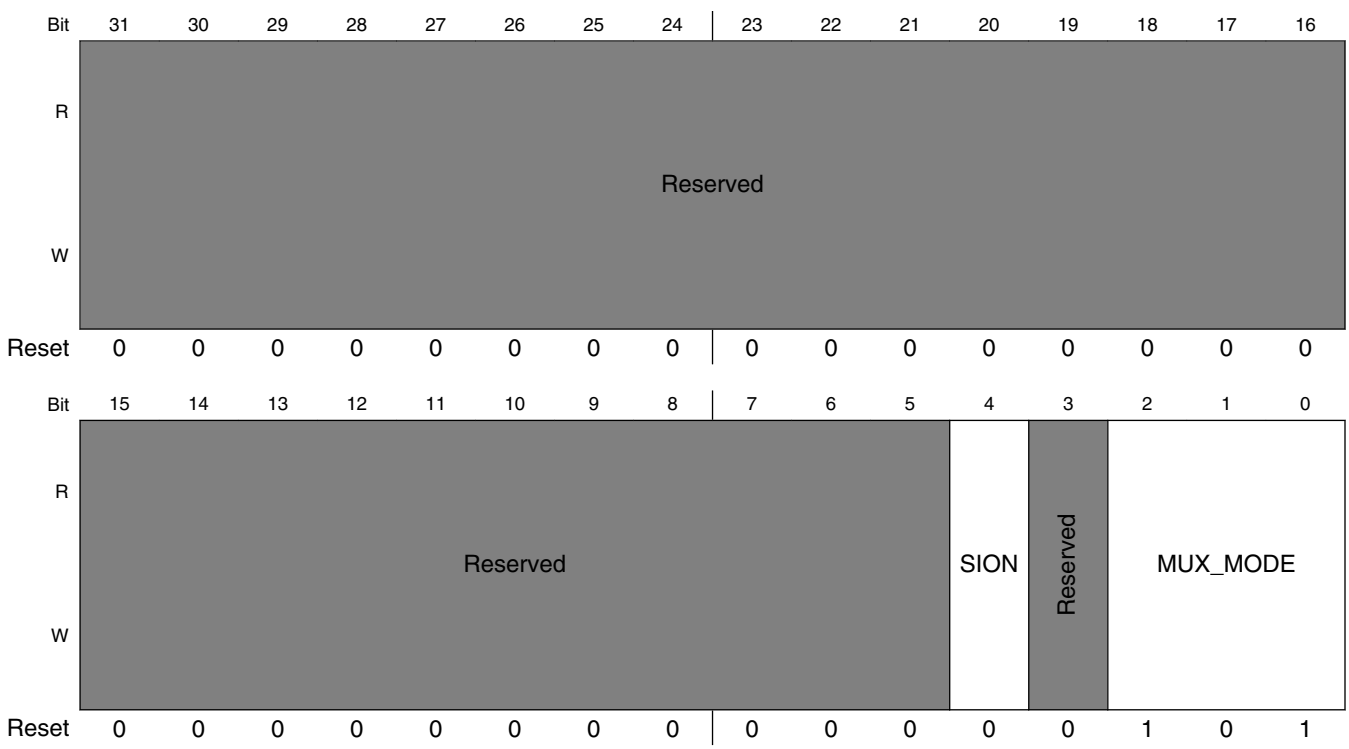
IOMUXC_SW_MUX_CTL_PAD_ENET1_CRS field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad ENET1_CRS 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ENET1_CRS. 000 **ALT0_ENET1_CRS** — Select mux mode: ALT0 mux port: CRS of instance: ENET1 001 **ALT1_WDOG2_WDOG_RST_B_DEB** — Select mux mode: ALT1 mux port: WDOG_RST_B_DEB of instance: WDOG2 010 **ALT2_SAI1_TX_SYNC** — Select mux mode: ALT2 mux port: TX_SYNC of instance: SAI1 011 **ALT3_GPT2_CAPTURE1** — Select mux mode: ALT3 mux port: CAPTURE1 of instance: GPT2 100 **ALT4_EPDC_PWR_CTRL0** — Select mux mode: ALT4 mux port: PWR_CTRL0 of instance: EPDC 101 **ALT5_GPIO7_IO14** — Select mux mode: ALT5 mux port: IO14 of instance: GPIO7 110 **ALT6_CCM_EXT_CLK3** — Select mux mode: ALT6 mux port: EXT_CLK3 of instance: CCM 111 **ALT7_CSU_ALARM_AUT2** — Select mux mode: ALT7 mux port: CSU_ALARM_AUT2 of instance: CSU

8.2.7.150 SW_MUX_CTL_PAD_ENET1_COL SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_ENET1_COL)

SW_MUX_CTL Register

Address: 3033_0000h base + 268h offset = 3033_0268h



IOMUXC_SW_MUX_CTL_PAD_ENET1_COL field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad ENET1_COL 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ENET1_COL. 000 **ALT0_ENET1_COL** — Select mux mode: ALT0 mux port: COL of instance: ENET1 001 **ALT1_WDOG1_WDOG_ANY** — Select mux mode: ALT1 mux port: WDOG_ANY of instance: WDOG1 010 **ALT2_SAI1_TX_DATA0** — Select mux mode: ALT2 mux port: TX_DATA0 of instance: SAI1 011 **ALT3_GPT2_CAPTURE2** — Select mux mode: ALT3 mux port: CAPTURE2 of instance: GPT2 100 **ALT4_EPDC_PWR_CTRL1** — Select mux mode: ALT4 mux port: PWR_CTRL1 of instance: EPDC 101 **ALT5_GPIO7_IO15** — Select mux mode: ALT5 mux port: IO15 of instance: GPIO7 110 **ALT6_CCM_EXT_CLK4** — Select mux mode: ALT6 mux port: EXT_CLK4 of instance: CCM 111 **ALT7_CSU_INT_DEB** — Select mux mode: ALT7 mux port: CSU_INT_DEB of instance: CSU

8.2.7.151 SW_PAD_CTL_PAD_GPIO1_IO08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO08)

SW_PAD_CTL Register

Address: 3033_0000h base + 26Ch offset = 3033_026Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO08 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO08 field descriptions (continued)

Field	Description
	Select one out of next values for pad: GPIO1_IO08 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO08 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO08 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO08 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO08 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.152 SW_PAD_CTL_PAD_GPIO1_IO09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO09)

SW_PAD_CTL Register

Address: 3033_0000h base + 270h offset = 3033_0270h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO09 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO09 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO09 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO09 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO09 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO09 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.153 SW_PAD_CTL_PAD_GPIO1_IO10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO10)

SW_PAD_CTL Register

Address: 3033_0000h base + 274h offset = 3033_0274h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO10 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO10 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO10 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO10 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO10 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO10 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO10 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.154 SW_PAD_CTL_PAD_GPIO1_IO11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO11)

SW_PAD_CTL Register

Address: 3033_0000h base + 278h offset = 3033_0278h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO11 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO11 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO11 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO11 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO11

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO11 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO11 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.155 SW_PAD_CTL_PAD_GPIO1_IO12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO12)

SW_PAD_CTL Register

Address: 3033_0000h base + 27Ch offset = 3033_027Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO12 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO12 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO12 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO12 field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO12 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO12 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO12 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.156 SW_PAD_CTL_PAD_GPIO1_IO13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO13)

SW_PAD_CTL Register

Address: 3033_0000h base + 280h offset = 3033_0280h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO13 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO13 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO13 field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO13 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO13 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO13 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO13 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.157 SW_PAD_CTL_PAD_GPIO1_IO14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO14)

SW_PAD_CTL Register

Address: 3033_0000h base + 284h offset = 3033_0284h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO14 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO14 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO14 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO14 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO14 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO14 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.158 SW_PAD_CTL_PAD_GPIO1_IO15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO15)

SW_PAD_CTL Register

Address: 3033_0000h base + 288h offset = 3033_0288h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO15 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: GPIO1_IO15 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: GPIO1_IO15 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO1_IO15 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: GPIO1_IO15 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: GPIO1_IO15 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO1_IO15 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.159 SW_PAD_CTL_PAD_JTAG_MOD SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_MOD)

SW_PAD_CTL Register

Address: 3033_0000h base + 28Ch offset = 3033_028Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								PS			PE	HYS	SRE	DSE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_JTAG_MOD field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: JTAG_MOD 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: JTAG_MOD 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_MOD 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled

Table continues on the next page...

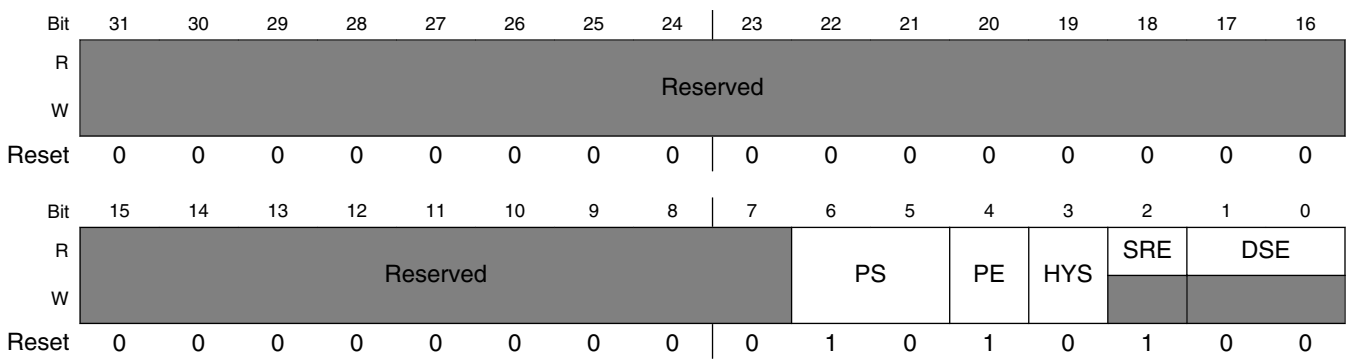
IOMUXC_SW_PAD_CTL_PAD_JTAG_MOD field descriptions (continued)

Field	Description
2 SRE	Slew Rate Field Read Only Field 1 **SRE** — Slow Slew Rate
DSE	Drive Strength Field Read Only Field 00 **DSE** — X1

8.2.7.160 SW_PAD_CTL_PAD_JTAG_TCK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TCK)

SW_PAD_CTL Register

Address: 3033_0000h base + 290h offset = 3033_0290h



IOMUXC_SW_PAD_CTL_PAD_JTAG_TCK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: JTAG_TCK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: JTAG_TCK

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_JTAG_TCK field descriptions (continued)

Field	Description
	0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TCK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Read Only Field 1 **SRE** — Slow Slew Rate
DSE	Drive Strength Field Read Only Field 00 **DSE** — X1

8.2.7.161 SW_PAD_CTL_PAD_JTAG_TDI SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TDI)

SW_PAD_CTL Register

Address: 3033_0000h base + 294h offset = 3033_0294h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_JTAG_TDI field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: JTAG_TDI 00 **PS_0_100K_PD** — 100K PD

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_JTAG_TDI field descriptions (continued)

Field	Description
	01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: JTAG_TDI 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TDI 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Read Only Field 1 **SRE** — Slow Slew Rate
DSE	Drive Strength Field Read Only Field 00 **DSE** — X1

8.2.7.162 SW_PAD_CTL_PAD_JTAG_TDO SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TDO)

SW_PAD_CTL Register

Address: 3033_0000h base + 298h offset = 3033_0298h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_JTAG_TDO field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: JTAG_TDO 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: JTAG_TDO 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TDO 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Read Only Field 0 **SRE** — Fast Slew Rate
DSE	Drive Strength Field Read Only Field 00 **DSE** — X1

8.2.7.163 SW_PAD_CTL_PAD_JTAG_TMS SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TMS)

SW_PAD_CTL Register

Address: 3033_0000h base + 29Ch offset = 3033_029Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_JTAG_TMS field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: JTAG_TMS 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: JTAG_TMS 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TMS 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Read Only Field 1 **SRE** — Slow Slew Rate
DSE	Drive Strength Field Read Only Field

Table continues on the next page...

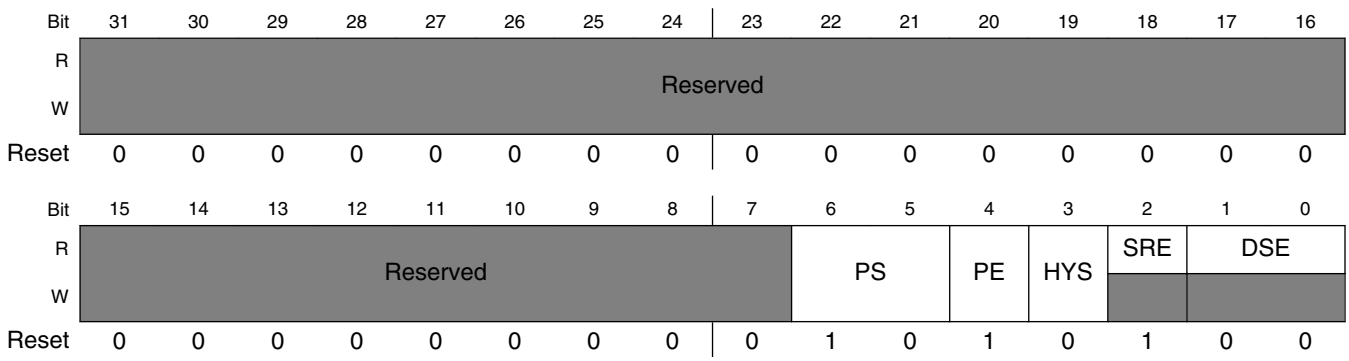
IOMUXC_SW_PAD_CTL_PAD_JTAG_TMS field descriptions (continued)

Field	Description
	00 **DSE** — X1

8.2.7.164 SW_PAD_CTL_PAD_JTAG_TRST_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_JTAG_TRST_B)

SW_PAD_CTL Register

Address: 3033_0000h base + 2A0h offset = 3033_02A0h



IOMUXC_SW_PAD_CTL_PAD_JTAG_TRST_B field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: JTAG_TRST_B 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: JTAG_TRST_B 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TRST_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_JTAG_TRST_B field descriptions (continued)

Field	Description
	Read Only Field 1 **SRE** — Slow Slew Rate
DSE	Drive Strength Field Read Only Field 00 **DSE** — X1

8.2.7.165 SW_PAD_CTL_PAD_EPDC_DATA00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA00)

SW_PAD_CTL Register

Address: 3033_0000h base + 2A4h offset = 3033_02A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA00 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA00 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA00 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA00

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA00 field descriptions (continued)

Field	Description
	0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA00 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA00 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.166 SW_PAD_CTL_PAD_EPDC_DATA01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA01)

SW_PAD_CTL Register

Address: 3033_0000h base + 2A8h offset = 3033_02A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA01 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA01 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA01 field descriptions (continued)

Field	Description
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA01 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA01 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA01 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.167 SW_PAD_CTL_PAD_EPDC_DATA02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA02)

SW_PAD_CTL Register

Address: 3033_0000h base + 2ACh offset = 3033_02ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA02 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA02 field descriptions (continued)

Field	Description
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA02 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA02 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA02 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA02 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.168 SW_PAD_CTL_PAD_EPDC_DATA03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA03)

SW_PAD_CTL Register

Address: 3033_0000h base + 2B0h offset = 3033_02B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA03 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA03 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA03 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA03 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA03 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.169 SW_PAD_CTL_PAD_EPDC_DATA04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA04)

SW_PAD_CTL Register

Address: 3033_0000h base + 2B4h offset = 3033_02B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA04 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA04 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA04 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA04 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA04 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA04 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.170 SW_PAD_CTL_PAD_EPDC_DATA05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA05)

SW_PAD_CTL Register

Address: 3033_0000h base + 2B8h offset = 3033_02B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA05 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA05 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA05 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA05

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA05 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA05 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.171 SW_PAD_CTL_PAD_EPDC_DATA06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA06)

SW_PAD_CTL Register

Address: 3033_0000h base + 2BCh offset = 3033_02BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA06 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA06 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA06 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA06 field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA06 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA06 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA06 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.172 SW_PAD_CTL_PAD_EPDC_DATA07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA07)

SW_PAD_CTL Register

Address: 3033_0000h base + 2C0h offset = 3033_02C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA07 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA07 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA07 field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA07 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA07 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA07 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA07 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.173 SW_PAD_CTL_PAD_EPDC_DATA08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA08)

SW_PAD_CTL Register

Address: 3033_0000h base + 2C4h offset = 3033_02C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA08 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA08 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA08 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA08 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA08 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA08 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.174 SW_PAD_CTL_PAD_EPDC_DATA09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA09)

SW_PAD_CTL Register

Address: 3033_0000h base + 2C8h offset = 3033_02C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA09 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA09 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA09 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA09 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA09 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA09 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA09 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.175 SW_PAD_CTL_PAD_EPDC_DATA10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA10)

SW_PAD_CTL Register

Address: 3033_0000h base + 2CCh offset = 3033_02CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA10 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA10 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA10 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA10 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA10

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA10 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA10 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.176 SW_PAD_CTL_PAD_EPDC_DATA11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA11)

SW_PAD_CTL Register

Address: 3033_0000h base + 2D0h offset = 3033_02D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA11 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA11 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA11 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA11 field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA11 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA11 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA11 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.177 SW_PAD_CTL_PAD_EPDC_DATA12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA12)

SW_PAD_CTL Register

Address: 3033_0000h base + 2D4h offset = 3033_02D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA12 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA12 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA12 field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA12 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA12 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA12 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA12 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.178 SW_PAD_CTL_PAD_EPDC_DATA13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA13)

SW_PAD_CTL Register

Address: 3033_0000h base + 2D8h offset = 3033_02D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA13 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA13 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA13 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA13 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA13 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA13 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.179 SW_PAD_CTL_PAD_EPDC_DATA14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA14)

SW_PAD_CTL Register

Address: 3033_0000h base + 2DCh offset = 3033_02DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA14 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA14 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA14 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA14 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA14 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA14 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA14 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.180 SW_PAD_CTL_PAD_EPDC_DATA15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA15)

SW_PAD_CTL Register

Address: 3033_0000h base + 2E0h offset = 3033_02E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA15 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_DATA15 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_DATA15 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_DATA15 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_DATA15

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_DATA15 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_DATA15 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.181 SW_PAD_CTL_PAD_EPDC_SDCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCLK)

SW_PAD_CTL Register

Address: 3033_0000h base + 2E4h offset = 3033_02E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCLK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_SDCLK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_SDCLK 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCLK field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_SDCLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_SDCLK 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDCLK 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.182 SW_PAD_CTL_PAD_EPDC_SDLE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDLE)

SW_PAD_CTL Register

Address: 3033_0000h base + 2E8h offset = 3033_02E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_SDLE field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_SDLE 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_SDLE field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_SDLE 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_SDLE 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_SDLE 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDLE 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.183 SW_PAD_CTL_PAD_EPDC_SDOE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDOE)

SW_PAD_CTL Register

Address: 3033_0000h base + 2ECh offset = 3033_02ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_SDOE field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_SDOE 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_SDOE 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_SDOE 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_SDOE 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDOE 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.184 SW_PAD_CTL_PAD_EPDC_SDSHR SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDSHR)

SW_PAD_CTL Register

Address: 3033_0000h base + 2F0h offset = 3033_02F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_SDSHR field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_SDSHR 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_SDSHR 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_SDSHR 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_SDSHR 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDSHR 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_SDSHR field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.185 SW_PAD_CTL_PAD_EPDC_SDCE0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE0)

SW_PAD_CTL Register

Address: 3033_0000h base + 2F4h offset = 3033_02F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE0 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_SDCE0 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_SDCE0 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_SDCE0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_SDCE0

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE0 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDCE0 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.186 SW_PAD_CTL_PAD_EPDC_SDCE1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE1)

SW_PAD_CTL Register

Address: 3033_0000h base + 2F8h offset = 3033_02F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE1 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_SDCE1 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_SDCE1 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE1 field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_SDCE1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_SDCE1 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDCE1 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.187 SW_PAD_CTL_PAD_EPDC_SDCE2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE2)

SW_PAD_CTL Register

Address: 3033_0000h base + 2FCh offset = 3033_02FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE2 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_SDCE2 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE2 field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_SDCE2 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_SDCE2 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_SDCE2 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDCE2 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.188 SW_PAD_CTL_PAD_EPDC_SDCE3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE3)

SW_PAD_CTL Register

Address: 3033_0000h base + 300h offset = 3033_0300h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_SDCE3 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_SDCE3 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_SDCE3 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_SDCE3 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_SDCE3 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDCE3 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.189 SW_PAD_CTL_PAD_EPDC_GDCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_GDCLK)

SW_PAD_CTL Register

Address: 3033_0000h base + 304h offset = 3033_0304h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_GDCLK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_GDCLK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_GDCLK 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_GDCLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_GDCLK 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_GDCLK 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_GDCLK field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.190 SW_PAD_CTL_PAD_EPDC_GDOE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_GDOE)

SW_PAD_CTL Register

Address: 3033_0000h base + 308h offset = 3033_0308h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_GDOE field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_GDOE 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_GDOE 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_GDOE 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_GDOE

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_GDOE field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_GDOE 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.191 SW_PAD_CTL_PAD_EPDC_GDRL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_GDRL)

SW_PAD_CTL Register

Address: 3033_0000h base + 30Ch offset = 3033_030Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_GDRL field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_GDRL 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_GDRL 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_GDRL field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_GDRL 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_GDRL 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_GDRL 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.192 SW_PAD_CTL_PAD_EPDC_GDSP SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_GDSP)

SW_PAD_CTL Register

Address: 3033_0000h base + 310h offset = 3033_0310h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_GDSP field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_GDSP 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_GDSP field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_GDSP 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_GDSP 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_GDSP 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_GDSP 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.193 SW_PAD_CTL_PAD_EPDC_BDR0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_BDR0)

SW_PAD_CTL Register

Address: 3033_0000h base + 314h offset = 3033_0314h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_BDR0 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_BDR0 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_BDR0 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_BDR0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_BDR0 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_BDR0 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.194 SW_PAD_CTL_PAD_EPDC_BDR1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_BDR1)

SW_PAD_CTL Register

Address: 3033_0000h base + 318h offset = 3033_0318h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_BDR1 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_BDR1 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_BDR1 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_BDR1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_BDR1 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_BDR1 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_BDR1 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.195 SW_PAD_CTL_PAD_EPDC_PWR_COM SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_PWR_COM)

SW_PAD_CTL Register

Address: 3033_0000h base + 31Ch offset = 3033_031Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_PWR_COM field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_PWR_COM 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_PWR_COM 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_PWR_COM 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_PWR_COM

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_PWR_COM field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_PWR_COM 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.196 SW_PAD_CTL_PAD_EPDC_PWR_STAT SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_EPDC_PWR_STAT)

SW_PAD_CTL Register

Address: 3033_0000h base + 320h offset = 3033_0320h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_EPDC_PWR_STAT field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: EPDC_PWR_STAT 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: EPDC_PWR_STAT 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_EPDC_PWR_STAT field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: EPDC_PWR_STAT 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: EPDC_PWR_STAT 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: EPDC_PWR_STAT 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.197 SW_PAD_CTL_PAD_LCD_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_CLK)

SW_PAD_CTL Register

Address: 3033_0000h base + 324h offset = 3033_0324h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_CLK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_CLK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_CLK field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_CLK 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_CLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_CLK 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_CLK 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.198 SW_PAD_CTL_PAD_LCD_ENABLE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_ENABLE)

SW_PAD_CTL Register

Address: 3033_0000h base + 328h offset = 3033_0328h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_ENABLE field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_ENABLE 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_ENABLE 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_ENABLE 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_ENABLE 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_ENABLE 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.199 SW_PAD_CTL_PAD_LCD_HSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_HSYNC)

SW_PAD_CTL Register

Address: 3033_0000h base + 32Ch offset = 3033_032Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_HSYNC field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_HSYNC 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_HSYNC 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_HSYNC 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_HSYNC 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_HSYNC 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_HSYNC field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.200 SW_PAD_CTL_PAD_LCD_VSYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_VSYNC)

SW_PAD_CTL Register

Address: 3033_0000h base + 330h offset = 3033_0330h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_VSYNC field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_VSYNC 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_VSYNC 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_VSYNC 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_VSYNC

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_VSYNC field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_VSYNC 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.201 SW_PAD_CTL_PAD_LCD_RESET SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_RESET)

SW_PAD_CTL Register

Address: 3033_0000h base + 334h offset = 3033_0334h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_RESET field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_RESET 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_RESET 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_RESET field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_RESET 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_RESET 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_RESET 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.202 SW_PAD_CTL_PAD_LCD_DATA00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA00)

SW_PAD_CTL Register

Address: 3033_0000h base + 338h offset = 3033_0338h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA00 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA00 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA00 field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA00 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA00 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA00 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA00 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.203 SW_PAD_CTL_PAD_LCD_DATA01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA01)

SW_PAD_CTL Register

Address: 3033_0000h base + 33Ch offset = 3033_033Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA01 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA01 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA01 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA01 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA01 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.204 SW_PAD_CTL_PAD_LCD_DATA02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA02)

SW_PAD_CTL Register

Address: 3033_0000h base + 340h offset = 3033_0340h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA02 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA02 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA02 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA02 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA02 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA02 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.205 SW_PAD_CTL_PAD_LCD_DATA03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA03)

SW_PAD_CTL Register

Address: 3033_0000h base + 344h offset = 3033_0344h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA03 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA03 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA03 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA03

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA03 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA03 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.206 SW_PAD_CTL_PAD_LCD_DATA04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA04)

SW_PAD_CTL Register

Address: 3033_0000h base + 348h offset = 3033_0348h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA04 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA04 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA04 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA04 field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA04 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA04 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.207 SW_PAD_CTL_PAD_LCD_DATA05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA05)

SW_PAD_CTL Register

Address: 3033_0000h base + 34Ch offset = 3033_034Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA05 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA05 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA05 field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA05 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA05 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA05 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.208 SW_PAD_CTL_PAD_LCD_DATA06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA06)

SW_PAD_CTL Register

Address: 3033_0000h base + 350h offset = 3033_0350h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA06 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA06 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA06 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA06 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA06 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA06 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.209 SW_PAD_CTL_PAD_LCD_DATA07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA07)

SW_PAD_CTL Register

Address: 3033_0000h base + 354h offset = 3033_0354h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA07 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA07 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA07 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA07 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA07 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA07 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA07 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.210 SW_PAD_CTL_PAD_LCD_DATA08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA08)

SW_PAD_CTL Register

Address: 3033_0000h base + 358h offset = 3033_0358h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA08 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA08 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA08 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA08 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA08

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA08 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA08 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.211 SW_PAD_CTL_PAD_LCD_DATA09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA09)

SW_PAD_CTL Register

Address: 3033_0000h base + 35Ch offset = 3033_035Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA09 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA09 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA09 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA09 field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA09 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA09 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA09 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.212 SW_PAD_CTL_PAD_LCD_DATA10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA10)

SW_PAD_CTL Register

Address: 3033_0000h base + 360h offset = 3033_0360h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA10 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA10 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA10 field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA10 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA10 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA10 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA10 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.213 SW_PAD_CTL_PAD_LCD_DATA11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA11)

SW_PAD_CTL Register

Address: 3033_0000h base + 364h offset = 3033_0364h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA11 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA11 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA11 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA11 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA11 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA11 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.214 SW_PAD_CTL_PAD_LCD_DATA12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA12)

SW_PAD_CTL Register

Address: 3033_0000h base + 368h offset = 3033_0368h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA12 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA12 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA12 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA12 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA12 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA12 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA12 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.215 SW_PAD_CTL_PAD_LCD_DATA13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA13)

SW_PAD_CTL Register

Address: 3033_0000h base + 36Ch offset = 3033_036Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA13 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA13 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA13 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA13 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA13

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA13 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA13 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.216 SW_PAD_CTL_PAD_LCD_DATA14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA14)

SW_PAD_CTL Register

Address: 3033_0000h base + 370h offset = 3033_0370h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA14 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA14 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA14 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA14 field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA14 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA14 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA14 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.217 SW_PAD_CTL_PAD_LCD_DATA15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA15)

SW_PAD_CTL Register

Address: 3033_0000h base + 374h offset = 3033_0374h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA15 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA15 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA15 field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA15 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA15 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA15 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA15 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.218 SW_PAD_CTL_PAD_LCD_DATA16 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA16)

SW_PAD_CTL Register

Address: 3033_0000h base + 378h offset = 3033_0378h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA16 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA16 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA16 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA16 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA16 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA16 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.219 SW_PAD_CTL_PAD_LCD_DATA17 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA17)

SW_PAD_CTL Register

Address: 3033_0000h base + 37Ch offset = 3033_037Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA17 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA17 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA17 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA17 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA17 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA17 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA17 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.220 SW_PAD_CTL_PAD_LCD_DATA18 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA18)

SW_PAD_CTL Register

Address: 3033_0000h base + 380h offset = 3033_0380h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA18 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA18 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA18 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA18 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA18

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA18 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA18 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.221 SW_PAD_CTL_PAD_LCD_DATA19 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA19)

SW_PAD_CTL Register

Address: 3033_0000h base + 384h offset = 3033_0384h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA19 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA19 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA19 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA19 field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA19 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA19 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA19 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.222 SW_PAD_CTL_PAD_LCD_DATA20 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA20)

SW_PAD_CTL Register

Address: 3033_0000h base + 388h offset = 3033_0388h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA20 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA20 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA20 field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA20 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA20 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA20 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA20 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.223 SW_PAD_CTL_PAD_LCD_DATA21 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA21)

SW_PAD_CTL Register

Address: 3033_0000h base + 38Ch offset = 3033_038Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA21 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA21 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA21 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA21 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA21 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA21 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.224 SW_PAD_CTL_PAD_LCD_DATA22 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA22)

SW_PAD_CTL Register

Address: 3033_0000h base + 390h offset = 3033_0390h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA22 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA22 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA22 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA22 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA22 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA22 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA22 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.225 SW_PAD_CTL_PAD_LCD_DATA23 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_LCD_DATA23)

SW_PAD_CTL Register

Address: 3033_0000h base + 394h offset = 3033_0394h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA23 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: LCD_DATA23 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: LCD_DATA23 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: LCD_DATA23 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: LCD_DATA23

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_LCD_DATA23 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: LCD_DATA23 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.226 SW_PAD_CTL_PAD_UART1_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_RX_DATA)

SW_PAD_CTL Register

Address: 3033_0000h base + 398h offset = 3033_0398h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_UART1_RX_DATA field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: UART1_RX_DATA 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: UART1_RX_DATA 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART1_RX_DATA field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: UART1_RX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: UART1_RX_DATA 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: UART1_RX_DATA 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.227 SW_PAD_CTL_PAD_UART1_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART1_TX_DATA)

SW_PAD_CTL Register

Address: 3033_0000h base + 39Ch offset = 3033_039Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_UART1_TX_DATA field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: UART1_TX_DATA 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART1_TX_DATA field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: UART1_TX_DATA 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: UART1_TX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: UART1_TX_DATA 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: UART1_TX_DATA 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.228 SW_PAD_CTL_PAD_UART2_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_RX_DATA)

SW_PAD_CTL Register

Address: 3033_0000h base + 3A0h offset = 3033_03A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_UART2_RX_DATA field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: UART2_RX_DATA 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: UART2_RX_DATA 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: UART2_RX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: UART2_RX_DATA 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: UART2_RX_DATA 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.229 SW_PAD_CTL_PAD_UART2_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART2_TX_DATA)

SW_PAD_CTL Register

Address: 3033_0000h base + 3A4h offset = 3033_03A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_UART2_TX_DATA field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: UART2_TX_DATA 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: UART2_TX_DATA 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: UART2_TX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: UART2_TX_DATA 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: UART2_TX_DATA 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART2_TX_DATA field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.230 SW_PAD_CTL_PAD_UART3_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_RX_DATA)

SW_PAD_CTL Register

Address: 3033_0000h base + 3A8h offset = 3033_03A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_UART3_RX_DATA field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: UART3_RX_DATA 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: UART3_RX_DATA 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: UART3_RX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: UART3_RX_DATA

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART3_RX_DATA field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: UART3_RX_DATA 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.231 SW_PAD_CTL_PAD_UART3_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_TX_DATA)

SW_PAD_CTL Register

Address: 3033_0000h base + 3ACh offset = 3033_03ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_UART3_TX_DATA field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: UART3_TX_DATA 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: UART3_TX_DATA 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART3_TX_DATA field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: UART3_TX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: UART3_TX_DATA 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: UART3_TX_DATA 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.232 SW_PAD_CTL_PAD_UART3_RTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_RTS_B)

SW_PAD_CTL Register

Address: 3033_0000h base + 3B0h offset = 3033_03B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_UART3_RTS_B field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: UART3_RTS_B 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_UART3_RTS_B field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: UART3_RTS_B 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: UART3_RTS_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: UART3_RTS_B 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: UART3_RTS_B 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.233 SW_PAD_CTL_PAD_UART3_CTS_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_UART3_CTS_B)

SW_PAD_CTL Register

Address: 3033_0000h base + 3B4h offset = 3033_03B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_UART3_CTS_B field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: UART3_CTS_B 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: UART3_CTS_B 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: UART3_CTS_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: UART3_CTS_B 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: UART3_CTS_B 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.234 SW_PAD_CTL_PAD_I2C1_SCL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C1_SCL)

SW_PAD_CTL Register

Address: 3033_0000h base + 3B8h offset = 3033_03B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_I2C1_SCL field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: I2C1_SCL 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: I2C1_SCL 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: I2C1_SCL 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: I2C1_SCL 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: I2C1_SCL 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_I2C1_SCL field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.235 SW_PAD_CTL_PAD_I2C1_SDA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C1_SDA)

SW_PAD_CTL Register

Address: 3033_0000h base + 3BCh offset = 3033_03BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_I2C1_SDA field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: I2C1_SDA 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: I2C1_SDA 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: I2C1_SDA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: I2C1_SDA

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_I2C1_SDA field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: I2C1_SDA 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.236 SW_PAD_CTL_PAD_I2C2_SCL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C2_SCL)

SW_PAD_CTL Register

Address: 3033_0000h base + 3C0h offset = 3033_03C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_I2C2_SCL field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: I2C2_SCL 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: I2C2_SCL 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_I2C2_SCL field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: I2C2_SCL 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: I2C2_SCL 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: I2C2_SCL 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.237 SW_PAD_CTL_PAD_I2C2_SDA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C2_SDA)

SW_PAD_CTL Register

Address: 3033_0000h base + 3C4h offset = 3033_03C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_I2C2_SDA field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: I2C2_SDA 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_I2C2_SDA field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: I2C2_SDA 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: I2C2_SDA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: I2C2_SDA 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: I2C2_SDA 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.238 SW_PAD_CTL_PAD_I2C3_SCL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C3_SCL)

SW_PAD_CTL Register

Address: 3033_0000h base + 3C8h offset = 3033_03C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_I2C3_SCL field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: I2C3_SCL 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: I2C3_SCL 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: I2C3_SCL 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: I2C3_SCL 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: I2C3_SCL 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.239 SW_PAD_CTL_PAD_I2C3_SDA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C3_SDA)

SW_PAD_CTL Register

Address: 3033_0000h base + 3CCh offset = 3033_03CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_I2C3_SDA field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: I2C3_SDA 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: I2C3_SDA 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: I2C3_SDA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: I2C3_SDA 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: I2C3_SDA 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_I2C3_SDA field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.240 SW_PAD_CTL_PAD_I2C4_SCL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C4_SCL)

SW_PAD_CTL Register

Address: 3033_0000h base + 3D0h offset = 3033_03D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_I2C4_SCL field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: I2C4_SCL 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: I2C4_SCL 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: I2C4_SCL 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: I2C4_SCL

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_I2C4_SCL field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: I2C4_SCL 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.241 SW_PAD_CTL_PAD_I2C4_SDA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_I2C4_SDA)

SW_PAD_CTL Register

Address: 3033_0000h base + 3D4h offset = 3033_03D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_I2C4_SDA field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: I2C4_SDA 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: I2C4_SDA 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_I2C4_SDA field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: I2C4_SDA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: I2C4_SDA 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: I2C4_SDA 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.242 SW_PAD_CTL_PAD_ECSPI1_SCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI1_SCLK)

SW_PAD_CTL Register

Address: 3033_0000h base + 3D8h offset = 3033_03D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ECSPI1_SCLK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ECSPI1_SCLK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ECSPI1_SCLK field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ECSPI1_SCLK 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ECSPI1_SCLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ECSPI1_SCLK 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ECSPI1_SCLK 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.243 SW_PAD_CTL_PAD_ECSPI1_MOSI SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI1_MOSI)

SW_PAD_CTL Register

Address: 3033_0000h base + 3DCh offset = 3033_03DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ECSPI1_MOSI field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ECSPI1_MOSI 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ECSPI1_MOSI 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ECSPI1_MOSI 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ECSPI1_MOSI 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ECSPI1_MOSI 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.244 SW_PAD_CTL_PAD_ECSPI1_MISO SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI1_MISO)

SW_PAD_CTL Register

Address: 3033_0000h base + 3E0h offset = 3033_03E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ECSPI1_MISO field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ECSPI1_MISO 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ECSPI1_MISO 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ECSPI1_MISO 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ECSPI1_MISO 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ECSPI1_MISO 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ECSPI1_MISO field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.245 SW_PAD_CTL_PAD_ECSPI1_SS0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI1_SS0)

SW_PAD_CTL Register

Address: 3033_0000h base + 3E4h offset = 3033_03E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ECSPI1_SS0 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ECSPI1_SS0 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ECSPI1_SS0 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ECSPI1_SS0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ECSPI1_SS0

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ECSPI1_SS0 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ECSPI1_SS0 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.246 SW_PAD_CTL_PAD_ECSPI2_SCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI2_SCLK)

SW_PAD_CTL Register

Address: 3033_0000h base + 3E8h offset = 3033_03E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ECSPI2_SCLK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ECSPI2_SCLK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ECSPI2_SCLK 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ECSPI2_SCLK field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: ECSPI2_SCLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ECSPI2_SCLK 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ECSPI2_SCLK 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.247 SW_PAD_CTL_PAD_ECSPI2_MOSI SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI2_MOSI)

SW_PAD_CTL Register

Address: 3033_0000h base + 3ECh offset = 3033_03ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ECSPI2_MOSI field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ECSPI2_MOSI 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ECSPI2_MOSI field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ECSPI2_MOSI 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ECSPI2_MOSI 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ECSPI2_MOSI 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ECSPI2_MOSI 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.248 SW_PAD_CTL_PAD_ECSPI2_MISO SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI2_MISO)

SW_PAD_CTL Register

Address: 3033_0000h base + 3F0h offset = 3033_03F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ECSPI2_MISO field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ECSPI2_MISO 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ECSPI2_MISO 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ECSPI2_MISO 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ECSPI2_MISO 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ECSPI2_MISO 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.249 SW_PAD_CTL_PAD_ECSPI2_SS0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ECSPI2_SS0)

SW_PAD_CTL Register

Address: 3033_0000h base + 3F4h offset = 3033_03F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ECSPI2_SS0 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ECSPI2_SS0 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ECSPI2_SS0 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ECSPI2_SS0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ECSPI2_SS0 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ECSPI2_SS0 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ECSPI2_SS0 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.250 SW_PAD_CTL_PAD_SD1_CD_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_CD_B)

SW_PAD_CTL Register

Address: 3033_0000h base + 3F8h offset = 3033_03F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_CD_B field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD1_CD_B 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD1_CD_B 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_CD_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD1_CD_B

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD1_CD_B field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD1_CD_B 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.251 SW_PAD_CTL_PAD_SD1_WP SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_WP)

SW_PAD_CTL Register

Address: 3033_0000h base + 3FCh offset = 3033_03FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_WP field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD1_WP 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD1_WP 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD1_WP field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_WP 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD1_WP 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD1_WP 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.252 SW_PAD_CTL_PAD_SD1_RESET_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_RESET_B)

SW_PAD_CTL Register

Address: 3033_0000h base + 400h offset = 3033_0400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_RESET_B field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD1_RESET_B 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD1_RESET_B field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD1_RESET_B 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_RESET_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD1_RESET_B 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD1_RESET_B 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.253 SW_PAD_CTL_PAD_SD1_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_CLK)

SW_PAD_CTL Register

Address: 3033_0000h base + 404h offset = 3033_0404h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_CLK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD1_CLK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD1_CLK 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_CLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD1_CLK 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD1_CLK 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.254 SW_PAD_CTL_PAD_SD1_CMD SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_CMD)

SW_PAD_CTL Register

Address: 3033_0000h base + 408h offset = 3033_0408h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_CMD field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD1_CMD 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD1_CMD 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_CMD 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD1_CMD 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD1_CMD 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD1_CMD field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.255 SW_PAD_CTL_PAD_SD1_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA0)

SW_PAD_CTL Register

Address: 3033_0000h base + 40Ch offset = 3033_040Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA0 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD1_DATA0 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD1_DATA0 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_DATA0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD1_DATA0

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA0 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD1_DATA0 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.256 SW_PAD_CTL_PAD_SD1_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA1)

SW_PAD_CTL Register

Address: 3033_0000h base + 410h offset = 3033_0410h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA1 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD1_DATA1 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD1_DATA1 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA1 field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_DATA1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD1_DATA1 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD1_DATA1 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.257 SW_PAD_CTL_PAD_SD1_DATA2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA2)

SW_PAD_CTL Register

Address: 3033_0000h base + 414h offset = 3033_0414h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA2 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD1_DATA2 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA2 field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD1_DATA2 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_DATA2 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD1_DATA2 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD1_DATA2 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.258 SW_PAD_CTL_PAD_SD1_DATA3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA3)

SW_PAD_CTL Register

Address: 3033_0000h base + 418h offset = 3033_0418h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD1_DATA3 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD1_DATA3 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD1_DATA3 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_DATA3 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD1_DATA3 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD1_DATA3 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.259 SW_PAD_CTL_PAD_SD2_CD_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_CD_B)

SW_PAD_CTL Register

Address: 3033_0000h base + 41Ch offset = 3033_041Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD2_CD_B field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD2_CD_B 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD2_CD_B 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD2_CD_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD2_CD_B 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD2_CD_B 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD2_CD_B field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.260 SW_PAD_CTL_PAD_SD2_WP SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_WP)

SW_PAD_CTL Register

Address: 3033_0000h base + 420h offset = 3033_0420h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD2_WP field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD2_WP 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD2_WP 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD2_WP 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD2_WP

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD2_WP field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD2_WP 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.261 SW_PAD_CTL_PAD_SD2_RESET_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_RESET_B)

SW_PAD_CTL Register

Address: 3033_0000h base + 424h offset = 3033_0424h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD2_RESET_B field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD2_RESET_B 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD2_RESET_B 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD2_RESET_B field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD2_RESET_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD2_RESET_B 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD2_RESET_B 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.262 SW_PAD_CTL_PAD_SD2_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_CLK)

SW_PAD_CTL Register

Address: 3033_0000h base + 428h offset = 3033_0428h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD2_CLK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD2_CLK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD2_CLK field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD2_CLK 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD2_CLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD2_CLK 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD2_CLK 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.263 SW_PAD_CTL_PAD_SD2_CMD SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_CMD)

SW_PAD_CTL Register

Address: 3033_0000h base + 42Ch offset = 3033_042Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD2_CMD field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD2_CMD 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD2_CMD 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD2_CMD 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD2_CMD 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD2_CMD 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.264 SW_PAD_CTL_PAD_SD2_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_DATA0)

SW_PAD_CTL Register

Address: 3033_0000h base + 430h offset = 3033_0430h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD2_DATA0 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD2_DATA0 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD2_DATA0 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD2_DATA0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD2_DATA0 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD2_DATA0 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD2_DATA0 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.265 SW_PAD_CTL_PAD_SD2_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_DATA1)

SW_PAD_CTL Register

Address: 3033_0000h base + 434h offset = 3033_0434h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD2_DATA1 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD2_DATA1 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD2_DATA1 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD2_DATA1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD2_DATA1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD2_DATA1 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD2_DATA1 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.266 SW_PAD_CTL_PAD_SD2_DATA2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_DATA2)

SW_PAD_CTL Register

Address: 3033_0000h base + 438h offset = 3033_0438h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD2_DATA2 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD2_DATA2 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD2_DATA2 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD2_DATA2 field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD2_DATA2 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD2_DATA2 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD2_DATA2 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.267 SW_PAD_CTL_PAD_SD2_DATA3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD2_DATA3)

SW_PAD_CTL Register

Address: 3033_0000h base + 43Ch offset = 3033_043Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD2_DATA3 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD2_DATA3 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD2_DATA3 field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD2_DATA3 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD2_DATA3 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD2_DATA3 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD2_DATA3 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.268 SW_PAD_CTL_PAD_SD3_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_CLK)

SW_PAD_CTL Register

Address: 3033_0000h base + 440h offset = 3033_0440h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD3_CLK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD3_CLK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD3_CLK 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD3_CLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD3_CLK 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD3_CLK 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.269 SW_PAD_CTL_PAD_SD3_CMD SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_CMD)

SW_PAD_CTL Register

Address: 3033_0000h base + 444h offset = 3033_0444h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD3_CMD field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD3_CMD 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD3_CMD 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD3_CMD 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD3_CMD 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD3_CMD 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD3_CMD field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.270 SW_PAD_CTL_PAD_SD3_DATA0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA0)

SW_PAD_CTL Register

Address: 3033_0000h base + 448h offset = 3033_0448h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA0 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD3_DATA0 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD3_DATA0 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD3_DATA0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD3_DATA0

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA0 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD3_DATA0 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.271 SW_PAD_CTL_PAD_SD3_DATA1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA1)

SW_PAD_CTL Register

Address: 3033_0000h base + 44Ch offset = 3033_044Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA1 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD3_DATA1 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD3_DATA1 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA1 field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD3_DATA1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD3_DATA1 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD3_DATA1 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.272 SW_PAD_CTL_PAD_SD3_DATA2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA2)

SW_PAD_CTL Register

Address: 3033_0000h base + 450h offset = 3033_0450h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA2 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD3_DATA2 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA2 field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD3_DATA2 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD3_DATA2 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD3_DATA2 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD3_DATA2 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.273 SW_PAD_CTL_PAD_SD3_DATA3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA3)

SW_PAD_CTL Register

Address: 3033_0000h base + 454h offset = 3033_0454h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA3 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD3_DATA3 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD3_DATA3 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD3_DATA3 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD3_DATA3 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD3_DATA3 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.274 SW_PAD_CTL_PAD_SD3_DATA4 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA4)

SW_PAD_CTL Register

Address: 3033_0000h base + 458h offset = 3033_0458h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA4 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD3_DATA4 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD3_DATA4 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD3_DATA4 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD3_DATA4 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD3_DATA4 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA4 field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.275 SW_PAD_CTL_PAD_SD3_DATA5 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA5)

SW_PAD_CTL Register

Address: 3033_0000h base + 45Ch offset = 3033_045Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA5 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD3_DATA5 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD3_DATA5 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD3_DATA5 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD3_DATA5

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA5 field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD3_DATA5 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.276 SW_PAD_CTL_PAD_SD3_DATA6 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA6)

SW_PAD_CTL Register

Address: 3033_0000h base + 460h offset = 3033_0460h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA6 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD3_DATA6 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD3_DATA6 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA6 field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD3_DATA6 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD3_DATA6 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD3_DATA6 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.277 SW_PAD_CTL_PAD_SD3_DATA7 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_DATA7)

SW_PAD_CTL Register

Address: 3033_0000h base + 464h offset = 3033_0464h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA7 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD3_DATA7 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD3_DATA7 field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD3_DATA7 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD3_DATA7 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD3_DATA7 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD3_DATA7 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.278 SW_PAD_CTL_PAD_SD3_STROBE SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_STROBE)

SW_PAD_CTL Register

Address: 3033_0000h base + 468h offset = 3033_0468h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD3_STROBE field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD3_STROBE 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD3_STROBE 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD3_STROBE 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD3_STROBE 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD3_STROBE 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.279 SW_PAD_CTL_PAD_SD3_RESET_B SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SD3_RESET_B)

SW_PAD_CTL Register

Address: 3033_0000h base + 46Ch offset = 3033_046Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SD3_RESET_B field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SD3_RESET_B 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SD3_RESET_B 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SD3_RESET_B 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SD3_RESET_B 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SD3_RESET_B 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SD3_RESET_B field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.280 SW_PAD_CTL_PAD_SAI1_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI1_RX_DATA)

SW_PAD_CTL Register

Address: 3033_0000h base + 470h offset = 3033_0470h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SAI1_RX_DATA field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SAI1_RX_DATA 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SAI1_RX_DATA 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SAI1_RX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SAI1_RX_DATA

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SAI1_RX_DATA field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SAI1_RX_DATA 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.281 SW_PAD_CTL_PAD_SAI1_TX_BCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI1_TX_BCLK)

SW_PAD_CTL Register

Address: 3033_0000h base + 474h offset = 3033_0474h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SAI1_TX_BCLK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SAI1_TX_BCLK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SAI1_TX_BCLK 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SAI1_TX_BCLK field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: SAI1_TX_BCLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SAI1_TX_BCLK 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SAI1_TX_BCLK 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.282 SW_PAD_CTL_PAD_SAI1_TX_SYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI1_TX_SYNC)

SW_PAD_CTL Register

Address: 3033_0000h base + 478h offset = 3033_0478h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SAI1_TX_SYNC field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SAI1_TX_SYNC 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SAI1_TX_SYNC field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SAI1_TX_SYNC 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SAI1_TX_SYNC 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SAI1_TX_SYNC 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SAI1_TX_SYNC 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.283 SW_PAD_CTL_PAD_SAI1_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI1_TX_DATA)

SW_PAD_CTL Register

Address: 3033_0000h base + 47Ch offset = 3033_047Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SAI1_TX_DATA field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SAI1_TX_DATA 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SAI1_TX_DATA 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SAI1_TX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SAI1_TX_DATA 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SAI1_TX_DATA 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.284 SW_PAD_CTL_PAD_SAI1_RX_SYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI1_RX_SYNC)

SW_PAD_CTL Register

Address: 3033_0000h base + 480h offset = 3033_0480h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SAI1_RX_SYNC field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SAI1_RX_SYNC 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SAI1_RX_SYNC 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SAI1_RX_SYNC 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SAI1_RX_SYNC 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SAI1_RX_SYNC 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SAI1_RX_SYNC field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.285 SW_PAD_CTL_PAD_SAI1_RX_BCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI1_RX_BCLK)

SW_PAD_CTL Register

Address: 3033_0000h base + 484h offset = 3033_0484h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SAI1_RX_BCLK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SAI1_RX_BCLK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SAI1_RX_BCLK 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SAI1_RX_BCLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SAI1_RX_BCLK

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SAI1_RX_BCLK field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SAI1_RX_BCLK 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.286 SW_PAD_CTL_PAD_SAI1_MCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI1_MCLK)

SW_PAD_CTL Register

Address: 3033_0000h base + 488h offset = 3033_0488h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SAI1_MCLK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SAI1_MCLK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SAI1_MCLK 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SAI1_MCLK field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: SAI1_MCLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SAI1_MCLK 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SAI1_MCLK 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.287 SW_PAD_CTL_PAD_SAI2_TX_SYNC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI2_TX_SYNC)

SW_PAD_CTL Register

Address: 3033_0000h base + 48Ch offset = 3033_048Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SAI2_TX_SYNC field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SAI2_TX_SYNC 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SAI2_TX_SYNC field descriptions (continued)

Field	Description
	10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SAI2_TX_SYNC 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SAI2_TX_SYNC 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SAI2_TX_SYNC 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SAI2_TX_SYNC 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.288 SW_PAD_CTL_PAD_SAI2_TX_BCLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI2_TX_BCLK)

SW_PAD_CTL Register

Address: 3033_0000h base + 490h offset = 3033_0490h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SAI2_TX_BCLK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SAI2_TX_BCLK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SAI2_TX_BCLK 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SAI2_TX_BCLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SAI2_TX_BCLK 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SAI2_TX_BCLK 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.289 SW_PAD_CTL_PAD_SAI2_RX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI2_RX_DATA)

SW_PAD_CTL Register

Address: 3033_0000h base + 494h offset = 3033_0494h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SAI2_RX_DATA field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SAI2_RX_DATA 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SAI2_RX_DATA 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SAI2_RX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SAI2_RX_DATA 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SAI2_RX_DATA 00 **DSE_0_X1** — X1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SAI2_RX_DATA field descriptions (continued)

Field	Description
	01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.290 SW_PAD_CTL_PAD_SAI2_TX_DATA SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_SAI2_TX_DATA)

SW_PAD_CTL Register

Address: 3033_0000h base + 498h offset = 3033_0498h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_SAI2_TX_DATA field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: SAI2_TX_DATA 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: SAI2_TX_DATA 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: SAI2_TX_DATA 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: SAI2_TX_DATA

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_SAI2_TX_DATA field descriptions (continued)

Field	Description
	0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: SAI2_TX_DATA 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.291 SW_PAD_CTL_PAD_ENET1_RGMII_RD0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD0)

SW_PAD_CTL Register

Address: 3033_0000h base + 49Ch offset = 3033_049Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD0 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_RGMII_RD0 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ENET1_RGMII_RD0 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD0 field descriptions (continued)

Field	Description
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RGMII_RD0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RGMII_RD0 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_RGMII_RD0 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.292 SW_PAD_CTL_PAD_ENET1_RGMII_RD1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD1)

SW_PAD_CTL Register

Address: 3033_0000h base + 4A0h offset = 3033_04A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD1 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_RGMII_RD1 00 **PS_0_100K_PD** — 100K PD

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD1 field descriptions (continued)

Field	Description
	01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ENET1_RGMII_RD1 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RGMII_RD1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RGMII_RD1 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_RGMII_RD1 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.293 SW_PAD_CTL_PAD_ENET1_RGMII_RD2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD2)

SW_PAD_CTL Register

Address: 3033_0000h base + 4A4h offset = 3033_04A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD2 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_RGMII_RD2 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ENET1_RGMII_RD2 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RGMII_RD2 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RGMII_RD2 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_RGMII_RD2 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.294 SW_PAD_CTL_PAD_ENET1_RGMII_RD3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD3)

SW_PAD_CTL Register

Address: 3033_0000h base + 4A8h offset = 3033_04A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD3 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_RGMII_RD3 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ENET1_RGMII_RD3 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RGMII_RD3 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RGMII_RD3 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_RGMII_RD3

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RD3 field descriptions (continued)

Field	Description
	00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.295 SW_PAD_CTL_PAD_ENET1_RGMII_RX_CTL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RX_CTL)

SW_PAD_CTL Register

Address: 3033_0000h base + 4ACh offset = 3033_04ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RX_CTL field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_RGMII_RX_CTL 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ENET1_RGMII_RX_CTL 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RGMII_RX_CTL 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RX_CTL field descriptions (continued)

Field	Description
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RGMII_RX_CTL 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_RGMII_RX_CTL 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.296 SW_PAD_CTL_PAD_ENET1_RGMII_RXC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RXC)

SW_PAD_CTL Register

Address: 3033_0000h base + 4B0h offset = 3033_04B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RXC field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_RGMII_RXC 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_RXC field descriptions (continued)

Field	Description
	Select one out of next values for pad: ENET1_RGMII_RXC 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RGMII_RXC 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RGMII_RXC 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_RGMII_RXC 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.297 SW_PAD_CTL_PAD_ENET1_RGMII_TD0 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD0)

SW_PAD_CTL Register

Address: 3033_0000h base + 4B4h offset = 3033_04B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD0 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD0 field descriptions (continued)

Field	Description
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_RGMII_TD0 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ENET1_RGMII_TD0 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RGMII_TD0 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RGMII_TD0 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_RGMII_TD0 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.298 SW_PAD_CTL_PAD_ENET1_RGMII_TD1 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD1)

SW_PAD_CTL Register

Address: 3033_0000h base + 4B8h offset = 3033_04B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD1 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_RGMII_TD1 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ENET1_RGMII_TD1 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RGMII_TD1 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RGMII_TD1 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_RGMII_TD1

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD1 field descriptions (continued)

Field	Description
	00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.299 SW_PAD_CTL_PAD_ENET1_RGMII_TD2 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD2)

SW_PAD_CTL Register

Address: 3033_0000h base + 4BCh offset = 3033_04BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD2 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_RGMII_TD2 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ENET1_RGMII_TD2 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RGMII_TD2 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD2 field descriptions (continued)

Field	Description
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RGMII_TD2 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_RGMII_TD2 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.300 SW_PAD_CTL_PAD_ENET1_RGMII_TD3 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD3)

SW_PAD_CTL Register

Address: 3033_0000h base + 4C0h offset = 3033_04C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD3 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_RGMII_TD3 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TD3 field descriptions (continued)

Field	Description
	Select one out of next values for pad: ENET1_RGMII_TD3 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RGMII_TD3 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RGMII_TD3 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_RGMII_TD3 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.301 SW_PAD_CTL_PAD_ENET1_RGMII_TX_CTL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TX_CTL)

SW_PAD_CTL Register

Address: 3033_0000h base + 4C4h offset = 3033_04C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TX_CTL field descriptions

Field	Description
31–7 -	This field is reserved. Reserved

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TX_CTL field descriptions (continued)

Field	Description
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_RGMII_TX_CTL 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ENET1_RGMII_TX_CTL 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RGMII_TX_CTL 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RGMII_TX_CTL 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_RGMII_TX_CTL 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.302 SW_PAD_CTL_PAD_ENET1_RGMII_TXC SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TXC)

SW_PAD_CTL Register

Address: 3033_0000h base + 4C8h offset = 3033_04C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TXC field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_RGMII_TXC 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ENET1_RGMII_TXC 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RGMII_TXC 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RGMII_TXC 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_RGMII_TXC

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RGMII_TXC field descriptions (continued)

Field	Description
	00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.303 SW_PAD_CTL_PAD_ENET1_TX_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_CLK)

SW_PAD_CTL Register

Address: 3033_0000h base + 4CCh offset = 3033_04CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_CLK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_TX_CLK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ENET1_TX_CLK 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_TX_CLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_TX_CLK field descriptions (continued)

Field	Description
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_TX_CLK 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_TX_CLK 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.304 SW_PAD_CTL_PAD_ENET1_RX_CLK SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_CLK)

SW_PAD_CTL Register

Address: 3033_0000h base + 4D0h offset = 3033_04D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_CLK field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_RX_CLK 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ENET1_RX_CLK

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_RX_CLK field descriptions (continued)

Field	Description
	0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_RX_CLK 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_RX_CLK 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_RX_CLK 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.305 SW_PAD_CTL_PAD_ENET1_CRS SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_CRS)

SW_PAD_CTL Register

Address: 3033_0000h base + 4D4h offset = 3033_04D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PS			PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_CRS field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_CRS

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_ENET1_CRS field descriptions (continued)

Field	Description
	00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ENET1_CRS 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_CRS 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_CRS 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_CRS 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.306 SW_PAD_CTL_PAD_ENET1_COL SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_ENET1_COL)

SW_PAD_CTL Register

Address: 3033_0000h base + 4D8h offset = 3033_04D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved									PS		PE	HYS	SRE	DSE	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

IOMUXC_SW_PAD_CTL_PAD_ENET1_COL field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6–5 PS	Pull Select Field Select one out of next values for pad: ENET1_COL 00 **PS_0_100K_PD** — 100K PD 01 **PS_1_5K_PU** — 5K PU 10 **PS_2_47K_PU** — 47K PU 11 **PS_3_100K_PU** — 100K PU
4 PE	Pull Enable Field Select one out of next values for pad: ENET1_COL 0 **PE_0_Pull_Disabled** — Pull Disabled 1 **PE_1_Pull_Enabled** — Pull Enabled
3 HYS	Hyst. Enable Field Select one out of next values for pad: ENET1_COL 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
2 SRE	Slew Rate Field Select one out of next values for pad: ENET1_COL 0 **SRE_0_Fast_Slew_Rate** — Fast Slew Rate 1 **SRE_1_Slow_Slew_Rate** — Slow Slew Rate
DSE	Drive Strength Field Select one out of next values for pad: ENET1_COL 00 **DSE_0_X1** — X1 01 **DSE_1_X4** — X4 10 **DSE_2_X2** — X2 11 **DSE_3_X6** — X6

8.2.7.307 FLEXCAN1_RX_SELECT_INPUT DAISY Register (IOMUXC_FLEXCAN1_RX_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 4DCh offset = 3033_04DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																													DAISY		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

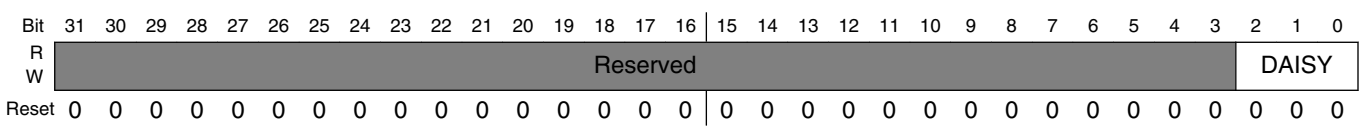
IOMUXC_FLEXCAN1_RX_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 000 **GPIO1_IO12_ALT3** — Selecting Pad: GPIO1_IO12 Mode: ALT3 for FLEXCAN1_RX 001 **I2C1_SCL_ALT2** — Selecting Pad: I2C1_SCL Mode: ALT2 for FLEXCAN1_RX 010 **SD3_DATA7_ALT4** — Selecting Pad: SD3_DATA7 Mode: ALT4 for FLEXCAN1_RX 011 **SAI1_RX_DATA_ALT3** — Selecting Pad: SAI1_RX_DATA Mode: ALT3 for FLEXCAN1_RX 100 **ENET1_RGMII_RD2_ALT1** — Selecting Pad: ENET1_RGMII_RD2 Mode: ALT1 for FLEXCAN1_RX

8.2.7.308 FLEXCAN2_RX_SELECT_INPUT DAISY Register (IOMUXC_FLEXCAN2_RX_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 4E0h offset = 3033_04E0h



IOMUXC_FLEXCAN2_RX_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 000 **GPIO1_IO14_ALT3** — Selecting Pad: GPIO1_IO14 Mode: ALT3 for FLEXCAN2_RX 001 **I2C3_SCL_ALT2** — Selecting Pad: I2C3_SCL Mode: ALT2 for FLEXCAN2_RX 010 **SD3_DATA4_ALT4** — Selecting Pad: SD3_DATA4 Mode: ALT4 for FLEXCAN2_RX 011 **SAI1_TX_SYNC_ALT3** — Selecting Pad: SAI1_TX_SYNC Mode: ALT3 for FLEXCAN2_RX 100 **ENET1_RGMII_TD2_ALT1** — Selecting Pad: ENET1_RGMII_TD2 Mode: ALT1 for FLEXCAN2_RX

8.2.7.309 CCM_EXT_CLK_1_SELECT_INPUT DAISY Register (IOMUXC_CCM_EXT_CLK_1_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 4E4h offset = 3033_04E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_CCM_EXT_CLK_1_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO12_ALT5** — Selecting Pad: GPIO1_IO12 Mode: ALT5 for CCM_EXT_CLK_1 01 **SD1_DATA0_ALT6** — Selecting Pad: SD1_DATA0 Mode: ALT6 for CCM_EXT_CLK_1 10 **ENET1_TX_CLK_ALT6** — Selecting Pad: ENET1_TX_CLK Mode: ALT6 for CCM_EXT_CLK_1

8.2.7.310 CCM_EXT_CLK_2_SELECT_INPUT DAISY Register (IOMUXC_CCM_EXT_CLK_2_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 4E8h offset = 3033_04E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_CCM_EXT_CLK_2_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO13_ALT5** — Selecting Pad: GPIO1_IO13 Mode: ALT5 for CCM_EXT_CLK_2 01 **SD1_DATA1_ALT6** — Selecting Pad: SD1_DATA1 Mode: ALT6 for CCM_EXT_CLK_2 10 **ENET1_RX_CLK_ALT6** — Selecting Pad: ENET1_RX_CLK Mode: ALT6 for CCM_EXT_CLK_2

8.2.7.311 CCM_EXT_CLK_3_SELECT_INPUT DAISY Register (IOMUXC_CCM_EXT_CLK_3_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 4ECh offset = 3033_04ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_CCM_EXT_CLK_3_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO14_ALT5** — Selecting Pad: GPIO1_IO14 Mode: ALT5 for CCM_EXT_CLK_3 01 **SD1_DATA2_ALT6** — Selecting Pad: SD1_DATA2 Mode: ALT6 for CCM_EXT_CLK_3 10 **ENET1_CRS_ALT6** — Selecting Pad: ENET1_CRS Mode: ALT6 for CCM_EXT_CLK_3

8.2.7.312 CCM_EXT_CLK_4_SELECT_INPUT DAISY Register (IOMUXC_CCM_EXT_CLK_4_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 4F0h offset = 3033_04F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_CCM_EXT_CLK_4_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO15_ALT5** — Selecting Pad: GPIO1_IO15 Mode: ALT5 for CCM_EXT_CLK_4 01 **SD1_DATA3_ALT6** — Selecting Pad: SD1_DATA3 Mode: ALT6 for CCM_EXT_CLK_4 10 **ENET1_COL_ALT6** — Selecting Pad: ENET1_COL Mode: ALT6 for CCM_EXT_CLK_4

8.2.7.313 CCM_PMIC_READY_SELECT_INPUT DAISY Register (IOMUXC_CCM_PMIC_READY_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 4F4h offset = 3033_04F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

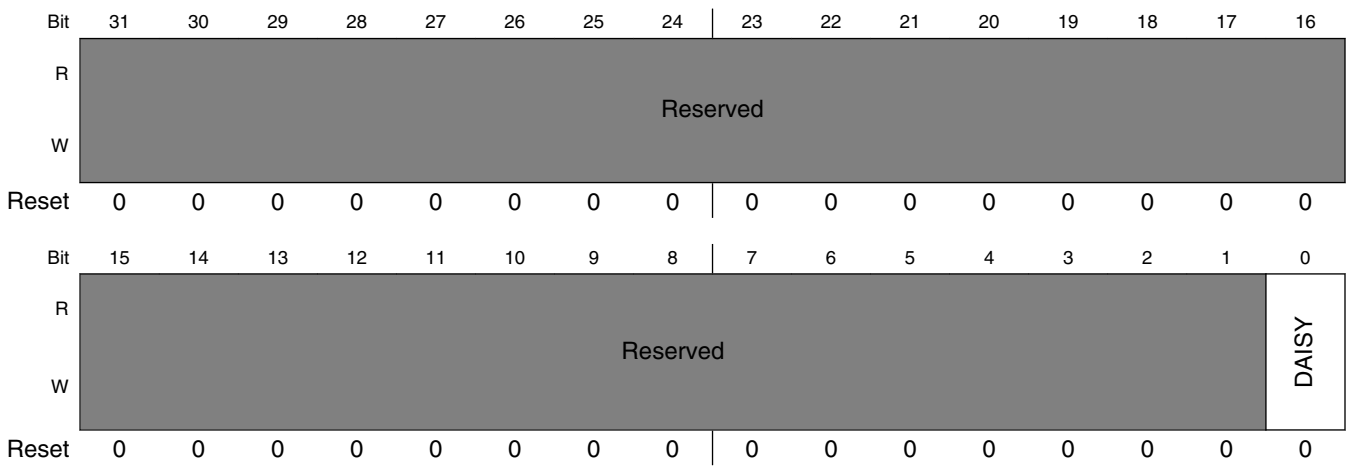
IOMUXC_CCM_PMIC_READY_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO09_ALT5** — Selecting Pad: GPIO1_IO09 Mode: ALT5 for CCM_PMIC_READY 01 **GPIO1_IO13_ALT4** — Selecting Pad: GPIO1_IO13 Mode: ALT4 for CCM_PMIC_READY 10 **UART1_RX_DATA_ALT2** — Selecting Pad: UART1_RX_DATA Mode: ALT2 for CCM_PMIC_READY 11 **SAI1_MCLK_ALT3** — Selecting Pad: SAI1_MCLK Mode: ALT3 for CCM_PMIC_READY

8.2.7.314 CSI_DATA2_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA2_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 4F8h offset = 3033_04F8h



IOMUXC_CSI_DATA2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA15_ALT3** — Selecting Pad: LCD_DATA15 Mode: ALT3 for CSI_DATA2 1 **ECSPI1_SCLK_ALT3** — Selecting Pad: ECSPI1_SCLK Mode: ALT3 for CSI_DATA2

8.2.7.315 CSI_DATA3_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA3_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 4FCh offset = 3033_04FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_CSI_DATA3_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA14_ALT3** — Selecting Pad: LCD_DATA14 Mode: ALT3 for CSI_DATA3 1 **ECSPI1_MOSI_ALT3** — Selecting Pad: ECSPI1_MOSI Mode: ALT3 for CSI_DATA3

8.2.7.316 CSI_DATA4_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA4_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 500h offset = 3033_0500h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

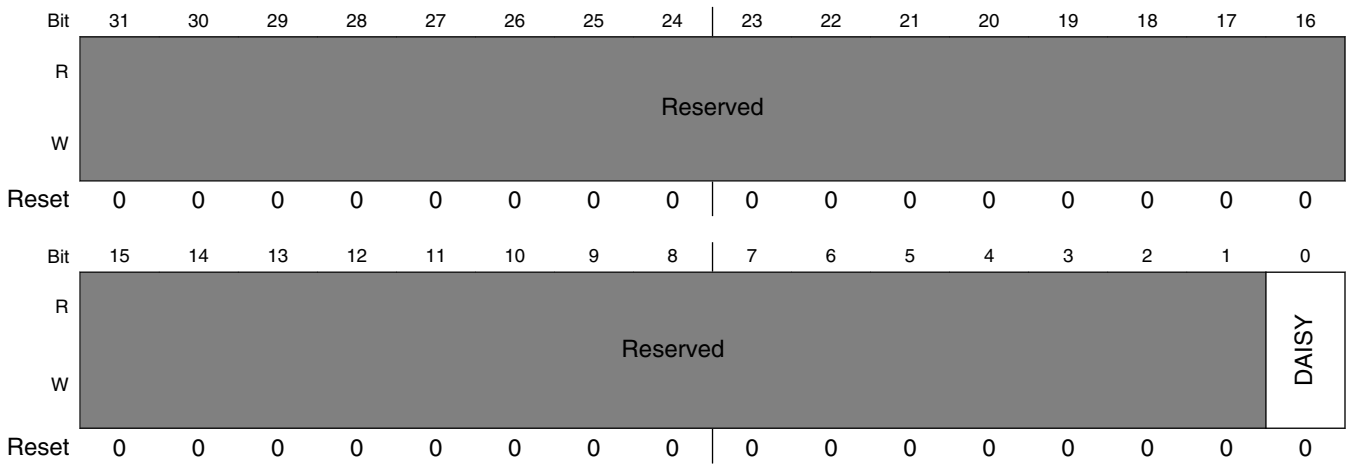
IOMUXC_CSI_DATA4_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA13_ALT3** — Selecting Pad: LCD_DATA13 Mode: ALT3 for CSI_DATA4 1 **ECSPI1_MISO_ALT3** — Selecting Pad: ECSPI1_MISO Mode: ALT3 for CSI_DATA4

8.2.7.317 CSI_DATA5_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA5_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 504h offset = 3033_0504h



IOMUXC_CSI_DATA5_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA12_ALT3** — Selecting Pad: LCD_DATA12 Mode: ALT3 for CSI_DATA5 1 **ECSPI1_SS0_ALT3** — Selecting Pad: ECSPI1_SS0 Mode: ALT3 for CSI_DATA5

8.2.7.318 CSI_DATA6_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA6_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 508h offset = 3033_0508h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

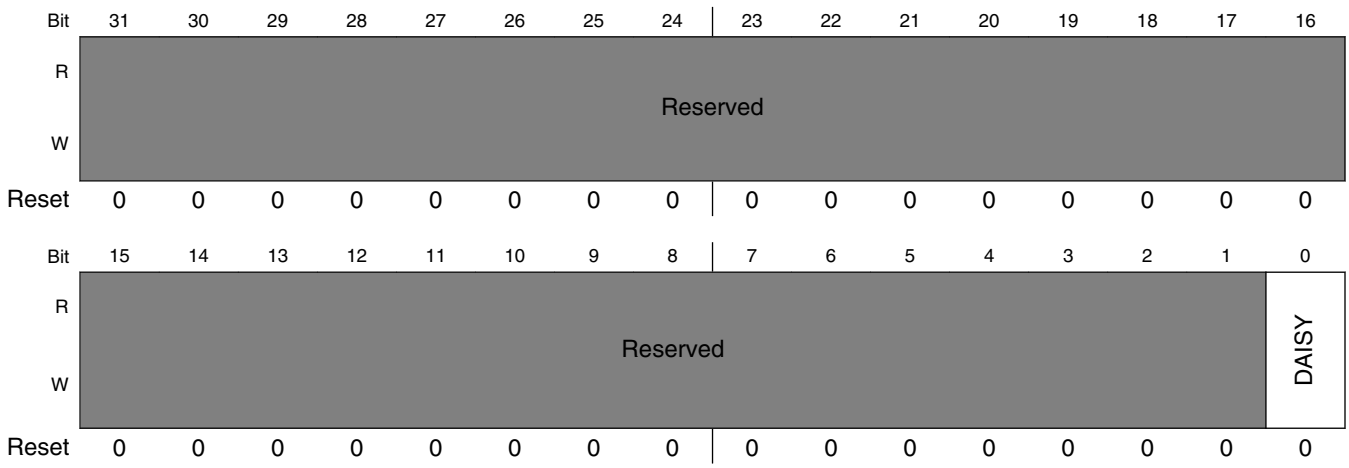
IOMUXC_CSI_DATA6_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA11_ALT3** — Selecting Pad: LCD_DATA11 Mode: ALT3 for CSI_DATA6 1 **ECSPI2_SCLK_ALT3** — Selecting Pad: ECSPI2_SCLK Mode: ALT3 for CSI_DATA6

8.2.7.319 CSI_DATA7_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA7_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 50Ch offset = 3033_050Ch



IOMUXC_CSI_DATA7_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA10_ALT3** — Selecting Pad: LCD_DATA10 Mode: ALT3 for CSI_DATA7 1 **ECSPI2_MOSI_ALT3** — Selecting Pad: ECSPI2_MOSI Mode: ALT3 for CSI_DATA7

8.2.7.320 CSI_DATA8_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA8_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 510h offset = 3033_0510h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_CSI_DATA8_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA09_ALT3** — Selecting Pad: LCD_DATA09 Mode: ALT3 for CSI_DATA8 1 **ECSPI2_MISO_ALT3** — Selecting Pad: ECSPI2_MISO Mode: ALT3 for CSI_DATA8

8.2.7.321 CSI_DATA9_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA9_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 514h offset = 3033_0514h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_CSI_DATA9_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA08_ALT3** — Selecting Pad: LCD_DATA08 Mode: ALT3 for CSI_DATA9 1 **ECSPI2_SS0_ALT3** — Selecting Pad: ECSPI2_SS0 Mode: ALT3 for CSI_DATA9

8.2.7.322 CSI_HSYNC_SELECT_INPUT DAISY Register (IOMUXC_CSI_HSYNC_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 518h offset = 3033_0518h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

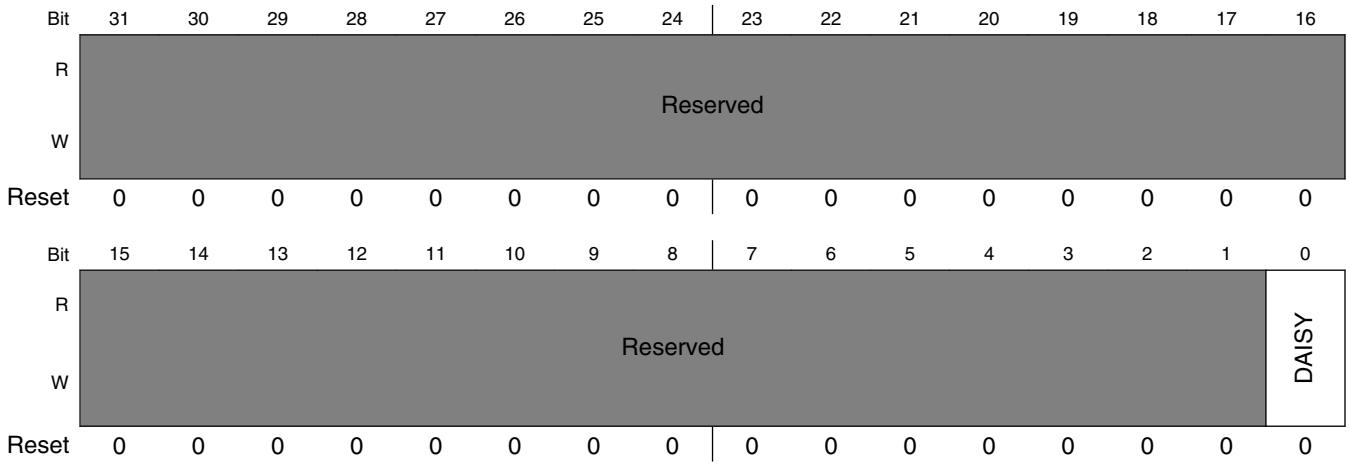
IOMUXC_CSI_HSYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA05_ALT3** — Selecting Pad: LCD_DATA05 Mode: ALT3 for CSI_HSYNC 1 **I2C3_SDA_ALT3** — Selecting Pad: I2C3_SDA Mode: ALT3 for CSI_HSYNC

8.2.7.323 CSI_PIXCLK_SELECT_INPUT DAISY Register (IOMUXC_CSI_PIXCLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 51Ch offset = 3033_051Ch



IOMUXC_CSI_PIXCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA06_ALT3** — Selecting Pad: LCD_DATA06 Mode: ALT3 for CSI_PIXCLK 1 **I2C4_SCL_ALT3** — Selecting Pad: I2C4_SCL Mode: ALT3 for CSI_PIXCLK

8.2.7.324 CSI_VSYNC_SELECT_INPUT DAISY Register (IOMUXC_CSI_VSYNC_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 520h offset = 3033_0520h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

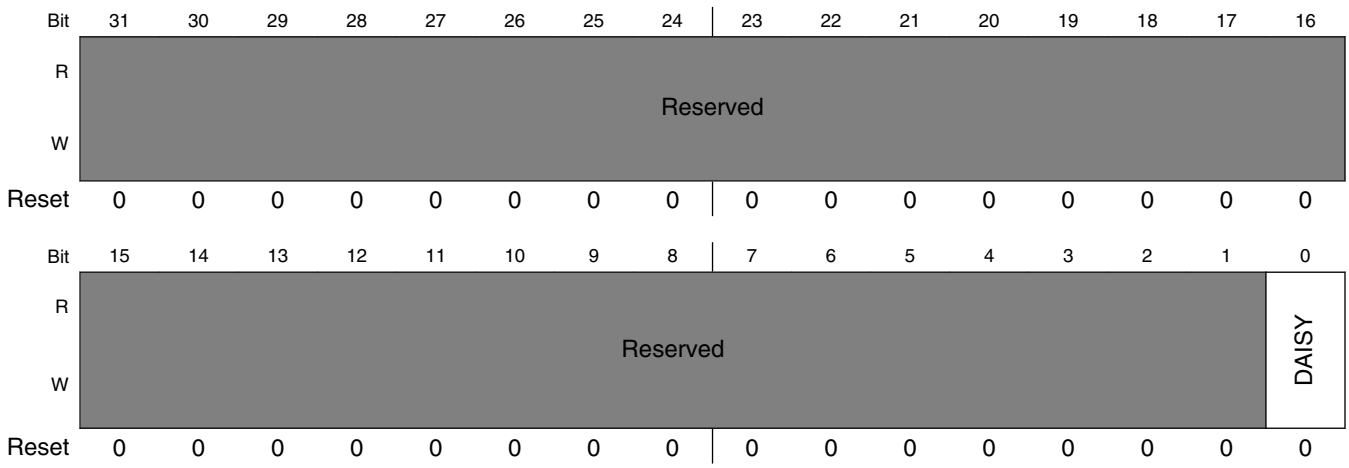
IOMUXC_CSI_VSYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA04_ALT3** — Selecting Pad: LCD_DATA04 Mode: ALT3 for CSI_VSYNC 1 **I2C3_SCL_ALT3** — Selecting Pad: I2C3_SCL Mode: ALT3 for CSI_VSYNC

8.2.7.325 ECSPI1_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_SCLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 524h offset = 3033_0524h



IOMUXC_ECSPI1_SCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **UART3_RTS_B_ALT3** — Selecting Pad: UART3_RTS_B Mode: ALT3 for ECSPI1_SCLK 1 **ECSPI1_SCLK_ALT0** — Selecting Pad: ECSPI1_SCLK Mode: ALT0 for ECSPI1_SCLK

8.2.7.326 ECSPI1_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_MISO_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 528h offset = 3033_0528h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

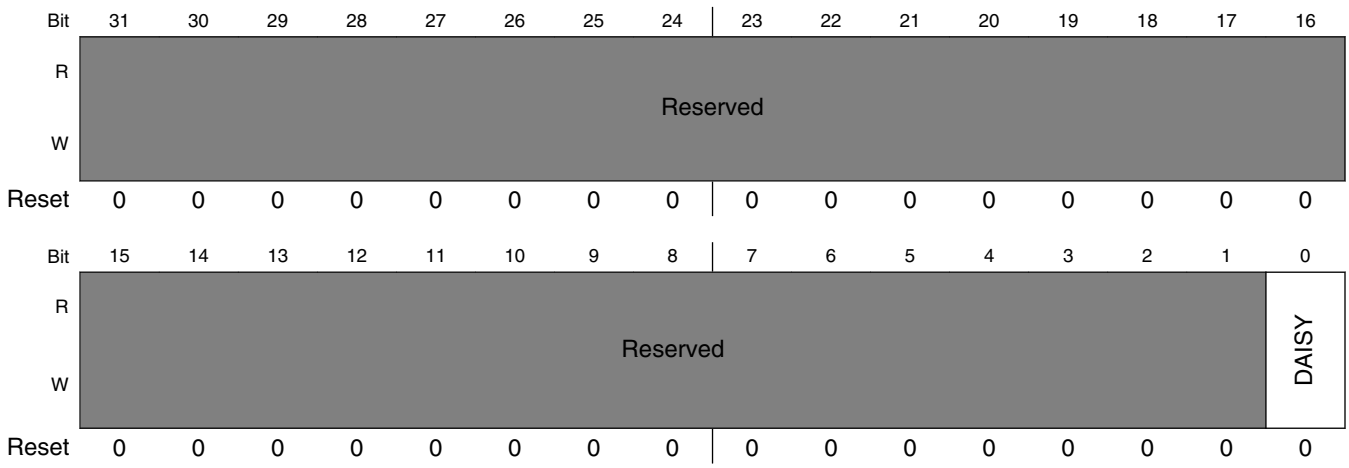
IOMUXC_ECSPI1_MISO_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **UART3_RX_DATA_ALT3** — Selecting Pad: UART3_RX_DATA Mode: ALT3 for ECSPI1_MISO 1 **ECSPI1_MISO_ALT0** — Selecting Pad: ECSPI1_MISO Mode: ALT0 for ECSPI1_MISO

8.2.7.327 ECSPI1_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_MOSI_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 52Ch offset = 3033_052Ch



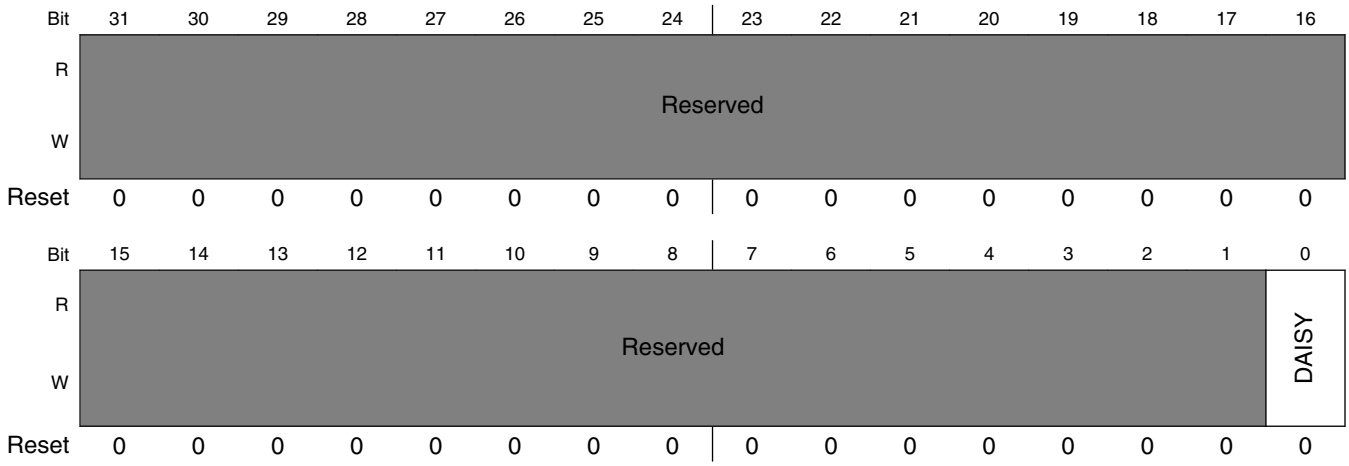
IOMUXC_ECSPI1_MOSI_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **UART3_TX_DATA_ALT3** — Selecting Pad: UART3_TX_DATA Mode: ALT3 for ECSPI1_MOSI 1 **ECSPI1_MOSI_ALT0** — Selecting Pad: ECSPI1_MOSI Mode: ALT0 for ECSPI1_MOSI

8.2.7.328 ECSPI1_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI1_SS0_B_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 530h offset = 3033_0530h



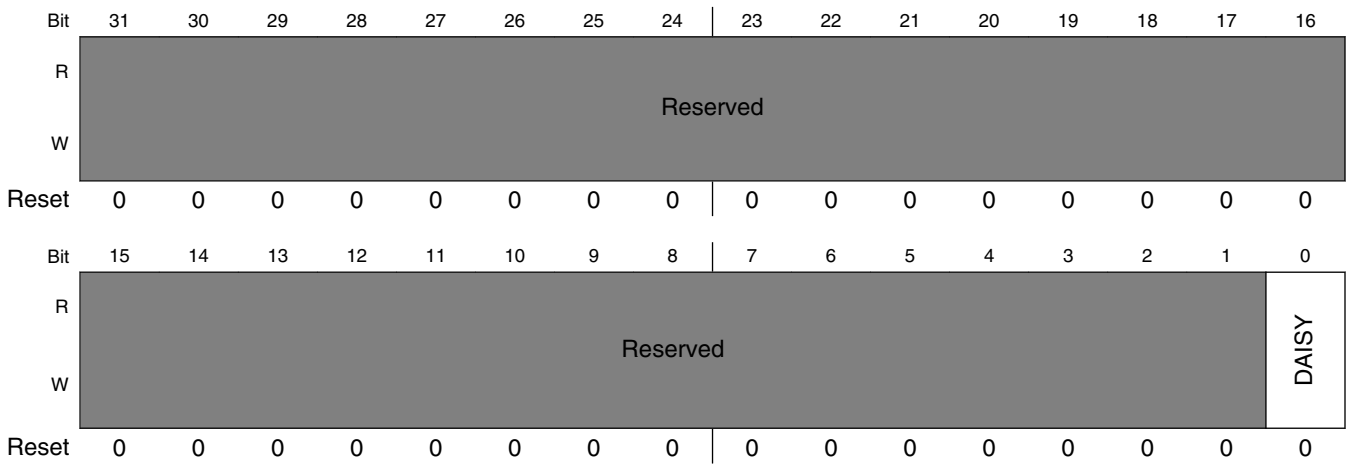
IOMUXC_ECSPI1_SS0_B_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **UART3_CTS_B_ALT3** — Selecting Pad: UART3_CTS_B Mode: ALT3 for ECSPI1_SS0_B 1 **ECSPI1_SS0_ALT0** — Selecting Pad: ECSPI1_SS0 Mode: ALT0 for ECSPI1_SS0_B

8.2.7.329 ECSPI2_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_SCLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 534h offset = 3033_0534h



IOMUXC_ECSPI2_SCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **ECSPI2_SCLK_ALT0** — Selecting Pad: ECSPI2_SCLK Mode: ALT0 for ECSPI2_SCLK 1 **ENET1_RGMII_RD2_ALT2** — Selecting Pad: ENET1_RGMII_RD2 Mode: ALT2 for ECSPI2_SCLK

8.2.7.330 ECSPI2_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_MISO_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 538h offset = 3033_0538h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

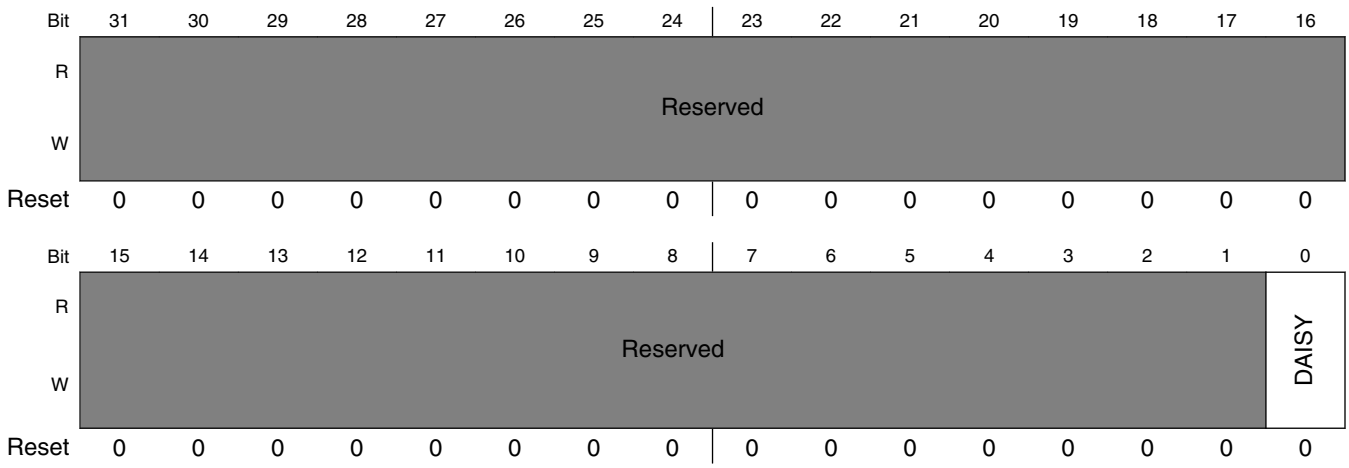
IOMUXC_ECSPI2_MISO_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **ECSPI2_MISO_ALT0** — Selecting Pad: ECSPI2_MISO Mode: ALT0 for ECSPI2_MISO 1 **ENET1_RGMII_TD2_ALT2** — Selecting Pad: ENET1_RGMII_TD2 Mode: ALT2 for ECSPI2_MISO

8.2.7.331 ECSPI2_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_MOSI_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 53Ch offset = 3033_053Ch



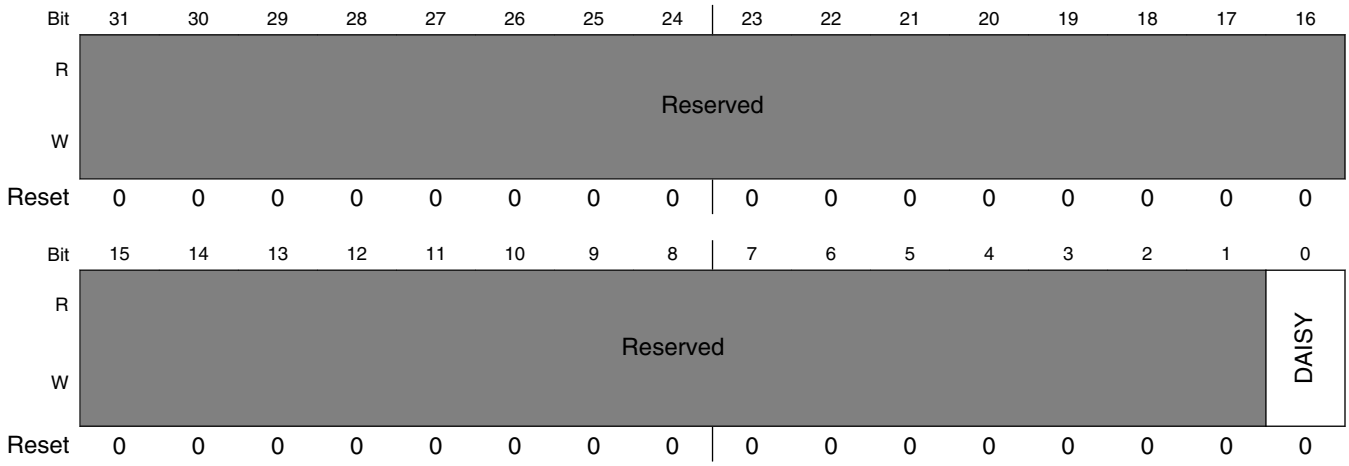
IOMUXC_ECSPI2_MOSI_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **ECSPI2_MOSI_ALT0** — Selecting Pad: ECSPI2_MOSI Mode: ALT0 for ECSPI2_MOSI 1 **ENET1_RGMII_RD3_ALT2** — Selecting Pad: ENET1_RGMII_RD3 Mode: ALT2 for ECSPI2_MOSI

8.2.7.332 ECSPI2_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI2_SS0_B_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 540h offset = 3033_0540h



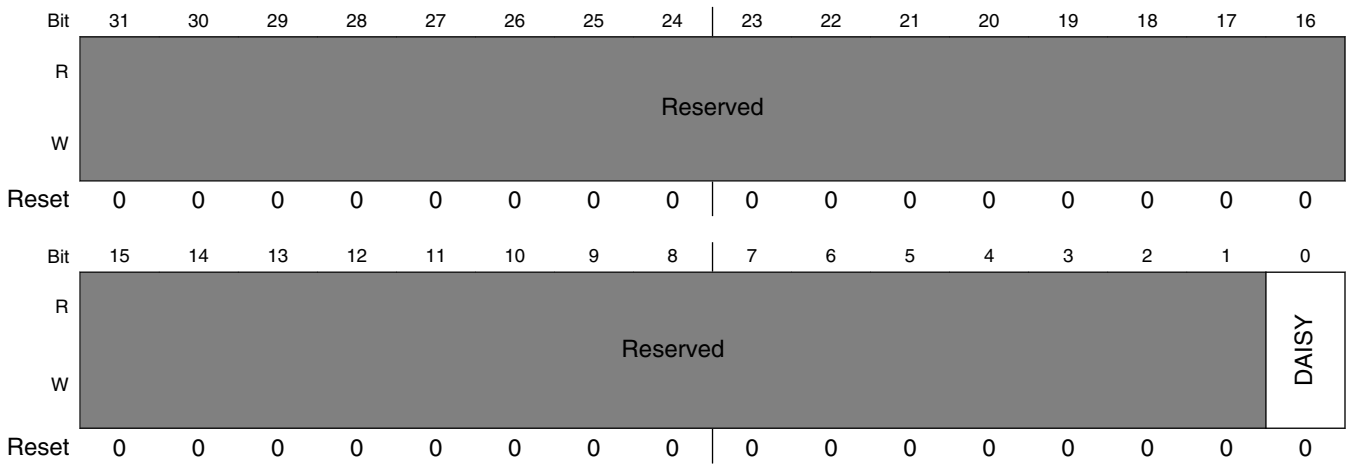
IOMUXC_ECSPI2_SS0_B_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **ECSPI2_SS0_ALT0** — Selecting Pad: ECSPI2_SS0 Mode: ALT0 for ECSPI2_SS0_B 1 **ENET1_RGMII_TD3_ALT2** — Selecting Pad: ENET1_RGMII_TD3 Mode: ALT2 for ECSPI2_SS0_B

8.2.7.333 ECSPI3_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_SCLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 544h offset = 3033_0544h



IOMUXC_ECSPI3_SCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **I2C2_SCL_ALT3** — Selecting Pad: I2C2_SCL Mode: ALT3 for ECSPI3_SCLK 1 **SAI2_RX_DATA_ALT1** — Selecting Pad: SAI2_RX_DATA Mode: ALT1 for ECSPI3_SCLK

8.2.7.334 ECSPI3_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_MISO_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 548h offset = 3033_0548h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

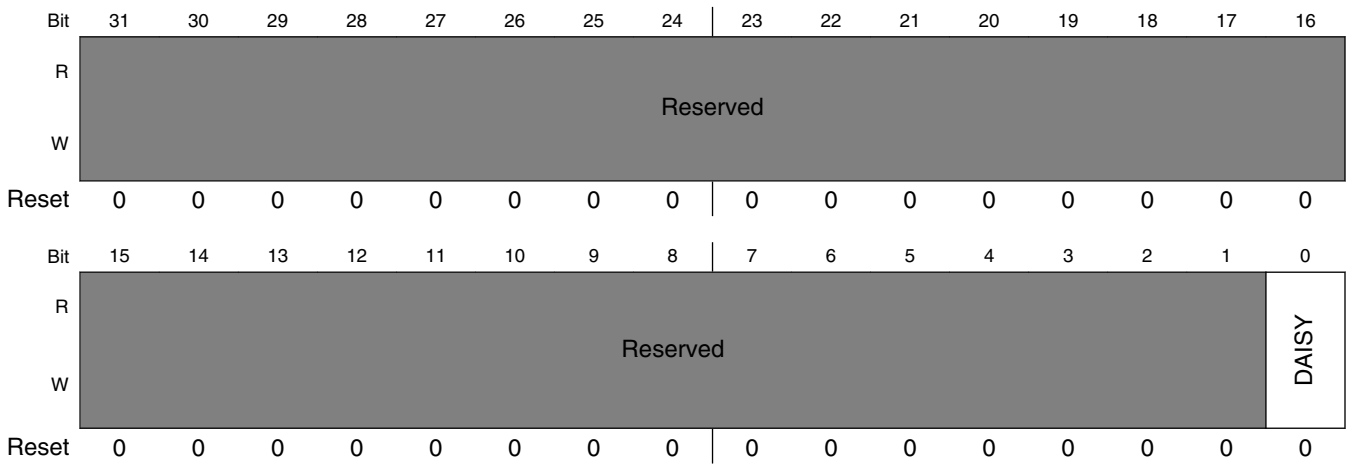
IOMUXC_ECSPI3_MISO_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **I2C1_SCL_ALT3** — Selecting Pad: I2C1_SCL Mode: ALT3 for ECSPI3_MIS 1 **SAI2_TX_SYNC_ALT1** — Selecting Pad: SAI2_TX_SYNC Mode: ALT1 for ECSPI3_MIS

8.2.7.335 ECSPI3_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_MOSI_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 54Ch offset = 3033_054Ch



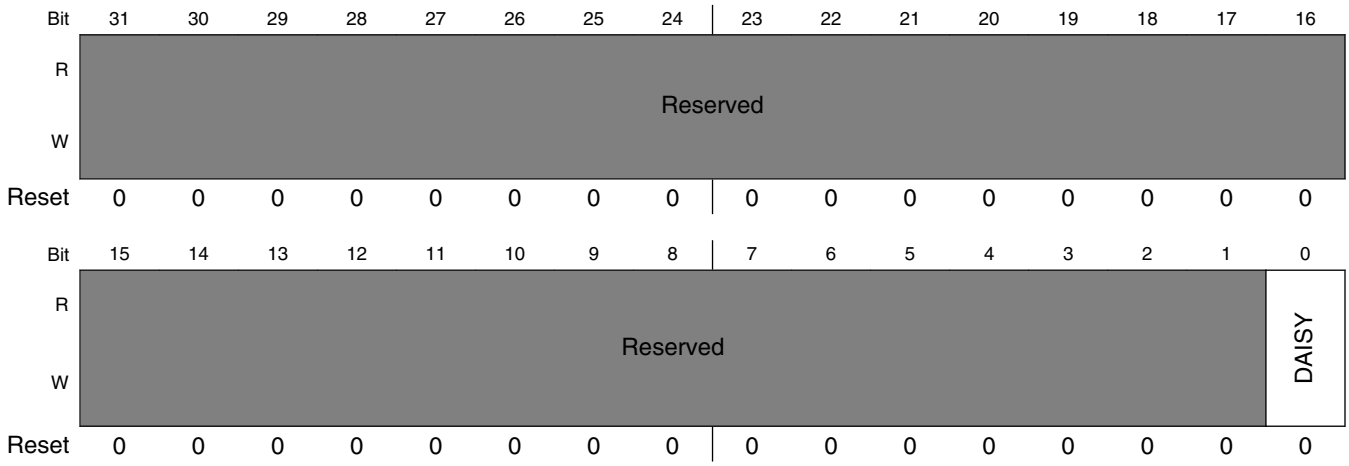
IOMUXC_ECSPI3_MOSI_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **I2C1_SDA_ALT3** — Selecting Pad: I2C1_SDA Mode: ALT3 for ECSPI3_MOSI 1 **SAI2_TX_BCLK_ALT1** — Selecting Pad: SAI2_TX_BCLK Mode: ALT1 for ECSPI3_MOSI

8.2.7.336 ECSPI3_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI3_SS0_B_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 550h offset = 3033_0550h



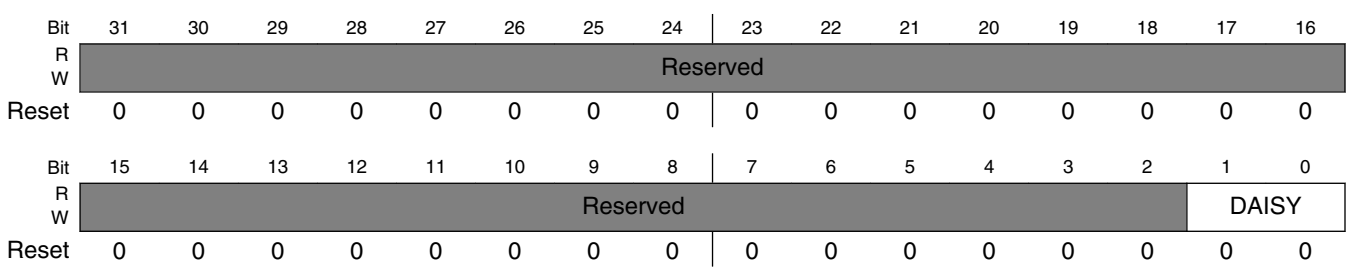
IOMUXC_ECSPI3_SS0_B_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **I2C2_SDA_ALT3** — Selecting Pad: I2C2_SDA Mode: ALT3 for ECSPI3_SS0_B 1 **SAI2_TX_DATA_ALT1** — Selecting Pad: SAI2_TX_DATA Mode: ALT1 for ECSPI3_SS0_B

8.2.7.337 ECSPI4_SCLK_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_SCLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 554h offset = 3033_0554h



IOMUXC_ECSPI4_SCLK_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **LCD_HSYNC_ALT1** — Selecting Pad: LCD_HSYNC Mode: ALT1 for ECSPI4_SCLK 01 **SD1_RESET_B_ALT3** — Selecting Pad: SD1_RESET_B Mode: ALT3 for ECSPI4_SCLK 10 **SD3_DATA1_ALT2** — Selecting Pad: SD3_DATA1 Mode: ALT2 for

8.2.7.338 ECSPI4_MISO_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_MISO_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 558h offset = 3033_0558h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_ECSPI4_MISO_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **LCD_CLK_ALT1** — Selecting Pad: LCD_CLK Mode: ALT1 for ECSPI4_MISO 01 **SD1_CD_B_ALT3** — Selecting Pad: SD1_CD_B Mode: ALT3 for ECSPI4_MISO 10 **SD3_CLK_ALT2** — Selecting Pad: SD3_CLK Mode: ALT2 for ECSPI4_MISO

8.2.7.339 ECSPI4_MOSI_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_MOSI_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 55Ch offset = 3033_055Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_ECSPI4_MOSI_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **LCD_ENABLE_ALT1** — Selecting Pad: LCD_ENABLE Mode: ALT1 for ECSPI4_MOSI 01 **SD1_WP_ALT3** — Selecting Pad: SD1_WP Mode: ALT3 for ECSPI4_MOSI 10 **SD3_CMD_ALT2** — Selecting Pad: SD3_CMD Mode: ALT2 for ECSPI4_MOSI

8.2.7.340 ECSPI4_SS0_B_SELECT_INPUT DAISY Register (IOMUXC_ECSPI4_SS0_B_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 560h offset = 3033_0560h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_ECSPI4_SS0_B_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **LCD_VSYNC_ALT1** — Selecting Pad: LCD_VSYNC Mode: ALT1 for ECSPI4_SS0_B 01 **SD1_CLK_ALT3** — Selecting Pad: SD1_CLK Mode: ALT3 for ECSPI4_SS0_B 10 **SD3_DATA0_ALT2** — Selecting Pad: SD3_DATA0 Mode: ALT2 for ECSPI4_SS0_B

8.2.7.341 CCM_ENET1_REF_CLK_SELECT_INPUT DAISY Register (IOMUXC_CCM_ENET1_REF_CLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 564h offset = 3033_0564h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_CCM_ENET1_REF_CLK_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO12_ALT2** — Selecting Pad: GPIO1_IO12 Mode: ALT2 for CCM_ENET1_REF_CLK 01 **I2C1_SDA_ALT4** — Selecting Pad: I2C1_SDA Mode: ALT4 for CCM_ENET1_REF_CLK 10 **ENET1_TX_CLK_ALT1** — Selecting Pad: ENET1_TX_CLK Mode: ALT1 for CCM_ENET1_REF_CLK 11 **GPIO1_IO02_ALT2** — Selecting Pad: GPIO1_IO02 Mode: ALT2 for CCM_ENET1_REF_CLK

8.2.7.342 ENET1_MDIO_SELECT_INPUT DAISY Register (IOMUXC_ENET1_MDIO_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 568h offset = 3033_0568h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_ENET1_MDIO_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO10_ALT2** — Selecting Pad: GPIO1_IO10 Mode: ALT2 for ENET1_MDIO 01 **UART1_RX_DATA_ALT6** — Selecting Pad: UART1_RX_DATA Mode: ALT6 for ENET1_MDIO 10 **SD2_CD_B_ALT1** — Selecting Pad: SD2_CD_B Mode: ALT1 for ENET1_MDIO

8.2.7.343 ENET1_RX_CLK_SELECT_INPUT DAISY Register (IOMUXC_ENET1_RX_CLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 56Ch offset = 3033_056Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_ENET1_RX_CLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **ENET1_RGMII_RXC_ALT0** — Selecting Pad: ENET1_RGMII_RXC Mode: ALT0 for ENET1_RGMII_RXC 1 **ENET1_RX_CLK_ALT0** — Selecting Pad: ENET1_RX_CLK Mode: ALT0 for ENET1_RX_CLK

8.2.7.344 CCM_ENET2_REF_CLK_SELECT_INPUT DAISY Register (IOMUXC_CCM_ENET2_REF_CLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 570h offset = 3033_0570h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_CCM_ENET2_REF_CLK_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO13_ALT2** — Selecting Pad: GPIO1_IO13 Mode: ALT2 for CCM_ENET2_REF_CLK 01 **EPDC_BDR0_ALT3** — Selecting Pad: EPDC_BDR0 Mode: ALT3 for CCM_ENET2_REF_CLK 10 **I2C2_SCL_ALT4** — Selecting Pad: I2C2_SCL Mode: ALT4 for CCM_ENET2_REF_CLK 11 **GPIO1_IO03_ALT2** — Selecting Pad: GPIO1_IO03 Mode: ALT2 for CCM_ENET2_REF_CLK

8.2.7.345 ENET2_MDIO_SELECT_INPUT DAISY Register (IOMUXC_ENET2_MDIO_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 574h offset = 3033_0574h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_ENET2_MDIO_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO14_ALT2** — Selecting Pad: GPIO1_IO14 Mode: ALT2 for ENET2_MDIO 01 **UART2_RX_DATA_ALT6** — Selecting Pad: UART2_RX_DATA Mode: ALT6 for ENET2_MDIO 10 **SD2_CD_B_ALT2** — Selecting Pad: SD2_CD_B Mode: ALT2 for ENET2_MDIO

8.2.7.346 ENET2_RX_CLK_SELECT_INPUT DAISY Register (IOMUXC_ENET2_RX_CLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 578h offset = 3033_0578h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

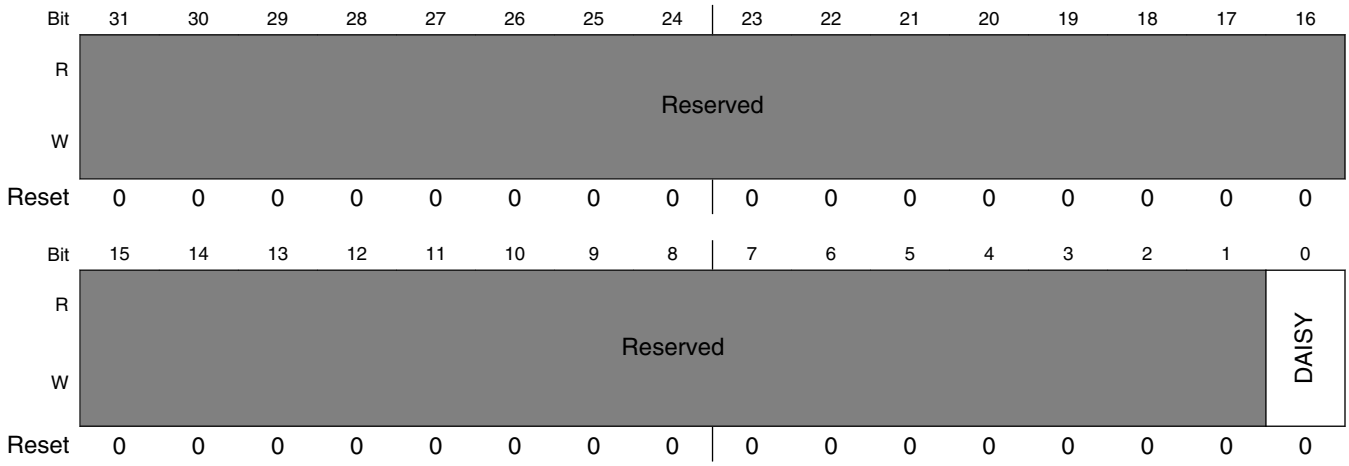
IOMUXC_ENET2_RX_CLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_SDCE1_ALT2** — Selecting Pad: EPDC_SDCE1 Mode: ALT2 for ENET2_RX_CLK 1 **EPDC_BDR1_ALT2** — Selecting Pad: EPDC_BDR1 Mode: ALT2 for ENET2_RX_CLK

8.2.7.347 EPDC_PWR_IRQ_SELECT_INPUT DAISY Register (IOMUXC_EPDC_PWR_IRQ_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 57Ch offset = 3033_057Ch



IOMUXC_EPDC_PWR_IRQ_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **ECSPI1_MISO_ALT6** — Selecting Pad: ECSPI1_MISO Mode: ALT6 for EPDC_PWR_IRQ 1 **ENET1_TX_CLK_ALT4** — Selecting Pad: ENET1_TX_CLK Mode: ALT4 for EPDC_PWR_IRQ

8.2.7.348 EPDC_PWR_STAT_SELECT_INPUT DAISY Register (IOMUXC_EPDC_PWR_STAT_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 580h offset = 3033_0580h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

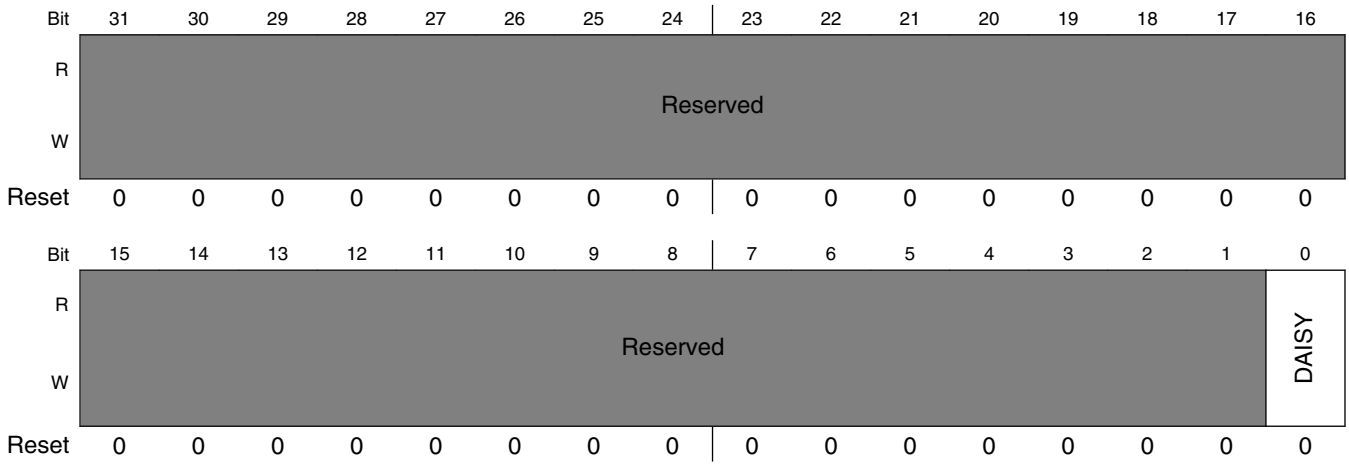
IOMUXC_EPDC_PWR_STAT_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_PWR_STAT_ALT0** — Selecting Pad: EPDC_PWR_STAT Mode: ALT0 for EPDC_PWR_STAT 1 **ECSPI1_MOSI_ALT6** — Selecting Pad: ECSPI1_MOSI Mode: ALT6 for EPDC_PWR_STAT

8.2.7.349 FLEXTIMER1_CH0_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH0_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 584h offset = 3033_0584h



IOMUXC_FLEXTIMER1_CH0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_SDOE_ALT1** — Selecting Pad: EPDC_SDOE Mode: ALT1 for FLEXTIMER1_CH0 1 **SD1_CD_B_ALT4** — Selecting Pad: SD1_CD_B Mode: ALT4 for FLEXTIMER1_CH0

8.2.7.350 FLEXTIMER1_CH1_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH1_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 588h offset = 3033_0588h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

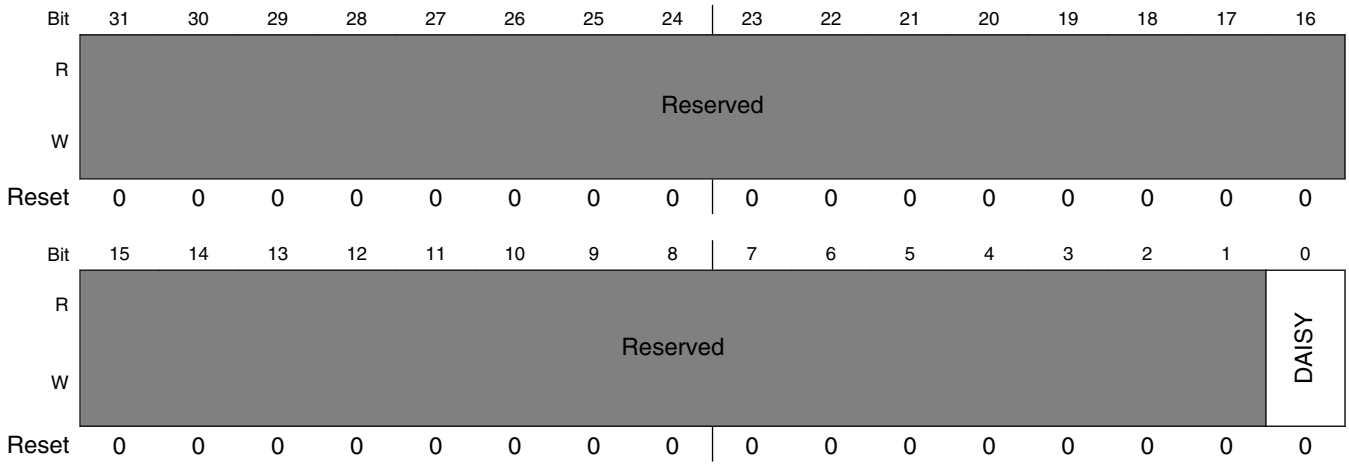
IOMUXC_FLEXTIMER1_CH1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_SDSHR_ALT1** — Selecting Pad: EPDC_SDSHR Mode: ALT1 for FLEXTIMER1_CH1 1 **SD1_WP_ALT4** — Selecting Pad: SD1_WP Mode: ALT4 for FLEXTIMER1_CH1

8.2.7.351 FLEXTIMER1_CH2_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH2_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 58Ch offset = 3033_058Ch



IOMUXC_FLEXTIMER1_CH2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_SDCE0_ALT1** — Selecting Pad: EPDC_SDCE0 Mode: ALT1 for FLEXTIMER1_CH2 1 **SD1_RESET_B_ALT4** — Selecting Pad: SD1_RESET_B Mode: ALT4 for FLEXTIMER1_CH2

8.2.7.352 FLEXTIMER1_CH3_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH3_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 590h offset = 3033_0590h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

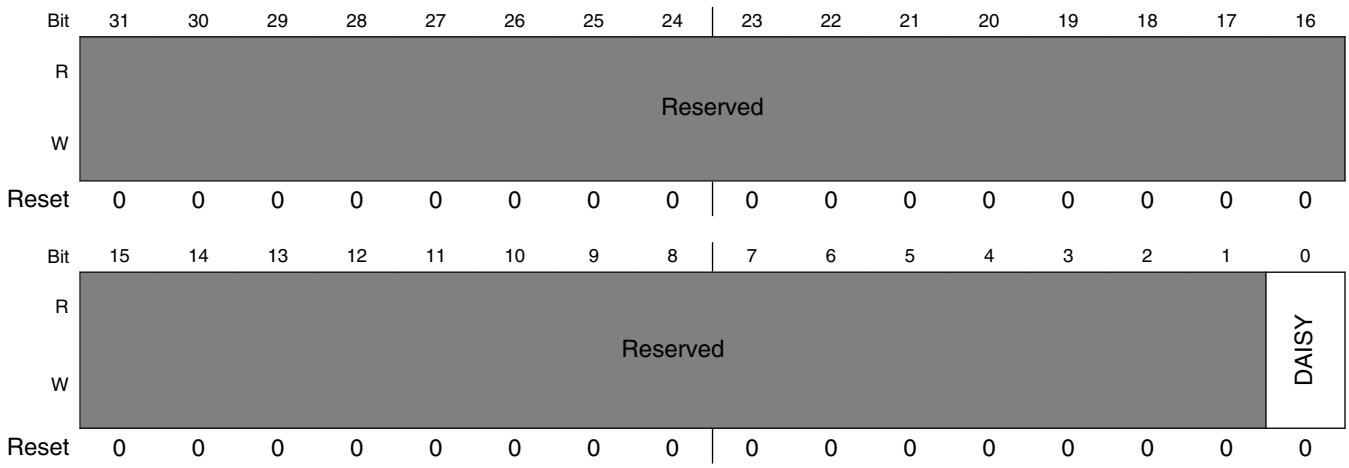
IOMUXC_FLEXTIMER1_CH3_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_SDCE1_ALT1** — Selecting Pad: EPDC_SDCE1 Mode: ALT1 for FLEXTIMER1_CH3 1 **SD1_CLK_ALT4** — Selecting Pad: SD1_CLK Mode: ALT4 for FLEXTIMER1_CH3

8.2.7.353 FLEXTIMER1_CH4_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH4_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 594h offset = 3033_0594h



IOMUXC_FLEXTIMER1_CH4_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA16_ALT1** — Selecting Pad: LCD_DATA16 Mode: ALT1 for FLEXTIMER1_CH4 1 **GPIO1_IO04_ALT2** — Selecting Pad: GPIO1_IO04 Mode: ALT2 for FLEXTIMER1_CH4

8.2.7.354 FLEXTIMER1_CH5_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH5_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 598h offset = 3033_0598h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

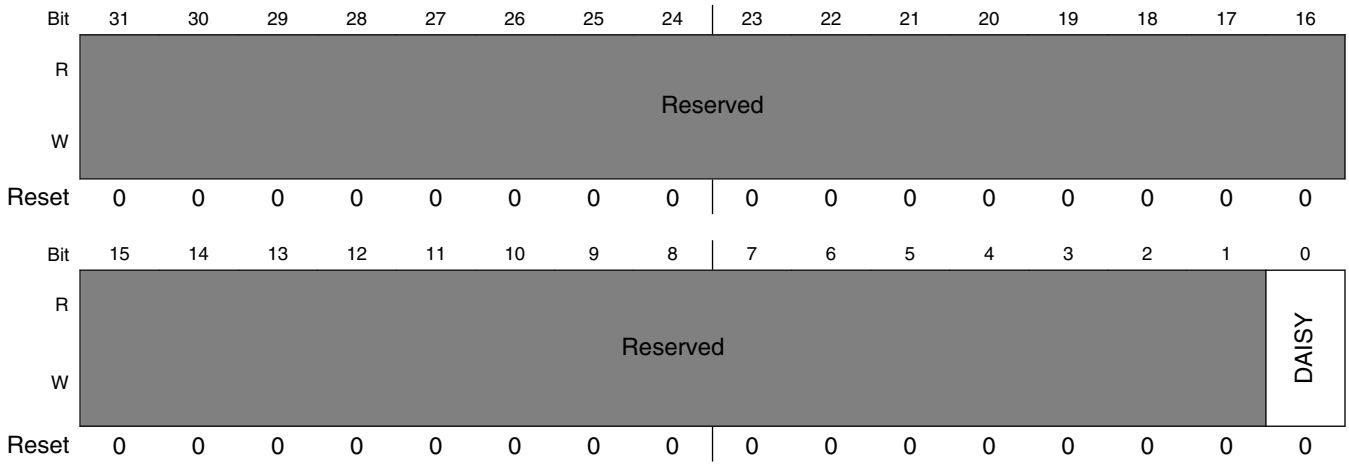
IOMUXC_FLEXTIMER1_CH5_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA17_ALT1** — Selecting Pad: LCD_DATA17 Mode: ALT1 for FLEXTIMER1_CH5 1 **GPIO1_IO05_ALT2** — Selecting Pad: GPIO1_IO05 Mode: ALT2 for FLEXTIMER1_CH5

8.2.7.355 FLEXTIMER1_CH6_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH6_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 59Ch offset = 3033_059Ch



IOMUXC_FLEXTIMER1_CH6_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA18_ALT1** — Selecting Pad: LCD_DATA18 Mode: ALT1 for FLEXTIMER1_CH6 1 **GPIO1_IO06_ALT2** — Selecting Pad: GPIO1_IO06 Mode: ALT2 for FLEXTIMER1_CH6

8.2.7.356 FLEXTIMER1_CH7_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_CH7_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5A0h offset = 3033_05A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

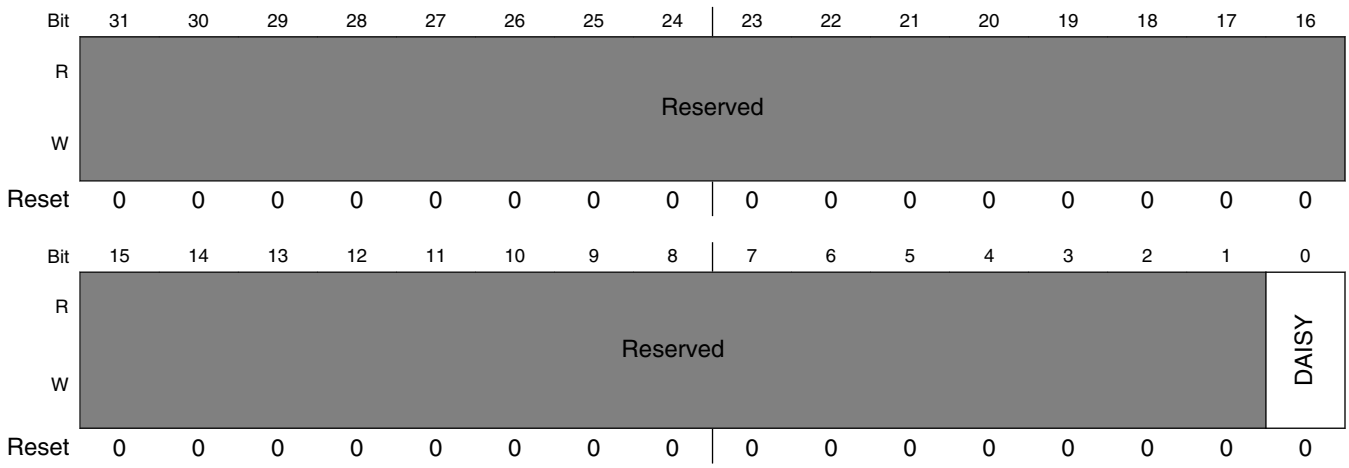
IOMUXC_FLEXTIMER1_CH7_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA19_ALT1** — Selecting Pad: LCD_DATA19 Mode: ALT1 for FLEXTIMER1_CH7 1 **GPIO1_IO07_ALT2** — Selecting Pad: GPIO1_IO07 Mode: ALT2 for FLEXTIMER1_CH7

8.2.7.357 FLEXTIMER1_PHA_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_PHA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5A4h offset = 3033_05A4h



IOMUXC_FLEXTIMER1_PHA_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **GPIO1_IO10_ALT5** — Selecting Pad: GPIO1_IO10 Mode: ALT5 for FLEXTIMER1_PHA 1 **SD1_DATA3_ALT4** — Selecting Pad: SD1_DATA3 Mode: ALT4 for FLEXTIMER1_PHA

8.2.7.358 FLEXTIMER1_PHB_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER1_PHB_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5A8h offset = 3033_05A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

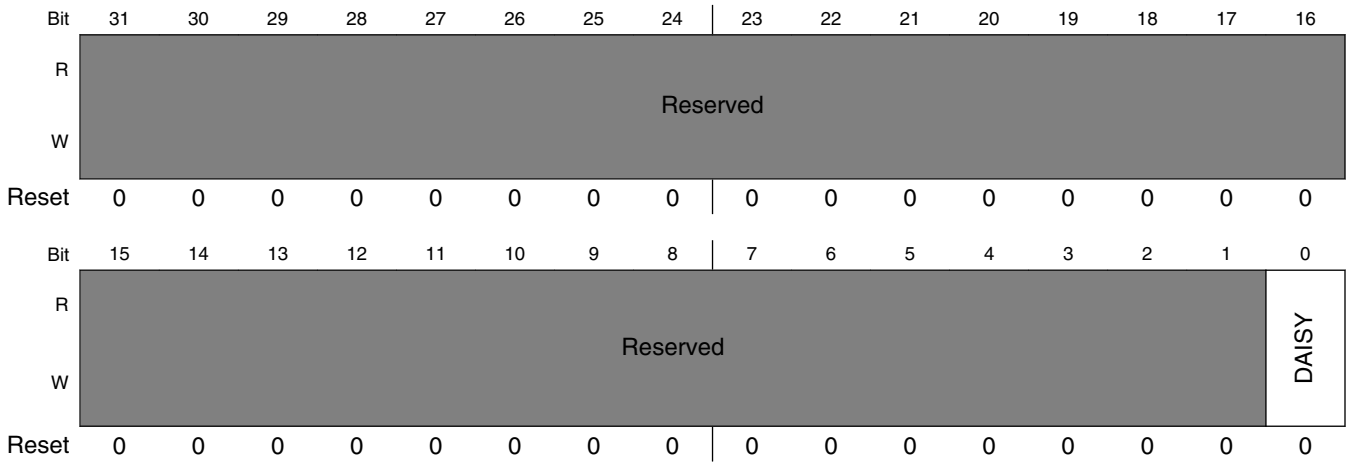
IOMUXC_FLEXTIMER1_PHB_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **GPIO1_IO11_ALT5** — Selecting Pad: GPIO1_IO11 Mode: ALT5 for FLEXTIMER1_PHB 1 **SD2_CD_B_ALT4** — Selecting Pad: SD2_CD_B Mode: ALT4 for FLEXTIMER1_PHB

8.2.7.359 FLEXTIMER2_CH0_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH0_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5ACh offset = 3033_05ACh



IOMUXC_FLEXTIMER2_CH0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_GDCLK_ALT1** — Selecting Pad: EPDC_GDCLK Mode: ALT1 for FLEXTIMER2_CH0 1 **SD1_CMD_ALT4** — Selecting Pad: SD1_CMD Mode: ALT4 for FLEXTIMER2_CH0

8.2.7.360 FLEXTIMER2_CH1_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH1_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5B0h offset = 3033_05B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

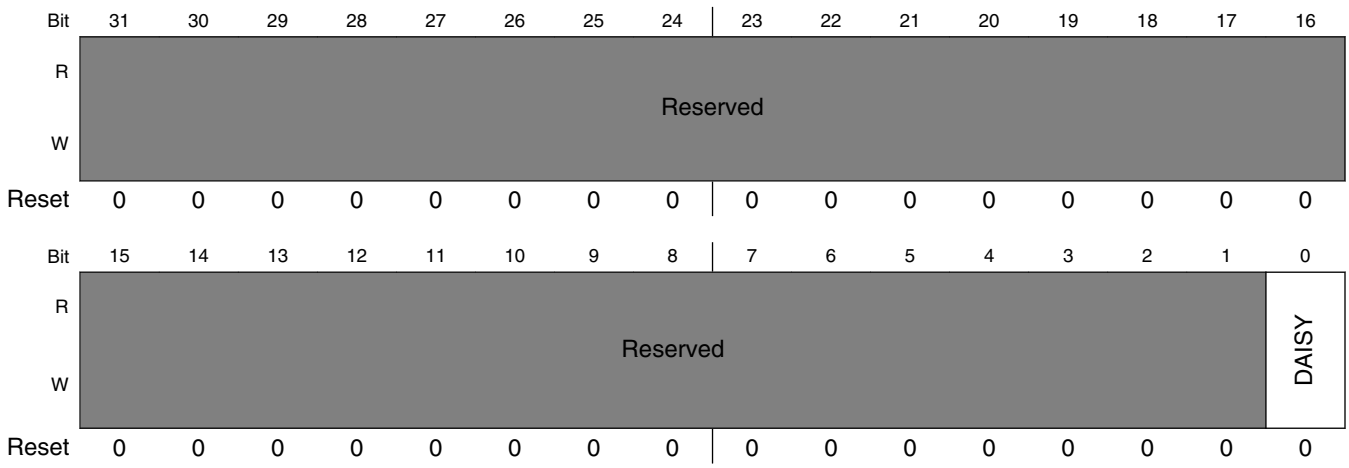
IOMUXC_FLEXTIMER2_CH1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_GDOE_ALT1** — Selecting Pad: EPDC_GDOE Mode: ALT1 for FLEXTIMER2_CH1 1 **SD1_DATA0_ALT4** — Selecting Pad: SD1_DATA0 Mode: ALT4 for FLEXTIMER2_CH1

8.2.7.361 FLEXTIMER2_CH2_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH2_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5B4h offset = 3033_05B4h



IOMUXC_FLEXTIMER2_CH2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_GDRL_ALT1** — Selecting Pad: EPDC_GDRL Mode: ALT1 for FLEXTIMER2_CH2 1 **SD1_DATA1_ALT4** — Selecting Pad: SD1_DATA1 Mode: ALT4 for FLEXTIMER2_CH2

8.2.7.362 FLEXTIMER2_CH3_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH3_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5B8h offset = 3033_05B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

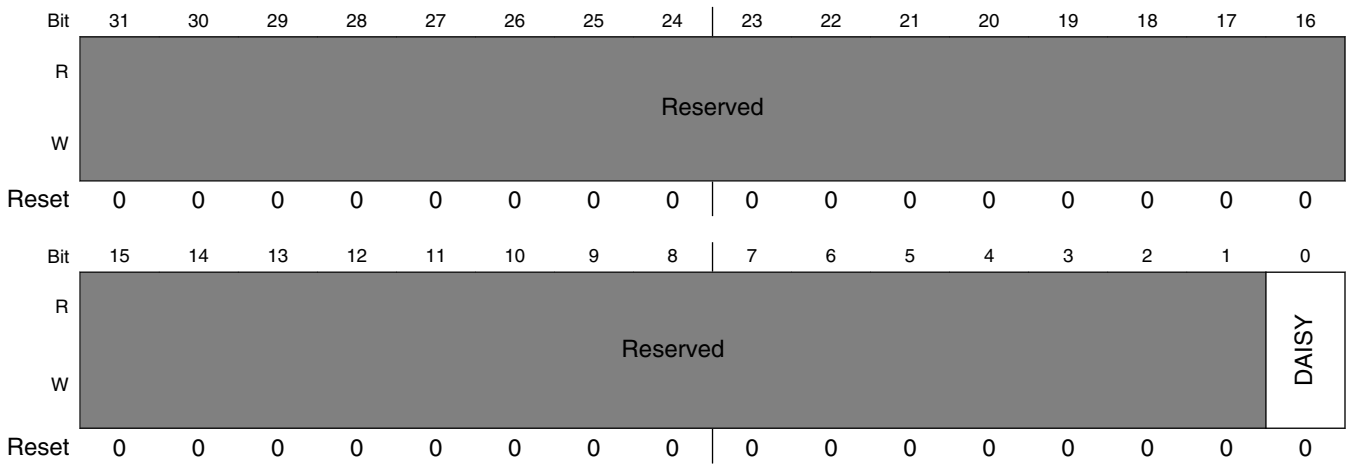
IOMUXC_FLEXTIMER2_CH3_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_GDSP_ALT1** — Selecting Pad: EPDC_GDSP Mode: ALT1 for FLEXTIMER2_CH3 1 **SD1_DATA2_ALT4** — Selecting Pad: SD1_DATA2 Mode: ALT4 for FLEXTIMER2_CH3

8.2.7.363 FLEXTIMER2_CH4_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH4_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5BCh offset = 3033_05BCh



IOMUXC_FLEXTIMER2_CH4_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA20_ALT1** — Selecting Pad: LCD_DATA20 Mode: ALT1 for FLEXTIMER2_CH4 1 **SAI2_TX_SYNC_ALT4** — Selecting Pad: SAI2_TX_SYNC Mode: ALT4 for FLEXTIMER2_CH4

8.2.7.364 FLEXTIMER2_CH5_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH5_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5C0h offset = 3033_05C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

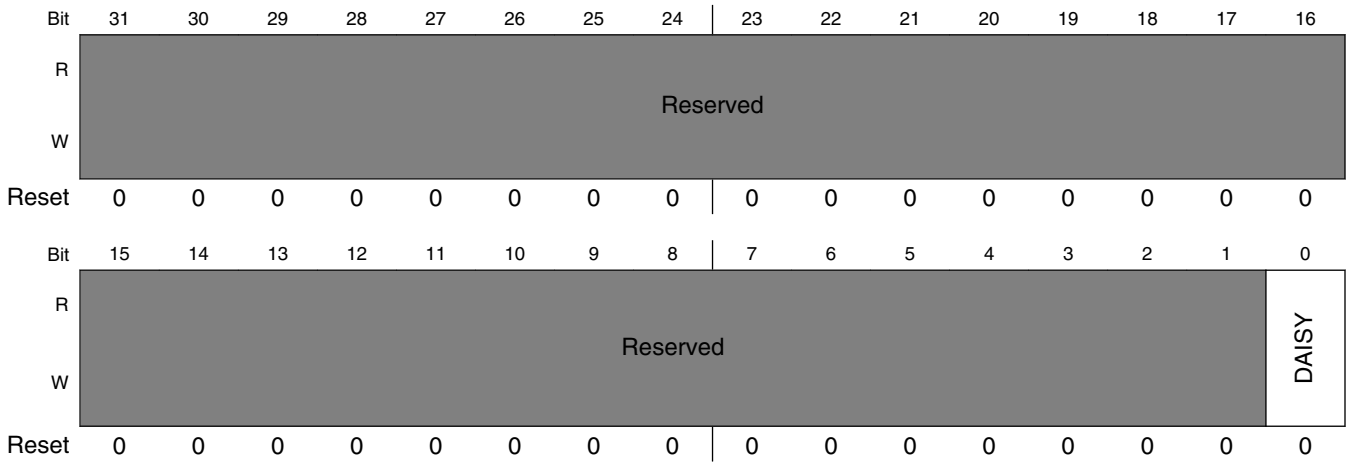
IOMUXC_FLEXTIMER2_CH5_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA21_ALT1** — Selecting Pad: LCD_DATA21 Mode: ALT1 for FLEXTIMER2_CH5 1 **SAI2_TX_BCLK_ALT4** — Selecting Pad: SAI2_TX_BCLK Mode: ALT4 for FLEXTIMER2_CH5

8.2.7.365 FLEXTIMER2_CH6_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH6_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5C4h offset = 3033_05C4h



IOMUXC_FLEXTIMER2_CH6_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA22_ALT1** — Selecting Pad: LCD_DATA22 Mode: ALT1 for FLEXTIMER2_CH6 1 **SAI2_RX_DATA_ALT4** — Selecting Pad: SAI2_RX_DATA Mode: ALT4 for FLEXTIMER2_CH6

8.2.7.366 FLEXTIMER2_CH7_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_CH7_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5C8h offset = 3033_05C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

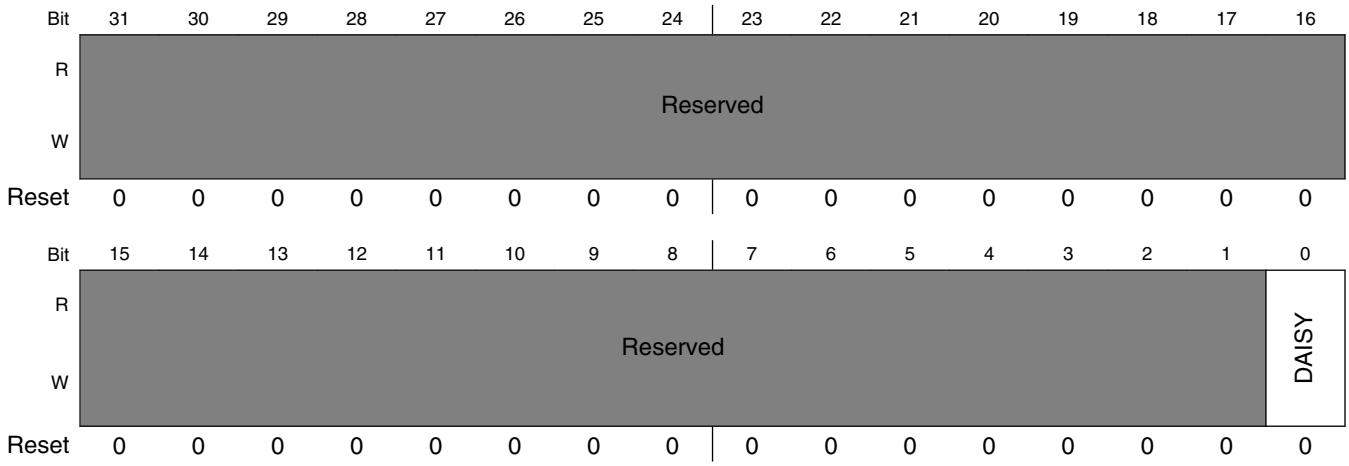
IOMUXC_FLEXTIMER2_CH7_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **LCD_DATA23_ALT1** — Selecting Pad: LCD_DATA23 Mode: ALT1 for FLEXTIMER2_CH7 1 **SAI2_TX_DATA_ALT4** — Selecting Pad: SAI2_TX_DATA Mode: ALT4 for FLEXTIMER2_CH7

8.2.7.367 FLEXTIMER2_PHA_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_PHA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5CCh offset = 3033_05CCh



IOMUXC_FLEXTIMER2_PHA_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_PWR_COM_ALT1** — Selecting Pad: EPDC_PWR_COM Mode: ALT1 for FLEXTIMER2_PHA 1 **SAI1_RX_BCLK_ALT4** — Selecting Pad: SAI1_RX_BCLK Mode: ALT4 for FLEXTIMER2_PHA

8.2.7.368 FLEXTIMER2_PHB_SELECT_INPUT DAISY Register (IOMUXC_FLEXTIMER2_PHB_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5D0h offset = 3033_05D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_FLEXTIMER2_PHB_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_PWR_STAT_ALT1** — Selecting Pad: EPDC_PWR_STAT Mode: ALT1 for FLEXTIMER2_PHB 1 **SAI1_MCLK_ALT4** — Selecting Pad: SAI1_MCLK Mode: ALT4 for FLEXTIMER2_PHB

8.2.7.369 I2C1_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C1_SCL_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5D4h offset = 3033_05D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_I2C1_SCL_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **UART1_RX_DATA_ALT1** — Selecting Pad: UART1_RX_DATA Mode: ALT1 for I2C1_SCL 01 **I2C1_SCL_ALT0** — Selecting Pad: I2C1_SCL Mode: ALT0 for I2C1_SCL 10 **GPIO1_IO04_ALT4** — Selecting Pad: GPIO1_IO04 Mode: ALT4 for I2C1_SCL

8.2.7.370 I2C1_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C1_SDA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5D8h offset = 3033_05D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_I2C1_SDA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **UART1_TX_DATA_ALT1** — Selecting Pad: UART1_TX_DATA Mode: ALT1 for I2C1_SDA 01 **I2C1_SDA_ALT0** — Selecting Pad: I2C1_SDA Mode: ALT0 for I2C1_SDA 10 **GPIO1_IO05_ALT4** — Selecting Pad: GPIO1_IO05 Mode: ALT4 for I2C1_SDA

8.2.7.371 I2C2_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C2_SCL_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5DCh offset = 3033_05DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_I2C2_SCL_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **UART2_RX_DATA_ALT1** — Selecting Pad: UART2_RX_DATA Mode: ALT1 for I2C2_SCL 01 **I2C2_SCL_ALT0** — Selecting Pad: I2C2_SCL Mode: ALT0 for I2C2_SCL 10 **GPIO1_IO06_ALT4** — Selecting Pad: GPIO1_IO06 Mode: ALT4 for I2C2_SCL

8.2.7.372 I2C2_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C2_SDA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5E0h offset = 3033_05E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_I2C2_SDA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **UART2_TX_DATA_ALT1** — Selecting Pad: UART2_TX_DATA Mode: ALT1 for I2C2_SDA 01 **I2C2_SDA_ALT0** — Selecting Pad: I2C2_SDA Mode: ALT0 for I2C2_SDA 10 **GPIO1_IO07_ALT4** — Selecting Pad: GPIO1_IO07 Mode: ALT4 for I2C2_SDA

8.2.7.373 I2C3_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C3_SCL_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5E4h offset = 3033_05E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																													DAISY		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_I2C3_SCL_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 000 **GPIO1_IO08_ALT4** — Selecting Pad: GPIO1_IO08 Mode: ALT4 for I2C3_SCL 001 **LCD_DATA20_ALT6** — Selecting Pad: LCD_DATA20 Mode: ALT6 for I2C3_SCL 010 **I2C3_SCL_ALT0** — Selecting Pad: I2C3_SCL Mode: ALT0 for I2C3_SCL 011 **SD3_DATA3_ALT2** — Selecting Pad: SD3_DATA3 Mode: ALT2 for I2C3_SCL 100 **ENET1_RGMII_RD0_ALT2** — Selecting Pad: ENET1_RGMII_RD0 Mode: ALT2 for I2C3_SCL

8.2.7.374 I2C3_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C3_SDA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5E8h offset = 3033_05E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																													DAISY		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_I2C3_SDA_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 000 **GPIO1_IO09_ALT4** — Selecting Pad: GPIO1_IO09 Mode: ALT4 for I2C3_SDA 001 **LCD_DATA21_ALT6** — Selecting Pad: LCD_DATA21 Mode: ALT6 for I2C3_SDA 010 **I2C3_SDA_ALT0** — Selecting Pad: I2C3_SDA Mode: ALT0 for I2C3_SDA 011 **SD3_DATA2_ALT2** — Selecting Pad: SD3_DATA2 Mode: ALT2 for I2C3_SDA 100 **ENET1_RGMII_RD1_ALT2** — Selecting Pad: ENET1_RGMII_RD1 Mode: ALT2 for I2C3_SDA

8.2.7.375 I2C4_SCL_SELECT_INPUT DAISY Register (IOMUXC_I2C4_SCL_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5ECh offset = 3033_05ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																													DAISY		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_I2C4_SCL_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 000 **GPIO1_IO10_ALT4** — Selecting Pad: GPIO1_IO10 Mode: ALT4 for I2C4_SCL

Table continues on the next page...

IOMUXC_I2C4_SCL_SELECT_INPUT field descriptions (continued)

Field	Description
	001 **LCD_DATA22_ALT6** — Selecting Pad: LCD_DATA22 Mode: ALT6 for I2C4_SCL 010 **I2C4_SCL_ALT0** — Selecting Pad: I2C4_SCL Mode: ALT0 for I2C4_SCL 011 **SAI1_RX_SYNC_ALT3** — Selecting Pad: SAI1_RX_SYNC Mode: ALT3 for I2C4_SCL 100 **ENET1_RGMII_TD2_ALT3** — Selecting Pad: ENET1_RGMII_TD2 Mode: ALT3 for I2C4_SCL

8.2.7.376 I2C4_SDA_SELECT_INPUT DAISY Register (IOMUXC_I2C4_SDA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5F0h offset = 3033_05F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																													DAISY		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_I2C4_SDA_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 000 **GPIO1_IO11_ALT4** — Selecting Pad: GPIO1_IO11 Mode: ALT4 for I2C4_SDA 001 **LCD_DATA23_ALT6** — Selecting Pad: LCD_DATA23 Mode: ALT6 for I2C4_SDA 010 **I2C4_SDA_ALT0** — Selecting Pad: I2C4_SDA Mode: ALT0 for I2C4_SDA 011 **SAI1_RX_BCLK_ALT3** — Selecting Pad: SAI1_RX_BCLK Mode: ALT3 for I2C4_SDA 100 **ENET1_RGMII_TD3_ALT3** — Selecting Pad: ENET1_RGMII_TD3 Mode: ALT3 for I2C4_SDA

8.2.7.377 KPP_COL0_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL0_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5F4h offset = 3033_05F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_KPP_COL0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_DATA07_ALT3** — Selecting Pad: EPDC_DATA07 Mode: ALT3 for KPP_COL0 1 **ENET1_RGMII_TD1_ALT6** — Selecting Pad: ENET1_RGMII_TD1 Mode: ALT6 for KPP_COL0

8.2.7.378 KPP_COL1_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL1_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5F8h offset = 3033_05F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_KPP_COL1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_DATA05_ALT3** — Selecting Pad: EPDC_DATA05 Mode: ALT3 for KPP_COL1 1 **ENET1_RGMII_RXC_ALT6** — Selecting Pad: ENET1_RGMII_RXC Mode: ALT6 for KPP_COL1

8.2.7.379 KPP_COL2_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL2_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 5FCh offset = 3033_05FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_KPP_COL2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_DATA03_ALT3** — Selecting Pad: EPDC_DATA03 Mode: ALT3 for KPP_COL2 1 **ENET1_RGMII_RD3_ALT6** — Selecting Pad: ENET1_RGMII_RD3 Mode: ALT6 for KPP_COL2

8.2.7.380 KPP_COL3_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL3_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 600h offset = 3033_0600h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_KPP_COL3_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_DATA01_ALT3** — Selecting Pad: EPDC_DATA01 Mode: ALT3 for KPP_COL3 1 **ENET1_RGMII_RD1_ALT6** — Selecting Pad: ENET1_RGMII_RD1 Mode: ALT6 for KPP_COL3

8.2.7.381 KPP_COL4_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL4_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 604h offset = 3033_0604h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_KPP_COL4_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_SDLE_ALT3** — Selecting Pad: EPDC_SDLE Mode: ALT3 for KPP_COL4 1 **GPIO1_IO07_ALT6** — Selecting Pad: GPIO1_IO07 Mode: ALT6 for KPP_COL4

8.2.7.382 KPP_COL5_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL5_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 608h offset = 3033_0608h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_KPP_COL5_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **GPIO1_IO08_ALT6** — Selecting Pad: GPIO1_IO08 Mode: ALT6 for KPP_COL5 1 **EPDC_SDOE_ALT3** — Selecting Pad: EPDC_SDOE Mode: ALT3 for KPP_COL5

8.2.7.383 KPP_COL6_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL6_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 60Ch offset = 3033_060Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_KPP_COL6_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **GPIO1_IO10_ALT6** — Selecting Pad: GPIO1_IO10 Mode: ALT6 for KPP_COL6 1 **EPDC_SDCE2_ALT3** — Selecting Pad: EPDC_SDCE2 Mode: ALT3 for KPP_COL6

8.2.7.384 KPP_COL7_SELECT_INPUT DAISY Register (IOMUXC_KPP_COL7_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 610h offset = 3033_0610h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_KPP_COL7_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_GDCLK_ALT3** — Selecting Pad: EPDC_GDCLK Mode: ALT3 for KPP_COL7 1 **SAI2_RX_DATA_ALT6** — Selecting Pad: SAI2_RX_DATA Mode: ALT6 for KPP_COL7

8.2.7.385 KPP_ROW0_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW0_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 614h offset = 3033_0614h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

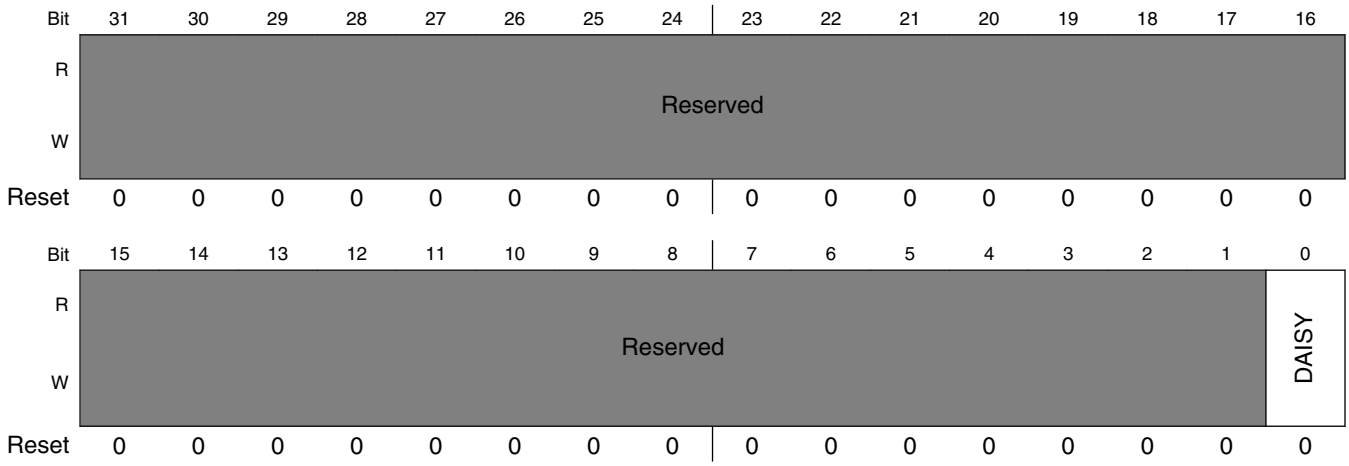
IOMUXC_KPP_ROW0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_DATA06_ALT3** — Selecting Pad: EPDC_DATA06 Mode: ALT3 for KPP_ROW0 1 **ENET1_RGMII_TD0_ALT6** — Selecting Pad: ENET1_RGMII_TD0 Mode: ALT6 for KPP_ROW0

8.2.7.386 KPP_ROW1_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW1_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 618h offset = 3033_0618h



IOMUXC_KPP_ROW1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_DATA04_ALT3** — Selecting Pad: EPDC_DATA04 Mode: ALT3 for KPP_ROW1 1 **ENET1_RGMII_RX_CTL_ALT6** — Selecting Pad: ENET1_RGMII_RX_CTL Mode: ALT6 for KPP_ROW1

8.2.7.387 KPP_ROW2_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW2_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 61Ch offset = 3033_061Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_KPP_ROW2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_DATA02_ALT3** — Selecting Pad: EPDC_DATA02 Mode: ALT3 for KPP_ROW2 1 **ENET1_RGMII_RD2_ALT6** — Selecting Pad: ENET1_RGMII_RD2 Mode: ALT6 for KPP_ROW2

8.2.7.388 KPP_ROW3_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW3_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 620h offset = 3033_0620h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_KPP_ROW3_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_DATA00_ALT3** — Selecting Pad: EPDC_DATA00 Mode: ALT3 for KPP_ROW3 1 **ENET1_RGMII_RD0_ALT6** — Selecting Pad: ENET1_RGMII_RD0 Mode: ALT6 for KPP_ROW3

8.2.7.389 KPP_ROW4_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW4_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 624h offset = 3033_0624h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_KPP_ROW4_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_SDCLK_ALT3** — Selecting Pad: EPDC_SDCLK Mode: ALT3 for KPP_ROW4 1 **GPIO1_IO06_ALT6** — Selecting Pad: GPIO1_IO06 Mode: ALT6 for KPP_ROW4

8.2.7.390 KPP_ROW5_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW5_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 628h offset = 3033_0628h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_KPP_ROW5_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **GPIO1_IO09_ALT6** — Selecting Pad: GPIO1_IO09 Mode: ALT6 for KPP_ROW5 1 **EPDC_SDSHR_ALT3** — Selecting Pad: EPDC_SDSHR Mode: ALT3 for KPP_ROW5

8.2.7.391 KPP_ROW6_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW6_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 62Ch offset = 3033_062Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_KPP_ROW6_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **GPIO1_IO11_ALT6** — Selecting Pad: GPIO1_IO11 Mode: ALT6 for KPP_ROW6 1 **EPDC_SDCE3_ALT3** — Selecting Pad: EPDC_SDCE3 Mode: ALT3 for KPP_ROW6

8.2.7.392 KPP_ROW7_SELECT_INPUT DAISY Register (IOMUXC_KPP_ROW7_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 630h offset = 3033_0630h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_KPP_ROW7_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_GDOE_ALT3** — Selecting Pad: EPDC_GDOE Mode: ALT3 for KPP_ROW7 1 **SAI2_TX_DATA_ALT6** — Selecting Pad: SAI2_TX_DATA Mode: ALT6 for KPP_ROW7

8.2.7.393 LCD_BUSY_SELECT_INPUT DAISY Register (IOMUXC_LCD_BUSY_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 634h offset = 3033_0634h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_BUSY_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_DATA08_ALT7** — Selecting Pad: EPDC_DATA08 Mode: ALT7 for LCD_BUSY 1 **EPDC_GDSP_ALT6** — Selecting Pad: EPDC_GDSP Mode: ALT6 for LCD_BUSY

8.2.7.394 LCD_DATA00_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA00_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 638h offset = 3033_0638h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA00_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA00_ALT6** — Selecting Pad: EPDC_DATA00 Mode: ALT6 for LCD_DATA00 01 **EPDC_DATA09_ALT7** — Selecting Pad: EPDC_DATA09 Mode: ALT7 for LCD_DATA 10 **LCD_DATA00_ALT0** — Selecting Pad: LCD_DATA00 Mode: ALT0 for LCD_DATA00

8.2.7.395 LCD_DATA01_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA01_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 63Ch offset = 3033_063Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA01_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA01_ALT6** — Selecting Pad: EPDC_DATA01 Mode: ALT6 for LCD_DATA01 01 **EPDC_DATA11_ALT7** — Selecting Pad: EPDC_DATA11 Mode: ALT7 for LCD_DATA01 10 **LCD_DATA01_ALT0** — Selecting Pad: LCD_DATA01 Mode: ALT0 for LCD_DATA01

8.2.7.396 LCD_DATA02_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA02_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 640h offset = 3033_0640h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA02_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA02_ALT6** — Selecting Pad: EPDC_DATA02 Mode: ALT6 for LCD_DATA02 01 **EPDC_SDCE3_ALT7** — Selecting Pad: EPDC_SDCE3 Mode: ALT7 for LCD_DATA02 10 **LCD_DATA02_ALT0** — Selecting Pad: LCD_DATA02 Mode: ALT0 for LCD_DATA02

8.2.7.397 LCD_DATA03_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA03_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 644h offset = 3033_0644h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA03_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA03_ALT6** — Selecting Pad: EPDC_DATA03 Mode: ALT6 for LCD_DATA03 01 **EPDC_SDCE2_ALT7** — Selecting Pad: EPDC_SDCE2 Mode: ALT7 for LCD_DATA03 10 **LCD_DATA03_ALT0** — Selecting Pad: LCD_DATA03 Mode: ALT0 for LCD_DATA03

8.2.7.398 LCD_DATA04_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA04_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 648h offset = 3033_0648h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA04_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA04_ALT6** — Selecting Pad: EPDC_DATA04 Mode: ALT6 for LCD_DATA04 01 **EPDC_SDCE1_ALT7** — Selecting Pad: EPDC_SDCE1 Mode: ALT7 for LCD_DATA04 10 **LCD_DATA04_ALT0** — Selecting Pad: LCD_DATA04 Mode: ALT0 for LCD_DATA04

8.2.7.399 LCD_DATA05_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA05_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 64Ch offset = 3033_064Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA05_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA05_ALT6** — Selecting Pad: EPDC_DATA05 Mode: ALT6 for LCD_DATA05 01 **EPDC_SDCE0_ALT7** — Selecting Pad: EPDC_SDCE0 Mode: ALT7 for LCD_DATA05 10 **LCD_DATA05_ALT0** — Selecting Pad: LCD_DATA05 Mode: ALT0 for LCD_DATA05

8.2.7.400 LCD_DATA06_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA06_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 650h offset = 3033_0650h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA06_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA06_ALT6** — Selecting Pad: EPDC_DATA06 Mode: ALT6 for LCD_DATA06 01 **EPDC_BDR1_ALT7** — Selecting Pad: EPDC_BDR1 Mode: ALT7 for LCD_DATA06 10 **LCD_DATA06_ALT0** — Selecting Pad: LCD_DATA06 Mode: ALT0 for LCD_DATA06

8.2.7.401 LCD_DATA07_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA07_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 654h offset = 3033_0654h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA07_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA07_ALT6** — Selecting Pad: EPDC_DATA07 Mode: ALT6 for LCD_DATA07 01 **EPDC_BDR0_ALT7** — Selecting Pad: EPDC_BDR0 Mode: ALT7 for LCD_DATA07 10 **LCD_DATA07_ALT0** — Selecting Pad: LCD_DATA07 Mode: ALT0 for LCD_DATA07

8.2.7.402 LCD_DATA08_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA08_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 658h offset = 3033_0658h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA08_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA08_ALT6** — Selecting Pad: EPDC_DATA08 Mode: ALT6 for LCD_DATA08 01 **EPDC_SDLE_ALT7** — Selecting Pad: EPDC_SDLE Mode: ALT7 for LCD_DATA08 10 **LCD_DATA08_ALT0** — Selecting Pad: LCD_DATA08 Mode: ALT0 for LCD_DATA08

8.2.7.403 LCD_DATA09_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA09_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 65Ch offset = 3033_065Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA09_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA09_ALT6** — Selecting Pad: EPDC_DATA09 Mode: ALT6 for LCD_DATA09 01 **EPDC_DATA10_ALT7** — Selecting Pad: EPDC_DATA10 Mode: ALT7 for LCD_DATA09 10 **LCD_DATA09_ALT0** — Selecting Pad: LCD_DATA09 Mode: ALT0 for LCD_DATA09

8.2.7.404 LCD_DATA10_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA10_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 660h offset = 3033_0660h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA10_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA10_ALT6** — Selecting Pad: EPDC_DATA10 Mode: ALT6 for LCD_DATA10 01 **EPDC_SDSHR_ALT7** — Selecting Pad: EPDC_SDSHR Mode: ALT7 for LCD_DATA10 10 **LCD_DATA10_ALT0** — Selecting Pad: LCD_DATA10 Mode: ALT0 for LCD_DATA10

8.2.7.405 LCD_DATA11_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA11_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 664h offset = 3033_0664h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA11_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA11_ALT6** — Selecting Pad: EPDC_DATA11 Mode: ALT6 for LCD_DATA11 01 **EPDC_PWR_COM_ALT7** — Selecting Pad: EPDC_PWR_COM Mode: ALT7 for LCD_DATA11 10 **LCD_DATA11_ALT0** — Selecting Pad: LCD_DATA11 Mode: ALT0 for LCD_DATA11

8.2.7.406 LCD_DATA12_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA12_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 668h offset = 3033_0668h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA12_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA12_ALT6** — Selecting Pad: EPDC_DATA12 Mode: ALT6 for LCD_DATA12 01 **EPDC_PWR_STAT_ALT7** — Selecting Pad: EPDC_PWR_STAT Mode: ALT7 for LCD_DATA12 10 **LCD_DATA12_ALT0** — Selecting Pad: LCD_DATA12 Mode: ALT0 for LCD_DATA12

8.2.7.407 LCD_DATA13_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA13_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 66Ch offset = 3033_066Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA13_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA13_ALT6** — Selecting Pad: EPDC_DATA13 Mode: ALT6 for LCD_DATA13 01 **LCD_DATA13_ALT0** — Selecting Pad: LCD_DATA13 Mode: ALT0 for LCD_DATA13 10 **ECSPI2_SCLK_ALT4** — Selecting Pad: ECSPI2_SCLK Mode: ALT4 for LCD_DATA13

8.2.7.408 LCD_DATA14_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA14_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 670h offset = 3033_0670h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA14_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA14_ALT6** — Selecting Pad: EPDC_DATA14 Mode: ALT6 for LCD_DATA14 01 **LCD_DATA14_ALT0** — Selecting Pad: LCD_DATA14 Mode: ALT0 for LCD_DATA14 10 **ECSPI2_MOSI_ALT4** — Selecting Pad: ECSPI2_MOSI Mode: ALT4 for LCD_DATA14

8.2.7.409 LCD_DATA15_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA15_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 674h offset = 3033_0674h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA15_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA15_ALT6** — Selecting Pad: EPDC_DATA15 Mode: ALT6 for LCD_DATA15 01 **LCD_DATA15_ALT0** — Selecting Pad: LCD_DATA15 Mode: ALT0 for LCD_DATA15 10 **ECSPI2_MISO_ALT4** — Selecting Pad: ECSPI2_MISO Mode: ALT4 for LCD_DATA15

8.2.7.410 LCD_DATA16_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA16_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 678h offset = 3033_0678h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA16_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_SDLE_ALT6** — Selecting Pad: EPDC_SDLE Mode: ALT6 for LCD_DATA16 01 **EPDC_GDCLK_ALT7** — Selecting Pad: EPDC_GDCLK Mode: ALT7 for LCD_DATA16 10 **LCD_DATA16_ALT0** — Selecting Pad: LCD_DATA16 Mode: ALT0 for LCD_DATA16

8.2.7.411 LCD_DATA17_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA17_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 67Ch offset = 3033_067Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA17_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_SDOE_ALT6** — Selecting Pad: EPDC_SDOE Mode: ALT6 for LCD_DATA17 01 **EPDC_GDSP_ALT7** — Selecting Pad: EPDC_GDSP Mode: ALT7 for LCD_DATA17 10 **LCD_DATA17_ALT0** — Selecting Pad: LCD_DATA17 Mode: ALT0 for LCD_DATA17

8.2.7.412 LCD_DATA18_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA18_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 680h offset = 3033_0680h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA18_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_SDSHR_ALT6** — Selecting Pad: EPDC_SDSHR Mode: ALT6 for LCD_DATA18 01 **EPDC_GDOE_ALT7** — Selecting Pad: EPDC_GDOE Mode: ALT7 for LCD_DATA18 10 **LCD_DATA18_ALT0** — Selecting Pad: LCD_DATA18 Mode: ALT0 for LCD_DATA18

8.2.7.413 LCD_DATA19_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA19_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 684h offset = 3033_0684h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA19_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_SDCE0_ALT6** — Selecting Pad: EPDC_SDCE0 Mode: ALT6 for LCD_DATA19 01 **EPDC_GDRL_ALT7** — Selecting Pad: EPDC_GDRL Mode: ALT7 for LCD_DATA19 10 **LCD_DATA19_ALT0** — Selecting Pad: LCD_DATA19 Mode: ALT0 for LCD_DATA19

8.2.7.414 LCD_DATA20_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA20_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 688h offset = 3033_0688h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA20_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_SDCLK_ALT7** — Selecting Pad: EPDC_SDCLK Mode: ALT7 for LCD_DATA20 01 **EPDC_SDCE1_ALT6** — Selecting Pad: EPDC_SDCE1 Mode: ALT6 for LCD_DATA20 10 **LCD_DATA20_ALT0** — Selecting Pad: LCD_DATA20 Mode: ALT0 for LCD_DATA20

8.2.7.415 LCD_DATA21_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA21_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 68Ch offset = 3033_068Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA21_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA12_ALT7** — Selecting Pad: EPDC_DATA12 Mode: ALT7 for LCD_DATA21 01 **EPDC_SDCE2_ALT6** — Selecting Pad: EPDC_SDCE2 Mode: ALT6 for LCD_DATA21 10 **LCD_DATA21_ALT0** — Selecting Pad: LCD_DATA21 Mode: ALT0 for LCD_DATA21

8.2.7.416 LCD_DATA22_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA22_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 690h offset = 3033_0690h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA22_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA14_ALT7** — Selecting Pad: EPDC_DATA14 Mode: ALT7 for LCD_DATA22 01 **EPDC_SDCE3_ALT6** — Selecting Pad: EPDC_SDCE3 Mode: ALT6 for LCD_DATA22 10 **LCD_DATA22_ALT0** — Selecting Pad: LCD_DATA22 Mode: ALT0 for LCD_DATA22

8.2.7.417 LCD_DATA23_SELECT_INPUT DAISY Register (IOMUXC_LCD_DATA23_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 694h offset = 3033_0694h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LCD_DATA23_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_SDOE_ALT7** — Selecting Pad: EPDC_SDOE Mode: ALT7 for LCD_DATA23 01 **EPDC_GDCLK_ALT6** — Selecting Pad: EPDC_GDCLK Mode: ALT6 for LCD_DATA23 10 **LCD_DATA23_ALT0** — Selecting Pad: LCD_DATA23 Mode: ALT0 for LCD_DATA23

8.2.7.418 LCD_VSYNC_SELECT_INPUT DAISY Register (IOMUXC_LCD_VSYNC_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 698h offset = 3033_0698h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

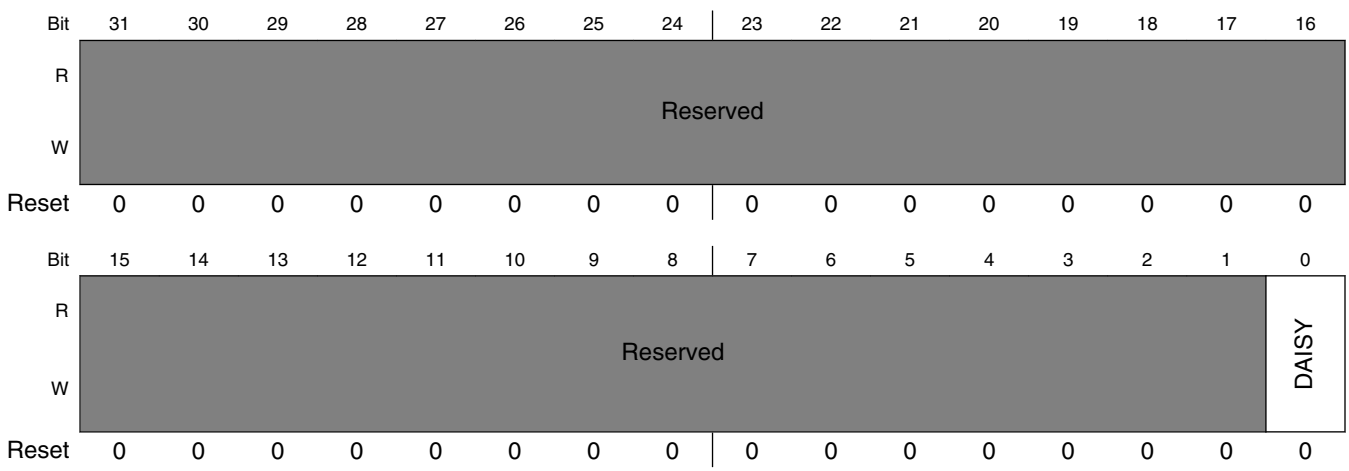
IOMUXC_LCD_VSYNC_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **EPDC_DATA02_ALT7** — Selecting Pad: EPDC_DATA02 Mode: ALT7 for LCD_VSYNC 01 **EPDC_PWR_STAT_ALT6** — Selecting Pad: EPDC_PWR_STAT Mode: ALT6 for LCD_VSYNC 10 **LCD_VSYNC_ALT0** — Selecting Pad: LCD_VSYNC Mode: ALT0 for LCD_VSYNC

8.2.7.419 SAI1_RX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_BCLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 69Ch offset = 3033_069Ch



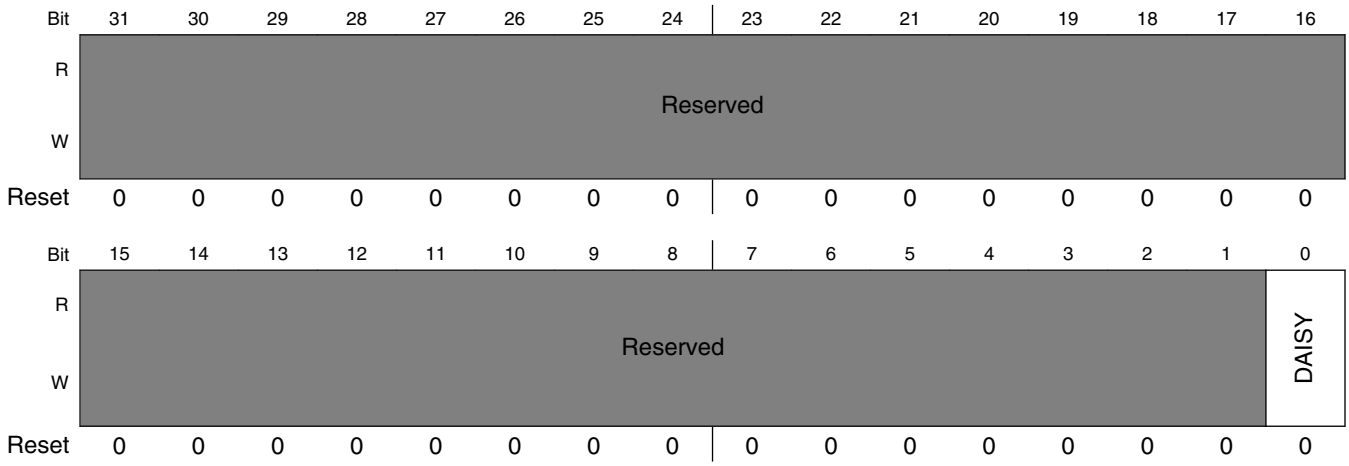
IOMUXC_SAI1_RX_BCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **SAI1_RX_BCLK_ALT0** — Selecting Pad: SAI1_RX_BCLK Mode: ALT0 for SAI1_RX_BCLK 1 **ENET1_TXC_ALT2** — Selecting Pad: ENET1_TXC Mode: ALT2 for SAI1_RX_BCLK

8.2.7.420 SAI1_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6A0h offset = 3033_06A0h



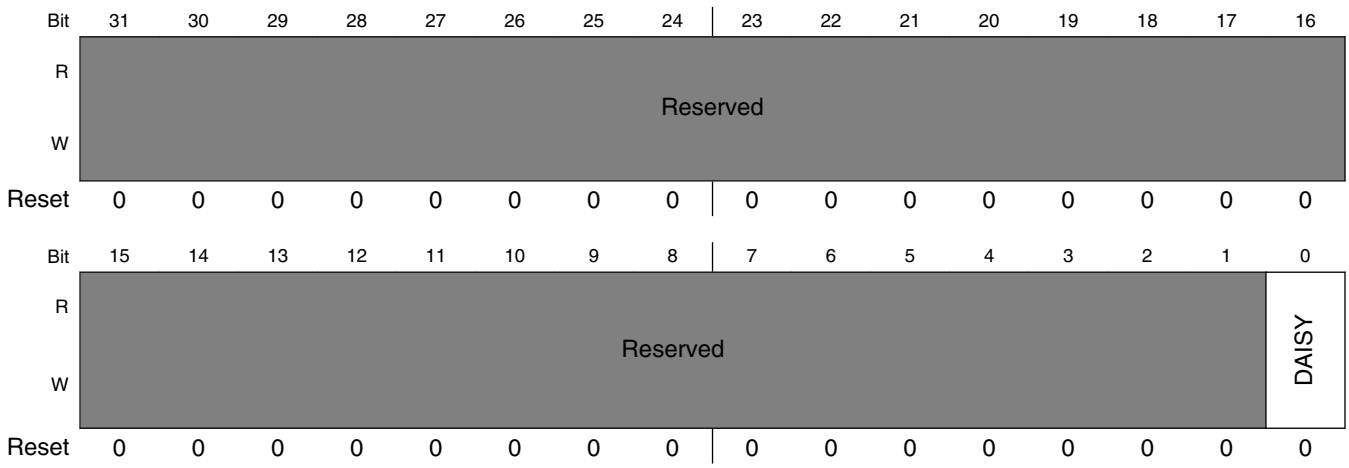
IOMUXC_SAI1_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **SAI1_RX_DATA_ALT0** — Selecting Pad: SAI1_RX_DATA Mode: ALT0 for SAI1_RX_DATA 1 **ENET1_TX_CLK_ALT2** — Selecting Pad: ENET1_TX_CLK Mode: ALT2 for SAI1_RX_DATA

8.2.7.421 SAI1_RX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_SYNC_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6A4h offset = 3033_06A4h



IOMUXC_SAI1_RX_SYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **SAI1_RX_SYNC_ALT0** — Selecting Pad: SAI1_RX_SYNC Mode: ALT0 for SAI1_RX_SYNC 1 **ENET1_TX_CTL_ALT2** — Selecting Pad: ENET1_TX_CTL Mode: ALT2 for SAI1_RX_SYNC

8.2.7.422 SAI1_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI1_TX_BCLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6A8h offset = 3033_06A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SAI1_TX_BCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **SAI1_TX_BCLK_ALT0** — Selecting Pad: SAI1_TX_BCLK Mode: ALT0 for SAI1_TX_BCLK 1 **ENET1_RX_CLK_ALT2** — Selecting Pad: ENET1_RX_CLK Mode: ALT2 for SAI1_TX_BCLK

8.2.7.423 SAI1_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI1_TX_SYNC_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6ACh offset = 3033_06ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SAI1_TX_SYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **SAI1_TX_SYNC_ALT0** — Selecting Pad: SAI1_TX_SYNC Mode: ALT0 for SAI1_TX_SYNC 1 **ENET1_CRS_ALT2** — Selecting Pad: ENET1_CRS Mode: ALT2 for SAI1_TX_SYNC

8.2.7.424 SAI2_RX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_BCLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6B0h offset = 3033_06B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

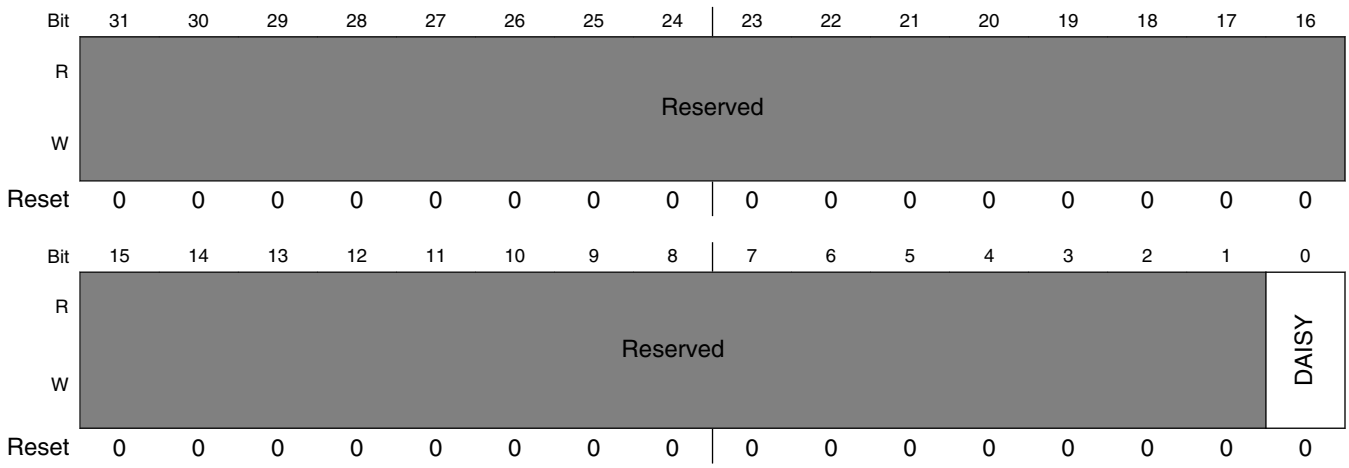
IOMUXC_SAI2_RX_BCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **SD2_CMD_ALT1** — Selecting Pad: SD2_CMD Mode: ALT1 for SAI2_RX_BCLK 1 **SAI1_RX_BCLK_ALT2** — Selecting Pad: SAI1_RX_BCLK Mode: ALT2 for SAI2_RX_BCLK

8.2.7.425 SAI2_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6B4h offset = 3033_06B4h



IOMUXC_SAI2_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **SD2_DATA0_ALT1** — Selecting Pad: SD2_DATA0 Mode: ALT1 for SAI2_RX_DATA 1 **SAI2_RX_DATA_ALT0** — Selecting Pad: SAI2_RX_DATA Mode: ALT0 for SAI2_RX_DATA

8.2.7.426 SAI2_RX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_SYNC_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6B8h offset = 3033_06B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

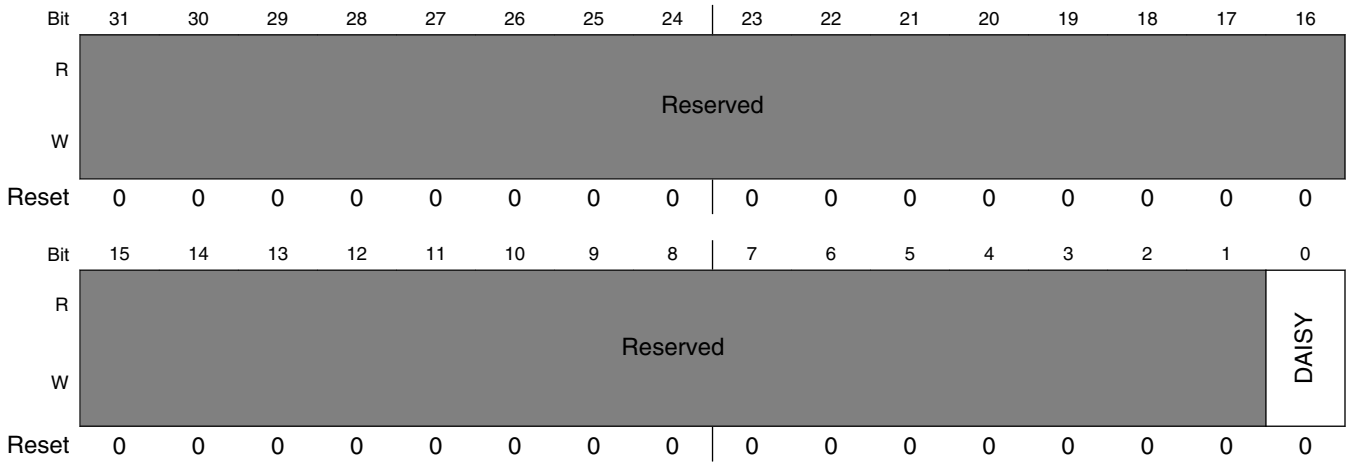
IOMUXC_SAI2_RX_SYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **SD2_CLK_ALT1** — Selecting Pad: SD2_CLK Mode: ALT1 for SAI2_RX_SYNC 1 **SAI1_RX_SYNC_ALT2** — Selecting Pad: SAI1_RX_SYNC Mode: ALT2 for SAI2_RX_SYNC

8.2.7.427 SAI2_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI2_TX_BCLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6BCh offset = 3033_06BCh



IOMUXC_SAI2_TX_BCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **SD2_DATA1_ALT1** — Selecting Pad: SD2_DATA1 Mode: ALT1 for SAI2_TX_BCLK 1 **SAI2_TX_BCLK_ALT0** — Selecting Pad: SAI2_TX_BCLK Mode: ALT0 for SAI2_TX_BCLK

8.2.7.428 SAI2_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI2_TX_SYNC_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6C0h offset = 3033_06C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SAI2_TX_SYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **SD2_DATA2_ALT1** — Selecting Pad: SD2_DATA2 Mode: ALT1 for SAI2_TX_SYNC 1 **SAI2_TX_SYNC_ALT0** — Selecting Pad: SAI2_TX_SYNC Mode: ALT0 for SAI2_TX_SYNC

8.2.7.429 SAI3_RX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI3_RX_BCLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6C4h offset = 3033_06C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SAI3_RX_BCLK_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **UART2_RX_DATA_ALT2** — Selecting Pad: UART2_RX_DATA Mode: ALT2 for SAI3_RX_BCLK 01 **SD1_CMD_ALT1** — Selecting Pad: SD1_CMD Mode: ALT1 for SAI3_RX_BCLK 10 **SD3_CMD_ALT3** — Selecting Pad: SD3_CMD Mode: ALT3 for SAI3_RX_BCLK

8.2.7.430 SAI3_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_SAI3_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6C8h offset = 3033_06C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SAI3_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **UART2_TX_DATA_ALT2** — Selecting Pad: UART2_TX_DATA Mode: ALT2 for SAI3_RX_DATA 01 **SD1_DATA0_ALT1** — Selecting Pad: SD1_DATA0 Mode: ALT1 for SAI3_RX_DATA 10 **SD3_DATA0_ALT3** — Selecting Pad: SD3_DATA0 Mode: ALT3 for

8.2.7.431 SAI3_RX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI3_RX_SYNC_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6CCh offset = 3033_06CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SAI3_RX_SYNC_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **UART3_RX_DATA_ALT2** — Selecting Pad: UART3_RX_DATA Mode: ALT2 for SAI3_RX_SYNC 01 **SD1_CLK_ALT1** — Selecting Pad: SD1_CLK Mode: ALT1 for SAI3_RX_SYNC 10 **SD3_CLK_ALT3** — Selecting Pad: SD3_CLK Mode: ALT3 for SAI3_RX_SYNC

8.2.7.432 SAI3_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI3_TX_BCLK_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6D0h offset = 3033_06D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SAI3_TX_BCLK_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **UART3_TX_DATA_ALT2** — Selecting Pad: UART3_TX_DATA Mode: ALT2 for SAI3_TX_BCLK 01 **SD1_DATA1_ALT1** — Selecting Pad: SD1_DATA1 Mode: ALT1 for SAI3_TX_BCLK 10 **SD3_DATA1_ALT3** — Selecting Pad: SD3_DATA1 Mode: ALT3 for SAI3_TX_BCLK

8.2.7.433 SAI3_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI3_TX_SYNC_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6D4h offset = 3033_06D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SAI3_TX_SYNC_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **UART3_CTS_B_ALT2** — Selecting Pad: UART3_CTS_B Mode: ALT2 for SAI3_TX_SYNC 01 **SD1_DATA2_ALT1** — Selecting Pad: SD1_DATA2 Mode: ALT1 for SAI3_TX_SYNC 10 **SD3_DATA2_ALT3** — Selecting Pad: SD3_DATA2 Mode: ALT3 for SAI3_TX_SYNC

8.2.7.434 SDMA_EVENTS0_SELECT_INPUT DAISY Register (IOMUXC_SDMA_EVENTS0_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6D8h offset = 3033_06D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SDMA_EVENTS0_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO14_ALT6** — Selecting Pad: GPIO1_IO14 Mode: ALT6 for SDMA_EVENTS0 01 **I2C3_SCL_ALT4** — Selecting Pad: I2C3_SCL Mode: ALT4 for SDMA_EVENTS0 10 **SD2_CD_B_ALT6** — Selecting Pad: SD2_CD_B Mode: ALT6 for SDMA_EVENTS0

8.2.7.435 SDMA_EVENTS1_SELECT_INPUT DAISY Register (IOMUXC_SDMA_EVENTS1_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6DCh offset = 3033_06DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

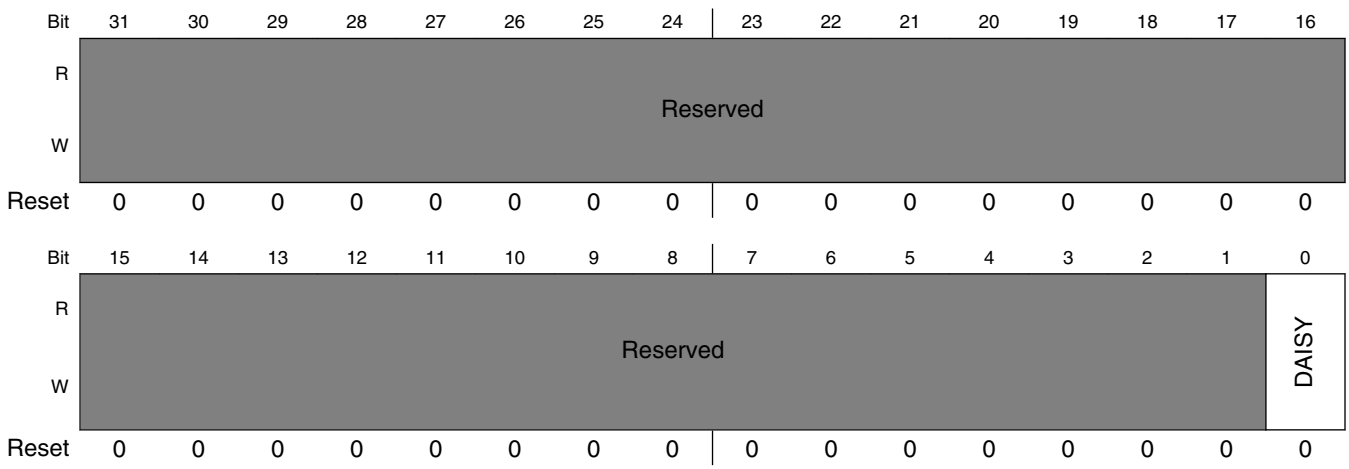
IOMUXC_SDMA_EVENTS1_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO15_ALT6** — Selecting Pad: GPIO1_IO15 Mode: ALT6 for SDMA_EVENTS1 01 **I2C3_SDA_ALT4** — Selecting Pad: I2C3_SDA Mode: ALT4 for SDMA_EVENTS1 10 **SD2_WP_ALT6** — Selecting Pad: SD2_WP Mode: ALT6 for SDMA_EVENTS1

8.2.7.436 SIM1_PORT1_PD_SELECT_INPUT DAISY Register (IOMUXC_SIM1_PORT1_PD_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6E0h offset = 3033_06E0h



IOMUXC_SIM1_PORT1_PD_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_DATA12_ALT1** — Selecting Pad: EPDC_DATA12 Mode: ALT1 for SIM1_PORT1_PD 1 **SAI1_RX_SYNC_ALT4** — Selecting Pad: SAI1_RX_SYNC Mode: ALT4 for SIM1_PORT1_PD

8.2.7.437 SIM1_PORT1_TRXD_SELECT_INPUT DAISY Register (IOMUXC_SIM1_PORT1_TRXD_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6E4h offset = 3033_06E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

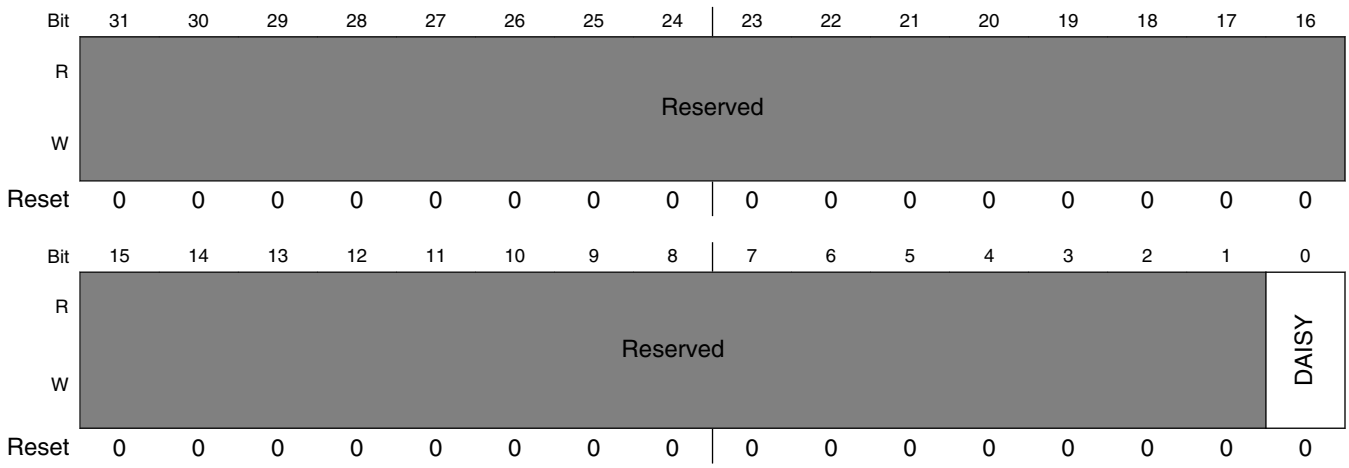
IOMUXC_SIM1_PORT1_TRXD_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_DATA08_ALT1** — Selecting Pad: EPDC_DATA08 Mode: ALT1 for SIM1_PORT1_TRXD 1 **SAI1_RX_DATA_ALT4** — Selecting Pad: SAI1_RX_DATA Mode: ALT4 for SIM1_PORT1_TRXD

8.2.7.438 SIM2_PORT1_PD_SELECT_INPUT DAISY Register (IOMUXC_SIM2_PORT1_PD_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6E8h offset = 3033_06E8h



IOMUXC_SIM2_PORT1_PD_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_SDCE3_ALT1** — Selecting Pad: EPDC_SDCE3 Mode: ALT1 for SIM2_PORT1_PD 1 **SD2_DATA3_ALT4** — Selecting Pad: SD2_DATA3 Mode: ALT4 for SIM2_PORT1_PD

8.2.7.439 SIM2_PORT1_TRXD_SELECT_INPUT DAISY Register (IOMUXC_SIM2_PORT1_TRXD_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6ECh offset = 3033_06ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SIM2_PORT1_TRXD_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **EPDC_DATA13_ALT1** — Selecting Pad: EPDC_DATA13 Mode: ALT1 for SIM2_PORT1_TRXD 1 **SD2_CMD_ALT4** — Selecting Pad: SD2_CMD Mode: ALT4 for SIM2_PORT1_TRXD

8.2.7.440 UART1_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART1_RTS_B_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6F0h offset = 3033_06F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_UART1_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **SAI2_TX_SYNC_ALT3** — Selecting Pad: SAI2_TX_SYNC Mode: ALT3 for UART1_RTS_B 01 **SAI2_TX_BCLK_ALT3** — Selecting Pad: SAI2_TX_BCLK Mode: ALT3 for UART1_RTS_B 10 **ENET1_RGMII_RD0_ALT3** — Selecting Pad: ENET1_RGMII_RD0 Mode: ALT3 for UART1_RTS_B 11 **ENET1_RGMII_RD1_ALT3** — Selecting Pad: ENET1_RGMII_RD1 Mode: ALT3 for UART1_RTS_B

8.2.7.441 UART1_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART1_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6F4h offset = 3033_06F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_UART1_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **UART1_RX_DATA_ALT0** — Selecting Pad: UART1_RX_DATA Mode: ALT0 for UART1_RX_DATA 01 **UART1_TX_DATA_ALT0** — Selecting Pad: UART1_TX_DATA Mode: ALT0 for UART1_RX_DATA 10 **ENET1_RGMII_RD2_ALT3** — Selecting Pad: ENET1_RGMII_RD2 Mode: ALT3 for UART1_RX_DATA 11 **ENET1_RGMII_RD3_ALT3** — Selecting Pad: ENET1_RGMII_RD3 Mode: ALT3 for UART1_RX_DATA

8.2.7.442 UART2_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART2_RTS_B_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6F8h offset = 3033_06F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_UART2_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **LCD_HSYNC_ALT4** — Selecting Pad: LCD_HSYNC Mode: ALT4 for UART2_RTS_B 01 **LCD_VSYNC_ALT4** — Selecting Pad: LCD_VSYNC Mode: ALT4 for UART2_RTS_B 10 **SAI2_RX_DATA_ALT3** — Selecting Pad: SAI2_RX_DATA Mode: ALT3 for UART2_RTS_B 11 **SAI2_TX_DATA_ALT3** — Selecting Pad: SAI2_TX_DATA Mode: ALT3 for UART2_RTS_B

8.2.7.443 UART2_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART2_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 6FCh offset = 3033_06FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

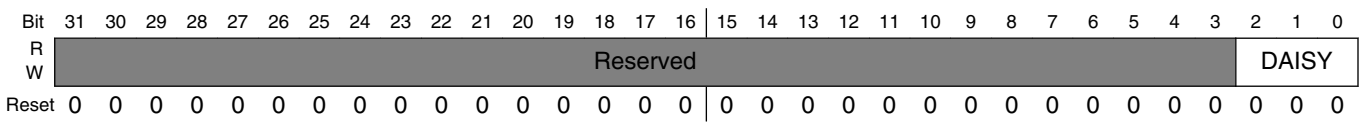
IOMUXC_UART2_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **LCD_CLK_ALT4** — Selecting Pad: LCD_CLK Mode: ALT4 for UART2_RX_DATA 01 **LCD_ENABLE_ALT4** — Selecting Pad: LCD_ENABLE Mode: ALT4 for UART2_RX_DATA 10 **UART2_RX_DATA_ALT0** — Selecting Pad: UART2_RX_DATA Mode: ALT0 for UART2_RX_DATA 11 **UART2_TX_DATA_ALT0** — Selecting Pad: UART2_TX_DATA Mode: ALT0 for UART2_RX_DATA

8.2.7.444 UART3_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART3_RTS_B_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 700h offset = 3033_0700h



IOMUXC_UART3_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 000 **GPIO1_IO10_ALT3** — Selecting Pad: GPIO1_IO10 Mode: ALT3 for UART3_RTS_B 001 **GPIO1_IO11_ALT3** — Selecting Pad: GPIO1_IO11 Mode: ALT3 for UART3_RTS_B 010 **UART3_RTS_B_ALT0** — Selecting Pad: UART3_RTS_B Mode: ALT0 for UART3_RTS_B 011 **UART3_CTS_B_ALT0** — Selecting Pad: UART3_CTS_B Mode: ALT0 for UART3_RTS_B 100 **SD3_DATA6_ALT3** — Selecting Pad: SD3_DATA6 Mode: ALT3 for UART3_RTS_B 101 **SD3_DATA7_ALT3** — Selecting Pad: SD3_DATA7 Mode: ALT3 for UART3_RTS_B

8.2.7.445 UART3_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART3_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 704h offset = 3033_0704h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																													DAISY		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_UART3_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 000 **GPIO1_IO08_ALT3** — Selecting Pad: GPIO1_IO08 Mode: ALT3 for UART3_RX_DATA 001 **GPIO1_IO09_ALT3** — Selecting Pad: GPIO1_IO09 Mode: ALT3 for UART3_RX_DATA 010 **UART3_RX_DATA_ALT0** — Selecting Pad: UART3_RX_DATA Mode: ALT0 for UART3_RX_DATA 011 **UART3_TX_DATA_ALT0** — Selecting Pad: UART3_TX_DATA Mode: ALT0 for UART3_RX_DATA 100 **SD3_DATA4_ALT3** — Selecting Pad: SD3_DATA4 Mode: ALT3 for UART3_RX_DATA 101 **SD3_DATA5_ALT3** — Selecting Pad: SD3_DATA5 Mode: ALT3 for UART3_RX_DATA

8.2.7.446 UART4_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART4_RTS_B_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 708h offset = 3033_0708h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																													DAISY		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_UART4_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain.

Table continues on the next page...

IOMUXC_UART4_RTS_B_SELECT_INPUT field descriptions (continued)

Field	Description
	000 **I2C1_SCL_ALT1** — Selecting Pad: I2C1_SCL Mode: ALT1 for UART4_RTS_B 001 **I2C1_SDA_ALT1** — Selecting Pad: I2C1_SDA Mode: ALT1 for UART4_RTS_B 010 **SD2_DATA2_ALT2** — Selecting Pad: SD2_DATA2 Mode: ALT2 for UART4_RTS_B 011 **SD2_DATA3_ALT2** — Selecting Pad: SD2_DATA3 Mode: ALT2 for UART4_RTS_B 100 **SAI2_RX_DATA_ALT2** — Selecting Pad: SAI2_RX_DATA Mode: ALT2 for UART4_RTS_B 101 **SAI2_TX_DATA_ALT2** — Selecting Pad: SAI2_TX_DATA Mode: ALT2 for UART4_RTS_B

8.2.7.447 UART4_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART4_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 70Ch offset = 3033_070Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																													DAISY		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_UART4_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 000 **I2C2_SCL_ALT1** — Selecting Pad: I2C2_SCL Mode: ALT1 for UART4_RX_DATA 001 **I2C2_SDA_ALT1** — Selecting Pad: I2C2_SDA Mode: ALT1 for UART4_RX_DATA 010 **SD2_DATA0_ALT2** — Selecting Pad: SD2_DATA0 Mode: ALT2 for UART4_RX_DATA 011 **SD2_DATA1_ALT2** — Selecting Pad: SD2_DATA1 Mode: ALT2 for UART4_RX_DATA 100 **SAI2_TX_SYNC_ALT2** — Selecting Pad: SAI2_TX_SYNC Mode: ALT2 for UART4_RX_DATA 101 **SAI2_TX_BCLK_ALT2** — Selecting Pad: SAI2_TX_BCLK Mode: ALT2 for UART4_RX_DATA

8.2.7.448 UART5_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART5_RTS_B_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 710h offset = 3033_0710h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																													DAISY		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

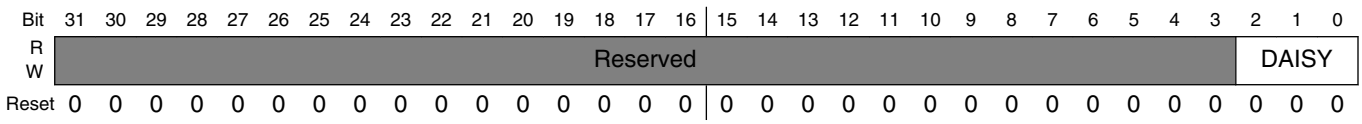
IOMUXC_UART5_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 000 **I2C3_SCL_ALT1** — Selecting Pad: I2C3_SCL Mode: ALT1 for UART5_RTS_B 001 **I2C3_SDA_ALT1** — Selecting Pad: I2C3_SDA Mode: ALT1 for UART5_RTS_B 010 **SAI1_TX_SYNC_ALT2** — Selecting Pad: SAI1_TX_SYNC Mode: ALT2 for UART5_RTS_B 011 **SAI1_TX_DATA_ALT2** — Selecting Pad: SAI1_TX_DATA Mode: ALT2 for UART5_RTS_B 100 **GPIO1_IO04_ALT3** — Selecting Pad: GPIO1_IO04 Mode: ALT3 for UART5_RTS_B 101 **GPIO1_IO05_ALT3** — Selecting Pad: GPIO1_IO05 Mode: ALT3 for UART5_RTS_B

8.2.7.449 UART5_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART5_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 714h offset = 3033_0714h



IOMUXC_UART5_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 000 **I2C4_SCL_ALT1** — Selecting Pad: I2C4_SCL Mode: ALT1 for UART5_RX_DATA 001 **I2C4_SDA_ALT1** — Selecting Pad: I2C4_SDA Mode: ALT1 for UART5_RX_DATA 010 **SAI1_RX_DATA_ALT2** — Selecting Pad: SAI1_RX_DATA Mode: ALT2 for UART5_RX_DATA 011 **SAI1_TX_BCLK_ALT2** — Selecting Pad: SAI1_TX_BCLK Mode: ALT2 for UART5_RX_DATA 100 **GPIO1_IO06_ALT3** — Selecting Pad: GPIO1_IO06 Mode: ALT3 for UART5_RX_DATA 101 **GPIO1_IO07_ALT3** — Selecting Pad: GPIO1_IO07 Mode: ALT3 for UART5_RX_DATA

8.2.7.450 UART6_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART6_RTS_B_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 718h offset = 3033_0718h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																													DAISY		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_UART6_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 000 **EPDC_DATA10_ALT3** — Selecting Pad: EPDC_DATA10 Mode: ALT3 for UART6_RTS_B 001 **EPDC_DATA11_ALT3** — Selecting Pad: EPDC_DATA11 Mode: ALT3 for UART6_RTS_B 010 **ECSPI1_MISO_ALT1** — Selecting Pad: ECSPI1_MISO Mode: ALT1 for UART6_RTS_B 011 **ECSPI1_SS0_ALT1** — Selecting Pad: ECSPI1_SS0 Mode: ALT1 for UART6_RTS_B 100 **SD1_RESET_B_ALT2** — Selecting Pad: SD1_RESET_B Mode: ALT2 for UART6_RTS_B 101 **SD1_CLK_ALT2** — Selecting Pad: SD1_CLK Mode: ALT2 for UART6_RTS_B

8.2.7.451 UART6_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART6_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 71Ch offset = 3033_071Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																													DAISY		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_UART6_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain.

Table continues on the next page...

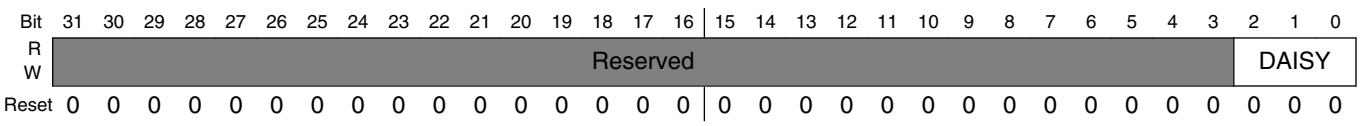
IOMUXC_UART6_RX_DATA_SELECT_INPUT field descriptions (continued)

Field	Description
	000 **EPDC_DATA08_ALT3** — Selecting Pad: EPDC_DATA08 Mode: ALT3 for UART6_RX_DATA 001 **EPDC_DATA09_ALT3** — Selecting Pad: EPDC_DATA09 Mode: ALT3 for UART6_RX_DATA 010 **ECSPI1_SCLK_ALT1** — Selecting Pad: ECSPI1_SCLK Mode: ALT1 for UART6_RX_DATA 011 **ECSPI1_MOSI_ALT1** — Selecting Pad: ECSPI1_MOSI Mode: ALT1 for UART6_RX_DATA 100 **SD1_CD_B_ALT2** — Selecting Pad: SD1_CD_B Mode: ALT2 for UART6_RX_DATA 101 **SD1_WP_ALT2** — Selecting Pad: SD1_WP Mode: ALT2 for UART6_RX_DATA

8.2.7.452 UART7_RTS_B_SELECT_INPUT DAISY Register (IOMUXC_UART7_RTS_B_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 720h offset = 3033_0720h



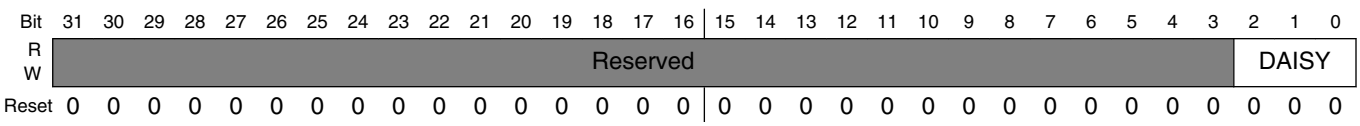
IOMUXC_UART7_RTS_B_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 000 **EPDC_DATA14_ALT3** — Selecting Pad: EPDC_DATA14 Mode: ALT3 for UART7_RTS_B 001 **EPDC_DATA15_ALT3** — Selecting Pad: EPDC_DATA15 Mode: ALT3 for UART7_RTS_B 010 **ECSPI2_MISO_ALT1** — Selecting Pad: ECSPI2_MISO Mode: ALT1 for UART7_RTS_B 011 **ECSPI2_SS0_ALT1** — Selecting Pad: ECSPI2_SS0 Mode: ALT1 for UART7_RTS_B 100 **SD1_DATA2_ALT2** — Selecting Pad: SD1_DATA2 Mode: ALT2 for UART7_RTS_B 101 **SD1_DATA3_ALT2** — Selecting Pad: SD1_DATA3 Mode: ALT2 for UART7_RTS_B

8.2.7.453 UART7_RX_DATA_SELECT_INPUT DAISY Register (IOMUXC_UART7_RX_DATA_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 724h offset = 3033_0724h



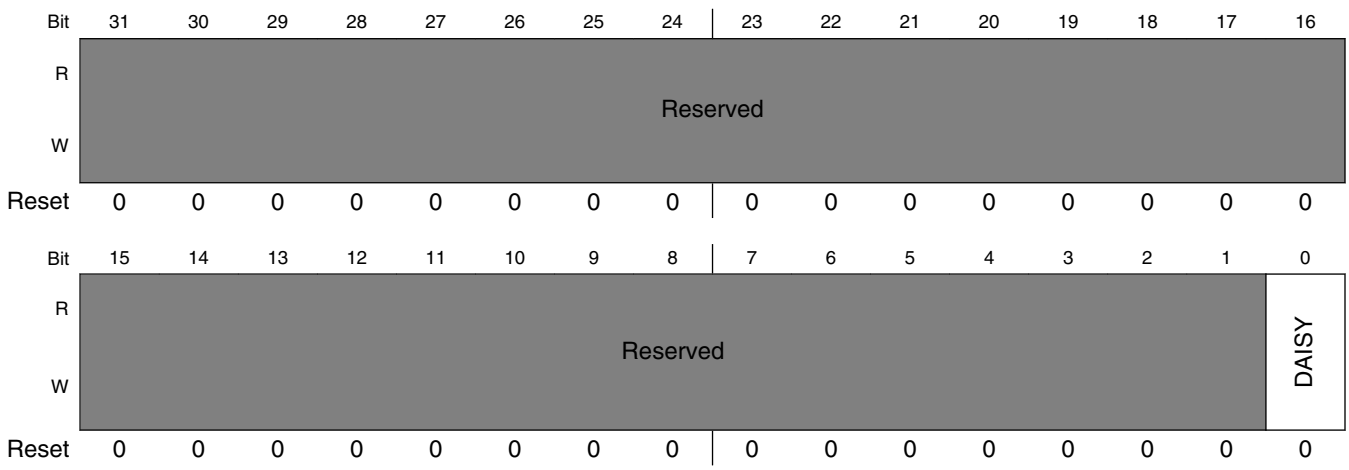
IOMUXC_UART7_RX_DATA_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 000 **EPDC_DATA12_ALT3** — Selecting Pad: EPDC_DATA12 Mode: ALT3 for UART7_RX_DATA 001 **EPDC_DATA13_ALT3** — Selecting Pad: EPDC_DATA13 Mode: ALT3 for UART7_RX_DATA 010 **ECSPI2_SCLK_ALT1** — Selecting Pad: ECSPI2_SCLK Mode: ALT1 for UART7_RX_DATA 011 **ECSPI2_MOSI_ALT1** — Selecting Pad: ECSPI2_MOSI Mode: ALT1 for UART7_RX_DATA 100 **SD1_DATA0_ALT2** — Selecting Pad: SD1_DATA0 Mode: ALT2 for UART7_RX_DATA 101 **SD1_DATA1_ALT2** — Selecting Pad: SD1_DATA1 Mode: ALT2 for UART7_RX_DATA

8.2.7.454 USB_OTG2_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG2_OC_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 728h offset = 3033_0728h



IOMUXC_USB_OTG2_OC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **UART3_RTS_B_ALT1** — Selecting Pad: UART3_RTS_B Mode: ALT1 for USB_OTG2_OC 1 **GPIO1_IO06_ALT1** — Selecting Pad: GPIO1_IO06 Mode: ALT1 for USB_OTG2_OC

8.2.7.455 USB_OTG1_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG1_OC_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 72Ch offset = 3033_072Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_USB_OTG1_OC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 0 **UART3_RX_DATA_ALT1** — Selecting Pad: UART3_RX_DATA Mode: ALT1 for USB_OTG1_OC 1 **GPIO1_IO04_ALT1** — Selecting Pad: GPIO1_IO04 Mode: ALT1 for USB_OTG1_OC

8.2.7.456 USB_OTG2_ID_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG2_ID_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 730h offset = 3033_0730h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_USB_OTG2_ID_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO13_ALT7** — Selecting Pad: GPIO1_IO13 Mode: ALT7 for USB_OTG2_ID 01 **I2C4_SDA_ALT4** — Selecting Pad: I2C4_SDA Mode: ALT4 for USB_OTG2_ID 10 **SD2_RESET_B_ALT4** — Selecting Pad: SD2_RESET_B Mode: ALT4 for USB_OTG2_ID 11 **GPIO1_IO03_ALT7** — Selecting Pad: GPIO1_IO03 Mode: ALT7 for USB_OTG2_ID

8.2.7.457 USB_OTG1_ID_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG1_ID_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 734h offset = 3033_0734h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_USB_OTG1_ID_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO12_ALT7** — Selecting Pad: GPIO1_IO12 Mode: ALT7 for USB_OTG1_ID 01 **I2C4_SCL_ALT4** — Selecting Pad: I2C4_SCL Mode: ALT4 for USB_OTG1_ID 10 **SD2_WP_ALT4** — Selecting Pad: SD2_WP Mode: ALT4 for USB_OTG1_ID 11 **GPIO1_IO02_ALT7** — Selecting Pad: GPIO1_IO02 Mode: ALT7 for USB_OTG1_ID

8.2.7.458 SD3_CD_B_SELECT_INPUT DAISY Register (IOMUXC_SD3_CD_B_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 738h offset = 3033_0738h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SD3_CD_B_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO14_ALT1** — Selecting Pad: GPIO1_IO14 Mode: ALT1 for SD3_CD_B 01 **I2C2_SCL_ALT6** — Selecting Pad: I2C2_SCL Mode: ALT6 for SD3_CD_B 10 **SD3_DATA7_ALT2** — Selecting Pad: SD3_DATA7 Mode: ALT2 for SD3_CD_B

8.2.7.459 SD3_WP_SELECT_INPUT DAISY Register (IOMUXC_SD3_WP_SELECT_INPUT)

DAISY Register

Address: 3033_0000h base + 73Ch offset = 3033_073Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SD3_WP_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Input Select (DAISY) Field Selecting Pads Involved in Daisy Chain. 00 **GPIO1_IO15_ALT1** — Selecting Pad: GPIO1_IO15 Mode: ALT1 for SD3_WP 01 **I2C2_SDA_ALT6** — Selecting Pad: I2C2_SDA Mode: ALT6 for SD3_WP 10 **SD3_DATA6_ALT2** — Selecting Pad: SD3_DATA6 Mode: ALT2 for SD3_WP

8.3 General Purpose Input/Output (GPIO)

8.3.1 Overview

The GPIO general-purpose input/output peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs.

When configured as an output, it is possible to write to an internal register to control the state driven on the output pin. When configured as an input, it is possible to detect the state of the input by reading the state of an internal register. In addition, the GPIO peripheral can produce CORE interrupts.

The GPIO is one of the blocks controlling the IOMUX of the chip.

Figure 8-4 shows the chip multiplexing scheme.

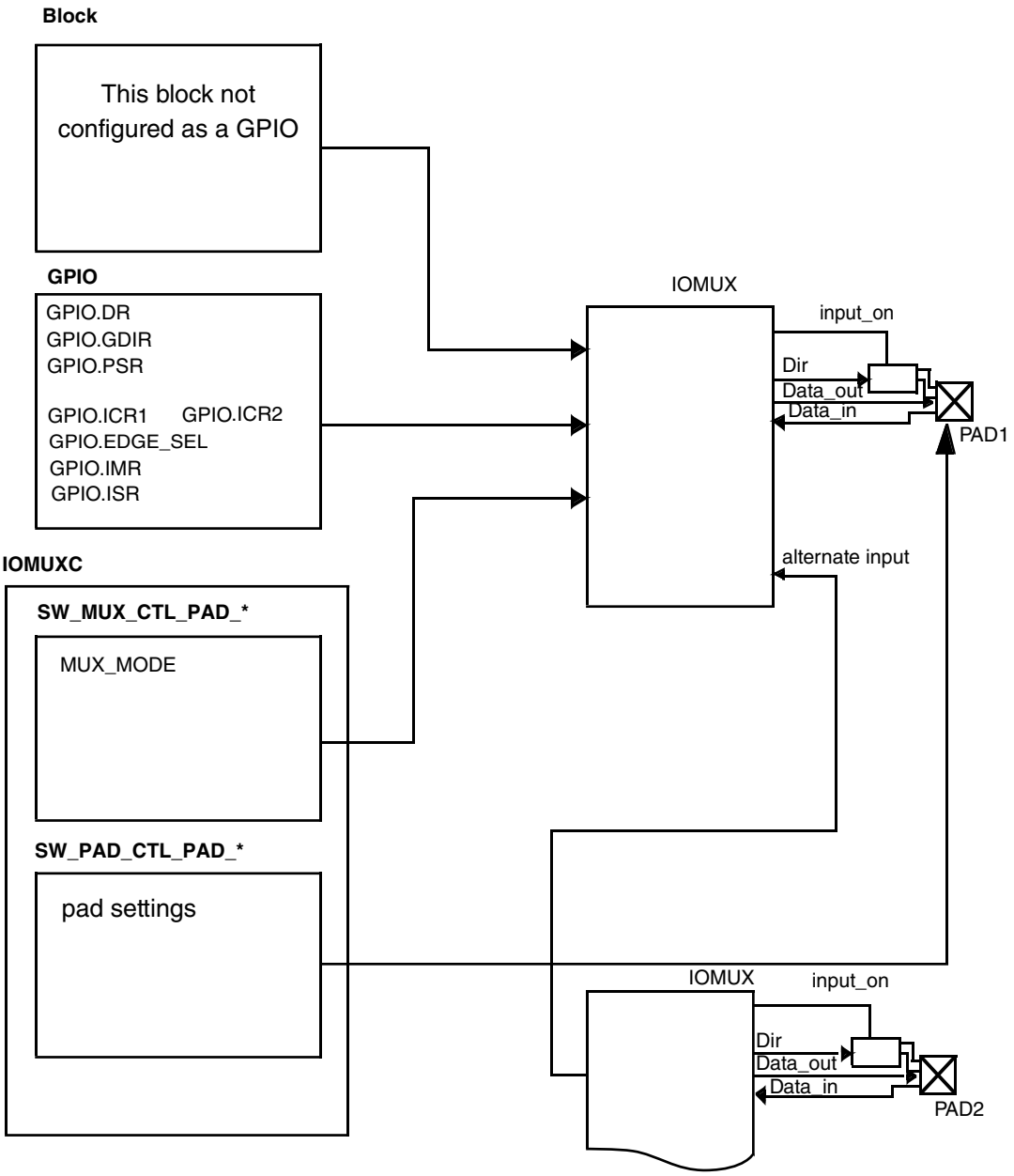


Figure 8-4. Chip IOMUX Scheme

The GPIO functionality is provided through eight registers, an edge-detect circuit, and interrupt generation logic.

The eight registers are:

- Data register (GPIO_DR)
- GPIO direction register (GPIO_GDIR)
- Pad sample register (GPIO_PSR)
- Interrupt control registers (GPIO_ICR1, GPIO_ICR2)

- Edge select register (GPIO_EDGE_SEL)
- Interrupt mask register (GPIO_IMR)
- Interrupt status register (GPIO_ISR)

These registers are described in detail in GPIO Memory Map/Register Definition.

Each GPIO input has a dedicated edge-detect circuit which can be configured through software to detect rising edges, falling edges, logic low-levels or logic high-levels on the input signals. The outputs of the edge detect circuits are optionally masked by setting the corresponding bit in the interrupt mask register (GPIO_IMR). These qualified outputs are OR'ed together to generate two one-bit interrupt lines:

- Combined interrupt indication for GPIOx signals 0 - 15
- Combined interrupt indication for GPIOx signals 16 - 31

In addition, GPIO1 provides visibility to each of its 8 low order interrupt sources (i.e. GPIO1 interrupt n, for n = 0 – 7). However, individual interrupt indications from other GPIOx are not available.

The GPIO edge detection is described further in Interrupt Control Unit.

The GPIO's overall functionality is described further in GPIO Functional Description.

8.3.1.1 Block Diagram

The GPIO subsystem contains 7 GPIO blocks which can generate and control up to 32 signals for general purpose.

A block diagram of the GPIO is shown in Figure 8-5 .

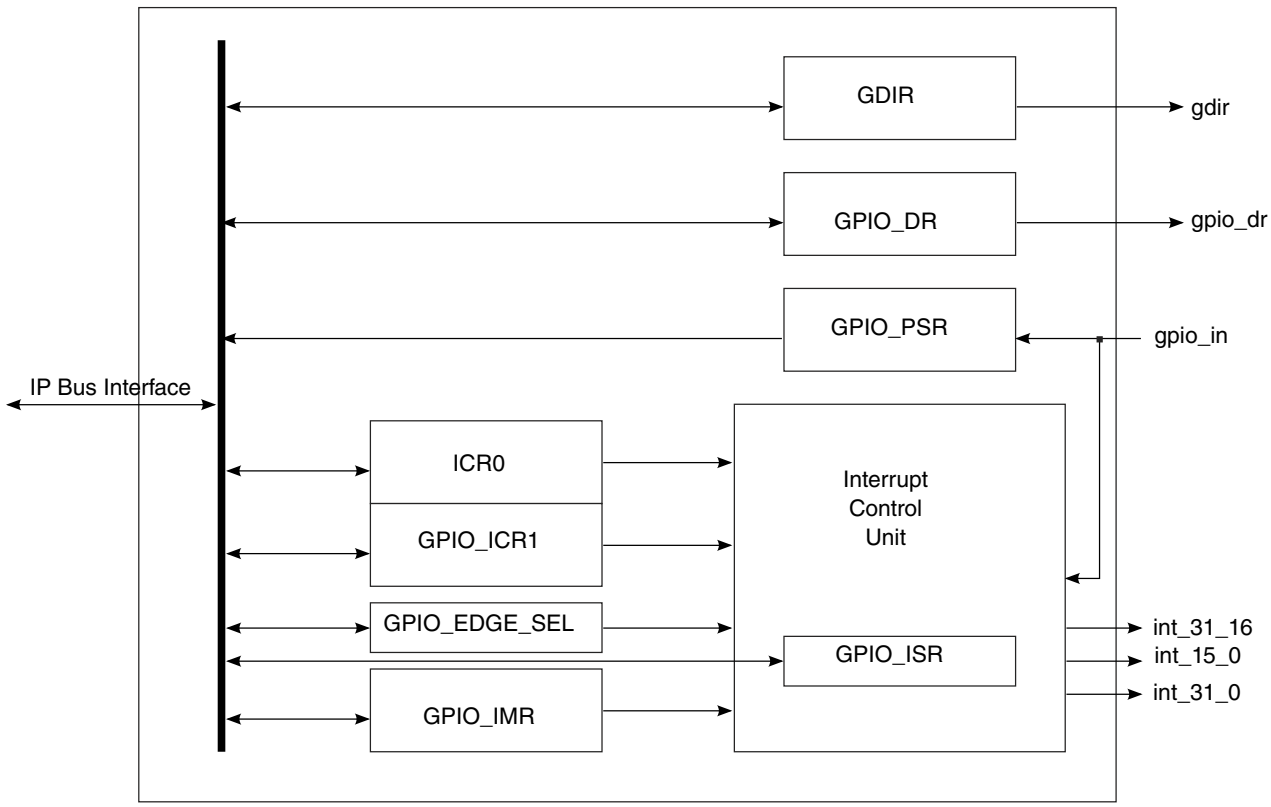


Figure 8-5. GPIO Block Diagram

8.3.1.2 Features

The GPIO includes the following features:

- General purpose input/output logic capabilities:
  - Drives specific data to output using the data register (GPIO_DR)
  - Controls the direction of the signal using the GPIO direction register (GPIO_GDIR)
  - Enables the core to sample the status of the corresponding inputs by reading the pad sample register (GPIO_PSR).
- GPIO interrupt capabilities:
  - Supports up to 32 interrupts

- Identifies interrupt edges
- Generates three active-high interrupts to the SoC interrupt controller

8.3.2 External Signals

The tables found here describe the external signals of GPIO.

Table 8-2. GPIO External Signals

Signal	Description	Pad	Mode	Direction
GPIO1_IO00	GPIO Signal	GPIO1_IO00	ALT0	IO
GPIO1_IO01	GPIO Signal	GPIO1_IO01	ALT0	IO
GPIO1_IO02	GPIO Signal	GPIO1_IO02	ALT0	IO
GPIO1_IO03	GPIO Signal	GPIO1_IO03	ALT0	IO
GPIO1_IO04	GPIO Signal	GPIO1_IO04	ALT0	IO
GPIO1_IO05	GPIO Signal	GPIO1_IO05	ALT0	IO
GPIO1_IO06	GPIO Signal	GPIO1_IO06	ALT0	IO
GPIO1_IO07	GPIO Signal	GPIO1_IO07	ALT0	IO
GPIO1_IO08	GPIO Signal	GPIO1_IO08	ALT0	IO
GPIO1_IO09	GPIO Signal	GPIO1_IO09	ALT0	IO
GPIO1_IO10	GPIO Signal	GPIO1_IO10	ALT0	IO
GPIO1_IO11	GPIO Signal	GPIO1_IO11	ALT0	IO
GPIO1_IO12	GPIO Signal	GPIO1_IO12	ALT0	IO
GPIO1_IO13	GPIO Signal	GPIO1_IO13	ALT0	IO
GPIO1_IO14	GPIO Signal	GPIO1_IO14	ALT0	IO
GPIO1_IO15	GPIO Signal	GPIO1_IO15	ALT0	IO
GPIO2_IO00	GPIO Signal	EPDC1_DATA00	ALT5	IO
GPIO2_IO01	GPIO Signal	EPDC1_DATA01	ALT5	IO
GPIO2_IO02	GPIO Signal	EPDC1_DATA02	ALT5	IO
GPIO2_IO03	GPIO Signal	EPDC1_DATA03	ALT5	IO
GPIO2_IO04	GPIO Signal	EPDC1_DATA04	ALT5	IO
GPIO2_IO05	GPIO Signal	EPDC1_DATA05	ALT5	IO
GPIO2_IO06	GPIO Signal	EPDC1_DATA06	ALT5	IO
GPIO2_IO07	GPIO Signal	EPDC1_DATA07	ALT5	IO
GPIO2_IO08	GPIO Signal	EPDC1_DATA08	ALT5	IO
GPIO2_IO09	GPIO Signal	EPDC1_DATA09	ALT5	IO
GPIO2_IO10	GPIO Signal	EPDC1_DATA10	ALT5	IO
GPIO2_IO11	GPIO Signal	EPDC1_DATA11	ALT5	IO
GPIO2_IO12	GPIO Signal	EPDC1_DATA12	ALT5	IO
GPIO2_IO13	GPIO Signal	EPDC1_DATA13	ALT5	IO
GPIO2_IO14	GPIO Signal	EPDC1_DATA14	ALT5	IO
GPIO2_IO15	GPIO Signal	EPDC1_DATA15	ALT5	IO

Table continues on the next page...

Table 8-2. GPIO External Signals (continued)

Signal	Description	Pad	Mode	Direction
GPIO2_IO16	GPIO Signal	EPDC1_SDCLK	ALT5	IO
GPIO2_IO17	GPIO Signal	EPDC1_SDLE	ALT5	IO
GPIO2_IO18	GPIO Signal	EPDC1_SDOE	ALT5	IO
GPIO2_IO19	GPIO Signal	EPDC1_SDSHR	ALT5	IO
GPIO2_IO20	GPIO Signal	EPDC1_SDCE0	ALT5	IO
GPIO2_IO21	GPIO Signal	EPDC1_SDCE1	ALT5	IO
GPIO2_IO22	GPIO Signal	EPDC1_SDCE2	ALT5	IO
GPIO2_IO23	GPIO Signal	EPDC1_SDCE3	ALT5	IO
GPIO2_IO24	GPIO Signal	EPDC1_GDCLK	ALT5	IO
GPIO2_IO25	GPIO Signal	EPDC1_GDOE	ALT5	IO
GPIO2_IO26	GPIO Signal	EPDC1_GDRL	ALT5	IO
GPIO2_IO27	GPIO Signal	EPDC1_GDSP	ALT5	IO
GPIO2_IO28	GPIO Signal	EPDC1_BDR0	ALT5	IO
GPIO2_IO29	GPIO Signal	EPDC1_BDR1	ALT5	IO
GPIO2_IO30	GPIO Signal	EPDC1_PWRCOM	ALT5	IO
GPIO2_IO31	GPIO Signal	EPDC1_PWRSTAT	ALT5	IO
GPIO3_IO00	GPIO Signal	LCD1_CLK	ALT5	IO
GPIO3_IO01	GPIO Signal	LCD1_ENABLE	ALT5	IO
GPIO3_IO02	GPIO Signal	LCD1_HSYNC	ALT5	IO
GPIO3_IO03	GPIO Signal	LCD1_VSYNC	ALT5	IO
GPIO3_IO04	GPIO Signal	LCD1_RESET	ALT5	IO
GPIO3_IO05	GPIO Signal	LCD1_DATA00	ALT5	IO
GPIO3_IO06	GPIO Signal	LCD1_DATA01	ALT5	IO
GPIO3_IO07	GPIO Signal	LCD1_DATA02	ALT5	IO
GPIO3_IO08	GPIO Signal	LCD1_DATA03	ALT5	IO
GPIO3_IO09	GPIO Signal	LCD1_DATA04	ALT5	IO
GPIO3_IO10	GPIO Signal	LCD1_DATA05	ALT5	IO
GPIO3_IO11	GPIO Signal	LCD1_DATA06	ALT5	IO
GPIO3_IO12	GPIO Signal	LCD1_DATA07	ALT5	IO
GPIO3_IO13	GPIO Signal	LCD1_DATA08	ALT5	IO
GPIO3_IO14	GPIO Signal	LCD1_DATA09	ALT5	IO
GPIO3_IO15	GPIO Signal	LCD1_DATA10	ALT5	IO
GPIO3_IO16	GPIO Signal	LCD1_DATA11	ALT5	IO
GPIO3_IO17	GPIO Signal	LCD1_DATA12	ALT5	IO
GPIO3_IO18	GPIO Signal	LCD1_DATA13	ALT5	IO
GPIO3_IO19	GPIO Signal	LCD1_DATA14	ALT5	IO
GPIO3_IO20	GPIO Signal	LCD1_DATA15	ALT5	IO
GPIO3_IO21	GPIO Signal	LCD1_DATA16	ALT5	IO
GPIO3_IO22	GPIO Signal	LCD1_DATA17	ALT5	IO

Table continues on the next page...

Table 8-2. GPIO External Signals (continued)

Signal	Description	Pad	Mode	Direction
GPIO3_IO23	GPIO Signal	LCD1_DATA18	ALT5	IO
GPIO3_IO24	GPIO Signal	LCD1_DATA19	ALT5	IO
GPIO3_IO25	GPIO Signal	LCD1_DATA20	ALT5	IO
GPIO3_IO26	GPIO Signal	LCD1_DATA21	ALT5	IO
GPIO3_IO27	GPIO Signal	LCD1_DATA22	ALT5	IO
GPIO3_IO28	GPIO Signal	LCD1_DATA23	ALT5	IO
GPIO4_IO00	GPIO Signal	UART1_RXD	ALT5	IO
GPIO4_IO01	GPIO Signal	UART1_TXD	ALT5	IO
GPIO4_IO02	GPIO Signal	UART2_RXD	ALT5	IO
GPIO4_IO03	GPIO Signal	UART2_TXD	ALT5	IO
GPIO4_IO04	GPIO Signal	UART3_RXD	ALT5	IO
GPIO4_IO05	GPIO Signal	UART3_TXD	ALT5	IO
GPIO4_IO06	GPIO Signal	UART3_RTS	ALT5	IO
GPIO4_IO07	GPIO Signal	UART3_CTS	ALT5	IO
GPIO4_IO08	GPIO Signal	I2C1_SCL	ALT5	IO
GPIO4_IO09	GPIO Signal	I2C1_SDA	ALT5	IO
GPIO4_IO10	GPIO Signal	I2C2_SCL	ALT5	IO
GPIO4_IO11	GPIO Signal	I2C2_SDA	ALT5	IO
GPIO4_IO12	GPIO Signal	I2C3_SCL	ALT5	IO
GPIO4_IO13	GPIO Signal	I2C3_SDA	ALT5	IO
GPIO4_IO14	GPIO Signal	I2C4_SCL	ALT5	IO
GPIO4_IO15	GPIO Signal	I2C4_SDA	ALT5	IO
GPIO4_IO16	GPIO Signal	ECSPI1_SCLK	ALT5	IO
GPIO4_IO17	GPIO Signal	ECSPI1_MOSI	ALT5	IO
GPIO4_IO18	GPIO Signal	ECSPI1_MISO	ALT5	IO
GPIO4_IO19	GPIO Signal	ECSPI1_SS0	ALT5	IO
GPIO4_IO20	GPIO Signal	ECSPI2_SCLK	ALT5	IO
GPIO4_IO21	GPIO Signal	ECSPI2_MOSI	ALT5	IO
GPIO4_IO22	GPIO Signal	ECSPI2_MISO	ALT5	IO
GPIO4_IO23	GPIO Signal	ECSPI2_SS0	ALT5	IO
GPIO5_IO00	GPIO Signal	SD1_CD_B	ALT5	IO
GPIO5_IO01	GPIO Signal	SD1_WP	ALT5	IO
GPIO5_IO02	GPIO Signal	SD1_RESET_B	ALT5	IO
GPIO5_IO03	GPIO Signal	SD1_CLK	ALT5	IO
GPIO5_IO04	GPIO Signal	SD1_CMD	ALT5	IO
GPIO5_IO05	GPIO Signal	SD1_DATA0	ALT5	IO
GPIO5_IO06	GPIO Signal	SD1_DATA1	ALT5	IO
GPIO5_IO07	GPIO Signal	SD1_DATA2	ALT5	IO
GPIO5_IO08	GPIO Signal	SD1_DATA3	ALT5	IO

Table continues on the next page...

Table 8-2. GPIO External Signals (continued)

Signal	Description	Pad	Mode	Direction
GPIO5_IO09	GPIO Signal	SD2_CD_B	ALT5	IO
GPIO5_IO10	GPIO Signal	SD2_WP	ALT5	IO
GPIO5_IO11	GPIO Signal	SD2_RESET_B	ALT5	IO
GPIO5_IO12	GPIO Signal	SD2_CLK	ALT5	IO
GPIO5_IO13	GPIO Signal	SD2_CMD	ALT5	IO
GPIO5_IO14	GPIO Signal	SD2_DATA0	ALT5	IO
GPIO5_IO15	GPIO Signal	SD2_DATA1	ALT5	IO
GPIO5_IO16	GPIO Signal	SD2_DATA2	ALT5	IO
GPIO5_IO17	GPIO Signal	SD2_DATA3	ALT5	IO
GPIO6_IO00	GPIO Signal	SD3_CLK	ALT5	IO
GPIO6_IO01	GPIO Signal	SD3_CMD	ALT5	IO
GPIO6_IO02	GPIO Signal	SD3_DATA0	ALT5	IO
GPIO6_IO03	GPIO Signal	SD3_DATA1	ALT5	IO
GPIO6_IO04	GPIO Signal	SD3_DATA2	ALT5	IO
GPIO6_IO05	GPIO Signal	SD3_DATA3	ALT5	IO
GPIO6_IO06	GPIO Signal	SD3_DATA4	ALT5	IO
GPIO6_IO07	GPIO Signal	SD3_DATA5	ALT5	IO
GPIO6_IO08	GPIO Signal	SD3_DATA6	ALT5	IO
GPIO6_IO09	GPIO Signal	SD3_DATA7	ALT5	IO
GPIO6_IO10	GPIO Signal	SD3_STROBE	ALT5	IO
GPIO6_IO11	GPIO Signal	SD3_RESET_B	ALT5	IO
GPIO6_IO12	GPIO Signal	SAI1_RXD	ALT5	IO
GPIO6_IO13	GPIO Signal	SAI1_TXC	ALT5	IO
GPIO6_IO14	GPIO Signal	SAI1_TXFS	ALT5	IO
GPIO6_IO15	GPIO Signal	SAI1_TXD	ALT5	IO
GPIO6_IO16	GPIO Signal	SAI1_RXFS	ALT5	IO
GPIO6_IO17	GPIO Signal	SAI1_RXC	ALT5	IO
GPIO6_IO18	GPIO Signal	SAI1_MCLK	ALT5	IO
GPIO6_IO19	GPIO Signal	SAI2_TXFS	ALT5	IO
GPIO6_IO20	GPIO Signal	SAI2_TXC	ALT5	IO
GPIO6_IO21	GPIO Signal	SAI2_RXD	ALT5	IO
GPIO6_IO22	GPIO Signal	SAI2_TXD	ALT5	IO
GPIO7_IO00	GPIO Signal	ENET1_RDATA0	ALT5	IO
GPIO7_IO01	GPIO Signal	ENET1_RDATA1	ALT5	IO
GPIO7_IO02	GPIO Signal	ENET1_RDATA2	ALT5	IO
GPIO7_IO03	GPIO Signal	ENET1_RDATA3	ALT5	IO
GPIO7_IO04	GPIO Signal	ENET1_RX_CTL	ALT5	IO
GPIO7_IO05	GPIO Signal	ENET1_RXC	ALT5	IO
GPIO7_IO06	GPIO Signal	ENET1_TDATA0	ALT5	IO

Table continues on the next page...

Table 8-2. GPIO External Signals (continued)

Signal	Description	Pad	Mode	Direction
GPIO7_IO07	GPIO Signal	ENET1_TDATA1	ALT5	IO
GPIO7_IO08	GPIO Signal	ENET1_TDATA2	ALT5	IO
GPIO7_IO09	GPIO Signal	ENET1_TDATA3	ALT5	IO
GPIO7_IO10	GPIO Signal	ENET1_TX_CTL	ALT5	IO
GPIO7_IO11	GPIO Signal	ENET1_TXC	ALT5	IO
GPIO7_IO12	GPIO Signal	ENET1_TX_CLK	ALT5	IO
GPIO7_IO13	GPIO Signal	ENET1_RX_CLK	ALT5	IO
GPIO7_IO14	GPIO Signal	ENET1_CRS	ALT5	IO
GPIO7_IO15	GPIO Signal	ENET1_COL	ALT5	IO

8.3.3 Clocks

The table found here describes the clock sources for GPIO.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 8-3. GPIO Clocks

Clock name	Clock Root	Description
ipg_clk_s	ipg_clk_root	Peripheral access clock

8.3.4 GPIO Functional Description

This section provides a complete functional description of the block.

8.3.4.1 GPIO Function

A GPIO signal can operate as a general-purpose input/output when the IOMUX is set to GPIO mode. Each GPIO signal may be independently configured as either an input or an output using the GPIO direction register (GPIO_GDIR).

When configured as an output (GPIO_GDIR bit = 1), the value in the data bit in the GPIO data register (GPIO_DR) is driven on the corresponding GPIO line. When a signal is configured as an input (GPIO_GDIR bit = 0), the state of the input can be read from the corresponding GPIO_PSR bit.

8.3.4.2 GPIO Programming

8.3.4.2.1 GPIO Read Mode

The programming sequence for reading input signals should be as follows:

1. Configure IOMUX to select GPIO mode (Via IOMUX Controller (IOMUXC)).
2. Configure GPIO direction register to input (GPIO_GDIR[GDIR] set to 0b).
3. Read value from data reg*ister*/pad status register.

A pseudocode description to read [input3:input0] values is as follows:

```
// SET INPUTS TO GPIO MODE.
write sw_mux_ctl_<input0>_<input1>_<input2>_<input3>, 32'h00000000
// SET GDIR TO INPUT.
write GDIR[31:4,input3_bit, input2_bit, input1_bit, input0_bit,] 32'hxxxxxxx0
// READ INPUT VALUE FROM DR.
read DR
// READ INPUT VALUE FROM PSR.
read PSR
```

NOTE

While the GPIO direction is set to input (GPIO_GDIR = 0), a read access to GPIO_DR does not return GPIO_DR data. Instead, it returns the GPIO_PSR data, which is the corresponding input signal value.

8.3.4.2.2 GPIO Write Mode

The programming sequence for driving output signals should be as follows:

1. Configure IOMUX to select GPIO mode (Via IOMUXC), also enable SION if need to read loopback pad value through PSR
2. Configure GPIO direction register to output (GPIO_GDIR[GDIR] set to 1b).
3. Write value to data register (GPIO_DR).

A pseudocode description to drive 4'b0101 on [output3:output0] is as follows:

```
// SET PADS TO GPIO MODE VIA IOMUX.
write sw_mux_ctl_pad_<output[0-3]>.mux_mode, <GPIO_MUX_MODE>
// Enable loopback so we can capture pad value into PSR in output mode
write sw_mux_ctl_pad_<output[0-3]>.sion, 1
// SET GDIR=1 TO OUTPUT BITS.
write GDIR[31:4,output3_bit,output2_bit, output1_bit, output0_bit,] 32'hxxxxxxxF
// WRITE OUTPUT VALUE=4'b0101 TO DR.
write DR, 32'hxxxxxxx5
// READ OUTPUT VALUE FROM PSR ONLY.
read_cmp PSR, 32'hxxxxxxx5
```

8.3.4.3 Interrupt Control Unit

In addition to the general-purpose input/output function, the edge-detect logic in the GPIO peripheral reflects whether a transition has occurred on a given GPIO signal that is configured as an input (GDIR bit = 0). The interrupt control registers (GPIO_ICR1 and GPIO_ICR2) may be used to independently configure the interrupt condition of each input signal (low-to-high transition, high-to-low transition, low, or high). For information about GPIO_ICR1 and GPIO_ICR2 settings, see GPIO Memory Map/Register Definition.

The interrupt control unit is built of 32 interrupt control subunits, where each subunit handles a single interrupt line.

8.3.5 GPIO Memory Map/Register Definition

There are eight 32-bit GPIO registers. All registers are accessible from the IP interface. Only 32-bit access is supported.

The GPIO memory map is shown in the following table.

GPIO memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3020_0000	GPIO data register (GPIO1_DR)	32	R/W	0000_0000h	8.3.5.1/ 2105
3020_0004	GPIO direction register (GPIO1_GDIR)	32	R/W	0000_0000h	8.3.5.2/ 2106
3020_0008	GPIO pad status register (GPIO1_PSR)	32	R	0000_0000h	8.3.5.3/ 2106
3020_000C	GPIO interrupt configuration register1 (GPIO1_ICR1)	32	R/W	0000_0000h	8.3.5.4/ 2107
3020_0010	GPIO interrupt configuration register2 (GPIO1_ICR2)	32	R/W	0000_0000h	8.3.5.5/ 2111
3020_0014	GPIO interrupt mask register (GPIO1_IMR)	32	R/W	0000_0000h	8.3.5.6/ 2114
3020_0018	GPIO interrupt status register (GPIO1_ISR)	32	w1c	0000_0000h	8.3.5.7/ 2115
3020_001C	GPIO edge select register (GPIO1_EDGE_SEL)	32	R/W	0000_0000h	8.3.5.8/ 2116

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3021_0000	GPIO data register (GPIO2_DR)	32	R/W	0000_0000h	8.3.5.1/ 2105
3021_0004	GPIO direction register (GPIO2_GDIR)	32	R/W	0000_0000h	8.3.5.2/ 2106
3021_0008	GPIO pad status register (GPIO2_PSR)	32	R	0000_0000h	8.3.5.3/ 2106
3021_000C	GPIO interrupt configuration register1 (GPIO2_ICR1)	32	R/W	0000_0000h	8.3.5.4/ 2107
3021_0010	GPIO interrupt configuration register2 (GPIO2_ICR2)	32	R/W	0000_0000h	8.3.5.5/ 2111
3021_0014	GPIO interrupt mask register (GPIO2_IMR)	32	R/W	0000_0000h	8.3.5.6/ 2114
3021_0018	GPIO interrupt status register (GPIO2_ISR)	32	w1c	0000_0000h	8.3.5.7/ 2115
3021_001C	GPIO edge select register (GPIO2_EDGE_SEL)	32	R/W	0000_0000h	8.3.5.8/ 2116
3022_0000	GPIO data register (GPIO3_DR)	32	R/W	0000_0000h	8.3.5.1/ 2105
3022_0004	GPIO direction register (GPIO3_GDIR)	32	R/W	0000_0000h	8.3.5.2/ 2106
3022_0008	GPIO pad status register (GPIO3_PSR)	32	R	0000_0000h	8.3.5.3/ 2106
3022_000C	GPIO interrupt configuration register1 (GPIO3_ICR1)	32	R/W	0000_0000h	8.3.5.4/ 2107
3022_0010	GPIO interrupt configuration register2 (GPIO3_ICR2)	32	R/W	0000_0000h	8.3.5.5/ 2111
3022_0014	GPIO interrupt mask register (GPIO3_IMR)	32	R/W	0000_0000h	8.3.5.6/ 2114
3022_0018	GPIO interrupt status register (GPIO3_ISR)	32	w1c	0000_0000h	8.3.5.7/ 2115
3022_001C	GPIO edge select register (GPIO3_EDGE_SEL)	32	R/W	0000_0000h	8.3.5.8/ 2116
3023_0000	GPIO data register (GPIO4_DR)	32	R/W	0000_0000h	8.3.5.1/ 2105
3023_0004	GPIO direction register (GPIO4_GDIR)	32	R/W	0000_0000h	8.3.5.2/ 2106
3023_0008	GPIO pad status register (GPIO4_PSR)	32	R	0000_0000h	8.3.5.3/ 2106
3023_000C	GPIO interrupt configuration register1 (GPIO4_ICR1)	32	R/W	0000_0000h	8.3.5.4/ 2107
3023_0010	GPIO interrupt configuration register2 (GPIO4_ICR2)	32	R/W	0000_0000h	8.3.5.5/ 2111
3023_0014	GPIO interrupt mask register (GPIO4_IMR)	32	R/W	0000_0000h	8.3.5.6/ 2114

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3023_0018	GPIO interrupt status register (GPIO4_ISR)	32	w1c	0000_0000h	8.3.5.7/ 2115
3023_001C	GPIO edge select register (GPIO4_EDGE_SEL)	32	R/W	0000_0000h	8.3.5.8/ 2116
3024_0000	GPIO data register (GPIO5_DR)	32	R/W	0000_0000h	8.3.5.1/ 2105
3024_0004	GPIO direction register (GPIO5_GDIR)	32	R/W	0000_0000h	8.3.5.2/ 2106
3024_0008	GPIO pad status register (GPIO5_PSR)	32	R	0000_0000h	8.3.5.3/ 2106
3024_000C	GPIO interrupt configuration register1 (GPIO5_ICR1)	32	R/W	0000_0000h	8.3.5.4/ 2107
3024_0010	GPIO interrupt configuration register2 (GPIO5_ICR2)	32	R/W	0000_0000h	8.3.5.5/ 2111
3024_0014	GPIO interrupt mask register (GPIO5_IMR)	32	R/W	0000_0000h	8.3.5.6/ 2114
3024_0018	GPIO interrupt status register (GPIO5_ISR)	32	w1c	0000_0000h	8.3.5.7/ 2115
3024_001C	GPIO edge select register (GPIO5_EDGE_SEL)	32	R/W	0000_0000h	8.3.5.8/ 2116
3025_0000	GPIO data register (GPIO6_DR)	32	R/W	0000_0000h	8.3.5.1/ 2105
3025_0004	GPIO direction register (GPIO6_GDIR)	32	R/W	0000_0000h	8.3.5.2/ 2106
3025_0008	GPIO pad status register (GPIO6_PSR)	32	R	0000_0000h	8.3.5.3/ 2106
3025_000C	GPIO interrupt configuration register1 (GPIO6_ICR1)	32	R/W	0000_0000h	8.3.5.4/ 2107
3025_0010	GPIO interrupt configuration register2 (GPIO6_ICR2)	32	R/W	0000_0000h	8.3.5.5/ 2111
3025_0014	GPIO interrupt mask register (GPIO6_IMR)	32	R/W	0000_0000h	8.3.5.6/ 2114
3025_0018	GPIO interrupt status register (GPIO6_ISR)	32	w1c	0000_0000h	8.3.5.7/ 2115
3025_001C	GPIO edge select register (GPIO6_EDGE_SEL)	32	R/W	0000_0000h	8.3.5.8/ 2116
3026_0000	GPIO data register (GPIO7_DR)	32	R/W	0000_0000h	8.3.5.1/ 2105
3026_0004	GPIO direction register (GPIO7_GDIR)	32	R/W	0000_0000h	8.3.5.2/ 2106
3026_0008	GPIO pad status register (GPIO7_PSR)	32	R	0000_0000h	8.3.5.3/ 2106
3026_000C	GPIO interrupt configuration register1 (GPIO7_ICR1)	32	R/W	0000_0000h	8.3.5.4/ 2107

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3026_0010	GPIO interrupt configuration register2 (GPIO7_ICR2)	32	R/W	0000_0000h	8.3.5.5/ 2111
3026_0014	GPIO interrupt mask register (GPIO7_IMR)	32	R/W	0000_0000h	8.3.5.6/ 2114
3026_0018	GPIO interrupt status register (GPIO7_ISR)	32	w1c	0000_0000h	8.3.5.7/ 2115
3026_001C	GPIO edge select register (GPIO7_EDGE_SEL)	32	R/W	0000_0000h	8.3.5.8/ 2116

8.3.5.1 GPIO data register (GPIOx_DR)

The 32-bit GPIO_DR register stores data that is ready to be driven to the output lines. If the IOMUXC is in GPIO mode and a given GPIO direction bit is set, then the corresponding DR bit is driven to the output. If a given GPIO direction bit is cleared, then a read of GPIO_DR reflects the value of the corresponding signal.Two wait states are required in read access for synchronization.

The results of a read of a DR bit depends on the IOMUXC input mode settings and the corresponding GDIR bit as follows:

- If GDIR[*n*] is set and IOMUXC input mode is GPIO, then reading DR[*n*] returns the contents of DR[*n*].
- If GDIR[*n*] is cleared and IOMUXC input mode is GPIO, then reading DR[*n*] returns the corresponding input signal's value.
- If GDIR[*n*] is set and IOMUXC input mode is not GPIO, then reading DR[*n*] returns the contents of DR[*n*].
- If GDIR[*n*] is cleared and IOMUXC input mode is not GPIO, then reading DR[*n*] always returns zero.

Address: Base address + 0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_DR field descriptions

Field	Description
DR	Data bits. This register defines the value of the GPIO output when the signal is configured as an output (GDIR[n]=1). Writes to this register are stored in a register. Reading GPIO_DR returns the value stored in the register if the signal is configured as an output (GDIR[n]=1), or the input signal's value if configured as an input (GDIR[n]=0).

GPIOx_DR field descriptions (continued)

Field	Description
	NOTE: The I/O multiplexer must be configured to GPIO mode for the GPIO_DR value to connect with the signal. Reading the data register with the input path disabled always returns a zero value.

8.3.5.2 GPIO direction register (GPIOx_GDIR)

GPIO_GDIR functions as direction control when the IOMUXC is in GPIO mode. Each bit specifies the direction of a one-bit signal. The mapping of each DIR bit to a corresponding SoC signal is determined by the SoC's pin assignment and the IOMUX table. For more details consult the IOMUXC chapter.

Address: Base address + 4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GDIR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_GDIR field descriptions

Field	Description
GDIR	GPIO direction bits. Bit n of this register defines the direction of the GPIO[n] signal. **NOTE:** GPIO_GDIR affects only the direction of the I/O signal when the corresponding bit in the I/O MUX is configured for GPIO. 0 **INPUT** — GPIO is configured as input. 1 **OUTPUT** — GPIO is configured as output.

8.3.5.3 GPIO pad status register (GPIOx_PSR)

GPIO_PSR is a read-only register. Each bit stores the value of the corresponding input signal (as configured in the IOMUX). This register is clocked with the ipg_clk_s clock, meaning that the input signal is sampled only when accessing this location. Two wait states are required any time this register is accessed for synchronization.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PSR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_PSR field descriptions

Field	Description
PSR	GPIO pad status bits (status bits). Reading GPIO_PSR returns the state of the corresponding input signal. Settings: **NOTE:** The IOMUXC must be configured to GPIO mode for GPIO_PSR to reflect the state of the corresponding signal.

8.3.5.4 GPIO interrupt configuration register1 (GPIOx_ICR1)

GPIO_ICR1 contains 16 two-bit fields, where each field specifies the interrupt configuration for a different input signal.

Address: Base address + Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ICR15		ICR14		ICR13		ICR12		ICR11		ICR10		ICR9		ICR8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ICR7		ICR6		ICR5		ICR4		ICR3		ICR2		ICR1		ICR0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_ICR1 field descriptions

Field	Description
31–30 ICR15	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 15. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
29–28 ICR14	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 14. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.

Table continues on the next page...

GPIOx_ICR1 field descriptions (continued)

Field	Description
27–26 ICR13	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 13. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
25–24 ICR12	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 12. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
23–22 ICR11	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 11. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
21–20 ICR10	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 10. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
19–18 ICR9	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 9. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.

Table continues on the next page...

GPIOx_ICR1 field descriptions (continued)

Field	Description
17–16 ICR8	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 8. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
15–14 ICR7	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 7. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
13–12 ICR6	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 6. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
11–10 ICR5	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 5. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
9–8 ICR4	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 4. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.

Table continues on the next page...

GPIOx_ICR1 field descriptions (continued)

Field	Description
7–6 ICR3	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 3. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
5–4 ICR2	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 2. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
3–2 ICR1	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 1. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
ICR0	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 0. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.

8.3.5.5 GPIO interrupt configuration register2 (GPIOx_ICR2)

GPIO_ICR2 contains 16 two-bit fields, where each field specifies the interrupt configuration for a different input signal.

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ICR31		ICR30		ICR29		ICR28		ICR27		ICR26		ICR25		ICR24	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ICR23		ICR22		ICR21		ICR20		ICR19		ICR18		ICR17		ICR16	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_ICR2 field descriptions

Field	Description
31–30 ICR31	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 31. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
29–28 ICR30	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 30. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
27–26 ICR29	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 29. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.

Table continues on the next page...

GPIOx_ICR2 field descriptions (continued)

Field	Description
25–24 ICR28	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 28. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
23–22 ICR27	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 27. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
21–20 ICR26	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 26. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
19–18 ICR25	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 25. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
17–16 ICR24	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 24. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.

Table continues on the next page...

GPIOx_ICR2 field descriptions (continued)

Field	Description
15–14 ICR23	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 23. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
13–12 ICR22	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 22. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
11–10 ICR21	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 21. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
9–8 ICR20	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 20. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
7–6 ICR19	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 19. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.

Table continues on the next page...

GPIOx_ICR2 field descriptions (continued)

Field	Description
5–4 ICR18	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 18. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
3–2 ICR17	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 17. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.
ICR16	Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 16. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 **LOW_LEVEL** — Interrupt n is low-level sensitive. 01 **HIGH_LEVEL** — Interrupt n is high-level sensitive. 10 **RISING_EDGE** — Interrupt n is rising-edge sensitive. 11 **FALLING_EDGE** — Interrupt n is falling-edge sensitive.

8.3.5.6 GPIO interrupt mask register (GPIOx_IMR)

GPIO_IMR contains masking bits for each interrupt line.

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_IMR field descriptions

Field	Description
IMR	Interrupt Mask bits. This register is used to enable or disable the interrupt function on each of the 32 GPIO signals. Settings:

GPIOx_IMR field descriptions (continued)

Field	Description
	Bit IMR[n] (n=0...31) controls interrupt n as follows: 0 **UNMASKED** — Interrupt n is disabled. 1 **MASKED** — Interrupt n is enabled.

8.3.5.7 GPIO interrupt status register (GPIOx_ISR)

The GPIO_ISR functions as an interrupt status indicator. Each bit indicates whether an interrupt condition has been met for the corresponding input signal. When an interrupt condition is met (as determined by the corresponding interrupt condition register field), the corresponding bit in this register is set. Two wait states are required in read access for synchronization. One wait state is required for reset.

Address: Base address + 18h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISR																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_ISR field descriptions

Field	Description
ISR	Interrupt status bits - Bit n of this register is asserted (active high) when the active condition (as determined by the corresponding ICR bit) is detected on the GPIO input and is waiting for service. The value of this register is independent of the value in GPIO_IMR. When the active condition has been detected, the corresponding bit remains set until cleared by software. Status flags are cleared by writing a 1 to the corresponding bit position.

8.3.5.8 GPIO edge select register (GPIOx_EDGE_SEL)

GPIO_EDGE_SEL may be used to override the ICR registers' configuration. If the GPIO_EDGE_SEL bit is set, then a rising edge or falling edge in the corresponding signal generates an interrupt. This register provides backward compatibility. On reset all bits are cleared (ICR is not overridden).

Address: Base address + 1Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPIO_EDGE_SEL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_EDGE_SEL field descriptions

Field	Description
GPIO_EDGE_SEL	Edge select. When GPIO_EDGE_SEL[*n*] is set, the GPIO disregards the ICR[*n*] setting, and detects any edge on the corresponding input signal.

Chapter 9
External Memory

9.1 External Memory Controllers

9.1.1 Overview

This chip has these external memory interfaces and controllers:

- Multi-Mode DDR Controller (MMDC)
- EIM-PSRAM/NOR flash controller

9.1.2 DDR controller (DDRC) overview and feature summary

The DDRC module is a DDR controller that supports several types of DDR memories.

Table 9-1. DDRC feature summary

Feature	Description
Supported standards	• DDR3/LPDDR2/LPDDR3 1ch x32

DDR interface	• Density of 256 MB - 2 GB
	  • Column size of 8-12 bits
	  • Row size of 11-16 bits
	• Supports burst length of 8 (aligned) for DDR3 and burst lengths of 4 for LPDDR2.

DDR performance	• Dynamic scheduling to optimize the bandwidth and latency. • Delayed writes for optimum bandwidth and latency. • For maximum SDRAM efficiency, commands are executed out of order. • Hardware configurable and software programmable QOS support. • Control options to avoid the starvation of lower priorities. • Write combine to allow multiple writes to the same address to be combined into a single write to the SDRAM. • Paging policy is selectable by the configuration registers. • Flexible address mapper logic to allow application-specific mapping of row, column, bank, and rank bits. • User-selectable refresh control options.

Table continues on the next page...

Table 9-1. DDRC feature summary (continued)

Feature	Description
AXI interface	• AXI bus compliant with a glueless interface to PL301 AXI network interconnect.
DDR calibration and delay-lines	• All calibrations can be done automatically by hardware or manually by software. • ZQ calibration for external DDR device (in DDR3 through the ZQ calibration command and in LPDDR2 through the MRW command).
DDR general	• Configurable timing parameters. • Configurable refresh scheme. • Supports dynamic voltage, frequency change, and low-power mode entry through the hardware negotiation with the system (req/ack handshake). • Suppors automatic self-refresh and power down entry and exit. • Supports fast and slow precharge power down in DDR3. • Supports various ODT control schemes. • Assertion/Deassertion of the ODT control per read or write accesses and for active or passive CS. • Supports MRW and MRR commands for LPDDR2. • Software control for moving to derated timing parameters and derated refresh rate according to the temperature variation.

9.1.3 EIM-PSRAM/NOR flash controller overview

EIM is an external interface module that manages the interface to external chip devices, including the generation of chip selects, clocks, and control for external peripherals and memory.

It provides asynchronous and synchronous access to devices with an SRAM-like interface.

9.1.3.1 EIM features

- Up to four (software configurable) chip selects for external devices
  - Flexible address decoding; each chip-select memory space is determined separately according to the GPR bits in IOMUXC.
  - Maximum supported density is 128 MB by default (AUS bit is cleared). When the AUS bit is set, the maximum supported density is 32 MB.
- Selectable write protection for each chip select
- Programmable wait-state generator for each chip select, for write and read accesses separately
- Asynchronous accesses with programmable setup and hold times for control signals
- Independent synchronous memory burst write mode support for PSRAM and NOR-flash memories (CellularRAM™ from Micron, Infineon, and Cypress, OneNAND™ and utRAM™ from Samsung, and COSMORAM™from Toshiba)

- Supports NAND-flash devices with a NOR-flash interfaces - OneNAND™ (Samsung)
- Independent programmable variable/fix latency support for read and write synchronous (burst) mode
- Support for little-endian operation
- ARM AXI slave interface accesses are only handled in parallel for single AXI ID transactions
- External interrupt support using the RDY_INT signal function as an external interrupt pin
- Boot from external device support according to boot signals, using the RDY_INT signal
  - RDY signal support assertion after reset
  - INT signal support assertion after reset for OneNAND™ (Samsung) device
  - Supports little-endian mode only

9.1.3.2 EIM boot scenarios

EIM allows booting from NOR-flash devices. To select the NOR flash as the boot source, use either the boot mode and configuration GPIO pins or the internal boot-related fuses.

See Sytem Boot for more information.

9.1.3.3 EIM boot configuration

This table shows the EIM boot configuration:

Table 9-2. EIM boot configuration

EIM_BOOT_CFG bus	EIM affected bits	EIM register
12	NUM16_BYP_GRANT	CS0GCR2
11	DSZ[2]	CS0GCR1
10	AUS	CS0GCR1
[9:8]	CSREC[2:1]	CS0GCR1
[7:5]	RWSC[4:2]	CS0GCR1
	WWSC[4:2]	CS0WCR
4	ERRST	WCR
3	RAL	CS0RCR1
	WAL	CS0WCR
2	MUM	CS0GCR1
	OEA[1]	CS0RCR1
[1:0]	DSZ[1:0]	CS0GCR1

9.1.3.4 OneNAND requirements

By default, the Ready/Busy pin is not in use, perform these steps to configure the OneNAND:

- Poll the device to see whether it is ready; software performs a read from the device.
- Connect the Ready/Busy signal of the OneNAND device to any GPIO pin and use it as an interrupt that indicates the OneNAND device ready state.

9.2 DDR Controller (DDRC)

9.2.1 Introduction

The DDR Memory Controller (DDRMC) combined with DFI-compatible PHY is a complete memory interface solution for DDR memory subsystems. The DDRMC is a flexible and advanced solution for ASIC and System-on-Chip (SoC) designers who need very low power while achieving industry leading high-efficiency, low- latency and high-performance from their memory interface.

Supported SDRAM types include:

- DDR3
- LPDDR2
- LPDDR3

The DDRMC receives transactions from the SoC core. These transactions are queued internally and scheduled for access to the SDRAM while satisfying SDRAM protocol timing requirements, transaction priorities, and dependencies between the transactions. The DDRMC in turn issues commands on the DFI interface to the PHY module, which launches and captures data to and from the SDRAM.

9.2.2 General Overview

The DDRMC and the combine create a complete solution for connecting an SoC application bus to DDR memory devices. The combined DDRMC and solution enables the highest DDR performance and bandwidth, with the DDRMC initiating DDR commands to transfer data with the maximum efficiency.

The DDRMC SoC application bus interface supports an AMBA 3 AXI interface. The DDRMC has flexible address mapper logic to allow application-specific mapping of row, column, bank, bank group, and rank bits. The DDRMC includes a DFI interface that supports DDR3 (JEDEC JESD79-3*), LPDDR2 (JEDECJESD209-2*), and LPDDR3 (JEDEC 209-3*) standards.

9.2.2.1 Block diagram

The DDRMC converts system bus transactions into memory commands on the DFI interface that are compliant with the DDR protocols. The following figure is the block diagram of the DDRMC .

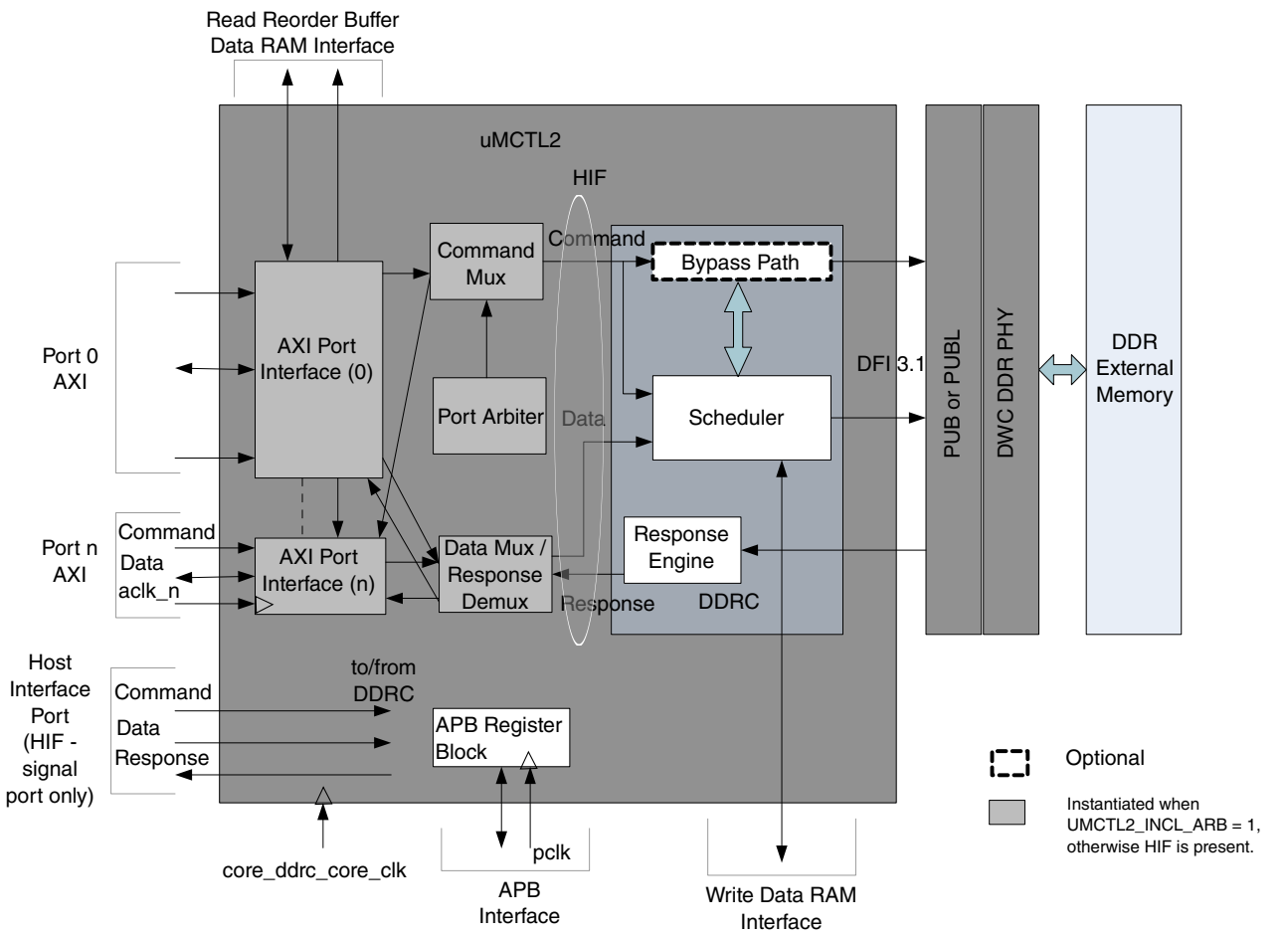


Figure 9-1. DDRMC Block Diagram

The DDRMC contains the following main architectural components:

- The AXI Port Interface (XPI) block: This block provides the interface to the application ports. It provides bus protocol handling, data buffering and reordering for read data, data bus size conversion (upsizing or downsizing), and memory burst address alignment.

- The Port Arbiter (PA) block: This block provides latency sensitive, priority based arbitration between the addresses issued by the XPIs (by the ports).
- The DDR Controller (DDRC) block: This block contains a logical CAM (Content Addressable Memory), which can be synthesized using standard cells. This holds information on the commands, which is used by the scheduling algorithms to optimally schedule commands to be sent to the PHY, based on priority, bank / rank status and DDR timing constraints. A bypass path is also provided (optionally).
- The APB Register Block: This block contains the software accessible registers.

9.2.2.2 Clocks

All application port clocks (ACLK_N) are independently configurable as asynchronous or synchronous with respect to the main DDRMC clock (CORE_DDRC_CORE_CLK). The APB interface clock (PCLK) is normally asynchronous to the CORE_DDRC_CORE_CLK. The interfaces that needs to be asynchronously configured, incur latency and area increase due to the additional clock domain crossing circuitry.

NOTE

PCLK frequency must be the same or less than CORE_DDRC_CORE_CLK frequency.

9.2.3 Features

The DDRMC supports the following features:

- Complete, integrated, single-vendor DDR3, LPDDR2, LPDDR3 solution
- DDR PHY Interface (DFI) support for easy integration with industry standard DFI 3.1-compliant PHYs
  - All signals supported on control, write data, and read data interfaces
  - Low Power Interface
- Scalable 1:2 frequency ratio architecture
- For DDR3, LPDDR2 and LPDDR3 configurations, direct software request control or programmable internal control for ZQ short calibration cycles
- For DDR3, LPDDR2 and LPDDR3 configurations, support for ZQ long calibration after self- refresh exit
- For LPDDR2 and LPDDR3 configurations, support for ZQ Reset feature through software
- Dynamic scheduling to optimize bandwidth and latency
- Read and write buffers in fully associative CAMs, configurable in powers of two, up to 64 reads and 64 writes

- Delayed writes for optimum performance on SDRAM data bus
- For maximum SDRAM efficiency, commands are executed out-of-order
- Hardware configurable and software programmable Quality of Service (QoS) support:
  - Support for three traffic classes on read commands—high priority reads, variable priority reads and low priority reads
  - Support for two traffic classes on write commands—normal priority writes and variable priority writes
  - Support for port urgent and port throttling control
- Configurable maximum SDRAM data-bus width (denoted “full data-bus width” below)
- Programmable support for all of the following SDRAM data-bus widths:
  - The full data-bus width or
  - Half of the full data-bus width or
  - One quarter of the full data-bus width.
- Two priorities for read transactions, and one for writes
- Control options to avoid starvation of lower priorities
- Guaranteed coherency for write-after-read (WAR) and read-after-write (RAW) hazards
- Write combine to allow multiple writes to the same address to be combined into a single write to SDRAM; supported for same starting address
- Paging policy selectable by configuration registers as any of the following:
  - Leave pages open after accesses
  - Close page when there are no further accesses available in the controller for that page, or
  - Auto-precharge with each access, with an optimization for page-close mode which leaves the page open after a flush for read-write and write-read collision cases
- Support automatic SDRAM power-down entry and exit caused by lack of transaction arrival for programmable time
- Support automatic Clock Stop (LPDDR2 / LPDDR3) entry and exit caused by lack of transaction arrival
- Support automatic power-down entry and exit caused by lack of transaction arrival for programmable time
- Support self-refresh entry and exit as follows:
  - Support automatic self-refresh entry and exit caused by lack of transaction arrival for programmable time
  - Support for self-refresh entry and exit under software control
  - Support for self-refresh entry and exit using dedicated DDRC hardware low power interface control (similar to the AMBA 3 AXI protocol low power control interface)

- Support for dynamically changing clock frequency while in self-refresh
  - DDR3 DLL-off mode supported
- Support for deep power-down entry and exit under software control (LPDDR2 / LPDDR3)
- Support for explicit SDRAM mode register updates under software control
- Flexible address mapper logic to allow application specific mapping of row, column, bank, and rank bits
- User-selectable refresh control options:
  - Controller-generated auto-refreshes at programmable average intervals
  - Ability to group up to 8 controller-generated refreshes together to be issued consecutively (this reduces the frequency of page closings, increasing overall efficiency)
  - When controller-generated refreshes are grouped, some refreshes can be issued speculatively when the controller is idle for a programmable period of time
  - Ability to disable controller-generated auto-refreshes
  - Ability to issue a refresh through direct software request
  - When LPDDR2 / LPDDR3 is used, user-selectable ability to perform per-bank refreshes rather than all-banks refreshes
- Advanced power-saving design includes no unnecessary toggling of command, address, and data pins (RAS / CAS / WE / BA / A hold last state after each command; DQ does not transition on writes when bytes are disabled)
- Low area, low power architecture
- 5-clock cycle typical command latency through the DDRMC (HIF interface)
  - Can be reduced to 4 cycles by choosing not to register DFI outputs (Configuration parameter)
  - 3-clock cycles for high priority read
- Leverages out of order requests with CAM to maximize throughput
- APB interface for the DDRMC software accessible registers

9.2.4 Functional Description

This chapter describes the functional about the DDRMC.

9.2.4.1 Functional overview

This section explains the functionality of the DDRMC. The DDRMC is composed of the following main architectural blocks:

- AXI Port Interface (XPI)
- DDR Controller (DDRC)
- Register Block

The DDRMC performs the following functions:

- Accepts requests from the SoC core with system addresses and associated data for writes.
- Performs address mapping from system addresses to SDRAM addresses (rank, bank, bank group, row).
- Prioritizes requests to minimize the latency of reads (especially high priority reads) and maximize page hits.
- Ensures that the SDRAM is properly initialized.
- Ensures that all requests made to the SDRAM are legal (accounting for associated SDRAM constraints).
- Ensures that refreshes and other SDRAM and PHY maintenance requests are inserted as required.
- Controls when the SDRAM enters and exits various power-saving modes appropriately.

9.2.4.2 Address mapper

Read and write requests are provided to the DDRMC with a system address. The system address is the command address of a transaction as presented on one of the data ports. The DDRMC converts this system address to a physical address. It maps the system address to the SDRAM rank, bank, row, and column addresses.

If the system address regions are enabled (DDRMC_A_NSAR > 0), the first part of the mapping is conversion of system address to AXI[1] byte address. The controller maps disjoint address regions to internal consecutive addresses. Otherwise, the controller assumes that the DRAM is always mapped as a monolithic block. For more information about this, see System address regions.

The second part of this mapping is conversion of AXI/AHB byte address to HIF word address. This is performed in the XPI block, and is applicable to AXI/AHB configurations only. For more information about this, see Application to HIF address mapping.

The last part is the conversion of HIF word address to SDRAM address. A flexible address mapper maps the HIF word address to the SDRAM rank / bank / bank group / row / column address. This address mapper is located within the DDRC.

The address mapping to be used depends on the use-case. The DDRMC provides a set of registers that allows flexible re-programming of logical to physical address mapping. For more information about this, see HIF address to SDRAM address mapping.

1. System Address Regions are only supported with AXI configurations.

The following figure explains the conversion of AXI to HIF to the SDRAM rank / bank / row / column / bank group addresses.

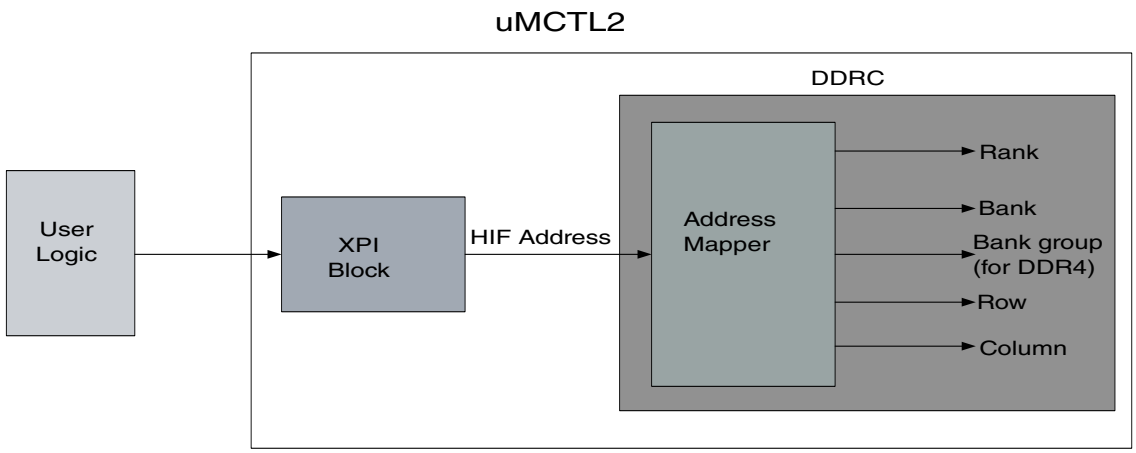


Figure 9-2. Conversion of AXI to HIF to SDRAM Rank / Bank / Row / Column / Bank Group Addresses

9.2.4.2.1 System address regions

The system address regions add the capability to define up to four disjoint memory regions mapping to the DRAM as consecutive addresses. The number of regions and minimum block size are determined by the hardware parameters DDRMC_A_NSAR and DDRMC_SARMINSIZE.

Each region is defined by:

- Base Address: Starting address of the region aligned to the minimum block size.
- Number of Blocks: Size of the region in multiples of minimum block size.

The base address of each region is specified by the register SARBASEn.base_addr and the total number of blocks is specified by the register SARSIZEn.nblocks (see SAR Size Register n (DDRC_MP_SARSIZE*n*)).

System address region specification is the same for all ports. Error response is not generated by the controller for addresses falling outside the specified address regions and the same address translation is applied from one base address to the next base address.

The base addresses must be specified so that they do not overlap. It is assumed that an outside agent can generate address decode errors if they happen to occur.

9.2.4.2.1.1 System address regions example

Consider the following example. The controller is set to have three system address regions (as specified on Table 9-3), with a minimum block size of 2 GB. Therefore, the DDRMC_SARMINSIZE parameter must be set to 4.

The system address bits [DDRMC_A_ADDRW-1:31] is used to represent the base addresses in the multiples of the minimum block size.

The registers must be programmed as follows:

- SARBASE0.base_addr = 1 (see SAR Base Address Register n (DDRC_MP_SARBASE*n*))
- SARBASE1.base_addr = 17
- SARBASE2.base_addr = 272
- SARSIZE0.nblocks = 0
- SARSIZE1.nblocks = 14
- SARSIZE2.nblocks = 15

The specifications in Table 9-3 translates to the register settings mentioned in Table 9-4.

Table 9-3. Address Region Mapping Example - Specification

Region	System Address (40-bit)	Offset	Internal Controller Address (36-bit)
Region 0 (2 GB)	0x00 8000 0000–0x00 FFFF FFFF	0x00 8000 0000	0x00 0000 0000–0x00 7FFF FFFF
Region 1 (30 GB)	0x08 8000 0000–0x0F FFFF FFFF	0x08 80000 0000	0x00 8000 0000–0x07 FFFF FFFF
Region 2 (32 GB)	0x88 0000 0000–0x8F FFFF FFFF	0x80 0000 0000	0x08 0000 0000–0x0F FFFF FFFF

Table 9-4. Example Base Addresses, Number of Blocks and Offset

Region	Base Address (Binary)	Register Value (base_addr)	Register Value (n blocks)	Calculated Base Address	Calculated Offset (Decimal / GB)
Region 0	9'b0_0000_0001	1	1	2 GB	1 / 2GB
Region 1	9'b0_0001_0001	17	15	34 GB	16 / 32GB
Region 2	9'b1_0001_0000	272	16	544 GB	256 / 512GB

Figure 9-3 shows the graphical illustration of the system address regions as specified in Table 9-3.

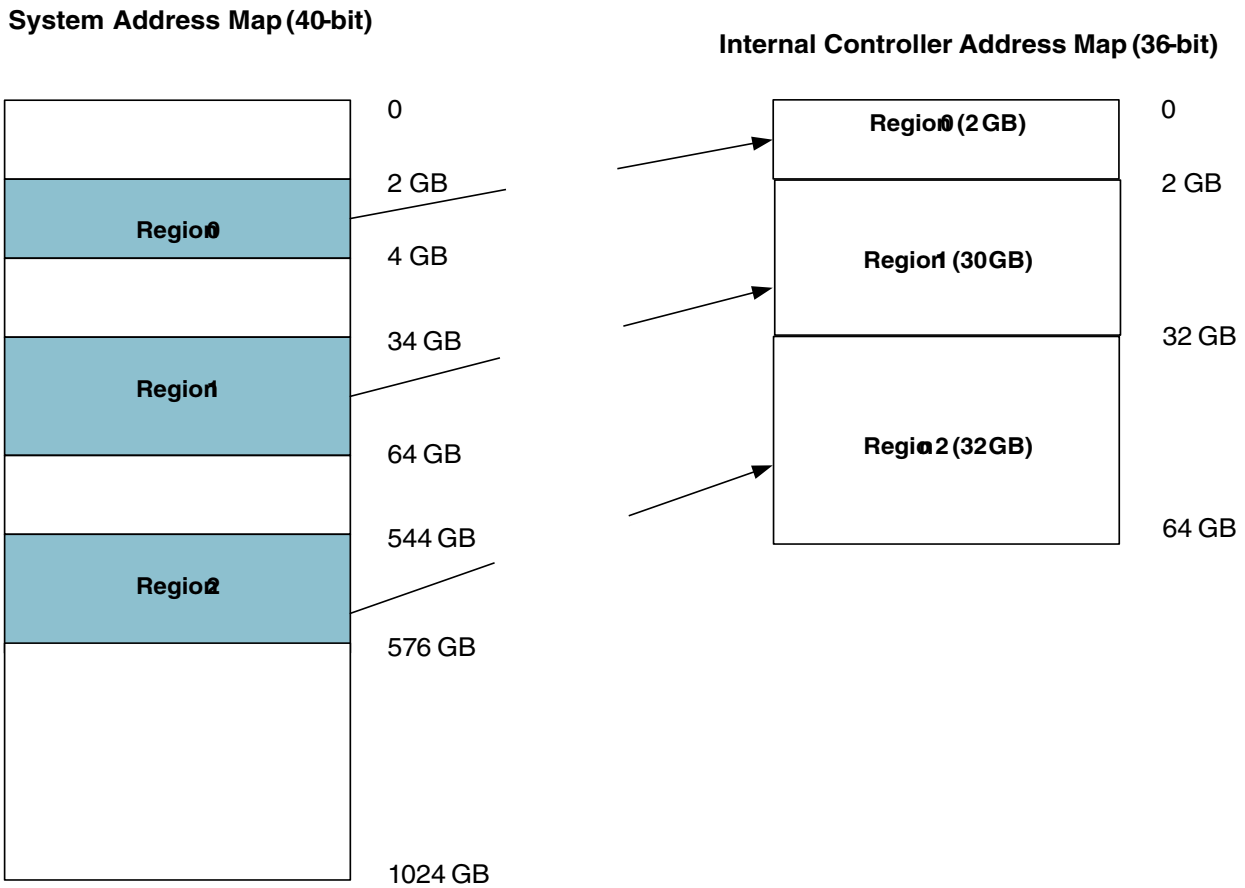


Figure 9-3. Address Region Mapping Example

NOTE

When DDRMC_AXI_BOUNDARY > DDRMC_SARMINSIZE + 27, portion of the address below the boundary can be changed by the conversion from system to AXI address. In this case, the user should ensure that the converted AXI address does not cross the boundary.

9.2.4.2.1.2 Application to HIF address mapping

The {ARADDR | AWADDR | HADDR} is a byte address. Based on the MEMC_BURST_LENGTH, XPI maps the MSB bits of the application address to the HIF address (hif_cmd_addr) in the following ways:

- HIF_CMD_ADDR [36: LOG2 (MEMC_BURST_LENGTH)] = {ARADDR | HADDR} [DDRMC_A_ADDRW - 1: LOG2 (MEMC_BURST_LENGTH) + LOG2 (MEMC_DRAM_DATA_WIDTH / 8]

- HIF_CMD_ADDR [LOG2 (MEMC_BURST_LENGTH) - 1: MEMC_FREQ_RATIO] is internally generated by XPI.
- HIF_CMD_ADDR [MEMC_FREQ_RATIO-1: 0] = 0

For example, if memory data width is 16, frequency ration is 1:1, and the application address width is 32:

- If MEMC_BURST_LENGTH = 8
  - HIF_CMD_ADDR [36:3] = {ARADDR | HADDR} [31:4]
  - HIF_CMD_ADDR [2:1] is internally generated by XPI.
  - HIF_CMD_ADDR [0] = 0
- If MEMC_BURST_LENGTH = 4
  - HIF_CMD_ADDR [36:2] = {ARADDR | AWADDR | HADDR} [31:3]
  - HIF_CMD_ADDR [1] is internally generated by XPI.
  - HIF_CMD_ADDR [0] = 0

9.2.4.2.1.3 HIF address to SDRAM address mapping

The address mapper maps HIF word addresses to SDRAM addresses by selecting the HIF address bit that maps to each and every applicable SDRAM address bit. While it is possible to map HIF address bits to a SDRAM address in any desired manner, the full available address space is accessible only when no two SDRAM address bits are determined by the same HIF address bit. Each SDRAM address bit has an associated register vector to determine its source.

Registers ADDRMAPx (x = 0 to 8) are used to program the address mapper. For more information on ADDRMAP registers, see DDRC.

The HIF address bit number is determined by adding the internal base of the ADDRMAPx (x = 0 to 8) register to the programmed value for that register, as described in the following equation:

HIF address bit number = [internal base] + [register value]

For example, for ADDRMAP3.addrmap_col_b7, the internal base is 7. When full data bus is in use, column bit 7 is determined by the following equation:

7 + [register value]

If this register is programmed to 2, the HIF address bit can be calculated by using following equation:

[HIF address bit number] = 7 + 2 = 9

In other words, the column address bit 7 sent to SDRAM would always be equal to hif_cmd_addr[9] of the corresponding HIF source address.

NOTE

- All of the column bits shift up 1 bit when only half of the data bus is in use. In this case, you need to look at ADDRMAP3.addrmap_col_b6 instead to determine the value of column address bit 7.
- All of the column bits shift up 2 bits when only a quarter of the data bus is in use. In this case, you need to look at ADDRMAP2.addrmap_col_b5 instead to determine the value of column address bit 7.
- All of the column bits shift up an additional 1 bit when the DDRMC is configured for BL8 mode (MEMC_BURST_LENGTH = 8).
- For Arbiter configurations (DDRMC_UNCL_ARB = 1) with mEMC_BURST_LENGTH = 8, it is required to program ADDRMAP2.addrmap_col_b2 = 0, so that the Arbiter can generate HIF address that are column aligned.
- If LPDDR2 or LPDDR3, or if MEMC_BURST_LENGTH = 8, it is required that HIF_CMD_ADDR [0] = 0, If DDRMC_INCL_ARB = 1, this is automatically the case. Otherwise, it is required that the HIF master generates HIF_CMD_ADDR [0] = 0.
- The register ADDRMAP5.addrmap_row_b2_10 maps multiple HIF address bits.
- For any address bits which cannot be in use in all cases, all bits of the associated address map register must be set to 1 when the associated SDRAM address bit is not in use.

The system address to physical address mapping can be completed by choosing any one of the possible combinations from the following figure. It explains the different possible ways to map the HIF address bits to the SDRAM rank / bank/ bank group / row / column address.

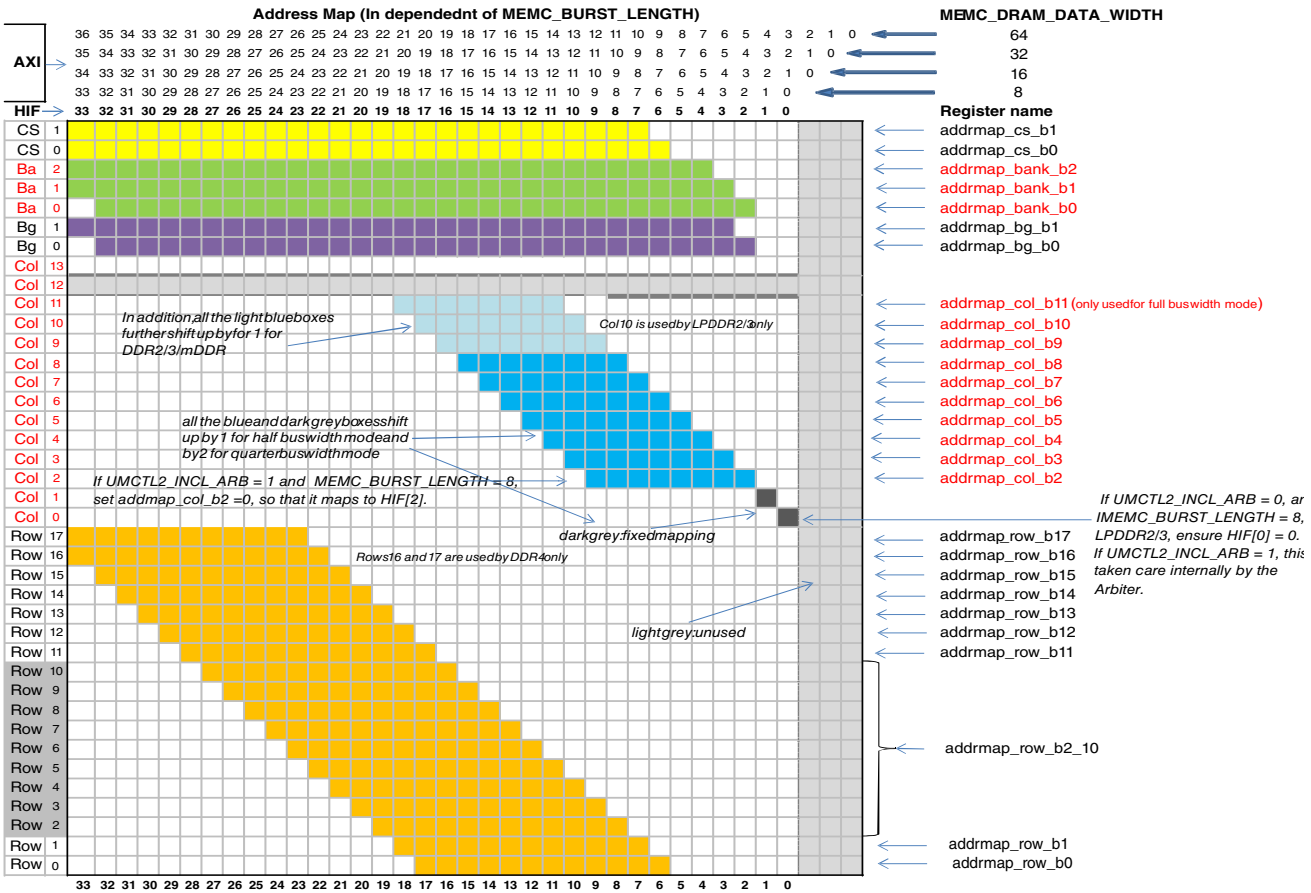


Figure 9-4. System Address to Physical Address Mapping (Independdent of MEMC_BURST_LENGTH)

NOTE

User should ensure that no two SDRAM address bits are determined by the same HIF address bit.

The following table shows some examples of address mapping schemes which might be used. This example assumes a full bus width, using an SDRAM with 10 column address bits (denoted “c” in the diagram), three bank address bits (denoted “b”) and 12 row address bits (denoted “r”). The non-interleaved example shows a simple address map where the lowest HIF address bits map to the SDRAM column addresses, the next three to the bank addresses, and the highest HIF address bits map to the row addresses. In the first interleaved example, the bank address bits are interleaved with the column address bits. In the last two examples, a more arbitrary mapping is shown. These examples are randomly selected to illustrate the flexibility of the DDRMC address mapper and they have no bearing on a typical system.

NOTE

The example "Non-Interleaved" and "Interleaved example1" are the same for both MEMC_BURST_LENGTH = 4 and MEMC_BURST_LENGTH = 8.

Table 9-5. Logical to Physical Address Mapping (Examples for DDR3)

	HIF Address Bits MEMC_BURST_LENGTH	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Non-interleaved	8	r11	r10	r9	r8	r7	r6	r5	r4	r3	r2	r1	r0	b2	b1	b0	c9	c8	c7	c6	c5	c4	c3	c2	c1	c0
	4	r11	r10	r9	r8	r7	r6	r5	r4	r3	r2	r1	r0	b2	b1	b0	c9	c8	c7	c6	c5	c4	c3	c2	c1	c0
Interleaved - example 1	8	r11	r10	r9	r8	r7	r6	r5	r4	r3	r2	r1	r0	c9	c8	c7	c6	c5	c4	b2	b1	b0	c3	c2	c1	c0
	4	r11	r10	r9	r8	r7	r6	r5	r4	r3	r2	r1	r0	c9	c8	c7	c6	c5	c4	b2	b1	b0	c3	c2	c1	c0
Interleaved - example 2	8	r11	r10	r9	r8	r7	r6	r5	r4	r3	r2	c6	c7	b2	c8	c9	r0	r1	b1	c3	c5	c4	b0	c2	c1	c0
Interleaved - example 3	4	r11	r10	r9	r8	r7	r6	r5	r4	r3	r2	c9	r0	r1	b0	c7	c8	c5	b1	c6	b2	c4	c3	c2	c1	c0

The following table illustrates how the ADDRMAP* registers should be set to achieve a particular address mapping. The address mapping is chosen from the interleaved example 2 of Table 9-5.

Table 9-6. Example of Register Programming to Implement a Desired Address Mapping Scheme (for DDR3)

HIF Address Bit	Desired SDRAM Address	Register Field to be Programmed	Internal Base	Value to Program
0	c0	None	—	—
1	c1	None	—	—
2	c2	ADDRMAP_MAP_COL_B2	2	0
3	b0	ADDRMAP_BANK_B0	2	0
4	c4	ADDRMAP_COL_B4	4	0
5	c5	ADDRMAP_COL_B5	5	0
6	c3	ADDRMAP_COL_B3	3	3
7	b1	ADDRMAP_BANK_B1	3	4
8	r1	ADDRMAP_ROW_B1	7	1
9	r0	ADDRMAP_ROW_B0	6	3
10	c9	ADDRMAP_COL_B8	8	2
11	c8	ADDRMAP_BANK_B7	7	4
12	b2	ADDRMAP_COL_B2	5	7

Table continues on the next page...

Table 9-6. Example of Register Programming to Implement a Desired Address Mapping Scheme (for DDR3) (continued)

HIF Address Bit	Desired SDRAM Address	Register Field to be Programmed	Internal Base	Value to Program
13	c7	ADDRMAP_COL_B6	6	7
14	c6	ADDRMAP_ROW_B5	5	9
15	r2	ADDRMAP_ROW_B2_10	8	7
...	...	...	...	...
23	r10	ADDRMAP_ROW_B2_10	16	7
24	r11	ADDRMAP_ROW_B11	17	7

9.2.4.2.1.4 LPDDR3 6-Gb and 12-Gb device densities

LPDDR3 6Gb and 12Gb device densities do not support addresses which have both row bits 13 and 14 high (row[14:13] = 2'b11). Any attempt to read or write to this address space could result in memory misbehavior or the system hangs. When such devices are used, the user must set register ADDRMAP6.lpddr3_6gb_12gb to 1 (see Address Map Register 6 (DDRC_ADDRMAP6)). This prevents the DDRMC from accessing this address space when there is a read or write request from the user / SoC core.

When ADDRMAP6.lpddr3_6gb_12gb is set to 1, any write or RMW request with row[14:13] = 2'b11 is discarded, while the HIF output HIF_WDATA_PTR_ADDR_ERR is asserted. Also, any read request with row[14:13] = 2'b11 is executed (dummy read), by changing row[14:13] from 2'b11 to 2'b10. The return data for such reads are masked to zeros, while the HIF output HIF_RDATA_ADDR_ERR is asserted.

For AXI / AHB configurations, any assertion of HIF_WDATA_PTR_ADDR_ERR or HIF_RDATA_ADDR_ERR results also in SLVERR / ERROR response.

9.2.4.3 Transaction service control

The transaction service control allows the user to carefully manage the following:

- Costly read / write bus turn-arounds
- Priorities of read requests to generally favor high priority traffic while also preventing starvation of low priority traffic

This functionality is implemented in a simple 2-state machine for each traffic type with completely configurable controls. The states of the 2-state machine determine when reads / writes are serviced and the relative priority (high priority versus low priority reads) at any given moment.

Following is the list of registers related to transaction service control:

- Low priority read transaction store
  - PERFLPR1.lpr_max_starve (see Low Priority Read CAM Register 1 (DDRC_PERFLPR1))
  - PERFLPR1.lpr_xact_run_length
- High priority read transaction store
  - PERFHPR1.hpr_max_starve
  - PERFHPR1.hpr_xact_run_length
- Write transaction store
  - PERFWR1.w_max_starve
  - PERFWR1.w_xact_run_length
- Bus turn-around control
  - SCHED.prefer_write
  - SCHED.rdwr_idle_gap
- Variable priority read transaction grouping
  - PERFVPR1.vpr_timeout_range
- Variable priority write transaction grouping
  - PERFVPW1.vpw_timeout_range

9.2.4.3.1 Transaction stores

Transactions in the DDRMC are separated into five categories:

- Low Priority Reads (LPR)
- Variable Priority Reads (VPR)
- High Priority Reads (HPR)
- Normal Priority Writes (NPW)
- Variable Priority Writes (VPW)

NOTE

VPR and VPW categories are only supported when the multi port arbiter is configured (DDRMC_INC_ARB = 1). Hardware parameters DDRMC_INCL_ARB and DDRMC_VPRW_EN should be defined to enable the VPR / VPW feature.

9.2.4.3.1.1 Read transaction store

This section describes how LPR, VPR and HPR traffic classes work inside the DDRC. See Quality of service for information on how to map the AXI arqos value to LPR, VPR, and HPR traffic queues.

SCHED.lpr_num_entries register splits the Read CAM in the controller into LPR and HPR sections. The LPR and VPR commands are sent to the LPR section of the CAM, and the HPR commands are sent to the HPR section. The VPR commands come in with an associated ‘timeout’ value. The LPR and HPR commands do not have ‘timeout’ value associated with them. The ‘timeout’ value of the VPR commands are counted down every cycle in which the commands are pending in the LPR store. If any VPR command has not been serviced and its ‘timeout’ value reaches 0, the command is considered as expired-VPR command. When there are any expired-VPR commands in the DDRC, those commands are given higher priority than HPR and LPR commands.

A register, PERFVPR1.vpr_timeout_range, is used to indicate the range of the ‘timeout’ counter value to be considered when the priority is switched from VPR to expired-VPR. This feature is used to execute a group of VPR commands that are in the temporal vicinity together, thereby possibly improving the bus utilization. For example, if the register value is set to 0xF, it indicates that the priorities of all the VPR entries whose timeout counters are 15 or below are promoted to expired-VPR when any of the VPR commands timeout.

The Read CAM handles three traffic classes - LPR, VPR and HPR. But the Scheduling Engine that gets its input from the Read CAM, handles only two traffic classes - LPR and HPR. When there are no expired-VPR commands, all the VPR commands are treated as LPR. When there are any expired-VPR commands in DDRC, all expired-VPR commands are treated as HPR commands.

9.2.4.3.1.2 Write transaction store

This section describes how NPW and VPW traffic classes work inside the DDRC. See Quality of service for information on how to map the AXI awqos value to NPW and VPW traffic queues.

The NPW and VPW commands are sent to the Write CAM. VPW commands do not have reserved space in the Write CAM. The VPW commands come in with an associated ‘timeout’ value. The NPW commands do not have ‘timeout’ value associated with them. The ‘timeout’ value of the VPW commands are counted down every cycle in which the commands are pending in the WR store. If any VPW command has not been serviced and its ‘timeout’ value reaches 0, the command is considered as expired-VPW command. When there are any expired-VPW commands in the DDRC, those commands are given higher priority than NPW commands.

A register, PERFVPW1.vpw_timeout_range, is used to indicate the range of the ‘timeout’ counter value to be considered when the priority is switched from VPW to expired-VPW. This feature is used to execute a group of VPW commands that are in the temporal vicinity together, thereby possibly improving the bus utilization. For example, if the

register value is set to 0xF, it indicates that the priorities of all the VPW entries whose timeout counters are 15 or below are promoted to expired-VPW when any of the VPW commands timeout.

The Write CAM handles two traffic classes - NPW and VPW. But the Scheduling Engine that gets its input from the Write CAM, handles only one traffic class - NPW. When there are no expired-VPW commands, all the VPW commands are treated as NPW. When there are any expired-VPW commands in DDRC, all expired-VPW commands are given higher preference over NPW commands.

9.2.4.3.1.3 Transaction store state transitions

Each class of transaction store (LPR, HPR, or WR) can be in any one of the following two states:

- Normal: It is the state where the transaction store starts.
- Critical: It indicates that the transaction store must be prioritized for service.

The transaction stores move between these two states under the control of *_MAX_STARVE and *_XACT_RUN_LENGTH registers.

The following table describes the transaction store state transitions.

Table 9-7. Transaction Store State Transitions

Current State	Next State	State Transition
Normal	Critical	This transaction store is not serviced for a count of *_MAX_STARVE clock cycles, or if an address collision happens.
Critical	Normal	*_XACT_RUN_LENGTH number of transactions is serviced from this transaction store.

Taking the low priority read transaction store as an example, it is expected that the transaction store generally functions independently based on the following registers (see Low Priority Read CAM Register 1 (DDRC_PERFLPR1)):

- PERFLPR1.lpr_max_starve
- PERFLPR1.lpr_xact_run_length

In the normal mode of operation, the FSM moves from normal to critical state when the starvation timer times out. The store remains in Critical state until the number of transactions equal to what is indicated in PERFLPR1.lpr_xact_run_length are serviced, or when there are no more commands available for that store, whichever happens first.

The presence of any expired-VPR command in the DDRC makes the HPR queue to be in Critical state regardless of the value of its starvation timer. The HPR queue stays in Critical state until all the expired-VPR commands are served.

Similarly, the presence of any expired-VPW command in DDRC makes the WR queue to be in Critical state regardless of the value of its starvation timer. The WR queue stays in Critical state until all the expired-VPW commands are served.

When the DDRMC is configured as HIF-only (DDRMC_INCL_ARB = 0), it provides a feature where the SoC core can force this state transition using external signals. It is useful in cases where the SoC core may have additional information that is helpful in determining state transitions. For example, if the data stream has real-time requirements and the SoC core has knowledge about FIFO depths or time-till-failure, it can use this information to explicitly request the DDRMC to prioritize a particular data stream when it is critical to do so. The signals associated with this feature are HIF_GO2CRITICAL_LPR and HIF_GO2CRITICAL_WR.

In case of multi port arbiter configuration (DDRMC_INCL_ARB = 1), the HIF_GO2CRITICAL_* SIGNALS are driven by the Port Arbiter. The generation of three signals involved in this case - HIF_GO2CRITICAL_WR, HIF_GO2CRITICAL_LPR and HIF_GO2CRITICAL_HPR is described in VPR / VPW timeout and Urgent signalling.

The assertion of HIF_GO2CRITICAL_* SIGNALS cause their respective queue FSMs to go to Critical state. The change in Read / Write mode switching and the Read priority level selection based on the presence of the HIF_GO2CRITICAL_* SIGNALS is mentioned in the following section.

9.2.4.3.1.4 Read / Write turn-around

The following terminology is used to describe the Read / Write turn-around algorithm:

- Write / Read pending: A Write or a Read command is pending in the CAM.
- Write / Read collision: A Write collision indicates that a new Read command that arrived on the HIF bus is to the same address as a pending Write in the CAM and that the collided Write has to be now flushed before accepting the Read command into the CAM (vice versa for Read collision).
- Write / LPR / HPR critical: Indicates that the Write, LPR or HPR queue is in critical state due to starvation.
- Read / Write idle timeout: Read/Write idle timer based on the register SCHED.rdwr_idle_gap has expired.

The DDRC starts in the Read mode (servicing read requests) and then switches to Write mode (servicing write requests) in the following conditions:

- Write pending && (Write collision || Write critical) && ! Read collision && ! HPR critical && ! LPR critical, or

- No Read pending && (SCHED.prefer_write || (Write pending && Read/Write idle timeout)), or
- Expired-VPW pending and no Expired-VPR pending

The DDRC switches back to Read mode in the following conditions:

- Read pending && (hif_go2critical_hpr || hif_go2critical_lpr) && !Write collision
- Read pending && (Read collision || HPR critical || LPR critical) && ! Write collision && ! Write critical,
- No Write pending && (~SCHED.prefer_write || (Read pending && Read/Write idle timeout))
- Expired-VPR pending

The Read / Write idle timeout logic is as given below:

- If SCHED.prefer_write = 0, count down when Write pending and No Reads.
- If SCHED.prefer_write = 1, count down when Read pending and No Writes.

When both expired-VPR and expired-VPW are present at the same time, expired-VPR is given higher preference.

NOTE

- The normal programming is expected to be SCHED.prefer_write = 0. This implies that read requests are immediately serviced when the DDRMC is idle. Also, it is often desirable to set the SCHED.rdwr_idle_gap to a lower number (such as 0, 1, or 2). This ensures that writes do not go unserviced in an otherwise-idle DDRMC for any length of time, wasting the bandwidth. The trade-off is that reads received on the HIF immediately after these writes may incur additional latency. This is because time is needed to turn the bus around, service the writes, and then turn the bus around again to service the reads.
- Ordering between reads and writes to the same address is guaranteed on all requests issued to the DDRMC. Therefore, write latency is not be a concern to system design. In the event that write data is required by a subsequent read, the DDRMC automatically forces the write data out to SDRAM before servicing the read.

9.2.4.3.1.5 Read priority management

In a read mode, high priority read requests are preferred for service over low priority read requests. However, if the low priority read transaction store is in Critical state due to a transaction that is pending for a long time (see Transaction store state transitions) and the high priority read transaction store is not, the low priority read requests are preferred over high priority read requests. This prevents starvation of low priority reads.

The exceptions to the above rule are as follows:

- When there are any expired-VPR commands in DDRC, the HPR queue is given priority over LPR queue, even if LPR queue is in Critical state.
- If HIF_GO2CRITICAL_HPR is high, the HPR queue is given priority over LPR queue, even if LPR queue is in Critical state.
- If HIF_GO2CRITICAL_LPR is high and HIF_GO2CRITICAL_HPR is low, the LPR queue is given priority over HPR queue, even if HPR queue is in Critical state

9.2.4.3.2 Address collision handling

The DDRC can execute transactions out-of-order while ensuring that all transactions appear as if they are executed in the order in which they are received. Every transaction that requires a response from the DDRC arrives with a token number which is provided back to the SoC core as part of the response. Since the DDRC queues transactions prior to execution, it is possible that multiple transactions to the same SDRAM address can arrive before the first transaction to that address is issued.

To enforce ordering of accesses to the same address, the DDRC uses the following algorithm:

1. **New read colliding with queued read**: This collision causes no problems. The two reads can end up being executed out-of-order.
2. **New write colliding with queued write**: If write combine is enabled, the DDRC overwrites the data for the old write with that from the new write and only performs one write transaction (write combine). For more information about this, see Write combine.
3. **New read (or write) colliding with queued write (or read) respectively**: In this case, the DDRC performs the following sequence:
   a. Holds the new transactions in a temporary buffer.
   b. Applies flow control back to the SoC core to prevent more transactions from arriving.
   c. Flushes the internal queue holding the colliding transaction until that transaction is serviced.
   d. Accepts the new transaction and removes flow control.

4. **New read colliding with both read and write**: This can happen when a read collides with a RMW command. In this case, the reads are flushed until the read collision is cleared, then the writes are flushed in the same manner as described in the step3.
5. **New write colliding with both read and write**: This can happen when a write collides with a RMW command. In this case, the new write is held in a temporary buffer until the read is completed. Then, it is combined with the queued write (if write combine is enabled).
6. **New RMW colliding with queued write**: In this case, the new RMW is stored in a temporary buffer until the queued write is completed.

9.2.4.3.3 Write combine

The write combine feature can combine multiple writes to the same address into a single write to SDRAM.

When a new write collides with a queued write in the CAM

- If write combine is enabled, the DDRC overwrites the data for the old write with that from the new write and only performs one write transaction (write combine).
- If write combine is disabled, the DDRC performs the following sequence:
  a. Holds the new write transaction in a temporary buffer.
  b. Applies flow control back to the SoC core to prevent more transactions from arriving.
  c. Flushes the internal queue holding the colliding transaction until that transaction is serviced.
  d. Accepts the new transaction and removes flow control.

9.2.4.3.4 Page policy

9.2.4.3.4.1 Explicit auto-precharge (Per command)

The explicit auto-precharge feature can enable auto-precharge on a per-command basis. If the HIF signal HIF_CMD_AUTOPRE is set during a valid command, then the auto-precharge bit for that command is set when it is sent to the SDRAM.

If you do not want to use the per-command auto-precharge feature, then this bit can be tied to 1’b1 or 1’b0. Tying it to 1’b1 causes all the commands to be executed with auto-precharge, and tying it to 1’b0 causes no commands to be executed with auto-precharge. For AXI configurations, HIF_CMD_AUTOPRE is tied to 0 internally, and is not accessible to the user.

In cases where a HIF transaction is translated into multiple DFI transactions, only the last DFI transaction is executed with auto-precharge. The earlier ones keep the page open, to allow subsequent transactions to benefit from the page hit. Also, if DBG0.dis_collision_page_opt (see Debug Register 0 (DDRC_DBG0)) is set to 0, the auto-precharge is automatically disabled for the flushed command in a collision case.

9.2.4.3.4.2 Intelligent precharges

The SCHED.pageclose register (see Scheduler Control Register (DDRC_SCHED)) enables precharge commands to be issued smartly via auto-precharges or explicit precharges. The exact functionality depends on the value programmed in SCHED1.pageclose_timer:

- If SCHED.pageclose is set to 1 and SCHED1.pageclose_timer = 0, a bank is kept open until there are page hit transactions available in the CAM to that bank. The last read or write command in the CAM with a bank and page hit is executed with auto-precharge. Even if this register is set to 1, explicit precharge (and not auto-precharge) can be issued in some cases where there is a mode switch between write and read or between LPR and HPR. The read and write commands that are executed as part of the ECC scrub requests are also executed with explicit precharge if the page needs to be closed.
- SCHED.pageclose is set to 1 and SCHED1.pageclose_timer > 0, a bank is kept open until there are page hit transactions available in the CAM to that bank. The last read or write command in the CAM with a bank and page hit is not executed with auto-precharge. Instead, a timer is started with pageclose_timer as the initial value. There is a timer on a per bank basis. The timer decrements unless the next read or write in the CAM to a bank is a page hit. It resets to pageclose_timer value if the next read or write in the CAM to a bank is a page hit. Once the timer reaches zero, an explicit precharge is scheduled.
- If SCHED.pageclose is set to 0, the bank remains open only until there is a need to close it (to open a different page or for page timeout or refresh timeout). This is also known as open page policy. The open page policy can be overridden by setting the per command auto pre bit on the HIF interface (HIF_CMD_AUTOPRE) as described in Explicit auto-precharge (Per command). When the multi port arbiter is configured, the HIF signal hif_cmd_autopre is tied to 0 and the user has no control over it.

The intelligent precharge feature provides a midway between open and close page policies. SCHED1.pageclose_timer (see Scheduler Control Register 1 (DDRC_SCHED1)) gives user control over the time waited when there are no page hits to a bank in the CAM before auto-precharge or explicit precharge is scheduled. It is useful when the multi port arbiter is configured. This timer allows the page to be kept open for a configurable number of clock cycles after there are no commands pending in

the CAM to a bank. If there are pending commands higher up in the stream (for example, XPI / PA of multi port arbiter) to the same bank / page, it gives a chance to be scheduled by the DDRMC as an open page command.

The following table provides a summary of all paging policy options available in the DDRMC.

Table 9-8. Paging Policy Options

Paging Policy	Availability	Description
Open page policy	All configurations	Setting SCHED.pageclose to 0 enables the open page policy. In HIF configurations, the open page policy can be overridden by sending commands with HIF_CMD_AUTOPRE = 1.
Intelligent precharge(Midway between Closed page and Open page policy)	All configurations	Setting SCHED.pageclose to 1 enables this smart paging policy as described in Intelligent precharges. The exact functionality is dependent on SCHED1.pageclose_timer value. InHIF configurations, this paging policy can be overridden by sending commands with HIF_CMD_AUTOPRE = 1.
Closed page policy	Only in HIF configurations. Not in multi port configurations	Control the paging policy per-command by asserting or de-asserting the auto precharge signal on the HIF interface – HIF_CMD_AUTOPRE. If all commands are to be executed with auto-precharge, HIF_CMD_AUTOPRE can be tied to 1.

9.2.4.4 Quality of service

9.2.4.4.1 Traffic classes

The DDRMC supports different traffic classes that are distinguished from each other by the arqos signal:

9.2.4.4.1.1 Read classes

- Low Priority Read (LPR): It is also called as the best effort traffic. There is no resource allocation and it shares the resources with the other traffic types. LPR is always treated as low priority in the PA and in the DDRC. There are timeout mechanisms that can be used to prevent starvation for both the PA and the DDRC. When there is a timeout in the PA, that port becomes the highest priority (priority0). When there is a timeout in the LPR store in the DDRC (in other words, LPR store becomes critical), the LPR entries are served before HPR entries.
- Variable Priority Read (VPR): It is also called as the maximum latency bound traffic for meeting real time deadlines in video or audio applications. VPR traffic shares the same resources with LPR in the DDRC. But, based on the configuration, it may have a dedicated queue (red or blue) in the XPI. It has the same priority as LPR, but lower priority than HPR for the PA. For the DDRC, VPR has the same initial priority as

LPR. Each command tagged as VPR has an associated latency timer. When expired, VPR transactions have the highest priority in the controller, both in the PA and in the DDRC. The purpose of this traffic class is to limit the maximum latency, where this latency bound is expected to be a few hundred clock cycles.

- High Priority Read (HPR): It has allocated resources. If the XPIs are configured to have dual read address queues, then the red queue can be allocated to HPR. For DDRC, HPR traffic goes to its own dedicated store. HPR traffic has higher priority than LPR. HPR is meant for latency critical but not real time applications such as CPU.

9.2.4.4.1.2 Write classes

- Normal Priority Write (NPW): It is also called as the best effort traffic. There is no resource allocation and it shares the resources with the other traffic types. It is always treated as normal priority in the PA and DDRC. There are timeout mechanisms that can be used to prevent starvation for the PA. When there is a timeout in the PA, that port becomes the highest priority (priority0).
- Variable Priority Write (VPW): It is also called as the maximum latency bound traffic to meet real time deadlines in video or audio applications. VPW traffic shares the same resources with NPW in the DDRC. For the DDRC, VPW has the same initial priority as NPW. Each command tagged as VPW has an associated latency timer. When expired, VPW transactions have the highest priority in the controller, both in the PA and in the DDRC. The purpose of this traffic class is to limit the maximum latency, where this latency bound is expected to be a few hundred clock cycles.

9.2.4.4.1.3 QoS mapping

The priority of an initiator (master) is susceptible to change by the intermediate agents (for example, interconnect) before reaching the destination slave depending on the service levels provided with respect to the required targets. Therefore, the relationship between the IDs and their class representations must be dynamically determined.

Arqos / awqos signal determines both port priorities and DDRC priorities dynamically. 16 QoS levels are divided to three regions for reads, two regions for writes. Each region can be assigned to any of the following traffic class—LPR, VPR, and HPR for reads, NPW and VPW for writes.

Region 0 and 1 are assigned to the blue address queue (See Dual read address queue). Typically, LPR / NPW and VPR / VPW may share this resource.

Region 2 is assigned to the red read address queue, this is not present for writes. The red address queue can be mapped to the HPR or VPR traffic. In the DDRC, LPR and VPR share the same CAM store called LPR store. HPR has its own store in the DDRC.

The regions are mapped per port using PCFGQOS0_n and PCFGWQOS0_n registers (see Port n Read QoS Configuration Register 0 (DDRC_MP_PCFGQOS0_0) and Port n Write QoS Configuration Register 0 (DDRC_MP_PCFGWQOS0_0)).

Fields for PCFGQOS0_n register are as follows:

- RQOS_MAP_LEVEL<x> (where x is 1 to 2) indicates two separation levels for three regions. Possible values 0 to 14 correspond to an arqos value. These registers indicate the upper end of the region. For example, if RQOS_MAP_LEVEL2 is set to 14, then Region 2 is identified as all transactions with arqos value of 15.
- RQOS_MAP_REGION<y> (where y is 0 to 2) for region identifier. This register indicates the traffic class of each of the three regions. RQOS_MAP_REGION2 is present only in dual read queue configurations.
  Valid values are:
  - 0–LPR
  - 1–VPR
  - 2–HPR

Fields for PCFGWQOS0_n register are as follows:

- WQOS_MAP_LEVEL indicates the separation level for two regions. Possible values 0 to 14 correspond to an awqos value. These registers indicate the upper end of the region. For example, if WQOS_MAP_LEVEL is set to 14, then Region 1 is identified as all transactions with awqos value of 15.
- WQOS_MAP_REGION<y> (where y is 0 to 1) for region identifier. This register indicates the traffic class of each of the two regions.
  Valid values are:
  - 0–NPW
  - 1–VPW

For dual read address queue configurations, Region 0 and Region 1 map to the blue queue, and Region 2 maps to the red queue. The blue queue can only be set to LPR, VPR or both. The red queue can only be set to VPR or HPR.

For single address queue configurations, Region 0 and Region 1 map to the blue queue. Being single queue, Region 2 is not present. In this case registers RQOS_MAP_LEVEL2 and RQOS_MAP_REGION2 don’t exist. In this case Region 0 can be set to HPR or VPR, Region 1 to VPR or LPR.

The following figure shows an example dynamic mapping from AXI QoS input to internal controller traffic classes.

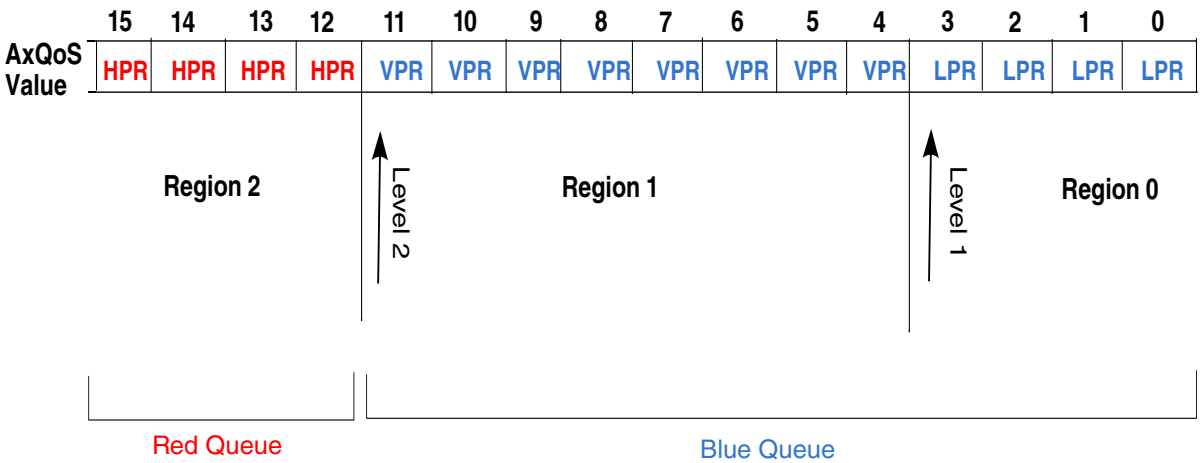


Figure 9-5. Example AxQoS Mapping to Traffic Classes

9.2.4.4.2 Dual read address queue

There are two separate read address queues in the XPI block—red and blue queue. They can be enabled by the parameter DDRMC_XPI_USE2RAQ_n for each port.

Both read queues request independently to the PA; therefore if enabled, maximum configurable number of XPI ports which is 16 is reduced. Each dual-queue XPI consumes two consecutive PA ports. The worst case is that there will be 8 ports where each XPI is configured to have dual read address queue.

Red queue can be used for one traffic class only and associated to Region 2 in the QoS mapper (HPR or VPR). Blue queue can be used for two traffic classes and associated to Region 0 and Region 1 in the QoS mapper (VPR and / or LPR).

There are separate time-outs for the blue and red queues. Timers are started to down count when the transaction is accepted in the XPI, which are then forwarded to DDRC together with the command (see VPR / VPW timeout).

Optional retime block (register slice for pipelining) at the input of the XPI read address channel is enabled by DDRMC_XPI_USE_INPUT_RAR. Optional retime block at the output of the XPI read address channel (enabled by DDRMC_XPI_USE_RAR) cannot be used if dual queue is enabled.

9.2.4.4.2.1 ID collisions

Due to the dynamic nature of the QoS with respect to the IDs, two transactions of the same ID can potentially exist in two queues. The collisions can be classified as:

- Blue after Red (BAR): A command with ID x is presented to the blue queue where a command with the same ID x is outstanding in the red queue.
- Red after Blue (RAB): A command with ID x is presented to the red queue where a command with same ID x is outstanding in the blue queue.

In the XPI, for both collision types, the ordering consistency within a given ID is preserved by stalling the incoming transaction to be pushed into the address queue it is mapped to until the colliding transaction in the opposite queue exists. This is required for functional correctness.

In case of any ID collision, the DDRMC does not speed up the drain of the stored transactions in the address queue causing collision even if the incoming transaction is HPR. In other words, address queues request arbitration to the Port Arbiter with their normal mapped priorities.

9.2.4.4.3 VPR / VPW timeout

There are separate timeouts for the blue and red queues for read transactions, and one timeout for write transactions. Timers are started to down count when the transaction is accepted in the XPI, which are then forwarded to DDRC together with the command.

Timeouts are set per port and queue using PCFGQOS1_n and PCFGWQOS1_n registers (see Port n Read QoS Configuration Register 1 (DDRC_MP_PCFGQOS1_0) and Port n Write QoS Configuration Register 1 (DDRC_MP_PCFGWQOS1_0)).

For PCFGQOS1_n register:

- RQOS_MAP_TIMEOUTB: Specifies the timeout value for transactions mapped to the blue queue.
- RQOS_MAP_TIMEOUTR: Specifies the timeout value for transactions mapped to the red queue.

For PCFGWQOS1_n register:

- WQOS_MAP_TIMEOUT: Specifies the timeout value for write transactions.

When expired, VPR/VPW transactions from XPI are tagged as expired-VPR or expired-VPW instead of LPR/NPW during normal priority transaction flow. The Port Arbiter treats these transactions with the highest priority (priority0). In addition, the Port Arbiter asserts hif_go2critical_lpr signal to the DDRC if there are no LPR credits available (LPR store of the read CAM is full) when an expired-VPR port is pending, and hif_go2critical_wr signal to the DDRC if there are no write credits available when an expired-VPW is pending. If an expired-VPW is issued as RMW at the HIF interface, the Port Arbiter asserts hif_go2critical_lpr signal to the DDRC if there are no LPR credits,

and / or hif_go2critical_wr signal if there are no write credits available. If both hif_go2critical_lpr and hif_go2critical_wr are asserted at the same time, priority in the DDRC is given to the read.

If VPR / VPW timeout registers are set to 0, the VPR / VPW transactions expire immediately as they enter the DDRMC, thereby making them the highest priority transaction class within the device. If multiple VPR / VPW transactions expire at the same time in the XPI, they are executed by the PA in round robin order.

To understand how the DDRC handles the VPR/VPW transactions, see Transaction service control. In addition to the registers mentioned in this section, PERFVPR1.vpr_timeout_range and PERFVPW1.vpw_timeout_range registers are also used by the DDRC and are related to the QoS.

9.2.4.4.3.1 Head of line blocking

Blue queue can be assigned to LPR and / or VPR if single or dual queue is selected, HPR and / or VPR if single queue is selected. Head of line blocking can occur in the XPI if more than one traffic class is mapped to the same address queue.

If a VPR expires inside the read address queue there can be two scenarios:

- Expired-VPR is blocked by one or more LPR transactions (single or dual queue configuration)
- Expired-VPR is blocked by one or more HPR transactions (only single queue configuration)

In case an LPR is blocking, that port becomes the highest priority (priority0) and if there are no LPR credits available, hif_go2critical_lpr is asserted.

In case an HPR is blocking, that port becomes the highest priority (priority0) and if there are no HPR credits available, hif_go2critical_hpr is asserted.

If a VPW expires inside the write address queue, expired-VPW may be blocked by one or more NPW transactions.

In case an NPW is blocking, that port becomes the highest priority (priority0) and if there are no write credits available, hif_go2critical_wr is asserted.

This behavior allows the queue where the VPR / VPW is expired to be flushed as quickly as possible.

This mechanism is applied only for the expired-VPR and expired-VPW commands.

For single read address queue configurations, HPR commands may be blocked by LPR or VPR commands (not-expired); in this case, the port priority is not increased.

9.2.4.4.4 Urgent signalling

AXI sideband signals per port, arurgent (arurgentb, arurgentr when DDRMC_XPI_USE2RAQ_n is defined) and awurgent, when asserted (and as long as asserted), set a given port to the highest priority (priority0) and when applicable- cause the read/write direction to switch immediately in the Port Arbiter if enabled by PCFGR_n.rd_port_urgent_en and PCFGW_n.wr_port_urgent_en registers (see Port n Configuration Read Register (DDRC_MP_PCFGR_0) and Port n Configuration Write Register (DDRC_MP_PCFGW_0)).

Urgent signals also cause the hif_go2critical_wr / hif_go2critical_lpr / hif_go2critical_hpr signals to be asserted at the HIF interface, which in turn force the read/write direction switching in the DDRC, if enabled by the PCFG.go2critical_en register. Urgent signals are ignored by the PA if there are no requesters from the asserted address channel and port. Similarly, hif_go2critical* signals are ignored by the DDRC if the corresponding store is empty.

9.2.4.5 Refresh controls

Refresh can be issued using the auto-refresh feature in the DDRMC or using the direct software request of refresh command. The RFSHCTL3.dis_auto_refresh register bit selects the refresh method (see Refresh Control Register 0 (DDRC_RFSHCTL3)). When this bit is set to ‘1’, the DDRMC uses direct software request of refresh command and when it is set to ‘0’, the internal auto-refresh feature is used.

This section describes the following topics about refresh controls:

- Refresh using direct software request of refresh command
- Refresh using auto refresh feature inside the DDRMC

9.2.4.5.1 Refresh using direct software request of refresh command

Follow these steps to put the DDRMC in direct software request of refresh command mode:

1. Set the RFSHCTL3.dis_auto_refresh bit to 1. When the register bit set, the DDRMC checks for any pending refreshes. Any pending refreshes are issued right away using the ‘critical refresh’ feature inside the DDRMC. After these refreshes are issued, all the refresh timers inside the DDRMC are reset to 0. They are re-activated only when the auto-refresh feature is enabled.
2. The SoC core must keep track of the refresh requirements of the SDRAM.
3. The refresh command can be issued by setting the register bits DBGCMD.rank*_refresh to 1 (see Command Debug Register (DDRC_DBGCMD)). When the rank*_refresh request is stored in the DDRMC, the corresponding register

bit is automatically cleared. The SoC core can initiate a rank*_refresh operation only if DBGSTAT.rank*_refresh_busy is low. The DDRMC issues refresh to the SDRAM at the earliest.

4. Software-driven refresh commands for each rank are loaded into a 9-entry buffer, and are issued by the DDRMC on the DFI as soon as it is legal to do so (the DDRMC controller must wait tRFC(min) between each refresh request). If the buffer saturates, the DBGSTAT.rank*_refresh_busy remains asserted to prevent software from initiating further refreshes.
5. Refresh control through direct software request is not allowed when the controller is in DDR4 mode and Fine Granularity Refresh (FGR) operation is turned on. In this case, RFSHCTL3.dis_auto_refresh bit should be set to 0 and refresh control should be done using the auto-refresh feature.

NOTE

It is possible to give a burst of back-to-back refresh commands without polling the DBGSTAT.rank*_refresh_busy register field to reduce the time between the software refresh commands to a minimum with the following risk:

- If the refresh buffer is not empty when the burst starts or the burst is longer than 9 refresh commands, the buffer saturates (DBGSTAT.rank*_refresh_busy is asserted) and some APB writes given in the burst are not added to the buffer. This means that the number of APB writes do not reflect the number of REF commands sent out to the memory.

9.2.4.5.2 Refresh using auto refresh feature inside the DDRMC

The DDRMC provides advanced refresh controls. Besides fully-configurable refresh constraints (tRFC(min) and tREFI), the DDRMC can also be programmed to gather refreshes to each rank of SDRAM to reduce the bandwidth consumed by refreshes and to increase the likelihood that refreshes can be serviced during an idle period.

Fine-gain control of the refreshes ensures these benefits can be balanced against worst case latencies associated with servicing refreshes together. Staggered refresh timers for multi-rank configurations of the DDRMC allow transactions to continue to other ranks while refreshes are taking place to just one rank.

Following signals are related to refresh controls:

- RFSHCTL0.refresh_burst (see Refresh Control Register 0 (DDRC_RFSHCTL0))
- RFSHCTL0.per_bank_refresh (LPDDR2/LPDDR3 feature)
- RFSHCTL0.refresh_margin
- RFSHCTL0.refresh_to_x32.

- RFSHCTL1.refresh_timer0_start_value_x32
- RSHCTL1.refresh_timer1_start_value_x32
- RFSHCTL2.refresh_timer2_start_value_x32
- RFSHCTL2._refresh_timer3_start_value_x32
- RFSHCTL3.refresh_update_level
- RFSHCTL3.dis_auto_refresh

For more information on “x32” register fields, refer to Computation for all the registers in units of 32, 1024, or 4096 clocks .

The purposes of refresh control are to:

- Reduce the bandwidth impact of refresh cycles
- Increase the likelihood of refreshes being serviced during idle periods
- Provide fine-gain control of the trading-off the above benefits (to gather refreshes) versus the increased worst case latencies associated with gathering refreshes
- Allow traffic to flow to other ranks while a given rank is being refreshed (for multi-rank configurations of the DDRMC only)

To minimize the worst case impact of a forced refresh cycle, the DDRMC can be programmed to issue single refreshes at a time by forcing RFSHCTL0.refresh_burst = 0 (see Refresh Control Register 0 (DDRC_RFSHCTL0)). It can be programmed to burst up to 8 refreshes (RFSHCTL0.refresh_burst = 7) in non-DDR4 modes. Burst refresh can be used to minimize the bandwidth lost to closing pages for refresh and to increase the likelihood that refreshes can be serviced during idle periods. It can also be programmed to any number in between to trade-off the benefits of each.

9.2.4.5.2.1 Single refresh

When using single refresh (RFSHCTL0.refresh_burst = 0), the DDRMC issues refreshes every time the refresh timer (t_{REFI}) expires. This is the optimal mode of operation for systems that minimize the maximum latency associated with refresh cycles.

9.2.4.5.2.2 Burst refresh

When burst refresh is enabled (RFSHCTL0.refresh_burst > 0), the DDRMC issues refreshes in bursts of (RFSHCTL0.refresh_burst + 1) refreshes at one time. Bursting refreshes reduces the total latency associated with those refreshes by reducing the number of pre-charges and activates required for refresh, as banks must be precharged only once to perform the entire group of refreshes, instead of once for each refresh.

9.2.4.5.2.2.1 Speculative refresh

When burst refresh is enabled (RFSHCTL0.refresh_burst > 0), the user can also make use of a feature called speculative refresh.

Burst refresh is implemented by counting the number of times t_{REFI} expires and issuing a group of refreshes when that number reaches the refresh burst number. Once t_{REFI} has expired at least once, the DDRMC can also perform speculative refreshes. This is done by automatically inserting refreshes when the DDRMC is idle.

The RFSHCTL0.refresh_to_x32 register determines how long the DDRMC must be idle before considering inserting these speculative refreshes. Each time a speculative refresh is performed, the count of t_{REFI} expirations is decremented, and thereby increasing the time before a burst of refreshes is required. This also ensures that speculative refreshes never occur more often than is required to keep the SDRAM properly refreshed.

If a new read or write transaction is accepted by the DDRMC during speculative refresh, the DDRMC services it as soon as legally possible. Most often it entails waiting for the required NOP cycles after a refresh before performing an activate and then servicing the read or write. If the DDRMC has begun closing pages for a speculative refresh but has not yet issued the refresh when the new transaction arrives, the speculative refresh is cancelled.

9.2.4.5.2.3 Per-bank refresh (LPDDR2/LPDDR3 only)

If RFSHCTL0.per_bank_refresh is set to 1 (see Refresh Control Register 0 (DDRC_RFSHCTL0)), the DDRMC performs per-bank refreshes instead of all-bank refreshes. In this case, RFSHTMG.t_rfc_nom_x32 and RFSHTMG.t_rfc_min should be set to the appropriate values for per-bank refresh (tREFIpb and tRFCpb respectively). In this mode, the DDRMC keeps track of which bank is being refreshed at any time, and is able to schedule commands to other banks immediately before and after the per-bank refresh commands, resulting in potential efficiency gains.

9.2.4.6 ZQ calibration

This feature is applicable for DDR3 and LPDDR2 / LPDDR3. The DDRMC uses ZQ calibration command to calibrate SDRAM RON (Resistor ON) and ODT (On-Die Termination) values over PVT (Process, Voltage, Temperature). DDR3 SDRAMs need more time to calibrate R_{ON} and ODT at initialization and relatively less time to perform periodic calibrations. For more information, refer to DDR3 (JEDEC JESD79-3*), LPDDR2 (JEDEC JESD209-2*) and LPDDR3 (JEDEC JESD209-3*) Specifications.

9.2.4.6.1 DDR3 devices

ZQCL (ZQ Calibration Long) command is used as follows:

- To perform the initial calibration during the power-up initialization sequence. This command is allowed a timing period of tZQinit determined by INIT5. dev_zqinit_x32 (see SDRAM Initialization Register 5 (DDRC_INIT5)) to perform full calibration and the transfer of values.
- To perform long ZQ Calibration after exiting from self-refresh. This command is allowed a timing period of tZQOPER, determined by ZQCTL0._t_zq_long_nop. This command is automatically issued when ZQCTL0.dis_srx_zqcl (ZQCL0.dis_mpsmx_zqcl) is set to 0. To disable issuing ZQCL after self-refresh exit , set ZQCTL0.dis_srx_zqcl (ZQCTL0.dis_mpsmx_zqcl) to 1.

ZQCS (ZQ Calibration Short) command is used to perform periodic calibration to account for VT (Voltage / Temperature) variations. A shorter timing window is provided to perform calibration and transfer of values as defined by the timing parameter tZQCS (determined by ZQCTL0._t_zq_short_nop). ZQCS can be performed automatically on a regular interval or through direct software request. For more information about this, see Automatic and software initiated ZQCs.

In DDR3 mode, the ZQ Calibration commands are sent out encoded on the DFI command bus as mentioned in the JEDEC Specification. The DDRMC performs no other activities for the duration of tZQinit, tZQOPER and tZQCS. The quiet time on the SDRAM channel helps in accurate calibration of SDRAM R_{ON} and ODT. All banks are precharged and tRP met before the DDRMC issues the ZQ Calibration commands.

9.2.4.6.2 LPDDR2 / LPDDR3 devices

The DDRMC supports the ZQ Calibration commands that are supported by the JEDEC Specification.

ZQInit (ZQ Initial Calibration) command performs the initial calibration during the power-up initialization sequence. This command is allowed a time period of tZQinit (determined by INIT5.dev_zqinit_x32).

ZQCL (ZQ Long Calibration) command performs long ZQ Calibration after exiting from self-refresh mode. This command is allowed a timing period of tZQCL (determined by ZQCTL0._t_zq_long_nop). This command is issued automatically when ZQCTL0. dis_srx_zqcl is set to 0. To disable issuing of ZQCL after self-refresh exit, set ZQCTL0. dis_srx_zqcl to 1.

ZQCS (ZQ Short Calibration) command is used to perform periodic calibration to account for VT (Voltage/Temperature) variations. A shorter timing window is provided to perform calibration and transfer of values as defined by the timing parameter tZQCS

(determined by ZQCTL0._t_zq_short_nop). ZQCS can be performed automatically on a regular interval or through direct software request. For more information about this, see Automatic and software initiated ZQCs.

ZQReset (ZQ Calibration Reset) command is used to reset the R_{ON} calibration to a default accuracy of ±30% across process, voltage and temperature. This command is used to ensure R_{ON} accuracy of ±30% when ZQCS and ZQCL are not used and is allowed a time period of tZQRESET, determined by ZQCTL1.t_zq_reset_nop. The command is issued by using the registers ZQCTL2.zq_reset and ZQSTAT.zq_reset_busy (see ZQ Control Register 2 (DDRC_ZQCTL2) and ZQ Status Register (DDRC_ZQSTAT)). For more information about this, see LPDDR2 / LPDDR3 ZQ reset command.

In LPDDR2 / LPDDR3 mode, ZQ Calibration commands are sent out as Mode Register Write commands to the DRAM. The MRW is done to MR10 and the calibration codes for the different commands are as follows: ZQInit – 0xFF, ZQCL – 0xAB, ZQCS – 0x56, and ZQRest – 0xC3.

NOTE

For LPDDR2 / LPDDR3, it is possible to perform ZQ commands from software using the mode register interface (see Mode register reads and writes). However, the DDRMC does not implement the correct timing for such a command, so it is recommended not to do this, and instead to use either of the two methods mentioned below.

9.2.4.6.3 Automatic and software initiated ZQCs

The DDRMC issues a ZQCS command in the following two ways:

- **Automatic ZQCS by the DDRMC** : In this case, the DDRMC sends ZQCS commands to the SDRAM periodically. The interval is determined by ZQCTL1.t_zq_short_interval_x1024. This method is used if ZQCTL0.dis_auto_zq is set to 0 (see ZQ Control Register 0 (DDRC_ZQCTL0)).
- **ZQCS using direct software request**: In this case, the SoC core sends a ZQCS command through software by setting DBGCMD.zq_calib_short to 1. When the ZQCS request is stored in the DDRMC, the register bit is automatically cleared. It is recommended not to set DBGCMD.zq_calib_short signal in init, self-refresh, deep power-down operating modes or maximum power saving mode (MPSM). The SoC core can initiate a ZQCS operation only if DBGSTAT.zq_calib_short_busy is low. The DBGSTAT.zq_calib_short_busy signal goes high in the clock after the DDRMC accepts a ZQCS request. It goes low when the ZQCS operation is initiated in the DDRMC. For proper SDRAM operation, user/SoC core should schedule this command frequently. This method is used if ZQCTL0.dis_auto_zq is set to 1.

9.2.4.6.4 LPDDR2 / LPDDR3 ZQ reset command

In LPDDR2/LPDDR3 mode, ZQ Reset command is issued by setting ZQCTL2.zq_reset to 1 (see ZQ Control Register 2 (DDRC_ZQCTL2)). When the ZQ Reset operation is complete, the DDRMC automatically clears this register bit. It is recommended not to set this signal in init, self-refresh or deep power-down operating modes. The SoC core can initiate a ZQ Reset operation only if ZQSTAT.zq_reset_busy is low. This signal goes high in the clock after the DDRMC accepts a ZQ Reset request. It goes low when the ZQ reset command is issued to the SDRAM and the associated NOP period is completed. It is recommended not to perform ZQ Reset command when the ZQSTAT.zq_reset_busy signal is high.

9.2.4.7 SDRAM initialization sequence

A specific command sequence must be followed to properly initialize SDRAM modules. Much of the sequence is standard, but parts of it must be controlled differently according to the system configuration, clock speeds, and specific SDRAM parts in use. For this reason, the DDRMC provides a great deal of flexibility in how the initialization sequence is performed.

This section describes the initialization sequence and programmability provided by the DDRMC.

The list of registers associated with SDRAM initialization sequence are as follows:

- MSTR.ddr3
- MSTR.lpddr2
- MSTR.lpddr3
- INIT0.skip_dram_init
- INIT0.pre_cke_x1024
- INIT0.post_cke_x1024
- DRAMTMG3.t_mrd
- DRAMTMG3.t_mod (This is only used for DDR3 SDRAMs)
- INIT1.reg_ddrc_pre_ocd_x32
- INIT1.final_wait_x32
- INIT3.mr
- INIT3.emr
- INIT4.emr2
- INIT4.emr3
- INIT5.dev_zqinit_x32. This is used for DDR3 and LPDDR2/LPDDR3 SDRAMs.

For LPDDR2 / LPDDR3 configurations, the following registers are used for the SDRAM initialization sequence:

- INIT2.min_stable_clock_x1
- INIT2.idle_after_reset_x32
- INIT5.max_auto_init_x1024
- DRAMTMG3.t_mrw

When user / SoC core appropriately sets the inputs listed above, the DDRMC automatically performs the necessary sequence to properly initialize the DDR SDRAM devices. These registers must be set before bringing the DDRMC out of reset.

NOTE

In addition to initializing the SDRAM after a hard reset, the initialization sequence can also be used to bring the SDRAM out of self-refresh mode (see Self-refresh).

9.2.4.7.1 DDR3 initialization sequence

For DDR3, the DDRMC initialization state machine executes the following initialization sequence:

1. Power-up.
2. Maintain dfi_reset_n low for duration specified by INIT1.dram_rstn_x1024. Specification requires at least 200 us with stable power.
3. Issue NOP / deselect for duration specified by INIT0.pre_cke_x1024. Specification requires at least 500 µs.
4. Assert CKE and issue NOP/deselect for INIT0.post_cke_x1024 (specification requires at least t_{XPR}).
5. Issue MRS (mode register set) command to load MR2 with INIT4.emr2 value followed by NOP/deselect for duration of DRAMTMG3.t_mrd.
6. Issue MRS command to load MR3 with INIT4.emr3 followed by NOP/deselect for duration of DRAMTMG3.t_mrd.
7. Issue MRS command to load MR1 with INIT4.emr followed by NOP/deselect for duration of DRAMTMG3.t_mrd.
8. Issue MRS command to load MR0 with INIT3.mr followed by NOP/deselect for duration of DRAMTMG3.t_mod.
9. Issue ZQCL command to start ZQ calibration and wait for INIT5.dev_zqinit_x32.
10. Wait for INIT5.dev_zqinit_x32 counting to finish. Ensure wait from Step 8 is larger than t_{DLLK}.
11. DDR3 SDRAM is now ready for normal operation.

9.2.4.7.2 LPDDR2 / LPDDR3 initialization sequence

For LPDDR2 / LPDDR3, the DDRMC initialization state machine executes the following initialization sequence:

1. Power-up.
2. CKE is held low for a duration specified by INIT0.pre_cke_x1024. The clock is checked to be stable for duration specified by INIT2.min_stable_clock_x1 (minimum of 5 clock cycles) prior to the first low to high transition of CKE.
3. Assert CKE for INIT0.post_cke_x1024 (specification requires at least 200 µs).
4. A MRW (Reset) command is issued to MRW63 register. Values of MA<7:0> = 3FH and OP<7:0> = 00H is used for this command. The MRW reset command brings the device to the device auto-initialization (resetting) state in the power-on initialization sequence.
5. Issue NOP / deselect for duration specified by INIT2.idle_after_reset_x32 (specification requires 1 us minimum) and INIT5.max_auto_init_x1024 (specification requires maximum time of 10 µs).
6. An MRW ZQ initialization calibration command is issued to the memory to register MR10 to initiate the ZQ calibration. Values of MA<7:0> = 0AH and OP<7:0> = FFH is used for this command.
7. Issue NOP / deselect for duration specified by INIT5.dev_zqinit_x32 (specification requires a minimum time of 1 µs).
8. Program MR2 register by setting MR2 register to INIT3.emr followed by a NOP / deselect for a duration specified by DRAMTMG3.t_mrw (typical value of 5 clock cycles).
9. Program MR1 register by setting MR1 register to INIT5.mr followed by a NOP / deselect for a duration specified by DRAMTMG3.t_mrw (typical value of 5 clock cycles).
10. Program MR3 register by setting MR3 register to INIT4.emr2 followed by a NOP / deselect for a duration specified by DRAMTMG3.t_mrw (typical value of 5 clock cycles).
11. Schedule multiple all bank refresh.
12. Begin normal operation.

9.2.4.8 ODT control

By default, ODT to memories (dfi_odt) is driven to all zeroes. Figure 9-6 shows the timing diagram for ODT writes.

The register inputs control the following:

- **ODTMAP.rank*_wr_odt**: The value desired for ODT following a write command (see ODT / Rank Map Register (DDRC_ODTMAP)).

- **ODTCFG.wr_odt_delay**: The number of cycles to delay following a write command before driving the programmed values for write ODT, which depends primarily on CAS latency.
- **ODTCFG.wr_odt_hold**: The number of cycles to hold the programmed write value after it is first driven.
- **ODTMAP.rank*_rd_odt**: The value desired for ODT following a read command.
- **ODTCFG.rd_odt_delay**: The number of cycles to delay following a read command before driving the programmed values for write ODT, which depends primarily on CAS latency.
- **ODTCFG.rd_odt_hold**: The number of cycles to hold the programmed read value after it is first driven.

NOTE

- ODT for memories is not required for mDDR / LPDDR2. It is a DDR2/DDR3 specific feature. For mDDR/LPDDR2, set ODTMAP.rank*_wr_odt to all zeroes.

All of these must be set by the SoC core before taking the DDRMC out of reset. They are then applied to every read or write issued by the DDRMC.

The recommended settings for ODT related registers in each protocol are given in ODT Configuration Register (DDRC_ODTCFG).

To avoid overlap of ODT for reads and writes, the following constraints must be enforced:

- ODTCFG.wr_odt_hold ≤ MEMC_FREQ_RATIO * RANKCTL.diff_rank_wr_gap + MSTR.burst_rdwr
  - If burst chop is enabled (MSTR.burstchop = 1), use burst_rdwr / 2 in the above equation
- ODTCFG.rd_odt_hold ≤ MEMC_FREQ_RATIO*RANKCTL.diff_rank_rd_gap + MSTR.burst_rdwr
  - If burst chop is enabled (MSTR.burstchop = 1), use burst_rdwr / 2 in the above equation
- ODTCFG.wr_odt_delay + ODTCFG.wr_odt_hold ≤ ODTCFG.rd_odt_delay + MEMC_FREQ_RATIO * DRAMTMG2.wr2rd
- ODTCFG.rd_odt_delay + ODTCFG.rd_odt_hold ≤ ODTCFG.wr_odt_delay + MEMC_FREQ_RATIO * DRAMTMG2.rd2wr

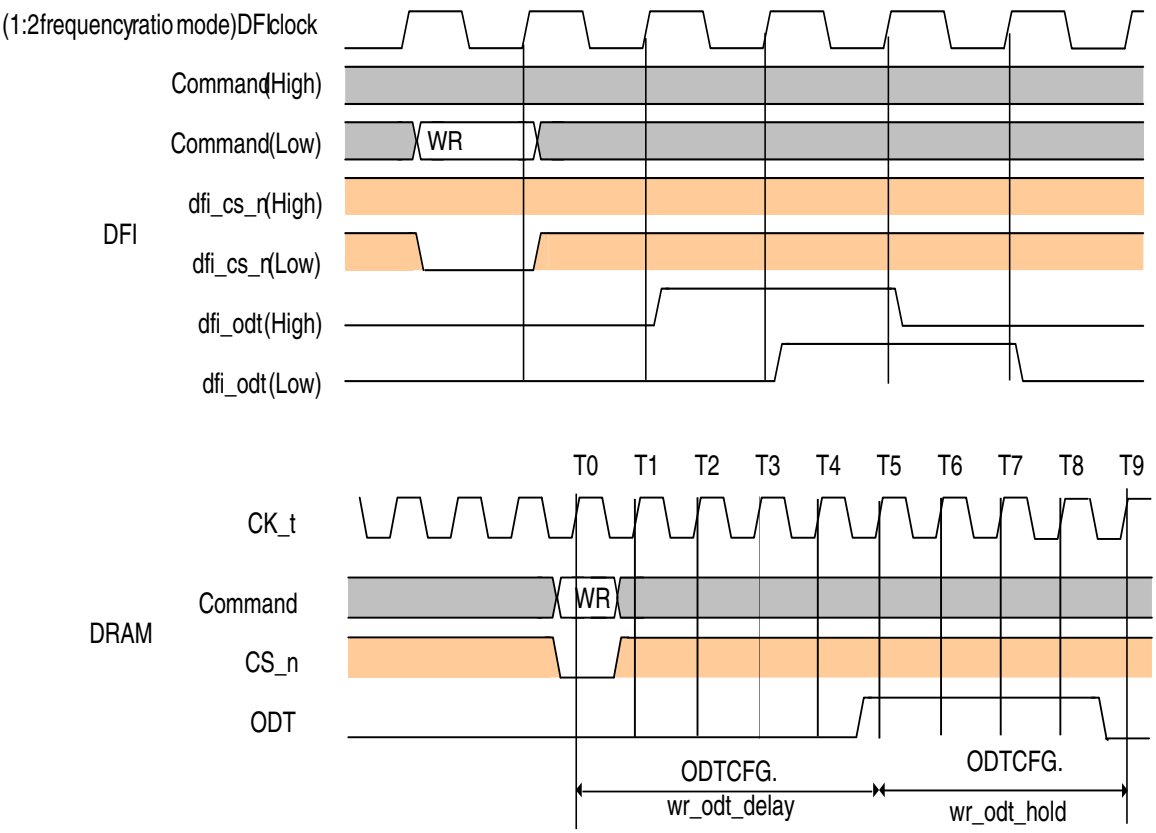


Figure 9-6. WR ODT Timing Diagram

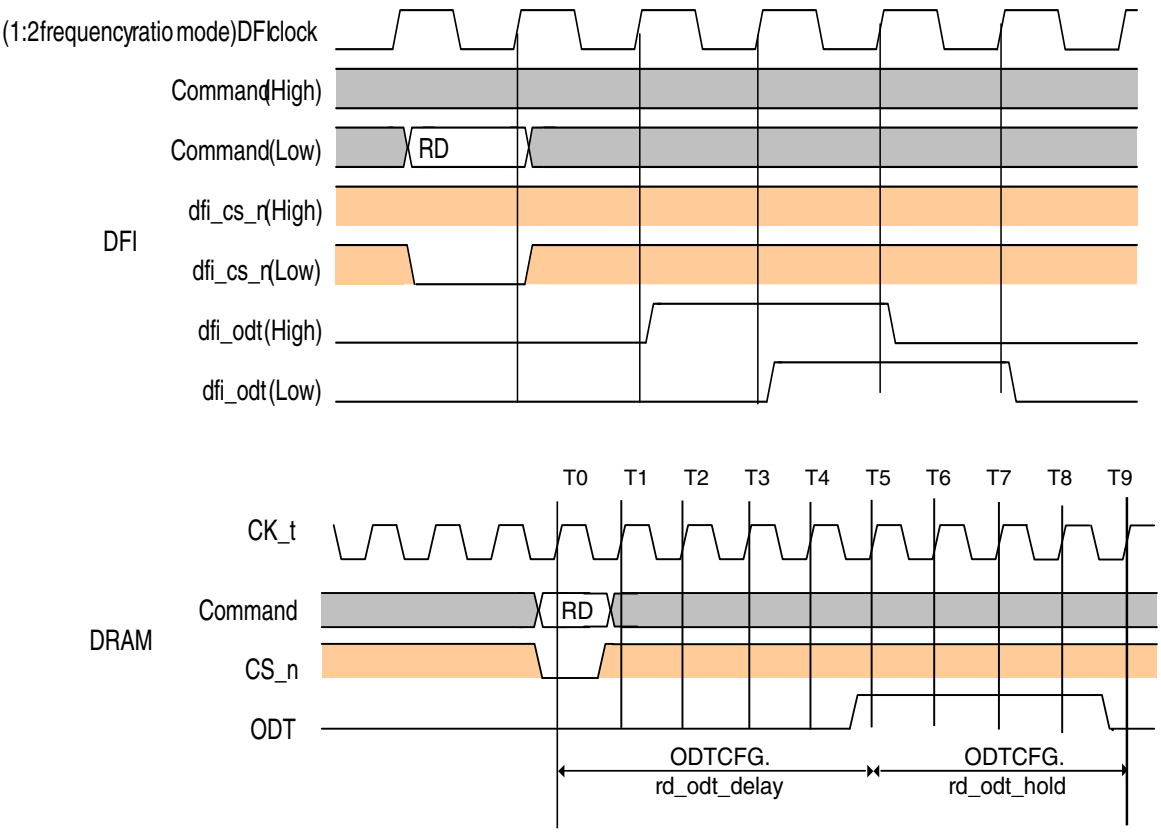


Figure 9-7. RD ODT Timing Diagram

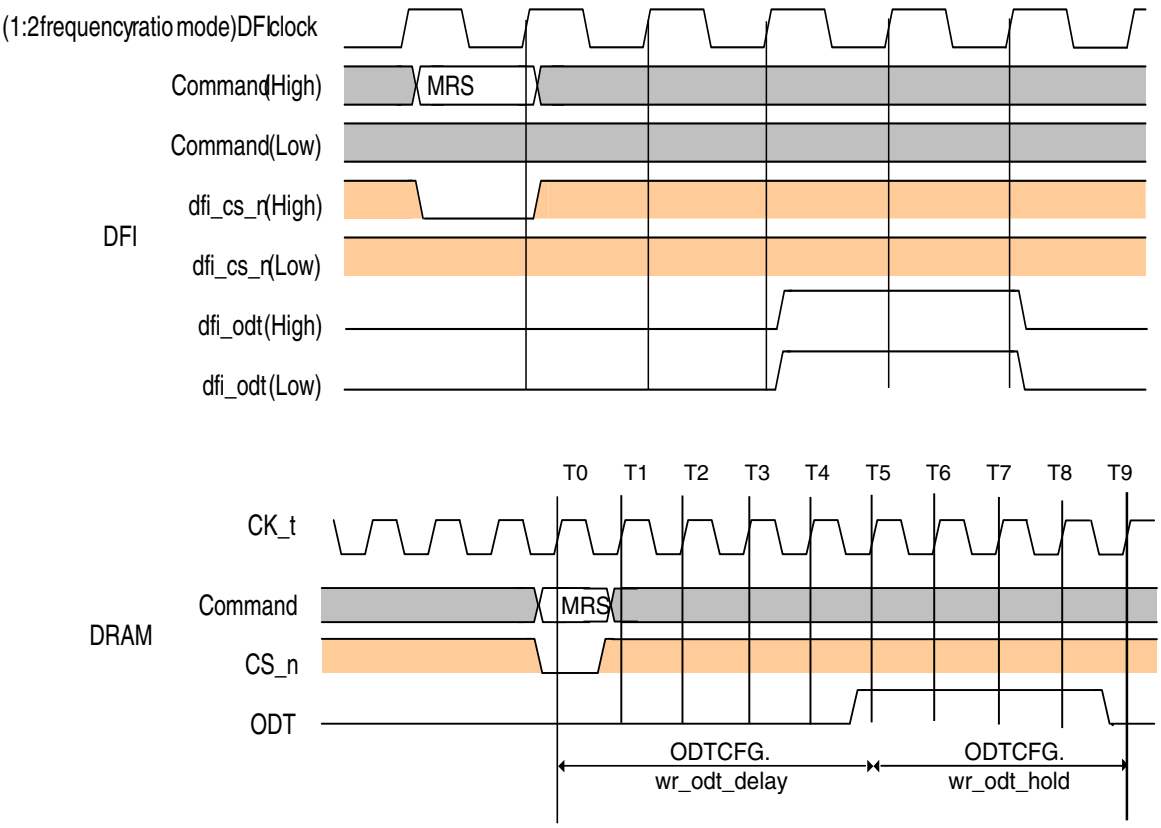


Figure 9-8. PDA ODT Timing Diagram

9.2.4.9 Power saving features

The DDRMC supports various methods to save power within the system:

- Power saving opportunities within the SDRAM. For more information about this, see SDRAM power saving features.

9.2.4.9.1 SDRAM power saving features

The DDRMC supports various SDRAM power-saving modes—precharge power-down, self-refresh, deep power-down, maximum power saving mode, and support for disabling clock to the DRAM via DFI_DRAM_CLK_DISABLE.

This section describes the following power-saving modes:

- Precharge power-down
- Self-refresh

- Deep power-down
- Assertion of dfi_dram_clk_disable

In multi-rank systems, these power-saving modes cannot be applied on a per-rank basis. If applied, they are always applied globally.

When enabled, the DDRMC automatically enters and exits precharge power-down mode based on a programmable idle timeout period.

Self-refresh can be entered/exited via three ways:
- Based on a programmable idle timeout period (similar to precharge power-down idle timeout)
- Explicitly controlled by the user via software
- Via the hardware low power interface(s)

In addition, DFI_DRAM_CLK_DISABLE can be asserted to disable the clocks to the DRAM. This can be done in self-refresh, power-down, deep power-down, and maximum power saving mode. It also includes clock stop if mDDR / LPDDR2 / LPDDR3 is used.

In relation to the hardware low power interface, optional support is provided to drive cactive_ddrc low under idle conditions in normal or power-down or automatic self-refresh modes. Optional support for power-down, self-refresh, and clock stop to be forcefully exited via CACTIVE_IN_DDRC = 1 is also provided.

Following registers are related to power saving features:
- DRAMTMG5.t_cke
- Precharge power-down controls:
  - PWRCTL.powerdown_en
  - PWRTMG.powerdown_to_x32
  - DRAMTMG1.t_xp
- Self-refresh controls:
  - PWRCTL.selfref_sw
  - PWRCTL.selfref_en
  - HWLPCTL.hw_lp_en
  - Hardware low power interface: csysreq_* / csysack_* / cactive_*
  - PWRTMG.selfref_to_x32
  - STAT.selfref_type
- Hardware low power interface:
  - HWLPCTL.hw_lp_exit_idle_en
  - HWLPCTL.hw_lp_idle_x32
- Deep power-down controls:
  - PWRCTL.deeppowerdown_en
  - PWRTMG.t_dpd_x4096
- Maximum power saving mode controls:

    - PWRCTL.mpsm_en
    - DRAMTMG11.t_mpx_lh
    - DRAMTMG11.t_mpx_s
    - DRAMTMG11.t_ckmpe
- Assertion of dfi_dram_clk_disable to disable the clocks to the DRAM controls:
    - PWRCTL.en_dfi_dram_clk_disable
    - DFITMG1.dfi_t_dram_clk_disable
    - DFITMG1.dfi_t_dram_clk_enable
    - DRAMTMG5.t_cksre
    - DRAMTMG5.t_cksrx
    - DRAMTMG6.t_ckpde
    - DRAMTMG6.t_ckpdx
    - DRAMTMG6.t_ckdpde
    - DRAMTMG6.t_ckdpdx
    - DRAMTMG6.t_ckcsx

For all the above power saving features, STAT.operating_mode signal can be used to monitor the current operating mode of the DDRMC.

NOTE

For more information on “x32 and x1024” register fields, refer to Computation for all the registers in units of 32, 1024, or 4096 clocks.

9.2.4.9.1.1 Precharge power-down

This section explains the following topics:

- Entering precharge power-down
- Exiting precharge power-down

9.2.4.9.1.1.1 Entering precharge power-down

When PWRCTL.powerdown_en = 1 (see Low Power Control Register (DDRC_PWRCTL)), the DDRMC automatically enters precharge power-down when the period specified by PWRTMG.powerdown_to_x32 has passed while the DDRMC is idle (except for issuing refreshes).

Entering precharge power-down mode involves the following steps:

1. Precharging (closing) all open pages. Pages are closed one-at-a-time in no specified order.
2. Waiting for tRP (row precharge) idle period.

3. Issuing the command to enter precharge power-down (NOP / deselect with CKE = 0).
4. This step occurs only if DFI low power interface for power-down is enabled (DFILPCFG0.dfi_lp_en_pd). Attempts an entry to low power mode via DFI low power interface with dfi_lp_wakeup set by DFILPCFG0.dfi_lp_wakeup_pd.

If the DDRMC receives a read or write request from the SoC core during Step 1 or Step 2 above, the power- down entry is immediately aborted. The same is true if PWRCTL.powerdown_en is driven to '0' during Step 1 or Step 2 . Once the power-down entry command is issued then proper power-down exit is required, as described in Exiting precharge power-down.

9.2.4.9.1.1.2 Exiting precharge power-down

Once the DDRMC has put the DDR SDRAM device(s) in precharge power-down mode, the DDRMC automatically performs the precharge power-down exit sequence for any of the following reasons:

- A refresh cycle is required to any rank in the system.
- The DDRMC receives a new request from the SoC core.
- A self-refresh entry is requested.
- PWRCTL.powerdown_en is set to '0' (see Low Power Control Register (DDRC_PWRCTL))

The DDRMC follows these steps when exiting precharge power-down mode:

1. Inserting any NOP / deselect commands required to satisfy the t_{CKE} requirement after entering precharge power-down.
2. This step occurs only if DFI low power mode entry during power-down entry is successful. Performs an exit from DFI low power mode.
3. Issuing the power-down exit command (NOP/deselect with CKE = 1). For multi-rank systems, all chip-selects are asserted so that all ranks exit precharge power-down simultaneously.
4. Issuing NOP/deselect for the period defined by t_{XP}.

NOTE

- DDR2: Fast Exit versus Slow Exit Active Power-down The DDR2 Specification describes two different variations on active power-down exit, depending on the programmed value of MR bit 12. As the DDRMC uses precharge power-down rather than active power-down, this programming has no effect on the DDRMC or the SDRAM devices.
- DDR3: Fast Exit versus Slow Exit Precharge Power-down The DDR3 Specification describes two different variations on precharge power-down exit, depending on the

programmed value of MR0 bit 12. If slow precharge power-down is used (MR0[12] = 0), DRAMTMG1.t_xp should be set to tXPDLL. If fast precharge power-down is used (MR0[12] = 1), DRAMTMG1.t_xp should be set to tXP.

- Other supported DDR protocols do not specify the difference between fast and slow power- down exit.

9.2.4.9.1.2 Self-refresh

This section explains the following topics:

- Entering self-refresh
- Exiting self-refresh

9.2.4.9.1.2.1 Entering self-refresh

The DDRMC puts the DDR SDRAM device(s) into self-refresh mode in the following cases:

- When the PWRCTL. selfref_en bit is set (see Low Power Control Register (DDRC_PWRCTL)) and no reads or writes are pending in the DDRMC for the period specified by PWRTMG.selfref_to_x32. This is referred to as automatic self-refresh. Dynamic changing of the PWRTMG.selfref_to_x32 register field is not allowed except when the DDRC is in reset (core_ddrc_rstn = 0).
- When the PWRCTL.selfref_sw bit is set. This is referred to as software self-refresh entry. The DDRC's hif_cmd_stall is driven high to stop new HIF commands from being accepted and existing core commands in DDRC are performed before sequence below . Note that, for arbiter configurations (MEMC_INCL_ARB = 1), commands mayoccurs be accepted by the AXI/AHB interface, but they do not enter the DDRC via the HIF as hif_cmd_stall = 1.
- When a hardware low power entry request occurs (on csysreq_ddrc / csysack_ddrc) with cactive_in_ddrc = 0, no outstanding commands, and as long as the DDRMC is in not in init or deep power-down or maximum power saving mode. This is referred to as an accepted hardware low power self-refresh entry. When accepted, the DDRC's hif_cmd_stall is driven high to stop new commands from being accepted and existing core commands in the DDRC are performed before the sequence mentioned below occurs.

Entering self-refresh mode involves the following steps:

1. If there is a self-refresh exit previously, wait for at least one refresh command (or 8 per-bank refresh commands if LPDDR2 / LPDDR3 per-bank refresh is enabled) to all active ranks. Auto-refresh logic must be enabled, or refresh should be issued using direct software requests of refresh command via DBGCMD.rank*_refresh.
2. Precharging (closing) all open pages. Pages are closed one-at-a-time in no specified order.
3. Waiting for t_{RP} (row precharge) idle period. If a new command is received on the HIF during this time, the self-refresh entry is aborted.
4. Issuing the command to enter self-refresh mode
5. This step occurs only if DFI low power interface for self-refresh is enabled (DFILPCFG0.dfi_lp_en_sr). Attempts an entry to low power mode via DFI low power interface with dfi_lp_wakeup set by DFILPCFG0.dfi_lp_wakeup_sr.

NOTE

STAT.selfref_type register field is 2'b11 if automatic self-refresh feature is the only cause of self- refresh. If software self-refresh or hardware low power self-refresh occurs, STAT.selfref_type = 2'b10.

Automatic self-refresh has lowest priority and is superseded by both software and hardware low power self-refresh. A software self-refresh entry means that a self-refresh exit occurs only if a software self-refresh exit occurs. Similarly, a hardware low power self-refresh entry means a self-refresh exit occurs only if a hardware low power self-refresh exit occurs. If both software and hardware low power self-refresh entry occurs, self-refresh exit occurs only if both software and hardware low power self-refresh exits occur.

9.2.4.9.1.2.2 Exiting self-refresh

The DDRMC takes the DDR SDRAM out of self-refresh mode under the following conditions:

- Whenever the PWRCTL.selfref_en input is de-asserted (see Low Power Control Register (DDRC_PWRCTL)), or new commands are received by the DDRMC, as long as automatic self-refresh is only cause for self-refresh entry (STAT.selfref_auto_flag = 1).
- Whenever the PWRCTL.selfref_sw bit is de-asserted. This is referred to as software self-refresh exit.
- When a hardware low power exit request occurs (on csysreq_ddrc / csysack_ddrc). This is referred to as hardware low power self-refresh exit.

Exiting self-refresh mode involves the following steps:

1. Inserting any NOP / deselect commands required to satisfy the t_{CKE} / t_{CKERSR} requirements after entering self-refresh.

2. This step occurs only if DFI low power mode entry during self-refresh entry is successful. Performs an exit from DFI low power mode.
3. Issuing the self-refresh exit command (refresh with CKE = 1).
4. Issuing NOP / deselect for the period defined by tX_{SDLL} / t_{XSNR} / t_{XSRD}.

9.2.4.9.1.3 Deep power-down

This power saving mode is applicable for LPDDR and LPDDR2 devices only. This section explains the following topics:

- Entering deep power-down
- Exiting deep power-down

9.2.4.9.1.3.1 Entering deep power-down

By setting the PWRCTL.deeppowerdown_en bit (see Low Power Control Register (DDRC_PWRCTL)), the user can put the SDRAM device(s) into deep power-down mode, if all the following conditions are true:

- The period specified by PWRTMG.powerdown_to_x32 has passed while the DDRMC is idle (except for issuing refreshes)
- PWRCTL.selfref_sw = 0
- PWRCTL.selfref_en = 0
- If HWLPCTL.hw_lp_en = 1, DPD is entered only when the hardware low power interface has completed a self-refresh exit. (This can be checked by observing STAT.operating_mode and STAT.selfref_type).
- If HWLPCTL.hw_lp_exit_idle_en = 1, DPD is entered only when all bits of

cactive_in_ddrc = 0 Entering Deep power-down involves the following steps:

1. Precharging (closing) all open pages. Pages are closed one-at-a-time in no specified order.
2. Waiting for tRP (row precharge) idle period.
3. Issuing the command to enter deep power-down. For multi-rank systems, all chip-selects are asserted so that all ranks enter deep power-down simultaneously. The deep power-down entry commands are as follows:

   Command for mDDR: CKE = 0, CSN = 0, RAS = 1, CAS = 1, WE = 0

   Command for LPDDR2 / LPDDR3: CKE = 0, CSN = 0, CA0 = 1, CA1 = 1, CA2 = 0

4. This step occurs only if DFI low power interface for deep power-down is enabled (DFILPCFG0.dfi_lp_en_dpd). It attempts an entry to low power mode via DFI low power interface with dfi_lp_wakeup set by DFILPCFG0.dfi_lp_wakeup_dpd.

If the DDRMC receives a read or write request from the SoC core during Step 1 or Step 2 above, the deep power-down entry is immediately aborted. The same is true if PWRCTL.deep_powerdown_en is driven to ‘0’ during Step 1 or Step 2 . Once the deep power-down entry command is issued, proper deep power-down exit is required, as described in the following section.

NOTE

Contents of SDRAM may be lost upon entry into deep power-down mode.

9.2.4.9.1.3.2 Exiting deep power-down

Once the DDRMC puts the DDR SDRAM device(s) in deep power-down mode, the DDRMC automatically exits deep power-down and re-runs the initialization sequence when PWRCTL.deeppowerdown_en is reset to ‘0’ (see Low Power Control Register (DDRC_PWRCTL)). An exit from DFI low power mode is performed prior to exiting deep power-down (this occurs only if DFI low power mode entry during deep power-down entry is successful).

9.2.4.9.1.4 Assertion of dfi_dram_clk_disable

Assertion of dfi_dram_clk_disable occurs only if PWRTL.en_dfi_dram_clk_disable = 1. dfi_dram_clk_disable is also dependent on the operating mode:

- In DDR2 / DDR3, dfi_dram_clk_disable can be asserted only in self-refresh mode.
- In mDDR / LPDDR2 / LPDDR3, dfi_dram_clk_disable can be asserted in the following modes:
  - In Self-refresh
  - In Power-down
  - In Deep power-down
  - In Normal mode. This is “Clock Stop” feature.

The timing of the assertion and de-assertion of dfi_dram_clk_disable in various modes is as follows:

- In Self-refresh:
  - Asserted at least DFITMG0.dfi_t_ctrl_delay + DRAMTMG5.t_cksre - DFITMG1.dfi_t_dram_clk_disable cycles after SRE command.
  - Deasserted at least DFITMG1.dfi_t_dram_clk_enable + DRAMTMG5.t_cksrx - DFITMG0.dfi_t_ctrl_delay cycles before SRX command.

■ In Power-down:

- Asserted at least DFITMG0.dfi_t_ctrl_delay + DRAMTMG7.t_ckpde - DFITMG1.dfi_t_dram_clk_disable cycles after PDE command.
- Deasserted at least DFITMG1.dfi_t_dram_clk_enable + DRAMTMG7.t_ckpdx - DFITMG0.dfi_t_ctrl_delay cycles before PDX command.

■ In Deep power-down:

- Asserted at least DFITMG0.dfi_t_ctrl_delay + DRAMTMG6.t_ckdpde - DFITMG1.dfi_t_dram_clk_disable cycles after DPDE command.
- Deasserted at least DFITMG1.dfi_t_dram_clk_enable + DRAMTMG6.t_ckdpdx - DFITMG0.dfi_t_ctrl_delay cycles before DPDX command.

- In Normal mode (Clock Stop):
  - Asserted at least DFITMG0.dfi_t_ctrl_delay - DFITMG0.dfi_t_dram_clk_disable cycles after any command other than SRE / PDE / DPDE.
  - Deasserted at least DFITMG1.dfi_t_dram_clk_enable + DRAMTMG6.t_ckcsx - DFITMG0.dfi_t_ctrl_delay cycles before any command other than SRX / PDX / DPDX.
- In Maximum power saving mode:
  - Asserted at least DFITMG0.dfi_t_ctrl_delay + DRAMTMG11.t_ckmpe - DFITMG1.dfi_t_dram_clk_disable cycles after MPSME.
  - Deasserted at least DFITMG1.dfi_t_dram_clk_enable + DRAMTMG5.t_cksrx - DFITMG0.dfi_t_ctrl_delay cycles before MPSMX.

9.2.4.9.1.5 DLL-off mode (DDR3)

DLL-off mode enables DDR3 SDRAMs to be operated at low frequencies. The DDRMC supports DLL-off mode, and transitions between DLL-on and DLL-off mode, under software control. To enable DLL-off mode from initialization, the following register fields must be set:

- INIT3.emr[0], so that the SDRAM mode register is set for DLL-off mode
- MSTR.dll_off_mode = 1
- If using the Synopsys DWC DDR PHY, it should be put in PLL-bypass mode

To perform a transition between DLL-off and DLL-on modes, the software must implement the sequence specified in the JEDEC specification. Full details of the software sequence is provided in the User Guide.

NOTE

DBI, geardown and programmable preamble are not supported in DLL-off mode. Also, geardown, C/A parity and CAL mode must be disabled when moving from DLL-off to DLL-on.

9.2.4.9.2 Software sequence for removal of clocks

Software can be used to keep the SDRAM in self-refresh. The AXI and DDRC clocks can be removed when in self-refresh by following the sequence described in the Table 9-9.

Table 9-9. Software Clock Removal Sequence

Step	Description	Comment
1	Write 0 to PCTRL_n.port_en	Blocks AXI port(s) from taking anymore transactions
2	Poll PSTAT.rd_port_busy_n = 0 Poll PSTAT.wr_port_busy_n = 0	Wait unit all AXI ports are idle
3	Write 0 to SBRCTL.scrub_en	Disable SBR, only required if SBR instantiated
4	Poll SBRSTAT.scrub_busy=0	Indicates that there are no outstanding SBR read commands, only required if SBR instantiated
5	Write 1 to PWRCTL.selfref_sw	Cause system to move to self-refresh state
6	Poll STAT.selfref_type= 2'b10	Wait until self-refresh state entered
7	Remove AXI clock(s)	
8[1]	Remove DDRC core clock	

1. Prior to this step, if PHY allows, it is recommended to disable generation of PHY initiated Update requests (dfi_phyupd_req). This avoids the possibility of the DDRMC missing dfi_phyupd_req assertions while its clock has been removed. The DDRMC cannot respond via dfi_phyupd_ack as its clock has been removed. If using a Synopsys PHY, this can be done via the PUB's DSGCR.PUREN.

The clocks should be re-enabled by following the sequence described in Table 9-10.

Table 9-10. Re-enabling the Clocks

Step	Description	Comment
1	Enable AXI clock(s)	
2[1]	Enable DDRC core clock	
3	Write 0 to PWRCTL.selfref_sw	Cause system to exit from self-refresh state
4	Poll STAT.selfref_type = 2'b00	Wait until self-refresh state is exited
5	Write 1 to PCTRL_n.port_en	AXI port(s) are no longer blocked from taking transactions
6	Write 1 to SBRCTL.scrub_en	Enable SBR if desired, only required if SBR instantiated

1. After this step, it is recommended to re-enable generation of PHY initiated Update requests (dfi_phyupd_req). If using a Synopsys PHY, this can be done via the PUB's DSGCR.PUREN.

.

9.2.4.9.3 Power removal flow

Table 9-11. Power Removal

Step	Description	Comment
1	Write 0 to PCTRL_n.port_en	Blocks AXI port(s) from taking anymore transactions

Table continues on the next page...

Table 9-11. Power Removal (continued)

Step	Description	Comment
2	Poll PSTAT.rd_port_busy_n = 0 Poll PSTAT.wr_port_busy_n = 0	Wait unit all AXI ports are idle
3	Write 0 to SBRCTL.scrub_en	Disable SBR, only required if SBR instantiated
4	Poll SBRSTAT.scrub_busy = 0	Indicates that there are no outstanding SBR read commands, only required if SBR instantiated
5	Write 1 to PWRCTL.selfref_sw	Cause system to move to self-refresh state
6	Poll STAT.selfref_type= 2'b10	Wait until self-refresh state entered
7	Place IOs in retention mode	Refer to relevant PHY databook for more information
8	Remove power	

Table 9-12. Re-enabling the power

Step	Description	Comment
1	Enable Power	
2	Reset controller / PHY by driving core_ddrc_rstn = 1'b0, aresetn_n = 1'b0, presetn = 'b0	
3	Remove APB reset, presetn = 1'b1, and re-reprogram the registers to pre-power removal values	
4	Program INIT0.skip_dram_init = 2'b11	Skips the DRAM init routine and starts up in self-refresh mode
5	Programs PWRCTL.selfref_sw = 1'b1	Keeps the controller in self-refresh mode
6	Program DFIMISC.dfi_init_complete_en to 1'b0	PHY initialization needs to be rerun so set to 0 until initialization complete
7	Remove the core reset core_ddrc_rstn = 1'b1 aresetn_n = 1'b1	
8	Remove IOs from retention mode	Refer to relevant PHY databook for more information
9	Run PHY initialization / training as required	Refer to relevant PHY databook for more information
10	Program DFIMISC.dfi_init_complete_en to 1'b1	Indicates to controller that PHY has completed re- training / initialization
11	Program PWRCTL.selfref_sw = 1'b0	trigger self-refresh exit
12	Poll STAT.selfref_type = 2'b00	Wait until self-refresh state is exited
13	Poll STAT.operating_mode for Normal Mode entry	
14	Write PCTRL.port_en = 1	AXI port(s) are no longer blocked from taking transactions
15	Write 1 to SBRCTL.scrub_en	Enable SBR if desired, only required if SBR instantiated

9.2.4.10 Mode register reads and writes

This section explains how to perform mode register reads and writes via software. Mode Register Reads (MRR) are applicable only to LPDDR2 / LPDDR3, and are used to read configuration and status data from mode registers in the SDRAM. Mode Register Writes (MRW or MRS) are applicable to all supported DDR protocols, and are used to write configuration data to mode registers in the SDRAM.

Access to the mode register is initiated by programming the MRCTRL0 and MRCTRL1 registers (see Mode Register Read / Write Control Register 0 (DDRC_MRCTRL0) and Mode Register Read / Write Control Register 1 (DDRC_MRCTRL1)). This must be done in three steps:

1. Poll MRSTAT.mr_wr_busy until it is '0'. This checks that there is no outstanding MR transaction. No writes should be performed to MRCTRL0 and MRCTRL1 if MRSTAT.mr_wr_busy = 1.
2. Write the MRCTRL0.mr_type, MRCTRL0.mr_addr, MRCTRL0.mr_rank and (for MRWs) MRCTRL1.mr_data to define the MR transaction.
3. In a separate APB transaction, write the MRCTRL0.mr_wr to 1. This bit is self-clearing, and triggers the MR transaction. The DDRMC then asserts the MRSTAT.mr_wr_busy while it performs the MR transaction to SDRAM, and no further accesses can be initiated until it is deasserted.

Some additional notes on mode register transactions are as follows:

- MRW transactions can be specified for either any single rank, or any combination of several ranks, by programming MRCTRL1.mr_rank.
- MRR transactions should only be performed to one rank at a time, to avoid bus contention.
- When a MRR is performed, the mode register contents are available on hif_mrr_data[7:0], qualified by hif_mrr_data_valid, after the mode register read command is issued by the DDRMC to the SDRAM.

9.2.4.10.1 Mode register write

The Mode Register Write (MRW) command is used to write configuration data to mode registers.

For non-LPDDR2 / non-LPDDR3 SDRAMs, MRW on <MR addr> with <MR Data> is triggered by writing MRCTRL0.mr_wr = '1', MRCTRL0.mr_type = '0', MRCTRL0.mr_addr = <MR addr>, MRCTRL1.mr_data = <MR Data> and MRCTRL0.mr_rank = <MR rank> for one clock.

For LPDDR2 / LPDDR3 SDRAMs, MRW on <MR addr> with <MR Data> is triggered by writing MRCTRL0.mr_wr = ‘1’, MRCTRL0.mr_type = ‘0’, MRCTRL1.mr_data[15:8] = <MR addr>, MRCTRL1.mr_data[7:0] = <MR Data> and MRCTRL0.mr_rank = <MR rank> for one clock.

In both the cases, the signal MRSTAT.mr_wr_busy is asserted by the DDRMC while the mode register access is performed, and no further accesses can be initiated until it is deasserted.

For user-initiated MR writes, the user can request mode registers programming operation for either any single rank / all even ranks / all odd ranks only. To implement this, a NUM_RANKS bits wide register is provided to indicate the rank of mode registers programming operation (MRCTRL1.mr_rank [NUM_RANKS - 1: 0]).

9.2.5 Register Descriptions

9.2.5.1 Description introduction

All registers are addressable on 32-bit boundaries; each unused bit or address location is reserved for future use and read back as 0. The register access abbreviations are as follows:

- R – Read only
- W – Write only
- R/W – Read / Write
- R/W1C - Read / Write 1 to clear
- R/WSC – Read / Write Self Clear. When a R/WSC bit is set to 1, the bit remains set until it is cleared by hardware after an internal event.

Some register fields might use the following optional attribute.

Table 9-13. Optional Attributes

Attribute	Description
Testable	As defined by the IP-XACT specification. Possible values are unconstrained, untestable, read only, write as read, restore. Untestablemeans that this field is untestable by a simple automated register test. For example, the read-write access of the register is controlled by a pin or another register. Read only means that you should not write to this register; only read from it. This might apply for a register that modifies the contents of another register.

NOTE

Computation for all the registers in units of 32, 1024, or 4096 clocks

This note refers to register fields whose names include “x32”, “x1024”, or “x4096”.

The DDRMC contains a timer which issues a pulse once every 32 clock cycles, another which issues a pulse once every 1024 clock cycles and another which issues a pulse once every 4096 clock cycles. “x32” register fields count pulses of the 32-cycle timer, “x1024” register fields count pulses of the 1024-cycle timer, and “x4096” register fields count pulses of the 4096-cycle timer. These timers are shared by the logic for all of these register fields, and the various periods start without any guaranteed relation to the phase of this timer.

Therefore for “x32” register fields:

- A programmed value of 0 gives 0 clock cycles
- A programmed value of 1 gives a 1 to 32 cycle delay
- A programmed value of 2 gives a 33 to 64 cycle delay and so on

For “x1024” register fields:

- A programmed value of 0 gives no delay
- A programmed value of 1 gives a 1 to 1,024 cycle delay
- A programmed value of 2 gives a 1,025 to 2,048 cycle delay and so on

For “x4096” register fields:

- A programmed value of 0 gives no delay
- A programmed value of 1 gives a 1 to 4,096 cycle delay
- A programmed value of 2 gives a 4,097 to 8,192 cycle delay and so on

For minimum delays, the control signal must be set to the smallest number for which the resulting delay is ALWAYS ≥ the required delay; for maximum delays or nominal refresh, this must be set such that the delay is ALWAYS ≤ the required delay.

This scheme is used to reduce the gate count that would be required to enforce these constraints more precisely.

NOTE

Most of the SDRAM timing registers are in terms of clock cycles. But the value given in an SDRAM datasheet may be in terms of ns / ps. These are primarily minimum timings. Before

using them to calculate value to program in the register, it is expected that these are divided by tCK (SDRAM clock period) and rounded up to the next integer value if division does not result in a whole integer value.

Exceptions to this rule are related to maximum timings, where these should be rounded down rather than rounded up, if division does not result in a whole integer value.

These include:

- RFSHTMG.t_rfc_nom_x32
- DRAMTMG0.t_ras_max

Certain SDRAM timing registers clarify this by using the following functions in their descriptions:

- RoundUp (X / Y) means that if X divided by Y does not result in a whole integer value, the result should be rounded up to the next integer value. For example, RoundUp (1000 / 150) = 7, RoundUp (1050 / 150) = 7.
- RoundDown (X / Y) means that if X divided by Y does not result in a whole integer value, the result should be rounded down to the next integer value. For example, RoundDown (1000 / 150) = 6, RoundDown (1050 / 150) = 7.

9.2.5.2 DDRC Registers

DDRC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
307A_0000	Master Register (DDRC_MSTR)	32	R/W	0000_0000h	9.2.5.2.1/ 2178
307A_0004	Operating Mode Status Register (DDRC_STAT)	32	R/W	0000_0000h	9.2.5.2.2/ 2181
307A_0010	Mode Register Read / Write Control Register 0 (DDRC_MRCTRL0)	32	R/W	0000_0000h	9.2.5.2.3/ 2182
307A_0014	Mode Register Read / Write Control Register 1 (DDRC_MRCTRL1)	32	R/W	0000_0000h	9.2.5.2.4/ 2184
307A_0018	Mode Register Read / Write Status Register (DDRC_MRSTAT)	32	R	0000_0000h	9.2.5.2.5/ 2185
307A_0020	Temperature Derate Enable Register (DDRC_DERATEEN)	32	R/W	0000_0000h	9.2.5.2.6/ 2186

Table continues on the next page...

DDRC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
307A_0024	Temperature Derate Interval Register (DDRC_DERATEINT)	32	R/W	0000_0000h	9.2.5.2.7/ 2187
307A_0030	Low Power Control Register (DDRC_PWRCTL)	32	R/W	0000_0000h	9.2.5.2.8/ 2188
307A_0034	Low Power Timing Register (DDRC_PWRTMG)	32	R/W	0000_0000h	9.2.5.2.9/ 2189
307A_0038	Hardware Low Power Control Register (DDRC_HWLPCTL)	32	R/W	0000_0000h	9.2.5.2.10/ 2191
307A_0050	Refresh Control Register 0 (DDRC_RFSHCTL0)	32	R/W	0000_0000h	9.2.5.2.11/ 2193
307A_0054	Refresh Control Register 1 (DDRC_RFSHCTL1)	32	R/W	0000_0000h	9.2.5.2.12/ 2195
307A_0060	Refresh Control Register 0 (DDRC_RFSHCTL3)	32	R/W	0000_0000h	9.2.5.2.13/ 2196
307A_0064	Refresh Timing Register (DDRC_RFSHTMG)	32	R/W	0000_0000h	9.2.5.2.14/ 2197
307A_00D0	SDRAM Initialization Register 0 (DDRC_INIT0)	32	R/W	0000_0000h	9.2.5.2.15/ 2198
307A_00D4	SDRAM Initialization Register 1 (DDRC_INIT1)	32	R/W	0000_0000h	9.2.5.2.16/ 2199
307A_00D8	SDRAM Initialization Register 2 (DDRC_INIT2)	32	R/W	0000_0000h	9.2.5.2.17/ 2200
307A_00DC	SDRAM Initialization Register 3 (DDRC_INIT3)	32	R/W	0000_0000h	9.2.5.2.18/ 2201
307A_00E0	SDRAM Initialization Register 4 (DDRC_INIT4)	32	R/W	0000_0000h	9.2.5.2.19/ 2202
307A_00E4	SDRAM Initialization Register 5 (DDRC_INIT5)	32	R/W	0000_0000h	9.2.5.2.20/ 2202
307A_00F4	Rank Control Register (DDRC_RANKCTL)	32	R/W	0000_0000h	9.2.5.2.21/ 2203
307A_0100	SDRAM Timing Register 0 (DDRC_DRAMTMG0)	32	R/W	0000_0000h	9.2.5.2.22/ 2205
307A_0104	SDRAM Timing Register 1 (DDRC_DRAMTMG1)	32	R/W	0000_0000h	9.2.5.2.23/ 2207
307A_0108	SDRAM Timing Register 2 (DDRC_DRAMTMG2)	32	R/W	0000_0000h	9.2.5.2.24/ 2208
307A_010C	SDRAM Timing Register 3 (DDRC_DRAMTMG3)	32	R/W	0000_0000h	9.2.5.2.25/ 2210
307A_0110	SDRAM Timing Register 4 (DDRC_DRAMTMG4)	32	R/W	0000_0000h	9.2.5.2.26/ 2211
307A_0114	SDRAM Timing Register5 (DDRC_DRAMTMG5)	32	R/W	0000_0000h	9.2.5.2.27/ 2212
307A_0118	SDRAM Timing Register 6 (DDRC_DRAMTMG6)	32	R/W	0000_0000h	9.2.5.2.28/ 2214

Table continues on the next page...

DDRC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
307A_011C	SDRAM Timing Register 7 (DDRC_DRAMTMG7)	32	R/W	0000_0000h	9.2.5.2.29/ 2215
307A_0120	SDRAM Timing Register 8 (DDRC_DRAMTMG8)	32	R/W	0000_0000h	9.2.5.2.30/ 2216
307A_0180	ZQ Control Register 0 (DDRC_ZQCTL0)	32	R/W	0000_0000h	9.2.5.2.31/ 2218
307A_0184	ZQ Control Register 1 (DDRC_ZQCTL1)	32	R/W	0000_0000h	9.2.5.2.32/ 2220
307A_0188	ZQ Control Register 2 (DDRC_ZQCTL2)	32	R/W	0000_0000h	9.2.5.2.33/ 2221
307A_018C	ZQ Status Register (DDRC_ZQSTAT)	32	R/W	0000_0000h	9.2.5.2.34/ 2222
307A_0190	DFI Timing Register 0 (DDRC_DFITMG0)	32	R/W	0000_0000h	9.2.5.2.35/ 2224
307A_0194	DFI Timing Register 1 (DDRC_DFITMG1)	32	R/W	0000_0000h	9.2.5.2.36/ 2226
307A_0198	DFI Low Power Configuration Register 0 (DDRC_DFILPCFG0)	32	R/W	0000_0000h	9.2.5.2.37/ 2227
307A_01A0	DFI Update Register 0 (DDRC_DFIUPD0)	32	R/W	0000_0000h	9.2.5.2.38/ 2230
307A_01A4	DFI Update Register 1 (DDRC_DFIUPD1)	32	R/W	0000_0000h	9.2.5.2.39/ 2231
307A_01A8	DFI Update Register 2 (DDRC_DFIUPD2)	32	R/W	0000_0000h	9.2.5.2.40/ 2232
307A_01AC	DFI Update Register 3 (DDRC_DFIUPD3)	32	R/W	0000_0000h	9.2.5.2.41/ 2233
307A_01B0	DFI Miscellaneous Control Register (DDRC_DFIMISC)	32	R/W	0000_0000h	9.2.5.2.42/ 2234
307A_0200	Address Map Register 0 (DDRC_ADDRMAP0)	32	R/W	0000_0000h	9.2.5.2.43/ 2235
307A_0204	Address Map Register 1 (DDRC_ADDRMAP1)	32	R/W	0000_0000h	9.2.5.2.44/ 2236
307A_0208	Address Map Register 2 (DDRC_ADDRMAP2)	32	R/W	0000_0000h	9.2.5.2.45/ 2237
307A_020C	Address Map Register 3 (DDRC_ADDRMAP3)	32	R/W	0000_0000h	9.2.5.2.46/ 2238
307A_0210	Address Map Register 4 (DDRC_ADDRMAP4)	32	R/W	0000_0000h	9.2.5.2.47/ 2240
307A_0214	Address Map Register 5 (DDRC_ADDRMAP5)	32	R/W	0000_0000h	9.2.5.2.48/ 2241
307A_0218	Address Map Register 6 (DDRC_ADDRMAP6)	32	R/W	0000_0000h	9.2.5.2.49/ 2243
307A_0240	ODT Configuration Register (DDRC_ODTCFG)	32	R/W	0000_0000h	9.2.5.2.50/ 2245

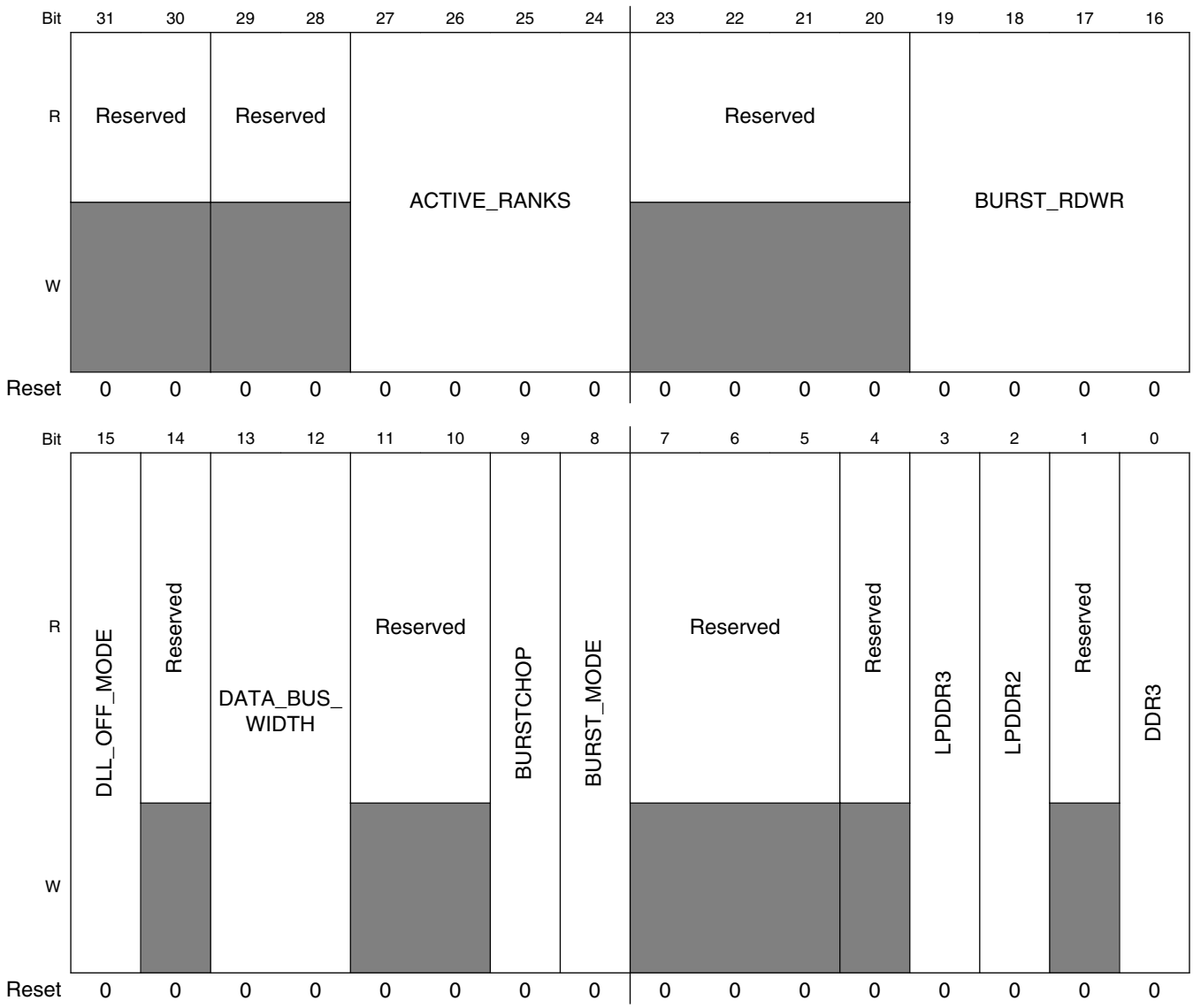
Table continues on the next page...

DDRC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
307A_0244	ODT / Rank Map Register (DDRC_ODTMAP)	32	R/W	0000_0000h	9.2.5.2.51/ 2246
307A_0250	Scheduler Control Register (DDRC_SCHED)	32	R/W	0000_0000h	9.2.5.2.52/ 2248
307A_0254	Scheduler Control Register 1 (DDRC_SCHED1)	32	R/W	0000_0000h	9.2.5.2.53/ 2250
307A_025C	High Priority Read CAM Register 1 (DDRC_PERFHPR1)	32	R/W	0000_0000h	9.2.5.2.54/ 2250
307A_0264	Low Priority Read CAM Register 1 (DDRC_PERFLPR1)	32	R/W	0000_0000h	9.2.5.2.55/ 2251
307A_026C	Write CAM Register 1 (DDRC_PERFWR1)	32	R/W	0000_0000h	9.2.5.2.56/ 2252
307A_0274	Variable Priority Read CAM Register 1 (DDRC_PERFVPR1)	32	R/W	0000_0000h	9.2.5.2.57/ 2252
307A_0278	Variable Priority Write CAM Register 1 (DDRC_PERFVPW1)	32	R/W	0000_0000h	9.2.5.2.58/ 2253
307A_0300	Debug Register 0 (DDRC_DBG0)	32	R/W	0000_0000h	9.2.5.2.59/ 2254
307A_0304	Debug Register 1 (DDRC_DBG1)	32	R/W	0000_0000h	9.2.5.2.60/ 2255
307A_0308	CAM Debug Register (DDRC_DBGCAM)	32	R/W	0000_0000h	9.2.5.2.61/ 2257
307A_030C	Command Debug Register (DDRC_DBGCMD)	32	R/W	0000_0000h	9.2.5.2.62/ 2260
307A_0310	Status Debug Register (DDRC_DBGSTAT)	32	R/W	0000_0000h	9.2.5.2.63/ 2262
307A_0320	Software Register Programming Control Enable (DDRC_SWCTL)	32	R/W	0000_0000h	9.2.5.2.64/ 2264
307A_0324	Software Register Programming Control Status (DDRC_SWSTAT)	32	R	0000_0000h	9.2.5.2.65/ 2265

9.2.5.2.1 Master Register (DDRC_MSTR)

Address: 307A_0000h base + 0h offset = 307A_0000h



DDRC_MSTR field descriptions

Field	Description
31–30 Reserved	This field is reserved. Reserved for future use
29–28 Reserved	This field is reserved. Reserved for future use
27–24 ACTIVE_RANKS	Only present for multi-rank configurations. Each bit represents one rank. For two-rank configurations, only bits [25:24] are present. • 1 - populated • 0 - unpopulated

Table continues on the next page...

DDRC_MSTR field descriptions (continued)

Field	Description
	LSB is the lowest rank number. For 2 ranks following combinations are legal: • 01 - One rank • 11 - Two ranks • Others - Reserved. For 4 ranks following combinations are legal: • 0001 - One rank • 0011 - Two ranks • 1111 - Four ranks **Value After Reset**: "(MEMC_NUM_RANKS == 4) ? 0xF :((MEMC_NUM_RANKS == 2) ? 0x3 : 0x1)" **Exists**: MEMC_NUM_RANKS > 1
23–20 Reserved	This field is reserved. Reserved for future use
19–16 BURST_RDWR	All other values are reserved. This controls the burst size used to access the SDRAM. This must match the burst length mode register setting in the SDRAM. Burst length of 2 is not supported with AXI ports when MEMC_BURST_LENGTH is 8. **Value After Reset**: 0x4 **Exists**: Always SDRAM burst length used: 0001 Burst length of 2 (only supported for mDDR) 0010 Burst length of 4 0100 Burst length of 8 1000 Burst length of 16 (only supported for mDDR and LPDDR2)
15 DLL_OFF_MODE	Set to 1 when the DDRC and DRAM has to be put in DLL-off mode for low frequency operation. Set to 0 to put DDRC and DRAM in DLL-on mode for normal frequency operation. **Value After Reset**: 0x0 **Exists**:MEMC_DDR3 == 1
14 Reserved	This field is reserved. Reserved for future use
13–12 DATA_BUS_ WIDTH	Note that half bus width mode is only supported when the SDRAM bus width is a multiple of 16, and quarter bus width mode is only supported when the SDRAM bus width is a multiple of 32 and the configuration parameter MEMC_QBUS_SUPPORT is set. Bus width refers to DQ bus width (excluding any ECC width). **Value After Reset**: 0x0 **Exists**: Always Selects proportion of DQ bus width that is used by the SDRAM: 00 Full DQ bus width to SDRAM 01 Half DQ bus width to SDRAM 10 Quater DQ bus width to SDRAM 11 Reserved
11–10 Reserved	This field is reserved.

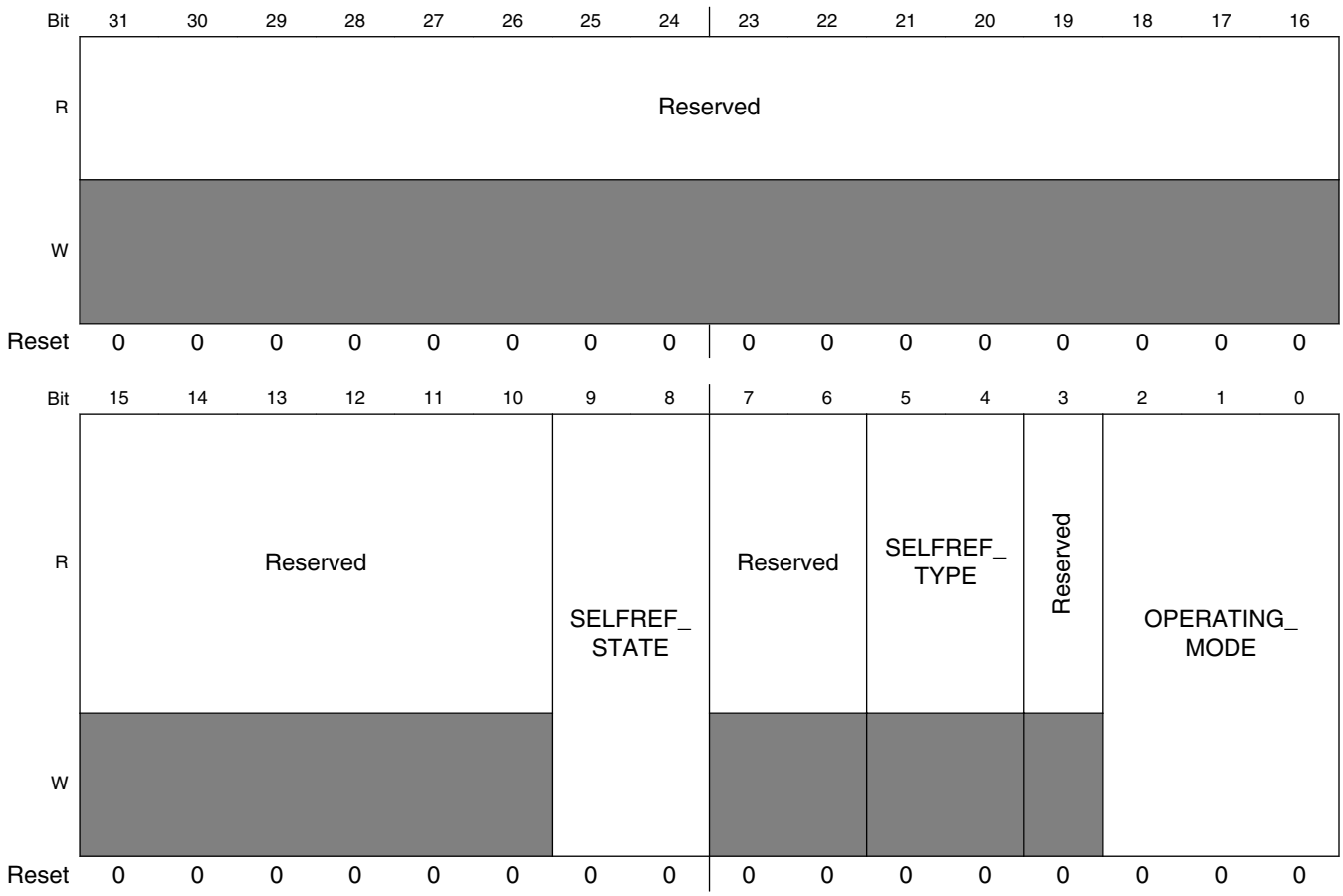
Table continues on the next page...

DDRC_MSTR field descriptions (continued)

Field	Description
9 BURSTCHOP	When set, enable burst-chop in DDR3. This is only supported in HIF configurations (DDRC_INCL_ARB not set) using full bus width mode (MSTR.data_bus_width = 00).**Value After Reset**: 0x0 **Exists**: MEMC_DDR3 == 1
8 BURST_MODE	Indicates Burst Mode **Value After Reset**: 0x0 **Exists**: DDRC_INCL_ARB == 0 0 Sequential burst mode 1 Interleaved burst mode
7–5 Reserved	This field is reserved. Reserved for future use
4 Reserved	This field is reserved.
3 LPDDR3	Select LPDDR3 SDRAM Present only in designs configured to support LPDDR3. **Value After Reset**: 0x0 **Exists**:MEMC_LPDDR3 == 1 1 LPDDR3 SDRAM device in use 0 Non-LPDDR3 device in use
2 LPDDR2	Select LPDDR2 SDRAM Present only in designs configured to support LPDDR2. **Value After Reset**: 0x0 **Exists**:MEMC_LPDDR2 == 1 1 LPDDR2 SDRAM device in use 0 Non-LPDDR2 device in use
1 Reserved	This field is reserved. Reserved for future use
0 DDR3	Only present in design that support DDR3. **Value After Reset**: "(MEMC_DDR3 == 1) ? 0x1 : 0x0" **Exists**:MEMC_DDR3 == 1 Select DDR3 SDRAM 1 DDR3 SDRAM device in use 0 Non-DDR3 device in use

9.2.5.2.2 Operating Mode Status Register (DDRC_STAT)

Address: 307A_0000h base + 4h offset = 307A_0004h



DDRC_STAT field descriptions

Field	Description
31–10 Reserved	This field is reserved. Reserved for future use
9–8 SELFREF_ STATE	Self refresh state This indicates self refresh or self refresh power down state for LPDDR4. This register is used for frequency change and MRR / MRW access during self-refresh. **Value After Reset**:0x0 **Exists**: MEMC_LPDDR4==1 00 SDRAM is not in Self Refresh. 01 Self refresh 1 10 Self refresh power down 11 Self refresh 2
7–6 Reserved	This field is reserved. Reserved for future use

Table continues on the next page...

DDRC_STAT field descriptions (continued)

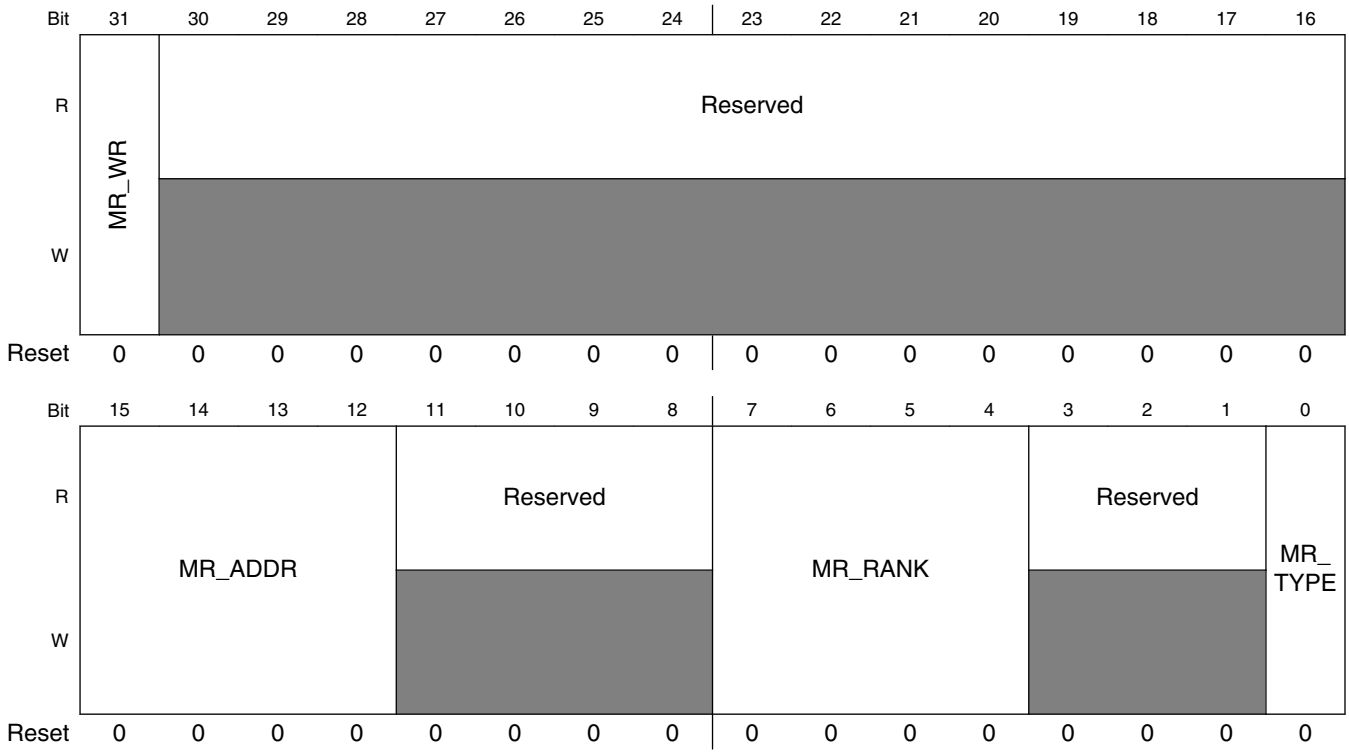
Field	Description
5–4 SELFREF_TYPE	Flags if self-refresh is entered. If it is under automatic, it is self-refresh control only or not.**Value After Reset**: 0x0 **Exists**: Always 00 SDRAM is not in self-refresh. If retry is enabled by CRCPARCTRL1.crc_parity_retry_enable, this also indicates SRE command is still in parity error window or retry is in-progress. 11 SDRAM is in self-refresh and self-refresh is caused by automatic self-refresh only. If retry is enabled, this guarantees SRE command is executed correctly without parity error. 10 SDRAM is in self-refresh and self-refresh is not caused solely under automatic self-refresh control. It could have been caused by Hardware Low Power Interface and/or Software (reg_ddrc_selfref_sw). If retry is enabled, this guarantees SRE command is executed correctly without parity error.
3 Reserved	This field is reserved. Reserved for future use
OPERATING_ MODE	Operating Mode This is 3-bits wide in configurations with mDDR / LPDDR2 / LPDDR3 support and 2-bits in all other configurations.Non-mDDR / LPDDR2 / LPDDR3 and non-DDR4 designs: • 00 - Init • 01 - Normal • 10 - Power-down • 11 - Self-refresh mDDR / LPDDR2 / LPDDR3 or DDR4 designs: • 000 - Init • 001 - Normal • 010 - Power-down • 011 - Self-refresh • 1XX - Deep power-down / Maximum Power Saving Mode **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.3 Mode Register Read / Write Control Register 0 (DDRC_MRCTRL0)

NOTE

Do not enable more than one of the following fields simultaneously:
- RCD_INIT_EN
- PDA_EN
- MPR_EN

Address: 307A_0000h base + 10h offset = 307A_0010h



DDRC_MRCTRL0 field descriptions

Field	Description
31 MR_WR	Setting this register bit to 1 triggers a mode register read or write operation. When the MR operation is complete, the DDRC automatically clears this bit. The other register fields of this register must be written in a separate APB transaction, before setting this mr_wr bit. It is recommended NOT to set this signal if in Init, Deep power-down or MPSM operating modes. **Value After Reset**: 0x0 **Exists**: Always
30–16 Reserved	This field is reserved. Reserved for future use
15–12 MR_ADDR	Address of the mode register that is to be written to Do not Care for LPDDR2 / LPDDR3 (see MRCTRL1.mr_data formode register addressing in LPDDR2 / LPDDR3).This signal is also used for writing to control words of RDIMMs. In that case, it corresponds to the bank address bits sent to the RDIMM **Value After Reset**: 0x0 **Exists**: Always 0000 MR0 0001 MR1 0010 MR2 0011 MR3 0100 MR4 0101 MR5 0110 MR6 0111 MR7

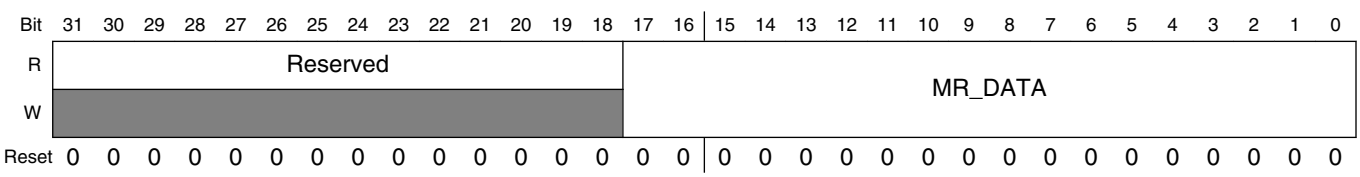
Table continues on the next page...

DDRC_MRCTRL0 field descriptions (continued)

Field	Description
11–8 Reserved	This field is reserved. Reserved for future use
7–4 MR_RANK	Controls which rank is accessed by MRCTRL0.mr_wr. Normally, it is desired to access all ranks, so all bits should be set to 1. However, for multi-rank UDIMMs / RDIMMs which implement address mirroring, it may be necessary to access ranks individually. Examples (assume DDRC is configured for 4 ranks): • 0x1 - select rank 0 only • 0x2 - select rank 1 only • 0x5 - select ranks 0 and 2 • 0xA - select ranks 1 and 3 • 0xF - select ranks 0, 1, 2 and 3 **Value After Reset**: "(MEMC_NUM_RANKS == 4) ? 0xF :((MEMC_NUM_RANKS == 2) ? 0x3 : 0x1)" **Exists**: Always
3–1 Reserved	This field is reserved.
0 MR_TYPE	Indicates whether the mode register operation is read or write. Only used for LPDDR2 / LPDDR3.**Value After Reset**: 0x0 **Exists**: MEMC_LPDDR2 == 1 0 Write 1 Read

9.2.5.2.4 Mode Register Read / Write Control Register 1 (DDRC_MRCTRL1)

Address: 307A_0000h base + 14h offset = 307A_0014h

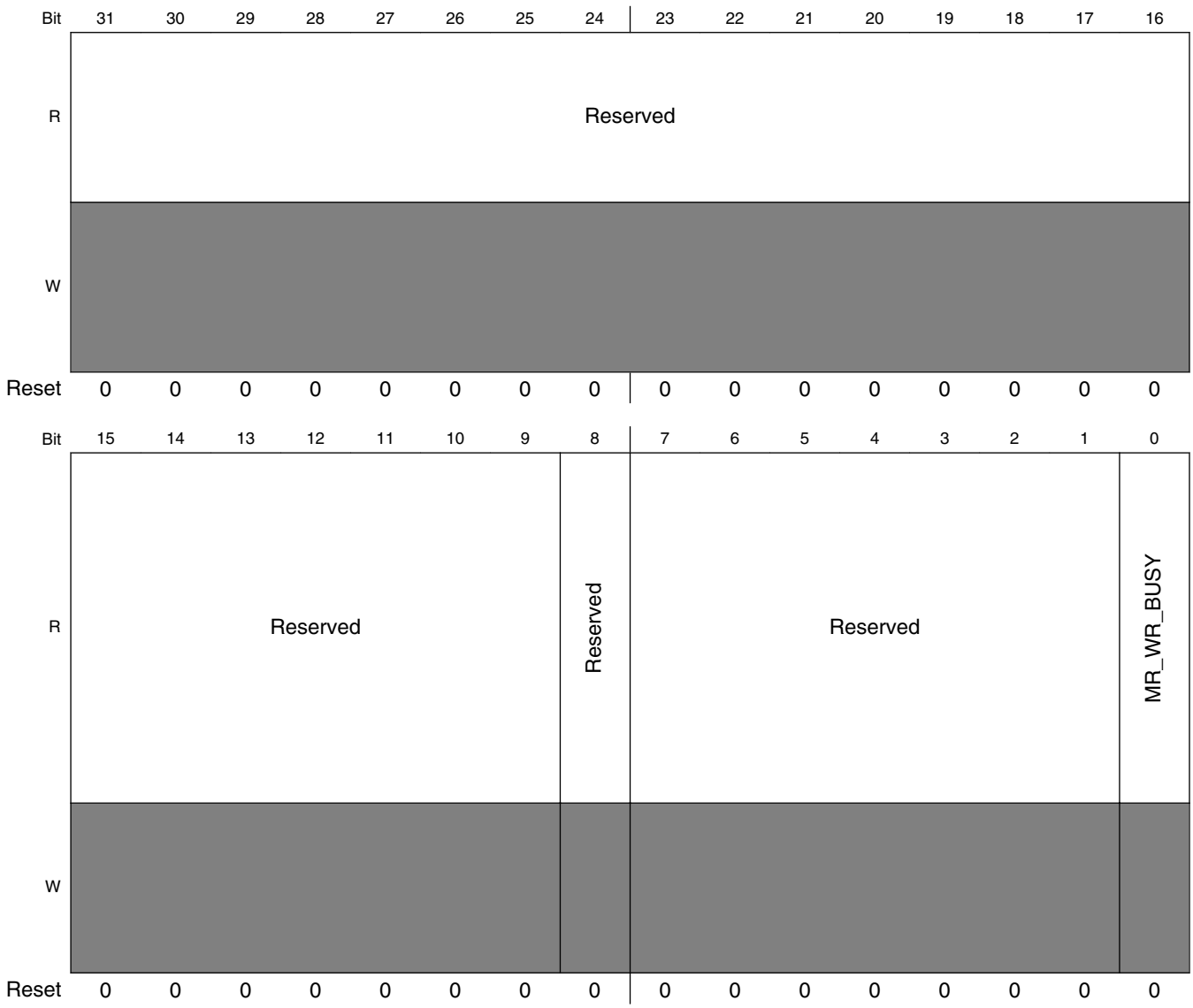


DDRC_MRCTRL1 field descriptions

Field	Description
31–18 Reserved	This field is reserved. Reserved for future use
MR_DATA	Mode register write data for all non- LPDDR2 / non-LPDDR3 modes. For LPDDR2 / LPDDR3, MRCTRL1 [15:0] are interpreted as [15:8] MR Address. [7:0] MR data for writes, do not care for reads. This is 18-bit wide in configurations with support and 16-bit in all other configurations. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.5 Mode Register Read / Write Status Register (DDRC_MRSTAT)

Address: 307A_0000h base + 18h offset = 307A_0018h



DDRC_MRSTAT field descriptions

Field	Description
31–9 Reserved	This field is reserved. Reserved for future use
8 Reserved	This field is reserved.
7–1 Reserved	This field is reserved. Reserved for future use
0 MR_WR_BUSY	The SoC core may initiate a MR write operation only if this signal is low.This signal goes high in the clock after the DDRC accepts the MRW / MRR request. It goes low when the MRW / MRR command is issued

Table continues on the next page...

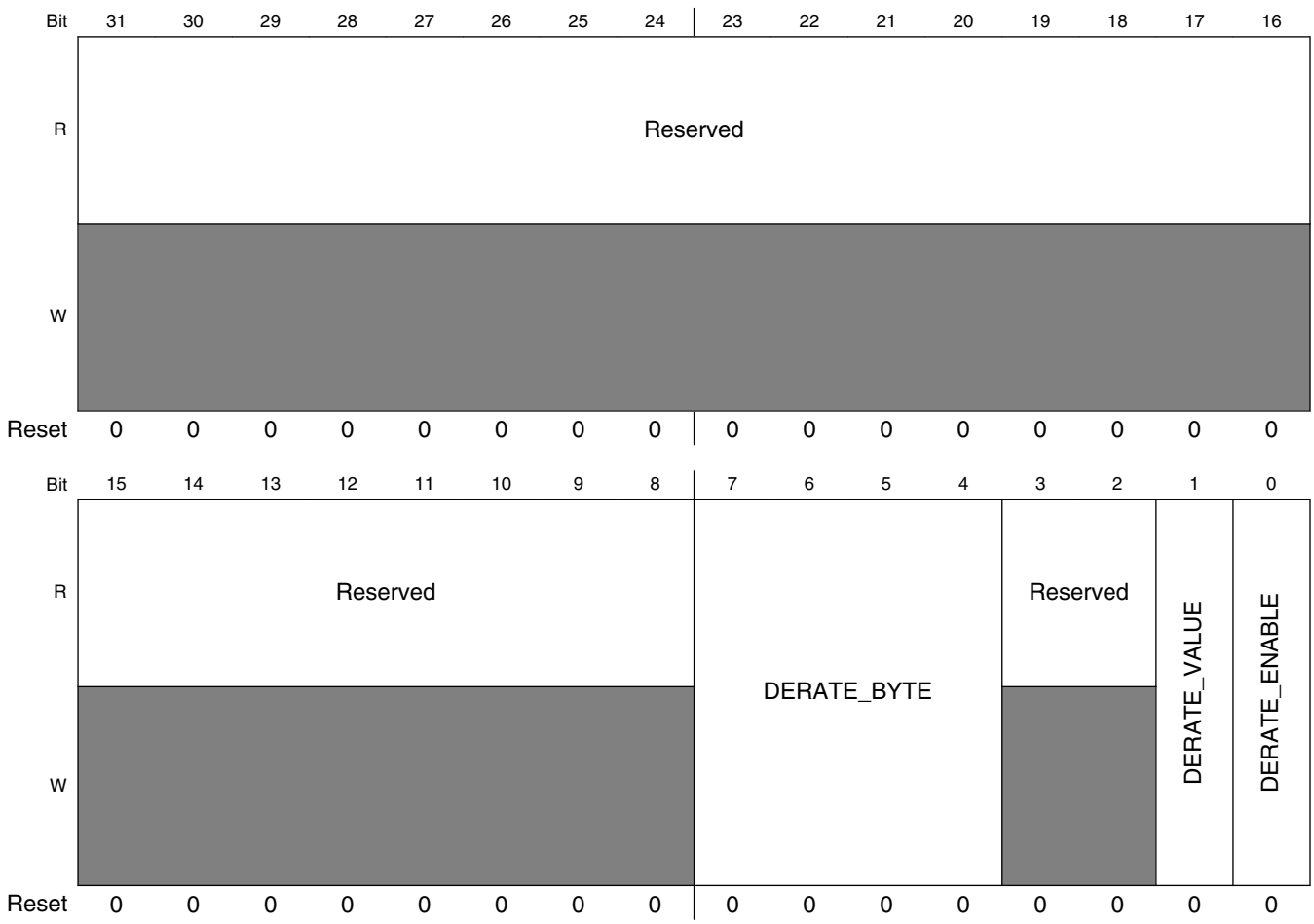
DDRC_MRSTAT field descriptions (continued)

Field	Description
	to the SDRAM. It is recommended not to perform MRW / MRR commands when 'MRSTAT.mr_wr_busy' is high. **Value After Reset**: 0x0 **Exists**: Always 0 Indicates that the SoC core can initiate a mode register write operation 1 Indicates that mode register write operation is in progress

9.2.5.2.6 Temperature Derate Enable Register (DDRC_DERATEEN)

Exists: MEMC_LPDDR2==1

Address: 307A_0000h base + 20h offset = 307A_0020h



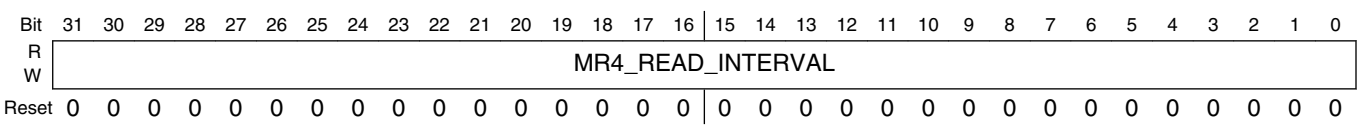
DDRC_DERATEEN field descriptions

Field	Description
31–8 Reserved	This field is reserved. Reserved for future use
7–4 DERATE_BYTE	Derate byte present only in designs configured to support LPDDR2 or LPDDR3 indicates which byte of the MRR data is used for derating. The maximum valid value depends on MEMC_DRAM_TOTAL_DATA_WIDTH. **Value After Reset**: 0x0 **Exists**: MEMC_LPDdR2==1
3–2 Reserved	This field is reserved. Reserved for future use.
1 DERATE_VALUE	Derate Value Present only in designs configured to support LPDDR2 or LPDDR3 Set to 0 for all LPDDR2 speed grades as derating value of +1.875 ns is less than a core_ddrc_core_clk period. It can be 0 or 1 for LPDDR3, depending if +1.875 ns is less than a core_ddrc_core_clk period or not. **Value After Reset**: 0x0 **Exists**:MEMC_LPDDR2==1 0 Derating uses + 1 1 Derating uses + 2
0 DERATE_ ENABLE	Enable Derating Present only in designs configured to support LPDDR2 / LPDDR3. This field must be set to '0' for non-LPDDR2 / LPDDR3 mode. **Value After Reset**: 0x0 **Exists**: MEMC_LPDDR2==1 0 Timing parameter derating is disabled 1 Timing parameter derating is enabled using MR4 read value

9.2.5.2.7 Temperature Derate Interval Register (DDRC_DERATEINT)

Exists: MEMC_LPDDR2==1

Address: 307A_0000h base + 24h offset = 307A_0024h

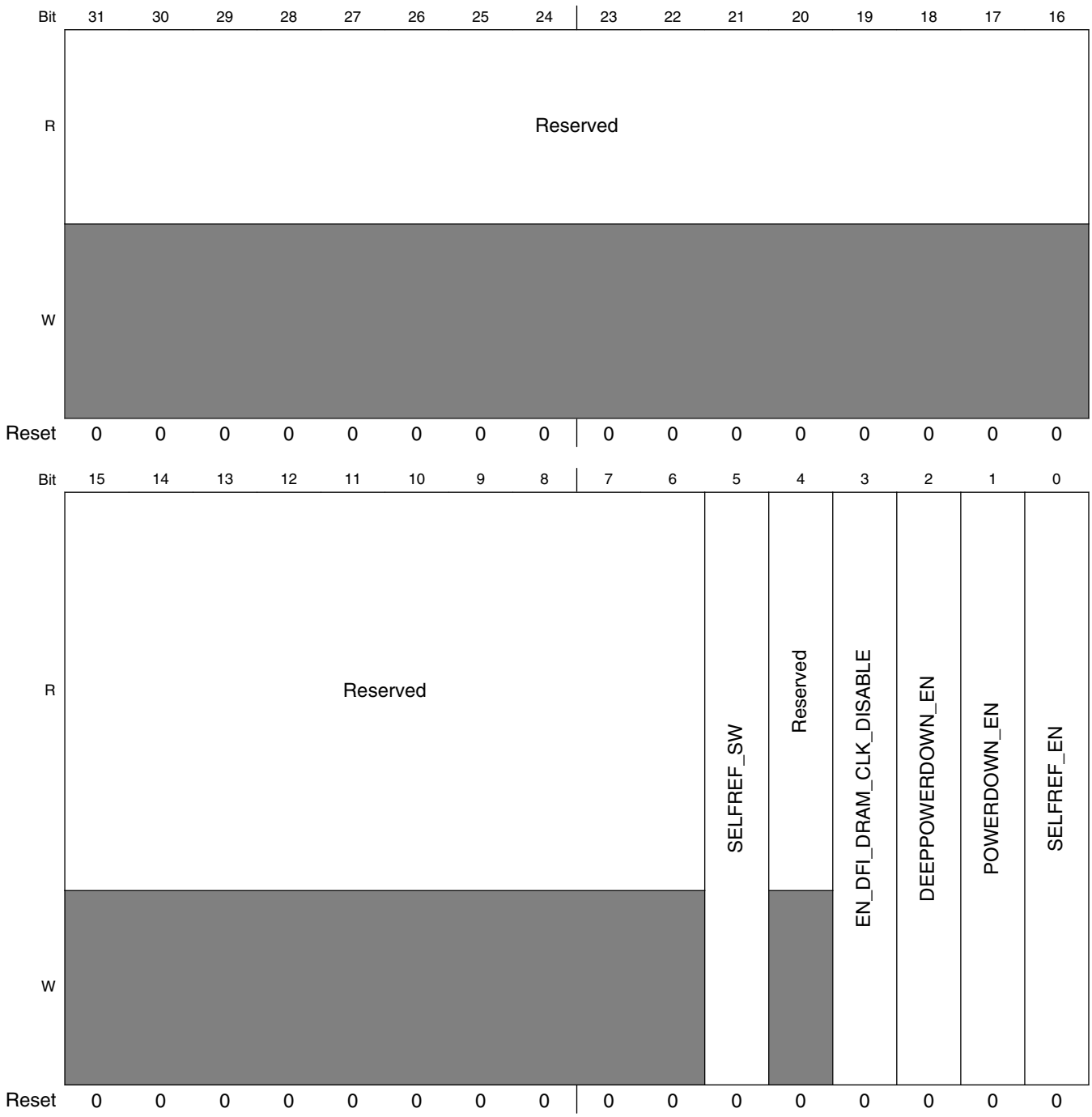


DDRC_DERATEINT field descriptions

Field	Description
MR4_READ_ INTERVAL	Interval between two MR4 reads, used to derate the timing parameters. Present only in designs configured to support LPDDR2 / LPDDR3. This register cannot not be set to zero.**Value After Reset**: 0x800000 **Exists**: MEMC_LPDDR2 == 1

9.2.5.2.8 Low Power Control Register (DDRC_PWRCTL)

Address: 307A_0000h base + 30h offset = 307A_0030h



DDRC_PWRCTL field descriptions

Field	Description
31–6 Reserved	This field is reserved. Reserved for future use

Table continues on the next page...

DDRC_PWRCTL field descriptions (continued)

Field	Description
5 SELFREF_SW	A value 0 of 1 to this register causes system to move to self-refresh state immediately, as long as it is not in INIT or DPD / MPSM operating_mode. This is referred to as Software Entry / Exit to Self-refresh.**Value After Reset**: 0x0 **Exists**: Always 1 Software Entry to Self-refresh 0 Software Exit from Self-refresh
4 Reserved	This field is reserved.
3 EN_DFI_DRAM_CLK_DISABLE	Enable the assertion of dfi_dram_clk_disable whenever a clock is not required by the SDRAM. If set to 0, dfi_dram_clk_disable is never asserted. Assertion of dfi_dram_clk_disable is as follows: In DDR2 / DDR3, can only be asserted in self-refresh.**Value After Reset**: 0x0 **Exists**: Always
2 DEEPPOWERDOWN_EN	When this is 1, DDRC puts the SDRAM into deep power-down mode when the transaction store is empty. This register must be reset to '0' to bring DDRC out of deep power-down mode. Controller performs automatic SDRAM initialization on deep power-down exit. Present only in designs configured to support mDDR or LPDDR2 or LPDDR3. FOR PERFORMANCE ONLY. **Value After Reset**: 0x0 **Exists**: MEMC_LPDDR2 == 1
1 POWERDOWN_EN	If true then the DDRC goes into power-down after a programmable number of cycles "maximum idle clocks before power down" (PWRTMG.powerdown_to_x32). This register bit may be re-programmed during the course of normal operation. **Value After Reset**: 0x0 **Exists**: Always
0 SELFREF_EN	If true then the DDRC puts the SDRAM into Self Refresh after a programmable number of cycles "maximum idle clocks before Self Refresh (PWRTMG.selfref_to_x32)". This register bit may be re-programmed during the course of normal operation. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.9 Low Power Timing Register (DDRC_PWRTMG)

Address: 307A_0000h base + 34h offset = 307A_0034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								SELFREF_TO_X32								T_DPD_X4096								Reserved			POWERDOWN_TO_X32				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_PWRTMG field descriptions

Field	Description
31–24 Reserved	This field is reserved. Reserved for future use

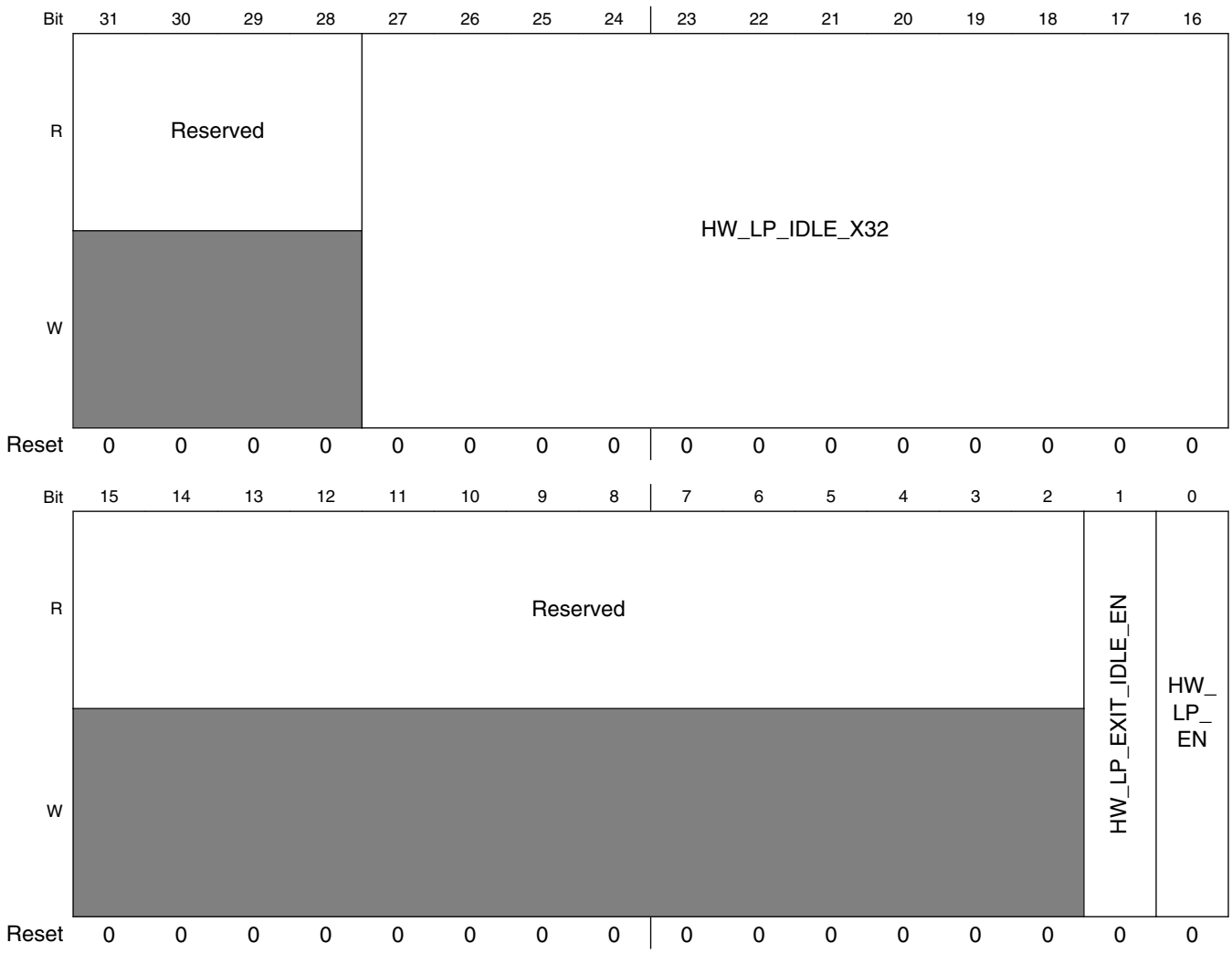
Table continues on the next page...

DDRC_PWRTMG field descriptions (continued)

Field	Description
23–16 SELFREF_TO_X32	After this many clocks of NOP or deselect the DDRC automatically puts the SDRAM into Self Refresh. This must be enabled in the PWRCTL.selfref_en.Unit: Multiples of 32 clocks. FOR PERFORMANCE ONLY. **Value After Reset**: 0x40 **Exists**: Always
15–8 T_DPD_X4096	Minimum deep power-down timeFor mDDR, value from the JEDEC specification is 0 as mDDR exits from deep power-down mode immediately after PWRCTL.deeppowerdown_en is de-asserted. For LPDDR2 / LPDDR3, value from the JEDEC specification is 500 µs. Unit: Multiples of 4096 clocks. Present only in designs configured to support mDDR, LPDDR2 or LPDDR3. FOR PERFORMANCE ONLY. **Value After Reset**: "(MEMC_MOBILE_OR_LPDDR2_EN) ? 0x20 : 0x0" **Exists**: MEMC_LPDDR2 == 1
7–5 Reserved	This field is reserved. Reserved for future use
POWERDOWN_TO_X32	After this many clocks of NOP or deselect the DDRC automatically puts the SDRAM into power-down. This must be enabled in the PWRCTL.powerdown_en. Unit: Multiples of 32 clocks FOR PERFORMANCE ONLY. **Value After Reset**: 0x10 **Exists**: Always

9.2.5.2.10 Hardware Low Power Control Register (DDRC_HWLPCTL)

Address: 307A_0000h base + 38h offset = 307A_0038h



DDRC_HWLPCTL field descriptions

Field	Description
31–28 Reserved	This field is reserved. Reserved for future use
27–16 HW_LP_IDLE_X32	Hardware idle period. The cactive_ddrc output is driven low if the system is idle for hw_lp_idle * 32 cycles if not in INIT or DPD / MPSM operating_mode. The hardware idle function is disabled when hw_lp_idle_x32 = 0. Unit: Multiples of 32 clocks. FOR PERFORMANCE ONLY. **Value After Reset**: 0x0 **Exists**: Always
15–2 Reserved	This field is reserved. Reserved for future use

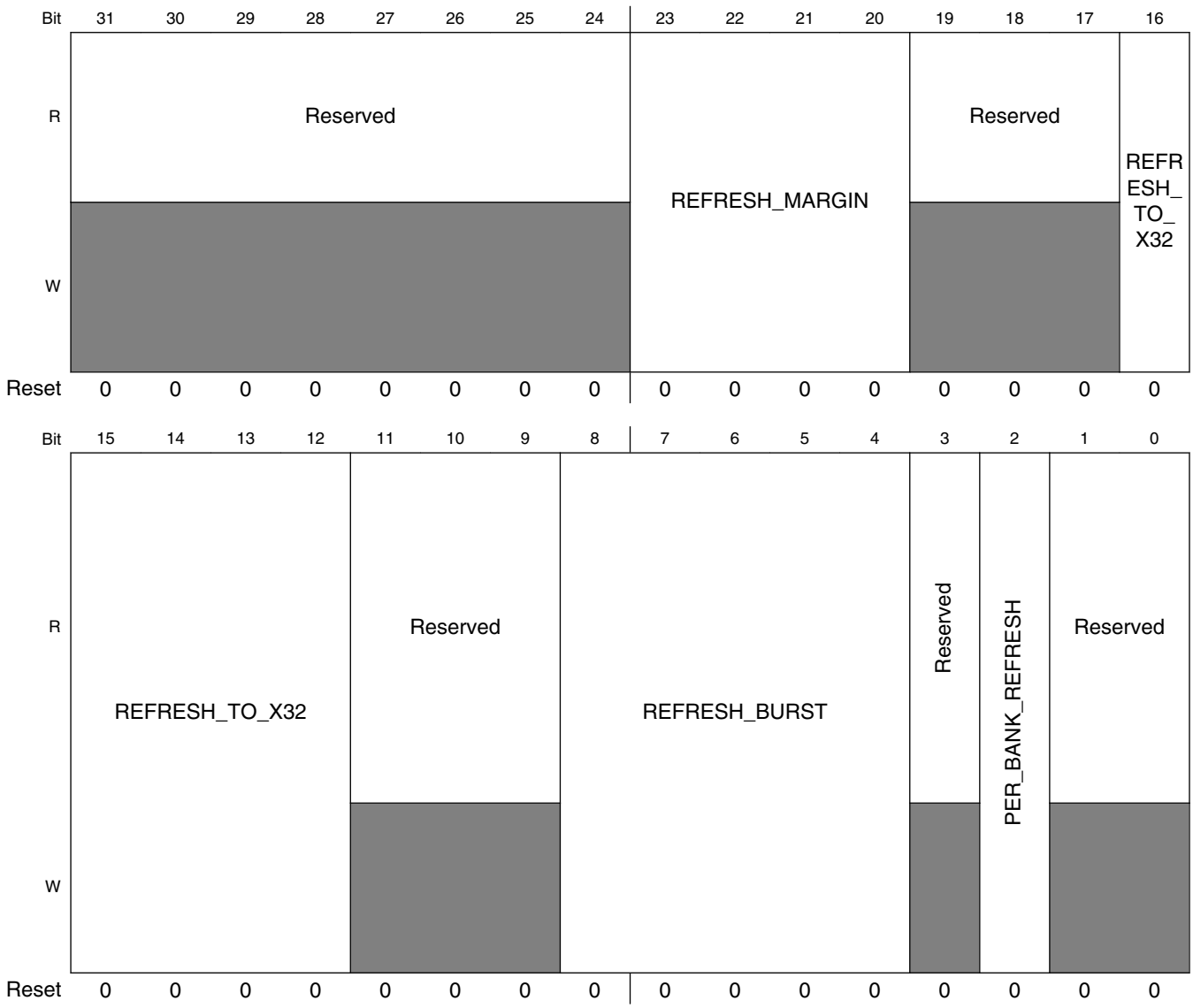
Table continues on the next page...

DDRC_HWLPCTL field descriptions (continued)

Field	Description
1 HW_LP_EXIT_IDLE_EN	When this bit is programmed to 1 the cactive_in_ddrc pin of the DDRC can be used to exit from the automatic clock stop, automatic power down or automatic self-refreshmodes. **NOTE:** It will not cause exit of Self-Refresh that is caused by Hardware Low Power Interface and / or Software (PWRCTL.selfref_sw). **Value After Reset**: 0x1 **Exists**: Always
0 HW_LP_EN	Enable for Hardware Low Power Interface **Value After Reset**: 0x1 **Exists**: Always

9.2.5.2.11 Refresh Control Register 0 (DDRC_RFSHCTL0)

Address: 307A_0000h base + 50h offset = 307A_0050h



DDRC_RFSHCTL0 field descriptions

Field	Description
31–24 Reserved	This field is reserved. Reserved for future use
23–20 REFRESH_MARGIN	Threshold value in number of clock cycles before the critical refresh or page timer expires. A critical refresh is to be issued before this threshold is reached. It is recommended that this not be changed from the default value, currently shown as 0x2. It must always be less than internally used t_rfc_nom_x32. Note that, in LPDDR2 / LPDDR3, internally used t_rfc_nom_x32 may be equal to RFSHTMG.t_rfc_nom_x32 >> 2 if derating is enabled (DERATEEN.derate_enable = 1). Otherwise, internally used t_rfc_nom_x32 will be equal to RFSHTMG.t_rfc_nom_x32. Unit: Multiples of 32 clocks. **Value After Reset**: 0x2

Table continues on the next page...

DDRC_RFSHCTL0 field descriptions (continued)

Field	Description
	Exists: Always
19–17 Reserved	This field is reserved. Reserved for future use
16–12 REFRESH_TO_X32	If the refresh timer (tRFCnom, also known as tREFI) has expired at least once, but it has not expired (RFSHCTL0.refresh_burst + 1) times yet, then a speculative refresh may be performed. A speculative refresh is a refresh performed at a time when refresh would be useful, but before it is absolutely required. When the SDRAM bus is idle for a period of time determined by thisRFSHCTL0.refresh_to_x32 and the refresh timer has expired at least once since the last refresh, then a speculative refresh is performed. Speculative refreshes continues successively until there are no refreshes pending or until new reads or writes are issued to the DDRC. FOR PERFORMANCE ONLY. **Value After Reset**: 0x10 **Exists**: Always
11–9 Reserved	This field is reserved. Reserved for future use
8–4 REFRESH_BURST	The programmed value + 1 is the number of refresh timeouts that is allowed to accumulate before traffic is blocked and the refreshes are forced to execute. Closing pages to perform a refresh is a one-time penalty that must be paid for each group of refreshes. Therefore, performing refreshes in a burst reduces the per-refresh penalty of these page closings. Higher numbers for RFSHCTL.refresh_burst slightly increases utilization; lower numbers decreases the worst-case latency associated with refreshes. For information on burst refresh feature refer to section 3.9 of DDR2 JEDEC specification - JESD79-2F.pdf. For DDR2 / DDR3, the refresh is always per-rank and not per-bank. The rank refresh can be accumulated over 8 * t_{REFI} cycles using the burst refresh feature. **Value After Reset**: 0x0 **Exists**: Always 0 Single refresh 1 Burst-of-2 refresh 7 Burst-of-8 refresh
3 Reserved	This field is reserved. Reserved for future use
2 PER_BANK_REFRESH	Per bank refresh allows traffic to flow to other banks. Per bank refresh is not supported by all LPDDR2 devices but should be supported by all LPDDR3 devices. Present only in designs configured to support LPDDR2 / LPDDR3.**Value After Reset**: 0x0 **Exists**: MEMC_LPDDR2 == 1 1 Per bank refresh 0 All bank refresh
Reserved	This field is reserved. Reserved for future use

9.2.5.2.12 Refresh Control Register 1 (DDRC_RFSHCTL1)

Exists: MEMC_NUM_RANKS > 1

Address: 307A_0000h base + 54h offset = 307A_0054h

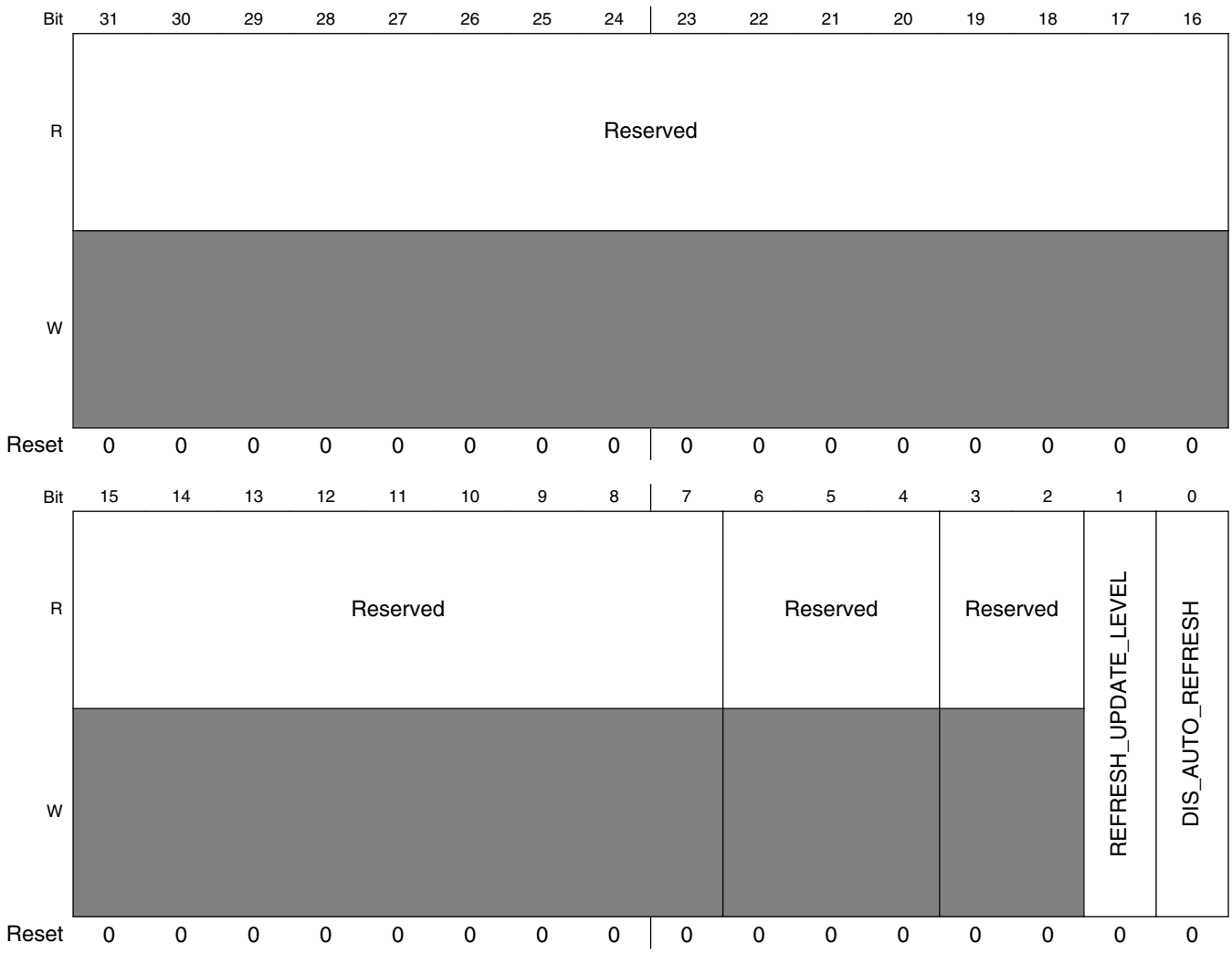
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				REFRESH_TIMER1_START_VALUE_X32												Reserved				REFRESH_TIMER0_START_VALUE_X32											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_RFSHCTL1 field descriptions

Field	Description
31–28 Reserved	This field is reserved. Reserved for future use
27–16 REFRESH_ TIMER1_ START_VALUE_ X32	Refresh timer start for rank 1 (only present in multi-rank configurations). This is useful in staggering the refreshes to multiple ranks to help traffic to proceed. This is explained in Refresh Controls section of architecture chapter. Unit: Multiples of 32 clocks. FOR PERFORMANCE ONLY. **Value After Reset**: 0x0 **Exists**: MEMC_NUM_RANKS > 1
15–12 Reserved	This field is reserved. Reserved for future use
REFRESH_ TIMER0_ START_VALUE_ X32	Refresh timer start for rank 0 (only present in multi-rank configurations). This is useful in staggering the refreshes to multiple ranks to help traffic to proceed. This is explained in Refresh Controls section of architecture chapter. Unit: Multiples of 32 clocks. FOR PERFORMANCE ONLY. **Value After Reset**: 0x0 **Exists**: MEMC_NUM_RANKS > 1

9.2.5.2.13 Refresh Control Register 0 (DDRC_RFSHCTL3)

Address: 307A_0000h base + 60h offset = 307A_0060h



DDRC_RFSHCTL3 field descriptions

Field	Description
31–7 Reserved	This field is reserved. Reserved for future use
6–4 Reserved	This field is reserved.
3–2 Reserved	This field is reserved. Reserved for future use
1 REFRESH_ UPDATE_LEVEL	Toggle this signal (either from 0 to 1 or from 1 to 0) to indicate that the refresh register(s) have been updated. The value is automatically updated when exiting reset, so it does not need to be toggled initially. **Value After Reset**: 0x0 **Exists**: Always

Table continues on the next page...

DDRC_RFSHCTL3 field descriptions (continued)

Field	Description
0 DIS_AUTO_ REFRESH	When '1', disable auto-refresh generated by the DDRC.When auto-refresh is disabled, the SoC core must generate refreshes using the registers reg_ddrc_rank0_refresh, reg_ddrc_rank1_refresh, reg_ddrc_rank2_refresh and reg_ddrc_rank3_refresh. When dis_auto_refresh transitions from 0 to 1, any pending refreshes are immediately scheduled by the DDRC. If DDR4 CRC / parity retry is enabled (CRCPARCTL1.crc_parity_retry_enable = 1), disable auto- refresh is not supported, and this bit must be set to '0'. This register field is changeable on the fly. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.14 Refresh Timing Register (DDRC_RFSHTMG)

Address: 307A_0000h base + 64h offset = 307A_0064h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				T_RFC_NOM_X32												Reserved						T_RFC_MIN									
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_RFSHTMG field descriptions

Field	Description
31–28 Reserved	This field is reserved. Reserved for future use
27–16 T_RFC_NOM_ X32	tREFI: Average time interval between refreshes per rank (Specification: 7.8us for DDR3. See JEDEC specification for mDDR, LPDDR2 and LPDDR3). For LPDDR2 / LPDDR3: • If using all-bank refreshes (RFSHCTL0.per_bank_refresh = 0), this register should be set to tREFIab • If using per-bank refreshes (RFSHCTL0.per_bank_refresh = 1), this register should be set to tREFIpb For configurations with MEMC_FREQ_RATIO = 2, program this to (tREFI / 2), no rounding up. Note that RFSHTMG.t_rfc_nom_x32 * 32 must be greater than RFSHTMG.t_rfc_min. Unit: Multiples of 32 clocks. **Value After Reset**: 0x62 **Exists**: Always
15–10 Reserved	This field is reserved. Reserved for future use
T_RFC_MIN	tRFC (min): Minimum time from refresh to refresh or activate. For MEMC_FREQ_RATIO = 1 configurations, t_rfc_min should be set to RoundUp (tRFCmin / tCK). For MEMC_FREQ_RATIO = 2 configurations, t_rfc_min should be set to RoundUp (RoundUp (tRFCmin / tCK) / 2). In LPDDR2 / LPDDR3 mode: • If using all-bank refreshes, the tRFCmin value in the above equations is equal to tRFCab • If using per-bank refreshes, the tRFCmin value in the above equations is equal to tRFCpb

Table continues on the next page...

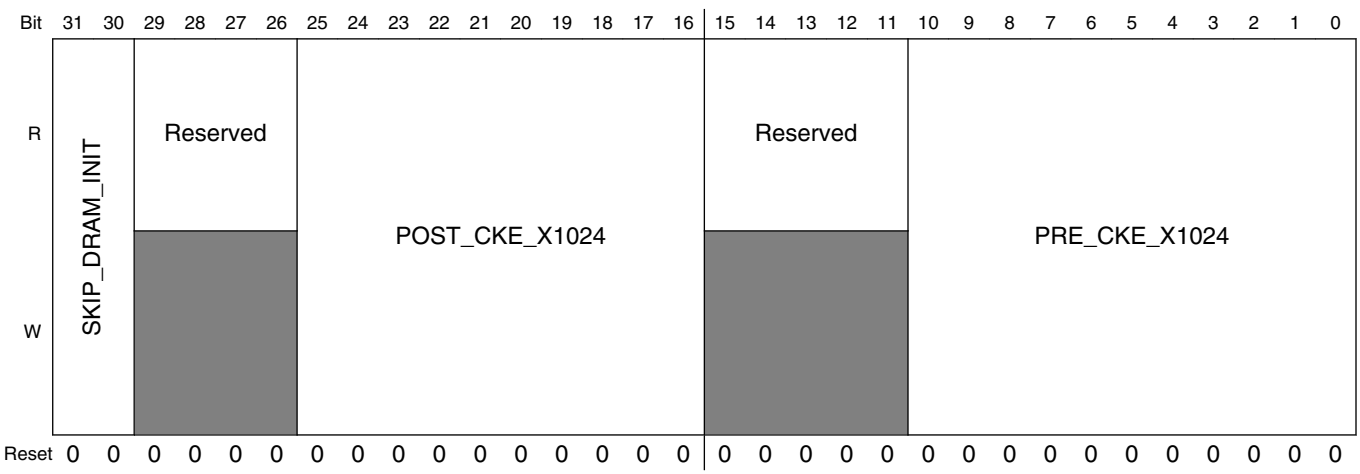
DDRC_RFSHTMG field descriptions (continued)

Field	Description
	Unit: Clocks. **Value After Reset**: 0x8c **Exists**: Always

9.2.5.2.15 SDRAM Initialization Register 0 (DDRC_INIT0)

Exists: Always

Address: 307A_0000h base + D0h offset = 307A_00D0h



DDRC_INIT0 field descriptions

Field	Description
31–30 SKIP_DRAM_INIT	If lower bit is enabled the SDRAM initialization routine is skipped. The upper bit decides what state the controller starts up in when reset is removed. **Value After Reset**: 0x0 **Exists**: Always 00 SDRAM Intialization routine is run after power-up. 01 SDRAM Intialization routine is skipped after power- up. Controller starts up in Normal Mode. 11 SDRAM Intialization routine is skipped after power- up. Controller starts up in Self-refresh Mode. 10 SDRAM Intialization routine is run after power-up.
29–26 Reserved	This field is reserved. Reserved for future use
25–16 POST_CKE_X1024	Cycles to wait after driving CKE high to start the SDRAM initialization sequence. Unit: 1024 clocks. DDR2 typically requires a 400 ns delay, requiring this value to be programmed to 2 at all clock speeds. LPDDR2 / LPDDR3 typically requires this to be programmed for a delay of 200 us.

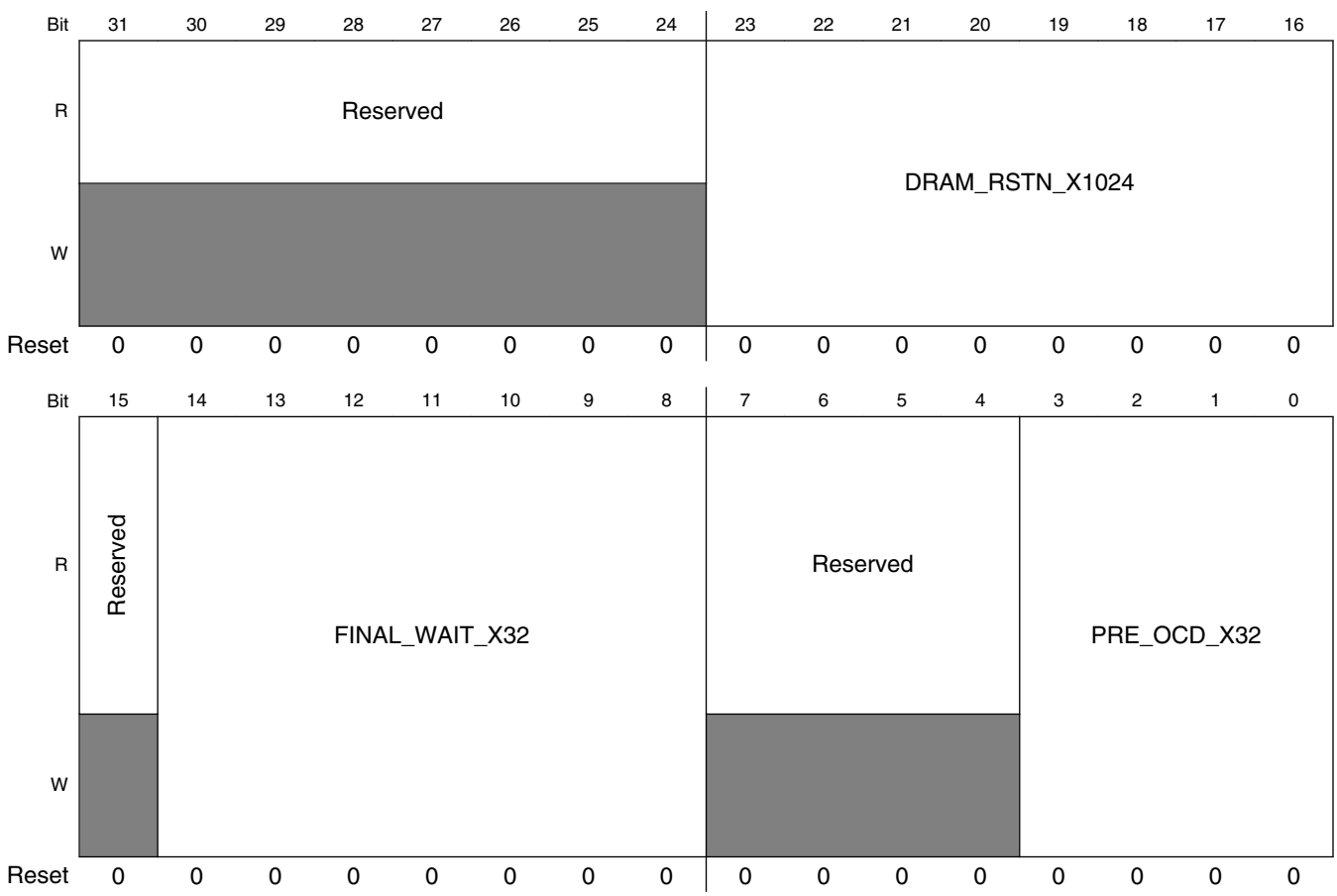
Table continues on the next page...

DDRC_INIT0 field descriptions (continued)

Field	Description
	For configurations with MEMC_FREQ_RATIO = 2, program this to JEDEC spec value divided by 2, and round it up to next integer value. **Value After Reset**: 0x2 **Exists**: Always
15–11 Reserved	This field is reserved. Reserved for future use
PRE_CKE_X1024	Cycles to wait after reset before driving CKE high to start the SDRAM initialization sequence. Unit: 1024 clock cycles. DDR2 specifications typically require this to be programmed for a delay of ≥ 200 µs. LPDDR2 / LPDDR3: tINIT1 of 100 ns (min) For configurations with MEMC_FREQ_RATIO = 2, program this to JEDEC spec value divided by 2, and round it up to next integer value. **Value After Reset**: 0x4e **Exists**: Always

9.2.5.2.16 SDRAM Initialization Register 1 (DDRC_INIT1)

Address: 307A_0000h base + D4h offset = 307A_00D4h



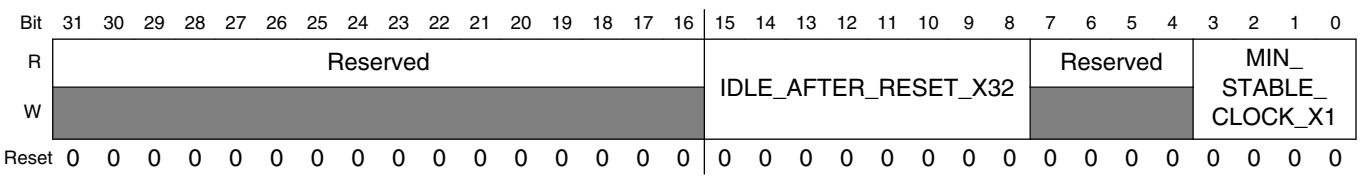
DDRC_INIT1 field descriptions

Field	Description
31–24 Reserved	This field is reserved. Reserved for future use
23–16 DRAM_RSTN_X1024	Number of cycles to assert SDRAM reset signal during init sequence. This is only present for designs supporting DDR3 devices. **Value After Reset**: 0x0 **Exists**: MEMC_DDR3 == 1
15 Reserved	This field is reserved. Reserved for future use
14–8 FINAL_WAIT_X32	Cycles to wait after completing the SDRAM initialization sequence before starting the dynamic scheduler. Unit: Counts of a global timer that pulses every 32 clock cycles. There is no known specific requirement for this; it may be set to zero. **Value After Reset**: 0x0 **Exists**: Always
7–4 Reserved	This field is reserved. Reserved for future use
PRE_OCD_X32	Wait period before driving the OCD complete command to SDRAM. Unit: Counts of a global timer that pulses every 32 clock cycles. There is no known specific requirement for this; it may be set to zero. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.17 SDRAM Initialization Register 2 (DDRC_INIT2)

Exists: MEMC_LPDDR2==1

Address: 307A_0000h base + D8h offset = 307A_00D8h



DDRC_INIT2 field descriptions

Field	Description
31–16 Reserved	This field is reserved. Reserved for future use
15–8 IDLE_AFTER_RESET_X32	Idle time after the reset command, tINIT4. Present only in designs configured to support LPDDR2. Unit: 32 clock cycles.

Table continues on the next page...

DDRC_INIT2 field descriptions (continued)

Field	Description
	Value After Reset: (MEMC_LPDDR2) ? 0xd : 0x0 **Exists**: MEMC_LPDDR2 == 1
7–4 Reserved	This field is reserved. Reserved for future use
MIN_STABLE_ CLOCK_X1	Time to wait after the first CKE high, tINIT2. Present only in designs configured to support LPDDR2 / LPDDR3. Unit: 1 clock cycle. LPDDR2 / LPDDR3 typically requires 5 x tCK delay. **Value After Reset**: (MEMC_LPDDR2) ? 0x5 : 0x0 **Exists**: MEMC_LPDDR2 == 1

9.2.5.2.18 SDRAM Initialization Register 3 (DDRC_INIT3)

Address: 307A_0000h base + DCh offset = 307A_00DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MR																EMR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_INIT3 field descriptions

Field	Description
31–16 MR	DDR2: Value to write to MR register. Bit 8 is for DLL and the setting here is ignored. The DDRC sets this bit appropriately. DDR3: Value loaded into MR0 register. mDDR: Value to write to MR register. LPDDR2 / LPDDR3 - Value to write to MR1 register **Value After Reset**: 0x0 **Exists**: Always
EMR	DDR2: Value to write to EMR register. Bits 9:7 are for OCD and the setting in this register is ignored. The DDRC sets those bits appropriately. • DDR3: Value to write to MR1 register. Set bit 7 to 0. If PHY-evaluation mode training is enabled, this bit is set appropriately by the DDRC during write leveling. • mDDR: Value to write to EMR register • LPDDR2 / LPDDR3: Value to write to MR2 register **Value After Reset**: 0x510 **Exists**: Always

9.2.5.2.19 SDRAM Initialization Register 4 (DDRC_INIT4)

Address: 307A_0000h base + E0h offset = 307A_00E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EMR2																EMR3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_INIT4 field descriptions

Field	Description
31–16 EMR2	DDR2: Value to write to EMR2 register. DDR3: Value to write to MR2 register LPDDR2 / LPDDR3: Value to write to MR3 register mDDR: Unused **Value After Reset**: 0x0 **Exists**: Always
EMR3	DDR2: Value to write to EMR3 register. DDR3: Value to write to MR3 register mDDR / LPDDR2 / LPDDR3: Unused **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.20 SDRAM Initialization Register 5 (DDRC_INIT5)

Exists: MEMC_DDR3==1 || MEMC_LPDDR2==1

Address: 307A_0000h base + E4h offset = 307A_00E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DEV_ZQINIT_X32								Reserved						MAX_AUTO_INIT_X1024									
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_INIT5 field descriptions

Field	Description
31–24 Reserved	This field is reserved. Reserved for future use
23–16 DEV_ZQINIT_X32	ZQ initial calibration, tZQINIT. Present only in designs configured to support DDR3 or LPDDR2 / LPDDR3. Unit: 32 clock cycles. DDR3 typically requires 512 clocks.

Table continues on the next page...

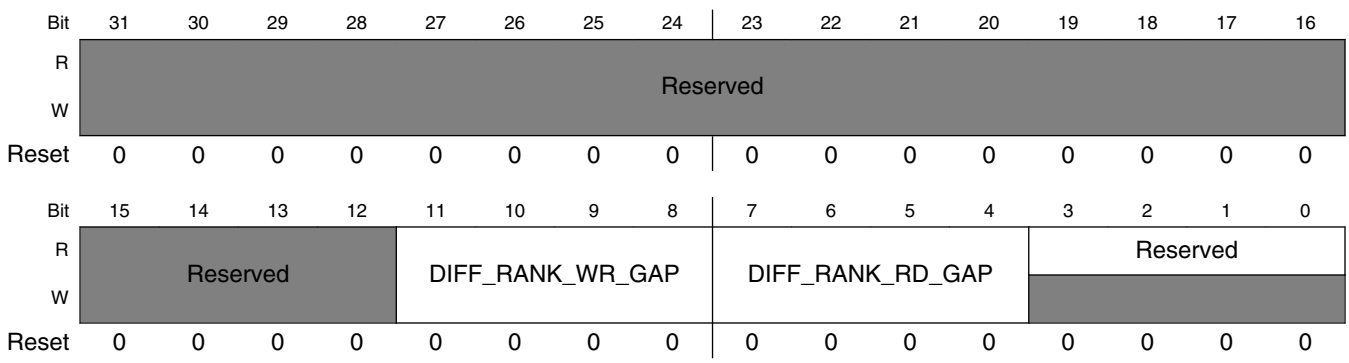
DDRC_INIT5 field descriptions (continued)

Field	Description
	LPDDR2 / LPDDR3 requires 1 µs. **Value After Reset**: 0x10 **Exists**: MEMC_DDR3 == 1 || MEMC_LPDDR2 == 1
15–10 Reserved	This field is reserved. Reserved for future use
MAX_AUTO_ INIT_X1024	Maximum duration of the auto initialization, tINIT5. Present only in designs configured to support LPDDR2 / LPDDR3. LPDDR2 / LPDDR3 typically requires 10 µs. **Value After Reset**: (MEMC_LPDDR2) ? 0x4 : 0x0 **Exists**: MEMC_LPDDR2 == 1

9.2.5.2.21 Rank Control Register (DDRC_RANKCTL)

Exists: MEMC_NUM_RANKs > 1

Address: 307A_0000h base + F4h offset = 307A_00F4h



DDRC_RANKCTL field descriptions

Field	Description
31–12 Reserved	This field is reserved. Reserved for future use
11–8 DIFF_RANK_ WR_GAP	**Description**: Only present for multi-rank configurations. Indicates the number of clocks of gap in data responses when performing consecutive writes to different ranks. This is used to switch the delays in the PHY to match the rank requirements. The value programmed in this register takes care of the ODT switch off timing requirement when switching ranks during writes. For configurations with MEMC_FREQ_RATIO = 2, program this to (N / 2) and round it up to the next integer value. N is value required by PHY, in terms of PHY clocks. If write preamble is set to 2tCK (DDR4 only), it increases by 1. For configurations with MEMC_FREQ_RATIO = 1, program this to N + 1.

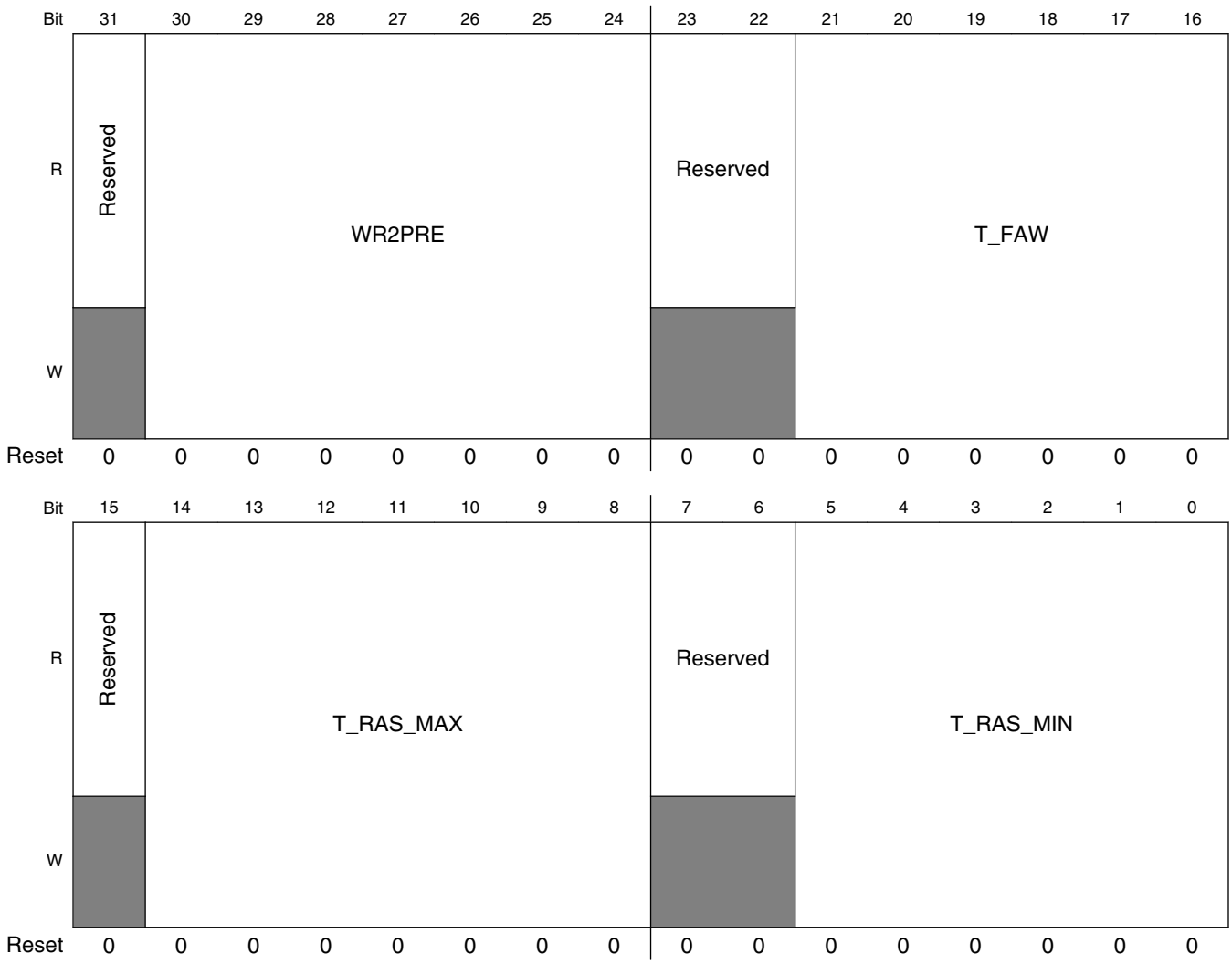
Table continues on the next page...

DDRC_RANKCTL field descriptions (continued)

Field	Description
	For configurations with MEMC_FREQ_RATIO = 2, program this to ((N + 1) / 2) and round it up to the next integer value. **Value After Reset**: 0x6 **Exists**: MEMC_NUM_RANKS > 1
7–4 DIFF_RANK_ RD_GAP	**Description**: Only present for multi-rank configurations. Indicates the number of clocks of gap in data responses when performing consecutive reads to different ranks. This is used to switch the delays in the PHY to match the rank requirements. The value programmed in this register takes care of the ODT switch off timing requirement when switching ranks during reads. For configurations with MEMC_FREQ_RATIO = 2, program this to (N / 2) and round it up to the next integer value. N is value required by PHY, in terms of PHY clocks. If read preamble is set to 2tCK (DDR4 only), it increases by 1. For configuration with MEMC_FREQ_RATIO = 1, program this to N + 1. For configuration with MEMC_FREQ_RATIO = 2, program this to ((N + 10 / 2) and round it up to the next integer value. **Value After Reset**: 0x6 **Exists**: MEMC_NUM_RANKS > 1
Reserved	This field is reserved. Reserved for future use

9.2.5.2.22 SDRAM Timing Register 0 (DDRC_DRAMTMG0)

Address: 307A_0000h base + 100h offset = 307A_0100h



DDRC_DRAMTMG0 field descriptions

Field	Description
31 Reserved	This field is reserved. Reserved for future use
30–24 WR2PRE	Minimum time between write and precharge to same bank Unit: Clocks Specifications: WL + BL/2 + tWR = approximately 8 cycles + 15 ns = 14 clocks @ 400 MHz and less for lower frequencies where: • WL = write latency • BL = burst length. This must match the value programmed in the BL bit of the mode register to the SDRAM. BST (burst terminate) is not supported at present. • tWR = Write recovery time. This comes directly from the SDRAMspecification.

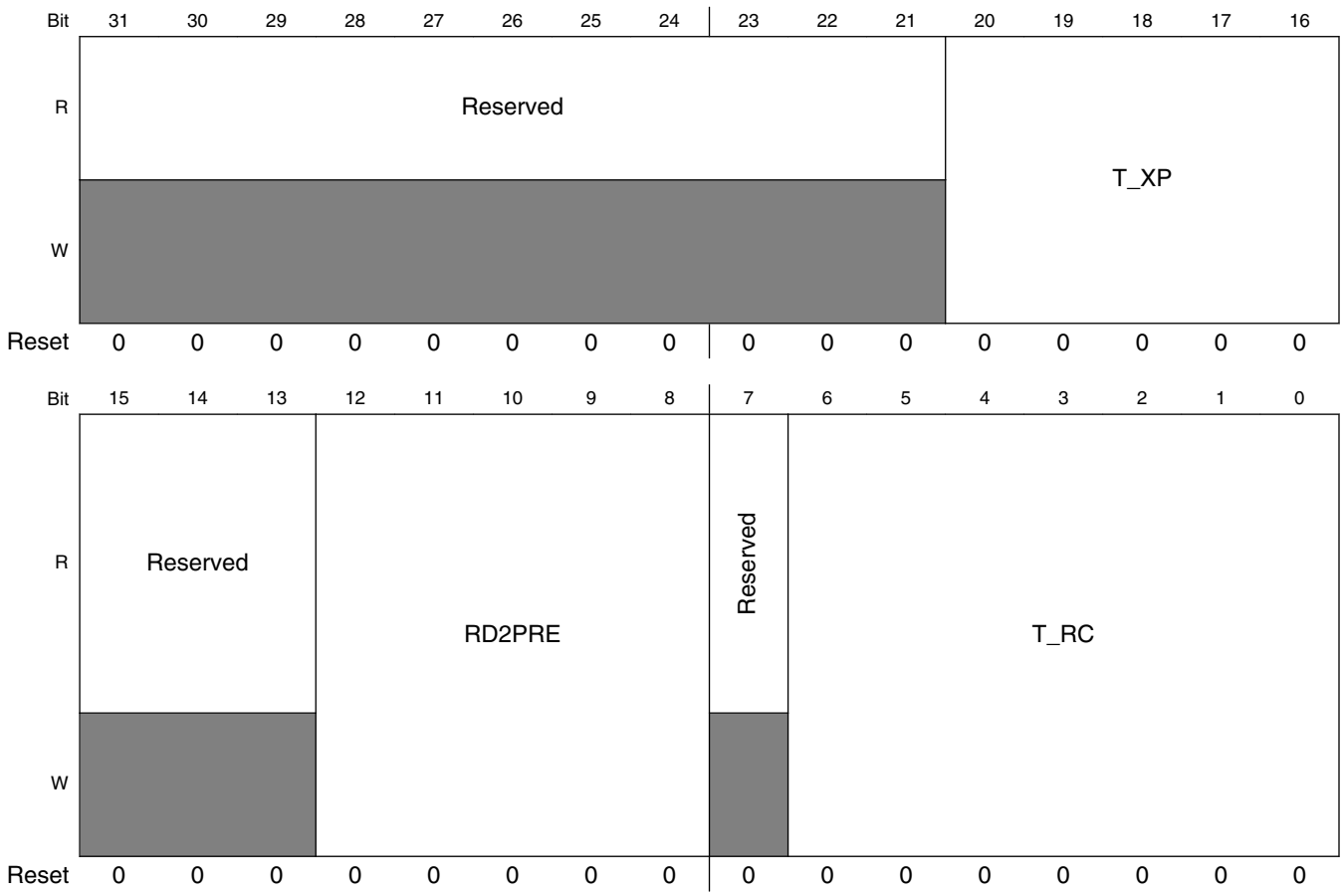
Table continues on the next page...

DDRC_DRAMTMG0 field descriptions (continued)

Field	Description
	Add one extra cycle for LPDDR2/LPDDR3 for this parameter. For configurations with MEMC_FREQ_RATIO = 2, 1T mode, divide the above value by 2. No rounding up. For configurations with MEMC_FREQ_RATIO = 2, 2T mode, divide the above value by 2 and add 1. No rounding up. **Value After Reset**: 0xf **Exists:**Always
23–22 Reserved	This field is reserved. Reserved for future use
21–16 T_FAW	tFAW Valid only when 8 or more banks (or banks x bank groups) are present. In 8-bank design, at most 4 banks must be activated in a rolling window of tFAW cycles. For configurations with MEMC_FREQ_RATIO = 2, program this to (tFAW / 2) and round up to next integer value. In a 4-bank design, set this register to 0x1 independent of the MEMC_FREQ_RATIO configuration. Unit: Clocks **Value After Reset**: 0x10 **Exists**: Always
15 Reserved	This field is reserved. Reserved for future use
14–8 T_RAS_MAX	tRAS(max): Maximum time between activate and precharge to same bank. This is the maximum time that a page can be kept open. Minimum value of this register is 1. Zero is invalid. For configurations with MEMC_FREQ_RATIO = 2, program this to (tRAS (max) - 1) / 2. No rounding up. Unit: Multiples of 1024 clocks. **Value After Reset**: 0x1b **Exists**: Always
7–6 Reserved	This field is reserved. Reserved for future use
T_RAS_MIN	tRAS (min): Minimum time between activate and precharge to the same bank. For configurations with MEMC_FREQ_RATIO = 2, 1T mode, program this to tRAS(min) / 2. No rounding up. For configurations with MEMC_FREQ_RATIO = 2, 2T mode, program this to (tRAS (min) / 2 + 1). No rounding up of the division operation. Unit: Clocks **Value After Reset**: 0xf **Exists**: Always

9.2.5.2.23 SDRAM Timing Register 1 (DDRC_DRAMTMG1)

Address: 307A_0000h base + 104h offset = 307A_0104h



DDRC_DRAMTMG1 field descriptions

Field	Description
31–21 Reserved	This field is reserved. Reserved for future use
20–16 T_XP	tXP: Minimum time after power-down exit to any operation. For DDR3, this should be programmed to tXPDLL if slow powerdown exit is selected in MR0[12]. Units: Clocks **Value After Reset**: 0x8 **Exists**: Always
15–13 Reserved	This field is reserved. Reserved for future use
12–8 RD2PRE	tRTP: Minimum time from read to precharge of same bank. • DDR3: tAL + max (tRTP, 4) • DDR4: Max of following two equations: tAL + max (tRTP, 4) or, RL + BL / 2 - tRP. • LPDDR2: Depends on if it's LPDDR2-S2 or LPDDR2-S4: • LPDDR3: BL / 2 + max (tRTP, 4) - 4

Table continues on the next page...

DDRC_DRAMTMG1 field descriptions (continued)

Field	Description
	For configurations with MEMC_FREQ_RATIO = 2, divide the above value by 2. No rounding up. Unit: Clocks. **Value After Reset**: 0x4 **Exists**: Always
7 Reserved	This field is reserved. Reserved for future use
T_RC	tRC: Minimumtime between activates to same bank. For configurations with MEMC_FREQ_RATIO = 2, program this to (tRC / 2) and round up to next integer value. Unit: Clocks. **Value After Reset**: 0x14 **Exists**: Always

9.2.5.2.24 SDRAM Timing Register 2 (DDRC_DRAMTMG2)

Address: 307A_0000h base + 108h offset = 307A_0108h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		WRITE_LATENCY						Reserved		READ_LATENCY					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		RD2WR						Reserved		WR2RD					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_DRAMTMG2 field descriptions

Field	Description
31–30 Reserved	This field is reserved. Reserved for future use
29–24 WRITE_ LATENCY	Set to WL Time from write command to write data on SDRAMinterface. This must be set to WL. Note that, depending on the PHY, if using RDIMM, it may be necessary to use a value of WL + 1 to compensate for the extra cycle of latency through the RDIMM For configurations with MEMC_FREQ_RATIO = 2, divide the value calculated using the above equation by 2, and round it up to next integer. This register field is not required for DDR3 as the DFI read and write latencies defined in DFITMG0 and DFITMG1 are sufficient for those protocols. Unit: clocks **Value After Reset**: 0x3

Table continues on the next page...

DDRC_DRAMTMG2 field descriptions (continued)

Field	Description
	Exists: MEMC_MOBILE==1 || MEMC_LPDDR2==1 || MEMC_DDR4==1 || MEMC_TRAINING==1
23–22 Reserved	This field is reserved. Reserved for future use
21–16 READ_ LATENCY	Set to RL Time from read command to read data on SDRAM interface.This must be set to RL. **NOTE:** Depending on the PHY, if using RDIMM, it may be necessary to use a value of RL + 1 to compensate for the extra cycle of latency through the RDIMM. For configurations with MEMC_FREQ_RATIO = 2, divide the value calculated using the above equation by 2, and round it up to next integer. This register field is not required for DDR2 and DDR3 as the DFI read and write latencies defined in DFITMG0 and DFITMG1 are sufficient for those protocols Unit: clocks **Value After Reset**: 0x5 **Exists**: MEMC_MOBILE==1 || MEMC_LPDDR2==1 || MEMC_DDR4==1 || MEMC_TRAINING==1
15–14 Reserved	This field is reserved. Reserved for future use
13–8 RD2WR	DDR2/3/mDDR: RL + BL / 2 - WL DDR4: RL + BL / 2 + 1 +WR_PREAMBLE - WL LPDDR2/LPDDR3: RL + BL / 2 + RU (tDQSCKmax / tCK) + 1 - WL. Minimum time from read command to write command. Include time for bus turnaround and all per-bank, per-rank, and global constraints. Unit: Clocks. Where • WL = write latency • BL = burst length. This must match the value programmed in the BL bit of the mode register to the SDRAM. • RL = read latency = CAS latency • WR_PREAMBLE = write preamble. This is unique to DDR4. For configurations with MEMC_FREQ_RATIO = 2, divide the value calculated using the above equation by 2, and round it up to next integer. Value After Reset: 0x6 Exists: Always
7–6 Reserved	This field is reserved. Reserved for future use
WR2RD	DDR4: CWL + PL + BL / 2 + tWTR_L Others: CWL + BL / 2 + tWTR In DDR4, minimum time from write command to read command for same bank group. In others, minimum time from write command to read command. Includes time for bus turnaround, recovery times, and all per-bank, per-rank, and global constraints. Unit: Clocks. Where: • CWL = CAS write latency

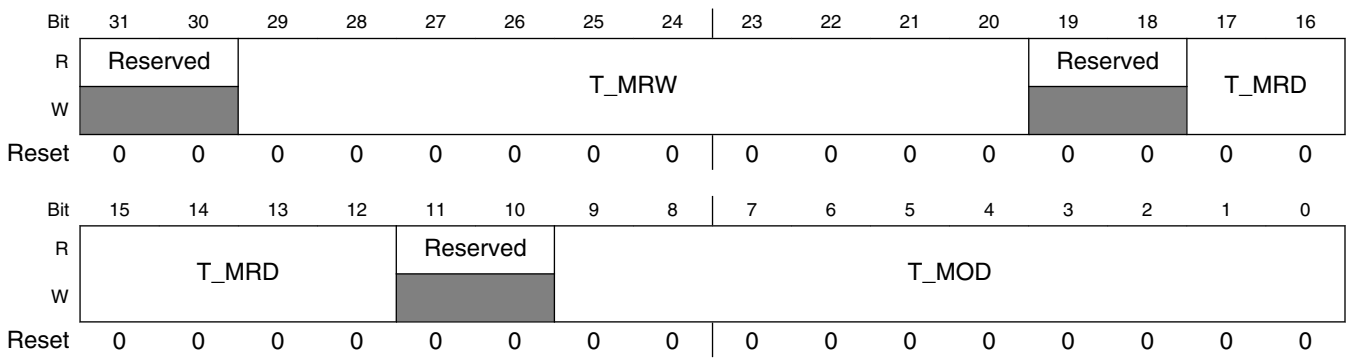
Table continues on the next page...

DDRC_DRAMTMG2 field descriptions (continued)

Field	Description
	• PL = Parity latency • BL = Burst length. This must match the value programmed in the BL bit of the mode register to the SDRAM. • tWTR_L = Internal write to read command delay for same bank group. This comes directly from the SDRAM specification. • tWTR = Internal write to read command delay. This comes directly from the SDRAM specification. Add one extra cycle for LPDDR2 / LPDDR3 operation. For configurations with MEMC_FREQ_RATIO = 2, divide the value calculated using the above equation by 2, and round it up to next integer. **Value After Reset**: 0xd **Exists**: Always

9.2.5.2.25 SDRAM Timing Register 3 (DDRC_DRAMTMG3)

Address: 307A_0000h base + 10Ch offset = 307A_010Ch



DDRC_DRAMTMG3 field descriptions

Field	Description
31–30 Reserved	This field is reserved. Reserved for future use
29–20 T_MRW	Time to wait after a mode register write or read (MRW or MRR). Present only in designs configured to support LPDDR2 / LPDDR3. LPDDR2 typically requires value of 5. LPDDR3 typically requires value of 10. **Value After Reset**: (MEMC_LPDDR2 == 1) ? 0x5 : 0x0 **Exists**: MEMC_LPDDR2 == 1
19–18 Reserved	This field is reserved. Reserved for future use
17–12 T_MRD	tMRD: Cycles between loadmode commands. If MEMC_DDR3 = 0, this parameter is also used to define the cycles between load mode command and following non-load mode command.

Table continues on the next page...

DDRC_DRAMTMG3 field descriptions (continued)

Field	Description
	For configurations with MEMC_FREQ_RATIO = 2, program this to (tMRD / 2) and round it up to the next integer value. If C/A parity for DDR4 is used, set to tMRD_PAR (tMOD + PL) instead. **Value After Reset**: 0x4 **Exists**: Always
11–10 Reserved	This field is reserved. Reserved for future use
T_MOD	tMOD: Present if MEMC_DDR3_OR_4 = 1. Cycles between loadmode command and following non-load mode command. This is required to be programmed even when a design that supports DDR3 / DDR4 is running in DDR2 mode. Set to tMOD if MEMC_FREQ_RATIO = 1, or tMOD / 2 (rounded up to next integer) if MEMC_FREQ_RATIO = 2. Note that if using RDIMM, depending on the PHY, it may be necessary to use a value of tMOD + 1 or (tMOD + 1) / 2 to compensate for the extra cycle of latency applied to mode register writes by the RDIMM chip. **Value After Reset**: (MEMC_DDR3 == 1) ? 0xc : 0x0 **Exists**: MEMC_DDR3 == 1

9.2.5.2.26 SDRAM Timing Register 4 (DDRC_DRAMTMG4)

Exists: Always

Address: 307A_0000h base + 110h offset = 307A_0110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			T_RCD					Reserved				T_CCD				Reserved				T_RRD				Reserved			T_RP				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_DRAMTMG4 field descriptions

Field	Description
31–29 Reserved	This field is reserved. Reserved for future use
28–24 T_RCD	tRCD - tAL: Minimum time from activate to read or write command to same bank. For configurations with MEMC_FREQ_RATIO = 2, program this to ((tRCD - tAL) / 2) and round it up to the next integer value. Minimum value allowed for this register is 1, which implies minimum (tRCD - tAL) value to be 2 in configurations with MEMC_FREQ_RATIO = 2. Unit: Clocks. **Value After Reset**: 0x5 **Exists**: Always
23–20 Reserved	This field is reserved. Reserved for future use

Table continues on the next page...

DDRC_DRAMTMG4 field descriptions (continued)

Field	Description
19–16 T_CCD	DDR4: tCCD_L: This is the minimum time between two reads or two writes for same bank group. Other: tCCD: This is the minimum time between two reads or two writes. For configurations with MEMC_FREQ_RATIO = 2, program this to (tCCD_L / 2 or tCCD / 2) and round it up to the next integer value. Unit: clocks. **Value After Reset**: 0x4 **Exists**: Always
15–12 Reserved	This field is reserved. Reserved for future use
11–8 T_RRD	DDR4: tRRD_L: Minimum time between activates from bank a to bank b for same bank group. Others: tRRD: Minimum time between activates from bank a to bank b. For configurations with MEMC_FREQ_RATIO = 2, program this to (tRRD_L / 2 or tRRD / 2) and round it up to the next integer value. Unit: Clocks. **Value After Reset**: 0x4 **Exists**: Always
7–5 Reserved	This field is reserved. Reserved for future use
T_RP	tRP: Minimum time from precharge to activate of same bank • For MEMC_FREQ_RATIO = 1 configurations, t_rp should be set to RoundUp (tRP / tCK). • For MEMC_FREQ_RATIO = 2 configurations, t_rp should be set to RoundDown (RoundUp (tRP / tCK) / 2) + 1. Unit: Clocks **Value After Reset:**0x5 **Exists:**Always

9.2.5.2.27 SDRAM Timing Register5 (DDRC_DRAMTMG5)

Address: 307A_0000h base + 114h offset = 307A_0114h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				T_CKSRX				Reserved				T_CKSRE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		T_CKESR						Reserved		T_CKE					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_DRAMTMG5 field descriptions

Field	Description
31–28 Reserved	This field is reserved. Reserved for future use
27–24 T_CKSRX	This is the time before Self Refresh Exit that CK is maintained as a valid clock before issuing SRX. Specifies the clock stable time before SRX. Recommended settings: • LPDDR2: 2 • LPDDR3: 2 • DDR3: tCKSRX For configurations with MEMC_FREQ_RATIO = 2, program this to recommended value divided by two and round it up to next integer. **Value After Reset**: 0x5 **Exists**: Always
23–20 Reserved	This field is reserved. Reserved for future use
19–16 T_CKSRE	This is the time after Self Refresh Down Entry that CK is maintained as a valid clock. Specifies the clock disable delay after SRE. Recommended settings: • LPDDR2: 2 • LPDDR3: 2 • DDR3: max (10 ns, 5 tCK) For configurations with MEMC_FREQ_RATIO = 2, program this to recommended value divided by two and round it up to next integer. **Value After Reset**: 0x5 **Exists**: Always
15–14 Reserved	This field is reserved. Reserved for future use
13–8 T_CKESR	Minimum CKE low width for Self refresh entry to exit timing in memory clock cycles. Recommended settings: • LPDDR2: tCKESR • LPDDR3: tCKESR • DDR3: tCKE + 1 For configurations with MEMC_FREQ_RATIO = 2, program this to recommended value divided by two and round it up to next integer. **Value After Reset**: 0x4 **Exists**: Always
7–5 Reserved	This field is reserved. Reserved for future use
T_CKE	Minimum number of cycles of CKE HIGH / LOW during power-down and self refresh. • LPDDR2 / LPDDR3 mode: Set this to the larger of tCKE or tCKESR • Non-LPDDR2 / non-LPDDR3 designs: Set this to tCKE value. For configurations with MEMC_FREQ_RATIO = 2, program this to (value described above) / 2 and round it up to the next integer value. Unit: Clocks. **Value After Reset**: 0x3

Table continues on the next page...

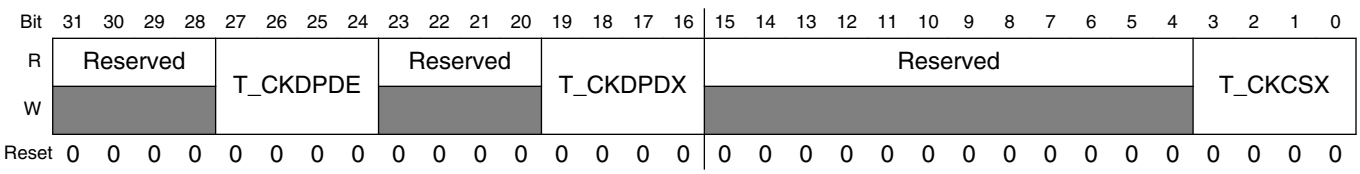
DDRC_DRAMTMG5 field descriptions (continued)

Field	Description
	Exists: Always

9.2.5.2.28 SDRAM Timing Register 6 (DDRC_DRAMTMG6)

Exists: MEMC_MOBILE==1 || MEMC_LPDDR2==1

Address: 307A_0000h base + 118h offset = 307A_0118h



DDRC_DRAMTMG6 field descriptions

Field	Description
31–28 Reserved	This field is reserved. Reserved for future use
27–24 T_CKDPDE	This is the time after Deep Power Down Entry that CK is maintained as a valid clock. Specifies the clock disable delay after DPDE. Recommended settings: • LPDDR2: 2 • LPDDR3: 2 For configurations with MEMC_FREQ_RATIO = 2, program this to recommended value divided by two and round it up to next integer. This is only present for designs supporting mDDR or LPDDR2 / LPDDR3 devices. **Value After Reset**: 0x2 **Exists**: MEMC_LPDDR2==1
23–20 Reserved	This field is reserved. Reserved for future use
19–16 T_CKDPDX	This is the time before Deep Power Down Exit that CK is maintained as a valid clock before issuing DPDX. Specifies the clock stable time before DPDX. Recommended settings: • LPDDR2: 2 • LPDDR3: 2 For configurations with MEMC_FREQ_RATIO = 2, program this to recommended value divided by two and round it up to next integer. This is only present for designs supporting mDDR or LPDDR2 devices. **Value After Reset**: 0x2 **Exists**: MEMC_LPDDR2==1
15–4 Reserved	This field is reserved. Reserved for future use

Table continues on the next page...

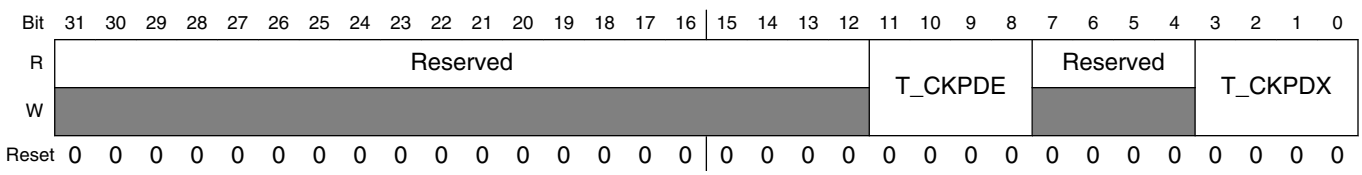
DDRC_DRAMTMG6 field descriptions (continued)

Field	Description
T_CKCSX	This is the time before Clock Stop Exit that CK is maintained as a valid clock before issuing Clock Stop Exit. Specifies the clock stable time before next command after Clock Stop Exit. Recommended settings: • LPDDR2: tXP + 2 • LPDDR3: tXP + 2 For configurations with MEMC_FREQ_RATIO = 2, program this to recommended value divided by two and round it up to next integer. This is only present for designs supporting mDDR or LPDDR2 / LPDDR3 devices. **Value After Reset**: 0x5 **Exists**:MEMC_LPDDR2 == 1

9.2.5.2.29 SDRAM Timing Register 7 (DDRC_DRAMTMG7)

Exists: MEMC_MOBILE==1 || MEMC_LPDDR2==1

Address: 307A_0000h base + 11Ch offset = 307A_011Ch



DDRC_DRAMTMG7 field descriptions

Field	Description
31–12 Reserved	This field is reserved. Reserved for future use
11–8 T_CKPDE	This is the time after Power Down Entry that CK is maintained as a valid clock. Specifies the clock disable delay after PDE. Recommended settings: • LPDDR2: 2 • LPDDR3: 2 For configurations with MEMC_FREQ_RATIO = 2, program this to recommended value divided by two and round it up to next integer. This is only present for designs supporting mDDR or LPDDR2 / LPDDR3 devices. **Value After Reset**: 0x2 **Exists**: MEMC_LPDDR2 == 1
7–4 Reserved	This field is reserved. Reserved for future use
T_CKPDX	This is the time before Power Down Exit that CK is maintained as a valid clock before issuing PDX. Specifies the clock stable time before PDX. Recommended settings:

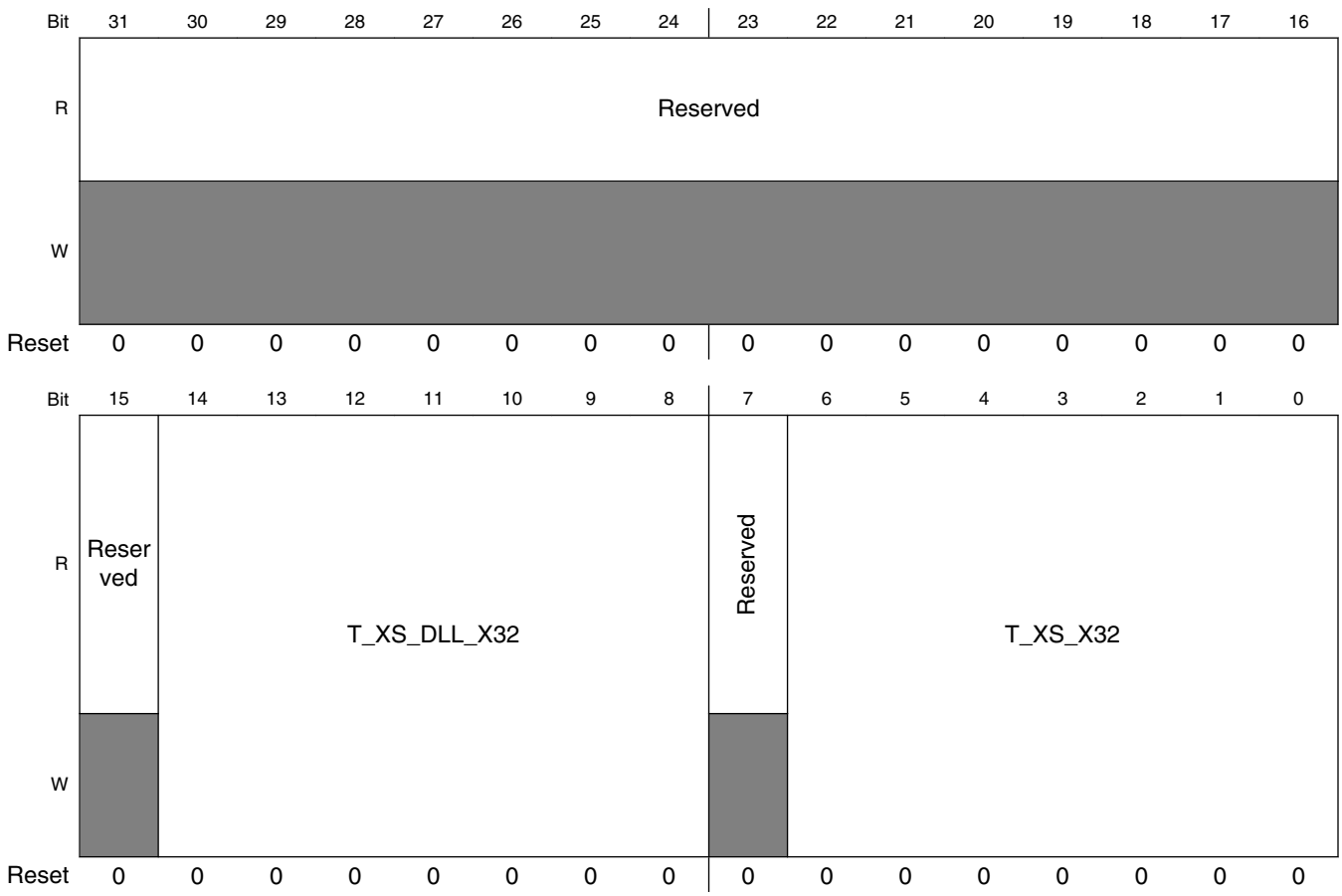
Table continues on the next page...

DDRC_DRAMTMG7 field descriptions (continued)

Field	Description
	• LPDDR2: 2 • LPDDR3: 2 For configurations with MEMC_FREQ_RATIO = 2, program this to recommended value divided by two and round it up to next integer. This is only present for designs supporting mDDR or LPDDR2 / LPDDR3 devices. **Value After Reset**: 0x2 **Exists**: MEMC_LPDDR2 == 1

9.2.5.2.30 SDRAM Timing Register 8 (DDRC_DRAMTMG8)

Address: 307A_0000h base + 120h offset = 307A_0120h



DDRC_DRAMTMG8 field descriptions

Field	Description
31–15 Reserved	This field is reserved. Reserved for future use

Table continues on the next page...

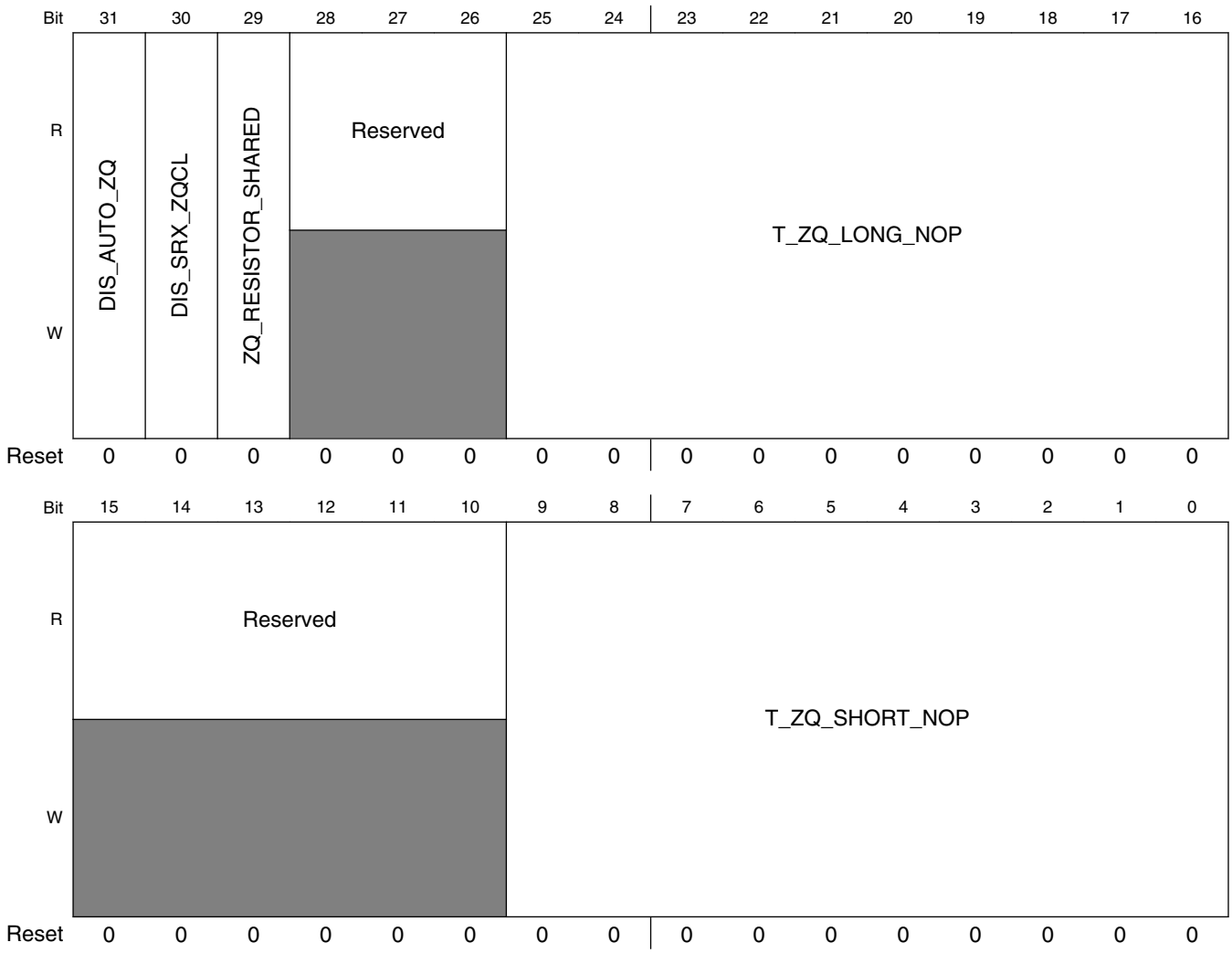
DDRC_DRAMTMG8 field descriptions (continued)

Field	Description
14–8 T_XS_DLL_X32	tXSDLL: Exit Self Refresh to commands requiring a locked DLL. For configurations with MEMC_FREQ_RATIO = 2, program this to the above value divided by 2 and round up to next integer value. Unit: Multiples of 32 clocks. **NOTE:** In LPDDR2 / LPDDR3 / Mobile DDR mode, t_xs_x32 and t_xs_dll_x32 must be set the same values derived from tXSR. **Value After Reset**: 0x44 **Exists**: Always
7 Reserved	This field is reserved. Reserved for future use
T_XS_X32	tXS: Exit Self Refresh to commands not requiring a locked DLL. For configurations with MEMC_FREQ_RATIO = 2, program this to the above value divided by 2 and round up to next integer value. Unit: Multiples of 32 clocks. **NOTE:** In LPDDR2 / LPDDR3 / Mobile DDR mode, t_xs_x32 and t_xs_dll_x32 must be set the same values derived from tXSR. **Value After Reset**: 0x5 **Exists**: Always

9.2.5.2.31 ZQ Control Register 0 (DDRC_ZQCTL0)

Exists: MEMC_DDR3==1 || MEMC_LPDDR2==1

Address: 307A_0000h base + 180h offset = 307A_0180h



DDRC_ZQCTL0 field descriptions

Field	Description
31 DIS_AUTO_ZQ	This is only present for designs supporting DDR3 or LPDDR2 / LPDDR3 devices. **Value After Reset**: 0x0 **Exists**: MEMC_DDR3 == 1 || MEMC_LPDDR2 == 1 1 Disable DDRC generation of ZQCS command. Register DBGCMD.zq_calib_short can be used instead to issue ZQ calibration request from APB module. 0 Internally generate ZQCS commands based on ZQCTL1.t_zq_short_interval_x1024.

Table continues on the next page...

DDRC_ZQCTL0 field descriptions (continued)

Field	Description
30 DIS_SRX_ZQCL	This is only present for designs supporting DDR3 or LPDDR2 / LPDDR3 devices. **Value After Reset**: 0x0 **Exists**: MEMC_DDR3 == 1 || MEMC_LPDDR2 == 1 1 Disable issuing of ZQCL command at Self-Refresh exit. Only applicable when run in DDR3 or LPDDR2 or LPDDR3 mode. 0 Enable issuing of ZQCL command at Self-Refresh exit. Only applicable when run in DDR3 or LPDDR2 or LPDDR3 mode.
29 ZQ_RESISTOR_SHARED	This is only present for designs supporting DDR3 or LPDDR2 / LPDDR3 devices. **Value After Reset**: 0x0 **Exists**: MEMC_DDR3 == 1 || MEMC_LPDDR2 == 1 1 Denotes that ZQ resistor is shared between ranks. Means ZQinit / ZQCL / ZQCS commands are sent to one rank at a time with tZQinit / tZQCL / tZQCS timing met between commands so that commands to different ranks do not overlap. 0 ZQ resistor is not shared.
28–26 Reserved	This field is reserved. Reserved for future use
25–16 T_ZQ_LONG_NOP	tZQoper for DDR3, tZQCL for LPDDR2 / LPDDR3: Number of cycles of NOP required after a ZQCL (ZQ calibration long) command is issued to SDRAM. For configurations with MEMC_FREQ_RATIO = 2: DDR3: program this to tZQoper / 2 and round it up to the next integer value. LPDDR2 / LPDDR3: program this to tZQCL / 2 and round it up to the next integer value. Unit: Clock cycles. This is only present for designs supporting DDR3 or LPDDR2 / LPDDR3 devices. **Value After Reset**: 0x200 **Exists**: MEMC_DDR3 == 1 || MEMC_LPDDR2 == 1
15–10 Reserved	This field is reserved. Reserved for future use
T_ZQ_SHORT_NOP	tZQCS: Number of cycles of NOP required after a ZQCS (ZQ calibration short) command is issued to SDRAM. For configurations with MEMC_FREQ_RATIO = 2, program this to tZQCS / 2 and round it up to the next integer value. Unit: Clock cycles. This is only present for designs supporting DDR3 or LPDDR2 / LPDDR3 devices. **Value After Reset**: 0x40 **Exists**: MEMC_DDR3 == 1 || MEMC_LPDDR2 == 1

9.2.5.2.32 ZQ Control Register 1 (DDRC_ZQCTL1)

Exists: MEMC_DDR3==1 || MEMC_LPDDR2==1

Address: 307A_0000h base + 184h offset = 307A_0184h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		T_ZQ_RESET_NOP										T_ZQ_SHORT_INTERVAL_X1024			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	T_ZQ_SHORT_INTERVAL_X1024															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

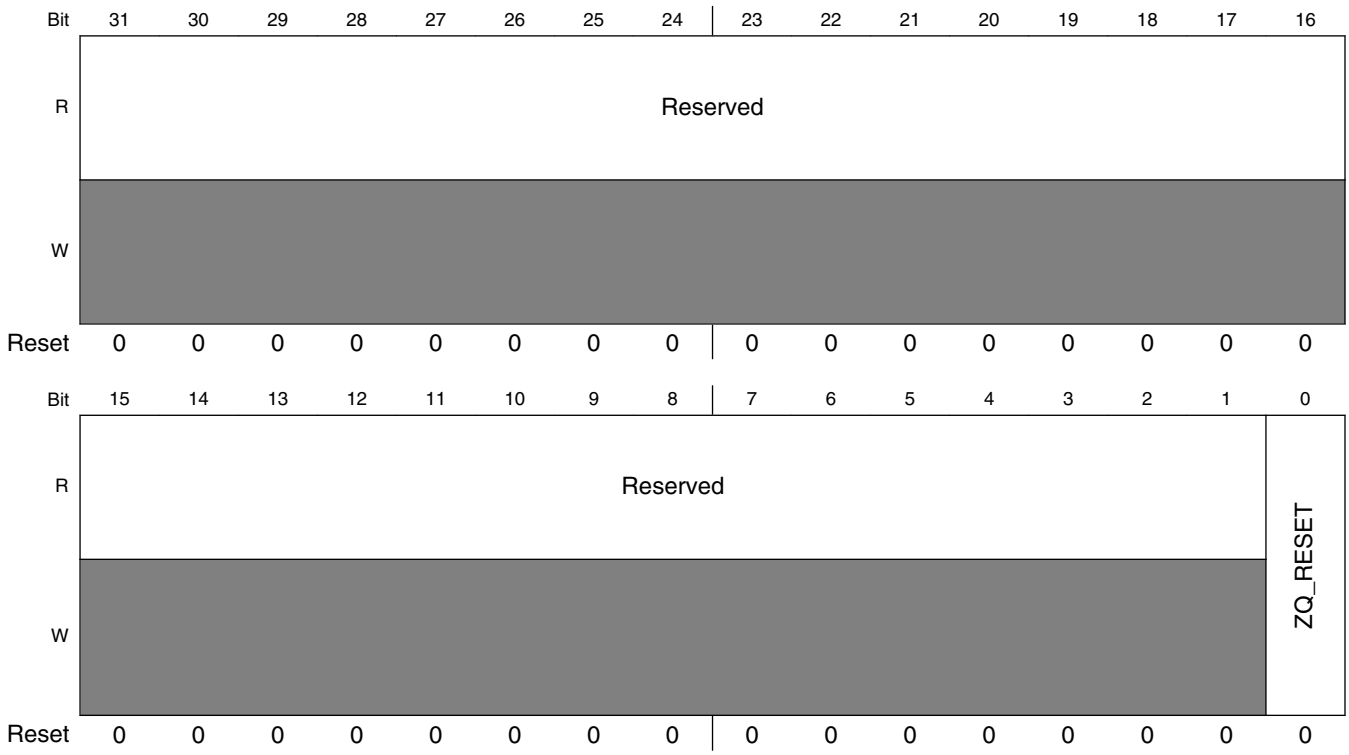
DDRC_ZQCTL1 field descriptions

Field	Description
31–30 Reserved	This field is reserved. Reserved for future use
29–20 T_ZQ_RESET_NOP	tZQReset: Number of cycles of NOP required after a ZQReset (ZQ calibration Reset) command is issued to SDRAM. For configurations with MEMC_FREQ_RATIO = 2, program this to tZQReset / 2 and round it up to the next integer value. Unit: Clock cycles. This is only present for designs supporting LPDDR2 / LPDDR3 devices. **Value After Reset**: 0x20 **Exists**: MEMC_LPDDR2 == 1
T_ZQ_SHORT_INTERVAL_X1024	Average interval to wait between automatically issuing ZQCS (ZQ calibration short) commands to DDR3 / LPDDR2 / LPDDR3 devices. Meaningless, if ZQCTL0.dis_auto_zq = 1. Unit: 1024 clock cycles. This is only present for designs supporting DDR3 or LPDDR2/LPDDR3 devices. **Value After Reset**: 0x100 **Exists**: MEMC_DDR3 == 1 || MEMC_LPDDR2 == 1

9.2.5.2.33 ZQ Control Register 2 (DDRC_ZQCTL2)

Exists: MEMC_LPDDR2==1

Address: 307A_0000h base + 188h offset = 307A_0188h



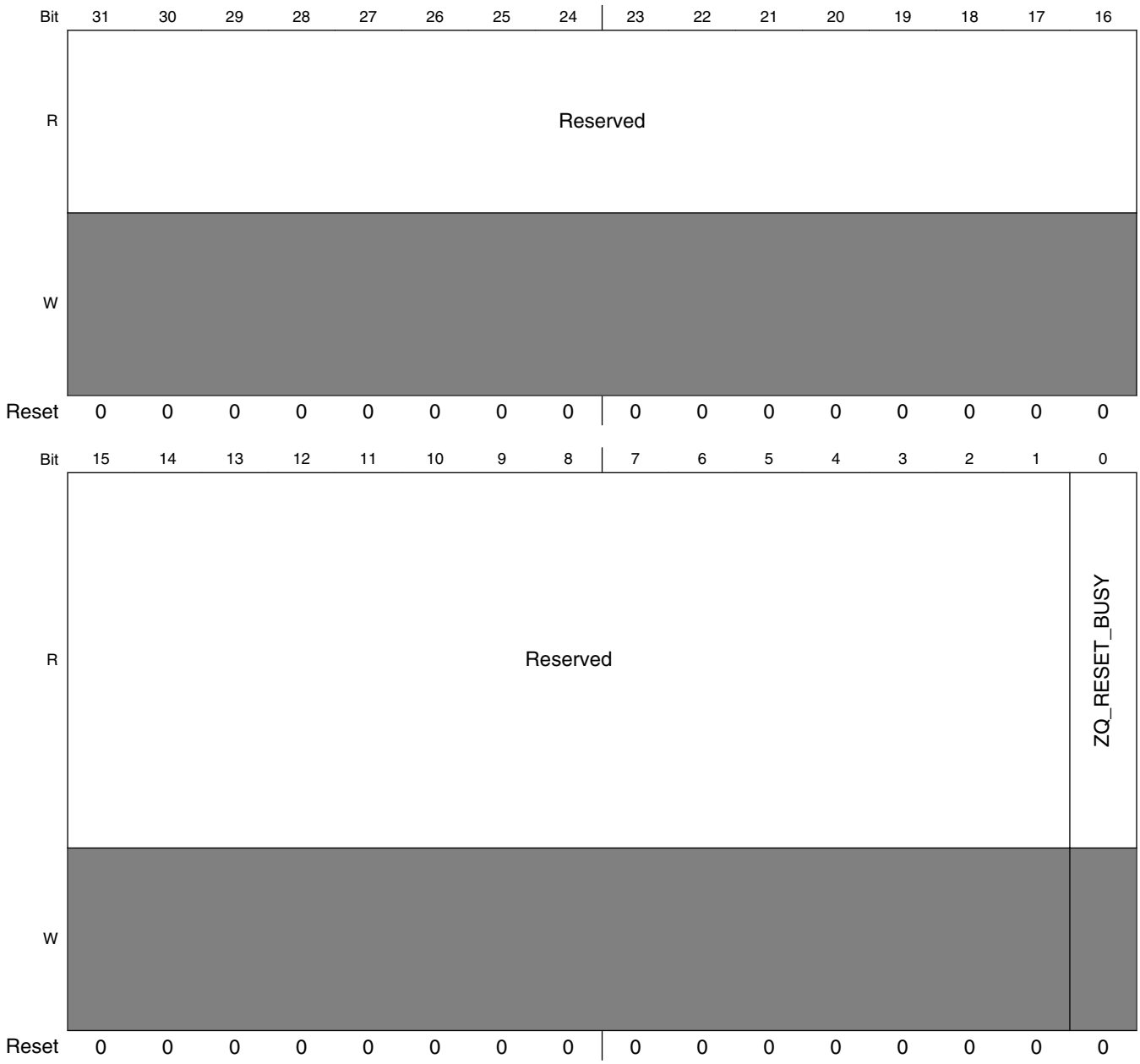
DDRC_ZQCTL2 field descriptions

Field	Description
31–1 Reserved	This field is reserved. Reserved for future use
0 ZQ_RESET	Setting this register bit to 1 triggers a ZQ Reset operation.When the ZQ Reset operation is complete, the DDRC automatically clears this bit. It is recommended NOT to set this signal if in Init, Self-Refresh or Deep power- down operating modes. This is only present for designs supporting LPDDR2 / LPDDR3 devices. **Value After Reset**: 0x0 **Exists**: MEMC_LPDDR2 == 1

9.2.5.2.34 ZQ Status Register (DDRC_ZQSTAT)

Exists: MEMC_LPDDR2==1

Address: 307A_0000h base + 18Ch offset = 307A_018Ch



DDRC_ZQSTAT field descriptions

Field	Description
31–1 Reserved	This field is reserved. Reserved for future use

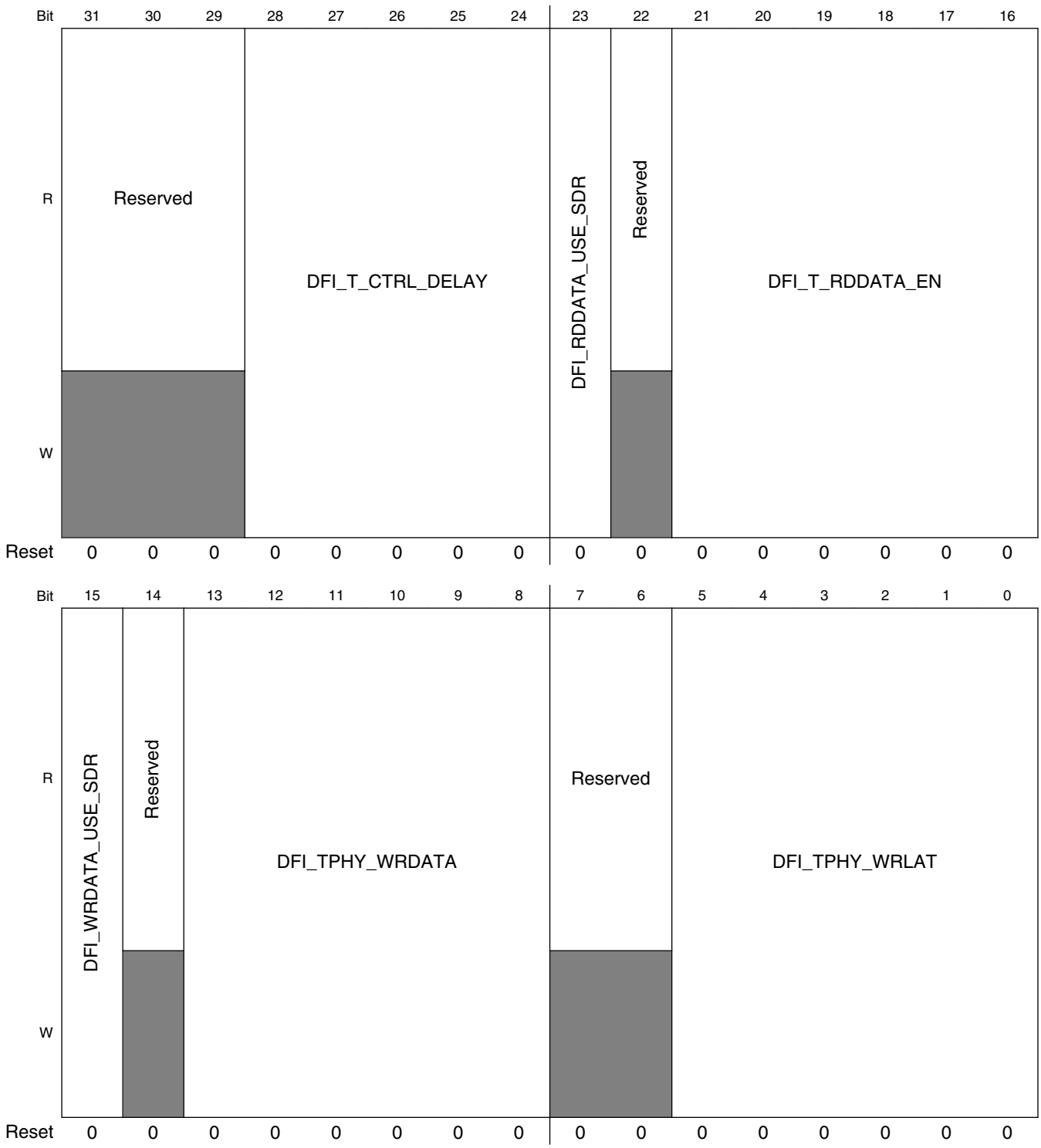
Table continues on the next page...

DDRC_ZQSTAT field descriptions (continued)

Field	Description
0 ZQ_RESET_ BUSY	SoC core may initiate a ZQ Reset operation only if this signal is low. This signal goes high in the clock after the DDRC accepts the ZQ Reset request. It goes low when the ZQ Reset command is issued to the SDRAM and the associated NOP period is over. It is recommended not to perform ZQ Reset commands when this signal is high. **Value After Reset**: 0x0 **Exists**: Always 0 Indicates that the SoC core can initiate a ZQ Reset operation. 1 Indicates that ZQ Reset operation is in progress.

9.2.5.2.35 DFI Timing Register 0 (DDRC_DFITMG0)

Address: 307A_0000h base + 190h offset = 307A_0190h



DDRC_DFITMG0 field descriptions

Field	Description
31–29 Reserved	This field is reserved. Reserved for future use
28–24 DFI_T_CTRL_DELAY	Specifies the number of DFI clock cycles after an assertion or de-assertion of the DFI control signals that the control signals at the PHY-DRAM interface reflect the assertion or deassertion. If the DFI clock and the memory clock are not phase-aligned, this timing parameter should be rounded up to the next integer value **Value After Reset**: 0x7. **Exists**: Always
23 DFI_RDDATA_USE_SDR	Defines whether dfi_rddata_en / dfi_rddata / dfi_rddata_valid is generated using HDR or SDR values Selects whether value in DFITMG0.dfi_t_rddata_en is in terms of SDR or HDR clock cycles. **Value After Reset**: 0x0 **Exists**: MEMC_FREQ_RATIO == 2 0 In terms of HDR clock cycles 1 In terms of SDR clock cycles Refer to PHY specification for correct value.
22 Reserved	This field is reserved. Reserved for future use
21–16 DFI_T_RDDATA_EN	Time from the assertion of a read command on the DFI interface to the assertion of the dfi_rddata_en signal. Refer to PHY specification for correct value. This corresponds to the DFI parameter trddata_en. Note that, depending on the PHY, if using RDIMM, it may be necessary to use the value (CL + 1) in the calculation of trddata_en. This is to compensate for the extra cycle of latency through the RDIMM. Unit: Clocks **Value After Reset**: 0x2 **Exists**: Always
15 DFI_WRDATA_USE_SDR	Defines whether dfi_wrdata_en / dfi_wrdata / dfi_wrdata_mask is generated using HDR or SDR values Selects whether value in DFITMG0.dfi_tphy_wrlat is in terms of SDR or HDR clock cycles Selects whether value in DFITMG0.dfi_tphy_wrdata is in terms of SDR or HDR clock cycles. Refer to PHY specification for correct value. **Value After Reset**: 0x0 **Exists**: MEMC_FREQ_RATIO==2 0 In terms of HDR clock cycles 1 In terms of SDR clock cycles
14 Reserved	This field is reserved. Reserved for future use
13–8 DFI_TPHY_WRDATA	Specifies the number of clock cycles between when dfi_wrdata_en is asserted to when the associated write data is driven on the dfi_wrdata signal. This corresponds to the DFI timing parameter tphy_wrdata. Refer to PHY specification for correct value. **NOTE:** The max supported value is 8. Unit: Clocks **Value After Reset**: 0x0 **Exists**: Always
7–6 Reserved	This field is reserved. Reserved for future use

Table continues on the next page...

DDRC_DFITMG0 field descriptions (continued)

Field	Description
DFI_TPHY_ WRLAT	Write latency Number of clocks from the write command to write data enable (dfi_wrdata_en). This corresponds to the DFI timing parameter tphy_wrlat. Refer to PHY specification for correct value.Note that, depending on the PHY, if using RDIMM, it may be necessary to use the value (CL + 1) in the calculation of tphy_wrlat. This is to compensate for the extra cycle of latency through the RDIMM. **Value After Reset**: 0x2 **Exists**: Always

9.2.5.2.36 DFI Timing Register 1 (DDRC_DFITMG1)

Address: 307A_0000h base + 194h offset = 307A_0194h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved						Reserved		Reserved			DFI_T_WRDATA_DELAY				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				DFI_T_DRAM_CLK_ DISABLE				Reserved				DFI_T_DRAM_CLK_ ENABLE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_DFITMG1 field descriptions

Field	Description
31–26 Reserved	This field is reserved. Reserved for future use
25–24 Reserved	This field is reserved.
23–21 Reserved	This field is reserved. Reserved for future use
20–16 DFI_T_ WRDATA_ DELAY	Specifies the number of DFI clocks between when the dfi_wrdata_en signal is asserted and when the corresponding write data transfer is completed on the DRAM bus. This corresponds to the DFI timing parameter twrdata_delay. Refer to PHY specification for correct value. For DFI 3.0 PHY, set to twrdata_delay, a new timing parameter introduced in DFI 3.0. For DFI 2.1 PHY, set to tphy_wrdata + (delay of DFI write data to the DRAM). Value to be programmed is in terms of DFI clocks, not PHY clocks. In FREQ_RATIO = 2, divide PHY's value by 2 and round up to next integer. If usingDFITMG0.dfi_wrdata_use_sdr = 1, add 1 to the value. Unit: Clocks **Value After Reset**: 0x0 **Exists**: Always
15–12 Reserved	This field is reserved. Reserved for future use

Table continues on the next page...

DDRC_DFITMG1 field descriptions (continued)

Field	Description
11–8 DFI_T_DRAM_CLK_DISABLE	Specifies the number of DFI clock cycles from the assertion of the dfi_dram_clk_disable signal on the DFI until the clock to the DRAM memory devices, at the PHY-DRAM boundary, maintains a low value. If the DFI clock and the memory clock are not phase aligned, this timing parameter should be rounded up to the next integer value. **Value After Reset**: 0x4 **Exists**: Always
7–4 Reserved	This field is reserved. Reserved for future use
DFI_T_DRAM_CLK_ENABLE	Specifies the number of DFI clock cycles from the de-assertion of the dfi_dram_clk_disable signal on the DFI until the first valid rising edge of the clock to the DRAM memory devices, at the PHY-DRAM boundary. If the DFI clock and the memory clock are not phase aligned, this timing parameter should be rounded up to the next integer value. **Value After Reset**: 0x4 **Exists**: Always

9.2.5.2.37 DFI Low Power Configuration Register 0 (DDRC_DFILPCFG0)

Address: 307A_0000h base + 198h offset = 307A_0198h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				DFI_TLP_RESP				DFI_LP_WAKEUP_DPD				Reserved			DFI_LP_EN_DPD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DFI_LP_WAKEUP_SR				Reserved			DFI_LP_EN_SR	DFI_LP_WAKEUP_PD				Reserved			DFI_LP_EN_PD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_DFILPCFG0 field descriptions

Field	Description
31–28 Reserved	This field is reserved. Reserved for future use
27–24 DFI_TLP_RESP	Setting for DFI's tlp_resp time. Same value is used for both Power Down, Self Refresh, Deep Power Down and Maximum Power Saving modes. DFI 2.1 specification onwards, recommends using a fixed value of 7 always. **Value After Reset**: 0x7 **Exists**: Always
23–20 DFI_LP_ WAKEUP_DPD	Value to drive on dfi_lp_wakeup signal when Deep Power Down mode is entered. Determines the DFI's tlp_wakeup time. This is only present for designs supporting mDDR or LPDDR2 / LPDDR3 devices. **Value After Reset**: 0x0 **Exists**: MEMC_MOBILE == 1 ||MEMC_LPDDR2 == 1 0x0 16 cycles 0x1 32 cycles 0x2 64 cycles 0x3 128 cycles 0x4 256 cycles 0x5 512 cycles 0x6 1024 cycles 0x7 2948 cycles 0x8 4096 cycles 0x9 8192 cycles 0xA 16384 cycles 0xB 32768 cycles 0xC 65536 cycles 0xD 131072 cycles 0xE 262144 cycles 0xF Unlimited
19–17 Reserved	This field is reserved. Reserved for future use
16 DFI_LP_EN_ DPD	Enables DFI Low Power interface handshaking during Deep Power Down Entry / Exit. This is only present for designs supporting mDDR or LPDDR2 / LPDDR3 devices. **Value After Reset**: 0x0 **Exists**: MEMC_LPDDR2 == 1 0 Disabled 1 Enabled
15–12 DFI_LP_ WAKEUP_SR	Value to drive on dfi_lp_wakeup signal when Self Refresh mode is entered. Determines the DFI's tlp_wakeup time. **Value After Reset**: 0x0 **Exists**: Always

Table continues on the next page...

DDRC_DFILPCFG0 field descriptions (continued)

Field	Description
	0x0 16 cycles 0x1 32 cycles 0x2 64 cycles 0x3 128 cycles 0x4 256 cycles 0x5 512 cycles 0x6 1024 cycles 0x7 2948 cycles 0x8 4096 cycles 0x9 8192 cycles 0xA 16384 cycles 0xB 32768 cycles 0xC 65536 cycles 0xD 131072 cycles 0xE 262144 cycles 0xF Unlimited
11–9 Reserved	This field is reserved. Reserved for future use
8 DFI_LP_EN_SR	Enables DFI Low Power interface handshaking during Self Refresh Entry / Exit. **Value After Reset**: 0x0 **Exists**: Always 0 Disabled 1 Enabled
7–4 DFI_LP_ WAKEUP_PD	Value to drive on dfi_lp_wakeup signal when Power Down mode is entered. Determines the DFI's tlp_wakeup time. **Value After Reset**: 0x0 **Exists**: Always 0x0 16 cycles 0x1 32 cycles 0x2 64 cycles 0x3 128 cycles 0x4 256 cycles 0x5 512 cycles 0x6 1024 cycles 0x7 2948 cycles 0x8 4096 cycles 0x9 8192 cycles 0xA 16384 cycles 0xB 32768 cycles 0xC 65536 cycles 0xD 131072 cycles 0xE 262144 cycles 0xF Unlimited

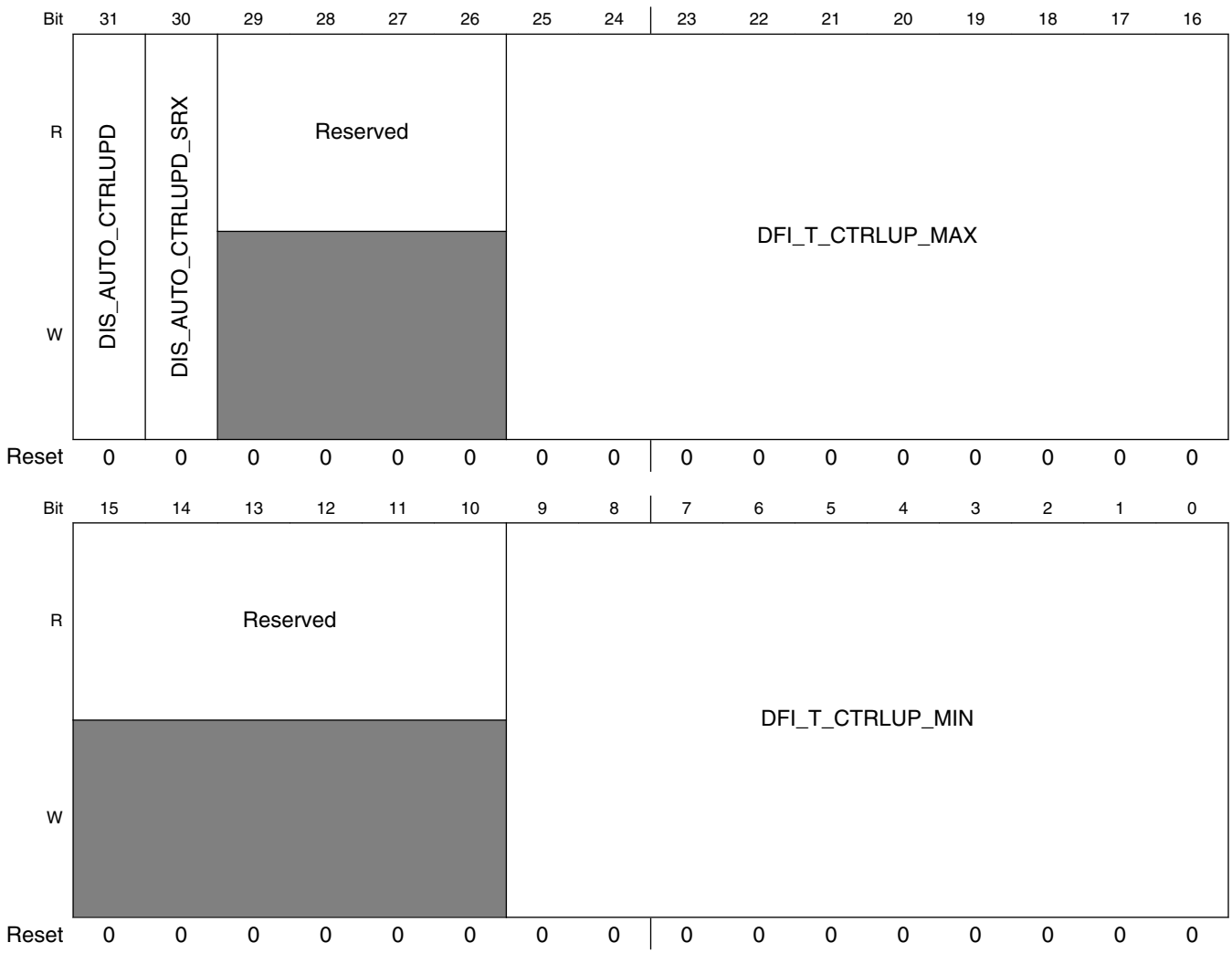
Table continues on the next page...

DDRC_DFILPCFG0 field descriptions (continued)

Field	Description
3–1 Reserved	This field is reserved. Reserved for future use
0 DFI_LP_EN_PD	Enables DFI Low Power interface handshaking during Power Down Entry / Exit. **Value After Reset**: 0x0 **Exists**: Always 0 Disabled 1 Enabled

9.2.5.2.38 DFI Update Register 0 (DDRC_DFIUPD0)

Address: 307A_0000h base + 1A0h offset = 307A_01A0h

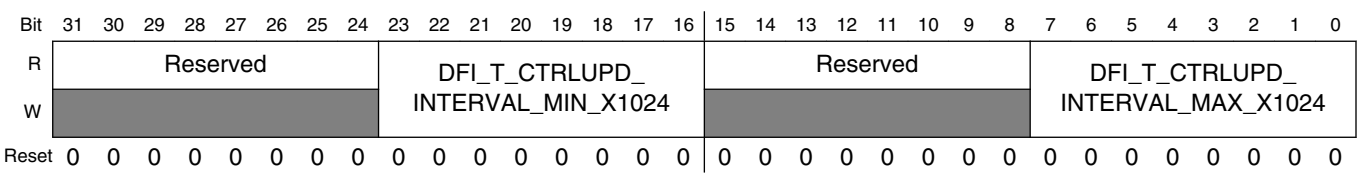


DDRC_DFIUPD0 field descriptions

Field	Description
31 DIS_AUTO_ CTRLUPD	When '1', disable the automatic dfi_ctrlupd_req generation by the DDRC. The core must issue the dfi_ctrlupd_req signal using register reg_ddrc_ctrlupd. This register field is changeable on the fly. When '0', DDRC issues dfi_ctrlupd_req periodically. **Value After Reset**: 0x0 **Exists**: Always
30 DIS_AUTO_ CTRLUPD_SRX	When '1', disable the automatic dfi_ctrlupd_req generation by the DDRC following a self-refresh exit. The core must issue the dfi_ctrlupd_req signal using register reg_ddrc_ctrlupd. This register field is changeable on the fly. When '0', DDRC issues a dfi_ctrlupd_req after exiting self- refresh. **Value After Reset**: 0x0 **Exists**: Always
29–26 Reserved	This field is reserved. Reserved for future use
25–16 DFI_T_CTRLUP_ MAX	Specifies themaximum number of clock cycles that the dfi_ctrlupd_req signal can assert. The lowest value to assign to this variable is 0x40. Unit: Clocks **Value After Reset**: 0x40 **Exists**: Always
15–10 Reserved	This field is reserved. Reserved for future use
DFI_T_CTRLUP_ MIN	Specifies the minimum number of clock cycles that the dfi_ctrlupd_req signal must be asserted. The DDRC expects the PHY to respond within this time. If the PHY does not respond, the DDRC will de-assert dfi_ctrlupd_req after dfi_t_ctrlup_min + 2 cycles. Lowest value to assign to this variable is 0x3. Unit: Clocks **Value After Reset**: 0x3 **Exists**: Always

9.2.5.2.39 DFI Update Register 1 (DDRC_DFIUPD1)

Address: 307A_0000h base + 1A4h offset = 307A_01A4h



DDRC_DFIUPD1 field descriptions

Field	Description
31–24 Reserved	This field is reserved. Reserved for future use
23–16 DFI_T_ CTRLUPD_	This is the minimum amount of time between DDRC initiated DFI update requests (which is executed whenever the DDRC is idle). Set this number higher to reduce the frequency of update requests, which can have a small impact on the latency of the first read request when the DDRC is idle.

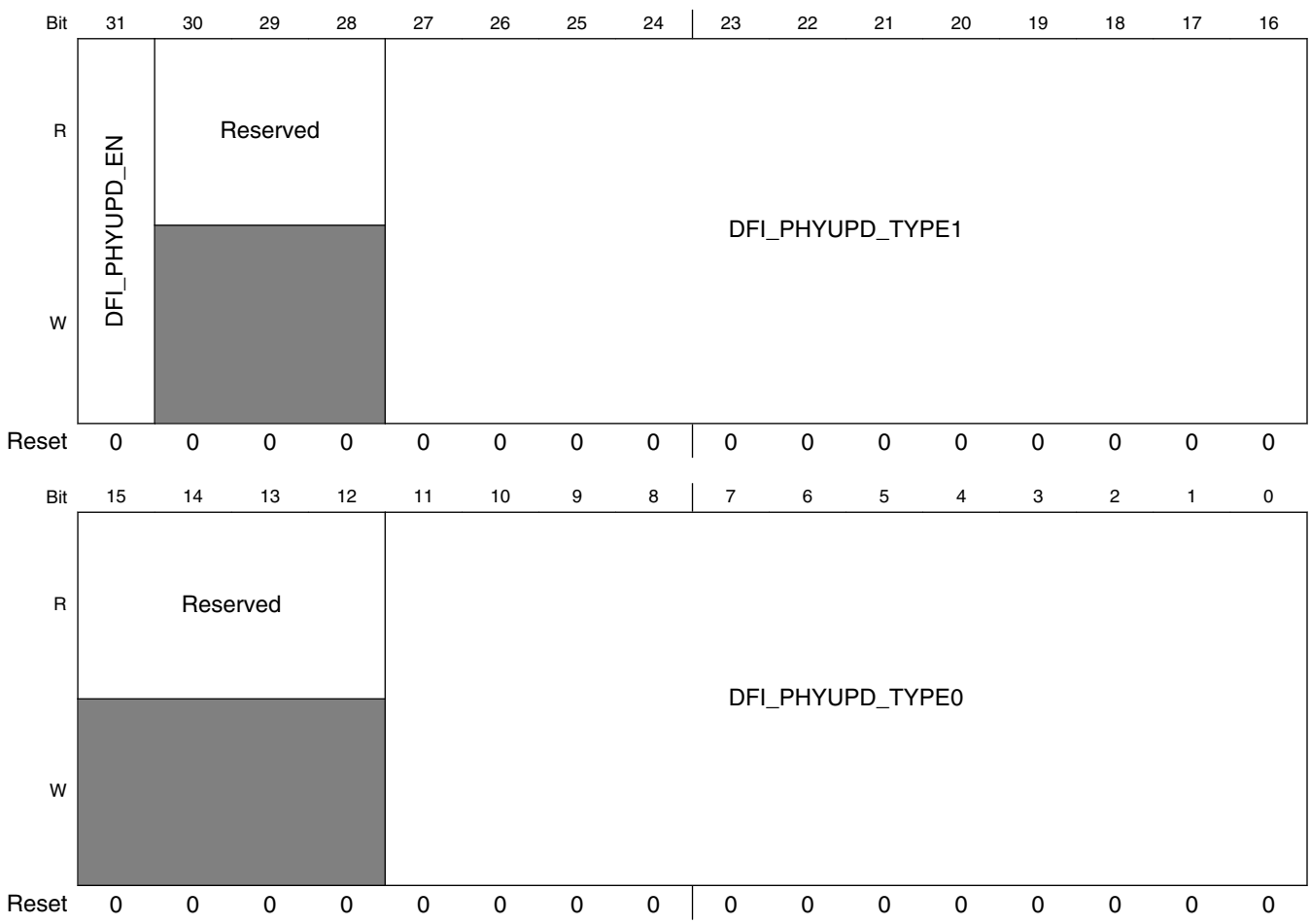
Table continues on the next page...

DDRC_DFIUPD1 field descriptions (continued)

Field	Description
INTERVAL_MIN_X1024	Unit: 1024 clocks **Value After Reset**: 0x0 **Exists**: Always
15–8 Reserved	This field is reserved. Reserved for future use
DFI_T_CTRLUPD_INTERVAL_MAX_X1024	This is the maximum amount of time between DDRC initiated DFI update requests. This timer resets with each update request; when the timer expires dfi_ctrlupd_req is sent and traffic is blocked until the dfi_ctrlupd_ackx is received. PHY can use this idle time to recalibrate the delay lines to the DLLs. The DFI controller update is also used to reset PHY FIFO pointers in case of data capture errors. Updates are required to maintain calibration over PVT, but frequent updates may impact performance. **NOTE:** Value programmed for DFIUPD1.dfi_t_ctrlupd_interval_max_x1024 must be greater than DFIUPD1.dfi_t_ctrlupd_interval_min_x1024. Unit: 1024 clocks **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.40 DFI Update Register 2 (DDRC_DFIUPD2)

Address: 307A_0000h base + 1A8h offset = 307A_01A8h

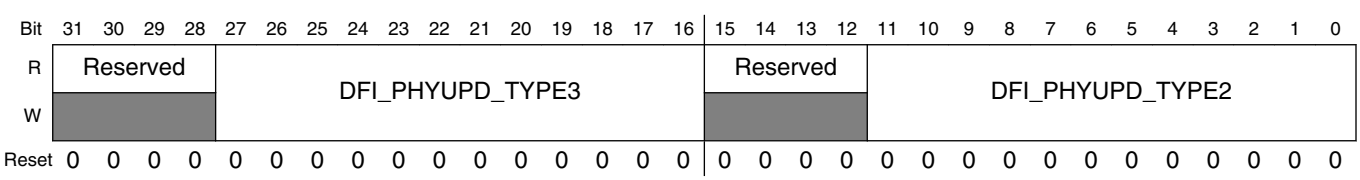


DDRC_DFIUPD2 field descriptions

Field	Description
31 DFI_PHYUPD_EN	Enables the support for acknowledging PHY- initiated updates. **Value After Reset**: 0x1 **Exists**: Always 0 Disabled 1 Enabled
30–28 Reserved	This field is reserved. Reserved for future use
27–16 DFI_PHYUPD_TYPE1	Specifies the maximum number of DFI clock cycles that the dfi_phyupd_req signal may remain asserted after the assertion of the dfi_phyupd_ack signal for dfi_phyupd_type = 2'b01. Is used in conjunction with value defined via DFIUPD4.dfi_phyupd_type1_mult. The dfi_phyupd_req signal may de-assert at any cycle after the assertion of the dfi_phyupd_ack signal. **Value After Reset**: 0x10 **Exists**: Always
15–12 Reserved	This field is reserved. Reserved for future use
DFI_PHYUPD_TYPE0	Specifies the maximum number of DFI clock cycles that the dfi_phyupd_req signal may remain asserted after the assertion of the dfi_phyupd_ack signal for dfi_phyupd_type = 2'b00. Is used in conjunction with value defined via DFIUPD4.dfi_phyupd_type0_mult. The dfi_phyupd_req signal may de-assert at any cycle after the assertion of the dfi_phyupd_ack signal. **Value After Reset**: 0x10 **Exists**: Always

9.2.5.2.41 DFI Update Register 3 (DDRC_DFIUPD3)

Address: 307A_0000h base + 1ACh offset = 307A_01ACh



DDRC_DFIUPD3 field descriptions

Field	Description
31–28 Reserved	This field is reserved. Reserved for future use
27–16 DFI_PHYUPD_TYPE3	Specifies the maximum number of DFI clock cycles that the dfi_phyupd_req signal may remain asserted after the assertion of the dfi_phyupd_ack signal for dfi_phyupd_type = 2'b11. Is used in conjunction with value defined via DFIUPD4.dfi_phyupd_type3_mult. The dfi_phyupd_req signal may de-assert at any cycle after the assertion of the dfi_phyupd_ack signal. **Value After Reset**: 0x10 **Exists**: Always

Table continues on the next page...

DDRC_DFIUPD3 field descriptions (continued)

Field	Description
15–12 Reserved	This field is reserved. Reserved for future use
DFI_PHYUPD_TYPE2	Specifies the maximum number of DFI clock cycles that the dfi_phyupd_req signal may remain asserted after the assertion of the dfi_phyupd_ack signal for dfi_phyupd_type = 2'b10. Is used in conjunction with value defined via DFIUPD4.dfi_phyupd_type2_mult. The dfi_phyupd_req signal may de-assert at any cycle after the assertion of the dfi_phyupd_ack signal. **Value After Reset**: 0x10 **Exists**: Always

9.2.5.2.42 DFI Miscellaneous Control Register (DDRC_DFIMISC)

Address: 307A_0000h base + 1B0h offset = 307A_01B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DFI_INIT_COMPLETE_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

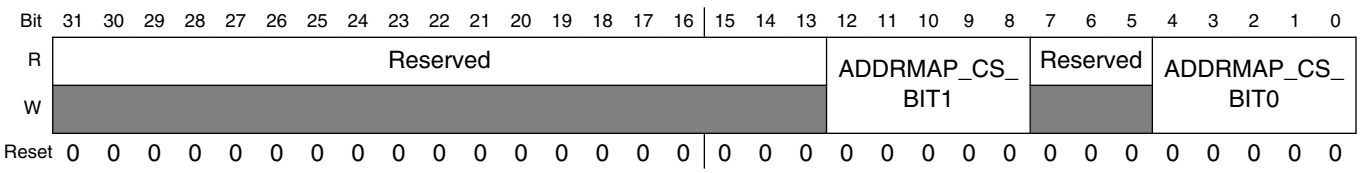
DDRC_DFIMISC field descriptions

Field	Description
31–1 Reserved	This field is reserved. Reserved for future use
0 DFI_INIT_ COMPLETE_EN	PHY initialization complete enable signal. When asserted the dfi_init_complete signal can be used to trigger SDRAM initialisation **Value After Reset**: 0x1 **Exists**: Always

9.2.5.2.43 Address Map Register 0 (DDRC_ADDRMAP0)

Exists: (MEMC_NUM_RANKS>1)

Address: 307A_0000h base + 200h offset = 307A_0200h

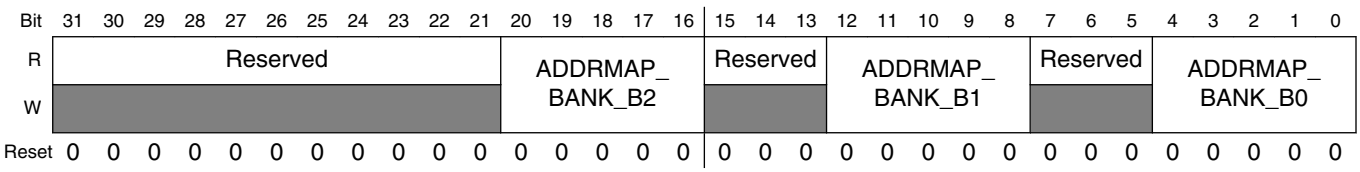


DDRC_ADDRMAP0 field descriptions

Field	Description
31–13 Reserved	This field is reserved. Reserved for future use
12–8 ADDRMAP_CS_ BIT1	Selects the HIF address bit used as rank address bit 1. Valid Range: 0 to 26, and 31 Internal Base: 7 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 31, rank address bit 1 is set to 0. **Value After Reset**: 0x0 **Exists**: MEMC_NUM_RANKS == 4
7–5 Reserved	This field is reserved. Reserved for future use
ADDRMAP_CS_ BIT0	Selects the HIF address bit used as rank address bit 0. Valid Range: 0 to 27, and 31 Internal Base: 6 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 31, rank address bit 0 is set to 0. **Value After Reset**: 0x0 **Exists**: MEMC_NUM_RANKS > 1

9.2.5.2.44 Address Map Register 1 (DDRC_ADDRMAP1)

Address: 307A_0000h base + 204h offset = 307A_0204h



DDRC_ADDRMAP1 field descriptions

Field	Description
31–21 Reserved	This field is reserved. Reserved for future use
20–16 ADDRMAP_ BANK_B2	Selects the HIF address bit used as bank address bit 2. Valid Range: 0 to 29 and 31 Internal Base: 4 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 31, bank address bit 2 is set to 0. **Value After Reset**: 0x0 **Exists**: Always
15–13 Reserved	This field is reserved. Reserved for future use
12–8 ADDRMAP_ BANK_B1	Selects the HIF address bits used as bank address bit 1. Valid Range: 0 to 30 Internal Base: 3 The selected HIF address bit for each of the bank address bits is determined by adding the internal base to the value of this field. **Value After Reset**: 0x0 **Exists**: Always
7–5 Reserved	This field is reserved. Reserved for future use
ADDRMAP_ BANK_B0	Selects the HIF address bits used as bank address bit 0. Valid Range: 0 to 30 Internal Base: 2 The selected HIF address bit for each of the bank address bits is determined by adding the internal base to the value of this field. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.45 Address Map Register 2 (DDRC_ADDRMAP2)

Address: 307A_0000h base + 208h offset = 307A_0208h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				ADDRMAP_COL_B5				Reserved				ADDRMAP_COL_B4				Reserved				ADDRMAP_COL_B3				Reserved				ADDRMAP_COL_B2			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_ADDRMAP2 field descriptions

Field	Description
31–28 Reserved	This field is reserved. Reserved for future use
27–24 ADDRMAP_COL_B5	**Description**: • Full bus width mode: Selects the HIF address bit used as column address bit 5. • Half bus width mode: Selects the HIF address bit used as column address bit 6. • Quarter bus width mode: Selects the HIF address bit used as column address bit 7. Valid Range: 0 to 7, and 15 Internal Base: 5 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 15, this column address bit is set to 0. **Value After Reset**: 0x0 **Exists**: Always
23–20 Reserved	This field is reserved. Reserved for future use
19–16 ADDRMAP_COL_B4	**Description**: • Full bus width mode: Selects the HIF address bit used as column address bit 4. • Half bus width mode: Selects the HIF address bit used as column address bit 5. • Quarter bus width mode: Selects the HIF address bit used as column address bit 6. Valid Range: 0 to 7, and 15 Internal Base: 4 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 15, this column address bit is set to 0. **Value After Reset**: 0x0 **Exists**: Always
15–12 Reserved	This field is reserved. Reserved for future use
11–8 ADDRMAP_COL_B3	• Full bus width mode: Selects the HIF address bit used as column address bit 3. • Half bus width mode: Selects the HIF address bit used as column address bit 4. • Quarter bus width mode: Selects the HIF address bit used as column address bit 5. Valid Range: 0 to 7 Internal Base: 3 The selected HIF address bit is determined by adding the internal base to the value of this field.

Table continues on the next page...

DDRC_ADDRMAP2 field descriptions (continued)

Field	Description
	Value After Reset: 0x0 **Exists**: Always
7–4 Reserved	This field is reserved. Reserved for future use
ADDRMAP_ COL_B2	• Full bus width mode: Selects the HIF address bit used as column address bit 2. • Half bus width mode: Selects the HIF address bit used as column address bit 3. • Quarter bus width mode: Selects the HIF address bit used as column address bit 4. Valid Range: 0 to 7 Internal Base: 2 The selected HIF address bit is determined by adding the internal base to the value of this field. **NOTE:** If DDRC_INCL_ARB = 1 and MEMC_BURST_LENGTH = 8, it is required to program this to 0. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.46 Address Map Register 3 (DDRC_ADDRMAP3)

Address: 307A_0000h base + 20Ch offset = 307A_020Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				ADDRMAP_ COL_B9				Reserved				ADDRMAP_ COL_B8				Reserved				ADDRMAP_ COL_B7				Reserved				ADDRMAP_ COL_B6			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_ADDRMAP3 field descriptions

Field	Description
31–28 Reserved	This field is reserved. Reserved for future use
27–24 ADDRMAP_ COL_B9	**Description**: • Full bus width mode: Selects the HIF address bit used as column address bit 9. • Half bus width mode: Selects the HIF address bit used as column address bit 11 (10 in LPDDR2 / LPDDR3 mode) • Quarter bus width mode: Selects the HIF address bit used as column address bit 13 (11 in LPDDR2 / LPDDR3 mode). Valid Range: 0 to 7, and 15 Internal Base: 9 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 15, this column address bit is set to 0. **NOTE:** Per JEDEC DDR2 / DDR3 / mDDR specification, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10. In LPDDR2 / LPDDR3, there is a dedicated bit for auto- precharge in the CA bus and hence column bit 10 is used.

Table continues on the next page...

DDRC_ADDRMAP3 field descriptions (continued)

Field	Description
	Value After Reset: 0x0 **Exists**: Always
23–20 Reserved	This field is reserved. Reserved for future use
19–16 ADDRMAP_ COL_B8	**Description**: • Full bus width mode: Selects the HIF address bit used as column address bit 8. • Half bus width mode: Selects the HIF address bit used as column address bit 9. • Quarter bus width mode: Selects the HIF address bit used as column address bit 11 (10 in LPDDR2 / LPDDR3 mode). Valid Range: 0 to 7, and 15 Internal Base: 8 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 15, this column address bit is set to 0. **NOTE:** Per JEDEC DDR2 / DDR3 / mDDR specification, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10. In LPDDR2 / LPDDR3, there is a dedicated bit for auto-precharge in the CA bus and hence column bit 10 is used. **Value After Reset**: 0x0 **Exists**: Always
15–12 Reserved	This field is reserved. Reserved for future use
11–8 ADDRMAP_ COL_B7	**Description**: • Full bus width mode: Selects the HIF address bit used as column address bit 7. • Half bus width mode: Selects the HIF address bit used as column address bit 8. • Quarter bus width mode: Selects the HIF address bit used as column address bit 9. Valid Range: 0 to 7, and 15 Internal Base: 7 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 15, this column address bit is set to 0. **Value After Reset**: 0x0 **Exists**: Always
7–4 Reserved	This field is reserved. Reserved for future use
ADDRMAP_ COL_B6	**Description**: • Full bus width mode: Selects the HIF address bit used as column address bit 6. • Half bus width mode: Selects the HIF address bit used as column address bit 7. • Quarter bus width mode: Selects the HIF address bit used as column address bit 8. Valid Range: 0 to 7, and 15 Internal Base: 6 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 15, this column address bit is set to 0. **Value After Reset**: 0x0

Table continues on the next page...

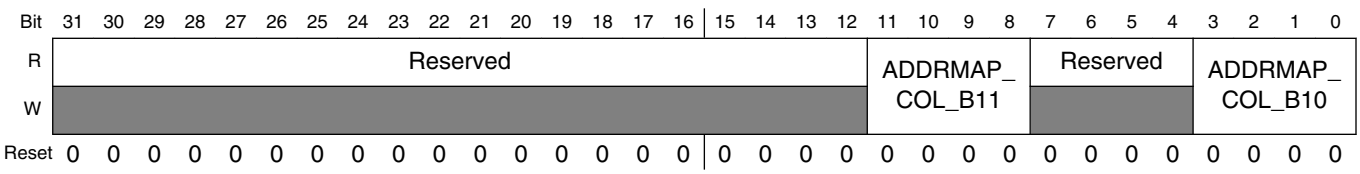
DDRC_ADDRMAP3 field descriptions (continued)

Field	Description
	Exists: Always

9.2.5.2.47 Address Map Register 4 (DDRC_ADDRMAP4)

Exists: Always

Address: 307A_0000h base + 210h offset = 307A_0210h



DDRC_ADDRMAP4 field descriptions

Field	Description
31–12 Reserved	This field is reserved. Reserved for future use
11–8 ADDRMAP_ COL_B11	**Description**: • Full bus width mode: Selects the HIF address bit used as column address bit 13 (11 in LPDDR2 / LPDDR3 mode). • Half bus width mode: Unused. Tomake it unused, this should be tied to 4'hF. • Quarter bus width mode: Unused. Tomake it unused, this must be tied to 4'hF. Valid Range: 0 to 7, and 15 Internal Base: 11 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 15, this column address bit is set to 0. **NOTE:** Per JEDEC DDR2 / DDR3 / mDDR specification, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10. In LPDDR2 / LPDDR3, there is a dedicated bit for auto-precharge in the CA bus and hence column bit 10 is used. **Value After Reset**: 0x0 **Exists**: Always
7–4 Reserved	This field is reserved. Reserved for future use
ADDRMAP_ COL_B10	**Description**: • Full bus width mode: Selects the HIF address bit used as column address bit 11 (10 in LPDDR2 / LPDDR3 mode). • Half bus width mode: Selects the HIF address bit used as column address bit 13 (11 in LPDDR2 / LPDDR3 mode). • Quarter bus width mode: UNUSED. To make it unused, this must be tied to 4'hF. Valid Range: 0 to 7, and 15

Table continues on the next page...

DDRC_ADDRMAP4 field descriptions (continued)

Field	Description
	Internal Base: 10 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 15, this column address bit is set to 0. **NOTE:** Per JEDEC DDR2 / DDR3 / mDDR specification, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10. In LPDDR2 / LPDDR3, there is a dedicated bit for auto-precharge in the CA bus and hence column bit 10 is used. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.48 Address Map Register 5 (DDRC_ADDRMAP5)

Address: 307A_0000h base + 214h offset = 307A_0214h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				ADDRMAP_ROW_B11				Reserved				ADDRMAP_ROW_B2_10				Reserved				ADDRMAP_ROW_B1				Reserved				ADDRMAP_ROW_B0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_ADDRMAP5 field descriptions

Field	Description
31–28 Reserved	This field is reserved. Reserved for future use
27–24 ADDRMAP_ROW_B11	Selects the HIF address bit used as row address bit 11. Valid Range: 0 to 11, and 15 Internal Base: 17 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 15, row address bit 11 is set to 0. **Value After Reset**: 0x0 **Exists**: Always
23–20 Reserved	This field is reserved. Reserved for future use
19–16 ADDRMAP_ROW_B2_10	Selects the HIF address bits used as row address bits 2 to 10. Valid Range: 0 to 11 Internal Base: 8 (for row address bit 2), 9 (for row address bit 3), 10 (for row address bit 4) etc increasing to 16 (for row address bit 10) The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field. **Value After Reset**: 0x0 **Exists**: Always

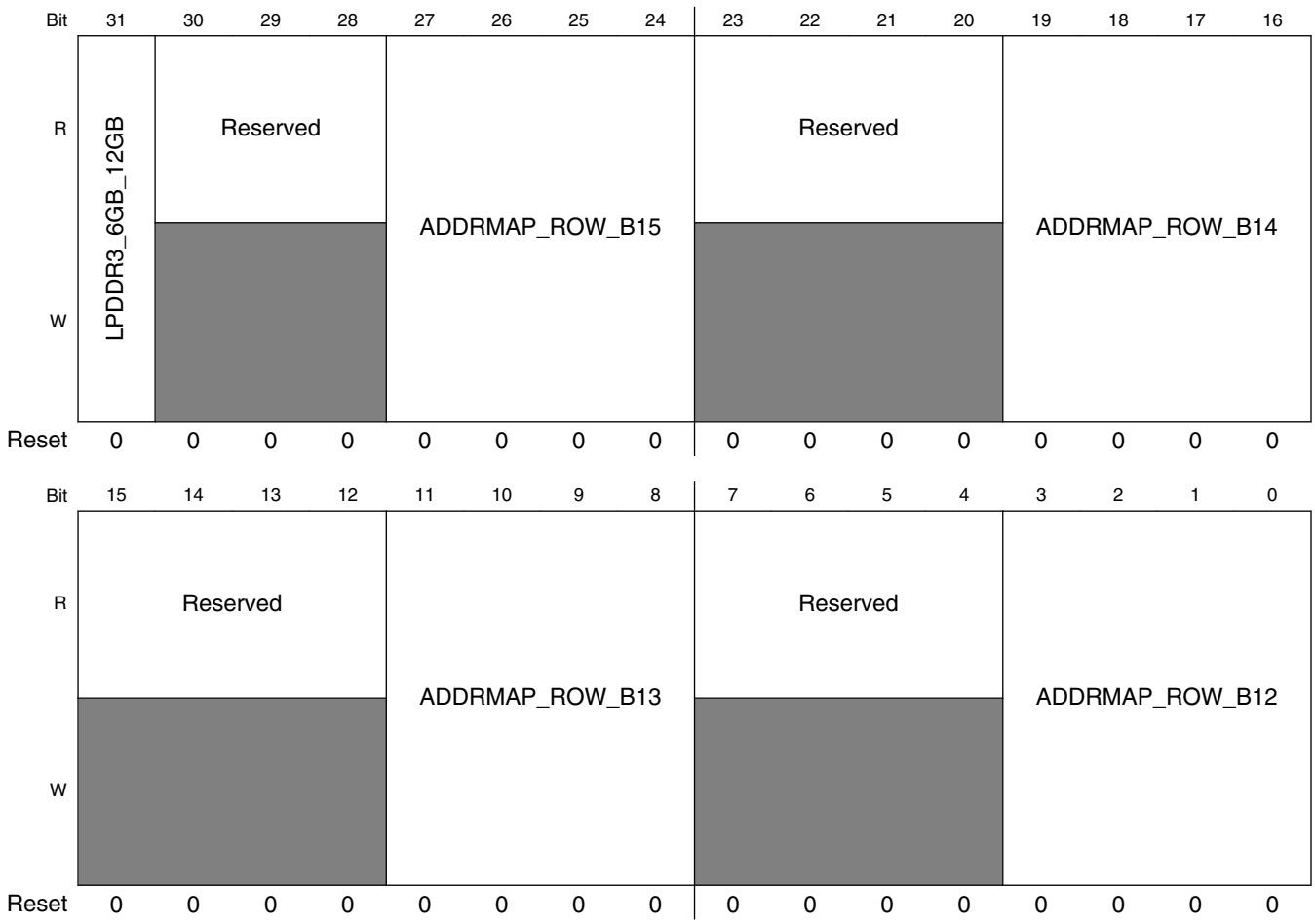
Table continues on the next page...

DDRC_ADDRMAP5 field descriptions (continued)

Field	Description
15–12 Reserved	This field is reserved. Reserved for future use
11–8 ADDRMAP_ ROW_B1	Selects the HIF address bits used as row address bit 1. Valid Range: 0 to 11 Internal Base: 7 The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field. **Value After Reset**: 0x0 **Exists**: Always
7–4 Reserved	This field is reserved. Reserved for future use
ADDRMAP_ ROW_B0	Selects the HIF address bits used as row address bit 0. Valid Range: 0 to 11 Internal Base: 6 The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.49 Address Map Register 6 (DDRC_ADDRMAP6)

Address: 307A_0000h base + 218h offset = 307A_0218h



DDRC_ADDRMAP6 field descriptions

Field	Description
31 LPDDR3_6GB_12GB	Set this to 1 if there is an LPDDR3 SDRAM 6 Gb or 12 Gb device in use. Present only in designs configured to support LPDDR3. **Value After Reset**: 0x0 **Exists**: MEMC_LPDDR3 == 1 1 LPDDR3 SDRAM 6 Gb / 12 Gb device in use. Every address having row[14:13] == 2'b11 is considered as invalid 0 Non-LPDDR3 6 Gb / 12 Gb device in use. All addresses are valid
30–28 Reserved	This field is reserved. Reserved for future use
27–24 ADDRMAP_ROW_B15	Selects the HIF address bit used as row address bit 15. Valid Range: 0 to 11, and 15 Internal Base: 21

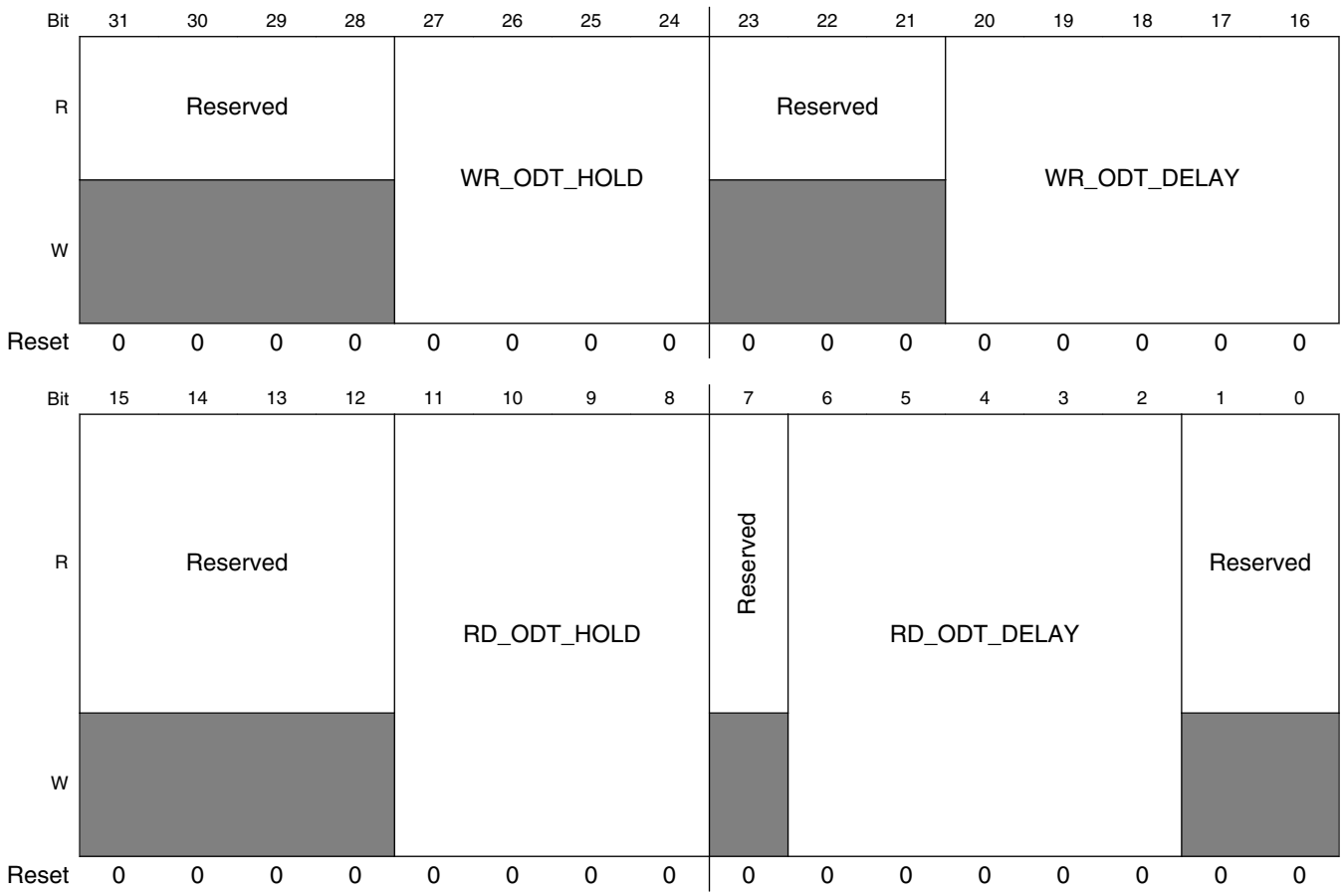
Table continues on the next page...

DDRC_ADDRMAP6 field descriptions (continued)

Field	Description
	The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 15, row address bit 15 is set to 0. **Value After Reset**: 0x0 **Exists**: Always
23–20 Reserved	This field is reserved. Reserved for future use
19–16 ADDRMAP_ ROW_B14	Selects the HIF address bit used as row address bit 14. Valid Range: 0 to 11, and 15 Internal Base: 20 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 15, row address bit 14 is set to 0. **Value After Reset**: 0x0 **Exists**: Always
15–12 Reserved	This field is reserved. Reserved for future use
11–8 ADDRMAP_ ROW_B13	Selects the HIF address bit used as row address bit 13. Valid Range: 0 to 11, and 15 Internal Base: 19 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 15, row address bit 13 is set to 0. **Value After Reset**: 0x0 **Exists**: Always
7–4 Reserved	This field is reserved. Reserved for future use
ADDRMAP_ ROW_B12	Selects the HIF address bit used as row address bit 12. Valid Range: 0 to 11, and 15 Internal Base: 18 The selected HIF address bit is determined by adding the internal base to the value of this field. If set to 15, row address bit 12 is set to 0. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.50 ODT Configuration Register (DDRC_ODTCFG)

Address: 307A_0000h base + 240h offset = 307A_0240h



DDRC_ODTCFG field descriptions

Field	Description
31–28 Reserved	This field is reserved. Reserved for future use
27–24 WR_ODT_HOLD	Cycles to hold ODT for a write command. The minimum supported value is 2. DDR3: • BL8 - 0x6 • BL4 - 0x4 **Value After Reset**: 0x4 **Exists**: Always
23–21 Reserved	This field is reserved. Reserved for future use
20–16 WR_ODT_ DELAY	The delay, in clock cycles, from issuing a write command to setting ODT values associated with that command. ODT setting must remain constant for the entire time that DQS is driven by the DDRC. ODT is used only in DDR3 designs. Recommended values: DDR3 - 0

Table continues on the next page...

DDRC_ODTCFG field descriptions (continued)

Field	Description
	Value After Reset: 0x0 **Exists**: Always
15–12 Reserved	This field is reserved. Reserved for future use
11–8 RD_ODT_HOLD	Cycles to hold ODT for a read command. The minimum supported value is 2. Recommended values: DDR3: • BL8 - 0x6 • BL4 - 0x4 **Value After Reset**: 0x4 **Exists**: Always
7 Reserved	This field is reserved. Reserved for future use
6–2 RD_ODT_ DELAY	The delay, in clock cycles, from issuing a read command to setting ODT values associated with that command. ODT setting must remain constant for the entire time that DQS is driven by the DDRC. Recommended values: DDR3: • (CL - CWL) **Value After Reset**: 0x0 **Exists**: Always
Reserved	This field is reserved. Reserved for future use

9.2.5.2.51 ODT / Rank Map Register (DDRC_ODTMAP)

Address: 307A_0000h base + 244h offset = 307A_0244h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RANK1_RD_ODT				RANK1_WR_ODT				RANK0_RD_ODT				RANK0_WR_ODT			
W													RANK0_WR_ODT			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_ODTMAP field descriptions

Field	Description
31–16 Reserved	This field is reserved. Reserved for future use

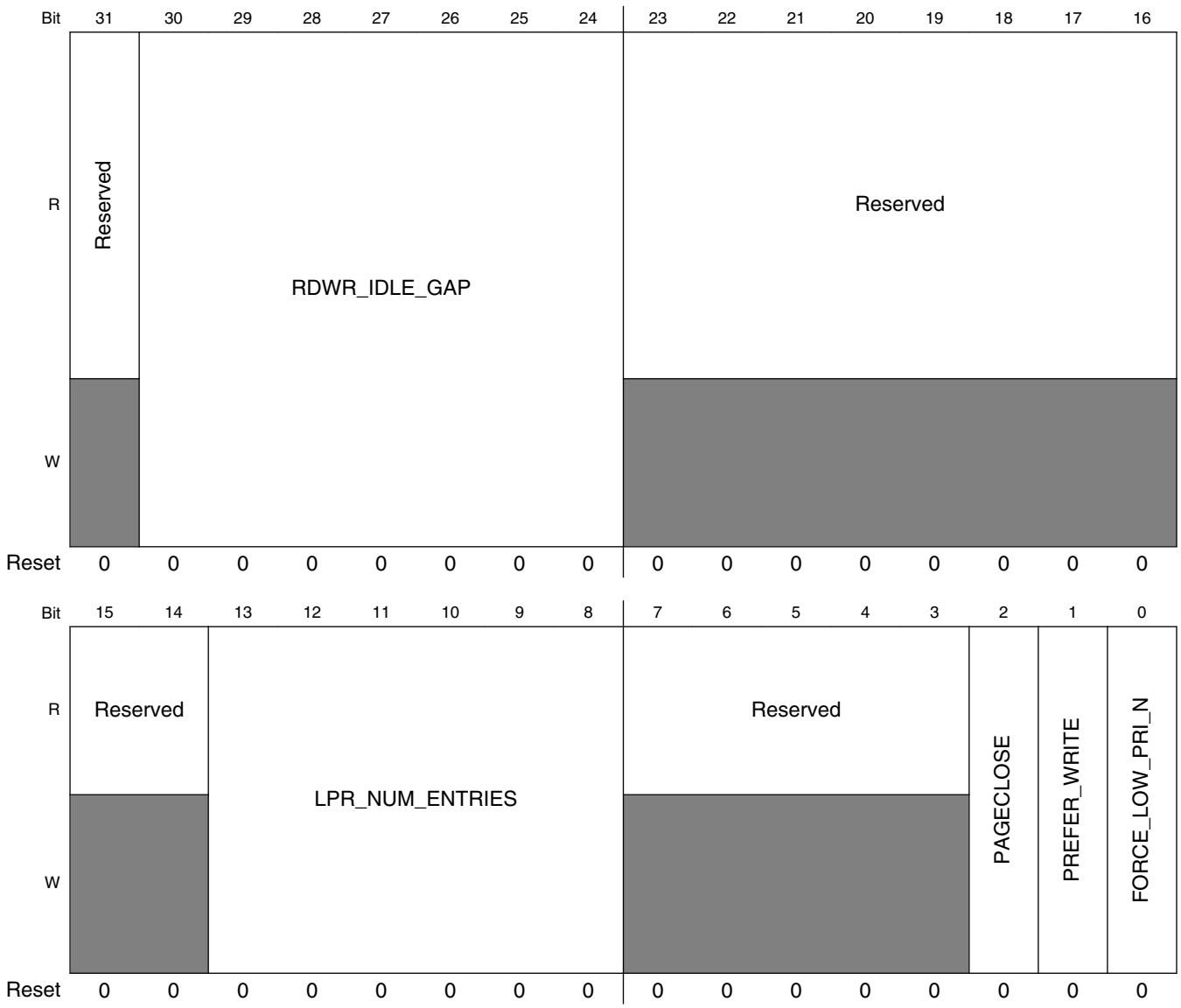
Table continues on the next page...

DDRC_ODTMAP field descriptions (continued)

Field	Description
15–12 RANK1_RD_ODT	Indicates which remote ODTs must be turned on during a read from rank 1. Each rank has a remote ODT (in the SDRAM) which can be turned on by setting the appropriate bit here. Rank 0 is controlled by the LSB; rank 1 is controlled by bit next to the LSB, etc. For each rank, set its bit to 1 to enable its ODT. Present only in configurations that have 2 or more ranks **Value After Reset**: "(MEMC_NUM_RANKS > 1) ? 0x2 : 0x0" **Exists**: MEMC_NUM_RANKS > 1
11–8 RANK1_WR_ODT	Indicates which remote ODTs must be turned on during a write to rank 1. Each rank has a remote ODT (in the SDRAM) which can be turned on by setting the appropriate bit here. Rank 0 is controlled by the LSB; rank 1 is controlled by bit next to the LSB, etc. For each rank, set its bit to 1 to enable its ODT. Present only in configurations that have 2 or more ranks **Value After Reset**: "(MEMC_NUM_RANKS > 1) ? 0x2 : 0x0" **Exists**: MEMC_NUM_RANKS > 1
7–4 RANK0_RD_ODT	Indicates which remote ODTs must be turned on during a read from rank 0. Each rank has a remote ODT (in the SDRAM) which can be turned on by setting the appropriate bit here. Rank 0 is controlled by the LSB; rank 1 is controlled by bit next to the LSB, etc. For each rank, set its bit to 1 to enable its ODT. **Value After Reset**: 0x1 **Exists**: Always
RANK0_WR_ODT	Indicates which remote ODTs must be turned on during a write to rank 0. Each rank has a remote ODT (in the SDRAM) which can be turned on by setting the appropriate bit here. Rank 0 is controlled by the LSB; rank 1 is controlled by bit next to the LSB, etc. For each rank, set its bit to 1 to enable its ODT. **Value After Reset**: 0x1 **Exists**: Always

9.2.5.2.52 Scheduler Control Register (DDRC_SCHED)

Address: 307A_0000h base + 250h offset = 307A_0250h



DDRC_SCHED field descriptions

Field	Description
31 Reserved	This field is reserved. Reserved for future use
30–24 RDWR_IDLE_GAP	When the preferred transaction store is empty for these many clock cycles, switch to the alternate transaction store if it is non-empty. The read transaction store (both high and low priority) is the default preferred transaction store and the write transaction store is the alternative store. When prefer write over read is set this is reversed. 0x0 is a legal value for this register. When set to 0x0, the transaction store switching will happen immediately when the switching conditions become true.

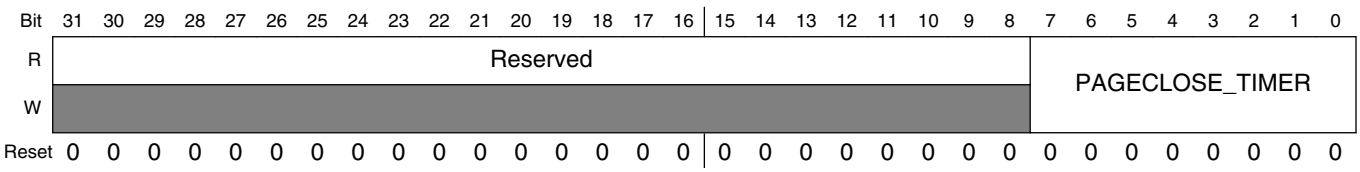
Table continues on the next page...

DDRC_SCHED field descriptions (continued)

Field	Description
	FOR PERFORMANCE ONLY **Value After Reset**: 0x0 **Exists**: Always
23–14 Reserved	This field is reserved. Reserved for future use
13–8 LPR_NUM_ENTRIES	Number of entries in the low priority transaction store is this value + 1. (MEMC_NO_OF_ENTRY - (SCHED.lpr_num_entries + 1)) is the number of entries available for the high priority transaction store. Setting this to maximum value allocates all entries to low priority transaction store. Setting this to 0 allocates 1 entry to low priority transaction store and the rest to high priority transaction store. **Value After Reset**: "MEMC_NO_OF_ENTRY / 2" **Exists**: Always
7–3 Reserved	This field is reserved. Reserved for future use
2 PAGECLOSE	If true, bank is kept open only until there are page hit transactions available in the CAM to that bank. The last read or write command in the CAM with a bank and page hit will be executed with auto-precharge if SCHED1.pageclose_timer = 0. Even if this register set to 1 and SCHED1.pageclose_timer is set to 0, explicit precharge (and not auto-precharge) may be issued in some cases where there is a mode switch betweenWrite and Read or between LPR and HPR. The Read andWrite commands that are executed as part of the ECC scrub requests are also executed without auto-precharge. If false, the bank remains open until there is a need to close it (to open a different page, or for page timeout or refresh timeout) - also known as open page policy. The open page policy can be overridden by setting the per-command- autopre bit on the HIF interface (hif_cmd_autopre). The pageclose feature provids a midway between Open and Close page policies. FOR PERFORMANCE ONLY. **Value After Reset**: 0x1 **Exists**: Always
1 PREFER_WRITE	If set then the bank selector prefers writes over reads. FOR DEBUG ONLY. **Value After Reset**: 0x0 **Exists**: Always
0 FORCE_LOW_PRI_N	Active low signal.When asserted ('0'), all incoming transactions are forced to low priority. This implies that all High Priority Read (HPR) and Variable Priority Read commands (VPR) will be treated as Low Priority Read (LPR) commands. On the write side, all Variable PriorityWrite (VPW) commands will be treated as Normal PriorityWrite (NPW) commands. Forcing the incoming transactions to low priority implicitly turns off Bypass path for read commands. FOR PERFORMANCE ONLY. **Value After Reset**: 0x1 **Exists**: Always

9.2.5.2.53 Scheduler Control Register 1 (DDRC_SCHED1)

Address: 307A_0000h base + 254h offset = 307A_0254h

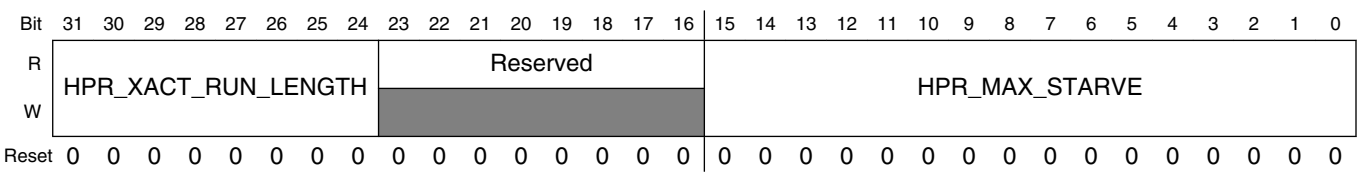


DDRC_SCHED1 field descriptions

Field	Description
31–8 Reserved	This field is reserved. Reserved for future use
PAGECLOSE_ TIMER	This field works in conjunction with SCHED.pageclose. It only has meaning if SCHED.pageclose == 1. If SCHED.pageclose == 1 and pageclose_timer == 0, then an auto-precharge may be scheduled for last read or write command in the CAM with a bank and page hit. Note, sometimes an explicit precharge is scheduled instead of the auto-precharge. See SCHED.pageclose for details of when this may happen. If SCHED.pageclose == 1 and pageclose_timer > 0, then an auto-precharge is not scheduled for last read or write command in the CAM with a bank and page hit. Instead, a timer is started, with pageclose_timer as the initial value. There is a timer on a per bank basis. The timer decrements unless the next read or write in the CAM to a bank is a page hit. It gets reset to pageclose_timer value if the next read or write in the CAM to a bank is a page hit. Once the timer has reached zero, an explcit precharge will be attempted to be scheduled. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.54 High Priority Read CAM Register 1 (DDRC_PERFHPR1)

Address: 307A_0000h base + 25Ch offset = 307A_025Ch



DDRC_PERFHPR1 field descriptions

Field	Description
31–24 HPR_XACT_ RUN_LENGTH	Number of transactions that are serviced once the HPR queue goes critical is the smaller of: • This number • Number of transactions available. Unit: Transaction FOR PERFORMANCE ONLY. **Value After Reset**: 0xf

Table continues on the next page...

DDRC_PERFHPR1 field descriptions (continued)

Field	Description
	Exists: Always
23–16 Reserved	This field is reserved.
HPR_MAX_ STARVE	Number of clocks that the HPR queue can be starved before it goes critical. The minimum valid functional value for this register is 0x1. Programming it to 0x0 will disable the starvation functionality; during normal operation, this function should not be disabled as it will cause excessive latencies. Unit: Clock cycles. FOR PERFORMANCE ONLY. **Value After Reset**: 0x1 **Exists**: Always

9.2.5.2.55 Low Priority Read CAM Register 1 (DDRC_PERFLPR1)

Address: 307A_0000h base + 264h offset = 307A_0264h

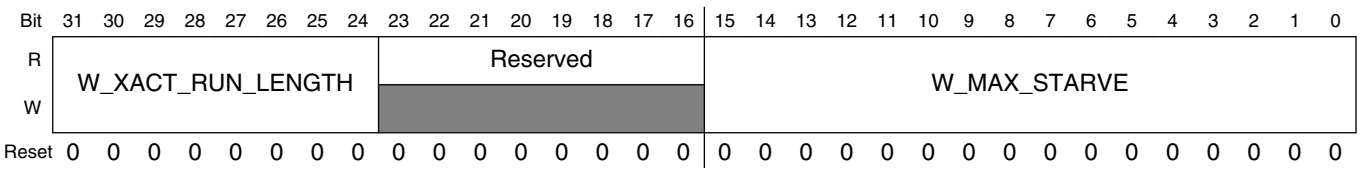
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LPR_XACT_RUN_LENGTH								Reserved								LPR_MAX_STARVE															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_PERFLPR1 field descriptions

Field	Description
31–24 LPR_XACT_ RUN_LENGTH	Number of transactions that are serviced once the LPR queue goes critical is the smaller of: 1. This number 2. Number of transactions available Unit: Transaction. FOR PERFORMANCE ONLY. **Value After Reset**: 0xf **Exists**: Always
23–16 Reserved	This field is reserved. Reserved for future use
LPR_MAX_ STARVE	Number of clocks that the LPR queue can be starved before it goes critical. The minimum valid functional value for this register is 0x1. Programming it to 0x0 will disable the starvation functionality; during normal operation, this function should not be disabled as it will cause excessive latencies. Unit: Clock cycles. FOR PERFORMANCE ONLY. **Value After Reset**: 0x7f **Exists**: Always

9.2.5.2.56 Write CAM Register 1 (DDRC_PERFWR1)

Address: 307A_0000h base + 26Ch offset = 307A_026Ch



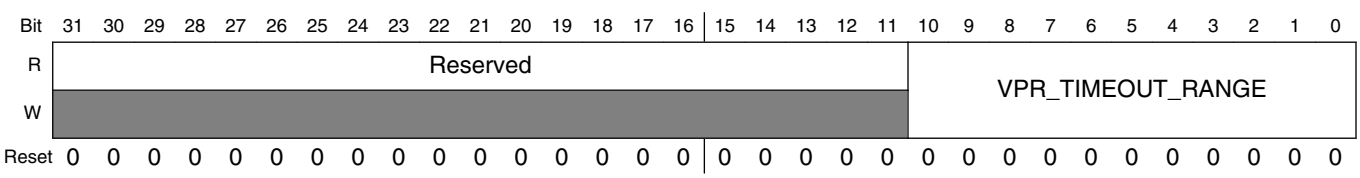
DDRC_PERFWR1 field descriptions

Field	Description
31–24 W_XACT_RUN_LENGTH	Number of transactions that are serviced once the WR queue goes critical is the smaller of: 1. This number 2. Number of transactions available Unit: Transaction. FOR PERFORMANCE ONLY. **Value After Reset**: 0xf **Exists**: Always
23–16 Reserved	This field is reserved. Reserved for future use
W_MAX_STARVE	Number of clocks that the WR queue can be starved before it goes critical. The minimum valid functional value for this register is 0x1. Programming it to 0x0 will disable the starvation functionality; during normal operation, this function should not be disabled as it will cause excessive latencies. Unit: Clock cycles. FOR PERFORMANCE ONLY. **Value After Reset**: 0x7f **Exists**: Always

9.2.5.2.57 Variable Priority Read CAM Register 1 (DDRC_PERFVPR1)

Exists: DDRC_VPR_EN==1

Address: 307A_0000h base + 274h offset = 307A_0274h



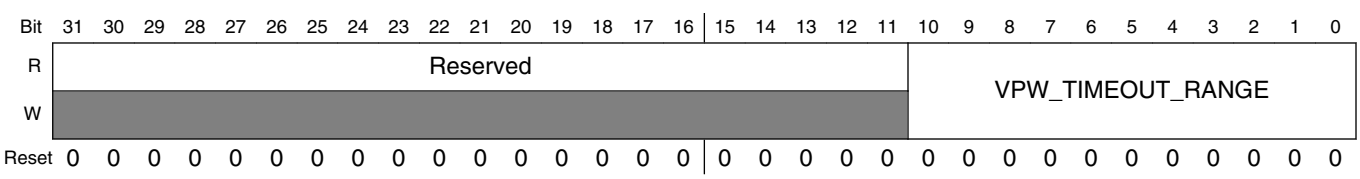
DDRC_PERFVPR1 field descriptions

Field	Description
31–11 Reserved	This field is reserved. Reserved for future use
VPR_TIMEOUT_RANGE	Indicates the range of the timeout value that is used for grouping the expired VPR commands in the CAM in DDRC. For example, if the register value is set to 0xF, then the priorities of all the VPR commands whose timeout counters are 15 or below will be considered as expired-VPR commands when the timeout value of any of the VPR commands reach 0. The expired-VPR commands, when present, are given higher priority than HPR commands. The VPR commands are expected to consist of largely page hit traffic and by grouping them together the bus utilization is expected to increase. This register applies to transactions inside the DDRC only. The Max value for this register is 0x7FF and the Min value is 0x0. When programmed to the Max value of 0x7FF, all the VPR commands that come in to DDRC will time-out right-away and will be considered as expired-VPR. When programmed to the Min value of 0x0, the timer of each command would have to reach a value of 0 before it will be considered as expired-VPR. Unit: Clock cycles. FOR PERFORMANCE ONLY. **Value After Reset**: 0x0 **Exists**: DDRC_VPR_EN == 1

9.2.5.2.58 Variable Priority Write CAM Register 1 (DDRC_PERFVPW1)

Exists: DDRC_VPW_EN==1

Address: 307A_0000h base + 278h offset = 307A_0278h



DDRC_PERFVPW1 field descriptions

Field	Description
31–11 Reserved	This field is reserved. Reserved for future use
VPW_TIMEOUT_RANGE	Indicates the range of the timeout value that is used for grouping the expired VPW commands in the CAM in DDRC. For example, if the register value is set to 0xF, then the priorities of all the VPW commands whose timeout counters are 15 or below will be considered as expired-VPW commands when the timeout value of any of the VPW commands reach 0. The expired-VPW commands, when present, are given higher priority than normalWrite commands. The VPW commands are expected to consist of largely page hit traffic and by grouping them together the bus utilization is expected to increase. This register applies to transactions inside the DDRC only. The Max value for this register is 0x7FF and the Min value is 0x0.

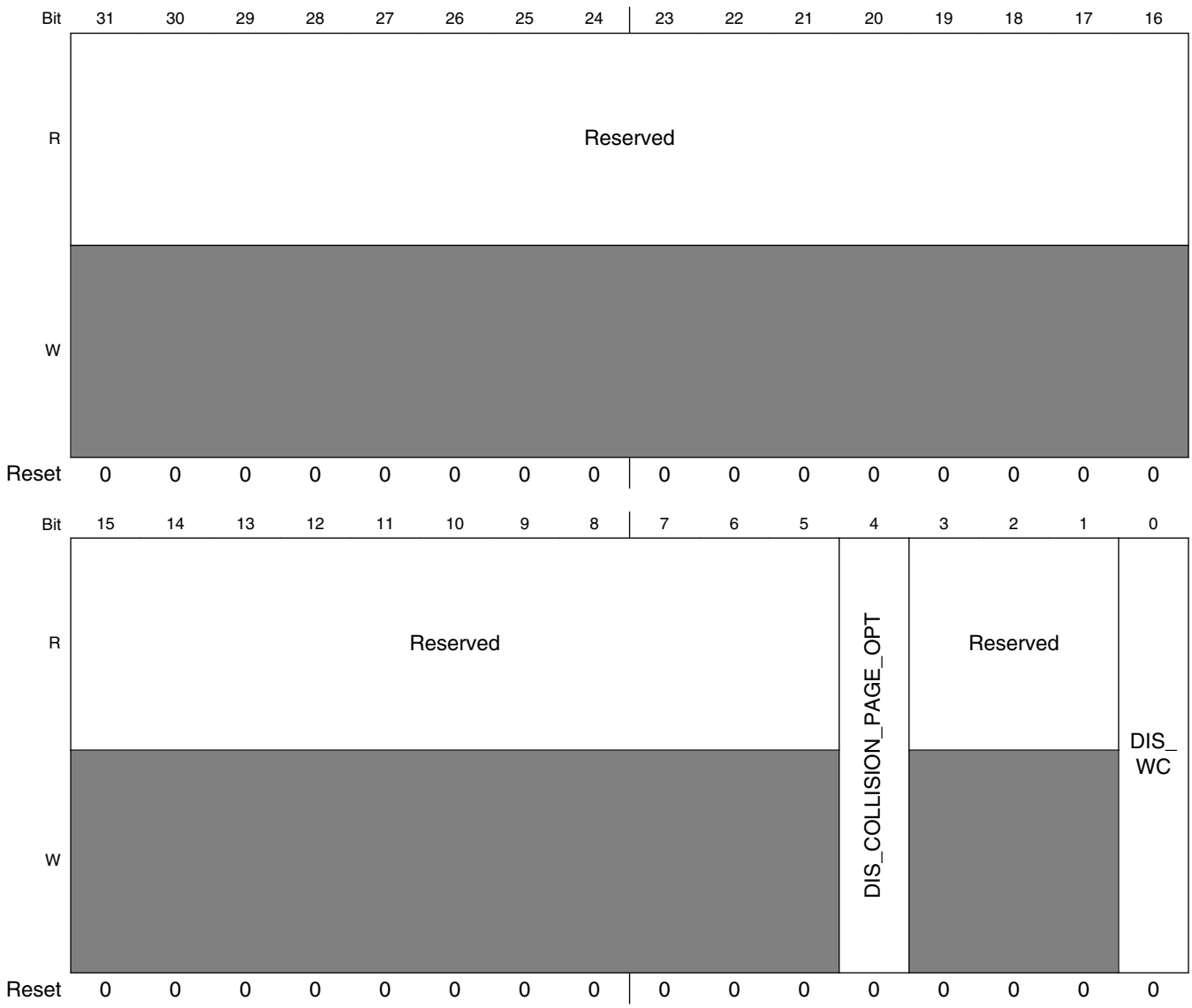
Table continues on the next page...

DDRC_PERFVPW1 field descriptions (continued)

Field	Description
	When programmed to the Max value of 0x7FF, all the VPW commands that come in to DDRC will time-out right-away and will be considered as expired-VPW. When programmed to the Min value of 0x0, the timer of each command would have to reach a value of 0 before it will be considered as expired-VPW. Unit: Clock cycles. FOR PERFORMANCE ONLY. **Value After Reset**: 0x0 **Exists**: DDRC_VPW_EN == 1

9.2.5.2.59 Debug Register 0 (DDRC_DBG0)

Address: 307A_0000h base + 300h offset = 307A_0300h

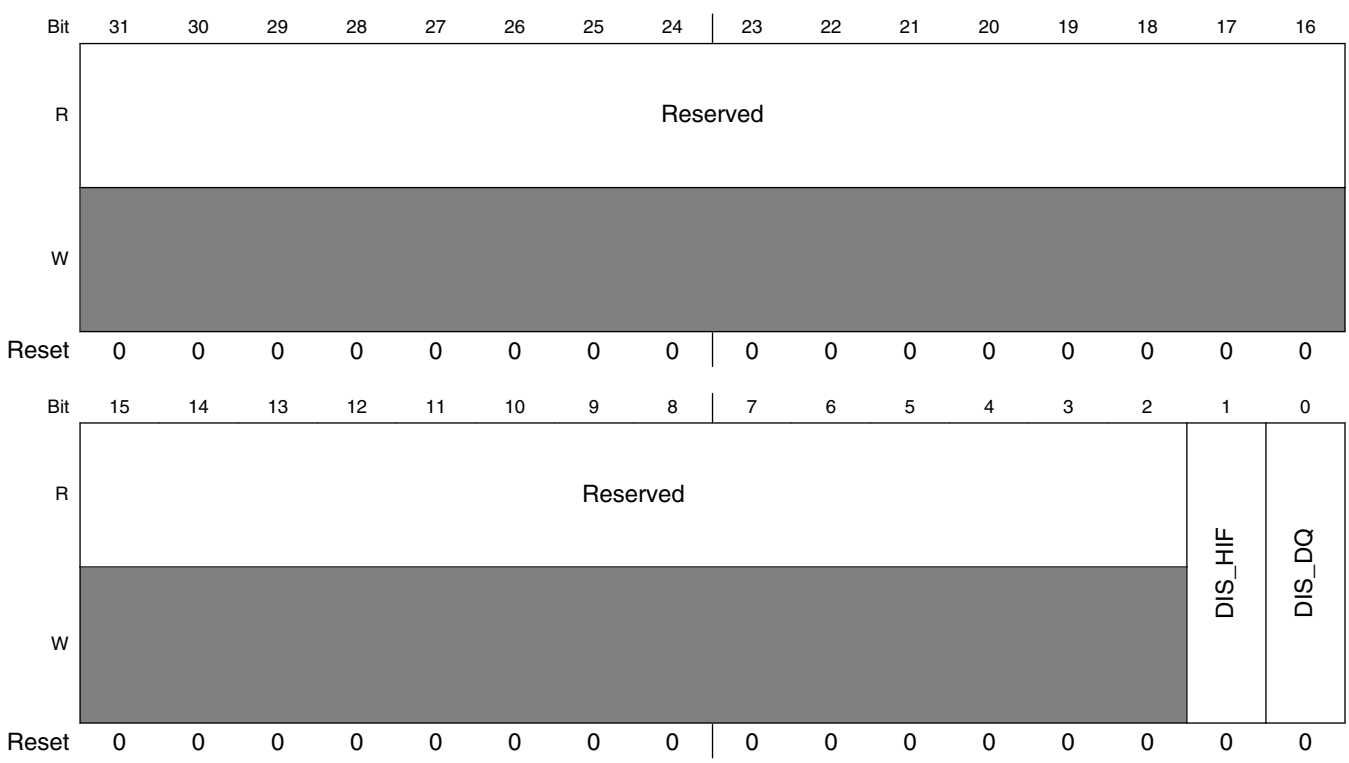


DDRC_DBG0 field descriptions

Field	Description
31–5 Reserved	This field is reserved. Reserved for future use
4 DIS_ COLLISION_ PAGE_OPT	When this is set to '0', auto-precharge is disabled for the flushed command in a collision case. Collision cases are write followed by read to same address, read followed by write to same address, or write followed by write to same address with DBG0.dis_wc bit = 1 (where same address comparisons exclude the two address bits representing critical word). FOR DEBUG ONLY. **Value After Reset**: 0x0 **Exists**: Always
3–1 Reserved	This field is reserved. Reserved for future use
0 DIS_WC	When the value is 1, disable write combine. FOR DEBUG ONLY **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.60 Debug Register 1 (DDRC_DBG1)

Address: 307A_0000h base + 304h offset = 307A_0304h

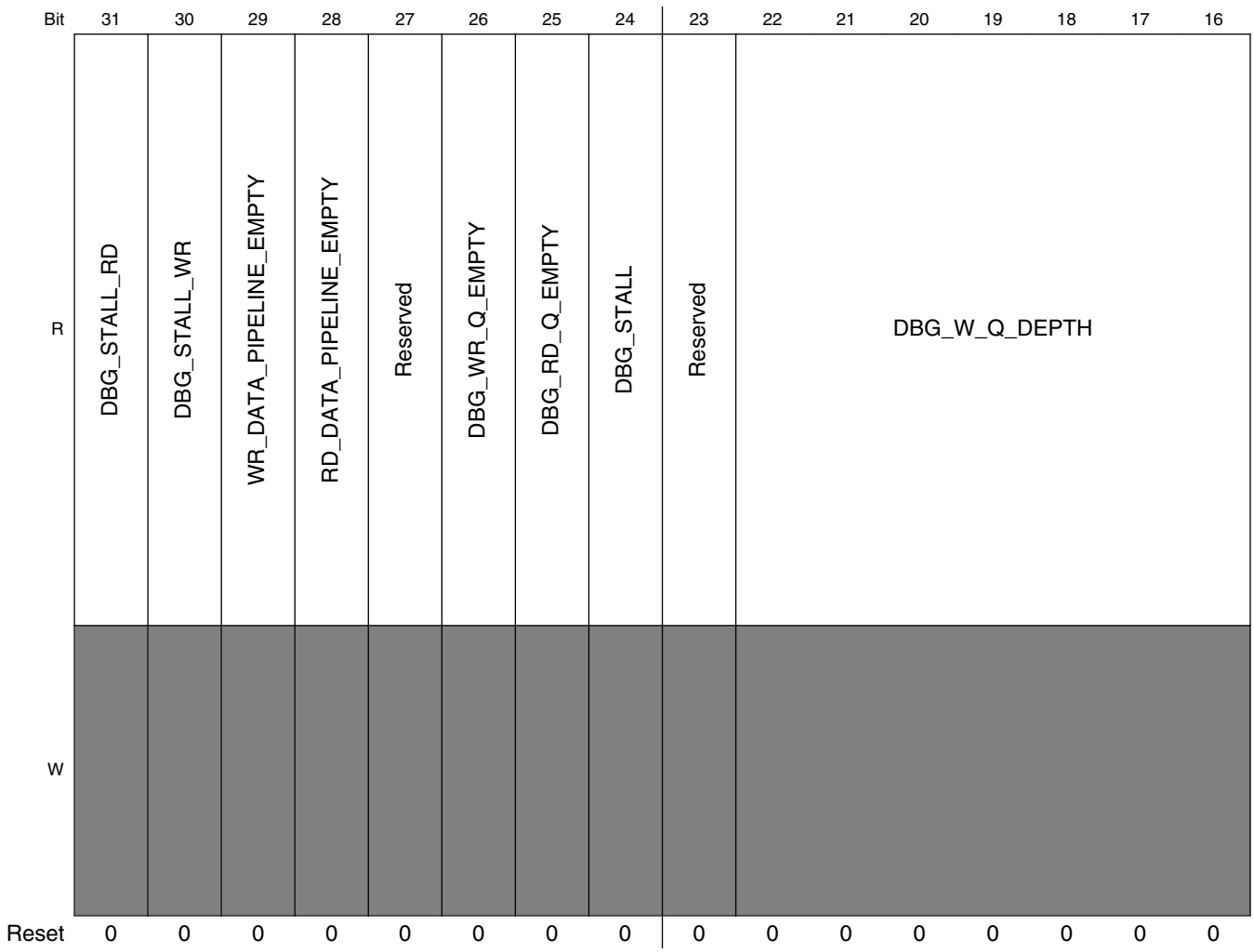


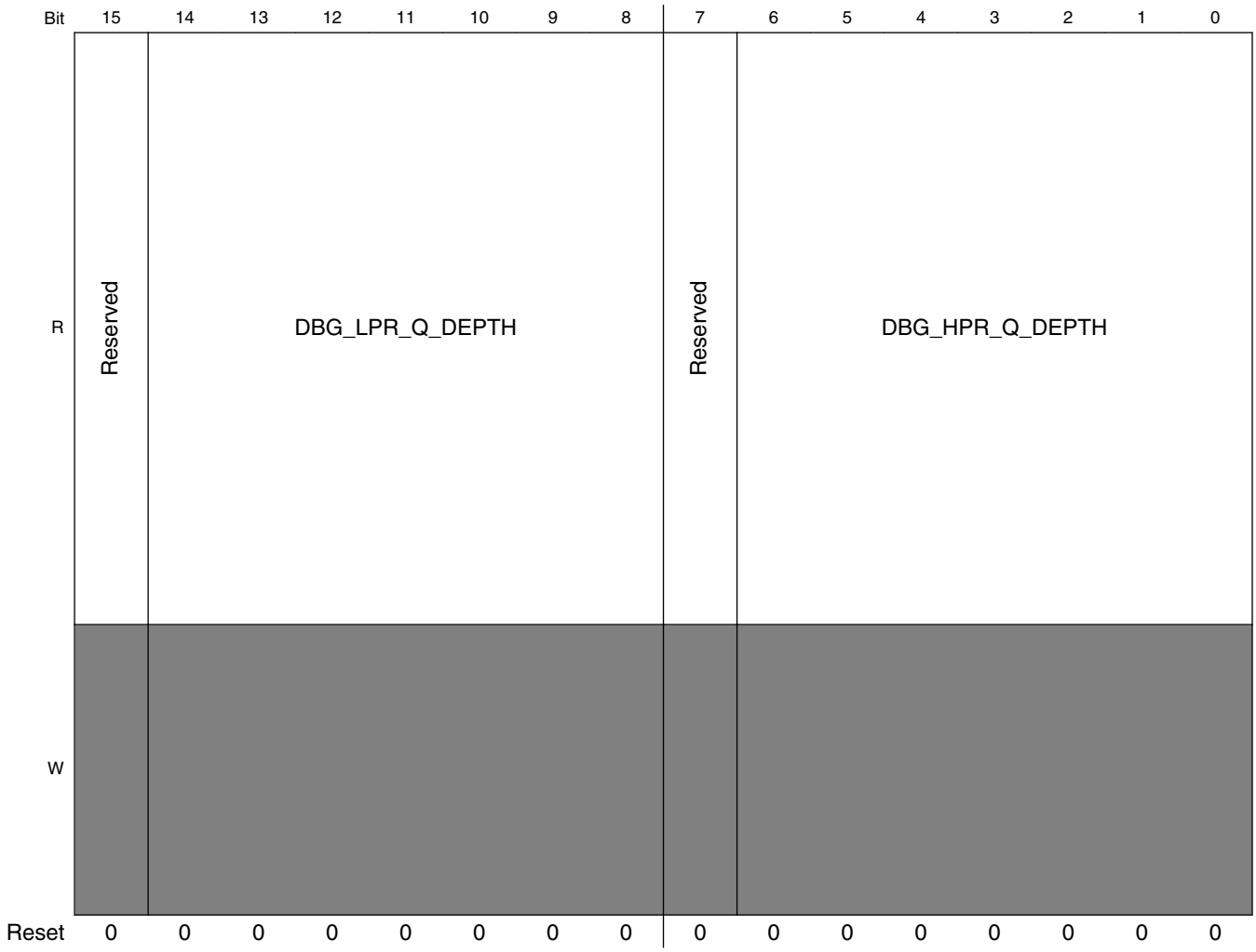
DDRC_DBG1 field descriptions

Field	Description
31–2 Reserved	This field is reserved. Reserved for future use
1 DIS_HIF	When 1, the DDRC asserts the HIF command signal hif_cmd_stall. The DDRC will ignore the hif_cmd_valid and all other associated request signals. This bit is intended to be switched on-the-fly. **Value After Reset**: 0x0 **Exists**: Always
0 DIS_DQ	When 1, DDRC will not de-queue any transactions from the CAM. Bypass is also disabled. All transactions are queued in the CAM. No reads or writes are issued to SDRAM as long as this is asserted. This bit may be used to prevent reads or writes being issued by the DDRC, which makes it safe to modify certain register fields associated with reads and writes (see User Guide for details). After setting this bit, it is strongly recommended to poll DBGCAM.wr_data_pipeline_empty and DBGCAM.rd_data_pipeline_empty, beforemaking changes to any registers which affect reads and writes. This will ensure that the relevant logic in the DDRC is idle. This bit is intended to be switched on-the-fly. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.61 CAM Debug Register (DDRC_DBGCAM)

Address: 307A_0000h base + 308h offset = 307A_0308h





DDRC_DBGCAM field descriptions

Field	Description
31 DBG_STALL_RD	Stall for Read channel FOR DEBUG ONLY **Value After Reset**: 0x0 **Exists**: DDRC_DUAL_PA_1 == 1
30 DBG_STALL_WR	Stall for Write channel FOR DEBUG ONLY **Value After Reset**: 0x0 **Exists**: DDRC_DUAL_PA_1 == 1
29 WR_DATA_PIPELINE_EMPTY	This bit indicates that the write data pipeline on the DFI interface is empty. This register is intended to be polled after setting DBG1.dis_dq, to ensure that all remaining commands / data have completed. **Value After Reset**: 0x0 **Exists**: Always

Table continues on the next page...

DDRC_DBGCAM field descriptions (continued)

Field	Description
28 RD_DATA_ PIPELINE_ EMPTY	This bit indicates that the read data pipeline on the DFI interface is empty. This register is intended to be polled after setting DBG1.dis_dq, to ensure that all remaining commands / data have completed. **Value After Reset**: 0x0 **Exists**: Always
27 Reserved	This field is reserved. Reserved for future use
26 DBG_WR_Q_ EMPTY	When 1, all theWrite command queues and Write data buffers inside DDRC are empty. This register is to be used for debug purpose. An example use-case scenario:When Controller enters Self- Refresh using the Low-Power entry sequence, Controller is expected to have executed all the commands in its queues and the write and read data drained. Hence this register should be 1 at that time. FOR DEBUG ONLY **Value After Reset**: 0x0 **Exists**: Always
25 DBG_RD_Q_ EMPTY	When 1, all the Read command queues and Read data buffers inside DDRC are empty. This register is to be used for debug purpose. An example use-case scenario:When Controller enters Self-Refresh using the Low-Power entry sequence, Controller is expected to have executed all the commands in its queues and the write and read data drained. Hence this register should be 1 at that time. FOR DEBUG ONLY **Value After Reset**: 0x0 **Exists**: Always
24 DBG_STALL	Stall FOR DEBUG ONLY **Value After Reset**: 0x0 **Exists**: Always
23 Reserved	This field is reserved. Reserved for future use
22–16 DBG_W_Q_ DEPTH	Write Queue Depth FOR DEBUG ONLY **Value After Reset**: 0x0 **Exists**: Always
15 Reserved	This field is reserved. Reserved for future use
14–8 DBG_LPR_Q_ DEPTH	Low Priority Read Queue Depth FOR DEBUG ONLY **Value After Reset**: 0x0 **Exists**: Always
7 Reserved	This field is reserved. Reserved for future use

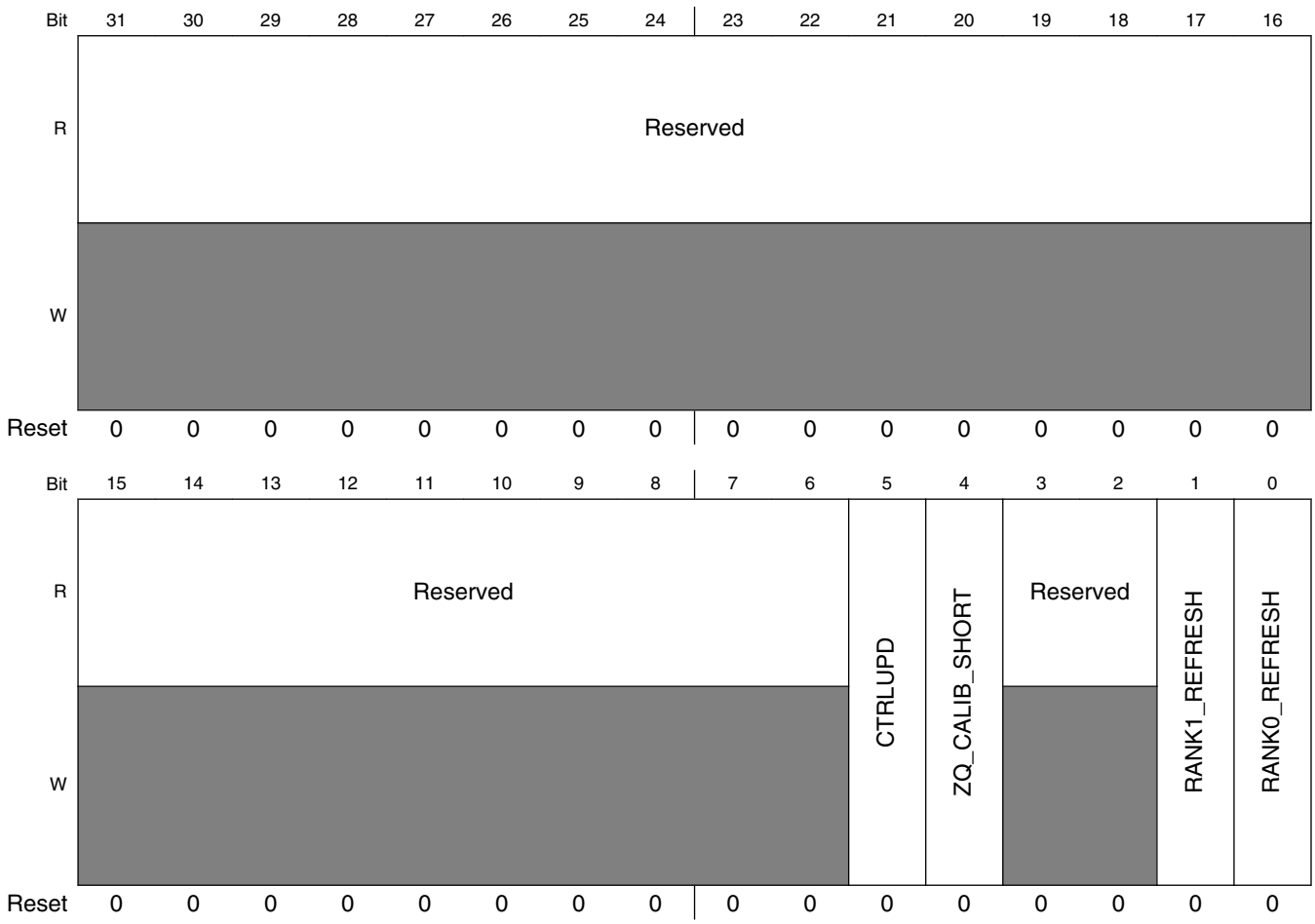
Table continues on the next page...

DDRC_DBGCAM field descriptions (continued)

Field	Description
DBG_HPR_Q_DEPTH	High Priority Read Queue Depth FOR DEBUG ONLY **Value After Reset**: 0x0 **Exists**: Always

9.2.5.2.62 Command Debug Register (DDRC_DBGCMD)

Address: 307A_0000h base + 30Ch offset = 307A_030Ch



DDRC_DBGCMD field descriptions

Field	Description
31–6 Reserved	This field is reserved. Reserved for future use
5 CTRLUPD	Setting this register bit to 1 indicates to the DDRC to issue a dfi_ctrlupd_req to the PHY. When this request is stored in the DDRC, the bit is automatically cleared. This operation must only be performed when DFIUPD0.dis_auto_ctrlupd = 1.

Table continues on the next page...

DDRC_DBGCMD field descriptions (continued)

Field	Description
	Value After Reset: 0x0 **Exists**: Always **Testable**: Read only
4 ZQ_CALIB_ SHORT	Setting this register bit to 1 indicates to the DDRC to issue a ZQCS (ZQ calibration short) command to the SDRAM. When this request is stored in the DDRC, the bit is automatically cleared. This operation can be performed only when ZQCTL0.dis_auto_zq = 1. It is recommended NOT to set this register bit if in Init operating mode. This register bit is ignored when in Self-Refresh and Deep power-down operating modes and Maximum Power Saving Mode. **Value After Reset**: 0x0 **Exists**: MEMC_DDR3_OR_4_OR_LPDDR2 == 1 **Testable**: Read only
3–2 Reserved	This field is reserved. Reserved for future use
1 RANK1_ REFRESH	Setting this register bit to 1 indicates to the DDRC to issue a refresh to rank 1.Writing to this bit causes DBGSTAT.rank1_refresh_busy to be set. When DBGSTAT.rank1_refresh_busy is cleared, the command has been stored in the DDRC. This operation can be performed only when RFSHCTL3.dis_auto_refresh = 1. It is recommended NOT to set this register bit if in Init or Deep power-down operating modes or Maximum Power Saving Mode. **Value After Reset**: 0x0 Exists:MEMC_NUM_RANKS > 1 **Testable**: Read only
0 RANK0_ REFRESH	Setting this register bit to 1 indicates to the DDRC to issue a refresh to rank 0.Writing to this bit causes DBGSTAT.rank0_refresh_busy to be set. When DBGSTAT.rank0_refresh_busy is cleared, the command has been stored in the DDRC. This operation can be performed only when RFSHCTL3.dis_auto_refresh = 1. It is recommended NOT to set this register bit if in Init or Deep power-down operating modes or Maximum Power Saving Mode. **Value After Reset**: 0x0 **Exists**: Always **Testable**: Read only

9.2.5.2.63 Status Debug Register (DDRC_DBGSTAT)

Address: 307A_0000h base + 310h offset = 307A_0310h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

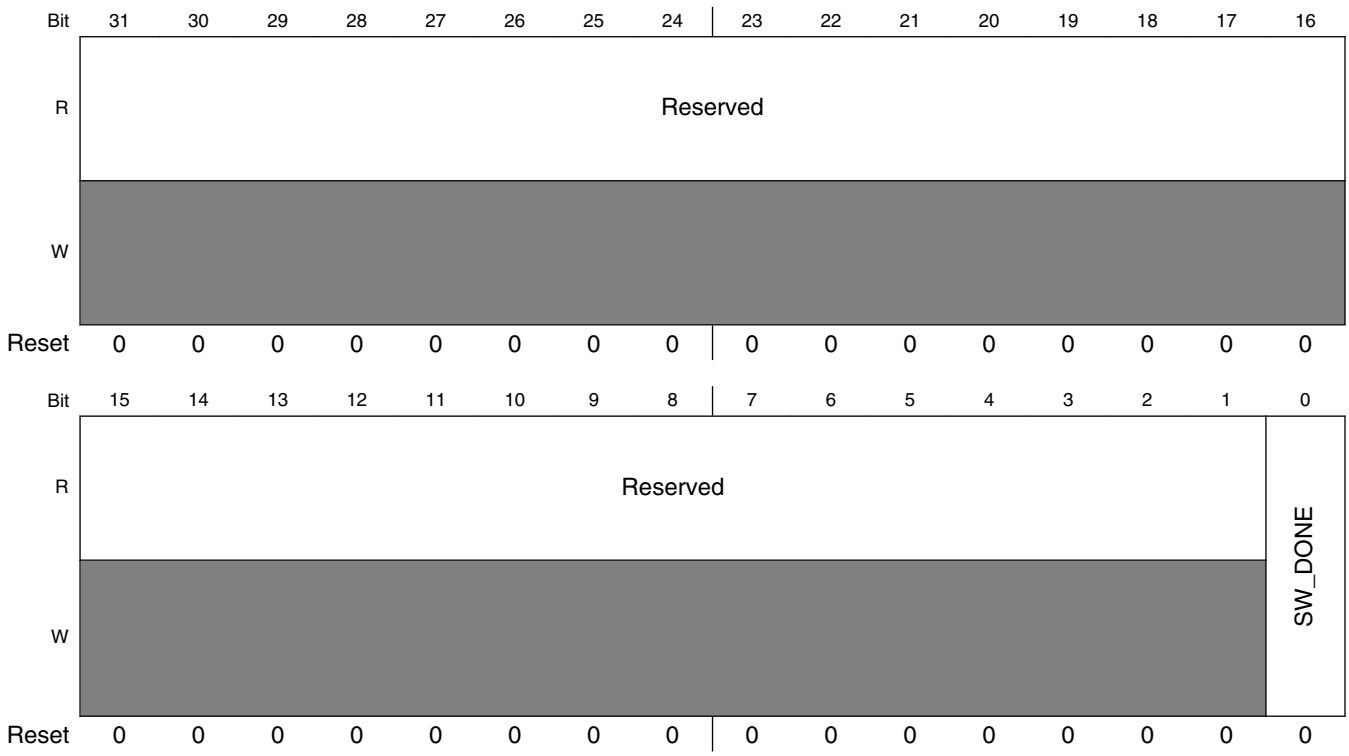
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved										CTRLUPD_BUSY	ZQ_CALIB_SHORT_BUSY	Reserved		RANK1_REFRESH_BUSY	RANK0_REFRESH_BUSY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_DBGSTAT field descriptions

Field	Description
31–6 Reserved	This field is reserved. Reserved for future use
5 CTRLUPD_ BUSY	SoC coremay initiate a ctrlupd operation only if this signal is low. This signal goes high in the clock after the DDRC accepts the ctrlupd request. It goes low when the ctrlupd operation is initiated in the DDRC. It is recommended not to perform ctrlupd operations when this signal is high. **Value After Reset**: 0x0 **Exists**: Always 0 Indicates that the SoC core can initiate a ctrlupd operation 1 Indicates that ctrlupd operation has not been initiated yet in the DDRC
4 ZQ_CALIB_ SHORT_BUSY	SoC core may initiate a ZQCS (ZQ calibration short) operation only if this signal is low. This signal goes high in the clock after the DDRC accepts the ZQCS request. It goes low when the ZQCS operation is initiated in the DDRC. It is recommended not to perform ZQCS operations when this signal is high. **Value After Reset**: 0x0 **Exists**: MEMC_DDR3_OR_LPDDR2 == 1
3–2 Reserved	This field is reserved. Reserved for future use
1 RANK1_ REFRESH_ BUSY	SoC core may initiate a rank1_refresh operation (refresh operation to rank 1) only if this signal is low. This signal goes high in the clock after DBGCMD.rank1_refresh is set to one. It goes low when the rank1_refresh operation is stored in the DDRC. It is recommended not to performrank1_refresh operationswhen this signal is high. **Value After Reset**: 0x0 **Exists**: MEMC_NUM_RANKS > 1 0 Indicates that the SoC core can initiate a rank1_refresh operation 1 Indicates that rank1_refresh operation has not been stored yet in the DDRC
0 RANK0_ REFRESH_ BUSY	SoC core may initiate a rank0_refresh operation (refresh operation to rank 0) only if this signal is low. This signal goes high in the clock after DBGCMD.rank0_refresh is set to one. It goes low when the rank0_refresh operation is stored in the DDRC. It is recommended not to performrank0_refresh operationswhen this signal is high. **Value After Reset**: 0x0 **Exists**: Always 0 Indicates that the SoC core can initiate a rank0_refresh operation 1 Indicates that rank0_refresh operation has not been stored yet in the DDRC

9.2.5.2.64 Software Register Programming Control Enable (DDRC_SWCTL)

Address: 307A_0000h base + 320h offset = 307A_0320h

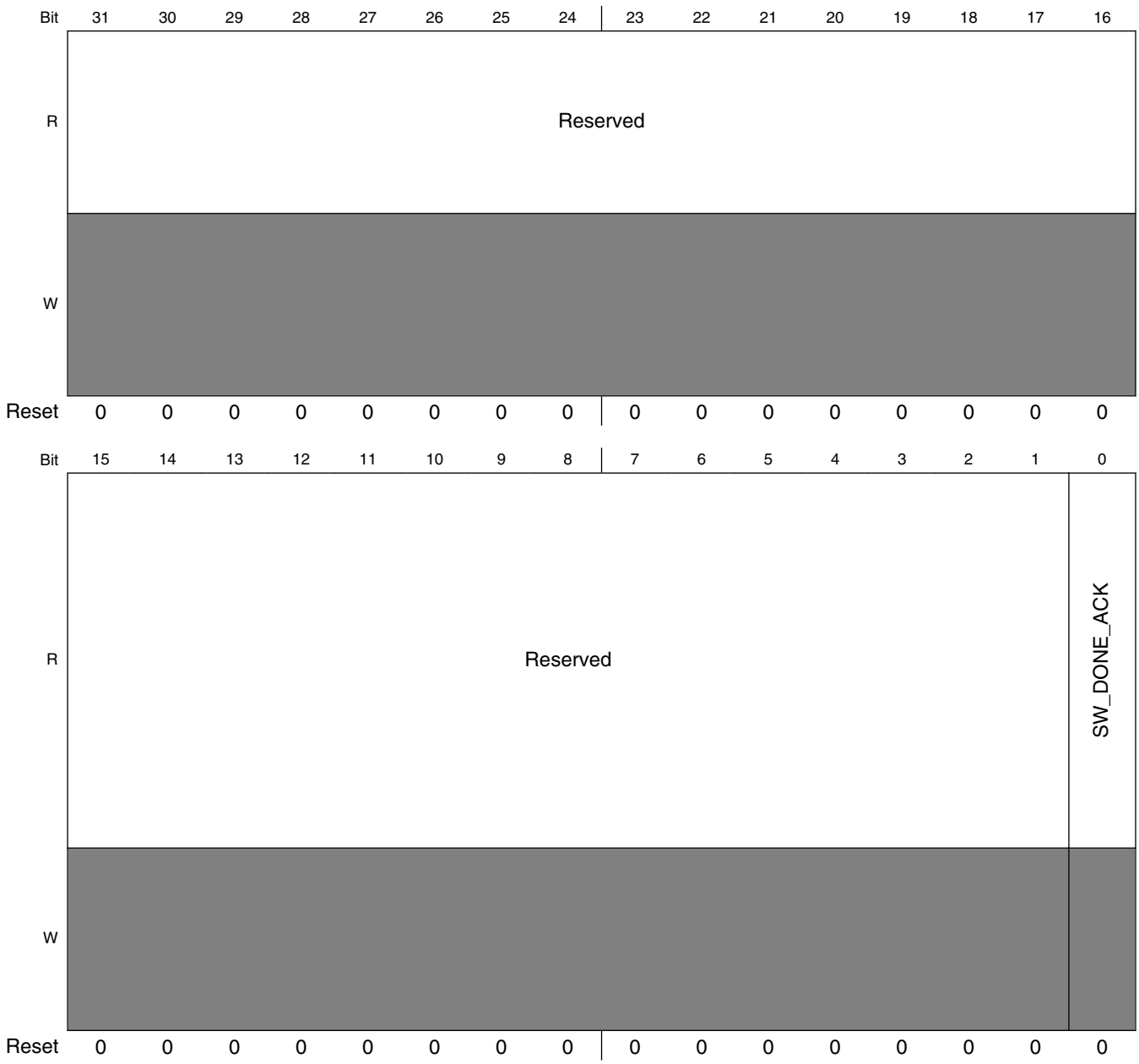


DDRC_SWCTL field descriptions

Field	Description
31–1 Reserved	This field is reserved. Reserved for future use
0 SW_DONE	Enable quasi dynamic register programming outside reset. Program register to 0 to enable quasi dynamic programming. Set back register to 1 once programming is done. **Value After Reset**: 0x1 **Exists**: Always

9.2.5.2.65 Software Register Programming Control Status (DDRC_SWSTAT)

Address: 307A_0000h base + 324h offset = 307A_0324h



DDRC_SWSTAT field descriptions

Field	Description
31–1 Reserved	This field is reserved. Reserved for future use
0 SW_DONE_ACK	Register programming done

Table continues on the next page...

DDRC_SWSTAT field descriptions (continued)

Field	Description
	This register is the echo of SWCTL.sw_done.Wait for sw_done value 1 to propagate to sw_done_ack at the end of the programming sequence to ensure that the correct registers values are propagated to the destination clock domains. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.3 DDRMC multi port registers

The following n and m is always 0 in the register.

DDRC_MP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
307A_03FC	Port Status Register (DDRC_MP_PSTAT)	32	R/W	0000_0000h	9.2.5.3.1/ 2269
307A_0400	Port Common Configuration Register (DDRC_MP_PCCFG)	32	R/W	0000_0000h	9.2.5.3.2/ 2270
307A_0404	Port n Configuration Read Register (DDRC_MP_PCFGR_0)	32	R/W	0000_0000h	9.2.5.3.3/ 2272
307A_0408	Port n Configuration Write Register (DDRC_MP_PCFGW_0)	32	R/W	0000_0000h	9.2.5.3.4/ 2274
307A_0410	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_00)	32	R/W	0000_0000h	9.2.5.3.5/ 2275
307A_0414	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_00)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0418	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_10)	32	R/W	0000_0000h	9.2.5.3.5/ 2275
307A_041C	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_10)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0420	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_20)	32	R/W	0000_0000h	9.2.5.3.5/ 2275
307A_0424	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_20)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0428	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_30)	32	R/W	0000_0000h	9.2.5.3.5/ 2275
307A_042C	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_30)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0430	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_40)	32	R/W	0000_0000h	9.2.5.3.5/ 2275

Table continues on the next page...

DDRC_MP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
307A_0434	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_40)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0438	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_50)	32	R/W	0000_0000h	9.2.5.3.5/ 2275
307A_043C	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_50)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0440	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_60)	32	R/W	0000_0000h	9.2.5.3.5/ 2275
307A_0444	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_60)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0448	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_70)	32	R/W	0000_0000h	9.2.5.3.5/ 2275
307A_044C	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_70)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0450	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_80)	32	R/W	0000_0000h	9.2.5.3.5/ 2275
307A_0454	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_80)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0458	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_90)	32	R/W	0000_0000h	9.2.5.3.5/ 2275
307A_045C	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_90)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0460	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_100)	32	R/W	0000_0000h	9.2.5.3.5/ 2275
307A_0464	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_100)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0468	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_110)	32	R/W	0000_0000h	9.2.5.3.5/ 2275
307A_046C	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_110)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0470	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_120)	32	R/W	0000_0000h	9.2.5.3.5/ 2275
307A_0474	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_120)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0478	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_130)	32	R/W	0000_0000h	9.2.5.3.5/ 2275
307A_047C	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_130)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0480	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_140)	32	R/W	0000_0000h	9.2.5.3.5/ 2275
307A_0484	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_140)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0488	Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_150)	32	R/W	0000_0000h	9.2.5.3.5/ 2275

Table continues on the next page...

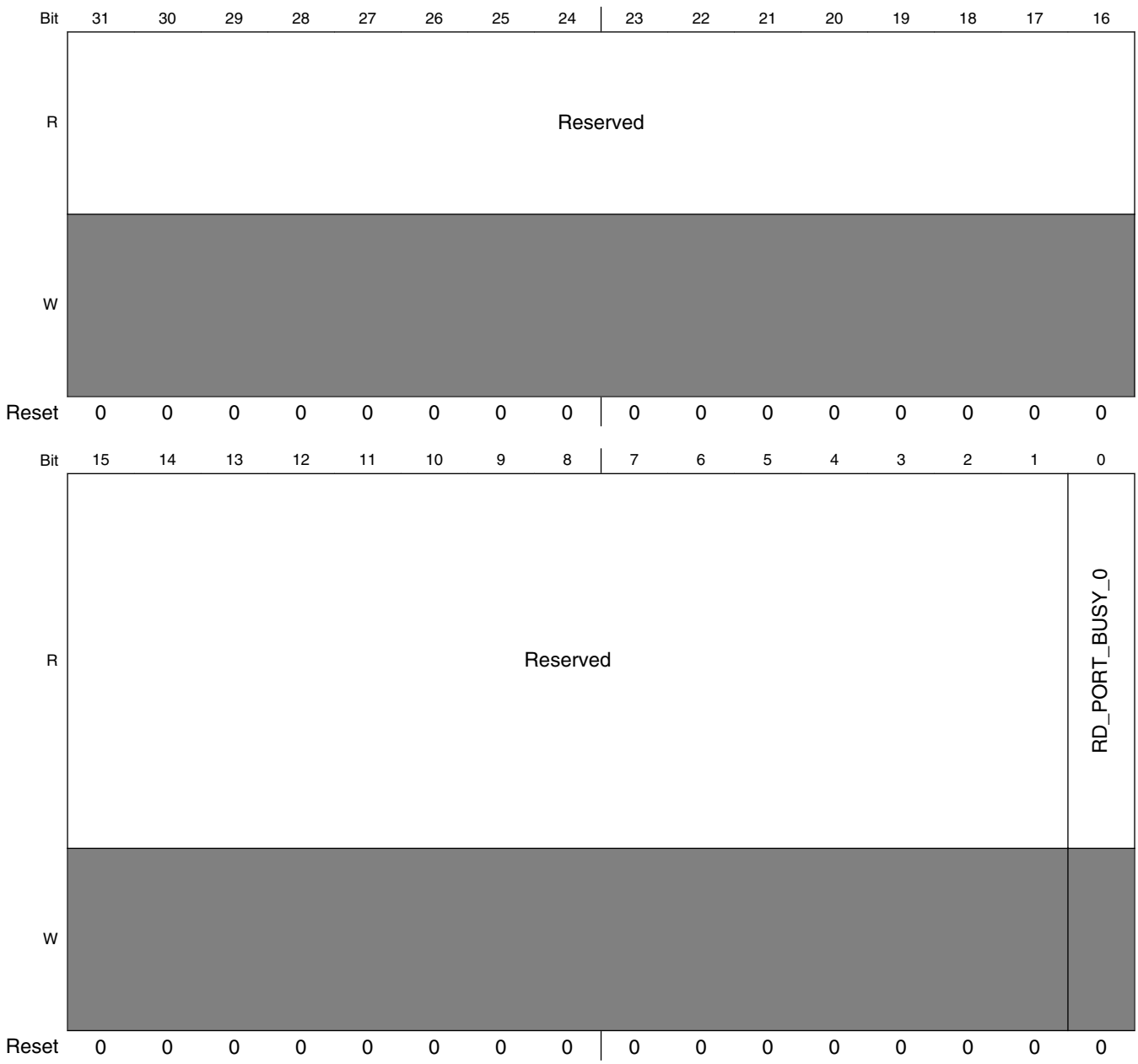
DDRC_MP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
307A_048C	Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_150)	32	R/W	0000_0000h	9.2.5.3.6/ 2276
307A_0490	Port n Control Register (DDRC_MP_PCTRL_0)	32	R/W	0000_0000h	9.2.5.3.7/ 2276
307A_0494	Port n Read QoS Configuration Register 0 (DDRC_MP_PCFGQOS0_0)	32	R/W	0000_0000h	9.2.5.3.8/ 2277
307A_0498	Port n Read QoS Configuration Register 1 (DDRC_MP_PCFGQOS1_0)	32	R/W	0000_0000h	9.2.5.3.9/ 2279
307A_049C	Port n Write QoS Configuration Register 0 (DDRC_MP_PCFGWQOS0_0)	32	R/W	0000_0000h	9.2.5.3.10/ 2280
307A_04A0	Port n Write QoS Configuration Register 1 (DDRC_MP_PCFGWQOS1_0)	32	R/W	0000_0000h	9.2.5.3.11/ 2281
307A_0F04	SAR Base Address Register n (DDRC_MP_SARBASE0)	32	R/W	0000_0000h	9.2.5.3.12/ 2281
307A_0F08	SAR Size Register n (DDRC_MP_SARSIZE0)	32	R/W	0000_0000h	9.2.5.3.13/ 2282
307A_0F0C	SAR Base Address Register n (DDRC_MP_SARBASE1)	32	R/W	0000_0000h	9.2.5.3.12/ 2281
307A_0F10	SAR Size Register n (DDRC_MP_SARSIZE1)	32	R/W	0000_0000h	9.2.5.3.13/ 2282
307A_0F14	SAR Base Address Register n (DDRC_MP_SARBASE2)	32	R/W	0000_0000h	9.2.5.3.12/ 2281
307A_0F18	SAR Size Register n (DDRC_MP_SARSIZE2)	32	R/W	0000_0000h	9.2.5.3.13/ 2282
307A_0F1C	SAR Base Address Register n (DDRC_MP_SARBASE3)	32	R/W	0000_0000h	9.2.5.3.12/ 2281
307A_0F20	SAR Size Register n (DDRC_MP_SARSIZE3)	32	R/W	0000_0000h	9.2.5.3.13/ 2282

9.2.5.3.1 Port Status Register (DDRC_MP_PSTAT)

Exists: DDRC_INCL_ARB==1

Address: 307A_0000h base + 3FCh offset = 307A_03FCh



DDRC_MP_PSTAT field descriptions

Field	Description
31–1 Reserved	This field is reserved. Reserved for future use

Table continues on the next page...

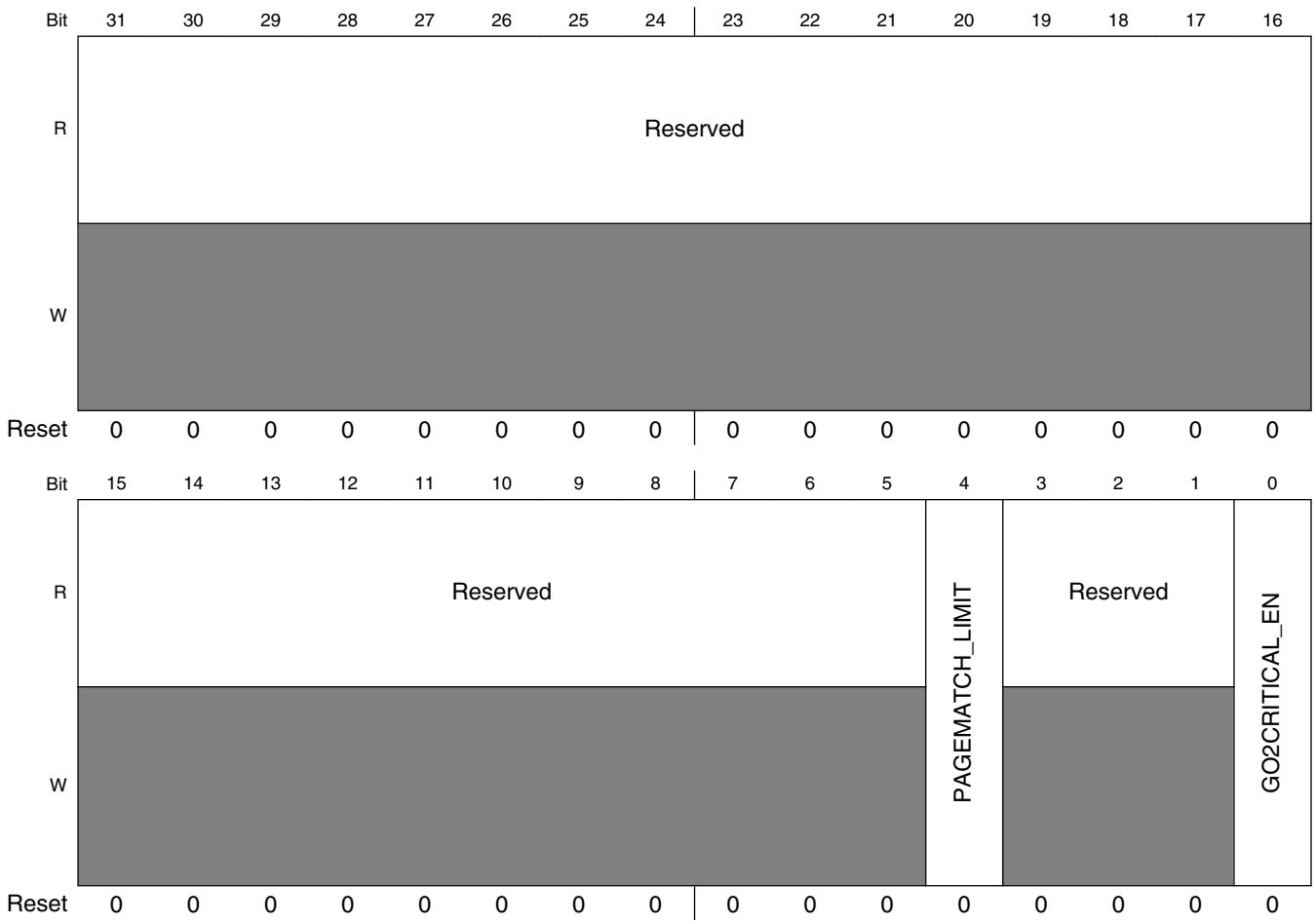
DDRC_MP_PSTAT field descriptions (continued)

Field	Description
0 RD_PORT_ BUSY_0	Indicates if there are outstanding reads for port 0. **Value After Reset**: 0x0 **Exists**: DDRC_PORT_0 == 1

9.2.5.3.2 Port Common Configuration Register (DDRC_MP_PCCFG)

Exists: DDRC_INCL_ARB==1

Address: 307A_0000h base + 400h offset = 307A_0400h



DDRC_MP_PCCFG field descriptions

Field	Description
31–5 Reserved	This field is reserved. Reserved for future use

Table continues on the next page...

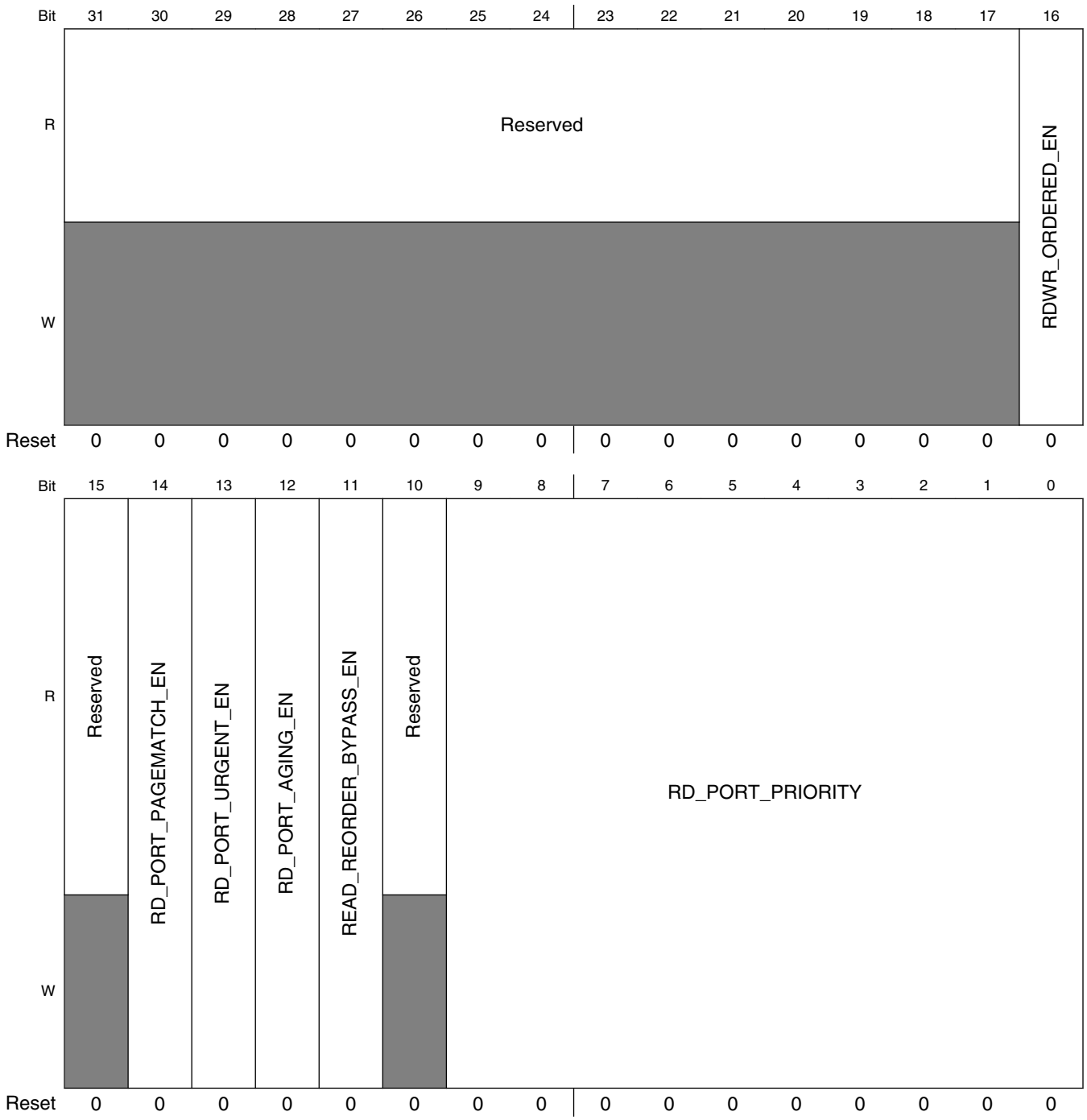
DDRC_MP_PCCFG field descriptions (continued)

Field	Description
4 PAGEMATCH_LIMIT	Page Match Four Limit If set to 1, limits the number of consecutive same page DDRC transactions that can be granted by the Port Arbiter to four when Page Match feature is enabled. If set to 0, there is no limit imposed on number of consecutive same page DDRC transactions. **Value After Reset**: 0x0 **Exists**: Always
3–1 Reserved	This field is reserved. Reserved for future use
0 GO2CRITICAL_EN	If set to 1 (enabled), sets hif_go2critical_wr and hif_go2critical_lpr / hif_go2critical_hpr signals going to DDRC based on urgent input (awurgent, arurgent) coming from AXI master. If set to 0 (disabled), hif_go2critical_wr and hif_go2critical_lpr / hif_go2critical_hpr signals at DDRC are driven to 1b'0. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.3.3 Port n Configuration Read Register (DDRC_MP_PCFGR_0)

Exists: DDRC_INCL_ARB==1 && DDRC_PORT_n==1

Address: 307A_0000h base + 404h offset = 307A_0404h



DDRC_MP_PCFGR_0 field descriptions

Field	Description
31–17 Reserved	This field is reserved. Reserved for future use
16 RDWR_ ORDERED_EN	Enable ordered read / writes. If set to 1, preserves the ordering between read transaction and write transaction issued to the same address, on a given port. In other words, the controller ensures that all same address read and write commands from the application port interface are transported to the DFI interface in the order of acceptance. This feature is useful in cases where software coherency is desired for masters issuing back-to-back read / write transactions without waiting for write/read responses. **NOTE:** This register has an effect only if necessary logic is instantiated via the DDRC_RDWR_ORDERED_0 parameter. **Value After Reset**: 0x0 **Exists**: DDRC_A_RDWR_ORDERED_0 == 1 && DDRC_A_AXI_0 == 1
15 Reserved	This field is reserved. Reserved for future use
14 RD_PORT_ PAGEMATCH_ EN	If set to 1, enables the Page Match feature. If enabled, once a requesting port is granted, the port is continued to be granted if the following immediate commands are to the same memory page (same bank and same row). See also related PCCFG.pagematch_limit register. **Value After Reset**:0x1 **Exists**: Always
13 RD_PORT_ URGENT_EN	If set to 1, enables the AXI urgent sideband signal (arurgent). When enabled and arurgent is asserted by the master, that port becomes the highest priority and hif_go2critical_lpr / hif_go2critical_hpr signal to DDRC is asserted if enabled in PCCFG.go2critical_en register. **NOTE:** Arurgent signal can be asserted anytime and as long as required which is independent of address handshaking (it is not associated with any particular command). **Value After Reset**: 0x0 **Exists**: Always
12 RD_PORT_ AGING_EN	If set to 1, enables aging function for the read channel of the port. **Value After Reset**: 0x0 **Exists**: Always
11 READ_ REORDER_ BYPASS_EN	If set to 1, read transactions with ID not covered by any of the virtual channel ID mapping registers are not reordered. **Value After Reset**: 0x0 **Exists**: DDRC_PORT_CHm_0 == 1
10 Reserved	This field is reserved. Reserved for future use
RD_PORT_ PRIORITY	Determines the initial load value of read aging counters. These counters will be parallel loaded after reset, or after each grant to the corresponding port. The aging counters down-count every clock cycle where the port is requesting but not granted. The higher significant 5-bit of the read aging counter sets the priority of the read channel of a given port. Port's priority will increase as the higher significant 5-bit of the counter starts to decrease. When the aging counter becomes 0, the corresponding port channel will have the highest priority level (timeout condition - Priority0). For multi port configurations, the aging counters cannot be used to set port priorities when external dynamic priority inputs (arqos) are enabled (timeout is still applicable). For single port configurations, the aging counters are only used when they timeout (become 0) to force read-write direction switching. In this

Table continues on the next page...

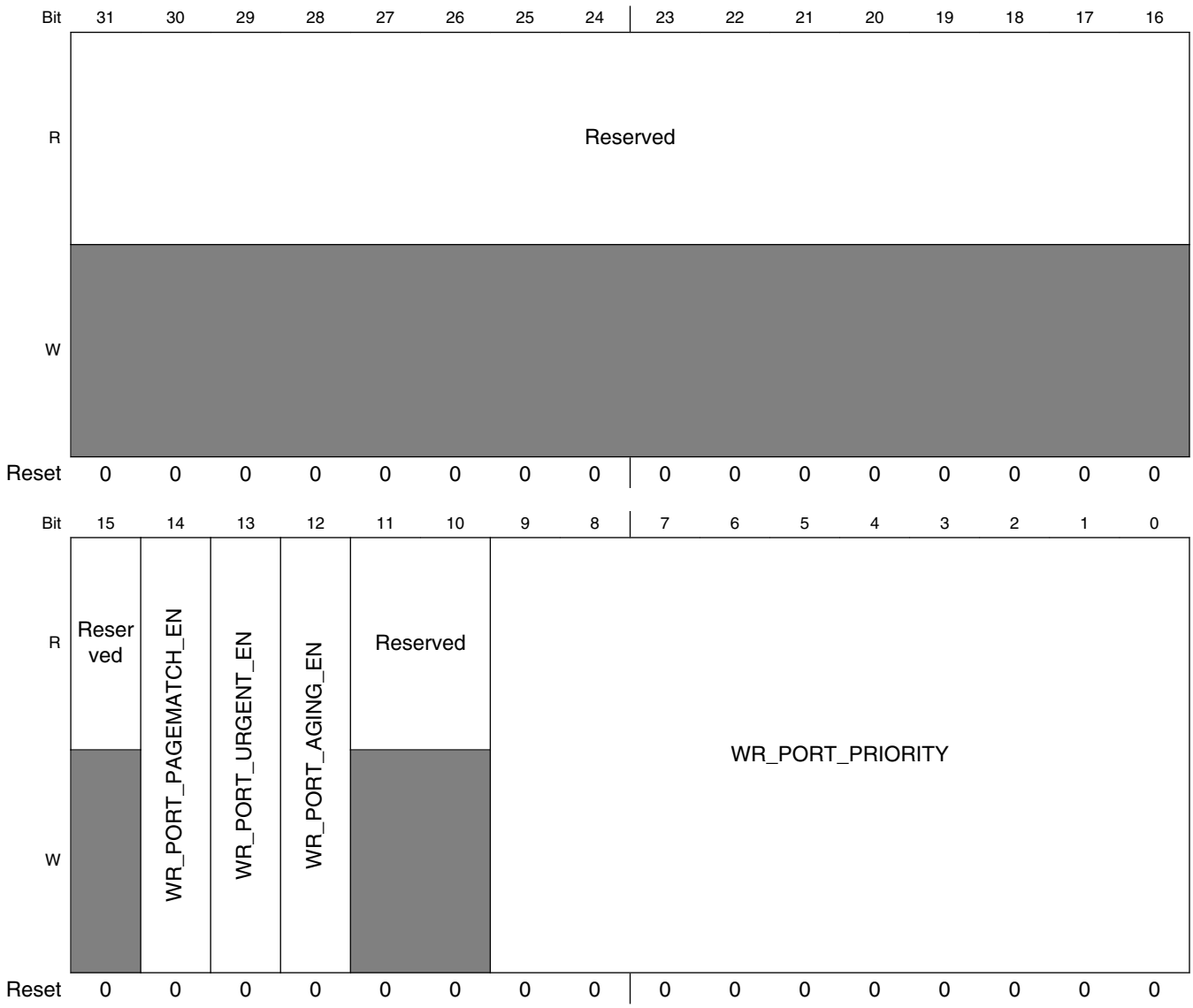
DDRC_MP_PCFGR_0 field descriptions (continued)

Field	Description
	case, external dynamic priority input, arqos (for reads only) can still be used to set the DDRC read priority (two priority levels: low priority read - LPR, high priority read - HPR) on a command by command basis. Note: The two LSBs of this register field are tied internally to 2'b00. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.3.4 Port n Configuration Write Register (DDRC_MP_PCFGW_0)

Exists: DDRC_INCL_ARB==1 && DDRC_PORT_0==1

Address: 307A_0000h base + 408h offset = 307A_0408h



DDRC_MP_PCFGW_0 field descriptions

Field	Description
31–15 Reserved	This field is reserved. Reserved for future use
14 WR_PORT_ PAGEMATCH_ EN	If set to 1, enables the Page Match feature. If enabled, once a requesting port is granted, the port is continued to be granted if the following immediate commands are to the same memory page (same bank and same row). See also related PCCFG.pagematch_limit register. **Value After Reset**: 0x1 **Exists**: Always
13 WR_PORT_ URGENT_EN	If set to 1, enables the AXI urgent sideband signal (awurgent). When enabled and awurgent is asserted by the master, that port becomes the highest priority and hif_go2critical_wr signal to DDRC is asserted if enabled in PCCFG.go2critical_enregister. **NOTE:** Awurgent signal can be asserted anytime and as long as required which is independent of address handshaking (it is not associated with any particular command). **Value After Reset**: 0x0 **Exists**: Always
12 WR_PORT_ AGING_EN	If set to 1, enables aging function for the write channel of the port. **Value After Reset**: 0x0 **Exists**: Always
11–10 Reserved	This field is reserved. Reserved for future use
WR_PORT_ PRIORITY	Determines the initial load value of write aging counters. These counters will be parallel loaded after reset, or after each grant to the corresponding port. The aging counters down-count every clock cycle where the port is requesting but not granted. The higher significant 5-bit of the write aging counter sets the initial priority of the write channel of a given port. Port's priority will increase as the higher significant 5-bit of the counter starts to decrease. When the aging counter becomes 0, the corresponding port channel will have the highest priority level. For multi port configurations, the aging counters cannot be used to set port priorities when external dynamic priority inputs (awqos) are enabled (timeout is still applicable). For single port configurations, the aging counters are only used when they timeout (become 0) to force read-write direction switching. **NOTE:** Two LSBs of this register field are tied internally to 2'b00. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.3.5 Port n Channel m Configuration ID Mask Register (DDRC_MP_PCFGIDMASKCH_*n*0)

M signifies the number of the virtual channel for the Port n where m = 0 to 15.

Exists: DDRC_PORT_CH m_0==1

Address: 307A_0000h base + 410h offset + (8d × i), where i=0d to 15d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ID_MASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_MP_PCFGIDMASKCH_*n*0 field descriptions

Field	Description
ID_MASK	Determines the mask used in the ID mapping function for virtual channel m. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.3.6 Port n Channel m Configuration ID Value Register (DDRC_MP_PCFGIDVALUECH_*n*0)

Exists: DDRC_PORT_CH m_0==1

Address: 307A_0000h base + 414h offset + (8d × i), where i=0d to 15d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ID_VALUE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_MP_PCFGIDVALUECH_*n*0 field descriptions

Field	Description
ID_VALUE	Determines the value used in the ID mapping function for virtual channel m. **Value After Reset**: 0x0 **Exists**: Always

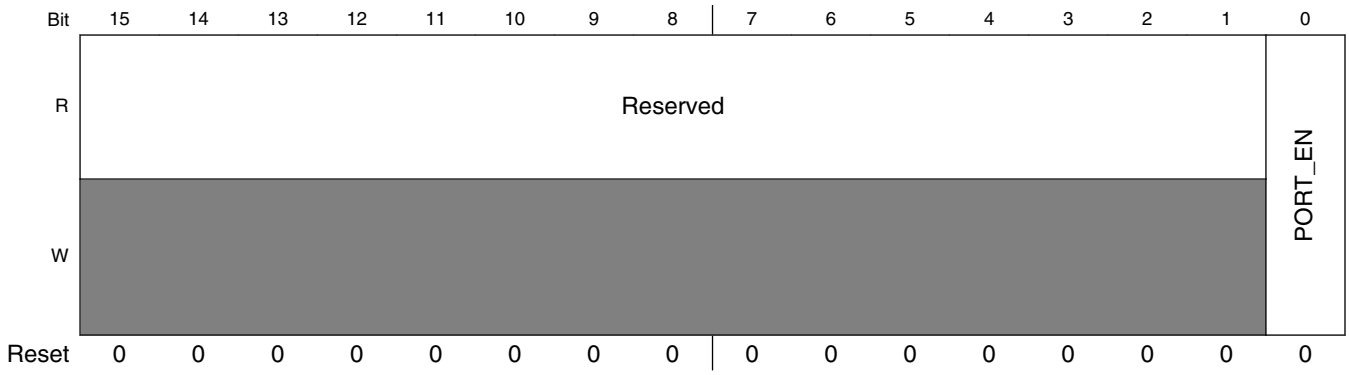
9.2.5.3.7 Port n Control Register (DDRC_MP_PCTRL_0)

n signifies the Port where n= 0 to 15.

Exists: DDRC_INCL_ARB==1 && DDRC_PORT_0==1

Address: 307A_0000h base + 490h offset = 307A_0490h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



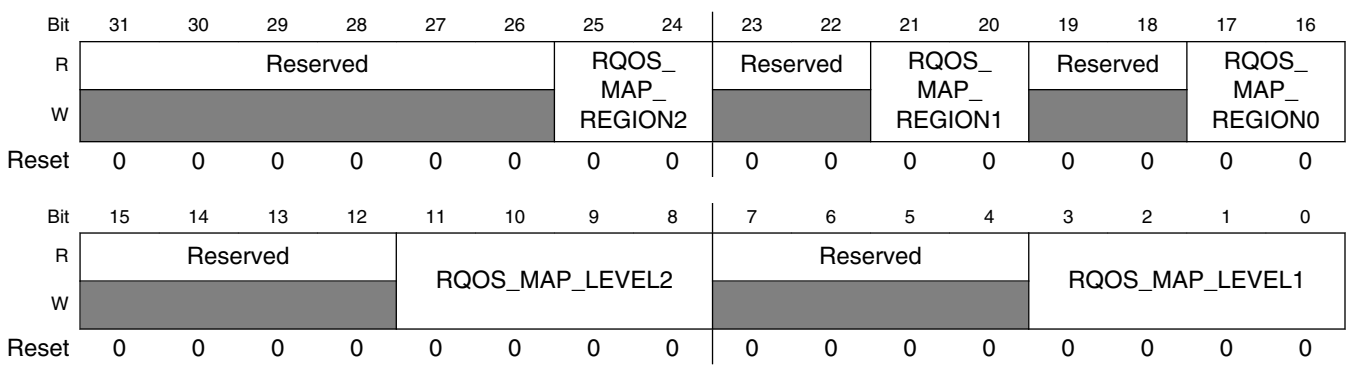
DDRC_MP_PCTRL_0 field descriptions

Field	Description
31–1 Reserved	This field is reserved. Reserved for future use
0 PORT_EN	Enables port n **Value After Reset**: DDRC_PORT_EN_RESET_VALUE **Exists**: Always

9.2.5.3.8 Port n Read QoS Configuration Register 0 (DDRC_MP_PCFGQOS0_0)

Exists: DDRC_A_AXI_0==1

Address: 307A_0000h base + 494h offset = 307A_0494h



DDRC_MP_PCFGQOS0_0 field descriptions

Field	Description
31–26 Reserved	This field is reserved. Reserved for future use

Table continues on the next page...

DDRC_MP_PCFGQOS0_0 field descriptions (continued)

Field	Description
25–24 RQOS_MAP_ REGION2	This bitfield indicates the traffic class of region2. For dual address queue configurations, region2 maps to the red address queue. Valid values are: • 1: VPR • 2: HPR only. When VPR support is disabled (DDRC_VPR_EN = 0) and traffic class of region2 is set to 1 (VPR), VPR traffic is aliased to LPR traffic. **Value After Reset**: 0x2 **Exists**: DDRC_A_USE2RAQ_0 == 1
23–22 Reserved	This field is reserved. Reserved for future use
21–20 RQOS_MAP_ REGION1	This bitfield indicates the traffic class of region 1. Valid values are: • 0 - LPR • 1 - VPR • 2 - HPR For dual address queue configurations, region1 maps to the blue address queue. In this case, valid values are: • 0: LPR • 1: VPR only When VPR support is disabled (DDRC_VPR_EN = 0) and traffic class of region 1 is set to 1 (VPR), VPR traffic is aliased to LPR traffic. **Value After Reset**: 0x0 **Exists**: Always
19–18 Reserved	This field is reserved. Reserved for future use
17–16 RQOS_MAP_ REGION0	This bitfield indicates the traffic class of region 0. Valid values are: • 0: LPR • 1: VPR • 2: HPR For dual address queue configurations, region 0 maps to the blue address queue. In this case, valid values are: • 0: LPR • 1: VPR only When VPR support is disabled (DDRC_VPR_EN = 0) and traffic class of region0 is set to 1 (VPR), VPR traffic is aliased to LPR traffic. **Value After Reset**: 0x0 **Exists**: Always
15–12 Reserved	This field is reserved. Reserved for future use

Table continues on the next page...

DDRC_MP_PCFGQOS0_0 field descriptions (continued)

Field	Description
11–8 RQOS_MAP_ LEVEL2	Separation level2 indicating the end of region1 mapping; start of region1 is (level1 + 1). Possible values for level2 are (level1 + 1) to 14 which corresponds to arqos. Region2 starts from (level2 + 1) up to 15. **NOTE:** For PA, arqos values are used directly as port priorities, where the higher the value corresponds to higher port priority. All of the map_level* registers must be set to distinct values. **Value After Reset**: 0xe **Exists**: DDRC_A_USE2RAQ_0 == 1
7–4 Reserved	This field is reserved. Reserved for future use
RQOS_MAP_ LEVEL1	Separation level1 indicating the end of region0 mapping; start of region0 is 0. Possible values for level1 are 0 to 13 (for dual RAQ) or 0 to 14 (for single RAQ) which corresponds to arqos. **NOTE:** For PA, arqos values are used directly as port priorities, where the higher the value corresponds to higher port priority. All of the map_level* registers must be set to distinct values. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.3.9 Port n Read QoS Configuration Register 1 (DDRC_MP_PCFGQOS1_0)

Exists: DDRC_A_AXI_n==1 && DDRC_XPI_VPR_n==1

Address: 307A_0000h base + 498h offset = 307A_0498h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved					RQOS_MAP_TIMEOUTR											Reserved					RQOS_MAP_TIMEOUTB										
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_MP_PCFGQOS1_0 field descriptions

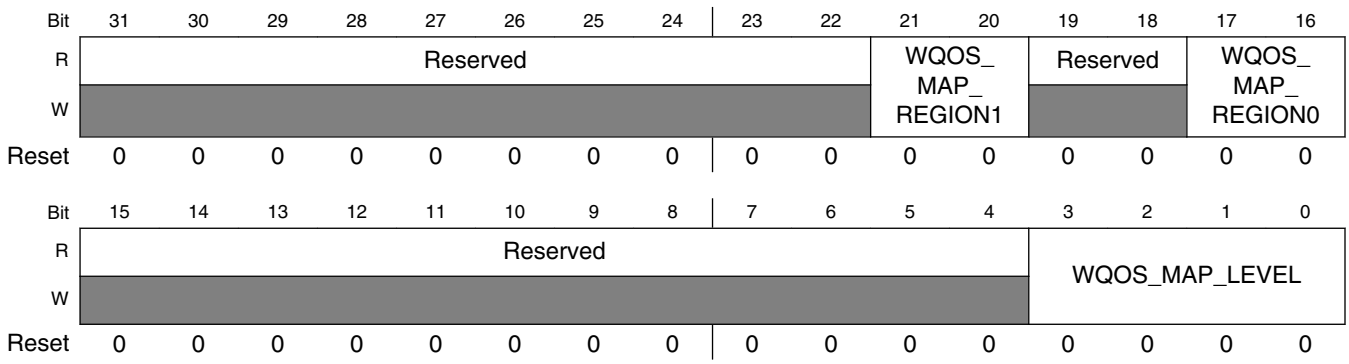
Field	Description
31–27 Reserved	This field is reserved. Reserved for future use
26–16 RQOS_MAP_ TIMEOUTR	Specifies the timeout value for transactions mapped to the red address queue. **Value After Reset**: 0x0 **Exists**: Always
15–11 Reserved	This field is reserved. Reserved for future use
RQOS_MAP_ TIMEOUTB	Specifies the timeout value for transactions mapped to the blue address queue. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.3.10 Port n Write QoS Configuration Register 0 (DDRC_MP_PCFGWQOS0_0)

n signifies the Port where n = 0 to 15.

Exists: DDRC_A_AXI_n==1 && DDRC_XPI_VPW_n==1

Address: 307A_0000h base + 49Ch offset = 307A_049Ch



DDRC_MP_PCFGWQOS0_0 field descriptions

Field	Description
31–22 Reserved	This field is reserved. Reserved for future use
21–20 WQOS_MAP_ REGION1	This bitfield indicates the traffic class of region 1. Valid values are: • 0: NPW • 1: VPW When VPW support is disabled (DDRC_VPW_EN = 0) and traffic class of region 1 is set to 1 (VPW), VPW traffic is aliased to LPW traffic. **Value After Reset**: 0x0 **Exists**: Always
19–18 Reserved	This field is reserved. Reserved for future use
17–16 WQOS_MAP_ REGION0	This bitfield indicates the traffic class of region 0. Valid values are: • 0: NPW • 1: VPW When VPW support is disabled (DDRC_VPW_EN = 0) and traffic class of region0 is set to 1 (VPW), VPW traffic is aliased to NPW traffic. **Value After Reset**: 0x0 **Exists**: Always
15–4 Reserved	This field is reserved. Reserved for future use
WQOS_MAP_ LEVEL	Separation level indicating the end of region0 mapping; start of region0 is 0. Possible values for level1 are 0 to 14 which corresponds to awqos.

Table continues on the next page...

DDRC_MP_PCFGWQOS0_0 field descriptions (continued)

Field	Description
	NOTE: For PA, awqos values are used directly as port priorities, where the higher the value corresponds to higher port priority. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.3.11 Port n Write QoS Configuration Register 1 (DDRC_MP_PCFGWQOS1_0)

Exists: DDRC_A_AXI_n==1 && DDRC_XPI_VPW_n==1

Address: 307A_0000h base + 4A0h offset = 307A_04A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																					WQOS_MAP_TIMEOUT										
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_MP_PCFGWQOS1_0 field descriptions

Field	Description
31–11 Reserved	This field is reserved. Reserved for future use
WQOS_MAP_TIMEOUT	Specifies the timeout value for write transactions. **Value After Reset**: 0x0 **Exists**: Always

9.2.5.3.12 SAR Base Address Register n (DDRC_MP_SARBASE*n*)

Where n= 0 to 3

Exits: DDRC_INCL_ARB==1 && DDRC_A_SAR_n==1

Address: 307A_0000h base + F04h offset + (8d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BASE_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRC_MP_SARBASE*n* field descriptions

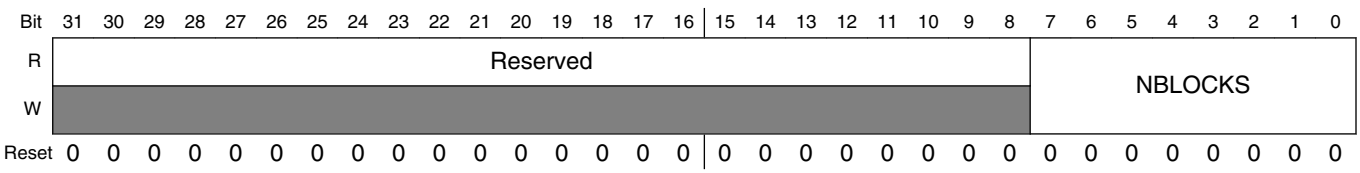
Field	Description
BASE_ADDR	Base address for address region n specified as awaddr[DDRC_A_ADDRW-1:x]and araddr[DDRC_A_ADDRW-1:x] where x is determined by the minimum block size parameter DDRC_SARMINSIZE: (x = log2 (block size)). **Value After Reset**: 0x0 **Exists**: Always

9.2.5.3.13 SAR Size Register n (DDRC_MP_SARSIZE*n*)

Where n= 0 to 3

Exists: DDRC_INCL_ARB==1 && DDRC_A_SAR_n==1

Address: 307A_0000h base + F08h offset + (8d × i), where i=0d to 3d



DDRC_MP_SARSIZE*n* field descriptions

Field	Description
31–8 Reserved	This field is reserved. Reserved for future use
NBLOCKS	Number of blocks for address region n. This register determines the total size of the region in multiples of minimum block size as specified by the hardware parameter DDRC_SARMINSIZE. The register value is encoded as number of blocks = nblocks + 1. For example, if register is programmed to 0, region will have 1 block. **Value After Reset**: 0x0 **Exists**: Always

9.3 DDR PHY (DDRP)

9.3.1 Overview

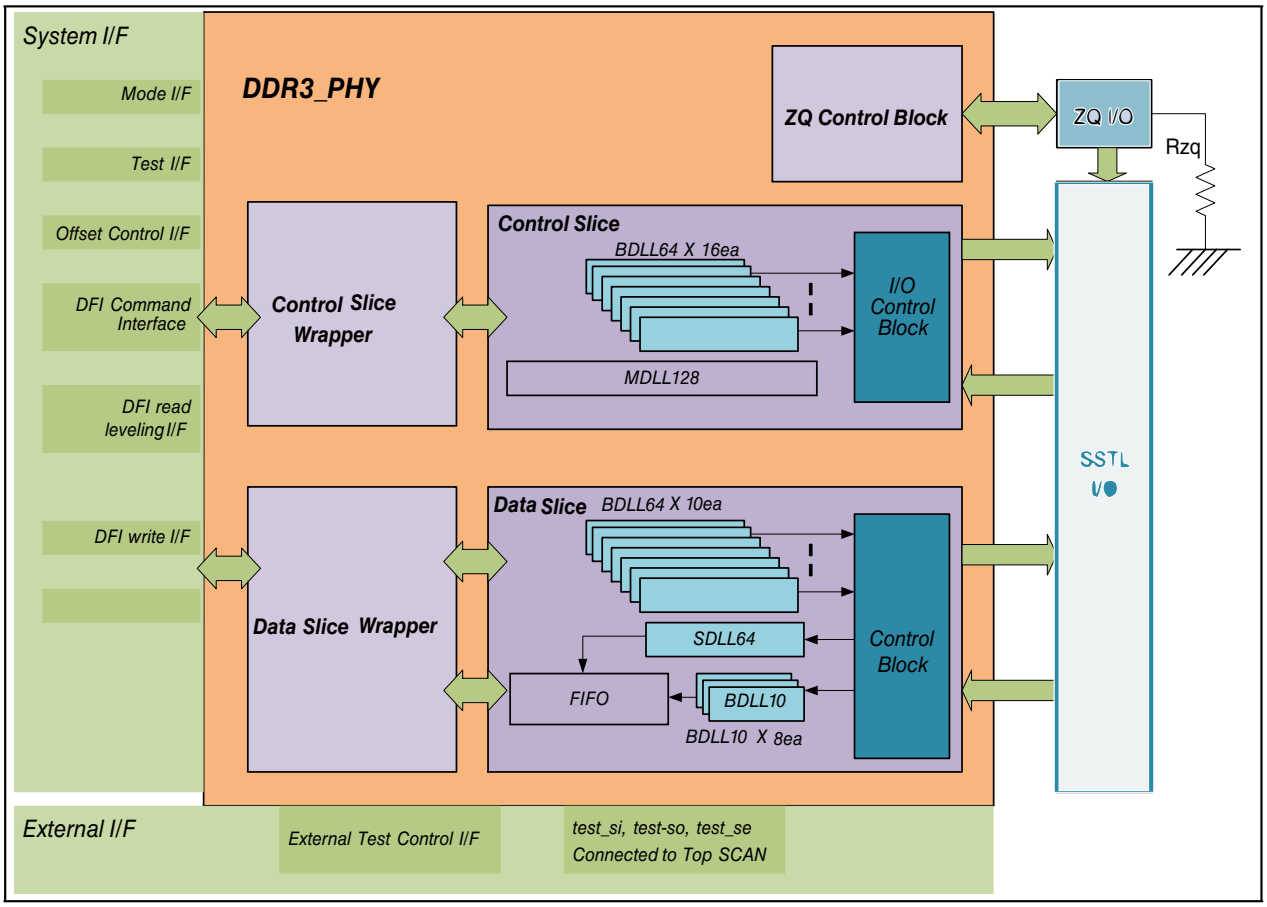


Figure 9-9. LPDDR3 HPHY

9.3.1.1 Features

The DDR3 PHY (V6R0) provides the following features:

- Fully supports DDR3, LPDDR2, LPDDR3 memory types.
- Fully digital DLL for 90° phase shift of strobe signal.
- FIFO (width:16-bit / depth:16 per 8-bit data slice) for programmable read timing.
- Provide feedback loop-back test scheme for at-speed data and control channel test.

NOTE

- RL should be greater than 4. If PHY is used with the dual-rank configurations, RL (Read Latency) and BL (Burst Length) should be used as the same value for those two ranks.
- ODT is not supported for LPDDR3.

9.3.2 Block diagram description

9.3.2.1 DLL and CONTROL I/F

9.3.2.1.1 DLL

DLL detects one clock period and generates delay line control signal. Delay line consists of 128 delay cells and is controlled by control logic. Delay line delays input clock and phase detector compares input clock and delayed clock. After phase comparison, phase detector detects whether delayed clock is lag or lead and generates INC / DEC signals to increase or decrease the delay amount of the delay line.

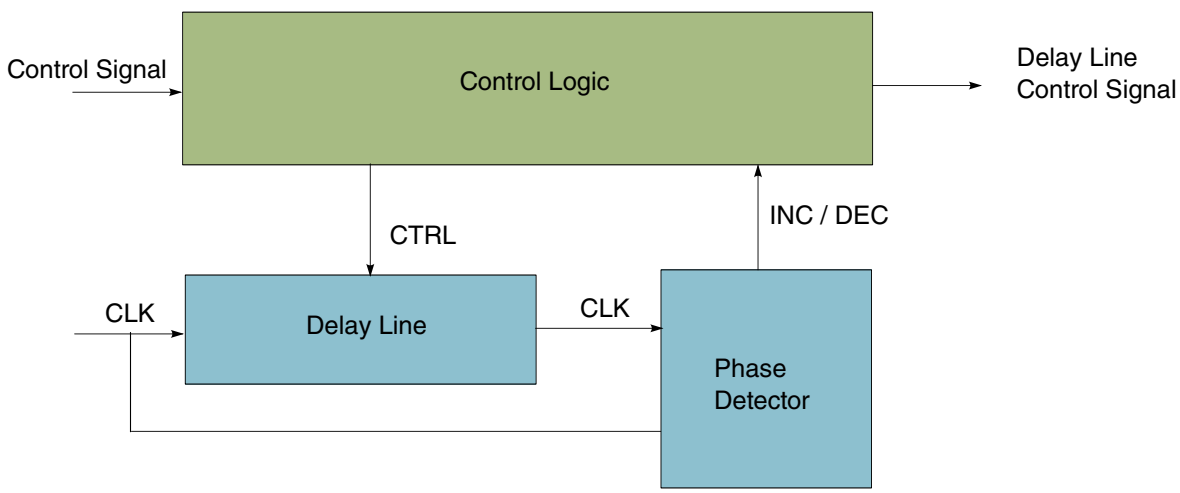


Figure 9-10. Block Diagram of DLL

9.3.2.1.2 Control

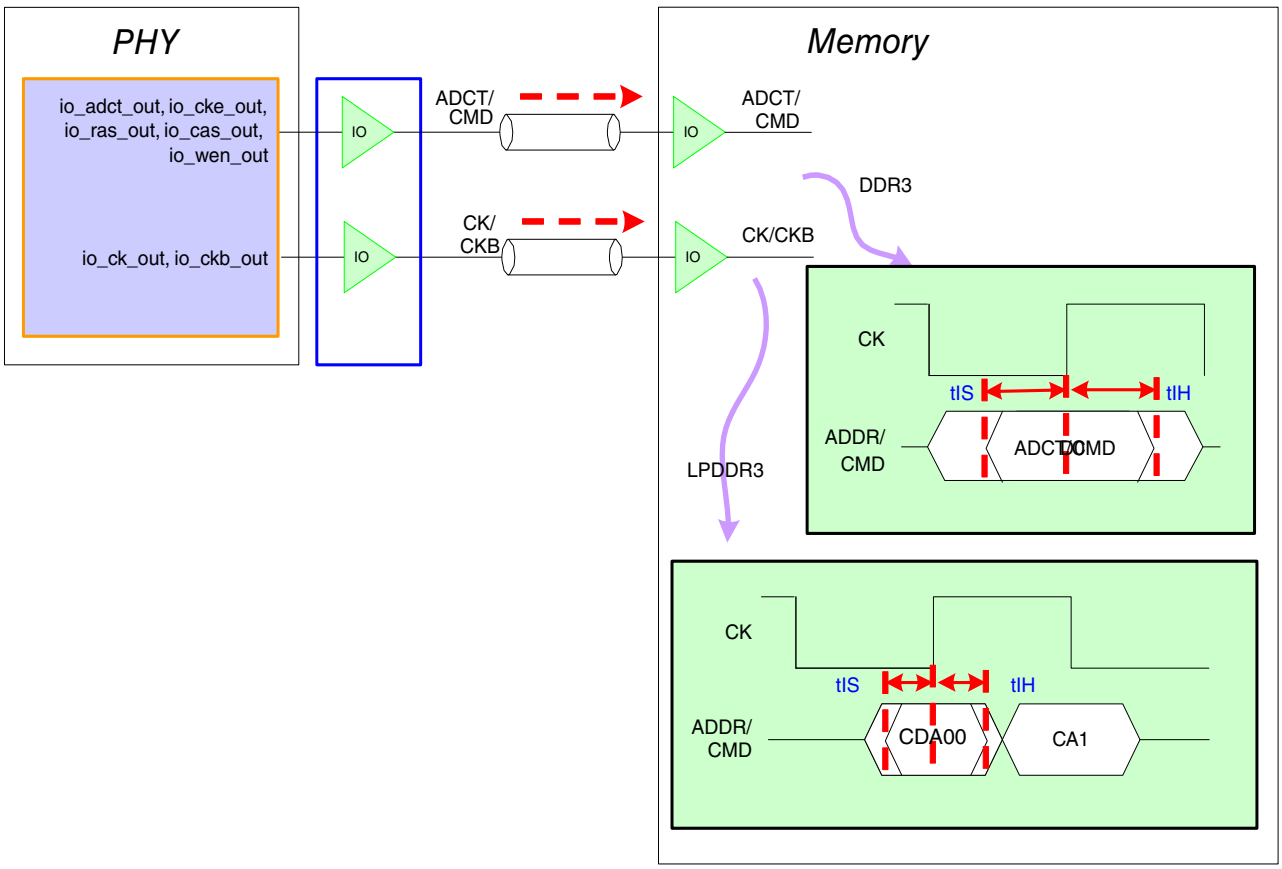


Figure 9-11. Block Diagram of the Control Path

The control block generates address, control, and clock signals that are sent to the I/O pads. ADCT[24:0] are the address traces that consist of Reset (RSN), ODT, Address[15:0], Bank Address[2:0], and Chip Selects[1:0] (chip selects [3:2] are not used in this implementation. Command signals consist of RAS, CAS, and WE. CK and CKB are differential clock signals. Address and control signals are center aligned at rising CK edge to maximize address / control signal setup / hold timing.

9.3.2.2 DATA I/F

9.3.2.2.1 Write path

Data I/F block generates data, mask and strobe signals to interface memory for memory write transaction. DQs are 8-bit width data signals, DM is data mask signal and DQS is strobe signal. Therefore, to interface 32-bit memory, 4 data slices are required.

DQS is edge-aligned signal to CK / CKB and DQs / DM are center-aligned signals to the edge of CK / CKB.

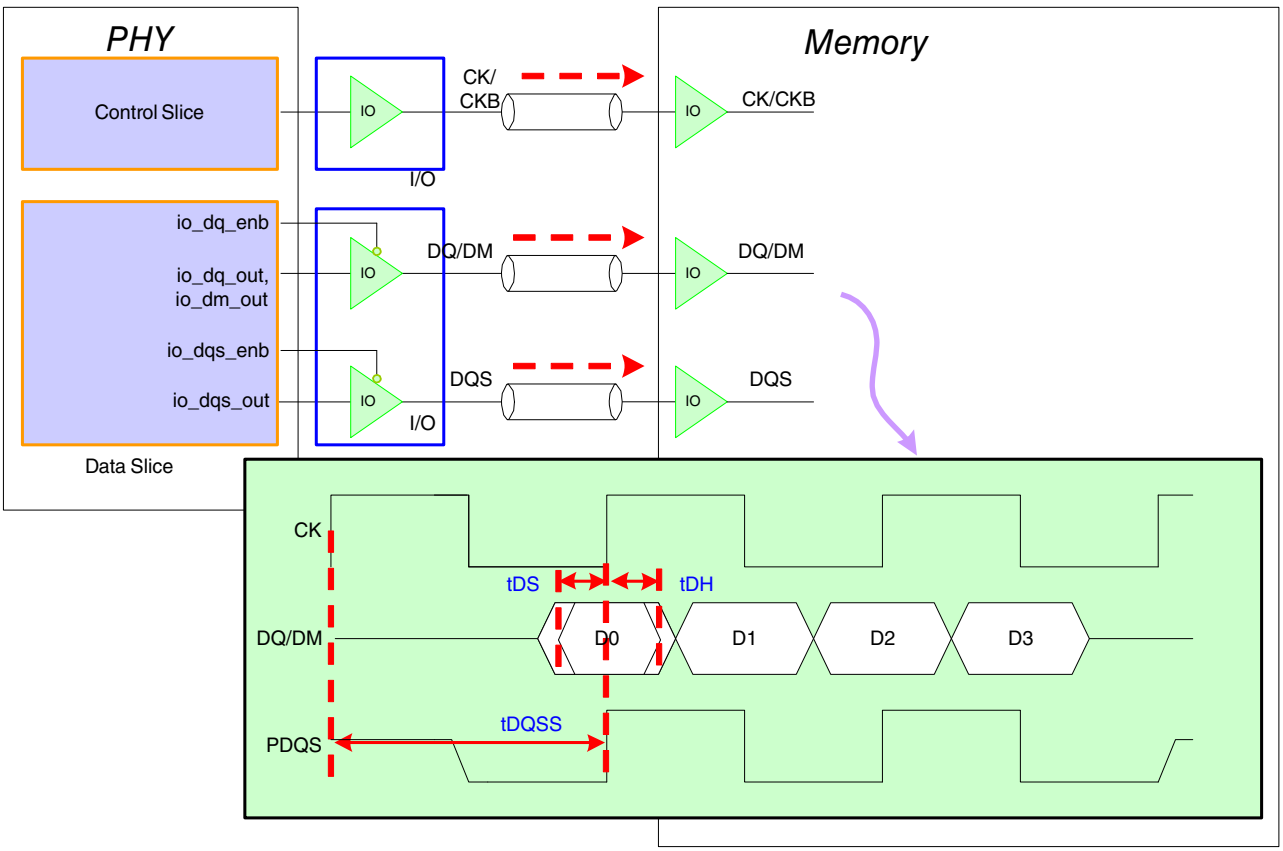


Figure 9-12. Block Diagram of the Write Path

9.3.2.2.2 Read path

Read blocks capture valid data using strobe signal which come from memory for read transaction. DQs are 8-bit width data signals, DQS is strobe signal.

DQs and DQS from memory are edge aligned to CK edge. Therefore, to capture DQs using DQS, DQS should be delayed to make 90° phase shifted DQS, i.e. DQs should be center aligned to the edge of delayed DQS and captured DQs are stored in the FIFO.

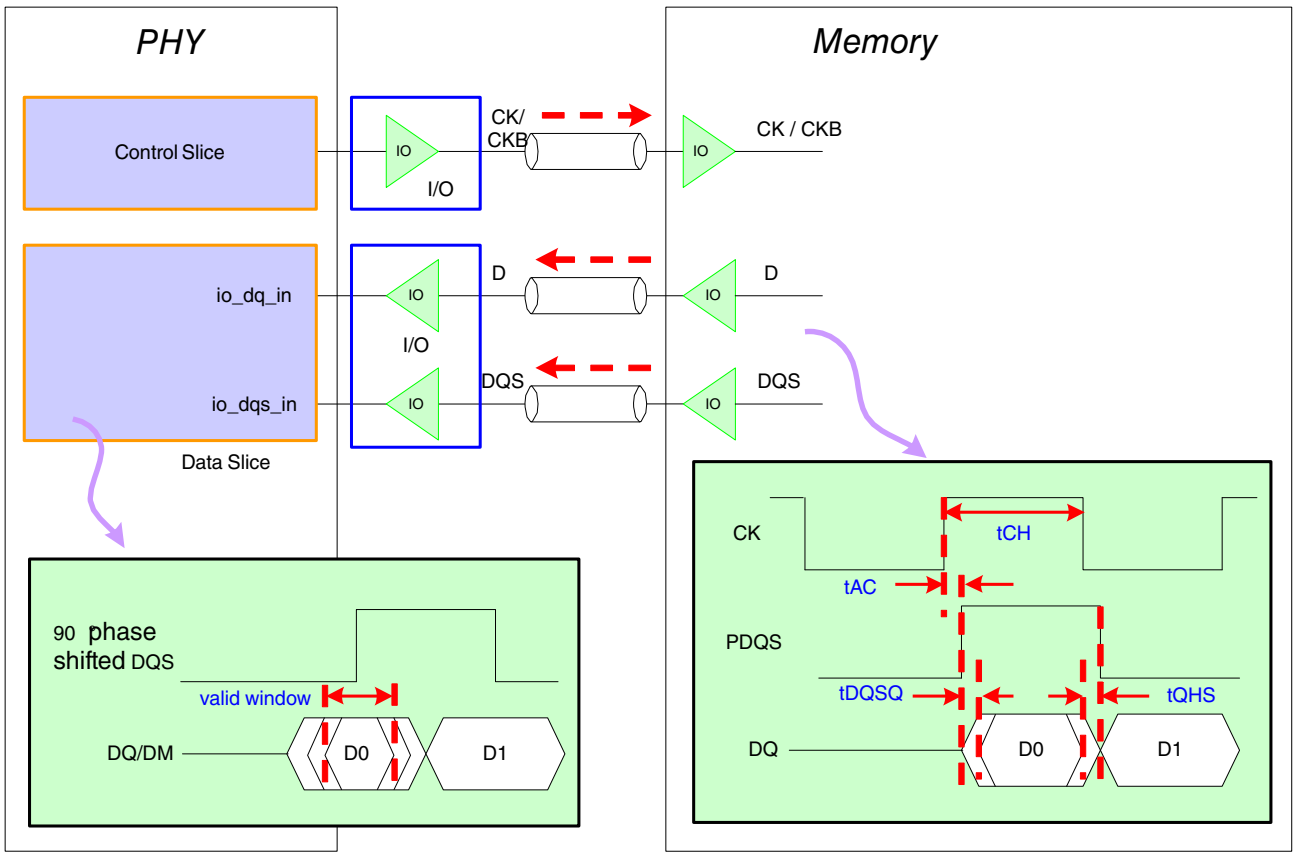


Figure 9-13. Block Diagram of the Read Path

9.3.2.2.3 ZQ CALIBRATION I/O

ZQ_IO calibrates the output and termination impedance and passes through impedance control signals to each I/O cells. RZQ should be 240 ohm.

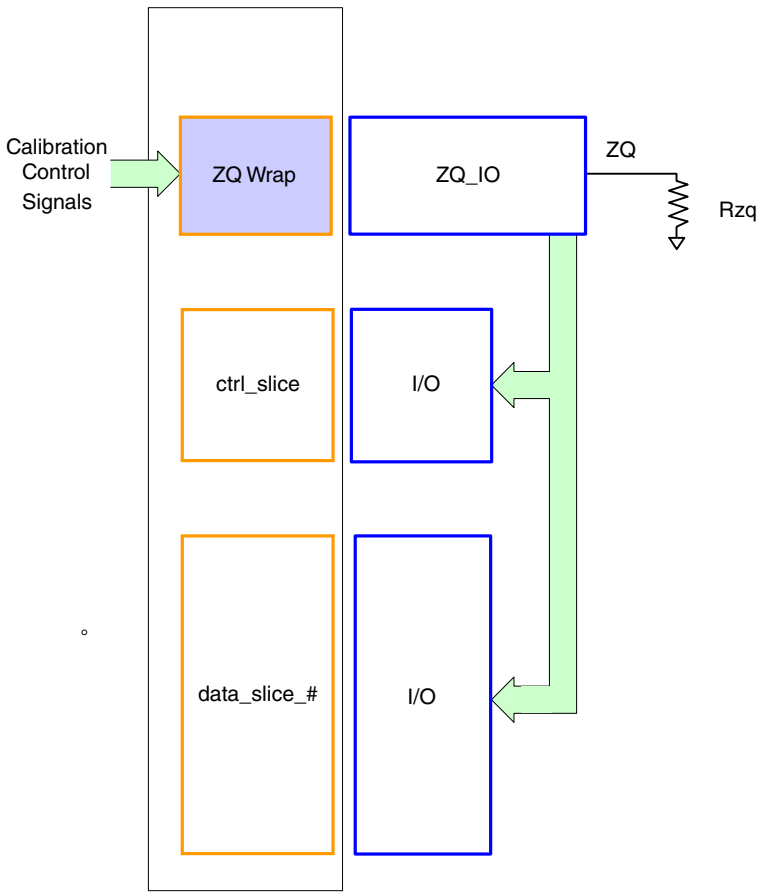


Figure 9-14. Connectivity Between ZQ I/O and the other I/Os

9.3.3 Functional description

9.3.3.1 DATAPATH

The timing parameters tphy_wrlat and tphy_wrdata define the delay from the write command to the dfi_wrdata_en_pN signal, and from the dfi_wrdata_en_pN signal to when data will be driven on the dfi_wrdata_pN signal respectively. These timing parameters are defined in terms of DFI PHY clocks.

- tphy_wrlat = WL - 1, tphy_wrdata = 2, for example ifWL = 4, tphy_wrlat = 3, tphy_wrdata = 2 (LPDDR3)
- tphy_wrlat = WL - 2, tphy_wrdata = 2, for example ifWL = 8, tphy_wrlat = 6, tphy_wrdata = 2(DDR3)

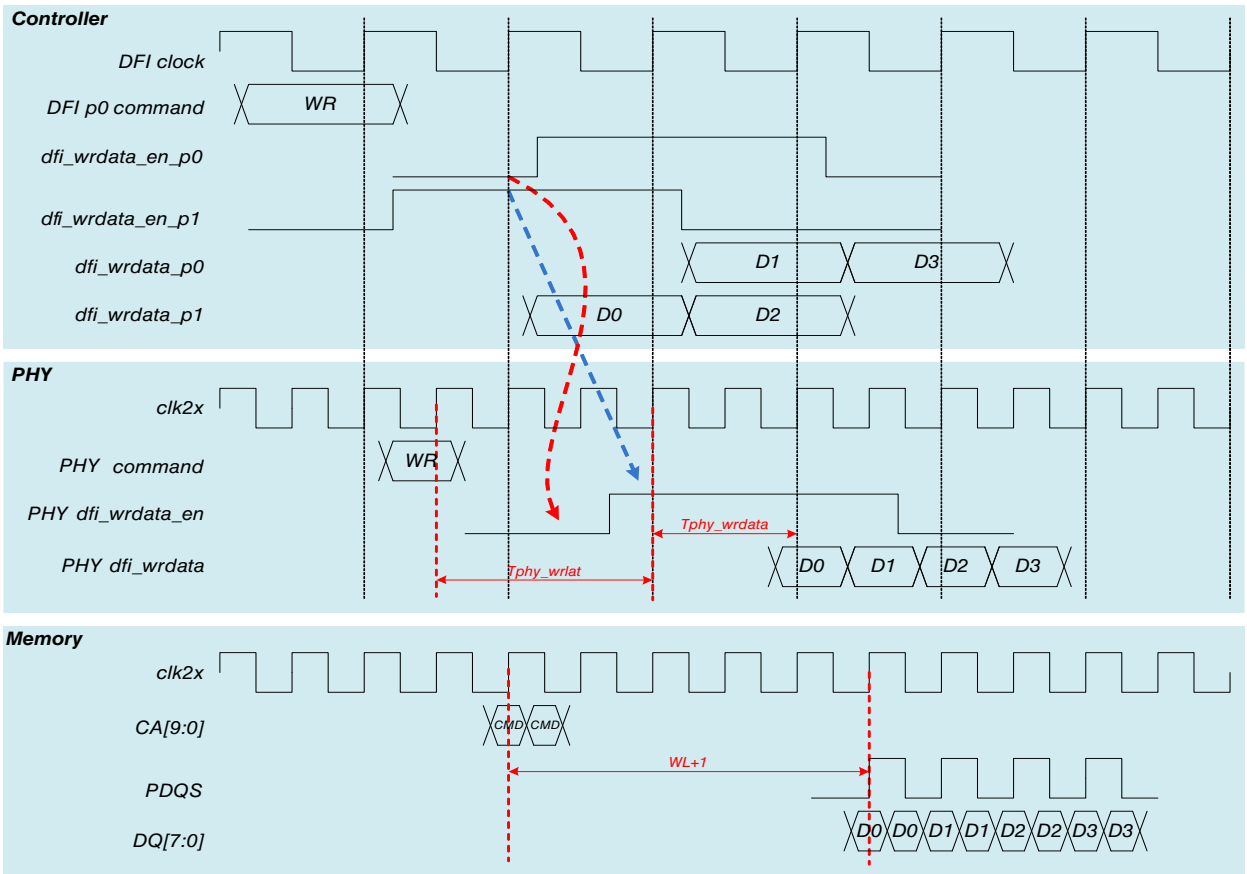


Figure 9-15. Data write Timing Diagram with phase 0 command (1:2 frequency ratio, LPDDR3)

Signal interface timing for data write is shown in the following figure. dfi_wrdata_en_p0/p1 is enable signal to indicate dfi_wrdata_p0/p1 and dfi_mask_p0/p1 are valid.

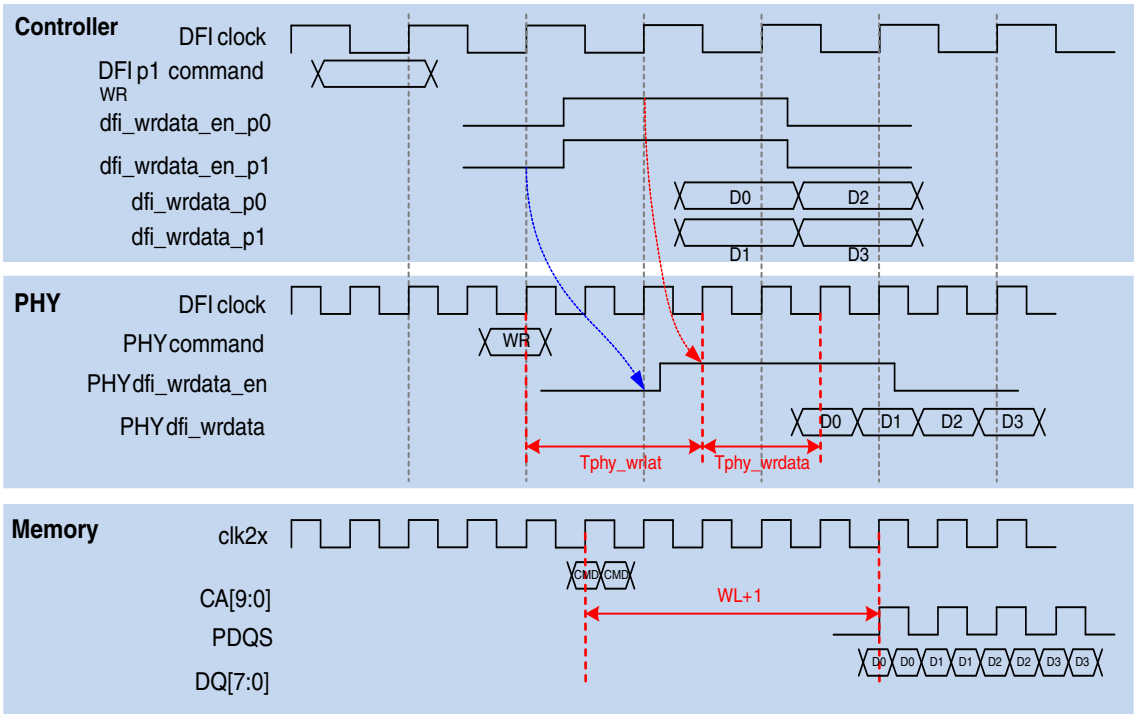


Figure 9-16. Data write Timing Diagram with phase 1 command (1:2 frequency ratio, LPDDR3)

The timing parameters trddata_en and tphy_rdlat define the delay from the read command to the dfi_rddata_en_pN signal, and from the dfi_rddata_en_pN signal to when data will be returned on the dfi_rddata_wN bus, respectively. These timing parameters are defined in terms of DFI PHY clocks and are measured relative to how the PHY interprets the data.

Signal interface timing for data read is shown in the following figure . After read command is issued, DQ and PDQS/NDQS are driven by the memory and DQ is stored in FIFO in read data path. After read data is stored in FIFO, when dfi_rddate_en_p0/p1 is issued from controller, the valid read data is driven during HIGH dfi_rddata_valid_w0/w1 signal at the rising edge of the clk signal.

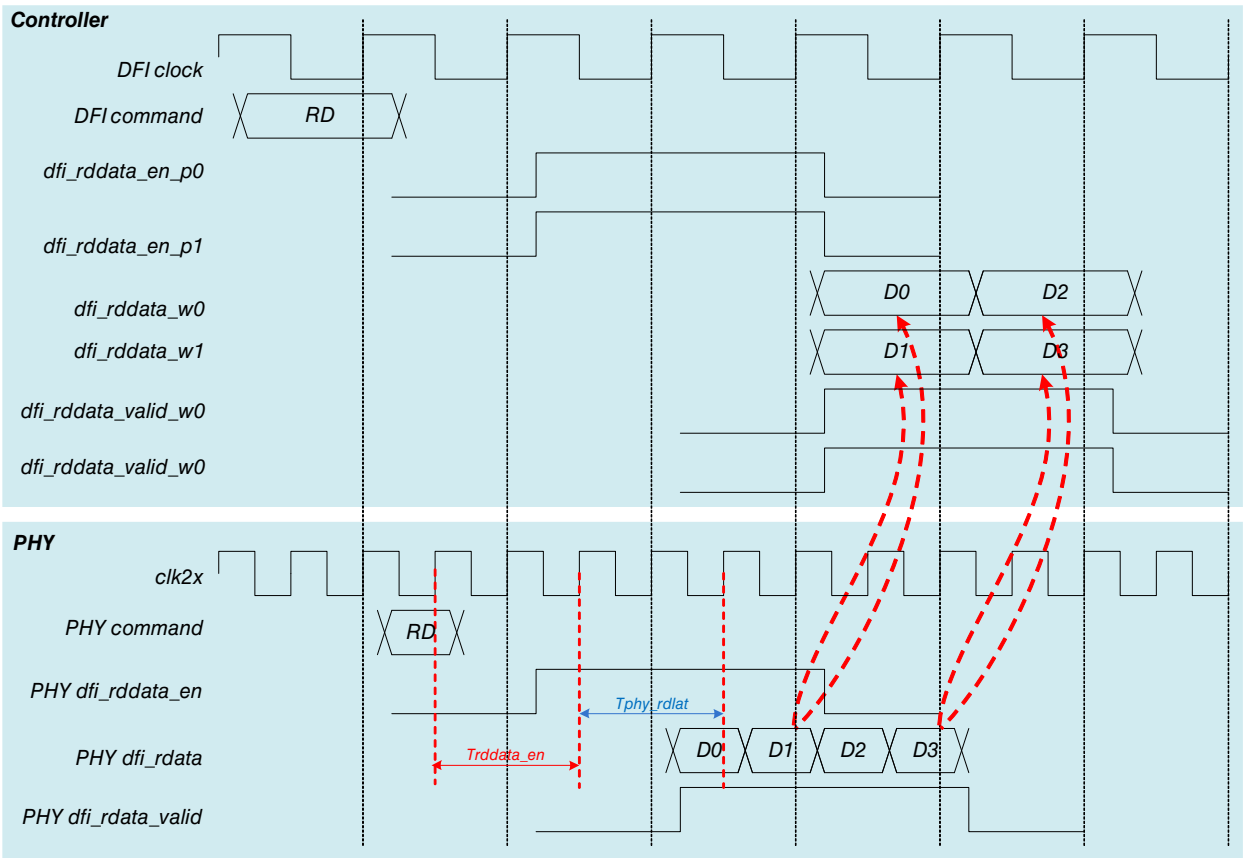


Figure 9-17. Data Read Timing Diagram (1:2 frequency ratio)

The following figure illustrates control signal and data read path. At the read path, DQS clean block is used to clean DQS signal and Delay line is used for 90° phase shift of DQS. If read command is issued at A point, read command is driven and memory drives DQ and DQS after CL(CAS Latency) at B point. DQ and DQS experience board and IO input delay and arrive at C point. Delay line execute 90° phase shift of DQS and shifted DQS is used as a clock signal to capture DQ in FIFO. After that dfi_rddata_en_p0/p1 is driven HIGH by the controller, the FIFO outputs valid read data at E point.

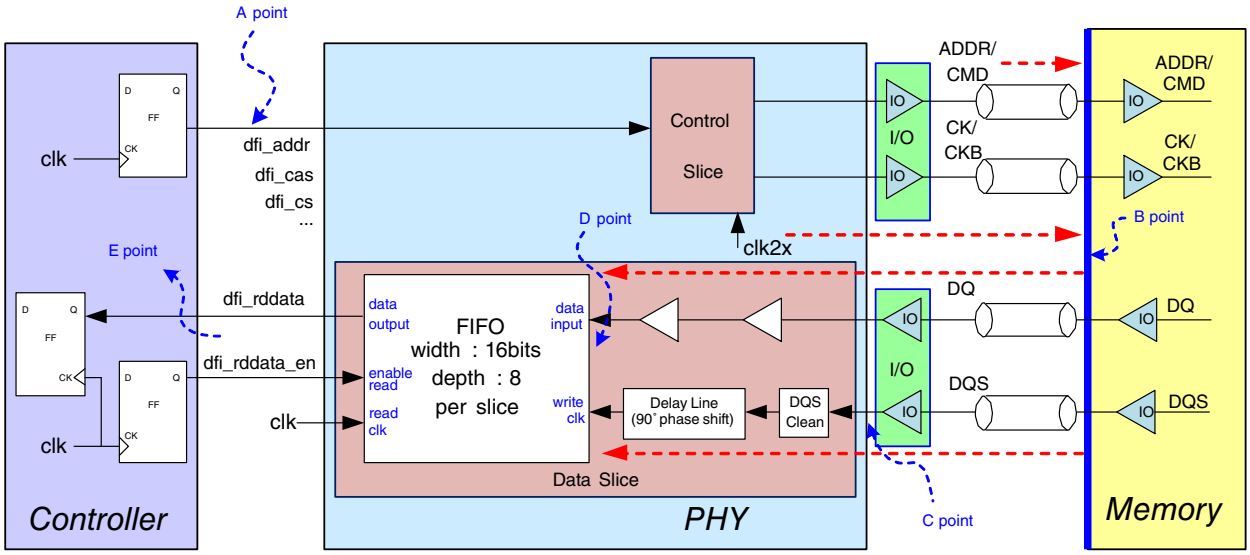


Figure 9-18. Read Data Path

The following figure illustrates the full timing diagram of command and data path for memory read. The deterministic timing values for the read path are latency to generate command (1 clock cycle), CL (CAS Latency) and data write time in FIFO (1 clock cycle). But the propagation delay of command (from PHY to memory) and DQ / DQS (from memory to PHY) varies according to the board designs and operating condition. And also internal logic delay in the following figure. is also different according to operating condition. For this reason, read data arrival time is quite different according to designs and environment and deterministic read timing calculation is difficult. To overcome this issue, FIFO of 8-depth is used in read path and data read timing can be programmed.

DQS cleaning is needed to remove high-Z state of DQS during read, but there is a lot of variation in DQS. We recommended that the pull-down mode in DQS (LP_CON0[NS-1:0]) should be used to remove high-Z state of DQS.

When it is in the pull-down mode, the duration of HIGH in "ctrl_gate_p0 /p1" should be extended to compensate the variation of DQS with ctrl_shgate = 0

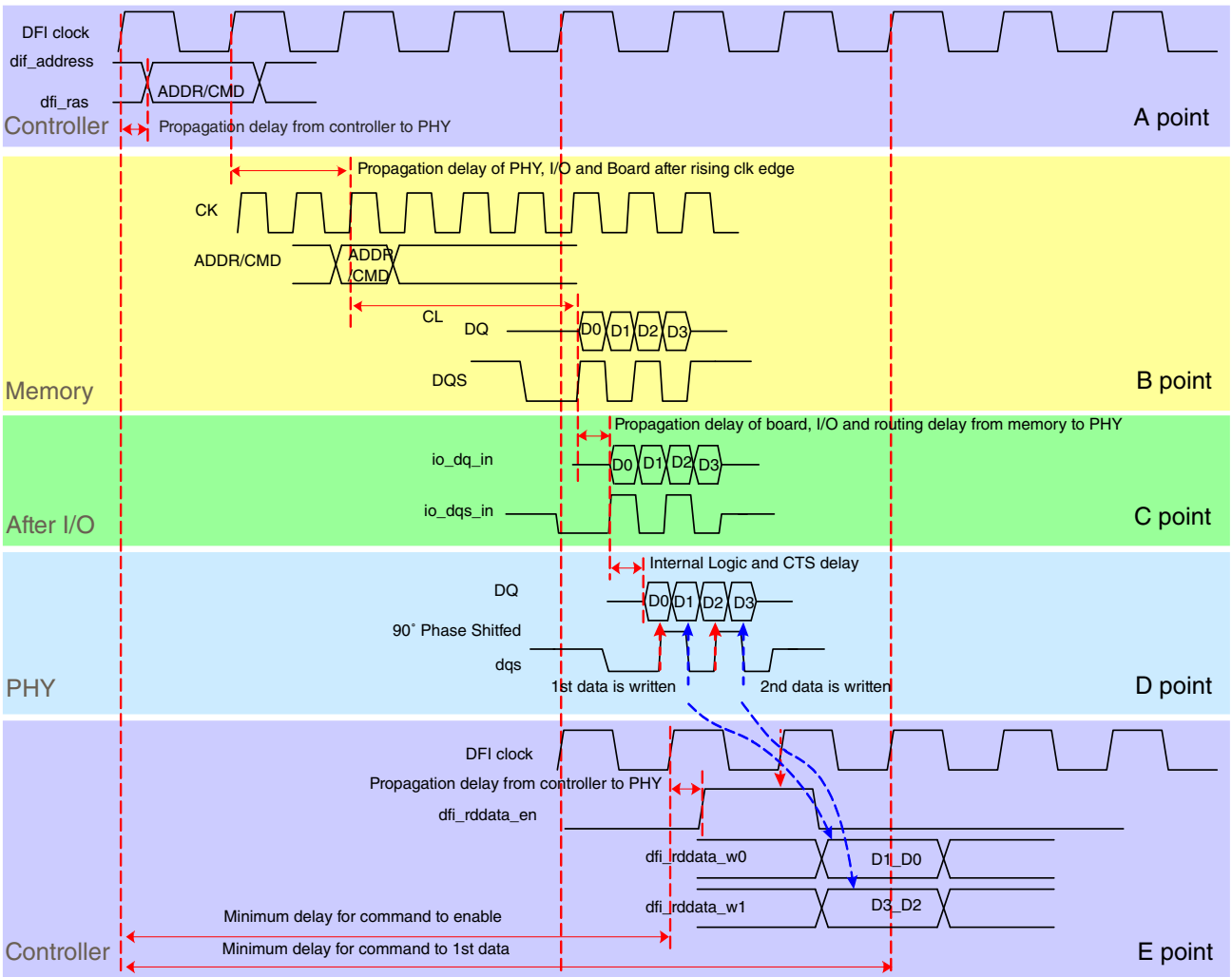


Figure 9-19. Full Read Path Timing Diagram

9.3.3.2 Package and board guide

In the following table, Skew in picoseconds is translated to physical length on the PCB.

Table 9-14. Skew Including Package and Board

Group	Signals	Skew	From	To
G0_1	CK / CKB	±10ps	Pad	Memory Pin
G0_2	PDQS[N] / NDQS[N]	±10ps	Pad	Memory Pin

Table continues on the next page...

Table 9-14. Skew Including Package and Board (continued)

Group	Signals	Skew	From	To
G1	PDQS[N] / NDQS[N] DQ[(N + 1) * 8 - 1:N * 8] DM[N]	±25ps from PDQS[N] / NDQS[N]	Pad	Memory Pin
G2	CK / CKB, RESET, ADCT[15:0], CS, RAS, CAS, BANK, CKE, WEN, ODT	±25ps from CK / CKB	Pad	Memory Pin
G3 (LPDDR2/LPDDR3)	CK / CKB, ADCT[9:0]	±25ps from CK / CKB	Pad	Memory Pin
G4	CK / CKB, P / NDQS[NS-1:0]	-100ps from CK / CKB	Pad	Memory Pin

NOTE

- Recommend that CK / CKB should be delayed than P/ NDQS[NS-1:0] (G4)
- When CK / CKB is connected to two rank, the skew of CK / CKB between two ranks should follow "G0_1" in.
- If using Write Leveling to support DIMM, The skew of "-100ps" for G4 can be changed by "-2500ps" which came from 2 cycles at 800 MHz.
- N means 0,1… NS-1.

9.3.3.3 Initialization

- After power-up and system PLL locking time, system reset (rst_n) is released.
- Select Memory Type (= PHY_CON0[12:11]).
  - ctrl_ddr_mode = 2'b11 (LPDDR3)
  - ctrl_ddr_mode = 2'b10 (LDDR2)
  - ctrl_ddr_mode = 2'b01 (DDR3)
- GATE default settings for DQS Cleaning
  - Set "ctrl_gateadj = 1, ctrl_gateduradj = 2, ctrl_shgate = 0, ctrl_pulld_dqs = 0xF, ctrl_shiftc = 0x0" (ctrl_atgate = 1)
- Set Read Latency (RL), Burst Length (BL) and Write Latency (WL)
  - SetRL (= PHY_CON4[4:0]), BL (= PHY_CON4[12:8]),WL (= PHY_CON4[20:16])
- ZQ Calibration (Please refer to ZQ I/O control procedure for more details)
  - Enable and Disable "zq_clk_div_en" in ZQ_CON0[18]
  - Enable "zq_manual_str" in ZQ_CON0[1]
  - Wait until "zq_cal_done" (ZQ_CON1[0]) is enabled.
  - Disable "zq_manual_str" in ZQ_CON0[1]

- Memory Controller should assert "dfi_init_start" from LOW to HIGH.
- Memory Controller should wait until "dfi_init_complete" is set to finish DLL lock.
- Enable DQS pull down mode
  - Please be careful that DQS pull down can be disabled only after Gate Leveling is done.
- Memory Controller should assert "dfi_ctrlupd_req" after "dfi_init_complete" is set.
- Start Memory Initialization by memory controller.

NOTE

Recommend to set "upd_mode = 0" (= OFFSETD_CON0[28])
for PHY-Initiated Update

Caution: Please do not change the frequency of "clkm" or voltage during operation. Those conditions should be changed without memory access. After changing, "ctrl_start" should be clear and set to lock again.

9.3.3.4 Training Procedure

9.3.3.4.1 WRITE LEVELING

Write Leveling compensates for the additional flight time skew delay introduced by the package, board and on- chip with respect to strobe(= DQS) and clock.Write Leveling can support 1 rank only (CS[0] or CS[1]), so CK and DQS delay at each rank should be same as possibe to support two ranks. For example, CK delay at rank0 and rank1 should be designed to be same on package / board.

9.3.3.4.1.1 H/W write leveling

- Memory Controller should configure Memory (DDR3, LPDDR3) in Write Level mode.
  - Set "cmd_default[8:7] = 2'b11" (LPDDR_CON4[8:7]) to enable "ODT[1:0]" signals during Write Leveling.
- Configure PHY in Write Level mode.
  - Enable "wrlvl_mode" in PHY_CON0[16].
- Start Write Leveling by setting "wrlvl_start = 1'b1" (= PHY_CON3[16])
- Wait until "wrlvl_resp = 1'b1" (= PHY_CON3[24])
- Finish Write Leveling by setting "wrlvl_start = 1'b0" (= PHY_CON3[16])
- Configure PHY in normal mode.

    - Disable "wrlvl_mode" in PHY_CON0[16].
- Set "cmd_default[8:7] = 2'b00" (LPDDR_CON4[8:7]) to disable "ODT[1:0]" signals.
- Set"reg_mode[0] = 1'b1" (= PHY_CON3[0]).
- Memory Controller should configure Memory (DDR3, LPDDR3) in normal mode.
- Enable and Disable "ctrl_resync" (= OFFSETD_CON0[24])" to make sure All SDLL is updated.
- Recommend to set "ctrl_readduradj = 1"

9.3.3.4.1.2 Write leveling dll manual setting

After knowing the board and package delay exactly, you can use the following manual setting instead ofWrite Leveling.

- Set WR_LVL_CON0 for Data Slices 0-3.
    - For example, suppose the following conditions
        - "ctrl_lock_value[8:0] = 0x7F(= 127)"
        - The delay of CK is about 118ps, 295ps, 512ns, and 704ps for Data Slices 0-3.
    - Fine step delay will be about 9.84ps by calculating "1250 / 127" at 800 MHz.
    - The delay of CK can be represented by 0x0C, 0x1E, 0x34, and 0x47 with Fine step delay unit.
        - If the delay of CK at DRAM is greater than 1 or 2 cycle, please ignore that cycle delay when setting "WR_LVL_CON0". For example, the delay of CK for Data Slice0 is 1329ns and the fine step delay should be 0x85, but after ignoring 0x7F (= 1250ns), the setting value will be 0x08.
    - Please set "WR_LVL_CON0 = 0x47341E0C"
- Set "wrlvl_mode = 1" (= PHY_CON0[16])
- • Set "wrlvl_mode = 0" (= PHY_CON0[16])
- • Enable and Disable "ctrl_resync" (= OFFSETD_CON0[24])" to make sure All SDLL is updated.

9.3.3.4.2 GATE LEVELING

NOTE

Gate Leveling is a feature used for DDR3 mode only. Do not use Gate Leveling for LPDDR2/LPDDR3.

The goal of gate training is to locate the delay at which the initial read DQS rising edge aligns with the rising edge of the read DQS gate (= ctrl_gate_p0 / p1). Once this point is identified, the read DQS gate can be adjusted prior to the DQS, to the approximate midpoint of the read DQS preamble.

- Controller should configure Memory (DDR3) in MPR mode.
- Set Gate Leveling Mode.(1)
  - Enable "gate_cal_mode" (= PHY_CON2[24])
  - Enable "ctrl_shgate" (= PHY_CON0[8])
  - Set "ctrl_gateduradj[3:0]" (= PHY_CON1[23:20]) in the following way.
    - Set " 4'b0000" (DDR3)
    - Set " 4'b1011" (LPDDR3)
    - Set " 4'b1001" (LPDDR2)
- Enable "gate_lvl_start (= PHY_CON3[18])" to do read leveling.(2)
- Wait until "rd_wr_cal_resp" (= PHY_CON3[26]) is set.(3)
  - The maximum waiting time will be 20μs. If the any command (the refresh or pre-charge command) is required within 20μs, please issue those commands before "(1)".
- Disable "gate_lvl_start (= PHY_CON3[18])" after "rd_wr_cal_resp"(= PHY_CON3[26]) is disabled.
- Disable DQS pull down mode.(4)
- Memory Controller should configure Memory (DDR3) in normal mode.

9.3.3.5 Low frequency operation

Even if the operation frequency of "clk2x" is out of the range of MDLL Input frequency (400 ~ 800 MHz), DDR PHY can operate at the low frequency because MDLL Input clock is separated from PHY Input clock. It is recommended that Read Leveling should be done in the normal operation. The following sequence is how to operate PHY at low frequency with the different MDLL clock.

- If "dfi_init_start" = 0, Controller should assert "dfi_init_start" from LOW to HIGH.
- After "dfi_init_complete" is checked by controller, read the value of "ctrl_lock_value".
- Enter Self-Refresh (CKE = 0)
- Go to the low frequency.
  - Change "clk2x" to the low frequency, but do not change the frequency of "clkm"(400 ~ 800 MHz).
- Write the multiplied value of ctrl_lock_value[8:0] to ctrl_force[8:0].
  - Enter DLL OFF mode before Leveling.
    - Set "ctrl_dll_on = 0" (= MDLL_CON0[5])
    - Read "ctrl_lock_value[8:0]" (= MDLL_CON1[16:8]) after setting "ctrl_dll_on = 0"
  - For example, if the operation frequency will be the half of MDLL clock, two multiplied value of ctrl_lock_value should be written to "ctrl_force".
  - If "the multiplied value" is more than 0x1FF, "ctrl_force" will be "0x1FF".

- The low frequency is under 100 MHz,
  - Enter DLL OFF mode by setting "ctrl_dll_on = 0".
  - Set "ctrl_force" = 0x1FF, "ctrl_offsetd" = 0x7F, ctrl_offsetr* = 0x7F, ctrl_offsetw* = 0x7F (*means 0 ~ 3)
  - Set CA0DeSkewCode = 0x60, CA1DeSkewCode = 0x60, ~ CA9DeSkewCode = 0x60.
- "dfi_ctrlupd_req" should be issued more than 10 cycles after ctrl_dll_on is disabled.
- Exit Self-Refresh (CKE = 1).
- Operate in the low frequency.

Caution: If using RL = 3 or 4, "ctrl_atgate" (= PHY_CON0[6]) should be set as "0", and "ctrl_gate_p0 / p1" and "ctrl_read_p0 / p1" should be generated from controller.

If it goes back to the original high frequency again, please refer to the following procedures.

- Enter Self-Refresh (CKE = 0)
- If "ctrl_dll_on = 0", turn on "ctrl_dll_on".
- "ctrl_offsetd" = 0x08, ctrl_offsetr* = 0x08, ctrl_offsetw* = 0x08 (*means 0 ~ 3)
- CA0DeSkewCode ~ CA9DeSkewCode = 0x8.
- Wait until "ctrl_clock = 1".
- Controller should assert "dfi_ctrlupd_req" to apply the new "ctrl_lock_value" after "ctrl_clock = 1".
- Exit Self-Refresh (CKE = 1)

9.3.3.6 Offset control

ctrl_offset0 ~ 3 control the offset of 90° phase shift of DQS or 270° clock.

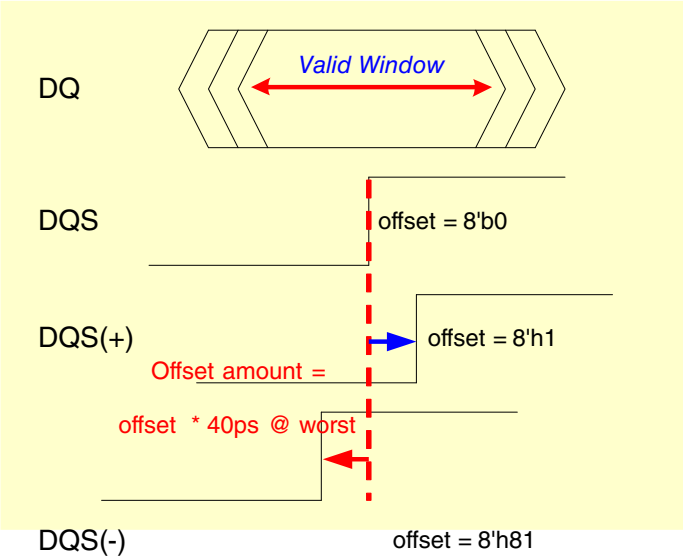


Figure 9-20. Offset Control Example for ctrl_offset0 ~ ctrl_offset3

9.3.3.7 DLL lock procedure

- After power-up and system PLL locking time, system reset (rst_n) is released.
- Assert "dfi_init_start" from LOW to HIGH.
- When DLL lock is finished, "dfi_init_complete" is set.
- Before memory access, "dfi_ctrlupd_req" should be applied. It is recommended that "dfi_ctrlupd_req" should be set and clear at the start of refresh cycle automatically by the memory controller to update DLL lock information periodically.

DLL is used to compensate PVT condition. Therefore DLL should not be turned-off for reliable operation except for the case of frequency scaling.(only to lower frequency scaling is permitted).

To turn-off the DLL, follow the next steps.

- After DLL locking, Enter DLL OFF mode by clearing ctrl_dll_on.
- CPU reads ctrl_lock_value and write ctrl_lock_value[8:0] to ctrl_force.
- "dfi_ctrlupd_req" should be issued 6 cycles after ctrl_dll_on was set and cleared.

9.3.3.8 ZQ I/O control procedure

ZQ I/O calibrates the I/Os to match the driving and termination impedance by referencing resistor value of resistor (RZQ) connected externally from ZQ pin to ground. For DDR3, RZQ should be 240ohm. One-time calibration is provided for ZQ I/O control procedure. There are two modes for one-time calibration. One is “Long calibration mode” and the other is “Short calibration mode”.

9.3.3.8.1 One-time calibration

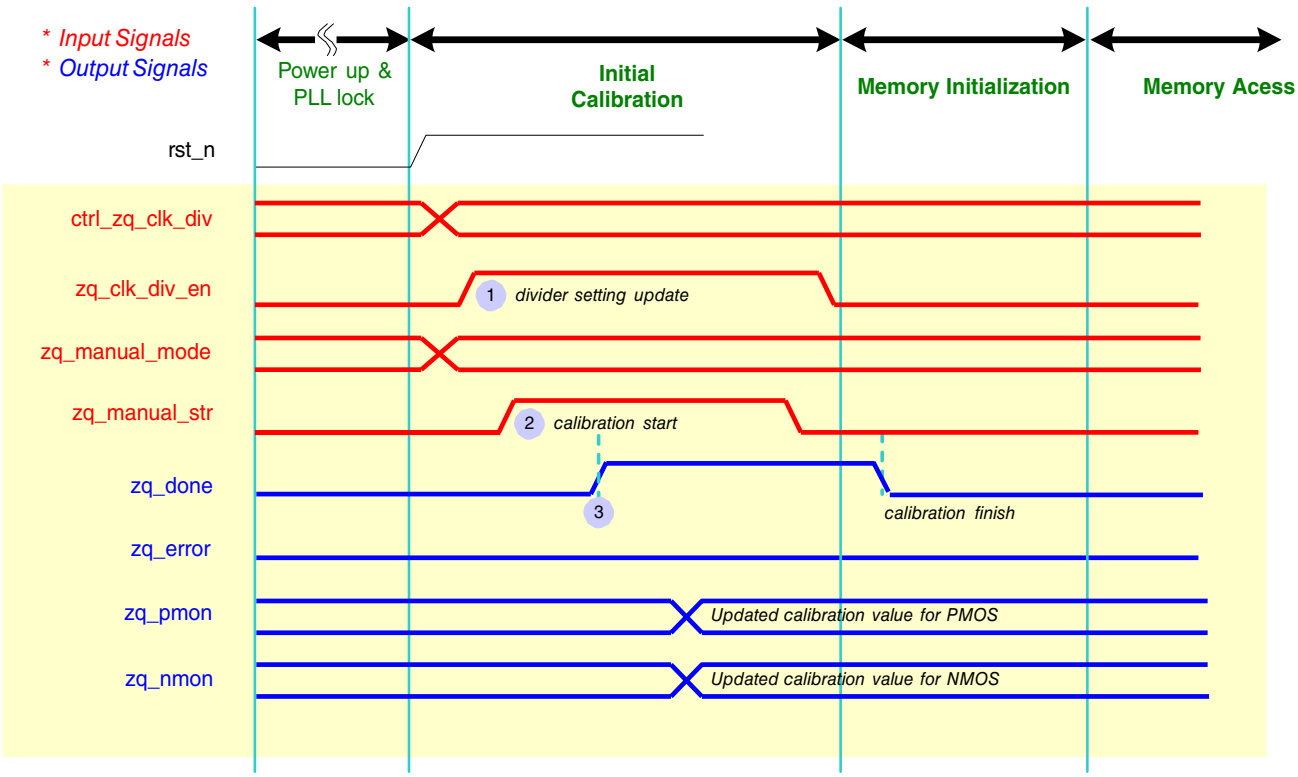


Figure 9-21. One-time Calibration Procedure

- After power-up and system PLL locking time, system reset (rst_n) is released.
- Set ctrl_zq_clk_div[31:0] to proper value (= 0x7)
- Set zq_clk_div_en from 1’b0 to 1’b1 to update divider settings (ctrl_zq_clk_div[31:0] = 0x7).
- Set zq_manual_mode
  - Long calibration mode: 2’b01
  - Short calibration mode: 2’b10
- Start ZQ I/O calibration by setting zq_manual_str from 1’b0 to 1’b1
- When calibration is done, zq_done (= PHY_CON17[0]) will be set.

- After zq_done (= PHY_CON17[0]) is asserted, clear zq_manual_str
- Clear zq_clk_div_en

9.3.3.8.2 Manual setting

- After power-up and system PLL locking time, system reset (rst_n) is released.
- Set ctrl_zq_clk_div[31:0] to proper value (= 0x7)
- Set zq_clk_div_en from 1’b0 to 1’b1 to update divider settings (ctrl_zq_clk_div[31:0] = 0x7).
- Set zq_manual_mode=2'b00 (= Force calibration mode).
- Start ZQ I/O calibration by setting zq_manual_str from 1’b0 to 1’b1
- After zq_done (= PHY_CON17[0]) is asserted, clear zq_manual_str.

9.3.3.9 DLL code update

Assertion of the "dfi_ctrlupd_ack" or "dfi_phyupd_ack" signal indicates the control, read and write interfaces on the DFI should be idle. While the "dfi_ctrlupd_ack" or "dfi_phyupd_ack" signal is asserted, the DFI bus may only be used for commands related to the update process.

The MC guarantees that dfi_ctrlupd_req signal will be asserted for at least tctrlupd_min cycles, allowing the PHY time to respond. To acknowledge the request, the dfi_ctrlupd_ack signal must be asserted while the dfi_ctrlupd_req signal is asserted. The dfi_ctrlupd_ack signal must de-assert at least one cycle before tctrlupd_max expires (Tctrlupd_min = 2, Tctrlupd_max = 21). Please use "MC-Initiated Update" until initialization including all calibrations is finished.

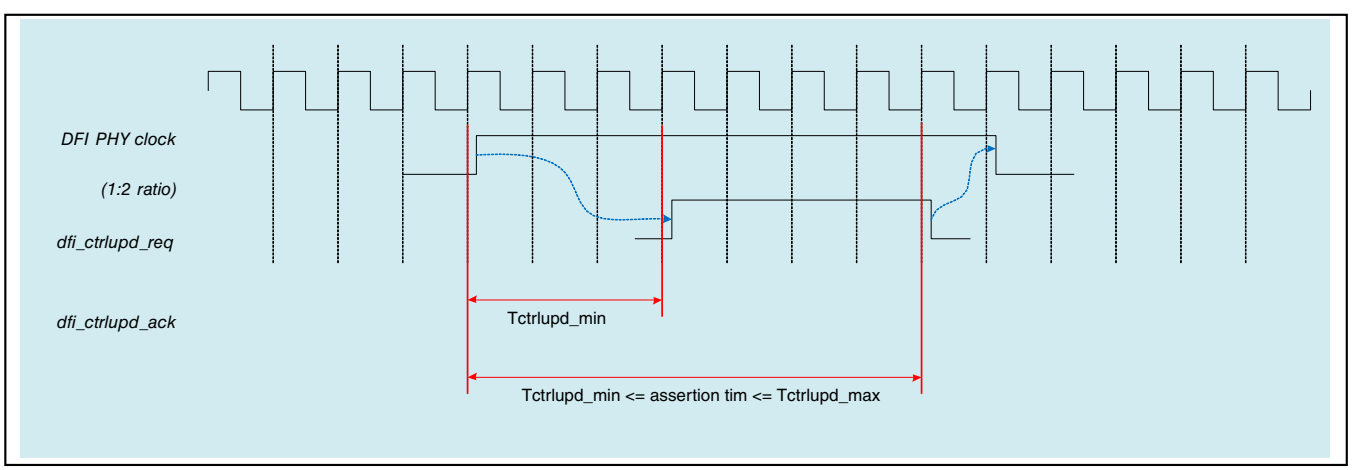


Figure 9-22. MC-Initiated Update Timing Diagram

Assertion of the dfi_phyupd_req signal indicates the control, read and write interfaces on the DFI are required. While the dfi_phyupd_ack signal is asserted, the DFI bus should be idle.

The Tphyupd_typeX parameters indicate the number of cycles of idle time on the DFI control, read and write data interfaces being requested. The dfi_phyupd_ack signal must assert within Tphyupd_resp cycles after the assertion of the dfi_phyupd_req signal (Tphyupd_resp = 64, Tphyupd_typeX = 4).

If MC (Memory Controller) does not assert dfi_phyupd_ack within Tphyupd_resp, PHY will keep dfi_phyupd_req asserting until dfi_phyupd_ack is asserted and Tphyupd_typeX will be changed up to "21".

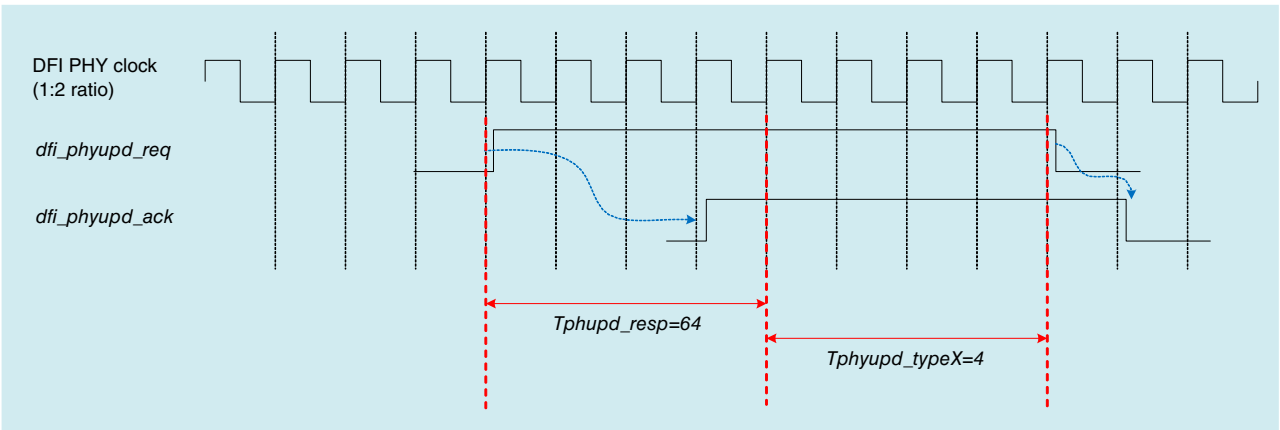


Figure 9-23. PHY-Initiated Update Timing Diagram

In case of LPDDR2 / LPDDR3, The PHY is expected to drive values on dfi_address[9:0] to ADCT[9:0] on the rising edge of clock and value on dfi_address[19:10] to ADCT[9:0] on falling edge of memory clock. But PHY will drive value on dfi_address[19:10] of controller to ADCT[9:0] during 4 clock cycles after "dfi_phyupd_ack" is issued.

When CPU or MC change the code value in SDLL or De-skew DLL by using APB Interface, "ctrl_resync" (PHY_CON[24]) or "wrlvl_resync" (= WR_LVL_CON3[0]) should be high and then low to apply the updated code values. And the following conditions should be met during the assertion of "dfi_ctrlupd_req", "ctrl_resync" or "wrlvl_resync".

- CS = 1 or CKE = 0 if any CA SDLL (LP_DDR_CON2) or DeSkew (WR_LVL_CON*) Code is changed.
- Any Write operation should not be permitted if Write SDLL (OFFSETW_CON*) Code is changed.

Caution: If the DeSkew Code for CKE[1:0], CS[1:0], CK, RESET is changed, CKE should be always "LOW" during the assertion of "dfi_ctrlupd_req", "ctrl_resync" or "wrlvl_resync".

9.3.3.10 Clock control

dfi_dram_clk_disable controls CK / CKB signals.

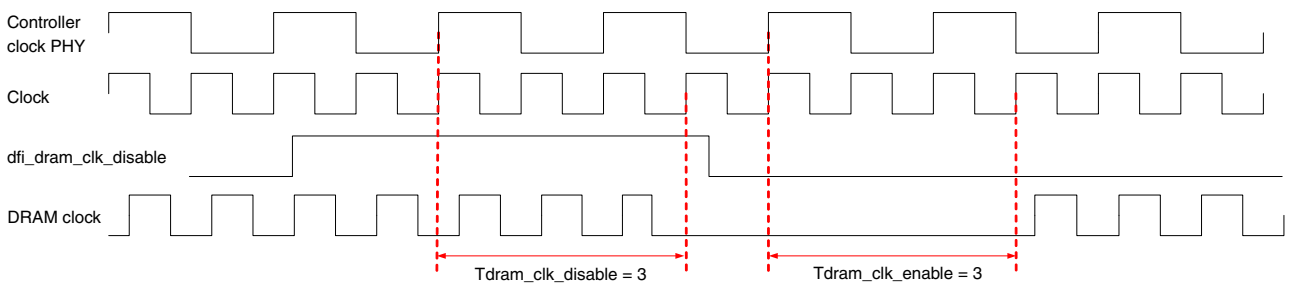


Figure 9-24. Clock Control Timing

9.3.4 Registers

DDR_PHY memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3079_0000	DDR_PHY_PHY_CON0	32	R/W	0000_0000h	9.3.4.1/ 2306
3079_0004	DDR_PHY_PHY_CON1	32	R/W	0000_0000h	9.3.4.2/ 2308
3079_0008	DDR_PHY_PHY_CON2	32	R/W	0000_0000h	9.3.4.3/ 2309
3079_000C	DDR_PHY_PHY_CON3	32	R/W	0000_0000h	9.3.4.4/ 2310
3079_0010	DDR_PHY_PHY_CON4	32	R/W	0000_0000h	9.3.4.5/ 2311
3079_0014	DDR_PHY_PHY_CON5	32	R/W	0000_0000h	9.3.4.6/ 2312
3079_0018	DDR_PHY_LP_CON0	32	R/W	0000_0000h	9.3.4.7/ 2313
3079_001C	DDR_PHY_RODT_CON0	32	R/W	0000_0000h	9.3.4.8/ 2314

Table continues on the next page...

DDR_PHY memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3079_0020	DDR_PHY_OFFSET_RD_CON0	32	R/W	0000_0000h	9.3.4.9/ 2315
3079_0030	DDR_PHY_OFFSET_WR_CON0	32	R/W	0000_0000h	9.3.4.10/ 2316
3079_0040	DDR_PHY_GATE_CODE_CON0	32	R/W	0000_0000h	9.3.4.11/ 2318
3079_004C	DDR_PHY_SHIFTC_CON0	32	R/W	0000_0000h	9.3.4.12/ 2319
3079_0050	DDR_PHY_CMD_SDLL_CON0	32	R/W	0000_0000h	9.3.4.13/ 2321
3079_006C	DDR_PHY_LVL_CON0	32	R/W	0000_0000h	9.3.4.14/ 2322
3079_0078	DDR_PHY_LVL_CON3	32	R/W	0000_0000h	9.3.4.15/ 2323
3079_007C	DDR_PHY_CMD_DESKEW_CON0	32	R/W	0000_0000h	9.3.4.16/ 2324
3079_0080	DDR_PHY_CMD_DESKEW_CON1	32	R/W	0000_0000h	9.3.4.17/ 2324
3079_0084	DDR_PHY_CMD_DESKEW_CON2	32	R/W	0000_0000h	9.3.4.18/ 2325
3079_0088	DDR_PHY_CMD_DESKEW_CON3	32	R/W	0000_0000h	9.3.4.19/ 2326
3079_0094	DDR_PHY_CMD_DESKEW_CON4	32	R/W	0000_0000h	9.3.4.20/ 2326
3079_009C	DDR_PHY_DRVDS_CON0	32	R/W	0000_0000h	9.3.4.21/ 2327
3079_00B0	DDR_PHY_MDLL_CON0	32	R/W	0000_0000h	9.3.4.22/ 2328
3079_00B4	DDR_PHY_MDLL_CON1	32	R/W	0000_0000h	9.3.4.23/ 2329
3079_00C0	DDR_PHY_ZQ_CON0	32	R/W	0000_0000h	9.3.4.24/ 2332
3079_00C4	DDR_PHY_ZQ_CON1	32	R/W	0000_0000h	9.3.4.25/ 2334
3079_00C8	DDR_PHY_ZQ_CON2	32	R/W	0000_0000h	9.3.4.26/ 2335
3079_0190	DDR_PHY_RD_DESKEW_CON0	32	R/W	0000_0000h	9.3.4.27/ 2335
3079_019C	DDR_PHY_RD_DESKEW_CON3	32	R/W	0000_0000h	9.3.4.28/ 2336
3079_01A8	DDR_PHY_RD_DESKEW_CON6	32	R/W	0000_0000h	9.3.4.29/ 2336
3079_01B4	DDR_PHY_RD_DESKEW_CON9	32	R/W	0000_0000h	9.3.4.30/ 2337

Table continues on the next page...

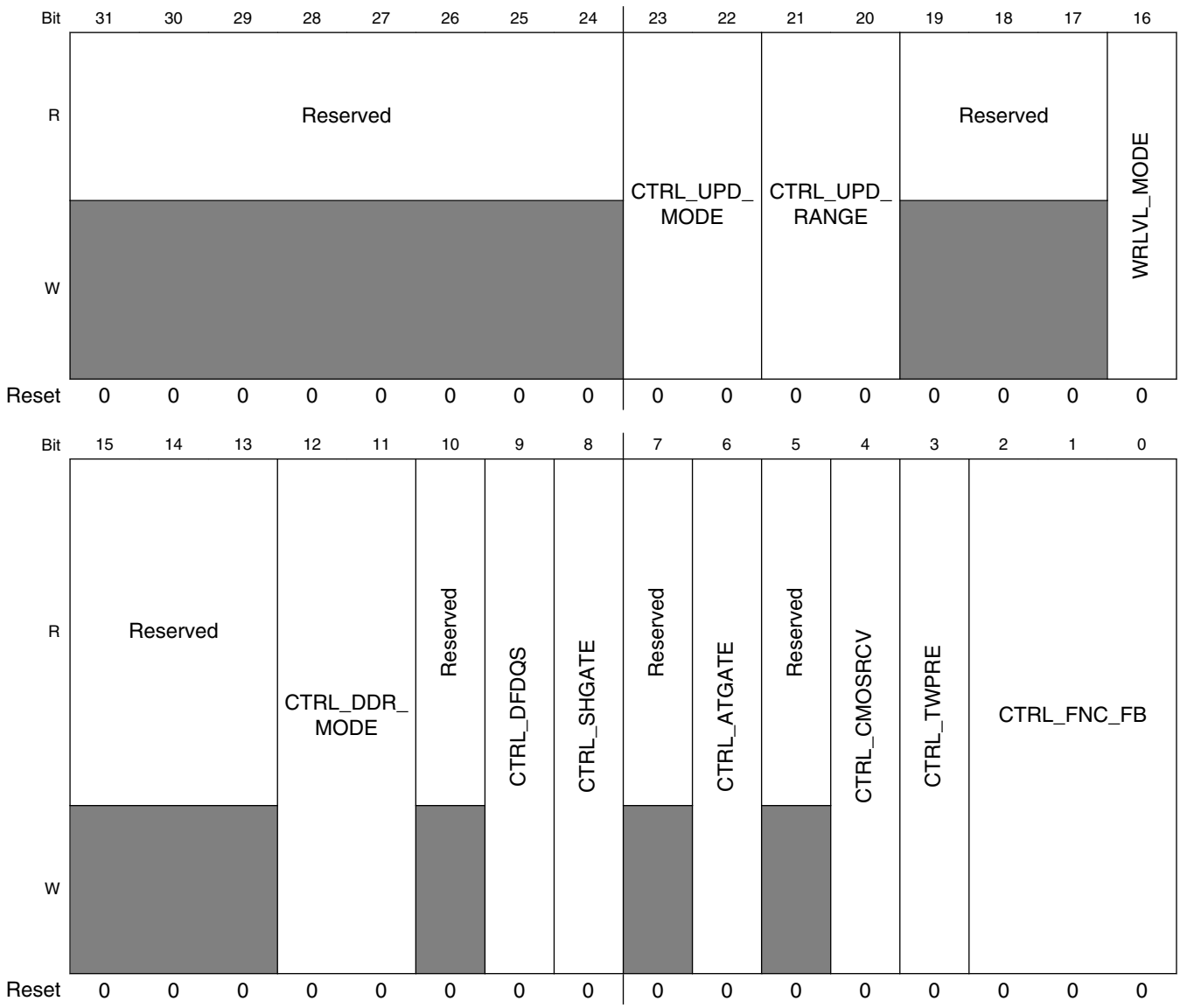
DDR_PHY memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3079_01C0	DDR_PHY_RD_DESKEW_CON12	32	R/W	0000_0000h	9.3.4.31/ 2337
3079_01CC	DDR_PHY_RD_DESKEW_CON15	32	R/W	0000_0000h	9.3.4.32/ 2338
3079_01D8	DDR_PHY_RD_DESKEW_CON18	32	R/W	0000_0000h	9.3.4.33/ 2338
3079_01E4	DDR_PHY_RD_DESKEW_CON21	32	R/W	0000_0000h	9.3.4.34/ 2339
3079_01F0	DDR_PHY_WR_DESKEW_CON0	32	R/W	0000_0000h	9.3.4.35/ 2339
3079_01FC	DDR_PHY_WR_DESKEW_CON3	32	R/W	0000_0000h	9.3.4.36/ 2340
3079_0208	DDR_PHY_WR_DESKEW_CON6	32	R/W	0000_0000h	9.3.4.37/ 2340
3079_0214	DDR_PHY_WR_DESKEW_CON9	32	R/W	0000_0000h	9.3.4.38/ 2341
3079_0220	DDR_PHY_WR_DESKEW_CON12	32	R/W	0000_0000h	9.3.4.39/ 2341
3079_022C	DDR_PHY_WR_DESKEW_CON15	32	R/W	0000_0000h	9.3.4.40/ 2342
3079_0238	DDR_PHY_WR_DESKEW_CON18	32	R/W	0000_0000h	9.3.4.41/ 2342
3079_0244	DDR_PHY_WR_DESKEW_CON21	32	R/W	0000_0000h	9.3.4.42/ 2343
3079_0250	DDR_PHY_DM_DESKEW_CON	32	R/W	0000_0000h	9.3.4.43/ 2343
3079_03A0	DDR_PHY_RDATA0	32	R/W	0000_0000h	9.3.4.44/ 2344
3079_03AC	DDR_PHY_STAT0	32	R/W	0000_0000h	9.3.4.45/ 2344

9.3.4.1 DDR_PHY_PHY_CON0

PHY Control Register

Address: 3079_0000h base + 0h offset = 3079_0000h



DDR_PHY_PHY_CON0 field descriptions

Field	Description
31–24 Reserved	This field is reserved. Initial Value = 8'h17
23–22 CTRL_UPD_ MODE	It controls when DLL is updated. Initial Value = 2'b01 2'b00 Update always

Table continues on the next page...

DDR_PHY_PHY_CON0 field descriptions (continued)

Field	Description
	2'b01 To update depending on "ctrl_flock" 2'b10 To update depending on "ctrl_flock" 2'b11 Do not update DLL
21–20 CTRL_UPD_ RANGE	It decides how many differences between the new lock value and the current lock value which is used in Slave-DLL is needed for updating lock value. Initial Value = 2'b00 2'b00 Update when difference is greater than 0 2'b01 Update when difference is greater than 1 2'b10 Update when difference is greater than 7 2'b11 Update when difference is greater than 15
19–17 Reserved	This field is reserved. Initial Value = 3'b001
16 WRLVL_MODE	Write Leveling Mode Enable Initial Value = 1'b0
15–13 Reserved	This field is reserved. Initial Value = 1'b0
12–11 CTRL_DDR_ MODE	Initial Value = 2'b11 2’b00 Reserved 2’b01 DDR3 2’b10 LPDDR2 2’b11 LPDDR3
10 Reserved	This field is reserved. Initial Value = 1'b1
9 CTRL_DFDQS	Initial Value = 1'b1 1’b0 Single-ended DQS — 1’b1 Differential DQS
8 CTRL_SHGATE	This field controls the gate control signal Initial Value = 1'b0 1’b0 Gate signal length = (burst length / 2) + N (DQS Pull-Down mode, ctrl_pulld_dqs[3:0] == 4'b1111, N = 0,1,2…) 1'b1 Gate signal length = (burst length / 2) – 1
7 Reserved	This field is reserved. Initial Value = 1'b0
6 CTRL_ATGATE	This bit should be set to 1’b1. Initial Value = 1'b1
5 Reserved	This field is reserved. Initial Value = 1'b0
4 CTRL_ CMOSRCV	This field controls the input mode of I/O. Initial Value = 1'b0

Table continues on the next page...

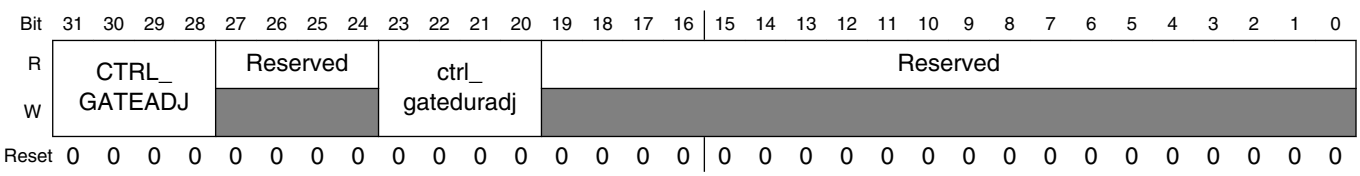
DDR_PHY_PHY_CON0 field descriptions (continued)

Field	Description
3 CTRL_TWPRE	Write preamble setting Initial Value = 1'b0
CTRL_FNC_FB	Valid only when {mode_phy, mode_nand, mode_scan, mode_mux} is 4'b00000. Initial Value = 3'b000 For normal operation • 3'b000: Normal operation mode. For ATE test purpose • 3'b010: External FNC read feedback test mode. • 3'b011: Internal FNC read feedback test mode. For Board test purpose • 3’b100: External PHY read feedback test mode. When memory is not attached on the board • 3’b101: Internal PHY read feedback test mode. mode_highz should be set. • 3’b110: Internal PHY write feedback test mode. mode_highz should be set.

9.3.4.2 DDR_PHY_PHY_CON1

PHY Control Register

Address: 3079_0000h base + 4h offset = 3079_0004h



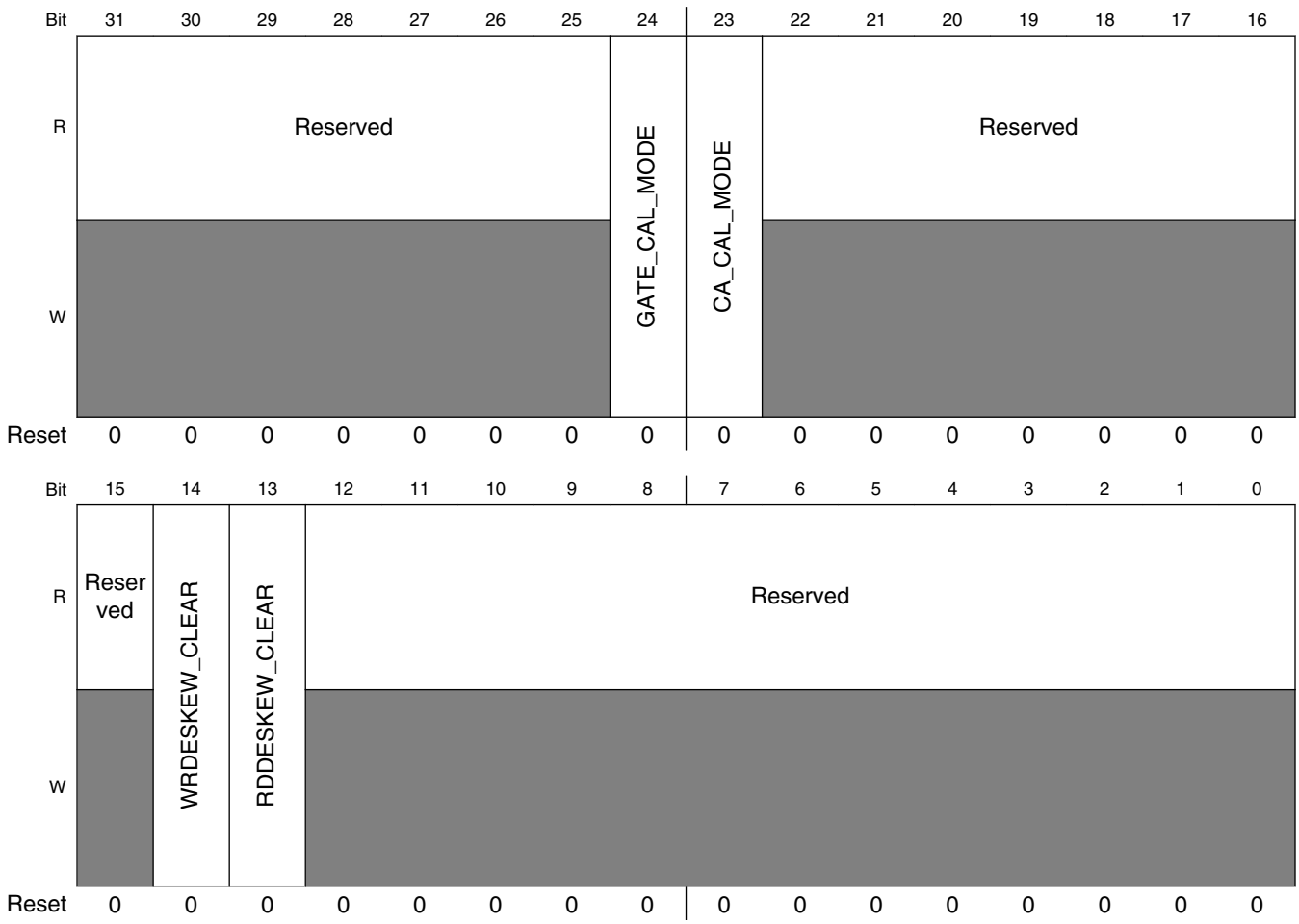
DDR_PHY_PHY_CON1 field descriptions

Field	Description
31–28 CTRL_GATEADJ	It adjusts the enable time of "ctrl_gate" on a clock cycle base. MSB (bit3) controls direction: 1'b1 (subtract delay), 1'b0 (add delay), Bit[2:0] set delay value. Initial Value = 4'h2
27–24 Reserved	This field is reserved. Initial Value = 4’h0
23–20 ctrl_gateduradj	It adjusts the duration cycle of "ctrl_gate" on a clock cycle base. MSB (bit3) controls direction: 1'b1 (subtract delay), 1'b0 (add delay), Bit[2:0] set delay value. Initial Value = 4’h2
Reserved	This field is reserved. Initial Value = 32'h10100

9.3.4.3 DDR_PHY_PHY_CON2

PHY Control Register

Address: 3079_0000h base + 8h offset = 3079_0008h



DDR_PHY_PHY_CON2 field descriptions

Field	Description
31–25 Reserved	This field is reserved. Initial Value = 7'b0
24 GATE_CAL_MODE	When gate_cal_mode = 1, Gate leveling offset value will be used instead of ctrl_shiftc*. If gate leveling is used, this value should be high during operation. Initial Value = 1'b0
23 CA_CAL_MODE	Not used Initial Value = 1'b0
22–15 Reserved	This field is reserved. Initial Value = 8'b02

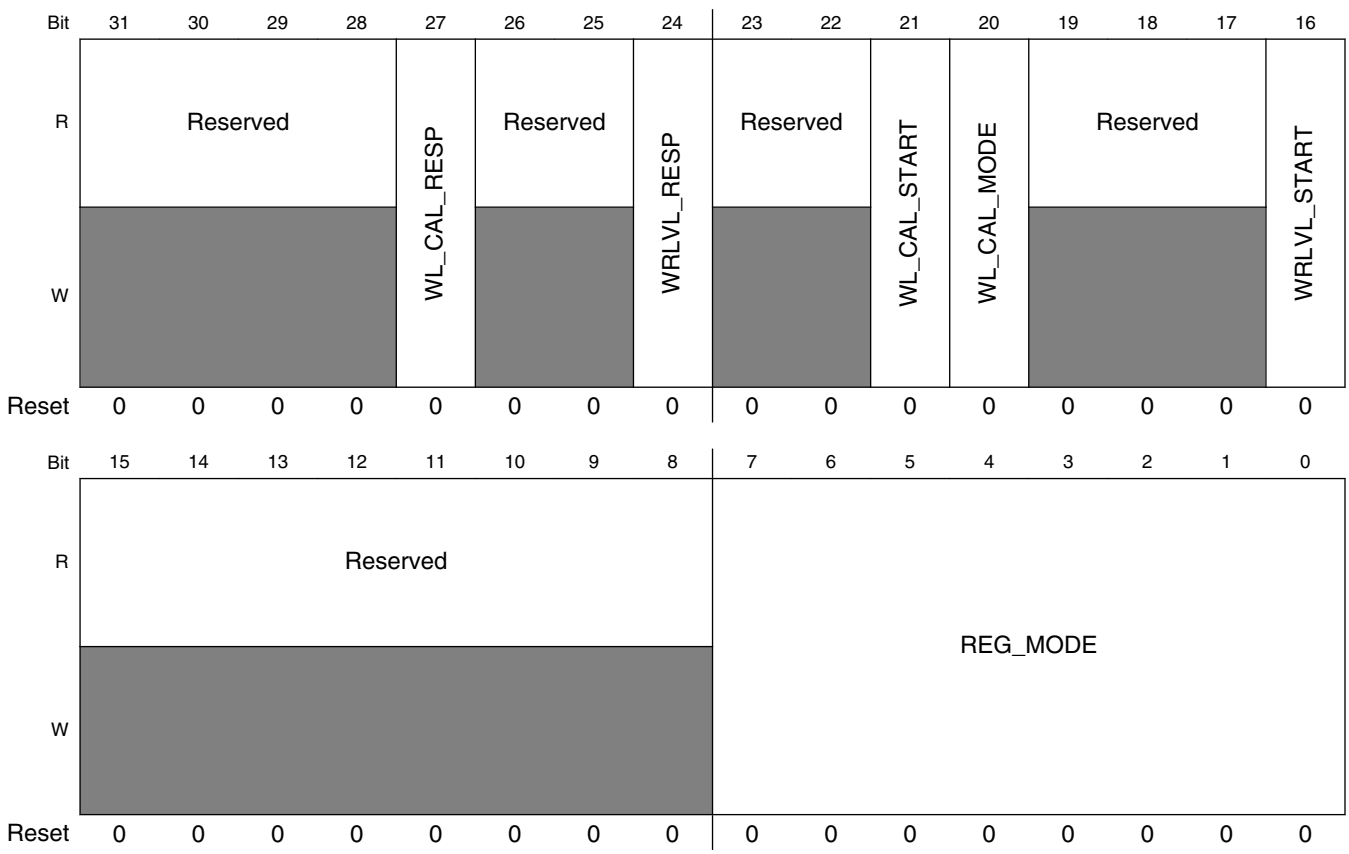
Table continues on the next page...

DDR_PHY_PHY_CON2 field descriptions (continued)

Field	Description
14 WRDESKEW_CLEAR	Write "ctrl_offsetw*" toWrDeSkewCode Initial Value = 1'b0
13 RDDESKEW_CLEAR	Clear "ctrl_offsetr*" to RdDeSkewCode Initial Value = 1'b0
Reserved	This field is reserved. Initial Value = 13'h4

9.3.4.4 DDR_PHY_PHY_CON3

Address: 3079_0000h base + Ch offset = 3079_000Ch



DDR_PHY_PHY_CON3 field descriptions

Field	Description
31–28 Reserved	This field is reserved.
27 WL_CAL_RESP	Response after Write Leveling Calibration Initial Value = 0x0

Table continues on the next page...

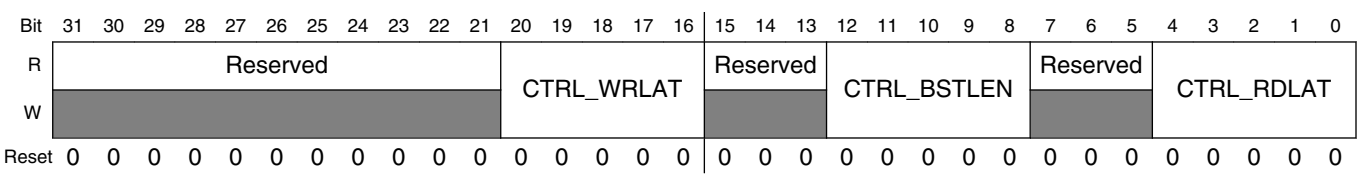
DDR_PHY_PHY_CON3 field descriptions (continued)

Field	Description
26–25 Reserved	This field is reserved.
24 WRLVL_RESP	Response after Write Leveling Initial Value = 0x0
23–22 Reserved	This field is reserved.
21 WL_CAL_START	Start Write Leveling Calibration Initial Value = 0x0
20 WL_CAL_MODE	Write Leveling Calibration mode Enable Initial Value = 0x0
19–17 Reserved	This field is reserved.
16 WRLVL_START	Start Write Leveling signal. It can be enabled when wrlvl_mode = 1. It should be disabled after Write Leveling Response is asserted. Initial Value = 0x0
15–8 Reserved	This field is reserved.
REG_MODE	Register mode control to write the information at each data Initial Value = 0x0

9.3.4.5 DDR_PHY_PHY_CON4

PHY Control Register

Address: 3079_0000h base + 10h offset = 3079_0010h



DDR_PHY_PHY_CON4 field descriptions

Field	Description
31–21 Reserved	This field is reserved.
20–16 CTRL_WRLAT	Clock cycles between write command and the first edge of DQS which can capture the first valid DQ. For example, if WL is 6, It should be set as 7(= WL + 1) in LPDDR3, 6(= WL) in DDR3. Initial Value = 5'h8
15–13 Reserved	This field is reserved.

Table continues on the next page...

DDR_PHY_PHY_CON4 field descriptions (continued)

Field	Description
12–8 CTRL_BSTLEN	Burst Length (BL) Initial Value = 5'h0
7–5 Reserved	This field is reserved.
CTRL_RDLAT	Read Latency (RL) Initial Value = 5'h0

9.3.4.6 DDR_PHY_PHY_CON5

NOTE

When "wl_cal_mode" = 1'b1, "PHY_CON5" will show the results of HW Write Latency Calibration. Please do not change "PHY_CON5" during "wl_cal_mode" = 1'b1.

Address: 3079_0000h base + 14h offset = 3079_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																				CTRL_WRLAT_PLUS3			CTRL_WRLAT_PLUS2			CTRL_WRLAT_PLUS1_1			CTRL_WRLAT_PLUS1_0		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_PHY_CON5 field descriptions

Field	Description
31–12 Reserved	This field is reserved.
11–9 CTRL_WRLAT_PLUS3	This field can control Write Latency (WL) by half cycle, one or two cycles for Data_Slice3. Initial Value = 3'h0 0 Write Latency Increases by half clock cycle when enabled. 1 Write Latency Increases by one clock cycle when enabled. 2 Write Latency Increases by wo clock cycle when enabled.
8–6 CTRL_WRLAT_PLUS2	This field can control Write Latency (WL) by half cycle, one or two cycles for Data_Slice2. Initial Value = 3'h0 0 Write Latency Increases by half clock cycle when enabled. 1 Write Latency Increases by one clock cycle when enabled. 2 Write Latency Increases by wo clock cycle when enabled.

Table continues on the next page...

DDR_PHY_PHY_CON5 field descriptions (continued)

Field	Description
5–3 CTRL_WRLAT_PLUS1_1	This field can control Write Latency (WL) by half cycle, one or two cycles for Data_Slice1. Initial Value = 3'h0 0 Write Latency Increases by half clock cycle when enabled. 1 Write Latency Increases by one clock cycle when enabled. 2 Write Latency Increases by wo clock cycle when enabled.
CTRL_WRLAT_PLUS1_0	This field can control Write Latency (WL) by half cycle, one or two cycles for Data_Slice0. Initial Value = 3'h0 0 Write Latency Increases by half clock cycle when enabled. 1 Write Latency Increases by one clock cycle when enabled. 2 Write Latency Increases by wo clock cycle when enabled.

9.3.4.7 DDR_PHY_LP_CON0

Address: 3079_0000h base + 18h offset = 3079_0018h

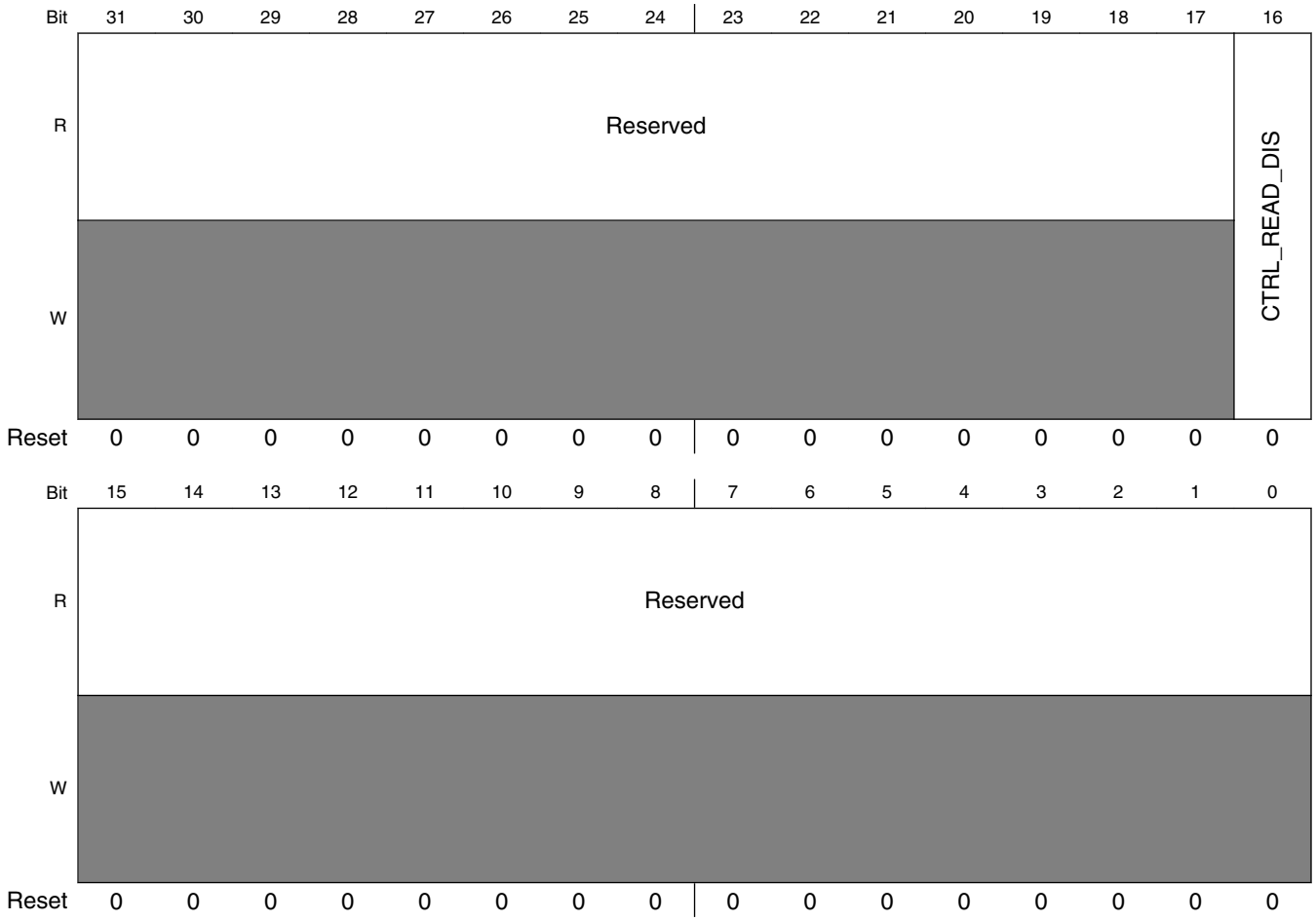
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved							CTRL_PULLD_DQ									Reserved							CTRL_PULLD_DQS								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_LP_CON0 field descriptions

Field	Description
31–25 Reserved	This field is reserved.
24–16 CTRL_PULLD_DQ	Active HIGH signal to down DQ signals. For normal operation this field should be zero. When bus is idle, it can be set to pull-down or up the bus not to make bus high-z state. Initial Value = 4'h0
15–9 Reserved	This field is reserved.
CTRL_PULLD_DQS	Active HIGH signal to pull-up or down PDQS / NDQS signals. When using Gate Leveling in DDR3, this field can be zero. When bus is idle, it can be set to pull-down or up the bus not to make bus high-z state (PDQS / NDQS is pulled-down / up). For LPDDR2 / LPDDR3 mode, this field should be always set for normal operation to make P / NDQS signals pull-down / up. Initial Value = 4'h0

9.3.4.8 DDR_PHY_RODT_CON0

Address: 3079_0000h base + 1Ch offset = 3079_001Ch



DDR_PHY_RODT_CON0 field descriptions

Field	Description
31–17 Reserved	This field is reserved. Initial Value = 15'h8
16 CTRL_READ_DIS	Read ODT (On-Die-Termination) Disable Signal. This signal should be one in LPDDR2 or LPDDR3 mode. Initial Value = 4'h0 1'b1 Drive ctrl_read_p* to 0 (If using LPDDR2 or LPDDR3, ctrl_read_p* will be always 0 by enabling this field). 1'b0 Drive ctrl_read_p* normally.
Reserved	This field is reserved. Initial Value = 16'h0

9.3.4.9 DDR_PHY_OFFSET_RD_CON0

To capture the DQs with DQS in read operation, DQS is shifted by 90° and this makes rising edge of DQS be located in the center of valid window of DQs. But according to the channel condition, i.e. PCB or SSN, rising edge of DQS could be biased from the center of valid window of DQs. ctrl_offset* is used to compensate the biased DQS and make it be placed in the center of DQs. If ctrl_offset*[7] is 0, ctrl_offset*[6:0] x tFS (tFS: fine step delay) is added to the 90° delay amount and if ctrl_offset*[7] is 1, ctrl_offset*[6:0] x tFS is subtracted from the 90° delay amount. ctrl_offset*[6:0] means the number of delay cells of the "delay line". Generally, ctrl_offset*[6:0] would be zero.

dfi_ctrlupd_req signal is required to update delay line control signals before the first normal operation after checking whether dfi_init_complete is set and at every memory refresh cycle. Before set and clear this signal during initialization, dfi_init_complete signal should be checked whether it's HIGH to confirm that DLL is locked.

Be careful that "ctrl_offsetr*" can be used for the other purpose (= Read Deskew Code Register). Please refer to mode (= PHY_CON3[7:0]).

Address: 3079_0000h base + 20h offset = 3079_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTRL_OFFSETR3								CTRL_OFFSETR2								CTRL_OFFSETR1								CTRL_OFFSETR0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_OFFSET_RD_CON0 field descriptions

Field	Description
31–24 CTRL_ OFFSETR3	This field can be used to give offset to read DQS. If this field is fixed, this should not be changed during operation. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. The right-shifted value is limited by the quarter of the maximum delay in Master Delay Line. Read DQS offset amount: ctrl_offsetr3[7] = 1: (tFS: fine step delay) Read DQS 90° delay amount – ctrl_offsetr0[6:0] x tFS ctrl_offsetr3[7] = 0: Read DQS 90° delay amount + ctrl_offsetr0[6:0] x tFS Initial Value = 0x8
23–16 CTRL_ OFFSETR2	This field can be used to give offset to read DQS. If this field is fixed, this should not be changed during operation. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. The right-shifted value is limited by the quarter of the maximum delay in Master Delay Line. Read DQS offset amount:

Table continues on the next page...

DDR_PHY_OFFSET_RD_CON0 field descriptions (continued)

Field	Description
	ctrl_offsetr2[7] = 1: (tFS: fine step delay) Read DQS 90° delay amount – ctrl_offsetr0[6:0] x tFS ctrl_offsetr2[7] = 0: Read DQS 90° delay amount + ctrl_offsetr0[6:0] x tFS Initial Value = 0x8
15–8 CTRL_ OFFSETR1	This field can be used to give offset to read DQS. If this field is fixed, this should not be changed during operation. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. The right-shifted value is limited by the quarter of the maximum delay in Master Delay Line. Read DQS offset amount: ctrl_offsetr1[7] = 1: (tFS: fine step delay) Read DQS 90° delay amount – ctrl_offsetr0[6:0] x tFS ctrl_offsetr1[7] = 0: Read DQS 90° delay amount + ctrl_offsetr0[6:0] x tFS Initial Value = 0x8
CTRL_ OFFSETR0	This field can be used to give offset to read DQS. If this field is fixed, this should not be changed during operation. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. The right-shifted value is limited by the quarter of the maximum delay in Master Delay Line. Read DQS offset amount: ctrl_offsetr0[7] = 1: (tFS: fine step delay) Read DQS 90° delay amount – ctrl_offsetr0[6:0] x tFS ctrl_offsetr0[7] = 0: Read DQS 90° delay amount + ctrl_offsetr0[6:0] x tFS Initial Value = 0x8

9.3.4.10 DDR_PHY_OFFSET_WR_CON0

Be careful that "ctrl_offsetw*" can be used for the other purpose (= Write Deskew Code Register). Please refer to reg_mode (= PHY_CON3[7:0]).

Address: 3079_0000h base + 30h offset = 3079_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTRL_OFFSETW3								CTRL_OFFSETW2								CTRL_OFFSETW1								CTRL_OFFSETW0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_OFFSET_WR_CON0 field descriptions

Field	Description
31–24 CTRL_ OFFSETW3	This field can be used to give offset to write DQ. If this field is fixed, this should not be changed during operation. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. The right-shifted value is limited by the quarter of the maximum delay in Master Delay Line. Write DQ offset amount: ctrl_offsetw2[7] = 1: (tFS: fine step delay) Write DQ 270° delay amount – ctrl_offsetw2[6:0] x tFS ctrl_offsetr2[7] = 0: Write DQ 270° delay amount + ctrl_offsetw2[6:0] x tFS Initial Value = 0x8
23–16 CTRL_ OFFSETW2	This field can be used to give offset to write DQ. If this field is fixed, this should not be changed during operation. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. The right-shifted value is limited by the quarter of the maximum delay in Master Delay Line. Write DQ offset amount: ctrl_offsetw2[7] = 1: (tFS: fine step delay) Write DQ 270° delay amount – ctrl_offsetw2[6:0] x tFS ctrl_offsetr2[7] = 0: Write DQ 270° delay amount + ctrl_offsetw2[6:0] x tFS Initial Value = 0x8
15–8 CTRL_ OFFSETW1	This field can be used to give offset to write DQ. If this field is fixed, this should not be changed during operation. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. The right-shifted value is limited by the quarter of the maximum delay in Master Delay Line. Write DQ offset amount: ctrl_offsetw2[7] = 1: (tFS: fine step delay) Write DQ 270° delay amount – ctrl_offsetw2[6:0] x tFS ctrl_offsetr2[7] = 0: Write DQ 270° delay amount + ctrl_offsetw2[6:0] x tFS Initial Value = 0x8
CTRL_ OFFSETW0	This field can be used to give offset to write DQ. If this field is fixed, this should not be changed during operation. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. The right-shifted value is limited by the quarter of the maximum delay in Master Delay Line. Write DQ offset amount: ctrl_offsetw2[7] = 1: (tFS: fine step delay) Write DQ 270° delay amount – ctrl_offsetw2[6:0] x tFS ctrl_offsetr2[7] = 0: Write DQ 270° delay amount + ctrl_offsetw2[6:0] x tFS

Table continues on the next page...

DDR_PHY_OFFSET_WR_CON0 field descriptions (continued)

Field	Description
	Initial Value = 0x8

9.3.4.11 DDR_PHY_GATE_CODE_CON0

GATE Control Register

Address: 3079_0000h base + 40h offset = 3079_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTRL_OFFSETC3								CTRL_OFFSETC2								CTRL_OFFSETC1								CTRL_OFFSETC0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_GATE_CODE_CON0 field descriptions

Field	Description
31–24 CTRL_ OFFSETC3	Gate offset amount for DDR3 Data Slice 3. Do not change this register value after the PHY has been initialized. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. ctrl_offsetc [7] = 1 : (tFS: fine step delay) GATEout delay amount – ctrl_offsetc [6:0] x tFS ctrl_offsetc [7] = 0 : GATEout delay amount + ctrl_offsetc [6:0] x tFS Initial Value = 0x0
23–16 CTRL_ OFFSETC2	Gate offset amount for DDR3 Data Slice 2. Do not change this register value after the PHY has been initialized. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. ctrl_offsetc [7] = 1 : (tFS: fine step delay) GATEout delay amount – ctrl_offsetc [6:0] x tFS ctrl_offsetc [7] = 0 : GATEout delay amount + ctrl_offsetc [6:0] x tFS Initial Value = 0x0
15–8 CTRL_ OFFSETC1	Gate offset amount for DDR3 Data Slice 1. Do not change this register value after the PHY has been initialized. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. ctrl_offsetc [7] = 1 : (tFS: fine step delay) GATEout delay amount – ctrl_offsetc [6:0] x tFS ctrl_offsetc [7] = 0 : GATEout delay amount + ctrl_offsetc [6:0] x tFS Initial Value = 0x0
CTRL_ OFFSETC0	Gate offset amount for DDR3 Data Slice 0. Do not change this register value after the PHY has been initialized. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. ctrl_offsetc [7] = 1 : (tFS: fine step delay)

Table continues on the next page...

DDR_PHY_GATE_CODE_CON0 field descriptions (continued)

Field	Description
	GATEout delay amount – ctrl_offsetc [6:0] x tFS ctrl_offsetc [7] = 0 : GATEout delay amount + ctrl_offsetc [6:0] x tFS Initial Value = 0x0

9.3.4.12 DDR_PHY_SHIFTC_CON0

Address: 3079_0000h base + 4Ch offset = 3079_004Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																				CTRL_ SHIFTC3			CTRL_ SHIFTC2			CTRL_ SHIFTC1			CTRL_ SHIFTC0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_SHIFTC_CON0 field descriptions

Field	Description
31–12 Reserved	This field is reserved. Initial Value = 3'b010
11–9 CTRL_SHIFTC3	GATEin signal delay amount for DDR. If this field is fixed, this should not be changed during operation. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. This value is limited by the half of the maximum delay in Master Delay Line. GATEin signal will be delayed by T/16 whenever it increases one. Initial Value = 3'b010 000 0 (0° shift) 001 T (365° shift) 010 T / 2 (180° shift) 011 T / 4 (90° shift) 100 T / 8 (45° shift) 101 T / 16 (22.5° shift)
8–6 CTRL_SHIFTC2	GATEin signal delay amount for DDR. If this field is fixed, this should not be changed during operation. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. This value is limited by the half of the maximum delay in Master Delay Line. GATEin signal will be delayed by T/16 whenever it increases one. Initial Value = 3'b010 000 0 (0° shift) 001 T (365° shift) 010 T / 2 (180° shift) 011 T / 4 (90° shift) 100 T / 8 (45° shift) 101 T / 16 (22.5° shift)
5–3 CTRL_SHIFTC1	GATEin signal delay amount for DDR. If this field is fixed, this should not be changed during operation. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. This value is limited by the half of

Table continues on the next page...

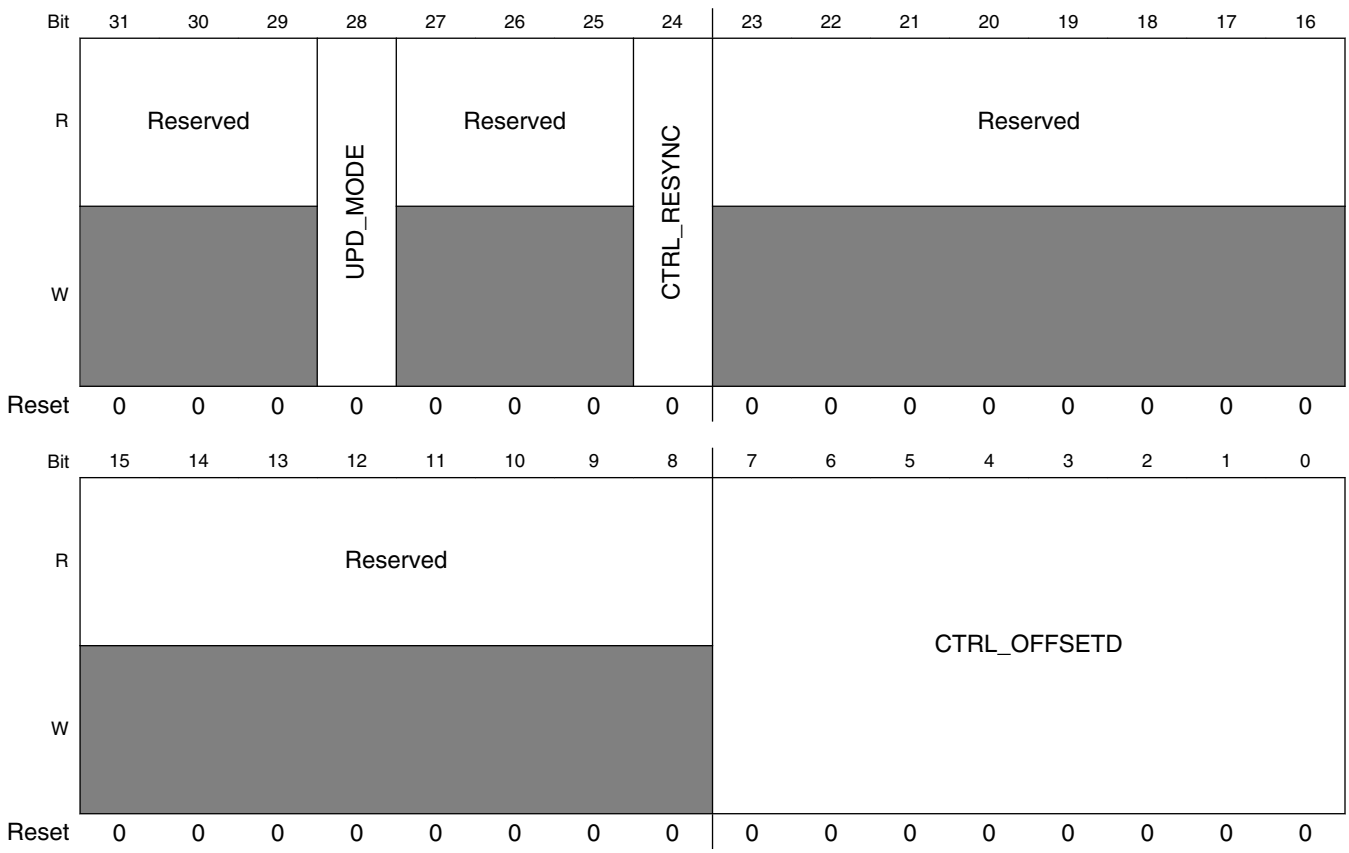
DDR_PHY_SHIFTC_CON0 field descriptions (continued)

Field	Description
	the maximum delay in Master Delay Line. GATEin signal will be delayed by T/16 whenever it increases one. Initial Value = 3'b010 000 0 (0° shift) 001 T (365° shift) 010 T / 2 (180° shift) 011 T / 4 (90° shift) 100 T / 8 (45° shift) 101 T / 16 (22.5° shift)
CTRL_SHIFTC0	GATEin signal delay amount for DDR. If this field is fixed, this should not be changed during operation. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. This value is limited by the half of the maximum delay in Master Delay Line. GATEin signal will be delayed by T/16 whenever it increases one. Initial Value = 3'b010 000 0 (0° shift) 001 T (365° shift) 010 T / 2 (180° shift) 011 T / 4 (90° shift) 100 T / 8 (45° shift) 101 T / 16 (22.5° shift)

9.3.4.13 DDR_PHY_CMD_SDLL_CON0

CMD SDLL Control Register

Address: 3079_0000h base + 50h offset = 3079_0050h



DDR_PHY_CMD_SDLL_CON0 field descriptions

Field	Description
31–29 Reserved	This field is reserved.
28 UPD_MODE	This field controls "PHY Update" Mode. Initial Value = 0x1 1'b1 MC-Initiated Update Mode 1'b0 PHY-Initiated Update Mode
27–25 Reserved	This field is reserved.
24 CTRL_RESYNC	Active RISIG-EDGE signal. This signal should become LOW after set HIGH for normal operation.When this bit transits from LOW to HIGH, pointers of FIFO and DLL information is updated. In general, this bit should be set during initialization and refresh cycles. Refer to "DLL Code Update" to use ctrl_resync.

Table continues on the next page...

DDR_PHY_CMD_SDLL_CON0 field descriptions (continued)

Field	Description
	Initial Value = 0x0
23–8 Reserved	This field is reserved.
CTRL_OFFSETD	This field is for debug purpose. (For LPDDR2) If this field is fixed, this should not be changed during operation. This value is valid only after dfi_ctrlupd_req becomes HIGH and LOW. The right-shifted value is limited by the quarter of the maximum delay in Master Delay Line. offset amount for 270° clock generation: ctrl_offsetd[7] = 1 : (tFS: fine step delay) 270° delay amount – ctrl_offsetd[6:0] x tFS ctrl_offsetd[7] = 0 : 270° delay amount + ctrl_offsetd[6:0] x tFS Initial Value = 0x8

9.3.4.14 DDR_PHY_LVL_CON0

Address: 3079_0000h base + 6Ch offset = 3079_006Ch

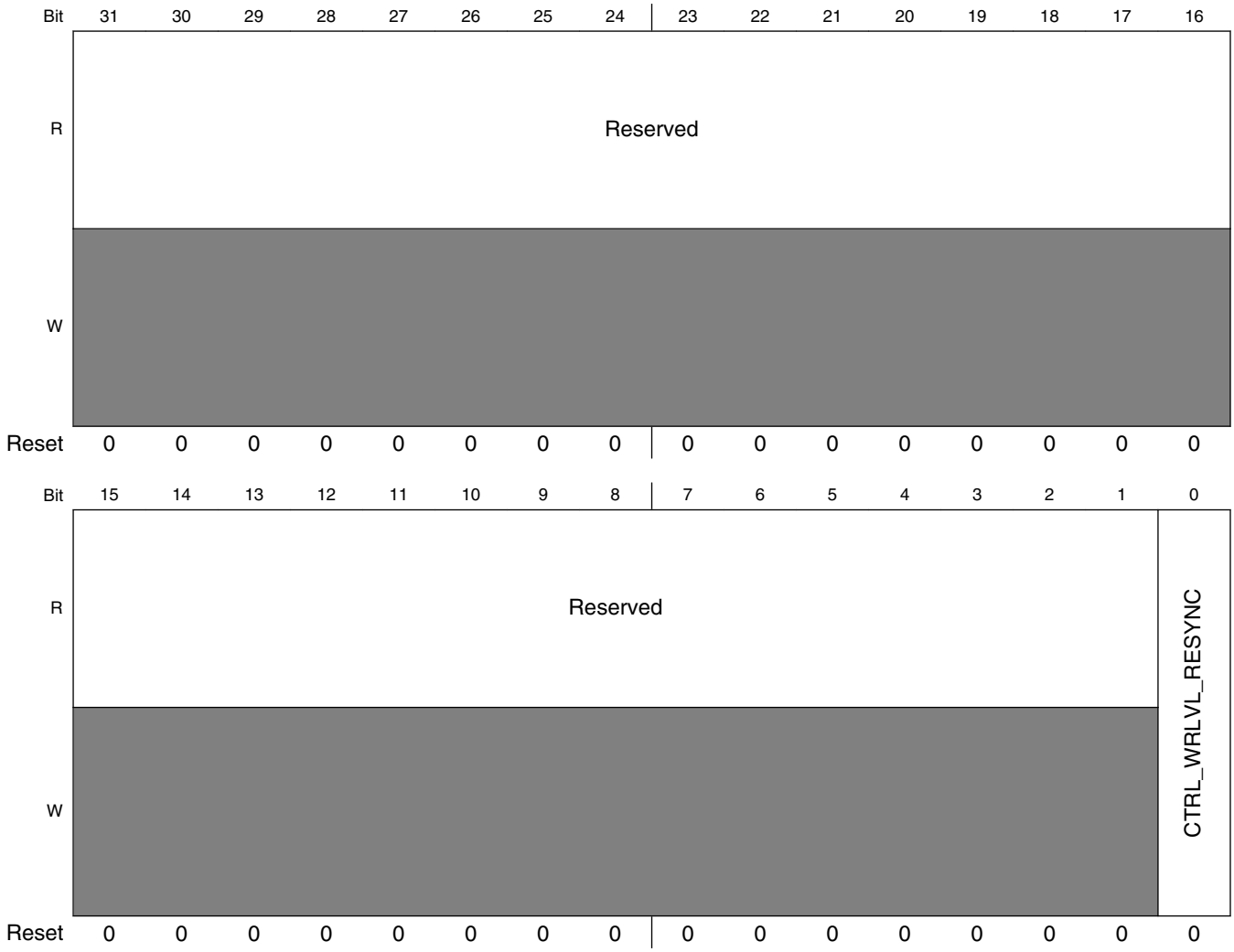
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ctrl_wrlvl3_code								ctrl_wrlvl2_code								ctrl_wrlvl1_code								ctrl_wrlvl0_code							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_LVL_CON0 field descriptions

Field	Description
31–24 ctrl_wrlvl3_code	Write Level Slave DLL Code Value for Data_Slice 3 Initial Value = 0x0
23–16 ctrl_wrlvl2_code	Write Level Slave DLL Code Value for Data_Slice 2 Initial Value = 0x0
15–8 ctrl_wrlvl1_code	Write Level Slave DLL Code Value for Data_Slice 1 Initial Value = 0x0
ctrl_wrlvl0_code	Write Level Slave DLL Code Value for Data_Slice 0 Initial Value = 0x0

9.3.4.15 DDR_PHY_LVL_CON3

Address: 3079_0000h base + 78h offset = 3079_0078h



DDR_PHY_LVL_CON3 field descriptions

Field	Description
31–1 Reserved	This field is reserved.
0 CTRL_WRLVL_RESYNC	Write Level DLL Code Update Enable Initial Value = 0x0

9.3.4.16 DDR_PHY_CMD_DESKEW_CON0

CMD DESKEW Control Register

Address: 3079_0000h base + 7Ch offset = 3079_007Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CA3DESKEWCODE								CA2DESKEWCODE								CA1DESKEWCODE_1								CA1DESKEWCODE_0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_CMD_DESKEW_CON0 field descriptions

Field	Description
31–24 CA3DESKEWCODE	DeSkew Code for CA[3] Initial Value = 0x0
23–16 CA2DESKEWCODE	DeSkew Code for CA[2] Initial Value = 0x0
15–8 CA1DESKEWCODE_1	DeSkew Code for CA[1] Initial Value = 0x0
CA1DESKEWCODE_0	DeSkew Code for CA[0] Initial Value = 0x0

9.3.4.17 DDR_PHY_CMD_DESKEW_CON1

CMD DESKEW Control Register

Address: 3079_0000h base + 80h offset = 3079_0080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CA7DESKEWCODE								CA6DESKEWCODE								CA5DESKEWCODE								CA4DESKEWCODE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_CMD_DESKEW_CON1 field descriptions

Field	Description
31–24 CA7DESKEWCODE	DeSkew Code for CA[7] Initial Value = 0x0
23–16 CA6DESKEWCODE	DeSkew Code for CA[6] Initial Value = 0x0

Table continues on the next page...

DDR_PHY_CMD_DESKEW_CON1 field descriptions (continued)

Field	Description
15–8 CA5DESKEWCODE	DeSkew Code for CA[5] Initial Value = 0x0
CA4DESKEWCODE	DeSkew Code for CA[4] Initial Value = 0x0

9.3.4.18 DDR_PHY_CMD_DESKEW_CON2

CMD DESKEW Control Register

Address: 3079_0000h base + 84h offset = 3079_0084h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CS0DESKEWCODE								CKDESKEWCODE								CA9DESKEWCODE								CA8DESKEWCODE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

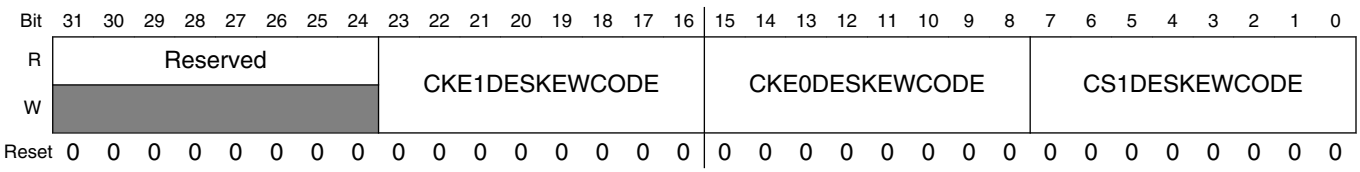
DDR_PHY_CMD_DESKEW_CON2 field descriptions

Field	Description
31–24 CS0DESKEWCODE	DeSkew Code for CS0 Initial Value = 0x0
23–16 CKDESKEWCODE	DeSkew Code for CK Initial Value = 0x0
15–8 CA9DESKEWCODE	DeSkew Code for CA[9] Initial Value = 0x0
CA8DESKEWCODE	DeSkew Code for CA[8] Initial Value = 0x0

9.3.4.19 DDR_PHY_CMD_DESKEW_CON3

If the DeSkew Code for CS[1:0], CK is changed, CKE should be always "LOW" during updating. If the DeSkew Code for CKE[1:0] is changed, Please initialize memory again.

Address: 3079_0000h base + 88h offset = 3079_0088h



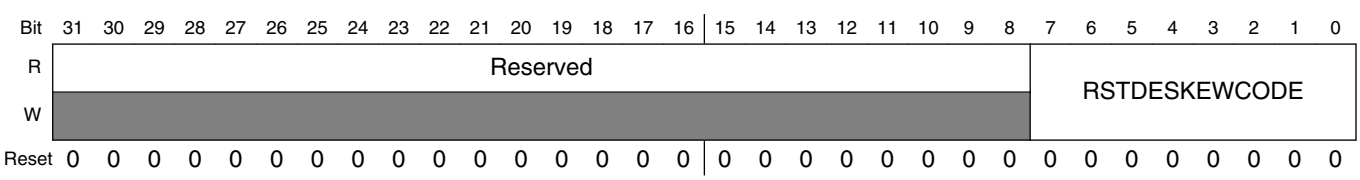
DDR_PHY_CMD_DESKEW_CON3 field descriptions

Field	Description
31–24 Reserved	This field is reserved.
23–16 CKE1DESKEWCODE	DeSkew Code for CKE1 Initial Value = 0x0
15–8 CKE0DESKEWCODE	DeSkew Code for CKE0 Initial Value = 0x0
CS1DESKEWCODE	DeSkew Code for CS1 Initial Value = 0x0

9.3.4.20 DDR_PHY_CMD_DESKEW_CON4

CMD DESKEW Control Register

Address: 3079_0000h base + 94h offset = 3079_0094h



DDR_PHY_CMD_DESKEW_CON4 field descriptions

Field	Description
31–8 Reserved	This field is reserved.

Table continues on the next page...

DDR_PHY_CMD_DESKEW_CON4 field descriptions (continued)

Field	Description
RSTDESKEWCODE	DeSkew Code for RST Initial Value = 0x0

9.3.4.21 DDR_PHY_DRVDS_CON0

Drive Strength Settings:

- 3'b000 : 240Ω Impedance output driver
- 3'b001 : 120Ω Impedance output driver
- 3'b010 : 80Ω Impedance output driver
- 3'b011 : 60Ω Impedance output driver
- 3'b100 : 48Ω Impedance output driver
- 3'b101 : 40Ω Impedance output driver
- 3'b110 : 34Ω Impedance output driver
- 3'b111 : 30Ω Impedance output driver

NOTE

It is recommended that Driver Strength will be one of the following settings instead of 3'h0.

- 3'b100 : 48Ω Impedance output driver
- 3'b101 : 40Ω Impedance output driver
- 3'b110 : 34Ω Impedance output driver
- 3'b111 : 30Ω Impedance output driver

Address: 3079_0000h base + 9Ch offset = 3079_009Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				CACKDRVRDS			CACKEDRVRDS			CACSDRVRDS			CAADRDRVRDS		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_DRVDS_CON0 field descriptions

Field	Description
31–12 Reserved	This field is reserved.
11–9 CACKDRVRDS	Driver strenght for CK Initial value = 0x0. See table above for register setting values.

Table continues on the next page...

DDR_PHY_DRVDS_CON0 field descriptions (continued)

Field	Description
8–6 CACKEDRVRDS	Driver strength for CKE[1:0] Initial value = 0x0
5–3 CACSDRVRDS	Driver strength for CS[1:0] Initial value = 0x0
CAADRDRVRDS	Driver strength for CA[9:0], RAS, CAS,WEN, ODT[1:0], RESET, BANK[2:0]. Initial value = 0x0

9.3.4.22 DDR_PHY_MDLL_CON0

MDLL Control Register

Address: 3079_0000h base + B0h offset = 3079_00B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved	CTRL_START_POINT							Reserved	CTRL_INC						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTRL_FORCE								CTRL_START	CTRL_DLL_ON	CTRL_REF					Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_MDLL_CON0 field descriptions

Field	Description
31 Reserved	This field is reserved. Initial Value = 1'b0
30–24 CTRL_START_POINT	Initial DLL lock start point. This is the number of delay cells and is the start point where "DLL" start tracing to be locked. Initial delay time is calculated by multiplying the unit delay of delay cell and this value. Initial Value = 7'h10
23 Reserved	This field is reserved. Initial Value = 1'b0
22–16 CTRL_INC	Increase amount of start point Initial Value = 7'h10
15–7 CTRL_FORCE	This field is used instead of ctrl_lock_value[8:0] found by the DLL only when ctrl_dll_on is LOW, i.e. If the DLL is off, this field is used to generate 270° clock and shift DQS by 90°. Initial Value = 9'h00
6 CTRL_START	This field is used to start DLL locking. Initial Value = 1'b1
5 CTRL_DLL_ON	HIGH active start signal to turn on the DLL. This signal should be kept HIGH for normal operation. If this signal becomes LOW, DLL is turned off. This bit should be kept set before ctrl_start is set to turn on the DLL.HIGH active start signal to turn on the DLL. This signal should be kept HIGH for normal operation. If this signal becomes LOW, DLL is turned off. This bit should be kept set before ctrl_start is set to turn on the DLL. Initial Value = 1'b1
4–1 CTRL_REF	This field determines the period of time when ctrl_locked is cleared. Initial Value = 4'h8 4'b0000 Do not use. 4'b0001 ctrl_flock is de-asserted during 16 clock cycles, ctrl_locked is deasserted. 4'b0010 ctrl_flock is de-asserted during 24 clock cycles, ctrl_locked is deasserted. 4'b1110 ctrl_flock is de-asserted during 120 clock cycles, ctrl_locked is deasserted. 4'b1111 Once ctrl_locked and dfi_init_complete are asserted, those won't be deasserted until rst_n is asserted.
0 Reserved	This field is reserved. Initial Value = 1'b0

9.3.4.23 DDR_PHY_MDLL_CON1

"DLL" is used to get how many delay cells should be passed through the "delay line" to delay a signal to an amount of one clock period and is controlled by ctrl_start, ctrl_start_point and ctrl_inc.

ctrl_start should be set and kept high to make "DLL" keep tracing one clock period after clock(PHY clock) becomes stable. If ctrl_start becomes LOW, "DLL" stops tracing one clock period.

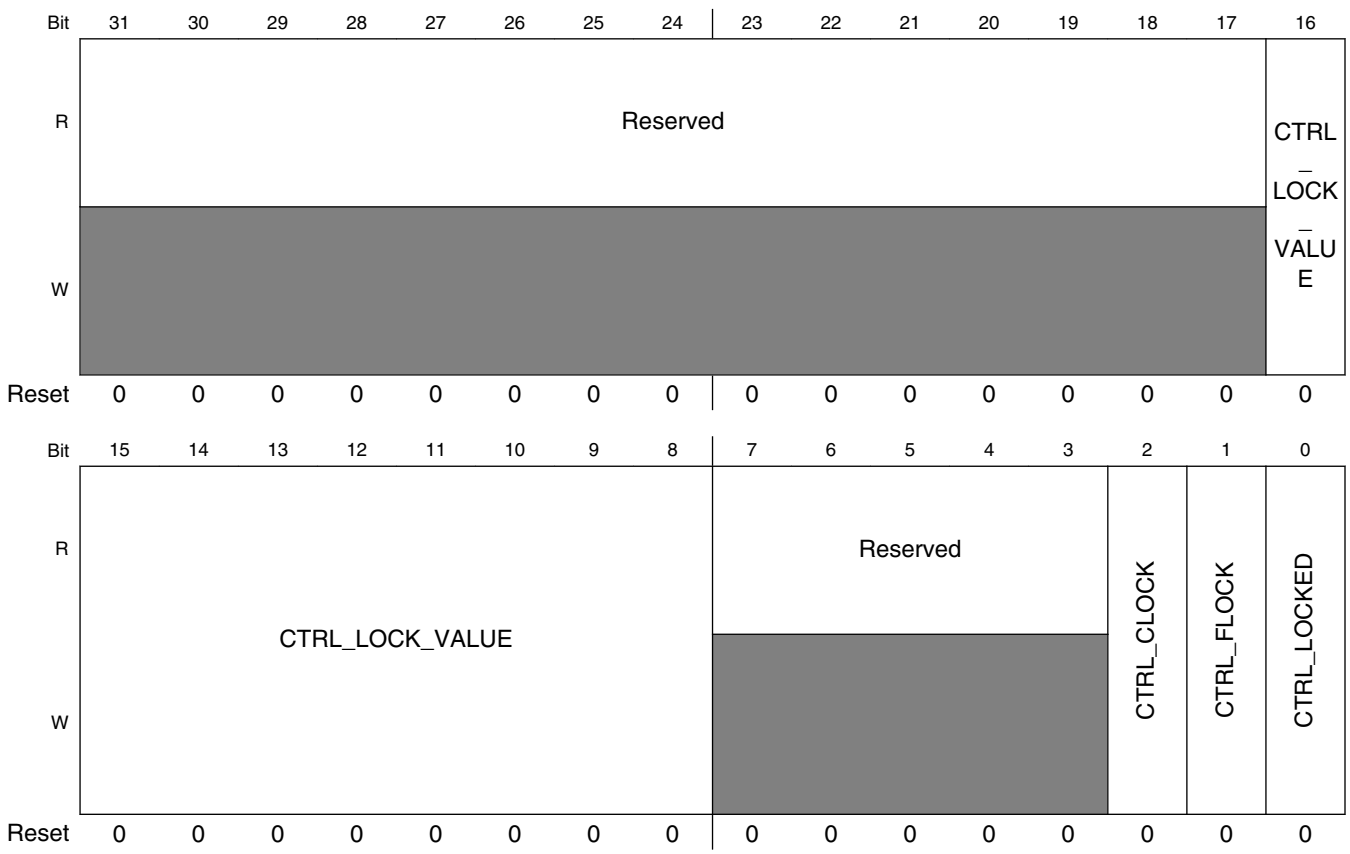
ctrl_clock and ctrl_flock are status fields indicating whether "DLL" is locked. After ctrl_start becomes HIGH, DLL starts tracing one clock period and controls (increases or decreases) the number of delay cells for the clock to pass through. And if ctrl_clock is set("DLL" is locked), "DLL" changes step delays of the "delay line" and controls the "delay line" in fine resolution to reduce the "phase offset error".When ctrl_flock is set, "DLL" is locked with fine resolution. ctrl_lock_value is information field to indicate the number of delay cells to delay a signal to one clock period through the "delay line".

- {ctrl_clock, ctrl_flock = 2'b00} : DLL is not locked.
- {ctrl_clock, ctrl_flock = 2'b01} : Impossible value.
- {ctrl_clock, ctrl_flock = 2'b10} : Locked.
- {ctrl_clock, ctrl_flock = 2'b11} : Locked.

NOTE

Recommended value for ctrl_start_point[7:0] and ctrl_inc[7:0] are 8'h10.

Address: 3079_0000h base + B4h offset = 3079_00B4h



DDR_PHY_MDLL_CON1 field descriptions

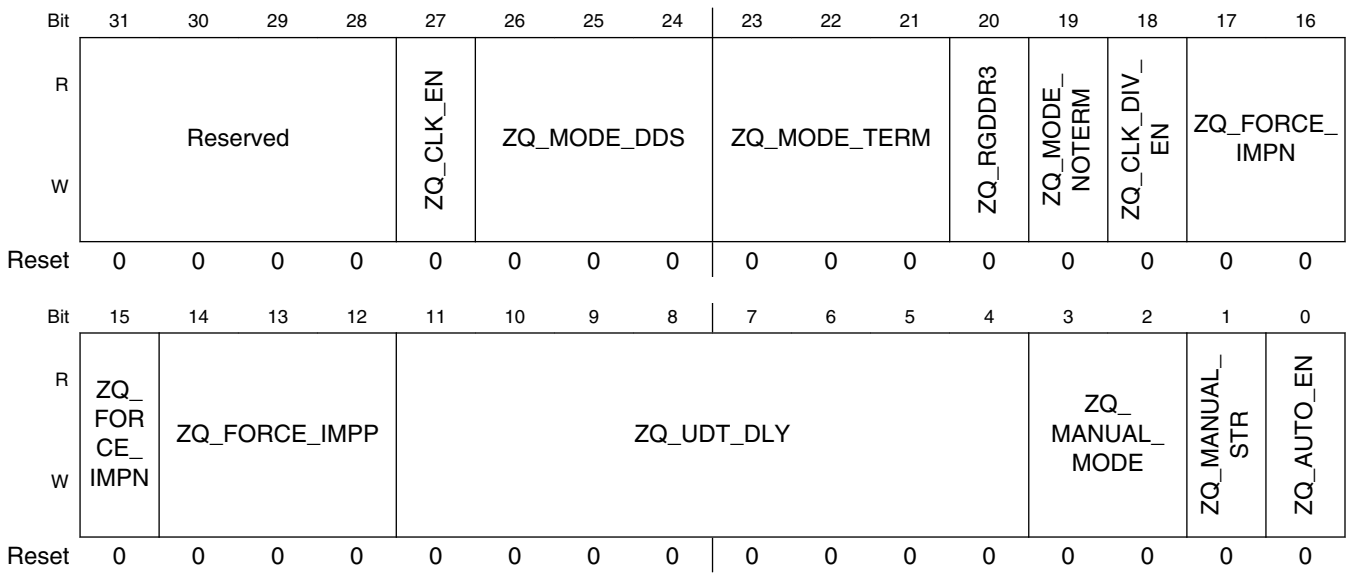
Field	Description
31–17 Reserved	This field is reserved.
16–8 CTRL_LOCK_VALUE	Locked delay line encoding value. Defines the number of Fine Steps to be used for one clock period. From ctrl_lock_value[8:0], tFS (fine step delay) can be calculated. tFS = tCK / ctrl_lock_value[8:0].
7–3 Reserved	This field is reserved.
2 CTRL_CLOCK	Coarse lock information. According to clock jitter, ctrl_clock can be de-asserted.
1 CTRL_FLOCK	Fine lock information. According to clock jitter, ctrl_flock can be de-asserted.
0 CTRL_LOCKED	DLL stable lock information. This field is set after ctrl_flock is set. This field is cleared when ctrl_flock is de-asserted for some period of time specified by ctrl_ref[3:0] value. This field is required for stable lock status check.

9.3.4.24 DDR_PHY_ZQ_CON0

NOTE

"zq_manual_str" (= ZQ_CON0[1]) should be toggled after ZQ_CON0[17:2] or ZQ_CON0[26:19] is changed. For example, if zq_mode_dds is written by 3'b111, "zq_manual_str" should be set and cleared to apply this new strength value (3'b111).

Address: 3079_0000h base + C0h offset = 3079_00C0h



DDR_PHY_ZQ_CON0 field descriptions

Field	Description
31–28 Reserved	This field is reserved. Initial Value = 0x0
27 ZQ_CLK_EN	ZQ I/O Clock enable Initial Value = 1'b1
26–24 ZQ_MODE_DDS	Driver strength selection. Initial Value = 3'h7 000 240 Ω Impedance output driver 001 120 Ω Impedance output driver 010 80 Ω Impedance output driver 011 60 Ω Impedance output driver 100 48 Ω Impedance output driver 101 40 Ω Impedance output driver

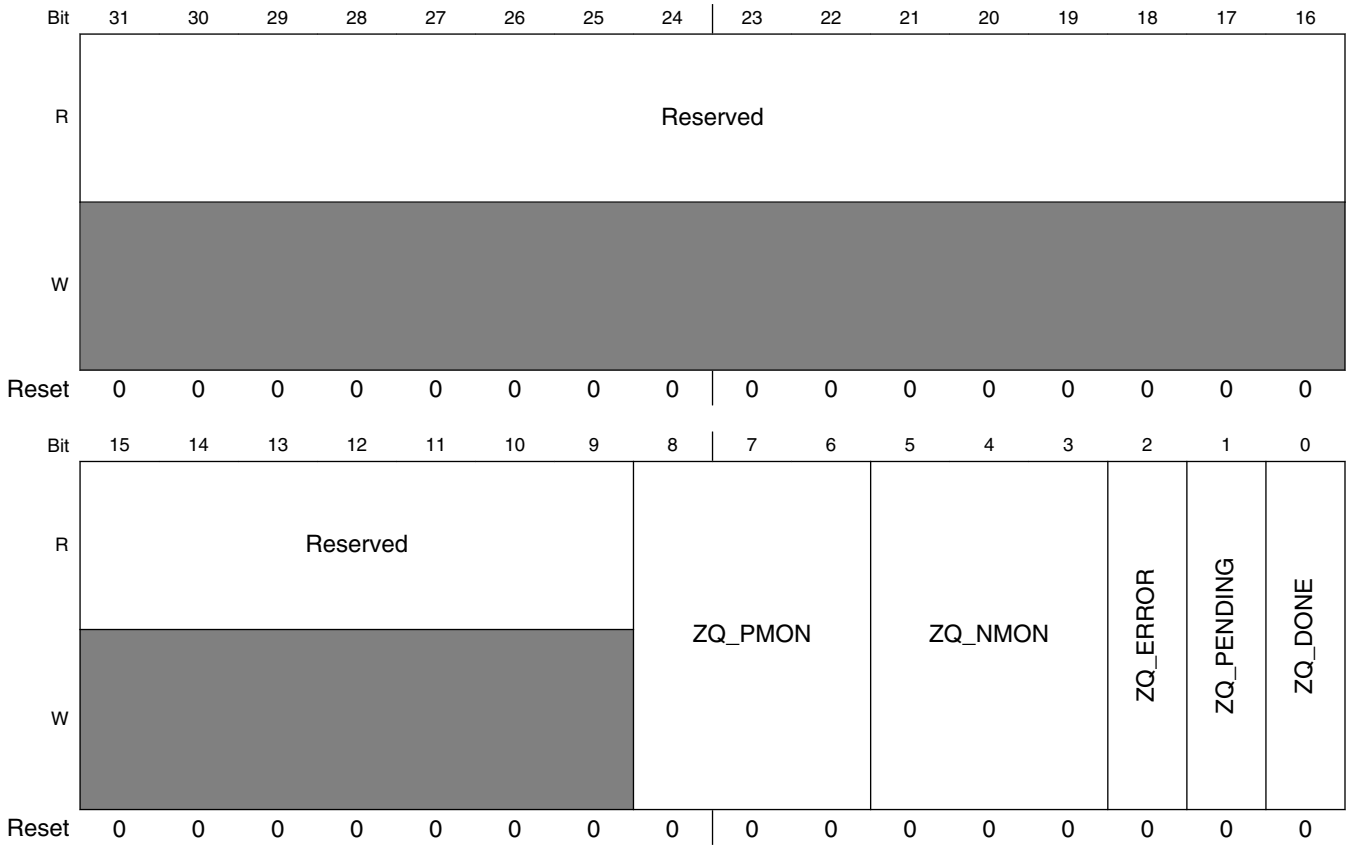
Table continues on the next page...

DDR_PHY_ZQ_CON0 field descriptions (continued)

Field	Description
	110 34 Ω Impedance output driver 111 30 Ω Impedance output driver
23–21 ZQ_MODE_TERM	On-die-termination (ODT) resistor value selection. Initial Value = 3'h0 3'b001 120 Ω Receiver termination 3'b010 60 Ω Receiver termination 3'b011 40 Ω Receiver termination 3'b100 30 Ω Receiver termination
20 ZQ_RGDDR3	Not used Initial Value = 1'b0
19 ZQ_MODE_NOTERM	Termination disable selection Initial Value = 1'b0 1 Termination disable 0 Termination enable
18 ZQ_CLK_DIV_EN	Clock dividing enable Initial Value = 1'b0
17–15 ZQ_FORCE_IMPN	Immediate control code for pull-down. Initial Value = 3'h0
14–12 ZQ_FORCE_IMPP	Immediate control code for pull-up. Initial Value = 3'h7
11–4 ZQ_UDT_DLY	ZQ I/O clock enable duration for auto calibration mode. Initial Value = 8'h30
3–2 ZQ_MANUAL_MODE	Manual calibration mode selection Initial Value = 2'b01 2'b00 Force calibration 2'b01 Long calibration 2'b10 Short calibration
1 ZQ_MANUAL_STR	Manual calibration start Initial Value = 1'b0
0 ZQ_AUTO_EN	Auto calibration enable Initial Value = 1'b0

9.3.4.25 DDR_PHY_ZQ_CON1

Address: 3079_0000h base + C4h offset = 3079_00C4h

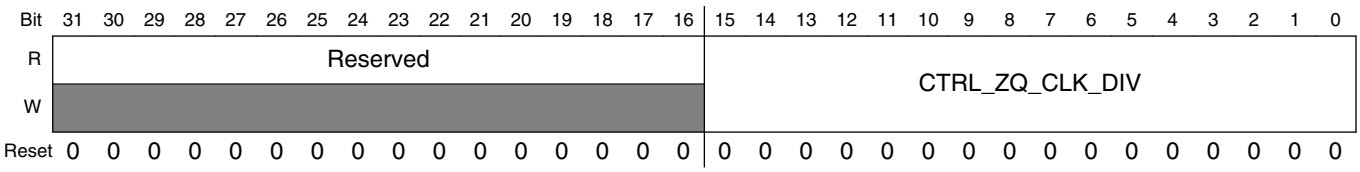


DDR_PHY_ZQ_CON1 field descriptions

Field	Description
31–9 Reserved	This field is reserved.
8–6 ZQ_PMON	Control code found by auto calibration for pull-up.
5–3 ZQ_NMON	Control code found by auto calibration for pull-down.
2 ZQ_ERROR	Calibration fail indication (High: calibration failed)
1 ZQ_PENDING	Auto calibration enable status
0 ZQ_DONE	ZQ Calibration is finished. Initial Value = 1'b0

9.3.4.26 DDR_PHY_ZQ_CON2

Address: 3079_0000h base + C8h offset = 3079_00C8h



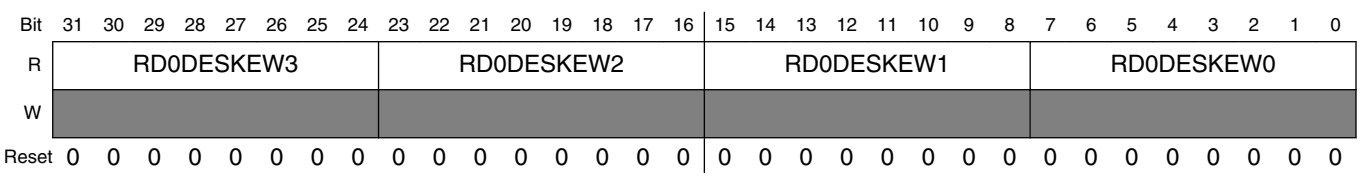
DDR_PHY_ZQ_CON2 field descriptions

Field	Description
31–16 Reserved	This field is reserved.
CTRL_ZQ_CLK_DIV	ZQ Clock (= io_zq_clk) divider setting value. The frequency will be the following formula. "io_zq_clk" (MHz) = clk2x (MHz) / ((ctrl_zq_clk_div + 1) * 4)

9.3.4.27 DDR_PHY_RD_DESKEW_CON0

RD DESKEW Control Register

Address: 3079_0000h base + 190h offset = 3079_0190h



DDR_PHY_RD_DESKEW_CON0 field descriptions

Field	Description
31–24 RD0DESKEW3	Read DQ24 De-Skew Code
23–16 RD0DESKEW2	Read DQ16 De-Skew Code
15–8 RD0DESKEW1	Read DQ8 De-Skew Code
RD0DESKEW0	Read DQ0 De-Skew Code

9.3.4.28 DDR_PHY_RD_DESKEW_CON3

Address: 3079_0000h base + 19Ch offset = 3079_019Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RD1DESKEW3								RD1DESKEW2								RD1DESKEW1								RD1DESKEW0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_RD_DESKEW_CON3 field descriptions

Field	Description
31–24 RD1DESKEW3	Read DQ25 De-Skew Code
23–16 RD1DESKEW2	Read DQ17 De-Skew Code
15–8 RD1DESKEW1	Read DQ9 De-Skew Code
RD1DESKEW0	Read DQ1 De-Skew Code

9.3.4.29 DDR_PHY_RD_DESKEW_CON6

Address: 3079_0000h base + 1A8h offset = 3079_01A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RD2DESKEW3								RD2DESKEW2								RD2DESKEW1								RD2DESKEW0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_RD_DESKEW_CON6 field descriptions

Field	Description
31–24 RD2DESKEW3	Read DQ26 De-Skew Code
23–16 RD2DESKEW2	Read DQ18 De-Skew Code
15–8 RD2DESKEW1	Read DQ10 De-Skew Code
RD2DESKEW0	Read DQ2 De-Skew Code

9.3.4.30 DDR_PHY_RD_DESKEW_CON9

Address: 3079_0000h base + 1B4h offset = 3079_01B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RD3DESKEW3								RD3DESKEW2								RD3DESKEW1								RD3DESKEW0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_RD_DESKEW_CON9 field descriptions

Field	Description
31–24 RD3DESKEW3	Read DQ27 De-Skew Code
23–16 RD3DESKEW2	Read DQ19 De-Skew Code
15–8 RD3DESKEW1	Read DQ11 De-Skew Code
RD3DESKEW0	Read DQ3 De-Skew Code

9.3.4.31 DDR_PHY_RD_DESKEW_CON12

Address: 3079_0000h base + 1C0h offset = 3079_01C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RD4DESKEW3								RD4DESKEW2								RD4DESKEW1								RD4DESKEW0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_RD_DESKEW_CON12 field descriptions

Field	Description
31–24 RD4DESKEW3	Read DQ28 De-Skew Code
23–16 RD4DESKEW2	Read DQ20 De-Skew Code
15–8 RD4DESKEW1	Read DQ12 De-Skew Code
RD4DESKEW0	Read DQ4 De-Skew Code

9.3.4.32 DDR_PHY_RD_DESKEW_CON15

Address: 3079_0000h base + 1CCh offset = 3079_01CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RD5DESKEW3								RD5DESKEW2								RD5DESKEW1								RD5DESKEW0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_RD_DESKEW_CON15 field descriptions

Field	Description
31–24 RD5DESKEW3	Read DQ29 De-Skew Code
23–16 RD5DESKEW2	Read DQ21 De-Skew Code
15–8 RD5DESKEW1	Read DQ13 De-Skew Code
RD5DESKEW0	Read DQ5 De-Skew Code

9.3.4.33 DDR_PHY_RD_DESKEW_CON18

Address: 3079_0000h base + 1D8h offset = 3079_01D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RD6DESKEW3								RD6DESKEW2								RD6DESKEW1								RD6DESKEW0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_RD_DESKEW_CON18 field descriptions

Field	Description
31–24 RD6DESKEW3	Read DQ30 De-Skew Code
23–16 RD6DESKEW2	Read DQ22 De-Skew Code
15–8 RD6DESKEW1	Read DQ14 De-Skew Code
RD6DESKEW0	Read DQ6 De-Skew Code

9.3.4.34 DDR_PHY_RD_DESKEW_CON21

Address: 3079_0000h base + 1E4h offset = 3079_01E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RD7DESKEW3								RD7DESKEW2								RD7DESKEW1								RD7DESKEW0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_RD_DESKEW_CON21 field descriptions

Field	Description
31–24 RD7DESKEW3	Read DQ31 De-Skew Code
23–16 RD7DESKEW2	Read DQ23 De-Skew Code
15–8 RD7DESKEW1	Read DQ15 De-Skew Code
RD7DESKEW0	Read DQ7 De-Skew Code

9.3.4.35 DDR_PHY_WR_DESKEW_CON0

WR DESKEW Control Register

Address: 3079_0000h base + 1F0h offset = 3079_01F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WR0DESKEW3								WR0DESKEW2								WR0DESKEW1								WR0DESKEW0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_WR_DESKEW_CON0 field descriptions

Field	Description
31–24 WR0DESKEW3	Wirte DQ24 De-Skew Code
23–16 WR0DESKEW2	Write DQ16 De-Skew Code
15–8 WR0DESKEW1	Write DQ8 De-Skew Code
WR0DESKEW0	Write DQ0 De-Skew Code

9.3.4.36 DDR_PHY_WR_DESKEW_CON3

Address: 3079_0000h base + 1FCh offset = 3079_01FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WR1DESKEW3								WR1DESKEW2								WR1DESKEW1								WR1DESKEW0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_WR_DESKEW_CON3 field descriptions

Field	Description
31–24 WR1DESKEW3	Write DQ25 De-Skew Code
23–16 WR1DESKEW2	Write DQ17 De-Skew Code
15–8 WR1DESKEW1	Write DQ9 De-Skew Code
WR1DESKEW0	Write DQ1 De-Skew Code

9.3.4.37 DDR_PHY_WR_DESKEW_CON6

Address: 3079_0000h base + 208h offset = 3079_0208h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WR2DESKEW3								WR2DESKEW2								WR2DESKEW1								WR2DESKEW0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_WR_DESKEW_CON6 field descriptions

Field	Description
31–24 WR2DESKEW3	Write DQ2 De-Skew Code for Data Slice3.
23–16 WR2DESKEW2	Write DQ2 De-Skew Code for Data Slice2.
15–8 WR2DESKEW1	Write DQ2 De-Skew Code for Data Slice1.
WR2DESKEW0	Write DQ2 De-Skew Code for Data Slice0.

9.3.4.38 DDR_PHY_WR_DESKEW_CON9

Address: 3079_0000h base + 214h offset = 3079_0214h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WR3DESKEW3								WR3DESKEW2								WR3DESKEW1								WR3DESKEW0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_WR_DESKEW_CON9 field descriptions

Field	Description
31–24 WR3DESKEW3	Write DQ3 De-Skew Code for Data Slice3.
23–16 WR3DESKEW2	Write DQ3 De-Skew Code for Data Slice2.
15–8 WR3DESKEW1	Write DQ3 De-Skew Code for Data Slice1.
WR3DESKEW0	Write DQ3 De-Skew Code for Data Slice0.

9.3.4.39 DDR_PHY_WR_DESKEW_CON12

Address: 3079_0000h base + 220h offset = 3079_0220h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WR4DESKEW3								WR4DESKEW2								WR4DESKEW1								WR4DESKEW0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_WR_DESKEW_CON12 field descriptions

Field	Description
31–24 WR4DESKEW3	Write DQ4 De-Skew Code for Data Slice3.
23–16 WR4DESKEW2	Write DQ4 De-Skew Code for Data Slice2.
15–8 WR4DESKEW1	Write DQ4 De-Skew Code for Data Slice1.
WR4DESKEW0	Write DQ4 De-Skew Code for Data Slice0.

9.3.4.40 DDR_PHY_WR_DESKEW_CON15

Address: 3079_0000h base + 22Ch offset = 3079_022Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WR5DESKEW3								WR5DESKEW2								WR5DESKEW1								WR5DESKEW0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_WR_DESKEW_CON15 field descriptions

Field	Description
31–24 WR5DESKEW3	Write DQ5 De-Skew Code for Data Slice3.
23–16 WR5DESKEW2	Write DQ5 De-Skew Code for Data Slice2.
15–8 WR5DESKEW1	Write DQ5 De-Skew Code for Data Slice1.
WR5DESKEW0	Write DQ5 De-Skew Code for Data Slice0.

9.3.4.41 DDR_PHY_WR_DESKEW_CON18

Address: 3079_0000h base + 238h offset = 3079_0238h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WR6DESKEW3								WR6DESKEW2								WR6DESKEW1								RD6DESKEW0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_WR_DESKEW_CON18 field descriptions

Field	Description
31–24 WR6DESKEW3	Write DQ6 De-Skew Code for Data Slice3.
23–16 WR6DESKEW2	Write DQ6 De-Skew Code for Data Slice2.
15–8 WR6DESKEW1	Write DQ6 De-Skew Code for Data Slice1.
RD6DESKEW0	Read DQ6 De-Skew Code for Data Slice0.

9.3.4.42 DDR_PHY_WR_DESKEW_CON21

Address: 3079_0000h base + 244h offset = 3079_0244h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WR7DESKEW3								WR7DESKEW2								WR7DESKEW1								WR7DESKEW0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_WR_DESKEW_CON21 field descriptions

Field	Description
31–24 WR7DESKEW3	Write DQ7 De-Skew Code for Data Slice3.
23–16 WR7DESKEW2	Write DQ7 De-Skew Code for Data Slice2.
15–8 WR7DESKEW1	Write DQ7 De-Skew Code for Data Slice1.
WR7DESKEW0	Write DQ7 De-Skew Code for Data Slice0.

9.3.4.43 DDR_PHY_DM_DESKEW_CON

Address: 3079_0000h base + 250h offset = 3079_0250h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMDESKEW3								DMDESKEW2								DMDESKEW1								DMDESKEW0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_DM_DESKEW_CON field descriptions

Field	Description
31–24 DMDESKEW3	Write DM De-Skew Code for Data Slice3.
23–16 DMDESKEW2	Write DM De-Skew Code for Data Slice2.
15–8 DMDESKEW1	Write DM De-Skew Code for Data Slice1.
DMDESKEW0	Write DM De-Skew Code for Data Slice0.

9.3.4.44 DDR_PHY_RDATA0

Address: 3079_0000h base + 3A0h offset = 3079_03A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DQ_IO_RD3								DQ_IO_RD2								DQ_IO_RD1								DQ_IO_RD0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_RDATA0 field descriptions

Field	Description
31–24 DQ_IO_RD3	DQ I/O Read Data for DS3
23–16 DQ_IO_RD2	DQ I/O Read Data for DS2
15–8 DQ_IO_RD1	DQ I/O Read Data for DS1
DQ_IO_RD0	DQ I/O Read Data for DS0

9.3.4.45 DDR_PHY_STAT0

Address: 3079_0000h base + 3ACh offset = 3079_03ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VERSION_INFO																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PHY_STAT0 field descriptions

Field	Description
VERSION_INFO	Version Information

9.4 AHB-to-APBH Bridge with DMA (APBH-Bridge-DMA)

9.4.1 Overview

The AHB-to-APBH bridge provides the chip with an inexpensive peripheral attachment bus running on the AHB's HCLK. (The H in APBH denotes that the APB*H* is synchronous to HCLK , as compared to APB*X*, which runs on the crystal-derived XCLK.)

As shown in the figure below, the AHB-to-APBH bridge includes the AHB-to-APB PIO bridge for a memory-mapped I/O to the APB devices, as well as a central DMA facility for devices on this bus and a vectored interrupt controller for the ARM Cortex-A8 core. Each one of the APB peripherals, including the vectored interrupt controller, is documented in their respective chapters.

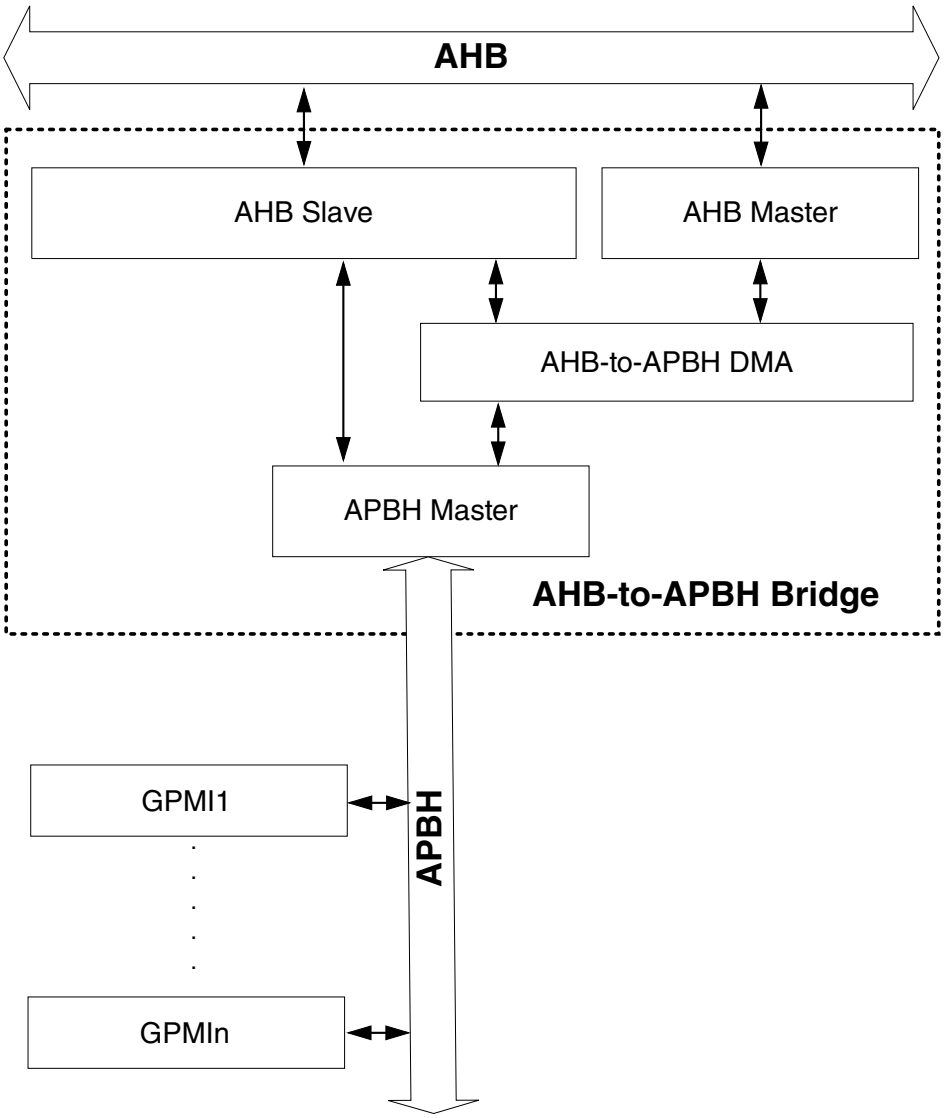


Figure 9-25. AHB-to-APBH Bridge DMA Block Diagram

The DMA controller uses the APBH bus to transfer read and write data to and from each peripheral. There is no separate DMA bus for these devices. Contention between the DMA's use of the APBH bus and the AHB-to-APB bridge functions' use of the APBH is mediated by an internal arbitration logic. For contention between these two units, the DMA is favored and the AHB slave will report "not ready" through its HREADY output until the bridge transfer can complete. The arbiter tracks repeated lockouts and inverts the priority, guaranteeing the ARM platform every fourth transfer on the APB.

9.4.2 Clocks

The table found here describes the clock sources for APBH. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 9-15. APBH Clocks

Clock name	Clock Root	Description
hclk	usdhc_clk_root	Module clock

9.4.3 APBH DMA

The DMA supports four sixteen channels of DMA services, as shown in the following table. The shared DMA resource allows each independent channel to follow a simple chained command list. Command chains are built up using the general structure, as shown in Figure 9-26.

Table 9-16. APBH DMA channel assignments

APBH DMA Channel #	Usage
0	SSP0GPMI0
1	SSP1GPMI1
2	SSP2GPMI2
3	SSP3GPMI3
4	GPMI0GPMI4
5	GPMI1GPMI5
6	GPMI2GPMI6
7	GPMI3GPMI7
8	GPMI4Reserved
9	GPMI5Unused
10	GPMI6Unused
11	GPMI7Unused
12	HSADCUnused
13	LCDIFUnused
14	Unused
15	Unused

A single command structure or channel command word specifies a number of operations to be performed by the DMA in support of a given device. Thus, the ARM platform can set up large units of work, chaining together many DMA channel command words, pass them off to the DMA, and have no further concern for the device until the DMA completion interrupt occurs. The goal is to have enough intelligence in the DMA and the devices to keep the interrupt frequency from any device below 1 KHz (arrival intervals longer than 1 ms).

A single command structure can issue 32-bit PIO write operations to key registers in the associated device using the same APB bus and controls that it uses to write DMA data bytes to the device. For example, this allows a chain of operations to be issued to the GPMI controller to send NAND command bytes, address bytes, and data transfers where the command and the address structure is completely under software control, but the administration of that transfer is handled autonomously by the DMA. Each DMA structure can have 0–15 PIO words appended to it. The CMDPIOWORDS field, if non-zero, instructs the DMA engine to copy these words to the APB, beginning at the first register address offset for the peripheral and and incrementing the register offset each cycle.

The DMA master generates only normal read/write transfers to the APBH. It does *not* generate set, clear, or toggle (SCT) transfers.

After any requested PIO words have been transferred to the peripheral, the DMA examines the two-bit command field in the channel command structure. Table 9-17 shows the four commands implemented by the DMA.

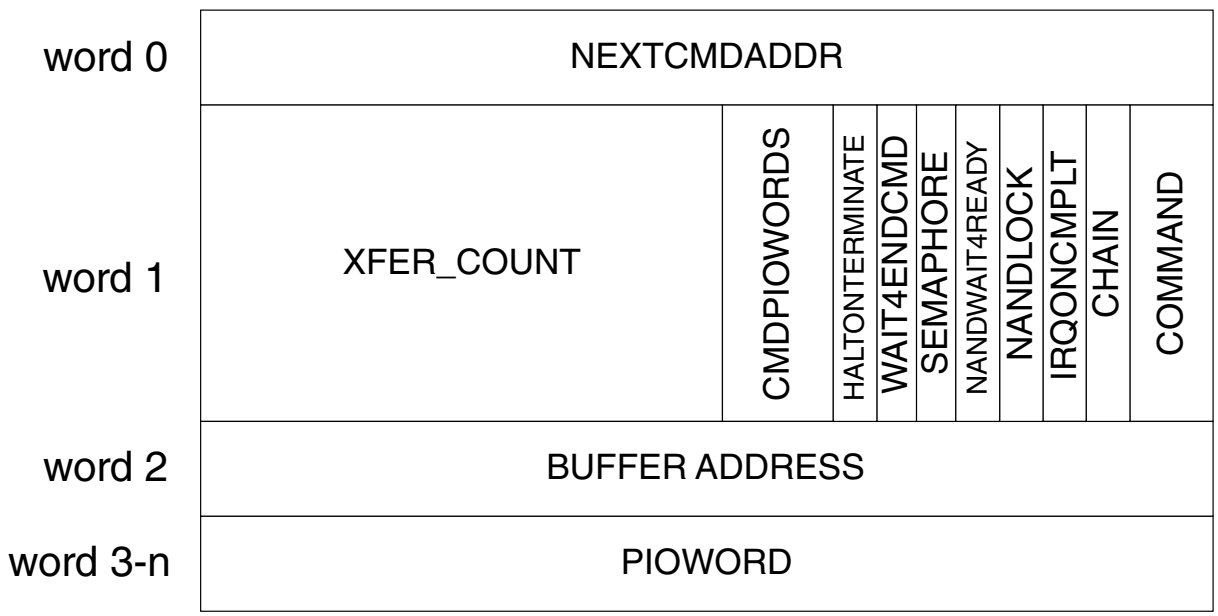


Figure 9-26. AHB-to-APBH Bridge DMA channel command structure

Table 9-17. APBH DMA commands

DMA Command	Usage
00	NO_DMA_XFER. Perform any requested PIO word transfers, but terminate the command before any DMA transfer.
01	DMA_WRITE. Perform any requested PIO word transfers, then perform a DMA transfer from the peripheral for the specified number of bytes.
10	DMA_READ. Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

Table continues on the next page...

Table 9-17. APBH DMA commands (continued)

DMA Command	Usage
11	DMA_SENSE. Perform any requested PIO word transfers, then perform a conditional branch to the next chained device. Follow the NEXTCMD_ADDR pointer if the peripheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true. This command becomes a no-operation for any channel other than a GPMI channel.

DMA_WRITE operations copy data bytes to the system memory (on-chip RAM or SDRAM) from the associated peripheral. The DMA_WRITE transfer uses the BUFFER_ADDRESS word in the command structure to point to the beginning byte to write data from the peripheral.

DMA_READ operations copy data bytes to the APB peripheral from the system memory. The DMA engine contains a shared byte aligner that aligns bytes from system memory to or from the peripherals. Peripherals always assume little-endian-aligned data arrives or departs on their 32-bit APB. The DMA_READ transfer uses the BUFFER_ADDRESS word in the command structure to point to the DMA data buffer to be read by the DMA_READ command.

The NO_DMA_XFER command is used to write PIO words to a device without performing any DMA data byte transfers. This command is useful in such applications as activating the NAND devices CHECKSTATUS operation. The check status command reads a status byte from the NAND device, performs an XOR and MASK against an expected value supplied as part of the PIO transfer. Once the read check completes (see NAND Read Status Polling Example), the NO_DMA_XFER command completes. The result in the peripheral is that its sense line is driven by the results of the comparison. The sense flip-flop is only updated by CHECKSTATUS for the device that is executed. At some future point, the chain contains a DMA command structure with the fourth and final command value, that is, the DMA_SENSE command.

As each DMA command completes, it triggers the DMA to load the next DMA command structure in the chain. The normal flow list of DMA commands is found by following the NEXTCMD_ADDR pointer in the DMA command structure. The DMA_SENSE command uses the DMA buffer pointer word of the command structure to point to an alternate DMA command structure chain or list. The DMA_SENSE command examines the sense line of the associated peripheral. If the sense line is false, then the DMA follows the standard list found whose next command is found from the pointer in the NEXTCMD_ADDR word of the command structure. If the sense line is true, then the DMA follows the alternate list whose next command is found from the pointer in the DMA Buffer Pointer word of the DMA_SENSE command structure (see Figure 9-26). The sense command ignores the CHAIN bit, so that both pointers must be valid when the DMA comes to a sense command.

If the wait-for-end-command bit (WAIT4ENDCMD) is set in a command structure, the DMA channel waits for the device to signal completion of a command by toggling the endcmd signal before proceeding to load and execute the next command structure. Then, if DECREMENT_SEMAPHORE is set, the semaphore is decremented after the end command is seen.

A detailed bit-field view of the DMA command structure is shown in the following table, which shows a field that specifies the number of bytes to be transferred by this DMA command. The transfer-count mechanism is duplicated in the associated peripheral, either as an implied or as a specified count in the peripheral.

Table 9-18. DMA channel command word in system memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NEXT_COMMAND_ADDRESS																															
Number DMA Bytes to Transfer																Number PIO Words to Write							HALTONTERMINATE	WAIT4ENDCMD	DECREMENT SEMAPHORE	NANDwAIT4READY	NANDLOCK	IRQ_COMPLETE	CHAIN	COMMAND	
DMA Buffer or Alternate CCW																															
Zero or More PIO Words to Write to the Associated Peripheral Starting at its Base Address on the APBH Bus																															

Figure 9-26 also shows the CHAIN bit in bit 2 of the second word of the command structure. This bit is set to 1, if the NEXT_COMMAND_ADDRESS contains a pointer to another DMA command structure. If a null pointer (0) is loaded into the NEXT_COMMAND_ADDRESS, it is not detected by the DMA hardware. Only the CHAIN bit indicates whether a valid list exists beyond the current structure.

If the IRQ_COMPLETE bit is set in the command structure, then the last act of the DMA before loading the next command is to set the interrupt-status bit corresponding to the current channel. The sticky interrupt request bit in the DMA CSR remains set until cleared by the software. It can be used to interrupt the ARM platform.

The NAND_LOCK bit is monitored by the DMA channel arbiter. After a NAND channel (from channel 4 to channel 11) Once a NAND channel (from channel 0 to channel 3) succeeds in the arbiter with its NAND_LOCK bit set, then the arbiter ignores the other NAND channels until a command is completed in which the NAND_LOCK is not set. Notice that the semantic here is that the NAND_LOCK state is to limit scheduling of a non-locked DMA. A DMA channel can go from unlocked to locked in the arbiter at the beginning of a command when the NAND_LOCK bit is set. When the last DMA

command of an atomic sequence is completed, the lock should be removed. To accomplish this, the last command does not have the NAND_LOCK bit. It is still locked in the atomic state within the arbiter when the command starts, so that it is the only NAND command that can be executed. At the end, it drops from the atomic state within the arbiter.

The NAND_WAIT4READY bit also has a special use for GPMI channels (from channel 4 0 to channel 11 3), i.e., the NAND device channels. The GPMI peripheral supplies a sample of the ready line from the NAND device. This ready value is used to hold off of a command with this bit set until the ready line is asserted to 1. Once the arbiter sees a command with a wait-for-ready set, it holds off that channel until ready is asserted.

Receiving an IRQ for HALTONTERMINATE (HOT) is a feature in the APBH DMA descriptor that allows GPMI (as well as SSP and I2C) to signal to the DMA engine that an error has occurred. If a commmand is stalled due to an error, a HOT signal is sent from the peripheral to the DMA engine and causes an IRQ after terminating the DMA descriptor being executed.

Therefore, it is recommended that software use this signal as follows:

- Always set HALTONTERMINATE to 1 in a DMA descriptor. That way, if a peripheral signals HOT, the transfer will end, leaving the peripheral block and the DMA engine synchronized (but at the end of a command).
- When an IRQ from an APBH channel is received, and the IRQ is determined to be due to an error (as opposed to an IRQONCOMPLETE interrupt) the software should:
    - Reset the channel.
    - Determine the error from error reporting in the peripheral block, then manage the error in the peripheral that is attached to that channel in whatever appropriate way exists for that device (software recovery, device reset, block reset, etc).

Each channel has an eight-bit counting semaphore that controls whether it is in the idle state. When the semaphore is non-zero, the channel is ready to run, process commands and perform DMA transfers. Whenever a command finishes its DMA transfer, it checks the DECREMENT_SEMAPHORE bit. If set, it decrements the counting semaphore. If the semaphore goes to 0 as a result, then the channel enters the idle state and remains there until the semaphore is incremented by the software. When the semaphore goes to non-zero and the channel is in its idle state, then it uses the value in the APBH_CHn_NXTCMDAR register (next command address register) to fetch a pointer to the next command to process.

NOTE

This is a double indirect case. This method allows the software to append to a running command list under the protection of the counting semaphore.

To start processing the first time, software creates the command list to be processed. It writes the address of the first command into the APBH_CHn_NXTCMDAR register, and then writes 1 to the counting semaphore in APBH_CHn_SEMA. The DMA channel loads APBH_CHn_CURCMDAR register and then enters the normal state machine processing for the next command. When the software writes a value to the counting semaphore, it is added to the semaphore count by hardware, protecting the case where both hardware and software are trying to change the semaphore on the same clock edge.

Software can examine the value of APBH_CHn_CURCMDAR at any time to determine the location of the command structure currently being processed.

9.4.4 NAND Read Status Polling Example

The following figure shows a more complicated scenario.

This subset of a NAND device workload shows that the first two command structures are used during the data-write phase of an NAND device write operation (CLE and ALE transfers omitted for clarity).

- After writing the data, one must wait until the NAND device status register indicates that the write charge has been transferred. This is built into the workload using a check status command in the NAND in a loop created from the next two DMA command structures.
- The NO_DMA_TRANSFER command is shown here performing the read check, followed by a DMA_SENSE command to branch the DMA command structure list, based on the status of a bit in the external NAND device.

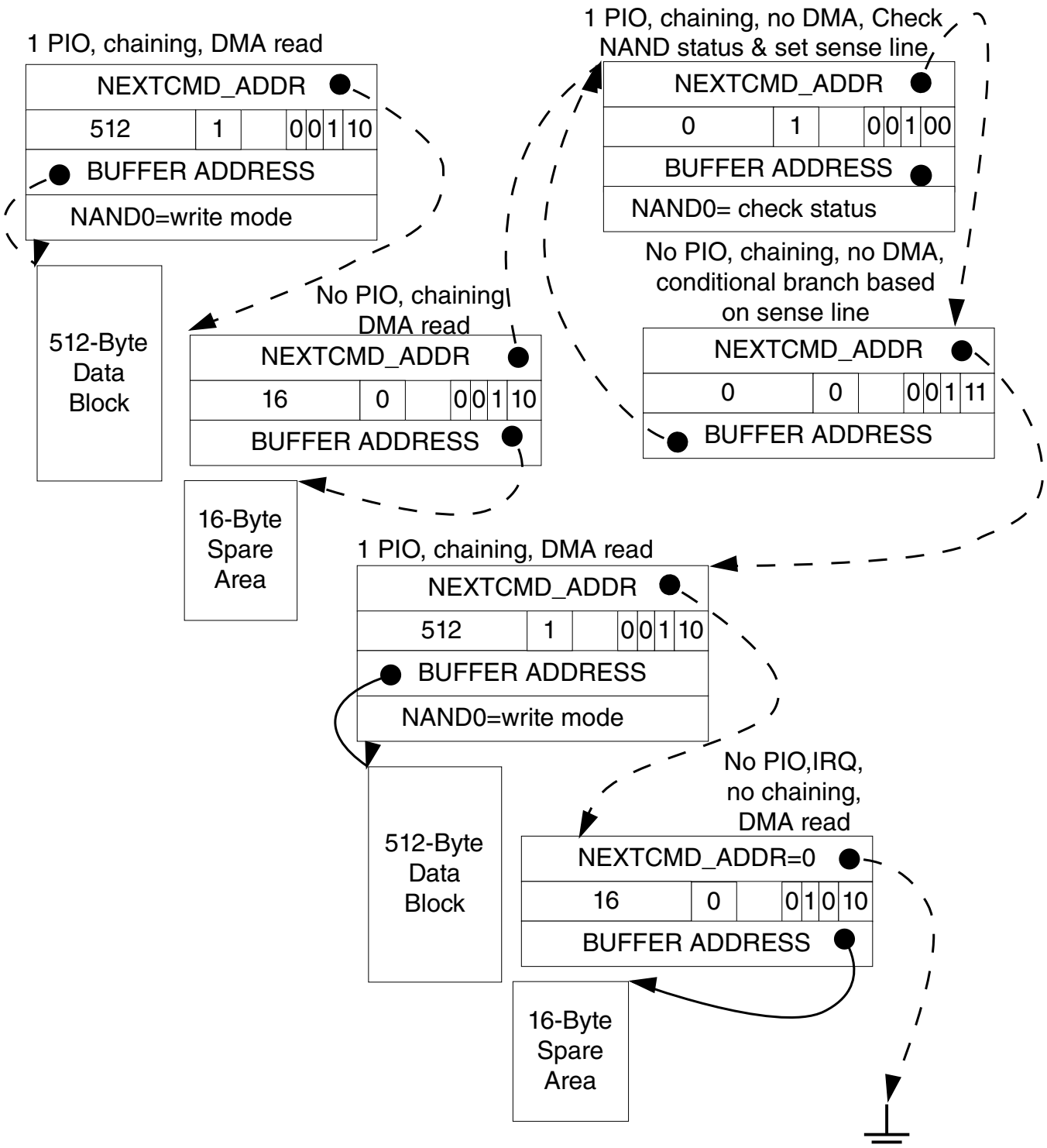


Figure 9-27. AHB-to-APBH Bridge DMA NAND Read Status Polling with DMA Sense Command

The example in the above figure shows the workload continuing immediately to the next NAND page transfer. However, one could perform a second sense operation to see if an error has occurred after the write. One could then point the sense command alternate branch at a NO_DMA_XFER command with the interrupt bit set. If the CHAIN bit is not

set on this failure branch, then the ARM platform is interrupted immediately, and the channel process is also immediately terminated in the presence of a workload-detected NAND error bit.

Note that each word of the three-word DMA command structure corresponds to a PIO register of the DMA that is accessible on the APBH bus. Normally, the DMA copies the next command structure onto these registers for processing at the start of each command by following the value of the pointer previously loaded into the NEXTCMD_ADDR register.

To start DMA processing for the first command, initialize the PIO registers of the desired channel, as follows:

- First, load the next command address register with a pointer to the first command to be loaded.
- Then, write 1 to the counting semaphore register. This causes the DMA to schedule the targeted channel for the DMA command structure load, just as if it had finished its previous command.

9.4.5 APBH Memory Map/Register Definition

APBH Hardware Register Format Summary

APBH memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3300_0000	AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0)	32	R/W	E000_0000h	9.4.5.1/ 2360
3300_0004	AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0_SET)	32	R/W	E000_0000h	9.4.5.1/ 2360
3300_0008	AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0_CLR)	32	R/W	E000_0000h	9.4.5.1/ 2360
3300_000C	AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0_TOG)	32	R/W	E000_0000h	9.4.5.1/ 2360
3300_0010	AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1)	32	R/W	0000_0000h	9.4.5.2/ 2362
3300_0014	AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1_SET)	32	R/W	0000_0000h	9.4.5.2/ 2362
3300_0018	AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1_CLR)	32	R/W	0000_0000h	9.4.5.2/ 2362

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3300_001C	AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1_TOG)	32	R/W	0000_0000h	9.4.5.2/ 2362
3300_0020	AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2)	32	R/W	0000_0000h	9.4.5.3/ 2365
3300_0024	AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2_SET)	32	R/W	0000_0000h	9.4.5.3/ 2365
3300_0028	AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2_CLR)	32	R/W	0000_0000h	9.4.5.3/ 2365
3300_002C	AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2_TOG)	32	R/W	0000_0000h	9.4.5.3/ 2365
3300_0030	AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL)	32	R/W	0000_0000h	9.4.5.4/ 2370
3300_0034	AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL_SET)	32	R/W	0000_0000h	9.4.5.4/ 2370
3300_0038	AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL_CLR)	32	R/W	0000_0000h	9.4.5.4/ 2370
3300_003C	AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL_TOG)	32	R/W	0000_0000h	9.4.5.4/ 2370
3300_0040	AHB to APBH DMA Device Assignment Register (APBH_DEVSEL)	32	R/W	0000_0000h	9.4.5.5/ 2371
3300_0050	AHB to APBH DMA burst size (APBH_DMA_BURST_SIZE)	32	R/W	0055_5555h	9.4.5.6/ 2372
3300_0060	AHB to APBH DMA Debug Register (APBH_DEBUG)	32	R/W	0000_0000h	9.4.5.7/ 2373
3300_0100	APBH DMA Channel n Current Command Address Register (APBH_CH0_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_0110	APBH DMA Channel n Next Command Address Register (APBH_CH0_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_0120	APBH DMA Channel n Command Register (APBH_CH0_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_0130	APBH DMA Channel n Buffer Address Register (APBH_CH0_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_0140	APBH DMA Channel n Semaphore Register (APBH_CH0_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_0150	AHB to APBH DMA Channel n Debug Information (APBH_CH0_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_0160	AHB to APBH DMA Channel n Debug Information (APBH_CH0_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_0170	APBH DMA Channel n Current Command Address Register (APBH_CH1_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_0180	APBH DMA Channel n Next Command Address Register (APBH_CH1_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_0190	APBH DMA Channel n Command Register (APBH_CH1_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3300_01A0	APBH DMA Channel n Buffer Address Register (APBH_CH1_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_01B0	APBH DMA Channel n Semaphore Register (APBH_CH1_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_01C0	AHB to APBH DMA Channel n Debug Information (APBH_CH1_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_01D0	AHB to APBH DMA Channel n Debug Information (APBH_CH1_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_01E0	APBH DMA Channel n Current Command Address Register (APBH_CH2_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_01F0	APBH DMA Channel n Next Command Address Register (APBH_CH2_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_0200	APBH DMA Channel n Command Register (APBH_CH2_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_0210	APBH DMA Channel n Buffer Address Register (APBH_CH2_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_0220	APBH DMA Channel n Semaphore Register (APBH_CH2_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_0230	AHB to APBH DMA Channel n Debug Information (APBH_CH2_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_0240	AHB to APBH DMA Channel n Debug Information (APBH_CH2_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_0250	APBH DMA Channel n Current Command Address Register (APBH_CH3_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_0260	APBH DMA Channel n Next Command Address Register (APBH_CH3_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_0270	APBH DMA Channel n Command Register (APBH_CH3_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_0280	APBH DMA Channel n Buffer Address Register (APBH_CH3_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_0290	APBH DMA Channel n Semaphore Register (APBH_CH3_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_02A0	AHB to APBH DMA Channel n Debug Information (APBH_CH3_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_02B0	AHB to APBH DMA Channel n Debug Information (APBH_CH3_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_02C0	APBH DMA Channel n Current Command Address Register (APBH_CH4_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_02D0	APBH DMA Channel n Next Command Address Register (APBH_CH4_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_02E0	APBH DMA Channel n Command Register (APBH_CH4_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_02F0	APBH DMA Channel n Buffer Address Register (APBH_CH4_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3300_0300	APBH DMA Channel n Semaphore Register (APBH_CH4_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_0310	AHB to APBH DMA Channel n Debug Information (APBH_CH4_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_0320	AHB to APBH DMA Channel n Debug Information (APBH_CH4_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_0330	APBH DMA Channel n Current Command Address Register (APBH_CH5_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_0340	APBH DMA Channel n Next Command Address Register (APBH_CH5_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_0350	APBH DMA Channel n Command Register (APBH_CH5_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_0360	APBH DMA Channel n Buffer Address Register (APBH_CH5_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_0370	APBH DMA Channel n Semaphore Register (APBH_CH5_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_0380	AHB to APBH DMA Channel n Debug Information (APBH_CH5_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_0390	AHB to APBH DMA Channel n Debug Information (APBH_CH5_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_03A0	APBH DMA Channel n Current Command Address Register (APBH_CH6_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_03B0	APBH DMA Channel n Next Command Address Register (APBH_CH6_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_03C0	APBH DMA Channel n Command Register (APBH_CH6_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_03D0	APBH DMA Channel n Buffer Address Register (APBH_CH6_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_03E0	APBH DMA Channel n Semaphore Register (APBH_CH6_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_03F0	AHB to APBH DMA Channel n Debug Information (APBH_CH6_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_0400	AHB to APBH DMA Channel n Debug Information (APBH_CH6_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_0410	APBH DMA Channel n Current Command Address Register (APBH_CH7_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_0420	APBH DMA Channel n Next Command Address Register (APBH_CH7_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_0430	APBH DMA Channel n Command Register (APBH_CH7_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_0440	APBH DMA Channel n Buffer Address Register (APBH_CH7_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_0450	APBH DMA Channel n Semaphore Register (APBH_CH7_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3300_0460	AHB to APBH DMA Channel n Debug Information (APBH_CH7_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_0470	AHB to APBH DMA Channel n Debug Information (APBH_CH7_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_0480	APBH DMA Channel n Current Command Address Register (APBH_CH8_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_0490	APBH DMA Channel n Next Command Address Register (APBH_CH8_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_04A0	APBH DMA Channel n Command Register (APBH_CH8_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_04B0	APBH DMA Channel n Buffer Address Register (APBH_CH8_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_04C0	APBH DMA Channel n Semaphore Register (APBH_CH8_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_04D0	AHB to APBH DMA Channel n Debug Information (APBH_CH8_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_04E0	AHB to APBH DMA Channel n Debug Information (APBH_CH8_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_04F0	APBH DMA Channel n Current Command Address Register (APBH_CH9_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_0500	APBH DMA Channel n Next Command Address Register (APBH_CH9_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_0510	APBH DMA Channel n Command Register (APBH_CH9_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_0520	APBH DMA Channel n Buffer Address Register (APBH_CH9_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_0530	APBH DMA Channel n Semaphore Register (APBH_CH9_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_0540	AHB to APBH DMA Channel n Debug Information (APBH_CH9_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_0550	AHB to APBH DMA Channel n Debug Information (APBH_CH9_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_0560	APBH DMA Channel n Current Command Address Register (APBH_CH10_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_0570	APBH DMA Channel n Next Command Address Register (APBH_CH10_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_0580	APBH DMA Channel n Command Register (APBH_CH10_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_0590	APBH DMA Channel n Buffer Address Register (APBH_CH10_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_05A0	APBH DMA Channel n Semaphore Register (APBH_CH10_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_05B0	AHB to APBH DMA Channel n Debug Information (APBH_CH10_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3300_05C0	AHB to APBH DMA Channel n Debug Information (APBH_CH10_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_05D0	APBH DMA Channel n Current Command Address Register (APBH_CH11_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_05E0	APBH DMA Channel n Next Command Address Register (APBH_CH11_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_05F0	APBH DMA Channel n Command Register (APBH_CH11_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_0600	APBH DMA Channel n Buffer Address Register (APBH_CH11_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_0610	APBH DMA Channel n Semaphore Register (APBH_CH11_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_0620	AHB to APBH DMA Channel n Debug Information (APBH_CH11_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_0630	AHB to APBH DMA Channel n Debug Information (APBH_CH11_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_0640	APBH DMA Channel n Current Command Address Register (APBH_CH12_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_0650	APBH DMA Channel n Next Command Address Register (APBH_CH12_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_0660	APBH DMA Channel n Command Register (APBH_CH12_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_0670	APBH DMA Channel n Buffer Address Register (APBH_CH12_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_0680	APBH DMA Channel n Semaphore Register (APBH_CH12_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_0690	AHB to APBH DMA Channel n Debug Information (APBH_CH12_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_06A0	AHB to APBH DMA Channel n Debug Information (APBH_CH12_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_06B0	APBH DMA Channel n Current Command Address Register (APBH_CH13_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_06C0	APBH DMA Channel n Next Command Address Register (APBH_CH13_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_06D0	APBH DMA Channel n Command Register (APBH_CH13_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_06E0	APBH DMA Channel n Buffer Address Register (APBH_CH13_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_06F0	APBH DMA Channel n Semaphore Register (APBH_CH13_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_0700	AHB to APBH DMA Channel n Debug Information (APBH_CH13_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_0710	AHB to APBH DMA Channel n Debug Information (APBH_CH13_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382

Table continues on the next page...

APBH memory map (continued)

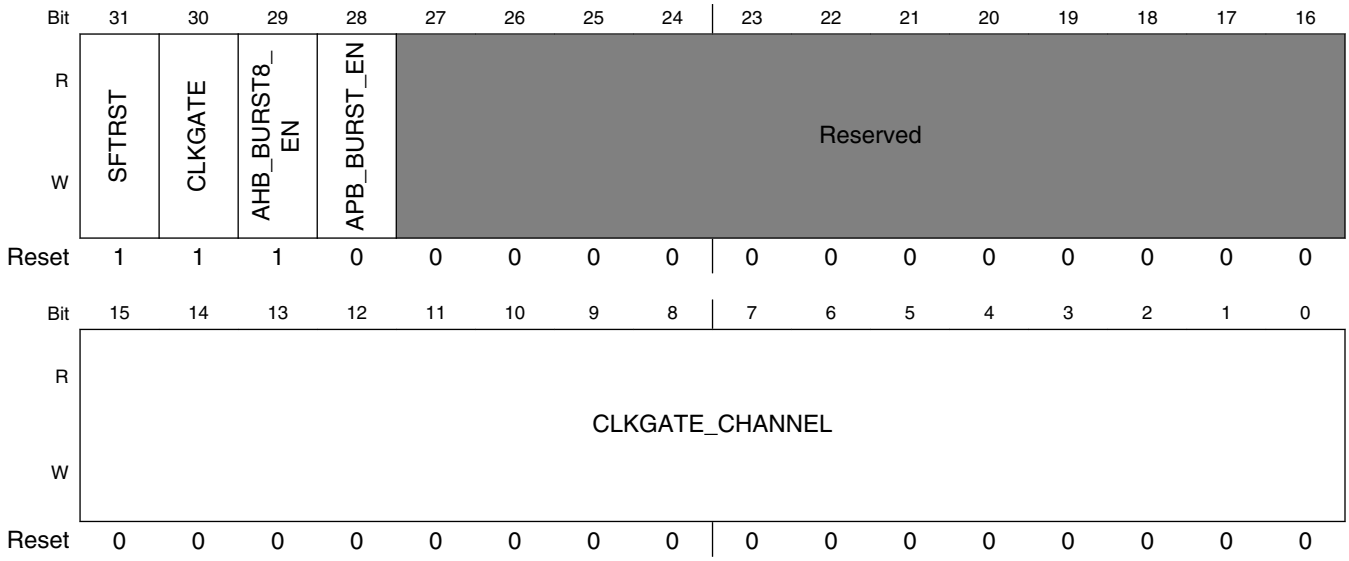
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3300_0720	APBH DMA Channel n Current Command Address Register (APBH_CH14_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_0730	APBH DMA Channel n Next Command Address Register (APBH_CH14_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_0740	APBH DMA Channel n Command Register (APBH_CH14_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_0750	APBH DMA Channel n Buffer Address Register (APBH_CH14_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_0760	APBH DMA Channel n Semaphore Register (APBH_CH14_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_0770	AHB to APBH DMA Channel n Debug Information (APBH_CH14_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_0780	AHB to APBH DMA Channel n Debug Information (APBH_CH14_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_0790	APBH DMA Channel n Current Command Address Register (APBH_CH15_CURCMDAR)	32	R/W	0000_0000h	9.4.5.8/ 2374
3300_07A0	APBH DMA Channel n Next Command Address Register (APBH_CH15_NXTCMDAR)	32	R/W	0000_0000h	9.4.5.9/ 2375
3300_07B0	APBH DMA Channel n Command Register (APBH_CH15_CMD)	32	R/W	0000_0000h	9.4.5.10/ 2375
3300_07C0	APBH DMA Channel n Buffer Address Register (APBH_CH15_BAR)	32	R/W	0000_0000h	9.4.5.11/ 2377
3300_07D0	APBH DMA Channel n Semaphore Register (APBH_CH15_SEMA)	32	R/W	0000_0000h	9.4.5.12/ 2378
3300_07E0	AHB to APBH DMA Channel n Debug Information (APBH_CH15_DEBUG1)	32	R/W	00A0_0000h	9.4.5.13/ 2379
3300_07F0	AHB to APBH DMA Channel n Debug Information (APBH_CH15_DEBUG2)	32	R/W	0000_0000h	9.4.5.14/ 2382
3300_0800	APBH Bridge Version Register (APBH_VERSION)	32	R/W	0301_0000h	9.4.5.15/ 2382

9.4.5.1 AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0*n*)

The APBH CTRL 0 provides overall control of the AHB to APBH bridge and DMA.

This register contains module softreset, clock gating, channel clock gating/freeze bits.

Address: 3300_0000h base + 0h offset + (4d × i), where i=0d to 3d



APBH_CTRL0*n* field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal APBH DMA operation. Set this bit to one (default) to disable clocking with the APBH DMA and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the APBH DMA block to its default state.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29 AHB_BURST8_EN	Set this bit to one (default) to enable AHB 8-beat burst. Set to zero to disable 8-beat burst on AHB interface.
28 APB_BURST_EN	Set this bit to one to enable apb master do a continous transfers when a device request a burst dma. Set to zero will treat a burst dma request as 4/8 individual requests.
27–16 RSVD0	This field is reserved. Reserved, always set to zero.
CLKGATE_CHANNEL	These bits must be set to zero for normal operation of each channel. When set to one they gate off the individual clocks to the channels. 0x0001 **NAND0** — 0x0002 **NAND1** — 0x0004 **NAND2** — 0x0008 **NAND3** — 0x0010 **NAND4** — 0x0020 **NAND5** — 0x0040 **NAND6** — 0x0080 **NAND7** — 0x0100 **SSP** —

9.4.5.2 AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1*n*)

The APBH CTRL one provides overall control of the interrupts generated by the AHB to APBH DMA. This register contains the per channel interrupt status bits and the per channel interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE

```
                    BF_WR(APBH_CTRL1, CH5_CMDCMPLT_IRQ, 0);  // use bitfield write macro
                    BF_APBH_CTRL1.CH5_CMDCMPLT_IRQ = 0;      // or, assign to register
struct's bitfield
```

Address: 3300_0000h base + 10h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CH15_CMDCMPLT_IRQ_EN	CH14_CMDCMPLT_IRQ_EN	CH13_CMDCMPLT_IRQ_EN	CH12_CMDCMPLT_IRQ_EN	CH11_CMDCMPLT_IRQ_EN	CH10_CMDCMPLT_IRQ_EN	CH9_CMDCMPLT_IRQ_EN	CH8_CMDCMPLT_IRQ_EN	CH7_CMDCMPLT_IRQ_EN	CH6_CMDCMPLT_IRQ_EN	CH5_CMDCMPLT_IRQ_EN	CH4_CMDCMPLT_IRQ_EN	CH3_CMDCMPLT_IRQ_EN	CH2_CMDCMPLT_IRQ_EN	CH1_CMDCMPLT_IRQ_EN	CH0_CMDCMPLT_IRQ_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CH15_CMDCMPLT_IRQ	CH14_CMDCMPLT_IRQ	CH13_CMDCMPLT_IRQ	CH12_CMDCMPLT_IRQ	CH11_CMDCMPLT_IRQ	CH10_CMDCMPLT_IRQ	CH9_CMDCMPLT_IRQ	CH8_CMDCMPLT_IRQ	CH7_CMDCMPLT_IRQ	CH6_CMDCMPLT_IRQ	CH5_CMDCMPLT_IRQ	CH4_CMDCMPLT_IRQ	CH3_CMDCMPLT_IRQ	CH2_CMDCMPLT_IRQ	CH1_CMDCMPLT_IRQ	CH0_CMDCMPLT_IRQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_CTRL1*n* field descriptions

Field	Description
31 CH15_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 15.
30 CH14_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 14.

Table continues on the next page...

APBH_CTRL1*n* field descriptions (continued)

Field	Description
29 CH13_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 13.
28 CH12_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 12.
27 CH11_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 11.
26 CH10_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 10.
25 CH9_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 9.
24 CH8_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 8.
23 CH7_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 7.
22 CH6_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 6.
21 CH5_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 5.
20 CH4_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 4.
19 CH3_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 3.
18 CH2_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 2.

Table continues on the next page...

APBH_CTRL1*n* field descriptions (continued)

Field	Description
17 CH1_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 1.
16 CH0_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 0.
15 CH15_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 15. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
14 CH14_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 14. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
13 CH13_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 13. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
12 CH12_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 12. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
11 CH11_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 11. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
10 CH10_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 10. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
9 CH9_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 9. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
8 CH8_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 8. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
7 CH7_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 7. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
6 CH6_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 6. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

Table continues on the next page...

APBH_CTRL1*n* field descriptions (continued)

Field	Description
5 CH5_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 5. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
4 CH4_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 4. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
3 CH3_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 3. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
2 CH2_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 2. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
1 CH1_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 1. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
0 CH0_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 0. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

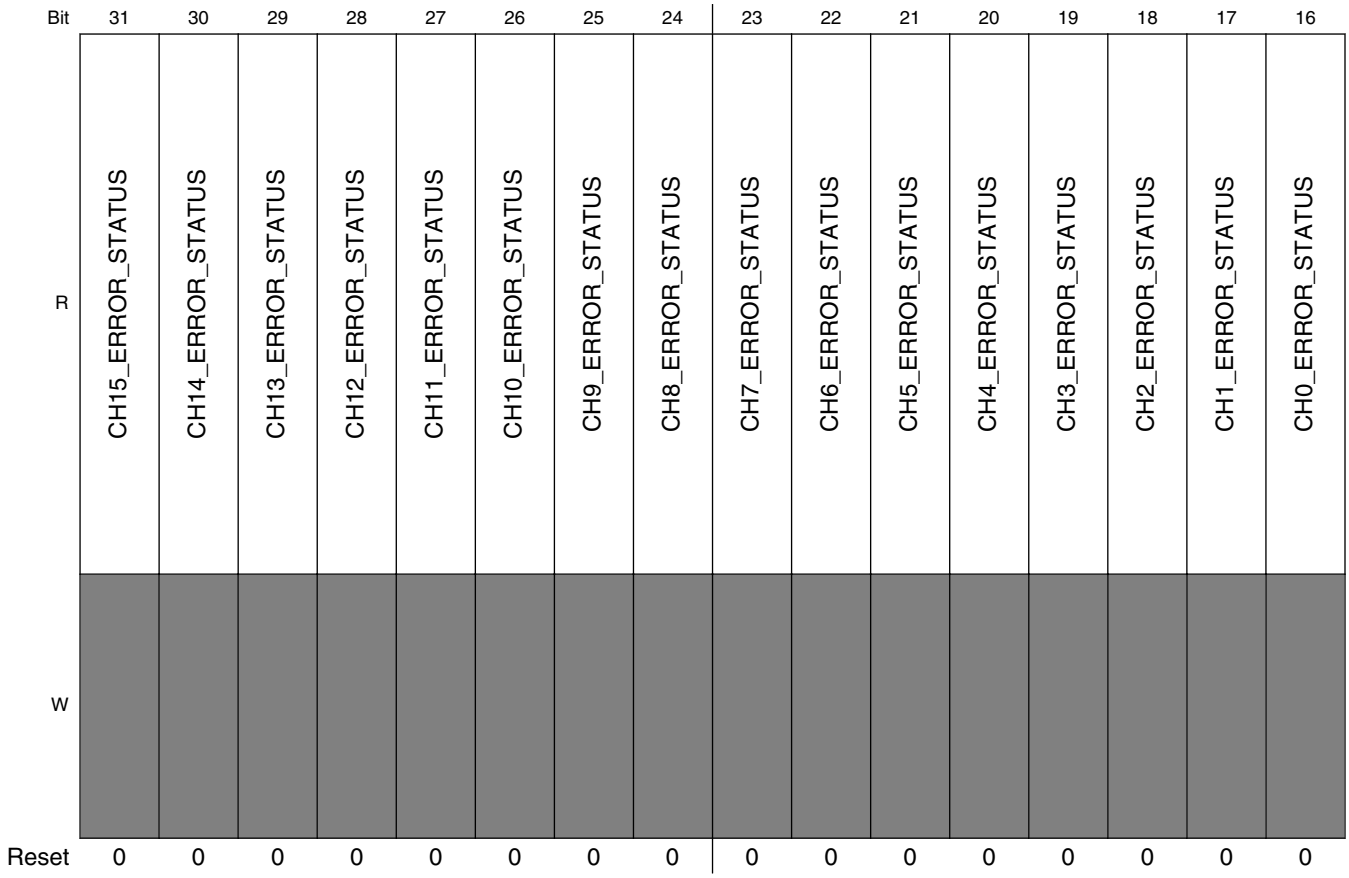
9.4.5.3 AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2*n*)

The APBH CTRL 2 provides channel error interrupts generated by the AHB to APBH DMA. This register contains the per channel interrupt status bits and the per channel interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE

```
                    BF_WR(APBH_CTRL1, CH5_CMDCMPLT_IRQ, 0);  // use bitfield write macro
                    BF_APBH_CTRL1.CH5_CMDCMPLT_IRQ = 0;      // or, assign to register
struct's bitfield
```

Address: 3300_0000h base + 20h offset + (4d × i), where i=0d to 3d



Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CH15_ERROR_STATUS	CH14_ERROR_STATUS	CH13_ERROR_STATUS	CH12_ERROR_STATUS	CH11_ERROR_STATUS	CH10_ERROR_STATUS	CH9_ERROR_STATUS	CH8_ERROR_STATUS	CH7_ERROR_STATUS	CH6_ERROR_STATUS	CH5_ERROR_STATUS	CH4_ERROR_STATUS	CH3_ERROR_STATUS	CH2_ERROR_STATUS	CH1_ERROR_STATUS	CH0_ERROR_STATUS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CH15_ERROR_IRQ	CH14_ERROR_IRQ	CH13_ERROR_IRQ	CH12_ERROR_IRQ	CH11_ERROR_IRQ	CH10_ERROR_IRQ	CH9_ERROR_IRQ	CH8_ERROR_IRQ	CH7_ERROR_IRQ	CH6_ERROR_IRQ	CH5_ERROR_IRQ	CH4_ERROR_IRQ	CH3_ERROR_IRQ	CH2_ERROR_IRQ	CH1_ERROR_IRQ	CH0_ERROR_IRQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_CTRL2*n* field descriptions

Field	Description
31 CH15_ERROR_STATUS	Error status bit for APBH DMA Channel 15. Valid when corresponding Error IRQ is set. 1 - AHB bus error

Table continues on the next page...

APBH_CTRL2*n* field descriptions (continued)

Field	Description
	0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
30 CH14_ERROR_ STATUS	Error status bit for APBH DMA Channel 14. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
29 CH13_ERROR_ STATUS	Error status bit for APBH DMA Channel 13. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
28 CH12_ERROR_ STATUS	Error status bit for APBH DMA Channel 12. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
27 CH11_ERROR_ STATUS	Error status bit for APBH DMA Channel 11. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
26 CH10_ERROR_ STATUS	Error status bit for APBH DMA Channel 10. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
25 CH9_ERROR_ STATUS	Error status bit for APBH DMA Channel 9. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
24 CH8_ERROR_ STATUS	Error status bit for APBH DMA Channel 8. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination.

Table continues on the next page...

APBH_CTRL2*n* field descriptions (continued)

Field	Description
	0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
23 CH7_ERROR_STATUS	Error status bit for APBX DMA Channel 7. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
22 CH6_ERROR_STATUS	Error status bit for APBX DMA Channel 6. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
21 CH5_ERROR_STATUS	Error status bit for APBX DMA Channel 5. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
20 CH4_ERROR_STATUS	Error status bit for APBX DMA Channel 4. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
19 CH3_ERROR_STATUS	Error status bit for APBX DMA Channel 3. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
18 CH2_ERROR_STATUS	Error status bit for APBX DMA Channel 2. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
17 CH1_ERROR_STATUS	Error status bit for APBX DMA Channel 1. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.

Table continues on the next page...

APBH_CTRL2*n* field descriptions (continued)

Field	Description
16 CH0_ERROR_STATUS	Error status bit for APBX DMA Channel 0. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
15 CH15_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 15. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
14 CH14_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 14. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
13 CH13_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 13. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
12 CH12_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 12. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
11 CH11_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 11. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
10 CH10_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 10. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
9 CH9_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 9. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
8 CH8_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 8. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
7 CH7_ERROR_IRQ	Error interrupt status bit for APBX DMA Channel 7. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
6 CH6_ERROR_IRQ	Error interrupt status bit for APBX DMA Channel 6. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
5 CH5_ERROR_IRQ	Error interrupt status bit for APBX DMA Channel 5. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
4 CH4_ERROR_IRQ	Error interrupt status bit for APBX DMA Channel 4. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
3 CH3_ERROR_IRQ	Error interrupt status bit for APBX DMA Channel 3. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
2 CH2_ERROR_IRQ	Error interrupt status bit for APBX DMA Channel 2. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.

Table continues on the next page...

APBH_CTRL2*n* field descriptions (continued)

Field	Description
1 CH1_ERROR_IRQ	Error interrupt status bit for APBX DMA Channel 1. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
0 CH0_ERROR_IRQ	Error interrupt status bit for APBX DMA Channel 0. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.

9.4.5.4 AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL*n*)

The APBH CHANNEL CTRL provides reset/freeze control of each DMA channel. This register contains individual channel reset/freeze bits.

Address: 3300_0000h base + 30h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RESET_CHANNEL																FREEZE_CHANNEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_CHANNEL_CTRL*n* field descriptions

Field	Description
31–16 RESET_CHANNEL	Setting a bit in this field causes the DMA controller to take the corresponding channel through its reset state. The bit is reset after the channel resources are cleared. 0x0001 **NAND0** — 0x0002 **NAND1** — 0x0004 **NAND2** — 0x0008 **NAND3** — 0x0010 **NAND4** — 0x0020 **NAND5** — 0x0040 **NAND6** — 0x0080 **NAND7** — 0x0100 **SSP** —
FREEZE_CHANNEL	Setting a bit in this field will freeze the DMA channel associated with it. This field is a direct input to the DMA channel arbiter. When frozen, the channel is deined access to the central DMA resources. 0x0001 **NAND0** — 0x0002 **NAND1** — 0x0004 **NAND2** — 0x0008 **NAND3** — 0x0010 **NAND4** — 0x0020 **NAND5** — 0x0040 **NAND6** —

Table continues on the next page...

APBH_CHANNEL_CTRL*n* field descriptions (continued)

Field	Description
	0x0080 **NAND7** — 0x0100 **SSP** —

9.4.5.5 AHB to APBH DMA Device Assignment Register (APBH_DEVSEL)

This register allows reassignment of the APBH device connected to the DMA Channels.

In this chip, APBH DMA channel resource is enough for high speed peripherals, so this register is of no use and reserved.

Address: 3300_0000h base + 40h offset = 3300_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_DEVSEL field descriptions

Field	Description
31–30 CH15	This field is reserved. Reserved.
29–28 CH14	This field is reserved. Reserved.
27–26 CH13	This field is reserved. Reserved.
25–24 CH12	This field is reserved. Reserved.
23–22 CH11	This field is reserved. Reserved.
21–20 CH10	This field is reserved. Reserved.
19–18 CH9	This field is reserved. Reserved.
17–16 CH8	This field is reserved. Reserved.
15–14 CH7	This field is reserved. Reserved.
13–12 CH6	This field is reserved. Reserved.

Table continues on the next page...

APBH_DEVSEL field descriptions (continued)

Field	Description
11–10 CH5	This field is reserved. Reserved.
9–8 CH4	This field is reserved. Reserved.
7–6 CH3	This field is reserved. Reserved.
5–4 CH2	This field is reserved. Reserved.
3–2 CH1	This field is reserved. Reserved.
CH0	This field is reserved. Reserved.

9.4.5.6 AHB to APBH DMA burst size (APBH_DMA_BURST_SIZE)

This register programs the apbh burst size of the APBH DMA devices when a DMA burst request is issued.

This register provides a mechanism for assigning the device.

Address: 3300_0000h base + 50h offset = 3300_0050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		CH8	
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH7		CH6		CH5		CH4		CH3		CH2		CH1		CH0	
Reset	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

APBH_DMA_BURST_SIZE field descriptions

Field	Description
31–30 CH15	This field is reserved. Reserved.
29–28 CH14	This field is reserved. Reserved.
27–26 CH13	This field is reserved. Reserved.
25–24 CH12	This field is reserved. Reserved.
23–22 CH11	This field is reserved. Reserved.

Table continues on the next page...

APBH_DMA_BURST_SIZE field descriptions (continued)

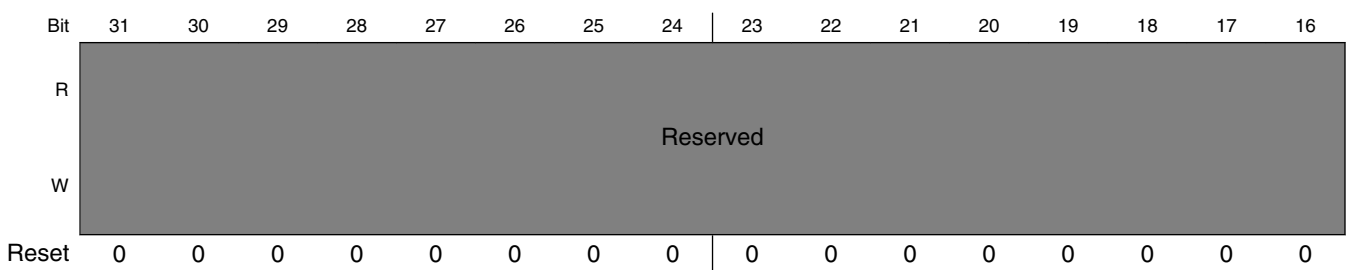
Field	Description
21–20 CH10	This field is reserved. Reserved.
19–18 CH9	This field is reserved. Reserved.
17–16 CH8	DMA burst size for SSP. 0x0 **BURST0** — 0x1 **BURST4** — 0x2 **BURST8** —
15–14 CH7	DMA burst size for GPMI channel 7. Do not change. GPMI only support burst size 4.
13–12 CH6	DMA burst size for GPMI channel 6. Do not change. GPMI only support burst size 4.
11–10 CH5	DMA burst size for GPMI channel 5. Do not change. GPMI only support burst size 4.
9–8 CH4	DMA burst size for GPMI channel 4. Do not change. GPMI only support burst size 4.
7–6 CH3	DMA burst size for GPMI channel 3. Do not change. GPMI only support burst size 4.
5–4 CH2	DMA burst size for GPMI channel 2. Do not change. GPMI only support burst size 4.
3–2 CH1	DMA burst size for GPMI channel 1. Do not change. GPMI only support burst size 4.
CH0	DMA burst size for GPMI channel 0. Do not change. GPMI only support burst size 4.

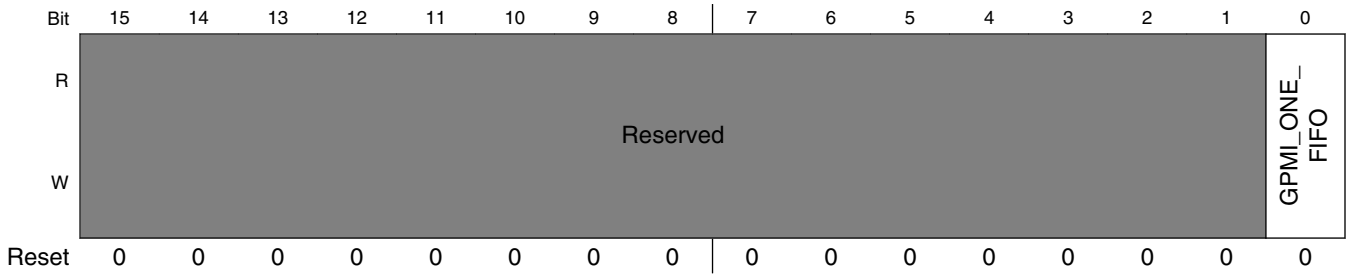
9.4.5.7 AHB to APBH DMA Debug Register (APBH_DEBUG)

This register is for debug purpose.

The debug register is for internal use only. Not recommend for customer useage.

Address: 3300_0000h base + 60h offset = 3300_0060h





APBH_DEBUG field descriptions

Field	Description
31–1 -	This field is reserved. Reserved, always set to zero.
0 GPMI_ONE_FIFO	Set to 0ne and the 8 GPMI channels will share the DMA FIFO, and when set to zero, the 8 GPMI channels will use its own DMA FIFO.

9.4.5.8 APBH DMA Channel n Current Command Address Register (APBH_CH*n*_CURCMDAR)

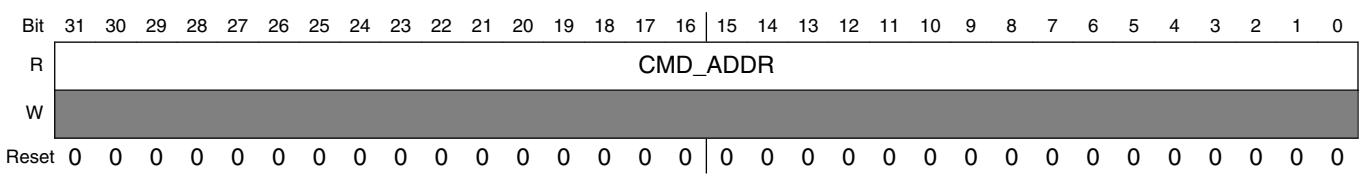
The APBH DMA channel n current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel n is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE

```
pCurCmd = (apbh_chn_cmd_t *) APBH_CHn_CURCMDAR_RD(0);              // read the whole
register, since there is only one field
pCurCmd = (apbh_chn_cmd_t *) BF_RDn(APBH_CHn_CURCMDAR, 0, CMD_ADDR);  // or, use multi-
register bitfield read macro
pCurCmd = (apbh_chn_cmd_t *) APBH_CHn_CURCMDAR(0).CMD_ADDR;        // or, assign from
bitfield of indexed register's struct
```

Address: 3300_0000h base + 100h offset + (112d × i), where i=0d to 15d



APBH_CH*n*_CURCMDAR field descriptions

Field	Description
CMD_ADDR	Pointer to command structure currently being processed for channel n.

9.4.5.9 APBH DMA Channel n Next Command Address Register (APBH_CH*n*_NXTCMDAR)

The APBH DMA Channel n Next Command Address register contains the address of the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to 1 in the DMA command word to process command lists.

APBH DMA Channel n is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel n semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE

```
APBH_CHn_NXTCMDAR_WR(0, (reg32_t) pCommandTwoStructure);         // write the entire
register, since there is only one field
BF_WRn(APBH_CHn_NXTCMDAR, 0, (reg32_t) pCommandTwoStructure);       // or, use multi-
register bitfield write macro
APBH_CHn_NXTCMDAR(0).CMD_ADDR = (reg32_t) pCommandTwoStructure;  // or, assign to bitfield
of indexed register's struct
```

Address: 3300_0000h base + 110h offset + (112d × i), where i=0d to 15d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_CH*n*_NXTCMDAR field descriptions

Field	Description
CMD_ADDR	Pointer to next command structure for channel n.

9.4.5.10 APBH DMA Channel n Command Register (APBH_CH*n*_CMD)

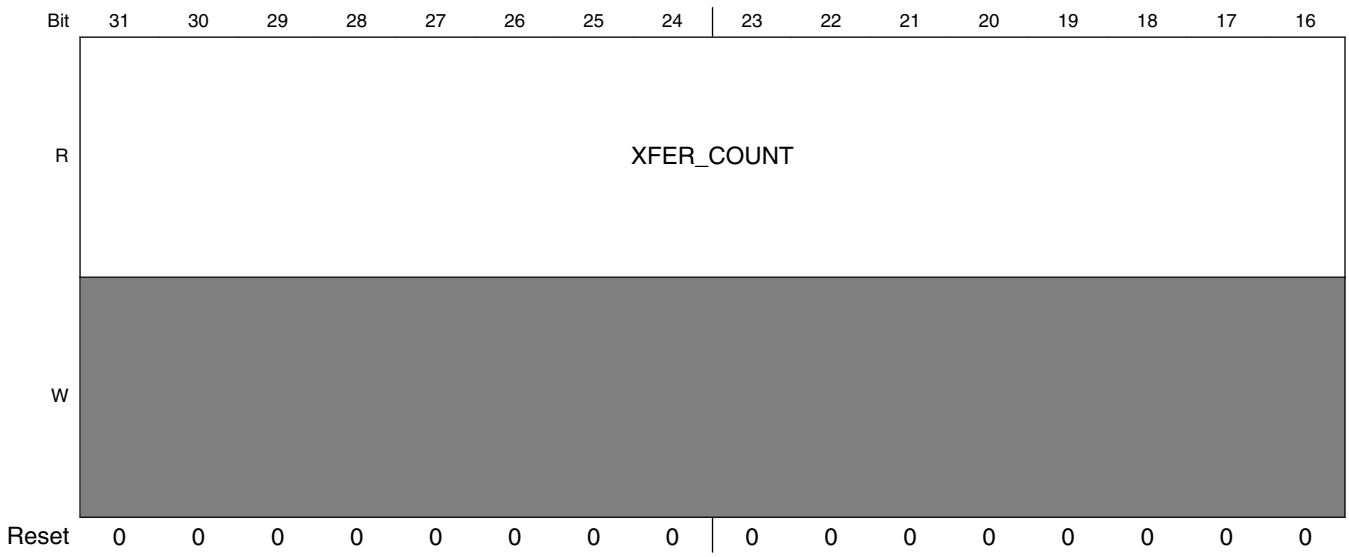
The APBH DMA Channel n command register specifies the DMA transaction to perform for the current command chain item.

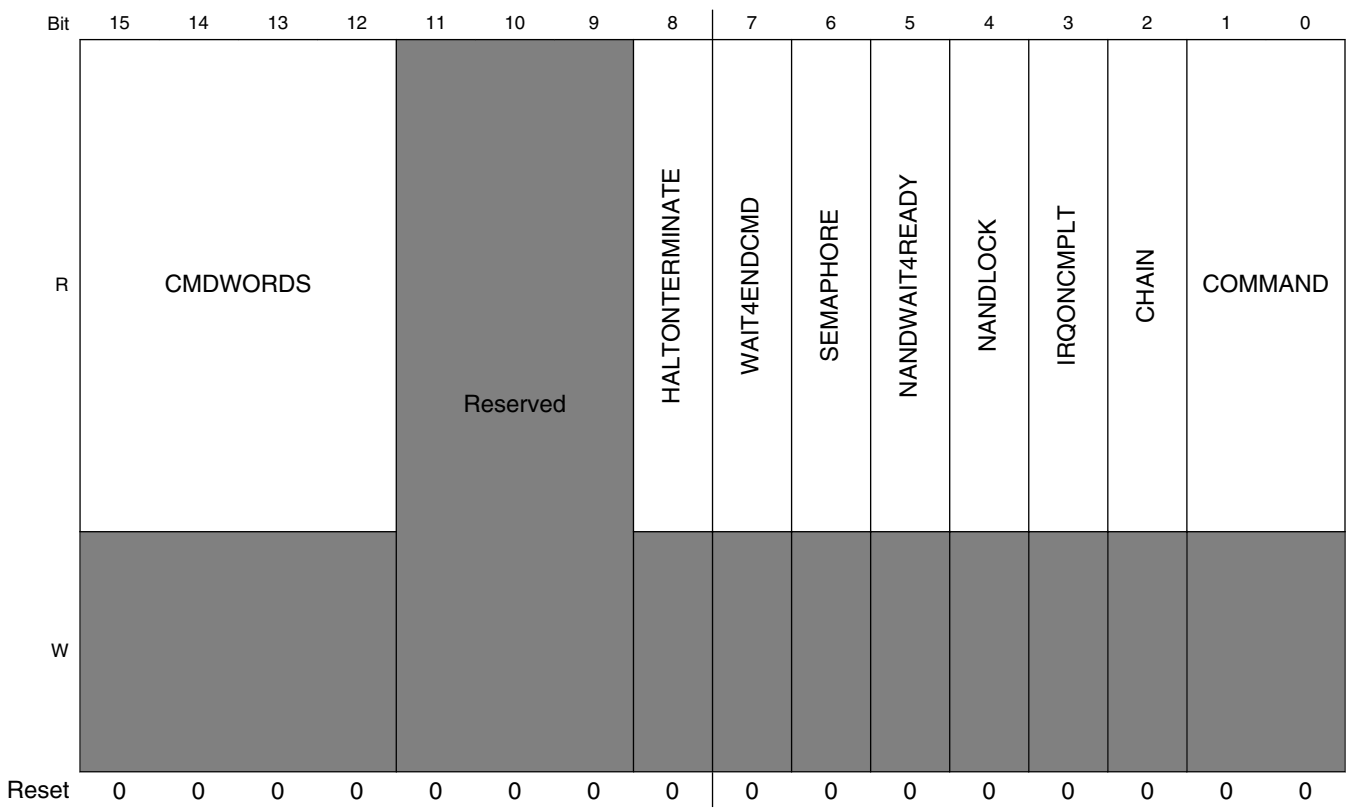
The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE

```
apbh_chn_cmd_t  dma_cmd;
dma_cmd.XFER_COUNT = 512;                                    // transfer 512 bytes
dma_cmd.COMMAND = BV_APBH_CHn_CMD_COMMAND__DMA_WRITE;  // transfer to system memory from
peripheral device
dma_cmd.CHAIN = 1;                                           // chain an additional command
structure on to the list
dma_cmd.IRQONCMPLT = 1;                                      // generate an interrupt on
completion of this command structure
```

Address: 3300_0000h base + 120h offset + (112d × i), where i=0d to 15d





APBH_CH*n*_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI0 device. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the GPMI0, starting with the base PIO address of the GPMI0 control register and incrementing from there. Zero means transfer NO command words
11–9 -	This field is reserved. Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel will will wait until the NAND device reports "ready" before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel will remain "locked" in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in APBH_CHn_CMDAR to find the next command.
COMMAND	This bitfield indicates the type of current command: 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

9.4.5.11 APBH DMA Channel n Buffer Address Register (APBH_CH*n*_BAR)

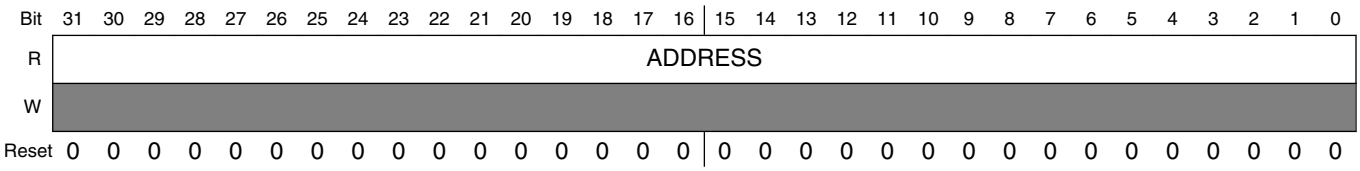
The APBH DMA Channel n buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE

```
apbh_chn_bar_t  dma_data;
dma_data.ADDRESS = (reg32_t) pDataBuffer;
```

Address: 3300_0000h base + 130h offset + (112d × i), where i=0d to 15d



APBH_CH*n*_BAR field descriptions

Field	Description
ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

9.4.5.12 APBH DMA Channel n Semaphore Register (APBH_CH*n*_SEMA)

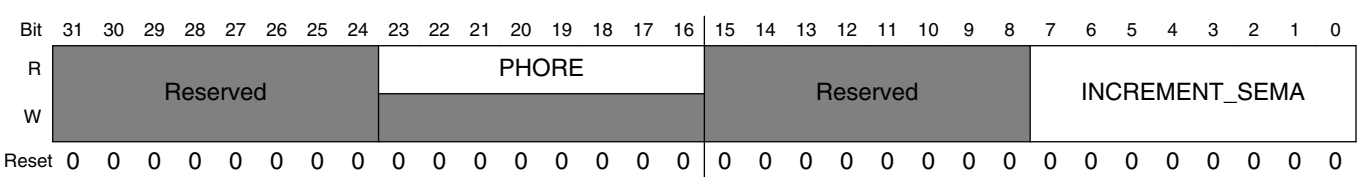
The APBH DMA Channel n semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE

```
BF_WR(APBH_CHn_SEMA, 0, INCREMENT_SEMA, 2);     // increment semaphore by two
current_sema = BF_RD(APBH_CHn_SEMA, 0, PHORE);  // get instantaneous value
```

Address: 3300_0000h base + 140h offset + (112d × i), where i=0d to 15d



APBH_CH*n*_SEMA field descriptions

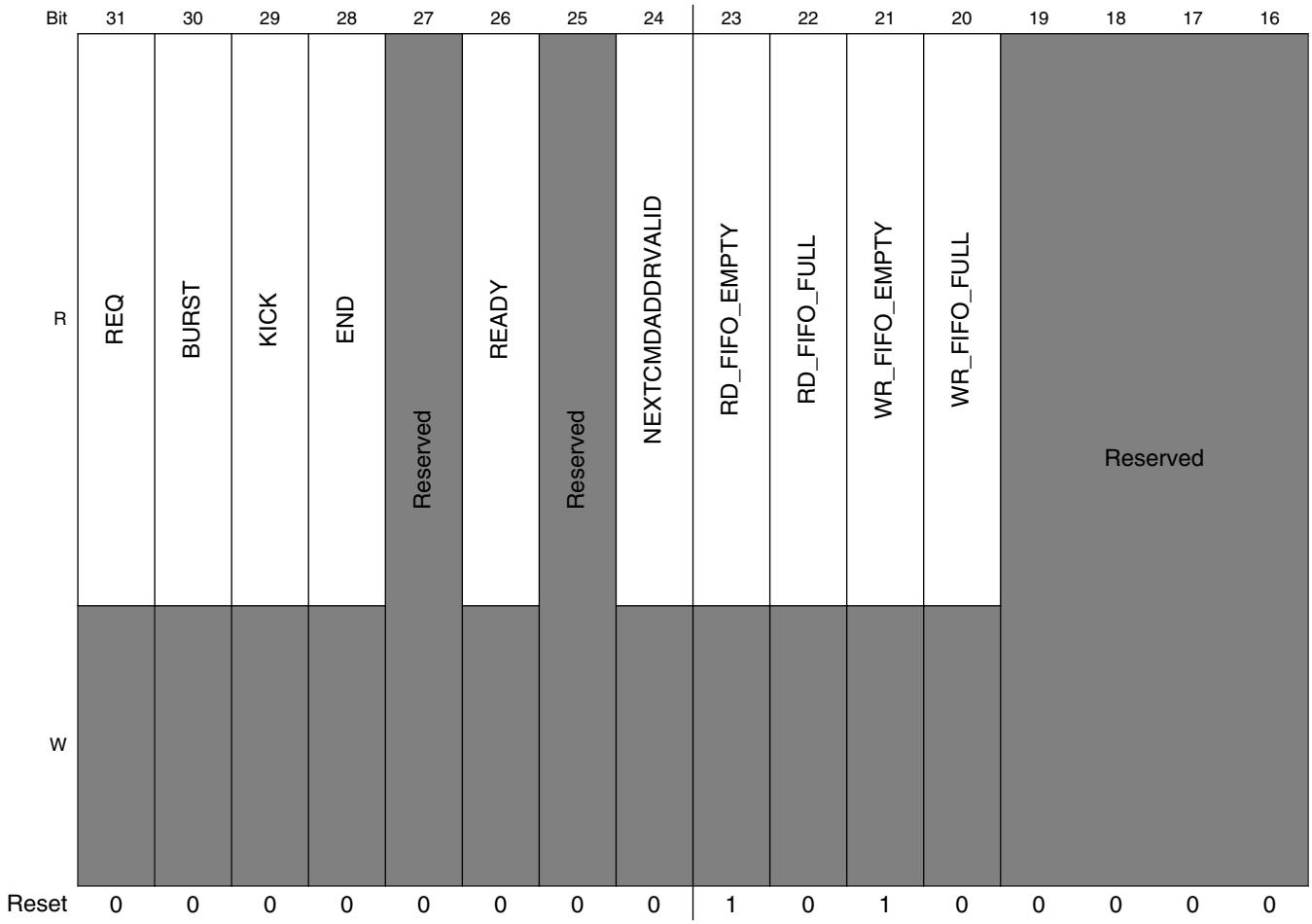
Field	Description
31–24 -	This field is reserved. Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	This field is reserved. Reserved, always set to zero.
INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

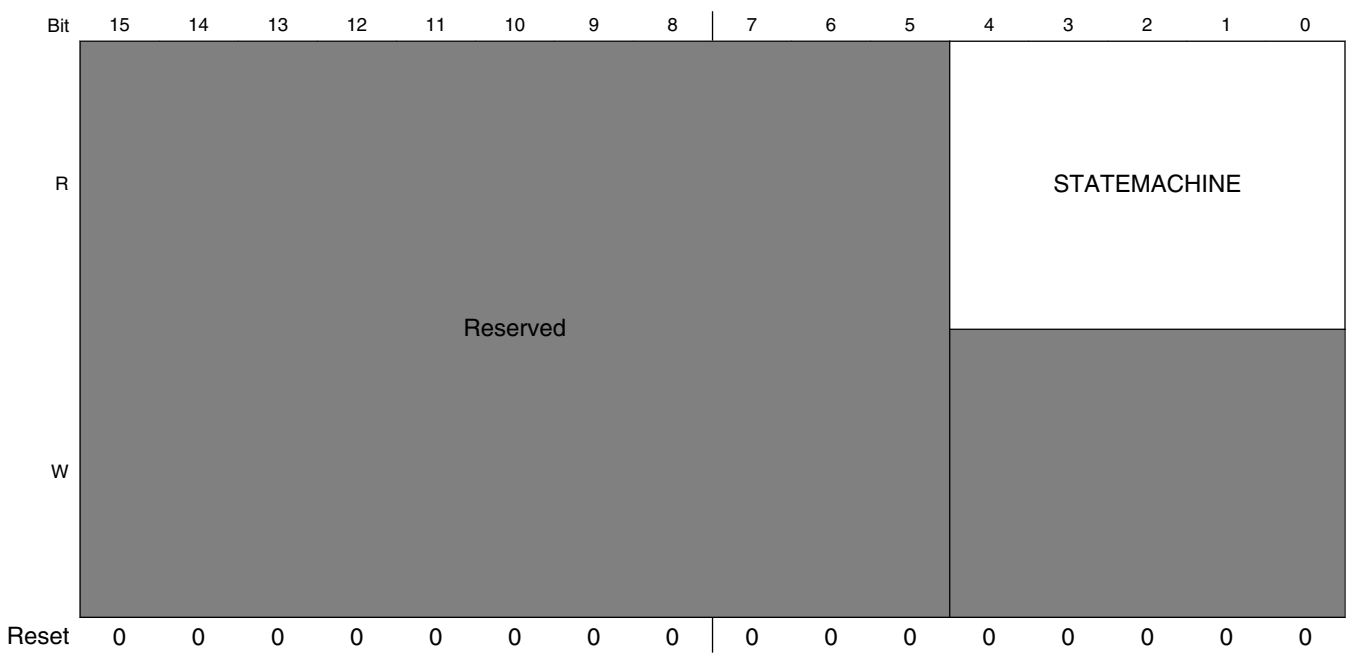
9.4.5.13 AHB to APBH DMA Channel n Debug Information (APBH_CH*n*_DEBUG1)

This register gives debug visibility into the APBH DMA Channel n state machine and controls.

This register allows debug visibility of the APBH DMA Channel n.

Address: 3300_0000h base + 150h offset + (112d × i), where i=0d to 15d





APBH_CH*n*_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27 SENSE	This field is reserved. This bit is reserved for this DMA Channel and always reads 0.
26 READY	This bit is reserved for this DMA Channel and always reads 0.
25 LOCK	This field is reserved. This bit is reserved for this Channel and always reads 0.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	This field is reserved. Reserved
STATEMACHINE	PIO Display of the DMA Channel n state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.

Table continues on the next page...

APBH_CH*n*_DEBUG1 field descriptions (continued)

Field	Description
	0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

9.4.5.14 AHB to APBH DMA Channel n Debug Information (APBH_CH*n*_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel n.

This register allows debug visibility of the APBH DMA Channel n.

Address: 3300_0000h base + 160h offset + (112d × i), where i=0d to 15d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_CH*n*_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

9.4.5.15 APBH Bridge Version Register (APBH_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

This register indicates the RTL version in use.

EXAMPLE

```
if (APBH_VERSION.B.MAJOR != 3)
   Error();
```

Address: 3300_0000h base + 800h offset = 3300_0800h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
STEP	Fixed read-only value reflecting the stepping of the RTL version.

9.5 62BIT Correcting ECC Accelerator (BCH)

9.5.1 Overview

The hardware ECC accelerator provides a forward error-correction function for improving the reliability of various storage media that may be attached to the device.

For example, NAND flash devices use a spare area to store ecc codes to correct some hard bit errors in data stored within the device, allowing higher device yields and, therefore, lower NAND device costs.

The Bose, Ray-Chaudhuri, Hocquenghem (BCH) Encoder and Decoder module is capable of correcting from 2 to 62 single bit errors within a block of data no larger than about 1900 bytes (512 bytes or 1024 bytes are typical) in applications such as protecting data and resources stored on modern NAND flash devices. The correction level in the BCH block is programmable to provide flexibility for varying applications and configurations of flash page size. The design can be programmed to encode protection of 2 to 62 bit errors when writing flash and to correct the corresponding number of errors on decode. The correction level when decoding MUST be programmed to the same correction level as was used during the encode phase.

BCH-codes are a type of block-code, which implies that all error-correction is performed over a block of N-symbols. The BCH operation will be performed over GF(2^{13} = 8192) or GF(2^{14} = 16384), which is the Galois Field consisting of 8191 or 16383 one-bit symbols. BCH-encoding (or encode for any block-code) can be performed by two algorithms: systematic encoding or multiplicative encoding. Systematic encoding is the process of reading all the symbols which constitute a block, dividing continuously these symbols by the generator polynomial for the GF(8192) or GF(16384) and appending the resulting ***t*** parity symbols to the block to create a BCH codeword (where ***t*** is the number of correctable bits).

The BCH encode process creates *t**13 (or *t**14)-bit parity symbols for each data block when the data is written to the flash device. The parity symbols are written to the flash device after the corresponding data block, and together these are collectively called the codeword. The codeword can be used during the decode process to correct errors that occur in either the data or parity blocks.

The BCH decoder processes code words in a 4-step fashion:

1. Syndrome Calculation (SC): This is the process of reading in all of the symbols of the codeword and continuously dividing by the generator polynomial for the field. 2**t* syndromes must be calculated for each codeword and inspection of the syndromes determines if there are errors: a non-zero set of syndromes indicates one or more errors. This process is implemented parallel hardware to minimize processing time since it must be done every time the decode is performed.
2. Key Equation Solver (KES): The syndromes represent 2t-linear equations with 2t-unknown variables. The process of solving these equations and selecting from the numerous solutions constitutes the KES module. When the KES block completes its operations, it generates an error locator polynomial (sigma) that is used in the proceeding block to determine the locations and values of the errors.
3. Chien Search (CS): This block takes input from the KES block and uses the Chien Algorithm for finding the locations of the errors based on the error locator polynomial. The method basically involves substituting all 8191 symbols from the GF(8192) or 16383 symbols from the GF(16383) into the locator polynomial. All evaluations that produce a zero solution indicate locations of the various errors. Since each located error corresponds to a single bit, the bit in the original data may be corrected by simply flipping the polarity of the incorrect location.
4. Correction: this block has to convert the symbol index and mask information to memory byte indexes and masks.

The BCH block, shown in the figure, was designed to operate in a pipelined fashion to maximize throughput. Aside from the initial latency to fill the pipeline stages, the BCH throughput is about 7/4 cycles/byte. Thus, the bottleneck in performing NAND reads and error corrections is the BCH rate. Current GPMI read rates are approximately 1/2 cycles/byte maximally for the current generation of NAND flash. Fortunately, BCH has a different master clock from GPMI, this gives some flexibility to match the throughput rate. The CPU is not directly involved in generating parity symbols, checking for errors, or correcting them.

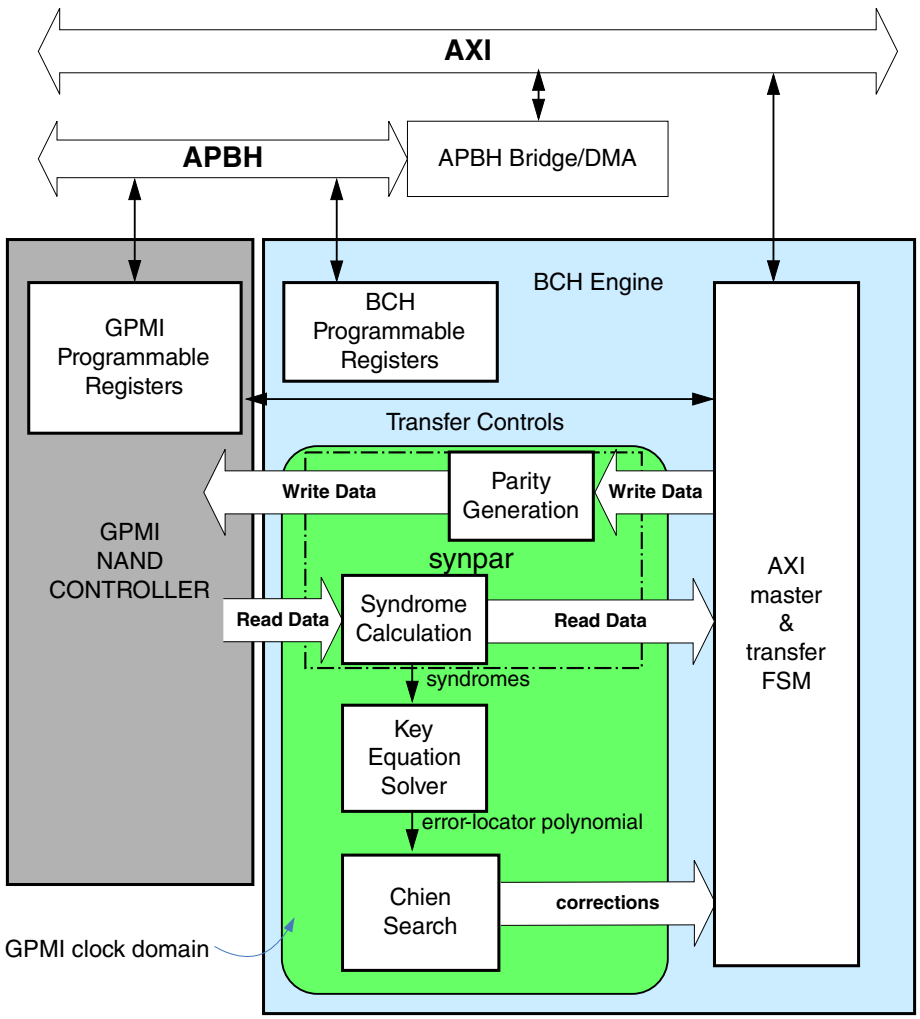


Figure 9-28. Hardware BCH Accelerator

9.5.2 Operation

Before performing any NAND flash read or write operations, software should first program the BCH's flash layout registers (see Flash Page Layout) to specify how data is to be formatted on the flash device.

The BCH hardware allows full programmability over the flash page layout to enable users flexibility in balancing ECC correction levels and ever-changing flash page sizes.

To initiate a NAND Flash write, software will program a GPMI DMA operation. The DMA need only program the GPMI control registers (and handle the requisite flash addressing handshakes) since the BCH will handle all data operations using its AXI bus interface. The BCH will then send the data to the GPMI controller to be written to flash as it computes the parity symbols. At the end of each data block the BCH will insert the parity symbols into the data stream so that the GPMI sees only a continuous stream of data to be written.

NAND Flash read operations operate in a similar manner. As the GPMI controller reads the device, all data is sent to the BCH hardware for error detection/correction. The BCH controller writes all incoming read data to system memory and in parallel computes the syndromes used to detect bit errors. If errors are detected within a block, the BCH hardware activates the error correction logic to determine where bit errors have occurred and ultimately correct them in the data buffer in system memory. After an entire flash page has been read and corrected, the BCH will signal an interrupt to the CPU.

The figure below indicates how data read from the GPMI is operated on within the BCH hardware. As the BCH receives data from the GPMI (top row), it is written to memory by the BCH's Bus Interface Unit (BIU) (second row). For blocks requiring correction, the KES logic will be activated after the entire block has been received. Once the error locator polynomial has been computed, the corrections are determined by the Chien Search and fed back to the BIU, which performs a read, modify, write operation on the buffer in memory to correct the data.

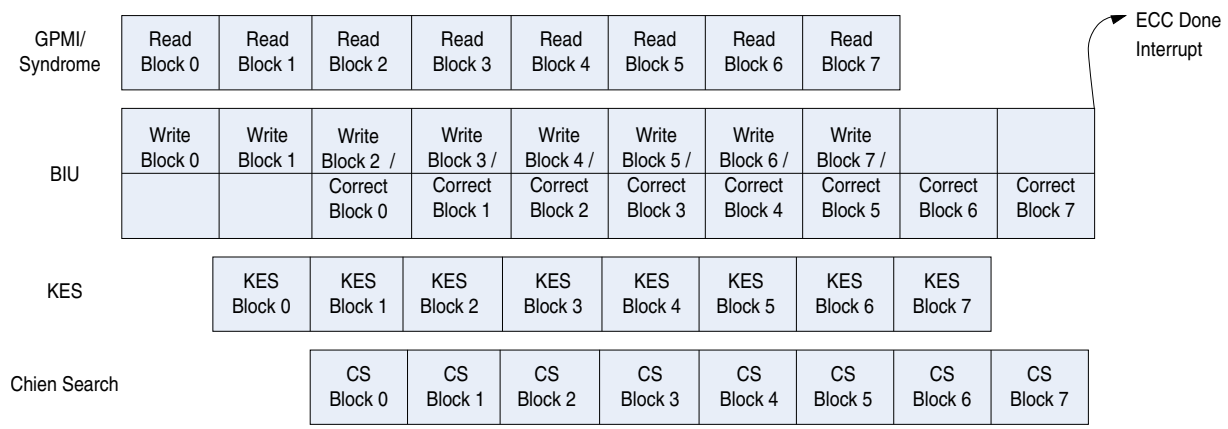


Figure 9-29. Block Pipeline while Reading Flash

9.5.2.1 BCH Limitations and Assumptions

- The BCH is programmable to support 2 to 62 bit error correction. ECC0 is supported as a pass-through, non-correcting mode.
- Data block sizes must be a multiple of 4 bytes and be aligned in system memory.
- The BCH supports a programmable number of metadata/auxiliary data bytes, from 0 to 255.
- Metadata will be written at the beginning of the flash page to facilitate fast access for filesystem operations.
- Metadata may be treated as an independent block for ECC purposes or combined with the first data block to conserve bits in the flash.
- The BCH does not support a partial page write (this can be accomplished by programming the BCH layout registers such that the BCH only sees a portion of the page).
- Flash read operations can read the entire page or the first block on the page.
- The BCH also supports a memory-to-memory mode of operation that does not require the use of DMA or the GPMI.

9.5.2.2 Flash Page Layout

The BCH supports a fully programmable flash page layout. The BCH maintains four independent layout registers that can describe four completely different NAND devices or layouts.

When the BCH initiates an operation, it selects one of the layouts by using the chip select as an index into the BCH_LAYOUTSELECT register that determines which layout should be used for the operation.

Three possible (generic) flash layout schemes are supported, as indicated in the figure below. (In each case, the metadata size may also be programmed to 0 bytes). Metadata may either be combined with the first block of data or the size of the first data block can be programmed to 0 to allow the metadata to be protected by its own ECC parity bits.

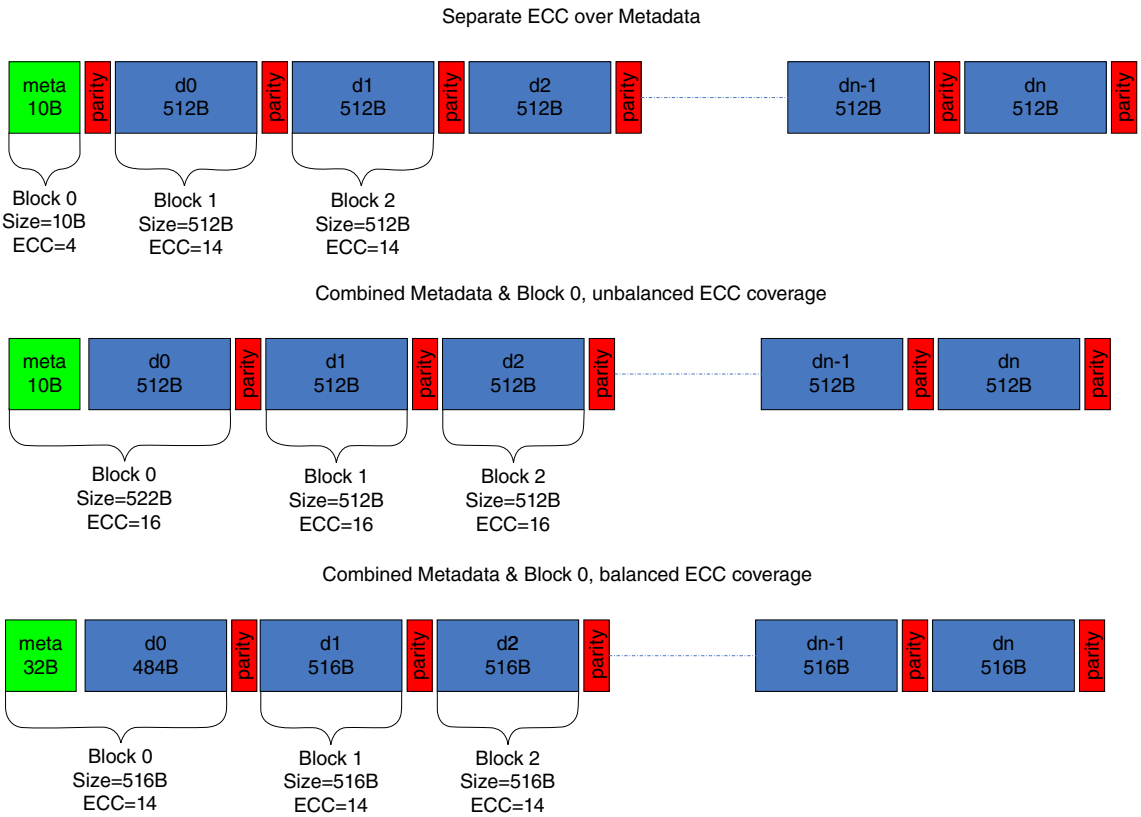


Figure 9-30. FLASH Page Layout Options

Each layout is determined by a pair of registers that define the following parameters:

- DATA0_SIZE: Indicates the number of data bytes in the first block on the page (this should not include parity or metadata bytes). This should be set to 0 when the metadata is to be covered separately with its own ECC. This must be a multiple of 4 bytes.
- ECC0: Indicates the ECC level to be used for the first block on the flash (data0+metadata).
- META_SIZE: Indicates the number of bytes (from 0-255) that are stored as metadata.
- NBLOCKS: Indicates the number of subsequent DATAN blocks on the flash, or the number of blocks following the DATA0 block.
- DATAN_SIZE: Indicates the number of data bytes in all subsequent data blocks. This MUST be a multiple of 4 bytes.
- ECCN: Indicates the ECC level to be used for the subsequent data blocks.

- GF0 or GFN: Indicates the Galois field the meta / data blocks are using
- PAGE_SIZE: Indicates the total number of bytes available per page on the physical flash device. This includes the spare area and is typically 4096+128, 4096+218, or 2048+64 bytes.

9.5.2.3 Determining the ECC layout for a device

Since the BCH is programmable, a system can trade off ECC levels for flash size and layout configurations.

The following examples indicate how to determine a valid layout based on the required storage space and flash size. For all cases, the size of the parity will be 13 (or 14 for GF(2^{14}))*ECC level ***bits***-- so for ECC8, 13 (or 14) bytes are required (per block).

9.5.2.3.1 4K+218 flash, 10 bytes metadata, 512 byte data blocks, separate metadata, Assuming GF(2^{13})

In this case, we have 8 data blocks each consisting of 512 bytes. Since the flash has 218 spare bytes (1744 bits), first estimate an ECC level for the data blocks by first subtracting the number of metadata bytes from the spare bytes (218 – 10 = 208 bytes = 1664 bits) then dividing the number of bits by 8 (number of blocks) and then by 13 (bits per ECC level).

(218 - 10) x 8 = 1664/13(8) = 16

Therefore all the data blocks could be covered by ECC16 if the metadata had no parity. This isn't acceptable, so assume ECC14 for all the data blocks. Now calculate the number of free bits for the metadata parity as

1664 - (14) x 13 x 8 = 208

Therefore, 208 bits remain for metadata parity. Dividing by 13 (bits/ECC) gives 16, so the metadata can be covered with ECC16. The settings for this device would then be

Table 9-19. Settings for 4K+218 FLASH

Setting	Value
PAGE_SIZE	4096+218=4314=0x10DA
META_SIZE	10=0x0A
DATA0_SIZE	0
ECC0	16=0x10
GF0	GF(2^{13})

Table continues on the next page...

Table 9-19. Settings for 4K+218 FLASH (continued)

Setting	Value
DATAN_SIZE	512=0x200 (in register interface, assigned as 0x80)
ECCN	14=0x0E
GFN	GF(2^{13})
NBLOCKS	8

9.5.2.3.2 4K+128 flash, 10 bytes metadata, 1024 byte data blocks, separate metadata, assuming GF(2^{13}) for data and GF(2^{14}) for metadata

This flash will have 118 bytes available for ECC (after subtracting the metadata size), therefore, 994 bits.

Dividing by 4*14 (number of blocks * ECC level) we get 17.75, therefore we can support ECC16 on the data blocks. The number of free spare bits becomes 944 - 16 * 4 * 14 = 944 - 896 = 48, divided by 13 = 3.69, therefore the metadata can be also covered by ECC2.

Table 9-20. Settings for 4K+128 FLASH

Setting	Value
PAGE_SIZE	4096+128=4224=0x1080
META_SIZE	10=0x0A
DATA0_SIZE	0
ECC0	2
GF0	GF(2^{13})
DATAN_SIZE	1024=0x400 (in register interface, assigned as 0x100)
ECCN	16
GFN	GF(2^{14})
NBLOCKS	4

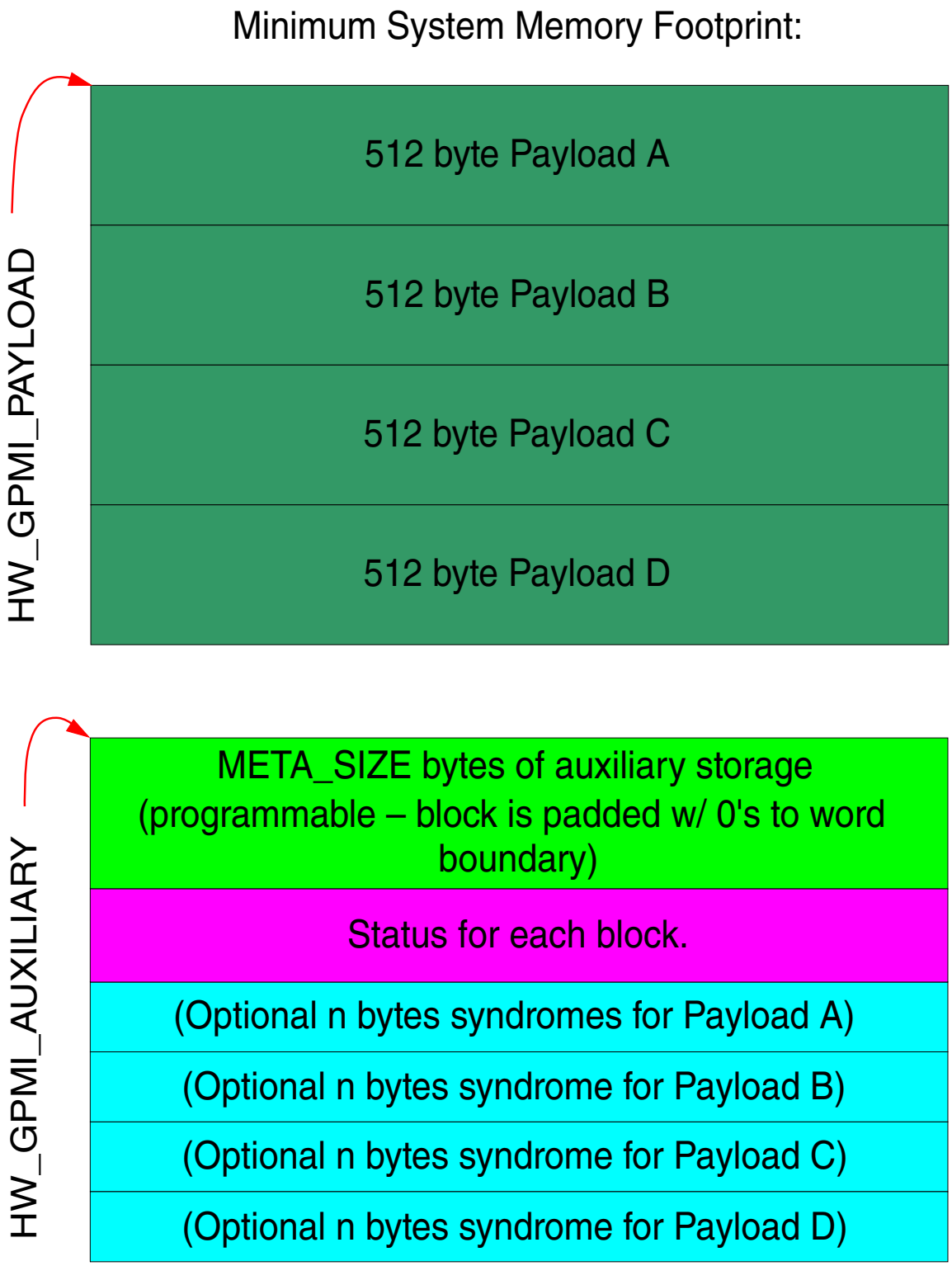
In this case, there will be additional unused spare bits, with the BCH will pad out with zeros.

9.5.2.4 Data Buffers in System Memory

While the data on the flash is interleaved with parity symbols, the BCH assumes that the data buffers in memory are contiguous.

Metadata read from the flash will be stored to the location pointed to by the GPMI_AUXILIARY register and data will be written to the address specified in the GPMI_PAYLOAD register as is shown in the following figure where the block length is 512 bytes for example. Since the number of blocks on a flash page is programmable, the BCH also writes individual block correction status to the auxiliary pointer at the word-aligned address following the end of the metadata. Optionally, the computed syndromes may also be written to the auxiliary area if the DEBUGSYNDROME bit is set in the control register.

As blocks complete processing, the bus master will accumulate the status for each block and write it to the auxiliary data buffer following the metadata. The metadata area will be padded with 0's until the next word boundary and the status for blocks 0-3 will be written to the next word. The status for subsequent blocks will then be written to the buffer. The status for the first block (metadata block) is also stored in the STATUS_BLK0 register in the BCH_STATUS register. The completion codes for the blocks are indicated in the Table 9-21. Note that the definition of the bytes and their ordering in the auxiliary and payload storage areas are user defined. When this data is read back from the flash and put into memory, it will resemble the original buffer that was written out to the flash.



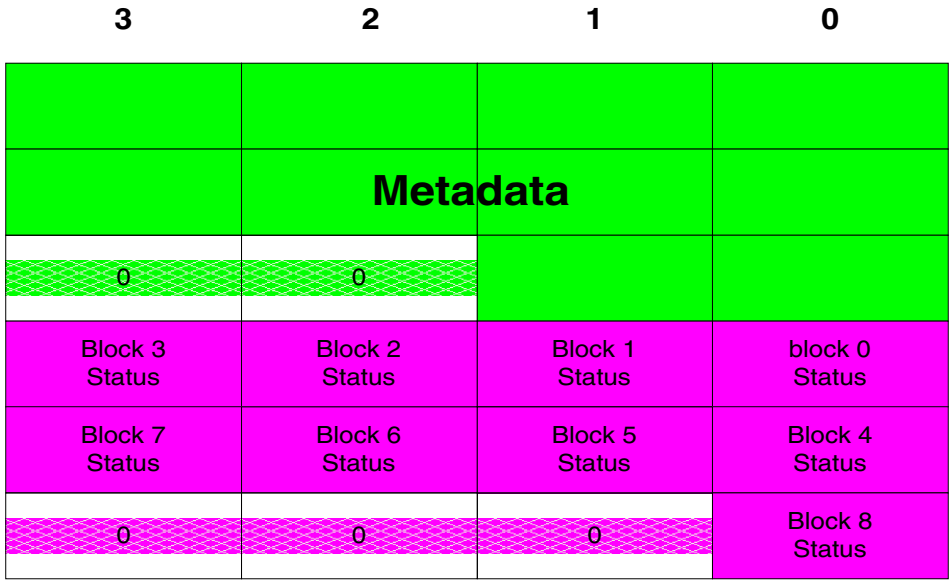
*Computed syndrome area consists of 2*t 13 (or 14)-bit symbols written as 16-bit half word.*

Figure 9-31. BCH Data Buffers in Memory

Table 9-21. Status Block Completion Codes

Code	Description
0xFF	Block is erased
0xFE	Block is uncorrectable
0x00	No errors found
0x01-0x3E	Number of errors corrected

The following figure shows the layout of the bytes within the status field.



Status bytes are allocated based on the NBLOCKS programmed into the flash format register. The number of status bytes will be computed by the NBLOCKS+1. The status area will be padded with zeros to the next word boundary.

Syndrome data written for debug purposes will follow the end of the status block.

Figure 9-32. Memory-to-Memory Operations

9.5.3 Memory to Memory (Loopback) Operation

The BCH supports a memory-to-memory mode of operation where both the encoded and decoded buffers reside in system memory.

This can be useful for applications where data must be protected by ECC, but the storage device does not reside on the GPMI bus.

The BCH operation in memory to memory mode is much simpler than in GPMI mode since DMAs are not required to manage the operation. Instead, software simply writes the BCH_DATAPTR and BCH_METAPTR with the addresses of the data and metadata

(auxiliary) buffers and the BCH_ENCODEPTR with the address of the buffer for encoded data. To initiate the operation, software simply sets the M2M_ENCODE and M2M_ENABLE bits in the control register. The BCH can be programmed to either issue an interrupt at the end of the operation or software may poll the status bits for completion.

Memory to memory decode operations work in a similar manner. The encoded data address is written to the BCH_ENCODEPTR and the data and meta pointers are written to buffers that correspond to the desired decoded data addresses. To initiate a decode, software must set the M2M_ENCODE bit to 0 while writing the M2M_ENABLE bit. Note that the addresses written to the BCH_DATAPTR, BCH_METAPTR and BCH_ENCODEPTR registers should always be aligned on a 4 byte boundary. In other words, the 2 lower bits of the address should always be written with zeros.

9.5.4 Programming the BCH/GPMI Interfaces

Programming the BCH for NAND operations consists largely of disabling the soft reset and clock bits (SFTRST and CLKGATE) from the BCH_CTRL register and then programming the flash layout registers to correspond to the format of the attached NAND device(s).

The BCH_LAYOUTSELECT register should also be programmed to map the chip select of each attached device into one of the four layout registers.

The bulk of the programming is actually applied to the GPMI through PIO operations embedded in DMA command structures. The DMA will perform all the requisite handshaking with the GPMI interface to negotiate the address portion of the transfer, then the BCH will handle all the movement of data from memory to the GPMI (writes) or the GPMI to memory (reads). The BCH will direct all data blocks to the buffer pointed to by the PAYLOAD_BUFFER and the metadata will be written to the AUXILIARY_BUFFER. Both of these registers are located in the GPMI PIO data space and are communicated to the BCH hardware at the beginning of the transfer. Thus, the normal multi-NAND DMA based device interleaving is preserved, that is, four NANDs on four separate chip selects can be scheduled for read or write operations using the BCH. Whichever channel finishes its ready wait first and enters the DMA arbiter with its lock bit set owns the GPMI command interface and through it owns the BCH resources for the duration of its processing.

9.5.4.1 BCH Encoding for NAND Writes

The BCH encoder flowchart in Figure 9-33 shows the detailed steps involved in programming and using the BCH encoder. This flowchart shows how to use the BCH block with the GPMI.

To use the BCH encoder with the GPMI's DMA, create a DMA command chain containing ten descriptor structures, as shown in Figure 9-35 and detailed in the DMA structure code example that follows it in DMA Structure Code Example. The ten descriptors perform the following tasks:

1. Disable the BCH block (in case it was enabled) and issue NAND write setup command byte (under CLE) and address bytes (under ALE).
2. Configure and enable the BCH and GPMI blocks to perform the NAND write.
3. Disable the BCH block and issue NAND write execute command byte (under CLE).
4. Wait for the NAND device to finish writing the data by watching the ready signal.
5. Check for NAND timeout through PSENSE.
6. Issue NAND status command byte (under CLE).
7. Read the status and compare against expected.
8. If status is incorrect or incomplete, branch to error handling descriptor chain.
9. Otherwise, write is complete and emit GPMI interrupt.

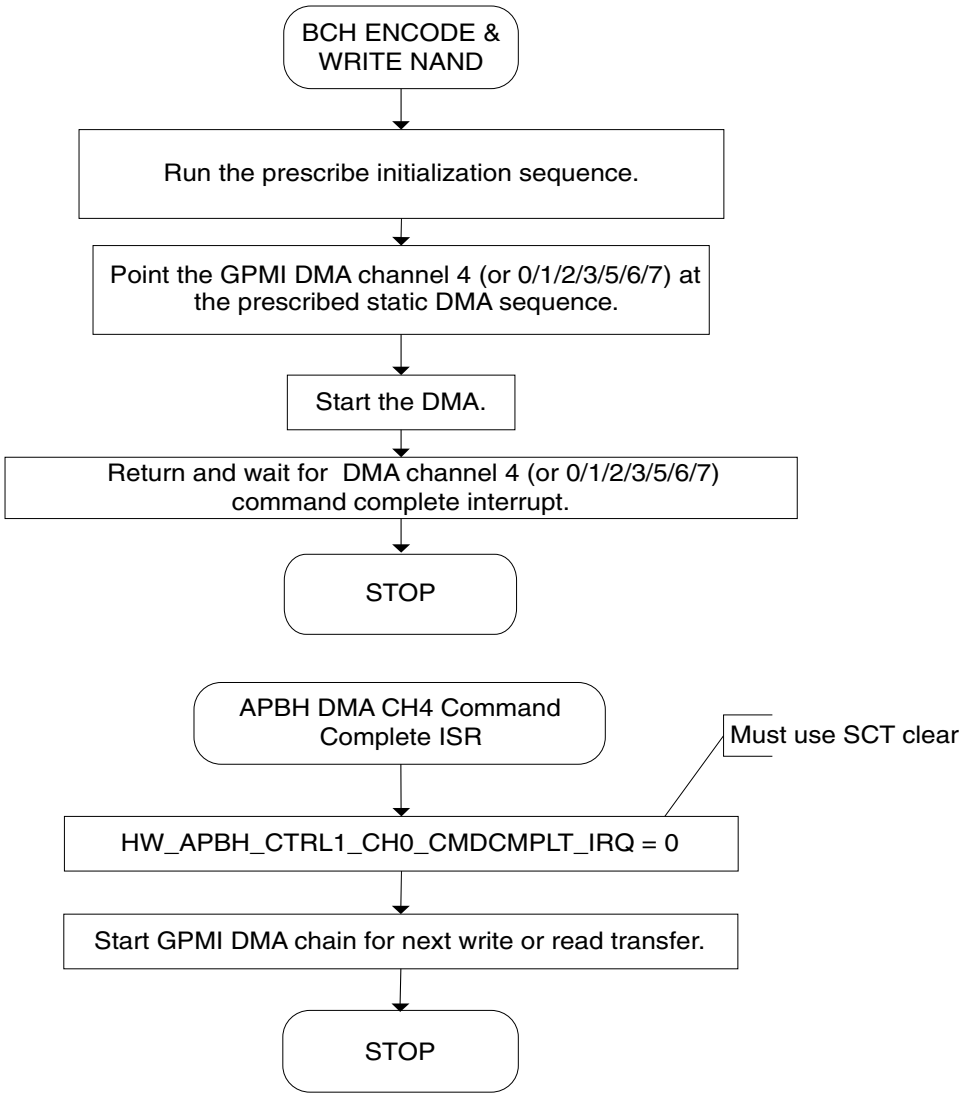


Figure 9-33. BCH Encode Flowchart

Descriptor Legend

NEXT CMD ADDR										
CMD	<=	xfer_count	cmdwords	wait4endcmd	semaphore	nandwait4ready	nandlock	irqoncmplt	chain	command
BUFFER ADDR										
HW_GPMI_CTRL0	<=	command_mode	word_length	lock_cs	CS	address	address_increment	xfer_count		
HW_GPMI_COMPARE	<=	mask				reference				
HW_GPMI_ECCCTRL	<=	ecc_cmd			enable_ecc				buffer_mask	
HW_GPMI_ECCCOUNT										
HW_GPMI_PAYLOAD										
HW_GPMI_AUXILIARY										

Figure 9-34. BCH DMA Descriptor Legend

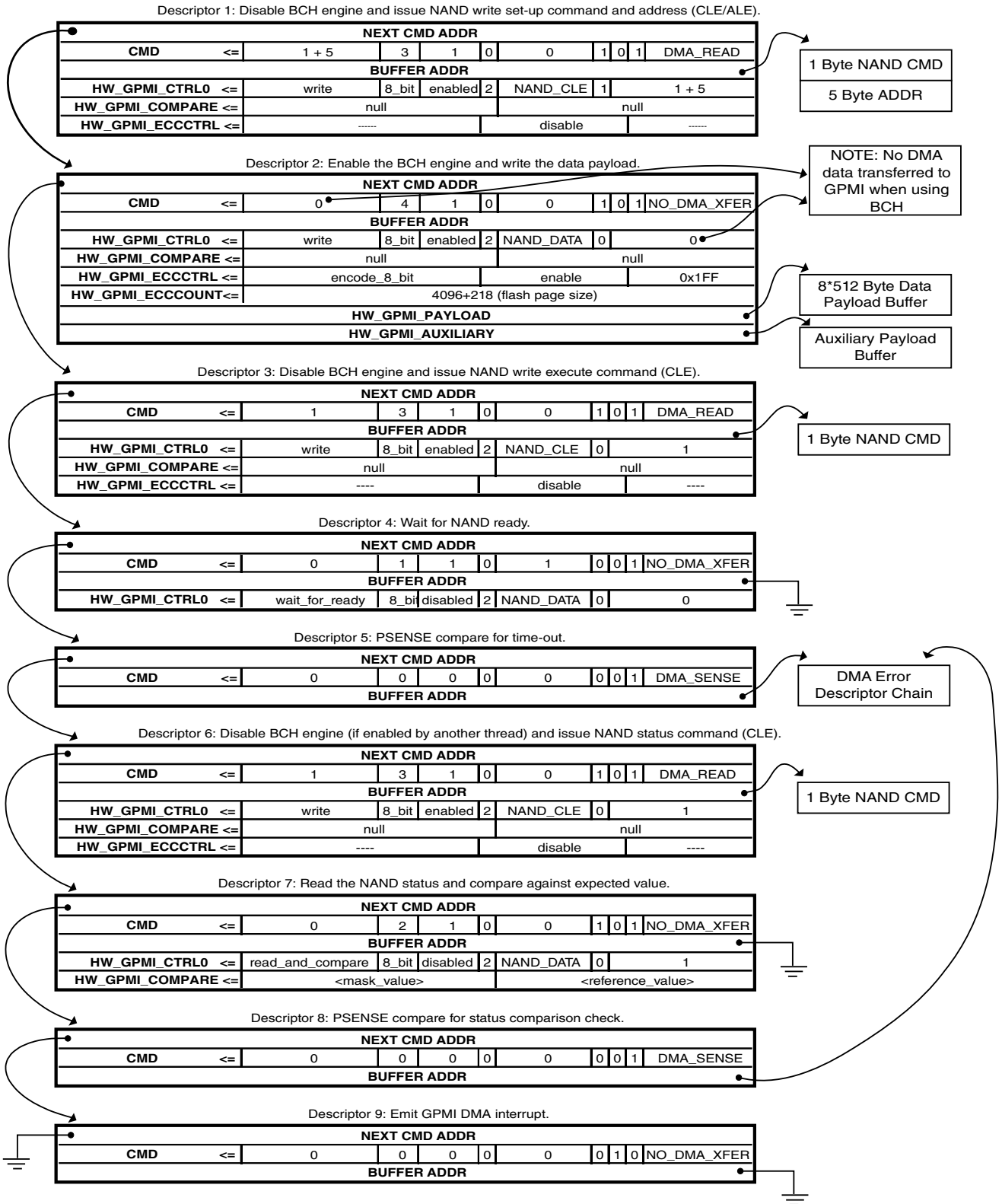


Figure 9-35. BCH Encode DMA Descriptor Chain

9.5.4.1.1 DMA Structure Code Example

The following code sample illustrates the coding for one write transaction involving 4096 bytes of data payload (eight 512-byte blocks) and 10 bytes of auxiliary payload (also referred to as metadata) to a 4K NAND page sitting on GPMI CS2.

```
//-----------------------------------------------------------------------------
// generic DMA/GPMI/ECC descriptor struct, order sensitive!
//-----------------------------------------------------------------------------
typedef struct {
  // DMA related fields
  unsigned int dma_nxtcmdar;
  unsigned int dma_cmd;
  unsigned int dma_bar;
  // GPMI related fields
  unsigned int gpmi_ctrl0;
  unsigned int gpmi_compare;
  unsigned int gpmi_eccctrl;
  unsigned int gpmi_ecccount;
  unsigned int gpmi_data_ptr;
  unsigned int gpmi_aux_ptr;
} GENERIC_DESCRIPTOR;
//-----------------------------------------------------------------------------
// allocate 10 descriptors for doing a NAND ECC Write
//-----------------------------------------------------------------------------
GENERIC_DESCRIPTOR write[10];
//-----------------------------------------------------------------------------
// DMA descriptor pointer to handle error conditions from psense checks
//-----------------------------------------------------------------------------
unsigned int * dma_error_handler;
//-----------------------------------------------------------------------------
// 8 byte NAND command and address buffer
// any alignment is ok, it is read by the GPMI DMA
//   byte 0 is write setup command
//   bytes 1-5 is the NAND address
//   byte 6 is write execute command
//   byte 7 is status command
//-----------------------------------------------------------------------------
unsigned char nand_cmd_addr_buffer[8];
//-----------------------------------------------------------------------------
// 4096 byte payload buffer used for reads or writes
// needs to be word aligned
//-----------------------------------------------------------------------------
unsigned int write_payload_buffer[(4096/4)];
//-----------------------------------------------------------------------------
// 65 byte meta-data to be written to NAND
// needs to be word aligned
//-----------------------------------------------------------------------------
unsigned int write_aux_buffer[65];
//-----------------------------------------------------------------------------
// Descriptor 1: issue NAND write setup command (CLE/ALE)
//-----------------------------------------------------------------------------
write[0].dma_nxtcmdar = &write[1];                          // point to the next descriptor
write[0].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1 + 5)|   // 1 byte command, 5 byte address
                   BF_APBH_CHn_CMD_CMDWORDS      (3)    |   // send 3 words to the GPMI
                 BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)    |  // wait for command to finish
before
                                                            //    continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)    |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)    |
                   BF_APBH_CHn_CMD_NANDLOCK      (1)    |   // prevent other DMA channels
from
                                                            //    taking over
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)    |
                   BF_APBH_CHn_CMD_CHAIN         (1)    |   // follow chain to next command
```

```
                BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ); // read data from DMA, write to
NAND
write[0].dma_bar = &nand_cmd_addr_buffer;           // byte 0 write setup, bytes 1 - 5 NAND
address
// 3 words sent to the GPMI
write[0].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                    BF_GPMI_CTRL0_CS              (2)      | // must correspond to NAND CS
used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (1)       | // send command and
address
                    BF_GPMI_CTRL0_XFER_COUNT       (1 + 5);    // 1 byte command, 5 byte
address
write[0].gpmi_compare = NULL;                            // field not used but necessary to
set eccctrl
write[0].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
//--------------------------------------------------------------------------
// Descriptor 2: write the data payload (DATA)
//--------------------------------------------------------------------------
write[1].dma_nxtcmdar = &write[2];                       // point to the next descriptor
write[1].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT      (0)|  // NOTE: No DMA data transfer
                   BF_APBH_CHn_CMD_CMDWORDS        (4)|  // send 4 words to the GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD     (1)|  // Wait to end
                   BF_APBH_CHn_CMD_SEMAPHORE       (0)|
                   BF_APBH_CHn_CMD_NANDWAIT4READY  (0)|
                   BF_APBH_CHn_CMD_NANDLOCK        (1)|  // maintain resource lock
                   BF_APBH_CHn_CMD_IRQONCMPLT      (0)|
                   BF_APBH_CHn_CMD_CHAIN           (1)|  // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_NO_XFER);  // No data transferred
write[1].dma_bar = &write_payload_buffer;                  // pointer for the 4K byte
data area
// 4 words sent to the GPMI
write[1].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                    BF_GPMI_CTRL0_CS                (2)       | // must correspond to NAND
CS used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)|
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                      BF_GPMI_CTRL0_XFER_COUNT         (0);        // NOTE: this field
contains
                                                                  //  the total amount
                              // DMA transferred to GPMI via DMA (0)!
write[1].gpmi_compare = NULL;                              // field not used but necessary
to
set eccctrl
write[1].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ECC_CMD,    ENCODE_8_BIT) | // specify t = 8
mode
                        BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, ENABLE)       | // enable ECC module
                     BF_GPMI_ECCCTRL_BUFFER_MASK   (0x1FF);          // write all 8 data
blocks
                                                                    //   and 1 aux block
write[1].gpmi_ecccount = BF_GPMI_ECCCOUNT_COUNT(4096+218);      // specify number of bytes
                                                                //   written to NAND
write[1].gpmi_data_pointer = &write_payload_pointer;            // data buffer address
write[1].gpmi_aux_pointer  = &write_aux_pointer;                // metadata pointer
//--------------------------------------------------------------------------
// Descriptor 3: issue NAND write execute command (CLE)
//--------------------------------------------------------------------------
write[2].dma_nxtcmdar = &write[3];                         // point to the next descriptor
write[2].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1) |     // 1 byte command
                   BF_APBH_CHn_CMD_CMDWORDS      (3) |     // send 3 words to the GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1) |     // wait for command to finish
before
                                                          //    continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0) |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0) |
                   BF_APBH_CHn_CMD_NANDLOCK      (1) |     // maintain resource lock
```

```
                    BF_APBH_CHn_CMD_IRQONCMPLT    (0) |
                    BF_APBH_CHn_CMD_CHAIN         (1) |     // follow chain to next command
                BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);   // read data from DMA, write to
NAND
write[2].dma_bar = &nand_cmd_addr_buffer[6];             // point to byte 6, write execute
command
// 3 words sent to the GPMI
write[2].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                    BF_GPMI_CTRL0_CS               (2)      | // must correspond to NAND CS
used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                      BF_GPMI_CTRL0_XFER_COUNT         (1);       // 1 byte command
write[2].gpmi_compare = NULL;                           // field not used but necessary to set
eccctrl
write[2].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
//-------------------------------------------------------------------------
// Descriptor 4: wait for ready (CLE)
//-------------------------------------------------------------------------
write[3].dma_nxtcmdar = &write[4];                       // point to the next descriptor

write[3].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0) |   // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (1) |   // send 1 word to the GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (1) |   // wait for command to finish before
                                                         //      continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0) |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(1) |   // wait for nand to be ready
                   BF_APBH_CHn_CMD_NANDLOCK      (0) |   // relinquish nand lock
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0) |
                   BF_APBH_CHn_CMD_CHAIN         (1) |   // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);          // no dma transfer
write[3].dma_bar = NULL;                                                // field not used
// 1 word sent to the GPMI
write[3].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WAIT_FOR_READY) | // wait for NAND
ready
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)         |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)      |
                  BF_GPMI_CTRL0_CS              (2)         | // must correspond to NAND CS
used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)     |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)            |
                      BF_GPMI_CTRL0_XFER_COUNT         (0);
//-------------------------------------------------------------------------
// Descriptor 5: psense compare (time out check)
//-------------------------------------------------------------------------
write[4].dma_nxtcmdar = &write[5];                         // point to the next descriptor
write[4].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)       | // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (0)       | // no words sent to GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)       | // do not wait to continue
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)       |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)       |
                   BF_APBH_CHn_CMD_NANDLOCK      (0)       |
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)       |
                   BF_APBH_CHn_CMD_CHAIN         (1)       | // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE);  // perform a sense check
write[4].dma_bar = dma_error_handler;                // if sense check fails, branch to error
handler
//-------------------------------------------------------------------------
// Descriptor 6: issue NAND status command (CLE)
//-------------------------------------------------------------------------
write[5].dma_nxtcmdar = &write[6];                         // point to the next descriptor
write[5].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1)     |  // 1 byte command
                   BF_APBH_CHn_CMD_CMDWORDS      (3)     |  // send 3 words to the GPMI
                 BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)     |  // wait for command to finish
before
continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)     |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)     |
```

```
                  BF_APBH_CHn_CMD_NANDLOCK      (1)    |  // prevent other DMA channels from
taking over
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)    |
                   BF_APBH_CHn_CMD_CHAIN         (1)    |  // follow chain to next command
                BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);  // read data from DMA, write to
NAND
write[5].dma_bar = &nand_cmd_addr_buffer[7];                // point to byte 7, status
command
write[5].gpmi_compare = NULL;                            // field not used but necessary to set
eccctrl
write[5].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
// 3 words sent to the GPMI
write[5].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                   BF_GPMI_CTRL0_CS               (2)     | // must correspond to NAND CS
used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                      BF_GPMI_CTRL0_XFER_COUNT         (1);       // 1 byte command
//-------------------------------------------------------------------------
// Descriptor 7: read status and compare (DATA)
//-------------------------------------------------------------------------
write[6].dma_nxtcmdar = &write[7];                          // point to the next descriptor
write[6].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)  |    // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (2)  |    // send 2 words to the GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)  |    // wait for command to finish
before
                                                          //      continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)  |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)  |
                   BF_APBH_CHn_CMD_NANDLOCK      (1)  |    // maintain resource lock
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)  |
                   BF_APBH_CHn_CMD_CHAIN         (1)  |    // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);  // no dma transfer
write[6].dma_bar = NULL;                                    // field not used
// 2 word sent to the GPMI
write[6].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ_AND_COMPARE) | // read from the
                                                                  //       NAND and
//                                                               // compare to expect
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)     |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)  |
                   BF_GPMI_CTRL0_CS             (2)       |   // must correspond to NAND CS
used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)        |
                      BF_GPMI_CTRL0_XFER_COUNT         (1);
write[6].gpmi_compare = <MASK_AND_REFERENCE_VALUE>;  // NOTE: mask and reference values are
NAND
                                                          // SPECIFIC to evaluate the NAND
status
//-------------------------------------------------------------------------
// Descriptor 8: psense compare (time out check)
//-------------------------------------------------------------------------
write[7].dma_nxtcmdar = &write[8];                            // point to the next descriptor
write[7].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)        | // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (0)        | // no words sent to GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)        | // do not wait to continue
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)        |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)        |
                   BF_APBH_CHn_CMD_NANDLOCK      (0)        | // relinquish nand lock
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)        |
                   BF_APBH_CHn_CMD_CHAIN         (1)        | // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE);  // perform a sense check
write[7].dma_bar = dma_error_handler;             // if sense check fails, branch to error
handler
//-------------------------------------------------------------------------
// Descriptor 9: emit GPMI interrupt
//-------------------------------------------------------------------------
write[8].dma_nxtcmdar = NULL;                                 // not used since this is
```

```
last
descriptor
write[8].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT     (0)          | // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS       (0)          | // no words sent to GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD    (0)          | // do not wait to continue
                   BF_APBH_CHn_CMD_SEMAPHORE      (0)          |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)           |
                   BF_APBH_CHn_CMD_NANDLOCK       (0)          |
                   BF_APBH_CHn_CMD_IRQONCMPLT     (1)          | // emit GPMI interrupt
                   BF_APBH_CHn_CMD_CHAIN          (0)          | // terminate DMA chain
processing
                   BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);  // no dma transfer
```

9.5.4.1.2 Using the BCH Encoder

To use the BCH encoder, first turn off the module-wide soft reset bit in both the GPMI and BCH blocks before starting any DMA activity.

Turning off the soft reset must take place by itself, prior to programming the rest of the control registers. Turn off the BCH bus master soft reset bit. Turn off the clock gate bits.

Program the remainder of the GPMI, BCH and APBH DMA as follows:

```
// bring APBH out of reset
APBH_CTRL0_CLR(BM_APBH_CTRL0_SFRST);
APBH_CTRL0_CLR(BM_APBH_CTRL0_CLKGATE);

// bring BCH out of reset
BCH_CTRL_CLR(BM_BCH_CTRL_SFTRST);
BCH_CTRL_CLR(BM_BCH_CTRL_CLKGATE);

// bring gpmi out of reset
GPMI_CTRL0_CLR(BM_GPMI_CTRL0_SFTRST);
GPMI_CTRL0_CLR(BM_GPMI_CTRL0_CLKGATE);
GPMI_CTRL1_SET(BM_GPMI_CTRL1_DEV_RESET | // deassert reset
                   BM_GPMI_CTRL1_BCH_MODE  );    // enable BCH mode

// enable pinctrl
PINCTRL_CTRL_WR(0x00000000);

// enable gpmi pins
PINCTRL_MUXSEL0_CLR(0x0000ffff);    // data bits
PINCTRL_MUXSEL1_CLR(0x03ffffff);    // control bits
PINCTRL_MUXSEL8_CLR(0x0003f3ff);    // control bits
PINCTRL_MUXSEL8_SET(0x00015155);    // control bits
```

Note that for writing NANDs (ECC encoding), only GPMI DMA command complete interrupts are used. The BCH engine is used for writing to the NAND but may optionally produces an interrupt. From the sample code in DMA Structure Code Example:

- DMA descriptor 1 prepares the NAND for data write by using the GPMI to issue a write setup command byte under CLE, then sends a 5-byte address under ALE. The BCH engine is disabled and not used for these commands.
- DMA descriptor 2 enables the BCH engine for encoding to begin the initial writing of the NAND data by specifying where the data and auxiliary payload are coming from in system memory.

- DMA descriptor 3 issues the write commit command byte under CLE to the NAND.
- DMA descriptor 4 waits for the NAND to complete the write commit/transfer by watching the NAND's ready line status. This descriptor relinquishes the NANDLOCK on the GPMI to enable the other DMA channels to initiate NAND transactions on different NAND CS lines.
- DMA descriptor 6 issues a NAND status command byte under "CLE" to check the status of the NAND device following the page write.
- DMA descriptor 7 reads back the NAND status and compares the status with an expected value. If there are differences, then the DMA processing engine follows an error-handling DMA descriptor path.
- DMA descriptor 8 disables the BCH engine and emits a GPMI interrupt to indicate that the NAND write has been completed.

9.5.4.2 BCH Decoding for NAND Reads

When a page is read from NAND flash, BCH syndromes will be computed and, if correctable errors are found, they will be corrected on a per block basis within the NAND page.

This decoding process is fully overlapped with other NAND data reads and with CPU execution. The BCH decoder flowchart in the figure below shows the steps involved in programming the decoder. The hardware flow of reading and decoding a 4096-byte page is shown in Figure 9-37.

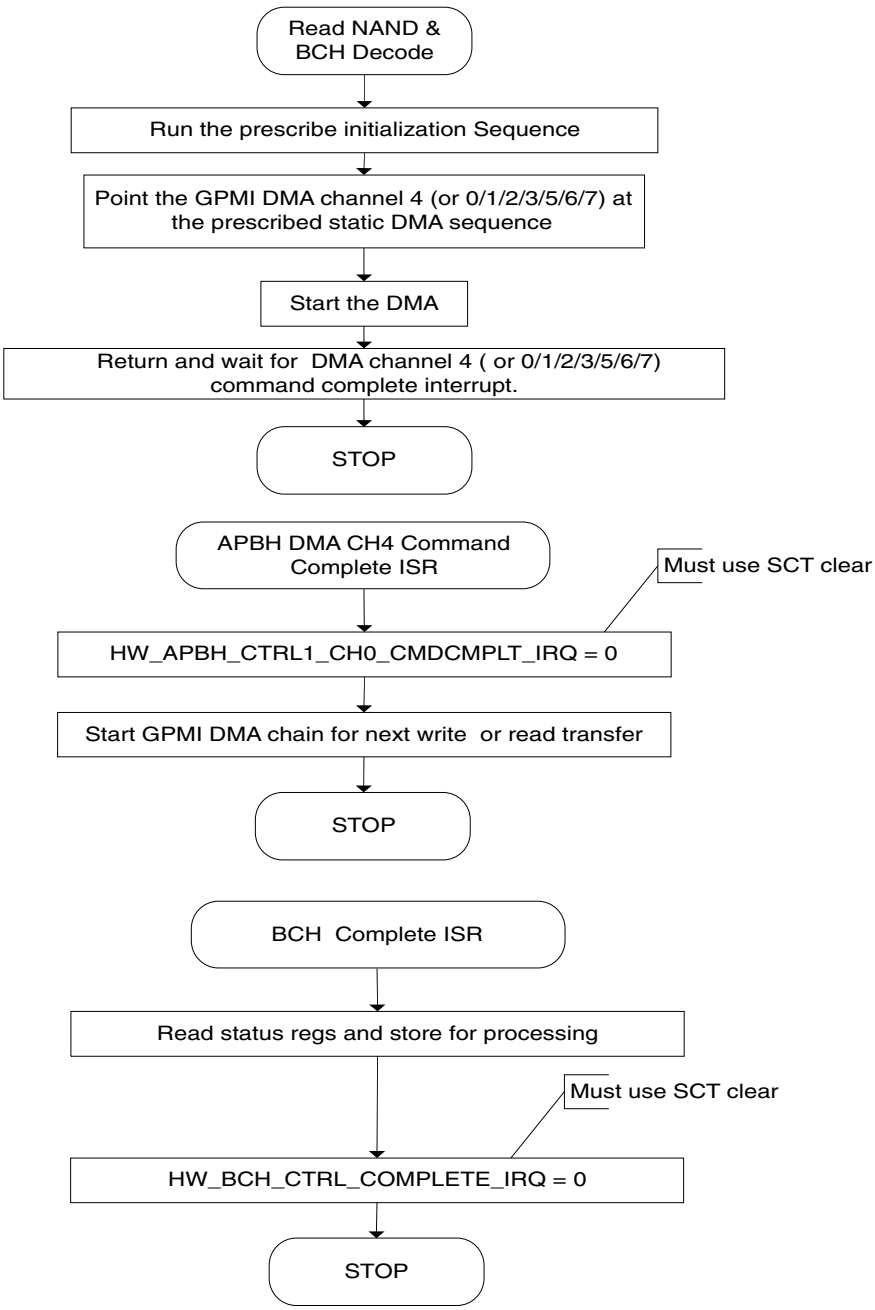


Figure 9-36. BCH Decode Flowchart

Conceptually, an APHB DMA Channel command chain with seven command structures linked together is used to perform the BCH decode operation (as shown in Figure 9-37).

Note

The GPMI's DMA command structures controls the BCH decode operation.

To use the BCH decoder with the GPMI's DMA, create a DMA command chain containing seven descriptor structures, as shown in the figure below and detailed in the DMA structure code example that follows it in DMA Structure Code Example. The seven DMA descriptors perform the following tasks:

1. Issue NAND read setup command byte (under "CLE") and address bytes (under "ALE").
2. Issue NAND read execute command byte (under "CLE").
3. Wait for the NAND device to complete accessing the block data by watching the ready signal.
4. Check for NAND timeout through "PSENSE".
5. Configure and enable the BCH block and read the NAND block data.
6. Disable the BCH block.
7. Descriptor NOP to allow NANDLOCK in the previous descriptor to the thread-safe.

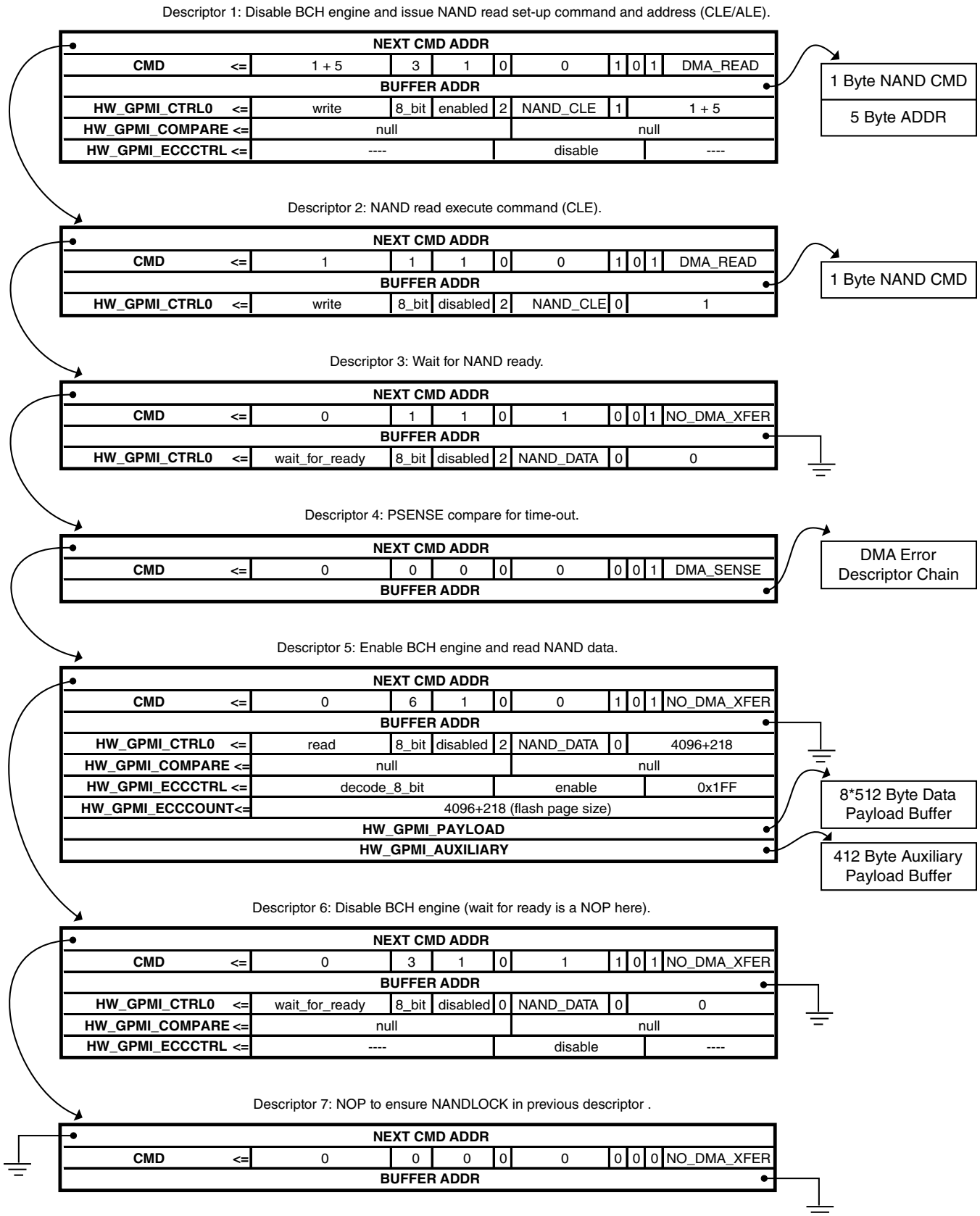


Figure 9-37. BCH Decode DMA Descriptor Chain

9.5.4.2.1 DMA Structure Code Example

The following sample code illustrates the coding for one read transaction, consisting of a seven DMA command structure chain for reading all 4096 bytes of payload data (eight 512-byte blocks) and 65 bytes of metadata with the associative parity bytes (8 * (18) + 9) from a 4K NAND page sitting on GPMI CS2.

```
//-----------------------------------------------------------------------------
// generic DMA/GPMI/ECC descriptor struct, order sensitive!
//-----------------------------------------------------------------------------
typedef struct {
  // DMA related fields
  unsigned int dma_nxtcmdar;
  unsigned int dma_cmd;
  unsigned int dma_bar;
  // GPMI related fields
  unsigned int gpmi_ctrl0;
  unsigned int gpmi_compare;
  unsigned int gpmi_eccctrl;
  unsigned int gpmi_ecccount;
  unsigned int gpmi_data_ptr;
  unsigned int gpmi_aux_ptr;
} GENERIC_DESCRIPTOR;
//-----------------------------------------------------------------------------
// allocate 7 descriptors for doing a NAND ECC Read
//-----------------------------------------------------------------------------
GENERIC_DESCRIPTOR read[7];
//-----------------------------------------------------------------------------
// DMA descriptor pointer to handle error conditions from psense checks
//-----------------------------------------------------------------------------
unsigned int * dma_error_handler;
//-----------------------------------------------------------------------------
// 7 byte NAND command and address buffer
// any alignment is ok, it is read by the GPMI DMA
//   byte 0 is read setup command
//   bytes 1-5 is the NAND address
//   byte 6 is read execute command
//-----------------------------------------------------------------------------
unsigned char nand_cmd_addr_buffer[7];
//-----------------------------------------------------------------------------
// 4096 byte payload buffer used for reads or writes
// needs to be word aligned
//-----------------------------------------------------------------------------
unsigned int read_payload_buffer[(4096/4)];
//-----------------------------------------------------------------------------
// 412 byte auxiliary buffer used for reads
// needs to be word aligned
//-----------------------------------------------------------------------------
unsigned int read_aux_buffer[(412/4)];
//-----------------------------------------------------------------------------
// Descriptor 1: issue NAND read setup command (CLE/ALE)
//-----------------------------------------------------------------------------
read[0].dma_nxtcmdar = &read[1];                              // point to the next descriptor
read[0].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1 + 5) |  // 1 byte command, 5 byte address
                  BF_APBH_CHn_CMD_CMDWORDS      (3)     |  // send 3 words to the GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)     |  // wait for command to finish
                                                           //     before continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)     |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)     |
                  BF_APBH_CHn_CMD_NANDLOCK      (1)     |  // prevent other DMA channels from
                                                           //    taking over
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)     |
                  BF_APBH_CHn_CMD_CHAIN         (1)     |   // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);  // read data from DMA, write to
```

```
NAND
read[0].dma_bar = &nand_cmd_addr_buffer;            // byte 0 read setup, bytes 1 - 5 NAND
address
// 3 words sent to the GPMI
read[0].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)     | // write to the NAND
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)     |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)   |
                     BF_GPMI_CTRL0_CS                 (2)        | // must correspond to NAND
CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE)  |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (1)        | // send command and address
                     BF_GPMI_CTRL0_XFER_COUNT         (1 + 5);    // 1 byte command, 5 byte
address
read[0].gpmi_compare = NULL;                         // field not used but necessary to set
eccctrl
read[0].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
//-------------------------------------------------------------------------
// Descriptor 2: issue NAND read execute command (CLE)
//-------------------------------------------------------------------------
read[1].dma_nxtcmdar = &read[2];                           // point to the next descriptor
read[1].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1)    |   // 1 byte read command
                  BF_APBH_CHn_CMD_CMDWORDS      (1)    |   // send 1 word to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)    |   // wait for command to finish
before
                                                         //    continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)    |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)    |
                  BF_APBH_CHn_CMD_NANDLOCK      (1)    |   // prevent other DMA channels from
                                                         //     taking over
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)    |
                  BF_APBH_CHn_CMD_CHAIN         (1)    |   // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);     // read data from DMA, write
to NAND
read[1].dma_bar = &nand_cmd_addr_buffer[6];              // point to byte 6, read execute
command
// 1 word sent to the GPMI
read[1].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)     | // write to the NAND
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)     |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)  |
                     BF_GPMI_CTRL0_CS                 (2)        | // must correspond to NAND
CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE)  |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)        |
                     BF_GPMI_CTRL0_XFER_COUNT         (1);        // 1 byte command
//-------------------------------------------------------------------------
// Descriptor 3: wait for ready (DATA)
//-------------------------------------------------------------------------
read[2].dma_nxtcmdar = &read[3];                           // point to the next descriptor
read[2].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)      | // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (1)      | // send 1 word to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)      | // wait for command to finish
before
                                                         //     continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)      |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(1)      | // wait for nand to be ready
                  BF_APBH_CHn_CMD_NANDLOCK      (0)      | // relinquish nand lock
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)      |
                  BF_APBH_CHn_CMD_CHAIN         (1)      | // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);   // no dma transfer
read[2].dma_bar = NULL;                                    // field not used
// 1 word sent to the GPMI
read[2].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WAIT_FOR_READY) | // wait for NAND
ready
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)          |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)       |
                     BF_GPMI_CTRL0_CS                 (2)             | // must correspond
to NAND CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)      |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)             |
                     BF_GPMI_CTRL0_XFER_COUNT         (0);
```

```
//------------------------------------------------------------------------
// Descriptor 4: psense compare (time out check)
//------------------------------------------------------------------------
read[3].dma_nxtcmdar = &read[4];                                    // point to the next
descriptor
read[3].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)         |       // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (0)         |       // no words sent to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)         |       // do not wait to continue
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)         |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)         |
                  BF_APBH_CHn_CMD_NANDLOCK      (0)         |
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)         |
                  BF_APBH_CHn_CMD_CHAIN         (1)         |       // follow chain to next
command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE);         // perform a sense check
read[3].dma_bar = dma_error_handler;                        // if sense check fails, branch to
error handler
//------------------------------------------------------------------------
// Descriptor 5: read 4K page plus 65 byte meta-data Nand data
//               and send it to ECC block (DATA)
//------------------------------------------------------------------------
read[4].dma_nxtcmdar = &read[5];                              // point to the next descriptor
read[4].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)     | // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (6)     | // send 6 words to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)     | // wait for command to finish before
                                                          //       continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)     |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)     |
                  BF_APBH_CHn_CMD_NANDLOCK      (1)     | // prevent other DMA channels from
taking over
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)     | // ECC block generates BCH interrupt
                                                          //      on completion
                  BF_APBH_CHn_CMD_CHAIN         (1)     | // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);     // no DMA transfer,
                                                                  // ECC block handles
transfer
read[4].dma_bar = NULL;                                           // field not used
// 6 words sent to the GPMI
read[4].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ)       |  // read from the NAND
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)      |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)   |
                     BF_GPMI_CTRL0_CS                 (2)         |  // must correspond to
NAND CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)  |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)         |
                     BF_GPMI_CTRL0_XFER_COUNT         (4096+218);    // eight 512 byte data
blocks
                                                                     // metadata, and parity

read[4].gpmi_compare = NULL;                              // field not used but necessary to set
eccctrl
// GPMI ECCCTRL PIO This launches the 4K byte transfer through BCH's
// bus master. Setting the ECC_ENABLE bit redirects the data flow
// within the GPMI so that read data flows to the BCH engine instead
// of flowing to the GPMI's DMA channel.
read[4].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ECC_CMD,    DECODE_8_BIT) |    // specify t = 8
mode
                       BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, ENABLE)       |    // enable ECC
module
                       BF_GPMI_ECCCTRL_BUFFER_MASK     (0X1FF);  // read all 8 data blocks
and 1 aux block
read[4].gpmi_ecccount = BF_GPMI_ECCCOUNT_COUNT(4096+218);         // specify number of bytes
                                                                  //        read from NAND
read[4].gpmi_data_ptr = &read_payload_buffer;                     // pointer for the 4K byte
                                                                  //    data area
read[4].gpmi_aux_ptr = &read_aux_buffer;                          // pointer for the 65 byte
aux area +
                                                                  // parity and syndrome
bytes for both
                                                                 // data and aux blocks.
```

```
//-------------------------------------------------------------------------
// Descriptor 6: disable ECC block
//-------------------------------------------------------------------------
read[5].dma_nxtcmdar = &read[6];                              // point to the next descriptor
read[5].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)       | // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (3)       | // send 3 words to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)       | // wait for command to finish
before
                                                            //     continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)       |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(1)       | // wait for nand to be ready
                  BF_APBH_CHn_CMD_NANDLOCK      (1)       | // need nand lock to be
                                                            // thread safe while turn-off BCH
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)       |
                  BF_APBH_CHn_CMD_CHAIN         (1)       | // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);      // no dma transfer
read[5].dma_bar = NULL;                                       // field not used
// 3 words sent to the GPMI
read[5].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ)      |
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)     |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)  |
                     BF_GPMI_CTRL0_CS                 (2)       | // must correspond to
NAND CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA) |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                     BF_GPMI_CTRL0_XFER_COUNT         (0);
read[5].gpmi_compare = NULL;                                  // field not used but necessary to set
eccctrl
read[5].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE);  // disable the ECC block
//-------------------------------------------------------------------------
// Descriptor 7: deassert nand lock
//-------------------------------------------------------------------------
read[6].dma_nxtcmdar = NULL;                                  // not used since this is last
descriptor
read[6].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)   |     // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (0)   |     // no words sent to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)   |     // wait for command to finish
before
                                                            //        continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)   |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)   |
                  BF_APBH_CHn_CMD_NANDLOCK      (0)   |     // relinquish nand lock
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)   |     // BCH engine generates interrupt
                  BF_APBH_CHn_CMD_CHAIN         (0)   |     // terminate DMA chain processing
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);      // no dma transfer
read[6].dma_bar = NULL;                                       // field not used
```

9.5.4.2.2 Using the Decoder

As illustrated in Figure 9-37 and the sample code in DMA Structure Code Example :

- DMA descriptor 1 prepares the NAND for data read by using the GPMI to issue a NAND read setup command byte under CLE, then sends a 5-byte address under ALE. The BCH engine is not used for these commands.
- DMA descriptor 2 issues a one-byte read execute command to the NAND device that triggers its read access. The NAND then goes not ready.
- DMA descriptor 3 performs a wait for ready operation allowing the DMA chain to remain dormant until the NAND device completes its read access time.

- DMA descriptor 5 handles the reading and error correction of the NAND data. This command's PIOs activate the BCH engine to write the read NAND data to system memory and to process it for any errors that need to be corrected. This DMA descriptor contains two PIO values that are system memory addresses pointing to the PAYLOAD data area and to the AUXILIARY data area. These addresses are used by the BCH engine's AHB master to move data into system memory and to correct it. While this example is reading an entire 4K page—payload plus metadata—it is equally possible to read just one 512-byte payload block or just the uniquely protected metadata block in a single 7 DMA structure transfer.
- DMA descriptor 6 disables the BCH engine with the NANDLOCK asserted. This is necessary to ensure that the GPMI resource is not arbitrated to another DMA channel when multiple DMA channels are active concurrently.
- DMA descriptor 7 deasserts the NANDLOCK to free up the GPMI resource to another channel.

As the BCH block receives data from the GPMI:

- The decoder transforms the read NAND data block into a BCH code word and computes the codeword syndrome.
- If no errors are present, then the BCH block can immediately report back to firmware. This report is passed as the BCH_CTRL_COMPLETE_IRQ interrupt status bit and the associated status registers in BCH_STATUS0/1 registers.
- If an error is present, then the BCH block corrects the necessary data block or parity block bytes, if possible (not all errors are correctable).

As the BCH decoder reads the data and parity blocks, it records a special condition, i.e., that all of the bits of a payload data block or metadata block are one, including any associated parity bytes. The all-ones case for both parity and data indicates an erased block in the NAND device.

The BCH_STATUS0 register contains a 4-bit field that indicates the final status of the auxiliary block. A value of 0x0 indicates no errors found for a block.

- A value of 1 to 20 inclusive indicates that many correctable errors were found and fixed.
- A value of 0xFE indicates uncorrectable errors detected on the block.

- A value of 0xFF indicates that the block was in the special ALL ONES state and is therefore considered to be an ERASED block.
- All other values are disallowed by the hardware design.

Recall that up to eight NAND devices can have DMA chains in-flight at once, i.e. they can all be contending for access to the GPMI data bus. It is impossible to predict which NAND device will enter the BCH engine with a transfer first, because each chain includes a wait4ready command structure. As a result, firmware should look at the BCH_STATUS0_COMPLETED_CE bit field to determine which block is being reported in the status register. There is also a 16-bit HANDLE field in the GPMI_ECCCTRL register that is passed down the pipeline with each transaction. This handle field can be used to speed firmware's detection of which transaction is being reported.

These examples of reading and writing have focused on full page transfers of 4K page NAND devices. Other device configurations can be specified by changing the ECCOUNT field in the GPMI registers and reprogramming the BCH's FLASHnLAYOUTm registers.

The BCH and GPMI blocks are designed to be very efficient at reading single 512 (or 1024)-byte pages in one transaction. With no errors, the transaction takes less than 20 HCLKs longer than the time to read the raw data from the NAND.

To summarize, the APBH DMA command chain for a BCH decode operation is shown in Figure 9-37. Seven DMA command structures must be present for each NAND read transaction decoded by the BCH. The seven DMA command structures for multiple NAND read transaction blocks can be chained together to make larger units of work for the BCH, and each will produce an appropriate error report in the BCH PIO space. Multiple NAND devices can have such multiple chains scheduled. The results can come back out of order with respect to the multiple chains.

9.5.4.3 Interrupts

There are two interrupt sources used in processing BCH protected NAND read and write transfers.

Since all BCH operations are initiated by GPMI DMA command structures, the DMA completion interrupt for the GPMI is an important ISR. Both of the flow charts of Figure 9-33 and Figure 9-36 show the GPMI DMA complete ISR skeleton. In both reads and writes, the GPMI DMA completion interrupt is used to schedule work ***INTO*** the error correction pipeline. As the front end processing completes, the DMA interrupt is

generated and additional work, such as DMA chains, are passed to the GPMI DMA to keep it *fed*. For write operations, this is the only interrupt that is generated for processing the NAND write transfer.

For reads, however, two interrupts are needed. Every read is started by a GPMI DMA command chain and the front end queue is fed as described above. The back end of the read pipeline is drained by monitoring the BCH completion interrupt found int HW_BCH_CTRL_COMPLETE_IRQ.

An BCH transaction consists of reading or writing all of the blocks requested in the HW_GPMI_ECCCTRL_BUFFER_MASK bit field. As every read transaction completes, it posts the status of all of the blocks to the HW_BCH_STATUS0 and HW_BCH_STATUS1 registers and sets the completion interrupt. The five stages of the BCH read pipeline completes, one in the GPMI and four in the BCH, are independently stalled as they complete and try to deliver to the next stage in the data flow. Several of these stages can be skipped if no-errors are found or once an uncorrectable error is found in a block.

In any case, the final stage will stall if the status register is busy waiting for the CPU to take status register results. The hardware monitors the state of the HW_BCH_CTRL_COMPLETE_IRQ bit. If it is still set when the last pipeline stage is ready to post data, then the stage will stall. It follows that the next previous stage will stall when it is ready to hand off work to the final stage, and so on up the pipeline.

CAUTION

It is important that firmware read the STATUS0/1 results and save them before clearing the interrupt request bit. Otherwise, a transaction and its results could be completely lost.

9.5.4.4 Randomizer

BCH ECC has a Randomizer module that is interfaced through the GPMI APBHDMA chain. The Randomizer can generate random data based on BCH ECC encoded/decoded data. It can be employed to reduce the disturbances caused by a neighboring cell in the NAND chip, thus reducing bit errors.

To enable the Randomizer module, set GPMI_ECCCTRL[RANDOMIZER_ENABLE] to 1, then set GPMI_ECCCOUNT[RANDOMIZER_PAGE] to select randomizer page number needed to be randomized. All these registers can be programmed by the DMA chain. The randomized data should start from the zero column address and be the size of the whole NAND page. If the randomizer function is enabled,

GPMI_ECCCTRL[ENABLE_ECC] should also be enabled. To bypass BCH error correction function, set BCH_FLASHxLAYOUT0[ECC0] and BCH_FLASHxLAYOUT1[ECCN] to 0.

9.5.5 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST.

The reset process gates the clocks automatically. The exemplary code is shown below.

```
// A soft reset can take multiple clocks to complete, so do NOT gate the
// clock when setting soft reset. The reset process will gate the clock
// automatically. Poll until this has happened before subsequently
// preparing soft-reset and clock gate
  BCH_CTRL_CLR(BM_BCH_CTRL_SFTRST);
  BCH_CTRL_CLR(BM_BCH_CTRL_CLKGATE);
// asserting soft-reset
  BCH_CTRL_SET(BM_BCH_CTRL_SFTRST);
// waiting for confirmation of soft-reset
  while (!BCH_CTRL.B.CLKGATE)
  {
// busy wait
  }
// Done.
  BCH_CTRL_CLR(BM_BCH_CTRL_SFTRST);
  BCH_CTRL_CLR(BM_BCH_CTRL_CLKGATE);
```

9.5.6 BCH Memory Map/Register Definition

BCH Hardware Register Format Summary

BCH memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3300_4000	Hardware BCH ECC Accelerator Control Register (BCH_CTRL)	32	R/W	C000_0000h	9.5.6.1/ 2419
3300_4004	Hardware BCH ECC Accelerator Control Register (BCH_CTRL_SET)	32	R/W	C000_0000h	9.5.6.1/ 2419
3300_4008	Hardware BCH ECC Accelerator Control Register (BCH_CTRL_CLR)	32	R/W	C000_0000h	9.5.6.1/ 2419
3300_400C	Hardware BCH ECC Accelerator Control Register (BCH_CTRL_TOG)	32	R/W	C000_0000h	9.5.6.1/ 2419
3300_4010	Hardware ECC Accelerator Status Register 0 (BCH_STATUS0)	32	R	0000_0010h	9.5.6.2/ 2421

Table continues on the next page...

BCH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3300_4014	Hardware ECC Accelerator Status Register 0 (BCH_STATUS0_SET)	32	R	0000_0010h	9.5.6.2/ 2421
3300_4018	Hardware ECC Accelerator Status Register 0 (BCH_STATUS0_CLR)	32	R	0000_0010h	9.5.6.2/ 2421
3300_401C	Hardware ECC Accelerator Status Register 0 (BCH_STATUS0_TOG)	32	R	0000_0010h	9.5.6.2/ 2421
3300_4020	Hardware ECC Accelerator Mode Register (BCH_MODE)	32	R/W	0000_0000h	9.5.6.3/ 2423
3300_4024	Hardware ECC Accelerator Mode Register (BCH_MODE_SET)	32	R/W	0000_0000h	9.5.6.3/ 2423
3300_4028	Hardware ECC Accelerator Mode Register (BCH_MODE_CLR)	32	R/W	0000_0000h	9.5.6.3/ 2423
3300_402C	Hardware ECC Accelerator Mode Register (BCH_MODE_TOG)	32	R/W	0000_0000h	9.5.6.3/ 2423
3300_4030	Hardware BCH ECC Loopback Encode Buffer Register (BCH_ENCODEPTR)	32	R/W	0000_0000h	9.5.6.4/ 2423
3300_4034	Hardware BCH ECC Loopback Encode Buffer Register (BCH_ENCODEPTR_SET)	32	R/W	0000_0000h	9.5.6.4/ 2423
3300_4038	Hardware BCH ECC Loopback Encode Buffer Register (BCH_ENCODEPTR_CLR)	32	R/W	0000_0000h	9.5.6.4/ 2423
3300_403C	Hardware BCH ECC Loopback Encode Buffer Register (BCH_ENCODEPTR_TOG)	32	R/W	0000_0000h	9.5.6.4/ 2423
3300_4040	Hardware BCH ECC Loopback Data Buffer Register (BCH_DATAPTR)	32	R/W	0000_0000h	9.5.6.5/ 2424
3300_4044	Hardware BCH ECC Loopback Data Buffer Register (BCH_DATAPTR_SET)	32	R/W	0000_0000h	9.5.6.5/ 2424
3300_4048	Hardware BCH ECC Loopback Data Buffer Register (BCH_DATAPTR_CLR)	32	R/W	0000_0000h	9.5.6.5/ 2424
3300_404C	Hardware BCH ECC Loopback Data Buffer Register (BCH_DATAPTR_TOG)	32	R/W	0000_0000h	9.5.6.5/ 2424
3300_4050	Hardware BCH ECC Loopback Metadata Buffer Register (BCH_METAPTR)	32	R/W	0000_0000h	9.5.6.6/ 2424
3300_4054	Hardware BCH ECC Loopback Metadata Buffer Register (BCH_METAPTR_SET)	32	R/W	0000_0000h	9.5.6.6/ 2424
3300_4058	Hardware BCH ECC Loopback Metadata Buffer Register (BCH_METAPTR_CLR)	32	R/W	0000_0000h	9.5.6.6/ 2424
3300_405C	Hardware BCH ECC Loopback Metadata Buffer Register (BCH_METAPTR_TOG)	32	R/W	0000_0000h	9.5.6.6/ 2424
3300_4070	Hardware ECC Accelerator Layout Select Register (BCH_LAYOUTSELECT)	32	R/W	E4E4_E4E4h	9.5.6.7/ 2425
3300_4074	Hardware ECC Accelerator Layout Select Register (BCH_LAYOUTSELECT_SET)	32	R/W	E4E4_E4E4h	9.5.6.7/ 2425
3300_4078	Hardware ECC Accelerator Layout Select Register (BCH_LAYOUTSELECT_CLR)	32	R/W	E4E4_E4E4h	9.5.6.7/ 2425

Table continues on the next page...

BCH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3300_407C	Hardware ECC Accelerator Layout Select Register (BCH_LAYOUTSELECT_TOG)	32	R/W	E4E4_E4E4h	9.5.6.7/ 2425
3300_4080	Hardware BCH ECC Flash 0 Layout 0 Register (BCH_FLASH0LAYOUT0)	32	R/W	070A_4080h	9.5.6.8/ 2426
3300_4084	Hardware BCH ECC Flash 0 Layout 0 Register (BCH_FLASH0LAYOUT0_SET)	32	R/W	070A_4080h	9.5.6.8/ 2426
3300_4088	Hardware BCH ECC Flash 0 Layout 0 Register (BCH_FLASH0LAYOUT0_CLR)	32	R/W	070A_4080h	9.5.6.8/ 2426
3300_408C	Hardware BCH ECC Flash 0 Layout 0 Register (BCH_FLASH0LAYOUT0_TOG)	32	R/W	070A_4080h	9.5.6.8/ 2426
3300_4090	Hardware BCH ECC Flash 0 Layout 1 Register (BCH_FLASH0LAYOUT1)	32	R/W	10DA_4080h	9.5.6.9/ 2428
3300_4094	Hardware BCH ECC Flash 0 Layout 1 Register (BCH_FLASH0LAYOUT1_SET)	32	R/W	10DA_4080h	9.5.6.9/ 2428
3300_4098	Hardware BCH ECC Flash 0 Layout 1 Register (BCH_FLASH0LAYOUT1_CLR)	32	R/W	10DA_4080h	9.5.6.9/ 2428
3300_409C	Hardware BCH ECC Flash 0 Layout 1 Register (BCH_FLASH0LAYOUT1_TOG)	32	R/W	10DA_4080h	9.5.6.9/ 2428
3300_40A0	Hardware BCH ECC Flash 1 Layout 0 Register (BCH_FLASH1LAYOUT0)	32	R/W	070A_4080h	9.5.6.10/ 2429
3300_40A4	Hardware BCH ECC Flash 1 Layout 0 Register (BCH_FLASH1LAYOUT0_SET)	32	R/W	070A_4080h	9.5.6.10/ 2429
3300_40A8	Hardware BCH ECC Flash 1 Layout 0 Register (BCH_FLASH1LAYOUT0_CLR)	32	R/W	070A_4080h	9.5.6.10/ 2429
3300_40AC	Hardware BCH ECC Flash 1 Layout 0 Register (BCH_FLASH1LAYOUT0_TOG)	32	R/W	070A_4080h	9.5.6.10/ 2429
3300_40B0	Hardware BCH ECC Flash 1 Layout 1 Register (BCH_FLASH1LAYOUT1)	32	R/W	10DA_4080h	9.5.6.11/ 2430
3300_40B4	Hardware BCH ECC Flash 1 Layout 1 Register (BCH_FLASH1LAYOUT1_SET)	32	R/W	10DA_4080h	9.5.6.11/ 2430
3300_40B8	Hardware BCH ECC Flash 1 Layout 1 Register (BCH_FLASH1LAYOUT1_CLR)	32	R/W	10DA_4080h	9.5.6.11/ 2430
3300_40BC	Hardware BCH ECC Flash 1 Layout 1 Register (BCH_FLASH1LAYOUT1_TOG)	32	R/W	10DA_4080h	9.5.6.11/ 2430
3300_40C0	Hardware BCH ECC Flash 2 Layout 0 Register (BCH_FLASH2LAYOUT0)	32	R/W	070A_4080h	9.5.6.12/ 2431
3300_40C4	Hardware BCH ECC Flash 2 Layout 0 Register (BCH_FLASH2LAYOUT0_SET)	32	R/W	070A_4080h	9.5.6.12/ 2431
3300_40C8	Hardware BCH ECC Flash 2 Layout 0 Register (BCH_FLASH2LAYOUT0_CLR)	32	R/W	070A_4080h	9.5.6.12/ 2431
3300_40CC	Hardware BCH ECC Flash 2 Layout 0 Register (BCH_FLASH2LAYOUT0_TOG)	32	R/W	070A_4080h	9.5.6.12/ 2431
3300_40D0	Hardware BCH ECC Flash 2 Layout 1 Register (BCH_FLASH2LAYOUT1)	32	R/W	10DA_4080h	9.5.6.13/ 2433

Table continues on the next page...

BCH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3300_40D4	Hardware BCH ECC Flash 2 Layout 1 Register (BCH_FLASH2LAYOUT1_SET)	32	R/W	10DA_4080h	9.5.6.13/ 2433
3300_40D8	Hardware BCH ECC Flash 2 Layout 1 Register (BCH_FLASH2LAYOUT1_CLR)	32	R/W	10DA_4080h	9.5.6.13/ 2433
3300_40DC	Hardware BCH ECC Flash 2 Layout 1 Register (BCH_FLASH2LAYOUT1_TOG)	32	R/W	10DA_4080h	9.5.6.13/ 2433
3300_40E0	Hardware BCH ECC Flash 3 Layout 0 Register (BCH_FLASH3LAYOUT0)	32	R/W	070A_4080h	9.5.6.14/ 2434
3300_40E4	Hardware BCH ECC Flash 3 Layout 0 Register (BCH_FLASH3LAYOUT0_SET)	32	R/W	070A_4080h	9.5.6.14/ 2434
3300_40E8	Hardware BCH ECC Flash 3 Layout 0 Register (BCH_FLASH3LAYOUT0_CLR)	32	R/W	070A_4080h	9.5.6.14/ 2434
3300_40EC	Hardware BCH ECC Flash 3 Layout 0 Register (BCH_FLASH3LAYOUT0_TOG)	32	R/W	070A_4080h	9.5.6.14/ 2434
3300_40F0	Hardware BCH ECC Flash 3 Layout 1 Register (BCH_FLASH3LAYOUT1)	32	R/W	10DA_4080h	9.5.6.15/ 2435
3300_40F4	Hardware BCH ECC Flash 3 Layout 1 Register (BCH_FLASH3LAYOUT1_SET)	32	R/W	10DA_4080h	9.5.6.15/ 2435
3300_40F8	Hardware BCH ECC Flash 3 Layout 1 Register (BCH_FLASH3LAYOUT1_CLR)	32	R/W	10DA_4080h	9.5.6.15/ 2435
3300_40FC	Hardware BCH ECC Flash 3 Layout 1 Register (BCH_FLASH3LAYOUT1_TOG)	32	R/W	10DA_4080h	9.5.6.15/ 2435
3300_4100	Hardware BCH ECC Debug Register0 (BCH_DEBUG0)	32	R/W	0000_0000h	9.5.6.16/ 2436
3300_4104	Hardware BCH ECC Debug Register0 (BCH_DEBUG0_SET)	32	R/W	0000_0000h	9.5.6.16/ 2436
3300_4108	Hardware BCH ECC Debug Register0 (BCH_DEBUG0_CLR)	32	R/W	0000_0000h	9.5.6.16/ 2436
3300_410C	Hardware BCH ECC Debug Register0 (BCH_DEBUG0_TOG)	32	R/W	0000_0000h	9.5.6.16/ 2436
3300_4110	KES Debug Read Register (BCH_DBGKESREAD)	32	R	0000_0000h	9.5.6.17/ 2438
3300_4114	KES Debug Read Register (BCH_DBGKESREAD_SET)	32	R	0000_0000h	9.5.6.17/ 2438
3300_4118	KES Debug Read Register (BCH_DBGKESREAD_CLR)	32	R	0000_0000h	9.5.6.17/ 2438
3300_411C	KES Debug Read Register (BCH_DBGKESREAD_TOG)	32	R	0000_0000h	9.5.6.17/ 2438
3300_4120	Chien Search Debug Read Register (BCH_DBGCSFEREAD)	32	R	0000_0000h	9.5.6.18/ 2438
3300_4124	Chien Search Debug Read Register (BCH_DBGCSFEREAD_SET)	32	R	0000_0000h	9.5.6.18/ 2438
3300_4128	Chien Search Debug Read Register (BCH_DBGCSFEREAD_CLR)	32	R	0000_0000h	9.5.6.18/ 2438

Table continues on the next page...

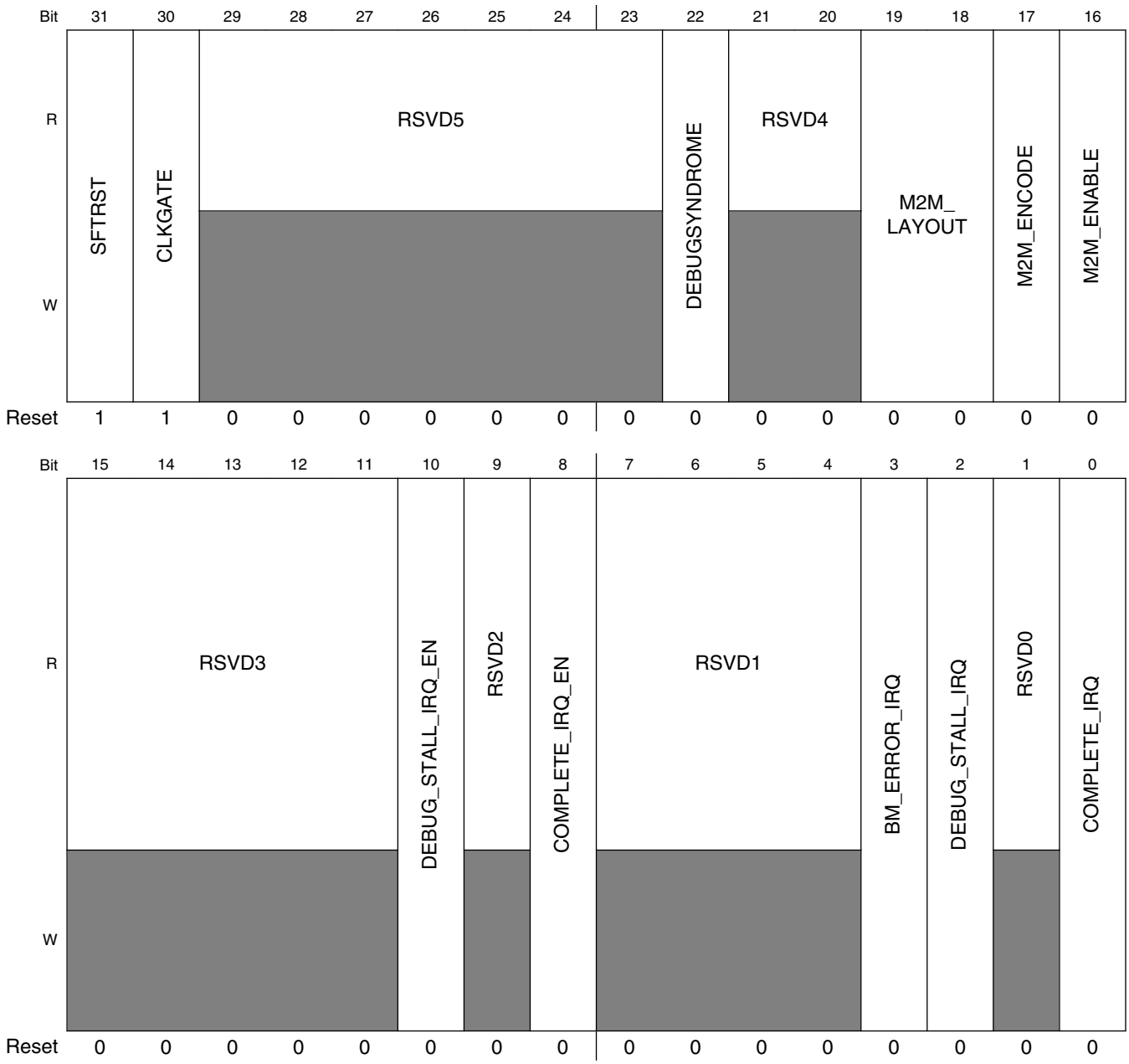
BCH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3300_412C	Chien Search Debug Read Register (BCH_DBGCSFEREAD_TOG)	32	R	0000_0000h	9.5.6.18/ 2438
3300_4130	Syndrome Generator Debug Read Register (BCH_DBGSYNDGENREAD)	32	R	0000_0000h	9.5.6.19/ 2439
3300_4134	Syndrome Generator Debug Read Register (BCH_DBGSYNDGENREAD_SET)	32	R	0000_0000h	9.5.6.19/ 2439
3300_4138	Syndrome Generator Debug Read Register (BCH_DBGSYNDGENREAD_CLR)	32	R	0000_0000h	9.5.6.19/ 2439
3300_413C	Syndrome Generator Debug Read Register (BCH_DBGSYNDGENREAD_TOG)	32	R	0000_0000h	9.5.6.19/ 2439
3300_4140	Bus Master and ECC Controller Debug Read Register (BCH_DBGAHBMREAD)	32	R	0000_0000h	9.5.6.20/ 2439
3300_4144	Bus Master and ECC Controller Debug Read Register (BCH_DBGAHBMREAD_SET)	32	R	0000_0000h	9.5.6.20/ 2439
3300_4148	Bus Master and ECC Controller Debug Read Register (BCH_DBGAHBMREAD_CLR)	32	R	0000_0000h	9.5.6.20/ 2439
3300_414C	Bus Master and ECC Controller Debug Read Register (BCH_DBGAHBMREAD_TOG)	32	R	0000_0000h	9.5.6.20/ 2439
3300_4150	Block Name Register (BCH_BLOCKNAME)	32	R	2048_4342h	9.5.6.21/ 2440
3300_4154	Block Name Register (BCH_BLOCKNAME_SET)	32	R	2048_4342h	9.5.6.21/ 2440
3300_4158	Block Name Register (BCH_BLOCKNAME_CLR)	32	R	2048_4342h	9.5.6.21/ 2440
3300_415C	Block Name Register (BCH_BLOCKNAME_TOG)	32	R	2048_4342h	9.5.6.21/ 2440
3300_4160	BCH Version Register (BCH_VERSION)	32	R	0100_0000h	9.5.6.22/ 2440
3300_4164	BCH Version Register (BCH_VERSION_SET)	32	R	0100_0000h	9.5.6.22/ 2440
3300_4168	BCH Version Register (BCH_VERSION_CLR)	32	R	0100_0000h	9.5.6.22/ 2440
3300_416C	BCH Version Register (BCH_VERSION_TOG)	32	R	0100_0000h	9.5.6.22/ 2440
3300_4170	Hardware BCH ECC Debug Register 1 (BCH_DEBUG1)	32	R/W	0000_0000h	9.5.6.23/ 2441
3300_4174	Hardware BCH ECC Debug Register 1 (BCH_DEBUG1_SET)	32	R/W	0000_0000h	9.5.6.23/ 2441
3300_4178	Hardware BCH ECC Debug Register 1 (BCH_DEBUG1_CLR)	32	R/W	0000_0000h	9.5.6.23/ 2441
3300_417C	Hardware BCH ECC Debug Register 1 (BCH_DEBUG1_TOG)	32	R/W	0000_0000h	9.5.6.23/ 2441

9.5.6.1 Hardware BCH ECC Accelerator Control Register (BCH_CTRL*n*)

The BCH CTRL provides overall control of the hardware ECC accelerator

Address: 3300_4000h base + 0h offset + (4d × i), where i=0d to 3d



BCH_CTRL*n* field descriptions

Field	Description
31 SFTRST	Set this bit to 0 to enable normal BCH operation. Set this bit to 1 (default) to disable clocking with the BCH and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the BCH block to its default state. This bit resets all state machines except for the AHB master state machine 0x0 **RUN** — Allow BCH to operate normally. 0x1 **RESET** — Hold BCH in reset.
30 CLKGATE	This bit must be set to 0 for normal operation. When set to 1 it gates off the clocks to the block. 0x0 **RUN** — Allow BCH to operate normally. 0x1 **NO_CLKS** — Do not clock BCH gates in order to minimize power consumption.
29–23 RSVD5	This field is reserved. This read-only field is reserved and always has the value 0.
22 DEBUGSYNDROME	(For debug purposes only). Enable write of computed syndromes to memory on BCH decode operations. Computed syndromes will be written to the auxiliary buffer after the status block. Syndromes will be written as padded 16-bit values.
21–20 RSVD4	This field is reserved. This read-only field is reserved and always has the value 0
19–18 M2M_LAYOUT	Selects the flash page format for memory-to-memory operations.
17 M2M_ENCODE	Selects encode (parity generation) or decode (correction) mode for memory-to-memory operations.
16 M2M_ENABLE	NOTE! WRITING THIS BIT INITIATES A MEMORY-TO-MEMORY OPERATION. The BCH module must be inactive (not processing data from the GPMI) when this bit is set. The M2M_ENCODE and M2M_LAYOUT bits as well as the ENCODEPTR, DATAPTR, and METAPTR registers are used for memory-to-memory operations and must be correctly programmed before writing this bit.
15–11 RSVD3	This field is reserved. This read-only field is reserved and always has the value 0
10 DEBUG_STALL_ IRQ_EN	1 = interrupt on debug stall mode is enabled. The IRQ is raised on every block
9 RSVD2	This field is reserved. This read-only field is reserved and always has the value 0.
8 COMPLETE_IRQ_ EN	1 = interrupt on completion of correction is enabled.
7–4 RSVD1	This field is reserved. This read-only field is reserved and always has the value 0.
3 BM_ERROR_IRQ	AHB Bus interface Error Interrupt Status. Write a 1 to the SCT clear address to clear the interrupt status bit.
2 DEBUG_STALL_IRQ	DEBUG STALL Interrupt Status. Write a 1 to the SCT clear address to clear the interrupt status bit.
1 RSVD0	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

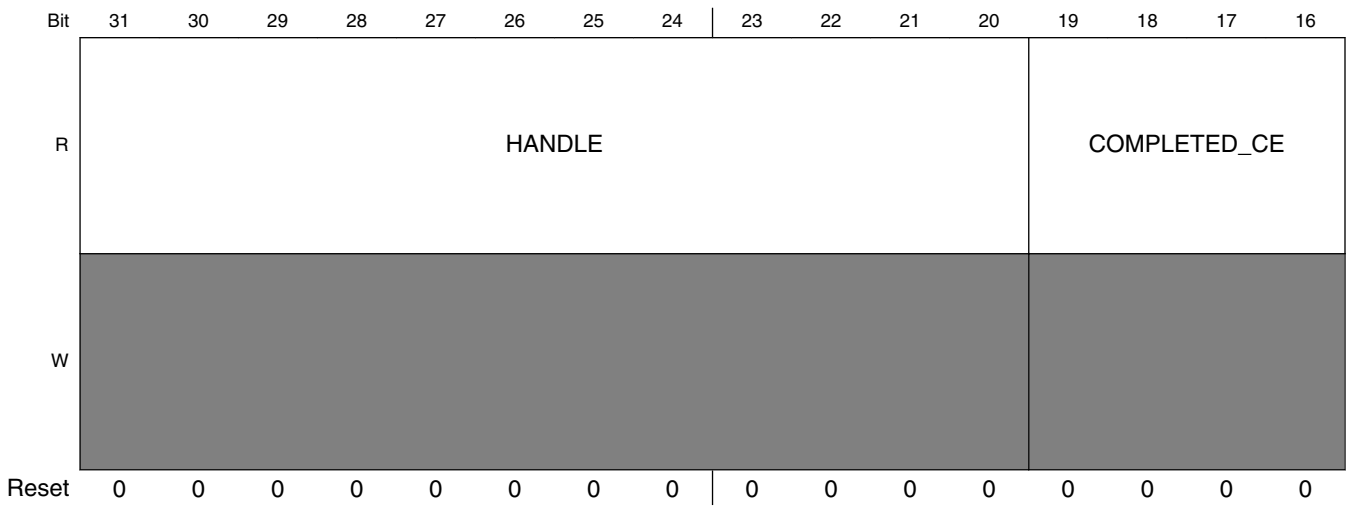
BCH_CTRL*n* field descriptions (continued)

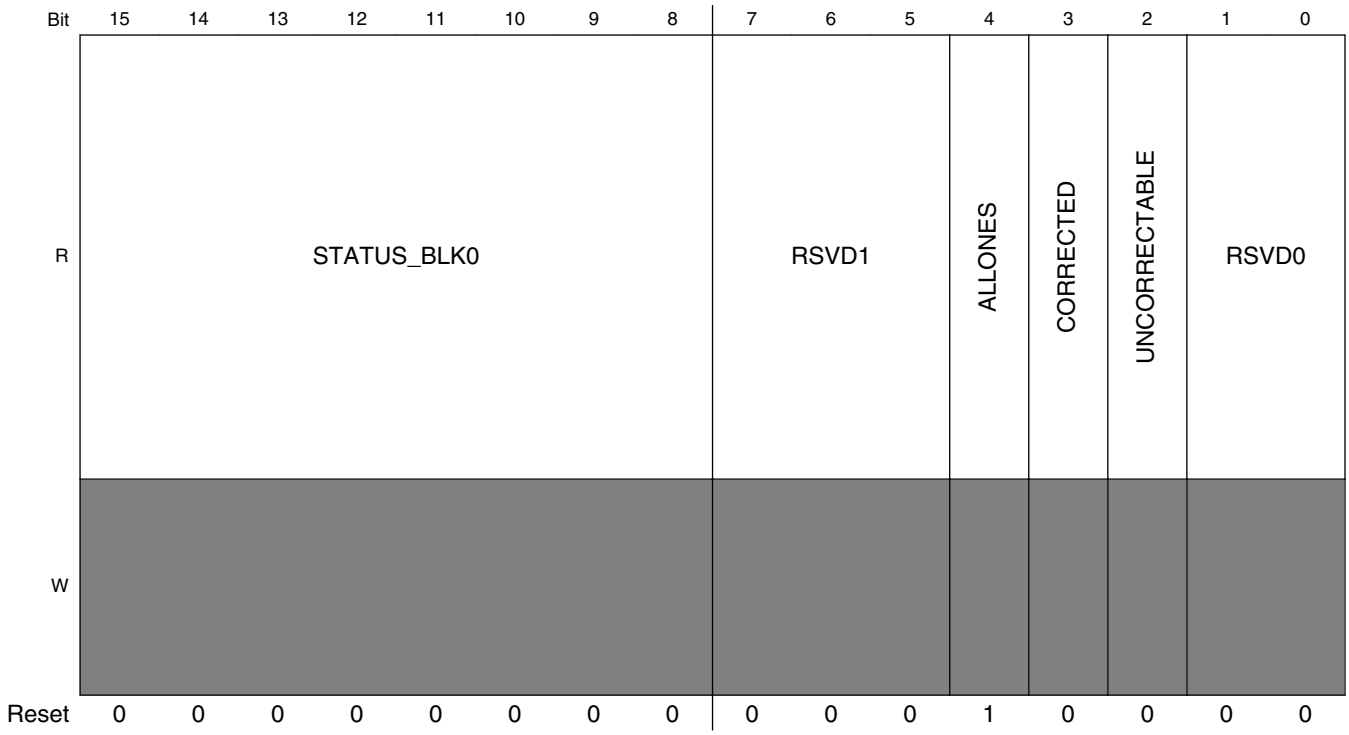
Field	Description
0 COMPLETE_IRQ	This bit indicates the state of the external interrupt line. Write a 1 to the SCT clear address to clear the interrupt status bit. NOTE: subsequent ECC completions will be held off as long as this bit is set. Be sure to read the data from BCH_STATUS0, 1 before clearing this interrupt bit.

9.5.6.2 Hardware ECC Accelerator Status Register 0 (BCH_STATUS0*n*)

The BCH STAT register provides visibility into the run-time status of the BCH and status information when processing is complete. It provides overall status of the hardware ECC accelerator.

Address: 3300_4000h base + 10h offset + (4d × i), where i=0d to 3d





BCH_STATUS0*n* field descriptions

Field	Description
31–20 HANDLE	Software supplies a 12 bit handle for this transfer as part of the GPMI DMA PIO operation that started the transaction. That handle passes down the pipeline and ends up here at the time the BCH interrupt is signaled.
19–16 COMPLETED_CE	This is the chip enable number corresponding to the NAND device from which this data came.
15–8 STATUS_BLK0	Count of symbols in error during processing of first block of flash (metadata block). The number of errors reported will be in the range of 0 to the ECC correction level for block 0. 0x00 **ZERO** — No errors found on block. 0x01 **ERROR1** — One error found on block. 0x02 **ERROR2** — One errors found on block. 0x03 **ERROR3** — One errors found on block. 0x04 **ERROR4** — One errors found on block. 0xFE **UNCORRECTABLE** — Block exhibited uncorrectable errors. 0xFF **ERASED** — Page is erased.
7–5 RSVD1	This field is reserved. This read-only field is reserved and always has the value 0.
4 ALLONES	1 = All data bits of this transaction are ONE.
3 CORRECTED	1 = At least one correctable error encountered during last processing cycle.
2 UNCORRECTABLE	1 = Uncorrectable error encountered during last processing cycle.
RSVD0	This field is reserved.

Table continues on the next page...

BCH_STATUS0*n* field descriptions (continued)

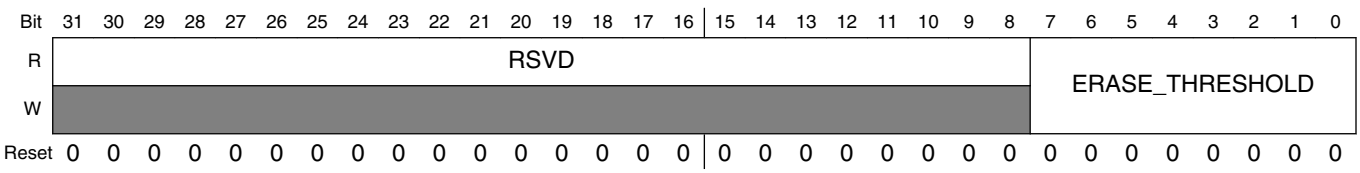
Field	Description
	This read-only field is reserved and always has the value 0.

9.5.6.3 Hardware ECC Accelerator Mode Register (BCH_MODE*n*)

The BCH MODE register provides additional mode controls.

Contains additional global mode controls for the BCH engine.

Address: 3300_4000h base + 20h offset + (4d × i), where i=0d to 3d



BCH_MODE*n* field descriptions

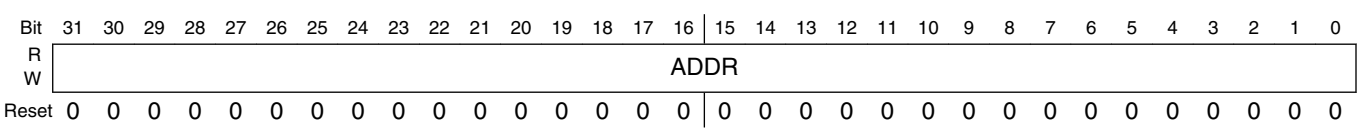
Field	Description
31–8 RSVD	This field is reserved. This read-only field is reserved and always has the value 0.
ERASE_ THRESHOLD	This value indicates the maximum number of zero bits on a flash subpage for it to be considered erased. For SLC NAND devices, this value should be programmed to 0 (meaning that the entire page should consist of bytes of 0xFF. For MLC NAND devices, bit errors may occur on reads (even on blank pages), so this threshold can be used to tune the erased page checking algorithm.

9.5.6.4 Hardware BCH ECC Loopback Encode Buffer Register (BCH_ENCODEPTR*n*)

When performing memory to memory operations, indicates the address of the encode buffer. This register should be programmed before writing a 1 to the M2M_ENABLE bit in the CTRL register.

For memory to memory operations, this register is used as the pointer to the encoded data, which is an output when encoding and an input while decoding.

Address: 3300_4000h base + 30h offset + (4d × i), where i=0d to 3d



BCH_ENCODEPTR*n* field descriptions

Field	Description
ADDR	Address pointer to encode buffer. This is the source for decode operations and the destination for encode operations. This value must be aligned on a 4 bytes boundary.

9.5.6.5 Hardware BCH ECC Loopback Data Buffer Register (BCH_DATAPTR*n*)

When performing memory to memory operations, indicates the address of the data buffer.

For memory to memory operations, this register is used as the pointer to the data to encode or the destination buffer for decode operations.

Address: 3300_4000h base + 40h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BCH_DATAPTR*n* field descriptions

Field	Description
ADDR	Address pointer to data buffer. This is the source for encode operations and the destination for decode operations. This register should be programmed before writing a 1 to the M2M_ENABLE bit in the CTRL register. This value must be aligned on a 4 byte boundary.

9.5.6.6 Hardware BCH ECC Loopback Metadata Buffer Register (BCH_METAPTR*n*)

When performing memory to memory operations, indicates the address of the metadata buffer.

For memory to memory operations, this register is used as the pointer to the metadata to encode or the extracted metadata for decode operations.

Address: 3300_4000h base + 50h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BCH_METAPTR*n* field descriptions

Field	Description
ADDR	Address pointer to metadata buffer. This is the source for encode metadata read operations and the destination for metadata decode operations. This register should be programmed before writing a 1 to the M2M_ENABLE bit in the CTRL register. This value must be aligned on a 4 bytes boundary.

9.5.6.7 Hardware ECC Accelerator Layout Select Register (BCH_LAYOUTSELECT*n*)

The BCH LAYOUTSELECT register provides a mapping of chip selects to layout registers.

When the BCH engine receives a request to process a data block from the GPMI interface, it will use this register to map the incoming chip select to one of the four possible flash layout registers

Address: 3300_4000h base + 70h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CS15_ SELECT		CS14_ SELECT		CS13_ SELECT		CS12_ SELECT		CS11_ SELECT		CS10_ SELECT		CS9_ SELECT		CS8_ SELECT	
Reset	1	1	1	0	0	1	0	0	1	1	1	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CS7_ SELECT		CS6_ SELECT		CS5_ SELECT		CS4_ SELECT		CS3_ SELECT		CS2_ SELECT		CS1_ SELECT		CS0_ SELECT	
Reset	1	1	1	0	0	1	0	0	1	1	1	0	0	1	0	0

BCH_LAYOUTSELECT*n* field descriptions

Field	Description
31–30 CS15_SELECT	Selects which layout is used for chip select 15.
29–28 CS14_SELECT	Selects which layout is used for chip select 14.
27–26 CS13_SELECT	Selects which layout is used for chip select 13.
25–24 CS12_SELECT	Selects which layout is used for chip select 12.
23–22 CS11_SELECT	Selects which layout is used for chip select 11.
21–20 CS10_SELECT	Selects which layout is used for chip select 10.
19–18 CS9_SELECT	Selects which layout is used for chip select 9.

Table continues on the next page...

BCH_LAYOUTSELECT*n* field descriptions (continued)

Field	Description
17–16 CS8_SELECT	Selects which layout is used for chip select 8.
15–14 CS7_SELECT	Selects which layout is used for chip select 7.
13–12 CS6_SELECT	Selects which layout is used for chip select 6.
11–10 CS5_SELECT	Selects which layout is used for chip select 5.
9–8 CS4_SELECT	Selects which layout is used for chip select 4.
7–6 CS3_SELECT	Selects which layout is used for chip select 3.
5–4 CS2_SELECT	Selects which layout is used for chip select 2.
3–2 CS1_SELECT	Selects which layout is used for chip select 1.
CS0_SELECT	Selects which layout is used for chip select 0.

9.5.6.8 Hardware BCH ECC Flash 0 Layout 0 Register (BCH_FLASH0LAYOUT0*n*)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH0LAYOUT1 register to control the format for the devices selecting layout 0 in the LAYOUTSELECT register.

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the four layout values. Each pair of registers contains settings that are used by the BCH block while reading / writing the flash page to control data, metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks.

See sections *Flash Page Layout* and *Determining the ECC layout for a device* for more detail information on setting up the flash layout registers.

Address: 3300_4000h base + 80h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NBLOCKS								META_SIZE							
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ECC0					GF13_0_GF14_1	DATA0_SIZE									
W																
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

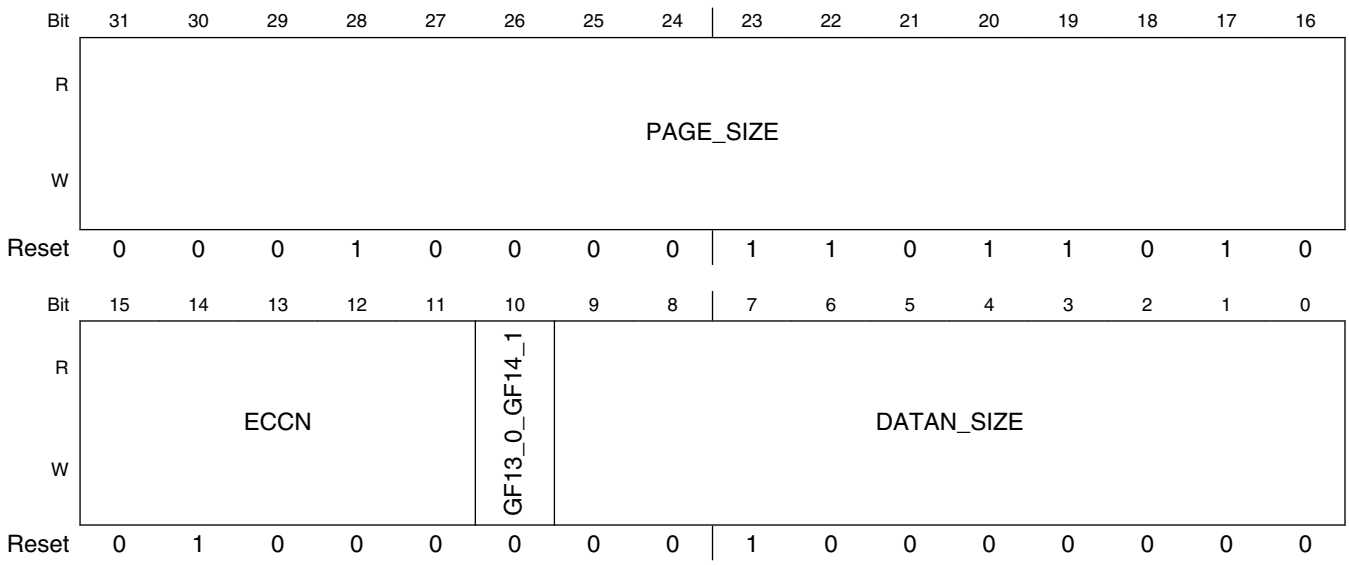
BCH_FLASH0LAYOUT0*n* field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata—if set to 0, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a 0, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and covered by a single parity block.
15–11 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed : — 0x1E **ECC60** — ECC 60 to be performed 0x1F **ECC62** — ECC 62 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATA0_SIZE	Indicates the size of the data 0 block (in DWORDS / four bytes) to be stored on the flash page. If set to 0, the first block only contains metadata.

9.5.6.9 Hardware BCH ECC Flash 0 Layout 1 Register (BCH_FLASH0LAYOUT1*n*)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH0LAYOUT0 register to control the format for the device selecting layout 0 in the LAYOUTSELECT register.

Address: 3300_4000h base + 90h offset + (4d × i), where i=0d to 3d



BCH_FLASH0LAYOUT1*n* field descriptions

Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–11 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed : — 0x1E **ECC60** — ECC 60 to be performed 0x1F **ECC62** — ECC 62 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATAN_SIZE	Indicates the size of the subsequent data blocks (in DWORDS / four bytes) to be stored on the flash page. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

9.5.6.10 Hardware BCH ECC Flash 1 Layout 0 Register (BCH_FLASH1LAYOUT0*n*)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH1LAYOUT1 register to control the format for the devices selecting layout 1 in the LAYOUTSELECT register.

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the four layout values. Each pair of registers contains settings that are used by the BCH block while reading / writing the flash page to control data, metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks.

See sections *Flash Page Layout* and *Determining the ECC layout for a device* for more detail information on setting up the flash layout registers.

Address: 3300_4000h base + A0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NBLOCKS								META_SIZE							
Reset	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ECC0					GF13_0_GF14_1	DATA0_SIZE									
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH1LAYOUT0*n* field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.

Table continues on the next page...

BCH_FLASH1LAYOUT0*n* field descriptions (continued)

Field	Description
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design supports from 0 to 255 bytes for metadata—if set to 0, no metadata will be stored. Metadata is stored before the associated data is in block 0. If the DATA0_SIZE field is programmed to a 0, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and covered by a single parity block.
15–11 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed : — 0x1E **ECC60** — ECC 60 to be performed 0x1F **ECC62** — ECC 62 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATA0_SIZE	Indicates the size of the data 0 block (in DWORDS / four bytes) to be stored on the flash page. If set to 0, the first block will contains metadata.

9.5.6.11 Hardware BCH ECC Flash 1 Layout 1 Register (BCH_FLASH1LAYOUT1*n*)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH1LAYOUT0 register to control the format for the device selecting layout 1 in the LAYOUTSELECT register.

Address: 3300_4000h base + B0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PAGE_SIZE															
Reset	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ECCN					GF13_0_GF14_1	DATAN_SIZE									
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH1LAYOUT1*n* field descriptions

Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–11 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed : — 0x1E **ECC60** — ECC 60 to be performed 0x1F **ECC62** — ECC 62 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATAN_SIZE	Indicates the size of the subsequent data blocks (in DWORDS / four bytes) to be stored on the flash page. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

9.5.6.12 Hardware BCH ECC Flash 2 Layout 0 Register (BCH_FLASH2LAYOUT0*n*)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH2LAYOUT1 register to control the format for the devices selecting layout 2 in the LAYOUTSELECT register.

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the four layout values. Each pair of registers contains settings that are used by the BCH block while reading / writing the flash page to control data, metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks.

See sections *Flash Page Layout* and *Determining the ECC layout for a device* for more detail information on setting up the flash layout registers.

Address: 3300_4000h base + C0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NBLOCKS								META_SIZE							
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ECC0					GF13_0_GF14_1	DATA0_SIZE									
W																
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH2LAYOUT0*n* field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that eight subsequent blocks are present for a total of nine blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata—if set to 0, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a 0, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and will be covered by a single parity block.
15–11 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed : — 0x1E **ECC60** — ECC 60 to be performed 0x1F **ECC62** — ECC 62 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATA0_SIZE	Indicates the size of the data 0 block (in DWORDS / four bytes) to be stored on the flash page. If set to 0, the first block will only contain metadata.

9.5.6.13 Hardware BCH ECC Flash 2 Layout 1 Register (BCH_FLASH2LAYOUT1*n*)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH2LAYOUT0 register to control the format for the device selecting layout 2 in the LAYOUTSELECT register.

Address: 3300_4000h base + D0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PAGE_SIZE															
W																
Reset	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ECCN					GF13_0_GF14_1	DATAN_SIZE									
W																
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH2LAYOUT1*n* field descriptions

Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–11 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed : — 0x1E **ECC60** — ECC 60 to be performed 0x1F **ECC62** — ECC 62 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATAN_SIZE	Indicates the size of the subsequent data blocks (in DWORDS / four bytes) to be stored on the flash page. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

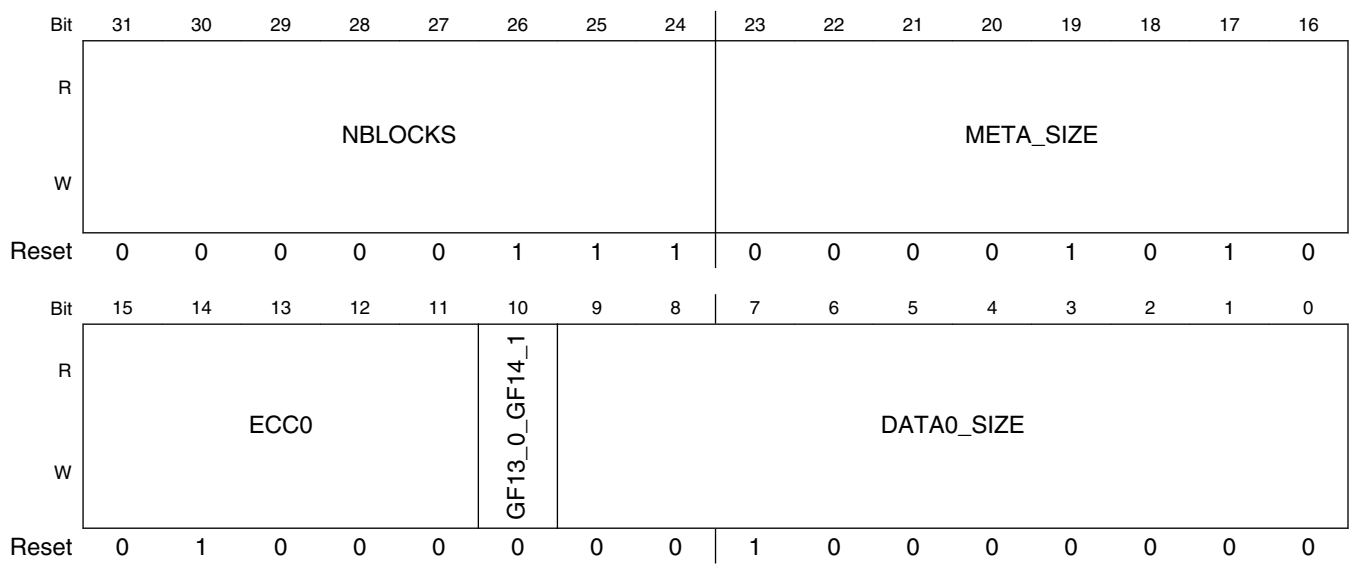
9.5.6.14 Hardware BCH ECC Flash 3 Layout 0 Register (BCH_FLASH3LAYOUT0*n*)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH3LAYOUT1 register to control the format for the devices selecting layout 3 in the LAYOUTSELECT register.

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the four layout values. Each pair of registers contains settings that are used by the BCH block while reading / writing the flash page to control data, metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks.

See sections *Flash Page Layout* and *Determining the ECC layout for a device* for more detail information on setting up the flash layout registers.

Address: 3300_4000h base + E0h offset + (4d × i), where i=0d to 3d



BCH_FLASH3LAYOUT0*n* field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.

Table continues on the next page...

BCH_FLASH3LAYOUT0*n* field descriptions (continued)

Field	Description
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata—if set to 0, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a 0, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and will be covered by a single parity block.
15–11 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed : — 0x1E **ECC60** — ECC 60 to be performed 0x1F **ECC62** — ECC 62 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATA0_SIZE	Indicates the size of the data 0 block (in DWORDS / four bytes) to be stored on the flash page. If set to 0, the first block will only contain metadata.

9.5.6.15 Hardware BCH ECC Flash 3 Layout 1 Register (BCH_FLASH3LAYOUT1*n*)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjunction with the FLASH3LAYOUT0 register to control the format for the device selecting layout 3 in the LAYOUTSELECT register.

Address: 3300_4000h base + F0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PAGE_SIZE															
Reset	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ECCN					GF13_0_GF14_1	DATAN_SIZE									
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH3LAYOUT1*n* field descriptions

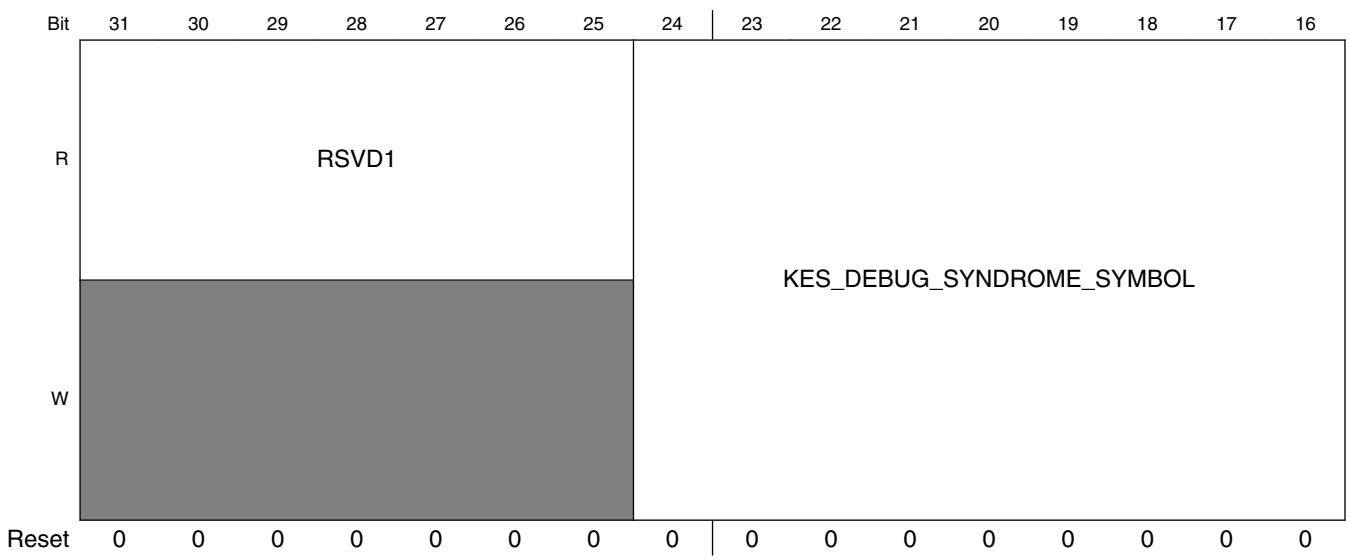
Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–11 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed : — 0x1E **ECC60** — ECC 60 to be performed 0x1F **ECC62** — ECC 62 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
DATAN_SIZE	Indicates the size of the subsequent data blocks (in DWORDS / four bytes) to be stored on the flash page. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

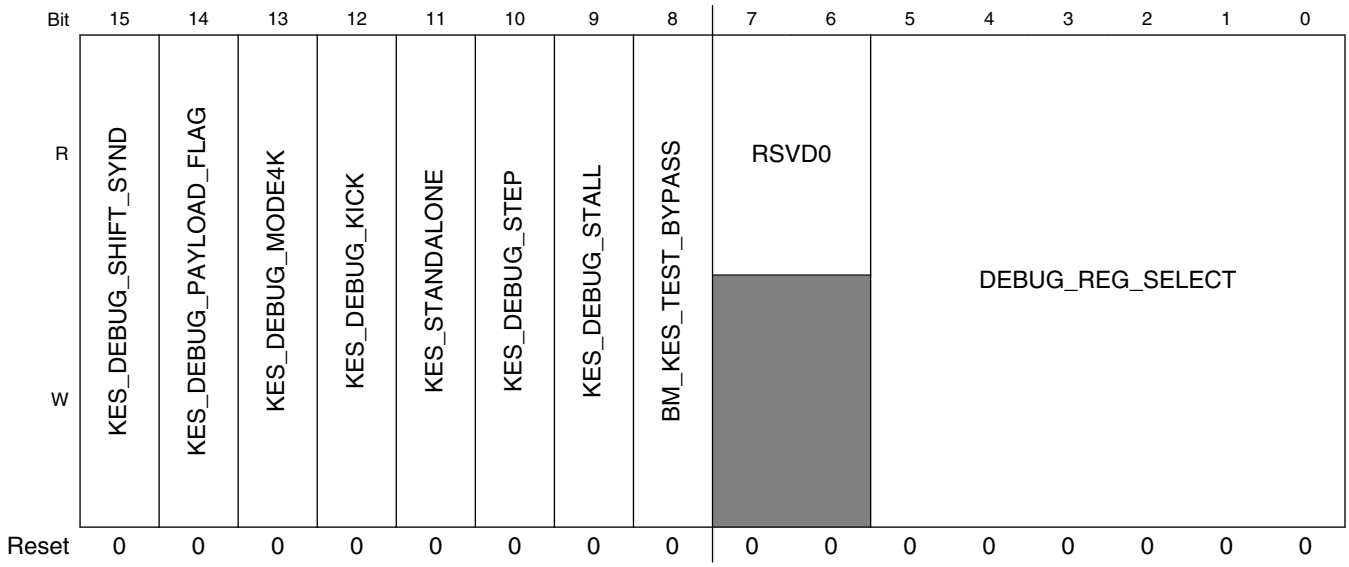
9.5.6.16 Hardware BCH ECC Debug Register0 (BCH_DEBUG0*n*)

The hardware BCH accelerator internal state machines and signals can be seen in the ECC debug register.

The BCH_DEBUG0 register provides access to various internal state information which might prove useful during hardware debug and validation.

Address: 3300_4000h base + 100h offset + (4d × i), where i=0d to 3d





BCH_DEBUG0*n* field descriptions

Field	Description
31–25 RSVD1	This field is reserved. This read-only field is reserved and always has the value 0.
24–16 KES_DEBUG_ SYNDROME_ SYMBOL	The 9 bit value in this bit field shifts into the syndrome register array at the input of the KES engine whenever BCH_DEBUG0_KES_DEBUG_SHIFT_SYND is toggled. 0x0 **NORMAL** — Bus master address generator for SYND_GEN writes operates normally. 0x1 **TEST_MODE** — Bus master address generator always addresses last four bytes in Auxiliary block.
15 KES_DEBUG_ SHIFT_SYND	Toggling this bit causes the value in BCH_DEBUG0_KES_SYNDROME_SYMBOL to be shift into the syndrome register array at the input to the KES engine. After shifting in 16 symbols, one can kick off both KES and CF cycles by toggling BCH_DEBUG0_KES_DEBUG_KICK. Make sure that set KES_BCH_DEBUG0_KES_STANDALONE mode to 1 before kicking.
14 KES_DEBUG_ PAYLOAD_FLAG	When running the stand alone debug mode on the error calculator, the state of this bit is presented to the KES engine as the input payload flag. 0x1 **DATA** — Payload is set for 512 bytes data block. 0x1 **AUX** — Payload is set for 65 or 19 bytes auxiliary block.
13 KES_DEBUG_ MODE4K	When running the stand alone debug mode on the error calculator, the state of this bit is presented to the KES engine as the input mode (4K or 2K pages). 0x1 **4k** — Mode is set for 4K NAND pages. 0x1 **2k** — Mode is set for 2K NAND pages.
12 KES_DEBUG_ KICK	Toggling causes KES engine FSM to start as if kick by the Bus Master. This allows stand alone testing of the KES and Chien Search engines. Be sure to set KES_BCH_DEBUG0_KES_STANDALONE mode to 1 before kicking.
11 KES_ STANDALONE	Set to one, cause the KES engine to suppress toggling the KES_BM_DONE signal to the bus master and suppress toggling the CF_BM_DONE signal by the CF engine. 0x0 **NORMAL** — Bus master address generator for SYND_GEN writes operates normally. 0x1 **TEST_MODE** — Bus master address generator always addresses last four bytes in Auxiliary block.

Table continues on the next page...

BCH_DEBUG0*n* field descriptions (continued)

Field	Description
10 KES_DEBUG_STEP	Toggling this bit causes the KES FSM to skip passed the stall state if it is in DEBUG_STALL mode and completed processing a block.
9 KES_DEBUG_STALL	Set to one to cause KES FSM to stall after notifying Chien search engine to start processing its block but before notifying the bus master that the KES computation is complete. This allows a diagnostic to stall the FSM after each blocks key equations are solved. This also has the effect of stalling the CSFE search engine so it's state can be examined after it finishes processing the KES stalled block. 0x0 **NORMAL** — KES FSM proceeds to next block supplied by bus master. 0x1 **WAIT** — KES FSM waits after current equations are solved and the search engine is started.
8 BM_KES_TEST_BYPASS	1 = Point all SYND_GEN writes to dummy area at the end of the AUXILLIARY block so that diagnostics can preload all payload, parity bytes and computed syndrome bytes for test the KES engine. 0x0 **NORMAL** — Bus master address generator for SYND_GEN writes operates normally. 0x1 **TEST_MODE** — Bus master address generator always addresses last four bytes in Auxiliary block.
7–6 RSVD0	This field is reserved. This read-only field is reserved and always has the value 0.
DEBUG_REG_SELECT	The value loaded in this bit field is used to select the internal register state view of KES engine or the Chien search engine.

9.5.6.17 KES Debug Read Register (BCH_DBGKESREAD*n*)

The hardware BCH ECC accelerator key equation solver internal state machines and signals can be seen in the ECC debug registers.

Address: 3300_4000h base + 110h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUES																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

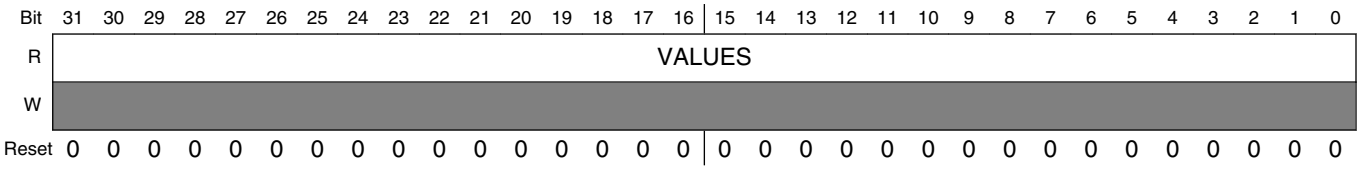
BCH_DBGKESREAD*n* field descriptions

Field	Description
VALUES	This register returns the ROM BIST CRC value after a BIST test.

9.5.6.18 Chien Search Debug Read Register (BCH_DBGCSFEREAD*n*)

The hardware BCH ECC accelerator Chien Search internal state machines and signals can be seen in the ECC debug registers.

Address: 3300_4000h base + 120h offset + (4d × i), where i=0d to 3d



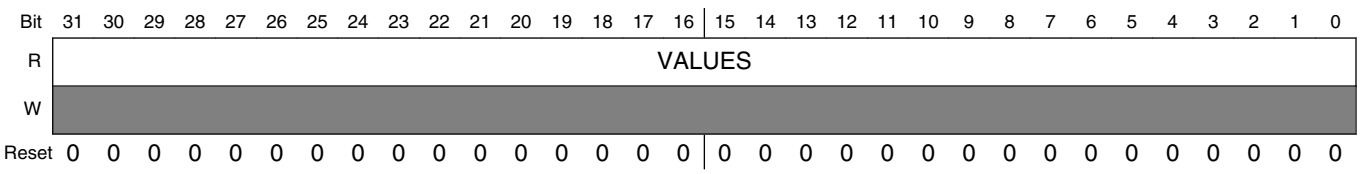
BCH_DBGCSFEREAD*n* field descriptions

Field	Description
VALUES	Reserved

9.5.6.19 Syndrome Generator Debug Read Register (BCH_DBGSYNDGENREAD*n*)

The hardware BCH ECC accelerator syndrome generator internal state machines and signals can be seen in the ECC debug registers.

Address: 3300_4000h base + 130h offset + (4d × i), where i=0d to 3d



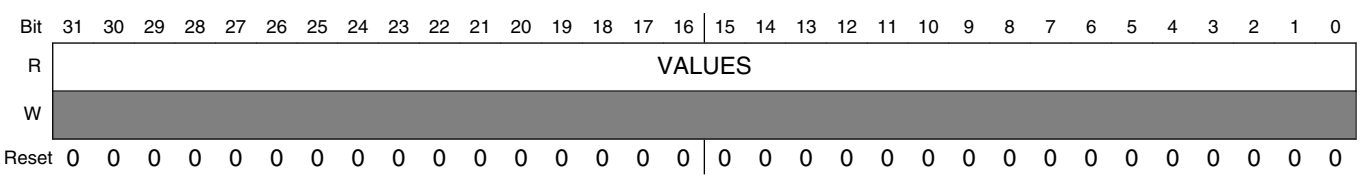
BCH_DBGSYNDGENREAD*n* field descriptions

Field	Description
VALUES	Reserved

9.5.6.20 Bus Master and ECC Controller Debug Read Register (BCH_DBGAHBMREAD*n*)

The hardware BCH ECC accelerator bus master, ECC controller internal state machines, and signals can be seen in the ECC debug registers.

Address: 3300_4000h base + 140h offset + (4d × i), where i=0d to 3d



BCH_DBGAHBMREAD*n* field descriptions

Field	Description
VALUES	Reserved

9.5.6.21 Block Name Register (BCH_BLOCKNAME*n*)

Read only view of the block name string BCH.

Fixed pattern read only value is for test purposes. It can be read as an ASCII string with the zero termination coming from the first byte of the BLOCKVERSION register.

Address: 3300_4000h base + 150h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NAME																															
W																																
Reset	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	1	0

BCH_BLOCKNAME*n* field descriptions

Field	Description
NAME	The name is in the ASCII characters BCH (0x20, H, C, B).

9.5.6.22 BCH Version Register (BCH_VERSION*n*)

This register always returns a known read value for debug purposes and indicates the version of the block and RTL version in use.

Address: 3300_4000h base + 160h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

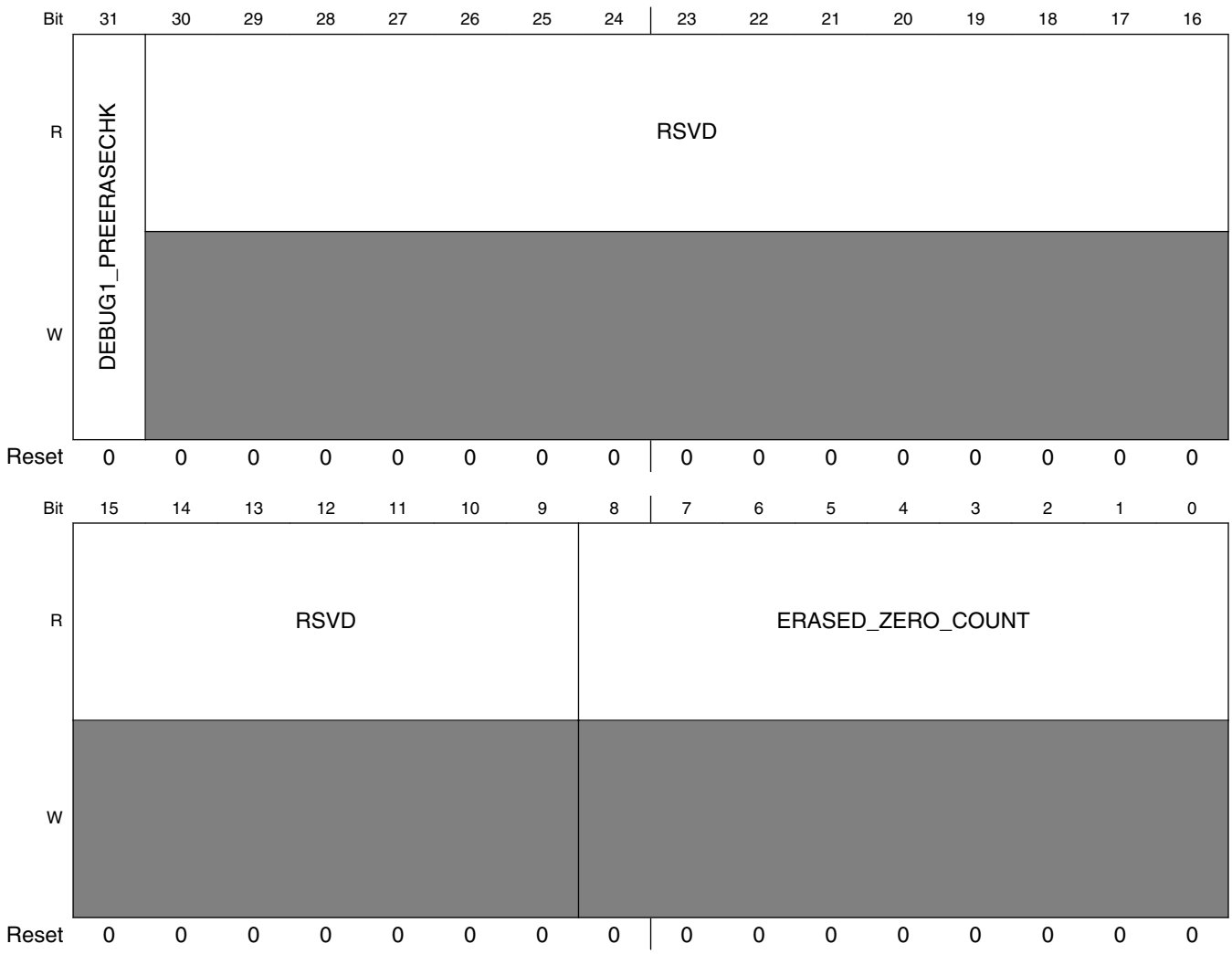
BCH_VERSION*n* field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value indicates the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value indicates the MINOR field of the RTL version.
STEP	Fixed read-only value reflecting the stepping of the RTL version.

9.5.6.23 Hardware BCH ECC Debug Register 1 (BCH_DEBUG1*n*)

The BCH_DEBUG1 register provides erased zero count information and pre-erase check.

Address: 3300_4000h base + 170h offset + (4d × i), where i=0d to 3d



BCH_DEBUG1*n* field descriptions

Field	Description
31 DEBUG1_ PREERASECHK	Blank page enables pre-erase check. 0x0 Turn off pre-erase check 0x1 Turn on pre-erase check
30–9 RSVD	This field is reserved. This read-only field is reserved and always has the value 0.
ERASED_ ZERO_COUNT	The zero counts on one page.

BCH_DEBUG1*n* field descriptions (continued)

Field	Description

9.6 General Purpose Media Interface (GPMI)

9.6.1 Overview

The GPMI controller is a flexible interface to supporting up to four NAND flash chip selects.

- ONFI3.2, DDR Mode, Samsung / Toshiba Toggle NAND protocol compatible.
- Fully configurable address and command behavior, providing support for future devices not yet specified.

The GPMI resides on the APBH. The GPMI also provides an interface to the BCH module to allow direct parity processing.

Registers are clocked on the HCLK domain. The I/O and pin timing are clocked on a dedicated GPMICLK domain. GPMICLK can be set to maximize I/O performance.

The following figure shows a block diagram of the GPMI controller.

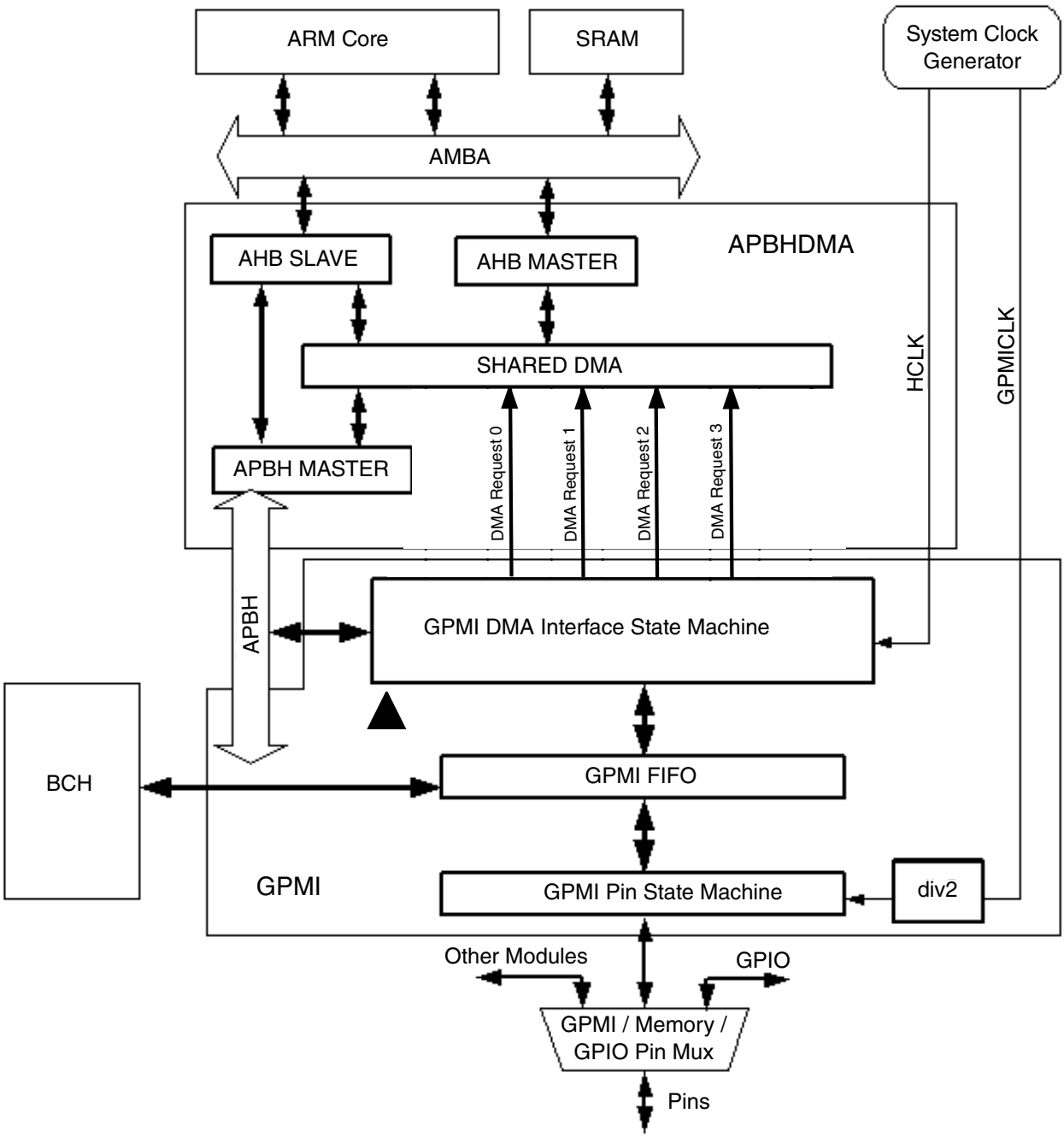


Figure 9-38. General-Purpose Media Interface Controller Block Diagram

9.6.2 External Signals

The table found here describes the external signals of GPMI.

Table 9-22. GPMI External Signals

Signal	Description	Pad	Mode	Direction
NAND_ALE	Address latch enable signal	SD3_CMD	ALT1	O
NAND_CE0_B	Chip enable signal	SAI1_TXC	ALT1	O
NAND_CE1_B	Chip enable signal	SAI1_RXD	ALT1	O
NAND_CE2_B	Chip enable signal	SAI1_RXFS	ALT1	O
NAND_CE3_B	Chip enable signal	SAI1_RXC	ALT1	O
NAND_CLE	Command latch enable signal	SD3_CLK	ALT1	O
NAND_DATA00	Data signal	SD3_DATA0	ALT1	IO
NAND_DATA01	Data signal	SD3_DATA1	ALT1	IO
NAND_DATA02	Data signal	SD3_DATA2	ALT1	IO
NAND_DATA03	Data signal	SD3_DATA3	ALT1	IO
NAND_DATA04	Data signal	SD3_DATA4	ALT1	IO
NAND_DATA05	Data signal	SD3_DATA5	ALT1	IO
NAND_DATA06	Data signal	SD3_DATA6	ALT1	IO
NAND_DATA07	Data signal	SD3_DATA7	ALT1	IO
NAND_DQS	DQS signal	SAI1_TXFS	ALT1	IO
NAND_RE_B	Read enable signal	SD3_STROBE	ALT1	O
NAND_READY	Ready signal	SAI1_TXD	ALT1	IO
NAND_WE_B	Write enable signal	SD3_RESET_B	ALT1	O
NAND_WP_B	Wait polarity signal	SAI1_MCLK	ALT1	O

9.6.3 Clocks

The table found here describes the clock sources for GPMI.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 9-23. GPMI Clocks

Clock name	Clock Root	Description
bch_input_apb_clk	nand_usdhc_bus_clk_root	BCH to APBH input clock
gpmi_bch_input_bch_clk	nand_usdhc_bus_clk_root	BCH input clock

Table continues on the next page...

Table 9-23. GPMI Clocks (continued)

Clock name	Clock Root	Description
gpmi_bch_input_gpmi_io_clk	nand_clk_root	GPMI IO input clock
gpmi_input_apb_clk	nand_usdhc_bus_clk_root	GPMI to APBH clock

9.6.4 GPMI NAND Mode

The general-purpose media interface has several features to efficiently support NAND:

- Individual chip select pins and ganged ready/busy pin for up to four NANDs.
- Individual state machine and DMA channel for each chip select.
- Special command modes work with DMA controller to perform all normal NAND functions without CPU intervention.
- Configurable timing based on a dedicated clock allows optimal balance of high NAND performance and low system power.

GPMI and DMA have been designed to handle complex multi-page operations without CPU intervention. The DMA uses a linked descriptor function with branching capability to automatically handle all of the operations needed to read/write multiple pages:

- **Data/Register Read/Write**-The GPMI can be programmed to read or write multiple cycles to the NAND address, command or data registers.
- **Wait for NAND Ready**-The GPMI's Wait-for-Ready mode can monitor the ready/busy signal of a single NAND flash and signal the DMA when the device has become ready. It also has a time-out counter and can indicate to the DMA that a time-out error has occurred. The DMAs can conditionally branch to a different descriptor in the case of an error.
- **Check Status**-The Read-and-Compare mode allows the GPMI to check NAND status against a reference. If an error is found, the GPMI can instruct the DMA to branch to an alternate descriptor, which attempts to fix the problem or asserts a CPU IRQ.

9.6.4.1 Multiple NAND Support

The GPMI supports up to four NAND chip selects, with ganged ready/busy pins.

Since they share a data bus and control lines, the GPMI can only actively communicate with a single NAND at a time. However, all NANDs can concurrently perform internal read, write, or erase operations. With fast NAND flash and software support for

concurrent NAND operations, this architecture allows the total throughput to approach the data bus speed, which can be as high as 50 MB/s (8-bit bus running at 50 MHz single clock edge) in asynchronous mode and 200MB/s (8-bit bus running at 100MHz both clock edges) in Source Synchronous mode.

There are two options for controlling the four NAND chip selects via the DMA interface. The first option is the one to one mapping, where the each DMA channel is tied to its own NAND chip select. For example DMA channel 'n' accesses only NAND attached to chip select 'n'. The second option is the decoupled mode where a DMA channel can access any or all NAND chip selects connected to the GPMI. A DMA channel will signify the NAND chip select it wants to access by writing its chip select value in the GPMI_CTRL0[CS] field and setting the GPMI_CTRL1[DECOUPLE_CS] to 1. This option is useful if software chooses to use only one DMA channel to access all the attached NAND devices.

9.6.4.2 GPMI NAND Timing and Clocking

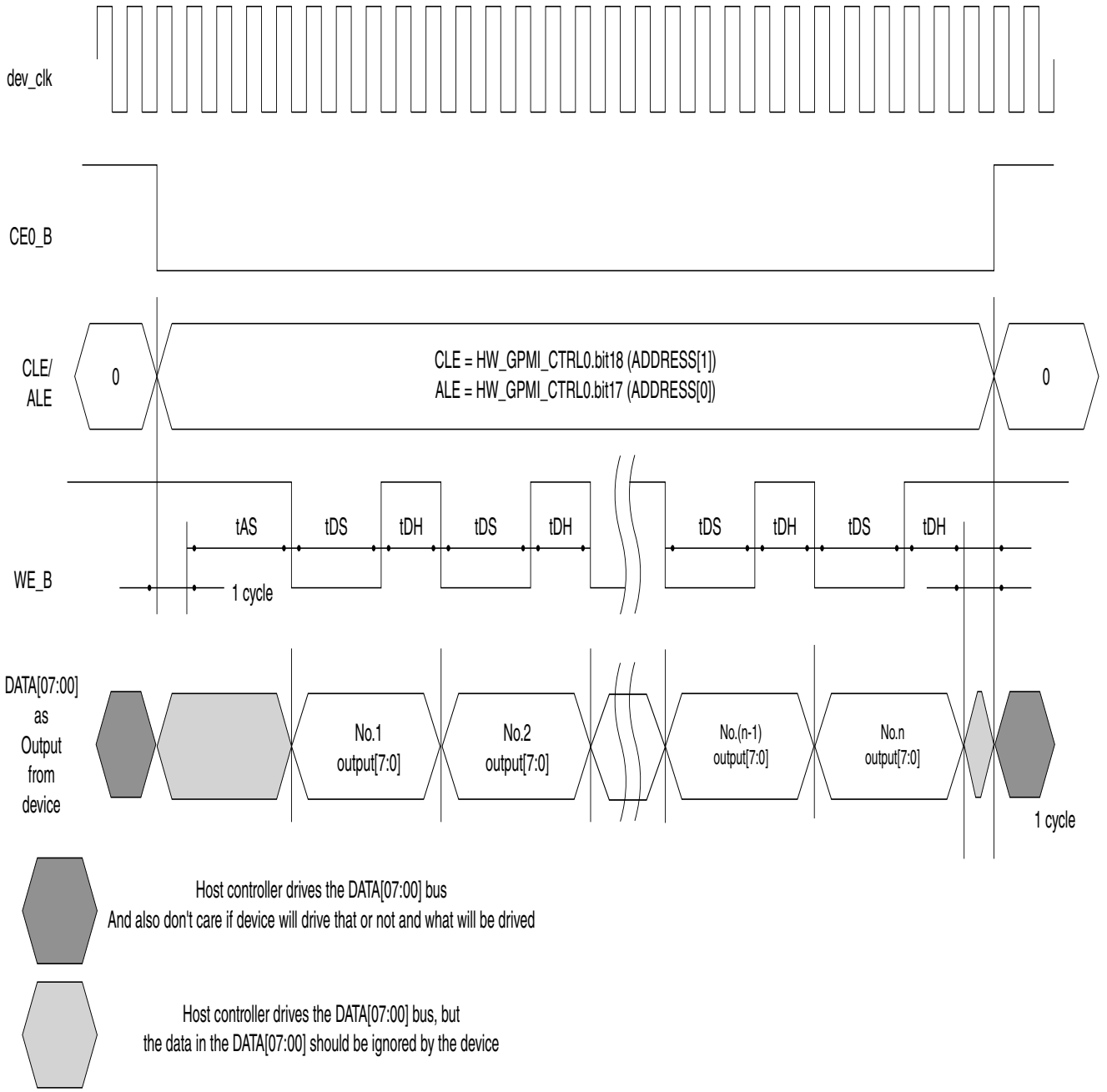
The dedicated clock, GPMICLK, is used as a timing reference for NAND flash I/O. Since various NANDs have different timing requirements, GPMICLK may need to be adjusted for each application.

While the actual pin timings are limited by the NAND chips used and the I/O pad configuration, the GPMI can support data bus speeds of up to 200 MHz x 8 bits. The actual read/write strobe timing parameters are adjusted as indicated in the register descriptions in Memory Map.

9.6.4.3 Basic NAND Timing

9.6.4.3.1 NAND Asynchronous Timing

Figure 9-40 and illustrates the operation of the output (from host to device) timing parameters in NAND ONFI asynchronous mode.



- tAS is configurable by programming HW_GPMI_TIMING0 Address_Setup: in this example, Address_Setup = 4, tAS is equal to 4 dev_clk cycles.
- tDS is configurable by programming HW_GPMI_TIMING0 Data_Setup; in this example, Data_Setup = 3, tDS is equal to 3 dev_clk cycles
- tDH is configuarble by programming HW_GPMI_TIMING0 Data_Hold: in this example, Data_Hold = 2, tDH is equal to 2 dev_clk cycles
- tAS/tDS/tDH will extend, if the output data is not ready in device fifo.

Figure 9-39. Asynchronous Mode Basic Write Timing Diagram (command write, address write, or data write

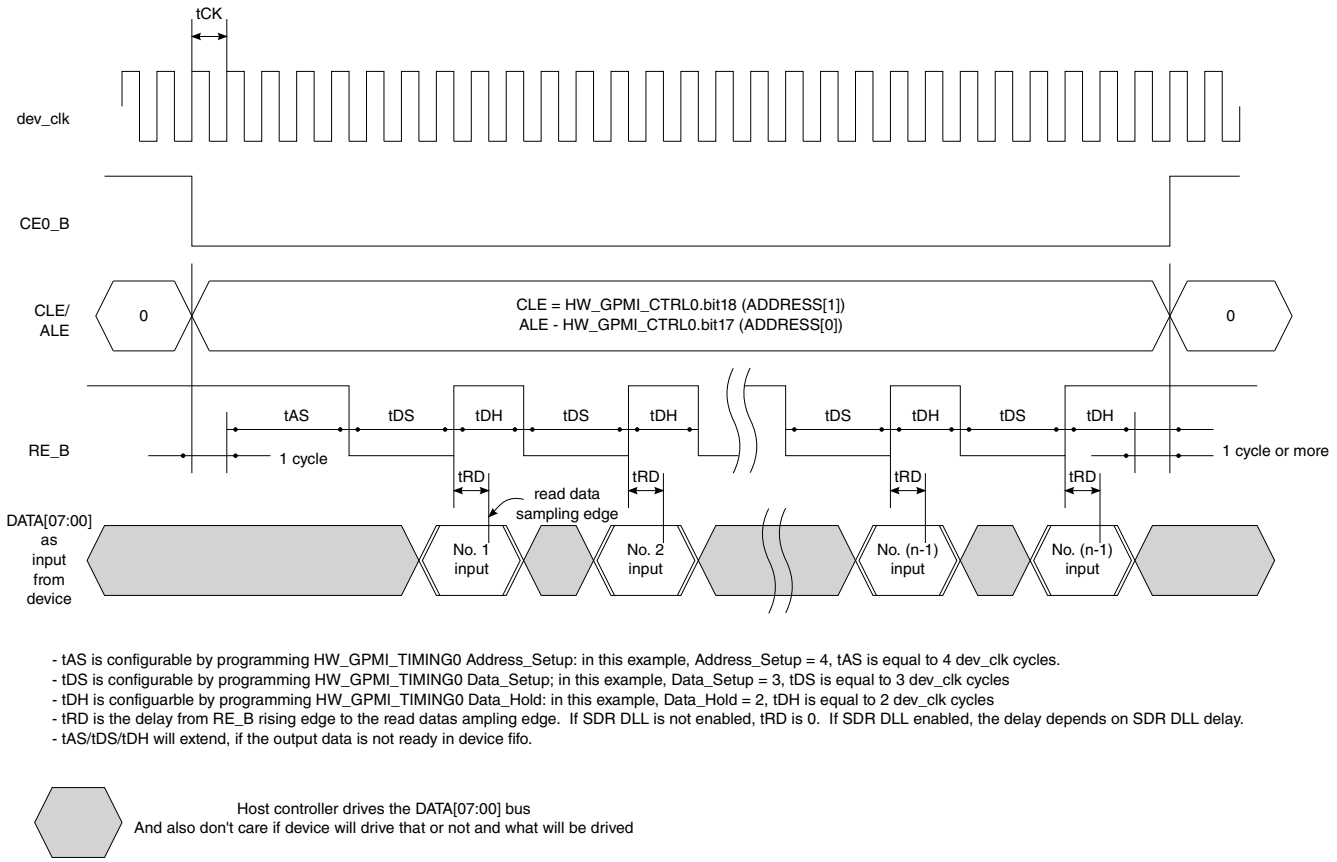


Figure 9-40. ONFI Asynchronous Mode Basic Read Timing Diagram (data read)

9.6.4.3.2 NAND Asynchronous EDO Mode Timing

In high-speed NANDS, the read data may not be valid until after the read strobe (NAND_RE_B) deasserts. This is the case when the minimum tDS is programmed to achieve higher bandwidth.

The GPMI implements a feedback read strobe to sample the read data. The feedback read strobe can be delayed to support fast nand EDO (Extended Data Out) timing where the read strobe may deassert before the read data is valid, and read data is valid for some time after read strobe.

Nand EDO timings is applied typically for read cycle frequency above 33 MHz. See Figure 9-41.

The GPMI provides control over the amount of delay applied to the feedback read strobe. This delay depends on the maximum read access time (tREA) of the nand and the read pulse width (tRP) used to access the nand. tRP is specified by GPMI_TIMING0[DATA_SETUP] register. When (tREA + 4ns) is less than tRP, no

delay is required to sample to nand read data. (The 4ns provides adequate data setup time for the GPMI.) In this case set GPMI_CTRL1[HALF_PERIOD] = 0; GPMI_CTRL1[RDN_DELAY] = 0; GPMI_CTRL1[DLL_ENABLE] = 0.

When (tREA + 4ns) is greater than or equal to tRP, a delay of the feedback read strobe is required to sample to nand read data. This delay is equal to the difference between these two timings:

DELAY = tREA + 4ns - tRP.

Since the GPMI delay chain is limited to ns maximum, if DELAY > ns then increase tRP by increasing the value of GPMI_TIMING0[DATA_SETUP] until DELAY is less than or equal to ns.

The GPMI programming for this DELAY depends on the GPMICLK period. The GPMI DLL will not function properly if the GPMICLK period is greater than ns: disable the DLL if this is the case. If the GPMICLK period is greater than ns (and not greater than ns), set the GPMI_CTRL1[HALF_PERIOD]=1; This will cause the DLL reference period (RP) to be one-half of the GPMICLK period. If the GPMICLK period is ns or less then set the GPMI_CTRL1[HALF_PERIOD]=0; This will cause the DLL reference period (RP) to be equal to the GPMICLK period. DELAY is a multiple (0 to 1.875) of RP.

The GPMI_CTRL1[RDN_DELAY] is encoded as a 1-bit integer and 3-bit fraction delay factor. DELAY is a multiple of the delay factor and the reference period. See table below for details.

Table 9-24. RDN DELAY

HW_GPMI_CTRL1[RDN_DELAY]	Delay Factor
0	0.000
1	0.125
2	0.250
3	0.375
4	0.500
5	0.625
6	0.750
7	0.875
8	1.000
9	1.125
10	1.250
11	1.375
12	1.500
13	1.625

Table continues on the next page...

Table 9-24. RDN DELAY (continued)

HW_GPMI_CTRL1[RDN_DELAY]	Delay Factor
14	1.750
15	1.875

DELAY = DelayFactor x RP or DELAY = GPMI_CTRL1[RDN_DELAY] x 0.125 x RP.

Use this equation to calculate the value for GPMI_CTRL1[RDN_DELAY]. Then set GPMI_CTRL1[DLL_ENABLE]=1.

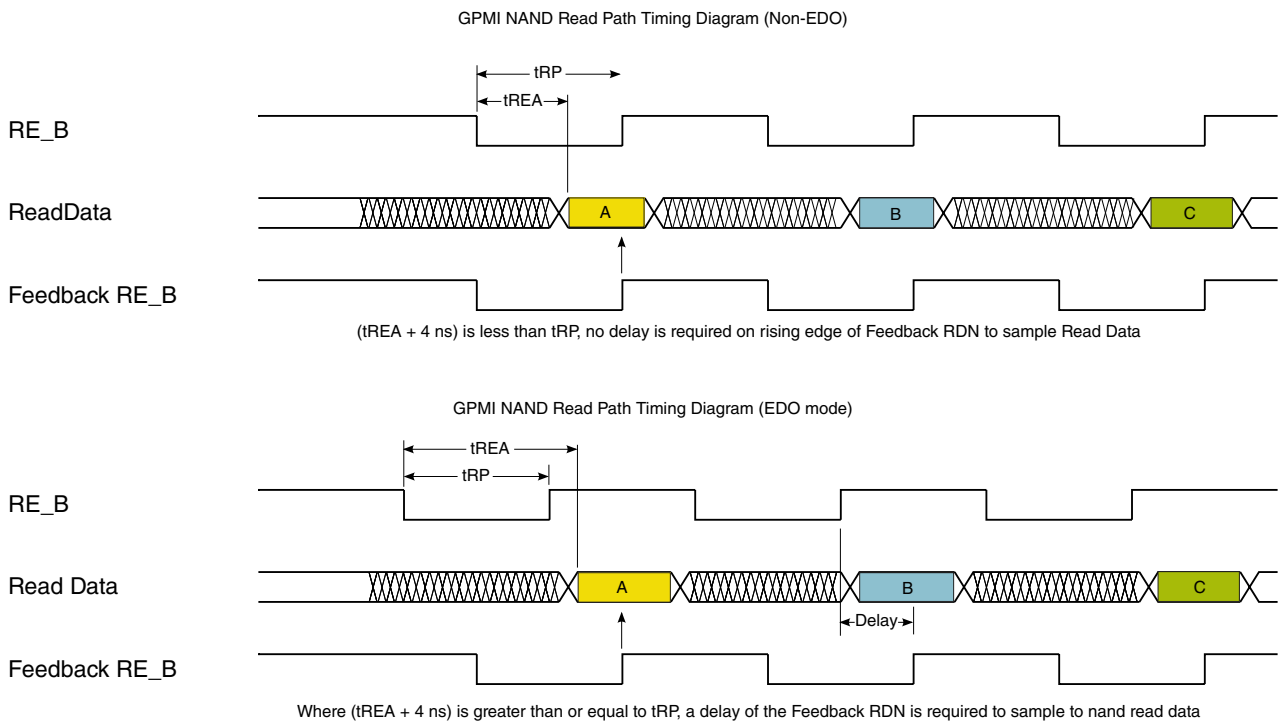


Figure 9-41. NAND Read Path Timing

For example, a NAND with tREAmax = 20 ns, tRPmin = 12 ns, and tRCmin = 25 ns (read cycle time) may be programmed as follows:

- GPMICLK clock frequency: Consider 480/6 = 80 MHz which is 12.5 ns clock period. This is too close to the minimum NAND spec if we program the data setup and hold to 1 GPMICLK cycle. Consider 480/7=68.57 MHz which is 14.58 ns clock period. With data setup and hold set to 1, we have a tRP of 14.58ns and a tRC of 29.16 ns (good margins).
- Since (tREA +4ns) is greater than tRP, required DELAY = tREA + 4ns - tRP = 20 + 4 - 14.58ns = 9.42 ns.

- GPMI_CTRL1[HALF_PERIOD] =, since GPMICLK period is ns. So RP=GPMICLK period = ns.
- DELAY = GPMI_CTRL1[RDN_DELAY] x 0.125 x RP. 9.42 ns = GPMI_CTRL1[RDN_DELAY] x 0.125 x ns. GPMI_CTRL1[RDN_DELAY] =

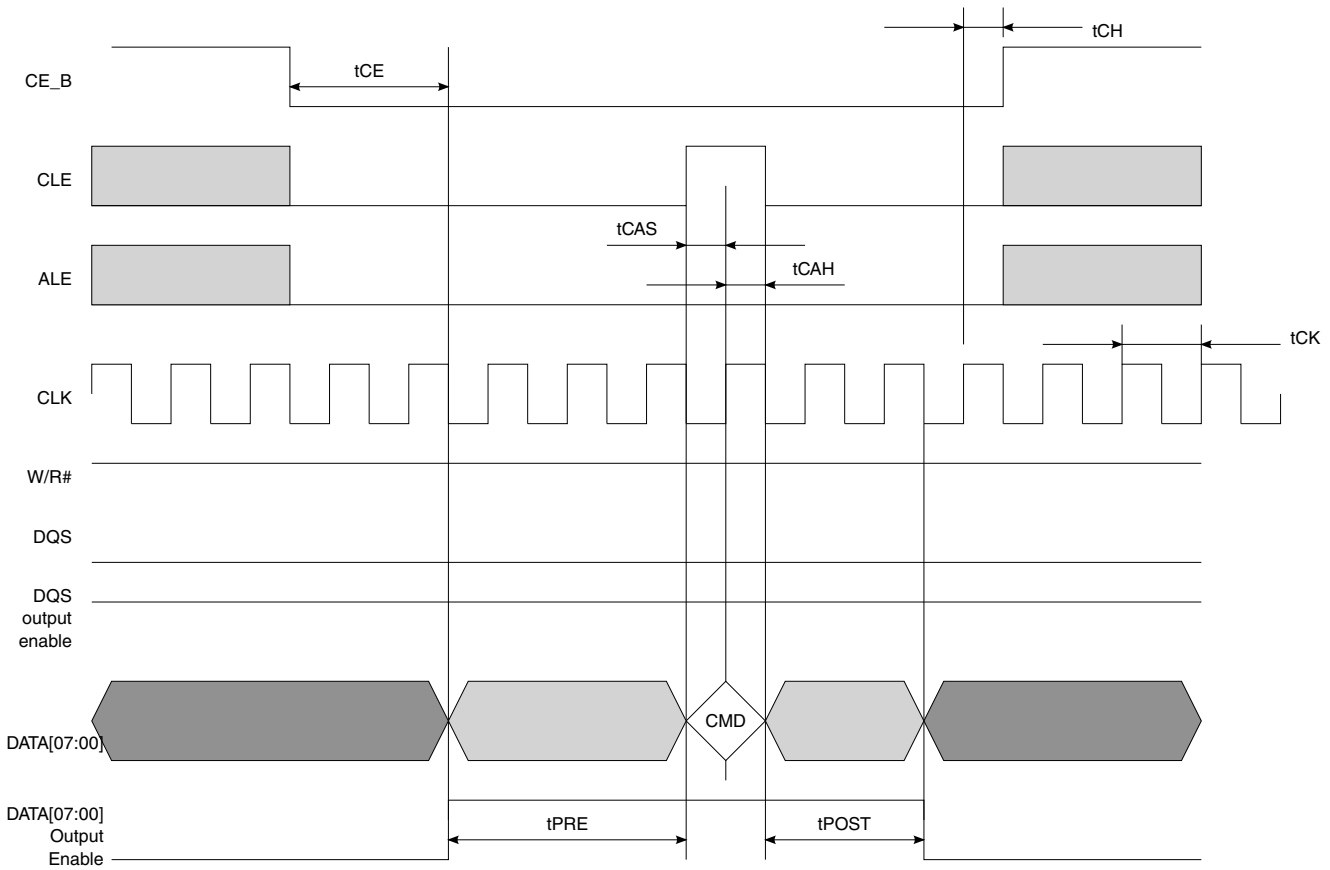
NOTE

It is recommended that the drive strength of NAND_RE_B and NAND_WE_B output pins be set to 8 mA. This will reduce the transition time under heavy loads. Low transition times will be important when NAND interface read and write cycle times are below 30 ns. The other GPMI pins may remain at 4 mA, since their frequency is only up to half that of NAND_RE_B and NAND_WE_B.

9.6.4.3.3 NAND ONFI Source Synchronous Mode Timing

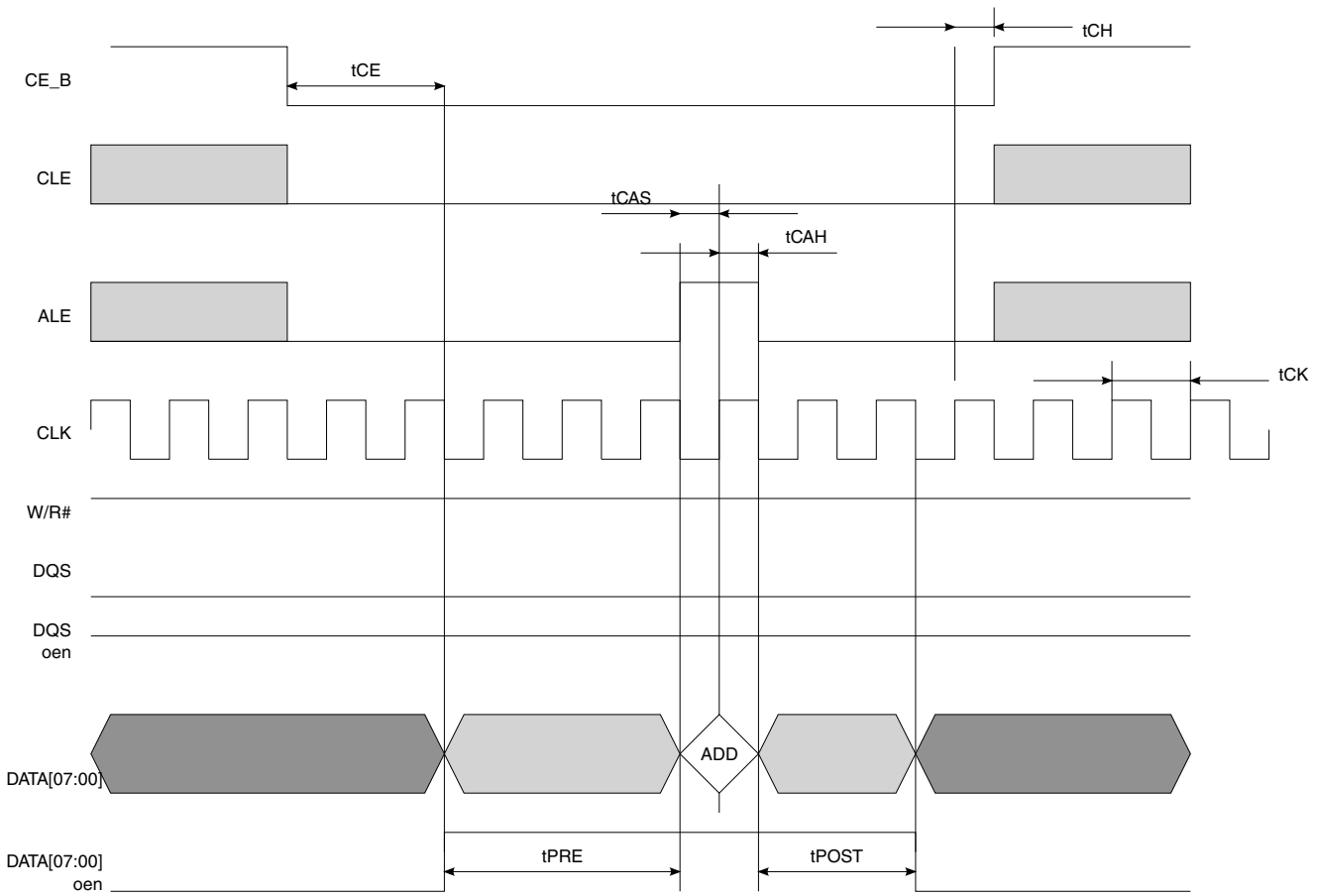
NOTE

In the following figures, CLK shares the same pin as WE_B in Async Mode. And W/R# shares the same pin as RE_B in Async Mode.



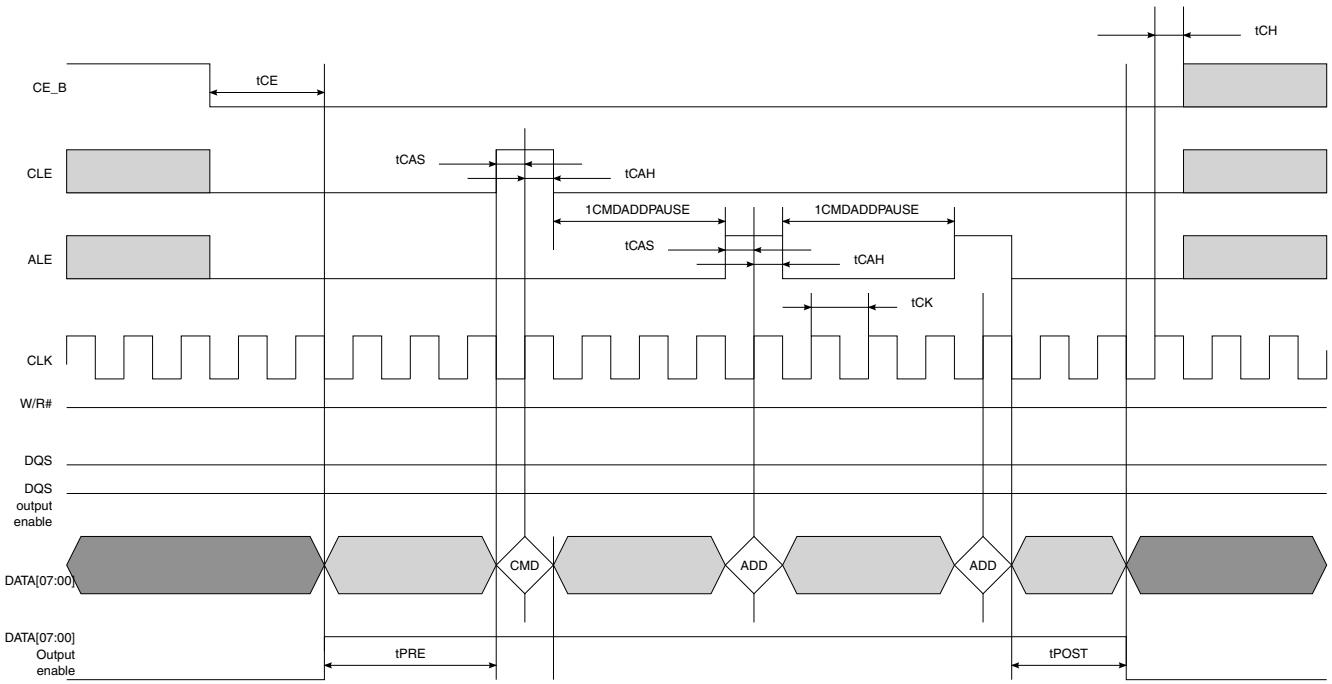
tCK is equal to device clock period
tCAS = 0.5 * tCK
tCAH = 0.5 * tCK
tCE = HW_GPMI_TIMING2.CE_DELAY * tCK
tCH = 0.5 * tCK
tPRE = HW_GPMI_TIMING2.PREAMBLE_DELAY * tCK
tPOST = HW_GPMI_TIMING2.POST_DELAY * tCK

Figure 9-42. ONFI Source Synchronous Mode Basic Command Write Timing Diagram



tCK is equal to device clock period
tCAS = 0.5 * tCK
tCAH = 0.5 * tCK
tCE = HW_GPMI_TIMING2.CE_DELAY * tCK
tCH = 0.5 * tCK
tPRE = HW_GPMI_TIMING2.PREAMBLE_DELAY * tCK
tPOST = HW_GPMI_TIMING2.POST_DELAY * tCK

Figure 9-43. ONFI Source Synchronous Mode Basic Address Write Timing Diagram

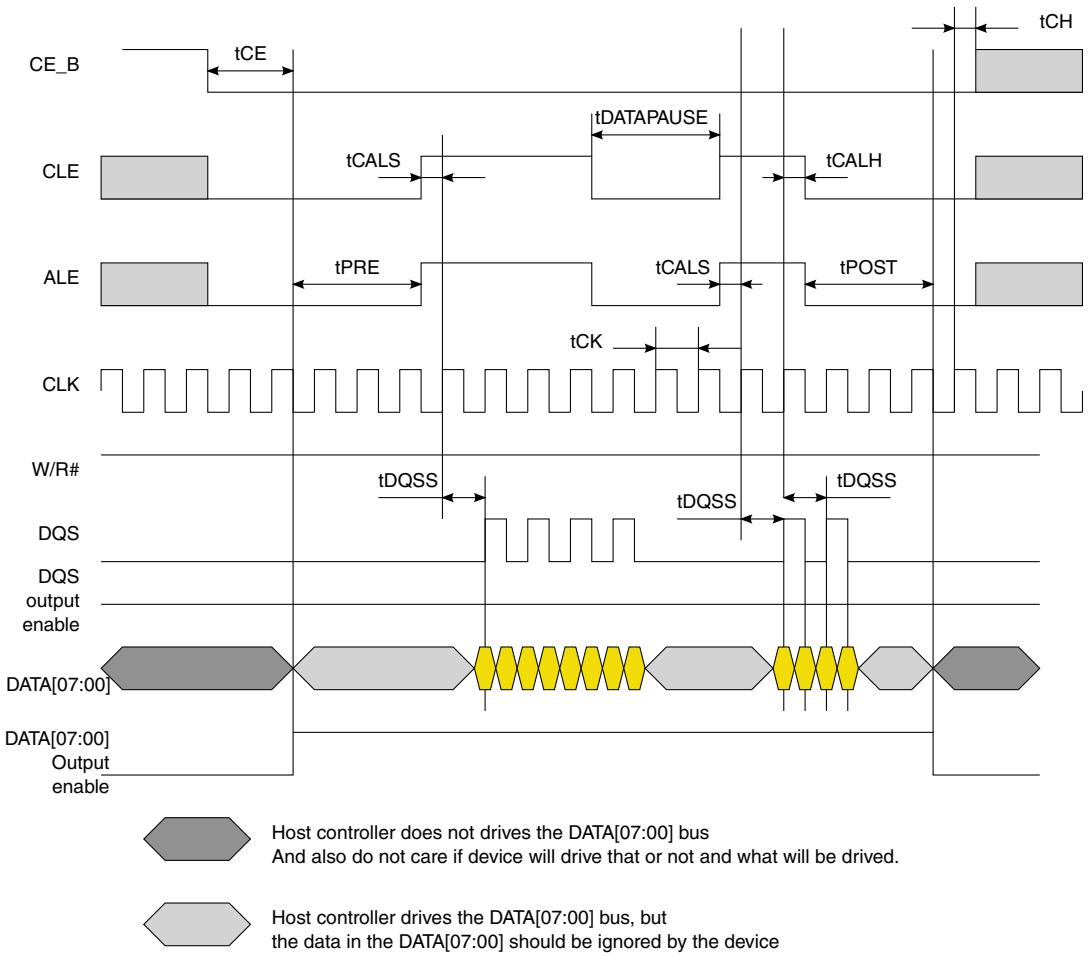


Host controller does not drives the DATA[07:00] bus
And also do not care if device will drive that or not and what will be drived.

Host controller drives the DATA[07:00] bus, but
the data in the DATA[07:00] should be ignored by the device

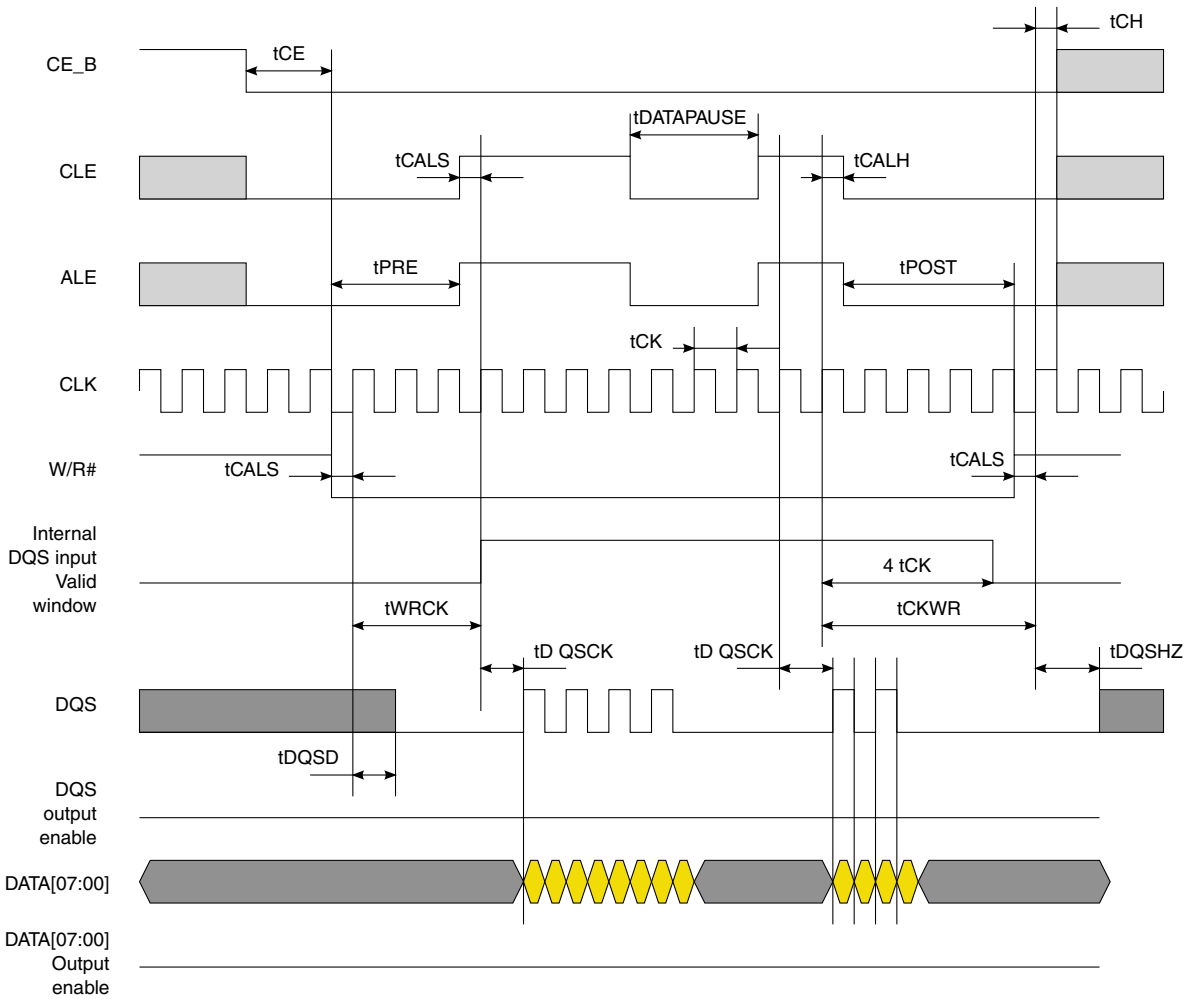
tCK is equal to device clock period
tCAS = 0.5 * tCK
tCAH = 0.5 *tCK
tCE = HW_GPMI_TIMING2.CE_DELAY * tCK
tCH = 0.5 * tCK
tPRE = HW_GPMI_TIMING2.PREAMBLE_DELAY * tCK
tPOST = HW_GPMI_TIMING2.POST_DELAY * tCK
tCMDADDPAUSE = HW_GPMI_TIMING2.CMDADD_PAUSE * tCK

Figure 9-44. ONFI Source Synchronous Mode Command + Address Write Timing Diagram



tCK is equal to device clock period
tCE = HW_GPI_TIMING2.CE_DELAY * tCK
tCH = 0.5 * tCK
tCALS = 0.5 * tCK
tCALH = 0.5 * tCK
tPRE = HW_GPMI_TIMING2.PREAMBLE_DELAY *tCK
tPOST = HW_GPMI_TIMING2.POST_DELAY * tCK
tDATAPAUSE = HW_GPMI_TIMING2.DATA_PAUSE * tCK
tDQSS = tCK

Figure 9-45. ONFI Source Synchronous Mode Data Write Timing Diagram



tCK is equal to device clock period
tCE = HW_GPI_TIMING2.CE_DELAY * tCK
tCH = 0.5 * tCK
tCALS = 0.5 * tCK
tCALH = 0.5 * tCK
tPRE = HW_GPMI_TIMING2.PREAMBLE_DELAY *tCK
tPOST = HW_GPMI_TIMING2.POST_DELAY * tCK
tDATAPAUSE = HW_GPMI_TIMING2.DATA_PAUSE * tCK
tWRCK/tCKWR/tDQSD/tDASCK/tDASHZ are device parameters

Figure 9-46. ONFI Source Synchronous Mode Data Read Timing Diagram

9.6.4.3.4 NAND Toggle Mode Timing

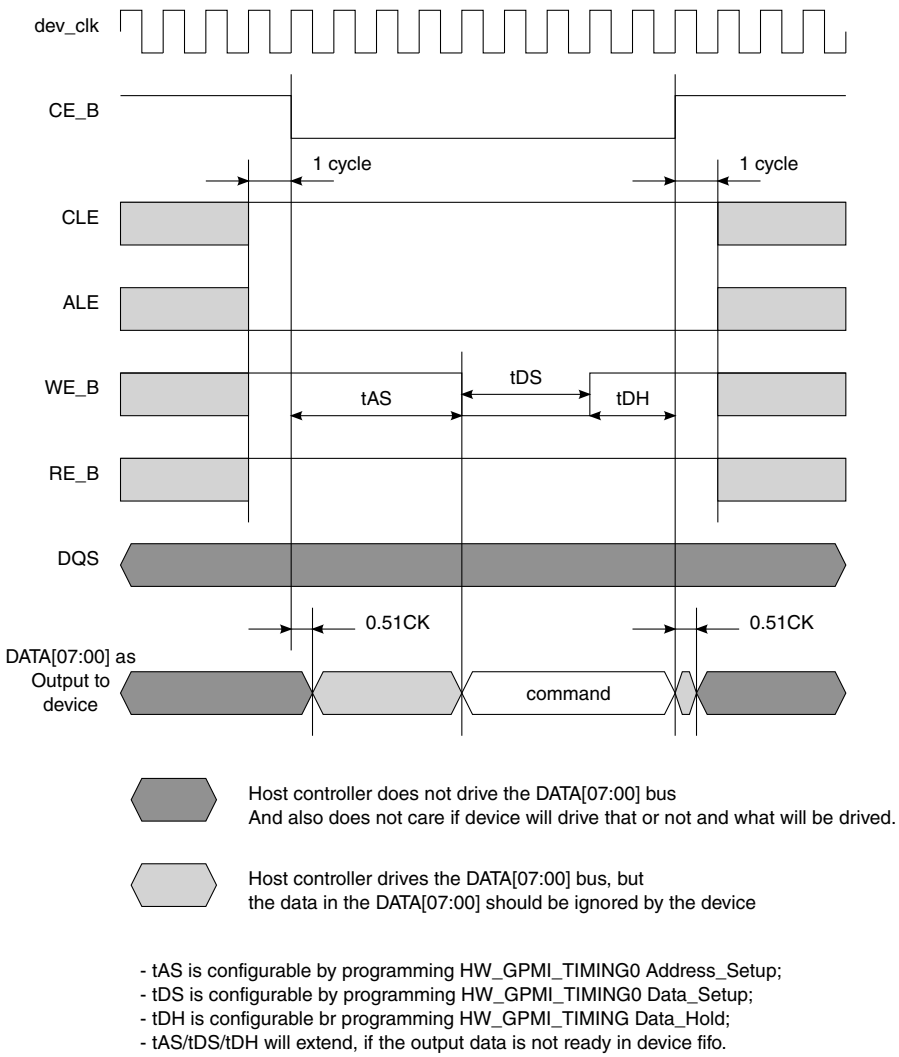
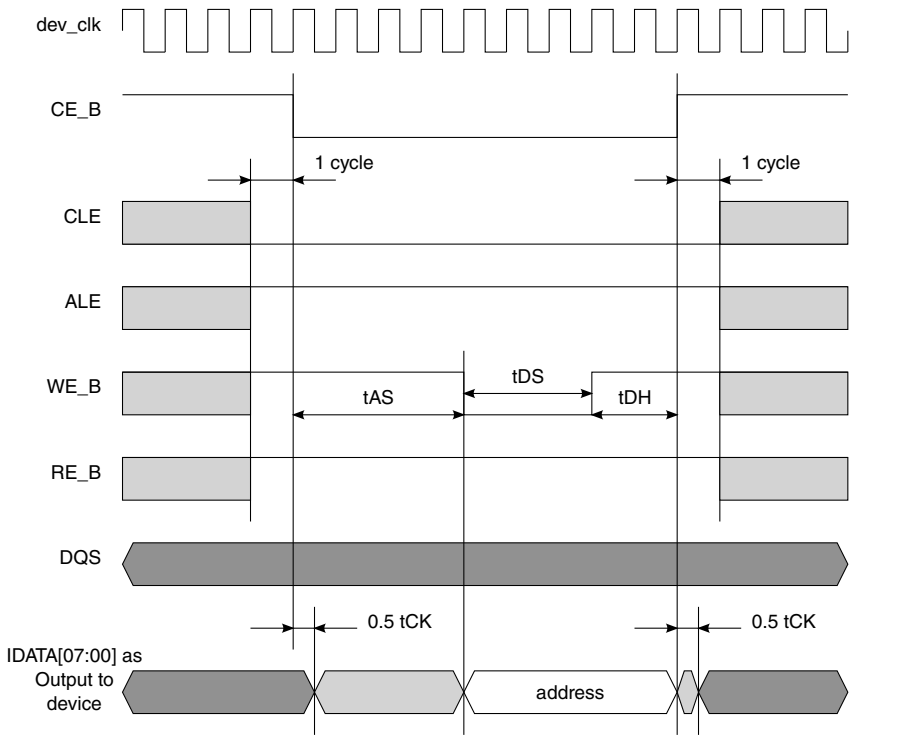


Figure 9-47. Samsung Toggle Mode Basic Command Write Timing Diagram



Host controller does not drive the DATA[07:00] bus
And also does not care if device will drive that or not and what will be drived.

Host controller drives the DATA[07:00] bus, but
the data in the DATA[07:00] should be ignored by the device

- tAS is configurable by programming HW_GPMI_TIMING0 Address_Setup;
- tDS is configurable by programming HW_GPMI_TIMING0 Data_Setup;
- tDH is configurable br programming HW_GPMI_TIMING Data_Hold;
- tAS/tDS/tDH will extend, if the output data is not ready in device fifo.

Figure 9-48. Samsung Toggle Mode Basic Address Write Timing Diagram

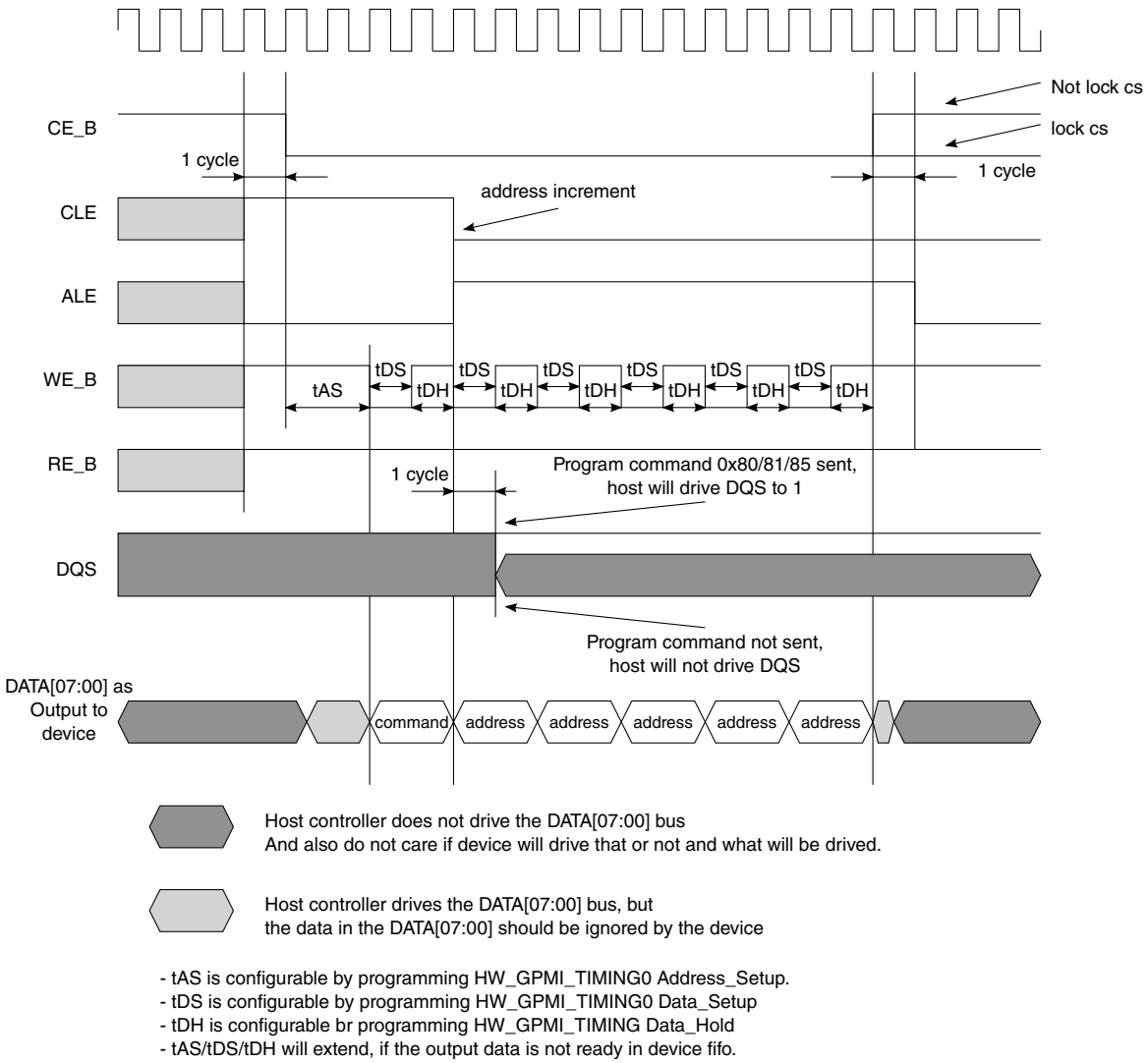


Figure 9-49. Samsung Toggle Mode Basic Command + Address Timing Diagram

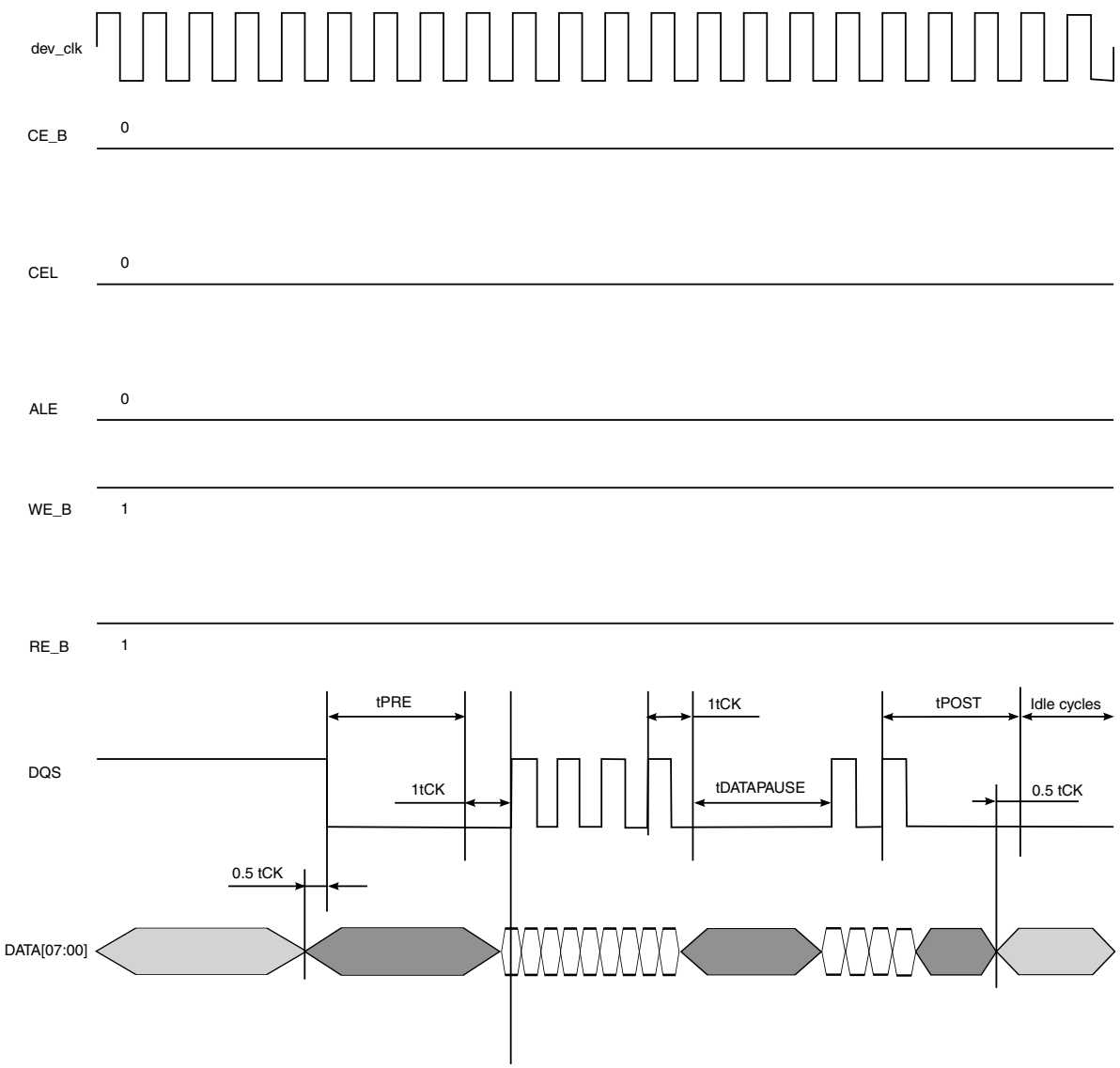
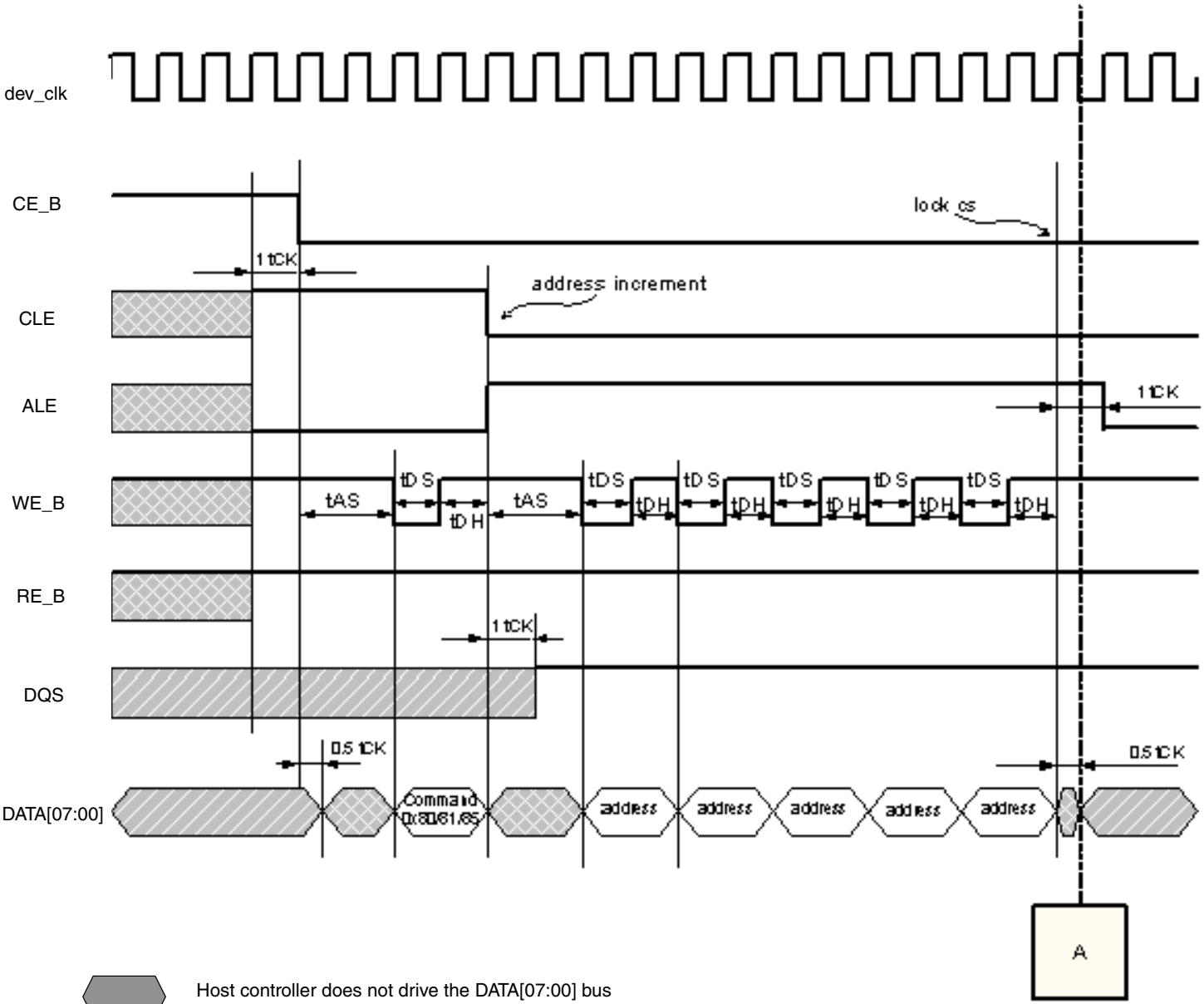


Figure 9-50. Toggle Mode Data Write Timing Diagram

Figure 9-51. Toggle Mode Data Read Timing Diagram

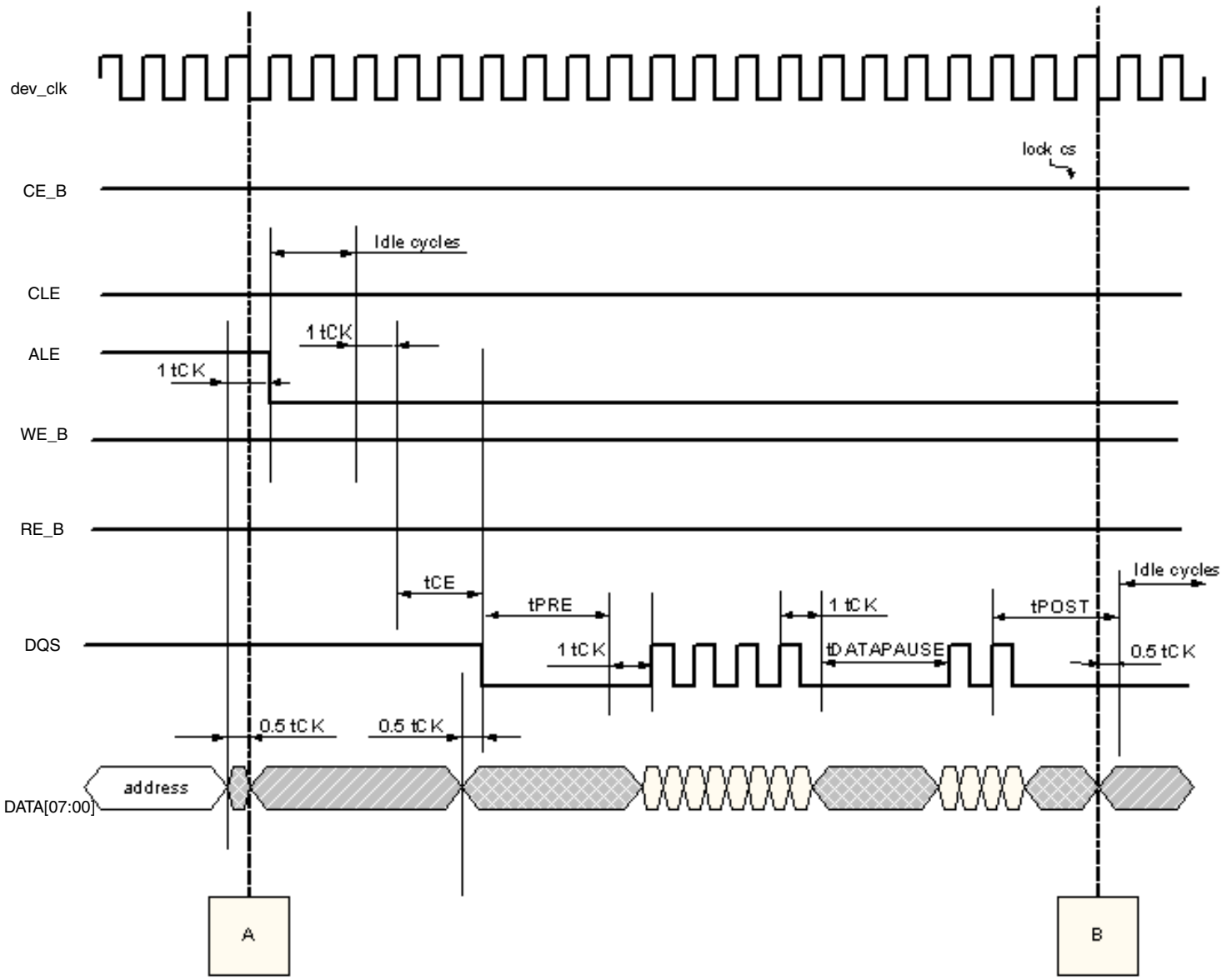


Host controller does not drive the DATA[07:00] bus
And also does not care if device will drive that or not and what will be drived.

Host controller drives the DATA[07:00] bus, but
the data in the DATA[07:00] should be ignored by the device

- tAS is configurable by programming HW_GPMI_TIMING0 Address_Setup:;
- tDS is configurable by programming HW_GPMI_TIMING0 Data_Setup;
- tDH is configuarble by programming HW_GPMI_TIMING0 Data_Hold;
- tAS/tDS/tDH will extend, if the output data is not ready in device fifo.

Figure 9-52. Toggle Mode Program Timing Diagram (A)



Host controller does not drive the DATA[07:00] bus
And also does not care if device will drive that or not and what will be drived.

Host controller drives the DATA[07:00] bus, but
the data in the DATA[07:00] should be ignored by the device

- tCK is equal to device clock period
- tCE = HW_GPMI_TIMING2.CE_DELAY * tCK
- tPRE = HW_GPMI_TIMING2.PREAMBLE_DELAY * tCK
- tPOST = HW_GPMI_TIMING2.POSTAMBLE_DELAY * tCK
- tDATAPULSE = HW_GPMI_TIMING2.DATA_PULSE * tCK

Figure 9-53. Toggle Mode Program Timing Diagram (B)

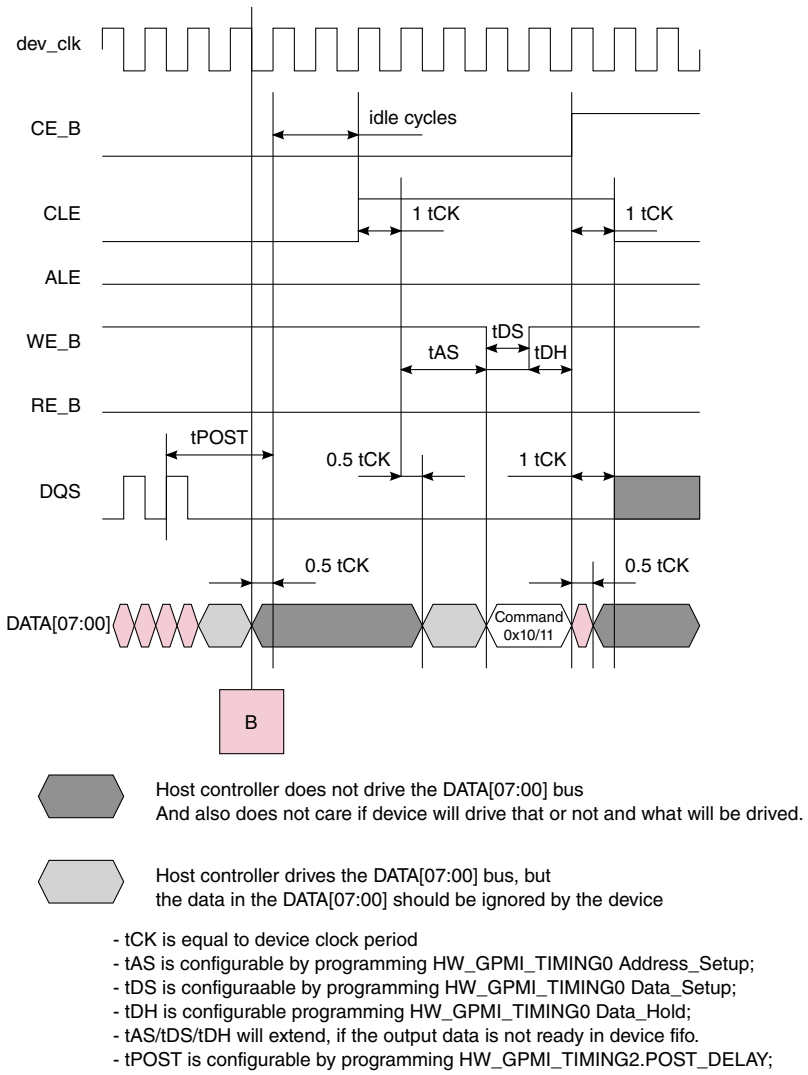


Figure 9-54. Toggle Mode Program Timing Diagram (C)

9.6.4.4 Hardware BCH Interface

The GPMI provides an interface to the BCH module. This reduces the SOC bus traffic and the software involvement.

The GPMI also provides an interface to the Randomizer module. The Randomizer can generate random data based on BCH ECC encoded / decoded data. It can be employed to reduce the disturbances caused by neighboring cells in the NAND chip, thus reducing bit errors. To enable the Randomizer module, set GPMI_ECCCTRL[RANDOMIZER_ENABLE] to 1, then set GPMI_ECCCOUNT[RANDOMIZER_PAGE] to select randomizer page number needed

to be randomized. All these registers can be programmed by the DMA chain. The randomized data should start from the zero column address and be the size of the whole NAND page. If the randomizer function is enabled, GPMI_ECCCTRL[ENABLE_ECC] should also be enabled. To bypass BCH error correction function, set BCH_FLASHxLAYOUT0[ECC0] and BCH_FLASHxLAYOUT1[ECCN] to 0.

When BCH ECC is enable, parity information is inserted on-the-fly during writes to 8-bit NAND devices. The BCH will supply payload and parity to the GPMI to write to the NAND. During NAND reads, parity is checked and ECC processing is performed after each read block. In this case the GPMI reads the NAND device and redirects the data and parity to the BCH module for ECC processing.

To program the BCH for NAND writes, remove the soft reset and clock gates from BCH_CTRL[SFTRST] and BCH_CTRL[CLKGATE]. The bulk of BCH programming is actually applied to the GPMI via PIO operations embedded in its DMA command structures. This has a subtle implication when writing to the GPMI ECC registers: access to the these registers must be written in progressive register order. Thus, to write to the GPMI_ECCCOUNT register, write first (in order) to registers GPMI_CTRL0, GPMI_COMPARE, and GPMI_ECCCTRL before writing to GPMI_ECCCOUNT. These additional register writes need to be accounted for in the CMDWORDS field of the respective DMA channel command register.

Note that the GPMI_PAYLOAD and GPMI_AUXILIARY pointers need to be word-aligned for proper ECC operation. If those pointers are non-word-aligned, then the BCH engine will not operate properly and could possibly corrupt system memory in the adjoining memory regions.

9.6.5 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically.

9.6.6 GPMI Memory Map/Register Definition

The following registers provide control for programmable elements of the GPMI module.

NOTE

All ATA or UDMA features are not supported for the chip.

NOTE

GPMI does not support the Set feature command in Toggle mode. The NANDF_DQS output is only enabled in program operation for Toggle mode, but the Set feature command also needs to use the NANDF_DQS signal to write data to Toggle NAND flash. So the Set feature command in Toggle mode is not supported.

GPMI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3300_2000	GPMI Control Register 0 Description (GPMI_CTRL0)	32	R/W	C000_0000h	9.6.6.1/ 2467
3300_2004	GPMI Control Register 0 Description (GPMI_CTRL0_SET)	32	R/W	C000_0000h	9.6.6.1/ 2467
3300_2008	GPMI Control Register 0 Description (GPMI_CTRL0_CLR)	32	R/W	C000_0000h	9.6.6.1/ 2467
3300_200C	GPMI Control Register 0 Description (GPMI_CTRL0_TOG)	32	R/W	C000_0000h	9.6.6.1/ 2467
3300_2010	GPMI Compare Register Description (GPMI_COMPARE)	32	R/W	0000_0000h	9.6.6.2/ 2469
3300_2020	GPMI Integrated ECC Control Register Description (GPMI_ECCCTRL)	32	R/W	0000_0000h	9.6.6.3/ 2470
3300_2024	GPMI Integrated ECC Control Register Description (GPMI_ECCCTRL_SET)	32	R/W	0000_0000h	9.6.6.3/ 2470
3300_2028	GPMI Integrated ECC Control Register Description (GPMI_ECCCTRL_CLR)	32	R/W	0000_0000h	9.6.6.3/ 2470
3300_202C	GPMI Integrated ECC Control Register Description (GPMI_ECCCTRL_TOG)	32	R/W	0000_0000h	9.6.6.3/ 2470
3300_2030	GPMI Integrated ECC Transfer Count Register Description (GPMI_ECCCOUNT)	32	R/W	0000_0000h	9.6.6.4/ 2471
3300_2040	GPMI Payload Address Register Description (GPMI_PAYLOAD)	32	R/W	0000_0000h	9.6.6.5/ 2472
3300_2050	GPMI Auxiliary Address Register Description (GPMI_AUXILIARY)	32	R/W	0000_0000h	9.6.6.6/ 2472

Table continues on the next page...

GPMI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3300_2060	GPMI Control Register 1 Description (GPMI_CTRL1)	32	R/W	0004_0004h	9.6.6.7/ 2473
3300_2064	GPMI Control Register 1 Description (GPMI_CTRL1_SET)	32	R/W	0004_0004h	9.6.6.7/ 2473
3300_2068	GPMI Control Register 1 Description (GPMI_CTRL1_CLR)	32	R/W	0004_0004h	9.6.6.7/ 2473
3300_206C	GPMI Control Register 1 Description (GPMI_CTRL1_TOG)	32	R/W	0004_0004h	9.6.6.7/ 2473
3300_2070	GPMI Timing Register 0 Description (GPMI_TIMING0)	32	R/W	0001_0203h	9.6.6.8/ 2476
3300_2080	GPMI Timing Register 1 Description (GPMI_TIMING1)	32	R/W	0000_0000h	9.6.6.9/ 2476
3300_2090	GPMI Timing Register 2 Description (GPMI_TIMING2)	32	R/W	0302_3336h	9.6.6.10/ 2477
3300_20A0	GPMI DMA Data Transfer Register Description (GPMI_DATA)	32	R/W	0000_0000h	9.6.6.11/ 2478
3300_20B0	GPMI Status Register Description (GPMI_STAT)	32	R	0000_0005h	9.6.6.12/ 2478
3300_20C0	GPMI Debug Information Register Description (GPMI_DEBUG)	32	R	0000_0000h	9.6.6.13/ 2481
3300_20D0	GPMI Version Register Description (GPMI_VERSION)	32	R	0502_0000h	9.6.6.14/ 2482
3300_20E0	GPMI Debug2 Information Register Description (GPMI_DEBUG2)	32	R/W	0000_F100h	9.6.6.15/ 2482
3300_20F0	GPMI Debug3 Information Register Description (GPMI_DEBUG3)	32	R	0000_0000h	9.6.6.16/ 2485
3300_2100	GPMI Double Rate Read DLL Control Register Description (GPMI_READ_DDR_DLL_CTRL)	32	R/W	0000_0038h	9.6.6.17/ 2486
3300_2110	GPMI Double Rate Write DLL Control Register Description (GPMI_WRITE_DDR_DLL_CTRL)	32	R/W	0000_0038h	9.6.6.18/ 2487
3300_2120	GPMI Double Rate Read DLL Status Register Description (GPMI_READ_DDR_DLL_STS)	32	R	0000_0000h	9.6.6.19/ 2489
3300_2130	GPMI Double Rate Write DLL Status Register Description (GPMI_WRITE_DDR_DLL_STS)	32	R	0000_0000h	9.6.6.20/ 2490

9.6.6.1 GPMI Control Register 0 Description (GPMI_CTRL0*n*)

The GPMI control register 0 specifies the GPMI transaction to perform for the current command chain item.

Address: 3300_2000h base + 0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SFTRST	CLKGATE	RUN	DEV_IRQ_EN	LOCK_CS	UDMA	COMMAND_MODE		WORD_LENGTH	CS			ADDRESS			ADDRESS_INCREMENT
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XFER_COUNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_CTRL0*n* field descriptions

Field	Description
31 SFTRST	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state. This will not work if the CLKGATE bit is already set to '1'. CLKGATE must be cleared to '0' before issuing a soft reset. Also the GPMICLK must be running for this to work properly. RUN = 0x0 Allow GPMI to operate normally. RESET = 0x1 Hold GPMI in reset.
30 CLKGATE	Set this bit zero for normal operation. Setting this bit to one (default), gates all of the block level clocks off for miniminizing AC energy consumption. RUN = 0x0 Allow GPMI to operate normally. NO_CLKS = 0x1 Do not clock GPMI gates in order to minimize power consumption.
29 RUN	The GPMI is busy running a command whenever this bit is set to '1'. The GPMI is idle whenever this bit set to zero. This can be set to one by a CPU write. In addition, the DMA sets this bit each time a DMA command has finished its PIO transfer phase. IDLE = 0x0 The GPMI is idle. BUSY = 0x1 The GPMI is busy running a command.

Table continues on the next page...

GPMI_CTRL0*n* field descriptions (continued)

Field	Description
28 DEV_IRQ_EN	When set to '1' and ATA_IRQ pin is asserted, the GPMI_IRQ output will assert.
27 LOCK_CS	For ATA/NAND mode: 0= Deassert chip select (CS) after RUN is complete. 1= Continue to assert chip select (CS) after RUN is complete. For Camera Mode: 0= Dont wait for VSYNC rising edge before capturing data. 1= Wait for VSYNC rising edge before capturing data (Camera mode only). DISABLED = 0x0 Deassert chip select (CS) after RUN is complete. ENABLED = 0x1 Continue to assert chip select (CS) after RUN is complete.
26 UDMA	DISABLED = 0x0 Use ATA-PIO mode on the external bus. ENABLED = 0x1 Use ATA-Ultra DMA mode on the external bus. 0 Use ATA-PIO mode on the external bus. 1 Use ATA-Ultra DMA mode on the external bus.
25–24 COMMAND_ MODE	WRITE = 0x0 Write mode. READ = 0x1 Read mode. READ_AND_COMPARE = 0x2 Read and Compare mode (setting sense flop). WAIT_FOR_READY = 0x3 Wait for Ready mode. For ATA WAIT_FOR_READY command set CS=01. 00 Write mode. 01 Read Mode. 10 Read and Compare Mode (setting sense flop). 11 Wait for Ready.
23 WORD_LENGTH	This bit should only be changed when RUN==0. Reserve = 0x0 Reserved. 8_BIT = 0x1 8-bit Data Bus mode. 0 Reserved. 1 8-bit Data Bus mode.
22–20 CS	Selects which chip select is active for this command. For ATA WAIT_FOR_READY command, this must be set to b01.
19–17 ADDRESS	Specifies the three address lines for ATA mode. In NAND mode, use A0 for CLE and A1 for ALE. NAND_DATA = 0x0 In NAND mode, this address is used to read and write data bytes. NAND_CLE = 0x1 In NAND mode, this address is used to write command bytes. NAND_ALE = 0x2 In NAND mode, this address is used to write address bytes.
16 ADDRESS_ INCREMENT	In ATA mode, the address will increment with each cycle. In NAND mode, the address will increment once, after the first cycle (going from CLE to ALE). DISABLED = 0x0 Address does not increment. ENABLED = 0x1 Increment address. 0 Address does not increment. 1 Increment address.
XFER_COUNT	Number of bytes to transfer for this command. A value of zero will transfer 64K bytes.

9.6.6.2 GPMI Compare Register Description (GPMI_COMPARE)

The GPMI compare register specifies the expect data and the xor mask for comparing to the status values read from the device. This register is used by the Read and Compare command.

GPMI_COMPARE 0x010

Address: 3300_2000h base + 10h offset = 3300_2010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MASK																REFERENCE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

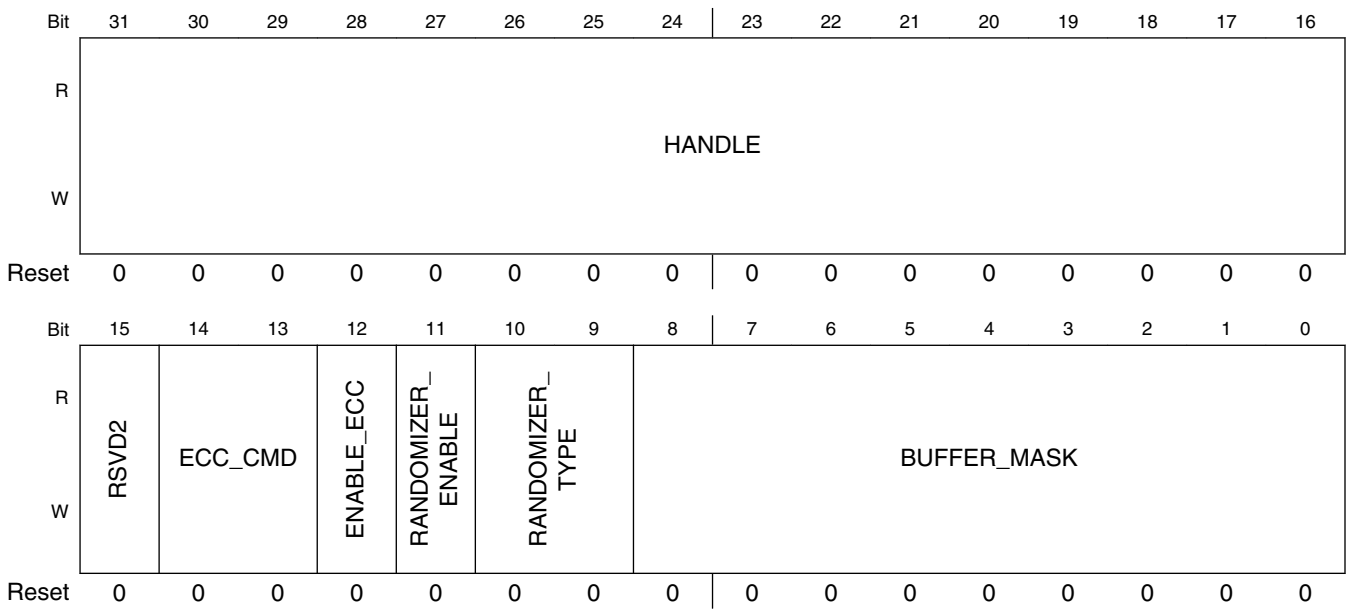
GPMI_COMPARE field descriptions

Field	Description
31–16 MASK	16-bit mask which is applied after the read data is XORed with the REFERENCE bit field.
REFERENCE	16-bit value which is XORed with data read from the NAND device.

9.6.6.3 GPMI Integrated ECC Control Register Description (GPMI_ECCCTRL*n*)

The GPMI ECC control register handles configuration of the integrated ECC / Randomizer accelerator.

Address: 3300_2000h base + 20h offset + (4d × i), where i=0d to 3d



GPMI_ECCCTRL*n* field descriptions

Field	Description
31–16 HANDLE	This is a register available to software to attach an identifier to a transaction in progress. This handle will be available from the ECC register space when the completion interrupt occurs.
15 RSVD2	Always write zeroes to this bit field.
14–13 ECC_CMD	ECC Command information. DECODE = 0x0 Decode. ENCODE = 0x1 Encode. RESERVE2 = 0x2 Reserved. RESERVE3 = 0x3 Reserved.
12 ENABLE_ECC	Enable ECC processing of GPMI transfers. ENABLE = 0x1 Use integrated ECC for read and write transfers. DISABLE = 0x0 Integrated ECC remains in idle.
11 RANDOMIZER_ ENABLE	Enable randomizer function. If this bit is set to enable, ENABLE_ECC should be also enable.

Table continues on the next page...

GPMI_ECCCTRL*n* field descriptions (continued)

Field	Description
	0x0 disable 0x1 enable
10–9 RANDOMIZER_ TYPE	Set randomizer type 0x00 Type 0 0x01 Type 1 0x10 Type 2
BUFFER_MASK	ECC buffer information. The BCH error correction only allows two configurations of the buffer mask - software may either read just the first block on the flash page or the entire flash page. Write operations must be for the entire flash page. Invalid buffer mask values will cause the DMA descriptor command to be terminated. BCH_AUXONLY = 0x100 Set to request transfer from only the auxiliary buffer (block 0 on flash). BCH_PAGE = 0x1FF Set to request transfer to/from the entire page.

9.6.6.4 GPMI Integrated ECC Transfer Count Register Description (GPMI_ECCCOUNT)

The GPMI ECC Transfer Count Register contains the count of bytes that flow through the ECC / Randomizer subsystem.

GPMI_ECCCOUNT 0x030

Address: 3300_2000h base + 30h offset = 3300_2030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								RANDOMIZER_PAGE								COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_ECCCOUNT field descriptions

Field	Description
31–24 -	Always write zeroes to this bit field.
23–16 RANDOMIZER_ PAGE	Set NAND page number needed to be randomized. The value is between 0-255
COUNT	Number of bytes to pass through ECC. This is the GPMI transfer count plus the syndrome count that will be inserted into the stream by the ECC. In DMA2ECC_MODE this count must match the GPMI_CTRL0_XFER_COUNT. A value of zero will transfer 64K words.

9.6.6.5 GPMI Payload Address Register Description (GPMI_PAYLOAD)

The GPMI payload address register specifies the location of the data buffers in system memory. This value must be word aligned.

GPMI_PAYLOAD 0x040

Address: 3300_2000h base + 40h offset = 3300_2040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDRESS														RSVD0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_PAYLOAD field descriptions

Field	Description
31–2 ADDRESS	Pointer to an array of one or more 512 byte payload buffers.
RSVD0	Always write zeroes to this bit field.

9.6.6.6 GPMI Auxiliary Address Register Description (GPMI_AUXILIARY)

The GPMI auxiliary address register specifies the location of the auxiliary buffers in system memory. This value must be word aligned.

GPMI_AUXILIARY 0x050

Address: 3300_2000h base + 50h offset = 3300_2050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS														RSVD0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_AUXILIARY field descriptions

Field	Description
31–2 ADDRESS	Pointer to ECC control structure and meta-data storage.
RSVD0	Always write zeroes to this bit field.

9.6.6.7 GPMI Control Register 1 Description (GPMI_CTRL1*n*)

The GPMI control register 1 specifies additional control fields that are not used on a per-transaction basis.

Address: 3300_2000h base + 60h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DEV_CLK_STOP	SSYNC_CLK_STOP	WRITE_CLK_STOP	TOGGLE_MODE	GPMI_CLK_DIV2_EN	UPDATE_CS	SSYNCMODE	DECOUPLE_CS	WRN_DLY_SEL		TEST_TRIGGER	TIMEOUT_IRQ_EN	GANGED_RDYBUSY	BCH_MODE	DLL_ENABLE	HALF_PERIOD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RDN_DELAY				DMA2ECC_MODE	DEV_IRQ	TIMEOUT_IRQ	BURST_EN	ABORT_WAIT_REQUEST	ABORT_WAIT_FOR_READY_CHANNEL			DEV_RESET	ATA_IRQRDY_POLARITY	CAMERA_MODE	GPMI_MODE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

GPMI_CTRL1*n* field descriptions

Field	Description
31 DEV_CLK_STOP	set this bit to 1 will stop gpmi io working clk.
30 SSYNC_CLK_STOP	set this bit to 1 will stop the source synchronous mode clk.

Table continues on the next page...

GPMI_CTRL1*n* field descriptions (continued)

Field	Description
29 WRITE_CLK_STOP	In onfi source synchronous mode, host may save power during the data write cycles by holding the CLK signal high (i.e. stopping the CLK). The host may only stop the CLK during data write, by setting this bit to 1, if the device supports this feature as indicated in the parameter page.
28 TOGGLE_MODE	enable samsung toggle mode.
27 GPMI_CLK_DIV2_EN	This bit should be reset to 0 in asynchronous mode. The frequence ratio of (device clock : ccm gpmi clock) will be (1 : 1). This bit should be set to 1, in source synchronous mode or toggle mode. The frequence raito of (device clock : ccm gpmi clock) will be (1 : 2). enable the gpmi clk divider. 0x0 internal factor-2 clock divider is disabled 0x1 internal factor-2 clock divider is enabled.
26 UPDATE_CS	force the CS value is be updated to external chip select pin, even GPMI is idle.
25 SSYNCMODE	source synchronouse mode 1 or asynchrous mode 0. ASYNC = 0x0 Asynchronous mode. SSYNC = 0x1 Source Synchronous mode.
24 DECOUPLE_CS	Decouple Chip Select from DMA Channel. Setting this bit to 1 will allow a DMA channel to specify any value in the CTRL0_CS register field. Software can use one DMA channel to access all 8 Nand devices.
23–22 WRN_DLY_SEL	Since the GPMI write strobe (WRN) is a fast clock pin, the delay on this signal can be programmed to match the load on this pin. 0 = ~2ns; 1 = ~4ns; 2 = ~6ns; 3 = no delay.
21 TEST_TRIGGER	Test Trigger Enable 0 Disable 1 Enable
20 TIMEOUT_IRQ_EN	Setting this bit to '1' will enable timeout IRQ for transfers in ATA mode only, and for WAIT_FOR_READY commands in both ATA and Nand mode. The Device_Busy_Timeout value is used for this timeout.
19 GANGED_RDYBUSY	Set this bit to 1 will force all Nand RDY_BUSY inputs to be sourced from (tied to) RDY_BUSY0. This will free up all, except one, RDY_BUSY input pins.
18 BCH_MODE	This bit selects which error correction unit will access GPMI. This bit must always be set to '1', since only the BCH unit is available in this design.
17 DLL_ENABLE	Set this bit to 1 to enable the GPMI DLL. This is required for fast NAND reads (above 30 MHz read strobe). After setting this bit, wait 64 GPMI clock cycles for the DLL to lock before performing a NAND read.
16 HALF_PERIOD	Set this bit to 1 if the GPMI clock period is greater than 16ns for proper DLL operation. DLL_ENABLE must be zero while changing this field.
15–12 RDN_DELAY	This variable is a factor in the calculated delay to apply to the internal read strobe for correct read data sampling. The applied delay (AD) is between 0 and 1.875 times the reference period (RP). RP is one half of the GPMI clock period if HALF_PERIOD=1 otherwise it is the full GPMI clock period. The equation is: AD = RDN_DELAY x 0.125 x RP. This value must not exceed 16ns.

Table continues on the next page...

GPMI_CTRL1*n* field descriptions (continued)

Field	Description
	This variable is used to achieve faster NAND access. For example if the Read Strobe is asserted from time 0 to 13ns but the read access time is 20ns, then choose AD=12ns will cause the data to be sampled at time 25ns (13+12) giving a 5ns data setup time. If RP=13ns then RDN_DELAY = 12/(0.125 x 13ns) = 7.38 (0111b). DLL_ENABLE must be zero while changing this field.
11 DMA2ECC_ MODE	This is mainly for testing HWECC without involving the Nand device. Setting this bit will cause DMA write data to redirected to HWECC module (instead of Nand Device) for encoding or decoding.
10 DEV_IRQ	This bit is set when an Interrupt is received from the ATA device. Write 0 to clear.
9 TIMEOUT_IRQ	This bit is set when a timeout occurs using the Device_Busy_Timeout value. Write 0 to clear.
8 BURST_EN	When set to 1 each DMA request will generate a 4-transfer burst on the APB bus.
7 ABORT_WAIT_ REQUEST	Request to abort "wait for ready" command on channel indicated by ABORT_WAIT_FOR_READY_CHANNEL. Hardware will clear this bit when abort is done.
6–4 ABORT_WAIT_ FOR_READY_ CHANNEL	Abort a wait for ready command on selected channel. Set the ABORT_WAIT_REQUEST to kick of operation.
3 DEV_RESET	ENABLED = 0x0 NANDF_WP_B(WPN) pin is held low (asserted). DISABLED = 0x1 NANDF_WP_B(WPN) pin is held high (de-asserted). 0 NANDF_WP_B pin is held low (asserted). 1 NANDF_WP_B pin is held high (de-asserted).
2 ATA_IRQRDY_ POLARITY	For ATA MODE: Note NAND_RDY_BUSY[3:2] are not affected by this bit. ACTIVELOW = 0x0 ATA IORDY and IRQ are active low, or NAND_RDY_BUSY[1:0] are active low ready. ACTIVEHIGH = 0x1 ATA IORDY and IRQ are active high, or NAND_RDY_BUSY[1:0] are active high ready. 0 External ATA IORDY and IRQ are active low. 1 External ATA IORDY and IRQ are active high. For NAND MODE: 0 External RDY_BUSY[1] and RDY_BUSY[0] pins are ready when low and busy when high. 1 External RDY_BUSY[1] and RDY_BUSY[0] pins are ready when high and busy when low.
1 CAMERA_MODE	When set to 1 and ATA UDMA is enabled the UDMA interface becomes a camera interface.
0 GPMI_MODE	ATA mode is only supported on channel zero. If ATA mode is selected, then only channel three is available for NAND use. NAND = 0x0 NAND mode. ATA = 0x1 ATA mode.

Table continues on the next page...

GPMI_CTRL1*n* field descriptions (continued)

Field	Description
	0 NAND mode. 1 ATA mode.

9.6.6.8 GPMI Timing Register 0 Description (GPMI_TIMING0)

The GPMI timing register 0 specifies the timing parameters that are used by the cycle state machine to guarantee the various setup, hold and cycle times for the external media type.

GPMI_TIMING0 0x070

Address: 3300_2000h base + 70h offset = 3300_2070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									ADDRESS_SETUP								DATA_HOLD								DATA_SETUP							
W	0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1

GPMI_TIMING0 field descriptions

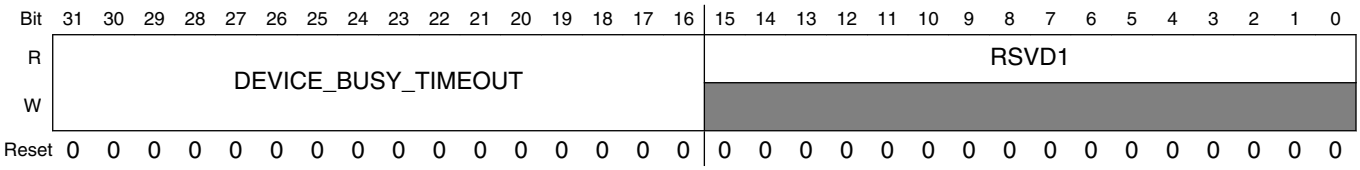
Field	Description
31–24 RSVD1	Always write zeroes to this bit field.
23–16 ADDRESS_ SETUP	Number of GPMICLK cycles that the CE/ADDR signals are active before a strobe is asserted. A value of zero is interpreted as 0. For ATA PIO modes this is known in the ATA7 specification as "Address valid to DIOR-/DIOW- setup"
15–8 DATA_HOLD	Data bus hold time in GPMICLK cycles. Also the time that the data strobe is de-asserted in a cycle. A value of zero is interpreted as 256. For ATA PIO modes this is known in the ATA7 specification as "DIOR-/DIOW- recovery time"
DATA_SETUP	Data bus setup time in GPMICLK cycles. Also the time that the data strobe is asserted in a cycle. This value must be greater than 2 for ATA devices that use IORDY to extend transfer cycles. A value of zero is interpreted as 256. For ATA PIO modes this is known in the ATA7 specification as ""DIOR-/DIOW-"

9.6.6.9 GPMI Timing Register 1 Description (GPMI_TIMING1)

The GPMI timing register 1 specifies the timeouts used when monitoring the NAND READY pin or the ATA IRQ and IOWAIT signals.

GPMI_TIMING1 0x080

Address: 3300_2000h base + 80h offset = 3300_2080h



GPMI_TIMING1 field descriptions

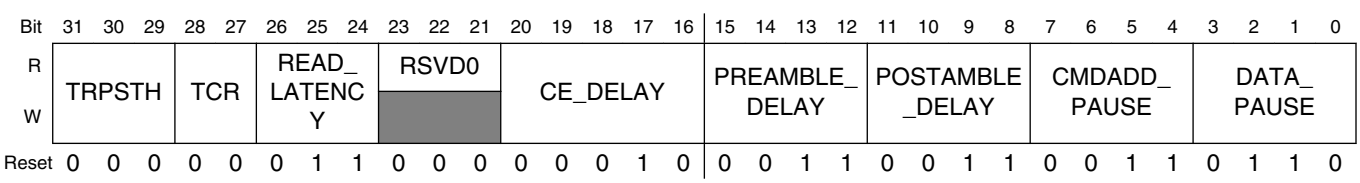
Field	Description
31–16 DEVICE_BUSY_ TIMEOUT	Timeout waiting for NAND Ready/Busy or ATA IRQ. Used in WAIT_FOR_READY mode. This value is the number of GPMI_CLK cycles multiplied by 4096.
RSVD1	Always write zeroes to this bit field.

9.6.6.10 GPMI Timing Register 2 Description (GPMI_TIMING2)

The GPMI timing register 2 specifies the double data rate timing parameters that are used by the cycle state machine to guarantee the various cs delay, pre-amble delay, post-amble delay, command/address delay, data delay, TCR, TRPSTH, and read latency cycle times for the external media type.

GPMI_TIMING2 0x090

Address: 3300_2000h base + 90h offset = 3300_2090h



GPMI_TIMING2 field descriptions

Field	Description
31–29 TRPSTH	Only for Toggle NAND timing control delay TRPSTH GPMICLK cycles for CEn_B high to RE_B high, A value of zero is interpreted as 8
28–27 TCR	Only for Toggle NAND timing control delay (TCR+1) GPMICLK cycles for CEn_B low to RE_B low, 0 is less than or equal to TCR, which is less than the PREAMBLE_DELAY
26–24 READ_ LATENCY	This field is for double data rate read latency configuration. others READ LATENCY is 3 000 READ LATENCY is 0 001 READ LATENCY is 1 010 READ LATENCY is 2 011 READ LATENCY is 3 100 READ LATENCY is 4 101 READ LATENCY is 5

Table continues on the next page...

GPMI_TIMING2 field descriptions (continued)

Field	Description
23–21 RSVD0	Always write zeroes to this bit field.
20–16 CE_DELAY	GPMI dealy from CEn assert to W/Rn changing edge. value of zero is interpreted as 32.
15–12 PREAMBLE_ DELAY	GPMI pre-amble delay in GPMICLK cycles. A value of zero is interpreted as 16.
11–8 POSTAMBLE_ DELAY	GPMI post-amble delay in GPMICLK cycles. A value of zero is interpreted as 16.
7–4 CMDADD_ PAUSE	GPMI delay time from command or addres pause to command or address resume in GPMICLK cycles. A value of zero is interpreted as 16.
DATA_PAUSE	GPMI delay time from data pause to data resume in GPMICLK cycles. A value of zero is interpreted as 16.

9.6.6.11 GPMI DMA Data Transfer Register Description (GPMI_DATA)

The GPMI DMA data transfer register is used by the DMA to read or write data to or from the ATA/NAND control state machine.

GPMI_DATA 0x0A0

Address: 3300_2000h base + A0h offset = 3300_20A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_DATA field descriptions

Field	Description
DATA	In 8-bit mode, one, two, three or four bytes can can be accessed to send the same number of bus cycles.

9.6.6.12 GPMI Status Register Description (GPMI_STAT)

The GPMI control and status register provides a read back path for various operational states of the GPMI controller.

GPMI_STAT 0x0B0

Address: 3300_2000h base + B0h offset = 3300_20B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	READY_BUSY								RDY_TIMEOUT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DEV7_ERROR	DEV6_ERROR	DEV5_ERROR	DEV4_ERROR	DEV3_ERROR	DEV2_ERROR	DEV1_ERROR	DEV0_ERROR	RSVD1			ATA_IRQ	INVALID_BUFFER_MASK	FIFO_EMPTY	FIFO_FULL	PRESENT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

GPMI_STAT field descriptions

Field	Description
31–24 READY_BUSY	Read-only view of NAND Ready_Busy Input pins.
23–16 RDY_TIMEOUT	State of the RDY/BUSY Timeout Flags. When any bit is set to '1' in this field, it indicates that a time out has occurred while waiting for the ready state of the requested NAND device. Multiple bits may be set simultaneously. When GPMI_CTRL1_DECOUPLE_CS = 0, RDY_TIMEOUT[n] is associated with the NAND device on chip_select[n]. When GPMI_CTRL1_DECOUPLE_CS = 1, these flags become associated to a DMA channel instead of a NAND device. For example if DMA channel 6 sends a WAIT_FOR_READY command for NAND Device 2, and a timeout occured on READY_BUSY2, then READY_TIMEOUT[6] will be set instead of READY_TIMEOUT[2].
15 DEV7_ERROR	DMA channel 7 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 7. 1 An Error has occurred on ATA/NAND Device accessed by
14 DEV6_ERROR	DMA channel 6 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 6. 1 An Error has occurred on ATA/NAND Device accessed by
13 DEV5_ERROR	DMA channel 5 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 5. 1 An Error has occurred on ATA/NAND Device accessed by
12 DEV4_ERROR	DMA channel 4 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 4. 1 An Error has occurred on ATA/NAND Device accessed by
11 DEV3_ERROR	DMA channel 3 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 3. 1 An Error has occurred on ATA/NAND Device accessed by
10 DEV2_ERROR	DMA channel 2 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 2. 1 An Error has occurred on ATA/NAND Device accessed by
9 DEV1_ERROR	DMA channel 1 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 1. 1 An Error has occurred on ATA/NAND Device accessed by
8 DEV0_ERROR	DMA channel 0 (Timeout or compare failure, depending on COMMAND_MODE). 0 No error condition present on ATA/NAND Device accessed by DMA channel 0. 1 An Error has occurred on ATA/NAND Device accessed by
7–5 RSVD1	Always write zeroes to this bit field.

Table continues on the next page...

GPMI_STAT field descriptions (continued)

Field	Description
4 ATA_IRQ	Status of the ATA_IRQ input pin.
3 INVALID_ BUFFER_MASK	Buffer Mask Validity bit. 0 ECC Buffer Mask is not invalid. 1 ECC Buffer Mask is invalid.
2 FIFO_EMPTY	NOT_EMPTY = 0x0 FIFO is not empty. EMPTY = 0x1 FIFO is empty. 0 FIFO is not empty. 1 FIFO is empty.
1 FIFO_FULL	NOT_FULL = 0x0 FIFO is not full. FULL = 0x1 FIFO is full. 0 FIFO is not full. 1 FIFO is full.
0 PRESENT	UNAVAILABLE = 0x0 GPMI is not present in this product. AVAILABLE = 0x1 GPMI is present in this product. 0 GPMI is not present in this product. 1 GPMI is present is in this product.

9.6.6.13 GPMI Debug Information Register Description (GPMI_DEBUG)

The GPMI debug information register provides a read back path for diagnostics to determine the current operating state of the GPMI controller.

GPMI_DEBUG 0x0C0

Address: 3300_2000h base + C0h offset = 3300_20C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WAIT_FOR_READY_END								DMA_SENSE								DMAREQ								CMD_END							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_DEBUG field descriptions

Field	Description
31–24 WAIT_FOR_ READY_END	Read Only view of the Wait_For_Ready End toggle signals to DMA. One per channel

Table continues on the next page...

GPMI_DEBUG field descriptions (continued)

Field	Description
23–16 DMA_SENSE	Read-only view of sense state of the 8 DMA channels. A value of "1" in any bit position indicates that a read and compare command failed or a timeout occured for the corresponding channel.
15–8 DMAREQ	Read-only view of DMA request line for 8 DMA channels. A toggle on any bit position indicates a DMA request for the corresponding channel.
CMD_END	Read Only view of the Command End toggle signals to DMA. One per channel

9.6.6.14 GPMI Version Register Description (GPMI_VERSION)

This register reflects the version number for the GPMI.

GPMI_VERSION 0x0D0

Address: 3300_2000h base + D0h offset = 3300_20D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_VERSION field descriptions

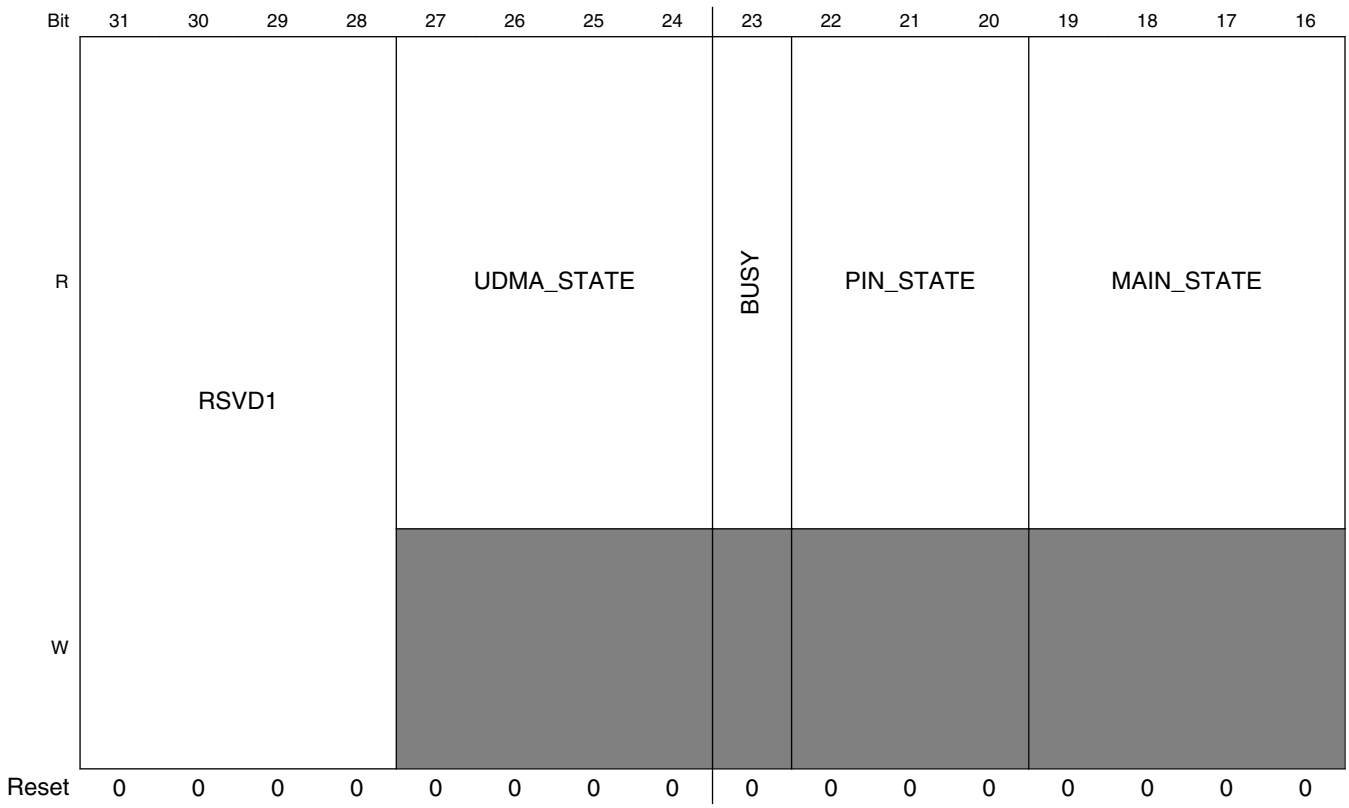
Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
STEP	Fixed read-only value reflecting the stepping of the RTL version.

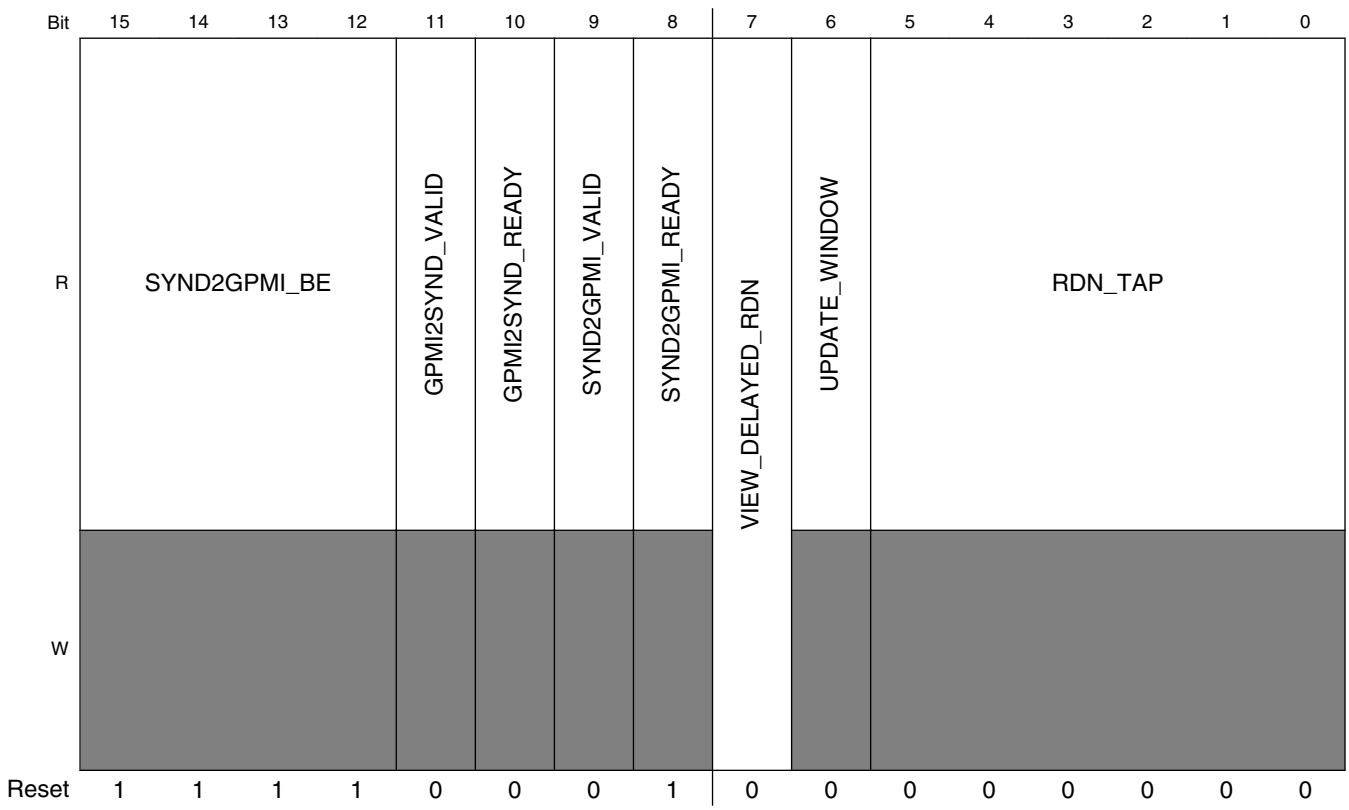
9.6.6.15 GPMI Debug2 Information Register Description (GPMI_DEBUG2)

The GPMI Debug2 information register provides a read back path for diagnostics to determine the current operating state of the GPMI controller.

GPMI_DEBUG2 0x0E0

Address: 3300_2000h base + E0h offset = 3300_20E0h





GPMI_DEBUG2 field descriptions

Field	Description
31–28 RSVD1	Always write zeroes to this bit field.
27–24 UDMA_STATE	USM_IDLE = 4'h0, idle USM_DMARQ = 4'h1, DMA req USM_ACK = 4'h2, DMA ACK USM_FIFO_E = 4'h3, Fifo empty USM_WPAUSE = 4'h4, WR DMA Paused by device USM_TSTRB = 4'h5, Toggle HSTROBE USM_CAPTUR = 4'h6, Capture Stage, (data sampled with DSTROBE is valid) USM_DATOUT = 4'h7, Change Burst DATAOUT USM_CRC = 4'h8, Source CRC to Device USM_WAIT_R = 4'h9, Waiting for DDMARDY- USM_END = 4'ha; Negate DMAACK (end of DMA) USM_WAIT_S = 4'hb, Waiting for DSTROBE USM_RPAUSE = 4'hc, Rd DMA Paused by Host USM_RSTOP = 4'hd, Rd DMA Stopped by Host USM_WTERM = 4'he, Wr DMA Termination State USM_RTERM = 4'hf, Rd DMA Termination state
23 BUSY	When asserted the GPMI is busy. Undefined results may occur if any registers are written when BUSY is asserted. DISABLED = 0x0 The GPMI is not busy. ENABLED = 0x1 The GPMI is busy.
22–20 PIN_STATE	parameter PSM_IDLE = 3'h0, PSM_BYTCNT = 3'h1, PSM_ADDR = 3'h2, PSM_STALL = 3'h3, PSM_STROBE = 3'h4, PSM_ATARDY = 3'h5, PSM_DHOLD = 3'h6, PSM_DONE = 3'h7. PSM_IDLE = 0x0 PSM_BYTCNT = 0x1 PSM_ADDR = 0x2 PSM_STALL = 0x3 PSM_STROBE = 0x4 PSM_ATARDY = 0x5 PSM_DHOLD = 0x6 PSM_DONE = 0x7
19–16 MAIN_STATE	parameter MSM_IDLE = 4'h0, MSM_BYTCNT = 4'h1, MSM_WAITFE = 4'h2, MSM_WAITFR = 4'h3, MSM_DMAREQ = 4'h4, MSM_DMAACK = 4'h5, MSM_WAITFF = 4'h6, MSM_LDFIFO = 4'h7, MSM_LDDMAR = 4'h8, MSM_RDCMP = 4'h9, MSM_DONE = 4'hA. MSM_IDLE = 0x0 MSM_BYTCNT = 0x1 MSM_WAITFE = 0x2 MSM_WAITFR = 0x3

Table continues on the next page...

GPMI_DEBUG2 field descriptions (continued)

Field	Description
	MSM_DMAREQ = 0x4 MSM_DMAACK = 0x5 MSM_WAITFF = 0x6 MSM_LDFIFO = 0x7 MSM_LDDMAR = 0x8 MSM_RDCMP = 0x9 MSM_DONE = 0xA
15–12 SYND2GPMI_BE	Data byte enable Input from BCH.
11 GPMI2SYND_VALID	Data handshake output to BCH.
10 GPMI2SYND_READY	Data handshake output to BCH.
9 SYND2GPMI_VALID	Data handshake Input from BCH.
8 SYND2GPMI_READY	Data handshake Input from BCH.
7 VIEW_DELAYED_RDN	Set to a 1 to select the delayed feedback RE_B to drive the GPMI_ADDR[0] (Nand CLE) pin. For debug purposes, this will allow you see if DLL is functioning properly.
6 UPDATE_WINDOW	A 1 indicates that the DLL is busy generating the required delay.
RDN_TAP	This is the DLL tap calculated by the DLL controller. The selects the amount of delay form the DLL chain.

9.6.6.16 GPMI Debug3 Information Register Description (GPMI_DEBUG3)

The GPMI Debug3 information register provides a read back path for diagnostics to determine the current operating state of the GPMI controller.

GPMI_DEBUG3 0x0F0

Address: 3300_2000h base + F0h offset = 3300_20F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_WORD_CNTR																DEV_WORD_CNTR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_DEBUG3 field descriptions

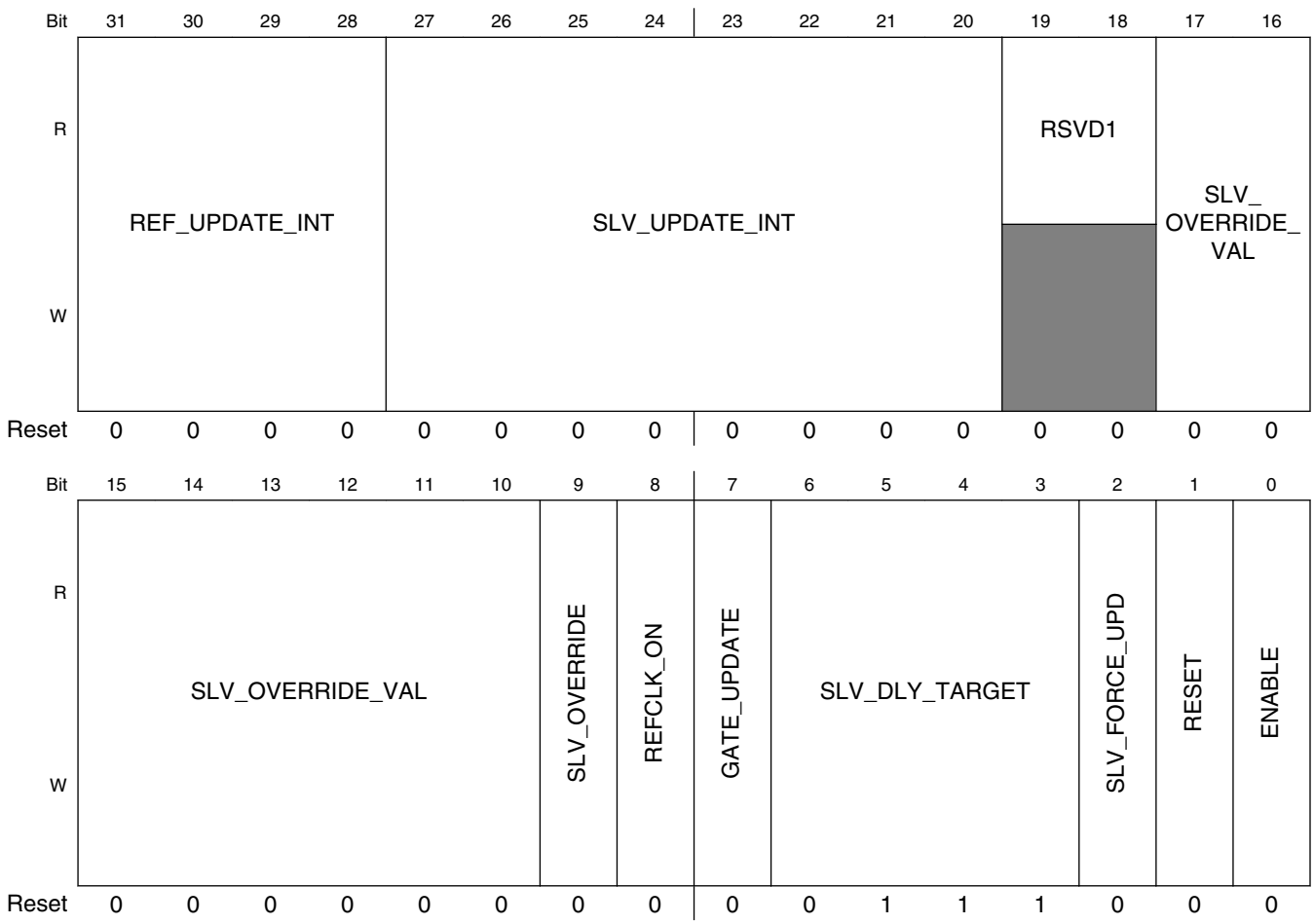
Field	Description
31–16 APB_WORD_ CNTR	Reflects the number of bytes remains to be transferred on the APB bus.
DEV_WORD_ CNTR	Reflects the number of bytes remains to be transferred on the ATA/Nand bus.

9.6.6.17 GPMI Double Rate Read DLL Control Register Description (GPMI_READ_DDR_DLL_CTRL)

GPMI DDR Read Delay Loop Lock Control Register. This register provides programmability in DDR mode for data input timing and data formats.

GPMI_READ_DDR_DLL_CTRL 0x100

Address: 3300_2000h base + 100h offset = 3300_2100h



GPMI_READ_DDR_DLL_CTRL field descriptions

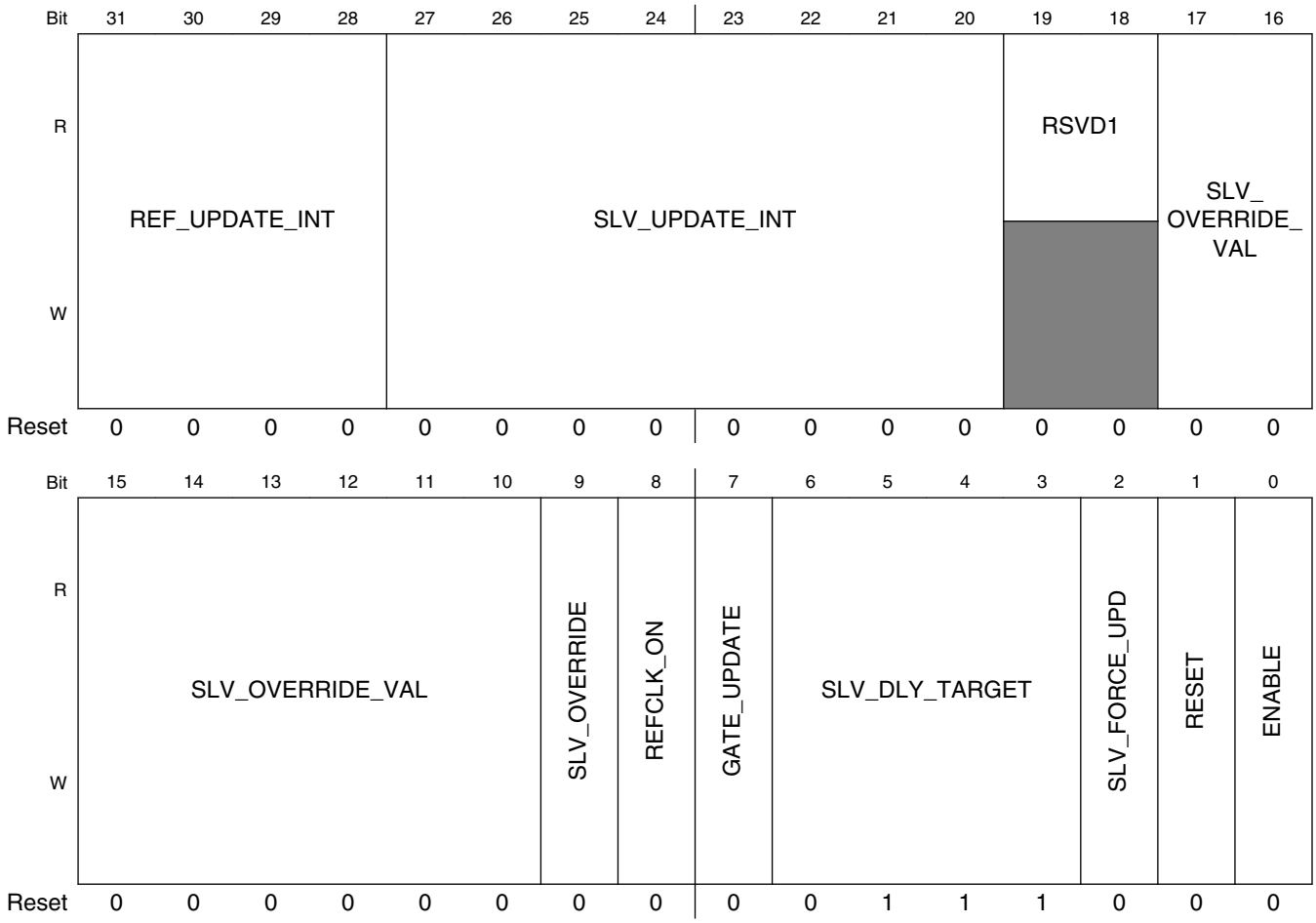
Field	Description
31–28 REF_UPDATE_INT	This field allows the user to add additional delay cycles to the DLL control loop (reference delay line control). By default, the DLL control loop shall update every two GPMICLK cycles. Programming this field results in a DLL control loop update interval of (2 + REF_UPDATE_INT) * GPMICLK. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature)
27–20 SLV_UPDATE_INT	Setting a value greater than 0 in this field, shall over-ride the default slave delay-line update interval of 256 GPMICLK cycles. A value of 0 results in an update interval of 256 GPMICLK cycles (default setting). A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line will also update automatically when the reference DLL transitions to a locked state (from an un-locked state).
19–18 RSVD1	Reserved
17–10 SLV_OVERRIDE_VAL	When SLV_OVERRIDE=1 This field is used to select 1 of 256 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 256.
9 SLV_OVERRIDE	Set this bit to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to tbe enabled using the ENABLE bit. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE=0
8 REFCLK_ON	set this bit to 1 will turn on the reference clock
7 GATE_UPDATE	Setting this bit to 1, forces the slave delay line not update
6–3 SLV_DLY_TARGET	The delay target for the read clock is can be programmed in 1/16th increments of an GPMICLK half-period. So the input read-clock can be delayed relative input data from (GPMICLK/2)/16 to GPMICLK/2.
2 SLV_FORCE_UPD	Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line shall update automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered).
1 RESET	Setting this bit to 1 force a reset on DLL. This will cause the DLL to lose lock and re-calibrate to detect an GPMICLK half period phase shift. This signal is used by the DLL as edge-sensitive, so in order to create a subsequent reset, RESET must be taken low and then asserted again.
0 ENABLE	Set this bit to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled.

9.6.6.18 GPMI Double Rate Write DLL Control Register Description (GPMI_WRITE_DDR_DLL_CTRL)

GPMI DDR Write Delay Loop Lock Control Register. This register provides programmability in DDR mode for data output timing and data formats.

GPMI_WRITE_DDR_DLL_CTRL 0x110

Address: 3300_2000h base + 110h offset = 3300_2110h



GPMI_WRITE_DDR_DLL_CTRL field descriptions

Field	Description
31–28 REF_UPDATE_ INT	This field allows the user to add additional delay cycles to the DLL control loop (reference delay line control). By default, the DLL control loop shall update every two GPMICLK cycles. Programming this field results in a DLL control loop update interval of (2 + REF_UPDATE_INT) * GPMICLK. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature)
27–20 SLV_UPDATE_ INT	Setting a value greater than 0 in this field, shall over-ride the default slave delay-line update interval of 256 GPMICLK cycles. A value of 0 results in an update interval of 256 GPMICLK cycles (default setting). A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line will also update automatically when the reference DLL transitions to a locked state (from an un-locked state).
19–18 RSVD1	Reserved
17–10 SLV_ OVERRIDE_VAL	When SLV_OVERRIDE=1 This field is used to select 1 of 256 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 256.
9 SLV_OVERRIDE	Set this bit to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to tbe enabled using the ENABLE bit. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE=0

Table continues on the next page...

GPMI_WRITE_DDR_DLL_CTRL field descriptions (continued)

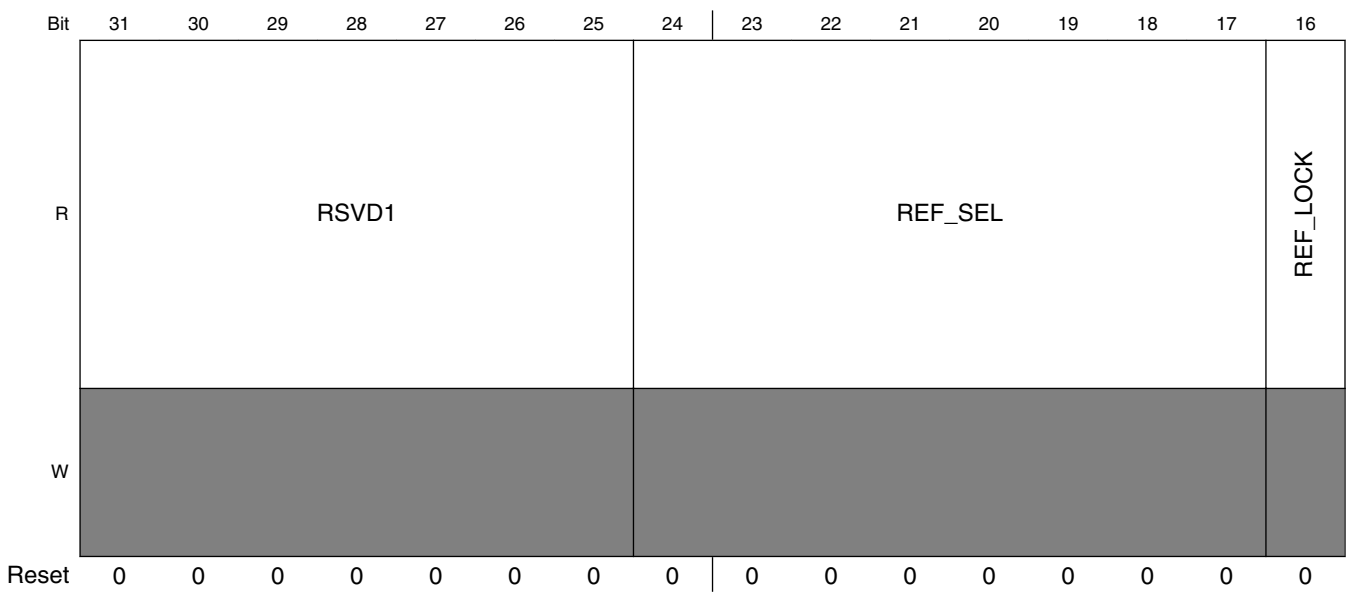
Field	Description
8 REFCLK_ON	set this bit to 1 will turn on the reference clock
7 GATE_UPDATE	Setting this bit to 1, forces the slave delay line not update
6–3 SLV_DLY_ TARGET	The delay target for the read clock can be programmed in 1/16th increments of an GPMICLK half-period. So the input read-clock can be delayed relative input data from (GPMICLK/2)/16 to GPMICLK/2.
2 SLV_FORCE_ UPD	Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line shall update automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered).
1 RESET	Setting this bit to 1 force a reset on DLL. This will cause the DLL to lose lock and re-calibrate to detect an GPMICLK half period phase shift. This signal is used by the DLL as edge-sensitive, so in order to create a subsequent reset, RESET must be taken low and then asserted again.
0 ENABLE	Set this bit to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled.

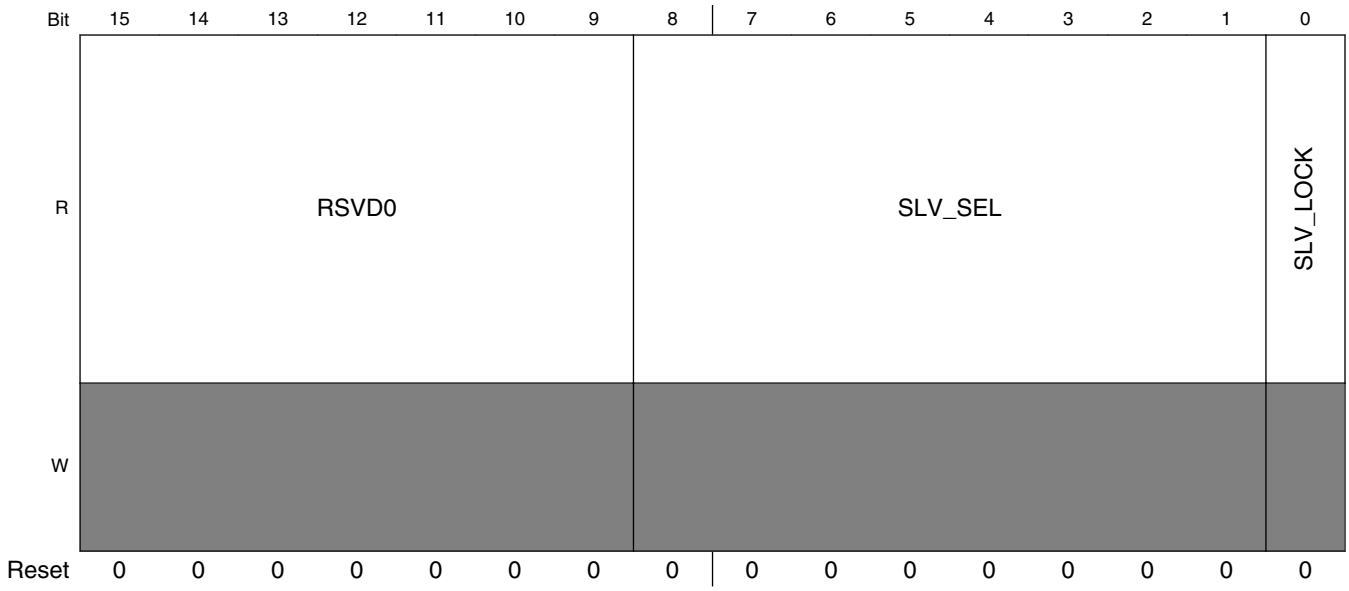
9.6.6.19 GPMI Double Rate Read DLL Status Register Description (GPMI_READ_DDR_DLL_STS)

GPMI Double Rate Read DLL Status Register, Read Only. GPMI DLL status fields are provided in this register.

GPMI_READ_DDR_DLL_STS 0x120

Address: 3300_2000h base + 120h offset = 3300_2120h





GPMI_READ_DDR_DLL_STS field descriptions

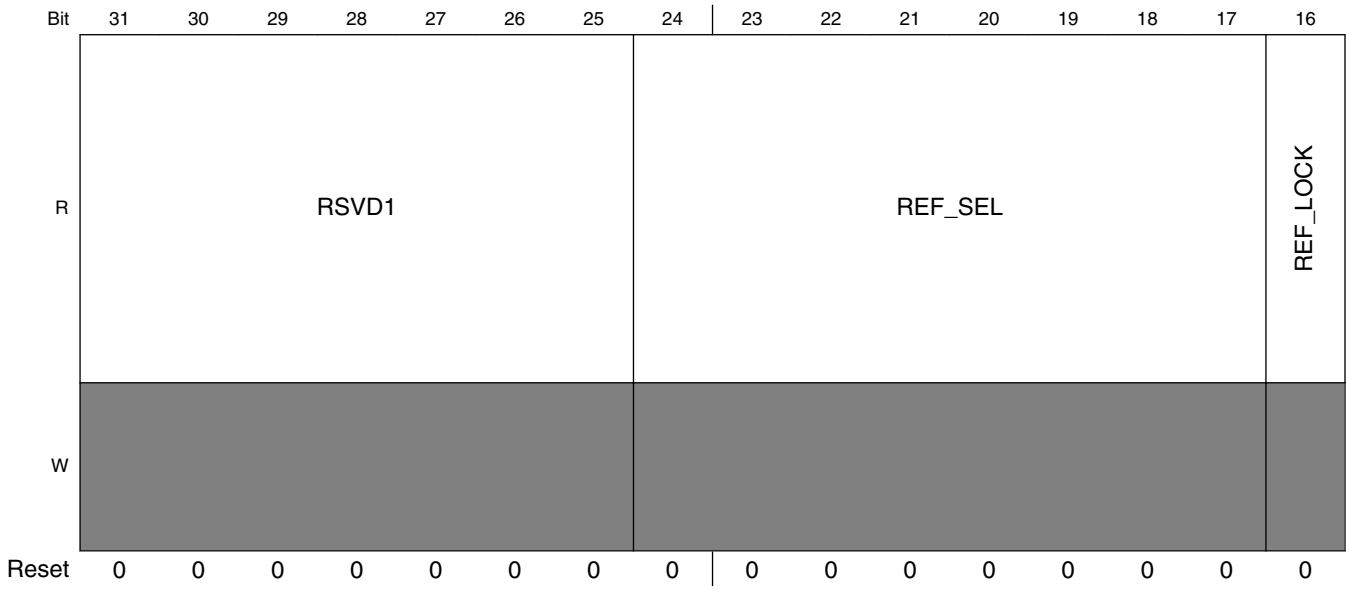
Field	Description
31–25 RSVD1	Reserved
24–17 REF_SEL	Reference delay line select status.
16 REF_LOCK	Reference DLL lock status. This signifies that the DLL has detected and locked to a half-phase GPMICLK shift, allowing the slave delay-line to perform programmed clock delays.
15–9 RSVD0	Reserved
8–1 SLV_SEL	Slave delay line select status
0 SLV_LOCK	Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value.

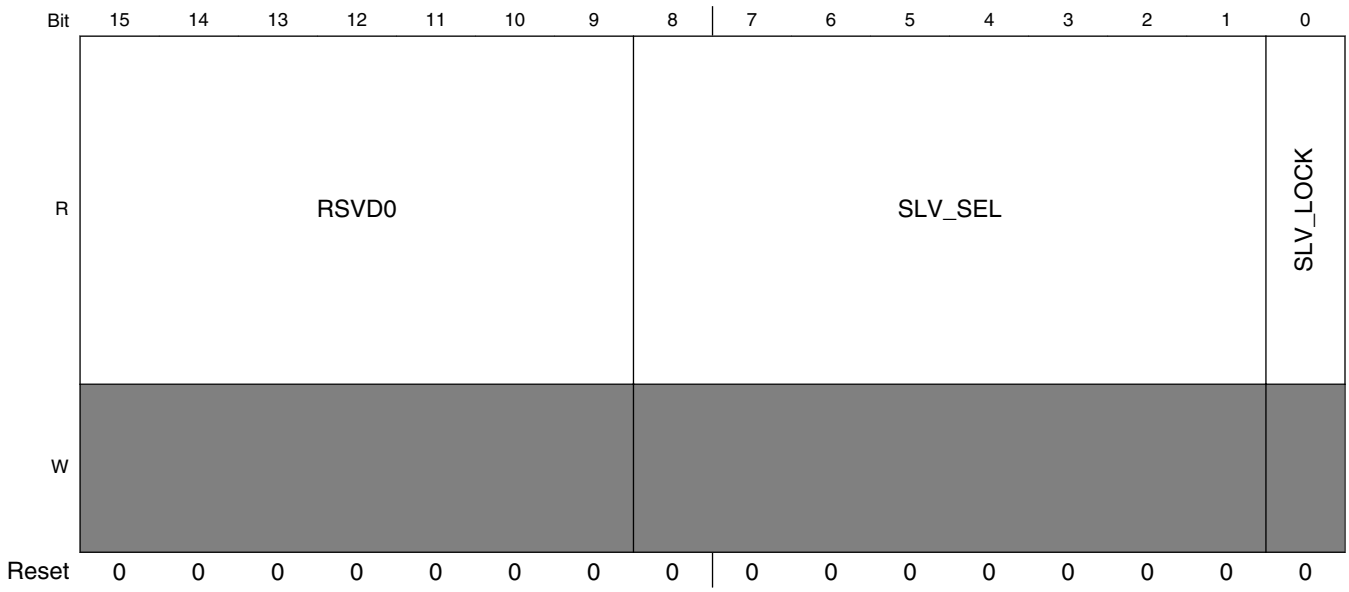
9.6.6.20 GPMI Double Rate Write DLL Status Register Description (GPMI_WRITE_DDR_DLL_STS)

GPMI Double Rate Write DLL Status Register, Read Only. GPMI DLL status fields are provided in this register.

GPMI_WRITE_DDR_DLL_STS 0x130

Address: 3300_2000h base + 130h offset = 3300_2130h





GPMI_WRITE_DDR_DLL_STS field descriptions

Field	Description
31–25 RSVD1	Reserved
24–17 REF_SEL	Reference delay line select status.
16 REF_LOCK	Reference DLL lock status. This signifies that the DLL has detected and locked to a half-phase GPMICLK shift, allowing the slave delay-line to perform programmed clock delays.
15–9 RSVD0	Reserved
8–1 SLV_SEL	Slave delay line select status

Table continues on the next page...

GPMI_WRITE_DDR_DLL_STS field descriptions (continued)

Field	Description
0 SLV_LOCK	Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value.

9.7 External Interface Module (EIM)

9.7.1 Overview

The EIM handles the interface to devices external to the chip, including generation of chip selects, clock and control for external peripherals and memory. It provides asynchronous access to devices with SRAM-like interface and synchronous access to devices with NOR-Flash-like or PSRAM-like interface.

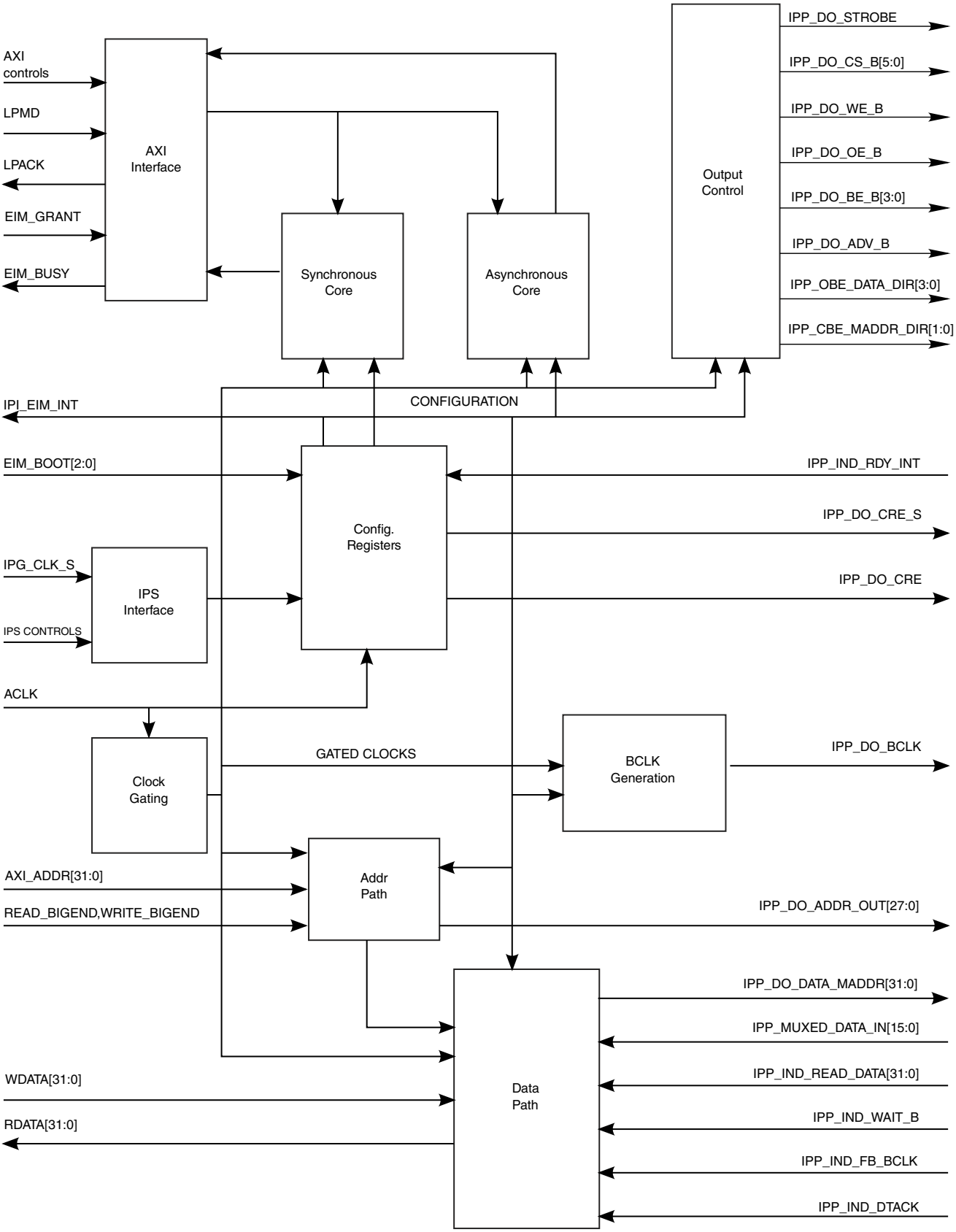


Figure 9-55. EIM Diagram

9.7.1.1 Features

- Six chip selects for external devices
  - Flexible address decoding. Each chip select memory space determined separately, according to VIA port configuration (see Chip Select Memory Map). Configurable Chip Select 0 base address (by VIA)
  - Individual select signal for each one of the memory space defined. Up to 6 memory spaces may be defined and programmed individually.
- Selectable Write Protection for each Chip Select
- Support for multiplexed address / data bus operation x16 port size
- Programmable Data Port Size for each Chip Select (x8, x16)
- Programmable Wait-State generator for each Chip Select, for write and read accesses separately
- Asynchronous accesses with programmable setup and hold times for control signals
- Support for Asynchronous page mode accesses (x16 port size)
- Independent synchronous Memory Burst Read Mode support for NOR-Flash and PSRAM memories (x16 port size)
- Independent synchronous Memory Burst Write Mode support for PSRAM and NOR-Flash like memories (CellularRAM™ from Micron, Infineon, and Cypress, OneNAND™ and utRAM™ from Samsung, and COSMORAM™ from Toshiba)
- Support of NAND-Flash devices with NOR-Flash like interface - MDOC™ (M-Systems), OneNAND™ (Samsung)
- Independent programmable variable/fix Latency support for read and write synchronous (burst) mode
- Support for Big Endian and Little Endian operation modes per access
- ARM AXI slave interface. One ID at a time support.

9.7.1.2 Modes of Operation

The EIM has the following modes of operation:

- Asynchronous Mode
- Asynchronous Page Mode
- Multiplexed Address/Data mode
- Burst Clock Mode
- Low Power Modes
- Boot Mode

See details in the EIM Operational Modes.

9.7.1.2.1 Asynchronous Mode

This is a non-burst mode that is used for SRAM access. In this mode, a single data is read/written with each access (asserted address).

All controls' timings are controlled by preset values in Chip Select Configuration Registers.

9.7.1.2.2 Asynchronous Page Read Mode

Setting the APR bit causes the EIM to perform memory burst accesses by emulating page mode operation.

The external address asserts for each piece of data. The initial access timing is according to RWSC field, and the next address assertions timing is according to PAT field. When APR bit is set, RCSN OEN, RADVN and RBEN fields are ignored for burst access to the external device.

The page size can be set via the BL field to 2, 4, 8, or 16 words (the word size is determined by the DSZ field).

9.7.1.2.3 Multiplexed Address/Data Mode

In this mode, multiplexing addresses and data bits on the same pins is supported for synchronous/asynchronous accesses to x8/x16 data width memory devices.

For more information about the pins that drive data/address in 8/16 non-muxed mode and 16 muxed mode, see the following table.

9.7.1.2.4 Burst Clock Mode

The controller has the ability to support burst synchronous operations in various frequencies, depending on the frequency of the input clock supplied by the system (EIM clock).

The EIM clock can be divided by one, two, three or four, and its frequency can be changed according to the requirements. Variable and fix latency are supported for this mode, according to the external device requirements.

- Synchronous read mode. This is a burst mode, which is used for reading from Flash/ PSRAM memory devices. In this mode, after address assertion a burst of sequential data can be read. Data exchange is carried out according to BCLK being generated

by EIM. An access is delayed according to external WAIT_B signal assertion (signal from the memory device).

- Synchronous write mode. A burst mode used for accessing external devices, which support synchronous write type of access (PSRAM protocol). In this mode, after address assertion a burst of sequential data can be written to the external device. Access may be delayed according to WAIT_B signal assertion (signal from the memory device) before first piece of data arrived to the external device.

NOTE

Maximum frequency of the EIM main clock is 133 Mhz. It may be reduced by the system for special cases of external devices, which demand a different frequency then integer division of the 133 MHz clock.

9.7.1.2.5 Low Power Modes

The input clock is gated by ACT_CS bits. When all the ACT_CS are negated (all CS disable) the internal clock is turned off; awready/wready & arready signal are de-asserted and the master can't access the EIM.

9.7.1.2.6 Boot Mode

It is possible to perform a boot operation from external device located on CS0. The configuration of the relevant bits are done with boot mode signals according to the external device parameters (for example, port size and protocol assertion).

See for more details.

9.7.2 External Signals

The following table describes the external signals of EIM:

Table 9-25. EIM External Signals

Signal	Description	Pad	Mode	Direction
EIM_ACLK_FREERUN	AXI clock signal	LCD1_DATA17	ALT4	I
EIM_AD00	LSB multiplexed Address/Data Bus signal	EPDC1_DATA00	ALT4	IO
EIM_AD01	LSB multiplexed Address/Data Bus signal	EPDC1_DATA01	ALT4	IO
EIM_AD02	LSB multiplexed Address/Data Bus signal	EPDC1_DATA02	ALT4	IO

Table continues on the next page...

Table 9-25. EIM External Signals (continued)

Signal	Description	Pad	Mode	Direction
EIM_AD03	LSB multiplexed Address/Data Bus signal	EPDC1_DATA03	ALT4	IO
EIM_AD04	LSB multiplexed Address/Data Bus signal	EPDC1_DATA04	ALT4	IO
EIM_AD05	LSB multiplexed Address/Data Bus signal	EPDC1_DATA05	ALT4	IO
EIM_AD06	LSB multiplexed Address/Data Bus signal	EPDC1_DATA06	ALT4	IO
EIM_AD07	LSB multiplexed Address/Data Bus signal	EPDC1_DATA07	ALT4	IO
EIM_AD08	LSB multiplexed Address/Data Bus signal	EPDC1_BDR1	ALT4	IO
EIM_AD09	LSB multiplexed Address/Data Bus signal	EPDC1_PWRCOM	ALT4	IO
EIM_AD10	LSB multiplexed Address/Data Bus signal	EPDC1_SDCLK	ALT4	IO
EIM_AD11	LSB multiplexed Address/Data Bus signal	EPDC1_SDLE	ALT4	IO
EIM_AD12	LSB multiplexed Address/Data Bus signal	EPDC1_SDOE	ALT4	IO
EIM_AD13	LSB multiplexed Address/Data Bus signal	EPDC1_SDSHR	ALT4	IO
EIM_AD14	LSB multiplexed Address/Data Bus signal	EPDC1_SDCE0	ALT4	IO
EIM_AD15	LSB multiplexed Address/Data Bus signal	EPDC1_SDCE1	ALT4	IO
EIM_ADDR16	MSB Address Bus signal	EPDC1_SDCE2	ALT4	O
EIM_ADDR17	MSB Address Bus signal	EPDC1_SDCE3	ALT4	O
EIM_ADDR18	MSB Address Bus signal	EPDC1_GDCLK	ALT4	O
EIM_ADDR19	MSB Address Bus signal	EPDC1_GDOE	ALT4	O
EIM_ADDR20	MSB Address Bus signal	EPDC1_GDRL	ALT4	O
EIM_ADDR21	MSB Address Bus signal	EPDC1_GDSP	ALT4	O
EIM_ADDR22	MSB Address Bus signal	EPDC1_BDR0	ALT4	O
EIM_ADDR23	MSB Address Bus signal	LCD1_DATA20	ALT4	O
EIM_ADDR24	MSB Address Bus signal	LCD1_DATA21	ALT4	O
EIM_ADDR25	MSB Address Bus signal	LCD1_DATA22	ALT4	O
EIM_ADDR26	MSB Address Bus signal	LCD1_DATA23	ALT4	O
EIM_BCLK	Burst Clock (BCLK). This active-high output signal is used to clock external burstcapable devices to synchronize the loading and incrementing of addresses and delivery of burst read and write data to/from the EIM. Its behavior is affected by the BCM field in the	EPDC1_DATA11	ALT4	O

Table continues on the next page...

Table 9-25. EIM External Signals (continued)

Signal	Description	Pad	Mode	Direction
	EIM_WCR and the SWR, SRD, BCD, and BCS fields of the EIM_CSxGCR1.			
EIM_CRE	Used as CRE/PS for CellularRam memory. It is used for the Mode Register Set command. This signal can be configured as active low or active high. See CRE and CREP field descriptions of the EIM_CSxGCR1 registers.	LCD1_DATA16	ALT4	O
EIM_CS0	Chip Selects. These signals are active-low. Behavior is affected by the RCSA and RCSN fields of the EIM_CSxRCR1 registers and the WCSA and WCSN fields of the EIM_CSxWCR1registers.	EPDC1_DATA10	ALT4	O
EIM_CS1	Chip Selects. These signals are active-low. Behavior is affected by the RCSA and RCSN fields of the EIM_CSxRCR1 registers and the WCSA and WCSN fields of the EIM_CSxWCR1registers.	EPDC1_DATA15	ALT4	O
EIM_CS2	Chip Selects. These signals are active-low. Behavior is affected by the RCSA and RCSN fields of the EIM_CSxRCR1 registers and the WCSA and WCSN fields of the EIM_CSxWCR1registers.	LCD1_DATA18	ALT4	O
EIM_CS3	Chip Selects. These signals are active-low. Behavior is affected by the RCSA and RCSN fields of the EIM_CSxRCR1 registers and the WCSA and WCSN fields of the EIM_CSxWCR1registers.	LCD1_DATA19	ALT4	O
EIM_DATA00	MSB Data Bus signal	LCD1_DATA00	ALT4	IO
EIM_DATA01	MSB Data Bus signal	LCD1_DATA01	ALT4	IO
EIM_DATA02	MSB Data Bus signal	LCD1_DATA02	ALT4	IO
EIM_DATA03	MSB Data Bus signal	LCD1_DATA03	ALT4	IO
EIM_DATA04	MSB Data Bus signal	LCD1_DATA04	ALT4	IO
EIM_DATA05	MSB Data Bus signal	LCD1_DATA05	ALT4	IO
EIM_DATA06	MSB Data Bus signal	LCD1_DATA06	ALT4	IO
EIM_DATA07	MSB Data Bus signal	LCD1_DATA07	ALT4	IO
EIM_DATA08	MSB Data Bus signal	LCD1_DATA08	ALT4	IO
EIM_DATA09	MSB Data Bus signal	LCD1_DATA09	ALT4	IO
EIM_DATA10	MSB Data Bus signal	LCD1_DATA10	ALT4	IO
EIM_DATA11	MSB Data Bus signal	LCD1_DATA11	ALT4	IO
EIM_DATA12	MSB Data Bus signal	LCD1_DATA12	ALT4	IO
EIM_DATA13	MSB Data Bus signal	LCD1_DATA13	ALT4	IO

Table continues on the next page...

Table 9-25. EIM External Signals (continued)

Signal	Description	Pad	Mode	Direction
EIM_DATA14	MSB Data Bus signal	LCD1_DATA14	ALT4	IO
EIM_DATA15	MSB Data Bus signal	LCD1_DATA15	ALT4	IO
EIM_DTACK_B	Data Acknowledge, asynchronous access. This input is used as a data acknowledge signal for single asynchronous accesses.	LCD1_RESET	ALT4	I
EIM_EB0	Byte Enable. These active-low output signals indicate valid data bytes for the current access. They may be configured to assert for write cycles only. EIM_EB[0] corresponds to DATA_OUT[7:0] For asynchronous write accesses, behavior is affected by the WBEA and WBEN fields of the EIM_CS1WCR1-EIM_CS5WCR1 Registers. On synchronous or asynchronous read accesses, these signals are always asserted at the start of the access and negated at end of the access.	EPDC1_DATA14	ALT4	O
EIM_EB1	Byte Enable. These active-low output signals indicate valid data bytes for the current access. They may be configured to assert for write cycles only. EIM_EB[1] corresponds to DATA_OUT[15:8] For asynchronous write accesses, behavior is affected by the WBEA and WBEN fields of the EIM_CS1WCR1-EIM_CS5WCR1 Registers. On synchronous or asynchronous read accesses, these signals are always asserted at the start of the access and negated at end of the access.	EPDC1_PWRSTAT	ALT4	O
EIM_LBA_B	Address Valid. This active-low output signal is asserted during burst mode accesses to cause the external burst capable device to load a new starting burst address. Assertion of LBA indicates that a valid address is present on the address bus. Its behavior is affected by the SWR, SRD, BCD, and BCS fields of the EIM_CSxGCR1 registers, the RADVA and RADVN fields of the EIM_CSxRCR1 registers, and the WADVA and WADVN fields of the EIM_CSxWCR1 registers. In asynchronous mode, LBA length is affected by the RADVA, WADVA,	EPDC1_DATA12	ALT4	O

Table continues on the next page...

Table 9-25. EIM External Signals (continued)

Signal	Description	Pad	Mode	Direction
	RADVN, and WADVN fields. Minimum length of LBA signal in all modes is one EIM clock cycle.			
EIM_OE	Output Enable. This active-low output signal indicates the bus access is a read and enables external devices to drive the data bus with read data. Its behavior is affected by the OEA and OEN bit fields in the Chip Select Configuration Registers.	EPDC1_DATA08	ALT4	O
EIM_RW	Memory Write Enable. This activelow output signal indicates the bus access is a write and enables external devices to sample the data bus. Its behavior is affected by the WEA and WEN bit fields in the Chip Select Configuration Registers.	EPDC1_DATA09	ALT4	O
EIM_WAIT	Ready/Busy/Wait signal. This activelow input signal is asserted by external burst capable devices which support fixed or variable latency of data. It is serviced in synchronous mode only (EIM_CSxGCR1[SWR, SRD] =1). WAIT will have a pull up resistor in pad. The signal indicates whether the External device is ready for data transaction or not. Busy cycles (or wait cycles) of the external device can occur at the start of a Burst access or at page boundary crossover. NOTE: For burst devices, WAIT output should be configured to change one cycle before data is ready (before delay). NOTE: Some External devices may not use this input signal for ready state indication (fix latency without WAIT signal monitoring). For these devices EIM should be configured accordingly (see RFL, WFL, and PSZ field descriptions). NOTE: This is same as what is shown in IP_IND_WAIT_B	EPDC1_DATA13	ALT4	I

9.7.2.1 Other Important Block I/O Signals Internal to the SoC

The following table provides a description of other signals which are internal to the i.MX 6UltraLite that are important to understand the function of EIM.

Name	I/O	Description
EIM_FB_BCLK	Input	Burst Clock Feedback. This block input is used to sample read data during high transfer speeds. The signal provides feedback from the I/O pad of the BCLK output pin and tends to align more closely with data from the external memory device.
EIM_BOOT	Input	EIM Boot Configuration. These block inputs determine the reset state of DSZ[1:0] and MUM.
ACLK	Input	AXI clock, maximum frequency 133 Mhz
IPG_CLK_S	Input	EIM module IPG clock
RST_B	Input	Active low HW reset
EIM_WARM_RESET	Input	Warm Reset. If this signal is asserted the rst_b will reset only the internal FF and state machine while S/W registers will keep their current state. This signal is active high signal.

9.7.3 Clocks

The following table describes the clock sources for EIM. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 9-26. EIM Clocks

Clock name	Clock Root	Description
aclk	aclk_eim_slow_clk_root	EIM clock (main)
aclk_slow	aclk_eim_slow_clk_root	EIM clock (slow)
ipg_clk_s	ipg_clk_root	Peripheral access clock
aclk_exsc	aclk_eim_slow_clk_root	EIM clock (external device)

- ACLK: EIM clock (main clock, AXI clock) with a Max frequency of 133 MHz. Can be gated externally when there is no active AXI access.
- ACLK_SLOW: EIM all time running ACLK. Used for flip-flops that must be active even when EIM is in low power down mode to provide clock for lpack/lpmd registers, IP registers and IP to AXI sync registers.
- IPG_CLK_S: IPG clock for IP accesses. IP registers are activated by ACLK_SLOW clock.
- ACLK_EXSC: Clock created from EIM clock for External device usage. Integer division by 1, 2, 3 and 4 of the clock can be use with BCD bit field configuration, according to external devices demands. EIM clock frequency may be reduced for lower frequency support which cannot be achieved via BCD bit field.

9.7.4 Chip Select Memory Map

The following four configurations are supported:

- CS0 (128 MB), CS1 (0 MB), CS2 (0 MB), CS3 (0 MB) [default configuration]
- CS0 (64 MB), CS1 (64 MB), CS2 (0 MB), CS3 (0 MB)
- CS0 (64 MB), CS1 (32 MB), CS2 (32 MB), CS3 (0 MB)
- CS0 (32 MB), CS1 (32 MB), CS2 (32 MB), CS3 (32 MB)

9.7.5 Functional Description

This section provides the functional description for the EIM.

9.7.5.1 Bus Sizing Configuration

The EIM supports byte, half word and word operands allowing access to x8, x16 ports. It can be address/data multiplexed in x16 ports. The port size is programmable via the DSZ bit field in the corresponding Chip Select Configuration Register. An 8-bit port can reside in each one of the bytes of the data bus.

In the case of a multi-cycle transfer, the lower two address bits (ADDR[1:0]) are incremented appropriately. The EIM address bus is configured according to DSZ bit field and AUS bits.

The EIM has a data multiplexer which takes the four bytes of the AXI data bus and routes them to their required positions to properly interface to memory.

NOTE

A word access to or from a x16 port requires two external bus cycles to complete the transfer.

A word access to or from a x8 port requires four external bus cycles to complete the transfer.

9.7.5.1.1 8 BIT PORT SUPPORT

EIM has limited support for mot68000 & intel 386 protocols.

9.7.5.1.1.1 MOTOROLA 68000

EIM has limited support for mot68000 protocol. Only basic read or write asynchronous operations are supported.

The following operations are not supported:

- Read modify write

- Sync access
- All special accesses (ARM platform space, bus arbitration, bus control, bus error & reset operations)
- FC outputs

9.7.5.1.1.2 INTEL 386

EIM has limited support for intel 386 protocol. Only basic read or write async non-pipelined operations are supported.

The following operations are not supported:

- Other bus cycles (interrupt, halt & refresh)
- Bus lock
- M/IO, DC, LBA, NA, REFRESH & BS8 signals

9.7.5.2 EIM Operational Modes

Listed here are the main operational modes for EIM selected by control bit fields settings.

For details, see the bit field descriptions of SWR / SRD / MUM. All modes are supported in with 8-, 16- or 32-bit port configuration, according to DSZ bit field.

Table 9-27. EIM Operation Modes Field Settings

Control bit fields			Brief mode description
MUM	SRD	SWR	
0	0	0	Asynchronous write / Asynchronous read for APR=0 / Asynchronous page read for APR=1, none multiplexed
		1	Synchronous write/ Asynchronous read or APR=0 / Asynchronous page read for APR=1,none multiplexed
	1	0	Asynchronous write/Synchronous read none multiplexed
		1	Synchronous write/read none multiplexed
1	0	0	Asynchronous write/read multiplexed
		1	Synchronous write/ Asynchronous read multiplexed
	1	0	Asynchronous write/Synchronous read multiplexed
		1	Synchronous write/read multiplexed

9.7.5.3 Burst Mode (Synchronous) Memory Operation

This mode is enabled for read or write access. Bit SWR sets the burst mode for write operations at the corresponding chip select and bit SRD sets it for read operation.

When this mode is set, the controller attempts to translate the Master burst accesses to memory burst accesses, being limited by the memory burst length, predefined by BL value, or memory and Master WRAP/INCR boundary crossing non-matching. Only the first address accessed is put by the controller on the external address bus in a memory burst sequence.

EIM may translate from some Master sequential accesses to one or several memory bursts, but not from two Master individual accesses to one memory burst.

For the first access in a memory burst sequence, the EIM asserts $\overline{\text{ADV}}$, causing the external burst device to latch the starting burst address; then toggle the burst clock (BCLK) for a predefined number of cycles in order to latch the first unit of data. Subsequent accessed data units can then be burst in fewer clock cycles, realizing an overall increase in bus bandwidth.

NOTE

The BCLK signal toggles only when burst access is executed toward the external device (BCM=1'b0 for normal mode use). It runs with a 50% duty cycle until the end of access is reached. When access is terminated, BCLK stops toggling.

Memory burst accesses are terminated by the EIM whenever it detects the following:

- The specific burst length has executed completely (end of access)
- Write access - missing data in write buffer (Master is delaying the data transfer toward the EIM)
- Next sequential access crosses boundary with unequal condition (wrap/increment, burst length) on the Master and memory
- Current memory burst length reached

9.7.5.4 Burst Clock Divisor (BCD)

In some cases, it may be necessary to slow the external bus in relation to the internal bus to allow accesses to burst devices that have a maximum operating frequency less than the operating frequency of the internal bus.

The internal bus frequency can be divided by one, two, three or four for presentation on the external bus in burst mode operation.

BCLK can only be set to integer divisions of the incoming clock frequency. To get a specific frequency on BCLK, configure the divider to change the incoming EIM clock accordingly.

By programming the BCD bit field to various values, two signals on the external bus are affected; $\overline{ADV}$ and BCLK. The $\overline{ADV}$ signal is asserted according to RADVA or WADVA bit fields programming, and is negated according to the formula mentioned in RADVN and WADVN bit fields description. The BCLK signal runs with a 50% duty cycle until the end of access is reached.

If BCM = 1, the BCLK runs at frequency according to GBCD bit field settings on every async memory access, regardless of the SWR and SRD bits configuration. Caution should be exercised when using BCM bit; GBCD bit field should be updated once and should not change when BCLK is toggling. The BCM bit is used mainly for system debug mode. It has no functional use of the EIM in normal mode.

9.7.5.5 Burst Clock Start (BCS)

In an effort to allow greater flexibility in achieving the minimum number of wait states on burst accesses, you can determine when you want the BCLK to start toggling after the start of access. This allows the BCLK to be skewed from point of data capture on the EIM clock by any number of EIM clock cycles.

Care must be exercised when setting BCS bit field in conjunction with the BCD and RWSC/WWSC bit fields. See the external timing diagrams in Burst (Synchronous Mode) Read Memory Accesses Timing Diagram - BCD=1 and Burst (Synchronous Mode) Read Memory Accesses Timing Diagram - BCD=0 for examples of how to use the BCS, BCD and RWSC/WWSC bit fields together.

9.7.5.6 Multiplexed Address/Data Mode Support

The control bit MUM allows support memory with multiplexed address/data bus both in asynchronous and in synchronous modes.

Caution should be exercised for using OEA/WEA & ADH bit fields. They should be configured according to the external device requirements, as it determines the time point of end of address phase and start of data phase.

9.7.5.7 Mixed Master/Memory Burst Modes Support

To provide mixed sequential/wrap accesses with different length, EIM interprets burst signal and generate additional $\overline{\text{ADV}}$ signals whenever there appear unequal address or burst boundary crossing condition.

BL bit field is used to notify EIM about current memory burst and wrap condition for properly external address generation. In case of non-matching boundaries in both the memory and Master access, EIM starts a new memory burst access by updating address from Master on address bus and generating $\overline{\text{ADV}}$ signal.

9.7.5.8 AXI (Master) Bus Cycles Support

The EIM uses an ARM AXI slave interface. It has a 32-bit bus and supports one access (one ID) at a time. No out of order or parallel accesses are supported.

The following AXI protocol signals are not supported:

- AWLOCK
- AWCACHE
- ARLOCK
- ARCACHE

ARID bus is sampled when:

- new read access is valid on the read address channel and is reflected on the RID bus output toward the master.

AWID bus is sampled when:

- new write access is valid on the write address channel and is reflected on the WID/BID bus output toward the master.

ARPROT and AWPROT signal are partially used. ARPROT[0] and AWPROT[0] bits are used for normal/privileged access detection. ARPROT[2:1] and AWPROT[2:1] are not used.

When sampling a valid access on both of the address channels, the read access will be performed first while write access is pending. After last data transfer completed, the pending write will be executed.

A new access may be executed one cycle after sampling a valid access on the read or write address channels, assuming there is no current access (back to back) which can cause a recovery or end of access penalty cycles, for write access, also assuming data is in write buffer for fast execution.

NOTE

- Only 32-bit word size accesses are supported for burst mode accesses.
- Only 8-bit (1 byte), 16-bit (2 byte) and 32-bit (4 byte) word size supported for single access.
- Maximum number of burst length is 16.
- According to AXI protocol, burst access should not cross 4 KB blocks. In case EIM gets an access that crosses the 4 KB, memory address calculation is invalid.

AXI transfers shown in the table below are also supported. These AXI cycles will be translated into the necessary cycles on the memory side. For example, for optimal operation in case ARM cache is configured to 8 beat burst with wrap, a synchronous flash and cellular RAM memory should be configured in 16 word wrap burst mode when using a 16-bit data port, and in 8 word wrap burst mode when using a 32-bit data port. EIM uses BL bit field to support different memory configurations. The controller splits the transaction when needed in some cases. See Table 9-29.

Table 9-28. AXI Burst Cycles Supported

Burst Length - Number of data transfers	Burst size - Bytes in transfer	Burst type	Description
1	1	INCR	Single transfer
1	2	INCR	Single transfer
1	4	INCR	Single transfer
2	4	WRAP	2-beat wrapping burst
4	4	WRAP	4-beat wrapping burst
8	4	WRAP	8-beat wrapping burst
16	4	WRAP	16-beat wrapping burst
2	4	INCR	2-beat incrementing burst
3	4	INCR	3-beat incrementing burst
4	4	INCR	4-beat incrementing burst
5	4	INCR	5-beat incrementing burst
6	4	INCR	6-beat incrementing burst
7	4	INCR	7-beat incrementing burst
8	4	INCR	8-beat incrementing burst
9	4	INCR	9-beat incrementing burst
10	4	INCR	10-beat incrementing burst
11	4	INCR	11-beat incrementing burst
12	4	INCR	12-beat incrementing burst
13	4	INCR	13-beat incrementing burst
14	4	INCR	14-beat incrementing burst

Table continues on the next page...

Table 9-28. AXI Burst Cycles Supported (continued)

Burst Length - Number of data transfers	Burst size - Bytes in transfer	Burst type	Description
15	4	INCR	15-beat incrementing burst
16	4	INCR	16-beat incrementing burst

Table 9-29. AXI to Memory Burst Splits Number

AXI Burst Type	Memory Burst Type Config.	# of accesses to X8 Memory Port size	# of accesses to X16 Memory Port size	# of accesses to X32 Memory Port size
INC16	WRAP4	16	8	4
Aligned Addr.	Cont.	1	1	1
INC16	WRAP4	17	9	5
Unaligned Addr.	Cont.	1	1	1
WRAP16 Aligned Addr.	WRAP16	4	2	1
	Cont.	1	1	1
WRAP16 Unaligned Addr.	WRAP16	5	3	1
	Cont.	2	2	2
INC8	WRAP8	4	2	1
Aligned Addr.	WRAP16	2	1	1
INC8	WRAP8	4 or 5	2 or 3	2
Unaligned Addr.	WRAP16	2 or 3	2	1 or 2
WRAP8	WRAP16	2	1	1
Aligned Addr.	Cont.	1	1	1
WRAP8 Unaligned Addr.	WRAP16	2 or 3	1	2
	Cont.	2	2	2

9.7.5.9 WAIT_B Signal, RWSC and WWSC bit fields Usage

Most of the external devices supporting burst mode for write or read accesses provide a signal which indicates data is valid on the memory bus (a.k.a. handshake mode). For this mode, RFL and WFL bits should be cleared and RWSC/ WWSC bit fields indicate when the controller should start sampling this signal from the external device or, in other words, how many BCLK cycles should be masked.

For devices which do not use this signal or have a fixed latency ability, the RFL and WFL bits may be set for internal calculation regarding BCLK cycles penalty until data is valid (memory initial access time). For this mode, RWSC/ WWSC indicates when the data is ready for sampling by the controller (read access) or the external device (write

access). There is separation between read and write accesses wait-state control. For read access, RWSC bit field is valid and WWSC bit field is ignored; for write access, WWSC is valid and RWSC is ignored.

9.7.5.10 IPS Register Interface

Access to the registers of the EIM, read or write, is made with IPS protocol signals. The system should avoid changing the registers while master/memory transaction is valid, as this can cause an unknown behavior of the controller.

Register access size is 32-bit as the register size definition, other size of access (byte or half word) is not supported.

9.7.5.11 MRS Set for PSRAM

Memory registers of PSRAM devices can be configured according to external signal, which indicates whether the access is to a memory array or memory register domain.

When the CRE bit is set, the following transactions to the external device will assert the CRE signal. The polarity of this signal is determined by the CREP bit for active low or active high assertion of the signal.

9.7.5.12 EIM Access Termination

EIM is monitoring the corresponding CSx control signal every time variable latency access or dtack access is performed toward the external device.

In variable latency accesses, the Watchdog Timer (WDOG-1) counts BCLK cycles. If it reaches the wdog_limit (according to the WDOG_LIMIT bit field in the WCR) before the device signals can drive/sample new data, the controller will terminate the access and generate an error response transfer toward the Master.

In dtack access, WDOG-1 counts ACLK cycles instead of BCLK and it reaches the wdog_limit before the device asserts the dtack signal, the controller will terminate the access and generate an error response transfer toward the Master.

WDOG-1 can by disabled by WDOG_EN bit in the WCR.

9.7.5.13 Error Conditions

The following conditions cause an error (AXI error or IPS error) response signal:

- AXI errors
  - Access to a disabled chip select - access to a mapped chip select address space where the CSEN bit in the corresponding chip select Configuration Register is clear
  - Access to a non mapped address - access to an address that is not mapped to any CS.
  - User access to a supervisor-protected chip select address space (the SP bit in the corresponding chip select Configuration Register is set)
  - User access in fixed mode access
  - User performs write access to write protected chip select
  - First write data ID and write address ID do not match. (No data is written to the memory.)
  - First Write Data ID and write address ID match but one or more of the other Write data IDs does not match the First Write data ID (data is written to memory according)
  - Access duration to external device from CSx signal assertion is 128/256/512/1024 cycles (access is terminated by the controller) - This error can be disabled be software.
- IPS errors
  - User read or write access to a reserved/non-valid address in the EIM Configuration Register

9.7.5.14 DTACK Mode

In DTACK mode, the EIM uses DTACK signal as an indication of when to end the access.

DTACK is an asynchronous edge/level sensitive signal. DTACK polarity is configurable by the DAP bit in CsxGCR2 (default value is 0).

In this case, EIM begins the access and after a few cycles (according DAPS field) and waits until DTACK (after synchronization) becomes asserted, then samples the data in read access and completes the current data access (see Figure 9-69, Figure 9-70 & Figure 9-71).

If more than one data is needed, CS will be negated between access (CSREC field is not zero) and the AXI burst access will be split into single accesses (see Figure 9-73).

9.7.5.15 EIM_GRANT / EIM_BUSY Handshake Description

Prior to executing command to one of the external device (chip select), EIM assert EIM_BUSY signal (1'b1) and checks the EIM_GRANT signal status.

If EIM_GRANT signal is high, it indicates external data bus is not used by other slaves (NAND Flash Controller) and EIM may start to execute the access. If EIM_GRANT is low, EIM waits until it is set (1'b1) before executing the access.

EIM keeps EIM_BUSY signal set until it completes the access toward the external device.

Once EIM_GRANT signal is set, it can not be reset until EIM_BUSY signal is cleared by EIM.

NOTE

In 16-bit Muxed EIM doesn't use the data bus, therefore there is no sharing of the data bus with NFC. EIM doesn't wait for EIM_GRANT signal from NFC and doesn't assert the EIM_BUSY signal.

9.7.5.16 LPMD / LPACK Handshake Description

These signals are used for frequency and/or voltage change, and for entering low power mode during normal operation of the EIM. Before any change can take place, the controller and all the relevant external devices should be in idle state, which means no access or data transfer is in process.

LPMD input signal is asserted once EIM detects the assertion of LPMD, all ready signals of the AXI channels are negated, and EIM is not sampling new accesses. It finishes all the ongoing accesses and already pending ones. When EIM is in idle state, the LPACK output signal is asserted. EIM will stay in idle state and the LPACK signal will stay asserted until the LPMD signal is negated.

9.7.5.17 Endianness

Big and Little endianness are supported by the controller according to the following table.

Table 9-30. EIM Out/in Data in Case AXI Out/in Data is 0xB3B2B1B0

Endian mode	AXI access	AXI address [1:0]	Port size and used bits								
			Word port				Half word port			Byte port	
			[31:24]	[23:16]	[15:8]	[7:0]	External address [0]	[31:24] ([15:8])	[23:16] ([7:0])	External address [1:0]	[31:24] ([23:16]) ([15:8]) ([7:0])
Big	Word	0	0xB3	0xB2	0xB1	0xB0	0	0xB3	0xB2	0	0xB3
										1	0xB2
							1	0xB1	0xB0	2	0xB1
										3	0xB0
	Half Word	0			0xB1	0xB0	0	0xB3	0xB2	0	0xB3
										1	0xB2
		2	0xB3	0xB2			1	0xB1	0xB0	2	0xB1
										3	0xB0
	Byte	0				0xB0	0		0xB3	0	0xB3
		1			0xB1			0xB2		1	0xB2
		2		0xB2			1		0xB1	2	0xB1
		3	0xB3					0xB0		3	0xB0
Little	Word	0	0xB3	0xB2	0xB1	0xB0	0	0xB1	0xB0	0	0xB0
										1	0xB1
							1	0xB3	0xB2	2	0xB2
										3	0xB3
	Half Word	0			0xB1	0xB0	0	0xB1	0xB0	0	0xB0
										1	0xB1
		2	0xB3	0xB2			1	0xB3	0xB2	2	0xB2
										3	0xB3
	Byte	0				0xB0	0		0xB0	0	0xB0
		1			0xB1			0xB1		1	0xB1
		2		0xB2			1		0xB2	2	0xB2
		3	0xB3					0xB3		3	0xB3

9.7.5.18 Strobe Signal Use

The strobe signal is toggling according to address/data valid condition on the external bus for read and write accesses, and for both synchronous and asynchronous modes.

At any time point when address/data is valid on the external bus, the strobe signal will generate a positive edge, which can be used to sample the external data and control signal.

NOTE

Strobe signal for read data is active (RL + 1) cycles after data on external bus is valid.

9.7.6 Initialization Information

9.7.6.1 Booting from EIM

EIM is ready to work with CS0 after the hardware reset, but it has been configured for very slow access (for boot purposes), with additional setup and hold time.

Other CSs are disabled by hardware reset. Therefore, all CSs must be properly initialized before use in writing values to the corresponding chip select configuration registers.

DSZ[1:0] and MUM fields are set according to EIM_BOOT [2:0] block inputs.

9.7.7 Typical Application

Application note uses following functions to illustrate EIM and memory accesses:

- WR16(address, data) is a 16 bit write access
- WR32(address, data) is a 32 bit write access
- RD16(address, data) is a 16 bit read access
- RD32(address, data) is a 32 bit read access
- WR_I(address, data, delta, counter) is a write data sequence, there data(i+1) = data(i) + delta
- COMMAND_SEQUENCE
- CHECK_STATUS

NOTE

COMMAND_SEQUENCE and CHECK_STATUS are described in AMD Flash Utility, Intel Sibley Flash Utility, MDOC Device Utility, Samsung OneNAND Utility, and Spansion Flash Utility.

All addresses are byte addresses. "CS0" is a Chip Select 0 base address. "EIM_" is a prefix of EIM's registers. 'h is a prefix of hexadecimal constant. "//" is a comment beginning. csba[cs] is a dimension of CS base addresses. "addr" means an address offset in current CS address space. Examples use CS0 address space, but it may apply to any CS except for boot mode functionality.

Configuration examples were verfied with the memory models listed below and may require some adjustments for other family members.

9.7.7.1 Access to Intel Sibley Flash

The following configurations are intended to Sibley family muxed and non-muxed devices.

9.7.7.1.1 Intel Sibley Flash Asynchronous Mode Configuration

- WR32('EIM_CS0GCR1,'h00210081);
- WR32('EIM_CS0RCR1,'h0e020000);
- WR32('EIM_CS0RCR2,'h00000000);
- WR32('EIM_CS0WCR1,'h0704a040);

9.7.7.1.2 Intel Sibley Flash Synchronous Mode Configuration

Configuration used for 133 MHz synchronous access to flash:

```
// Set memory to synchronous read mode
     WR16('CS0+('h5903<<1),'h0060);
     WR16('CS0+('h5903<<1),'h0003);
     WR16('CS0+('h0000<<1),'h00ff);
// Set EIM configuration to synchronous timing
     WR32('EIM_CS0GCR1,'h50214225);                    // 133 MHz
     WR32('EIM_CS0RCR1,'h0c000000);                    // 12 cycles on memory
```

Configuration used for 66 MHz synchronous access to muxed flash:

```
// Set memory to synchronous read mode
     WR16('CS0+('h3103<<1),'h0060);
     WR16('CS0+('h3103<<1),'h0003);
     WR16('CS0+('h0000<<1),'h00ff);
//-------------------------------------------------------------------------
// Set EIM configuration to synchronous timing
     WR32('EIM_CS0GCR1,'h5021122d);                    // 66 MHz
     WR32('EIM_CS0RCR1,'h07000000);                    // 7cycles on memory
```

9.7.7.1.3 Intel Sibley Flash Utility

```
// Single data word programming to addr
     WR16('CS0+addr,'h0060);                 // Unlock
     WR16('CS0+addr,'h00d0);
     WR16('CS0+addr, 'h0041);
     WR16('CS0+addr, data);
     WR16('CS0+caddr, 'h0070);                 // Read Status command
     while('CS0+data[7] == 0)                              // Wait / Polling
         RD16('CS0+addr, data);              // Read status
     RD16('CS0+addr, data);                  // Read status
     WR16('CS0+'h0000,'h00ff);
// Write buffer programming
     WR16('CS0+addr,'h0060);                 // Unlock
     WR16('CS0+addr,'h00d0);
     data    = 0;
```

```
WR16('CS0+addr, 'h0070);                // Read Status command
while(data[7] == 0)                           // Wait
    RD16('CS0+addr, data);              // Read status
WR16('CS0+'h0000,'h00ff);
WR16('CS0+addr,'h00e9);             // Write Buffer command
WR16('CS0+addr,255);                // Word counter (<256)
for(i=0; i<'h200; i = i + 'h40)
   WR_I('CS0+addr+i, data+((i>2)*'h0010_0001), 'h0010_0001, 16); // Data
WR16('CS0+addr,'h00d0);             // Write Confirm command
data  = 0;
while(data[7] == 0)                          // Wait
   RD16('CS0+addr, data);           // Read status
RD16('CS0+addr, data);              // Read status
WR16('CS0+'h0000,'h00ff);
```

9.7.7.2 Access to MDOC Device

The following configurations are intended to MDOC H3 device.

9.7.7.2.1 MDOC Device Boot

To boot from the MDOC device the ERRST bit should be configured to 1, so that EIM will hold the first read access to CS0 until the MDOC asserts the RDY signal.

9.7.7.2.2 MDOC Device Asynchronous Mode Configuration

```
// Non-muxed mode
WR32('EIM_CS0GCR1,'h00410081);
WR32('EIM_CS0RCR1,'h0e121010);
WR32('EIM_CS0RCR2,'h00000000);
WR32('EIM_CS0WCR1,'h12092492);
// Muxed mode
WR32('EIM_CS0GCR1,'h00410081);
WR32('EIM_CS0RCR1,'h0e121010);
WR32('EIM_CS0RCR2,'h00000000);
WR32('EIM_CS0WCR1,'h12092492);
```

9.7.7.2.3 MDOC Device Utility

```
// Read Manufacturer ID and Device ID
    RE16('CS0+'h9400,'h4833);
    RE16('CS0+'h9422,'hb7cc);
```

9.7.7.3 Access to Micron PSRAM

The following configurations are intended to mt45w4mw16bfb_706.

9.7.7.3.1 Micron PSRAM Asynchronous Mode Configuration

```
// 16 bit memory
    WR32('EIM_CS0GCR1,'h403104b1);
    WR32('EIM_CS0RCR1,'h0b010000);
```

```
    WR32('EIM_CS0RCR2,'h00000008);
    WR32('EIM_CS0WCR1,'h0b040040);
// 32 bit memory
    WR32('EIM_CS0GCR1,'h403304b1);
    WR32('EIM_CS0RCR1,'h0f010000);
    WR32('EIM_CS0RCR2,'h00000008);
    WR32('EIM_CS0WCR1,'h0f040040);
```

9.7.7.3.2 Micron PSRAM Synchronous Mode Configuration

```
// 16 bit memory
    WR32('EIM_CS0GCR1,'h403104b1);
    WR32('EIM_CS0WCR1,'h0b040000);
    WR16('CS0+('h85947<<1),'h0040); // memory configuration
    WR32('EIM_CS0GCR1,'h4021_5487);// fixed latency memory wrap 4
    WR32('EIM_CS0RCR1,'h04000000);
    WR32('EIM_CS0RCR2,'h00000008);
    WR32('EIM_CS0WCR1,'h04000000);
// 32 bit memory
    WR32('EIM_CS0GCR1,'h6003_04f1);
    WR32('EIM_CS0WCR1,'h0b04_0000);
    WR32('CS0+('h85947<<2),'h0040); // memory configuration
    WR32('EIM_CS0GCR1,'h4003_1487); // var latency memory inc. page size 128
    WR32('EIM_CS0RCR1,'h04000000);
    WR32('EIM_CS0RCR2,'h00000008);
    WR32('EIM_CS0WCR1,'h04000000);
```

9.7.7.4 Access to Samsung OneNAND

Mentioned below are the configurations intended for Samsung OneNAND muxed and non-muxed devices.

9.7.7.4.1 Samsung OneNAND Boot

There are two ways to boot from Samsung OneNAND. In the first way, the ERRST bit is set to 0 and the user has to poll the interrupt status in the OneNAND interrupt register (or set interrupt handler there). In the second way, the ERRST bit is set to 1 and the user should enable the device interrupt output before the first read from CS0 access is issued.

Load sectors 2,3 to DataRAM, page 0 done in the next example:

- WR16('CS0+('hF241<<1),'h0); // Clear interrupt status
- WR16('CS0+('hF100<<1),'h0); // block[8:0] address
- WR16('CS0+('hF107<<1),'h2); // sector[1:0] and page[7:2] addresses
- WR16('CS0+('hF200<<1),'h802); // buffer[11:8] address and counter[1:0]
- WR16('CS0+('hF101<<1),'h0); // DDP choose
- WR16('CS0+('hF220<<1),'h0); // Set command

9.7.7.4.2 Samsung OneNAND Asynchronous Mode Configuration

```
// Non-muxed memory
    WR32('EIM_CS0GCR1,'h00410081);
```

```
    WR32('EIM_CS0RCR1,'h0b010000);
    WR32('EIM_CS0RCR2,'h00000000);
    WR32('EIM_CS0WCR1,'h0c092480);
// Muxed memory
    WR32('EIM_CS0GCR1,'h00410089);
    WR32('EIM_CS0RCR1,'h0b010000);
    WR32('EIM_CS0RCR2,'h00000000);
    WR32('EIM_CS0WCR1,'h0c092480);
```

9.7.7.4.3 Samsung OneNAND Synchronous Mode Configuration

Set memory and EIM to synchronous read mode is shown in the next example:

```
    WR16('CS0+('hF221<<1),'hc0e0); // Synchronous read, 4 clk latency
    WR32('EIM_CS0GCR1,'h50412405); // 44 MHz (non-muxed)
    WR32('EIM_CS0RCR1,'h05010000);
```

The muxed Samsung OneNAND supports synchronous write, too:

```
// Set memory & EIM to synchronous read and write mode
    WR16('CS0+('hF221<<1),'hc0f2); // Sync. read and write, 4 clk latency
    WR32('EIM_CS0GCR1,'h5041240f); // 44 MHz
    WR32('EIM_CS0RCR1,'h05010000);
    WR32('EIM_CS0WCR1,'h05040000);
```

9.7.7.4.4 Samsung OneNAND Utility

The following utility algorithms are used on the Samsung OneNAND:

```
// Unlock Block command
    WR16('CS0+('hF100<<1),'h0);        // DFS
    WR16('CS0+('hF100<<1),'h0);        // DBS
    WR16('CS0+('hF24c<<1),'h2);        // SBA - block number (2)
    WR16('CS0+('hF241<<1),'h0);        // Clear interrupt status
    WR16('CS0+('hF220<<1),'h23);       // Unlock command
    data ='h0;
    while(!(data &'h0004))             // Polling
       RD32('WIAR, data); // Read status
// Erase block command
    WR16('CS0+('hF100<<1),'h2);        // DFS and block ([8:0]) address
    WR16('CS0+('hF101<<1),'h0);        // DBS
    WR16('CS0+('hF241<<1),'h0);        // Clear interrupt status
            WR16('CS0+('hF220<<1),'h94);           // Erase command
    data ='h0;
    while(!(data &'h0004))                  // Wait
        RD32('WIAR, data); // Read status
// Program page command
    WR16('CS0+('hF100<<1),'h2);             // DFS and block[8:0] address
    WR16('CS0+('hF107<<1),'h0);             // sector[1:0] and page[7:2] addresses
    WR16('CS0+('hF200<<1),'h800);           // buffer[11:8] address and counter[1:0]
    WR16('CS0+('hF241<<1),'h0);             // Clear interrupt status
    WR16('CS0+('hF220<<1),'h80);            // Program command
    data ='h0;
    while(!(data &'h0004))                  // Wait
        RD32('WIAR, data); // Read status
```

9.7.7.5 Access to Samsung UtRAM

Below mentioned configurations are intended for Samsung UtRAM.

9.7.7.5.1 Samsung UtRAM Asynchronous Mode Configuration

```
WR32('EIM_CS0GCR1,'h400104b1);
WR32('EIM_CS0RCR1,'h0a010000);
WR32('EIM_CS0RCR2,'h00000008);
WR32('EIM_CS0WCR1,'h0b040040);
```

9.7.7.5.2 Samsung UtRAM Synchronous Mode Configuration

```
RD16('CS0+('hff_ffff<<1),data);             // command sequence
RD16('CS0+('hff_ffff<<1),data);
RD16('CS0+('hff_ffff<<1),data);
RD16('CS0+('hff_feff<<1),data);
RD16('CS0+('h00_82a0<<1),data);             // memory sync. configuration
WR32('EIM_CS0GCR1,'h4021_53b7);            // fixed latency memory wrap 32
WR32('EIM_CS0RCR1,'h0500_0000);
WR32('EIM_CS0WCR1,'h0300_0000);
```

9.7.7.6 Access to Spansion Flash

Below mentioned configurations are intended for Spansion Flash.

9.7.7.6.1 Spansion Flash Asynchronous Mode Configuration

```
          WR32('EIM_CS0GCR1,'h00410081);
          WR32('EIM_CS0RCR1,'h0a018000);
          WR32('EIM_CS0RCR2,'h00000000);
          WR32('EIM_CS0WCR1,'h0704a240);
WR16('CS0+('hF220<<1),'h94);            // Erase command
          data ='h0;
          while(!(data &'h0004))                    // Wait
              RD32('WIAR, data); // Read status
     // Program page command
          WR16('CS0+('hF100<<1),'h2);               // DFS and block[8:0] address
          WR16('CS0+('hF107<<1),'h0);               // sector[1:0] and page[7:2] addresses
          WR16('CS0+('hF200<<1),'h800);             // buffer[11:8] address and counter[1:0]
          WR16('CS0+('hF241<<1),'h0);               // Clear interrupt status
          WR16('CS0+('hF220<<1),'h80);              // Program command
          data ='h0;
          while(!(data &'h0004))                    // Wait
              RD32('WIAR, data); // Read status
```

9.7.7.6.2 Spansion Flash Synchronous Mode Configuration

```
          WR16('CS0+('h0555<<1),'h00aa);   // command sequence
          WR16('CS0+('h02aa<<1),'h0055);
          WR16('CS0+('h0555<<1),'hd0);
          WR16('CS0+('h0000<<1),'h1ec4);   // memory sync. configuration
          WR32('EIM_CS0GCR1,'h50411325); // 66 MHz
          WR32('EIM_CS0RCR1,'h05000000); // 5 cycles on memory
```

9.7.7.6.3 Spansion Flash Utility

```
      // Single word programming
          COMMAND_SEQUENCE(cs,16,'ha0);           // single word programming
```

```
            WR16('CS0+addr, data);
            CHECK_STATUS('CS0+addr,data,16,1,errst);
        // Write buffer programming
            COMMAND_SEQUENCE(0,16,'h25);              // write buffer programming
            WR16('CS0+addr,'h001f);                     // counter-1
            WR_I('CS0+addr, data, 'h0010_0001, 16); // data
            WR16('CS0+addr,'h0029);                     // write buffer to flash
            CHECK_STATUS('CS0+addr+'h3e,data[31:16]+'h00f0,16,1,errst);
```

There COMMAND_SEQUENCE and CHECK_STATUS are next functions:

```
        task COMMAND_SEQUENCE;
            input[2:0]    cs;
            input[7:0]    port_size;
            input[31:0]   code;
        begin
            if(port_size == 16)
               begin
               WR16(csba[cs]+('h0555<<1),'h00aa);
               WR16(csba[cs]+('h02aa<<1),'h0055);
               WR16(csba[cs]+('h0555<<1),code);
               end
            else
               begin
               WR32(csba[cs]+('h0555<<2),'h00aa);
               WR32(csba[cs]+('h02aa<<2),'h0055);
               WR32(csba[cs]+('h0555<<2),code);
               end
        end
        endtask
        task     CHECK_STATUS;
            input[31:0] addr;
            input[31:0] edata;
            input[7:0]   port_size;
            input[7:0]   opcode;
            output[7:0] errst;
            reg[31:0]    data;
            reg[31:0]    data3;
        begin
            errst = 0;
            data  = 0;
            data3 = 0;
while(!(data == edata) && !errst) // Wait operation
          begin: BR_EN
          RD16(addr, data);             // Read status
          if(data[7] !== edata[7])
              begin
              if(data[5] == 1)
                   begin
                   RD16(addr, data3);
                   RD16(addr, data);
                   if(data[6] !== data3[6])
                        begin
                        $display("CHECK_STATUS: Error timout on single data program");
                        errst = 1;
                        disable BR_EN;
                        end
                   end
              else
                   begin
                   if(opcode == 2)
                        if(data[1] == 1)
                            begin
                            RD16(addr, data3);
                            if(port_size == 32)
                               RD32(addr, data);
                            else
                               RD16(addr, data);
                            if(data[1] == 1 && data !== edata)
```

```
                                begin
                                $display("CHECK_STATUS: Error on write buffer");
                                errst =3;
                                disable BR_EN;
                                end
                            end
                    end
                end
            else
                begin
                RD16(addr, data3);
                if(port_size == 32)
                    RD32(addr, data);
                else
                    begin
                    RD16(addr, data);
                    edata[31:16] = 16'h0;
                    end
                if(data !== edata)
                    begin
                    $display("CHECK_STATUS: Error in data write on single data program");
                    errst =2;
                    disable BR_EN;
                    end
                end
            end
end
endtask
```

9.7.7.7 8 bit support

This section details the pin connections for Intel mode and Motorola mode.

Intel Mode - For intel mode use the following connection:

Table 9-31. Intel Mode pin connections

ARM platform Pin	EIM Pin	Notes
ADS#	IPP_DO_ADV_B	WAL = 1,RAL = 1
W/R	IPP_DO_BE_B	WBED = 1
WR#	WE#	
RD#	OE#	

Mot. Mode - For intel mode use the following connection:

Table 9-32. Motorola Mode pin connections

ARM platform Pin	EIM Pin	Notes
AS#	IPP_DO_CS_B	
R/W#	WE#	
LDS#	BE#	

9.7.8 External Bus Timing Diagrams

The following timing diagrams show the timing of accesses to memory or a peripheral with different timing parameters. All examples done for CS0, but are valid for any others chip select. BE means one from current used BE[3:0].

9.7.8.1 Asynchronous Read Memory Accesses Timing Diagram

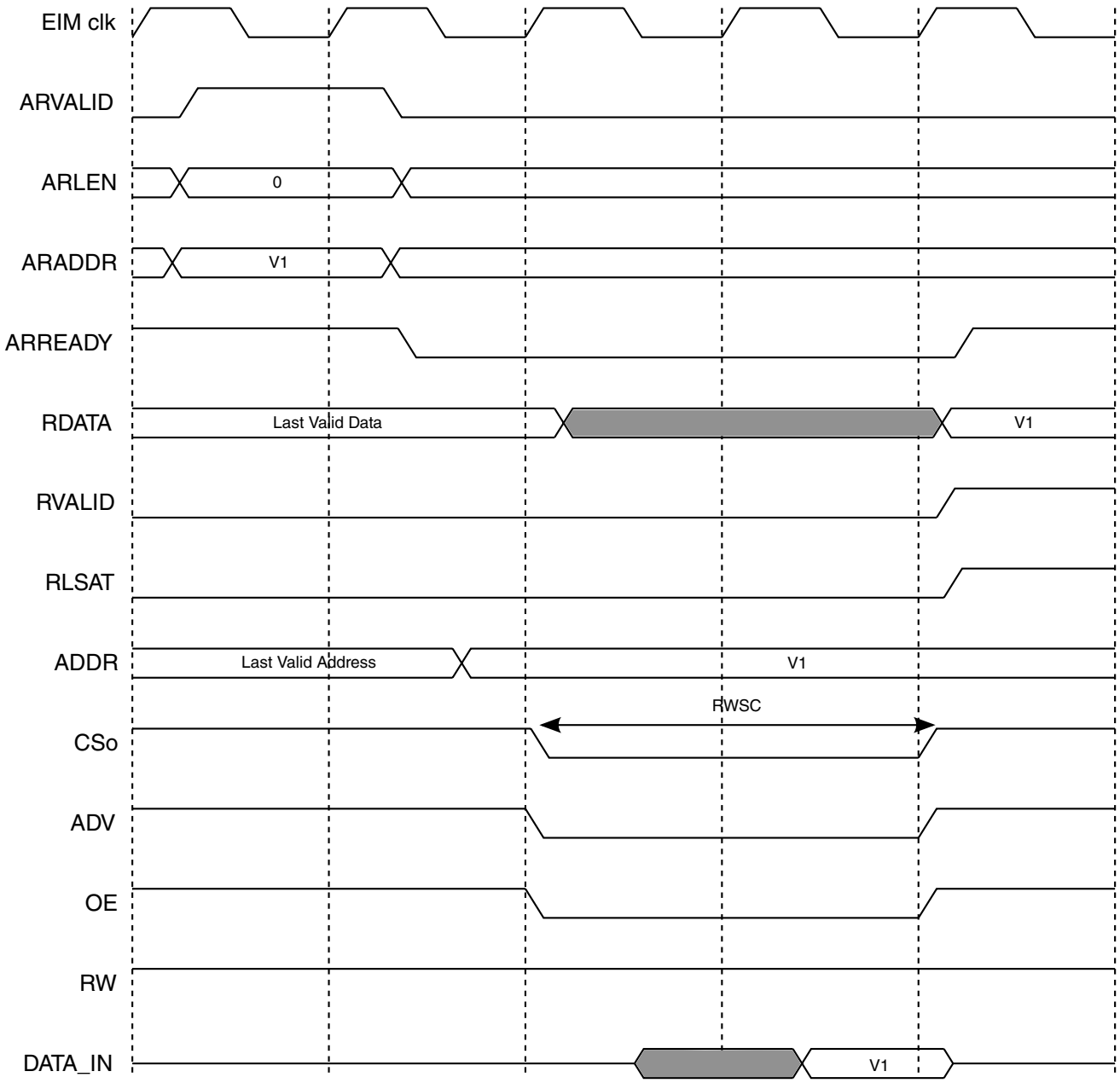


Figure 9-56. Read Access, RWSC=2,RCSA=0,OEA=0,RCSN=0,OEN=0, RAL=1

9.7.8.2 Asynchronous Write Memory Accesses Timing Diagram

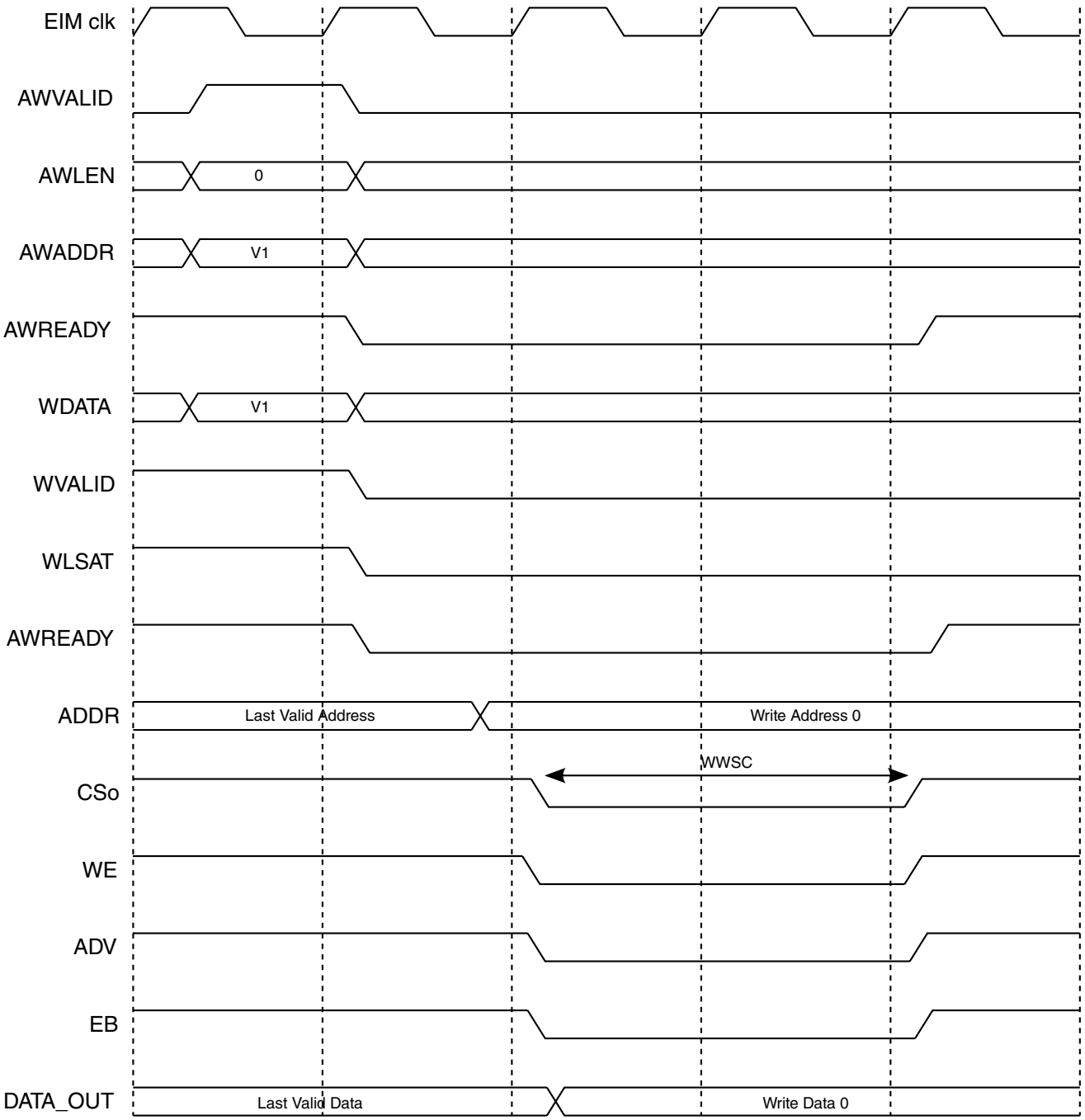


Figure 9-57. Write Access, WWSC=2,WCSA=0,WEA=0,WCSN=0,WEN=0,BEA=0, BEN=0, WAL=1

9.7.8.3 Asynchronous Read/Write Memory Accesses Timing Diagram

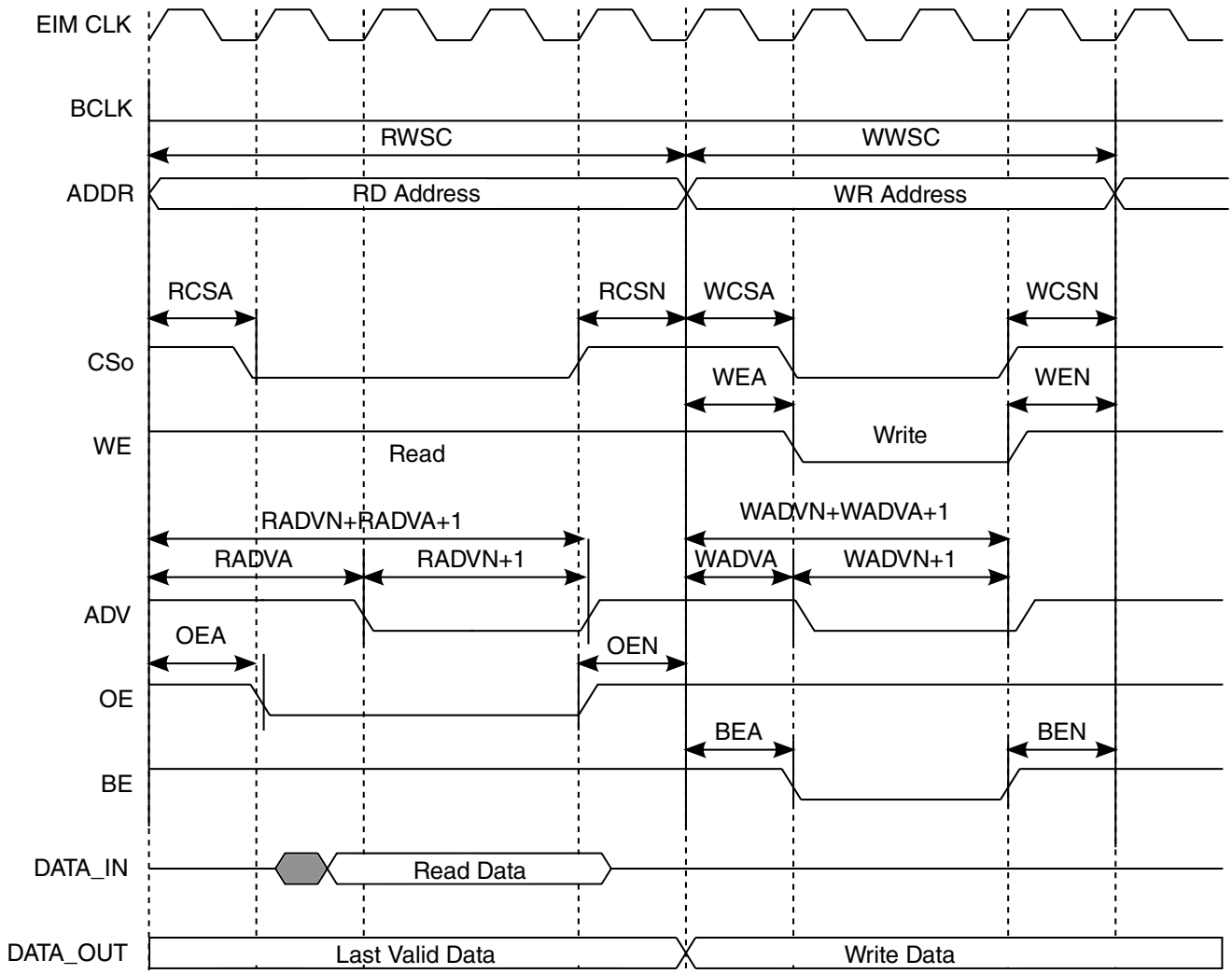


Figure 9-58.
RCSA=1,RADVA=2,OEA=1,RADVN=1,RCSN=1,OEN=1,WCSA=1,WEA=1,WADVA=1,BEA=1,WADVN=1,WCSN=1,WEN=1,BEN=1

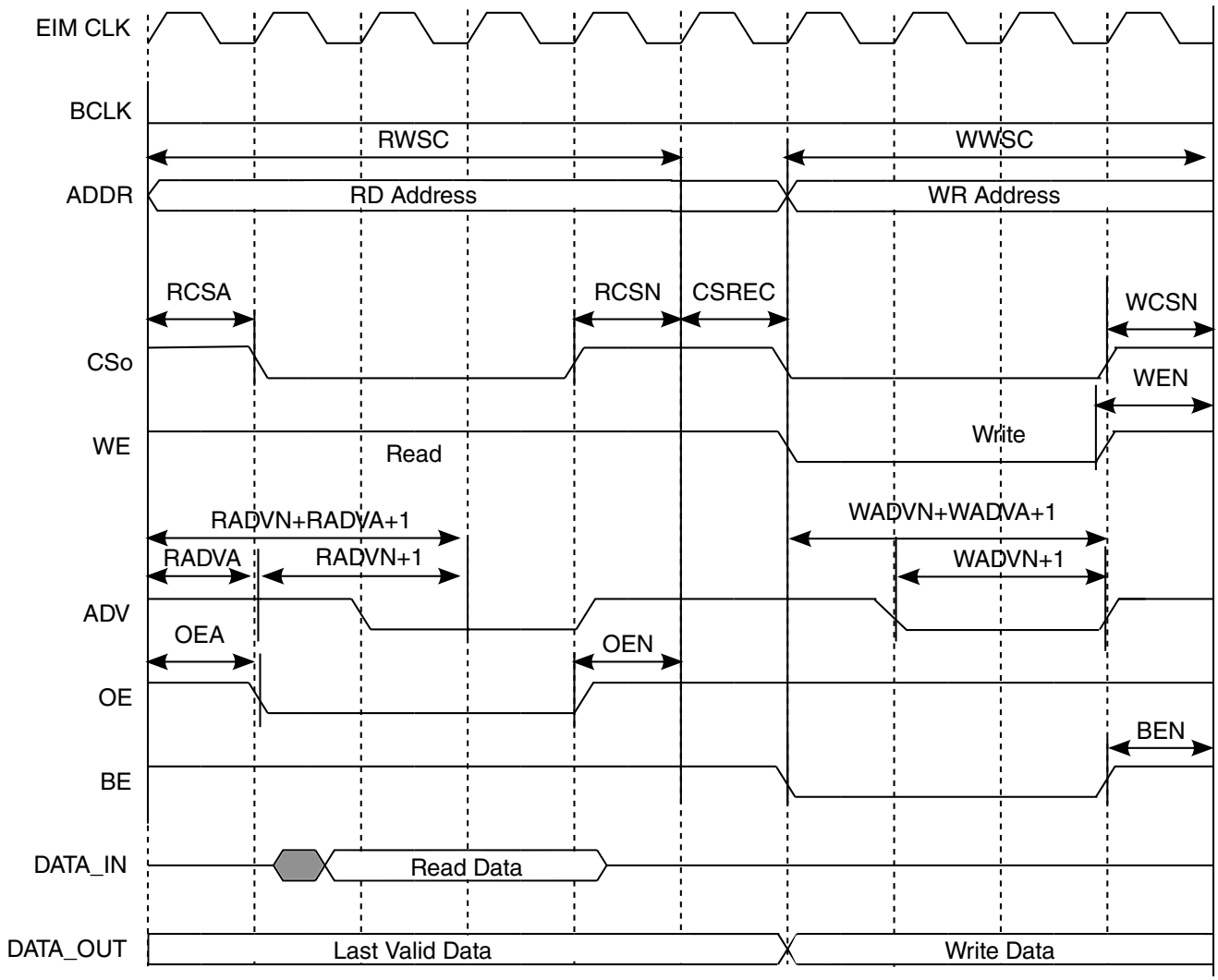


Figure 9-59.
RWSC=5,RCSA=1,RCSN=1,RADVA=1,RADVN=1,OEA=1,OEN=1,WWSC=4,WCSA=0,WCSN=1,WEA=0,WEN=1,WADVA=1,WADVN=1,BEA=0,BEN=1,CSREC=1

9.7.8.4 Asynchronous Read/Write Using RAL, WAL and CSREC

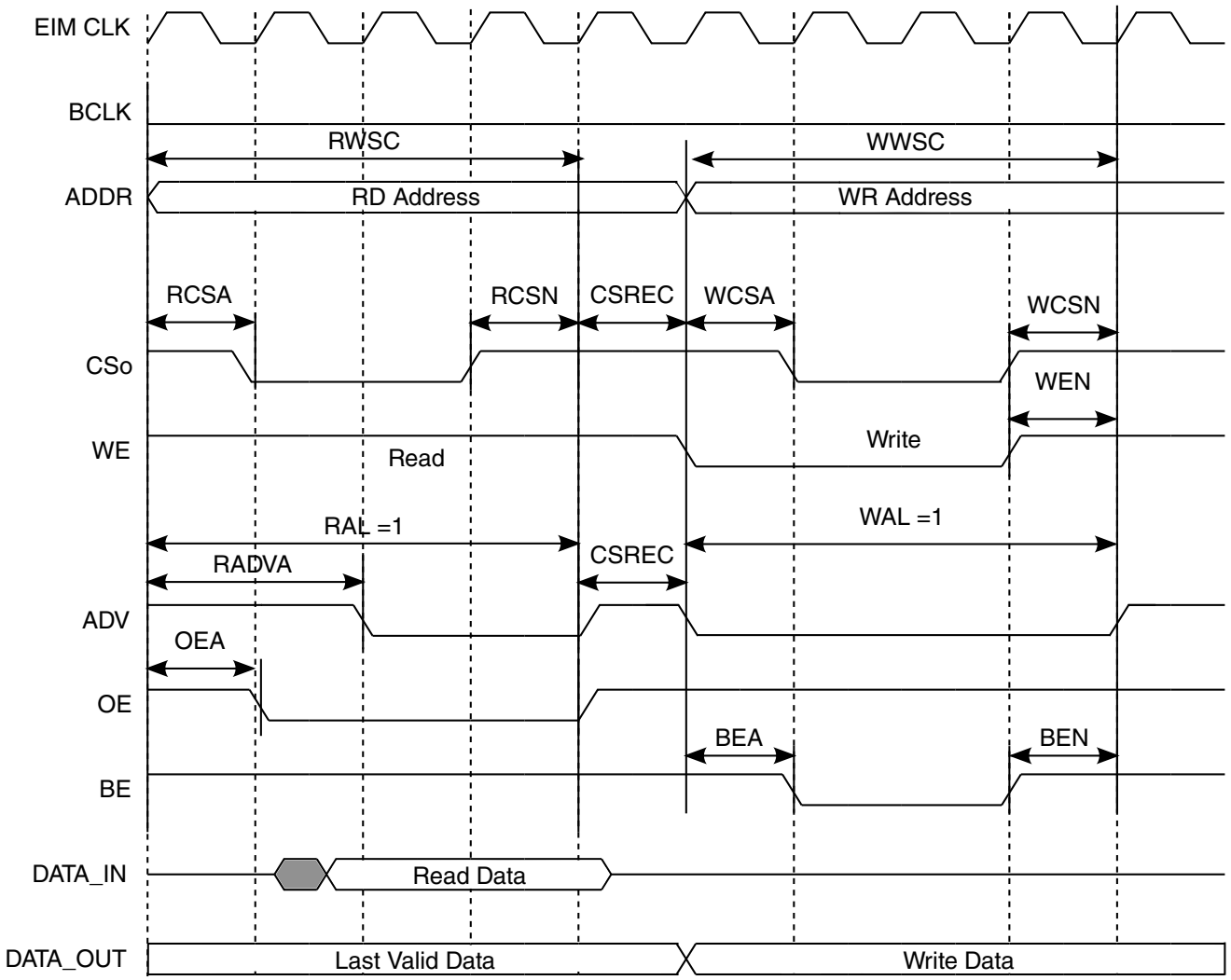


Figure 9-60.
RAL=1,RCSN=1,RADVA=2,OEA=1,RCSN=1,CSREC=1,WCSA=1,WEA=0,WADVA=0,BEA=1,WAL=1,WCSN=1,WEN=1,BEN=1

9.7.8.5 Consecutive Asynchronous Write Memory Accesses Timing Diagram

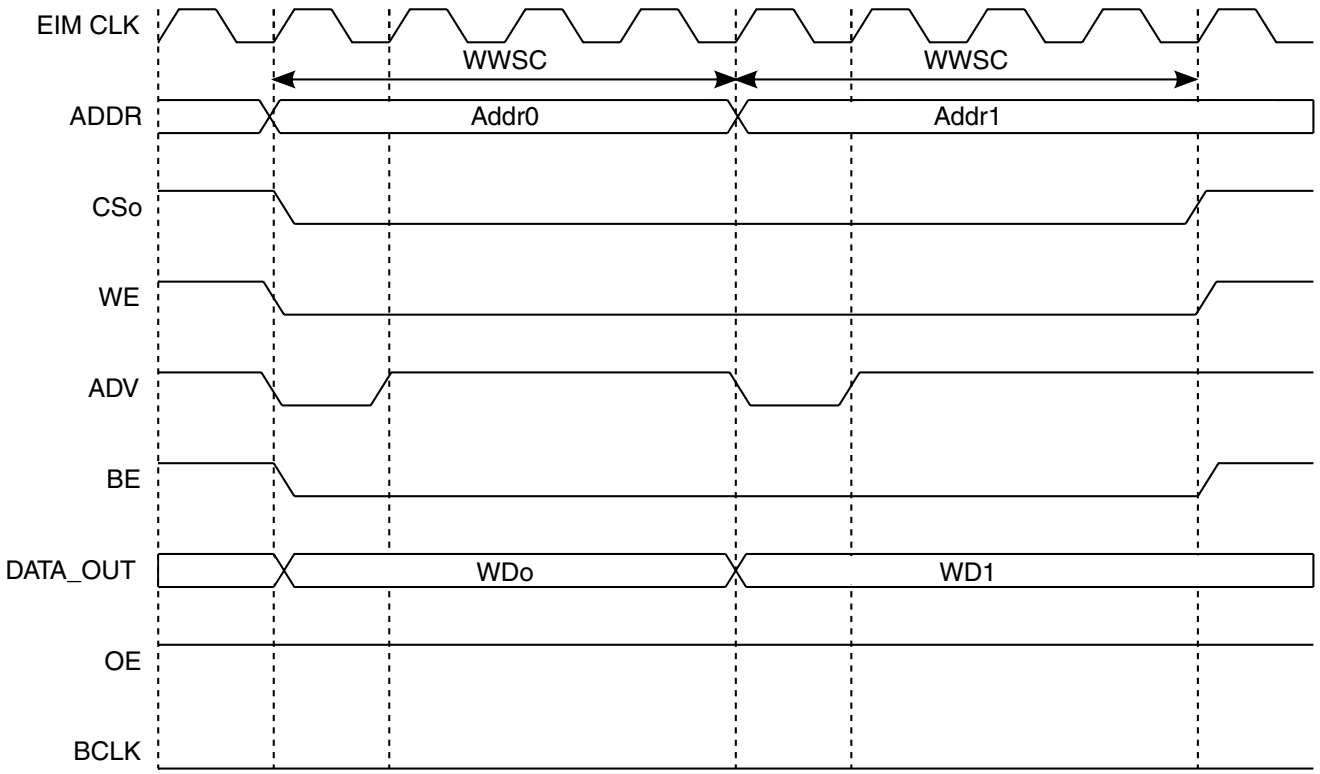


Figure 9-61. WWSC=4,WCSA=0,WEA=0,WADVA=0,BEA=0,WCSN=0,WEN=0,WADVN=0,BEN=0,CSREC=0

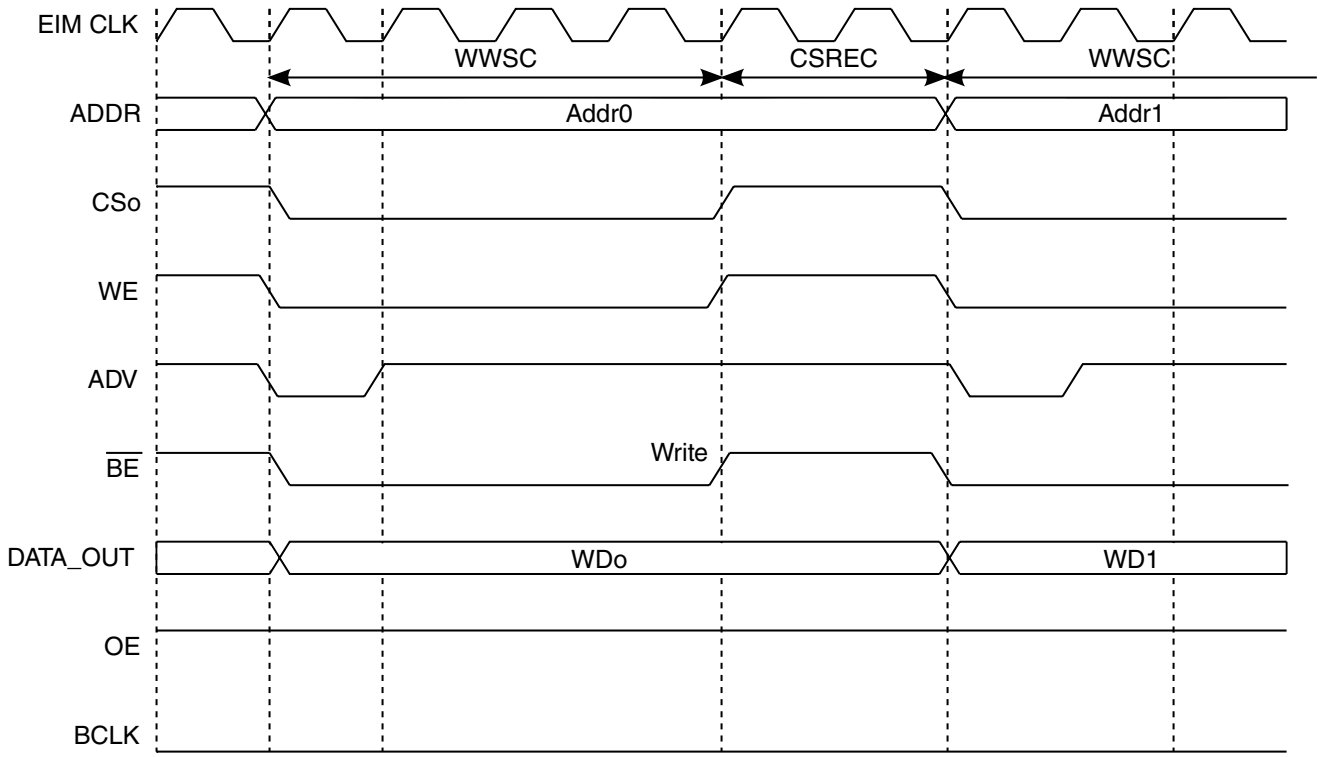


Figure 9-62. WWSC=4,WCSA=0,WEA=0,WADVA=0,BEA=0,WCSN=0,WEN=0,WADVN=0,BEN=0,CSREC=2

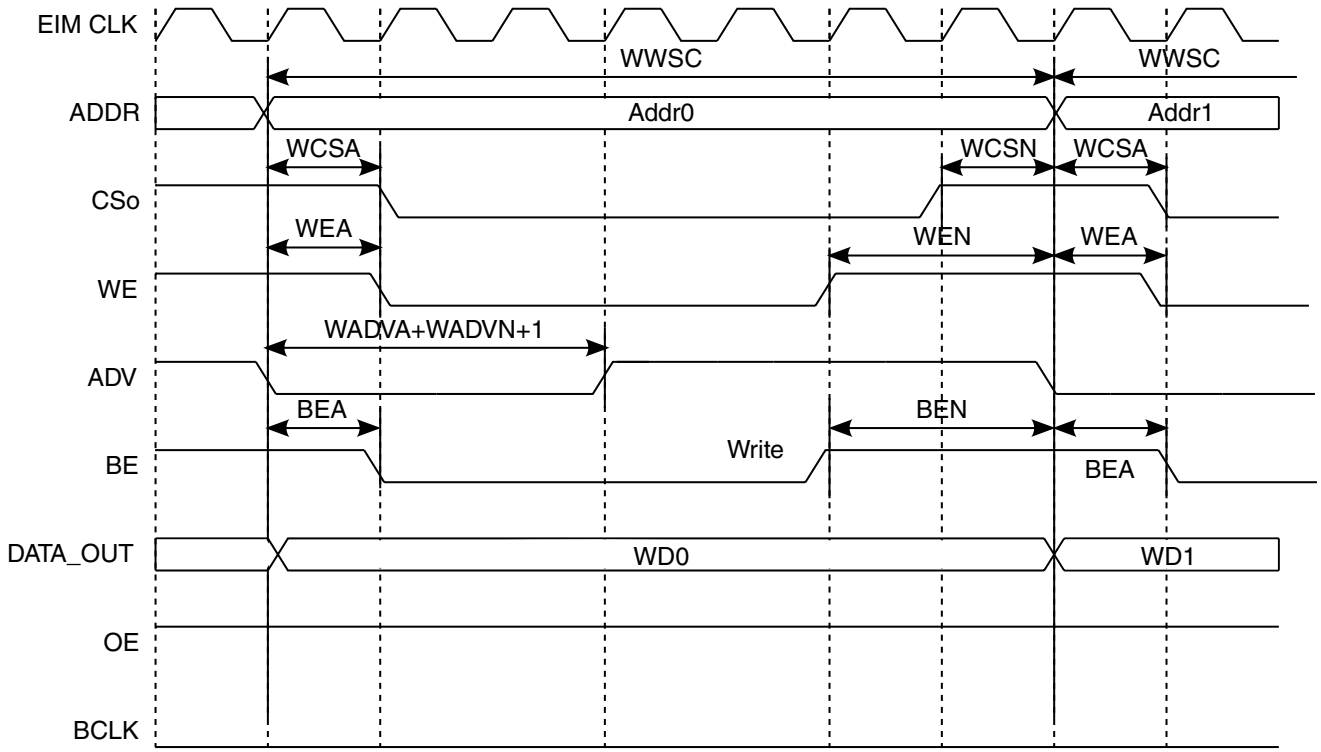


Figure 9-63.
WWSC=7,WCSA=1,WCSN=1,WEA=1,WEN=2,WADVA=0,WADVN=2,BEA=1,BEN=2

9.7.8.6 Consecutive Asynchronous Read Memory Accesses Timing Diagram

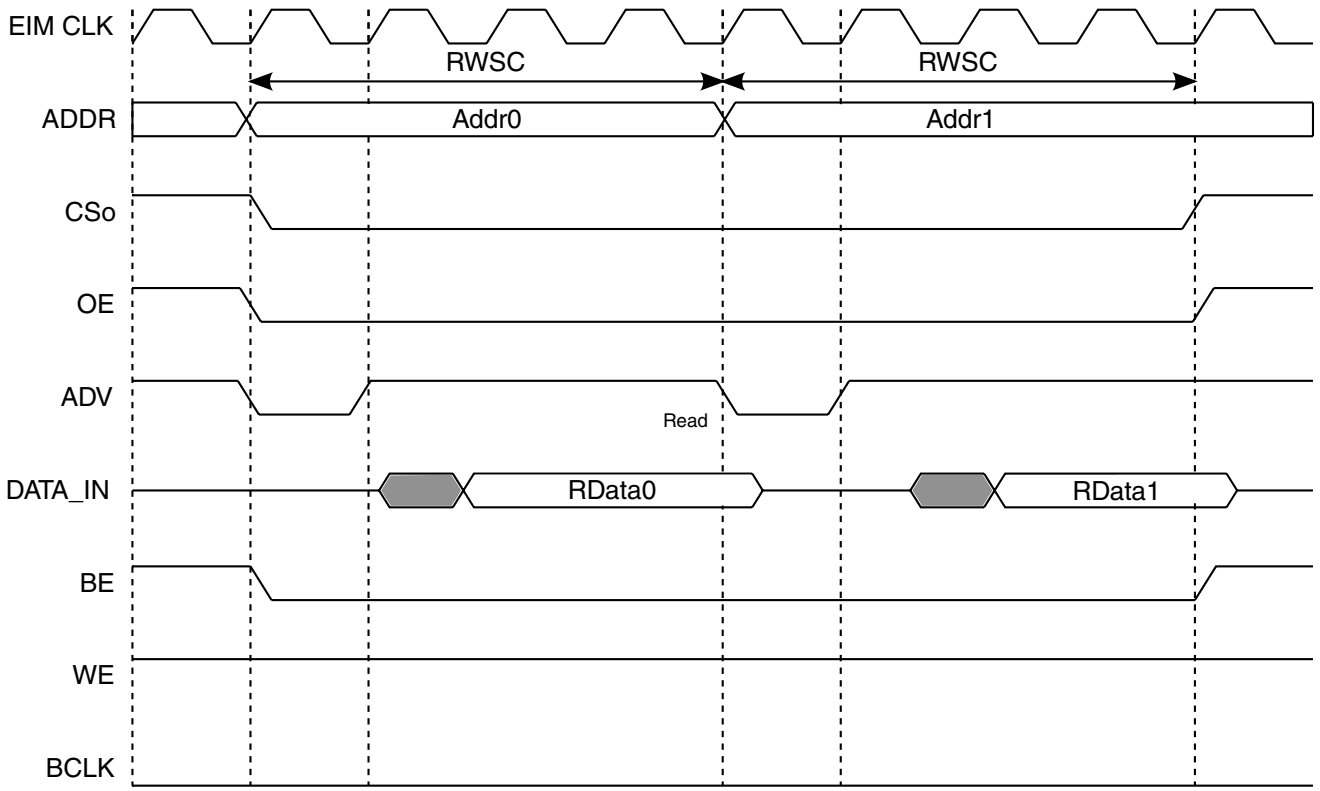


Figure 9-64. RWSC=4,RCSA=0,OEA=0,RADVA=0,RCSN=0,OEN=0,RADVN=0,CSREC=0

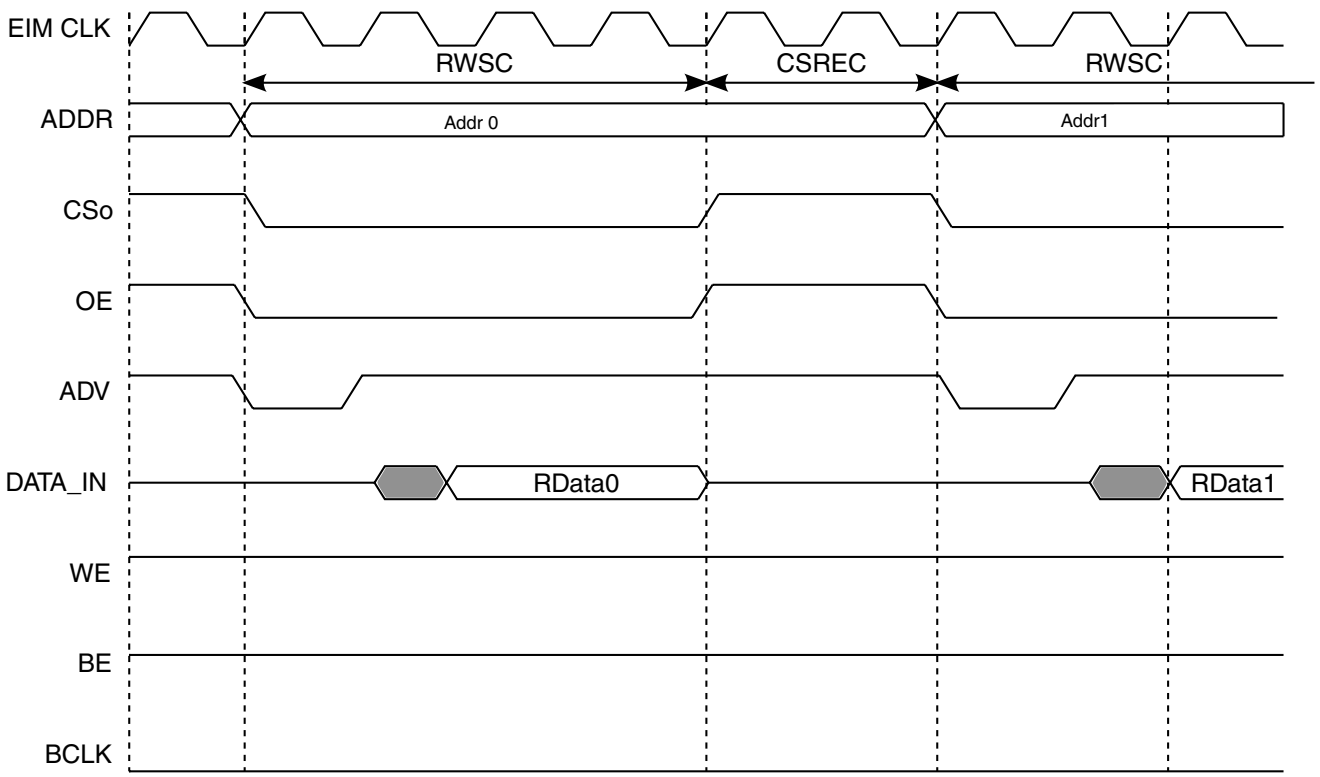


Figure 9-65. RWSC=4,RCSA=0,OEA=0,RADVA=0,RCSN=0,OEN=0,RADVN=0,CSREC=2

9.7.8.7 Burst (Synchronous Mode) Read Memory Accesses Timing Diagram - BCD=0

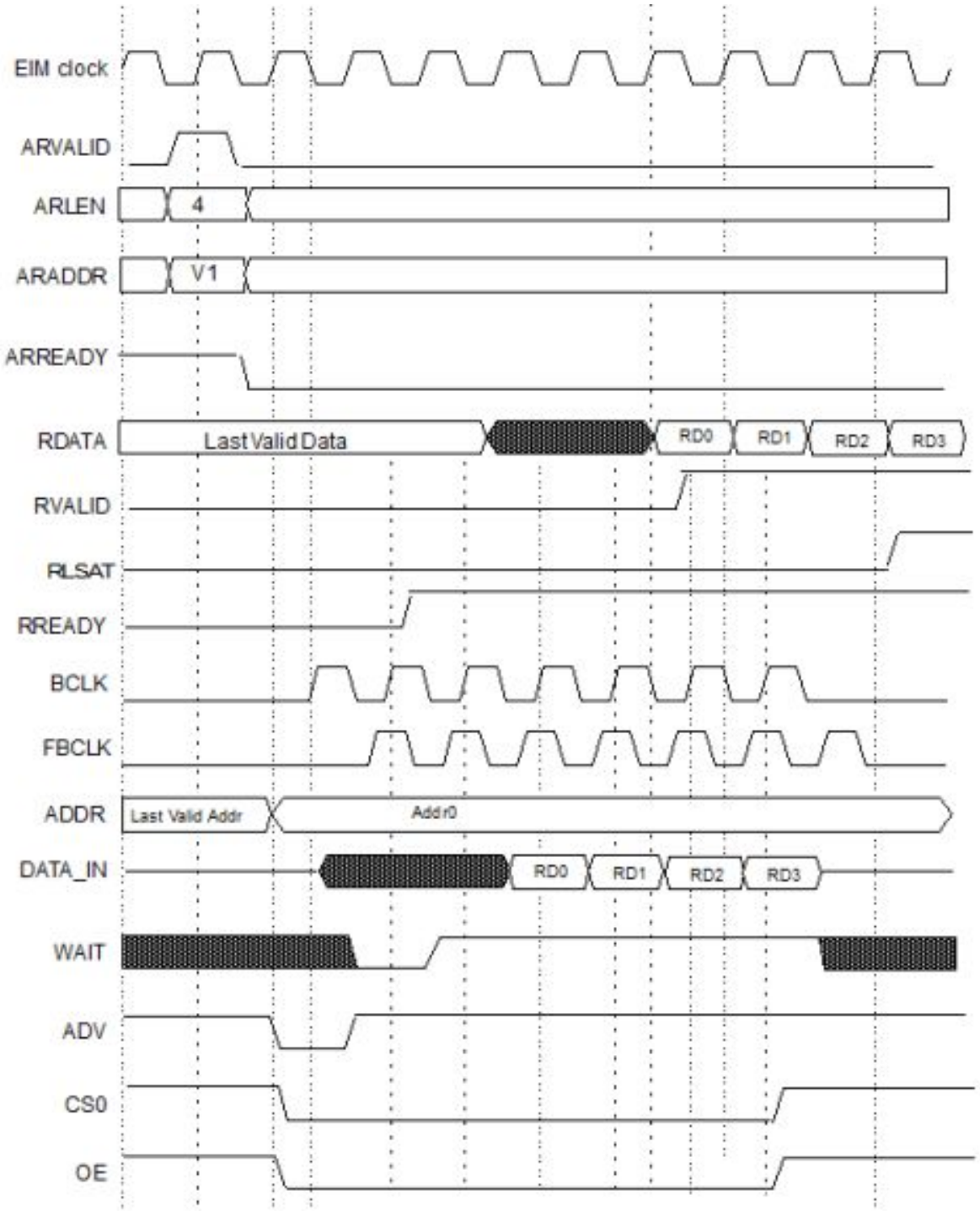


Figure 9-66. SRD=1,BCD=0,BCS=0,RWSC=1,RADVA=0,RADVN=0,RFL=0,RL=0

9.7.8.8 Burst (Synchronous Mode) Read Memory Accesses Timing Diagram - BCD=1

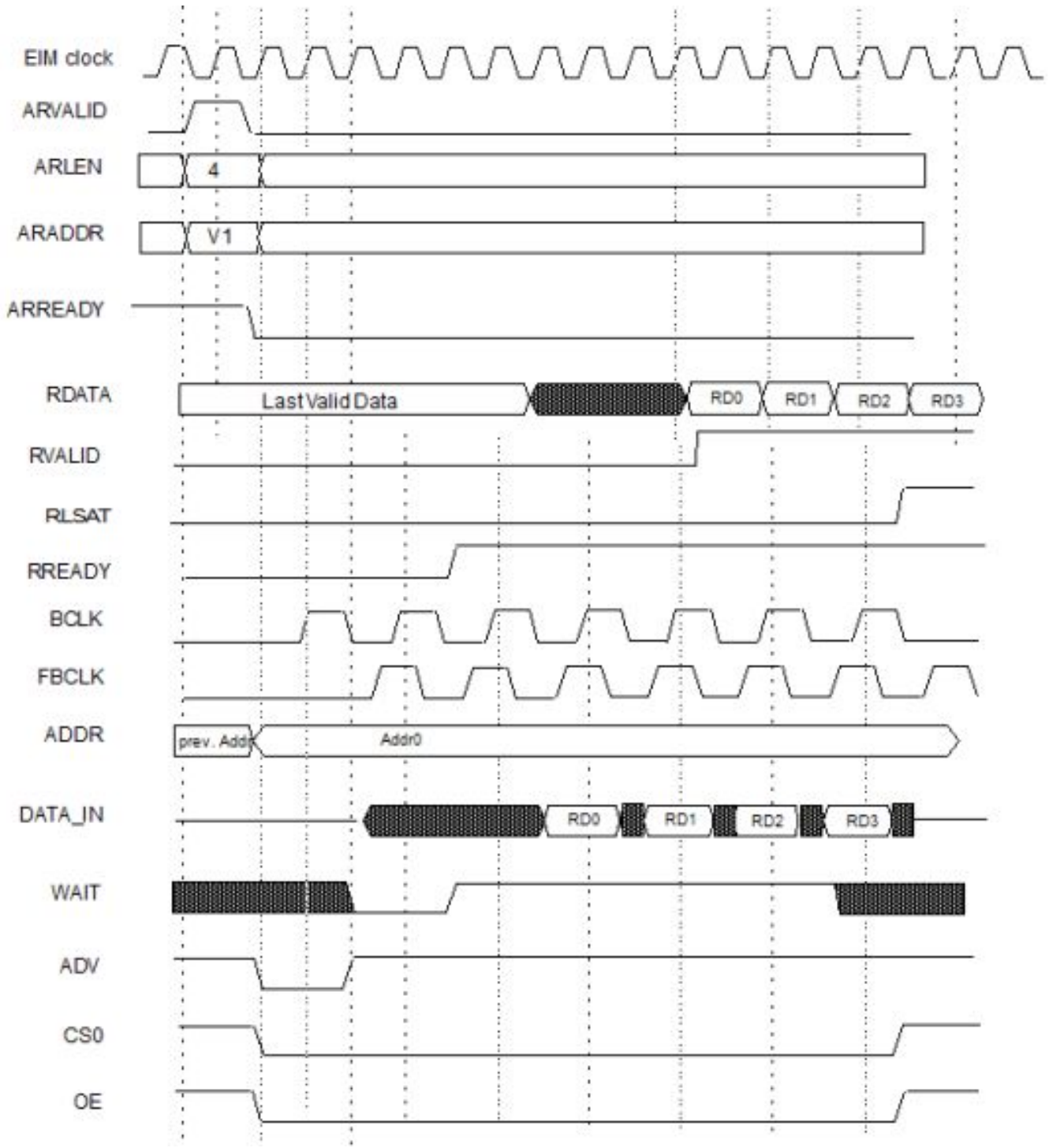
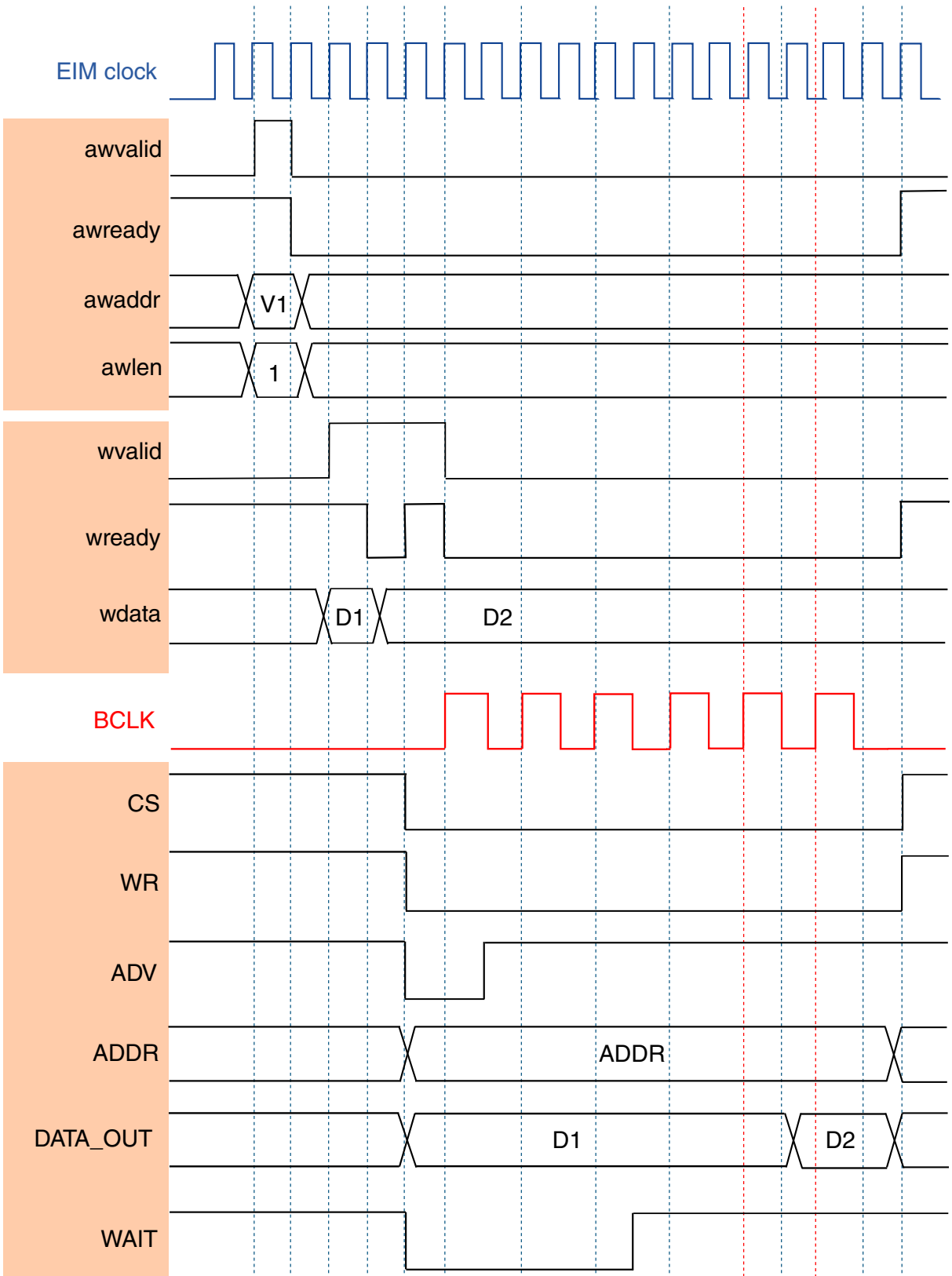


Figure 9-67. BCD=1, RL = 3

9.7.8.9 Burst (Synchronous Mode) Write Memory Access Timing - BCD=1



9.7.8.10 Asynchronous Page Mode Access

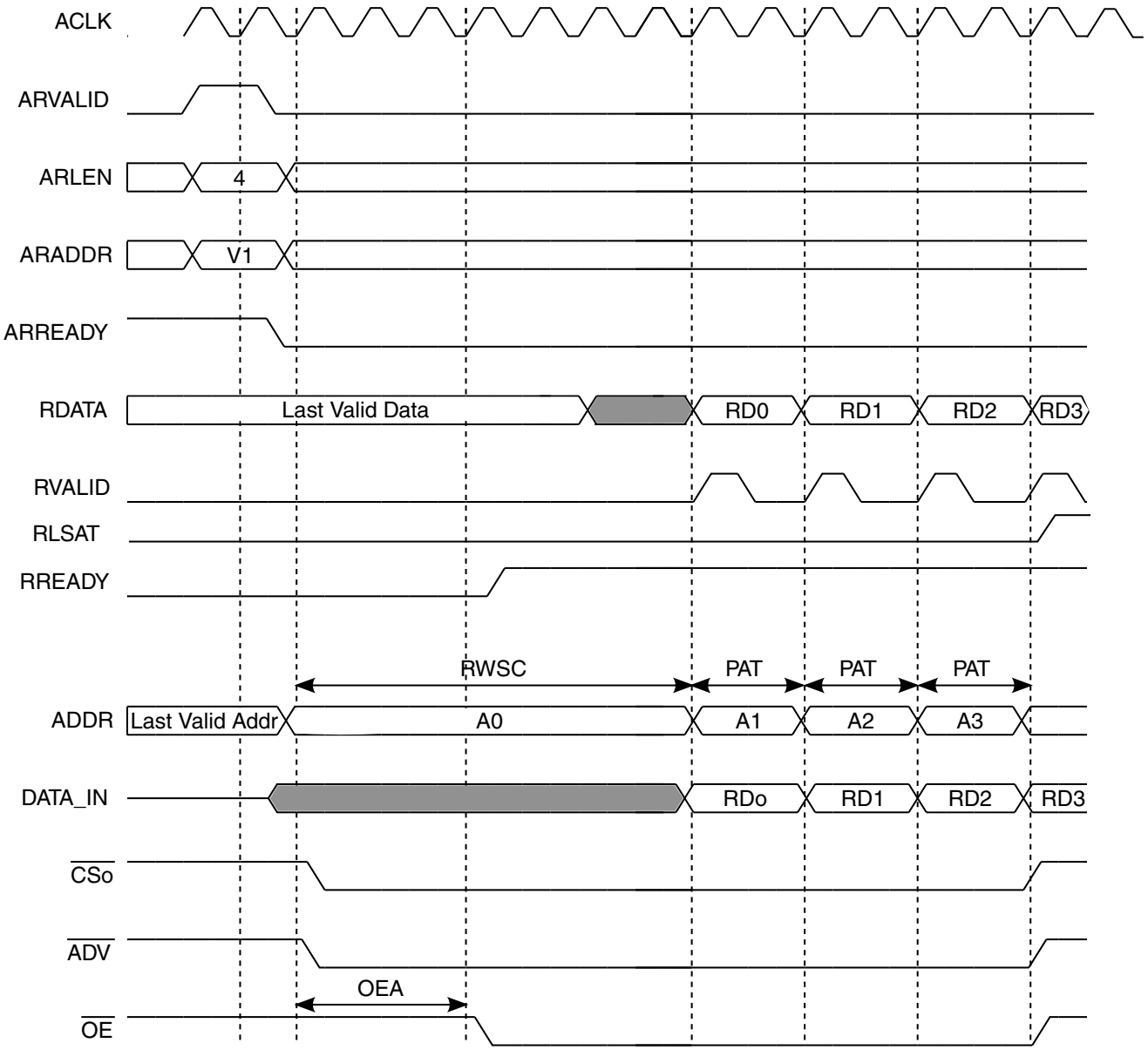


Figure 9-68. PAT = 2

9.7.8.11 DTACK Mode - AXI Single Access

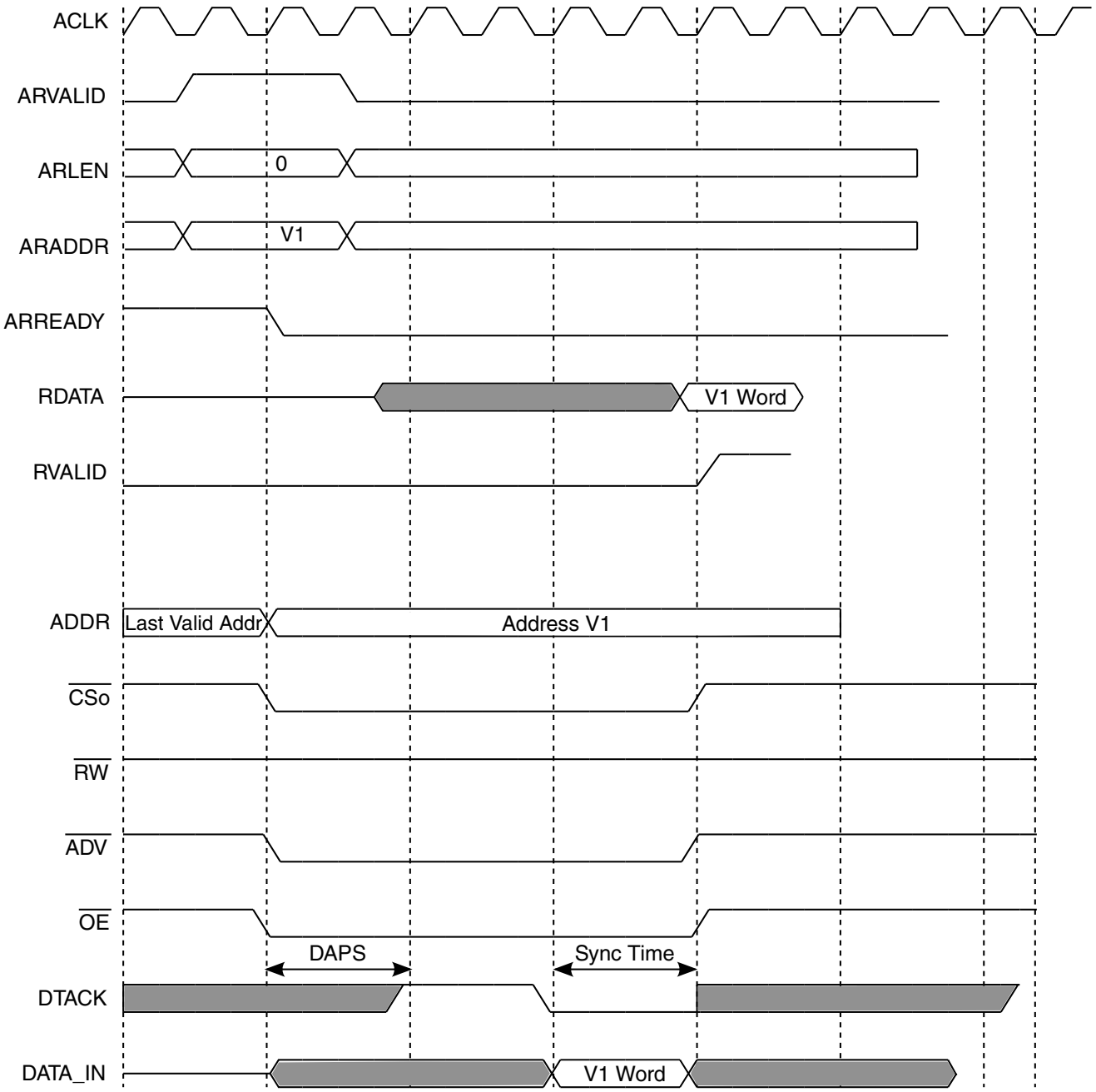


Figure 9-69. DAPS = 2

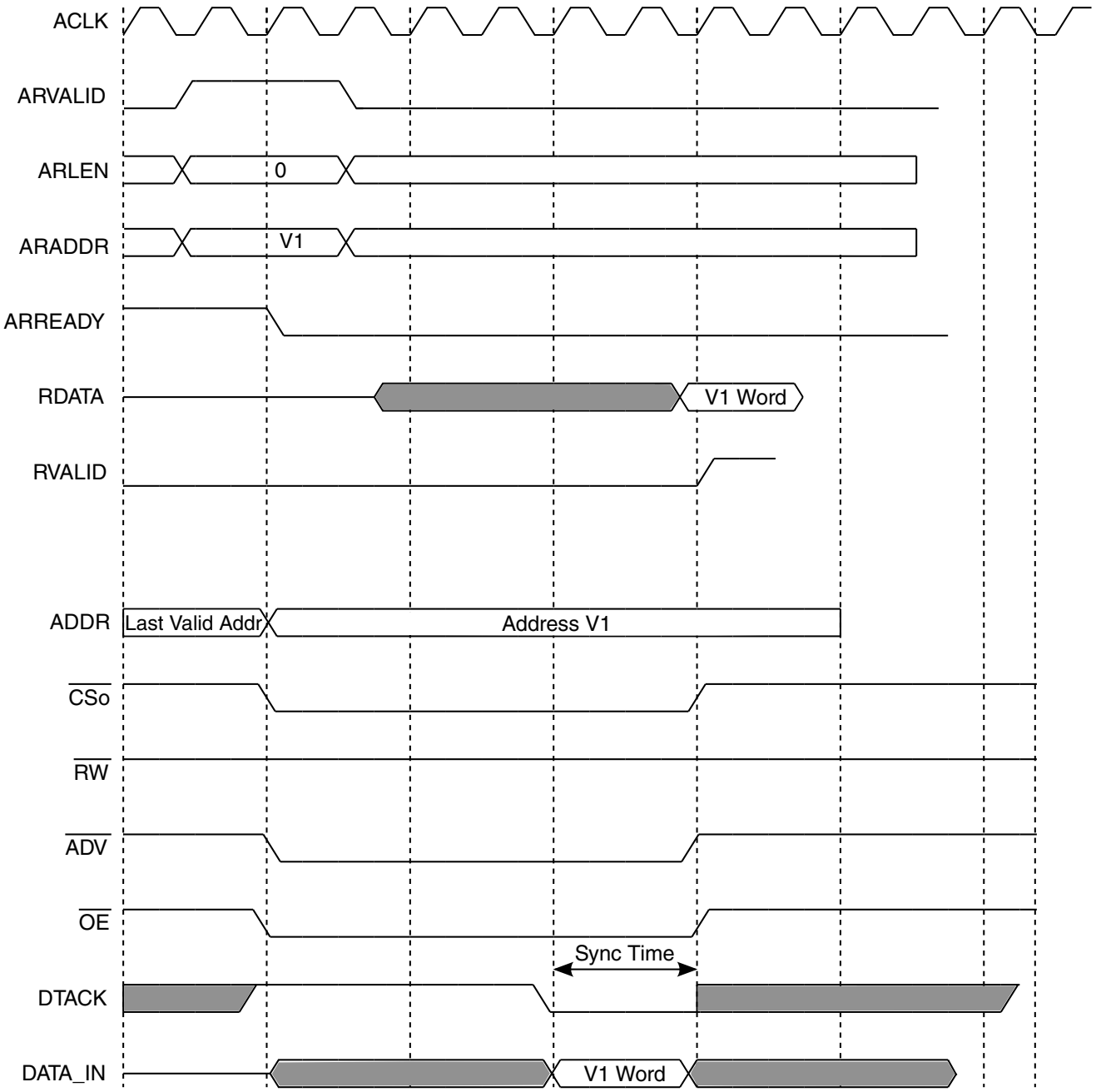


Figure 9-70. DAPS = 0

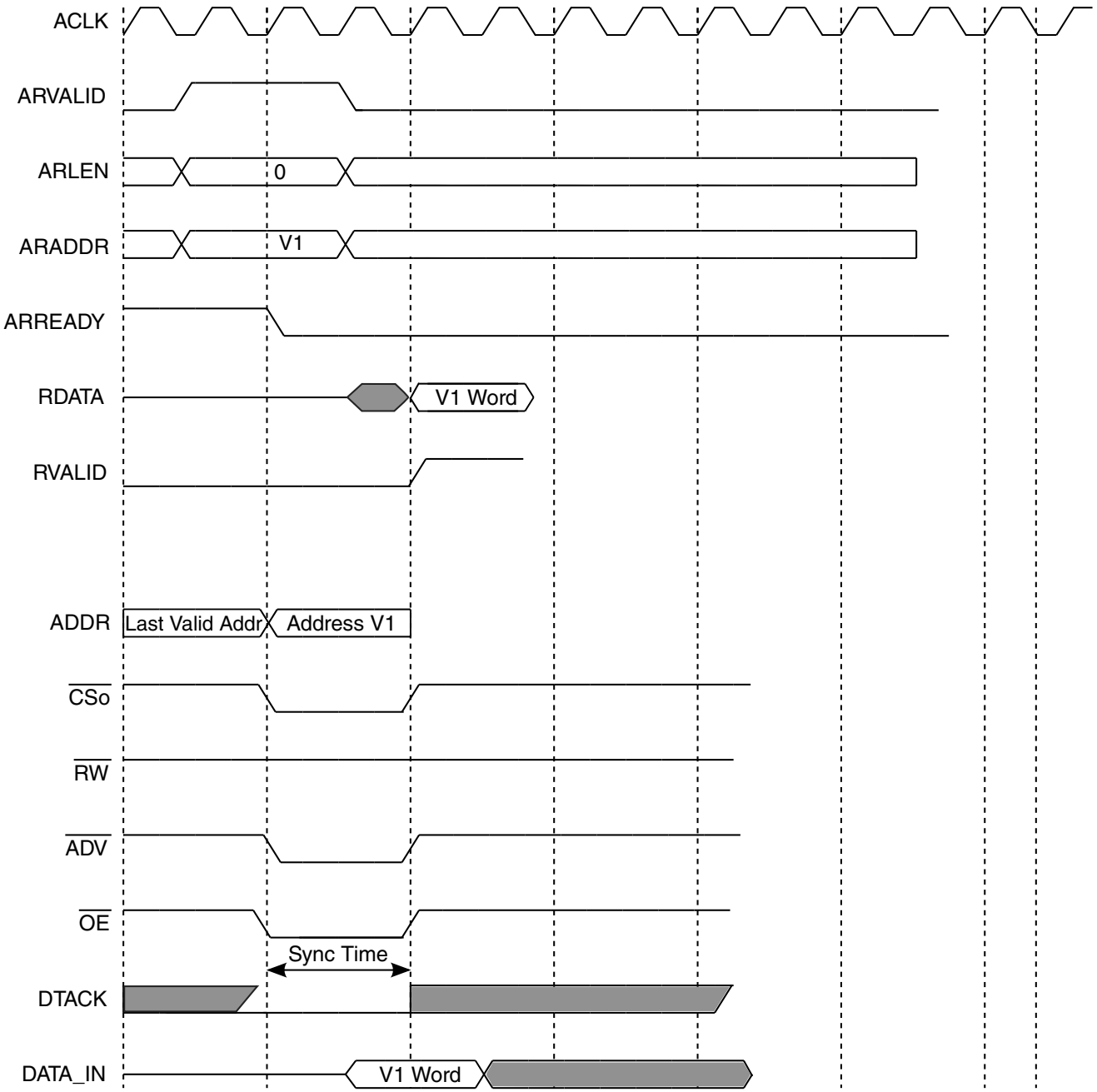


Figure 9-71. DAPS = 0

9.7.8.12 DTACK Mode - AXI Single Write Access

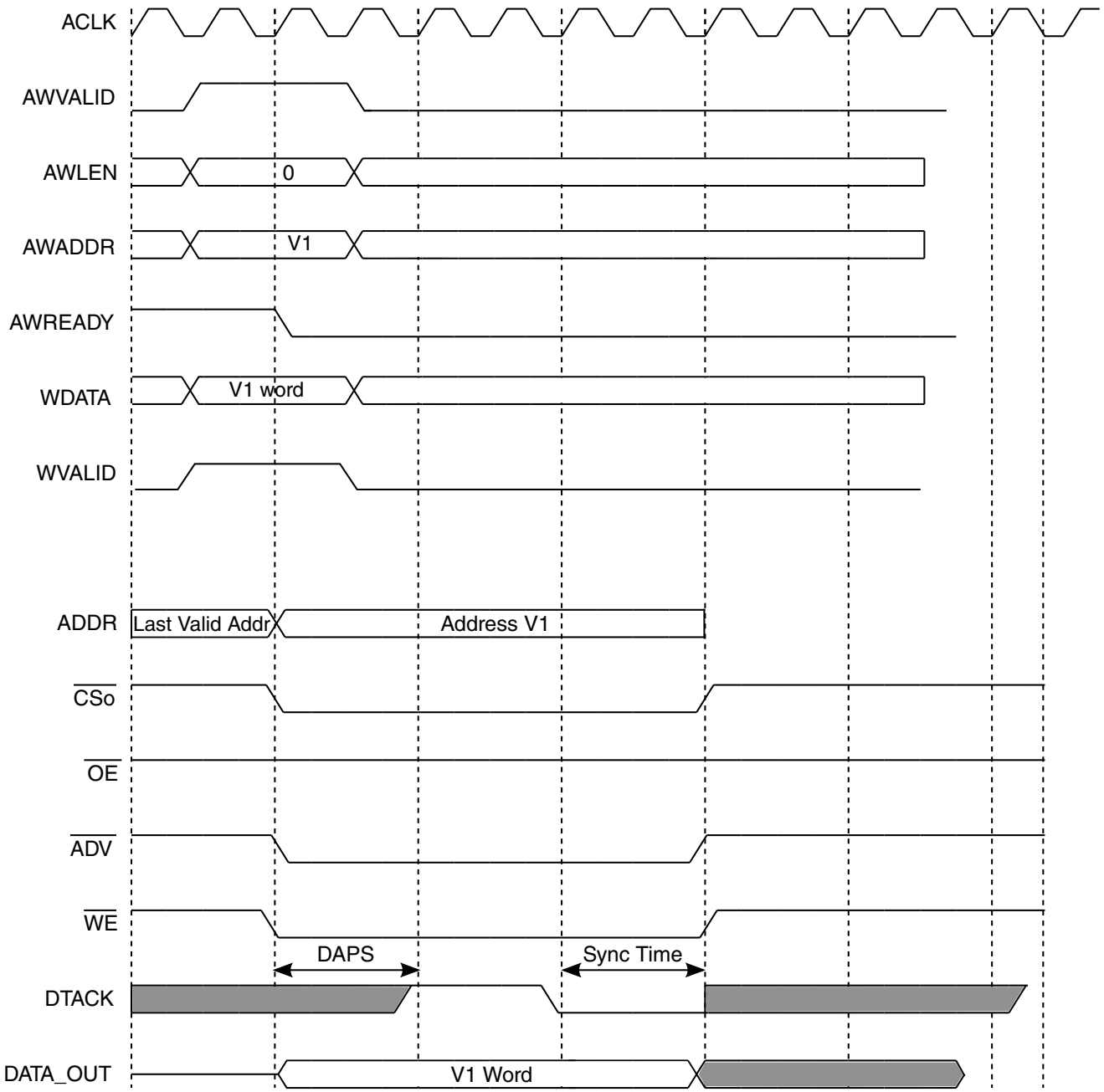


Figure 9-72. DAPS = 2

9.7.8.13 DTACK Mode - AXI Burst Access

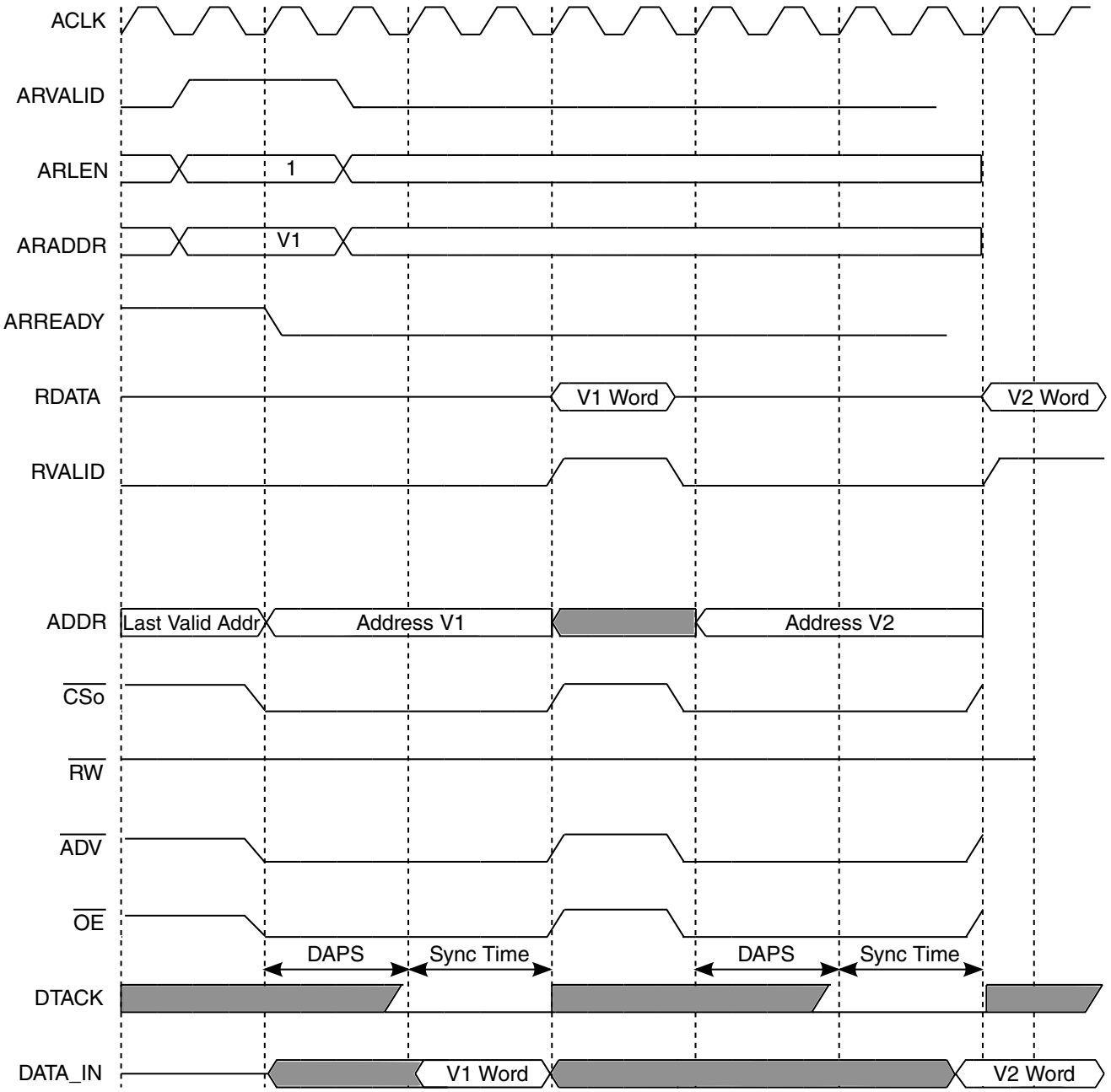


Figure 9-73. DAPS = 2 CSREC = 2

9.7.9 EIM Memory Map/Register Definition

The EIM includes 33 user-accessible 32-bit registers. The the EIM Configuration Register (EIM_WCR) contains control bits that configure the EIM for certain operation modes.

The 160 bits used to control Individual Chip Select are divided into five registers:

- Chip Select n General Configuration Register 1 (EIM_CSnGCR1)
- Chip Select n General Configuration Register 2 (EIM_CSnGCR2)
- Chip Select n Read Configuration Register 1 (EIM_CSnRCR1)
- Chip Select n Read Configuration Register 2 (EIM_CSnRCR2)
- Chip Select n Write Configuration Register (EIM_CSnWCR)

In addition there are 3 general registers: EIM_WCR, EIM_WIAR & EIM_EAR.

NOTE

- All EIM registers are sampled by IPG_CLK_S, therefore IPG_CLK_S must be active when accessing through IP bus.
- Read access from all registers (except EIM_WIAR & EIM_EAR) will generate one IPG_XFR_WAIT cycle.
- Read access from EIM_WIAR & EIM_EAR will generate six IPG_XFR_WAIT cycles.
- Write access to all registers (except EIM_EAR) will generate three IPG_XFR_WAIT cycles.
- Write access to EIM_EAR will generate six IPG_XFR_WAIT cycles.

EIM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BC_0000	Chip Select n General Configuration Register 1 (EIM_CS0GCR1)	32	R/W	0001_0080h	9.7.9.1/ 2543
30BC_0004	Chip Select n General Configuration Register 2 (EIM_CS0GCR2)	32	R/W	0000_1000h	9.7.9.2/ 2548
30BC_0008	Chip Select n Read Configuration Register 1 (EIM_CS0RCR1)	32	R/W	0000_0000h	9.7.9.3/ 2549
30BC_000C	Chip Select n Read Configuration Register 2 (EIM_CS0RCR2)	32	R/W	0000_0000h	9.7.9.4/ 2552
30BC_0010	Chip Select n Write Configuration Register 1 (EIM_CS0WCR1)	32	R/W	0000_0000h	9.7.9.5/ 2553
30BC_0014	Chip Select n Write Configuration Register 2 (EIM_CS0WCR2)	32	R/W	0000_0000h	9.7.9.6/ 2556
30BC_0018	Chip Select n General Configuration Register 1 (EIM_CS1GCR1)	32	R/W	0001_0080h	9.7.9.1/ 2543

Table continues on the next page...

EIM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BC_001C	Chip Select n General Configuration Register 2 (EIM_CS1GCR2)	32	R/W	0000_1000h	9.7.9.2/ 2548
30BC_0020	Chip Select n Read Configuration Register 1 (EIM_CS1RCR1)	32	R/W	0000_0000h	9.7.9.3/ 2549
30BC_0024	Chip Select n Read Configuration Register 2 (EIM_CS1RCR2)	32	R/W	0000_0000h	9.7.9.4/ 2552
30BC_0028	Chip Select n Write Configuration Register 1 (EIM_CS1WCR1)	32	R/W	0000_0000h	9.7.9.5/ 2553
30BC_002C	Chip Select n Write Configuration Register 2 (EIM_CS1WCR2)	32	R/W	0000_0000h	9.7.9.6/ 2556
30BC_0030	Chip Select n General Configuration Register 1 (EIM_CS2GCR1)	32	R/W	0001_0080h	9.7.9.1/ 2543
30BC_0034	Chip Select n General Configuration Register 2 (EIM_CS2GCR2)	32	R/W	0000_1000h	9.7.9.2/ 2548
30BC_0038	Chip Select n Read Configuration Register 1 (EIM_CS2RCR1)	32	R/W	0000_0000h	9.7.9.3/ 2549
30BC_003C	Chip Select n Read Configuration Register 2 (EIM_CS2RCR2)	32	R/W	0000_0000h	9.7.9.4/ 2552
30BC_0040	Chip Select n Write Configuration Register 1 (EIM_CS2WCR1)	32	R/W	0000_0000h	9.7.9.5/ 2553
30BC_0044	Chip Select n Write Configuration Register 2 (EIM_CS2WCR2)	32	R/W	0000_0000h	9.7.9.6/ 2556
30BC_0048	Chip Select n General Configuration Register 1 (EIM_CS3GCR1)	32	R/W	0001_0080h	9.7.9.1/ 2543
30BC_004C	Chip Select n General Configuration Register 2 (EIM_CS3GCR2)	32	R/W	0000_1000h	9.7.9.2/ 2548
30BC_0050	Chip Select n Read Configuration Register 1 (EIM_CS3RCR1)	32	R/W	0000_0000h	9.7.9.3/ 2549
30BC_0054	Chip Select n Read Configuration Register 2 (EIM_CS3RCR2)	32	R/W	0000_0000h	9.7.9.4/ 2552
30BC_0058	Chip Select n Write Configuration Register 1 (EIM_CS3WCR1)	32	R/W	0000_0000h	9.7.9.5/ 2553
30BC_005C	Chip Select n Write Configuration Register 2 (EIM_CS3WCR2)	32	R/W	0000_0000h	9.7.9.6/ 2556
30BC_0060	Chip Select n General Configuration Register 1 (EIM_CS4GCR1)	32	R/W	0001_0080h	9.7.9.1/ 2543
30BC_0064	Chip Select n General Configuration Register 2 (EIM_CS4GCR2)	32	R/W	0000_1000h	9.7.9.2/ 2548
30BC_0068	Chip Select n Read Configuration Register 1 (EIM_CS4RCR1)	32	R/W	0000_0000h	9.7.9.3/ 2549
30BC_006C	Chip Select n Read Configuration Register 2 (EIM_CS4RCR2)	32	R/W	0000_0000h	9.7.9.4/ 2552
30BC_0070	Chip Select n Write Configuration Register 1 (EIM_CS4WCR1)	32	R/W	0000_0000h	9.7.9.5/ 2553

Table continues on the next page...

EIM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BC_0074	Chip Select n Write Configuration Register 2 (EIM_CS4WCR2)	32	R/W	0000_0000h	9.7.9.6/ 2556
30BC_0078	Chip Select n General Configuration Register 1 (EIM_CS5GCR1)	32	R/W	0001_0080h	9.7.9.1/ 2543
30BC_007C	Chip Select n General Configuration Register 2 (EIM_CS5GCR2)	32	R/W	0000_1000h	9.7.9.2/ 2548
30BC_0080	Chip Select n Read Configuration Register 1 (EIM_CS5RCR1)	32	R/W	0000_0000h	9.7.9.3/ 2549
30BC_0084	Chip Select n Read Configuration Register 2 (EIM_CS5RCR2)	32	R/W	0000_0000h	9.7.9.4/ 2552
30BC_0088	Chip Select n Write Configuration Register 1 (EIM_CS5WCR1)	32	R/W	0000_0000h	9.7.9.5/ 2553
30BC_008C	Chip Select n Write Configuration Register 2 (EIM_CS5WCR2)	32	R/W	0000_0000h	9.7.9.6/ 2556
30BC_0090	EIM Configuration Register (EIM_WCR)	32	R/W	See section	9.7.9.7/ 2557
30BC_0094	DLL Control Register (EIM_DCR)	32	R/W	0014_0000h	9.7.9.8/ 2559
30BC_0098	DLL Status Register (EIM_DSR)	32	R	0000_0000h	9.7.9.9/ 2560
30BC_009C	EIM IP Access Register (EIM_WIAR)	32	R/W	0000_0010h	9.7.9.10/ 2561
30BC_00A0	Error Address Register (EIM_EAR)	32	R/W	0000_0000h	9.7.9.11/ 2562

9.7.9.1 Chip Select n General Configuration Register 1 (EIM_CS*n*GCR1)

Address: 30BC_0000h base + 0h offset + (24d × i), where i=0d to 5d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PSZ				WP	GBC			AUS	CSREC			SP	DSZ		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BCS		BCD		WC	BL			CREP	CRE	RFL	WFL	MUM	SRD	SWR	CSEN
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

EIM_CS*n*GCR1 field descriptions

Field	Description
31–28 PSZ	Page Size. This bit field indicates memory page size in words (word is defined by the DSZ field). PSZ is used when fix latency mode is applied, WFL=1 for sync. write accesses, RFL=1 for sync. Read accesses. When working in fix latency mode WAIT signal from the external device is not being monitored, PSZ is used to determine if page boundary is reached and renewal of access is preformed. This bit field is ignored when sync. Mode is disabled or fix latency mode is not being used for write or read access separately. It can be valid for both access type, read or write, or only for one type, according to configuration. PSZ is cleared by a hardware reset. 0000 8 words page size 0001 16 words page size 0010 32 words page size 0011 64 words page size 0100 128 words page size 0101 256 words page size 0110 512 words page size 0111 1024 (1k) words page size 1000 2048 (2k) words page size 1001 - 1111 Reserved
27 WP	Write Protect. This bit prevents writes to the address range defined by the corresponding chip select. WP is cleared by a hardware reset. 0 Writes are allowed in the memory range defined by chip. 1 Writes are prohibited. All attempts to write to an address mapped by this chip select result in a error response and no assertion of the chip select output.
26–24 GBC	Gap Between Chip Selects. This bit field, according to the settings shown below, determines the minimum time between end of access to the current chip select and start of access to different chip select. GBC is cleared by a hardware reset. Example settings: 000 minimum of 0 EIM clock cycles before next access from different chip select (async. mode only) 001 minimum of 1 EIM clock cycles before next access from different chip select 010 minimum of 2 EIM clock cycles before next access from different chip select 111 minimum of 7 EIM clock cycles before next access from different chip select
23 AUS	Address UnShifted. This bit indicates an unshifted mode for address assertion for the relevant chip select accesses. AUS bit is cleared by hardware reset. 0 Address shifted according to port size (DSZ config) (128 Mbyte maximum supported memory density). 1 Address unshifted (32 Mbyte maximum supported memory density).
22–20 CSREC	CS Recovery. This bit field, according to the settings shown below, determines the minimum pulse width of CS, OE, and WE control signals before executing a new back to back access to the same chip select. CSREC is cleared by a hardware reset. **NOTE:** The reset value for EIM_CS0GCR1, CSREC[2:0] is 0b110. For EIM_CS1GCR1 - EIM_CS5GCR, the reset value is 0b000. Example settings: 000 0 EIM clock cycles minimum width of CS, OE and WE signals (read async. mode only) 001 1 EIM clock cycles minimum width of CS, OE and WE signals

Table continues on the next page...

EIM_CS*n*GCR1 field descriptions (continued)

Field	Description
	010 2 EIM clock cycles minimum width of CS, OE and WE signals 111 7 EIM clock cycles minimum width of CS, OE and WE signals
19 SP	Supervisor Protect. This bit prevents accesses to the address range defined by the corresponding chip select when the access is attempted in the User mode. SP is cleared by a hardware reset. 0 User mode accesses are allowed in the memory range defined by chip select. 1 User mode accesses are prohibited. All attempts to access an address mapped by this chip select in User mode results in an error response and no assertion of the chip select output.
18–16 DSZ	Data Port Size. This bit field defines the width of an external device's data port as shown below. **NOTE:** Only async. access supported for 8 bit port. **NOTE:** The reset value for EIM_CS0GCR1, DSZ[2] = 0, DSZ[1:0] = EIM_BOOT[1:0]. For EIM_CS1GCR1 - EIM_CS5GCR1, the reset value is 0b001. 000 Reserved. 001 16 bit port resides on DATA[15:0] 010 16 bit port resides on DATA[31:16] 011 32 bit port resides on DATA[31:0] 100 8 bit port resides on DATA[7:0] 101 8 bit port resides on DATA[15:8] 110 8 bit port resides on DATA[23:16] 111 8 bit port resides on DATA[31:24]
15–14 BCS	Burst Clock Start. When SRD=1 or SWR=1,this bit field determines the number of EIM clock cycles delay from start of access before the first rising edge of BCLK is generated. When BCD=0 value of BCS=0 results in a half clock delay after the start of access. For other values of BCD a one clock delay after the start of access is applied, not an immediate assertion. BCS is cleared by a hardware reset. 00 0 EIM clock cycle additional delay 01 1 EIM clock cycle additional delay 10 2 EIM clock cycle additional delay 11 3 EIM clock cycle additional delay
13–12 BCD	Burst Clock Divisor. This bit field contains the value used to program the burst clock divisor for BCLK generation. It is used to divide the internal EIMbus frequency. BCD is cleared by a hardware reset. **NOTE:** For other then the mentioned below frequency such as 104 MHz, EIM clock (input clock) should be adjust accordingly. 00 Divide EIM clock by 1 01 Divide EIM clock by 2 10 Divide EIM clock by 3 11 Divide EIM clock by 4
11 WC	Write Continuous. The WI bit indicates that write access to the memory are always continuous accesses regardless of the BL field value. WI is cleared by hardware reset. 0 Write access burst length occurs according to BL value. 1 Write access burst length is continuous.
10–8 BL	Burst Length. The BL bit field indicates memory burst length in words (word is defined by the DSZ field) and should be properly initialized for mixed wrap/increment accesses support. Continuous BL value corresponds to continuous burst length setting of the external memory device. For fix memory burst size,

Table continues on the next page...

EIM_CS*n*GCR1 field descriptions (continued)

Field	Description
	type is always wrap. In case not matching wrap boundaries in both the memory (BL field) and Master access on the current address, EIM update address on the external device address bus and regenerates the access. BL is cleared by a hardware reset. When APR=1, Page Read Mode is applied, BL determine the number of words within the read page burst. BL is cleared by a hardware reset for EIM_CS0GCR1 - EIM_CS5GCR1. 000 4 words Memory wrap burst length (read page burst size when APR = 1) 001 8 words Memory wrap burst length (read page burst size when APR = 1) 010 16 words Memory wrap burst length (read page burst size when APR = 1) 011 32 words Memory wrap burst length (read page burst size when APR = 1) 100 Continuous burst length (2 words read page burst size when APR = 1) 101 Reserved 110 Reserved 111 Reserved
7 CREP	Configuration Register Enable Polarity. This bit indicates CRE memory pin assertion state, active-low or active-high, while executing a memory register set command to the external device (PSRAM memory type). CREP is set by a hardware reset. **NOTE:** Whenever PSRAM is connected the CREP value must be correct also for accesses where CRE is disabled. For Non-PSRAM memory CREP value should be 1. 0 CRE signal is active low 1 CRE signal is active high
6 CRE	Configuration Register Enable. This bit indicates CRE memory pin state while executing a memory register set command to PSRAM external device. CRE is cleared by a hardware reset. 0 CRE signal use is disable 1 CRE signal use is enable
5 RFL	Read Fix Latency. This bit field determine if the controller is monitoring the WAIT signal from the External device connected to the chip select (handshake mode - fix or variable data latency) or if it start sampling data according to RWSC field, it only valid in synchronous mode. RFL is cleared by a hardware reset. When RFL=1 Burst access is terminated on page boundary and resume on the following page according to BL bit field configuration, because WAIT signal is not monitored from the external device. 0 the External device WAIT signal is being monitored, and it reflect the external data bus state 1 the state of the External devices is determined internally (Fix latency mode only)
4 WFL	Write Fix Latency. This bit field determine if the controller is monitoring the WAIT signal from the External device connected to the chip select (handshake mode - fix or variable data latency) or if it start data transfer according to WWSC field, it only valid in synchronous mode. WFL is cleared by a hardware reset. When WFL=1 Burst access is terminated on page boundary and resume on the following page according to BL bit field configuration, because WAIT signal is not monitored from the external device 0 the External device WAIT signal is being monitored, and it reflect the external data bus state 1 the state of the External devices is determined internally (Fix latency mode only)
3 MUM	Multiplexed Mode. This bit determines the address/data multiplexed mode for asynchronous and synchronous accesses for 8 bit, 16 bit or 32 bit devices (DSZ config. dependent).

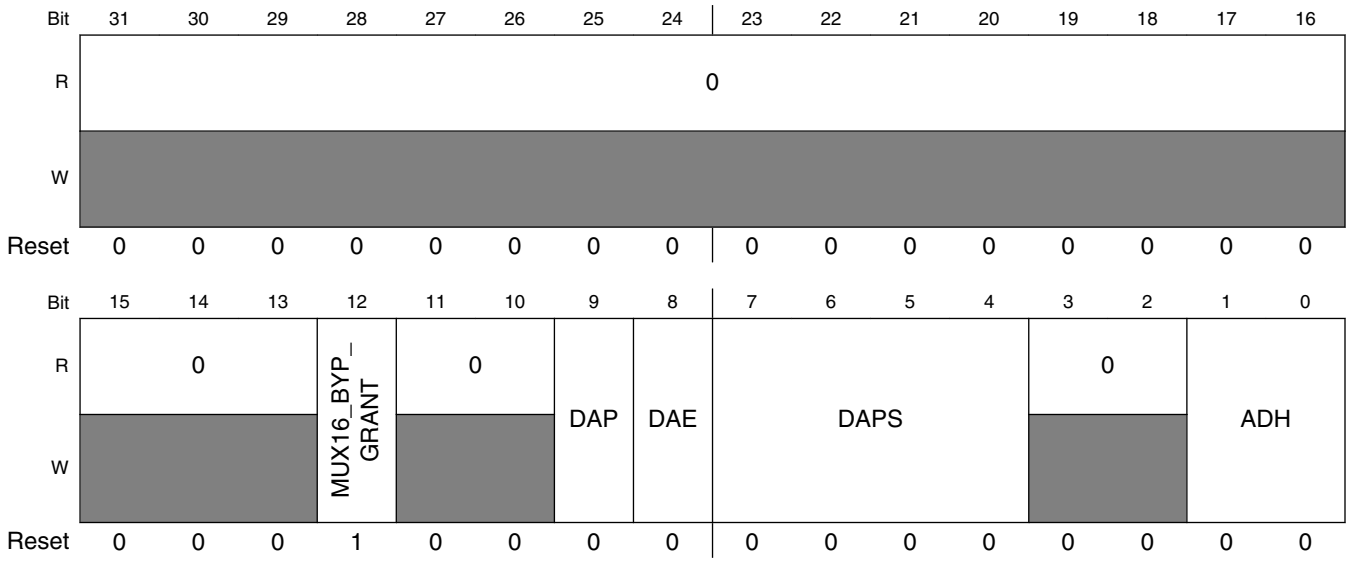
Table continues on the next page...

EIM_CS*n*GCR1 field descriptions (continued)

Field	Description
	NOTE: The reset value for EIM_CS0GCR1[MUM] = EIM_BOOT[2]. For EIM_CS1GCR1 - EIM_CS5GCR1 the reset value is 0. 0 Multiplexed Mode disable 1 Multiplexed Mode enable
2 SRD	Synchronous Read Data. This bit field determine the read accesses mode to the External device of the chip select. The External device should be configured to the same mode as this bit implicates. SRD is cleared by a hardware reset. **NOTE:** Sync. accesses supported only for 16/32 bit port. 0 read accesses are in Asynchronous mode 1 read accesses are in Synchronous mode
1 SWR	Synchronous Write Data. This bit field determine the write accesses mode to the External device of the chip select. The External device should be configured to the same mode as this bit implicates. SWR is cleared by a hardware reset. **NOTE:** Sync. accesses supported only for 16/32 bit port. 0 write accesses are in Asynchronous mode 1 write accesses are in Synchronous mode
0 CSEN	CS Enable. This bit controls the operation of the chip select pin. CSEN is set by a hardware reset for CSGCR0 to allow external boot operation. CSEN is cleared by a hardware reset to CSGCR1-CSGCR5. **NOTE:** Reset value for EIM_CS0GCR1 for CSEN is 1. For EIM_CS1GCR1-CS1GCR5 reset value is 0. 0 Chip select function is disabled; attempts to access an address mapped by this chip select results in an error respond and no assertion of the chip select output 1 Chip select is enabled, and is asserted when presented with a valid access.

9.7.9.2 Chip Select n General Configuration Register 2 (EIM_CS*n*GCR2)

Address: 30BC_0000h base + 4h offset + (24d × i), where i=0d to 5d



EIM_CS*n*GCR2 field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value 0.
12 MUX16_BYP_GRANT	Muxed 16 bypass grant. This bit when asserted causes EIM to bypass the grant/ack. arbitration with NFC (only for 16 bit muxed mode accesses). 0 EIM waits for grant before driving a 16 bit muxed mode access to the memory. 1 EIM ignores the grant signal and immediately drives a 16 bit muxed mode access to the memory.
11–10 Reserved	This read-only field is reserved and always has the value 0.
9 DAP	Data Acknowledge Polarity. This bit indicates DTACK memory pin assertion state, active-low or active-high, while executing an async access using DTACK signal from the external device. DAP is cleared by a hardware reset. 0 DTACK signal is active high 1 DTACK signal is active low
8 DAE	Data Acknowledge Enable. This bit indicates external device is using DTACK pin as strobe/terminator of an async. access. DTACK signal may be used only in asynchronous single read (APR=0) or write accesses. DTACK poling start point is set by DAPS bit field. polarity of DTACK is set by DAP bit field. DAE is cleared by a hardware reset. 0 DTACK signal use is disable 1 DTACK signal use is enable
7–4 DAPS	Data Acknowledge Poling Start. This bit field determine the starting point of DTACK input signal polling. DAPS is used only in asynchronous single read or write accesses.

Table continues on the next page...

EIM_CS*n*GCR2 field descriptions (continued)

Field	Description
	NOTE: Since DTACK is an async. signal the start point of DTACK signal polling is at least 3 cycles after the start of access. DAPS is cleared by a hardware reset. Example settings: 0000 3 EIM clk cycle between start of access and first $\overline{\text{DTACK}}$ check 0001 4 EIM clk cycles between start of access and first $\overline{\text{DTACK}}$ check 0010 5 EIM clk cycles between start of access and first $\overline{\text{DTACK}}$ check 0111 10 EIM clk cycles between start of access and first $\overline{\text{DTACK}}$ check 1011 14 EIM clk cycles between start of access and first DTACK check 1111 18 EIM clk cycles between start of access and first DTACK check
3–2 Reserved	This read-only field is reserved and always has the value 0.
ADH	Address hold time - This bit field determine the address hold time after ADV negation when mum = 1 (muxed mode). When mum = 0 this bit has no effect. For read accesses the field determines when the pads direction will be switched. **NOTE:** Reset value for EIM_CS0GCR2 for ADH is 10. For EIM_CS1GCR2-EIM_CS5GCR2 reset value is 00. 00 0 cycle after ADV negation 01 1 cycle after ADV negation 10 2 cycle after ADV negation 11 Reserved

9.7.9.3 Chip Select n Read Configuration Register 1 (EIM_CS*n*RCR1)

Address: 30BC_0000h base + 8h offset + (24d × i), where i=0d to 5d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0		RWSC						0	RADVA			RAL	RADVN		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	OEA			0	OEN			0	RCSA			0	RCSN		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_CS*n*RCR1 field descriptions

Field	Description
31–30 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

EIM_CS*n*RCR1 field descriptions (continued)

Field	Description
29–24 RWSC	Read Wait State Control. This bit field programs the number of wait-states, according to the settings shown below, for synchronous or asynchronous read access to the external device connected to the chip select. When SRD=1 and RFL=0, RWSC indicates the number of burst clock (BCLK) cycles from the start of an access, before the controller can start sample data.Since WAIT signal can be asserted one cycle before the first data can be sampled, the controller starts evaluating the WAIT signal state one cycle before, this is referred as handshake mode or variable latency mode. When SRD=1 and RFL=1, RWSC indicates the number of burst clock (BCLK) cycles from the start of an access, until the external device is ready for data transfer, this is referred as fix latency mode. When SRD=0, RFL bit is ignored, RWSC indicates the asynchronous access length and the number of EIM clock cycles from the start of access until the external device is ready for data transfer. RWSC is cleared by a hardware reset. **NOTE:** The reset value for EIM_CS0RCR1, RWSC[5:0] = 0b011100. For CG1RCR1 - CS1RCR5 the reset value is 0b000000. Example settings: 000000 Reserved 000001 RWSC value is 1 000010 RWSC value is 2 111101 RWSC value is 61 111110 RWSC value is 62 111111 RWSC value is 63
23 Reserved	This read-only field is reserved and always has the value 0.
22–20 RADVA	ADV Assertion. This bit field determines when ADV signal is asserted for synchronous or asynchronous read modes according to the settings shown below. RADVA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of access and ADV assertion 001 1 EIM clock cycles between beginning of access and ADV assertion 010 2 EIM clock cycles between beginning of access and ADV assertion 111 7 EIM clock cycles between beginning of access and ADV assertion
19 RAL	Read ADV Low. This bit field determine ADV signal negation time. When RAL=1, RADVN bit field is ignored and ADV signal will stay asserted until end of access. When RAL=0 negation of ADV signal is according to RADVN bit field configuration.
18–16 RADVN	ADV Negation. This bit field determines when ADV signal to memory is negated during read accesses. When SRD=1 (synchronous read mode), ADV negation occurs according to the following formula: (RADVN + RADVA + BCD + BCS + 1) EIM clock cycles from start of access. When asynchronous read mode is applied (SRD=0) and RAL=0 ADV negation occurs according to the following formula: (RADVN + RADVA + 1) EIM clock cycles from start of access. RADVN is cleared by a hardware reset. **NOTE:** the reset value for EIM_CS0RCR1[RADVN] = 2. For EIM_CS1RCR1 - EIM_CS5RCR1, the reset value is 0b000. **NOTE:** This field should be configured so ADV negation will occur before the end of access. For ADV negation at the same time with the end of access user should RAL bit.

Table continues on the next page...

EIM_CS*n*RCR1 field descriptions (continued)

Field	Description
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 OEA	OE Assertion. This bit field determines when OE signal are asserted during read cycles (synchronous or asynchronous mode), according to the settings shown below. OEA is cleared by a hardware reset. In muxed mode OE assertion occurs (OEA + RADVN + RADVA + ADH +1) EIM clock cycles from start of access. **NOTE:** The reset value for EIM_CS0RCR1[OEA] is 0b000 if EIM_BOOT[2] = 0. If EIM_BOOT[2] is 1, the reset value for EIM_CS0RCR1 is 0b010. The reset value of this field for EIM_CS1RCR1 - EIM_CS5RCR1 is 0b000. Example settings: 000 0 EIM clock cycles between beginning of access and OE assertion 001 1 EIM clock cycles between beginning of access and OE assertion 010 2 EIM clock cycles between beginning of access and OE assertion 111 7 EIM clock cycles between beginning of access and OE assertion
11 Reserved	This read-only field is reserved and always has the value 0.
10–8 OEN	OE Negation. This bit field determines when OE signal is negated during read cycles in asynchronous single mode only (SRD=0 & APR = 0), according to the settings shown below. This bit field is ignored when SRD=1. OEN is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between end of access and OE negation 001 1 EIM clock cycles between end of access and OE negation 010 2 EIM clock cycles between end of access and OE negation 111 7 EIM clock cycles between end of access and OE negation
7 Reserved	This read-only field is reserved and always has the value 0.
6–4 RCSA	Read CS Assertion. This bit field determines when CS signal is asserted during read cycles (synchronous or asynchronous mode), according to the settings shown below. RCSA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of read access and CS assertion 001 1 EIM clock cycles between beginning of read access and CS assertion 010 2 EIM clock cycles between beginning of read access and CS assertion 111 7 EIM clock cycles between beginning of read access and CS assertion
3 Reserved	This read-only field is reserved and always has the value 0.
RCSN	Read CS Negation. This bit field determines when CS signal is negated during read cycles in asynchronous single mode only (SRD=0 & APR = 0), according to the settings shown below. This bit field is ignored when SRD=1. RCSN is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between end of read access and CS negation 001 1 EIM clock cycles between end of read access and CS negation 010 2 EIM clock cycles between end of read access and CS negation 111 7 EIM clock cycles between end of read access and CS negation

9.7.9.4 Chip Select n Read Configuration Register 2 (EIM_CS*n*RCR2)

Address: 30BC_0000h base + Ch offset + (24d × i), where i=0d to 5d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APR	PAT			0		RL		0	RBEA			RBE	RBEN		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_CS*n*RCR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 APR	Asynchronous Page Read. This bit field determine the asynchronous read mode to the external device. When APR=0, the async. read access is done as single word (where word is defined by the DSZ field). when APR=1, the async. read access executed as page read. page size is according to BL field config., RCSN,RBEN,OEN and RADVN are being ignored. APR is cleared by a hardware reset for EIM_CS1GCR1 - EIM_CS5GCR1. **NOTE:** SRD=0 and MUM=0 must apply when APR=1
14–12 PAT	Page Access Time. This bit field is used in Asynchronous Page Read mode only (APR=1). the initial access is set by RWSC as in regular asynchronous mode. the consecutive address assertions width determine by PAT field according to the settings shown below. when APR=0 this field is ignored. PAT is cleared by a hardware reset for EIM_CS1GCR1 - EIM_CS5GCR1. 000 Address width is 2 EIM clock cycles 001 Address width is 3 EIM clock cycles 010 Address width is 4 EIM clock cycles 011 Address width is 5 EIM clock cycles 100 Address width is 6 EIM clock cycles 101 Address width is 7 EIM clock cycles 110 Address width is 8 EIM clock cycles 111 Address width is 9 EIM clock cycles
11–10 Reserved	This read-only field is reserved and always has the value 0.
9–8 RL	Read Latency. This bit field indicates cycle latency when executing a synchronous read operation. The fields holds the feedback clock loop delay in aclk cycle units. This field is cleared by a hardware reset. 00 Feedback clock loop delay is up to 1 cycle for BCD = 0 or 1.5 cycles for BCD != 0 01 Feedback clock loop delay is up to 2 cycles for BCD = 0 or 2.5 cycles for BCD != 0 10 Feedback clock loop delay is up to 3 cycles for BCD = 0 or 3.5 cycles for BCD != 0 11 Feedback clock loop delay is up to 4 cycles for BCD = 0 or 4.5 cycles for BCD != 0

Table continues on the next page...

EIM_CS*n*RCR2 field descriptions (continued)

Field	Description
7 Reserved	This read-only field is reserved and always has the value 0.
6–4 RBEA	Read BE Assertion. This bit field determines when BE signal is asserted during read cycles (synchronous or asynchronous mode), according to the settings shown below. RBEA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of read access and BE assertion 001 1 EIM clock cycles between beginning of read access and BE assertion 010 2 EIM clock cycles between beginning of read access and BE assertion 111 7 EIM clock cycles between beginning of read access and BE assertion
3 RBE	Read BE enable. This bit field determines if BE will be asserted during read access. 0 - BE are disabled during read access. 1- BE are enable during read access according to value of RBEA & RBEN bit fields.
RBEN	Read BE Negation. This bit field determines when BE signal is negated during read cycles in asynchronous single mode only (SRD=0 & APR=0), according to the settings shown below. This bit field is ignored when SRD=1. RBEN is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between end of read access and BE negation 001 1 EIM clock cycles between end of read access and BE negation 010 2 EIM clock cycles between end of read access and BE negation 111 7 EIM clock cycles between end of read access and BE negation

9.7.9.5 Chip Select n Write Configuration Register 1 (EIM_CS*n*WCR1)

Address: 30BC_0000h base + 10h offset + (24d × i), where i=0d to 5d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WAL	WBED	WWSC						WADVA			WADVN			WBEA	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WBEA	WBEN			WEA				WEN			WCSA			WCSN	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_CS*n*WCR1 field descriptions

Field	Description
31 WAL	Write ADV Low. This bit field determine ADV signal negation time in write accesses. When WAL=1, WADVN bit field is ignored and ADV signal will stay asserted until end of access. When WAL=0 negation of ADV signal is according to WADVN bit field configuration.
30 WBED	Write Byte Enable Disable. When asserted this bit prevent from IPP_DO_BE_B[x] to be asserted during write accesses.This bit is cleared by hardware reset.
29–24 WWSC	Write Wait State Control. This bit field programs the number of wait-states, according to the settings shown below, for synchronous or asynchronous write access to the external device connected to the chip select. When SWR=1 and WFL=0, WWSC indicates the number of burst clock (BCLK) cycles from the start of an access, before the memory can sample the first data.Since WAIT signal can be asserted one cycle before the first data can be sampled, the controller starts evaluating the WAIT signal state one cycle before, this is referred as handshake mode or variable latency mode. When SWR=1 and WFL=1, WWSC indicates the number of burst clock (BCLK) cycles from the start of an access, until the external device is ready for data transfer, this is referred as fix latency mode. When SWR=0, WFL bit is ignored, WWSC indicates the asynchronous access length and the number of EIM clock cycles from the start of access until the external device is ready for data transfer. WWSC is cleared by a hardware reset. **NOTE:** The reset value for EIM_CS0WCR1, WWSC[5:0] = 0b011100. For EIM_CS1WCR1 - EIM_CS5WCR1, the reset value of this field is 0b000000. Example settings: 000000 Reserved 000001 WWSC value is 1 000010 WWSC value is 2 000011 WWSC value is 3 111111 WWSC value is 63
23–21 WADVA	ADV Assertion. This bit field determines when ADV signal is asserted for synchronous or asynchronous write modes according to the settings shown below. WADVA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of access and ADV assertion 001 1 EIM clock cycles between beginning of access and ADV assertion 010 2 EIM clock cycles between beginning of access and ADV assertion 111 7 EIM clock cycles between beginning of access and ADV assertion
20–18 WADVN	ADV Negation. This bit field determines when ADV signal to memory is negated during write accesses. When SWR=1 (synchronous write mode), ADV negation occurs according to the following formula: (WADVN + WADVA + BCD + BCS + 1) EIM clock cycles. When asynchronous read mode is applied (SWR=0) ADV negation occurs according to the following formula: (WADVN + WADVA + 1) EIM clock cycles. **NOTE:** Reset value for EIM_CS0WCR for WADVN is 2. For EIM_CS1WCR - EIM_CS5WCR reset value is 000. **NOTE:** This field should be configured so ADV negation will occur before the end of access. For ADV negation at the same time as the end of access, S/W should set the WAL bit.
17–15 WBEA	BE Assertion. This bit field determines when BE signal is asserted during write cycles in async. mode only (SWR=0), according to the settings shown below. BEA is cleared by a hardware reset.

Table continues on the next page...

EIM_CS*n*WCR1 field descriptions (continued)

Field	Description
	NOTE: Reset value for EIM_CS0WCR for WBEA is 2. For EIM_CS1WCR - EIM_CS5WCR reset value is 000. Example settings: 000 0 EIM clock cycles between beginning of access and BE assertion 001 1 EIM clock cycles between beginning of access and BE assertion 010 2 EIM clock cycles between beginning of access and BE assertion 111 7 EIM clock cycles between beginning of access and BE assertion
14–12 WBEN	BE[3:0] Negation. This bit field determines when BE[3:0] bus signal is negated during write cycles in async. mode only (SWR=0), according to the settings shown below. This bit field is ignored when SWR=1. BEN is cleared by a hardware reset. **NOTE:** Reset value for EIM_CS0WCR for WBEN is 2. For EIM_CS1WCR - EIM_CS5WCR reset value is 000. Example settings: 000 0 EIM clock cycles between end of access and WE negation 001 1 EIM clock cycles between end of access and WE negation 010 2 EIM clock cycles between end of access and WE negation 111 7 EIM clock cycles between end of access and WE negation
11–9 WEA	WE Assertion. This bit field determines when WE signal is asserted during write cycles (synchronous or asynchronous mode), according to the settings shown below. This bit field is ignored when executing a read access to the external device. WEA is cleared by a hardware reset. **NOTE:** Reset value for EIM_CS0WCR for WEA is 2. For EIM_CS1WCR - EIM_CS5WCR reset value is 000. Example settings: 000 0 EIM clock cycles between beginning of access and WE assertion 001 1 EIM clock cycles between beginning of access and WE assertion 010 2 EIM clock cycles between beginning of access and WE assertion 111 7 EIMclock cycles between beginning of access and WE assertion
8–6 WEN	WE Negation. This bit field determines when WE signal is negated during write cycles in asynchronous mode only (SWR=0), according to the settings shown below. This bit field is ignored when SWR=1. WEN is cleared by a hardware reset. **NOTE:** Reset value for EIM_CS0WCR for WEN is 2. For EIM_CS1WCR - EIM_CS5WCR reset value is 000. Example settings: 000 0 EIM clock cycles between beginning of access and WE assertion 001 1 EIM clock cycles between beginning of access and WE assertion 010 2 EIM clock cycles between beginning of access and WE assertion 111 7 EIM clock cycles between beginning of access and WE assertion
5–3 WCSA	Write CS Assertion. This bit field determines when CS signal is asserted during write cycles (synchronous or asynchronous mode), according to the settings shown below.this bit field is ignored when executing a read access to the external device. WCSA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of write access and CS assertion

Table continues on the next page...

EIM_CS*n*WCR1 field descriptions (continued)

Field	Description
	001 1 EIM clock cycles between beginning of write access and CS assertion 010 2 EIM clock cycles between beginning of write access and CS assertion 111 7 EIMclock cycles between beginning of write access and CS assertion
WCSN	Write CS Negation. This bit field determines when CS signal is negated during write cycles in asynchronous mode only (SWR=0), according to the settings shown below. This bit field is ignored when SWR=1. WCSN is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between end of read access and CS negation 001 1 EIM clock cycles between end of read access and CS negation 010 2 EIM clock cycles between end of read access and CS negation 111 7 EIM clock cycles between end of read access and CS negation

9.7.9.6 Chip Select n Write Configuration Register 2 (EIM_CS*n*WCR2)

Address: 30BC_0000h base + 14h offset + (24d × i), where i=0d to 5d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

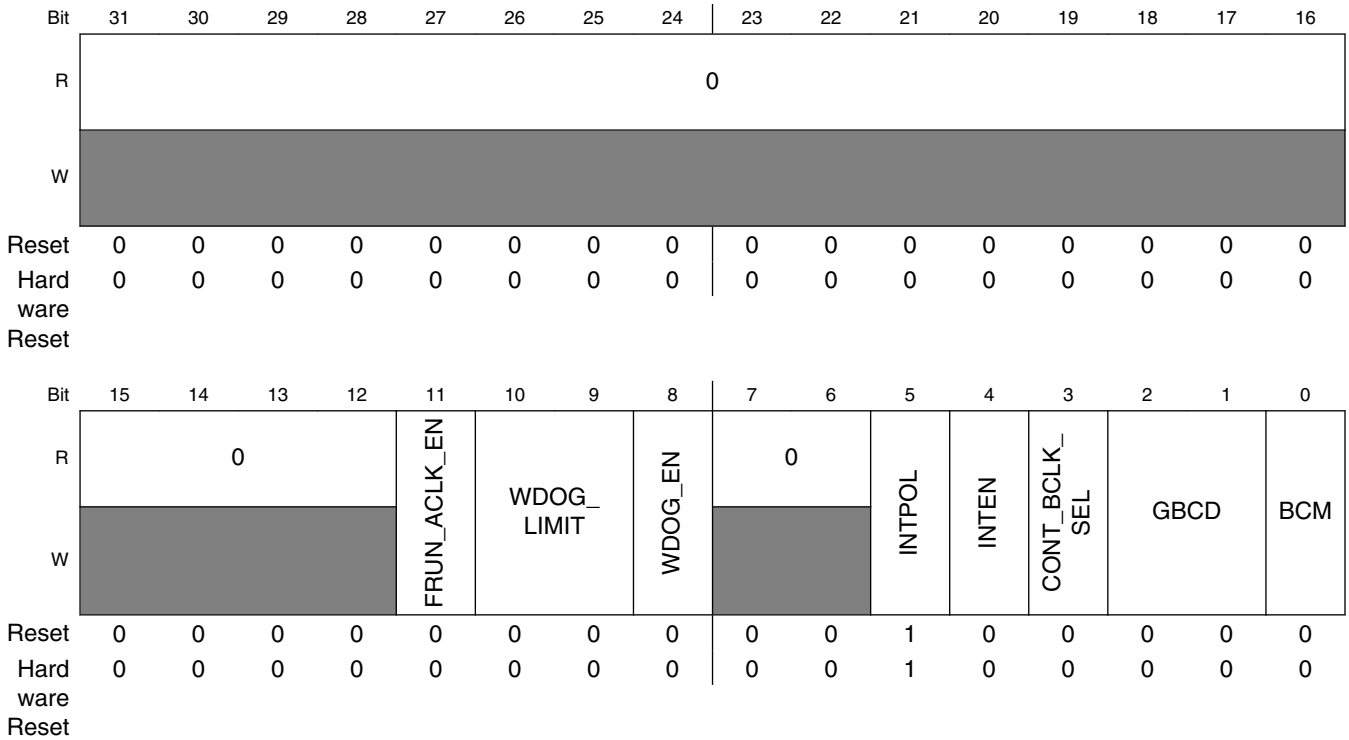
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															WBCDD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_CS*n*WCR2 field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 WBCDD	Write Burst Clock Divisor Decrement. If this bit is asserted and BCD value is 0 sync. write access will be preformed as if BCD value is 1.When this bit is negated or BCD value is not 0 this bit has no affect. This bit is cleared by hardware reset.

9.7.9.7 EIM Configuration Register (EIM_WCR)

Address: 30BC_0000h base + 90h offset = 30BC_0090h



EIM_WCR field descriptions

Field	Description
31–12 Reserved	Reserved This read-only field is reserved and always has the value 0.
11 FRUN_ACLK_EN	Free run ACLK enable
10–9 WDOG_LIMIT	Memory Watchdog (WDOG) cycle limit. This bit field determines the number of BCLK cycles (ACLK cycles in dtack mode) before the WDOG counter terminates the access and send an error response to the master. 00 128 BCLK cycles 01 256 BCLK cycles 10 512 BCLK cycles 11 1024 BCLK cycles
8 WDOG_EN	Memory WDOG enable. This bit controls the operation of the wdog counter that terminates the EIM access. 0 Memory WDOG is Disabled 1 Memory WDOG is Enabled

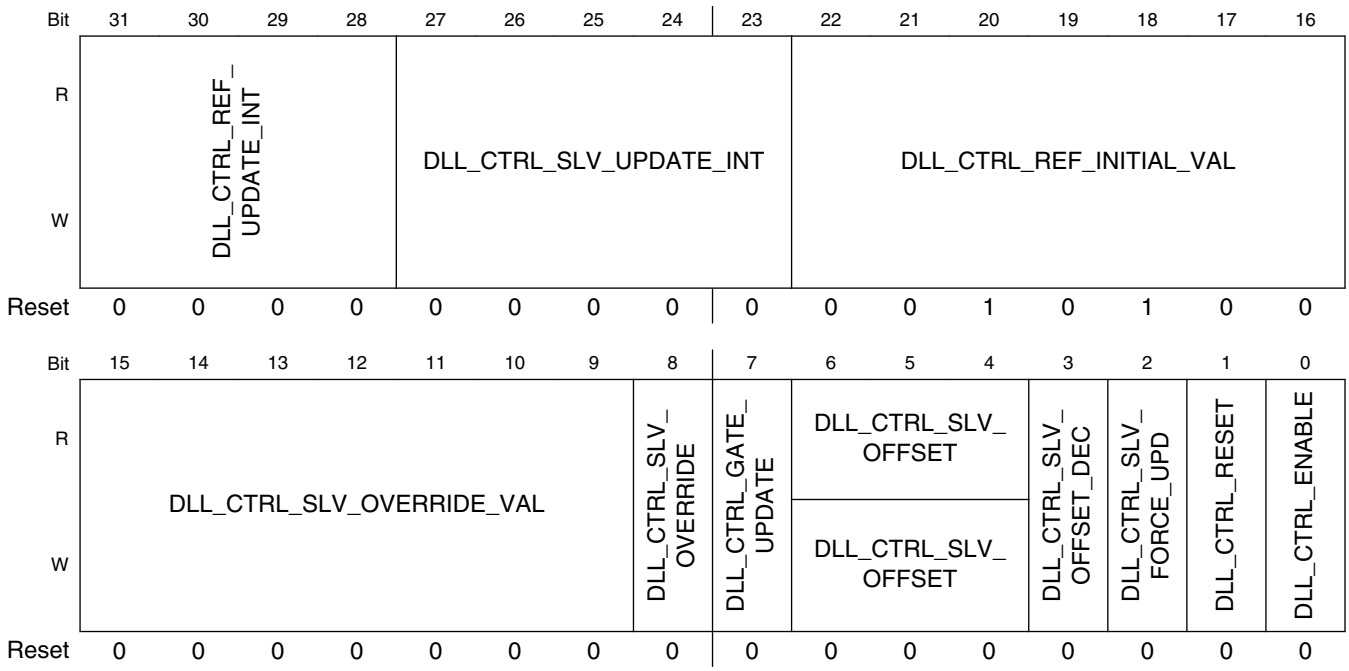
Table continues on the next page...

EIM_WCR field descriptions (continued)

Field	Description
7–6 Reserved	This read-only field is reserved and always has the value 0.
5 INTPOL	Interrupt Polarity. This bit field determines the polarity of the external device interrupt. 0 External interrupt polarity is active low 1 External interrupt polarity is active high
4 INTEN	Interrupt Enable. When this bit is set the External signal RDY_INT as active interrupt. When interrupt occurs, INT bit at the WCR will be set and t EIM_EXT_INT signal will be asserted correspondingly. This bit is cleared by a hardware reset. 0 External interrupt Disable 1 External interrupt Enable
3 CONT_BCLK_ SEL	Continuous BCLK select When this bit is set BCLK pin output continuous clock. Otherwize, BCLK will output clock only when nesserary. 0 BCLK When nesserary 1 BCLK Continuous
2–1 GBCD	General Burst Clock Divisor. When BCM bit is set, this bit field contains the value used to program the burst clock divisor for Continuous BCLK generation. The other BCD bit fields for each chip select are ignored. It is used to divide the internal AXI bus frequency. When BCM=0 GBCD bit field has no influence. GBCD is cleared by a hardware reset. 00 Divide EIM clock by 1 01 Divide EIM clock by 2 10 Divide EIM clock by 3 11 Divide EIM clock by 4
0 BCM	Burst Clock Mode. This bit selects the burst clock mode of operation. It is used for system debug mode. BCM is cleared by a hardware reset. **NOTE:** The BCLK frequency in this mode is according to GBCD bit field. **NOTE:** The BCLK phase is opposite to the EIM clock in this mode if GBCD is 0. **NOTE:** This bit should be used only in async. accesses. No sync access can be executed if this bit is set. **NOTE:** When this bit is set bcd field shouldn't be configured to 0. 0 The burst clock runs only when accessing a chip select range with the SWR/SRD bits set. When the burst clock is not running it remains in a logic 0 state. When the burst clock is running it is configured by the BCD and BCS bit fields in the chip select Configuration Register. 1 The burst clock runs whenever ACLK is active (independent of chip select configuration)

9.7.9.8 DLL Control Register (EIM_DCR)

Address: 30BC_0000h base + 94h offset = 30BC_0094h



EIM_DCR field descriptions

Field	Description
31–28 DLL_CTRL_REF_UPDATE_INT	Reference DLL Update Interval DLL control loop update interval. The interval cycle is (2 + REF_UPDATE_INT) * ref_clock. By default, the DLL control loop shall update every two ref_clock cycles. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature)
27–23 DLL_CTRL_SLV_UPDATE_INT	Slave DLL Update Interval If default 0 is used, it means 256 cycles of ref_clock. A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line will also update automatically when the reference DLL transitions to a locked state (from an un-locked state).
22–16 DLL_CTRL_REF_INITIAL_VAL	This field is used to select the initial value of reference chain before DLL enabled. It's recommended to set the initial value close to the locked value to accelerate the locking.
15–9 DLL_CTRL_SLV_OVERRIDE_VAL	When SLV_OVERRIDE=1 This field is used to select 1 of 128 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 128.
8 DLL_CTRL_SLV_OVERRIDE	Set this bit to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to be enabled using the ENABLE bit. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE=0

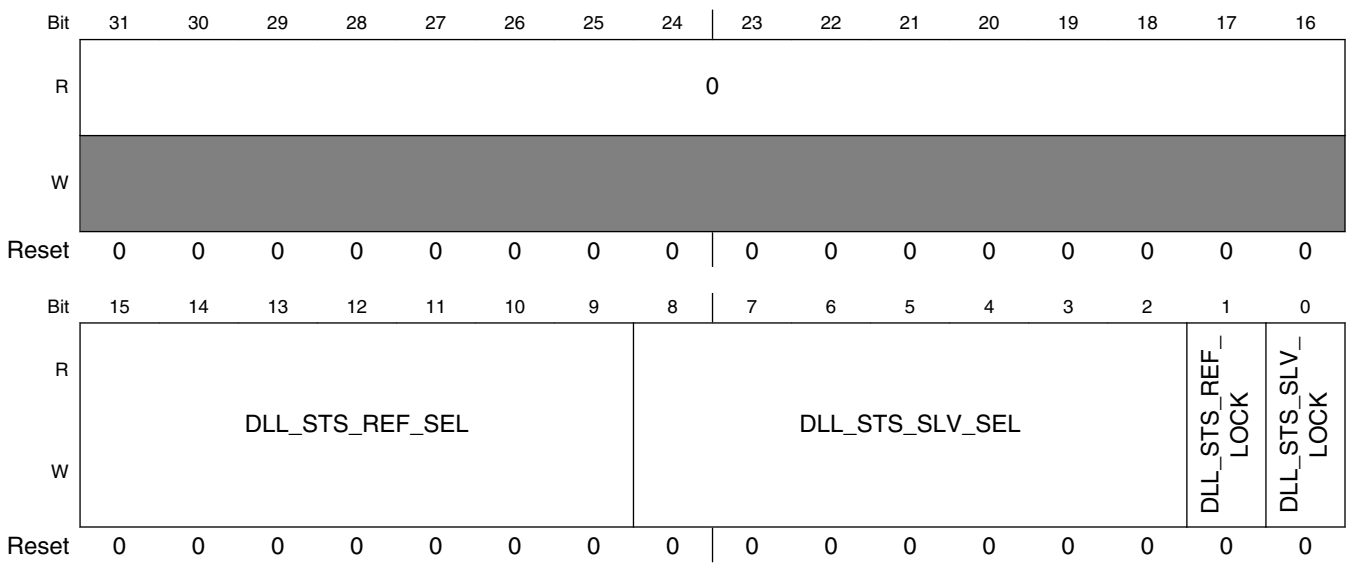
Table continues on the next page...

EIM_DCR field descriptions (continued)

Field	Description
7 DLL_CTRL_ GATE_UPDATE	Set this bit to 1 to force DLL not update from now on. Since when clock exists, glitches might appear during update. This bit is used by software if we met such kind of condition. Set it to 0 to let DLL update automatically
6–4 DLL_CTRL_ SLV_OFFSET	OFFSET value for DLL_CTRL_SLV_SEL
3 DLL_CTRL_ SLV_OFFSET_ DEC	Slave Chain Offset Decrease Decrease(or increase) the value defined by DLL_CTRL_SLV_OFFSET when calculating DLL_STS_SLV_SEL 0 DLL_STS_SLV_SEL = DLL_STS_REF_SEL + DLL_CTRL_SLV_OFFSET 1 DLL_STS_SLV_SEL = DLL_STS_REF_SEL - DLL_CTRL_SLV_OFFSET
2 DLL_CTRL_ SLV_FORCE_ UPD	Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line shall update automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered). Be sure to use it when EIM is idle.
1 DLL_CTRL_ RESET	DLL Reset Bit Setting this bit to 1 force a reset on DLL. This will cause the DLL to lose lock and re-lock. This signal is used by the DLL as edge-sensitive, so in order to create a subsequent reset, RESET must be taken low and then asserted again
0 DLL_CTRL_ ENABLE	Set this bit to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled

9.7.9.9 DLL Status Register (EIM_DSR)

Address: 30BC_0000h base + 98h offset = 30BC_0098h

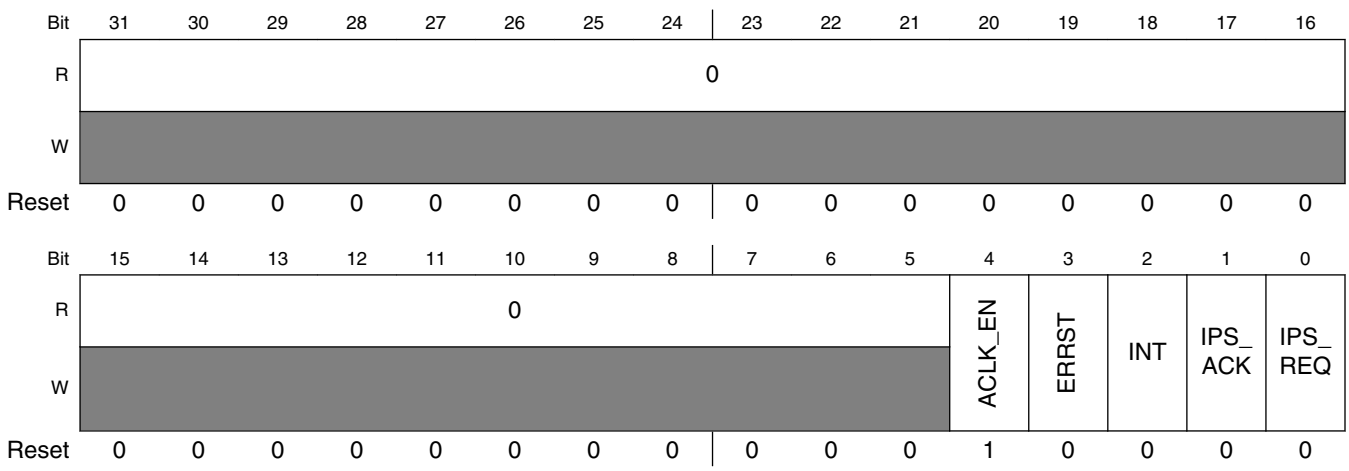


EIM_DSR field descriptions

Field	Description
31–16 Reserved	Reserved This read-only field is reserved and always has the value 0.
15–9 DLL_STS_REF_SEL	Reference delay line select taps. Be noted this is encoded by 7 bits for 127taps.
8–2 DLL_STS_SLV_SEL	Slave delay line select status. This is the instant value generated from reference chain. Since only when ref_lock is detected can the reference chain get updated, this value should be the right value next be update to the slave line when reference is locked.
1 DLL_STS_REF_LOCK	Reference DLL lock status. This signifies that the DLL has detected and locked to feedback BCLK, allowing the slave delay-line to perform programmed clock delays
0 DLL_STS_SLV_LOCK	Slave delay-line lock status. This signifies that a valid delay has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value

9.7.9.10 EIM IP Access Register (EIM_WIAR)

Address: 30BC_0000h base + 9Ch offset = 30BC_009Ch



EIM_WIAR field descriptions

Field	Description
31–5 Reserved	This read-only field is reserved and always has the value 0.
4 ACLK_EN	ACLK enable. This bit gates the ACLK for the EIM except from FFs that get ipg_aclk_s. After reset ACLK is enabled. 0 ACLK is disabled 1 ACLK is enabled

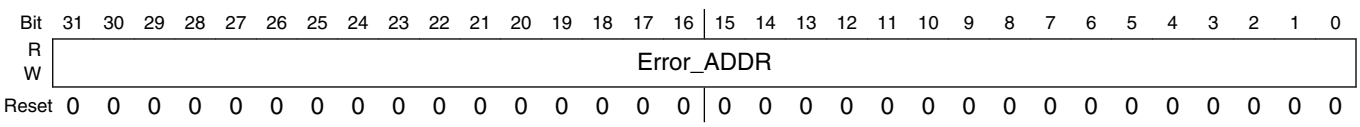
Table continues on the next page...

EIM_WIAR field descriptions (continued)

Field	Description
3 ERRST	READY After Reset. This bit controls the initial ready/busy status for external devices on CS0 immediately after hardware reset. This is a sticky bit which is cleared once the RDY_INT signal is asserted by the external device. When ERRST = 1 the first fetch access from EIM to the external device located on CS0 will be pending until RDY_INT signal indicates that the external device is ready, then EIM will execute the access. **NOTE:** Reset value for ERRST is EIM_BOOT[4]. 0 RDY_INT After Reset Disable 1 RDY_INT After Reset Enable
2 INT	Interrupt. This bit indicates interrupt assertion by an external device according to RDY_INT signal. When polling this bit, INT=0 indicates interrupt not occurred and INT=1 indicates assertion of the external device interrupt. This bit is cleared by a hardware reset.
1 IPS_ACK	IPS ACK. The EIM is ready for ips access. There is no active AXI access and no new AXI access is accepted till this bit is cleared. This bit is cleared by the master after it completes the ips accesses. 0 Master cannot access ips. 1 Master can access ips.
0 IPS_REQ	IPS request. The Master requests to access one of the IPS registers.Dutring such access the EIM should not preform any AXI/memory accesses. The EIM finishes the AXI accesses that already starts and asserts the IPS_ACK bit. 0 No Master requests ips access 1 Master requests ips access

9.7.9.11 Error Address Register (EIM_EAR)

Address: 30BC_0000h base + A0h offset = 30BC_00A0h



EIM_EAR field descriptions

Field	Description
Error_ADDR	Error Address. This bit field holds the AXI address of the last access that caused error. This register is read only register.

Chapter 10
Mass Storage

10.1 Enhanced Configurable SPI (ECSPI)

10.1.1 Overview

The Enhanced Configurable Serial Peripheral Interface (ECSPI) is a full-duplex, synchronous, four-wire serial communication block.

The ECSPI contains a 64 x 32 receive buffer (RXFIFO) and a 64 x 32 transmit buffer (TXFIFO). With data FIFOs, the ECSPI allows rapid data communication with fewer software interrupts. The figure below shows a block diagram of the ECSPI.

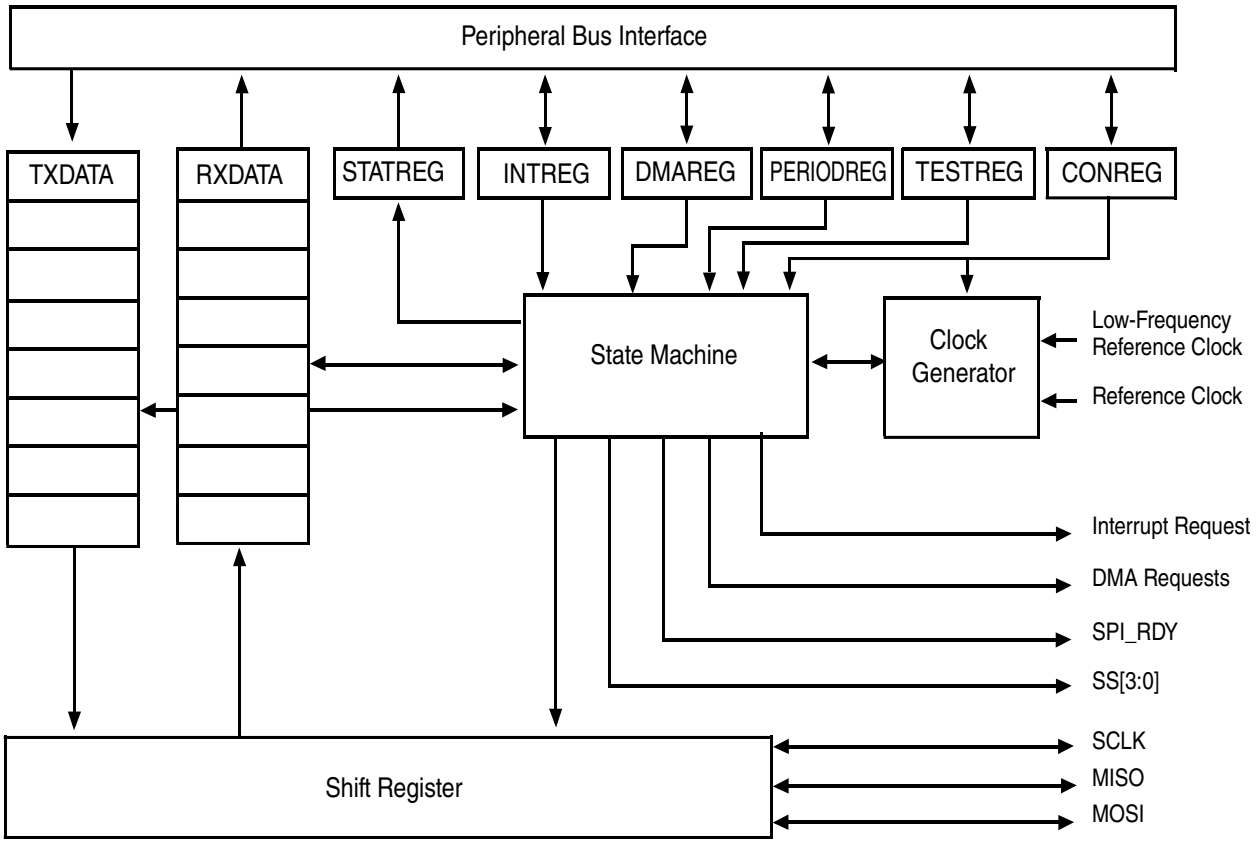


Figure 10-1. ECSPI Block Diagram

10.1.1.1 Features

Key features of the ECSPI include:

- Full-duplex synchronous serial interface
- Master/Slave configurable
- Four Chip Select (SS) signals to support multiple peripherals
- Transfer continuation function allows unlimited length data transfers
- 32-bit wide by 64-entry FIFO for both transmit and receive data
- Polarity and phase of the Chip Select (SS) and SPI Clock (SCLK) are configurable
- Direct Memory Access (DMA) support
- Max operation frequency up to the reference clock frequency.

10.1.1.2 Modes and Operations

The ECSPI supports the modes described in the indicated sections:

- Master Mode
- Slave Mode
- Low Power Modes

As described in Operations, the ECSPI supports the operations described in the indicated sections:

- Typical Master Mode
  - Master Mode with SPI_RDY
  - Master Mode with Wait States
  - Master Mode with SS_CTL[3:0] Control
  - Master Mode with Phase Control
- Typical Slave Mode

10.1.2 External Signals

Table 10-1. eCSPI External Signals

Signal	Description	Pad	Mode	Direction
ECSPI1_MISO	Master data in; slave data out	ECSPI1_MISO	ALT0	IO
		UART3_RXD	ALT3	
ECSPI1_MOSI	Master data out; slave data in	ECSPI1_MOSI	ALT0	IO
		UART3_TXD	ALT3	
ECSPI1_RDY	SPI data ready signal	UART2_TXD	ALT3	I
ECSPI1_SCLK	SPI clock signal	ECSPI1_SCLK	ALT0	IO
		UART3_RTS	ALT3	
ECSPI1_SS0	Chip select signal	ECSPI1_SS0	ALT0	IO
		UART3_CTS	ALT3	
ECSPI1_SS1	Chip select signal	UART1_RXD	ALT3	IO
ECSPI1_SS2	Chip select signal	UART1_TXD	ALT3	IO
ECSPI1_SS3	Chip select signal	UART2_RXD	ALT3	IO
ECSPI2_MISO	Master data in; slave data out	ECSPI2_MISO	ALT0	IO
		ENET1_TDATA2	ALT2	
ECSPI2_MOSI	Master data out; slave data in	ECSPI2_MOSI	ALT0	IO
		ENET1_RDATA3	ALT2	
ECSPI2_RDY	SPI data ready signal	ENET1_TDATA1	ALT2	I
ECSPI2_SCLK	SPI clock signal	ECSPI2_SCLK	ALT0	IO
		ENET1_RDATA2	ALT2	
ECSPI2_SS0	Chip select signal	ECSPI2_SS0	ALT0	IO

Table continues on the next page...

Table 10-1. eCSPI External Signals (continued)

Signal	Description	Pad	Mode	Direction
	Chip select signal	ENET1_TDATA3	ALT2	
ECSPI2_SS1	Chip select signal	ENET1_RX_CTL	ALT2	IO
ECSPI2_SS2	Chip select signal	ENET1_RXC	ALT2	IO
ECSPI2_SS3	Chip select signal	ENET1_TDATA0	ALT2	IO
ECSPI3_MISO	Master data in; slave data out	I2C1_SCL	ALT3	IO
		SAI2_TXFS	ALT1	
ECSPI3_MOSI	Master data out; slave data in	I2C1_SDA	ALT3	IO
		SAI2_TXC	ALT1	
ECSPI3_RDY	SPI data ready signal	SD2_RESET_B	ALT3	I
ECSPI3_SCLK	SPI clock signal	I2C2_SCL	ALT3	IO
		SAI2_RXD	ALT1	
ECSPI3_SS0	Chip select signal	I2C2_SDA	ALT3	IO
	Chip select signal	SAI2_TXD	ALT1	
ECSPI3_SS1	Chip select signal	SD1_DATA3	ALT3	IO
ECSPI3_SS2	Chip select signal	SD2_CD_B	ALT3	IO
ECSPI3_SS3	Chip select signal	SD2_WP	ALT3	IO
ECSPI4_MISO	Master data in; slave data out	LCD1_CLK	ALT1	IO
		SD1_CD_B	ALT3	
		SD3_CLK	ALT2	
ECSPI4_MOSI	Master data out; slave data in	LCD1_ENABLE	ALT1	IO
		SD1_WP	ALT3	
		SD3_CMD	ALT2	
ECSPI4_RDY	SPI data ready signal	SD1_DATA2	ALT3	I
ECSPI4_SCLK	SPI clock signal	LCD1_HSYNC	ALT1	IO
		SD1_RESET_B	ALT3	
		SD3_DATA1	ALT2	
ECSPI4_SS0	Chip select signal	LCD1_VSYNC	ALT1	IO
		SD1_CLK	ALT3	
		SD3_DATA0	ALT2	
ECSPI4_SS1	Chip select signal	SD1_CMD	ALT3	IO
ECSPI4_SS2	Chip select signal	SD1_DATA0	ALT3	IO
ECSPI4_SS3	Chip select signal	SD1_DATA1	ALT3	IO

Figure 10-2 shows the ECSPI in master mode connected to four external devices in a one-way communication link.

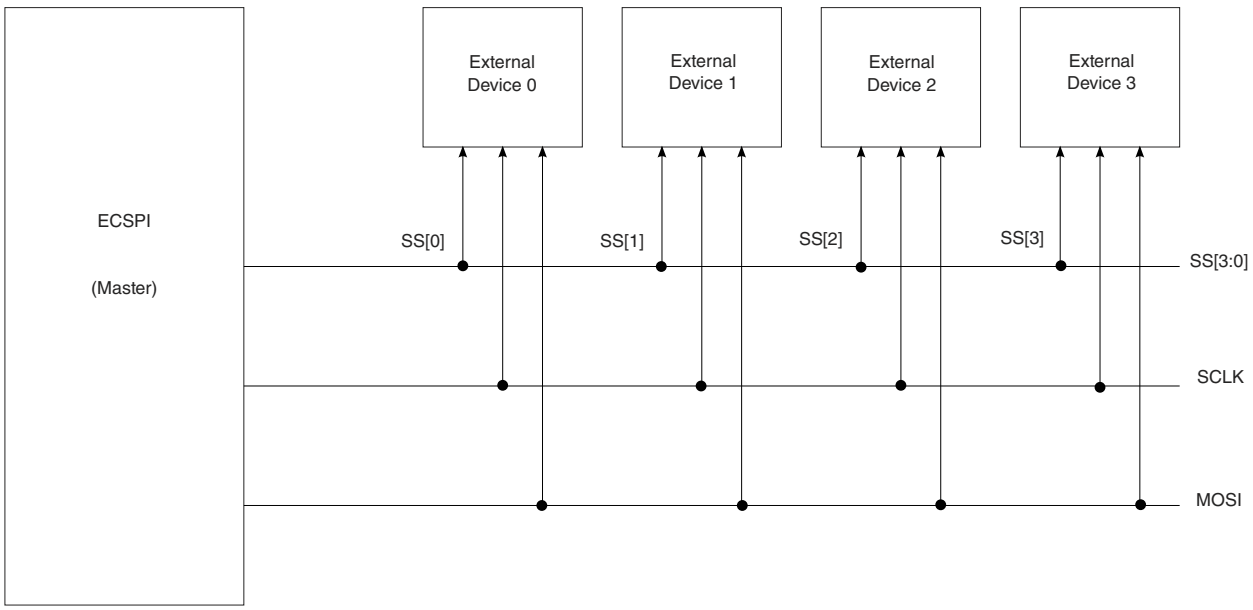


Figure 10-2. Example Connection Diagram

10.1.3 Clocks

The following table describes the clock sources for eCSPI. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 10-2. eCSPI Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_32k	ckil_sync_clk_root	Low-frequency reference clock (32kHz)
ipg_clk_per	ecspi_clk_root	eCSPI module clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

10.1.4 Functional Description

This section provides a complete functional description of the ECSPI. The figure found here shows the relationship of SCLK and data lines while ECSPI has been configured with different POL and PHA settings.

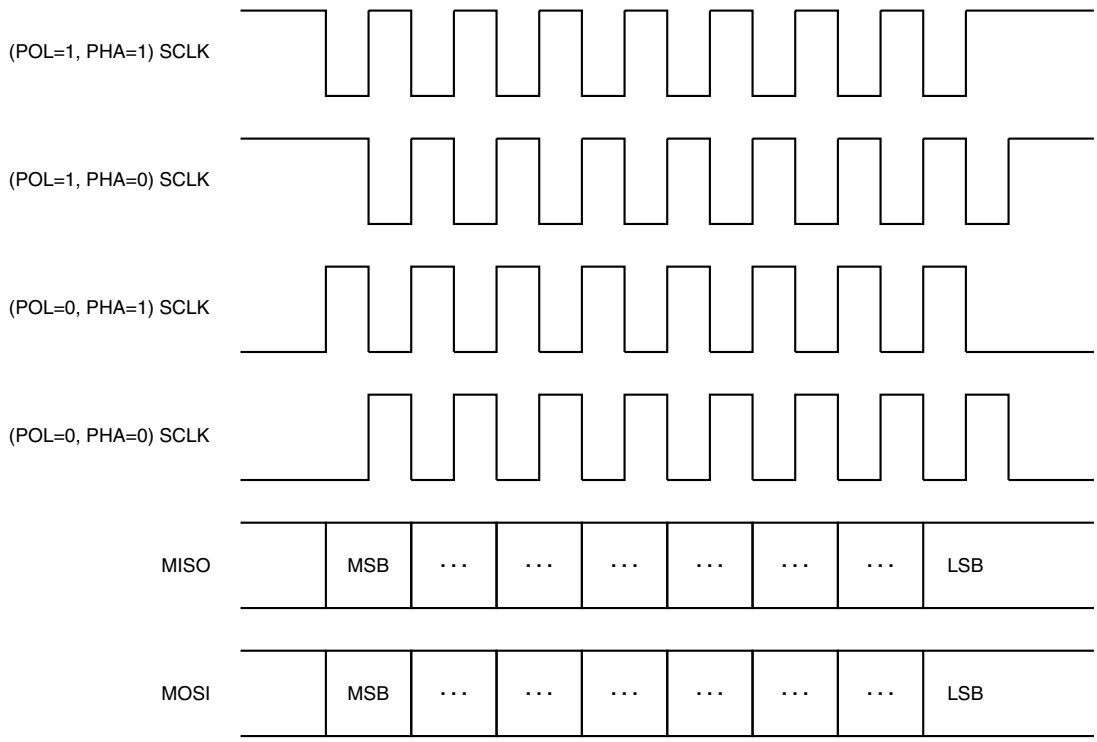


Figure 10-3. ECSPI SCLK, MISO, and MOSI Relationship

10.1.4.1 Master Mode

When the ECSPI is configured as a master, it uses a serial link to transfer data between the ECSPI and an external slave device.

One of the Chip Select (SS) signals and the clock signal (SCLK) are used to transfer data between two devices. If the external device is a transmit-only device, the ECSPI master's output port can be ignored and used for other purposes. In order to use the internal TXFIFO and RXFIFO, two auxiliary output signals, Chip Select (SS) and SPI_RDY, are used for data transfer rate control. Software can also configure the sample period control register to a fixed data transfer rate.

10.1.4.2 Slave Mode

When the ECSPI is configured as a slave, software can configure the ECSPI Control register to match the external SPI master's timing.

In this configuration, Chip Select ($\overline{SS}$) becomes an input signal, and is used to control data transfers through the Shift register, as well as to load/store the data FIFO.

Slave mode only supports the case when ECSPIx_CONFIGREG[SS_CTL] is cleared. The accurate burst length should always be specified using the BURST_LENGTH parameter. ECSPIx_CONFIGREG[SS_CTL] set to 1 is not supported in slave mode.

10.1.4.3 Low Power Modes

The ECSPI does not operate under low power mode.

It holds its operation when its clock is gated off in master mode. In slave mode, the ECSPI does not respond when its clock is gated off.

10.1.4.4 Operations

The information found here describes the ECSPI's operations.

10.1.4.4.1 Typical Master Mode

The ECSPI master uses the Chip Select (SS) signal to enable an external SPI device, and uses the SCLK signal to transfer data in and out of the Shift register.

The SPI_RDY enables fast data communication with fewer software interrupts. By programming the ECSPI_PERIODREG register accordingly, the ECSPI can be used for a fixed data transfer rate.

When the ECSPI is in Master mode the SS, SCLK, and MOSI are output signals, and the MISO signal is an input.

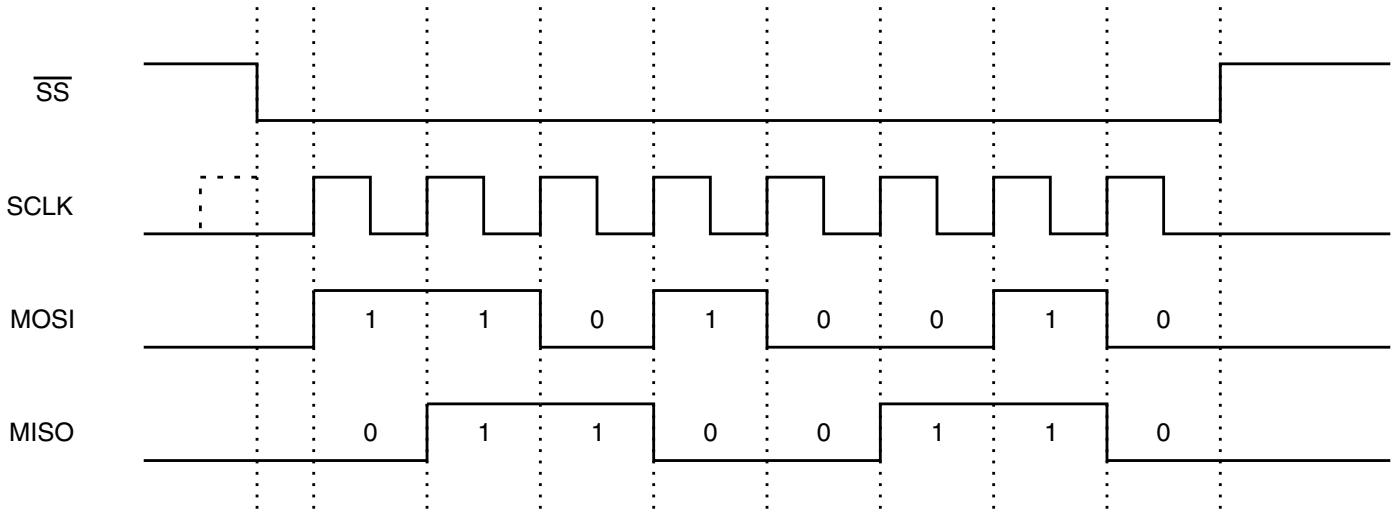


Figure 10-4. Typical SPI Burst (8-bit Transfer)

In the above figure, the Chip Select (SS) signal enables the selected external SPI device, and the SCLK synchronizes the data transfer. The MOSI and MISO signals change on rising edge of SCLK and the MISO signal is latched on the falling edge of the SCLK. The figure above shows a data of 0xD2 is shifted out, and a data of 0x66 is shifted in.

10.1.4.4.1.1 Master Mode with SPI_RDY

By default, the ECSPI does not use the SPI_RDY signal in master mode (MODE =1).

A SPI burst begins when the following events happen:

- The ECSPI is enabled, TXFIFO has data in it, and ECSPI_CONREG[XCH] bit or the ECSPI_CONREG[SMC] bit is set.
- When the SPI Data Ready Control (ECSPI_CONREG[DRCTL]) bits contains either 01 or 10, the SPI_RDY signal controls when a SPI burst starts.

A SPI burst is defined as a bus transaction that starts when the slave select is asserted and ends when the slave select is negated. The Chip Select (SS) signal will remain asserted until all the bits in a SPI burst are shifted out.

If ECSPI_CONREG[DRCTL] is set to 01, the SPI burst can be triggered only if a falling edge of the SPI_RDY signal has been detected.

The following figure shows the relationship between a SPI burst and the falling edge of SPI_RDY signal.

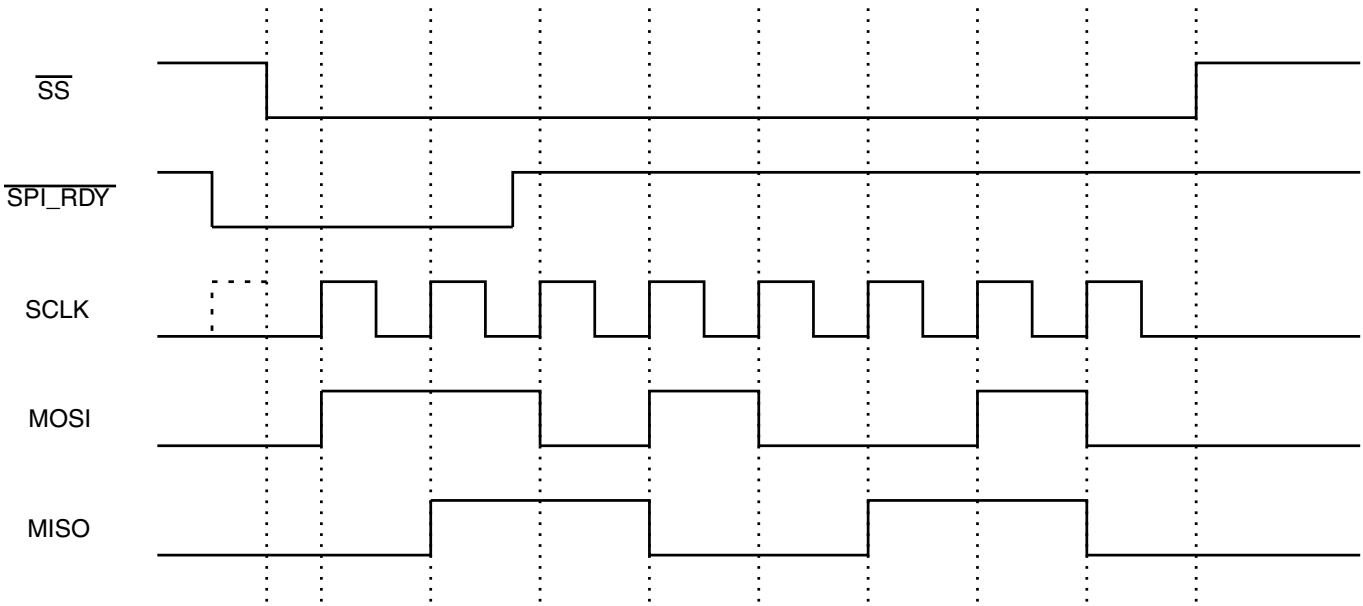


Figure 10-5. Relationship Between a SPI Burst and SPI_RDY: Falling-Edge Triggered

A SPI burst does not start until the falling edge of the SPI_RDY signal is detected. The next SPI burst starts when the next SPI_RDY falling edge is detected, after the last burst has finished.

If SPI Data Ready Control (ECSPI_CONREG[DRCTL]) is set to 10, the SPI burst can be triggered only if the SPI_RDY signal is low.

The following figure shows the relationship between a SPI burst and the SPI_RDY signal. The SPI burst does not begin until the SPI_RDY signal goes low. The ECSPI will keep transmitting SPI burst if the SPI_RDY signal remains low.

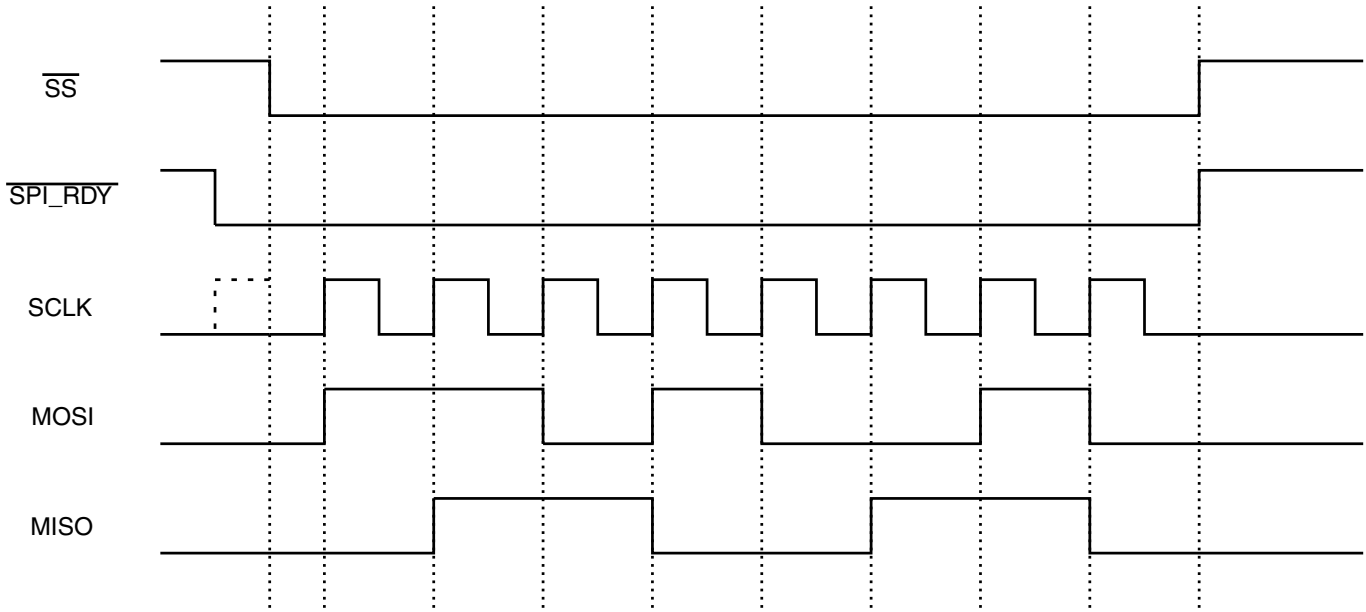


Figure 10-6. Relationship Between a SPI Burst and SPI_RDY: Low-Level Triggered

10.1.4.4.1.2 Master Mode with Wait States

Wait states can be inserted between SPI bursts. This provides a way for software to slow down the SPI burst to meet the timing requirements of a slower SPI device.

The following figure shows wait states inserted between SPI bursts.

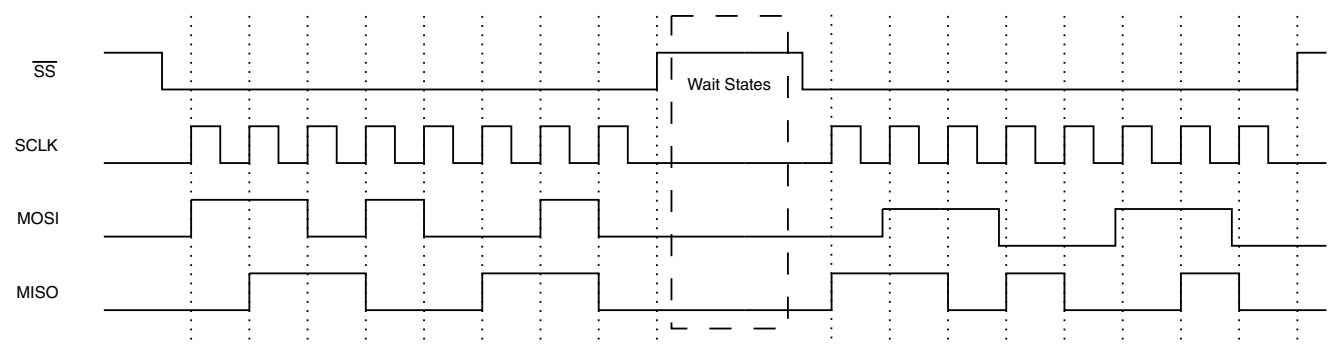


Figure 10-7. SPI Bursts with Wait States

In this case, the number of wait states is controlled by ECSPI_PERIODREG[SAMPLE PERIOD] and the wait states' clock source is selected by ECSPI_PERIODREG[CSRC].

10.1.4.4.1.3 Master Mode with SS_CTL[3:0] Control

The SPI SS Control (SS_CTL[3:0]) controls whether the current operation is single burst or multiple bursts.

When the SPI SS Wave Form Select (SS_CTL[3:0]) is set, the current operation is multiple bursts transfer. When the SPI SS Wave Form Select (SS_CTL[3:0]) bit is cleared, the current operation is single burst transfer. A SPI burst can contains multiple words as defined in the BURST LENGTH field of the ECSPI_CONREG register.

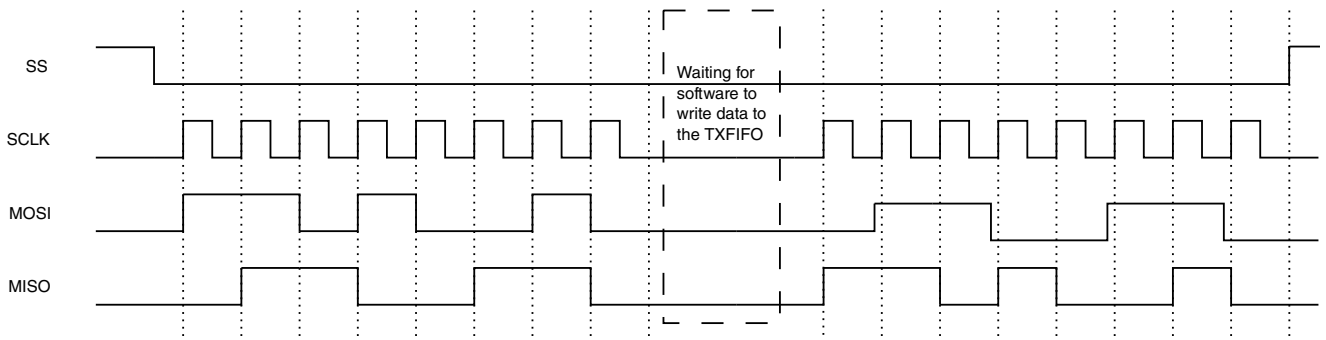


Figure 10-8. SPI Burst While SS_CTL[3:0] is Clear

In Figure 10-8, two 8-bit bursts in the TXFIFO have been combined and transmitted in one SPI burst. The maximum length of a single SPI burst is defined in the BURST LENGTH field of the ECSPI_CONREG control register. (Figure 10-8 corresponds to a BURST LENGTH of 8.) This provides a way for transferring a longer SPI burst by writing data into TXFIFO while the ECSPI is transmitting.

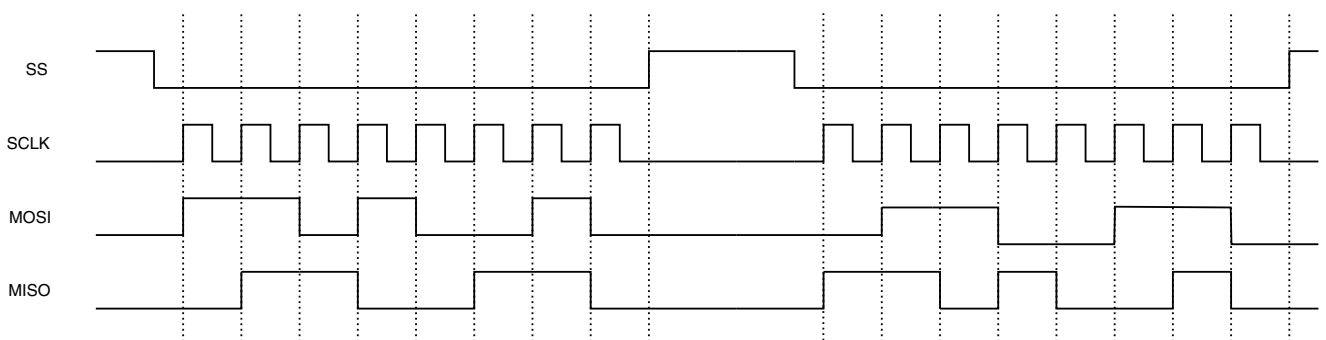


Figure 10-9. SPI Bursts While SS_CTL[3:0] is Set

In Figure 10-9, two FIFO entries are transmitted, one entry with each SPI burst. The ECSPI will continue to transmit SPI bursts until the TXFIFO is empty. When wait states can be inserted between SPI bursts, the SS will negate between SPI bursts until the wait states finish.

10.1.4.4.1.4 Master Mode with Phase Control

The Phase Control (ECSPI_CONREG[PHA]) bit controls how the transmit data shifts out and the receive data shifts in.

When the Phase control (ECSPI_CONREG[PHA]) bit is set, the transmit data will shift out on the rising edge of SCLK, and the receive data is latched on the falling edge of SCLK. The most-significant bit is output on the first rising SCLK edge.

When ECSPI_CONREG[PHA] is cleared, the transmit data is shifted out on the falling edge of SCLK and the receive data is latched on the rising edge of SCLK. The MSB is output when the host processor loads the transmitted data.

Inverting the SCLK polarity does not impact the edge-triggered operations because they are internal to the serial peripheral interface master. Figure 10-10 shows how SPI burst works with different POL and PHA configuration.

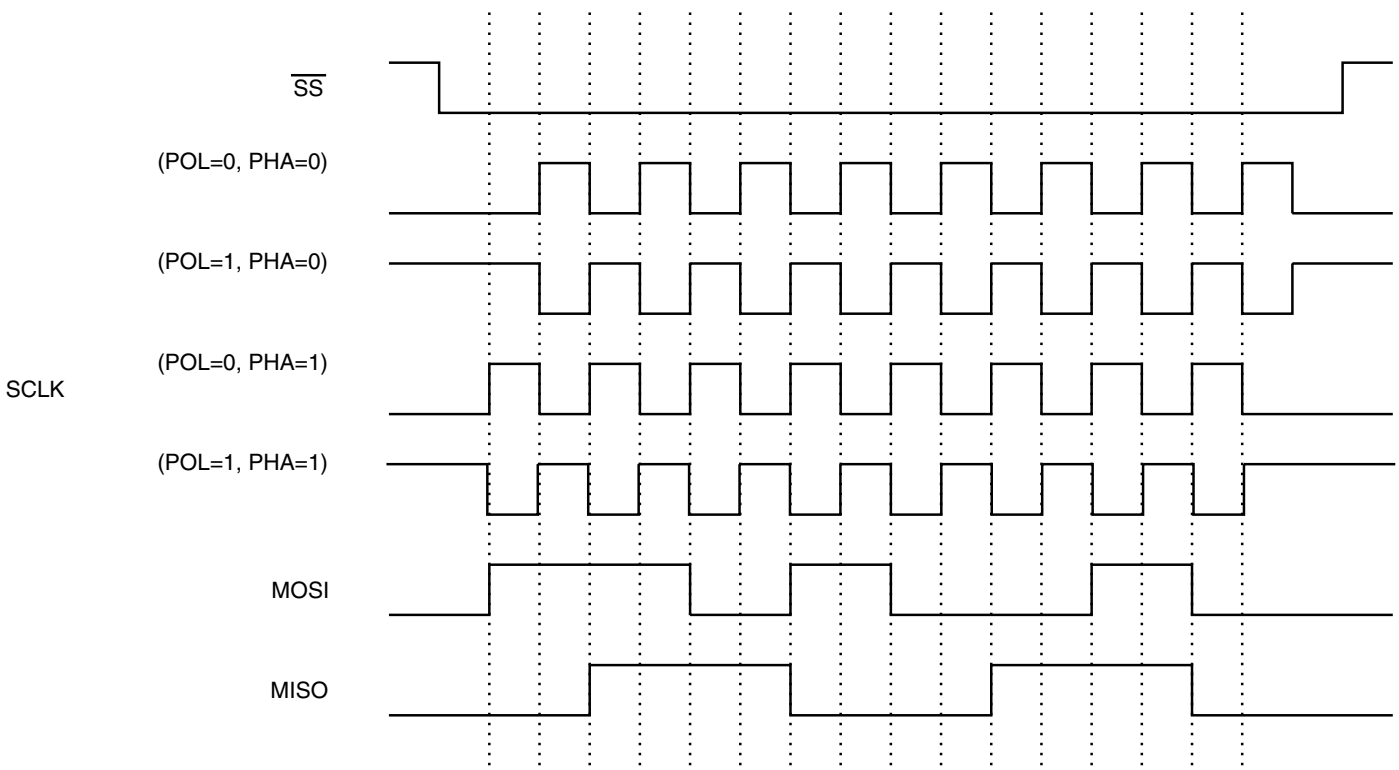


Figure 10-10. SPI Burst with Different POL and PHA Configurations

10.1.4.4.2 Typical Slave Mode

When the ECSPI is configured as a slave (Mode = 0), software can configure the ECSPI Control register to match the external SPI master's timing. In this configuration, SS becomes an input signal, and is used to latch data in and out of the internal data Shift registers, as well as to advance the data FIFO.

The SS, SCLK, and MOSI are inputs and MISO is output. Most of the timing diagrams are similar to the diagrams shown previously for the SPI in Master mode (Mode = 1), because the inputs come from a SPI master device.

However, the timing is different when SS is used to advance the data FIFO. When the SS_POL=0 while the ECSPI is configured in Slave mode, the data FIFO will advance on the rising edge of the SS signal. When the polarity is reversed (SS_POL = 1), the data FIFO will advance on the falling edge of the SS signal.

The figure below shows a SPI burst in which the data FIFO is advanced by the rising edge of the SS signal.

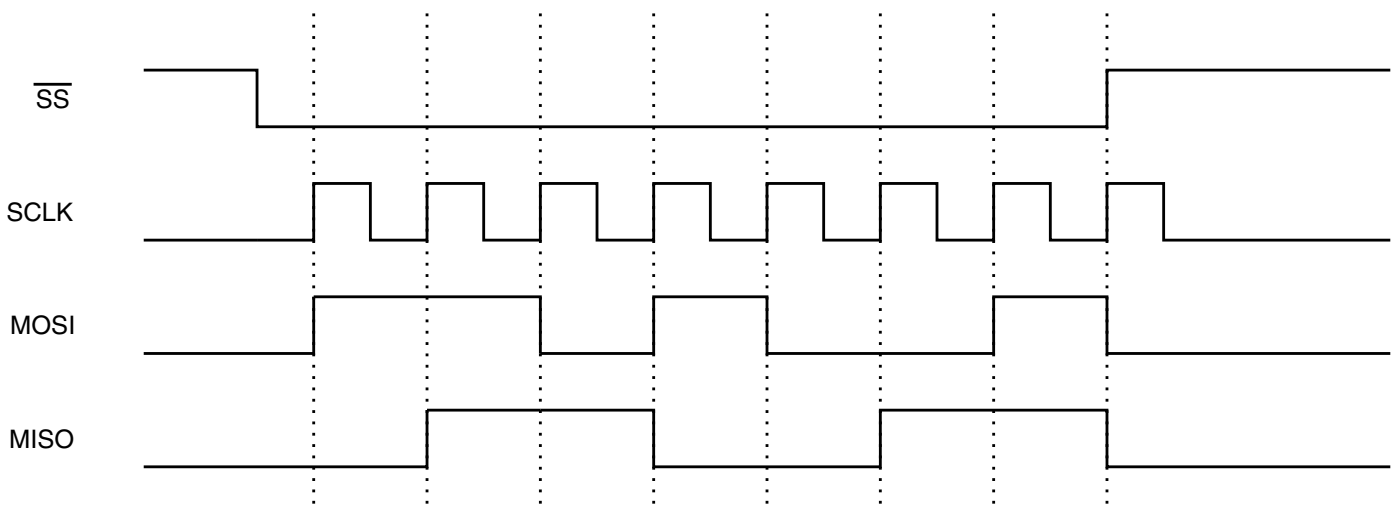


Figure 10-11. Advancing the Data FIFO on the Rising Edge of SS

In the above case, only the most significant 7 bits are loaded to the RXFIFO.

10.1.4.5 Reset

Whenever a device reset occurs, a reset is performed on the ECSPI, resetting all registers to their default values.

Software can reset the block using the CONREG[EN] bit; see ECSPI.

10.1.4.6 Interrupts

Interrupt control provides a way to manage the ECSPI FIFOs:

- For transmitting data, software can enable the TXFIFO empty, TXFIFO data request, and TXFIFO full interrupts to maintain the TXFIFO using an interrupt service routine.
- For receiving data, software can enable the RXFIFO ready, RXFIFO data request, and RXFIFO full interrupts to retrieve data from the RXFIFO using an interrupt service routine.

Other interrupt sources can be used to control or debug the SPI bursts:

- The transfer-completed interrupt means that there is no data left in the TXFIFO and that the data in the Shift register has been shifted out.
- The RXFIFO overflow interrupt means that the RXFIFO received more than 64 words and will not accept any other words.

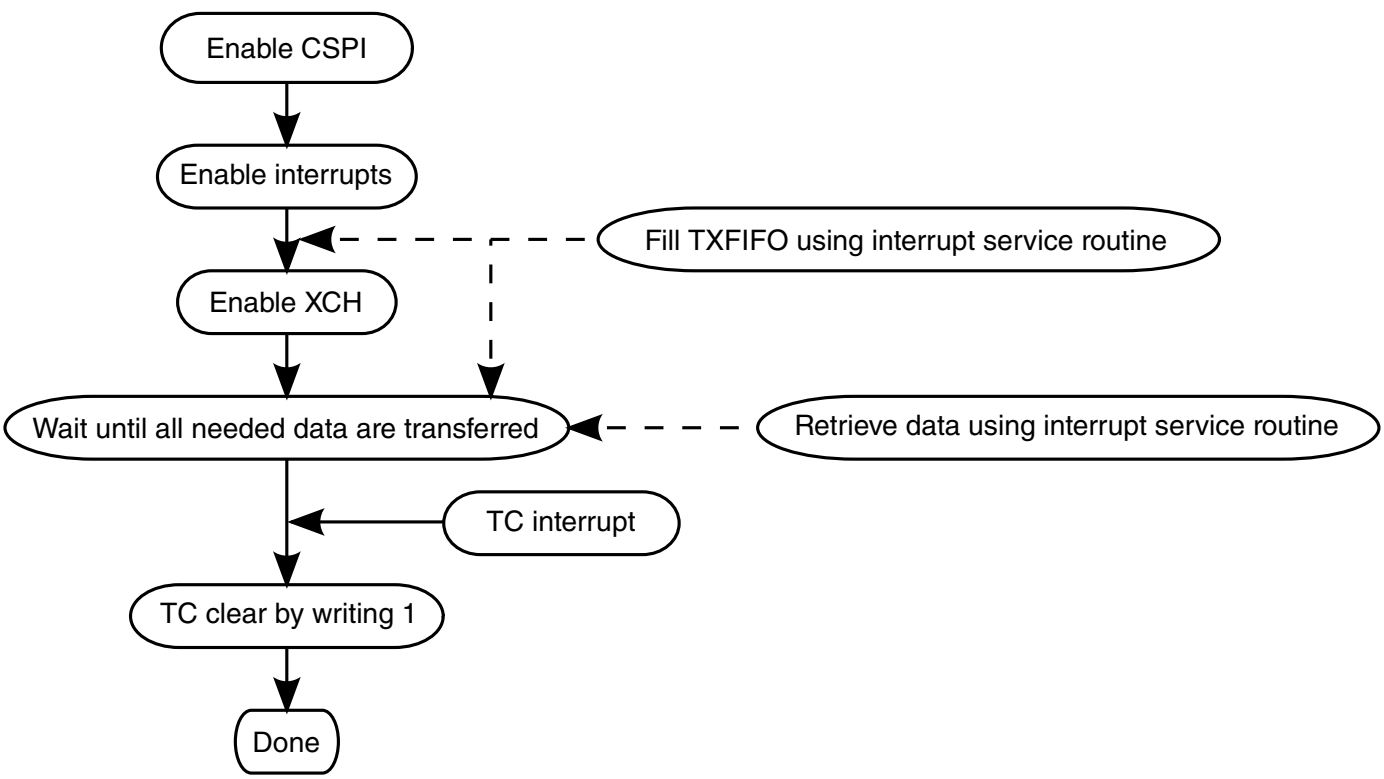


Figure 10-12. Program Sequence of SPI Burst Using Interrupt Control

10.1.4.7 DMA

DMA control provides another method to utilize the FIFOs in the ECSPI. By using DMA request and acknowledge signals, larger amounts of data can be transferred, and will reduce interrupts and host processor loading. When the appropriate conditions are matched, the block will send out a DMA request.

The DMA can deal with the following conditions:

- TXFIFO empty
- TXFIFO data request
- RXFIFO data request
- RXFIFO full

The figure below shows a program sequence of SPI bursts using DMA control.

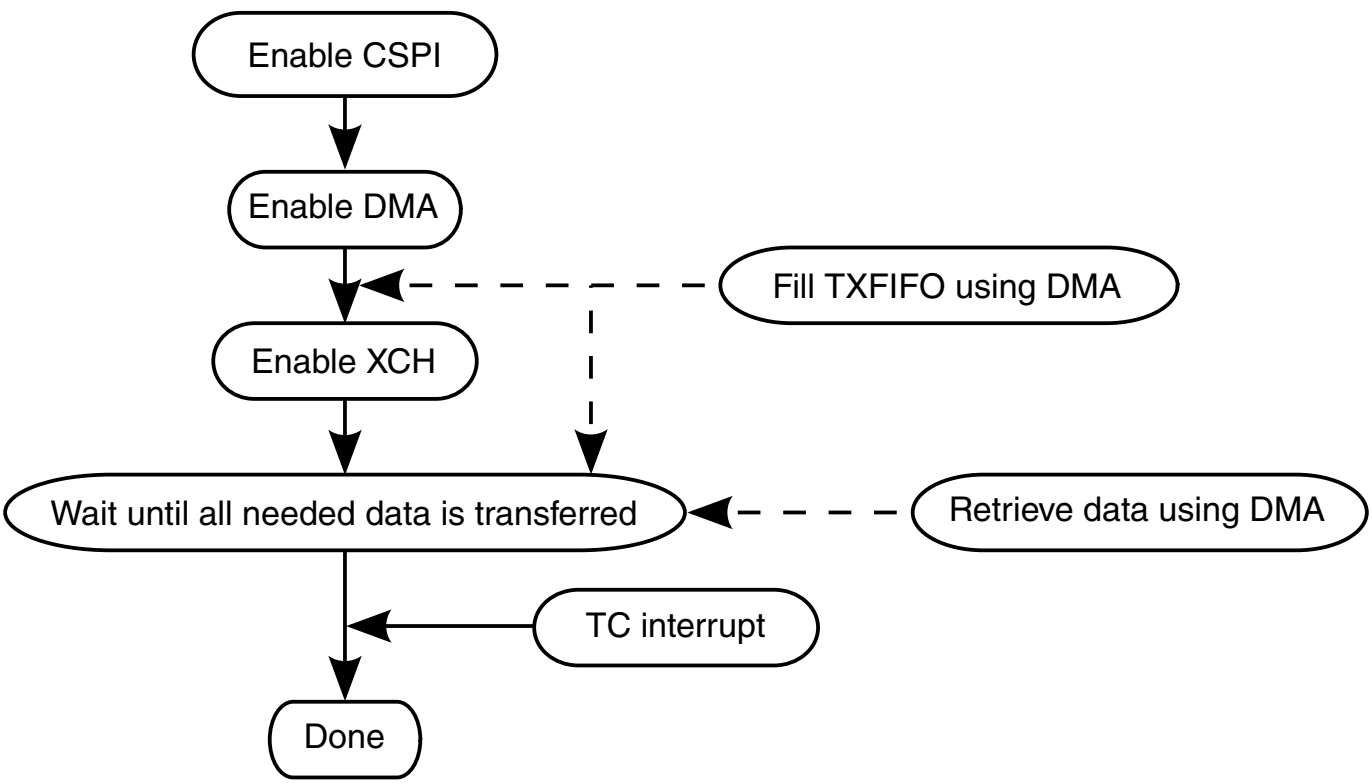


Figure 10-13. Program Sequence of SPI Burst Using DMA

10.1.4.8 Byte Order

The ECSPI does not support byte re-ordering in hardware.

10.1.5 Initialization

This section provides initialization information for ECSPI.

To initialize the block:

1. Clear the EN bit in ECSPI_CONREG to reset the block.
2. Enable the clocks for ECSPI.
3. Set the EN bit in ECSPI_CONREG to put ECSPI out of reset.
4. Configure corresponding IOMUX for ECSPI external signals.
5. Configure registers of ECSPI properly according to the specifications of the external SPI device.

10.1.6 Applications

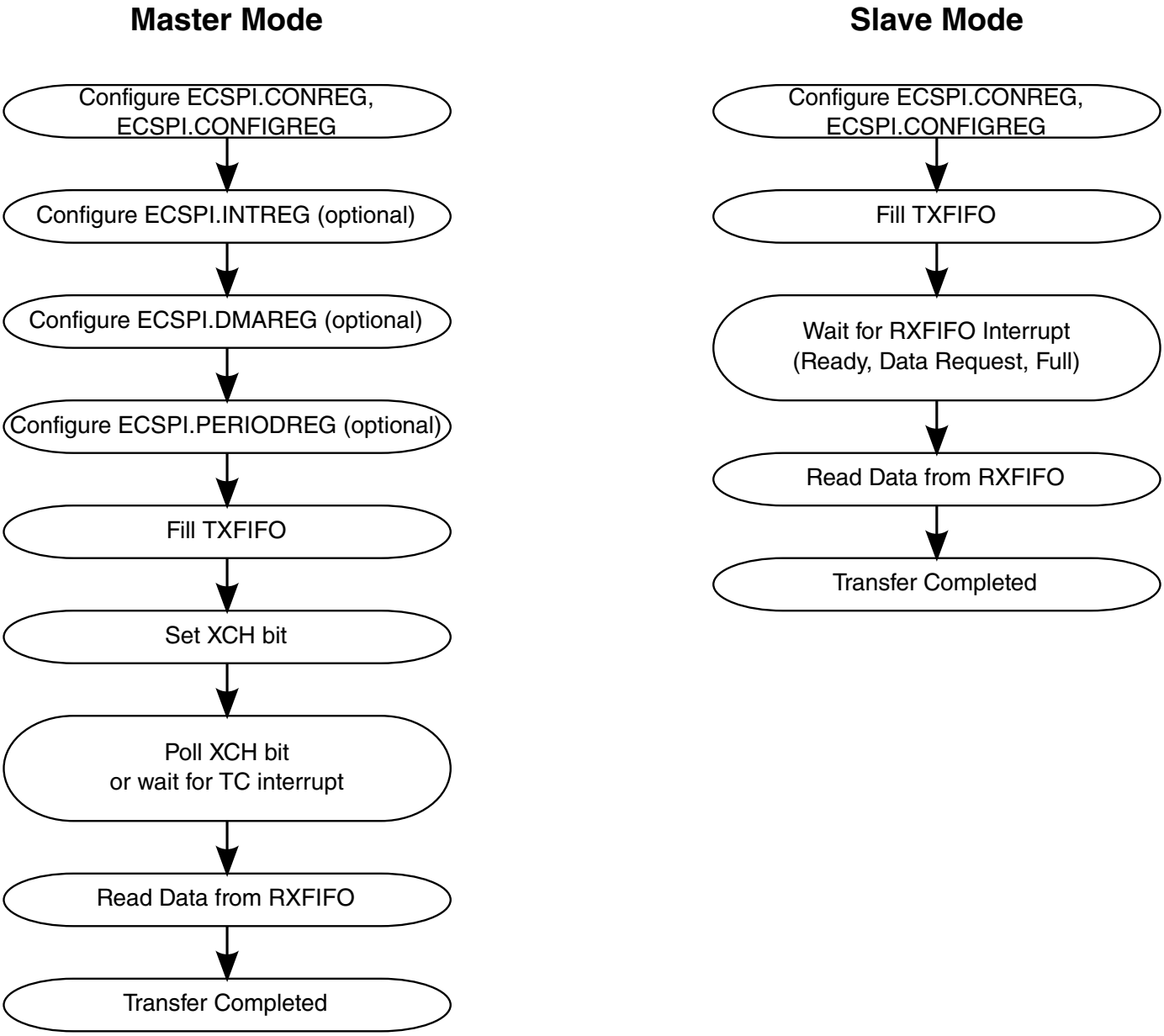


Figure 10-14. Flowchart of the ECSPI Operation

10.1.7 ECSPI Memory Map/Register Definition

This section includes the block memory map and detailed descriptions of all registers. For the base address of a particular block instantiation, see the system memory map.

ECSPI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3063_0000	Receive Data Register (ECSPI4_RXDATA)	32	R	0000_0000h	10.1.7.1/ 2582
3063_0004	Transmit Data Register (ECSPI4_TXDATA)	32	W	0000_0000h	10.1.7.2/ 2582
3063_0008	Control Register (ECSPI4_CONREG)	32	R/W	0000_0000h	10.1.7.3/ 2583
3063_000C	Config Register (ECSPI4_CONFIGREG)	32	R/W	0000_0000h	10.1.7.4/ 2585
3063_0010	Interrupt Control Register (ECSPI4_INTREG)	32	R/W	0000_0000h	10.1.7.5/ 2588
3063_0014	DMA Control Register (ECSPI4_DMAREG)	32	R/W	0000_0000h	10.1.7.6/ 2589
3063_0018	Status Register (ECSPI4_STATREG)	32	R/W	0000_0003h	10.1.7.7/ 2591
3063_001C	Sample Period Control Register (ECSPI4_PERIODREG)	32	R/W	0000_0000h	10.1.7.8/ 2592
3063_0020	Test Control Register (ECSPI4_TESTREG)	32	R/W	0000_0000h	10.1.7.9/ 2593
3063_0040	Message Data Register (ECSPI4_MSGDATA)	32	W	0000_0000h	10.1.7.10/ 2594
3082_0000	Receive Data Register (ECSPI1_RXDATA)	32	R	0000_0000h	10.1.7.1/ 2582
3082_0004	Transmit Data Register (ECSPI1_TXDATA)	32	W	0000_0000h	10.1.7.2/ 2582
3082_0008	Control Register (ECSPI1_CONREG)	32	R/W	0000_0000h	10.1.7.3/ 2583
3082_000C	Config Register (ECSPI1_CONFIGREG)	32	R/W	0000_0000h	10.1.7.4/ 2585
3082_0010	Interrupt Control Register (ECSPI1_INTREG)	32	R/W	0000_0000h	10.1.7.5/ 2588
3082_0014	DMA Control Register (ECSPI1_DMAREG)	32	R/W	0000_0000h	10.1.7.6/ 2589
3082_0018	Status Register (ECSPI1_STATREG)	32	R/W	0000_0003h	10.1.7.7/ 2591

Table continues on the next page...

ECSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3082_001C	Sample Period Control Register (ECSPI1_PERIODREG)	32	R/W	0000_0000h	10.1.7.8/ 2592
3082_0020	Test Control Register (ECSPI1_TESTREG)	32	R/W	0000_0000h	10.1.7.9/ 2593
3082_0040	Message Data Register (ECSPI1_MSGDATA)	32	W	0000_0000h	10.1.7.10/ 2594
3083_0000	Receive Data Register (ECSPI2_RXDATA)	32	R	0000_0000h	10.1.7.1/ 2582
3083_0004	Transmit Data Register (ECSPI2_TXDATA)	32	W	0000_0000h	10.1.7.2/ 2582
3083_0008	Control Register (ECSPI2_CONREG)	32	R/W	0000_0000h	10.1.7.3/ 2583
3083_000C	Config Register (ECSPI2_CONFIGREG)	32	R/W	0000_0000h	10.1.7.4/ 2585
3083_0010	Interrupt Control Register (ECSPI2_INTREG)	32	R/W	0000_0000h	10.1.7.5/ 2588
3083_0014	DMA Control Register (ECSPI2_DMAREG)	32	R/W	0000_0000h	10.1.7.6/ 2589
3083_0018	Status Register (ECSPI2_STATREG)	32	R/W	0000_0003h	10.1.7.7/ 2591
3083_001C	Sample Period Control Register (ECSPI2_PERIODREG)	32	R/W	0000_0000h	10.1.7.8/ 2592
3083_0020	Test Control Register (ECSPI2_TESTREG)	32	R/W	0000_0000h	10.1.7.9/ 2593
3083_0040	Message Data Register (ECSPI2_MSGDATA)	32	W	0000_0000h	10.1.7.10/ 2594
3084_0000	Receive Data Register (ECSPI3_RXDATA)	32	R	0000_0000h	10.1.7.1/ 2582
3084_0004	Transmit Data Register (ECSPI3_TXDATA)	32	W	0000_0000h	10.1.7.2/ 2582
3084_0008	Control Register (ECSPI3_CONREG)	32	R/W	0000_0000h	10.1.7.3/ 2583
3084_000C	Config Register (ECSPI3_CONFIGREG)	32	R/W	0000_0000h	10.1.7.4/ 2585
3084_0010	Interrupt Control Register (ECSPI3_INTREG)	32	R/W	0000_0000h	10.1.7.5/ 2588
3084_0014	DMA Control Register (ECSPI3_DMAREG)	32	R/W	0000_0000h	10.1.7.6/ 2589
3084_0018	Status Register (ECSPI3_STATREG)	32	R/W	0000_0003h	10.1.7.7/ 2591
3084_001C	Sample Period Control Register (ECSPI3_PERIODREG)	32	R/W	0000_0000h	10.1.7.8/ 2592
3084_0020	Test Control Register (ECSPI3_TESTREG)	32	R/W	0000_0000h	10.1.7.9/ 2593

Table continues on the next page...

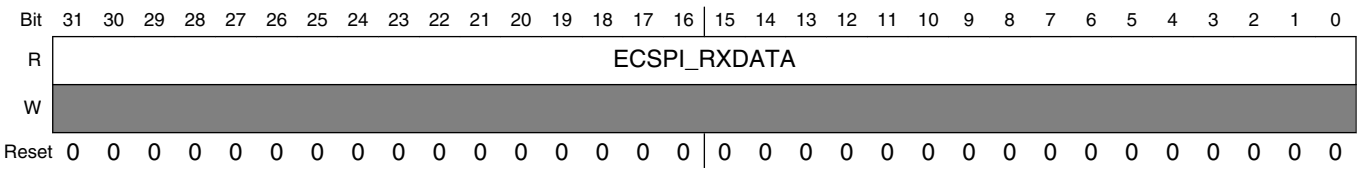
ECSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3084_0040	Message Data Register (ECSPI3_MSGDATA)	32	W	0000_0000h	10.1.7.10/ 2594

10.1.7.1 Receive Data Register (ECSPIx_RXDATA)

The Receive Data register (ECSPI_RXDATA) is a read-only register that forms the top word of the 64 x 32 receive FIFO. This register holds the data received from an external SPI device during a data transaction. Only word-sized read operations are allowed.

Address: Base address + 0h offset



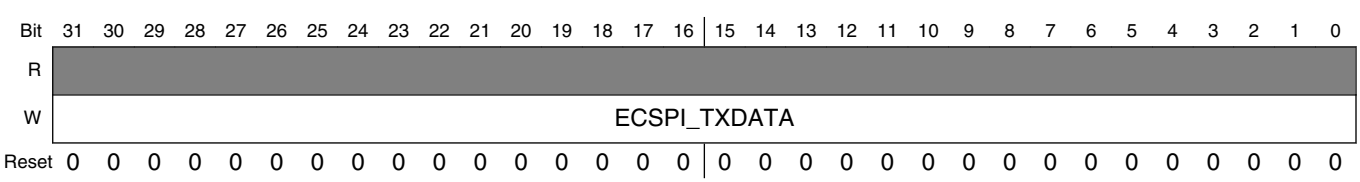
ECSPIx_RXDATA field descriptions

Field	Description
ECSPI_RXDATA	Receive Data. This register holds the top word of the receive data FIFO. The FIFO is advanced for each read of this register. The data read is undefined when the Receive Data Ready (RR) bit in the Interrupt Control/Status register is cleared. Zeros are read when ECSPI is disabled.

10.1.7.2 Transmit Data Register (ECSPIx_TXDATA)

The Transmit Data (ECSPI_TXDATA) register is a write-only data register that forms the bottom word of the 64 x 32 TXFIFO. The TXFIFO can be written to as long as it is not full, even when the SPI Exchange bit (XCH) in ECSPI_CONREG is set. This allows software to write to the TXFIFO during a SPI data exchange process. Writes to this register are ignored when the ECSPI is disabled (ECSPI_CONREG[EN] bit is cleared).

Address: Base address + 4h offset



ECSPIx_TXDATA field descriptions

Field	Description
ECSPI_TXDATA	Transmit Data. This register holds the top word of data loaded into the FIFO. Data written to this register must be a word operation. The number of bits actually transmitted is determined by the BURST_LENGTH field of the corresponding SPI Control register. If this field contains more bits than the number specified by BURST_LENGTH, the extra bits are ignored. For example, to transfer 10 bits of data, a 32-bit word must be written to this register. Bits 9-0 are shifted out and bits 31-10 are ignored. When the ECSPI is operating in Slave mode, zeros are shifted out when the FIFO is empty. Zeros are read when ECSPI is disabled.

10.1.7.3 Control Register (ECSPIx_CONREG)

The Control Register (ECSPI_CONREG) allows software to enable the ECSPI , configure its operating modes, specify the divider value, and SPI_RDY control signal, and define the transfer length.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BURST_LENGTH												CHANNEL_ SELECT		DRCTL	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRE_DIVIDER				POST_DIVIDER				CHANNEL_MODE				SMC	XCH	HT	EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ECSPIx_CONREG field descriptions

Field	Description
31–20 BURST_ LENGTH	Burst Length. This field defines the length of a SPI burst to be transferred. The Chip Select (SS) will remain asserted until all bits in a SPI burst are shifted out. A maximum of 2^12 bits can be transferred in a single SPI burst. In master mode, it controls the number of bits per SPI burst. Since the shift register always loads 32-bit data from transmit FIFO, only the n least-significant (n = BURST LENGTH + 1) will be shifted out. The remaining bits will be ignored. Number of Valid Bits in a SPI burst. 0x000 A SPI burst contains the 1 LSB in a word. 0x001 A SPI burst contains the 2 LSB in a word. 0x002 A SPI burst contains the 3 LSB in a word. ... 0x01F A SPI burst contains all 32 bits in a word. 0x020 A SPI burst contains the 1 LSB in first word and all 32 bits in second word. 0x021 A SPI burst contains the 2 LSB in first word and all 32 bits in second word. ...

Table continues on the next page...

ECSPIx_CONREG field descriptions (continued)

Field	Description
	0xFFE A SPI burst contains the 31 LSB in first word and 2^7 -1 words. 0xFFF A SPI burst contains 2^7 words.
19–18 CHANNEL_ SELECT	SPI CHANNEL SELECT bits. Select one of four external SPI Master/Slave Devices. In master mode, these two bits select the external slave devices by asserting the Chip Select (SSn) outputs. Only the selected Chip Select (SSn) signal can be active at a given time; the remaining three signals will be negated. 00 Channel 0 is selected. Chip Select 0 (SS0) will be asserted. 01 Channel 1 is selected. Chip Select 1 (SS1) will be asserted. 10 Channel 2 is selected. Chip Select 2 (SS2) will be asserted. 11 Channel 3 is selected. Chip Select 3 (SS3) will be asserted.
17–16 DRCTL	SPI Data Ready Control. This field selects the utilization of the $\overline{\text{SPI\_RDY}}$ signal in master mode. ECSPI checks this field before it starts an SPI burst. 00 The $\overline{\text{SPI\_RDY}}$ signal is a don't care. 01 Burst will be triggered by the falling edge of the SPI_RDY signal (edge-triggered). 10 Burst will be triggered by a low level of the SPI_RDY signal (level-triggered). 11 Reserved.
15–12 PRE_DIVIDER	SPI Pre Divider. ECSPI uses a two-stage divider to generate the SPI clock. This field defines the pre-divider of the reference clock. 0000 Divide by 1. 0001 Divide by 2. 0010 Divide by 3. ... 1101 Divide by 14. 1110 Divide by 15. 1111 Divide by 16.
11–8 POST_DIVIDER	SPI Post Divider. ECSPI uses a two-stage divider to generate the SPI clock. This field defines the post-divider of the reference clock using the equation: 2^n . 0000 Divide by 1. 0001 Divide by 2. 0010 Divide by 4. ... 1110 Divide by 2^{14} . 1111 Divide by 2^{15} .
7–4 CHANNEL_ MODE	SPI CHANNEL MODE selects the mode for each SPI channel. CHANNEL MODE[3] is for SPI channel 3. CHANNEL MODE[2] is for SPI channel 2. CHANNEL MODE[1] is for SPI channel 1. CHANNEL MODE[0] is for SPI channel 0. 0 Slave mode. 1 Master mode.
3 SMC	Start Mode Control. This bit applies only to channels configured in Master mode (CHANNEL MODE = 1).

Table continues on the next page...

ECSPIx_CONREG field descriptions (continued)

Field	Description
	It controls how the ECSPI starts a SPI burst, either through the SPI exchange bit, or immediately when the TXFIFO is written to. 0 SPI Exchange Bit (XCH) controls when a SPI burst can start. Setting the XCH bit will start a SPI burst or multiple bursts. This is controlled by the SPI SS Wave Form Select (SS_CTL). Refer to XCH and SS_CTL descriptions. 1 Immediately starts a SPI burst when data is written in TXFIFO.
2 XCH	SPI Exchange Bit. This bit applies only to channels configured in Master mode (CHANNEL MODE = 1). If the Start Mode Control (SMC) bit is cleared, writing a 1 to this bit starts one SPI burst or multiple SPI bursts according to the SPI SS Wave Form Select (SS_CTL). The XCH bit remains set while either the data exchange is in progress, or when the ECSPI is waiting for an active input if SPIRDY is enabled through DRCTL. This bit is cleared automatically when all data in the TXFIFO and the shift register has been shifted out. 0 Idle. 1 Initiates exchange (write) or busy (read).
1 HT	Hardware Trigger Enable. This bit is used in master mode only. It enables hardware trigger (HT) mode. Note, HT mode is not supported by this product. 0 Disable HT mode. 1 Enable HT mode.
0 EN	SPI Block Enable Control. This bit enables the ECSPI. This bit must be set before writing to other registers or initiating an exchange. Writing zero to this bit disables the block and resets the internal logic with the exception of the ECSPI_CONREG. The block's internal clocks are gated off whenever the block is disabled. 0 Disable the block. 1 Enable the block.

10.1.7.4 Config Register (ECSPIx_CONFIGREG)

The Config Register (ECSPI_CONFIGREG) allows software to configure each SPI channel, configure its operating modes, specify the phase and polarity of the clock, configure the Chip Select (SS), and define the HT transfer length. Note, HT mode is not supported by this product.

Address: Base address + Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved			HT_LENGTH					SCLK_CTL				DATA_CTL				SS_POL				SS_CTL				SCLK_POL				SCLK_PHA			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ECSPIx_CONFIGREG field descriptions

Field	Description
31–29 -	This field is reserved. Reserved
28–24 HT_LENGTH	HT LENGTH. This field defines the message length in HT Mode. Note, HT mode is not supported by this product. The length in bits of one message is (HT LENGTH + 1).
23–20 SCLK_CTL	SCLK CTL. This field controls the inactive state of SCLK for each SPI channel. SCLK CTL[3] is for SPI channel 3. SCLK CTL[2] is for SPI channel 2. SCLK CTL[1] is for SPI channel 1. SCLK CTL[0] is for SPI channel 0. 0 Stay low. 1 Stay high.
19–16 DATA_CTL	DATA CTL. This field controls inactive state of the data line for each SPI channel. DATA CTL[3] is for SPI channel 3. DATA CTL[2] is for SPI channel 2. DATA CTL[1] is for SPI channel 1. DATA CTL[0] is for SPI channel 0. 0 Stay high. 1 Stay low.
15–12 SS_POL	SPI SS Polarity Select. In both Master and Slave modes, this field selects the polarity of the Chip Select (SS) signal. SS POL[3] is for SPI channel 3. SS POL[2] is for SPI channel 2. SS POL[1] is for SPI channel 1. SS POL[0] is for SPI channel 0. 0 Active low. 1 Active high.
11–8 SS_CTL	SPI SS Wave Form Select. In master mode, this field controls the output wave form of the Chip Select (SS) signal when the SMC (Start Mode Control) bit is cleared. The SS_CTL are ignored if the SMC bit is set. SS CTL[3] is for SPI channel 3. SS CTL[2] is for SPI channel 2. SS CTL[1] is for SPI channel 1. SS CTL[0] is for SPI channel 0. In slave mode, this bit controls when the SPI burst is completed. An SPI burst is completed by the Chip Select (SS) signal edges. (SSPOL = 0: rising edge; SSPOL = 1: falling edge) The RXFIFO is advanced whenever a Chip Select (SS) signal edge is detected or the shift register contains 32-bits of valid data. 0 In master mode - only one SPI burst will be transmitted.

Table continues on the next page...

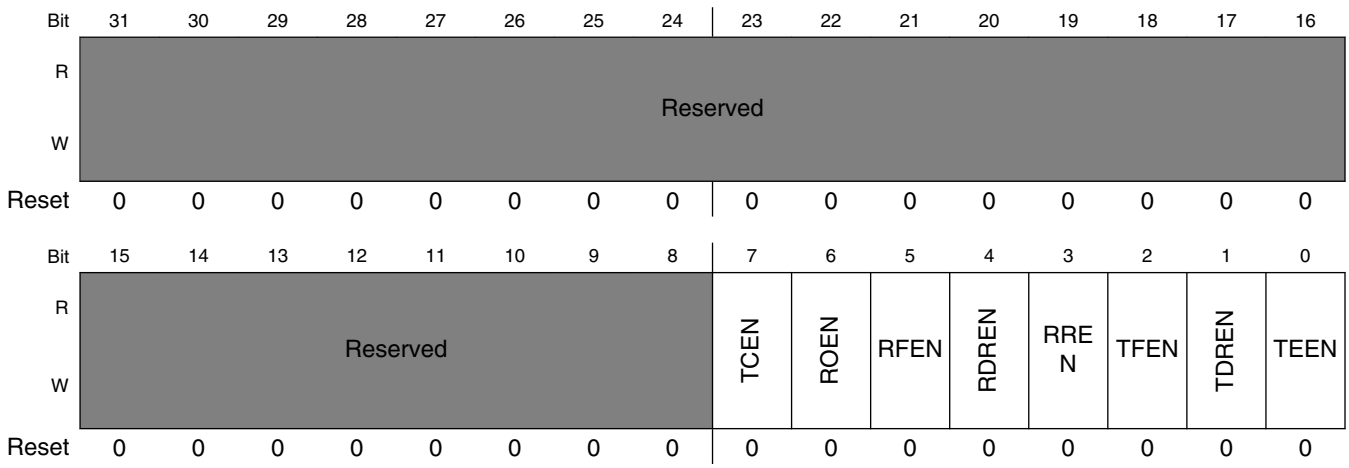
ECSPIx_CONFIGREG field descriptions (continued)

Field	Description
	1 In master mode - Negate Chip Select (SS) signal between SPI bursts. Multiple SPI bursts will be transmitted. The SPI transfer will automatically stop when the TXFIFO is empty. 0 In slave mode - an SPI burst is completed when the number of bits received in the shift register is equal to (BURST LENGTH + 1). Only the n least-significant bits (n = BURST LENGTH[4:0] + 1) of the first received word are valid. All bits subsequent to the first received word in RXFIFO are valid. 1 Reserved
7–4 SCLK_POL	SPI Clock Polarity Control. This field controls the polarity of the SCLK signal. See Figure 10-10 for more information. SCLK_POL[3] is for SPI channel 3. SCLK_POL[2] is for SPI channel 2. SCLK_POL[1] is for SPI channel 1. SCLK_POL[0] is for SPI channel 0. 0 Active high polarity (0 = Idle). 1 Active low polarity (1 = Idle).
SCLK_PHA	SPI Clock/Data Phase Control. This field controls the clock/data phase relationship. See Figure 10-10 for more information. SCLK PHA[3] is for SPI channel 3. SCLK PHA[2] is for SPI channel 2. SCLK PHA[1] is for SPI channel 1. SCLK PHA[0] is for SPI channel 0. 0 Phase 0 operation. 1 Phase 1 operation.

10.1.7.5 Interrupt Control Register (ECSPIx_INTREG)

The Interrupt Control Register (ECSPI_INTREG) enables the generation of interrupts to the host processor. If the ECSPI is disabled, this register reads zero.

Address: Base address + 10h offset



ECSPIx_INTREG field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
7 TCEN	Transfer Completed Interrupt enable. This bit enables the Transfer Completed Interrupt. 0 Disable 1 Enable
6 ROEN	RXFIFO Overflow Interrupt enable. This bit enables the RXFIFO Overflow Interrupt. 0 Disable 1 Enable
5 RFEN	RXFIFO Full Interrupt enable. This bit enables the RXFIFO Full Interrupt. 0 Disable 1 Enable
4 RDREN	RXFIFO Data Request Interrupt enable. This bit enables the RXFIFO Data Request Interrupt when the number of data entries in the RXFIFO is greater than RX_THRESHOLD. 0 Disable 1 Enable
3 RREN	RXFIFO Ready Interrupt enable. This bit enables the RXFIFO Ready Interrupt. 0 Disable 1 Enable

Table continues on the next page...

ECSPIx_INTREG field descriptions (continued)

Field	Description
2 TFEN	TXFIFO Full Interrupt enable. This bit enables the TXFIFO Full Interrupt. 0 Disable 1 Enable
1 TDREN	TXFIFO Data Request Interrupt enable. This bit enables the TXFIFO Data Request Interrupt when the number of data entries in the TXFIFO is less than or equal to TX_THRESHOLD. 0 Disable 1 Enable
0 TEEN	TXFIFO Empty Interrupt enable. This bit enables the TXFIFO Empty Interrupt. 0 Disable 1 Enable

10.1.7.6 DMA Control Register (ECSPIx_DMAREG)

The Direct Memory Access Control Register (ECSPI_DMAREG) provides software a way to use an on-chip DMA controller for ECSPI data. Internal DMA request signals enable direct data transfers between the ECSPI FIFOs and system memory. The ECSPI sends out DMA requests when the appropriate FIFO conditions are matched.

If the ECSPI is disabled, this register is read as 0.

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RXTDEN	Reserved	RX_DMA_LENGTH						RXDEN	Reserved	RX_THRESHOLD					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								TEDEN	Reserved	TX_THRESHOLD					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ECSPIx_DMAREG field descriptions

Field	Description
31 RXTDEN	RXFIFO TAIL DMA Request Enable. This bit enables an internal counter that is increased at each read of the RXFIFO. This counter is cleared automatically when it reaches RX DMA LENGTH. If the number of words remaining in the RXFIFO is greater than or equal to RX DMA LENGTH, a DMA request is generated even if it is less than or equal to RX_THRESHOLD. 0 Disable 1 Enable
30 -	This field is reserved. Reserved
29–24 RX_DMA_ LENGTH	RX DMA LENGTH. This field defines the burst length of a DMA operation. Applies only when RXTDEN is set.
23 RXDEN	RXFIFO DMA Request Enable. This bit enables/disables the RXFIFO DMA Request. 0 Disable 1 Enable
22 -	This field is reserved. Reserved
21–16 RX_ THRESHOLD	RX THRESHOLD. This field defines the FIFO threshold that triggers a RX DMA/INT request. A RX DMA/INT request is issued when the number of data entries in the RXFIFO is greater than RX_THRESHOLD.
15–8 -	This field is reserved. Reserved
7 TEDEN	TXFIFO Empty DMA Request Enable. This bit enables/disables the TXFIFO Empty DMA Request. 0 Disable 1 Enable
6 -	This field is reserved. Reserved
TX_ THRESHOLD	TX THRESHOLD. This field defines the FIFO threshold that triggers a TX DMA/INT request. A TX DMA/INT request is issued when the number of data entries in the TXFIFO is not greater than TX_THRESHOLD.

10.1.7.7 Status Register (ECSPIx_STATREG)

The ECSPI Status Register (ECSPI_STATREG) reflects the status of the ECSPI's operating condition. If the ECSPI is disabled, this register reads 0x0000_0003.

Address: Base address + 18h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								TC	RO	RF	RDR	RR	TF	TDR	TE
W									w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

ECSPIx_STATREG field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
7 TC	Transfer Completed Status bit. Writing 1 to this bit clears it. 0 Transfer in progress. 1 Transfer completed.
6 RO	RXFIFO Overflow. When set, this bit indicates that RXFIFO has overflowed. Writing 1 to this bit clears it. 0 RXFIFO has no overflow. 1 RXFIFO has overflowed.
5 RF	RXFIFO Full. This bit is set when the RXFIFO is full. 0 Not Full. 1 Full.
4 RDR	RXFIFO Data Request. 0 When RXTDE is set - Number of data entries in the RXFIFO is not greater than RX_THRESHOLD. 1 When RXTDE is set - Number of data entries in the RXFIFO is greater than RX_THRESHOLD or a DMA TAIL DMA condition exists. 0 When RXTDE is clear - Number of data entries in the RXFIFO is not greater than RX_THRESHOLD. 1 When RXTDE is clear - Number of data entries in the RXFIFO is greater than RX_THRESHOLD.
3 RR	RXFIFO Ready. This bit is set when one or more words are stored in the RXFIFO. 0 No valid data in RXFIFO. 1 More than 1 word in RXFIFO.
2 TF	TXFIFO Full. This bit is set when if the TXFIFO is full.

Table continues on the next page...

ECSPIx_STATREG field descriptions (continued)

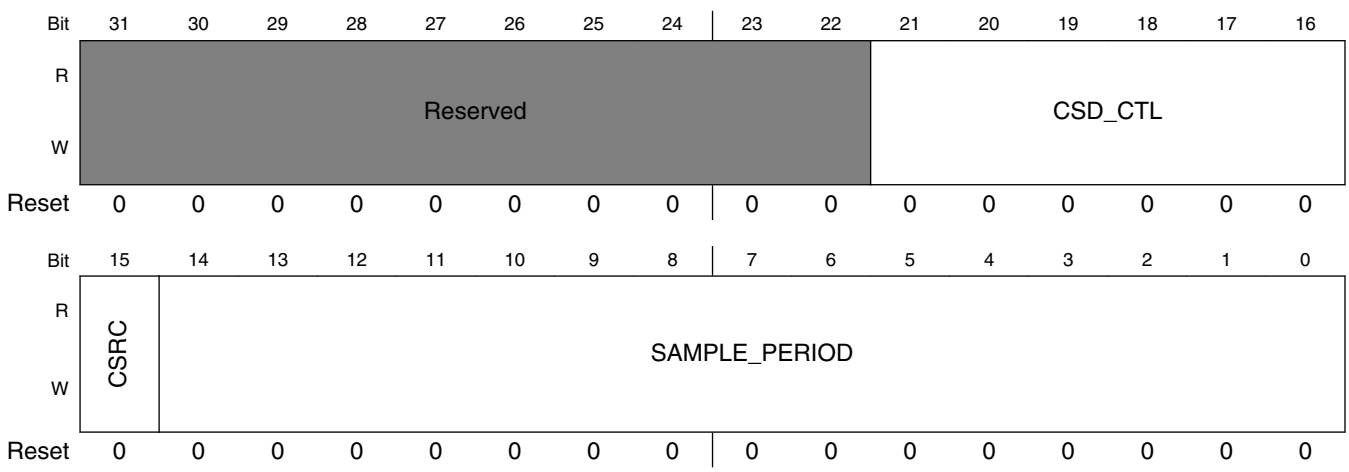
Field	Description
	0 TXFIFO is not Full. 1 TXFIFO is Full.
1 TDR	TXFIFO Data Request. 0 Number of valid data slots in TXFIFO is greater than TX_THRESHOLD. 1 Number of valid data slots in TXFIFO is not greater than TX_THRESHOLD.
0 TE	TXFIFO Empty. This bit is set if the TXFIFO is empty. 0 TXFIFO contains one or more words. 1 TXFIFO is empty.

10.1.7.8 Sample Period Control Register (ECSPIx_PERIODREG)

The Sample Period Control Register (ECSPI_PERIODREG) provides software a way to insert delays (wait states) between consecutive SPI transfers. Control bits in this register select the clock source for the sample period counter and the delay count indicating the number of wait states to be inserted between data transfers.

The delay counts apply only when the current channel is operating in Master mode (ECSPI_CONREG[CHANNEL MODE] = 1).ECSPI_PERIODREG also contains the CSD CTRL field used to insert a delay between the Chip Select's active edge and the first SPI Clock edge.

Address: Base address + 1Ch offset



ECSPIx_PERIODREG field descriptions

Field	Description
31–22 -	This field is reserved. Reserved

Table continues on the next page...

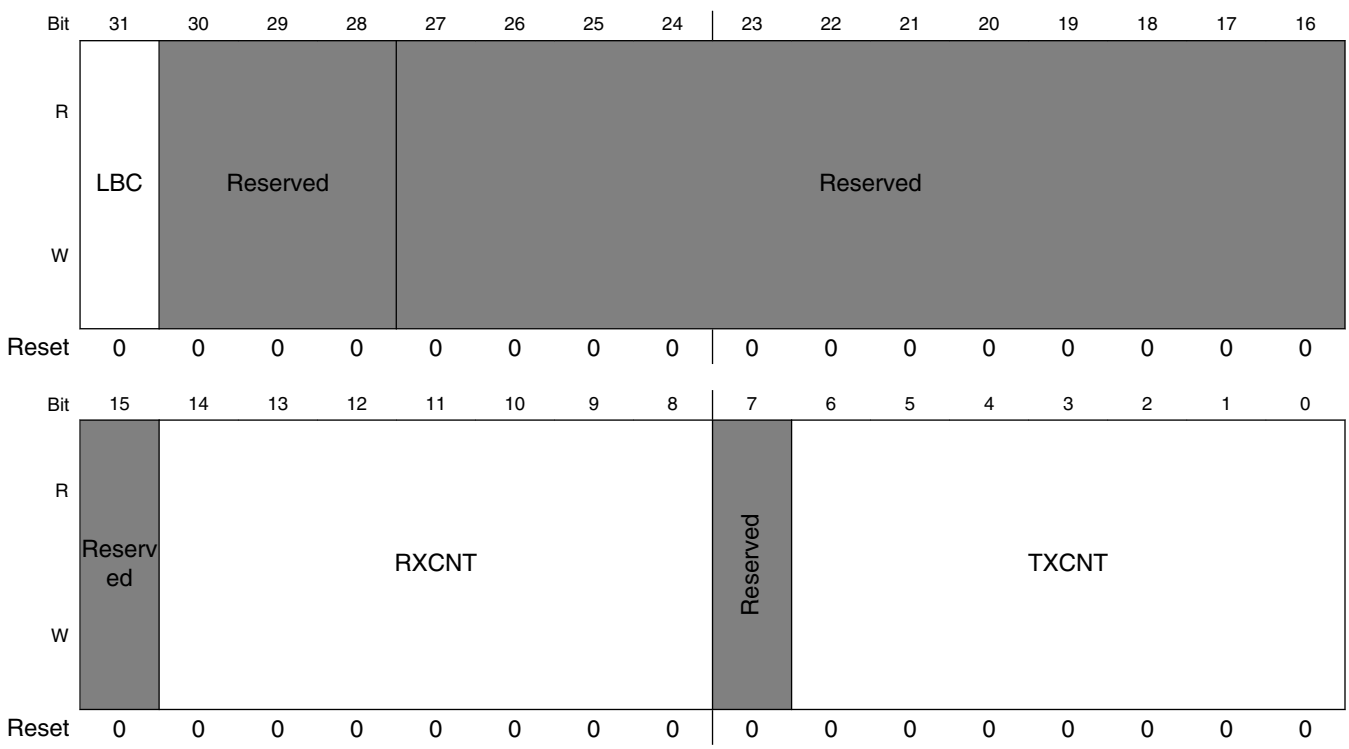
ECSPIx_PERIODREG field descriptions (continued)

Field	Description
21–16 CSD_CTL	Chip Select Delay Control bits. This field defines how many SPI clocks will be inserted between the chip select's active edge and the first SPI clock edge. The range is from 0 to 63.
15 CSRC	Clock Source Control. This bit selects the clock source for the sample period counter. 0 SPI Clock (SCLK) 1 Low-Frequency Reference Clock (32.768 KHz)
SAMPLE_ PERIOD	Sample Period Control. These bits control the number of wait states to be inserted in data transfers. During the idle clocks, the state of the SS output will operate according to the SS_CTL control field in the ECSPI_CONREG register. 0x0000 0 wait states inserted 0x0001 1 wait state inserted ... ... 0x7FFE 32766 wait states inserted 0x7FFF 32767 wait states inserted

10.1.7.9 Test Control Register (ECSPIx_TESTREG)

The Test Control Register (ECSPI_TESTREG) provides software a mechanism to internally connect the receive and transmit devices of the ECSPI, and monitor the contents of the receive and transmit FIFOs.

Address: Base address + 20h offset



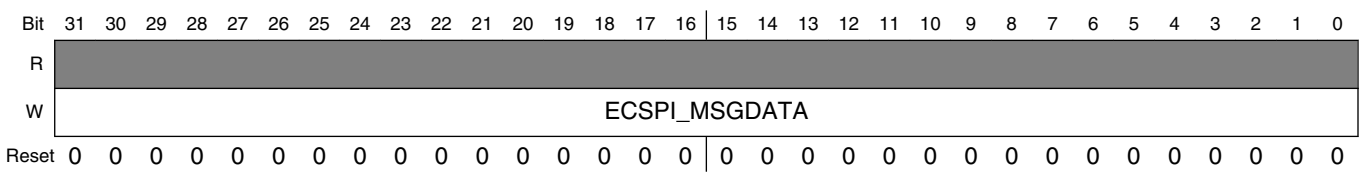
ECSPIx_TESTREG field descriptions

Field	Description
31 LBC	Loop Back Control. This bit is used in Master mode only. When this bit is set, the ECSPI connects the transmitter and receiver sections internally, and the data shifted out from the most-significant bit of the shift register is looped back into the least-significant bit of the Shift register. In this way, a self-test of the complete transmit/receive path can be made. The output pins are not affected, and the input pins are ignored. 0 Not connected. 1 Transmitter and receiver sections internally connected for Loopback.
30–28 -	This field is reserved. Reserved, all bits should be ignored.
27–15 -	This field is reserved. Reserved
14–8 RXCNT	RXFIFO Counter. This field indicates the number of words in the RXFIFO.
7 -	This field is reserved. Reserved
TXCNT	TXFIFO Counter. This field indicates the number of words in the TXFIFO.

10.1.7.10 Message Data Register (ECSPIx_MSGDATA)

The Message Data Register (ECSPI_MSGDATA) forms the top word of the 16 x 32 MSG Data FIFO. Only word-size accesses are allowed for this register. Reads to this register return zero, and writes to this register store data in the MSG Data FIFO.

Address: Base address + 40h offset



ECSPIx_MSGDATA field descriptions

Field	Description
ECSPI_ MSGDATA	ECSPI_MSGDATA holds the top word of MSG Data FIFO. The MSG Data FIFO is advanced for each write of this register. The data read is zero. The data written to this register is stored in the MSG Data FIFO.

10.2 Quad Serial Peripheral Interface (QuadSPI)

10.2.1 Overview

The Quad Serial Peripheral Interface (QuadSPI) block acts as an interface to one or two external serial flash devices, each with up to four bidirectional data lines. The following figure is a block diagram of the QuadSPI module.

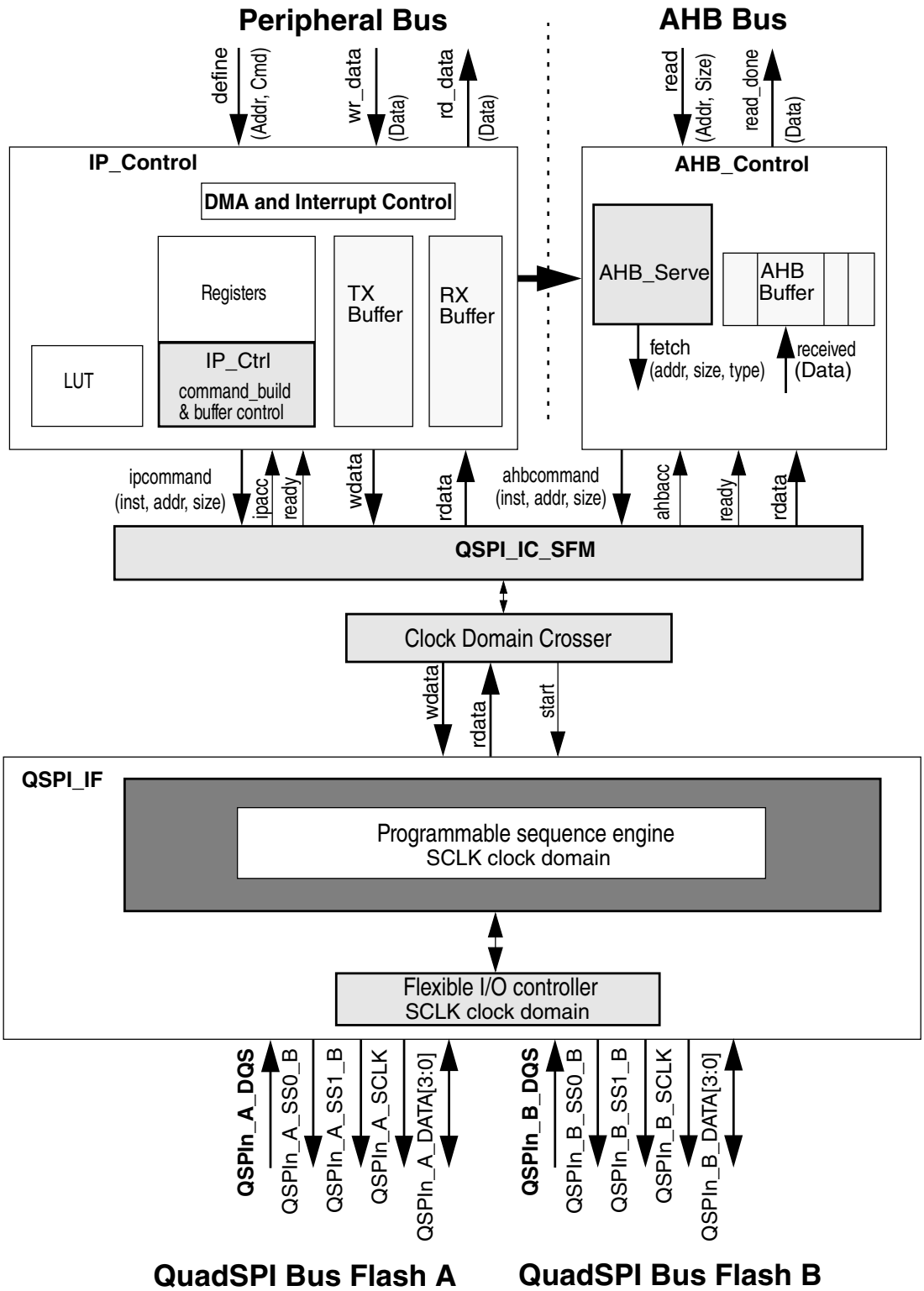


Figure 10-15. QuadSPI Block Diagram

The following figure describes the serial flash clock diagram in QuadSPI:

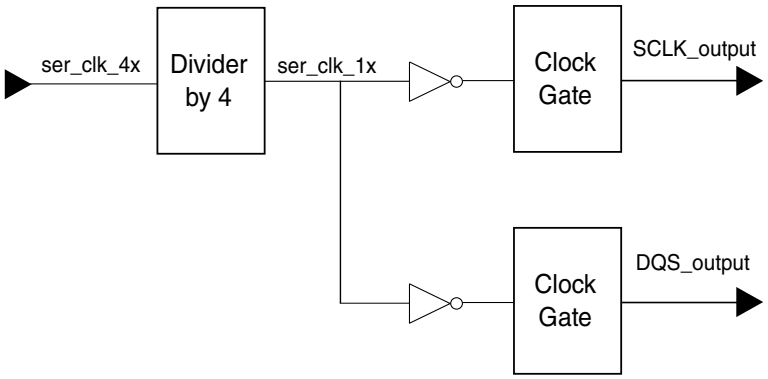


Figure 10-16. Flash Clock Diagram

ser_clk_4x is the serial clock root from CCM

ser_clk_1x' is generated clock from 'ser_clk_4x' divided by 4

SCLK_output is serial output clock generated from 'ser_clk_1x' with inverter and clock gate, it's used as Clock by external serial flash device.

DQS_output is serial flash data strobe signal generated from 'ser_clk_1x' with inverter and clock gate. This signal could be loopback from pad and used as sampling clock for serial flash data.

10.2.1.1 Features

The QuadSPI supports the following features:

- Flexible sequence engine to support various flash vendor devices.
- Single, dual, quad mode of operation.
- DDR/DTR mode wherein the data is generated on every edge of the serial flash clock.
- Support for flash data strobe signal for data sampling in DDR and SDR mode.
- Two identical serial flash devices can be connected and accessed in parallel for data read operations, forming one (virtual) flash memory with doubled readout bandwidth.
- DMA support to read RX Buffer data via AMBA AHB bus (64-bit width interface) or IP registers space (32-bit access).
    - Inner loop size of DMA access can be configured.
- Multi master accesses with priority

- Flexible and configurable buffer for each master

- Thirteen interrupt conditions (see Table 10-12)
- Memory mapped read access to connected flash devices.
- Programmable sequence engine to cater to future command/protocol changes and able to support all existing vendor commands and operations.
  - Supports 3-byte and 4-byte addressing.
  - TXFIFO size is 512Byte
  - RXFIFO size is 512Byte
  - AHB BUF size is 1KByte

10.2.1.2 QuadSPI Modes of Operation

This section provides information about the modes in which the QuadSPI module can be used.

10.2.1.2.1 Normal Mode

In this mode, one or two external serial flash memory devices can be accessed. Further details about this mode of operation can be found in Modes of Operation (Normal Mode).

10.2.1.2.2 Module Disable Mode

This mode is used for power management of the device containing the QuadSPI module. It is controlled by signals external to the QuadSPI. The clock to the non-memory mapped logic in the QuadSPI can be stopped while in Module Disable Mode.

10.2.1.3 Acronyms and Abbreviations

The following table contains acronyms and abbreviations used in this document.

Table 10-3. Acronyms and Abbreviations

Terms	Description
AHB	Advanced High-performance Bus, version of AMBA
AMBA	Advanced Microcontroller Bus Architecture
BE	Big Endian Byte Ordering
CS	Chip Select
DMA	Direct Memory Access

Table continues on the next page...

Table 10-3. Acronyms and Abbreviations (continued)

Terms	Description
MB	Megabyte. Each MB is 1024 * 1024 bytes
IFM	Individual Flash Mode
PFM	Parallel Flash Mode
LSB	Least Significant Bit
MSB	Most Significant Bit
PCS	Peripheral Chip Select
QSPI, QuadSPI	Quad Serial Peripheral Interface
SCK	Serial Communications Clock
w1c	Write 1 to clear, writing a 1 to this field resets the flag

10.2.1.4 Glossary for QuadSPI module

Table 10-4. Glossary

Term	Definition
AHB Command	An AHB Command is a SFM Command triggered by a read access to the address range belonging to the memory mapped access defined in Table 10-9. Refer to AHB Commands for details.
Asserted	A signal that is asserted is in its active state. An active low signal changes from logic level one to logic level zero when asserted, and an active high signal changes from logic level zero to logic level one.
Clear	To clear a bit or bits means to establish logic level zero on the bit or bits.
Clock Phase	Determines when the data should be sampled relative to the active edge of SCK
Clock Polarity	Determines the idle state of the SCK signal.
Drain	To remove entries from a FIFO by software or hardware.
Endianness	Byte Ordering scheme.
Field	Two or more register bits grouped together.
Fill	To add entries to a FIFO by software or hardware.
Host	Refers to another functional block in the device containing the QuadSPI module
Instruction Code	8 bits defining the type of command to be executed.
IP Command	An IP Command is a SFM Command triggered by writing into the QSPI_IPCR[SEQID] field.
Logic level one	The voltage that corresponds to Boolean true (1) state.
Logic level zero	The voltage that corresponds to Boolean false (0) state.
Negated	A signal that is negated is in its inactive state. An active low signal changes from logic level 0 to logic level 1 when negated, and an active high signal changes from logic level 1 to logic level 0.
QSPI_AMBA_BASE	First address of QuadSPI address space on system memory map.
QSPI_ARDB_BASE	First address of QuadSPI Rx Buffer on system memory map.
Set	To set a bit or bits means to establish logic level one on the bit or bits.

Table continues on the next page...

Table 10-4. Glossary (continued)

Term	Definition
RX Buffer PUSH Event	Addition of valid entries into the RX Buffer. In the default case each Buffer PUSH Event adds 2 entries to the RX Buffer since the interface to the serial clock domain is 64 bits in width. Depending on the number of bytes read from the serial flash device it is possible for the very last Buffer PUSH Event that only one entry is added. The QSPI_RBSR[RDBFL] field is incremented by the number of entries added to the RX Buffer.
RX Buffer POP Event	Removal of valid entries from the RX Buffer. Each Buffer POP Event removes (QSPI_RBCT[WMRK] + 1) valid entries from the buffer. The QSPI_RBSR[RDBFL] field is decremented by the same number and the QSPI_RBSR[RDCTR] field is incremented accordingly.
Individual Flash Mode	Access to a single, individual serial flash device. Refer to Serial Flash Access Schemes for details.
Parallel Flash Mode	Read access to two serial flash devices attached to the QuadSPI module in parallel. Refer to Serial Flash Access Schemes for details.
SFM Command	Serial Flash Memory Command. A SFM command consists of an instruction code and all other parameters (for example, size or mode bytes) needed for that specific instruction code. Triggering a command either initiates a transaction on the external serial flash or results in an error. Refer to Table 10-22 for details on errors.
Single/Dual/Quad Instructions	Depending on the serial flash device connected to the QuadSPI module there will be instructions using a different number of data lines. • Single: Single line I/O with one data out and one data in line to/from the serial flash device. • Dual: Dual line I/O with two bidirectional I/O lines, driven alternatively by the serial flash device or the QuadSPI module • Quad: Quad line I/O with 4 bidirectional I/O lines, driven alternatively by the serial flash device or the QuadSPI
Transaction	A transaction consists of all flags, data and signals in either direction to execute a command for an attached serial flash device. It is a combination of chip select, sclk, instruction code, address, mode- and/or dummy bytes, transmit and/or receive data.
LUT	Look-up table.

10.2.2 External Signals

The following table describes the external signals of QSPI:

Table 10-5. QSPI External Signals

Signal	Description	Pad	Mode	Direction
QSPI_A_DATA0	I/O data signal 1 port 0 for serial flash device A	EPDC1_DATA00	ALT2	IO
QSPI_A_DATA1	I/O data signal 1 port 1 for serial flash device A	EPDC1_DATA01	ALT2	IO
QSPI_A_DATA2	I/O data signal 1 port 2 for serial flash device A	EPDC1_DATA02	ALT2	IO
QSPI_A_DATA3	I/O data signal 1 port 3 for serial flash device A	EPDC1_DATA03	ALT2	IO
QSPI_A_DQS	Data strobe signal 1 to serial flash device A	EPDC1_DATA04	ALT2	I

Table continues on the next page...

Table 10-5. QSPI External Signals (continued)

Signal	Description	Pad	Mode	Direction
QSPI_A_SCLK	Serial clock output 1 to serial flash device A	EPDC1_DATA05	ALT2	O
QSPI_A_SS0_B	Chip select 1 port 0 for serial flash device A	EPDC1_DATA06	ALT2	O
QSPI_A_SS1_B	Chip select 1 port 1 for serial flash device A	EPDC1_DATA07	ALT2	O
QSPI_B_DATA0	I/O data signal 1 port 0 for serial flash device B	EPDC1_DATA08	ALT2	IO
QSPI_B_DATA1	I/O data signal 1 port 1 for serial flash device B	EPDC1_DATA09	ALT2	IO
QSPI_B_DATA2	I/O data signal 1 port 2 for serial flash device B	EPDC1_DATA10	ALT2	IO
QSPI_B_DATA3	I/O data signal 1 port 3 for serial flash device B	EPDC1_DATA11	ALT2	IO
QSPI_B_DQS	Data strobe signal 1 to serial flash device B	EPDC1_DATA12	ALT2	I
QSPI_B_SCLK	Serial clock output 1 to serial flash device B	EPDC1_DATA13	ALT2	O
QSPI_B_SS0_B	Chip select 1 port 0 for serial flash device B	EPDC1_DATA14	ALT2	O
QSPI_B_SS1_B	Chip select 1 port 1 for serial flash device B	EPDC1_DATA15	ALT2	O

10.2.2.1 Driving External Signals

The different phases of serial flash access scheme are shown in the following figure.

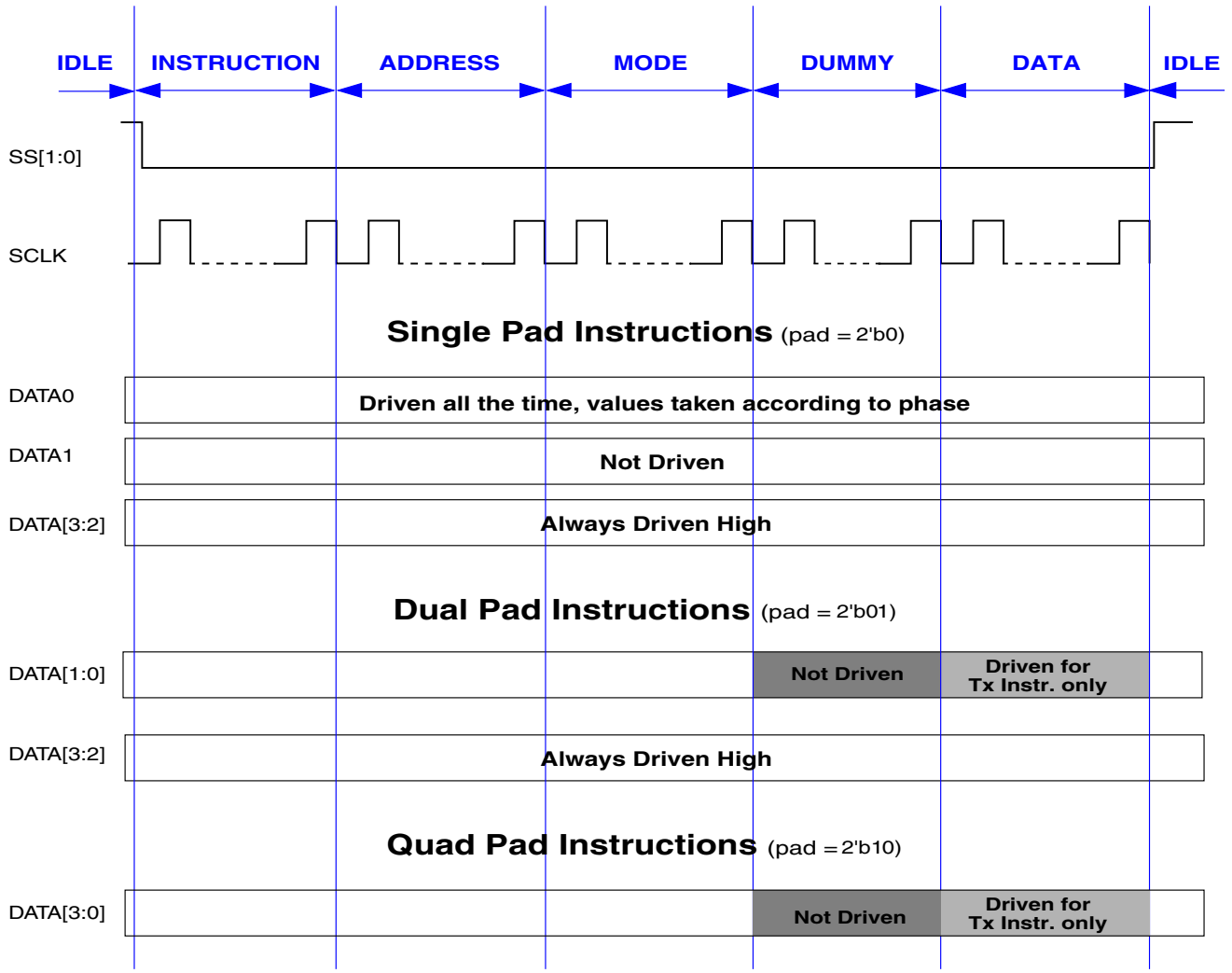


Figure 10-17. Serial Flash Access Scheme

The different phases and the I/O driving characteristics of the QuadSPI module are characterized in the following way:

- IDLE: Serial flash device not selected. No interaction with the serial flash device. All DATA signals driven.
- INSTRUCTION: Serial flash device selected. The instruction is sent to the serial flash device. All DATA signals are driven.
- ADDRESS: Serial Flash Address is sent to the device. All DATA signals are driven. Note that this phase is not applicable for all SFM Commands.
- MODE: Mode bytes are sent to the serial flash device. All DATA signals are driven. Note that this phase is not applicable for all SFM Commands.

- DUMMY: Dummy clocks are provided to the serial flash device. Refer to the Figure 10-17 for the DATA signals driven. The actual data lines required for the SFM Command executed are not driven for data read commands. Note that this phase is not applicable for all SFM Commands.
- DATA: Serial flash data are sent to or received from the serial flash device. Refer to the preceding figure for the DATA signals driven. The actual data lines required for the SFM Command executed are not driven for data read commands. Note that this phase is not applicable for all SFM Commands.

The SS[1:0] and SCLK signals are driven permanently throughout all the phases.

In Individual Flash Mode this applies to the selected flash device. In Parallel Flash Mode this applies to both serial flash devices simultaneously.

10.2.3 Memory Map and Register Definition

This section provides the memory map and register definitions of the QuadSPI module.

10.2.3.1 Register Write Access

This section describes the write access restriction terms that apply to all registers, which can be one of the following:

- **Register Write Access Restriction**

  For each register bit and register field, the write access conditions are specified in the detailed register description. A description of the write access conditions is given in the following table. If, for a specific register bit or field, none of the given write access conditions is fulfilled, any write attempt to this register bit or field is ignored without any notification. The values of the bits or fields are not changed.

  The condition term [A or B] indicates that the register or field can be written to if at least one of the conditions is fulfilled.

Table 10-6. Register Write Access Restrictions

Condition	Description
Anytime	No write access restriction.
Disabled Mode	Write access only if *QSPI_MCR[MDIS] = 1*.
Normal Mode	Write access only if the module is in *Normal Mode*.

- **Register Write Access Requirements**

  All registers can be accessed with 8-bit, 16-bit, and 32-bit wide operations. For some of the registers, at least a 16/32-bit wide write access is required to ensure correct operation. This write access requirement is stated in the detailed register description for each register affected.

10.2.3.2 Serial Flash Address Assignment

The serial flash address assignment may be modified by writing into Serial Flash A1 Top Address (QuadSPI_SFA1AD) and Serial Flash A2 Top Address (QuadSPI_SFA2AD) for device A and into Serial Flash B1Top Address (QuadSPI_SFB1AD) and Serial Flash B2Top Address (QuadSPI_SFB2AD) for device B. The following table shows how different access modes are related to the address specified for the next SFM Command. Note that this address assignment is valid for both IP and AHB commands.

Table 10-7. Serial Flash Address Assignment

Parameter	Function	Access Mode
QSPI_AMBA_BASE ((31:10) - 22 bits)	QuadSPI AHB base address	
TOP_ADDR_MEMA1(TPADA1)	Top address for the external flash A1 (first device of the dual die flash A, or the first of the two independent flashes sharing the IOFA)	Any access to the address space between TOP_ADDR_MEMA1 and QSPI_AMBA_BASE will be routed to **Serial Flash A1**
TOP_ADDR_MEMA2(TPADA2)	Top address for the external flash A2 (second device of the dual die flash A, or the second of the two independent flashes sharing the IOFA).	Any access to the address space between TOP_ADDR_MEMA2 and TOP_ADDR_MEMA1 will be routed to **Serial Flash A2**
TOP_ADDR_MEMB1(TPADB1)	Top address for the external flash B1 (first device of the dual die flash B, or the first of the two independent flashes sharing the IOFB)	Any access to the address space between TOP_ADDR_MEMB1 and TOP_ADDR_MEMA2 will be routed to **Serial Flash B1**
TOP_ADDR_MEMB2(TPADB2)	Top address for the external flash B2 (second device of the dual die flash B or the second of the two independent flashes sharing the IOFB)	Any access to the address space between TOP_ADDR_MEMB2 and TOP_ADDR_MEMB1 will be routed to **Serial Flash A2**

10.2.3.3 AMBA Bus Register Memory Map

QSPI_AMBA_BASE defines the address to be used as start address of the serial flash device as defined by the system memory map..

Table 10-8. QuadSPI AMBA Bus Memory Map

Address	Register Name
Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A	
QSPI_AMBA_BASE to (TOP_ADDR_MEMA2 - 0x01)	Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A Refer to Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A for details and to Table 10-16 and Table 10-20 for information about the byte ordering.
Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B	
TOP_ADDR_MEMA2 to (TOP_ADDR_MEMB2 - 0x01)	Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B Refer to Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B for details and to Table 10-16 and Table 10-20 for information about the byte ordering.
Parallel Flash Mode	
QSPI_AMBA_BASE to (TOP_ADDR_MEMB2 - 0x01)	Parallel Flash Mode Refer to Parallel Flash Mode for details and to Table 10-19 and Table 10-20 for information about the byte ordering.
AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31)	
QSPI_ARDB_BASE to… (32 * 4 Byte) QSPI_ARDB_BASE + 0x0000_01FF	AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31) Refer to Table 10-16 and Table 10-18 for information about the byte ordering.

Note

Any read access to non-implemented addresses will provide undefined results.

In case single die flash devices, TOP_ADDR_MEMA2 and TOP_ADDR_MEMB2 should be initialized/programmed to TOP_ADDR_MEMA1 and TOP_ADDR_MEMB1 respectively- in effect, setting the size of these devices to 0. This would ensure that the complete memory map is assigned to only one flash device.

Parallel Flash Mode is valid only for commands related to data read from the serial flash. The first device of flash A has to be paired with the first device of flash B and the second device of flash A has to be paired with the second device of flash B in parallel mode. Parallel mode is selected via the QSPI_BFGENCR[PAR_EN] bit for all masters in AHB driven mode and via the QSPI_IPCR[PAR_EN] in IP driven mode. In parallel mode, the incoming address (SFAR address in case of IP initiated transactions and the incoming AHB address in case of AHB initiated transactions) is divided by 2 and sent to the two flashes connected in parallel.

Any IP Command other than data read in Parallel Flash Mode will result in the assertion of the QSPI_FR[IUEF] flag and any AHB Command other than data read in Parallel Flash Mode will result in the assertion of the QSPI_FR[ABSEF] flag.

In the Individual Flash Modes, the 3/4 address bytes (as programmed in the instruction/operand in the sequence) available for the flash address is determined by SFADR [23:0] or SFADR [31:0]as given in the table above.

In Parallel Flash Mode, both flashes are read with the same starting address of 3/4 (as programmed in the instruction/ operand in the sequence) bytes in size. This address is derived from SFADR [24:1] or SFADR [31:1] as given in the table above. The LSB of the SFADR field is used to select the appropriate bits of both flash devices to combine the byte corresponding to the selected address.

10.2.3.4 AHB Bus Register Memory Map Descriptions

This chapter contains definitions of registers in the AMBA address space.

10.2.3.4.1 AHB Bus Access Considerations

It has to be noted that all logic in the QuadSPI module implementing the AHB Bus access is designed to read the content of an external serial flash device. Therefore the following restrictions apply to the QuadSPI module with respect to accesses to the AHB bus:

- Any write access is answered with the ERROR condition according to the AMBA AHB Specification. No write occurs.
- Any AHB Command resulting in the assertion of the QSPI_FR[ABSEF] flag is answered with the ERROR condition according to the AMBA_AHB specification. The resulting AHB Command is ignored.
- AHB Bus access types fully supported are NONSEQ and BUSY.
- AHB access type SEQ is treated in the same way like NONSEQ. Refer to the AMBA AHB Specification for further details.

10.2.3.4.2 Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A

Starting with address QSPI_AMBA_BASE the content of the first external serial flash devices is mapped into the address space of the device containing the QuadSPI module. Serial flash address byte address 0x0 corresponds to bus address QSPI_AMBA_BASE with increasing order. Assuming that a dual-die flash is connected on the first set of external pads, the address space is divided into two parts, one for each device of the dual die package. Refer to the following table for the address mapping. The byte ordering for 32 bit access is given in Table 10-16 and for 64 bit read access the byte ordering is given in Table 10-20.

Table 10-9. Memory Mapped Individual Flash Mode - Flash A Address Scheme

Memory Mapped Address 32 Bit Access	Memory Mapped Address 64 Bit Access	Serial Flash Byte Address	Flash Device
QSPI_AMBA_BASE + 0x00	QSPI_AMBA_BASE + 0x00	0x00_0000 to 0x00_0003	A1
QSPI_AMBA_BASE + 0x04		0x00_0004 to 0x00_0007	
…	…	…	
TOP_ADDR_MEMA1 - 0x08	TOP_ADDR_MEMA1 - 0x08	(TOP_ADDR_MEMA1- 0x08) to (TOP_ADDR_MEMA1 - 0x04 -0x01)	
TOP_ADDR_MEMA1 - 0x04		(TOP_ADDR_MEMA1 - 0x04) to (TOP_ADDR_MEMA1 - 0x01)	
TOP_ADDR_MEMA1 + 0x00	TOP_ADDR_MEMA1 + 0x00_0000	0x00_0000 to 0x00_0003	A2
TOP_ADDR_MEMA1 + 0x04		0x00_0004 to 0x00_0007	
…..	…	…	
TOP_ADDR_MEMA2 - 0x08	TOP_ADDR_MEMA2 - 0x08	(TOP_ADDR_MEMA2 - 0x08) to (TOP_ADDR_MEMA2 - 0x04 - 0x01)	
TOP_ADDR_MEMA2 - 0x04		(TOP_ADDR_MEMA2 - 0x04) to (TOP_ADDR_MEMA2 - 0x01)	

The available address range depends from the size of the external serial flash device. Any access beyond the size of the external serial flash provides undefined results.

For details concerning the read process refer to Flash Read.

10.2.3.4.3 Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B

Starting with address TOP_ADDR_MEMA2 the content of the first external serial flash devices is mapped into the address space of the device containing the QuadSPI module. Serial flash address byte address 0x0 corresponds to bus address TOP_ADDR_MEMA2 with increasing order. Assuming that a dual-die flash is connected on the first set of external pads, the address space is divided into two parts, one for each device of the dual

die package. Refer the following table for the address mapping. The byte ordering for 32 bit access is given in Table 10-16 and for 64 bit read access the byte ordering is given in Table 10-20.

Table 10-10. Memory Mapped Individual Flash Mode - Flash B Address Scheme

Memory Mapped Address 32 Bit Access	Memory Mapped Address 64 Bit Access	Serial Flash Byte Address	Flash Device
TOP_ADDR_MEMA2 + 0x00	TOP_ADDR_MEMA2 + 0x00	0x00_0000 to 0x00_0003	B1
TOP_ADDR_MEMA2 + 0x04		0x00_0004 to 0x00_0007	
…	…	…	
TOP_ADDR_MEMB1 - 0x08	TOP_ADDR_MEMB1 - 0x08	(TOP_ADDR_MEMB1- TOP_ADDR_MEMA2 - 0x08) to (TOP_ADDR_MEMB1 - TOP_ADDR_MEMA2 - 0x04 -0x01)	
TOP_ADDR_MEMB1 - 0x04		(TOP_ADDR_MEMB1- TOP_ADDR_MEMA2 - 0x04) to (TOP_ADDR_MEMB1- TOP_ADDR_MEMA2- 0x01)	
TOP_ADDR_MEMB1 + 0x00	TOP_ADDR_MEMB1 + 0x00_0000	0x00_0000 to 0x00_0003	B2
TOP_ADDR_MEMB1 + 0x04		0x00_0004 to 0x00_0007	
…..	…	…	
TOP_ADDR_MEMB2 - 0x08	TOP_ADDR_MEMA2 - 0x08	(TOP_ADDR_MEMB2 - TOP_ADDR_MEMB1- 0x08) to (TOP_ADDR_MEMB2 - TOP_ADDR_MEMB1 - 0x04 - 0x01)	
TOP_ADDR_MEMB2 - 0x04		(TOP_ADDR_MEMB2- TOP_ADDR_MEMB1 - 0x04) to (TOP_ADDR_MEMB2 - TOP_ADDR_MEMB1 - 0x01)	

The available address range depends from the size of the external serial flash device. Any access beyond the size of the external serial flash provides undefined results.

For details concerning the read process refer to Flash Read.

10.2.3.4.4 Parallel Flash Mode

Any of the AHB flexible-buffers can be configured to work in parallel flash mode by programming the QSPI_BFGENCR[PAR_EN] bit to '1'. When parallel mode is set, Flash A1 is paired with Flash B1 and Flash A2 is paired with Flash B2. In parallel mode, software should ensure that the size of Flash A1(A2) is equal to the size of Flash B1(B2).

Reads from any even AHB bus address provides bits [7:4] of both serial flash devices and reads from any odd AHB bus address provides bits [3:0] of both flash devices. Refer to the following table for the address mapping. The byte ordering for 32 bit access is given in Table 10-18 and for 64 bit read access the byte ordering is given in Table 10-20.

Table 10-11. Memory Mapped Parallel Flash Mode Address Scheme

Memory Mapped Address 32 Bit Access	Memory Mapped Address 64 Bit Access	Serial Flash A Byte Address	Serial Flash B Byte Address
QSPI_AMBA_BASE + 0x0000_0000	QSPI_AMBA_BASE + 0x00 For details, please refer to Parallel mode and Dual Die Flashes.	0x00_0000 - 0x00_0001	0x00_0000 - 0x00_0001
QSPI_AMBA_BASE + 0x0000_0004		0x00_002 - 0x00_0003	0x00_0002 - 0x00_0003
QSPI_AMBA_BASE + 0x0000_0008	QSPI_AMBA_BASE + 0x08	0x00_0004 - 0x00_0005	0x00_0004 - 0x00_0005
QSPI_AMBA_BASE + 0x0000_000C		0x00_0006 - 0x00_0007	0x00_0006 - 0x00_0007
…	…	…	…
TOP_ADDR_MEMB2 - 0x08	TOP_ADDR_MEMB2 - 0x08	(TOP_ADDR_MEMB2 - QSPI_AMBA_BASE - 0x08)/2	(TOP_ADDR_MEMB2 - QSPI_AMBA_BASE - 0x08)/2
TOP_ADDR_MEMB2 - 0x04		(TOP_ADDR_MEMB2 - QSPI_AMBA_BASE - 0x04)/2 +0x01	(TOP_ADDR_MEMB2 - QSPI_AMBA_BASE- 0x04)/2 + 0x01

The available address range covers 27 address bits, corresponding to 128 MB per flash device. The usable space depends from the size of the external serial flash devices. Any access beyond the size of the external serial flash provides undefined results.

For details concerning the read process refer to Flash Read.

10.2.4 Interrupt Signals

The interrupt request lines of the QuadSPI module are mapped to the internal flags according to the following table.

Table 10-12. Assignment of Interrupt Request Lines

IRQ/DMA line	QSPI_FR Flag	Interrupt Description
ipi_int_tfff	TBFF	TX Buffer Fill
ipi_int_tcf	TFF	Peripheral Command Transaction Finished
ipi_int_rfdf	RBDF	RX Buffer Drain
ipi_int_overrun		Buffer Overflow/Underrun Error Logical OR from:

Table continues on the next page...

Table 10-12. Assignment of Interrupt Request Lines (continued)

IRQ/DMA line	QSPI_FR Flag	Interrupt Description
	RBOF	RX Buffer Overrun
	TBUF	TX Buffer Underrun
	ABOF	AHB Buffer Overflow
ipi_int_cerr		Serial Flash Command Error Logical OR from:
	IPAEF	Peripheral access while AHB busy Error
	IPIEF	Peripheral Command could not be triggered Error
	IPGEF	Peripheral access while AHB Grant Error
	IUEF	Peripheral Command Usage Error
ipi_int_ored	DLPFF, TBFF, TFF, ILLINE, RBDF, RBOF, TBUF, ABSEF, ABOF, IPAEF, IPIEF, IPGEF, IUEF	Logical OR from all the QSPI_FR flags mentioned

10.2.5 Functional Description

This section provides the functional information of the QuadSPI module.

10.2.5.1 Serial Flash Access Schemes

The Quad Serial Peripheral Interface (QuadSPI) block acts as an interface to one single or two external serial flash devices, each with up to 4 bidirectional data lines. Depending from the serial flash devices attached to the QuadSPI module the following access schemes are possible:

Table 10-13. Access Schemes for Serial Flash Data Access

Access Scheme	One Flash Device on Port A	One Flash Device on Port B	Two identical Flash Devices connected on Port A and Port B
Individual Flash Mode: Access to Flash A	**Yes**	**N/a**	**Yes**
Individual Flash Mode: Access to Flash B	**N/a**	**Yes**	**Yes**
Parallel Flash Mode: Read from Flash A and Flash B	**N/a**	**N/a**	**Yes**

Note

If two flash devices are accessed in Parallel Flash Mode, they are accessed with identical control signals. Special alignment on per-flash basis is **not** possible. It is within the responsibility of the application to ensure that the identical signals are applicable to both flash devices.

In Parallel Flash Mode, both external serial flash devices appear logically as one single memory doubled in size with respect to one individual flash device.

If two different flash devices are attached, they can be operated only in Individual Flash Mode.

In the Parallel Flash Mode, only data read commands are supported. Any other IP Command will result in an error condition signaled by the assertion of the QSPI_FR[IUEF] flag and any other AHB Command will result in the assertion of the QSPI_FR[ABSEF] flag.

In the Individual Flash Mode, all supported commands are available.

Unless explicitly noted, all the following descriptions relate to the Individual Flash Mode.

10.2.5.2 Modes of Operation

Refer to QuadSPI Modes of Operation for an overview over the possible operational modes of the QuadSPI block.

- Normal Mode can be used for write or read accesses to an external serial flash device.
  - Serial Flash Write: Data can be programmed into the flash via the IP interface only. Refer to Flash Programming for further details.
  - Serial Flash Read: Read the contents of the serial flash device. Two separate read channels are available via RX Buffer and AHB Buffer, see Flash Read.

- Stop Mode: The mode is used for power management. When a request is made to enter Stop Mode, the QuadSPI block acknowledges the request and completes the SFM Command in progress, then the system clocks to the QuadSPI block may be shut off
- Module Disable Mode: The mode is used for power management. The clock to the non-memory mapped logic in the QuadSPI can be stopped while in Module Disable Mode. The module enters the mode by setting QSPI_MCR[MDIS].

10.2.5.3 Normal Mode

This mode is used to allow communication with an external serial flash device. Compared to the standard SPI protocol, this communication method uses up to 4 bidirectional data lines operating at high data rates. The communication to the external serial flash device consists of an instruction code and optional address, mode, dummy and data transfers. The flexible programmable core engine described below is immune to a wide variety of command/protocol differences in the serial flash devices provided by various flash vendors.

10.2.5.3.1 Programmable Sequence Engine

The core of the QuadSPI module is a programmable sequence engine that works on "instruction-operand" pairs. The core controller executes each programmed instruction sequentially. The complete list of instructions and the corresponding operands is given in the following table.

Table 10-14. Instruction set

Instruction	Instruction encoding	Pins	Operand	Action on Serial Flash(es)
CMD	6'd1	N=2'd{0,1,2} 2'd0 - One pad 2'd1 - Two pads 2'd2 - Four pads	8 bit command value	Provide the serial flash with operand on the number of pads specified
ADDR	6'd2		Number of address bits to be sent (for example, 8'd24 => 24 address bits required)	Provide the serial flash with address cycles according to the operand on the number of pads specified. The actual address to be provided will be derived from the incoming address in case of AHB initiated transactions and the value of SFAR in case of IPS initiated transactions .
DUMMY	6'd3		Number of dummy clock cycles (should be <= 64 cycles)	Provide the serial flash with dummy cycles as per the operand. The PAD information defines the number of pads in input mode. (for example, one pad implies that pad 1 is not driven, rest all are driven)

Table continues on the next page...

Table 10-14. Instruction set (continued)

Instruction	Instruction encoding	Pins	Operand	Action on Serial Flash(es)
MODE	6'd4		8 bit mode value	Provide the serial flash with 8 bit operand on the number of pads specified
MODE2	6'd5	N=2'd{0,1}	2 bit mode value	Provide the serial flash with 2 bit operand on the number of pads[1] specified
MODE4	6'd6	N=2'd{0,1,2}	4 bit mode value	Provide the serial flash with 4 bit operand on the number of pads[2] specified
READ	6'd7	N=2'd{0,1,2} 2'd0 - One pad 2'd1 - Two pads 2'd2 - Four pads	Read data size in bytes (for AHB transactions, the user's application should ensure that data size is a multiple of 8 bytes)	Read data from flash on the number of pads specified. The data size may be overwritten by writing to the ADATSZ field of the QSPI_BUFxCR registers for AHB initiated transactions and IDATSZ field of IP Configuration Register (QuadSPI_IPCR) for IP initiated transactions.
WRITE	6'd8		Write data size in bytes	Write data on number of pads sepcified. The data size may be overwritten by writing to the IDATSZ field of IP Configuration Register (QuadSPI_IPCR) register
JMP_ON_CS	6'd9	NA	Instruction number	Every time the CS is deasserted, jump to the instruction pointed to by the operand. This instruction allows the programmer to specify the behavior of the controller when a new read transaction is initiated following a CS deassertion.
ADDR_DDR	6'd10	N=2'd{0,1,2} 2'd0 - One pad 2'd1 - Two pads 2'd2 - Four pads	Number of address bits to be sent (for example, 8'd24 => 24 address bits required)	Provide the serial flash with address cycles according to the operand on the number of pads specified at each clock edge of serial flash clock. The actual address to be provided will be derived from the incoming address in case of AHB initiated transactions and the value of QSPI_SFAR in case of IPS initiated transactions .
MODE_DDR	6'd11		8 bit mode value	Provide the serial flash with 8 bit operand on the number of pads specified at each clock edge of serial flash.
MODE2_DDR	6'd12	N=2'd{0}	2 bit mode value	Provide the serial flash with 2 bit operand on the number of pads specified at each clock edge of serial flash[3]
MODE4_DDR	6'd13	N=2'd{0,1}	4 bit mode value	Provide the serial flash with 4 bit operand on the number of pads specified at each clock edge of serial flash[4].
READ_DDR	6'd14	N=2'd{0,1,2} 2'd0 - One pad 2'd1 - Two pads 2'd2 - Four pads	Read data size in bytes (for AHB transactions, the user's application should ensure that data size is in multiple of 8 bytes)	Read data from flash on the number of pads specified at each clock edge of serial flash. The data size may be overwritten by writing to the ADATSZ field of the QSPI_BUFxCR registers for AHB initiated transactions and IDATSZ field of IP Configuration Register (QuadSPI_IPCR) for IP initiated transactions
WRITE_DDR	6'd15		Write data size in bytes	Write data on the number of pads specified at each clock edge of serial flash. The data size may be overwritten by writing to the IDATSZ field of IP Configuration Register (QuadSPI_IPCR) register

Table continues on the next page...

Table 10-14. Instruction set (continued)

Instruction	Instruction encoding	Pins	Operand	Action on Serial Flash(es)
DATA_LEARN[5]	6'd16		8 bit Data learning pattern	Find the correct sampling point with the data learning pattern. When this instruction is encountered, the QSPI_SMPR[DDRSMP] values are ignored and the controller finds the correct sampling point on its own by sampling the data learning pattern.[6]
STOP	8'd0	NA	NA	Stop execution; deassert CS

1. For a one pad instruction, MODE2 will take 2 serial flash clock cycles on the flash interface.
2. For a one pad instruction, MODE4 will take 4 serial flash clock cycles on the flash interface. For a 4 pad instruction, MODE4 will take 1 serial flash clock cycle on the flash interface.
3. For a one pad instruction, MODE2_DDR will take 1 serial flash clock cycle on the flash interface.
4. For a one pad instruction, MODE4_DDR will take 2 serial flash clock cycles on the flash interface. For a 4 pad instruction MODE4_DDR will take half a cycle on the serial flash interface.
5. Data learning is not implemented on this chip.
6. t is not recommended to have 0x00 or 0xFF as the data learning pattern.

A sequence of such instruction-operand pairs may be pre-populated in the LUT according to the device connected on board. Each instruction-operand pair is of 16 bits (2 bytes) each. Every sequence pre-programmed in the LUT is referred to by its index.

The programmable sequence engine allows the user to configure the QuadSPI module according to the serial flash connected on board. The flexible structure is easily adaptable to new command/protocol changes from different vendors.

10.2.5.3.2 Flexible AHB buffers

In order to reduce the latency of the reads for AHB masters, the data read from the serial flash is buffered in flexible AHB buffers. There are four such flexible buffers. The size of each of these buffers is configurable with the minimum size being 0 Bytes and maximum size being the size of the complete buffer instantiated.The size of buffer 0 is defined as being from 0 to QSPI_BUF0IND. The Size of buffer 1 is from QSPI_BUF0IND to QSPI_BUF1IND, buffer2 is from QSPI_BUF1IND to QSPI_BUF2IND and buffer 3 is from QSPI_BUF2IND to the size of the complete buffer, which is given in the chip-specific QuadSPI information.

Each flexible AHB buffer is associated with the following

1. An AHB master. Optionally, buffer3 may be configured as an "all master" buffer by setting the QSPI_BUF3CR[ALLMST] bit. When buffer3 is configured in such a way, any access from a master not associated with any other buffer is routed to buffer3.
2. A datasize field representing the amount of data to be fetched from the flash on every "missed" access.

The master port number of every incoming request is checked and the data is returned/ fetched into the corresponding associated buffer. Every "missed" access to the buffer causes the controller to clear the buffer and fetch QSPI_BUFxCR[ADATSZ] amount of data from the serial flash.As such, there is no benefit in configuring a buffer size of greater than ADATSZ, as the locations greater than ADATSZ will never be used. For any AHB access, the sequence pointed to by the QSPI_BFGENCR[SEQID] field is used for the flash transaction initiated. The data is returned to the master as soon as the requested amount is read from the serial flash. The controller however, continues to prefetch the rest of the data in anticipation of a next consecutive request. Figure 10-18 shows the flexible AHB buffers.

The QSPI_BFGENCR[SEQID] field points to an index of the LUT. Refer to Look-up Table for details.

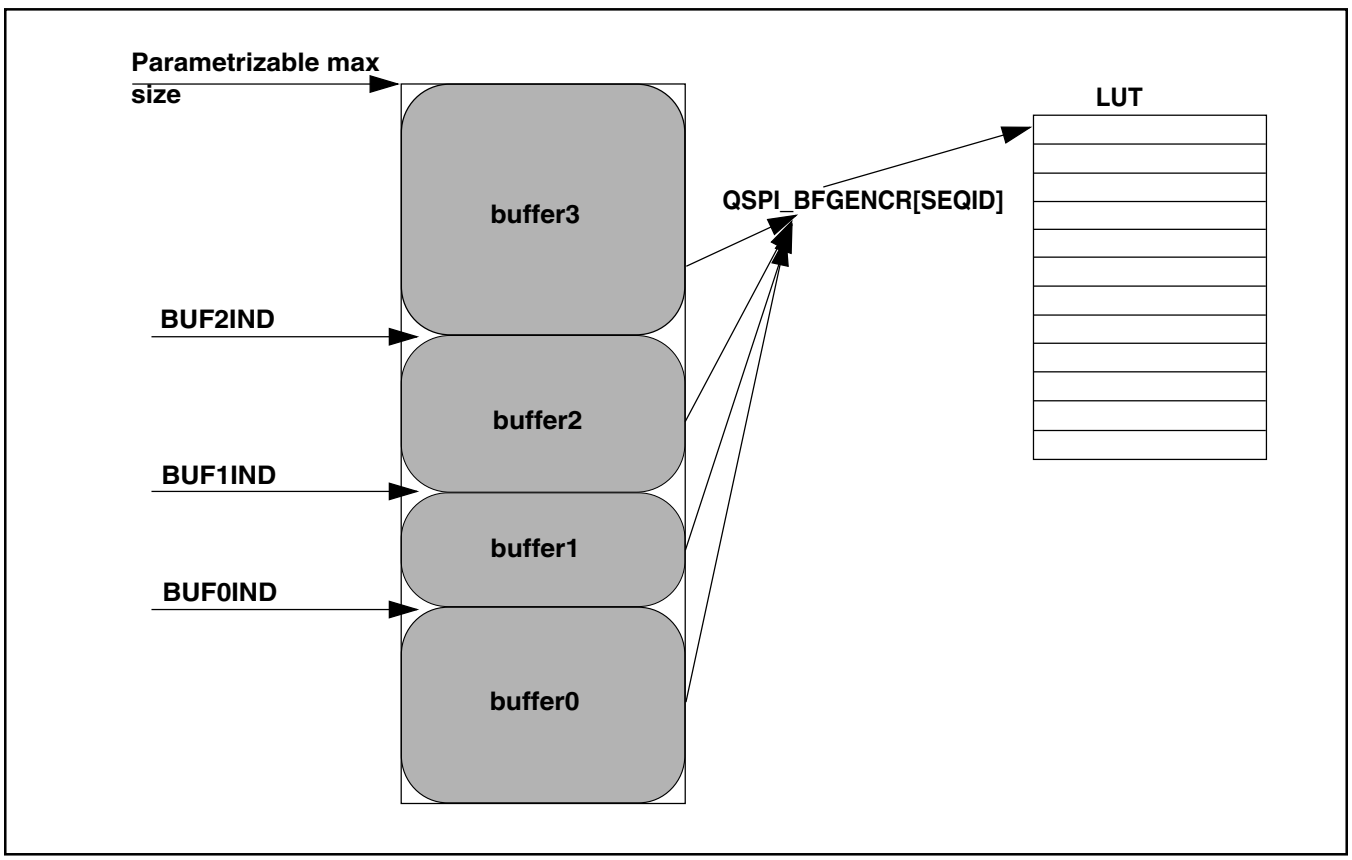


Figure 10-18. Flexible AHB Buffers

Buffer0 may optionally be configured to be associated with a high priority master by setting the QSPU_BUF0CR[HP_EN] bit. An access by a high priority master suspends any ongoing prefetch to any of the other buffers. The ongoing prefetch is suspended and the high priority master is serviced first. Once the high priority masters access completes,

the suspended transaction is resumed (before any other AHB access is entertained). The status of the suspended buffer can be read from Sequence Suspend Status Register (QuadSPI_SPNDST).

10.2.5.3.3 Suspend-Abort Mechanism

Any low priority AHB access can be suspended by a high priority AHB master request. The ongoing transaction is suspended at 64 bit boundary. The suspended transaction is restarted after the high priority master is served and the high priority transaction including data prefetch is completed. While a transaction is in suspended state, it may be aborted if a transaction by the same suspended master is made to a location which is different from the location of the suspended transaction.

Any ongoing transaction is aborted if a request from the same master arrives for a location other than the location at which the transaction is going on. The abort can happen at any point of time.

10.2.5.3.4 Look-up Table

The Look-up-table or LUT consists of a number of pre-programmed sequences. Each sequence is basically a sequence of instruction-operand pairs which when executed sequentially generates a valid serial flash transaction. Each sequence can have a maximum of 8 instruction-operand pairs. The LUT can hold a maximum of 16 sequences. The figure below shows the basic structure of the sequence in the LUT.

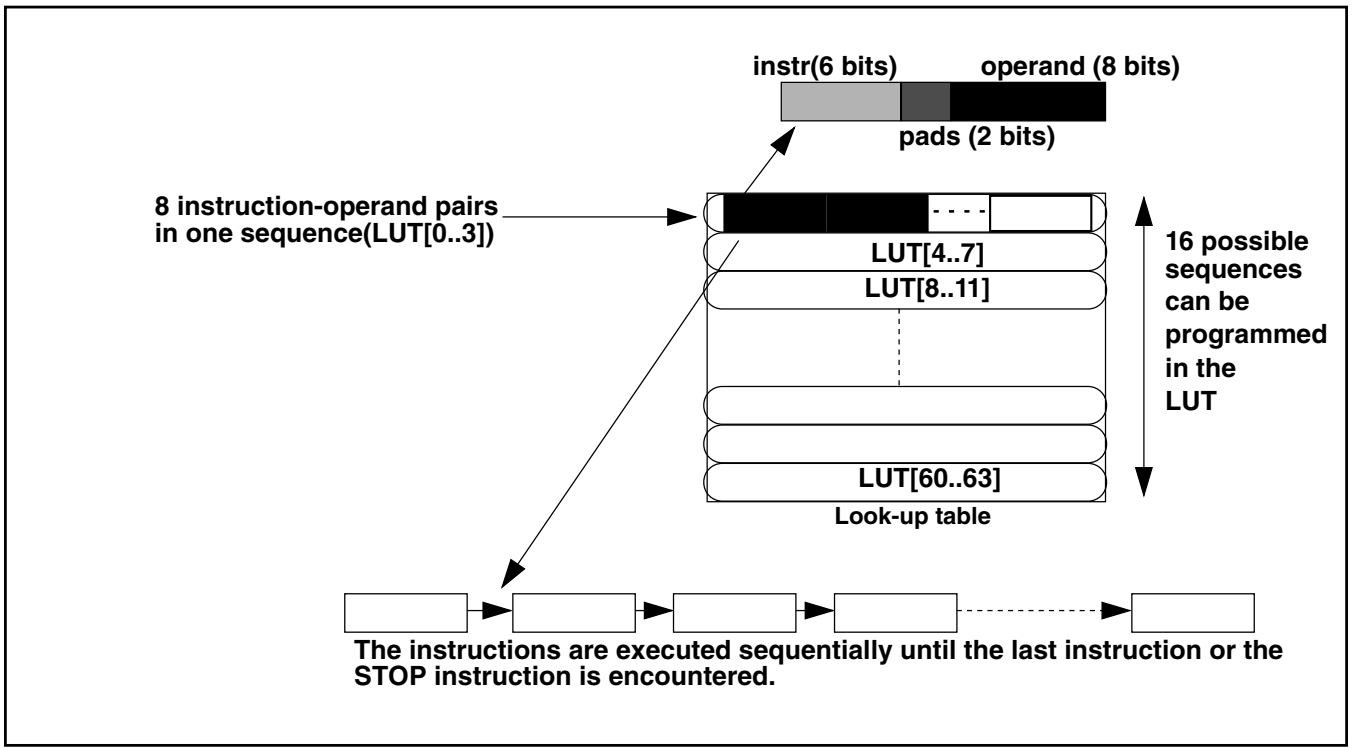


Figure 10-19. LUT and sequence structure

At reset, the index 0 of the look-up-table (LUT[0..3]) is programmed with a basic read sequence as given in Table 10-15. After reset the complete LUT may be reprogrammed according to the device connected on board.In order to protect its contents during a code runover the LUT may be locked, after which a write to the LUT will not be successful until it has been unlocked again.The key for locking or unlocking the LUT is **0x5AF05AF0**. The process for locking and un-locking the LUT is as follows:

Locking the LUT

1. Write the key **(0x5AF05AF0)** in to the LUT Key Register (QuadSPI_LUTKEY).
2. Write 0b01 to the LUT Lock Configuration Register (QuadSPI_LCKCR). Note that this IPS transaction should immediately follow the above IPS transaction (no other IPS transaction can be issued in between). A successful write into this register locks the LUT.

Unlocking the LUT

1. Write the key **(0x5AF05AF0)** into the LUT Key Register (QuadSPI_LUTKEY)
2. Write 0b10 to the LUT Lock Configuration Register (QuadSPI_LCKCR). Note that this IPS transaction should immediately follow the above IPS transaction (no other IPS transaction can be issued in between). A successful write into this register unlocks the LUT.

The lock status of the LUT can be read from QSPI_LCKCR[UNLOCK] and QSPI_LCKCR[LOCK] bit.

Some example sequences are defined in Example Sequences. The reset sequence at LUT index 0 is given in the following table.

Table 10-15. Reset sequence

Instruction	Pad	Operand	Comment
CMD	0x00	0x03	Read Data byte command on one pad
ADDR	0x00	0x18	24 Addr bits to be sent on one pad
READ	0x00	0x08	Read 64 bits
JMP_ON_CS	0x00	0x00	Jump to instruction 0 (CMD)

10.2.5.3.5 Issuing SFM Commands

Each access to the external device follows the same sequence:

1. The user must pre-populate the LUT with the serial flash command sequences that are required for the flash device being used.
2. The QuadSPI module starts executing the instructions in the sequence one by one. The transaction starts and the status bit QSPI_SR[BUSY] is set.
3. Communication with the external serial flash device is started and the transaction is executed.
4. When the transaction is finished (all transmit- and receive operations with the external serial flash device are finished) the status bit QSPI_SR[BUSY] is reset. In case of an IP Command the QSPI_FR[TFF] flag is asserted.

Further details are given in below in Flash Programming and Flash Read.

You can trigger the processing of SFM commands in the QuadSPI module in one of the following ways:

- **Using IP commands**

  For IP Commands the required components need to be written into the following registers:

  - Write the serial flash address to be used by the instruction into QSPI_SFAR, refer to Serial Flash Address Register (QSPI_SFAR). For IP Commands not related to specific addresses, the base address of the related flash need to be

programmed.For example, for an instruction which does not require an address (i.e. write enable instruction) the SFAR should be programmed with the base address of the memory the command is to be sent to.

- Write the sequence ID and data size details in the IP Configuration Register (QSPI_IPCR).
- Note that the write into the QSPI_IPCR[SEQID] field must be the last step of the sequence. It is possible to combine all fields of the QSPI_IPCR into one single write. Refer to IP Configuration Register (QSPI_IPCR) for details.

Note that there are some conditions where no IP Command is executed after writing the QSPI_IPCR[SEQID] field and the write operation itself is ignored. They are described in Command Arbitration.

- **Using AHB commands**

  Any AHB memory mapped access is routed to one of the buffers depending on the master port number of the request. If the access is a "miss", a new serial flash transaction is started. The transaction is based on the sequence pointed to by the BFGENCR[SEQID] field as described in Flexible AHB buffers.

  An AHB access is termed memory mapped when the access is to the memory mapped serial flashes, as described in Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A and Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B.

  Again the possible error conditions are described in Command Arbitration .

10.2.5.3.6 Flash Programming

In all cases the memory sector to be written needs to be erased first. The programming sequence itself is then initiated in the following way:

1. Check that the TX Buffer is empty. If the QSPI_SR[TXEDA] bit is set then the TX Buffer must be cleared by writing 1 into the QSPI_MCR[CLR_TXF] bit.
2. Program the address related to the command in the QSPI_SFAR register.
3. Provide initial data for the program command into the circular buffer via register TX Buffer Data Register (QSPI_TBDR) . At least four word of data must be written into the TX Buffer up to a maximum of 32.

4. Program the QSPI_IPCR register to trigger the command. The QSPI_IPCR[SEQID] should point to an index of the LUT which has the flash program sequence pre-programmed.The IDATSZ field should be set to denote the size of the write.
5. Depending on the amount of data required, step 3 must be repeated until all the required data have been written into the QSPI_TBDR register.The QSPI_SR[TXFULL] can be used to check if the buffer is ready to receive more data. At any time, the QSPI_TBSR[TRCTR] field can be read to check how many words have been written actually into the TX Buffer.

Upon writing the QSPI_IPCR[SEQID] field (refer to step 4) the QuadSPI module will start to execute the programmed sequence. It is the responsibility of the software to ensure that a correct sequence is programmed into the LUT in accordance with the flash memory connected to the module. The data is fetched from the TX Buffer. It consists of **32** entries of 32-bits and is organized as a circular FIFO, whose read pointer is incremented by four after each fetch. When all data are transmitted, the QuadSPI module will return from 'busy' to 'idle'. However, this is not true for the external device since the internal programming is still ongoing. It is up to the user to monitor the relevant status information available from the serial flash device and to ensure that the programming is finished properly.

10.2.5.3.7 Flash Read

Host access to the data stored in the external serial flash device is done in two steps. First, the data must be read into the internal buffers and in the second step these internal buffers can be read by the host.

1. **Reading Serial Flash Data into the QuadSPI Module Internal Buffers**

   A read access to the external serial flash device can be triggered in two different ways:

   - **IP Command Read**: For **reading flash data into the RX Buffer** the user must provide the correct sequence ID in the QuadSPI_IPCR[SEQID] register. The sequence ID points to a sequence in the LUT. It is the responsibility of the software to ensure that a correct read sequence is programmed in the LUT in accordance with the serial flash device connected on board. The user should program the Serial Flash Address Register (QSPI_SFAR) and the IP Configuration Register (QSPI_IPCR) registers. All available read commands supported by the external serial flash are possible.

     Optionally it is possible to clear the RX Buffer pointer prior to triggering the IP Command by writing a 1 into the QuadSPI_MCR[CLR_RXF] field.

From these inputs, the complete transaction is built when the QSPI_IPCR[SEQID] field is written. The transaction related to the read access starts and the requested number of bytes is fetched from the external serial flash device into the RX Buffer. Since the read access is triggered by an IP command, the IP_ACC status bit and the BUSY bit are both set (both are located in the Status Register (QSPI_SR)). A count of the number of entries currently in the Rx Buffer can be obtained from QSPI_RBCT[RXBRD].

The communication with the external serial flash is stopped when the specified number of bytes has been read (successful completion of the transaction).

- **AHB Command Read**: For **reading flash data into the AHB Buffer** the user must set up a read access by a master to the address range in the system memory map which the external serial flash devices are mapped to. The user should also program the buffer registers corresponding to the AHB master initiating the request, this is depends on the configuration of the QSPI_RBCT[RXBRD]. The user should provide the correct sequence ID into the buffer generic configuration register (QSPI_BFGENCR). It is the responsibility of the software to ensure that a correct read sequence in programmed in the LUT in accordance with the serial flash device connected on board. Flash device selection and access mode are determined by the address accessed in the AHB address space associated to the QuadSPI module (refer to Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A, Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B and Parallel Flash Mode.)

  On each AHB read access to the memory mapped area the valid data in the AHB Buffer is checked against the address requested in the actual read. When the AHB read request can't be served from the content of the AHB Buffer, the buffer is flushed and the sequence pointed to by the sequence ID is executed by the controller. The requested number of buffer entries defined in the QSPI_BUFxCR[ADATSZ] field is then fetched from the external serial flash device into the internal AHB Buffer. Since the read access is triggered via the AHB bus, the QSPI_SR[AHB_ACC] status bit is set driving in turn the QSPI_SR[BUSY] bit until the transaction is finished. The communication with the external serial flash is stopped when the specified number of entries has been filled.

2. **Data Transfer from the QuadSPI Module Internal Buffers**

   The data read out from the external serial flash device by the QuadSPI module is stored in the internal buffers.The means of accessing the data from the buffer differs depending on which buffer the data has been loaded to. Refer to Block Diagram for details about the two available buffers, the RX Buffer and the AHB Buffer, in the QuadSPI module:

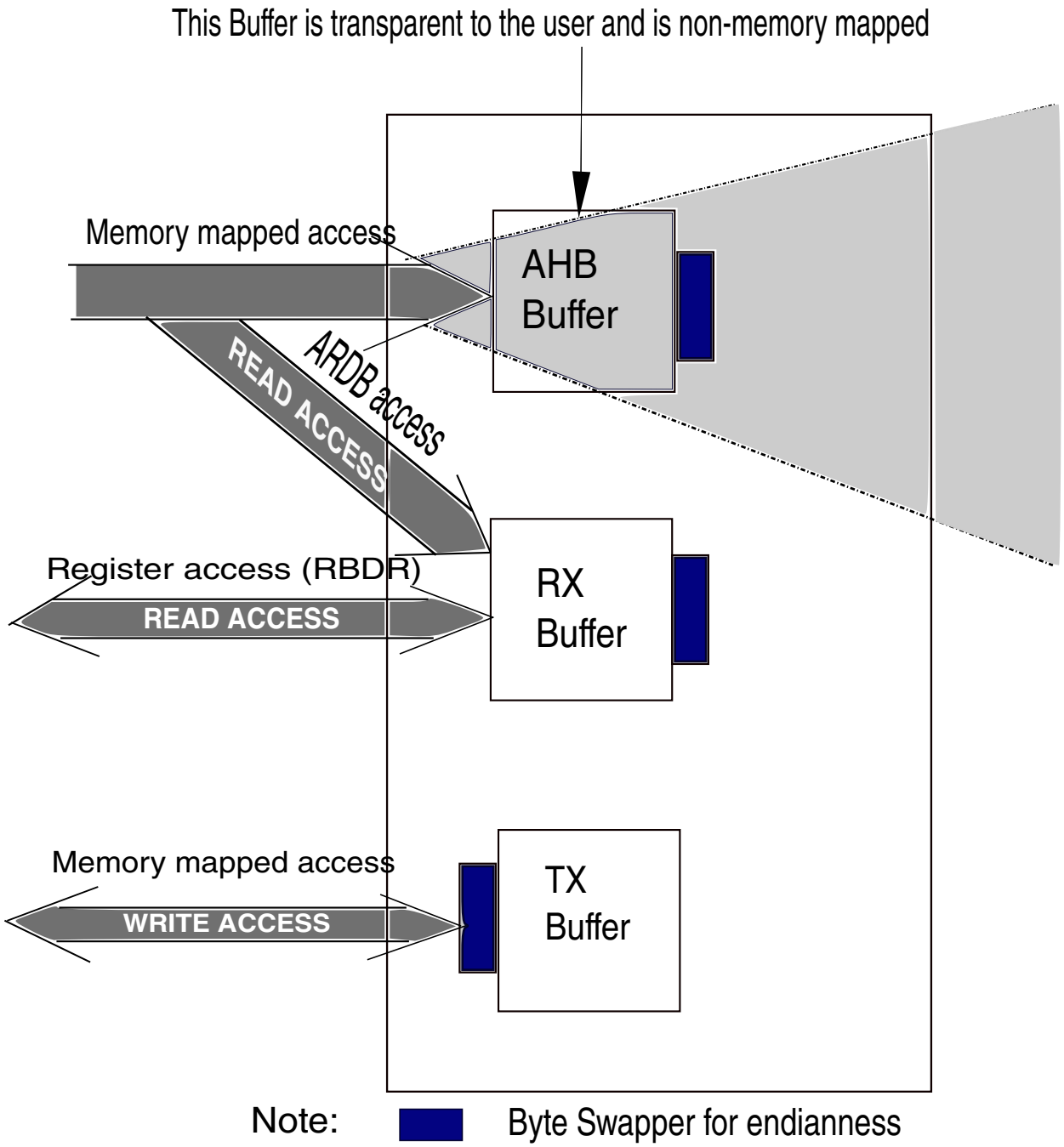


Figure 10-20. QuadSPI memory map

- The RX Buffer is implemented as FIFO of depth 32 entries of 4 bytes. Its content is accessible in two different address areas both referring to the identical data and the same physical memory.

  In the IPS address space in the area associated to QSPI_RBDR0 to QSPI_RBDR31

  In the AHB address space in the area associated to QSPI_ARDB0 to QSPI_ARDB31. Two successive entries are accessed with one single 64 bit AHB read operation.

RX Buffer operation can be summarized as follows: The QSPI_RBCT[WMRK] field determines at which fill level the RXWE bit is asserted and how many entries are removed from the RX Buffer on each Buffer POP operation. So the QSPI_SR[RXWE] bit indicates that the configured number of data entries is available in the RX Buffer and the QSPI_RBSR[RDBFL] field indicates how many valid entries are available in total. Note that the first entry (QSPI_RBDR0 or QSPI_ARDB0) always corresponds to the first valid entry in the RX Buffer. The software needs to manage the number of valid data bytes itself.

Further details can be found in RX Buffer Data Register (QuadSPI_RBDR*n*) and in AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31).

- **Flag-based Data Read of the RX Buffer** is done by polling the QSPI_SR[RXWE] bit. When it is asserted the valid entries can be read either via the IPS address space (QSPI_RBDRn) or the AHB address space (QSPI_ARDBn). A Buffer POP operation must be triggered by the application by writing a 1 into the QSPI_FR[RBDF] bit - this automatically updates the FIFO to point to the next entry as defined by RBCT[WMRK]. For example, if WMRK is set to 3, then the buffer will discard 16 bytes of data.
- **DMA controlled Data Read of the RX Buffer** is done by using the DMA module.The application must ensure that the DMA controller of the related device is programmed appropriately like it is described in DMA Usage.

  DMA controlled read out is triggered fully automatically by the assertion of the QSPI_SR[RXWE] bit. The related Buffer POP operation is also handled completely inside the QuadSPI module. Like in the case above, accessing the RX Buffer content either on QSPI_RBDRn or QSPI_ARDBn related addresses is equivalent.

- **AHB Buffer data read via memory mapped access:** This kind of access is done by reading one of the addresses assigned to the external serial flash device(s) within the range given in Table 10-8 table *under the condition that the data requested are already present in the AHB Buffer or it is currently being read from the serial flash device by the instruction in progress.* If this is not the case a memory mapped AHB command read is triggered as described above. If the requested data is already available in the AHB Buffer they are provided directly to the host.

  When AHB access are made to the flash memory mapped address, the data will be fetched and returned to the AHB interface. Till the data is being fetched the AHB interface would be stalled. As soon as the data from the requested address has been read by the QuadSPI module the AHB read access is served. So it is possible to run sequential reads from the AHB buffer at arbitrary speed without

the need to monitor any information about the availability of the data. Nevertheless this access scheme stalls the AHB bus for the time required to read the data from the serial flash device. A better way is (when it is known the access is sequential) to have a prefetch enabled (by programming the ADATSZ field) such that before the next sequential AHB access come, the data is already fetched into the buffer.

As long as the host restricts its accesses to the data already in the buffer and the data currently fetched from the serial flash, it is possible to run the host read from the AHB Buffer in parallel to the serial flash read into the AHB Buffer.

10.2.5.3.8 Byte Ordering of Serial Flash Read Data

In this paragraph the byte ordering of the serial flash data is given. The basic scheme is that the **first** byte read out of the serial flash device - which is addressed by the QSPI_SFAR[SFADR] field - corresponds to bit position QSPI_RBDR0[31:24] register for IP Command read. In contrast to that for AHB Command read the bytes are always positioned according to the byte ordering of the AHB bus.

- **Byte Ordering in Individual Flash Mode**

  The following table gives the byte ordering scheme of how the byte oriented data space of the serial flash device is mapped into one single 32 bit entry of the RX Buffer or the AHB Buffer. The table is valid within the following context:

  - Flash A or Flash B in Individual Flash Mode
  - All AHB data read commands with access size of 32 bit

Table 10-16. Byte Ordering in Individual Flash Mode

Serial Flash Byte Numbering	3	2	1	0
Buffer Entry Bit Position [31:0] (32 Bit data width)	[31:24]	[23:16]	[15:8]	[7:0]

Note

For IP Commands the read size can be given in number of bytes. If this number is not a multiple of 4, then the last buffer entry is not completely filled with the missing higher numbered bytes at undefined values.

For AHB Commands, reads, starting from an address not aligned to 32 bit boundaries, the requested bytes are given at the appropriate positions according to the AMBA AHB specification.

- **Byte Ordering in Parallel Flash Mode**

  In Parallel Flash Mode each byte is combined out of 2 half bytes which are read in parallel from the two serial flash devices. The following tables shows how the flash content is separated into the half bytes and how the half bytes are assembled to the content of the QSPI_RBDR0 register.

Table 10-17. Serial Flash Device Half Byte Ordering

Serial Flash Device Byte #	Flash A Bit Position		Flash B Bit Position	
	[7:4]	[3:0]	[7:4]	[3:0]
0	fah0	fal0	fbh0	fbl0
1	fah1	fal1	fbh1	fbl1
2	fah2	fal2	fbh2	fbl2
3	fah3	fal3	fbh3	fbl3
4	fah4	fal4	fbh4	fbl4
5	fah5	fal5	fbh5	fbl5
6	fah6	fal6	fbh6	fbl6
7	fah7	fal7	fbh7	fbl7
8	fah8	fal8	fbh8	fbl8

The table entry naming reflects the half byte positioning in the serial flash devices:

- **<fa>**h0 means **Flash A**, <fb>h0 means Flash B.
- fa**<h>**0 means half byte in **high position**, fa<l>o means half byte in low position.
- fah**<0>** means **physical byte address 0** in the serial flash device, fal<1> means physical byte address 1 in the serial flash device.

Table 10-18. Byte Ordering in Parallel Flash Mode - RX Buffer

QSPI_SFAR[SFADR] set to 0x000_0000

Table continues on the next page...

Table 10-18. Byte Ordering in Parallel Flash Mode - RX Buffer (continued)

QSPI_RBDR0 **QSPI_ARDB0**	fal1	fbl1	fah1	fbh1	fal0	fbl0	fah0	fbh0
QSPI_RBDR1 **QSPI_ARDB1**	fal3	fbl3	fah3	fbh3	fal2	fbl2	fah2	fbh2
QSPI_SFAR[SFADR] set to 0x000_0001								
QSPI_RBDR0 **QSPI_ARDB0**	fal2	fbl2	fah2	fbh2	fal1	fbl1	fah1	fbh1
QSPI_RBDR1 **QSPI_ARDB1**	fal4	fbl4	fah4	fbh4	fal3	fbl3	fah3	fbh3

Note

For IP Commands the read size can be given in number of bytes. If this number is not a multiple of 4 the last buffer entry is not completely filled with the missing higher numbered bytes at undefined values.

* Appicable only for single io mode.

Table 10-19. Byte Ordering in Parallel Flash Mode - AHB Buffer

AHB Address **(32 Bit Access)**	fal1	fbl1	fah1	fbh1	fal0	fbl0	fah0	fbh0
AHB Address 0x800_0004 **(32 Bit Access)**	fal3	fbl3	fah3	fbh3	fal2	fbl2	fah2	fbh2

Note

For AHB Command read starting from an address not aligned to 32 bit boundaries or AHB access size smaller than 32 bit the requested bytes are given at the appropriate positions according to the AMBA AHB specification.

- **Buffer Entry Ordering for 64 Bit Read Access**

For read access via the AHB interface 64 bit access is possible. Each 64 bit access reads 2 32 bit entries simultaneously. The ordering of these 32 bit entries within the 64 bit word is given in the following table.

Table 10-20. 64 Bit Read Access Buffer Entry Ordering

AHB Read Data Bit Position [63:0]	[63:32]	[31:0]
Buffer Entry #	Odd (1, 3, 5, ...)	Even (0, 2, 4, ...)

10.2.5.3.9 Normal Mode Interrupt and DMA Requests

The QuadSPI module has different flags that can only generate interrupt requests and one flag that can generate interrupt as well as DMA requests. The following table lists the eight conditions. Note that the flags mentioned in the table are related to the Flag Register (QSPI_FR).

Table 10-21. Interrupt and DMA Request Conditions

Condition	Flag(QSPI_FR)	DMA
Data Learn patttern Failure	DLPFF	-
TX Buffer Fill	TBFF	-
TX Buffer Underrun	TBUF	-
Illegal Instruction Error	ILLINE	-
RX Buffer Drain	RBDF	X
RX Buffer Overflow	RBOF	-
AHB Buffer Overflow	ABOF	-
AHB Sequence Error	ABSEF	-
IP Command Usage Error	IUEF	-
IP Command Trigger during AHB Access Error	IPAEF	-
IP Command Trigger could not be executed Error	IPIEF	-
IP Access during AHB Grant Error	IPGEF	-
IP Command related Transaction Finished	TFF	-

Each condition has a flag bit in the Flag Register (QSPI_FR) and a Request Enable bit in the DMA Request Select and Enable Register (QSPI_RSER) . The RX Buffer Drain Flag (RBDF) has separate enable bits for generating IRQ and DMA requests. Note that not all flags have an individual IRQ line. Check the devices Interrupt Vector Table for more details.

- Transmit Buffer Fill Interrupt Request:

  The Transmit Buffer Fill IRQ indicates that the TX Buffer can accept new data. It is asserted if the QSPI_FR[TBFF] flag is asserted and if the corresponding enable bit (QSIP_RSER[TBFIE]) is set. Refer to TX Buffer Operation, for details about the assertion of the QSPI_FR[TBFF] flag.

- Receive Buffer Drain Interrupt or DMA Request:

  The Receive Buffer Drain IRQ derived from the QSPI_FR[RBDF] flag indicates that the RX Buffer of the QuadSPI module has data available from the serial flash device to be read by the host. It remains set as long as the QSPI_RBSR[RXWE] bit is set. The QSPI_RSER[RBDIE] bit enables the related IRQ.

  Aside from the IRQ it is possible to handle RX Buffer drain by DMA. If the QSPI_RSER[RBDDE] bit is set, a DMA request will be triggered when the RX Buffer contains more than QSPI_RBCT[WMRK] valid entries.The application must set the environment appropriately (for example, the DMA controller) for the DMA transfers.

- Buffer Overflow/Underrun Interrupt Request:

  The Buffer Overflow/Underrun IRQ is a combination of the following flags (all located in the QSPI_FR register with the related enable bits in the QSPI_RSER register):

  - TBUF - TX Buffer Underrun, enabled by TBUIE
  - RBOF - RX Buffer Overflow, enabled by RBOIE
  - ABOF - AHB Buffer Overflow, enabled by ABOIE

    The Transmit Buffer Underrun indicates that an underrun condition in the TX Buffer has occurred. It is generated when a write instruction is triggered whilst the Tx Buffer is empty and the QSPI_RSER[TFUFIE] bit is set.

    The Receive Buffer Overflow indicates that an overflow condition in the RX Buffer has occurred. It is generated when the RX Buffer is full, an additional read transfer attempts to write into the RX Buffer and the QSPI_RSER[RBOIE] bit is set.

    The AHB Buffer Overflow indicates that an overflow condition in the AHB Buffer has occurred. It is generated when the AHB Buffer is full, an additional read transfer attempts to write into the AHB Buffer and the QSPI_RSER[ABOIE] bit is set.

    The data from the transfers that generated the individual overflow conditions is ignored.

- Serial Flash Command Error Interrupt Request

  If the IPAEF, IPIEF, IPGEF or IUEF flags in the QSPI_FR are set, and the related interrupt enable bits in the QSPI_RSER are also set, then an interrupt is requested.

- Transaction Finished Interrupt Request

  The IP Command Transaction Finished IRQ indicates the completion of the current IP Command.It is triggered by the QSPI_FR[TFF] flag and is masked by the QSPI_RSER[TFIE] bit.

10.2.5.3.10 TX Buffer Operation

The TX Buffer provides the data used for page programming. For proper operation it is required to provide at least four entry in the TX Buffer prior to starting the execution of the page programming command. The application must ensure that the required number of data bytes is written into the TX Buffer fast enough as long as the command is executed without a TX Buffer overflow or underrun.

The QuadSPI module sets the QSPI_FR[TBFF] flag so long as the TX Buffer is not full and can accept more data.

When the QuadSPI module tries to pull data out of an empty TX Buffer the TX Buffer underrun is signaled by the QSPI_FR[TBUF] flag.The TX buffer underrun flag is also asserted when TX buffer contains less than 128 bits of data and QuadSPI module tries to pull out data from it. The current IP Command leading to the underrun condition is continued until the specified number of bytes has been sent to the serial flash device, in the underrun condition when QuadSPI module tries to pull out data of empty TX buffer, the data transferred is all F's i.e. once the underrun flag is set under this condition, it will return F's until the required number of bytes are not sent. This has been done to ensure that the software need not to erase whole sector after underrun, just reprogramming from failure point will serve the purpose. When this Sequence Command is finished, the QSPI_FR[TBFF] flag is asserted indicating that the Tx Buffer is ready to be written again.

The TX Buffer overflow isn't signaled explicitly, but the TX Buffer fill level can be monitored by the QSPI_TBSR[TRBFL] field.

Refer to TX Buffer Status Register (QuadSPI_TBSR) and Flag Register (QuadSPI_FR) for details about the TX Buffer related registers.

10.2.5.3.11 Address scheme

Earlier serial flash memories supported only 24-bit address space hence restricting the maximum memory size of the serial flash as 16 MB. The new memory specification supports two types of 32-bit addressing mode in addition to legacy 24-bit address mode.

- **Extended Address Mode**

  In this mode, the legacy 24-bit commands are converted to accept 32-bit address commands. The flash memory needs to be configured for 32-bit address mode. Also, while programming the LUT sequence in QuadSPI for 32-bit mode, the ADDR and ADDR_DDR command should be programmed with 8'd32 as the operand value. By default, the QuadSPI is in 24-bit legacy address mode. Each of the memory vendors have a different way of enabling this mode (Refer to the memory specification from memory vendors). For example, the command B7h sent to Macronix flash will enable it for 32-bit address mode.

- **Extended Address register**

  In this mode, the upper 8-bit of the 32-bit address is provided by the Extended address register in the memory itself. The memory provides a specific register which is updated according to the address to be accessed. This effectively converts the legacy 24-bit address command into 32-bit address commands. The memories greater in size than 16 MB, consists of banks of 16 MB. The 8-bit written in the extended address register effectively enables a bank. For example in Spansion memory, when the extended address register is updated with a value of 0x01 with the help of the command 17h, it will open Bank1 of the memory. The consequent 24-bit address commands will lead to Bank1. The extended address register needs to be update with the respective value for access to other banks. This effectively converts the legacy 24-bit address command into 32-bit address commands.

10.2.6 Initialization/Application Information

This section provides the initialization and application information of the QuadSPI module.

10.2.6.1 Power Up and Reset

Note that the serial flash devices connected to the QuadSPI module may require special voltage characteristics of their inputs during power up or reset. It is the responsibility of the application to ensure this.

10.2.6.2 Available Status/Flag Information

This paragraph gives an overview of the different status and flag information available and their interdependencies for different use cases. Related registers are QSPI_SR and QSPI_FR. Refer to the related descriptions how to set up the QuadSPI module appropriately.

10.2.6.2.1 IP Commands

Refer to IP Configuration Register (QuadSPI_IPCR) for additional details not explicitly covered in this paragraph.

- **IP Commands - Normal Operation**

  Writing the QSPI_IPCR[SEQID] field triggers the execution of a new IP Command. Given that this is a legal command the QSPI_SR[IPACC] and the QSPI_SR[BUSY] bits are asserted simultaneously, immediately after the execution is started.

  When the instruction on the serial flash device has been finished these bits are de-asserted and the QSPI_FR[TFF] flag is set.

- **IP Commands - Error Situations**

  Refer to Table 10-22 below.

10.2.6.2.2 AHB Commands

Refer to Section 1, Reading Serial Flash Data into the QuadSPI Module, in Flash Read for additional details not explicitly covered in this paragraph.

- **AHB Commands - Normal Operation**

  Memory mapped read access to a serial flash address not contained in the AHB Buffer, triggers the execution of an AHB Command. Given that this is a legal command the QSPI_SR[AHBACC] and the QSPI_SR[BUSY] bits are asserted simultaneously immediately after the execution is started. When the instruction on the serial flash device has been finished these bits are de-asserted.

- **IP Commands - Error Situations**

  Refer to Table 10-22 below.

10.2.6.2.3 Overview of Error Flags

The following table gives an overview of the different error flags in the QSPI_FR register and additional error-related details.

Table 10-22. Overview of QSPI_FR Error Flags

Error Category	Error Flag in QSPI_FR	Command Execution on Serial Flash Device TFF Behavior (in case of IP commands only)	Description
AHB Error Flag	ABSEF	Flash transaction is aborted	AHB sequence contains • WRITE instruction • WRITE_DDR instruction
AHB Error Flag	ABOF	Flash transaction continues until it finishes	Set when the module tried to push data into the AHB buffer that exceeded the size of the AHB buffer. Only occurs due to wrong programming of the QSPI_BUFxCR[ADATSZ].
Miscellaneous Error Flag	DLPFF[1]	Flash transaction continues until it finishes	Set when DATA_LEARN instruction was encountered in a sequence but no sampling point was found for the data learning pattern.
Miscellaneous Error Flag	ILLINE	Flash transaction aborted	Illegal instruction Error Flag – Set when an illegal instruction is encountered by the controller in any of the sequences.
Command Arbitration Error	IPIEF	TFF not asserted in conjunction with that command	IP Command Error - caused when IP access is currently in progress (IP_ACC set) and • write attempt to QSPI_IPCR register. • write attempt to QSPI_SFAR register. • write attempt to QSPI_RBCT register.
Command Arbitration Error	IPAEF		• AHB Command already running, another IP Command could not be executed. • AHB Command already running, write attempt to QSPI_IPCR[SEQID] field.
Command Arbitration Error	IPGEF		• Exclusive access to the serial flash granted for AHB Commands, write attempt to QSPI_IPCR[SEQID] field.
IP Command Error	IUEF	—	• IP Command Usage Error
Buffer Related Error	RBOF	TFF is asserted on completion	• RX Buffer Overrun

Table continues on the next page...

Table 10-22. Overview of QSPI_FR Error Flags (continued)

Error Category	Error Flag in QSPI_FR	Command Execution on Serial Flash Device TFF Behavior (in case of IP commands only)	Description
Buffer Related Error	TBUF		• TX Buffer Underrun

1. Data learning is not implemented on this chip.

Note that only the buffer related errors are related to a transaction on the external serial flash. All the other errors do not trigger an actual transaction.

10.2.6.2.4 IP Bus and AHB Access Command Collisions

There are two flags related to this topic, the QSPI_FR[IPAEF] and QSPI_FR[IPIEF]. Refer to sub-section "Reading Serial Flash Data into the QuadSPI Module" of Flash Read section, for a description of the flags and Command Arbitration , for details about possible command collisions.

10.2.6.3 Exclusive Access to Serial Flash for AHB Commands

It is possible that several masters need to access the serial flash device connected to the QuadSPI module separately, one master by triggering IP Commands and reading the RX Buffer (via RBDR*n* register) and the other masters by triggering AHB Commands (via ARDB*n* Registers). These two set of buffer (RBDR and ARDB Buffer) points to the same physical buffer.Refer to Figure 10-20 To avoid command collisions resulting in excessive latencies the QuadSPI module implements a request-handshake mechanism between the master triggering AHB Commands and the QuadSPI module allowing this specific master to request exclusive access to the serial flash device for AHB Commands. If this exclusive access is granted the execution of IP Commands is blocked. This resolves command collisions and excessive times where the AHB interface may be blocked.

If this capability is used in the device there is additional status and flag information available related to this mechanism. The QSPI_SR[AHBGNT] bit reflects the module-internal state that the exclusive access mentioned above is granted, any attempt to trigger an IP Command is rejected and results in the assertion of the QSPI_FR[IPGEF] flag. Refer to the descriptions of the related bit and flag for details.

It is within the responsibility of the application to set up the master using this mechanism appropriately, if used incorrectly no IP Commands at all can be triggered.

Two different cases can be distinguished:

10.2.6.3.1 RX Buffer Read via QSPI_ARDB Registers

In this case all masters share the AHB bus for RX Buffer as well as for AHB Buffer read. In this case the access to the AHB interface by the master triggering AHB Commands must be deferred until any pending IP Command has been finished **and** the RX Buffer readout has been finished as well. The QSPI ARDB Buffers access the Rx buffer i.e the data from the Rx Buffer is returned and no data from AHB Buffer is touched. This is the conservative use case, corresponding to the reset value 0 of the QSPI_RBCT[RXBRD] bit.

In this case the QSPI_SR[AHBGNT] bit is asserted not earlier than any running IP Command has been finished (QSPI_SR[IP_ACC] is 0), the RX Buffer has been read out completely (QSPI_RBSR[RDBFL] equal to 0) or no DMA read is pending (QSPI_SR[RXDMA] equal to 0 and Rx Buffer readout is via AHB(QSPI_RBCT[RXBRD]) equal to 1.

10.2.6.3.2 RX Buffer Read via QSPI_RBDR Registers

This is the preferred use case as an access to the AHB buffer (memory mapped flash) does not interfere with any IPS access to read the RBDR buffer. It is not possible that a pending AHB bus access triggered by an AHB Command stalls the AHB bus and blocks the RX Buffer readout since the RX Buffer is read via the IP bus based registers QSPI_RBDR0 to QSPI_RBDR31.

For this case it is recommended to program the QSPI_RBCT[RXBRD] bit to 1. The QSPI_SR[AHBGNT] bit is asserted immediately after any running IP Command has been finished (QSPI_SR[IP_ACC] is 0), the RX Buffer has been read out completely (QSPI_RBSR[RDBFL] equal to 0) or no DMA read is pending (QSPI_SR[RXDMA] equal to 0, allowing the master triggering AHB Commands to trigger AHB Commands as soon as possible without the need to wait for the RX Buffer readout to be finished.

10.2.6.4 Command Arbitration

In case of overlapping commands, the arbitration scheme is described in the following paragraphs under the assumption that the priority mechanism described in Exclusive Access to Serial Flash for AHB Commands is **not** used:

- During the execution of an IP Command, the running IP Command can't be terminated by issuing another IP Command or AHB Command. The QSPI_FR[IPIEF] flag is asserted when the host tries to write into the QSPI_IPCR

register. When the host triggers an AHB Command (refer to sub-section "Reading Serial Flash Data into the QuadSPI Module" of Flash Read section, for details), this command is stalled until the currently running IP Command is finished.

- During the execution of an AHB Command, the running AHB Command can't be terminated by issuing an IP Command. The command is ignored and the QSPI_FR[IPAEF] flag is asserted. Refer to Flag Register (QuadSPI_FR) for the description of these flags.

  When another AHB Command is triggered the address of the memory mapped access is considered. If the requested address is currently read from the serial flash device, the running command is continued. If this is not the case the currently running command is terminated and another AHB Command related to the requested address is executed. Refer to sub-section "Reading Serial Flash Data into the QuadSPI Module" of Flash Read section, for further details.

In case of coinciding commands the IP Command is triggered and the AHB Command is stalled until the IP Command has been finished (QSPI_SR[IP_ACC] has been deasserted).

The IP Commands ignored in case of command collision will not result in the assertion of the QSPI_FR[TFF] flag.

10.2.6.5 Flash Device Selection

Regardless of the SFM Command (IP or AHB) the access mode is selected by specifying the 32 bit address value for the following SFM Command.

For IP Commands the access mode is selected with the address programmed into the QSPI_SFAR register. Refer to Serial Flash Address Register (QuadSPI_SFAR) for details.

For AHB Commands the access mode is determined by the memory mapped address which is accessed Refer to AMBA Bus Register Memory Map for details.

10.2.6.6 DMA Usage

For the complete description of the DMA module refer to the related DMA Controller chapter. In this paragraph only the details specific to the DMA usage related to the QuadSPI module are given.

10.2.6.6.1 DMA Usage in Normal Mode

10.2.6.6.1.1 Bandwidth considerations

Careful consideration of the throughput rate of the entire chain (serial flash -> AHB bus / IP Bus -> DMA controller) involved in the read data process is essential for proper operation. Such analysis must take into account not only the data rate provided by the serial flash but also the data rate of the AHB bus and the performance of the DMA controller in reading data from the RX buffer.

Two figures must match for proper operation, that means that the data rate provided by the serial flash device must not exceed the average RX Buffer readout data rate. Otherwise, the longer this state persists, a RX Buffer overflow will result.

AHB Bus Side (data read):

The total number of bus cycles for each DMA Minor Loop completion is added from the following components:

- Overhead for each minor loop, given by DMA controller: Assume 10 cycles
- Overhead due to clock domain crossing: Assume 2 cycles
- Number of bus clock cycles required for 8 bytes (64 bit read size): Assume 2 cycles (read/write sequence of DMA controller)

Note that the size of the minor loop is determined by the size of the QSPI_RBCT[WMRK] field, therefore the overhead given above distributes among (QSPI_RBCT[WMRK]+1)/2 read accesses of 64 bit each.

The following table gives some examples for typical use cases:

Table 10-23. Access Duration Examples - Bus Clock Side

QSPI_RBCT[WMRK]	Number of Bytes per DMA Loop [1]	Number of Bus Clock cycles for DMA Minor Loop	Time Duration of DMA Minor Loop for 120Mhz Bus clock Frequency
0	4	12+2 = 14	~117ns
1	8	12+2 = 14	~117ns
3	16	12+4 = 16	~133ns
7	32	12+8 = 20	~167ns
11	48	12+12 = 24	~200ns

1. DMA Loop means one Minor Loop Completion which is equivalent to one.

NOTE

The table figure represents ideal scenario, actual performance will depend on how the system is integrated.

Serial Flash Device Side (data read):

The number of serial flash cycles can be determined in the following way:

- Number of serial flash clock cycles required to read 4 bytes, corresponding to one RX Buffer entry (setup of command and address not considered): 2 cycles for Quad DDR mode instructions in Parallel Flash Mode, 4 cycles for Quad (SDR) mode instruction in parallel flash mode or Dual IO DDR mode instruction in parallel flash mode, 8 cycles for Quad Mode (SDR) instructions in Individual Flash Mode etc.
- Overhead due to clock domain crossing : 1 cycle.

The following table lists the number of clock cycles required to read the data from the serial flash corresponding to the different settings of the QSPI_RBCT[WMRK] field:

Table 10-24. Access Duration Examples - Serial Flash side

QSPI_RBCT[WMRK] setting	Num Bytes per DMA Loop [1]	Num SCKFx for 60MHz SCKFx			Time duration of Flash data readout for 60MHz SCKFx (~16.6ns period)		
		IFM [2] Quad	IFM Quad DDR	PFM [3] Quad DDR	IFM Quad	IFM Quad DDR	PFM Quad DDR
0	4	9	5	3	~150ns	~83ns	~50ns
1	8	17	9	5	~282ns	~150ns	~83ns
3	16	33	17	9	~548ns	~282ns	~150ns
7	32	65	33	17	~1079ns	~548ns	~282ns
11	48	97	49	25	~1610ns	~813ns	~415ns

1. DMA Loop means one Minor loop completion which is equivalent to one Major Loop iteration.
2. Individual flash mode.
3. Parallel flash mode.

From the examples given in the two tables above, it can be seen that depending on the relationship between the Bus clock and Serial flash clock frequencies, there are settings possible where the serial flash provides the read data faster than the AHB bus can read out the RX buffer. In the above tables, it is the case of PFM Quad DDR mode with Watermark up to 3 and other cases. In these cases, the RX buffer data keeps accumulating over time and will eventually overflow. To avoid RX Buffer overflow, the data transaction size should be small enough.

A complementary example would be when the watermark is set to be too high. In such a case, the time taken by the DMA to read out the RX buffer entries should be smaller than the time taken by the controller to push in the remaining entries in the buffer.

NOTE

The tables mentioned above are only examples which must be correlated with the DMA in the system.

10.2.6.7 Parallel mode

QuadSPI can access two flashes in parallel. This increases the throughput of the QuadSPI by two times. Both write and read operations are supported in parallel mode. When dual die flashes are accessed in parallel mode, it is mandatory for flash A1 to be of the same size as B1 and A2 to be of the same size as B2. The following figure shows how QuadSPI maps the incoming addresses to the different flashes connected on board.

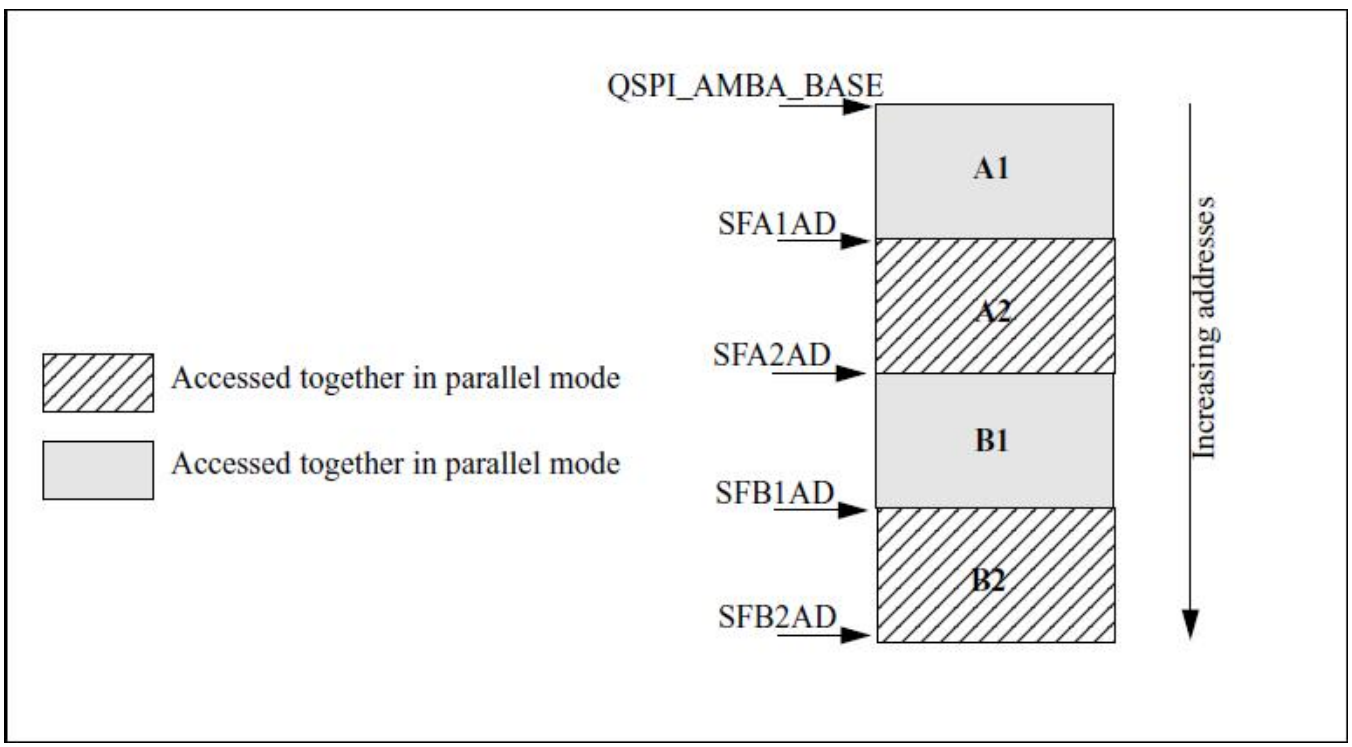


Figure 10-21. Flash addressing

An example programming for parallel mode access is given below (flash sizes are assumed to be 256MB):

- QSPI_AMBA_BASE - 0x10000000
- QSPI_SFA1AD[TPADA1] - 0x20000000
- QSPI_SFA2AD[TPADA2] - 0x30000000
- QSPI_SFB1AD[TPADB1] - 0x40000000
- QSPI_SFB2AD[TPADB2] - 0x50000000

In order to access the first location of A1/B1 pair, the incoming address should be 0x10000000. QSPI_AMBA_BASE is subtracted from this address and the result is divided by two. Therefore, address provided to flash A1 and B1

Flash Address = (Memory mapped address - QSPI_AMBA_BASE)/2

For Memory Mapped address:

- 0x10000000, flash address: 0x0 (Or, the first address of flash A1 and B1)
- 0x10000004, flash address: 0x2
- 0x10000008, flash address: 0x4 etc.

Similarly, in order to access the first location of A2/B2 pair, the incoming address should be 0x30000000.

Flash Address = (Memory mapped address - SFA2AD)/2

For Memory Mapped address:

- 0x30000000, flash address: 0x0 (Or, the first address of flash A2 and B2)
- 0x30000004, flash address: 0x2
- 0x30000008, flash address: 0x4 etc.

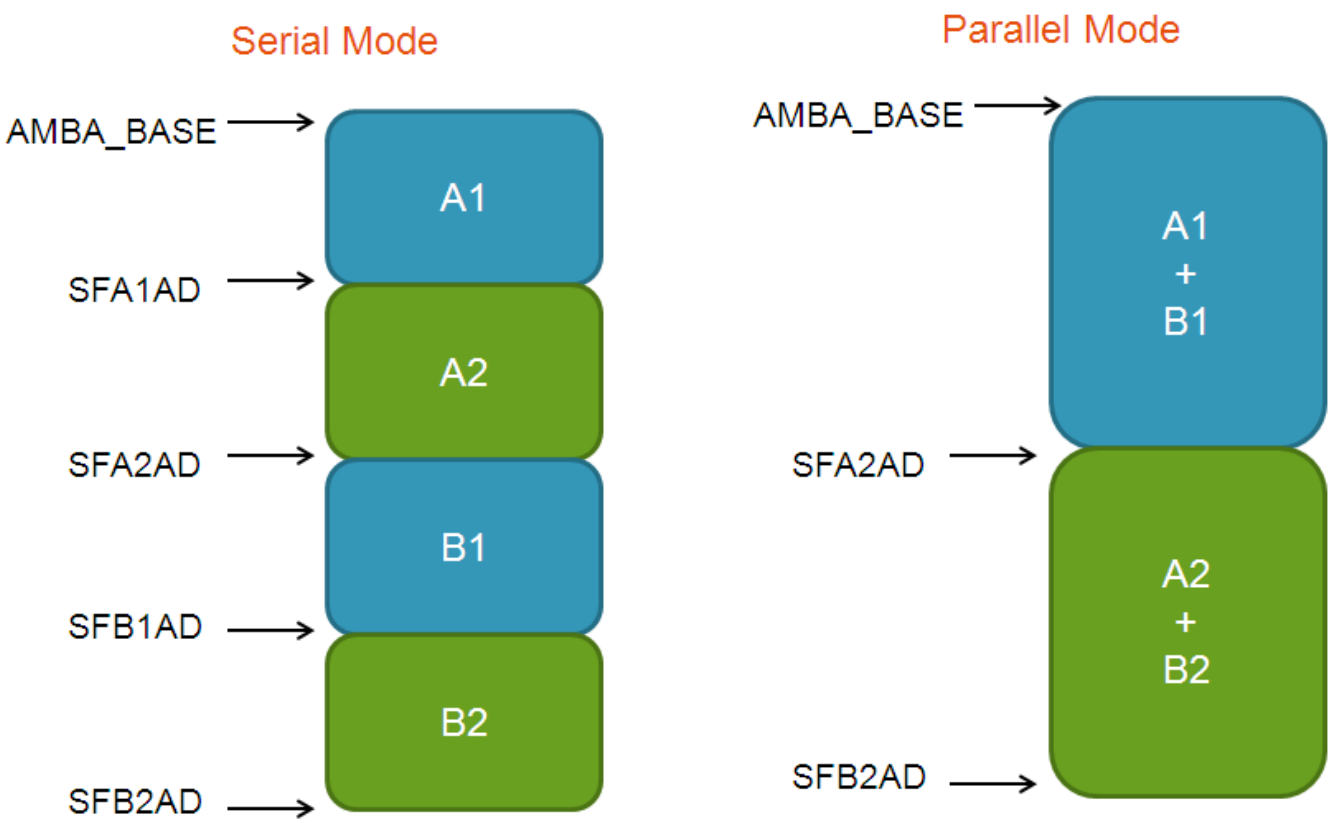


Figure 10-22. Memory map - Serial and Parallel

10.2.7 Byte Ordering - Endianness

QuadSPI provides support for swapping the flash read/write data based on the configuration of the QSPI_MCR[END_CFG]. By default the data is always returned in 64 bit LE format on the AHB bus and 32 bit LE format on the IPS interface when read via the RX buffer and written in 32 bit LE format when written via the TX buffer.

The table(QSPI_MCR[END_CFG]) below shows the complete bit ordering. BE signifies Big Endian which means the high order bits of the associated data vectors are associated with low order address positions. LE signifies Little Endian which means the lower order bits of the associated data vectors are associated with low order address positions.Refer to figure Figure 10-20

Table 10-25. QSPI_MCR[END_CFG]

00	64 bit BE
01	32 bit LE
10	32 bit BE
11	64 bit LE

The tables below (Byte ordering configuration in AHB) and (Byte ordering configuration in IPS) show how this configuration is implemented in QSPI AHB and IPS interfaces respectively. B in the table signifies Byte and the index 1-8 refers to the byte position i.e. 1 refer to bits[7:0], 8 refer to bits[63:56] and so on.

Table 10-26. Byte ordering configuration in AHB

64 bit BE	B1	B2	B3	B4	B5	B6	B7	B8
64 bit LE	B8	B7	B6	B5	B4	B3	B2	B1
32 bit BE	B5	B6	B7	B8	B1	B2	B3	B4
32 bit LE	B4	B3	B2	B1	B8	B7	B6	B5

Table 10-27. Byte ordering configuration in IPS

32BE	B1	B2	B3	B4
32LE	B4	B3	B2	B1

The examples below show the byte ordering in 64 bit BE configuration for AHB Buffer and 32 bit BE for TX/RX Buffer:

10.2.7.1 Programming Flash Data

CPU write instructions to the QSPI_TBDR register like

- Write QSPI_TBDR -> 0x01_02_03_04
- Write QSPI_TBDR -> 0x05_06_07_08

result in the following content of the TX Buffer:

Table 10-28. Example of QuadSPI TX Buffer

TX Buffer Entry	Content
0	32'h01_02_03_04
1	32'h05_06_07_08

Programming the TX Buffer into the external serial flash device results in the following byte order to be sent to the serial flash:

- 01…02…03…04…05…06…07…08

10.2.7.2 Reading Flash Data into the RX Buffer

Reading the content from the same address provides the following sequence of bytes, identical to the write case:

- 01…02…03…04…05…06…07…08

This results in the RX Buffer filled with:

Table 10-29. Resulting RX Buffer Content

RX Buffer Entry	Content
0	32'h01_02_03_04
1	32'h05_06_07_08

10.2.7.2.1 Readout of the RX Buffer via QSPI_RBDRn

The RX Buffer content appears at CPU read access via the Peripheral bus interface in the following order:

- Read QSPI_RBDR0 <- 0x01_02_03_04
- Read QSPI_RBDR1 <- 0x05_06_07_08

10.2.7.2.2 Readout of the RX Buffer via ARDBn

The RX Buffer content appears at read access on the AMBA AHB interface at the QuadSPI module boundary:

- (1a): 32 Bit Access: Read QSPI_ARDB0 <- 0x01_02_03_04
- (2a): 32 Bit Access: Read QSPI_ARDB1 <- 0x05_06_07_08
- (1b/2b): 64 Bit Access: Read QSPI_ARDB0 <- 0x01_02_03_04_05_06_07_08

10.2.7.3 Reading Flash Data into the AHB Buffer

Reading the content from the same address as it was written to provides the following sequence of bytes, identical to the write case:

- 01…02…03…04…05…06…07…08

This results in the AHB Buffer filled with:

Table 10-30. Resulting AHB Buffer Content

AHB Buffer Entry	Content
0	64'h01_02_03_04_05_06_07_08

10.2.7.3.1 Readout of the AHB Buffer via Memory Mapped Read

The AHB Buffer content appears at read access on the AMBA AHB interface at the QuadSPI module boundary:

- (1a): 32 Bit Read Access: <- 0x01_02_03_04
- (2a): 32 Bit Read Access: <- 0x05_06_07_08
- (1/2): 64 Bit Read Access: <- 0x01_02_03_04_05_06_07_08

10.2.8 Serial Flash Devices

Several different vendors make flash devices with a QuadSPI interface. At present there is no set standard for the QuadSPI instruction set. Most common commands currently have the same instruction code for all vendors, however some commands are unique to specific vendors. Some example sequences are provided below.

10.2.8.1 Example Sequences

This section provides the example sequences of the QuadSPI module.

Table 10-31. Exit 4 x I/O Read Enhance Performance Mode (XIP) (Macronix) and Read Status

INSTR	PAD	OPERAND	COMMENT
CMD	0x0	0xEB	4xIO Read Command
ADDR	0x2	0x18	24 Bit address to be send on 4 pads
MODE	0x2	0x00	2 mode cycles (exit XIP)
DUMMY	0x0	0x04	4 dummy cycles
READ	0x2	0x08	Read 64 bits
CMD	0x0	0x05	Read Status register
READ	0x0	0x01	Status register data
STOP	0x0	0x00	STOP, Instruction over

10.2.8.1.1 Fast Read Sequence (Macronix/Numonyx/Spansion/Winbond)

The following table shows the fast read sequence for Macronix/Numonyx/Spansion/ Winbond flashes.

Table 10-32. Fast Read sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0x0B	Fast Read command = 0x0B
ADDR	0x0	0x18	24 Addr bits to be sent on one pad
DUMMY	0x0	0x08	8 Dummy cycles
READ	0x0	0x04	Read 32 Bits on one pad
JMP_ON_CS	0x0	0x00	Jump to instruction 0 (CMD)

10.2.8.1.2 Fast Dual I/O DT Read Sequence (Macronix)

The following table shows the Fast Dual I/O DT read sequence for Macronix flashes.

Table 10-33. Fast Dual I/O DT Read sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0xBD	Fast Dual I/O DT read command = 0xBD
ADDR_DDR	0x1	0x18	24 Addr bits to be sent on 2 pads in DDR mode
MODE4_DDR	0x1	0x00	P2=P0 or P3=P1 is necessary. Refer to Macronix datasheet for details. One clock cycle for mode.
DUMMY	0x0	0x06	6 Dummy cycles
READ_DDR	0x1	0x04	Read 32 Bits on 2 pads in DDR mode
JMP_ON_CS	0x0	0x00	Jump to instruction 0 (CMD)

10.2.8.1.3 Fast Read Quad Output (Winbond)

The following table shows the Fast read quad output sequence for Winbond memories

Table 10-34. Fast Read Quad output sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0x6B	Fast read quad output command = 0x6B
ADDR	0x0	0x18	24 Addr bits to be sent on 1 pad
DUMMY	0x0	0x08	8 Dummy cycles
READ	0x2	0x04	Read 32 Bits on 4 pads
JMP_ON_CS	0x0	0x00	Jump to instruction 0 (CMD)

10.2.8.1.4 4 x I/O Read Enhance Performance Mode (XIP) (Macronix)

The following table shows the 4 x I/O Read Enhance Performance Mode for Macronix flashes. The enhanced performance mode is also known as XIP mode.

Table 10-35. Fast Read Quad output sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0xEB	4xI/O Read command = 0xEB
ADDR	0x2	0x18	24 Addr bits to be sent on 4 pads
MODE	0x2	0xA5	2 mode cycles
DUMMY	0x0	0x04	4 Dummy cycles
READ	0x2	0x04	Read 32 Bits on 4 pads
JMP_ON_CS	0x0	0x01	Jump to instruction 1 (ADDR)

When in XIP mode the software should ensure that all the flashes connected to the controller are in XIP mode. As a part of initializing the controller, all the flashes may be enabled with XIP by carrying out dummy reads.

10.2.8.1.5 Dual Command Page Program (Numonyx)

The following table shows the Dual command page program sequence for Numonyx flashes.

Table 10-36. Dual Command Page Program sequence

Instruction	Pad	Operand	Comment
CMD	0x1	0x02	Dual command page program = 0x02 on 2 pads
ADDR	0x1	0x18	24 Addr bits to be sent on 2 pads
WRITE	0x1	0x20	Write 32 Bytes on 2 pads
STOP	0x0	0x00	STOP, Instruction over

10.2.8.1.6 Sector Erase (Macronix/Spansion/Numonyx)

The following table shows the Sector erase sequence for Macronix/Spansion/Numonyx flashes

Table 10-37. Sector Erase sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0xD8	Sector erase command = 0xD8
ADDR	0x0	0x18	24 Addr bits to be sent on 1 pad
STOP	0x0	0x00	STOP, Instruction over

10.2.8.1.7 Read Status Register (Macronix/Spansion/Numonyx/Winbond)

The following table shows the Read status register sequence for Macronix/Spansion/ Numonyx/Winbond flashes.

Table 10-38. Read Status Register Sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0x05	Read status register command = 0x05
READ	0x0	0x01	Read status register data
STOP	0x0	0x00	STOP, Instruction over

10.2.8.2 Dual Die Flashes

Certain serial flash vendors provide dual-die packages which are essentially two devices (dies) stacked within the same package to increase the memory capacity of a single package. These two devices within a package share the same data and clock pins, but have individual Chip Selects. QuadSPI controller provides support for two dual-die packages to be connected simultaneously. The figure below shows the two dual-die packages and the naming conventions used in this document. For simplicity, the data pins are shown to be unidirectional.

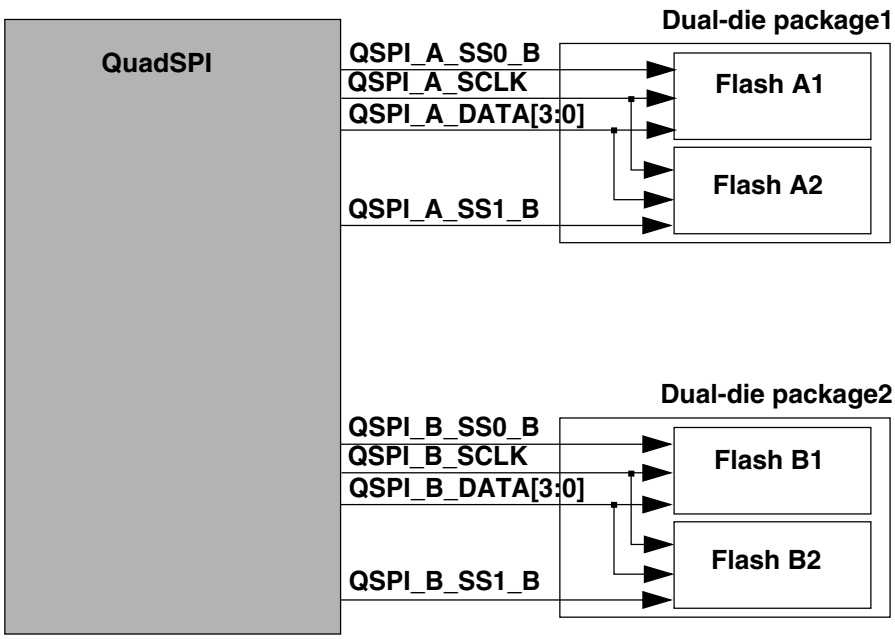


Figure 10-23. Dual-die support

Since the two devices within one package share the same i/o pads, they cannot function in parallel mode. Software should ensure that when QuadSPI is configured in parallel mode the two selected flash devices are from different dual-die packages.

10.2.8.3 Boot initialization sequence

The following are the recommended sequence of steps for booting from QuadSPI:

- System out of reset and flash available (300us)
- Clocks still at very low frequency. Clock tree configured, I/O pins configured. First request sent to QuadSPI for address 0x0 of flash.

- The reset command sequence in QuadSPI has 0x03 (basic read command) which is applicable to all flashes at < 50MHz serial flash clock
- The first few bytes of data is read from the flash which contains the following information:
  - The total sizes of all the flashes connected on board
  - Whether DDR mode supported
  - Frequency of DDR operation
  - Continuous mode entry sequence
  - 24bit or 32bit addressing (assuming 24bit for first accesses)
- All the serial flashes are configured
  - Quad Mode enabled
  - Dummy reads to enter into XIP
- QuadSPI is configured
  - Parallel enable set
  - LUT configured for highest performance reads
  - DDR mode enabled (if applicable)
  - Buffers configured
- Serial flash clock frequency increased.
- Boot reads happen in parallel, DDR enabled, quad output mode @66MHz.

10.2.9 Sampling of Serial Flash Input Data

10.2.9.1 Internal Sampling of Serial Flash Input Data

Depending from the actual implementation there is a delay between the internal clocking in the QuadSPI module and the external serial flash device.Refer to the following figure for an overview of this scheme.

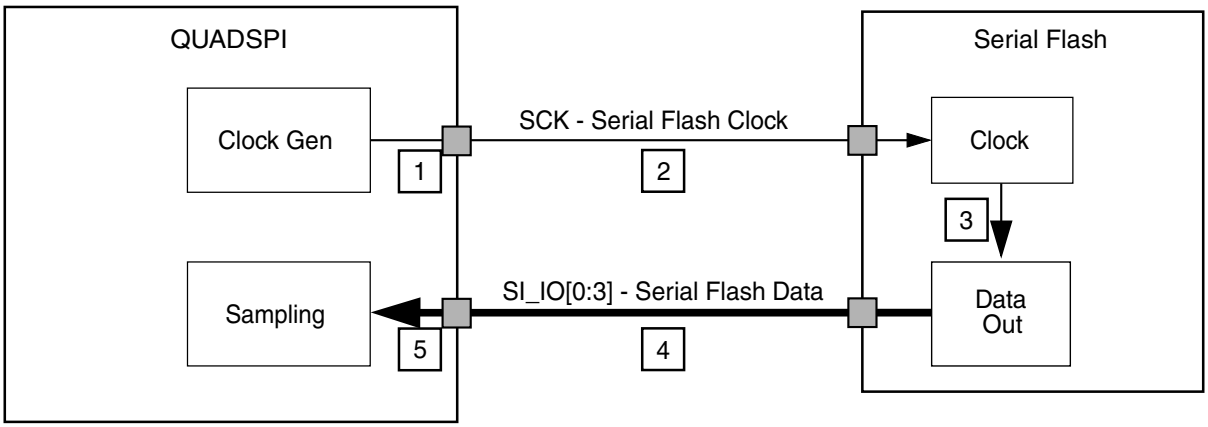


Figure 10-24. Serial Flash Sampling Clock Overview

Note

The arrival of the serial flash data in the sampling stage of the QuadSPI module are given in the following figure. Note that the amount of the total delay $t_{Del,total}$ is very specific to the characteristics of the actual implementation.

Note also that the serial flash device clock SCK is inverted with respect to the QuadSPI internal reference clock.

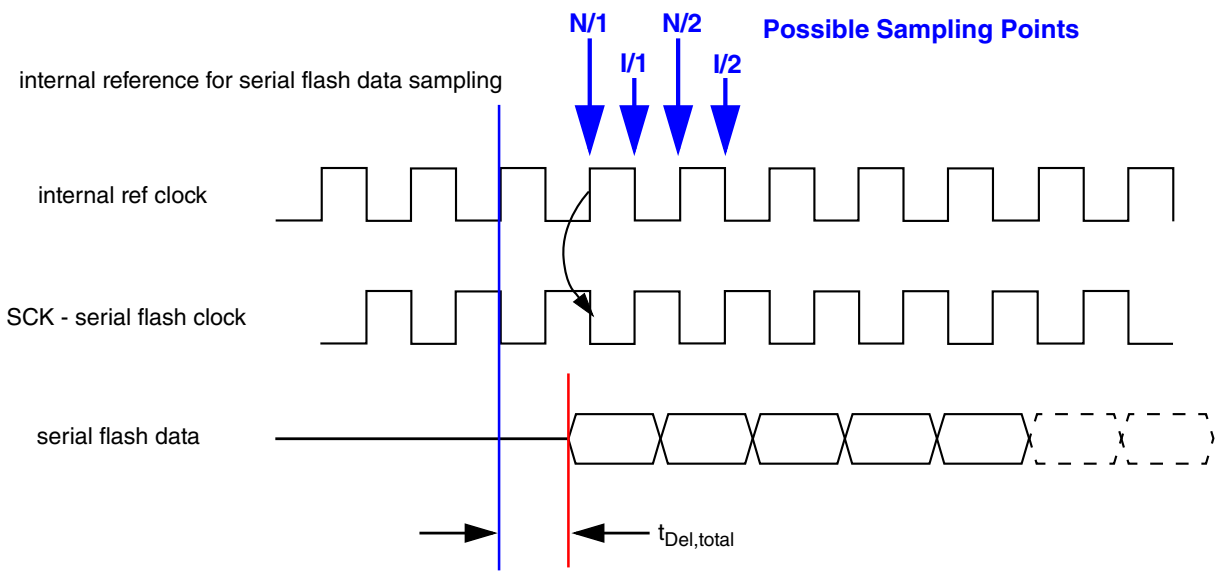


Figure 10-25. Serial Flash Sampling Clock Timing

The rising edge of the internal reference clock is taken as timing reference for the data output of the serial flash. After a time of $t_{Del,total}$ the data arrive at the internal sampling stage of the QuadSPI module.

According to the Serial Flash Sampling Clock Overview figure, the following parts of the delay chain contribute to $t_{Del,total}$:

1. Output delay of the serial flash clock output of the device containing the QuadSPI module
2. Wire delay of application/PCB from the device containing the QuadSPI module to the external serial flash device
3. Clock to data out delay of the external serial flash device, including input and output delays
4. Wire delay of application/PCB from the external serial flash device to the device containing the QuadSPI module
5. Input delay belonging to the data in input

The possible points in time for the sampling of the incoming data are denoted as N/1, I/1, N/2 and I/2 above. The sampling point relevant for the internal sampling is configured in the QSPI_SMPR register, refer to Sampling Register (QuadSPI_SMPR) for details. Note that the falling edges of the reference clock are not actually used, instead the inverted clock is used for sampling at these positions. The following table gives an overview of the available configurations for the commands running at regular (full) speed:

Table 10-39. Sampling Configuration

Sampling Point	Description	Delay [FSDLY] [HSDLY]	Phase [FSPHS] [HSPHS]	QSPI_SMPR for Full Speed Setting[1]
N/1	sampling with non-inverted clock, 1 sample delay	0	0	0x0000000x
I/1	sampling with inverted clock, 1 sample delay	0	1	0x0000002x
N/2	sampling with non-inverted clock, 2 samples delay	1	0	0x0000004x
I/2	sampling with inverted clock, 2 samples delay	1	1	0x0000006x

1. 'x' is not considered here

Depending from the actual delay and the serial flash clock frequency the appropriate sampling point can be chosen. The following remarks should be considered when selecting the appropriate setting:

- Theoretically there should be 2 settings possible to capture the correct data since the serial flash output is valid for 1 clock cycle, disregarding rise and fall times and timing uncertainties.
- Depending from the timing uncertainties it may turn out in actual applications that only one possible sample positions remains. This is subject to careful consideration depending from the actual implementation.

- The delay $t_{Del,total}$ is an absolute size to shift the point in time when the serial flash date get valid at the QuadSPI input.
- For decreasing frequency of the serial flash clock the distance between the edges increases. So for large differences in the frequency the required setting may change.
- For commands running at half of the regular serial flash clock (QSPI_SMPR[HSENA] bit set) the sampling point must be figured separately to allow for the compensation of the absolute shift in time with respect to the sample-relative setting in the QSPI_SPMPR register.

10.2.9.2 DDR Mode

The increasing requirement of improved throughput has introduced the double data rate (DDR) mode. In DDR mode, the data is transferred on both the rising and falling edges of the serial flash clock. The DDR serial flashes sample as well as drive the data on both rising and falling edges of serial flash clock.

10.2.10 Serial Flash Data Input Timing

There are sampling modes for input flash data:

- Internal sampling - Input serial flash data is captured by internal serial clock. In SDR mode, serial data is sampled by serial clock 1x (ser_clk_1x) rise edge. In DDR mode, serial data is sampled by serial clock 4x (ser_clk_4x) rise edge.
- Loopback DQS sampling - Soc will output serial data strobe with internal serial clock. This serial data strobe would be loopback from pad and used to sample input serial data. In SDR mode, serial data is sampled by loopback DQS rise edge. In DDR mode, serial data is sampled by loopback DQS both edge

NOTE

DQS pad need to be set to force input for loopback.

- Flash DQS sampling - Some serial Flash device provide the data strobe output together with serial data. This strobe signal is used to sample input serial data directly. In SDR mode, serial data is sampled by Flash DQS rise edge. In DDR mode, serial data is sampled by Flash DQS both edge

Following table shows selection for sampling mode:

Sampling mode	DQS_EN	DQS_LOOPBACK_EN
Internal sampling	0	doesn't matter

Table continues on the next page...

Sampling mode	DQS_EN	DQS_LOOPBACK_EN
Loopback DQS sampling	1	1
Flash DQS sampling	1	0

Serial flash data and clock path for input timing is shown in the following figure.

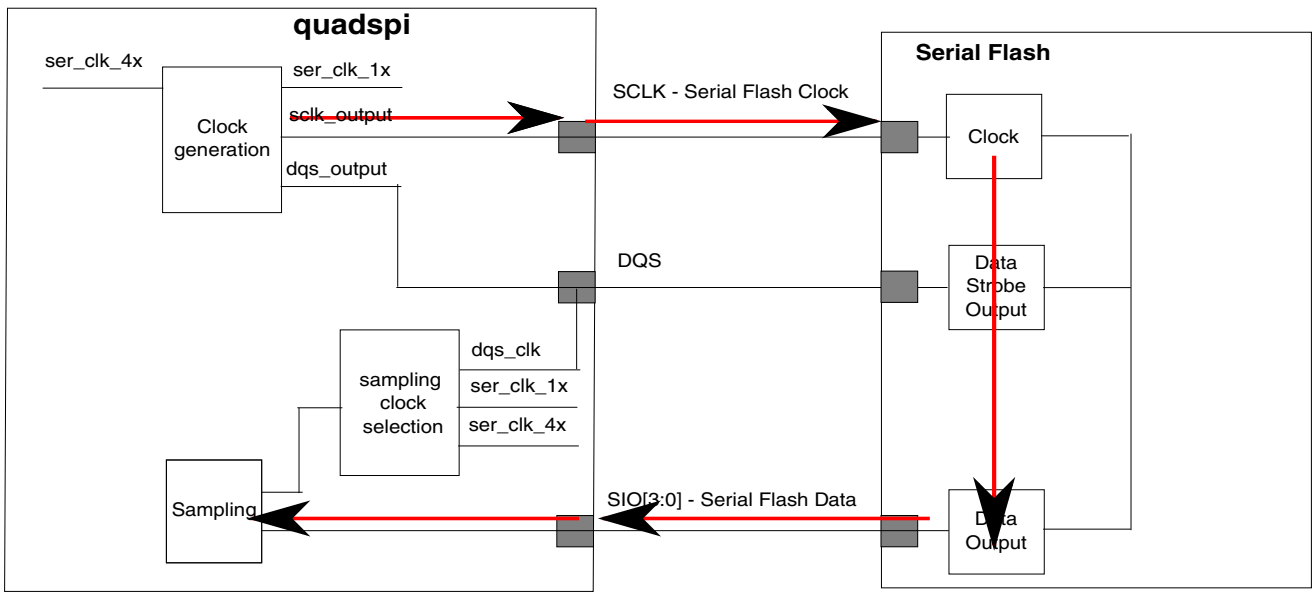


Figure 10-26. Serial flash data and clock path

NOTE

The red line is for data input path, the blue line is for DQS loopback path, and the purple line is for DQS input path from Serial Flash.

Total delay (Ttotal_delay_data) for serial flash data input is the sum of following delay:

1. Output delay of serial flash clock from internal serial clock to SCLK pad inside SOC
2. Wire delay of serial flash clock (SCLK) from SOC to external Serial Flash Device
3. Clock to Output Valid time of external Serial Flash Device
4. Wire delay of serial flash data (SIO) from external serial Flash Device to SOC
5. Input delay of serial flash data from SIO pad to internal sample register

Total delay (Ttotal_delay_loopback_dqs) for loopback DQS clock is the sum of following delay:

1. Output delay of DQS clock from internal serial clock to DQS pad inside SOC
2. Input delay of DQS clock from DQS pad to internal sample register

Total delay (Ttotal_delay_flash_dqs) for Flash DQS clock is the sum of following delay:

1. Output delay of serial flash clock from internal serial clock to SCLK pad inside SOC
2. Wire delay of serial flash clock (SCLK) from SOC to external Serial Flash Device

3. Clock to DQS Output time of external Serial Flash Device
4. Wire delay of serial flash data strobe (DQS) from external serial Flash Device to SOC
5. Input delay of DQS clock from DQS pad to internal sample register

10.2.10.1 Input timing in SDR mode with internal sampling

Input Timing diagram in SDR mode with internal sampling is show in the figure below.

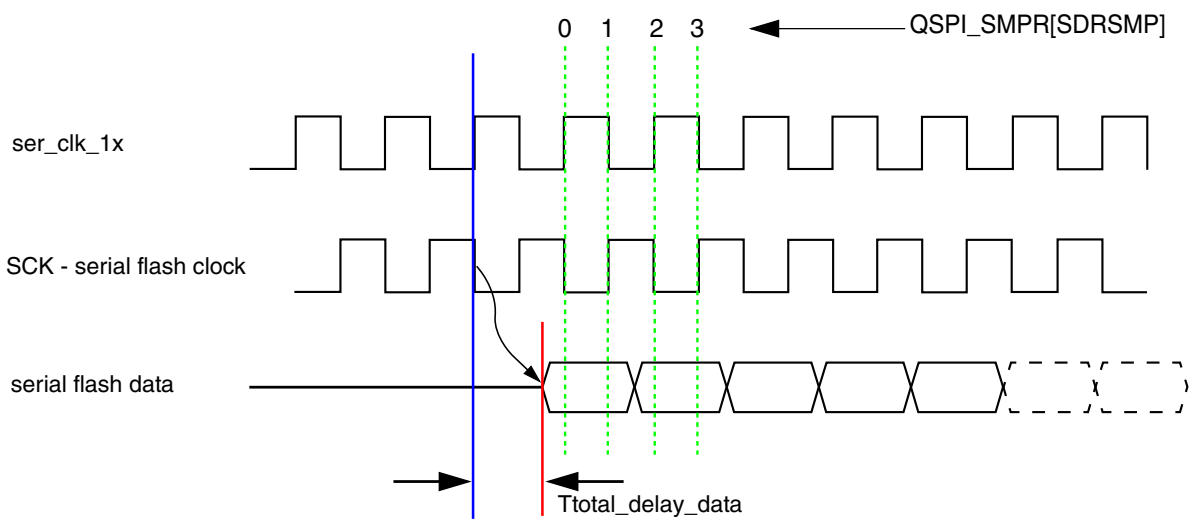


Figure 10-27. Internal sample SDR

There are four sample points for this sampling mode, which is determined by register field QSPI_SMPR.SDRSMP.

Sampling point need to be select correctly to meet both Setup and Hold timing for internal sample registers:

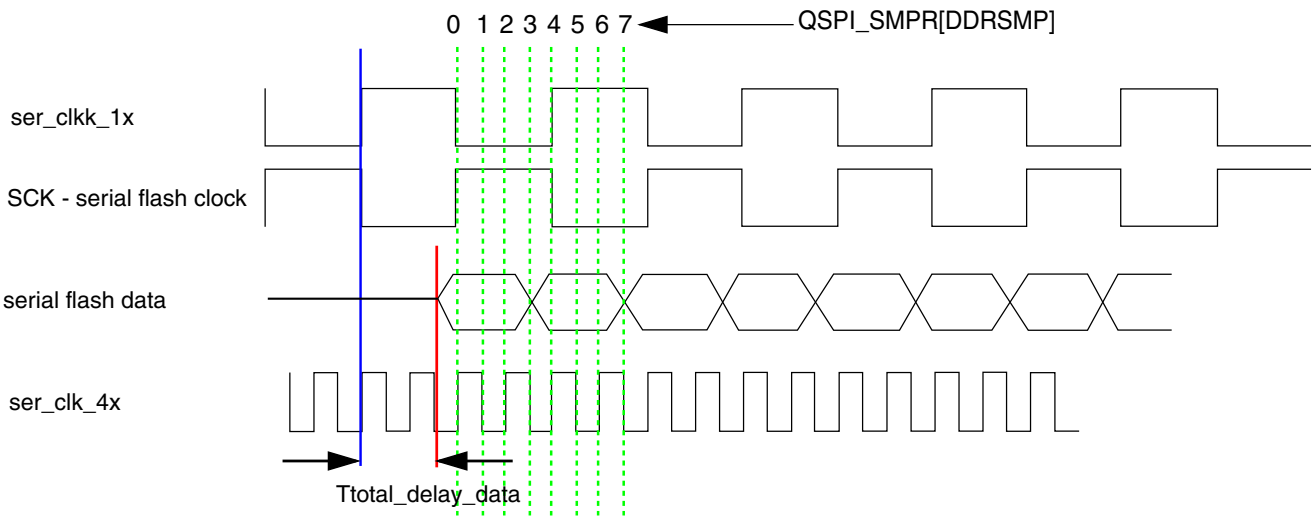
- For sample point N, Setup requirement is: Tcycle*(N+2)/2>Ttotal_delay_data,max
- For sample point N, Hold requirement is: Ttotal_delay_data,min>Tcycle*N/2

NOTE

Tcycle is the cycle of ser_clk_1x. Ttotal_delay_data,max is maximum delay of serial data input path. Ttotal_delay_data,min is the minimum delay of serial data input path. N=0,1,2,3

10.2.10.2 Input timing in DDR mode with internal sampling

Input Timing diagram in SDR mode with internal sampling is show in the following figure.



There are 8 sample points for this sampling mode, which is determined by register field DDRSMP.

Sampling point need to be select correctly to meet both Setup and Hold timing for internal sample registers:

- For sample point N, Setup requirement is: Tcycle * (N+2)/8 >Ttotal_delay_data,max
- For sample point N, Hold requirement is: Ttotal_delay_data,min> Tcycle * N/8

10.2.10.3 Input timing in SDR mode with loopback DQS sampling

Input Timing diagram in SDR mode with internal sampling is show in the following figure.

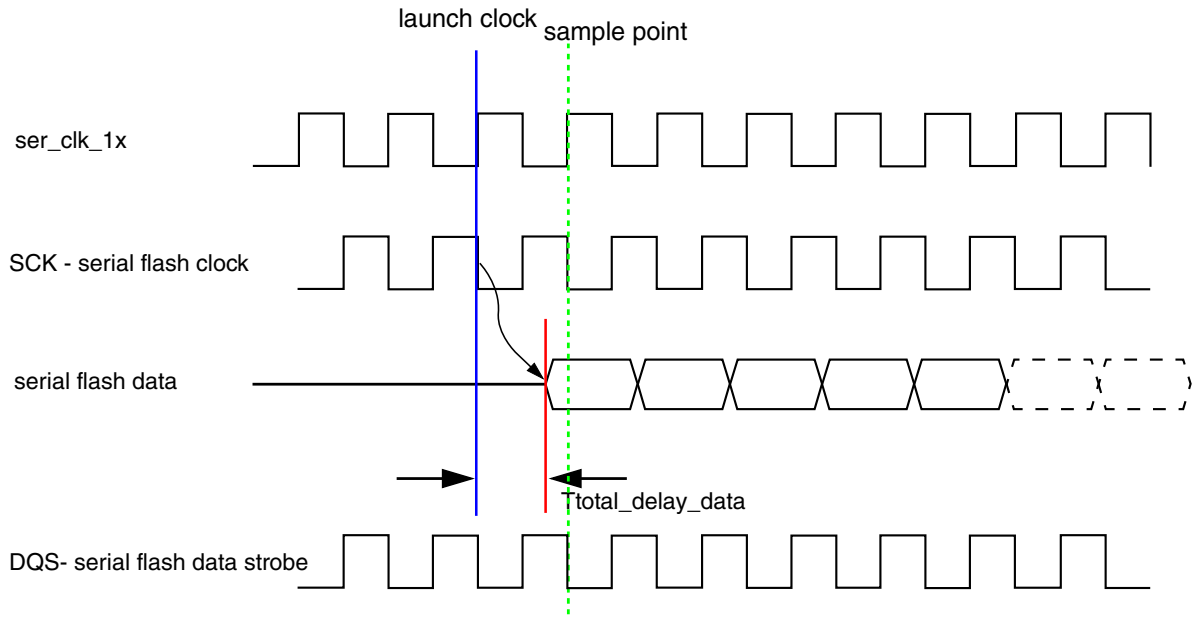


Figure 10-28. Input timing in SDR mode with loopback DQS sampling

In SDR mode and loopback sampling mode, DQS_PHASE_EN should be set 1.

For this sample point, the Setup requirement is: Tcycle >max(Ttotal_delay_data-Ttotal_delay_loopback_dqs). The Hold requirement is: min(Ttotal_delay_data-Ttotal_delay_loopback_dqs)> 0.

10.2.10.4 Input timing in DDR mode with loopback DQS sampling

Input Timing diagram in DDR mode with internal sampling is show in the following figure.

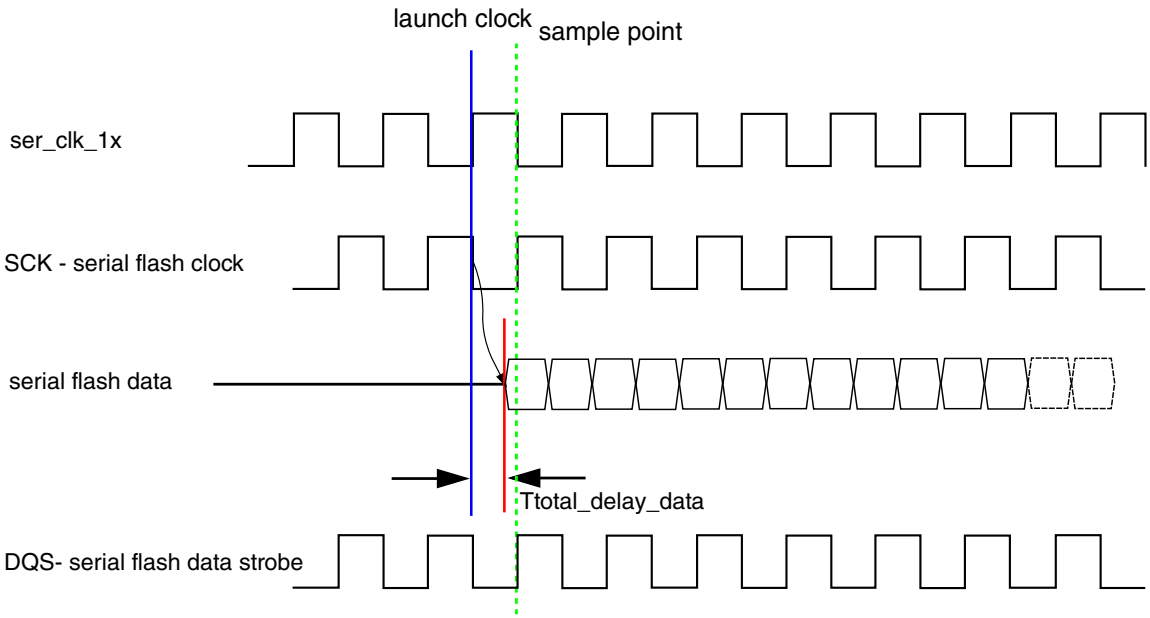


Figure 10-29. Input timing in DDR mode with loopback DQS sampling

In DDR mode and loopback sampling mode, DQS_PHASE_EN should be set 0.

For this sample point, the Setup requirement is: Tcycle >max(Ttotal_delay_data-Ttotal_delay_loopback_dqs). The Hold requirement is: min(Ttotal_delay_data-Ttotal_delay_loopback_dqs)> 0.

10.2.10.5 Input timing in SDR mode with flash DQS sampling

Input Timing diagram in SDR mode with internal sampling is show in the following figure.

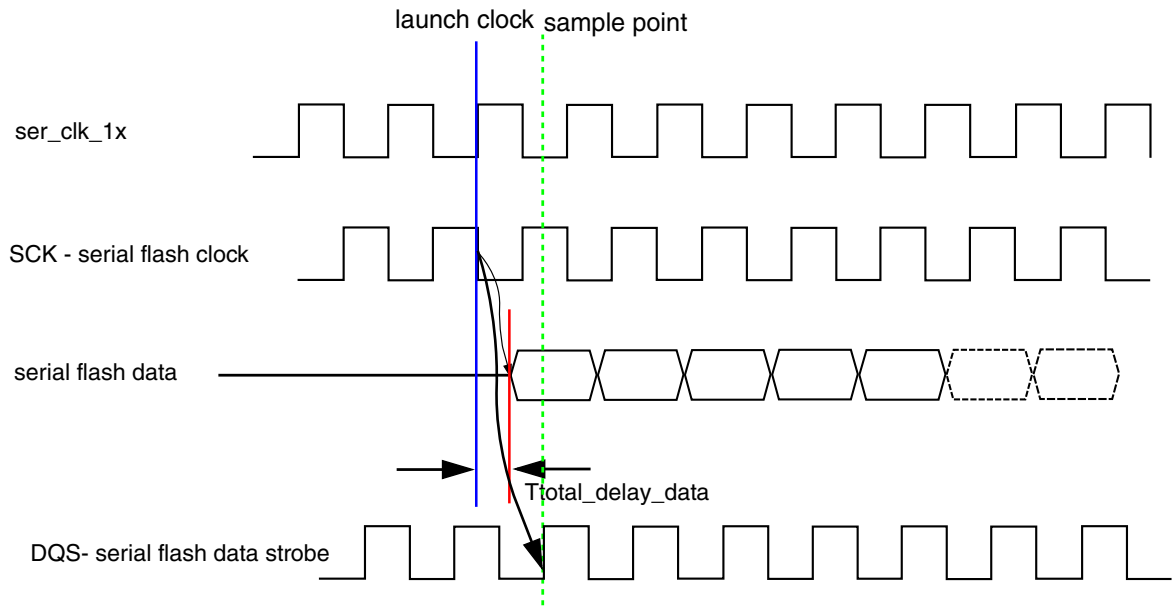


Figure 10-30. Input timing in SDR mode with flash DQS sampling

In this sampling mode, there will be a setup/hold requirement on Flash serial Data and Flash serial Data Strobe. This value will be specified in the data sheet.

10.2.10.6 Input timing in DDR mode with flash DQS sampling

Input Timing diagram in DDR mode with internal sampling is show in the following figure.

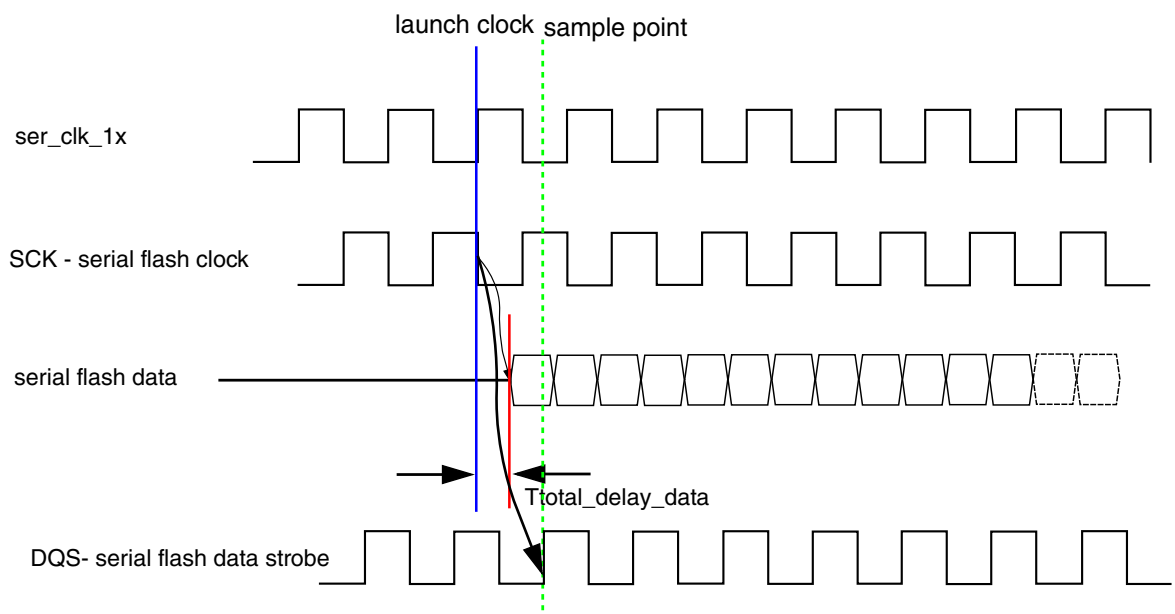


Figure 10-31. Input timing in DDR mode with flash DQS sampling

In this sampling mode, there will be a setup/hold requirement on Flash serial Data and Flash serial Data Strobe. This value will be specified in the data sheet.

10.2.10.7 Data Strobe Signal functionality

Some external serial flashes provide the data strobe (DQS/RDS) output which is fed directly to the QuadSPI module. The strobe (DQS/RDS) signal needs to be delayed to have the edges aligned to the data valid period. QuadSPI internally samples the incoming data at posedge of the strobe signal for SDR and on both the edges of the strobe signal for DDR. Refer to the figure for more detail.

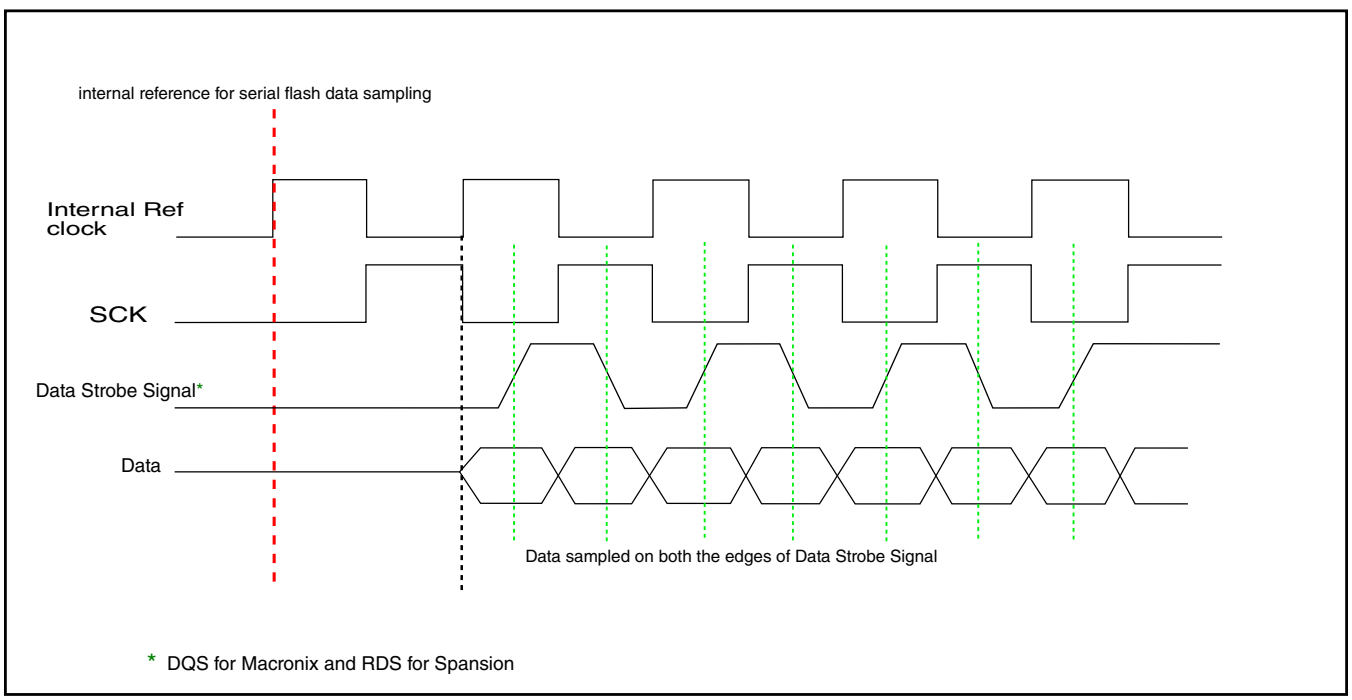


Figure 10-32. Data strobe signal functionality

10.2.11 Output timing in SDR mode

Output timing diagram in SDR mode is show in the following figure.

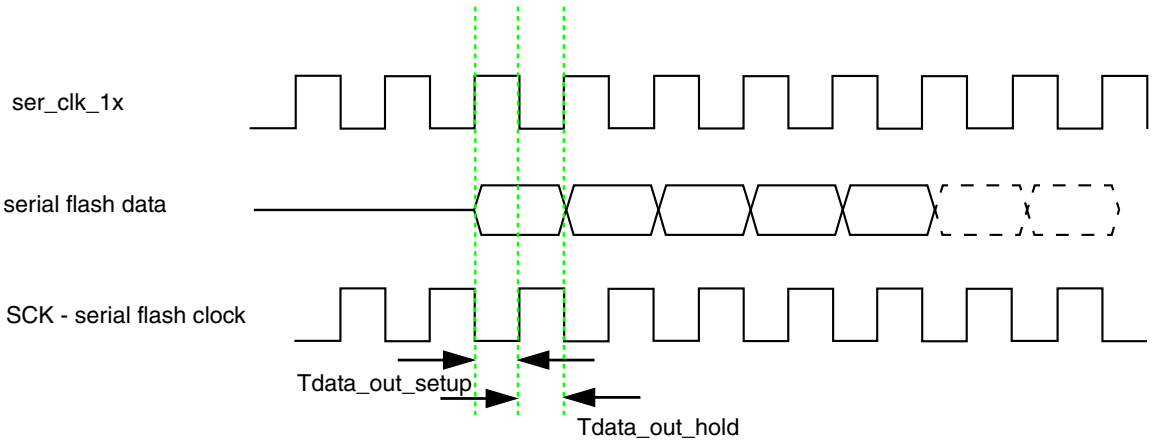


Figure 10-33. Output timing in SDR mode

In SDR mode, quadspi output serial data with internal serial clock rise edge 1x (ser_clk_1x). Flash Device has requirement on Data and Clock setup and hold timing.

10.2.12 Output timing in DDR mode

Output Timing diagram in DDR mode is show in the following figure.

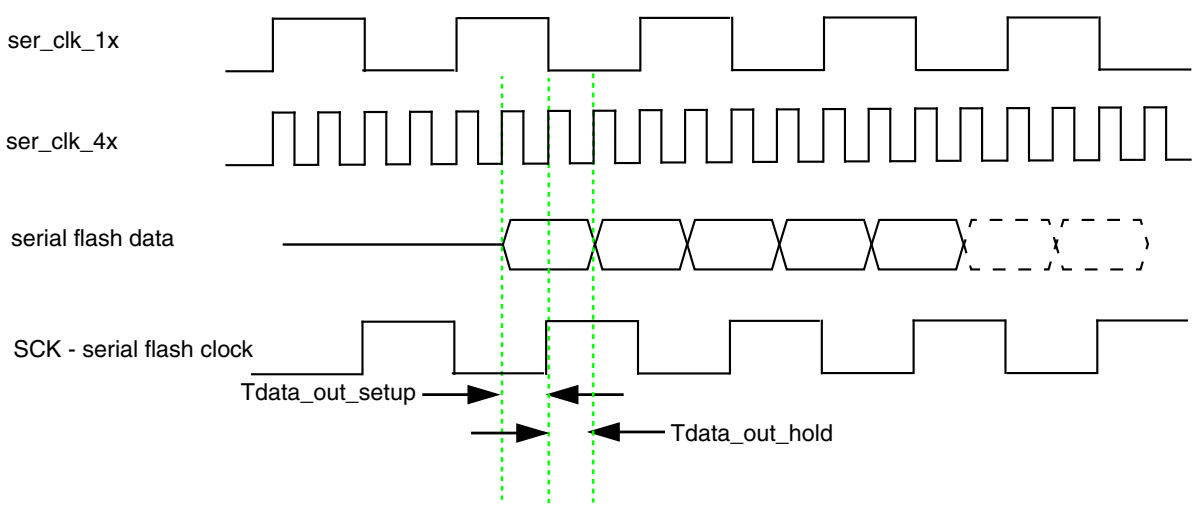


Figure 10-34. Output timing in DDR mode

In DDR mode, quadspi output serial data with internal serial clock both edge 1x (ser_clk_1x) and then delay one ser_clk_4x cycle for hold timing.

NOTE

TX_DDR_DELAY_EN should be set to 1 for DDR mode.

10.2.13 AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31)

10.2.13.1 AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31)

NOTE

See the System Memory map in this document for the base address of the QSPI AHB RX Data Buffer.

memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	AHB RX Data Buffer register (ARDB0)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
4	AHB RX Data Buffer register (ARDB1)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
8	AHB RX Data Buffer register (ARDB2)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
C	AHB RX Data Buffer register (ARDB3)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
10	AHB RX Data Buffer register (ARDB4)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
14	AHB RX Data Buffer register (ARDB5)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
18	AHB RX Data Buffer register (ARDB6)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
1C	AHB RX Data Buffer register (ARDB7)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
20	AHB RX Data Buffer register (ARDB8)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
24	AHB RX Data Buffer register (ARDB9)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
28	AHB RX Data Buffer register (ARDB10)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
2C	AHB RX Data Buffer register (ARDB11)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
30	AHB RX Data Buffer register (ARDB12)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
34	AHB RX Data Buffer register (ARDB13)	32	R/W	0000_0000h	10.2.13.1.1/ 2659

Table continues on the next page...

memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
38	AHB RX Data Buffer register (ARDB14)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
3C	AHB RX Data Buffer register (ARDB15)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
40	AHB RX Data Buffer register (ARDB16)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
44	AHB RX Data Buffer register (ARDB17)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
48	AHB RX Data Buffer register (ARDB18)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
4C	AHB RX Data Buffer register (ARDB19)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
50	AHB RX Data Buffer register (ARDB20)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
54	AHB RX Data Buffer register (ARDB21)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
58	AHB RX Data Buffer register (ARDB22)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
5C	AHB RX Data Buffer register (ARDB23)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
60	AHB RX Data Buffer register (ARDB24)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
64	AHB RX Data Buffer register (ARDB25)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
68	AHB RX Data Buffer register (ARDB26)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
6C	AHB RX Data Buffer register (ARDB27)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
70	AHB RX Data Buffer register (ARDB28)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
74	AHB RX Data Buffer register (ARDB29)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
78	AHB RX Data Buffer register (ARDB30)	32	R/W	0000_0000h	10.2.13.1.1/ 2659
7C	AHB RX Data Buffer register (ARDB31)	32	R/W	0000_0000h	10.2.13.1.1/ 2659

10.2.13.1.1 AHB RX Data Buffer register (ARDB*n*)

The AHB RX Data Buffer register 0 to 31 can be used to read the buffer content of the RX Buffer from successive addresses. QSPI_ARDB0 corresponds to the RX Buffer register entry corresponding to the current value of the read pointer with increasing order.

The increment of the read pointer depends from the access scheme (DMA or flag-driven). Refer to "Data Transfer from the QuadSPI Module Internal Buffers" section in Flash Read section, RX Buffer, data read via register interface and AHB read, for the description of successive accesses to the RX Buffer content. Refer also to Byte Ordering of Serial Flash Read Data for the byte ordering scheme.

Valid address range accessible in the QSPI_ARDBn range depends from the number of RX Buffer entries implemented and from the number of valid buffer entries available in the RX Buffer.

- Example 1, RX Buffer filled completely with 32 words: In this case the address range for valid read access extends from QSPI_ARDB0 to QSPI_ARDB31.
- Example 2, RX Buffer filled with 5 valid words, RX Buffer fill level QSPI_RBSR[RDBFL] is 5. In this case an access to QSPI_ARDB4 provides the last valid entry.

Address: 0h base + 0h offset + (4d × i), where i=0d to 31d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ARXD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ARDB*n* field descriptions

Field	Description
ARXD	ARDB provided RX Buffer Data. Byte order (endianness) is identical to the RX Buffer Data Registers.

10.2.14 Peripheral Bus Register Descriptions

This section provides the peripheral bus register information of the QuadSPI module.

This section provides the memory map and register definitions of the QuadSPI module.

QuadSPI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BB_0000	Module Configuration Register (QuadSPI1_MCR)	32	R/W	000F_4000h	10.2.14.1/ 2673
30BB_0008	IP Configuration Register (QuadSPI1_IPCR)	32	R/W	0000_0000h	10.2.14.2/ 2675

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BB_000C	Flash Configuration Register (QuadSPI1_FLSHCR)	32	R/W	0000_0303h	10.2.14.3/ 2676
30BB_0010	Buffer0 Configuration Register (QuadSPI1_BUF0CR)	32	R/W	0000_0000h	10.2.14.4/ 2677
30BB_0014	Buffer1 Configuration Register (QuadSPI1_BUF1CR)	32	R/W	0000_0000h	10.2.14.5/ 2678
30BB_0018	Buffer2 Configuration Register (QuadSPI1_BUF2CR)	32	R/W	0000_0000h	10.2.14.6/ 2679
30BB_001C	Buffer3 Configuration Register (QuadSPI1_BUF3CR)	32	R/W	See section	10.2.14.7/ 2680
30BB_0020	Buffer Generic Configuration Register (QuadSPI1_BFGENCR)	32	R/W	0000_0000h	10.2.14.8/ 2681
30BB_0030	Buffer0 Top Index Register (QuadSPI1_BUF0IND)	32	R/W	0000_0000h	10.2.14.9/ 2682
30BB_0034	Buffer1 Top Index Register (QuadSPI1_BUF1IND)	32	R/W	0000_0000h	10.2.14.10/ 2682
30BB_0038	Buffer2 Top Index Register (QuadSPI1_BUF2IND)	32	R/W	0000_0000h	10.2.14.11/ 2683
30BB_0100	Serial Flash Address Register (QuadSPI1_SFAR)	32	R/W	0000_0000h	10.2.14.12/ 2684
30BB_0108	Sampling Register (QuadSPI1_SMPR)	32	R/W	0000_0000h	10.2.14.13/ 2684
30BB_010C	RX Buffer Status Register (QuadSPI1_RBSR)	32	R	0000_0000h	10.2.14.14/ 2685
30BB_0110	RX Buffer Control Register (QuadSPI1_RBCT)	32	R/W	0000_0000h	10.2.14.15/ 2686
30BB_0150	TX Buffer Status Register (QuadSPI1_TBSR)	32	R	0000_0000h	10.2.14.16/ 2687
30BB_0154	TX Buffer Data Register (QuadSPI1_TBDR)	32	R/W	0000_0000h	10.2.14.17/ 2687
30BB_015C	Status Register (QuadSPI1_SR)	32	R	0000_3800h	10.2.14.18/ 2689
30BB_0160	Flag Register (QuadSPI1_FR)	32	w1c	0800_0000h	10.2.14.19/ 2692
30BB_0164	Interrupt and DMA Request Select and Enable Register (QuadSPI1_RSER)	32	R/W	0000_0000h	10.2.14.20/ 2695
30BB_0168	Sequence Suspend Status Register (QuadSPI1_SPNDST)	32	R	0000_0000h	10.2.14.21/ 2698
30BB_016C	Sequence Pointer Clear Register (QuadSPI1_SPTRCLR)	32	R/W	0000_0000h	10.2.14.22/ 2700
30BB_0180	Serial Flash A1 Top Address (QuadSPI1_SFA1AD)	32	R/W	0000_0000h	10.2.14.23/ 2700
30BB_0184	Serial Flash A2 Top Address (QuadSPI1_SFA2AD)	32	R/W	0000_0000h	10.2.14.24/ 2701

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BB_0188	Serial Flash B1Top Address (QuadSPI1_SFB1AD)	32	R/W	0000_0000h	10.2.14.25/ 2701
30BB_018C	Serial Flash B2Top Address (QuadSPI1_SFB2AD)	32	R/W	0000_0000h	10.2.14.26/ 2702
30BB_0200	RX Buffer Data Register (QuadSPI1_RBDR0)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0204	RX Buffer Data Register (QuadSPI1_RBDR1)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0208	RX Buffer Data Register (QuadSPI1_RBDR2)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_020C	RX Buffer Data Register (QuadSPI1_RBDR3)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0210	RX Buffer Data Register (QuadSPI1_RBDR4)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0214	RX Buffer Data Register (QuadSPI1_RBDR5)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0218	RX Buffer Data Register (QuadSPI1_RBDR6)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_021C	RX Buffer Data Register (QuadSPI1_RBDR7)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0220	RX Buffer Data Register (QuadSPI1_RBDR8)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0224	RX Buffer Data Register (QuadSPI1_RBDR9)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0228	RX Buffer Data Register (QuadSPI1_RBDR10)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_022C	RX Buffer Data Register (QuadSPI1_RBDR11)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0230	RX Buffer Data Register (QuadSPI1_RBDR12)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0234	RX Buffer Data Register (QuadSPI1_RBDR13)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0238	RX Buffer Data Register (QuadSPI1_RBDR14)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_023C	RX Buffer Data Register (QuadSPI1_RBDR15)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0240	RX Buffer Data Register (QuadSPI1_RBDR16)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0244	RX Buffer Data Register (QuadSPI1_RBDR17)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0248	RX Buffer Data Register (QuadSPI1_RBDR18)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_024C	RX Buffer Data Register (QuadSPI1_RBDR19)	32	R/W	0000_0000h	10.2.14.27/ 2702

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BB_0250	RX Buffer Data Register (QuadSPI1_RBDR20)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0254	RX Buffer Data Register (QuadSPI1_RBDR21)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0258	RX Buffer Data Register (QuadSPI1_RBDR22)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_025C	RX Buffer Data Register (QuadSPI1_RBDR23)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0260	RX Buffer Data Register (QuadSPI1_RBDR24)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0264	RX Buffer Data Register (QuadSPI1_RBDR25)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0268	RX Buffer Data Register (QuadSPI1_RBDR26)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_026C	RX Buffer Data Register (QuadSPI1_RBDR27)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0270	RX Buffer Data Register (QuadSPI1_RBDR28)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0274	RX Buffer Data Register (QuadSPI1_RBDR29)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0278	RX Buffer Data Register (QuadSPI1_RBDR30)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_027C	RX Buffer Data Register (QuadSPI1_RBDR31)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_0300	LUT Key Register (QuadSPI1_LUTKEY)	32	R/W	5AF0_5AF0h	10.2.14.28/ 2703
30BB_0304	LUT Lock Configuration Register (QuadSPI1_LCKCR)	32	R/W	0000_0002h	10.2.14.29/ 2704
30BB_0310	Look-up Table register (QuadSPI1_LUT0)	32	R/W	0818_0403h	10.2.14.30/ 2705
30BB_0314	Look-up Table register (QuadSPI1_LUT1)	32	R/W	2400_1C08h	10.2.14.31/ 2706
30BB_0318	Look-up Table register (QuadSPI1_LUT2)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_031C	Look-up Table register (QuadSPI1_LUT3)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0320	Look-up Table register (QuadSPI1_LUT4)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0324	Look-up Table register (QuadSPI1_LUT5)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0328	Look-up Table register (QuadSPI1_LUT6)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_032C	Look-up Table register (QuadSPI1_LUT7)	32	R/W	0000_0000h	10.2.14.32/ 2707

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BB_0330	Look-up Table register (QuadSPI1_LUT8)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0334	Look-up Table register (QuadSPI1_LUT9)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0338	Look-up Table register (QuadSPI1_LUT10)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_033C	Look-up Table register (QuadSPI1_LUT11)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0340	Look-up Table register (QuadSPI1_LUT12)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0344	Look-up Table register (QuadSPI1_LUT13)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0348	Look-up Table register (QuadSPI1_LUT14)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_034C	Look-up Table register (QuadSPI1_LUT15)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0350	Look-up Table register (QuadSPI1_LUT16)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0354	Look-up Table register (QuadSPI1_LUT17)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0358	Look-up Table register (QuadSPI1_LUT18)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_035C	Look-up Table register (QuadSPI1_LUT19)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0360	Look-up Table register (QuadSPI1_LUT20)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0364	Look-up Table register (QuadSPI1_LUT21)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0368	Look-up Table register (QuadSPI1_LUT22)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_036C	Look-up Table register (QuadSPI1_LUT23)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0370	Look-up Table register (QuadSPI1_LUT24)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0374	Look-up Table register (QuadSPI1_LUT25)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0378	Look-up Table register (QuadSPI1_LUT26)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_037C	Look-up Table register (QuadSPI1_LUT27)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0380	Look-up Table register (QuadSPI1_LUT28)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0384	Look-up Table register (QuadSPI1_LUT29)	32	R/W	0000_0000h	10.2.14.32/ 2707

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BB_0388	Look-up Table register (QuadSPI1_LUT30)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_038C	Look-up Table register (QuadSPI1_LUT31)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0390	Look-up Table register (QuadSPI1_LUT32)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0394	Look-up Table register (QuadSPI1_LUT33)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0398	Look-up Table register (QuadSPI1_LUT34)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_039C	Look-up Table register (QuadSPI1_LUT35)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03A0	Look-up Table register (QuadSPI1_LUT36)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03A4	Look-up Table register (QuadSPI1_LUT37)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03A8	Look-up Table register (QuadSPI1_LUT38)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03AC	Look-up Table register (QuadSPI1_LUT39)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03B0	Look-up Table register (QuadSPI1_LUT40)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03B4	Look-up Table register (QuadSPI1_LUT41)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03B8	Look-up Table register (QuadSPI1_LUT42)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03BC	Look-up Table register (QuadSPI1_LUT43)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03C0	Look-up Table register (QuadSPI1_LUT44)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03C4	Look-up Table register (QuadSPI1_LUT45)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03C8	Look-up Table register (QuadSPI1_LUT46)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03CC	Look-up Table register (QuadSPI1_LUT47)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03D0	Look-up Table register (QuadSPI1_LUT48)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03D4	Look-up Table register (QuadSPI1_LUT49)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03D8	Look-up Table register (QuadSPI1_LUT50)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03DC	Look-up Table register (QuadSPI1_LUT51)	32	R/W	0000_0000h	10.2.14.32/ 2707

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BB_03E0	Look-up Table register (QuadSPI1_LUT52)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03E4	Look-up Table register (QuadSPI1_LUT53)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03E8	Look-up Table register (QuadSPI1_LUT54)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03EC	Look-up Table register (QuadSPI1_LUT55)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03F0	Look-up Table register (QuadSPI1_LUT56)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03F4	Look-up Table register (QuadSPI1_LUT57)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03F8	Look-up Table register (QuadSPI1_LUT58)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_03FC	Look-up Table register (QuadSPI1_LUT59)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0400	Look-up Table register (QuadSPI1_LUT60)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0404	Look-up Table register (QuadSPI1_LUT61)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_0408	Look-up Table register (QuadSPI1_LUT62)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_040C	Look-up Table register (QuadSPI1_LUT63)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4000	Module Configuration Register (QuadSPI2_MCR)	32	R/W	000F_4000h	10.2.14.1/ 2673
30BB_4008	IP Configuration Register (QuadSPI2_IPCR)	32	R/W	0000_0000h	10.2.14.2/ 2675
30BB_400C	Flash Configuration Register (QuadSPI2_FLSHCR)	32	R/W	0000_0303h	10.2.14.3/ 2676
30BB_4010	Buffer0 Configuration Register (QuadSPI2_BUF0CR)	32	R/W	0000_0000h	10.2.14.4/ 2677
30BB_4014	Buffer1 Configuration Register (QuadSPI2_BUF1CR)	32	R/W	0000_0000h	10.2.14.5/ 2678
30BB_4018	Buffer2 Configuration Register (QuadSPI2_BUF2CR)	32	R/W	0000_0000h	10.2.14.6/ 2679
30BB_401C	Buffer3 Configuration Register (QuadSPI2_BUF3CR)	32	R/W	See section	10.2.14.7/ 2680
30BB_4020	Buffer Generic Configuration Register (QuadSPI2_BFGENCR)	32	R/W	0000_0000h	10.2.14.8/ 2681
30BB_4030	Buffer0 Top Index Register (QuadSPI2_BUF0IND)	32	R/W	0000_0000h	10.2.14.9/ 2682
30BB_4034	Buffer1 Top Index Register (QuadSPI2_BUF1IND)	32	R/W	0000_0000h	10.2.14.10/ 2682

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BB_4038	Buffer2 Top Index Register (QuadSPI2_BUF2IND)	32	R/W	0000_0000h	10.2.14.11/ 2683
30BB_4100	Serial Flash Address Register (QuadSPI2_SFAR)	32	R/W	0000_0000h	10.2.14.12/ 2684
30BB_4108	Sampling Register (QuadSPI2_SMPR)	32	R/W	0000_0000h	10.2.14.13/ 2684
30BB_410C	RX Buffer Status Register (QuadSPI2_RBSR)	32	R	0000_0000h	10.2.14.14/ 2685
30BB_4110	RX Buffer Control Register (QuadSPI2_RBCT)	32	R/W	0000_0000h	10.2.14.15/ 2686
30BB_4150	TX Buffer Status Register (QuadSPI2_TBSR)	32	R	0000_0000h	10.2.14.16/ 2687
30BB_4154	TX Buffer Data Register (QuadSPI2_TBDR)	32	R/W	0000_0000h	10.2.14.17/ 2687
30BB_415C	Status Register (QuadSPI2_SR)	32	R	0000_3800h	10.2.14.18/ 2689
30BB_4160	Flag Register (QuadSPI2_FR)	32	w1c	0800_0000h	10.2.14.19/ 2692
30BB_4164	Interrupt and DMA Request Select and Enable Register (QuadSPI2_RSER)	32	R/W	0000_0000h	10.2.14.20/ 2695
30BB_4168	Sequence Suspend Status Register (QuadSPI2_SPNDST)	32	R	0000_0000h	10.2.14.21/ 2698
30BB_416C	Sequence Pointer Clear Register (QuadSPI2_SPTRCLR)	32	R/W	0000_0000h	10.2.14.22/ 2700
30BB_4180	Serial Flash A1 Top Address (QuadSPI2_SFA1AD)	32	R/W	0000_0000h	10.2.14.23/ 2700
30BB_4184	Serial Flash A2 Top Address (QuadSPI2_SFA2AD)	32	R/W	0000_0000h	10.2.14.24/ 2701
30BB_4188	Serial Flash B1Top Address (QuadSPI2_SFB1AD)	32	R/W	0000_0000h	10.2.14.25/ 2701
30BB_418C	Serial Flash B2Top Address (QuadSPI2_SFB2AD)	32	R/W	0000_0000h	10.2.14.26/ 2702
30BB_4200	RX Buffer Data Register (QuadSPI2_RBDR0)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4204	RX Buffer Data Register (QuadSPI2_RBDR1)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4208	RX Buffer Data Register (QuadSPI2_RBDR2)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_420C	RX Buffer Data Register (QuadSPI2_RBDR3)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4210	RX Buffer Data Register (QuadSPI2_RBDR4)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4214	RX Buffer Data Register (QuadSPI2_RBDR5)	32	R/W	0000_0000h	10.2.14.27/ 2702

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BB_4218	RX Buffer Data Register (QuadSPI2_RBDR6)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_421C	RX Buffer Data Register (QuadSPI2_RBDR7)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4220	RX Buffer Data Register (QuadSPI2_RBDR8)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4224	RX Buffer Data Register (QuadSPI2_RBDR9)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4228	RX Buffer Data Register (QuadSPI2_RBDR10)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_422C	RX Buffer Data Register (QuadSPI2_RBDR11)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4230	RX Buffer Data Register (QuadSPI2_RBDR12)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4234	RX Buffer Data Register (QuadSPI2_RBDR13)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4238	RX Buffer Data Register (QuadSPI2_RBDR14)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_423C	RX Buffer Data Register (QuadSPI2_RBDR15)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4240	RX Buffer Data Register (QuadSPI2_RBDR16)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4244	RX Buffer Data Register (QuadSPI2_RBDR17)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4248	RX Buffer Data Register (QuadSPI2_RBDR18)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_424C	RX Buffer Data Register (QuadSPI2_RBDR19)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4250	RX Buffer Data Register (QuadSPI2_RBDR20)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4254	RX Buffer Data Register (QuadSPI2_RBDR21)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4258	RX Buffer Data Register (QuadSPI2_RBDR22)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_425C	RX Buffer Data Register (QuadSPI2_RBDR23)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4260	RX Buffer Data Register (QuadSPI2_RBDR24)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4264	RX Buffer Data Register (QuadSPI2_RBDR25)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4268	RX Buffer Data Register (QuadSPI2_RBDR26)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_426C	RX Buffer Data Register (QuadSPI2_RBDR27)	32	R/W	0000_0000h	10.2.14.27/ 2702

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BB_4270	RX Buffer Data Register (QuadSPI2_RBDR28)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4274	RX Buffer Data Register (QuadSPI2_RBDR29)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4278	RX Buffer Data Register (QuadSPI2_RBDR30)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_427C	RX Buffer Data Register (QuadSPI2_RBDR31)	32	R/W	0000_0000h	10.2.14.27/ 2702
30BB_4300	LUT Key Register (QuadSPI2_LUTKEY)	32	R/W	5AF0_5AF0h	10.2.14.28/ 2703
30BB_4304	LUT Lock Configuration Register (QuadSPI2_LCKCR)	32	R/W	0000_0002h	10.2.14.29/ 2704
30BB_4310	Look-up Table register (QuadSPI2_LUT0)	32	R/W	0818_0403h	10.2.14.30/ 2705
30BB_4314	Look-up Table register (QuadSPI2_LUT1)	32	R/W	2400_1C08h	10.2.14.31/ 2706
30BB_4318	Look-up Table register (QuadSPI2_LUT2)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_431C	Look-up Table register (QuadSPI2_LUT3)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4320	Look-up Table register (QuadSPI2_LUT4)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4324	Look-up Table register (QuadSPI2_LUT5)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4328	Look-up Table register (QuadSPI2_LUT6)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_432C	Look-up Table register (QuadSPI2_LUT7)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4330	Look-up Table register (QuadSPI2_LUT8)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4334	Look-up Table register (QuadSPI2_LUT9)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4338	Look-up Table register (QuadSPI2_LUT10)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_433C	Look-up Table register (QuadSPI2_LUT11)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4340	Look-up Table register (QuadSPI2_LUT12)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4344	Look-up Table register (QuadSPI2_LUT13)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4348	Look-up Table register (QuadSPI2_LUT14)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_434C	Look-up Table register (QuadSPI2_LUT15)	32	R/W	0000_0000h	10.2.14.32/ 2707

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BB_4350	Look-up Table register (QuadSPI2_LUT16)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4354	Look-up Table register (QuadSPI2_LUT17)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4358	Look-up Table register (QuadSPI2_LUT18)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_435C	Look-up Table register (QuadSPI2_LUT19)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4360	Look-up Table register (QuadSPI2_LUT20)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4364	Look-up Table register (QuadSPI2_LUT21)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4368	Look-up Table register (QuadSPI2_LUT22)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_436C	Look-up Table register (QuadSPI2_LUT23)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4370	Look-up Table register (QuadSPI2_LUT24)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4374	Look-up Table register (QuadSPI2_LUT25)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4378	Look-up Table register (QuadSPI2_LUT26)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_437C	Look-up Table register (QuadSPI2_LUT27)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4380	Look-up Table register (QuadSPI2_LUT28)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4384	Look-up Table register (QuadSPI2_LUT29)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4388	Look-up Table register (QuadSPI2_LUT30)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_438C	Look-up Table register (QuadSPI2_LUT31)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4390	Look-up Table register (QuadSPI2_LUT32)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4394	Look-up Table register (QuadSPI2_LUT33)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4398	Look-up Table register (QuadSPI2_LUT34)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_439C	Look-up Table register (QuadSPI2_LUT35)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43A0	Look-up Table register (QuadSPI2_LUT36)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43A4	Look-up Table register (QuadSPI2_LUT37)	32	R/W	0000_0000h	10.2.14.32/ 2707

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BB_43A8	Look-up Table register (QuadSPI2_LUT38)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43AC	Look-up Table register (QuadSPI2_LUT39)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43B0	Look-up Table register (QuadSPI2_LUT40)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43B4	Look-up Table register (QuadSPI2_LUT41)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43B8	Look-up Table register (QuadSPI2_LUT42)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43BC	Look-up Table register (QuadSPI2_LUT43)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43C0	Look-up Table register (QuadSPI2_LUT44)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43C4	Look-up Table register (QuadSPI2_LUT45)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43C8	Look-up Table register (QuadSPI2_LUT46)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43CC	Look-up Table register (QuadSPI2_LUT47)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43D0	Look-up Table register (QuadSPI2_LUT48)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43D4	Look-up Table register (QuadSPI2_LUT49)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43D8	Look-up Table register (QuadSPI2_LUT50)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43DC	Look-up Table register (QuadSPI2_LUT51)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43E0	Look-up Table register (QuadSPI2_LUT52)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43E4	Look-up Table register (QuadSPI2_LUT53)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43E8	Look-up Table register (QuadSPI2_LUT54)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43EC	Look-up Table register (QuadSPI2_LUT55)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43F0	Look-up Table register (QuadSPI2_LUT56)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43F4	Look-up Table register (QuadSPI2_LUT57)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43F8	Look-up Table register (QuadSPI2_LUT58)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_43FC	Look-up Table register (QuadSPI2_LUT59)	32	R/W	0000_0000h	10.2.14.32/ 2707

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BB_4400	Look-up Table register (QuadSPI2_LUT60)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4404	Look-up Table register (QuadSPI2_LUT61)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_4408	Look-up Table register (QuadSPI2_LUT62)	32	R/W	0000_0000h	10.2.14.32/ 2707
30BB_440C	Look-up Table register (QuadSPI2_LUT63)	32	R/W	0000_0000h	10.2.14.32/ 2707

10.2.14.1 Module Configuration Register (QuadSPIx_MCR)

The QuadSPI_MCR holds configuration data associated with QuadSPI operation.

Write:

- *All other fields: Anytime*

Address: Base address + 0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved					DQS_PHASE_EN	DQS_LOOPBACK_EN	DQS_LOOPBACK_FROM_PAD	Reserved				Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved	MDIS	Reserved		CLR_TXF	CLR_RXF	Reserved		DDR_EN	DQS_EN	Reserved		END_CFG		SWRSTHD	SWRSTSD
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_MCR field descriptions

Field	Description
31–27 Reserved	This field is reserved.

Table continues on the next page...

QuadSPIx_MCR field descriptions (continued)

Field	Description
26 DQS_PHASE_EN	This bit controls internal DQS output phase. If DQS_EN and DQS_LOOPBACK_EN are both set to 1, this bit should be set in SDR mode, and cleared in DDR mode. If either DQS_EN or DQS_LOOPBACK_EN is set to 0, this bit is ignored.
25 DQS_LOOPBACK_EN	Quadspi will output serial data strobe signal which will be loopback from pad to sample input flash serial data. Please note pad should be force input for loopback.Quadspi will output serial data strobe signal which will be loopback from pad to sample input flash serial data. Please note pad should be force input for loopback. This bit is a don't care when DQS_EN is set to 0. 1'b1 DQS loopback sampling enabled 1'b0 DQS loopback sampling disabled
24 DQS_LOOPBACK_FROM_PAD	This bit should always be set to '1' when DQS_LOOPBACK_EN is set to '1'. When DQS_LOOPBACK_EN is set to '1', this bit is ignored.
23–20 Reserved	This field is reserved.
19–16 Reserved	This field is reserved. This field is reserved and should always be set to 0xF.
15 Reserved	This field is reserved. This field is reserved.
14 MDIS	Module Disable. The MDIS bit allows the clock to the non-memory mapped logic in the QuadSPI to be stopped, putting the QuadSPI in a software controlled power-saving state. Please refer to , for more information. 0 Enable QuadSPI clocks. 1 Allow external logic to disable QuadSPI clocks.
13–12 Reserved	This field is reserved.
11 CLR_TXF	Clear TX FIFO/Buffer. Invalidate the TX Buffer content. 0 No action. 1 Read and write pointers of the TX Buffer are reset to 0. QSPI_TBSR[TRCTR] is reset to 0.
10 CLR_RXF	Clear RX FIFO. Invalidate the RX Buffer. 0 No action. 1 Read and write pointers of the RX Buffer are reset to 0. QSPI_RBSR[RDBFL] is reset to 0.
9–8 Reserved	This field is reserved.
7 DDR_EN	DDR mode enable: 0 2x and 4x clocks are disabled for SDR instructions only 1 2x and 4x clocks are enabled supports both SDR and DDR instruction.
6 DQS_EN	DQS enable: This field is valid for both SDR and DDR mode.For more details Refer Data Strobe Signal Functionality

Table continues on the next page...

QuadSPIx_MCR field descriptions (continued)

Field	Description
	0 DQS disabled. 1 DQS enabled- When enabled, the incoming data is sampled on both the edges of DQS input when QSPI_MCR[DDR_EN] is set, else, on only one edge when QSPI_MCR[DDR_EN] is 0. The QSPI_SMPR[DDR_SMP] values are ignored.
5–4 Reserved	This field is reserved.
3–2 END_CFG	Defines the endianness of the QSPI module.For more details refer to Byte Ordering Endianess
1 SWRSTHD	Software reset for AHB domain **NOTE:** Please keep other fields value when write to SWRSTHD and SWRSTSD **NOTE:** These software reset don't reset register setting but only reset internal flip-flops in quadspi controller **NOTE:** To remove the reset, need to write 0 to SWRSTHD and SWRSTSD 0 No action 1 AHB domain flops are reset. Does not reset configuration registers. It is advisable to reset both the serial flash domain and AHB domain at the same time. Resetting only one domain might lead to side effects.
0 SWRSTSD	Software reset for Serial Flash domain **NOTE:** Please keep other fields value when write to SWRSTHD and SWRSTSD **NOTE:** These software reset don't reset register setting but only reset internal flip-flops in quadspi controller **NOTE:** To remove the reset, need to write 0 to SWRSTHD and SWRSTSD 0 No action 1 Serial Flash domain flops are reset. Does not reset configuration registers. It is advisable to reset both the serial flash domain and AHB domain at the same time. Resetting only one domain might lead to side effects.

10.2.14.2 IP Configuration Register (QuadSPIx_IPCR)

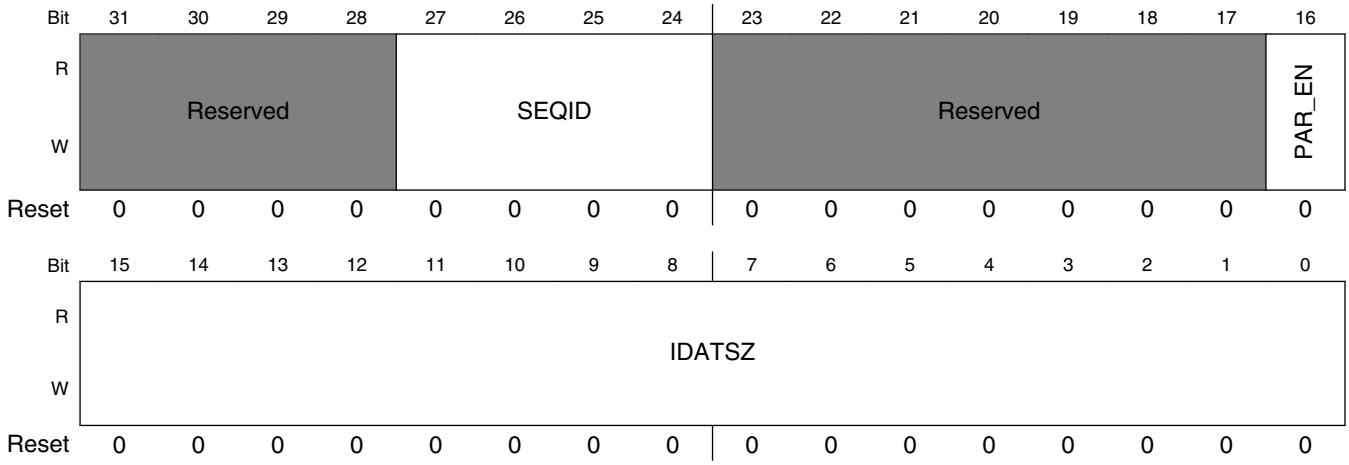
The IP configuration register provides all the configuration required for an IP initiated command. An IP command can be triggered by writing in the SEQID field of this register. If the SEQID field is written successfully, a new command to the external serial flash is started as per the sequence pointed to by the SEQID field. Refer to Normal Mode , for details about the command triggering and command execution.

Write:

- *QSPI_SR[IP_ACC]=0*

AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31)

Address: Base address + 8h offset



QuadSPIx_IPCR field descriptions

Field	Description
31–28 Reserved	This field is reserved.
27–24 SEQID	Points to a sequence in the Look-up-table. The SEQID defines the bits [6:2] of the LUT index. The bits [1:0] are always assumed to be 0. Refer to Look-up Table for more details. A write to this bit -field triggers a transaction on the serial flash interface.
23–17 Reserved	This field is reserved.
16 PAR_EN	When set, a transaction to two serial flash devices is triggered in parallel mode. Refer to Parallel Flash Mode for more details.
IDATSZ	IP data transfer size: Defines the data transfer size in bytes of the IP command.

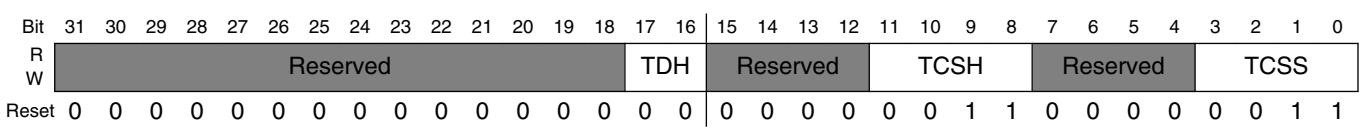
10.2.14.3 Flash Configuration Register (QuadSPIx_FLSHCR)

The Flash configuration register contains the flash device specific timings that must be met by the QuadSPI controller for the device to function correctly.

Write:

- *QSPI_SR[AHB_ACC] = 0*
- *QSPI_SR[IP_ACC] = 0*

Address: Base address + Ch offset



QuadSPIx_FLSHCR field descriptions

Field	Description
31–18 Reserved	This field is reserved.
17–16 TDH	Serial flash Data In hold time: This helps in meeting the Data In Hold time requirement of a Flash. This is valid only in DDR mode. Refer to Data input hold requirement of Flash for details. **NOTE:** This field should be set to 01 in DDR mode (DDR_EN=1) and set to 00 in SDR mode (DDR_EN=0). Other value are reserved. 00 Data aligned with the posedge of Internal reference clock of QuadSPI 01 Data aligned with 2x serial flash half clock 10 Reserved 11 Reserved
15–12 Reserved	This field is reserved.
11–8 TCSH	Serial flash CS hold time in terms of serial flash clock cycles. **NOTE:** The actual delay between chip select assertion and clock fall edge is defined as: • 1 SCK cycle if TCSH is set 0 or 1 • N SCK cycle if TCSH is set N (N>1)
7–4 Reserved	This field is reserved. Reserved.
TCSS	Serial flash CS setup time in terms of serial flash clock cycles. **NOTE:** 1. The actual delay between chip select assertion and clock fall edge is defined as: • 0.5 SCK cycle if TCSS is set 0 or 1 • N+0.5 SCK cycle if TCSS is set N (N>1) 2. Any update to TCSS register bits is visible on the flash interface only from the second transaction following the update.

10.2.14.4 Buffer0 Configuration Register (QuadSPIx_BUF0CR)

This register provides the configuration for any access to buffer0. An access is routed to buffer0 when the master port number of the incoming AHB request matches the MSTRID field of the BUF0CR. Any buffer "miss" leads to a serial flash transaction being triggered as per the sequence pointed to the SEQID field. Buffer0 may also be configured as a high priority buffer by setting the HP_EN field of this register.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HP_EN	Reserved														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADATSZ								Reserved				MSTRID			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_BUF0CR field descriptions

Field	Description
31 HP_EN	High Priority Enable: When set, the master associated with this buffer is assigned a priority higher than the rest of the masters. An access by a high priority master will suspend any ongoing prefetch by another AHB master and will be serviced on high priority. Refer to Flexible AHB Buffers for details.
30–16 Reserved	This field is reserved.
15–8 ADATSZ	AHB data transfer size: Defines the data transfer size in 8 bytes of an AHB triggered access to serial flash. For example,a value of 0x2 will set transfer size to 16bytes.When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. SW should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved. Reserved.
MSTRID	Master ID: The ID of the AHB master associated with BUFFER0. Any AHB access with this master port number is routed to this buffer.

10.2.14.5 Buffer1 Configuration Register (QuadSPIx_BUF1CR)

This register provides the configuration for any access to buffer1. An access is routed to buffer1 when the master port number of the incoming AHB request matches the MSTRID field of the BUF1CR. Any buffer "miss" leads to the buffer being flushed and a serial flash transaction being triggered as per the sequence pointed to by the SEQID field.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																ADATSZ								Reserved				MSTRID			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_BUF1CR field descriptions

Field	Description
31–16 Reserved	This field is reserved.
15–8 ADATSZ	AHB data transfer size: Defines the data transfer size in 8 bytes of an AHB triggered access to serial flash.For example, a value of 0x2 will set transfer size to 16bytes. When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. SW should ensure that this transfer size is not greater than the size of this buffer.

Table continues on the next page...

QuadSPIx_BUF1CR field descriptions (continued)

Field	Description
7–4 Reserved	This field is reserved.
MSTRID	Master ID: The ID of the AHB master associated with BUFFER1. Any AHB access with this master port number is routed to this buffer.

10.2.14.6 Buffer2 Configuration Register (QuadSPIx_BUF2CR)

This register provides the configuration for any access to buffer2. An access is routed to buffer2 when the master port number of the incoming AHB request matches the MSTRID field of the BUF2CR. Any buffer "miss" leads to the buffer being flushed and a serial flash transaction being triggered as per the sequence pointed to by the SEQID field.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: Base address + 18h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																ADATSZ								Reserved				MSTRID			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_BUF2CR field descriptions

Field	Description
31–16 Reserved	This field is reserved. Reserved.
15–8 ADATSZ	AHB data transfer size: Defines the data transfer size in 8 Bytes of an AHB triggered access to serial flash. For example, a value of 0x2 will set transfer size to 16bytes.When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. SW should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved. Reserved.
MSTRID	Master ID: The ID of the AHB master associated with BUFFER2. Any AHB access with this master port number is routed to this buffer.

10.2.14.7 Buffer3 Configuration Register (QuadSPIx_BUF3CR)

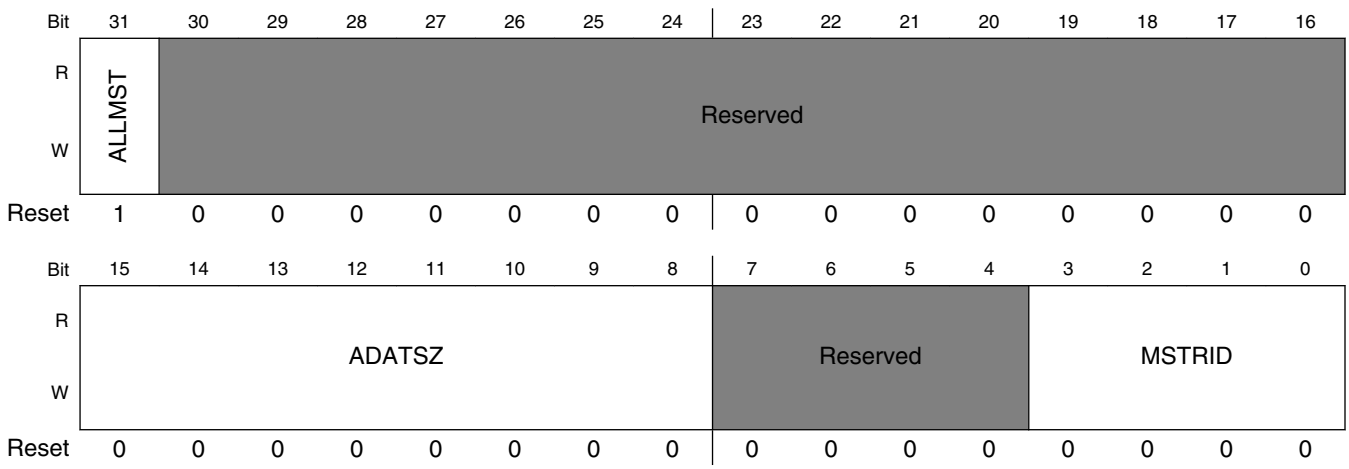
This register provides the configuration for any access to buffer3. An access is routed to buffer3 when the master port number of the incoming AHB request matches the MSTRID field of the BUF3CR. Any buffer "miss" leads to the buffer being flushed a serial flash transaction being triggered as per the sequence pointed to by the SEQID field. If the ALLMST field is set, any transaction where the master port number does not match any of the buffer MSTRID fields will be routed to this buffer. In the case that the ALLMST field is not set, any such transaction (where master port number does not match any of the MSTRID fields) will be returned an ERROR response.

Write:

- *QSPI_SR[AHB_ACC] = 0*

.

Address: Base address + 1Ch offset



QuadSPIx_BUF3CR field descriptions

Field	Description
31 ALLMST	All master enable: When set, buffer3 acts as an all-master buffer. Any AHB access with a master port number not matching with the master ID of buffer0 or buffer1 or buffer2 is routed to buffer3. When set, the MSTRID field of this register is ignored.
30–16 Reserved	This field is reserved. Reserved.
15–8 ADATSZ	AHB data transfer size: Defines the data transfer size in 8 Bytes of an AHB triggered access to serial flash. When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. SW should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved.
MSTRID	Master ID: The ID of the AHB master associated with BUFFER3. Any AHB access with this master port number is routed to this buffer.

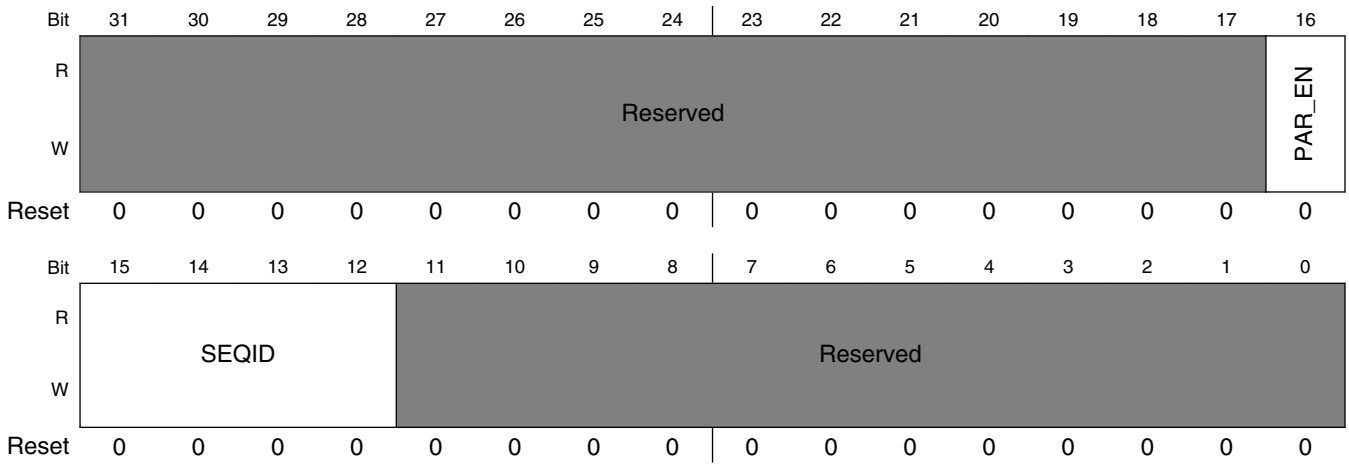
10.2.14.8 Buffer Generic Configuration Register (QuadSPIx_BFGENCR)

This register provides the generic configuration to any of the buffer accesses. Any buffer "miss" leads to the buffer being flushed and a serial flash transaction being triggered as per the sequence pointed to by the SEQID field. If the PAR_EN field is set, all the buffer accesses result in parallel accesses to the flashes.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: Base address + 20h offset



QuadSPIx_BFGENCR field descriptions

Field	Description
31–17 Reserved	This field is reserved.
16 PAR_EN	When set, a transaction to two serial flash devices is triggered in parallel mode. Refer to Parallel Flash Mode for more details.
15–12 SEQID	Points to a sequence in the Look-up-table. The SEQID defines the bits [6:2] of the LUT index. The bits [1:0] are always assumed to be 0. Refer to Look-up Table. **NOTE:** If the sequence pointer differs between the new and previous sequence then the user should reset this. See QSPI_SPTRCLR for more information.
Reserved	This field is reserved.

10.2.14.9 Buffer0 Top Index Register (QuadSPIx_BUF0IND)

This register specifies the top index of buffer0, which defines its size. Note that that the 3 LSBs of this register are set to zero - this ensures that the buffer is 64bit aligned, as each buffer entry is 64bits long.

The register value should be set to the desired number of bytes less 8. For example, setting BUF0IND to 0 gives 8 bytes, 1 give 16bytes etc.

The size of buffer0 is the difference between the BUF0IND+8 and 0.

It is the responsibility of the software to ensure that BUF0IND value is not greater than the overall size of the buffer. The hardware does not provide any protection against illegal programming.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: Base address + 30h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TPINDX0																													Reserved		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_BUF0IND field descriptions

Field	Description
31–3 TPINDX0	Top index of buffer 0.
Reserved	This field is reserved. Reserved.

10.2.14.10 Buffer1 Top Index Register (QuadSPIx_BUF1IND)

This register specifies the top index of buffer1, which defines its size. Note that the 3 LSBs of this register are set to zero - this ensures that the buffer is 64bit aligned as each buffer entry is 64bits long.

The register value should be set to the desired number of bytes less. The size of buffer1 is the difference between the BUF1IND and BUF0IND.

It is the responsibility of the software to ensure that BUF1IND value is not greater than the overall size of the buffer. The hardware does not provide any protection against illegal programming.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: Base address + 34h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TPINDX1																													Reserved		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_BUF1IND field descriptions

Field	Description
31–3 TPINDX1	Top index of buffer 1.
Reserved	This field is reserved.

10.2.14.11 Buffer2 Top Index Register (QuadSPIx_BUF2IND)

This register specifies the top index of buffer2, which defines its size. Note that that the 3 LSBs of this register are set to zero - this ensures that the buffer is 64bit aligned as each buffer entry is 64bits long.

The register value should be set to the desired number of bytes less 8. The size of buffer2 is the difference between the BUF2IND and BUF1IND.

It is the responsibility of the software to ensure that BUF2IND value is not greater than the overall size of the buffer. The hardware does not provide any protection against illegal programming.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: Base address + 38h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TPINDX2																													Reserved		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_BUF2IND field descriptions

Field	Description
31–3 TPINDX2	Top index of buffer 2.
Reserved	This field is reserved.

10.2.14.12 Serial Flash Address Register (QuadSPIx_SFAR)

The module automatically translates this address on the memory map to the address on the flash itself. When operating in 24bit mode, only bits 23-0 are sent to the flash, in 32bit mode, bits 27-0 are used with bits 31-28 driven to 0 Refer to Table 10-7 for the mapping between the access mode and the QSPI_SFAR content and to Normal Mode for details about the command triggering and command execution. The software should ensure that the serial flash address provided in the QSPI_SFAR register lies in the valid flash address range as defined in Table 10-7.

Write:

- *QSPI_SR[IP_ACC] = 0*

Address: Base address + 100h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SFADR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_SFAR field descriptions

Field	Description
SFADR	Serial Flash Address. The register content is used as byte address for all following IP Commands.

10.2.14.13 Sampling Register (QuadSPIx_SMPR)

The Sampling Register allows configuration of how the incoming data from the external serial flash devices are sampled in the QuadSPI module.

Write:Disabled Mode

Address: Base address + 108h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0													DDRSMP		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								SDRSMP			0		0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

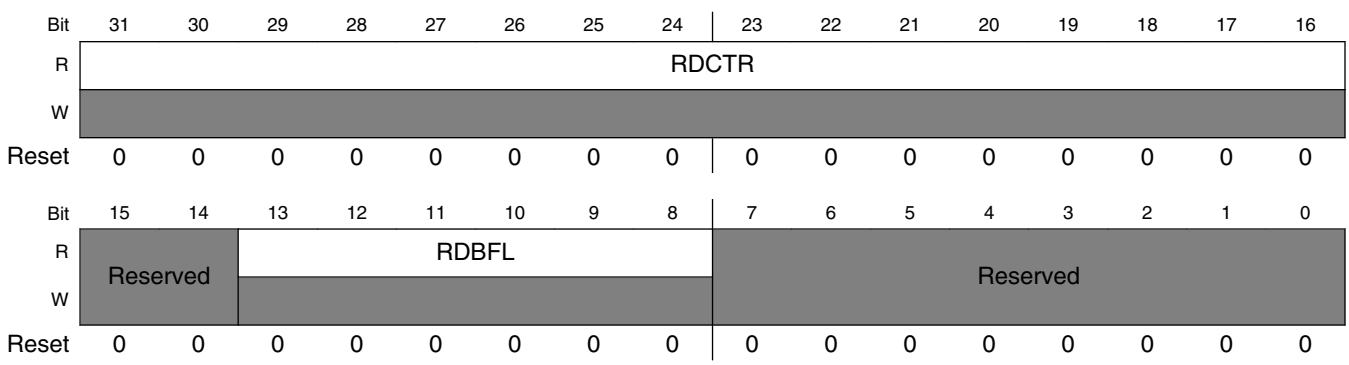
QuadSPIx_SMPR field descriptions

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–16 DDRSMP	DDR Sampling point. Select the sampling point for incoming data when serial flash is executing a DDR instruction. Refer to Input timing in DDR mode with internal sampling , for details on the sampling points.
15–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–5 SDRSMP	SDR sampling point.
4–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

10.2.14.14 RX Buffer Status Register (QuadSPIx_RBSR)

This register contains information related to the receive data buffer.

Address: Base address + 10Ch offset



QuadSPIx_RBSR field descriptions

Field	Description
31–16 RDCTR	Read Counter, indicates how many entries of 4 bytes have been removed from the RX Buffer.For example a value of 0x2 would indicate 8bytes have been removed It is incremented by the number (QSPI_RBCT[WMRK] + 1) on RX Buffer POP event. The RX Buffer can be popped using DMA or pop flag QSPI_FR[RBDF]. The QSPI_RSER[RBDDE] defines which pop has to be done. For further details please refer to AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31) and "Data Transfer from the QuadSPI Module Internal Buffers section in Flash Read section.
15–14 Reserved	This field is reserved.
13–8 RDBFL	RX Buffer Fill Level, indicates how many entries of 4 bytes are still available in the RX Buffer.For example a value of 0x2 would indicate 8bytes are available.
Reserved	This field is reserved.

10.2.14.15 RX Buffer Control Register (QuadSPIx_RBCT)

This register contains control data related to the receive data buffer.

Write:

- *QSPI_SR[IP_ACC] = 0*

Address: Base address + 110h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved							RXBRD	Reserved			WMRK				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_RBCT field descriptions

Field	Description
31–9 Reserved	This field is reserved.
8 RXBRD	RX Buffer Readout: This bit specifies the access scheme for the RX Buffer readout. 0 RX Buffer content is read using the AHB Bus registers QSPI_ARDB0 to QSPI_ARDB31. For details, refer to Exclusive Access to Serial Flash for AHB Commands. 1 RX Buffer content is read using the IP Bus registers QSPI_RBDR0 to QSPI_RBDR31.
7–5 Reserved	This field is reserved.
WMRK	RX Buffer Watermark: This field determines when the readout action of the RX Buffer is triggered. When the number of valid entries in the RX Buffer is equal to or greater than the number given by (WMRK+1) the QSPI_SR[RXWE] flag is asserted.The value should be entered as the number of 4byte entries minus 1. For example a value of 0x0 would set the watermark to 4bytes, 1 to 8bytes, 2 to 12bytes etc. For details, refer to DMA Usage.

10.2.14.16 TX Buffer Status Register (QuadSPIx_TBSR)

This register contains information related to the transmit data buffer.

Address: Base address + 150h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TRCTR																Reserved			TRBFL					Reserved								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_TBSR field descriptions

Field	Description
31–16 TRCTR	Transmit Counter. This field indicates how many entries of 4 bytes have been written into the TX Buffer by host accesses. It is reset to 0 when a 1 is written into the QSPI_MCR[CLR_TXF] bit. It is incremented on each write access to the QSPI_TBDR register when another word has been pushed onto the TX Buffer. When it is not cleared the TRCTR field wraps around to 0. Refer to TX Buffer Data Register (QuadSPI_TBDR) for details.
15–13 Reserved	This field is reserved.
12–8 TRBFL	TX Buffer Fill Level. The TRBFL field contains the number of entries of 4 bytes each available in the TX Buffer for the QuadSPI module to transmit to the serial flash device.
Reserved	This field is reserved.

10.2.14.17 TX Buffer Data Register (QuadSPIx_TBDR)

The QSPI_TBDR register provides access to the circular TX Buffer of depth 128 bytes. This buffer provides the data written into it as write data for the page programming commands to the serial flash device. Refer to Table 10-16 for the byte ordering scheme. A write transaction on the flash with data size of less than 32 bits will lead to the removal of four data entry from the TX buffer. The valid bits will be used and the rest of the bits will be discarded.

Write:

- *QSPI_SR[TXFULL] = 0*

32-bit write access required

NOTE

There is no limitation on Flash programming size. But the data size written to TX Buffer should be 64 Byte aligned. If flash programming size is not 64 bytes aligned, please write redundant data to TX Buffers. The redundant data will be ignored by QuadSPI controller.

Address: Base address + 154h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TXDATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

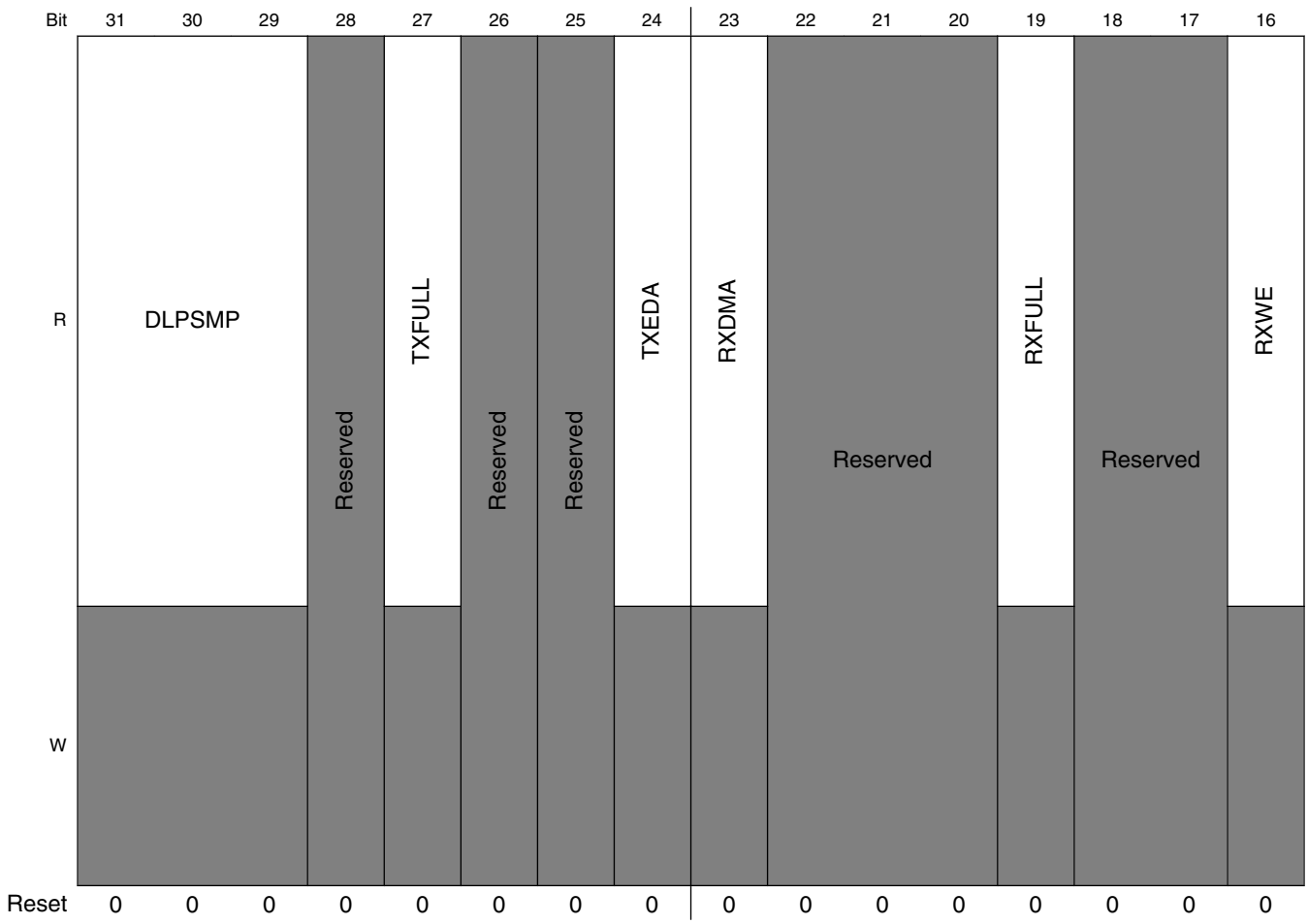
QuadSPIx_TBDR field descriptions

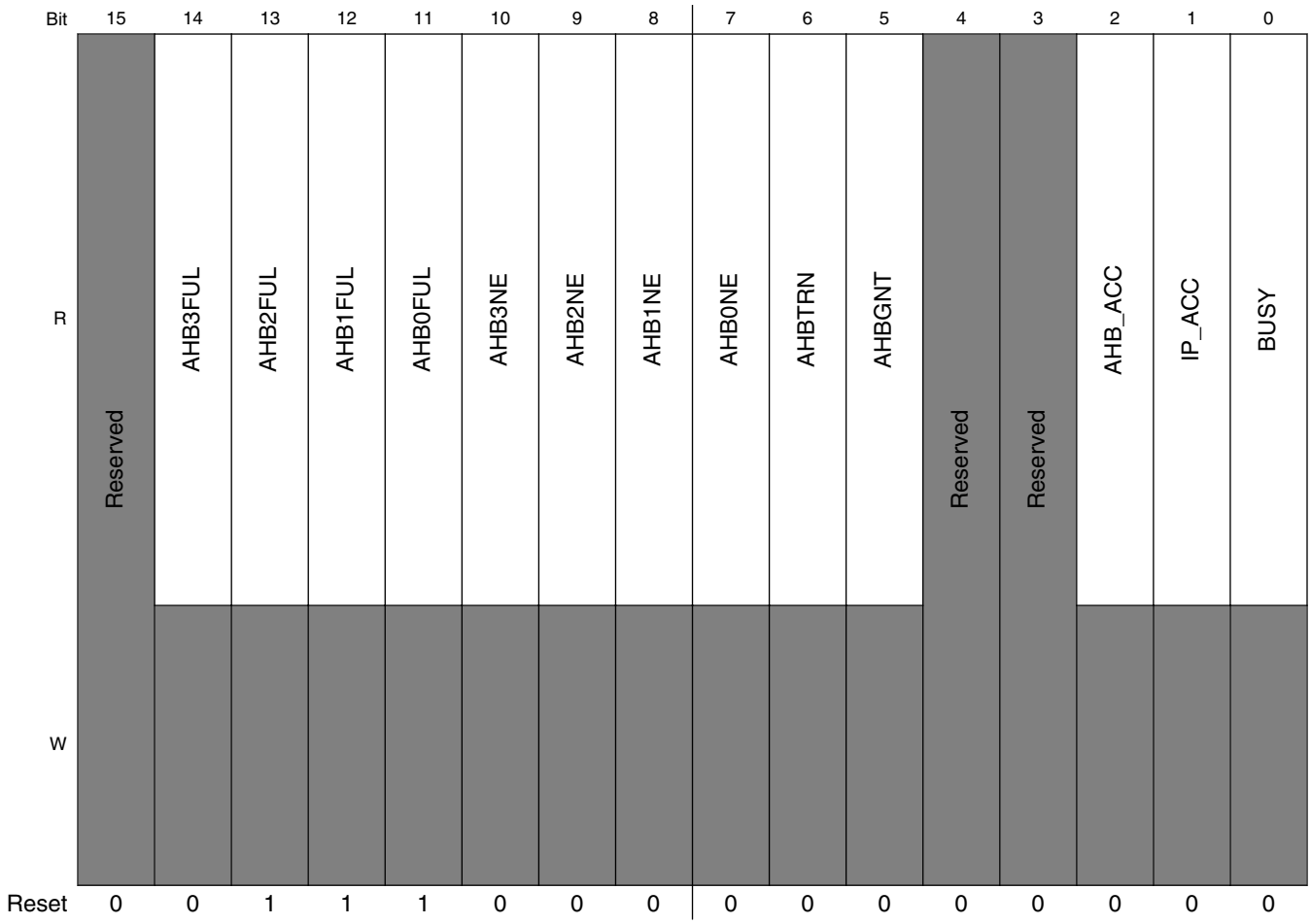
Field	Description
TXDATA	TX Data On write access the data is written into the next available entry of the TX Buffer and the QPSI_TBSR[TRBFL] field is updated accordingly. On a read access, the last data written to the register is returned.

10.2.14.18 Status Register (QuadSPIx_SR)

The QSPI_SR register provides all available status information about SFM command execution and arbitration, the RX Buffer and TX Buffer and the AHB Buffer.

Address: Base address + 15Ch offset





QuadSPIx_SR field descriptions

Field	Description
31–29 DLPSMP	**NOTE:** Data learning is not implemented on this chip. Data learning pattern sampling point: The sampling point found by the controller with the data learning pattern. • This is used for DDR only. • If the learning fails, this field will return garbage and DLPFF bit will be set. •
28 Reserved	This field is reserved.
27 TXFULL	TX Buffer Full: Asserted when no more data can be stored.
26 Reserved	This field is reserved.
25 Reserved	This field is reserved.
24 TXEDA	Tx Buffer Enough Data Available Asserted when TX Buffer contains enough data for any pop operation to take place. There must be atleast 128bit data available in TX FIFO for any pop operation otherwise QSPI_FR[TBUF] will be set.

Table continues on the next page...

QuadSPIx_SR field descriptions (continued)

Field	Description
23 RXDMA	RX Buffer DMA: Asserted when RX Buffer read out via DMA is active i.e DMA is requested or running.
22–20 Reserved	This field is reserved.
19 RXFULL	RX Buffer Full: Asserted when the RX Buffer is full, i.e. that QSPI_RBSR[RDBFL] field is equal to 32.
18–17 Reserved	This field is reserved.
16 RXWE	RX Buffer Watermark Exceeded: Asserted when the number of valid entries in the RX Buffer exceeds the number given in the QSPI_RBCT[WMRK] field.
15 Reserved	This field is reserved.
14 AHB3FUL	AHB 3 Buffer Full: Asserted when AHB 3 buffer is full.
13 AHB2FUL	AHB 2 Buffer Full: Asserted when AHB 2 buffer is full.
12 AHB1FUL	AHB 1 Buffer Full: Asserted when AHB 1 buffer is full.
11 AHB0FUL	AHB 0 Buffer Full: Asserted when AHB 0 buffer is full.
10 AHB3NE	AHB 3 Buffer Not Empty: Asserted when AHB 3 buffer contains data.
9 AHB2NE	AHB 2 Buffer Not Empty: Asserted when AHB 2 buffer contains data.
8 AHB1NE	AHB 1 Buffer Not Empty: Asserted when AHB 1 buffer contains data.
7 AHB0NE	AHB 0 Buffer Not Empty: Asserted when AHB 0 buffer contains data.
6 AHBTRN	AHB Access Transaction pending: Asserted when there is a pending request on the AHB interface. Refer to the AMBA specification for details.
5 AHBGNT	AHB Command priority Granted: Asserted when another module has been granted priority of AHB Commands against IP Commands. For details refer to Command Arbitration.
4 Reserved	This field is reserved.
3 RESERVED	This field is reserved.
2 AHB_ACC	AHB Access: Asserted when the transaction currently executed was initiated by AHB bus.
1 IP_ACC	IP Access: Asserted when transaction currently executed was initiated by IP bus.
0 BUSY	Module Busy: Asserted when module is currently busy handling a transaction to an external flash device.

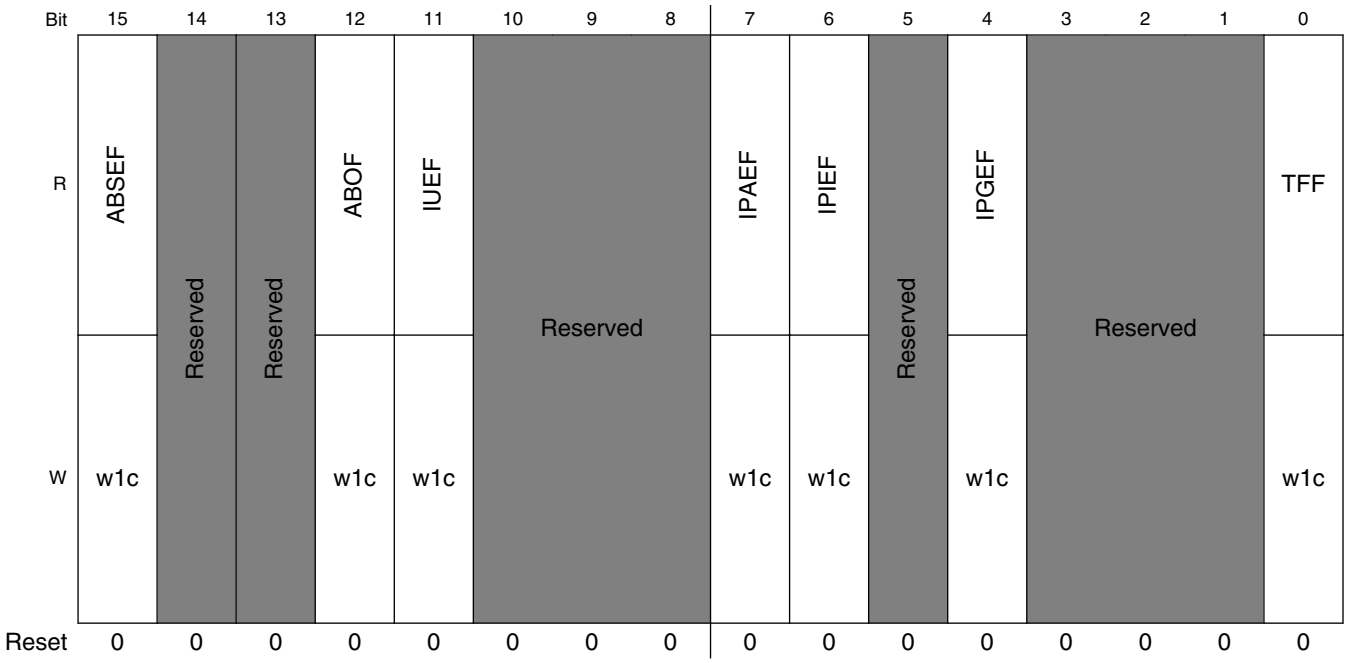
10.2.14.19 Flag Register (QuadSPIx_FR)

The QSPI_FR register provides all available flags about SFM command execution and arbitration which may serve as source for the generation of interrupt service requests. Note that the error flags in this register do not relate directly to the execution of the transaction in the serial flash device itself but only to the behavior and conditions visible in the QuadSPI module.

Write:Enabled Mode

Address: Base address + 160h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DLPFF	Reserved	Reserved		TBFF	TBUF	Reserved		ILLINE	Reserved					RBOF	RBDF
W	w1c				w1c	w1c			w1c						w1c	w1c
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0



QuadSPIx_FR field descriptions

Field	Description
31 DLPFF	**NOTE:** Data learning is not implemented on this chip. Data Learning Pattern Failure Flag: Set when DATA_LEARN instruction was encountered in a sequence but no sampling point was found for the data learning pattern The controller automatically starts sampling using the value in QSPI_SMPR[DDRSMP].
30 Reserved	This field is reserved.
29–28 Reserved	This field is reserved.
27 TBFF	TX Buffer Fill Flag: Before writing to the TX buffer, this bit should be cleared. Then this bit has to be read back. If the bit is set, the TX Buffer can take more data. If the bit remains cleared, the TX buffer is full. Refer to Tx Buffer Operation for details.
26 TBUF	TX Buffer Underrun Flag: Set when the module tried to pull data although TX Buffer was emptyor the buffer contains less than 128bits of data. The application must ensure that the buffer never goes empty during a transaction expect for the last data fetch. The IP Command leading to the TX Buffer underrun is continued (data sent to the serial flash device is all F in case of valid tx underrun. The application must clear the TX Buffer in response to this event by writing a 1 into the QSPI_MCR[CLR_TXF] bit.
25–24 Reserved	This field is reserved.
23 ILLINE	Illegal Instruction Error Flag: Set when an illegal instruction is encountered by the controller in any of the sequences. Refer to Table 10-14 for a list of legal instructions.
22–18 Reserved	This field is reserved.
17 RBOF	RX Buffer Overflow Flag: Set when not all the data read from the serial flash device could be pushed into the RX Buffer. The IP Command leading to this condition is continued until the number of bytes according to the QSPI_IPCR[IDATSZ] field has been read from the serial flash device.

Table continues on the next page...

QuadSPIx_FR field descriptions (continued)

Field	Description
	The content of the RX Buffer is not changed.
16 RBDF	RX Buffer Drain Flag: Will be set if the QuadSPI_SR[RXWE] status bit is asserted. Writing 1 into this bit triggers one of the following actions: • If the RX Buffer has up to QuadSPI_RBCT[WMRK] valid entries then the flag is cleared. • If the RX Buffer has more than QuadSPI_RBCT[WMRK] valid entries and the QuadSPI_RSER[RBDDE] bit is not set (flag driven mode) a RX Buffer POP event is triggered. The flag remains set if the RX Buffer contains more than QuadSPI_RBCT[WMRK] valid entries after the RX Buffer POP event is finished. The flag is cleared if the RX Buffer contains less than or equal to QuadSPI_RBCT[WMRK] valid entries after the RX Buffer POP event is finished. Refer to "Receive Buffer Drain Interrupt or DMA Request" section in Normal Mode Interrupt and DMA Requests, for details.
15 ABSEF	AHB Sequence Error Flag: Set when the execution of an AHB Command is started with an WRITE or WRITE_DDR Command in the sequence pointed to by the QSPI_BUFxCR register Communication with the serial flash device is terminated before the execution of WRITE/WRITE_DDR command by the QuadSPI module. The AHB bus request which triggered this command is answered with an ERROR response.
14 Reserved	Reserved This field is reserved.
13 Reserved	Reserved This field is reserved.
12 ABOF	AHB Buffer Overflow Flag: Set when the size of the AHB access exceeds the size of the AHB buffer. This condition can occur only if the QSPI_BUFxCR[ADATSZ] field is programmed incorrectly. The AHB Command leading to this condition is continued until the number of entries according to the QSPI_BUFxCR[ADATSZ] field has been read from the serial flash device. The content of the AHB Buffer is not changed.
11 IUEF	IP Command Usage Error Flag: Set when in parallel flash mode the execution of an IP Command is started and the sequence pointed to by the sequence ID contains a WRITE or a WRITE_DDR command. Refer to Table 10-14 table for the related commands. Communication with the serial flash device is terminated before the execution of WRITE/WRITE_DDR command by the QuadSPI module.
10–8 Reserved	This field is reserved.
7 IPAEF	IP Command Trigger during AHB Access Error Flag. Set when the following condition occurs: • A write access occurs to the QSPI_IPCR[SEQID] field and the QSPI_SR[AHB_ACC] bit is set. Any command leading to the assertion of the IPAEF flag is ignored.
6 IPIEF	IP Command Trigger could not be executed Error Flag. Set when the QSPI_SR[IP_ACC] bit is set (i.e. an IP triggered command is currently executing) and any of the following conditions occurs: • Write access to the QSPI_IPCR register. Any command leading to the assertion of the IPIEF flag is ignored • Write access to the QSPI_SFAR register. • Write access to the QSPI_RBCT register.
5 Reserved	This field is reserved.

Table continues on the next page...

QuadSPIx_FR field descriptions (continued)

Field	Description
4 IPGEF	IP Command Trigger during AHB Grant Error Flag: Set when the following condition occurs: • A write access occurs to the QSPI_IPCR[SEQID] field and the QSPI_SR[AHBGNT] bit is set. Any command leading to the assertion of the IPGEF flag is ignored.
3–1 Reserved	This field is reserved.
0 TFF	IP Command Transaction Finished Flag: Set when the QuadSPI module has finished a running IP Command. If an error occurred the related error flags are valid, at the latest, in the same clock cycle when the TFF flag is asserted.

1. QSPI_BUFxCR implies anyone of QSPI_BUF0CR/QSPI_BUF1CR/QSPI_BUF2CR/QSPI_BUF3CR

10.2.14.20 Interrupt and DMA Request Select and Enable Register (QuadSPIx_RSER)

The QuadSPI_RSER register provides enables and selectors for the interrupts in the QuadSPI module.

NOTE

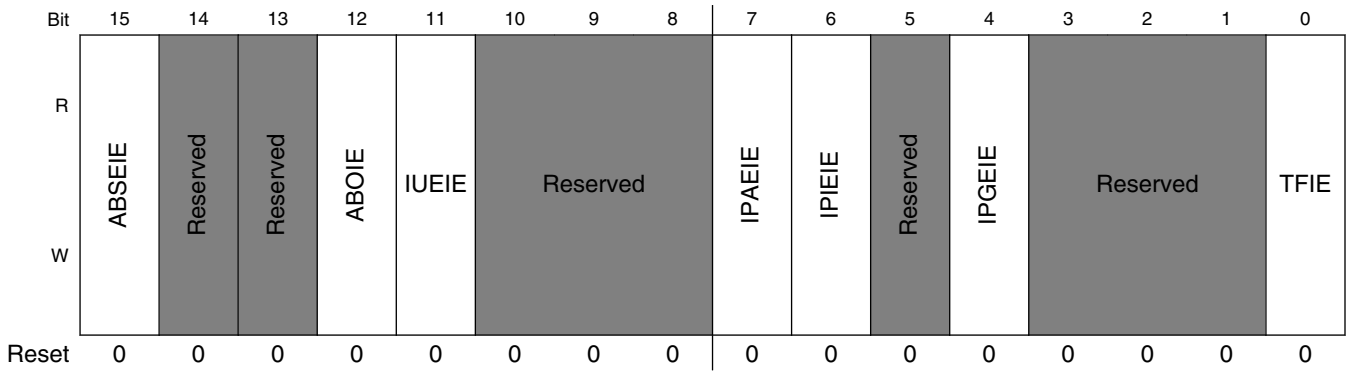
Each flag of the QuadSPI_FR register enabled as source for an interrupt prevents the QuadSPI module from entering Stop Mode or Module Disable Mode when this flag is set.

>

Write:Anytime

Address: Base address + 164h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DLPFIE	Reserved	Reserved		TBFIE	TBUIE	Reserved	Reserved	ILLINIE	Reserved	RBDDE	Reserved			RBOIE	RBDIE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



QuadSPIx_RSER field descriptions

Field	Description
31 DLPFIE	**NOTE:** Data learning is not implemented on this chip. Data Learning Pattern Failure Interrupt enable . Triggered by DLPFF flag in QSPI_FR register 0 No DLPFF interrupt will be generated 1 DLPFF interrupt will be generated
30 Reserved	This field is reserved.
29–28 RESERVED	This field is reserved.
27 TBFIE	TX Buffer Fill Interrupt Enable 0 No TBFF interrupt will be generated 1 TBFF interrupt will be generated
26 TBUIE	TX Buffer Underrun Interrupt Enable 0 No TBUF interrupt will be generated 1 TBUF interrupt will be generated
25 Reserved	This field is reserved.
24 Reserved	This field is reserved.
23 ILLINIE	Illegal Instruction Error Interrupt Enable. Triggered by ILLINE flag in QSPI_FR 0 No ILLINE interrupt will be generated 1 ILLINE interrupt will be generated
22 Reserved	This field is reserved.
21 RBDDE	RX Buffer Drain DMA Enable: Enables generation of DMA requests for RX Buffer Drain. When this bit is set DMA requests are generated as long as the QSPI_SR[RXWE] status bit is set. 0 No DMA request will be generated 1 DMA request will be generated
20–18 Reserved	This field is reserved.

Table continues on the next page...

QuadSPIx_RSER field descriptions (continued)

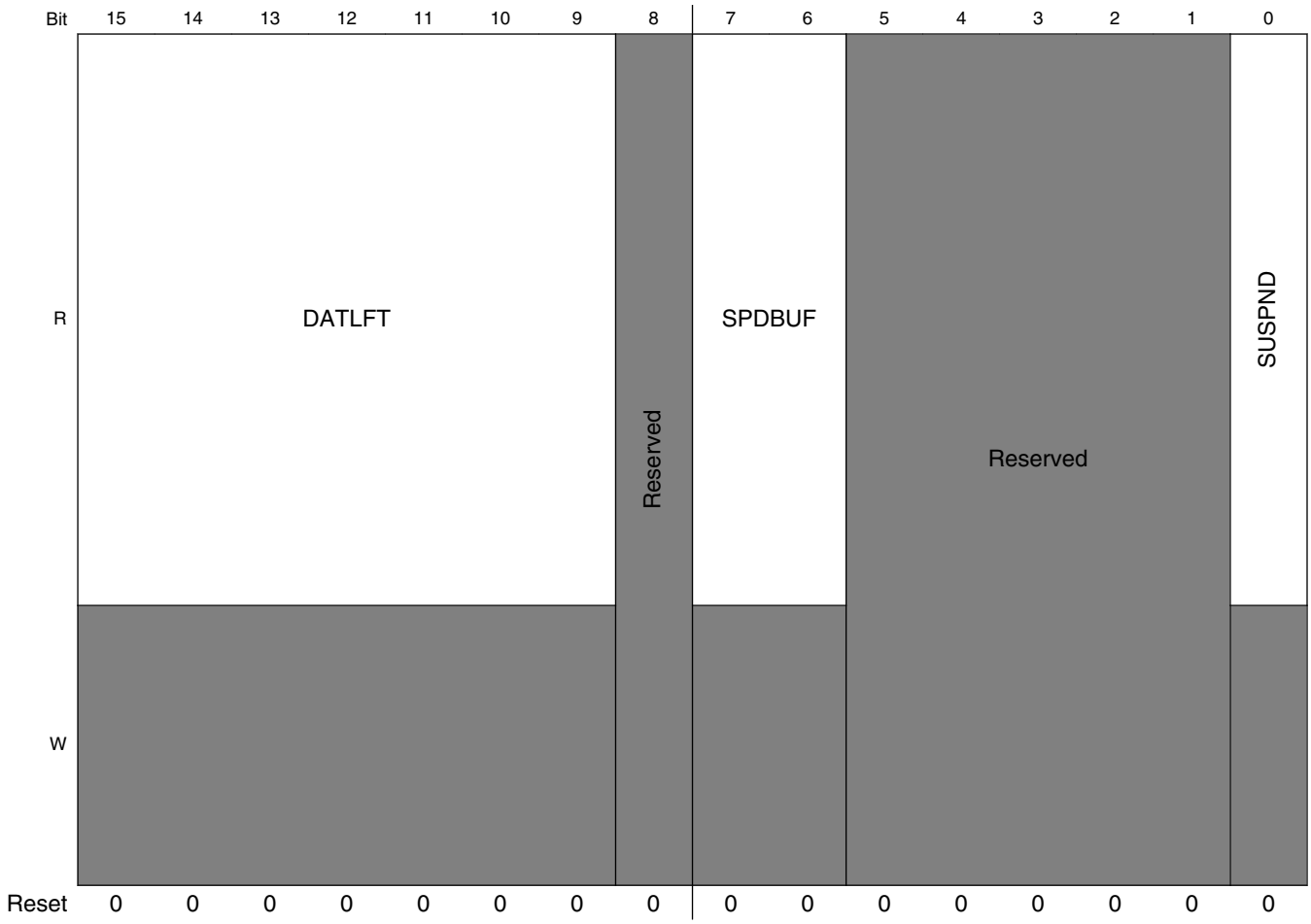
Field	Description
17 RBOIE	RX Buffer Overflow Interrupt Enable 0 No RBOF interrupt will be generated 1 RBOF interrupt will be generated
16 RBDIE	RX Buffer Drain Interrupt Enable: Enables generation of IRQ requests for RX Buffer Drain. When this bit is set the interrupt is asserted as long as the QuadSPI_SR[RBDF] flag is set. 0 No RBDF interrupt will be generated 1 RBDF Interrupt will be generated
15 ABSEIE	AHB Sequence Error Interrupt Enable: Triggered by ABSEF flags of QSPI_FR 0 No ABSEF interrupt will be generated 1 ABSEF interrupt will be generated
14 Reserved	Reserved This field is reserved.
13 Reserved	Reserved This field is reserved.
12 ABOIE	AHB Buffer Overflow Interrupt Enable 0 No ABOF interrupt will be generated 1 ABOF interrupt will be generated
11 IUEIE	IP Command Usage Error Interrupt Enable 0 No IUEF interrupt will be generated 1 IUEF interrupt will be generated
10–8 Reserved	This field is reserved.
7 IPAEIE	IP Command Trigger during AHB Access Error Interrupt Enable 0 No IPAEF interrupt will be generated 1 IPAEF interrupt will be generated
6 IPIEIE	IP Command Trigger during IP Access Error Interrupt Enable 0 No IPIEF interrupt will be generated 0 IPIEF interrupt will be generated
5 Reserved	This field is reserved.
4 IPGEIE	IP Command Trigger during AHB Grant Error Interrupt Enable 0 No IPGEF interrupt will be generated 1 IPGEF interrupt will be generated
3–1 Reserved	This field is reserved. Reserved.
0 TFIE	Transaction Finished Interrupt Enable 0 No TFF interrupt will be generated 1 TFF interrupt will be generated

10.2.14.21 Sequence Suspend Status Register (QuadSPIx_SPNDST)

The sequence suspend status register provides information specific to any suspended sequence. An AHB sequence may be suspended when a high priority AHB master makes an access before the AHB sequence completes the data transfer requested.

Address: Base address + 168h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



QuadSPIx_SPNDST field descriptions

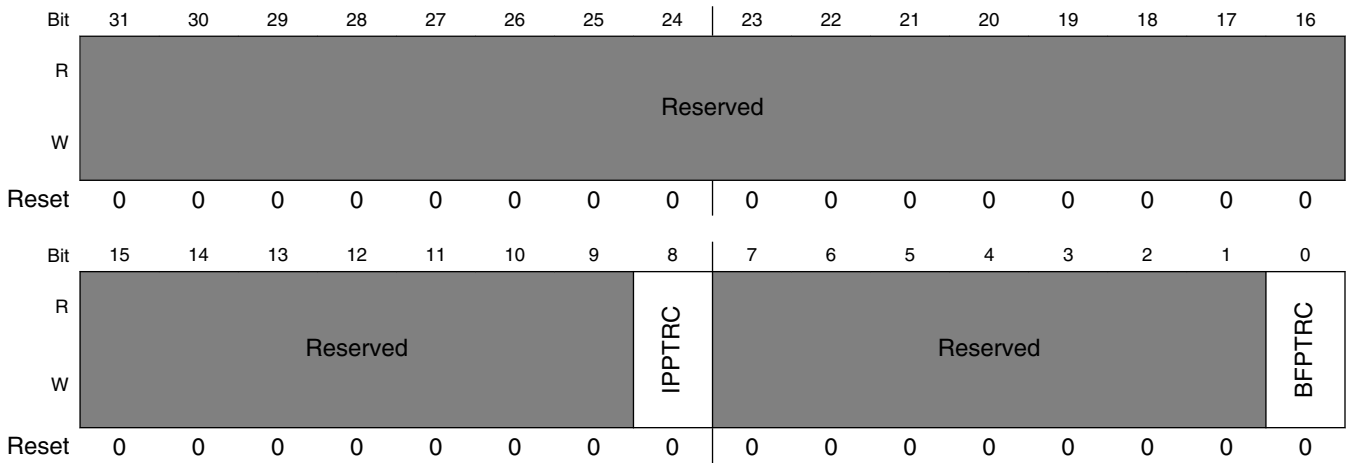
Field	Description
31–16 Reserved	This field is reserved.
15–9 DATLFT	Data left: Provides information about the amount of data left to be read in the suspended sequence. Valid only when SUSPND is set to 1'b1. Value in terms of 64 bits or 8 bytes
8 Reserved	This field is reserved.
7–6 SPDBUF	Suspended Buffer: Provides the suspended buffer number. Valid only when SUSPND is set to 1'b1
5–1 Reserved	This field is reserved.
0 SUSPND	When set, it signifies that a sequence is in suspended state

10.2.14.22 Sequence Pointer Clear Register (QuadSPIx_SPTRCLR)

The sequence pointer clear register provides bits to reset the IP and Buffer sequence pointers. The sequence pointer contains the index of which instruction within the LUT entry is to be executed next. For example, if the LUT entry ends on a JMP_ON_CS value of 2, the index will be stored as 2.

The software should reset the sequence pointers whenever the sequence ID is changed by updating the SEQID field in QSPI_IPCR or QSPI_BFGENCR.

Address: Base address + 16Ch offset



QuadSPIx_SPTRCLR field descriptions

Field	Description
31–9 Reserved	This field is reserved.
8 IPPTRC	IP Pointer Clear: 1: Clears the sequence pointer for IP accesses as defined in QuadSPI_IPCR
7–1 Reserved	This field is reserved. Reserved.
0 BFPTRC	Buffer Pointer Clear: 1: Clears the sequence pointer for AHB accesses as defined in QuadSPI_BFGENCR.

10.2.14.23 Serial Flash A1 Top Address (QuadSPIx_SFA1AD)

The QSPI_SFA1AD register provides the address mapping for the serial flash A1.The difference between QSPI_SFA1AD[TPADA1] and QSPI_AMBA_BASE defines the size of the memory map for serial flash A1.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: Base address + 180h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TPADA1																Reserved																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_SFA1AD field descriptions

Field	Description
31–10 TPADA1	Top address for Serial Flash A1. In effect, TPADxx is the first location of the next memory.
Reserved	This field is reserved.

10.2.14.24 Serial Flash A2 Top Address (QuadSPIx_SFA2AD)

The QSPI_SFA2AD register provides the address mapping for the serial flash A2.The difference between QSPI_SFA2AD[TPADA2] and QSPI_SFA1AD[TPADA1] defines the size of the memory map for serial flash A2.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: Base address + 184h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TPADA2																Reserved																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_SFA2AD field descriptions

Field	Description
31–10 TPADA2	Top address for Serial Flash A2. In effect, TPxxAD is the first location of the next memory.
Reserved	This field is reserved.

10.2.14.25 Serial Flash B1Top Address (QuadSPIx_SFB1AD)

The QSPI_SFB1AD register provides the address mapping for the serial flash B1.The difference between QSPI_SFB1AD[TPADB1] and QSPI_SFA2AD[TPADA2] defines the size of the memory map for serial flash B1.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: Base address + 188h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TPADB1																						Reserved									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_SFB1AD field descriptions

Field	Description
31–10 TPADB1	Top address for Serial Flash B1.In effect, TPxxAD is the first location of the next memory.
Reserved	This field is reserved.

10.2.14.26 Serial Flash B2Top Address (QuadSPIx_SFB2AD)

The QSPI_SFB2AD register provides the address mapping for the serial flash B2.The difference between QSPI_SFB2AD[TPADB2] and QSPI_SFB1AD[TPADB1] defines the size of the memory map for serial flash B2.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: Base address + 18Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TPADB2																						Reserved									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_SFB2AD field descriptions

Field	Description
31–10 TPADB2	Top address for Serial Flash B2. In effect, TPxxAD is the first location of the next memory.
Reserved	This field is reserved.

10.2.14.27 RX Buffer Data Register (QuadSPIx_RBDR*n*)

The QuadSPI_RBDR registers provide access to the individual entries in the RX Buffer. Refer to Table 10-16 for the byte ordering scheme.

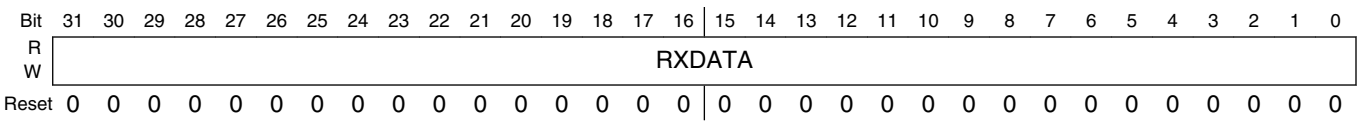
QuadSPI_RBDR0 corresponds to the actual position of the read pointer within the RX Buffer. The number of valid entries available depends from the number of RX Buffer entries implemented and from the number of valid buffer entries available in the RX Buffer.

Example 1, RX Buffer filled completely with 32 words: In this case the address range for valid read access extends from QuadSPI_RBDR0 to QuadSPI_RBDR31.

Example 2, RX Buffer filled with 5 valid words: RX Buffer fill level QuadSPI_RBSR[RDBFL] is 5. In this case an access to QuadSPI_RBDR4 provides the last valid entry.

Any access beyond the range of valid RX Buffer entries provides undefined results.

Address: Base address + 200h offset + (4d × i), where i=0d to 31d



QuadSPIx_RBDR*n* field descriptions

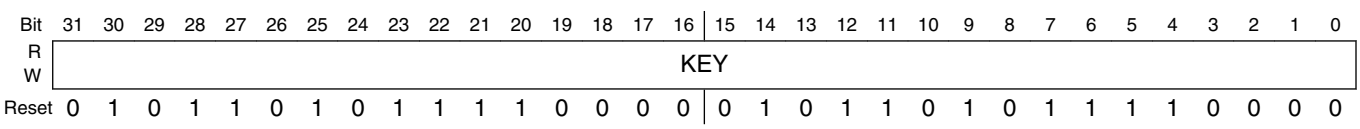
Field	Description
RXDATA	RX Data. The RXDATA field contains the data associated with the related RX Buffer entry. Data format and byte ordering is given in Byte Ordering of Serial Flash Read Data .

10.2.14.28 LUT Key Register (QuadSPIx_LUTKEY)

The LUT Key register contains the key to lock and unlock the Look-up-table. Refer to Look-up Table for details.

Write:Anytime

Address: Base address + 300h offset



QuadSPIx_LUTKEY field descriptions

Field	Description
KEY	The key to lock or unlock the LUT. The KEY is 0x5AF05AF0. The read value is always 0x5AF05AF0

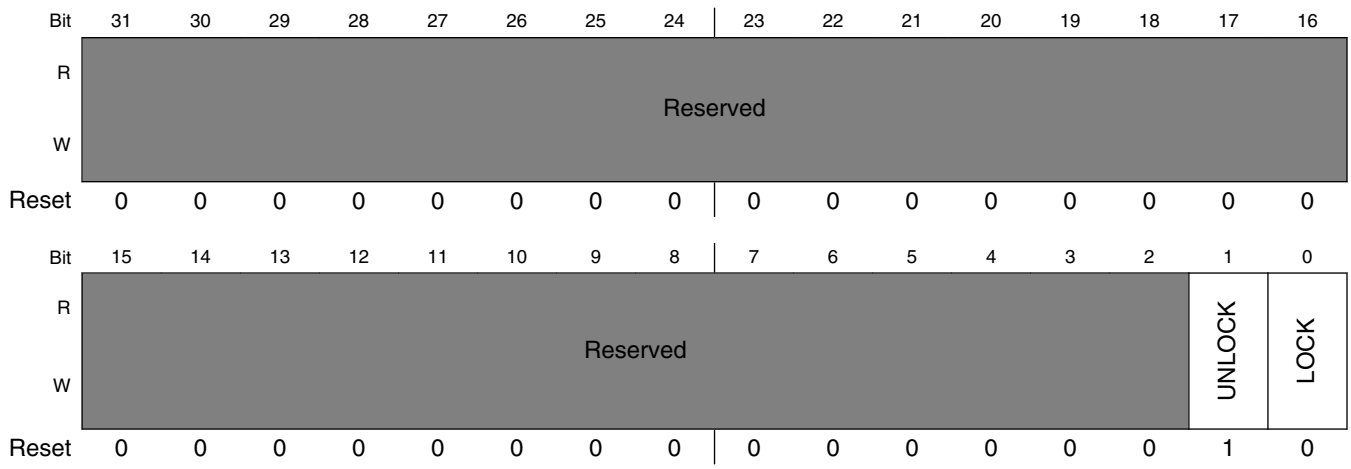
10.2.14.29 LUT Lock Configuration Register (QuadSPIx_LCKCR)

The LUT lock configuration register is used along with QSPI_LUTKEY register to lock or unlock the LUT. This register has to be written immediately after QSPI_LUTKEY register for the lock or unlock operation to be successful. Refer to Look-up Table for details. Setting both the LOCK and UNLOCK bits as "00" or "11" is not allowed.

Write:Just after writing the LUT Key Register

(QSPI_LUTKEY)

Address: Base address + 304h offset



QuadSPIx_LCKCR field descriptions

Field	Description
31–2 Reserved	This field is reserved.
1 UNLOCK	Unlocks the LUT when the following two conditions are met: 1. This register is written just after the LUT Key Register (QuadSPI_LUTKEY) 2. The LUT key register was written with 0x5AF05AF0 key
0 LOCK	Locks the LUT when the following condition is met: 1. This register is written just after the LUT Key Register (QuadSPI_LUTKEY) 2. The LUT key register was written with 0x5AF05AF0 key

10.2.14.30 Look-up Table register (QuadSPIx_LUT0)

The LUT registers are a look-up-table for sequences of instructions. The programmable sequence engine executes the instructions in these sequences to generate a valid serial flash transaction. There are a total of 64 LUT registers. These 64 registers are divided into groups of 4 registers that make a valid sequence. Therefore, QSPI_LUT[0], QSPI_LUT[4], QSPI_LUT[8] ..... QSPI_LUT[60] are the starting registers of a valid sequence. Each of these sets of 4 registers can have a maximum of 8 instructions. A maximum of 16 sequences can be defined at one time. Look-up Table describes the LUT registers in detail.

Write:Once the LUT is unlocked

Address: Base address + 310h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INSTR1						PAD1		OPRND1							
Reset	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INSTR0						PAD0		OPRND0							
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1

QuadSPIx_LUT0 field descriptions

Field	Description
31–26 INSTR1	Instruction 1
25–24 PAD1	Pad information for INSTR1. 00 1 Pad 01 2 Pads 10 4 Pads 11 NA
23–16 OPRND1	Operand for INSTR1.
15–10 INSTR0	Instruction 0
9–8 PAD0	Pad information for INSTR0. 00 1 Pad 01 2 Pads 10 4 Pads 11 NA
OPRND0	Operand for INSTR0.

10.2.14.31 Look-up Table register (QuadSPIx_LUT1)

The LUT registers are a look-up-table for sequences of instructions. The programmable sequence engine executes the instructions in these sequences to generate a valid serial flash transaction. There are a total of 64 LUT registers. These 64 registers are divided into groups of 4 registers that make a valid sequence. Therefore, QSPI_LUT[0], QSPI_LUT[4], QSPI_LUT[8] ..... QSPI_LUT[60] are the starting registers of a valid sequence. Each of these sets of 4 registers can have a maximum of 8 instructions. A maximum of 16 sequences can be defined at one time. Look-up Table describes the LUT registers in detail.

Write:Once the LUT is unlocked

Address: Base address + 314h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INSTR1						PAD1		OPRND1							
Reset	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INSTR0						PAD0		OPRND0							
Reset	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0

QuadSPIx_LUT1 field descriptions

Field	Description
31–26 INSTR1	Instruction 1
25–24 PAD1	Pad information for INSTR1. 00 1 Pad 01 2 Pads 10 4 Pads 11 NA
23–16 OPRND1	Operand for INSTR1.
15–10 INSTR0	Instruction 0
9–8 PAD0	Pad information for INSTR0. 00 1 Pad 01 2 Pads 10 4 Pads 11 NA
OPRND0	Operand for INSTR0.

10.2.14.32 Look-up Table register (QuadSPIx_LUT*n*)

The LUT registers are a look-up-table for sequences of instructions. The programmable sequence engine executes the instructions in these sequences to generate a valid serial flash transaction. There are a total of 64 LUT registers. These 64 registers are divided into groups of 4 registers that make a valid sequence. Therefore, QSPI_LUT[0], QSPI_LUT[4], QSPI_LUT[8] ..... QSPI_LUT[60] are the starting registers of a valid sequence. Each of these sets of 4 registers can have a maximum of 8 instructions. A maximum of 16 sequences can be defined at one time. Look-up Table describes the LUT registers in detail.

Write:Once the LUT is unlocked

Address: Base address + 318h offset + (4d × i), where i=0d to 61d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INSTR1						PAD1		OPRND1							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INSTR0						PAD0		OPRND0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_LUT*n* field descriptions

Field	Description
31–26 INSTR1	Instruction 1
25–24 PAD1	Pad information for INSTR1. 00 1 Pad 01 2 Pads 10 4 Pads 11 NA
23–16 OPRND1	Operand for INSTR1.
15–10 INSTR0	Instruction 0
9–8 PAD0	Pad information for INSTR0. 00 1 Pad 01 2 Pads 10 4 Pads 11 NA
OPRND0	Operand for INSTR0.

10.3 Ultra Secured Digital Host Controller (uSDHC)

10.3.1 Overview

The Ultra Secured Digital Host Controller (uSDHC) provides the interface between the host system and the SD/SDIO/MMC cards, as depicted in Figure 10-35.

The uSDHC acts as a bridge, passing host bus transactions to the SD/SDIO/MMC cards by sending commands and performing data accesses to/from the cards.

It handles the SD/SDIO/MMC protocols at the transmission level.

The following are brief descriptions of the cards supported by the uSDHC:

The Multi Media Card (MMC) is a universal low cost data storage and communication media designed to cover a wide array of applications including mobile video and gaming. Previous MMC cards were based on a 7-pin serial bus with a single data pin, while the new high speed MMC communication is based on an advanced 11-pin serial bus designed to operate in the low voltage range.

The Secure Digital Card (SD) is an evolution of the old MMC technology. It is specifically designed to meet the security, capacity, performance, and environment requirements inherent in newly-emerging audio and video consumer electronic devices. The physical form factor, pin assignment and data transfer protocol are forward-compatible with the old MMC (with some additions).

Under the SD protocol, it can be categorized into Memory card, I/O card and Combo card, which has both memory and I/O functions. The memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard. The I/O card, which is also known as SDIO card, provides high-speed data I/O with low power consumption for mobile electronic devices. For the sake of simplicity, the following figure does not show cards with reduced size or mini cards.

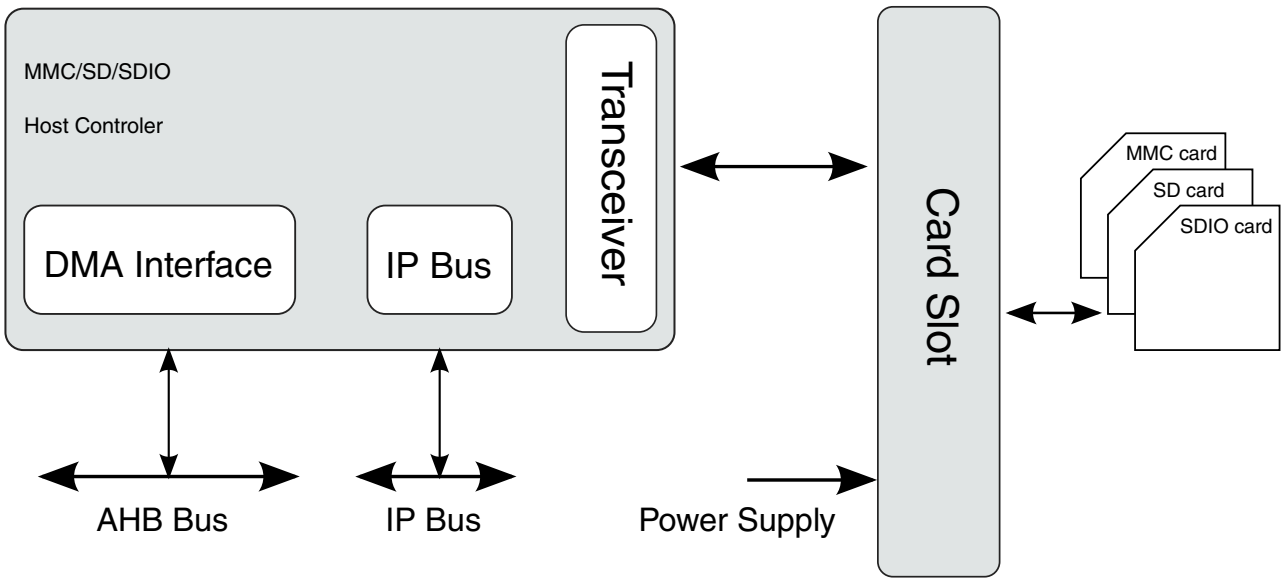


Figure 10-35. System Connection of the uSDHC

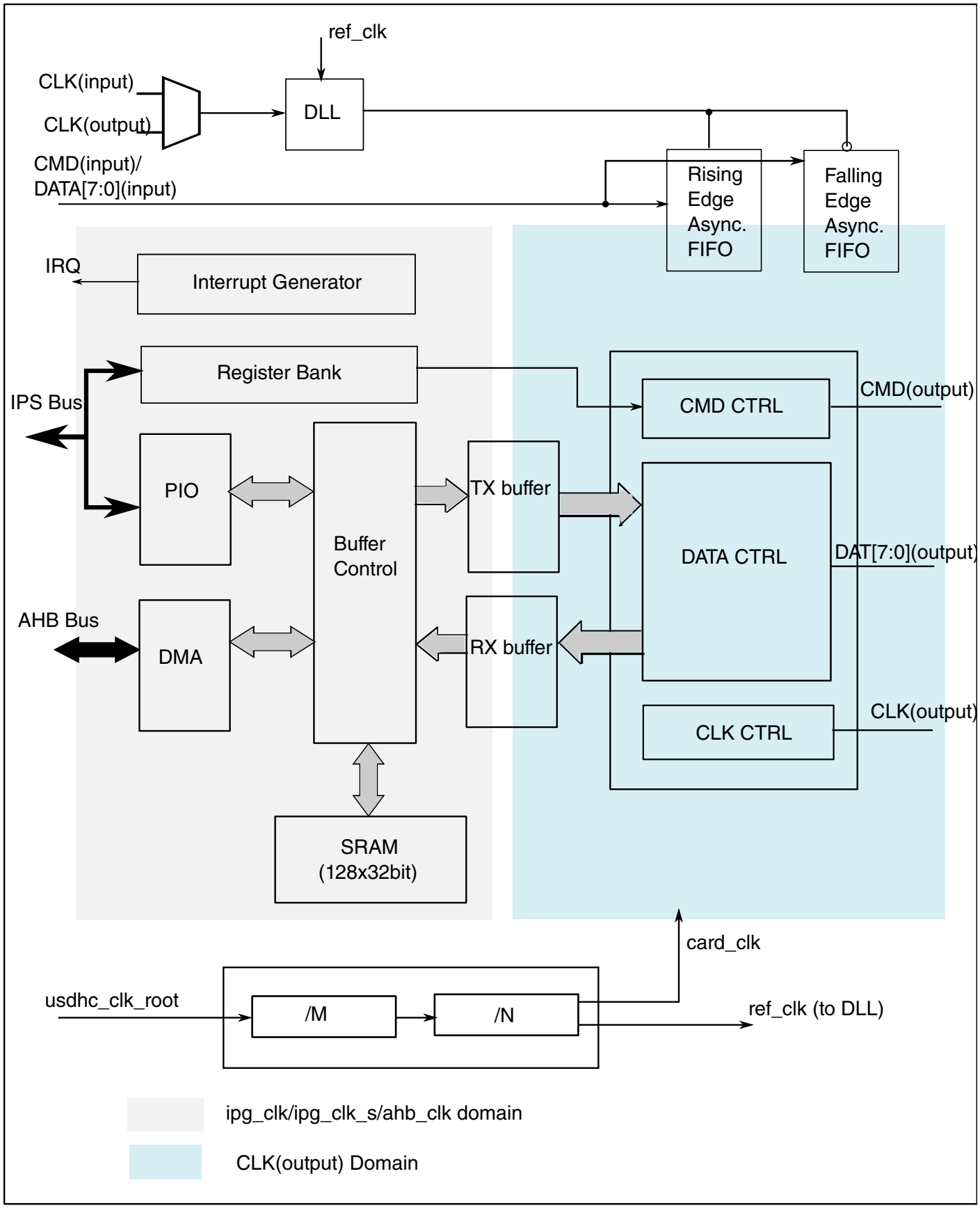


Figure 10-36. ultra Secure Digital Host Controller Block Diagram

10.3.1.1 Features

The features of the uSDHC module include the following:

- Conforms to the SD Host Controller Standard Specification version 3.0
- Compatible with the MMC System Specification version 4.2/4.3/4.4/4.41/4.5/5.0
- Compatible with the SD Memory Card Specification version 3.0 and supports the Extended Capacity SD Memory Card
- Compatible with the SDIO Card Specification version 3.0
- Designed to work with SD Memory, miniSD Memory, SDIO, miniSDIO, SD Combo, MMC, MMC plus, and MMC RS cards
- Card bus clock frequency up to 208 MHz
- Supports 1-bit / 4-bit SD and SDIO modes, 1-bit / 4-bit / 8-bit MMC modes
    - Up to 832 Mbps of data transfer for SDIO cards using 4 parallel data lines in SDR(Single Data Rate) mode
    - Up to 400 Mbps of data transfer for SDIO card using 4 parallel data lines in DDR(Dual Data Rate) mode
    - Up to 832 Mbps of data transfer for SDXC cards using 4 parallel data lines in SDR(Single Data Rate) mode
    - Up to 400 Mbps of data transfer for SDXC card using 4 parallel data lines in DDR(Dual Data Rate) mode
    - Up to 1600 Mbps of data transfer for MMC cards using 8 parallel data lines in SDR(Single Data Rate) mode
    - Up to 3200 Mbps of data transfer for MMC cards using 8 parallel data lines in DDR(Dual Data Rate) mode
- Supports single block/multi-block read and write
- Supports block sizes of 1 ~ 4096 bytes
- Supports the write protection switch for write operations
- Supports both synchronous and asynchronous abort
- Supports pause during the data transfer at block gap
- Supports SDIO Read Wait and Suspend Resume operations
- Supports Auto CMD12 for multi-block transfer
- Host can initiate non-data transfer command while data transfer is in progress
- Allows cards to interrupt the host in 1-bit and 4-bit SDIO modes, also supports interrupt period
- Embodies a fully configurable 128x32-bit FIFO for read/write data
- Supports internal and external DMA capabilities
- Support voltage selection by configuring vendor specific register bit
- Supports Advanced DMA to perform linked memory access

10.3.1.2 Modes and Operations

10.3.1.2.1 Data transfer Modes

The uSDHC can select the following modes for data transfer:

- SD 1-bit
- SD 4-bit
- MMC 1-bit
- MMC 4-bit
- MMC 8-bit
- Identification Mode (up to 400 kHz)
- MMC full speed mode (up to 26 MHz)
- MMC high speed mode (up to 52 MHz)
- MMC HS400 mode (200 MHz both edges)
- MMC DDR mode (52 MHz both edges)
- SD/SDIO full speed mode (up to 25 MHz)
- SD/SDIO high speed mode (up to 50 MHz)
- SD/SDIO UHS-I mode(up to 208 MHz in SDR mode, up to 50 MHz in DDR mode)

10.3.2 External Signals

The following table describes the external signals of USDHC:

Table 10-40. USDHC External Signals

Signal	Description	Pad	Mode	Direction
SD1_CD_B	Card detection pin If not used(for the embedded memory),tie low to indicate there is a card attached.	SD1_CD_B	ALT0	I
SD1_CLK	Clock for MMC/SD/SDIO card	SD1_CLK	ALT0	O
SD1_CMD	CMD line connect to card	SD1_CMD	ALT0	IO
SD1_DATA0	DATA0 line in all modes Also used to detect busy state	SD1_DATA0	ALT0	IO
SD1_DATA1	DATA1 line in 4/8-bit mode Also used to detect interrupt in 1/4- bit mode	SD1_DATA1	ALT0	IO
SD1_DATA2	DATA2 line or Read Wait in 4-bit mode. Read Wait in 1-bit mode	SD1_DATA2	ALT0	IO
SD1_DATA3	DATA3 line in 4/8-bit mode or configured as card detection pin May be configured as card detection pin in 1-bit mode	SD1_DATA3	ALT0	IO

Table continues on the next page...

Table 10-40. USDHC External Signals (continued)

Signal	Description	Pad	Mode	Direction
SD1_DATA4	DATA4 line in 8-bit mode, not used in other modes	ECSPI2_SCLK	ALT2	IO
SD1_DATA5	DATA5 line in 8-bit mode, not used in other modes	ECSPI2_MOSI	ALT2	IO
SD1_DATA6	DATA6 line in 8-bit mode, not used in other modes	ECSPI2_MISO	ALT2	IO
SD1_DATA7	DATA7 line in 8-bit mode, not used in other modes	ECSPI2_SS0	ALT2	IO
SD1_LCTL	LED control used to drive an external LED Active high Fully controlled by the driver Optional output	GPIO1_IO09	ALT1	O
		UART3_RXD	ALT6	
SD1_RESET_B	Card hardware reset signal, active LOW	SD1_RESET_B	ALT0	O
SD1_VSELECT	IO power voltage selection signal	GPIO1_IO08	ALT1	O
		UART3_CTS	ALT6	
SD1_WP	Card write protect detect If not used(for the embedded memory), tie low to indicate it's not write protected.	SD1_WP	ALT0	I
SD2_CD_B	Card detection pin If not used(for the embedded memory),tie low to indicate there is a card attached.	SD2_CD_B	ALT0	I
SD2_CLK	Clock for MMC/SD/SDIO card	SD2_CLK	ALT0	O
SD2_CMD	CMD line connect to card	SD2_CMD	ALT0	IO
SD2_DATA0	DATA0 line in all modes Also used to detect busy state	SD2_DATA0	ALT0	IO
SD2_DATA1	DATA1 line in 4/8-bit mode Also used to detect interrupt in 1/4- bit mode	SD2_DATA1	ALT0	IO
SD2_DATA2	DATA2 line or Read Wait in 4-bit mode. Read Wait in 1-bit mode	SD2_DATA2	ALT0	IO
SD2_DATA3	DATA3 line in 4/8-bit mode or configured as card detection pin May be configured as card detection pin in 1-bit mode	SD2_DATA3	ALT0	IO
SD2_DATA4	DATA4 line in 8-bit mode, not used in other modes	ECSPI1_SCLK	ALT2	IO
SD2_DATA5	DATA5 line in 8-bit mode, not used in other modes	ECSPI1_MOSI	ALT2	IO
SD2_DATA6	DATA6 line in 8-bit mode, not used in other modes	ECSPI1_MISO	ALT2	IO
SD2_DATA7	DATA7 line in 8-bit mode, not used in other modes	ECSPI1_SS0	ALT2	IO
SD2_LCTL	LED control used to drive an external LED Active high Fully	GPIO1_IO10	ALT1	O
		UART3_TXD	ALT6	

Table continues on the next page...

Table 10-40. USDHC External Signals (continued)

Signal	Description	Pad	Mode	Direction
	controlled by the driver Optional output			
SD2_RESET_B	Card hardware reset signal, active LOW	SD2_RESET_B	ALT2	O
		SD2_RESET_B	ALT0	
SD2_VSELECT	IO power voltage selection signal	GPIO1_IO12	ALT1	O
		I2C1_SCL	ALT6	
SD2_WP	Card write protect detect If not used(for the embedded memory), tie low to indicate it's not write protected.	SD2_WP	ALT0	I
SD3_CD_B	Card detection pin If not used(for the embedded memory),tie low to indicate there is a card attached.	GPIO1_IO14	ALT1	I
		I2C2_SCL	ALT6	
		SD3_DATA7	ALT2	
SD3_CLK	Clock for MMC/SD/SDIO card	SD3_CLK	ALT0	O
SD3_CMD	CMD line connect to card	SD3_CMD	ALT0	IO
SD3_DATA0	DATA0 line in all modes Also used to detect busy state	SD3_DATA0	ALT0	IO
SD3_DATA1	DATA1 line in 4/8-bit mode Also used to detect interrupt in 1/4- bit mode	SD3_DATA1	ALT0	IO
SD3_DATA2	DATA2 line or Read Wait in 4-bit mode. Read Wait in 1-bit mode	SD3_DATA2	ALT0	IO
SD3_DATA3	DATA3 line in 4/8-bit mode or configured as card detection pin May be configured as card detection pin in 1-bit mode	SD3_DATA3	ALT0	IO
SD3_DATA4	DATA4 line in 8-bit mode, not used in other modes	SD3_DATA4	ALT0	IO
SD3_DATA5	DATA5 line in 8-bit mode, not used in other modes	SD3_DATA5	ALT0	IO
SD3_DATA6	DATA6 line in 8-bit mode, not used in other modes	SD3_DATA6	ALT0	IO
SD3_DATA7	DATA7 line in 8-bit mode, not used in other modes	SD3_DATA7	ALT0	IO
SD3_LCTL	LED control used to drive an external LED Active high Fully controlled by the driver Optional output	GPIO1_IO11	ALT1	O
		UART3_RTS	ALT6	
SD3_RESET_B	Card hardware reset signal, active LOW	SD3_RESET_B	ALT2	O
		SD3_RESET_B	ALT0	
SD3_STROBE	Strobe Signal	SD3_STROBE	ALT0	IO
SD3_VSELECT	IO power voltage selection signal	GPIO1_IO13	ALT1	O
		I2C1_SDA	ALT6	
SD3_WP	Card write protect detect If not used(for the embedded memory), tie	GPIO1_IO15	ALT1	I
		I2C2_SDA	ALT6	

Table continues on the next page...

Table 10-40. USDHC External Signals (continued)

Signal	Description	Pad	Mode	Direction
	low to indicate it's not write protected.	SD3_DATA6	ALT2	

10.3.2.1 Signals Overview

The uSDHC has 14 associated I/O signals.

- The CLK is an internally generated clock used to drive the MMC, SD, SDIO cards.
- The CMD I/O is used to send commands and receive responses to and from the card. Eight data lines (DAT7~DAT0) are used to perform data transfers between the uSDHC and the card.
- The CD and WP are card detection and write protection signals directly routed from the socket. These two signals are active low (0). A low on CD# means that a card is inserted, and a high on WP means that the write protect switch is active.
- LCTL is an output signal used to drive an external LED to indicate that the SD interface is busy.
- RST is an output signal used to reset the MMC card.
- VSELECT is an output signal used to change the voltage of the external power supplier.

CD, WP, LCTL, RST and VSELECT are all optional for system implementation. If the uSDHC needs to support a 4-bit data transfer, DAT7~DAT4 can also be optional and tied to high.

10.3.3 Clocks

The table found here describes the clock sources for uSDHC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 10-41. uSDHC Clocks

Clock name	Clock Root	Description
hclk	ahb_clk_root	AHB bus clock
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_perclk	usdhc_clk_root	Base clock
ipg_clk_s	ipg_clk_root	Peripheral access clock for register accesses

10.3.4 Functional Description

The following sections provide a brief functional description of the major system blocks, including the Data Buffer, DMA AHB interface, register bank as well as IP Bus interface, dual-port memory wrapper, data/command controller, clock & reset manager and clock generator.

10.3.4.1 Data Buffer

The uSDHC uses one configurable data buffer to transfer data between the system bus (IP Bus or AHB Bus) and the SD card in an optimized manner, maximizing throughput between the two clock domains (IP peripheral clock and the master clock).

The buffer is used as temporary storage for data being transferred between the host system and the card. The watermark levels for read and write are both configurable and can be from 1 to 128 words. The burst lengths for read and write are also configurable and can be from 1 to 31 words.

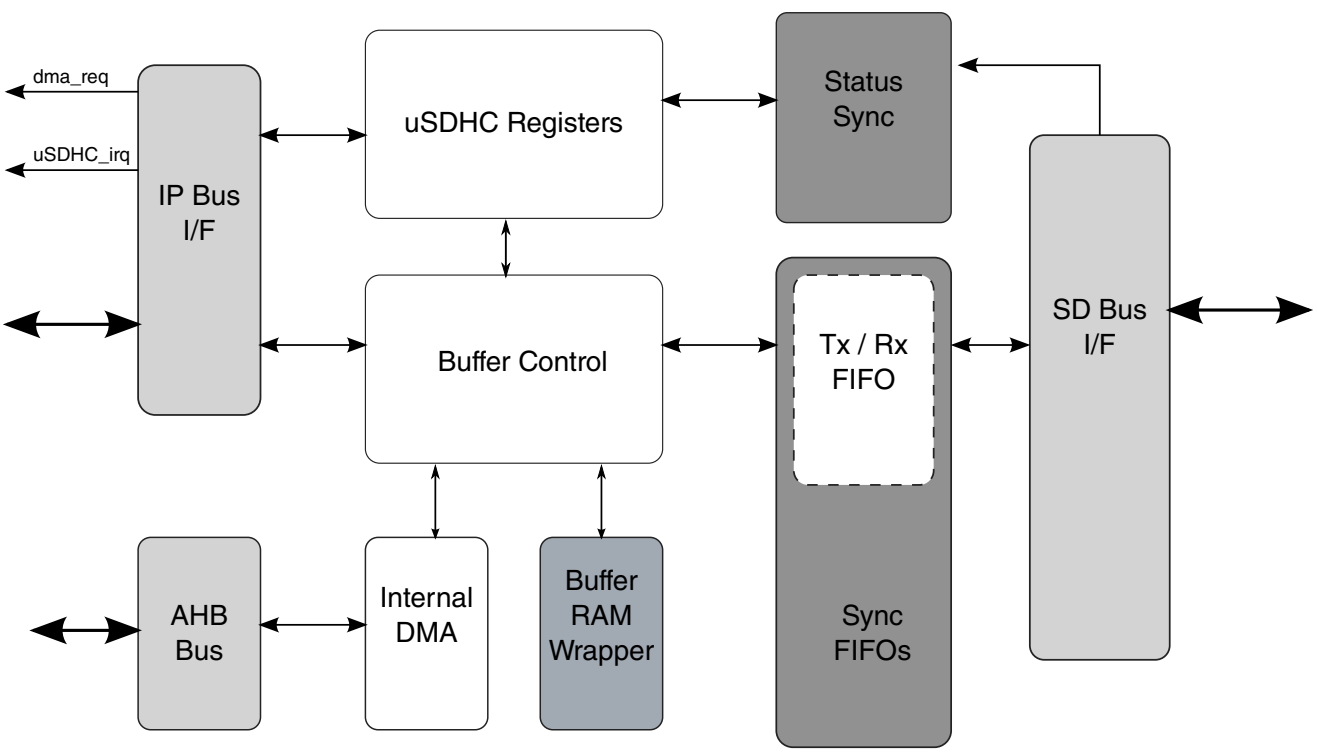


Figure 10-37. uSDHC Buffer Scheme

There are 3 transfer modes to access the data buffer:

- CPU polling mode:
  - For a host read operation, when the number of words received in the buffer meets or exceeds the RD_WML watermark value, by polling the BRR bit, the Host Driver can read the Buffer Data Port register to fetch the amount of words set in the RD_WML register from the buffer. The write operation is similar.
- External DMA mode:
  - For a read operation, when there are more words received in the buffer than the amount set in the RD_WML register, a DMA request is sent out to inform the external DMA to fetch the data. The request will be immediately de-asserted when there is an access on the Buffer Data Port register. If the number of words in the buffer after the current burst meets or exceeds RD_WML value, the DMA request is asserted again. For instance, if there are twice as many words in the buffer as there are in the RD_WML value, there are two successive DMA requests with only one cycle of de-assertion between. The write operation is similar. Note the accesses CPU polling mode and external DMA mode both use the IP bus, and if the external DMA is enabled, in both modes an external DMA request is sent when the buffer is ready.
- Internal DMA mode (includes simple and advanced DMA accesses):
  - The internal DMA access, either by simple or advanced DMA, is over the AHB bus. For internal DMA access mode, the external DMA request will never be sent out.

For a read operation, when there are more words in the buffer than the amount set in the RD_WML register, the internal DMA starts fetching data over the AHB bus. Except for INCR4 and INCR8, the burst type is always INCR mode and the burst length depends on the shortest of following factors:

- Burst length configured in the burst length field of the Watermark Level register
- Watermark Level boundary
- Block size boundary
- Data boundary configured in the current descriptor (if the ADMA is active)
- 1 Kbyte address boundary defined in the AHB protocol

Write operation is similar.

Sequential and contiguous access is necessary to ensure the pointer address value is correct. Random or skipped access is not possible. The byte order, by reset, is little endian mode. The actual byte order is swapped inside the buffer, according to the endian mode configured by software (see the following figures). For a host write operation, byte order is swapped after data is fetched from the buffer and ready to send to the SD Bus. For a host read operation, byte order is swapped before the data is stored in the buffer.

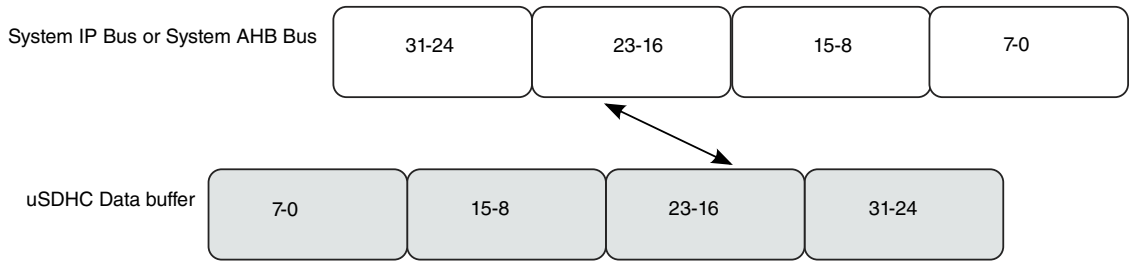


Figure 10-38. Data Swap between System Bus and uSDHC Data Buffer in Byte Little Endian Mode

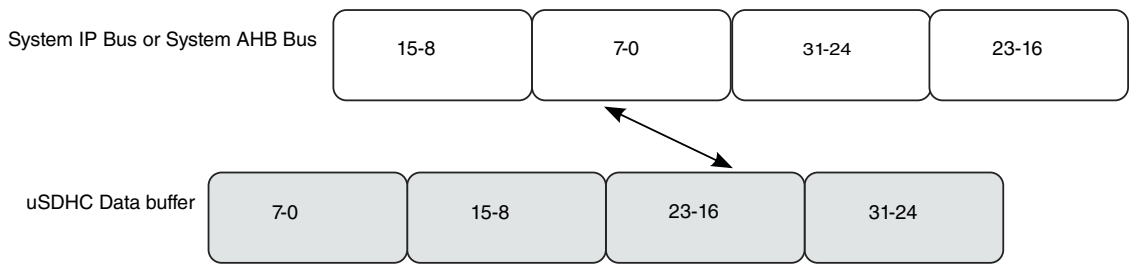


Figure 10-39. Data Swap between System Bus and uSDHC Data Buffer in Half Word Big Endian Mode

10.3.4.1.1 Write Operation Sequence

There are three ways to write data into the buffer when the user transfers data to the card:

- External DMA through the uSDHC DMA request signal
- Processor core polling through the BWR bit in Interrupt Status register (interrupt or polling)
- Internal DMA

When the internal DMA is not used, (the DMAEN bit in the Transfer Type register is not set when the command is sent), the uSDHC asserts a DMA request when the amount of buffer space exceeds the value set in the WR_WML register, and is ready for receiving new data. At the same time, the uSDHC sets the BWR bit. The buffer write ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the uSDHC will not inform the system before all the required number of bytes are transferred (if no error was encountered). When an error occurs during the data transfer, the uSDHC will abort the data transfer and abandon the current block. The Host Driver should read the contents of the DMA System Address register to obtain the starting address of the abandoned data block. If the current data transfer is in multi-block mode, the uSDHC will not automatically send CMD12, even

though the AC12EN bit in the Transfer Type register is set. The Host Driver sends CMD12 in this scenario and re-starts the write operation from that address. It is recommended that a Software Reset for Data be applied before the transfer is re-started.

The uSDHC will not start data transmission until the number of words set in the WR_WML register can be held in the buffer. If the buffer is empty and the Host System does not write data in time, the uSDHC will stop the CLK to avoid the data buffer under-run situation.

10.3.4.1.2 Read Operation Sequence

There are three ways to read data from the buffer when the user transfers data to the card:

- External DMA through the uSDHC DMA request signal
- Processor core polling through the BRR bit in Interrupt Status register (interrupt or polling)
- Internal DMA

When internal DMA is not used (DMAEN bit in Transfer Type register is not set when the command is sent), the uSDHC asserts a DMA request when the amount of data exceeds the value set in the RD_WML register, that is available and ready for system fetching data. At the same time, the uSDHC sets the BRR bit. The buffer read ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the uSDHC will not inform the system before all the required number of bytes are transferred (if no error was encountered). When an error occurs during the data transfer, the uSDHC will abort the data transfer and abandon the current block. The Host Driver should read the content of the DMA System Address register to get the starting address of the abandoned data block. If the current data transfer is in multi-block mode, the uSDHC will not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The Host Driver sends CMD12 in this scenario and re-starts the read operation from that address. It is recommended that a Software Reset for Data be applied before the transfer is re-started.

For any write transfer mode, the uSDHC will not start data transmission until the number of words set in the RD_WML register are in the buffer. If the buffer is full and the Host System does not read data in time, the uSDHC will stop the CLK to avoid the data buffer over-run situation.

10.3.4.1.3 Data Buffer and Block Size

The user needs to know the buffer size for the buffer operation during a data transfer to utilize it in the most optimized way. In the uSDHC, the only data buffer can hold up to 128 words (32-bit) and the watermark levels for write and read can be configured accordingly.

For both read and write, the watermark level can be from 1 to 128 words. For both read and write the burst length can be from 1 to 31 words. The Host Driver may configure the value according to the system situation and requirement.

During a multi-block data transfer, the block length can be set to any value between 1 and 4096 bytes, satisfying the requirements of the external card. The only restriction is from the external card, which can be limited in size or support of a partial block access (which is not the integer times of 512 bytes).

As uSDHC treats each block individually, for block sizes which are not multiples of four (not word-aligned) stuffed bytes are required at the end of each block. For example, if the block size is 7 bytes and there are 12 blocks to write, the system side must write two times for each block. For each block the ending byte will be abandoned by uSDHC because it only sends 7 bytes to the card and picks data from the following system write, resulting in 24 beats of write access in total.

10.3.4.1.4 Dividing Large Data Transfer

This SDIO command CMD53 definition limits the maximum data size of data transfers according to the following formula:

Max data size = Block size x Block count

The length of a multiple block transfer needs to be in block size units. If the total data length can't be divided evenly into a multiple of the block size, then there are two ways to transfer the data which depend on the function and the card design. Option 1 is for the Host Driver to split the transaction. The remainder of the block size data is then transferred by using a single block command at the end. Option 2 is to add dummy data in the last block to fill the block size. For option 2, the card must manage the removal of the dummy data.

See the figure below for an example showing the dividing of large data transfers, assuming a kind of WLAN SDIO card that only supports a block size up to 64 bytes. Although the uSDHC supports a block size of up to 4096 bytes, the SDIO can only accept a block size less than 64 bytes, so the data must be divided (see example below).

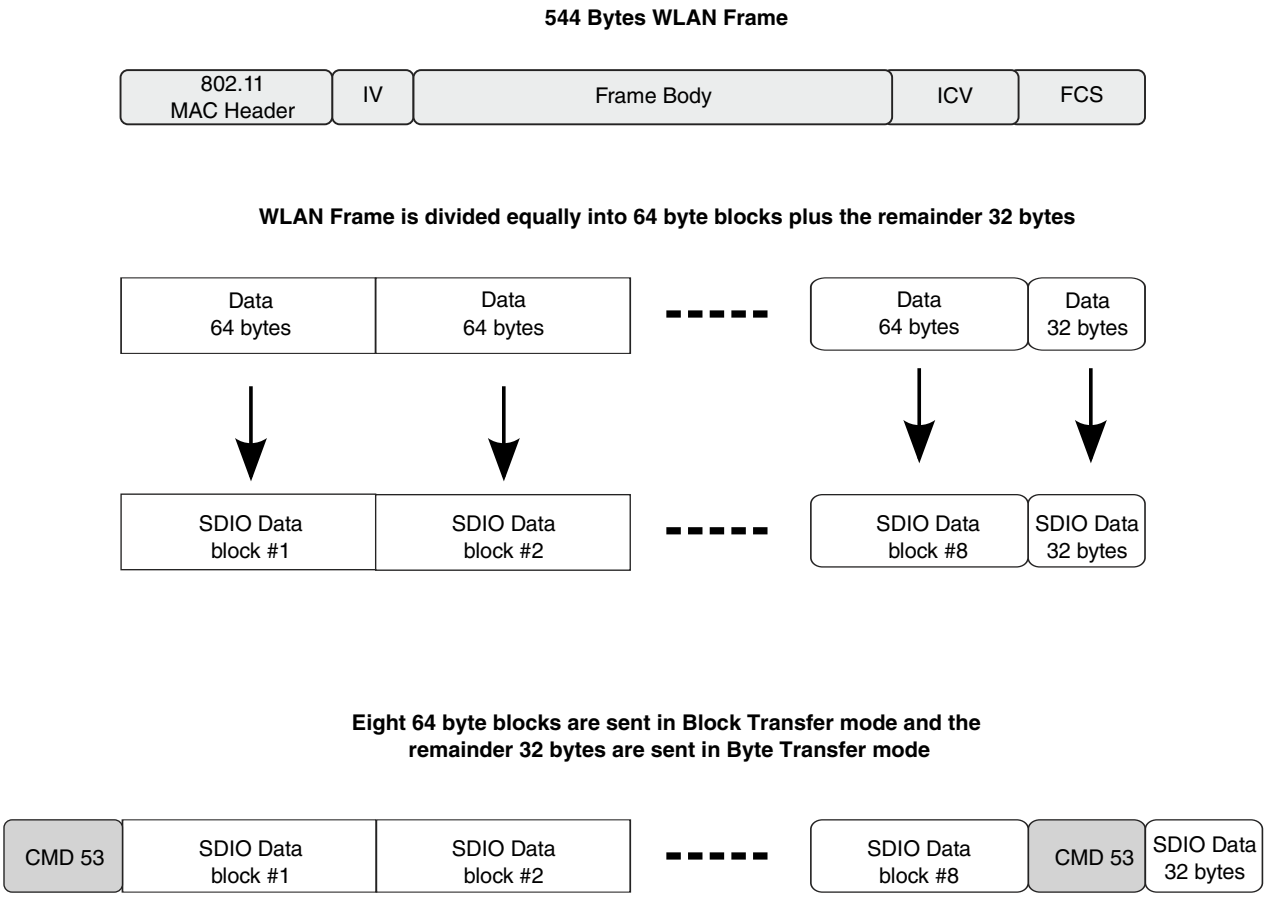


Figure 10-40. Example for Dividing Large Data Transfers

10.3.4.1.5 External DMA Request

When the internal DMA is not in use and external DMA is enabled, the Data Buffer will generate a DMA request to the system. During a write operation, when the number of WR_WML words can be held in the buffer free space, the signal uSDHC_dreq_b is asserted to 0, informing the Host System of a DMA write.

The BWR bit in the Interrupt Status register is also set, as long as the BWRSEN bit in the Interrupt Status Enable register is set. The DMA request is de-asserted after several accesses to the Data Port register are made while the buffer's free space can't meet the watermark condition (free space> write watermark level).

On read operation, when the number of RD_WML words are already in the buffer, the signal uSDHC_dreq_b is asserted to 0, informing the Host System for a DMA read. The BRR bit in the Interrupt Status register is also set, as long as the BRRSEN bit in the

Interrupt Status Enable register is set. The DMA request is de-asserted after several accesses to the Data Port register are made while the buffer's data can't meet the watermark condition (the number of data in buffer > read watermark level).

If the DMA burst length can't change during a data transfer for an external DMA transfer, the watermark level (read or write) must be a divisor of the block size. If it is not, transferring the block may cause buffer under-run (read operation) or over-run (write operation). For example, if the block size is 512 bytes, the watermark level of read (or write) must be a power of two between 1 and 128. For processor core polling access there is no such issue, as the last access in the block transfer can be controlled by software. The watermark level can be any value, even larger than the block size (but no greater than 128 words) because the actual number of bytes transferred by the software can be controlled and does not exceed the block size in each transfer.

The uSDHC also supports non-word aligned block size, as long as the card supports that block size. In this case, the watermark level should be set as the number of words. For example, if the block size is 31 bytes, the watermark level can be set to any number of words. For this case, the BLKSIZE bits of the Block Attribute register will be set as 1fh. For the CPU polling access, the burst length can be 1 to 128 words, without restriction. This is because the software will transfer 8 words, and the uSDHC will also set the BWR or BRR bits when the remaining data does not violate data buffer. See DMA Burst Length for more details about the dynamic watermark level of the data buffer.

For the above example, even though 8 words are transferred via the Data Port register, the uSDHC will transfer only 31 bytes over the SD Bus, as required by the BLKSIZE bits. In this data transfer, with non-word aligned block size, the endian mode should be set cautiously or invalid data will be transferred to and from the card.

10.3.4.2 DMA AHB Interface

The internal DMA implements a DMA engine and the AHB master. When the internal DMA is enabled, the uSDHC_dreq_b will not be asserted during the transfer, but the BWR and BRR bits will be set if the BWRSEN and BRRSEN bits have been set in the Interrupt Status Enable register.

See the figure below for an illustration of the DMA AHB interface block.

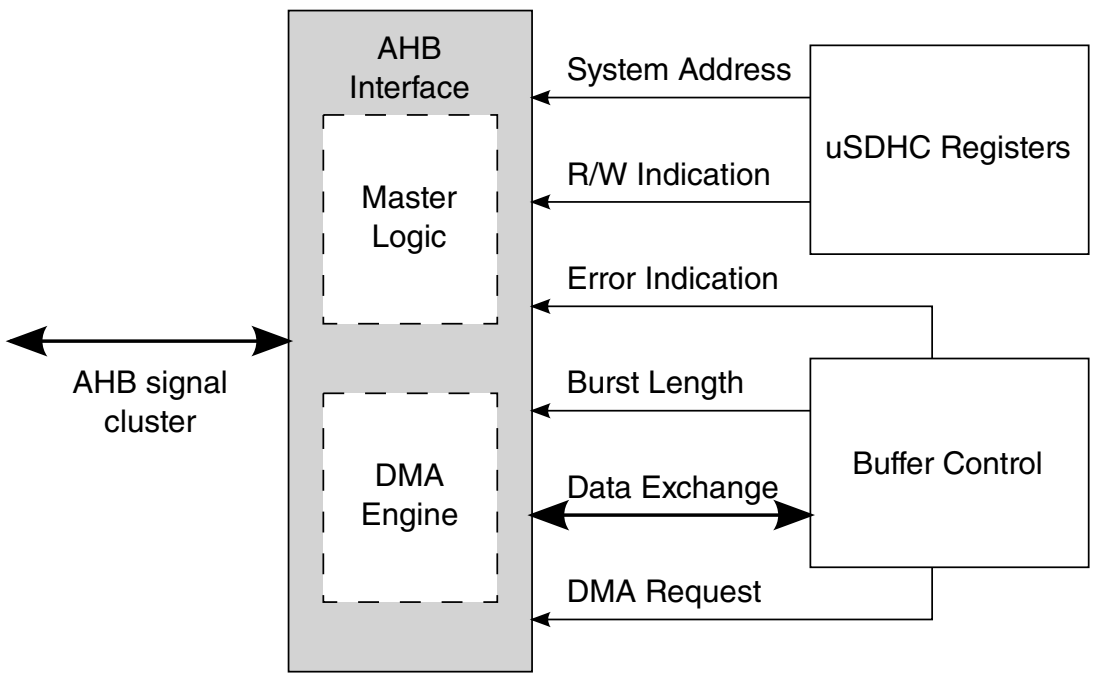


Figure 10-41. DMA AHB Interface Block

10.3.4.2.1 Internal DMA Request

If the watermark level requirement is met in data transfer or if the last data of current block is ready in the data buffer, and the Internal DMA is enabled, the Data Buffer block will send a DMA request to AHB interface. Meanwhile, the external DMA request signal (uSDHC_dreq_b) is disabled.

The delay in response from the internal DMA engine depends on the system AHB bus loading and the priority assigned to the uSDHC. The DMA engine does not respond to the request during its burst transfer, but is ready to serve as soon as the burst is over. The Data Buffer de-asserts the request if the data buffer space(for write) or bytes in data buffer is smaller than the watermark level. Upon access to the buffer by internal DMA, the Data Buffer updates its internal buffer pointer, and when the watermark level is satisfied or the last data of current block is ready in the data buffer, another DMA request is sent.

The data transfer is in the block unit, and the subsequent watermark level is always set as the remaining number of words. For instance, for a multi block data read with each block size of 31 bytes, and the burst length set to 6 words. After the first burst transfer, if there are more than 2 words in the buffer (which might contain some data of the next block), another DMA request is sent. This is because the remaining number of words to send for the current block is (31 - 6 * 4) / 4 = 2. The uSDHC will read 2 words out of the buffer, with 7 valid bytes and 1 stuffed byte.

10.3.4.2.2 DMA Burst Length

Just like a CPU polling access, the DMA burst length for the internal DMA engine can be from 1 to 16 words. The actual burst length for the DMA depends on the lesser of the configured burst length or the remaining words of the current block.

See the example in Internal DMA Request. After 6 words are read, the burst length will be 2 words, then the next burst length will be 6 words. This is because the next block starts, which is 31 bytes, more than 6 words. The Host Driver may take this variable burst length into account. It is also acceptable to configure the burst length as the divisor of the block size, so that each time the burst length will be the same.

10.3.4.2.3 AHB Master Interface

It is possible that the internal AHB DMA engine could fail during the data transfer. Upon detection of an AHB bus error during DMA transfer, the DMA engine stops the transfer and goes to the idle state.At that point, the internal data buffer stops receiving incoming data and sending out data. The DMAE bit in the Interrupt Status register will be generated to host CPU to report a bus error condition.

Once the DMAE interrupt is received, the software shall send a CMD12 to abort the current transfer and read the DS_ADDR bits of the DMA System Address register to get the starting address of the corrupted block. After the DMA error is fixed, the software should apply a data reset and re-start the transfer from this address to recover the corrupted block. DMA operation will resume when the interrupt is serviced by software.

10.3.4.2.4 ADMA Engine

In the SD Host Controller Standard, a new DMA transfer algorithm called the ADMA (Advanced DMA) is defined. For Simple DMA, once the page boundary is reached, a DMA interrupt will be generated and the new system address shall be programmed by the Host Driver.

The ADMA defines the programmable descriptor table in the system memory. The Host Driver can calculate the system address at the page boundary and program the descriptor table before executing ADMA. It reduces the frequency of interrupts to the host system. Therefore, higher speed DMA transfers could be realized since the Host MCU intervention would not be needed during long DMA based data transfers.

There are two types of ADMA: ADMA1 and ADMA2 in Host Controller. ADMA1 can support data transfer of 4KB aligned data in system memory. ADMA2 improves the restriction so that data of any location and any size can be transferred in system memory. Their formats of Descriptor Table are different.

ADMA can recognize all kinds of descriptors define in SD Host Controller Standard, and if 'End' flag is detected in the descriptor, ADMA will stop after this descriptor is processed.

10.3.4.2.4.1 ADMA Concept and Descriptor Format

For ADMA1, including the following descriptors:

- Valid/Invalid descriptor.
- Nop descriptor.
- Set data length descriptor.
- Set data address descriptor.
- Link descriptor.
- Interrupt flag and End flag in descriptor.

For ADMA2, including the following descriptors:

- Valid/Invalid descriptor.
- Nop descriptor.
- Rsv descriptor.
- Set data length & address descriptor.
- Link descriptor.
- Interrupt flag and End flag in descriptor.

See Figure 10-42 for the format of the descriptor table for ADMA1.

Figure 10-45 explains the ADMA2 format. ADMA2 deals with the lower 32-bit first, and then the higher 32-bit. If the 'Valid' flag of descriptor is 0, it will ignore the high 32-bit. Address field shall be set on word aligned(lower 2-bit is always set to 0). Data length is in byte unit.

ADMA will start read/write operation after it reaches the Tran state, using the data length and data address analyzed from most recent descriptor(s).

For ADMA1, the valid data length descriptor is the last Set type descriptor before Tran type descriptor. Every Tran type will trigger a transfer, and the transfer data length is extracted from the most recent Set type descriptor. If there is no Set type descriptor after the previous Trans descriptor, the data length will be the value for previous transfer, or 0 if no Set descriptor is ever met.

For ADMA2, Tran type descriptor contains both data length and transfer data address, so only a Tran type descriptor can start a data transfer

Address/ Page Field		Address/ Page Field		Attribute Field					
31	12	11	6	5	4	3	2	1	0
Address or Data Length		000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	31- 28	27- 12
0	0	Nop	No Operation	Don't Care	
0	1	Set	Set Data Length	0000	Data Length
1	0	Tran	Transfer Data	Data Address	
1	1	Link	Link Descriptor	Descriptor Address	

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Figure 10-42. Format of the ADMA1 Descriptor Table

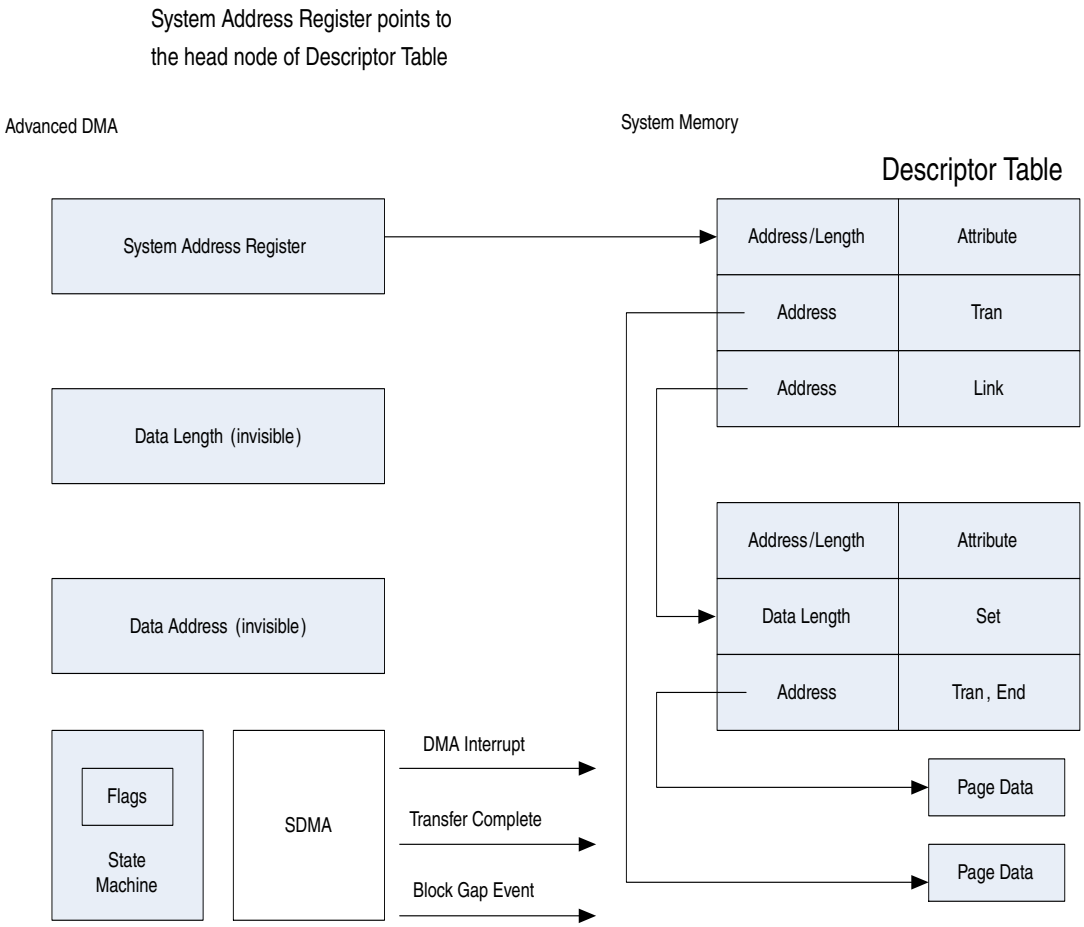


Figure 10-43. Concept and Access Method of ADMA1 Descriptor Table

Address Field		Length		Reserved		Attribute Field					
63	32	31	16	15	06	05	04	03	02	01	00
32-bit Address		16-bit length		0000000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	Operation
0	0	Nop	No Operation	Don't Care
0	1	Rsv	Reserved	Same as Nop. Read this line and go to next one
1	0	Tran	Transfer Data	Transfer data with address and length set in this descriptor line
1	1	Link	Link Descriptor	Link to another descriptor

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Figure 10-44. Format of the ADMA2 Descriptor Table

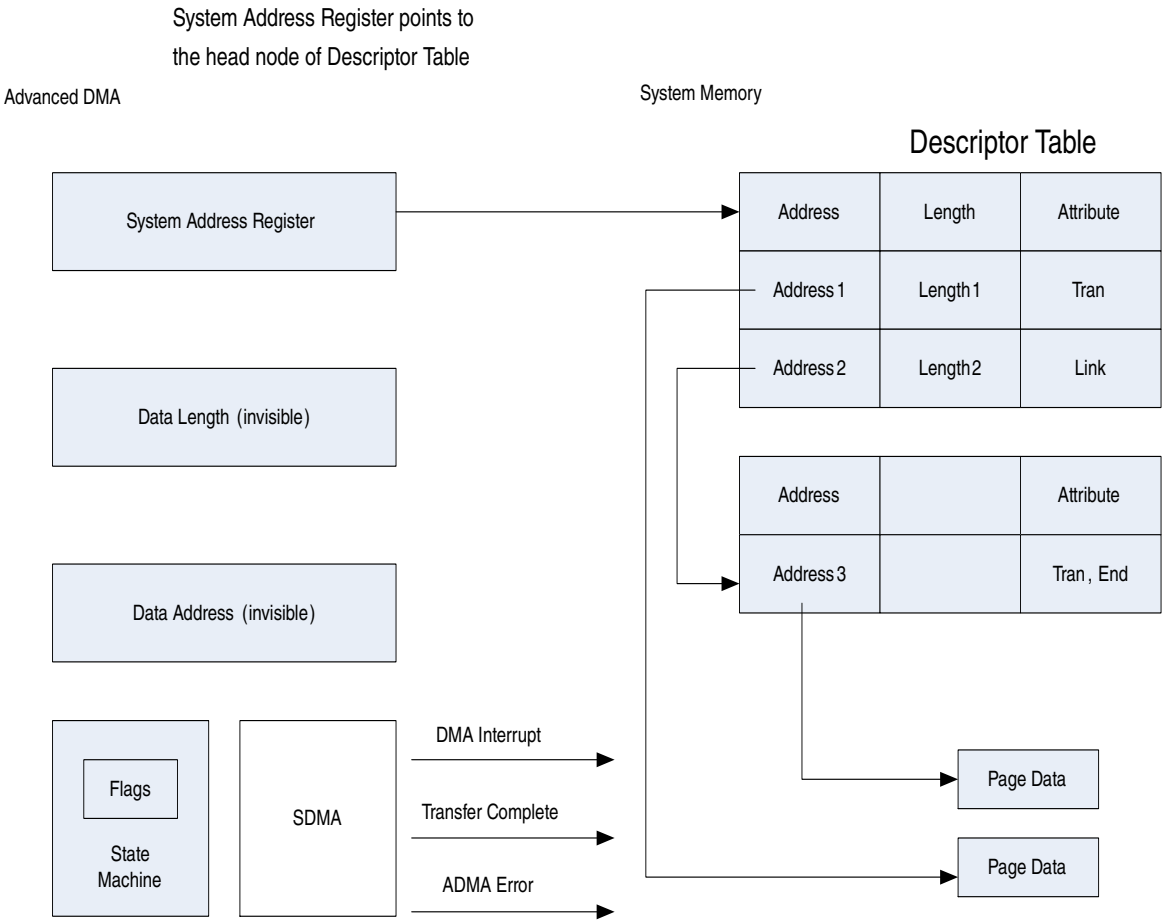


Figure 10-45. Concept and Access Method of ADMA2 Descriptor Table

10.3.4.2.4.2 ADMA Interrupt

If the interrupt flag descriptor is set, ADMA will generate an interrupt according to various types of descriptors:

For ADMA1:

- Set type of descriptor: interrupt is generated when data length is set.
- Tran type descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop type of descriptor: interrupt is generated just after this descriptor is fetched.

For ADMA2:

- Tran type of descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop/Rsv type of descriptor: interrupt is generated just after this descriptor is fetched.

10.3.4.2.4.3 ADMA Error

The ADMA will stop whenever any error is encountered. These errors include:

- Fetching descriptor error
- AHB response error
- Data length mismatch error

An ADMA descriptor error will be generated when it fails to detect a 'Valid' flag in the descriptor. If an ADMA descriptor error occurs, the interrupt is not generated even if the 'Interrupt' flag of this descriptor is set.

When BLKCNTEN bit is set, data length set in buffer must be equal to the whole data length set in descriptor nodes, otherwise data length mismatch error will be generated.

When BLKCNTEN bit is not set, then whole data length set in descriptor should be a multiple of block lengths; otherwise, when data set in the descriptor nodes are not performed at block boundaries, then data mismatch errors will occur.

10.3.4.3 Register Bank with IP Bus Interface

Register accesses via the IP Bus interface are actually on the Register Bank.

See Figure 10-46 below for the block diagram.

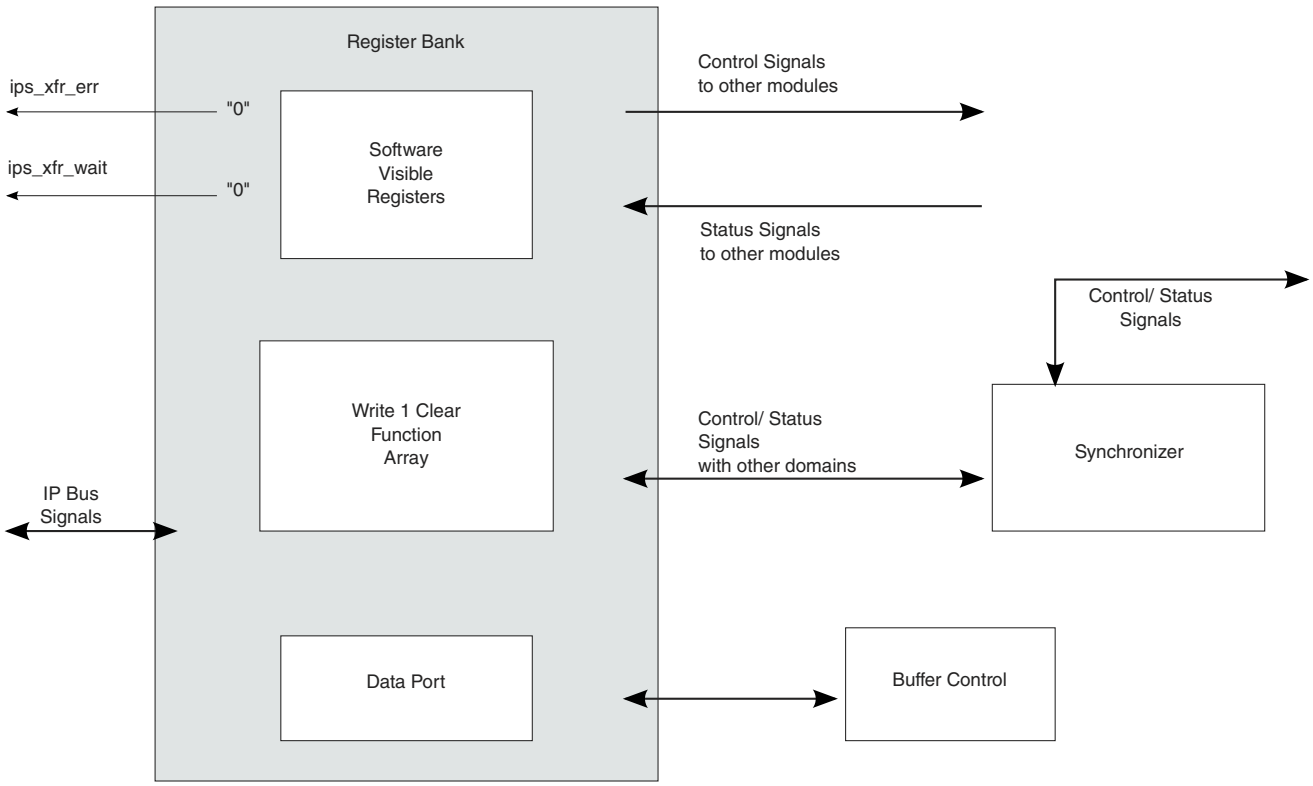


Figure 10-46. Register Bank Diagram

Only 32-bit access is allowed, and no partial read / write is supported, thus all accesses are word aligned.

10.3.4.3.1 SD Protocol Unit

The SD protocol unit deals with all SD protocol affairs.

The SD Protocol Unit performs the following functions:

- Acts as the bridge between the internal buffer and the SD bus
- Sends the command data as well as its argument serially
- Stores the serial response bit stream into corresponding registers
- Detects the bus state on the CMD/DAT lines
- Monitors the interrupt from the SDIO card
- Asserts the read wait signal
- Gates off the SD clock when buffer is announcing danger status
- Detects the write protect state

The SD Protocol Unit consists of four sub modules:

1. SD control misc.
2. Command control.

3. Data control.
4. Clock control

10.3.4.3.2 SD control misc

In the SD control misc unit, the card detect(include the CD_B and DATA3 used as Card Detection), write protection and card interrupt are implemented.

This module monitors the signal level on all 8 data lines, the command lines, and directly routes the level values into the Register Bank. The driver can use this for debug purposes.

The module also detects the WP (Write Protect) line. If WP is active, writes to the register bank will be ignored.

This module also drives the LCTL output signal when the LCTL bit is set by the driver.

10.3.4.3.3 SD Clock control

If the internal data buffer is near full(for read) or near empty(for write), the SD clock must be gated off to avoid buffer over/under-run, this module will assert the gate of the output SD clock to shut the clock off. After the buffer has space(for read) or has data(for write), the clock gate of this module will open and the SD clock will be active again.

10.3.4.3.4 Command control

The Command Control module deals with the transactions on the CMD line.

See the figure below for an illustration of the structure for the Command CRC Shift Register.

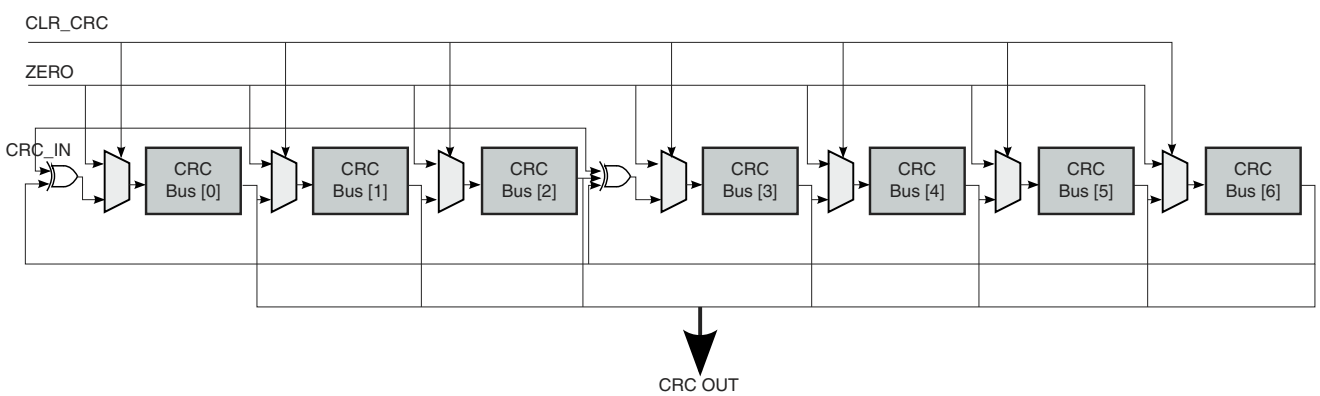


Figure 10-47. Command CRC Shift Register

The CRC polynomials for the CMD are as follows:

Generator polynomial: $G(x) = x^7 + x^3 + 1$
$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + ... + (\text{last bit}) * x^0$
$CRC[6:0] = \text{Remainder } [(M(x) * x^7) / G(x)]$

10.3.4.3.5 Data control

The Data Agent deals with the transactions on the eight data lines. Moreover, this module also detects the busy state on the DATA0 line, and generates the Read Wait state by the request from the Transceiver.

The CRC polynomials for the DATA are as follows:

Generator polynomial: $G(x) = x^{16} + x^{12} + x^5 + 1$
$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + ... + (\text{last bit}) * x^0$
$CRC[15:0] = \text{Remainder } [(M(x) * x^{16}) / G(x)]$

10.3.4.4 Clock & Reset Manager

This module controls all the reset signals within the uSDHC.

There are four kinds of reset signals within uSDHC:

1. Hardware reset.
2. Software reset for all logic.
3. Software reset for the data logic.
4. Software reset for the command logic.

All these signals are fed into this module and stable signals are generated inside the module to reset all other modules. The module also gates off all the inside signals.

10.3.4.5 Clock Generator

The Clock Generator generates the card CLK by peripheral source clock in two stages.

Refer to the figure below for the structure of the divider. The term "Base" represents the frequency of peripheral source clock.

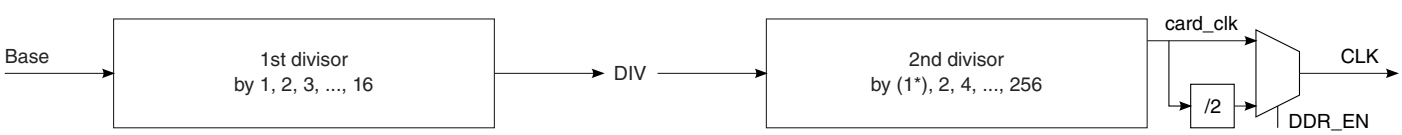


Figure 10-48. Two Stages of the Clock Divider

The first stage outputs an intermediate clock (DIV), which can be Base, Base/2, Base/3, ..., or Base/16.

The second stage is a prescaler, and outputs the actual internal working clock (card_clk). This clock is the driving clock for all sub modules of the SD Protocol Unit, and the sync FIFOs (see Figure 10-37) to synchronize with the data rate from the internal data buffer. The frequency of the clock output from this stage, can be DIV, DIV/2, DIV/4,..., or DIV/256. Thus the highest frequency of the card_clk is Base, and the next highest is Base/2, while the lowest frequency is Base/4096. If the duty cycle of Base clock is 50%, the duty cycle of card_clk is also 50%, even when the compound divisor is an odd value.

Please note, in SDR mode and DDR mode, the CLK are different.

- In SDR mode, CLK is equal to the internal working clock(card_clk).

- In DDR mode, CLK is equal to the card_clk/2.

10.3.4.6 SDIO Card Interrupt

Information on Interrupts in 1-bit Mode, Interrupts in 4-bit Mode, and Card Interrupt Handling are detailed in the sections below.

10.3.4.6.1 Interrupts in 1-bit Mode

In this case the DATA1 pin is dedicated to providing the interrupt function. An interrupt is asserted by pulling the DATA1 low from the SDIO card, until the interrupt service is finished to clear the interrupt.

10.3.4.6.2 Interrupt in 4-bit Mode

Since the interrupt and data line 1 share Pin 8 in 4-bit mode, an interrupt will only be sent by the card and recognized by the host during a specific time. This is known as the Interrupt Period. The uSDHC will only sample the level on Pin 8 during the Interrupt Period. At all other times, the host will ignore the level on Pin 8, and treat it as the data signal. The definition of the Interrupt Period is different for operations with single block and multiple block data transfers.

In the case of normal single data block transmissions, the Interrupt Period becomes active two clock cycles after the completion of a data packet. This Interrupt Period lasts until after the card receives the end bit of the next command that has a data block transfer associated with it.

For multiple block data transfers in 4-bit mode, there is only a limited period of time that the Interrupt Period can be active due to the limited period of data line availability between the multiple blocks of data. This requires a more strict definition of the Interrupt Period. For this case, the Interrupt Period is limited to two clock cycles. This begins two

clocks after the end bit of the previous data block. During this 2-clock cycle interrupt period, if an interrupt is pending, the DATA1 line will be held low for one clock cycle with the last clock cycle pulling DATA1 high. On completion of the Interrupt Period, the card releases the DATA1 line into the high Z state. The uSDHC samples the DATA1 during the Interrupt Period when the IABG bit in the Protocol Control register is set.

Refer to SDIO Card Specification v1.10f for further information about the SDIO card interrupt.

10.3.4.6.3 Card Interrupt Handling

When the CINTIEN bit in the Interrupt Signal Enable Register is set to 0, the uSDHC clears the interrupt request to the Host System. The Host Driver should clear this bit before servicing the SDIO Interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.

The SDIO Card Interrupt Status can be cleared by writing 1 to this bit. But as the interrupt source from the SDIO card does not clear, this bit is set again. In order to clear this bit, it is required to reset the interrupt source from the external card followed by a writing 1 to this bit. In 1-bit mode, the uSDHC will detect the SDIO Interrupt with or without the SD clock (to support wakeup). In 4-bit mode, the interrupt signal is sampled during the Interrupt Period, so there are some sample delays between the interrupt signal from the SDIO card and the interrupt to the Host System Interrupt Controller. When the SDIO status has been set, and the Host Driver needs to service this interrupt, so the SDIO bit in the Interrupt Control Register of SDIO card will be cleared. This is required to clear the SDIO interrupt status latched in the uSDHC and to stop driving the interrupt signal to the System Interrupt Controller. The Host Driver must issue a CMD52 to clear the card interrupt. After completion of the card interrupt service, the SDIO Interrupt Status Enable bit is set to 1, and the uSDHC starts sampling the interrupt signal again.

See the figure below for an illustration of the SDIO card interrupt scheme and for the sequences of software and hardware events that take place during a card interrupt handling procedure.

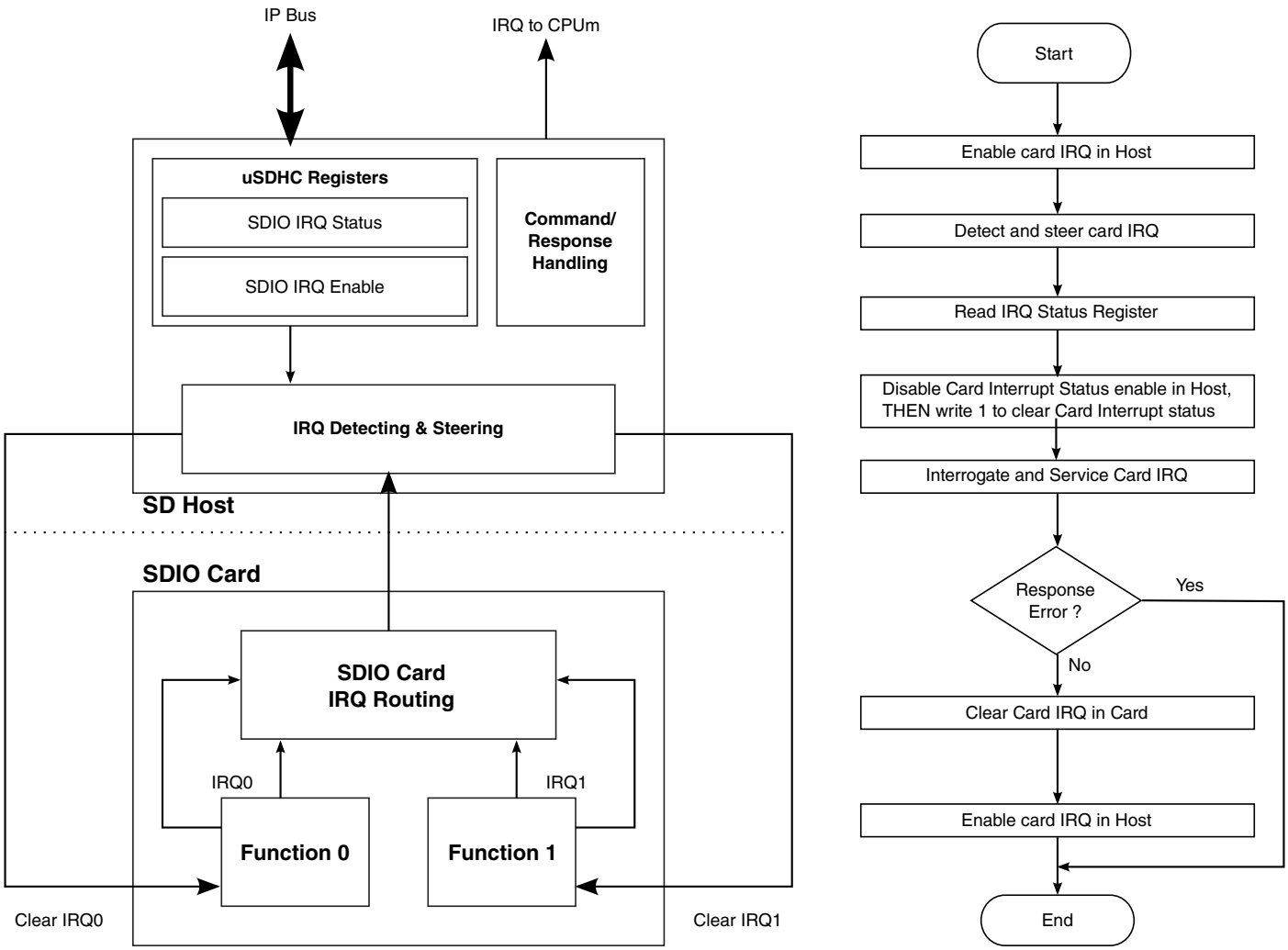


Figure 10-49. Card Interrupt Scheme and Card Interrupt Detection and Handling Procedure

10.3.4.7 Card Insertion and Removal Detection

The uSDHC uses either the DATA3 pin or the CD_B pin to detect card insertion or removal. When there is no card on the MMC/SD bus, the DATA3 will be pulled to a low voltage level by default.

When any card is inserted to or removed from the socket, the uSDHC detects the logic value changes on the DATA3 pin and generates an interrupt. When the DATA3 pin is not used for card detection (for example, it is implemented in GPIO), the CD_B pin must be connected for card detection. Whether DATA3 is configured for card detection or not, the CD_B pin is always a reference for card detection. Whether the DATA3 pin or the CD_B pin is used to detect card insertion, the uSDHC will send an interrupt (if enabled) to inform the Host system that a card is inserted.

10.3.4.8 Power Management and Wake Up Events

When there is no operation between the uSDHC and the card through the SD bus, the user can completely disable the ipg_clk and ipg_perclk in the chip level clock control module to save power. When the user needs to use the uSDHC to communicate with the card, it can enable the clock and start the operation.

In some circumstances, when the clocks to the uSDHC are disabled, for instance, when the system is in low power mode, there are some events for which the user needs to enable the clock and handle the event. These events are called wakeup interrupts. The uSDHC can generate these interrupt even when there are no clocks enabled. The three interrupts which can be used as wake up events are:

1. Card Removal Interrupt
2. Card Insertion Interrupt
3. Interrupt from SDIO card

The uSDHC offers a power management feature. By clearing the clock enabled bits in the System Control Register, the clocks are gated in the low position to the uSDHC. For maximum power saving, the user can disable all the clocks to the uSDHC when there is no operation in progress.

These three wake up events (or wakeup interrupts) can also be used to wake up the system from low-power modes.

NOTE

To make the interrupt a wakeup event, when all the clocks to the uSDHC are disabled or when the whole system is in low power mode, the corresponding wakeup enabled bit needs to be set. Refer to Protocol Control (uSDHC_PROT_CTRL) for more information on the uSDHC Protocol Control register.

10.3.4.8.1 Setting Wake Up Events

For the uSDHC to respond to a wakeup event, the software must set the respective wakeup enable bit before the CPU enters sleep mode.

Before the software disables the host clock, it should ensure that all of the following conditions have been met:

- No Read or Write Transfer is active
- Data and Command lines are not active
- No interrupts are pending
- Internal data buffer is empty

10.3.4.9 MMC fast boot

The Embedded MultiMediaCard (eMMC4.3) specification adds a fast boot feature which requires hardware support. There are two types of fast boot mode, boot operation, and alternative boot operation in the eMMC4.3 specification. Each type also has with-acknowledge and without-acknowledge modes.

In boot operation mode, the master (MultiMediaCard host) can read boot data from the slave (MMC device) by keeping CMD line low after power-on, or sending CMD0 with argument + 0xFFFFFFFA (optional for slave), before issuing CMD1.

NOTE

For the eMMC4.3 card setting, please see the eMMC4.3 specification.

10.3.4.9.1 Boot operation

NOTE

For the purposes of this documentation, fast boot is called "normal fast boot mode".

If the CMD line is held LOW for 74 clock cycles and more after power-up before the first command is issued, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within 1 second after the CMD line goes LOW, the slave starts to send the first boot data to the master on the DATA line(s). The master must keep the CMD line LOW to read all of the boot data.

If boot acknowledge is enabled, the slave has to send acknowledge pattern '010' to the master within 50ms after the CMD line goes LOW. If boot acknowledge is disabled, the slave will not send out acknowledge pattern '010'.

The master can terminate boot mode with the CMD line HIGH.

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.

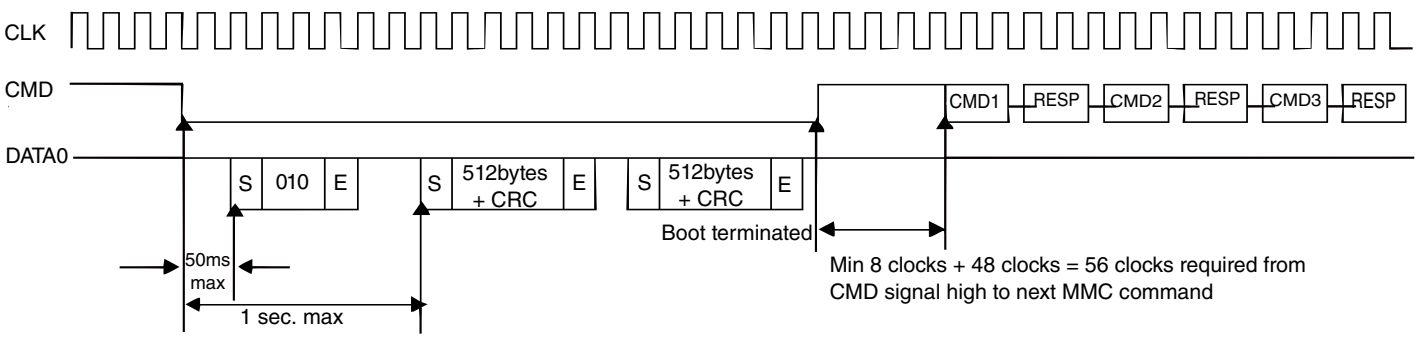


Figure 10-50. MultiMediaCard state diagram (normal boot mode)

10.3.4.9.2 Alternative boot operation

This boot function is optional for the device. If bit 0 in the extended CSD byte[228] is set to '1', the device supports the alternative boot operation.

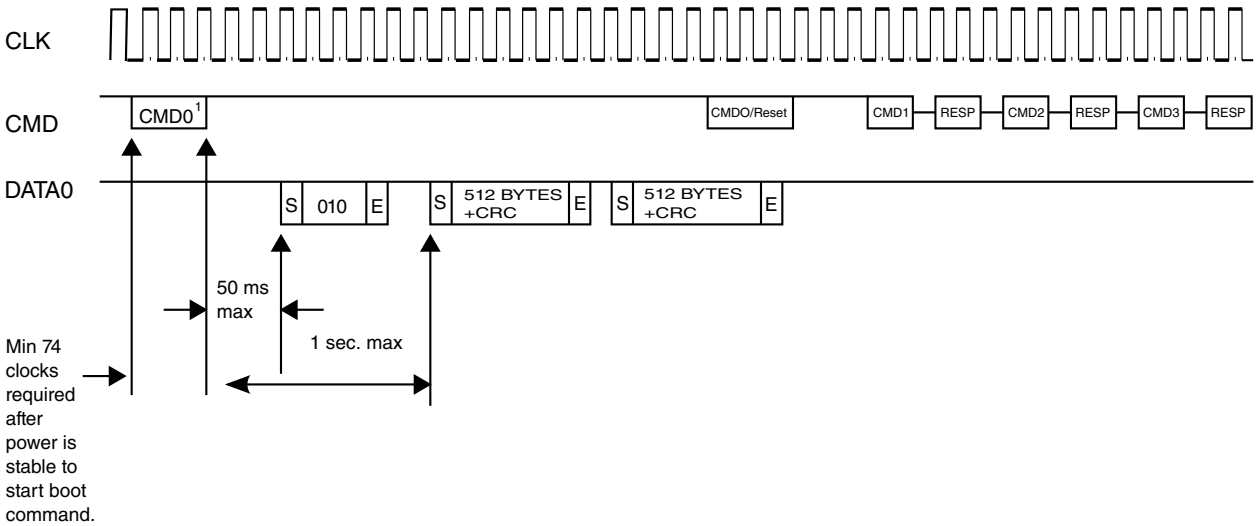
After power-up, if the host issues CMD0 with the argument of 0xFFFFFFFA after 74 clock cycles, before CMD1 is issued or the CMD line goes low, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within 1 second after CMD0 with the argument of 0xFFFFFFFA is issued, the slave starts to send the first boot data to the master on the DATA line(s).

If boot acknowledge is enabled, the slave has to send the acknowledge pattern '010' to the master within 50ms after the CMD0 with the argument of 0xFFFFFFFA is received. If boot acknowledge is disabled, theslave will not send out acknowledge pattern '010'.

The master can terminate boot mode by issuing CMD0 (Reset).

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.



NOTE 1. CMD0 with argument 0xFFFFFFFA

Figure 10-51. MultiMediaCard state diagram (alternative boot mode)

10.3.5 Initialization/Application of uSDHC

All communication between the system and cards are controlled by the host. The host sends commands of two types: broadcast and addressed (point-to-point).

Broadcast commands are intended for all cards, such as GO_IDLE_STATE, SEND_OP_COND, ALL_SEND_CID. In Broadcast mode, all cards are in the open-drain mode to avoid bus contention. Refer to Commands for MMC/SD/SDIO for the commands of bc and bcr categories.

After the Broadcast command CMD3 is issued, the cards enter standby mode. Addressed type commands are used from this point. In this mode, the CMD/DATA I/O pads will turn to push-pull mode, to have the driving capability for maximum frequency operation. Refer to Commands for MMC/SD/SDIO for the commands of ac and adtc categories.

10.3.5.1 Command Send & Response Receive Basic Operation

Assuming the data type WORD is an unsigned 32-bit integer, the below flow is a guideline for sending a command to the card(s):

```
send_command(cmd_index, cmd_arg, other requirements)
{
WORD wCmd; // 32-bit integer to make up the data to write into Transfer Type register, it is
recommended to implement in a bit-field manner
wCmd = (<cmd_index> & 0x3f) >> 24; // set the first 8 bits as '00'+<cmd_index>
set CMDTYP, DPSEL, CICEN, CCCEN, RSTTYP, DTDSEL accorind to the command index;
if (internal DMA is used) wCmd |= 0x1;
if (multi-block transfer) {
          set MSBSEL bit;
          if (finite block number) {
                    set BCEN bit;
                    if (auto12 command is to use) set AC12EN bit;
          }
}
write_reg(CMDARG, <cmd_arg>); // configure the command argument
write_reg(XFERTYP, wCmd); // set Transfer Type register as wCmd value to issue the command
}
wait_for_response(cmd_index)
{
while (CC bit in IRQ Status register is not set); // wait until Command Complete bit is set
read IRQ Status register and check if any error bits about Command are set
if (any error bits are set) report error;
write 1 to clear CC bit and all Command Error bits;
}
```

For the sake of simplicity, the function wait_for_response is implemented here by means of polling. For an effective and formal way, the response is usually checked after the Command Complete Interrupt is received. When doing this, make sure the corresponding interrupt status bits are enabled.

For some scenarios, the response time-out is expected. For instance, after all cards respond to CMD3 and go to the Standby State, no response to the Host when CMD2 is sent. The Host Driver will deal with "fake" errors like this with caution.

10.3.5.2 Card Identification Mode

When a card is inserted to the socket or the card was reset by the host, the host needs to validate the operation voltage range, identify the cards, request the cards to publish the Relative Card Address (RCA) or to set the RCA for the MMC cards.

10.3.5.2.1 Card Detect

See the figure below for a flow diagram showing the detection of MMC, SD and SDIO cards using the uSDHC.

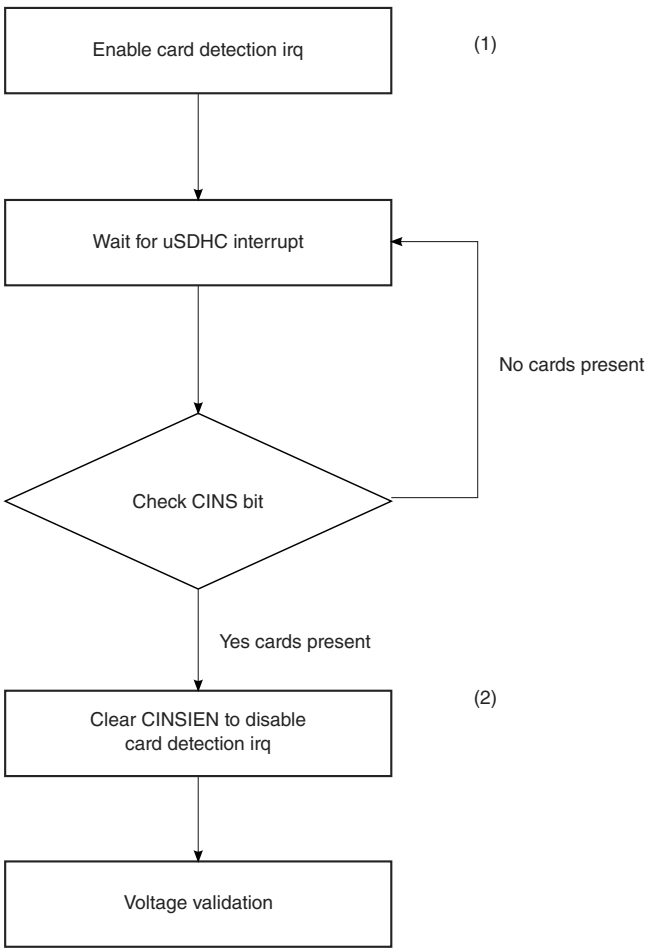


Figure 10-52. Flow Diagram for Card Detection

Here is the card detect sequence:

- Set the CINSIEN bit to enable card detection interrupt
- When an interrupt from the uSDHC is received, check the CINS bit in the Interrupt Status register to see if it was caused by card insertion
- Clear the CINSIEN bit to disable the card detection interrupt and ignore all card insertion interrupts afterwards

10.3.5.2.2 Reset

The host consists of three types of resets:

- Hardware reset (Card and Host) which is driven by POR (Power On Reset)

- Software reset (Host Only) is initiated by the write operation on the RSTD, RSTC, or RSTA bits of the System Control register to reset the data part, command part, or all parts of the Host Controller, respectively
- Card reset (Card Only). The command, "Go_Idle_State" (CMD0), is the software reset command for all types of MMC cards, SD Memory cards. This command sets each card into the Idle State regardless of the current card state. For an SD I/O Card, CMD52 is used to write an I/O reset in the CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host needs to validate the voltage range of the card. See the figure below for the software flow to reset both the uSDHC and the card.

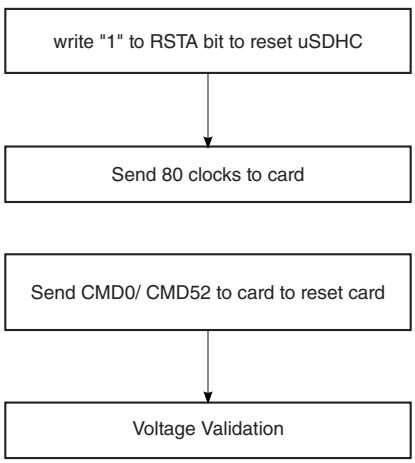


Figure 10-53. Flow Chart for Reset of the uSDHC and SD I/O Card

```
software_reset()
{
set_bit(SYSCTRL, RSTA); // software reset the Host
set DTOCV and SDCLKFS bit fields to get the CLK of frequency around 400kHz
configure IO pad to set the power voltage of external card to around 3.0V
poll bits CIHB and CDIHB bits of PRSSTAT to wait both bits are cleared
set_bit(SYSCTRL, INTIA); // send 80 clock ticks for card to power up
send_command(CMD_GO_IDLE_STATE, <other parameters>); // reset the card with CMD0
or send_command(CMD_IO_RW_DIRECT, <other parameters>);
}
```

10.3.5.2.3 Voltage Validation

All cards should be able to establish communication with the host using any operation voltage in the maximum allowed voltage range specified in the card specification. However, the supported minimum and maximum values for Vdd are defined in the Operation Conditions Register (OCR) and may not cover the whole range.

Cards that store the CID and CSD data in the preload memory are only able to communicate this information under data transfer Vdd conditions. This means if the host and card have non-common Vdd ranges, the card will not be able to complete the identification cycle, nor will it be able to send CSD data.

Therefore, a special command Send_Op_Cont (CMD1 for MMC), SD_Send_Op_Cont (ACMD41 for SD Memory) and IO_Send_Op_Cont (CMD5 for SD I/O) is used. The voltage validation procedure is designed to provide a mechanism to identify and reject cards which do not match the Vdd range(s) desired by the host. This is accomplished by the host sending the desired Vdd voltage window as the operand of this command. Cards that can't perform the data transfer in the specified range must discard themselves from further bus operations and go into the Inactive State. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the Inactive State. This query should be used if the host is able to select a common voltage range or if a notification shall be sent to the system when a non-usable card in the stack is detected.

The following steps show how to perform voltage validation when a card is inserted:

```
voltage_validation(voltage_range_arguement)
{
label the card as UNKNOWN;
send_command(IO_SEND_OP_COND, 0x0, <other parameters are omitted>); // CMD5, check SDIO
operation voltage, command argument is zero
if (RESP_TIMEOUT != wait_for_response(IO_SEND_OP_COND)) { // SDIO command is accepted
          if (0 < number of IO functions) {
                    label the card as SDIO;
                    IORDY = 0;
                    while (!(IORDY in IO OCR response)) { // set voltage range for each IO
function
                              send_command(IO_SEND_OP_COND, <voltage range>, <other
parameter>);
                              wait_for_response(IO_SEND_OP_COND);
                    } // end of while ...
          } // end of if (0 < ...
          if (memory part is present inside SDIO card) Label the card as SDCombo; // this is
an
SD-Combo card
} // end of if (RESP_TIMEOUT ...
if (the card is labelled as SDIO card) return; // card type is identified and voltage range
is
set, so exit the function;
send_command(APP_CMD, 0x0, <other parameters are omitted>); // CMD55, Application specific
CMD
prefix
if (no error calling wait_for_response(APP_CMD, <...>) { // CMD55 is accepted
          send_command(SD_APP_OP_COND, <voltage range>, <...>); // ACMD41, to set voltage
range
for memory part or SD card
          wait_for_response(SD_APP_OP_COND); // voltage range is set
          if (card type is UNKNOWN) label the card as SD;
          return; //
} // end of if (no error ...
else if (errors other than time-out occur) { // command/response pair is corrupted
          deal with it by program specific manner;
} // of else if (response time-out
else { // CMD55 is refuse, it must be MMC card  if (card is already labelled as SDCombo)
{ //
change label
```

```
                    re-label the card as SDIO;
                    ignore the error or report it;
                    return; // card is identified as SDIO card
          } // of if (card is ...
          send_command(SEND_OP_COND, <voltage range>, <...>);
          if (RESP_TIMEOUT == wait_for_response(SEND_OP_COND)) { // CMD1 is not accepted,
either
                    label the card as UNKNOWN;
                    return;
          } // of if (RESP_TIMEOUT ...
} // of else
}
```

10.3.5.2.4 Card Registry

Card registry for the MMC and SD/SDIO/SD Combo cards are different. For the SD Card, the Identification process starts at a clock rate lower than 400 kHz and the power voltage higher than 2.7 V (as defined by the Card spec). At this time, the CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated, the host will request the card to send their valid operation conditions.

The response to ACMD41 is the operation condition register of the card. The same command shall be send to all of the new cards in the system. Incompatible cards are put into the Inactive State. The host then issues the command, All_Send_CID (CMD2), to each card to get its unique card identification (CID) number. Cards that are currently unidentified (in the Ready State), send their CID number as the response. After the CID is sent by the card, the card goes into the Identification State.

The host then issues Send_Relative_Addr (CMD3), requesting the card to publish a new relative card address (RCA) that is shorter than the CID. This RCA will be used to address the card for future data transfer operations. Once the RCA is received, the card changes its state to the Standby State. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.

The host repeats the identification process with CMD2 and CMD3 for each card in the system until the last CMD2 gets no response from any of the cards in system.

For MMC operation, the host starts the card identification process in open-drain mode with the identification clock rate lower than 400 kHz and the power voltage higher than 2.7 V. The open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is activated the host will request the cards to send their valid operation conditions (CMD1). The response to CMD1 is the "wired OR" operation on the condition restrictions of all cards in the system. Incompatible cards are sent into the Inactive State. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards (the cards in Ready State) simultaneously start sending their CID numbers serially, while

bit-wise monitoring their outgoing bit stream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop sending their CID immediately and must wait for the next identification cycle. Since the CID is unique for each card, only one card can be successfully send its full CID to the host. This card then goes into the Identification State. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to the card a relative card address (RCA). Once the RCA is received the card state changes to the Stand-by State, and the card does not react in further identification cycles, and its output driver switches from open-drain to push-pull. The host repeats the process, mainly CMD2 and CMD3, until the host receives a time-out condition to recognize the completion of the identification process.

For operation as MMC cards:

```
card_registry()
{
do { // decide RCA for each card until response time-out
          if(card is labelled as SDCombo or SDIO) { // for SDIO card like device
                    send_command(SET_RELATIVE_ADDR, 0x00, <...>); // ask SDIO card to
publish its
RCA
                    retrieve RCA from response;
          } // end if (card is labelled as SDCombo ...
          else if (card is labelled as SD) { // for SD card
                    send_command(ALL_SEND_CID, <...>);
                    if (RESP_TIMEOUT == wait_for_response(ALL_SEND_CID)) break;
                    send_command(SET_RELATIVE_ADDR, <...>);
                    retrieve RCA from response;
          } // else if (card is labelled as SD ...
          else if (card is labelled as MMC) {
                    send_command(ALL_SEND_CID, <...>);
                    rca = 0x1; // arbitrarily set RCA, 1 here for example
                    send_command(SET_RELATIVE_ADDR, 0x1 << 16, <...>); // send RCA at upper
16
bits
          } // end of else if (card is labelled as MMC ...
} while (response is not time-out);
}
```

10.3.5.3 Card Access

Information about Block Write, Block Read, Suspense Resume, ADMA Usage, Transfer Error, and Card Interrupt are detailed in the sections below.

10.3.5.3.1 Block Write

Information on Normal Write, DDR Write, and Write with Pause are detailed in the sections below.

10.3.5.3.1.1 Normal Write

During a block write (CMD24 - 27, CMD60, CMD61), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. If the CRC fails, the card shall indicate the failure on the DATA line. The transferred data will be discarded and not written, and all further transmitted blocks (in multiple block write mode) will be ignored.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card detects the block misalignment error and aborts the programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, and while ignoring all further data transfer, waits in the Receive-data-State for a stop command. The write operation is also aborted if the host tries to write over a write protected area.

For MMC and SD cards, programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card will report an error and not change any register contents.

For all types of cards, some may require long and unpredictable periods of time to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DATA line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) or other means for SDIO cards at any time, and the card will respond with its status. The responded status indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing a CMD7 (to select a different card) to place the card into the Standby State and release the DATA line without interrupting the write operation. When re-selecting the card, it will reactivate the busy indication by pulling DATA to low if the programming is still in progress and the write buffer is unavailable.

The software flow to write to a card incorporates the internal DMA and the write operation is a multi-block write with the Auto CMD12 enabled. For the other two methods (by means of external DMA or CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to write to a card is described below:

1. Check the card status, wait until the card is ready for data.
2. Set the card block length/size:

- For SD/MMC cards, use SET_BLOCKLEN (CMD16)
- For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT (CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)

3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

10.3.5.3.1.2 DDR Write

uSDHC supports dual data rate mode.

The software flow to write to a card in ddr mode is described as below:

1. Check the card status, wait until the card is ready for data.
2. For eMMC4.4 card, block length only can be set to 512byte.
3. Set the uSDHC number block register (NOB), nob is 5 (for instance).
4. Set eMMC4.4 card to high speed mode, use SWITCH(CMD6).
5. Set eMMC4.4 card bus with (4-bit /8-bit ddr mode), use SWITCH(CMD6).
6. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer.The DDR_EN bit should be set. The AC12EN bit should also be set.
7. Wait for the Transfer Complete interrupt.
8. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

10.3.5.3.1.3 Write with Pause

The write operation can be paused during the transfer. Instead of stopping the CLK at any time to pause all the operations, which is also inaccessible to the Host Driver, the Driver can set the Stop At Block Gap Request(SABGREQ) bit in the Protocol Control register to pause the transfer between the data blocks. As there is no time-out condition in a write operation during the data blocks, a write to all types of cards can be paused in this way, and if the DATA0 line is not required to de-assert to release the busy state, no suspend command is needed.

Like in the flow described in Normal Write, the write with pause is shown with the same kind of write operation:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Set the SABGREQ bit.
7. Wait for the Transfer Complete interrupt.
8. Clear the SABGREQ bit.
9. Check the status bit to see if a write CRC error occurred.
10. Set the CREQ bit to continue the write operation.
11. Wait for the Transfer Complete interrupt.
12. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

The number of blocks left during the data transfer is accessible by reading the contents of the BLKCNT field in the Block Attribute register. As the data transfer and the setting of the SABGREQ bit are concurrent, and the delay of register read and the register setting, the actual number of blocks left may not be exactly the value read earlier. The Driver shall read the value of BLKCNT after the transfer is paused and the Transfer Complete interrupt is received.

It is also possible the last block has begun when the Stop At Block Gap Request is sent to the buffer. In this case, the next block gap is actually the end of the transfer. These types of requests are ignored and the Driver should treat this as a non-pause transfer and deal with it as a common write operation.

When the write operation is paused, the data transfer inside the Host System is not stopped, and the transfer is active until the data buffer is full. Because of this (if not needed), it is recommended to avoid using the Suspend Command for the SDIO card. This is because when such a command is sent, the uSDHC thinks the System will switch to another function on the SDIO card, and flush the data buffer. The uSDHC takes the Resume Command as a normal command with data transfer, and it is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to

send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set as well as the AC12EN bit. However, the uSDHC will automatically send a CMD12 to mark the end of the multi-block transfer.

10.3.5.3.2 Block Read

Information about Normal Read, DDR Read, Read with Pause, and DLL (Delay Line) in Read Path are detailed in the sections below.

10.3.5.3.2.1 Normal Read

For block reads, the basic unit of data transfer is a block whose maximum size is stored in areas defined by the corresponding card specification. A CRC is appended to the end of each block, ensuring data transfer integrity. The CMD17, CMD18, CMD53, CMD60, CMD61, and so on, can initiate a block read. After completing the transfer, the card returns to the Transfer State. For multi blocks read, data blocks will be continuously transferred until a stop command is issued.

The software flow to read from a card incorporates the internal DMA and the read operation is a multi-block read with the Auto CMD12 enabled. For the other two methods (by means of external DMA or CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to read from a card is described below:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer read ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

10.3.5.3.2.2 DDR Read

uSDHC supports dual data rate mode.

The software flow to write to a card in ddr mode is described below:

1. Check the card status, wait until the card is ready for data.
2. For eMMC4.4 card, block length only can be set to 512byte.
3. Set the uSDHC number block register (NOB), nob is 5 (for instance).
4. Set eMMC4.4 card to high speed mode, use SWITCH(CMD6).
5. Set eMMC4.4 card bus with (4-bit /8-bit ddr mode), use SWITCH(CMD6).
6. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer.The DDR_EN bit should be set. The AC12EN bit should also be set.
7. Wait for the Transfer Complete interrupt.
8. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

10.3.5.3.2.3 Read with Pause

The read operation is not generally able to pause. Only the SDIO card (and SDCombo card working under I/O mode) supporting the Read Wait feature can pause during the read operation. If the SDIO card supports Read Wait (SRW bit in CCCR register is 1), the Driver can set the SABGREQ bit in the Protocol Control register to pause the transfer between the data blocks.

Before setting the SABGREQ bit, make sure the RWCTL bit in the Protocol Control register is set, otherwise the uSDHC will not assert the Read Wait signal during the block gap and data corruption occurs. It is recommended to set the RWCTL bit once the Read Wait capability of the SDIO card is recognized.

Like in the flow described in Normal Read, the read with pause is shown with the same kind of read operation:

1. Check the SRW bit in the CCR register on the SDIO card to confirm the card supports Read Wait.
2. Set the RWCTL bit.
3. Check the card status and wait until the card is ready for data.
4. Set the card block length/size:

- For SD/MMC, use SET_BLOCKLEN (CMD16)
- For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)

5. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
6. Set the uSDHC number block register (NOB), nob is 5 (for instance).
7. Disable the buffer read ready interrupt, configure the DMA setting and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set
8. Set the SABGREQ bit.
9. Wait for the Transfer Complete interrupt.
10. Clear the SABGREQ bit.
11. Check the status bit to see if read CRC error occurred.
12. Set the CREQ bit to continue the read operation.
13. Wait for the Transfer Complete interrupt.
14. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

Like the write operation, it is possible to meet the ending block of the transfer when paused. In this case, the uSDHC will ignore the Stop At Block Gap Request and treat it as a command read operation.

Unlike the write operation, there is no remaining data inside the buffer when the transfer is paused. All data received before the pause will be transferred to the Host System. No matter if the Suspend Command is sent or not, the internal data buffer is not flushed.

If the Suspend Command is sent and the transfer is later resumed by means of a Resume Command, the uSDHC takes the command as a normal one accompanied with data transfer. It is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set, as well as the AC12EN bit. However, the uSDHC will automatically send the CMD12 to mark the end of multi-block transfer.

10.3.5.3.2.4 DLL (Delay Line) in Read Path

The DLL(Delay Line) is newly added to assist in sampling read data. The DLL provides the ability to programmatically select a quantized delay (in fractions of the clock period) regardless of on-chip variations such as process, voltage and temperature (PVT).

The reasons why the DLL is needed for uSDHC are 1.) the path of read data traveling from card to host varies. 2.) in SD/MMC DDR mode the minimum input setup and hold time are both at 2.5 ns. The data sampling window is so small that the delay of loopback clock needs to be accurate and consistent regardless of PVT. The DLL takes the divided card_clk as the reference clock and loopback clock as the input clock. It then generates a delayed version of the input clock according to the programmed target delay.

The DLL can be disabled or bypassed, and it can also be manually set for a fixed delay in override mode. The override value set is the number of delay cells. In override mode, there is no need to set the DLL_enable. Another DLL mode is target value mode. In this mode, the DLL will automatically adjust the number of delay cells according to target value set by user and PVT changes. Be aware that target value is in units of 1/32 of the clock reference period. If the card_clk is 100Mhz, then the reference clock period is 10ns, setting target vaule of 16 means 5ns =(16/32)*10ns. Software can disable automatic update by setting dll_gate_update bit. Please refer to Figure 10-54.

Since the user may change the frequency of the card_clk from time to time by changing SDCLKFS[7:0]/DVS[3:0], the software must adjust the delay value to ensure it works correctly when the reference clock (card_clk) is changed. The following is the correct flow, which should be ignored if DLL_CTRL_ENABLE is not set.

Step 1: Set DLL_CTRL_RESET bit

Step 2: Configure the SDCLKFS[7:0] and DVS[3:0]

Step 3: Wait until SDSTB is asserted

Step 4: clear DLL_CTRL_RESET bit

Step 5: Wait until both DLL_STS_SLV_LOCK and DLL_STS_REF_LOCK are asserted

Step 6: set DLL_CTRL_SLV_FORCE_UPD

Step 7: clear DLL_CTRL_SLV_FORCE_UPD

NOTE:Software should make sure the DLL_CTRL_SLV_FORCE_UPD is lasted for at least one card_clk. So software may need to add some delay between step6 and step7.

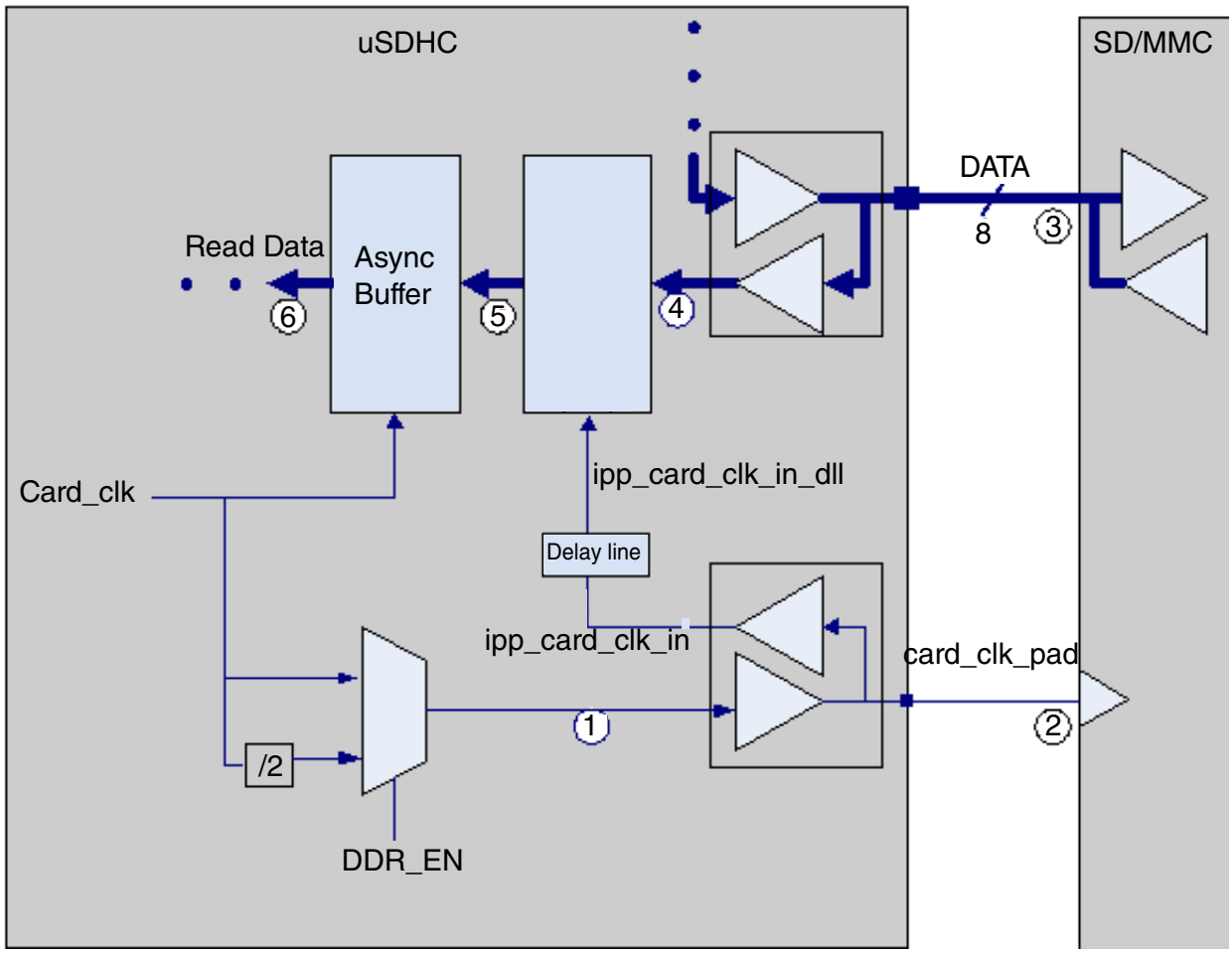


Figure 10-54. DLL(Delay Line) in Read Path

10.3.5.3.3 Suspend Resume

The uSDHC supports the Suspend Resume operations of SDIO cards, although slightly differently than the suggested implementation of Suspend in the SDIO card specification.

10.3.5.3.3.1 Suspend

After setting the SABGREQ bit, the Host Driver may send a Suspend command to switch to another function of the SDIO card. The uSDHC does not monitor the content of the response, so it doesn't know if the Suspend command succeeded or not. Accordingly, it doesn't de-assert Read Wait for read pause. To solve this problem, the Driver shall not mark the Suspend command as a "Suspend", (i.e. setting the CMDTYP bits to 01). Instead, the Driver shall send this command as if it were a normal command, and only when the command succeeds, and the BS bit is set in the response, can the Driver send another command marked as "Suspend" to inform the uSDHC that the current transfer is suspended. As shown in the following sequence for Suspend operation:

1. Set the SABREQ bit to pause the current data transfer at block gap.
2. After the BGE bit is set, send the Suspend command to suspend the active function. The CMDTYP bit field must be 2'b00.
3. Check the BS bit of the CCCR in the response. If it is 1, repeat this step until the BS bit is cleared or abandon the suspend operation according to the Driver strategy.
4. Send another normal I/O command to the suspended function. The CMDTYP of this command must be 2'b01, so the uSDHC can detect this special setting and be informed that the paused operation has successfully suspended. If the paused transfer is a read operation, the uSDHC stops driving DATA2 and goes to the idle state.
5. Save the context registers in the system memory for later use, including the DMA System Address Register (for internal DMA operation), and the Block Attribute Register.
6. Begin operation for another function on the SDIO card.

10.3.5.3.3.2 Resume

To resume the data transfer, a Resume command shall be issued:

1. To resume the suspended function, restore the context register with the saved value in step #5 of the Suspend operation above.
2. Send the Resume command. In the Transfer Type register, all bit fields are set to the value as if this were another ordinary data transfer, instead of a transfer resume (except the CMDTYP is set to 2'b10).
3. If the Resume command has responded, the data transfer will be resumed.

10.3.5.3.4 ADMA Usage

To use the ADMA in a data transfer, the Host Driver must prepare the correct descriptor chain prior to sending the read/write command.

The steps to prepare the correct descriptor chain are:

1. Create a descriptor to set the data length that the current descriptor group is about to transfer. The data length should be even numbers of the block size.
2. Create another descriptor to transfer the data from the address setting in this descriptor. The data address must be at a page boundary (4kB address aligned).
3. If necessary, create a Link descriptor containing the address of the next descriptor. The descriptor group is created in steps 1 ~ 3.
4. Repeat steps 1 ~ 3 until all descriptors are created.
5. In the last descriptor, set the End flag to 1 and make sure the total length of all descriptors match the product of the block size and block number configured in the Block Attribute Register.

6. Set the ADMA System Address Register to the address of the first descriptor and set the DMAS field in the Protocol Control Register to 01 to select the ADMA.
7. Issue a write or read command with the DMAEN bit set to 1 in the Transfer Type Register.

Steps 1 ~ 5 are independent of step 6, so step 6 can finish before steps 1 ~ 5. Regarding the descriptor configuration, it is recommended not to use the Link descriptor as it requires extra system memory access.

10.3.5.3.5 Transfer Error

Information about CRC, Internal DMA, Transfer ADMA, and Auto CMD12 Errors are detailed in the sections below.

10.3.5.3.5.1 CRC Error

It is possible at the end of a block transfer, that a write CRC status error or read CRC error occurs. For this type of error the latest block received shall be discarded. This is because the integrity of the data block is not guaranteed. It is recommended to discard the following data blocks and re-transfer the block from the corrupted one.

For a multi-block transfer, the Host Driver shall issue a CMD12 to abort the current process and start the transfer by a new data command. In this scenario, even when the AC12EN and BCEND bits are set, the uSDHC does not automatically send a CMD12 because the last block is not transferred. On the other hand, if it is within the last block that the CRC error occurs, an Auto CMD12 will be sent by the uSDHC. In this case, the Driver shall re-send or re-obtain the last block with a single block transfer.

10.3.5.3.5.2 Internal DMA Error

During the data transfer with internal Simple DMA, if the DMA engine encounters some error on the AHB bus, the DMA operation is aborted and DMA Error interrupt is sent to the Host System. When acknowledged by such an interrupt, the Driver shall calculate the start address of data block in which the error occurs.

The start address can be calculated by either:

1. Reading the DMA System Address register. The error occurs during the previous burst. Taking the block size, the previous burst length and the start address of the next burst transfer into account, it is straight forward to obtain the start address of the corrupted block.

2. Reading the BLKCNT field of the Block Attribute register. By the number of blocks left, the total number to transfer, the start address of transfer, and the size of each block, the start address of corrupted block can be determined. When the BCEN bit is not set, the contents of the Block Attribute register does not change, so this method does not work.

When a DMA error occurs, it is recommended to abort the current transfer by means of a CMD12 (for multi block transfer), apply a reset for data, and re-start the transfer from the corrupted block to recover from the error.

10.3.5.3.5.3 Transfer ADMA Error

There are 3 kinds of possible ADMA errors. The AHB transfer, invalid descriptor, and data-length mismatch errors. Whenever these errors occur, the DMA transfer stops and the corresponding error status bit is set.

For acknowledging the status, the Host Driver should recover the error as shown below and re-transfer from the place of interruption.

1. AHB transfer error: Such errors may occur during data transfer or descriptor fetch. For either scenario, it is recommended to retrieve the transfer context, reset for the data part and re-transfer the block that was corrupted, or the next block if no block is corrupted.
2. Invalid descriptor error: For such errors, it is recommended to retrieve the transfer context, reset for the data part and re-create the descriptor chain from the invalid descriptor and issue a new transfer. As the data to transfer now may be less than the previous setting, the data length configured in the new descriptor chain should match the new value.
3. Data-length mismatch error: It is similar to recover from this error. The Host Driver polls relating registers to retrieve the transfer context, apply a reset for the data part, configure a new descriptor chain, and make another transfer if there is data left. Like the previous scenario of the invalid descriptor error, the data length must match the new transfer.

10.3.5.3.5.4 Auto CMD12 Error

After the last block of the multi block transfer is sent or received, and the AC12EN bit is set when the data transfer is initiated by the data command, the uSDHC automatically sends a CMD12 to the card to stop the transfer.

When errors with this command occur, it is recommended to the Driver to deal with the situations in the following manner:

1. Auto CMD12 response time-out. It is not certain whether the command is accepted by the card or not. The Driver should clear the Auto CMD12 error status bits and re-send the CMD12 until it is accepted by the card.
2. Auto CMD12 response CRC error. Since card responds to the CMD12, the card will abort the transfer. The Driver may ignore the error and clear the error status bit.
3. Auto CMD12 conflict error or not sent. The command is not sent, so the Driver shall send a CMD12 manually.

10.3.5.3.6 Card Interrupt

The external cards can inform the Host Controller by means of some special signals. For the SDIO card, it can be the low level on the DATA1 line during some special period. The uSDHC only monitors the DATA1 line and supports the SDIO interrupt.

When the SDIO interrupt is captured by the uSDHC, and the Host System is informed by the uSDHC asserting the uSDHC interrupt line, the interrupt service from the Host Driver is called.

As the interrupt source is controlled by the external card, the interrupt from the SDIO card must be serviced before the CINT bit is cleared by written. Refer to Card Interrupt Handling for the card interrupt handling flow.

10.3.5.4 Switch Function

A switch command shall be issued by the Host Driver to enable new features added to the SD/MMC spec. SD/MMC cards can transfer data at bus widths other than 1-bit. Different speed mode are also defined. To enable these features, a switch command shall be issued by the Host Driver.

For SDIO cards, the high speed mode/DDR50/SDR50/SDR104 are enabled by writing the EHS bit in the CCCR register after the SHS bit is confirmed. For SD cards, the high speed mode/DDR50/SDR50/SDR104 are queried and enabled by a CMD6 (with the mnemonic symbol as SWITCH_FUNC). For MMC cards , the high speed mode/HS400 are queried by a CMD8 and enabled by a CMD6 (with the mnemonic symbol as SWITCH).

The SDR4-bit, SDR8-bit ,DDR4-bit and DDR8-bit width of the MMC is also enabled by the SWITCH command, but with a different argument.

These new functions can also be disabled by a software reset. For SDIO cards it can be done by setting the RES bit in the CCCR register. For other cards, it can be accomplished by issuing a CMD0. This method of restoring to the normal mode is not recommended because a complete identification process is needed before the card is ready for data transfer.

For the sake of simplicity, the following pseudocode examples do not show current capability check, which is recommended in the function switch process.

10.3.5.4.1 Query, Enable and Disable SDIO High Speed Mode

```
enable_sdio_high_speed_mode(void)
{
send CMD52 to query bit SHS at address 0x13;
if (SHS bit is '0') report the SDIO card does not support high speed mode and return;
send CMD52 to set bit EHS at address 0x13 and read after write to confirm EHS bit is set;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 50MHz;
(data transactions like normal peers)
}
disable_sdio_high_speed_mode(void)
{
send CMD52 to clear bit EHS at address 0x13 and read after write to confirm EHS bit is
cleared;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

10.3.5.4.2 Query, Enable and Disable SD High Speed Mode/SDR50/ SDR104/DDR50

```
enable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0xFFFFFx and read 64 bytes of data accompanying the R1 response;
(high speed mode,x=1; SDR50,x=2; SDR104,x=3; DDR50,x=4;)
wait data transfer done bit is set;
check if the bit x of received 512 bits is set;
if (bit 401 is '0') report the SD card does not support high speed mode and return;
if (bit 402 is '0') report the SD card does not support SDR50 mode and return;
if (bit 403 is '0') report the SD card does not support SDR104 mode and return;
if (bit 404 is '0') report the SD card does not support DDR50 mode and return;
send CMD6, with argument 0x80FFFFFx and read 64 bytes of data accompanying the R1 response;
(high speed mode,x=1; SDR50,x=2; SDR104 x=3; DDR50 x=4;)
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 50MHz for high speed mode, 100Mhz for SDR50, 200Mhz for SDR104, 50Mhz for
DDR50;
(data transactions like normal peers)
}
disable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0x80FFFFF0 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 25MHz;
```

```
(data transactions like normal peers)
}
```

10.3.5.4.3 Query, Enable and Disable MMC High Speed Mode

```
enable_mmc_high_speed_mode(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support high speed mode and
return;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
extract the value of CARD_TYPE field to check the 'high speed mode' in this MMC is 26MHz or
52MHz;
send CMD6 with argument 0x1B90100;
send CMD13 to wait card ready (busy line released);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report MMC switching to high speed mode failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 26MHz or 52MHz according to the CARD_TYPE;
(data transactions like normal peers)
}
disable_mmc_high_speed_mode(void)
{
send CMD6 with argument 0x2B90100;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 0;
if (HS_TIMING is not 0) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 20MHz;
(data transactions like normal peers)
}
```

10.3.5.4.4 Set MMC Bus Width

```
change_mmc_bus_width(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support multiple bit width and
return;
send CMD6 with argument 0x3B70x00; (8-bit(dual data rate), x=6; 4-bit(dual data rate), x=5;8-
bit, x=2; 4-bit, x=1; 1-bit, x=0)
send CMD13 to wait card ready (busy line released);
(data transactions like normal peers)
}
```

10.3.5.5 ADMA Operation

Code for ADMA1 Operation and ADMA2 Operation can be found here.

10.3.5.5.1 ADMA1 Operation

```
Set_adma1_descriptor
{
if (to start data transfer) {
// Make sure the address is 4KB align.
```

```
Set 'Set' type descriptor;
{
Set Act bits to 01;
Set [31:12] bits data length (byte unit);
}
Set 'Tran' type descriptor;
{
Set Act bits to 10;
Set [31:12] bits address (4KB align);
}
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to 11;
Set [31:12] bits the next descriptor address (4KB aligned);
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set End bit to 1;
}
if (to generate interrupt for this descriptor) {
Set Int bit to 1;
}
Set Valid bit to 1;
}
```

10.3.5.5.2 ADMA2 Operation

```
Set_adma2_descriptor
{
if (to start data transfer) {
// Make sure the address is a 32-bit boundary (lower 2-bit are always '00').
Set higher 32-bit of descriptor for this data transfer initial address;
Set [31:16] bits data length (byte unit);
Set Act bits to '10';
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to '11';
// Make sure the address is 32-bit boundary (lower 2-bit are always set to '00').
Set higher 32-bit of descriptor for the next descriptor address;
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set 'End' bit '1';
}
if (to generate interrupt for this descriptor) {
Set 'Int' bit '1';
}
Set the 'Valid' bit to '1';
}
```

10.3.5.6 Fast Boot Operation

10.3.5.6.1 Normal fast boot flow

1. Software must configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.

2. Software must configure the MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 0 (normal fast boot), and bit 4 to select the ack mode or not. If the data will be sent through DMA mode, the software should configure bit 7 to enable the automatic stop at block gap feature, and configure bit 3-bit 0 to select the ack timeout value according to the SD CLK frequency.
3. Software then needs to configure the Block Attributes Register to set the block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software must configure the Protocol control register to set DTW (data transfer width). If in DDR fast boot mode, DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software needs to configure the Command Argument Register to set argument if needed (no need in normal fast boot).
6. Software must configure the Transfer Type Register to start the boot process. In normal boot mode, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept at the default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1.
7. DMAEN should be configured as 0 in polling mode. And if BCEN is configured as 1, it is recommended to configure the number of blocks in the Block Attributes Register to the maximum value. If in DDR fast boot mode, DDR_EN needs to be set to 1.
8. When the step 6 is configured, the boot process will begin. Software needs to poll the data buffer ready status to read the data from the buffer in time. If a boot timeout happens (ack times out or the first data read times out), an interrupt will be triggered, and software must configure MMC Boot Register to bit 6 to 0 to disable boot. This makes CMD high, then after at least 56 clocks, it is ready to begin a normal initialization process.
9. If there is no timeout, software needs to determine when the data read is finished and then configure MMC Boot Register bit 6 to 0 to disable boot. This will make CMD line high and command completed asserted. After at least 56 clocks, it is ready to begin normal initialization process.
10. Reset the host and then can begin the normal process.

10.3.5.6.2 Alternative fast boot flow

1. Software needs to configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software needs to configure MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 1 (alternative boot), and bit 4 to select the ack mode or not. If data needs to be sent through DMA mode, then configure bit 7 to enable the automatic stop at block gap feature. Software should also configure bit 3-bit 0 to select the ack timeout value according to the SD clock frequency.

3. Software then needs to configure Block Attributes Register to set the block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software needs to configure the Protocol control register to set the DTW (data transfer width). If in ddr fast boot mode, DTW only can be configure to 4-bit/8-bit dataline mode.
5. Software needs to configure Command Argument Register to set argument to 0xFFFFFFFA.
6. Software needs to configure the Transfer Type Register to start the boot process by CMD0 with 0xFFFFFFFA argument . In alternative boot, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1. Note DMAEN should be configured as 0 in polling mode. And if BCEN is configured as 1 in polling mode, it is recommended to configure the block count in the Block Attributes Register to the maximum value. If in DDR fast boot mode, DDR_EN needs to be set to 1.
7. When the step 6 is configured, the boot process will begin. Software needs to poll the data buffer ready status to read the data from the buffer in time. If there is a boot timeout (ack data timeout in 50ms or data timeout in 1s), the host will send out the interrupt and software needs to send CMD0 with reset and then configure the boot enable bit to 0 to stop this process..
8. If there is no time out, software needs to decide when to stop the boot process, and send out the CMD0 with reset and then after the command is completed, configure the MMC Boot Register bit 6 to stop the process. After 8 clocks from the command completion, the slave (card) is ready for the identification step.
9. Reset the host and then begin the normal process.

10.3.5.6.3 Fast boot application case (in DMA mode)

In the boot application case, because the image destination and the image size are contained in the beginning of the image, it is necessary to switch DMA parameters on the fly during MMC fast boot.

In fast boot, the host can use ADMA2 (Advanced DMA2) with two destinations.

The detail flow is described below:

1. Software needs to configure INIT_ACTIVE bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software needs to configure the MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot); and bit 5 to 0 (normal fast boot) or 1 (alternative boot); and bit 4 to select the ack mode or not. In DMA mode, configure bit 7 to 1 to enable the automatic stop at block gap feature. Also configure bits[31-16] to set the (BLK_CNT - VALUE1).Here VALUE1 is the value of the block count that needs to transfer the

first time, so that that the hostwill stop at the block gap when the uSDHC controller gets VAULE1 blocks from the device. Also configure bits[3-0] to select the ack timeout value according to the SD clock frequency.

3. Software then needs to configure the Block Attributes Register to set block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes. In DMA mode, it is recommended to set the block count (BLK_CNT) to the max value (16'hffff).
4. Software needs to configure Protocol Control Register to set DTW (data transfer width). If in DDR fast boot mode, the DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software enbable ADMA2 by configuring Protocol Control Register bits [9-8].
6. Software need to set at least three pairs ADMA2 descriptor in boot memory (ie, in IRAM, at least 6 word). The first pair descriptor define the start address (ie, IRAM)and data length(ie,512byte*VALUE1) of fisrt part boot code. Software also need to set the second pair descriptor, the second start address (any value that is writeable), data length is suggest to set 1~2word(record as VALUE2). Note: the second couple desc also transfer useful data even at lease 1 word. Because our ADMA2 can't support 0 data_length data transfer descriptor.
7. Software needs to configure Command Argument Register to set argument to 0xFFFFFFFA in alternative fast boot, and don't need set in normal fast boot.
8. Software needs to configure Transfer Type Register to start the boot process . CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1. DMAEN is configured as 1 in DMA mode. And if BCEN is configured as 1, better to configure blk no in Bock Attributes Register to the max value. And if in ddr fast boot mode, DDR_EN need to be set to 1.
9. When the step 8 is configured, boot process will begin, the first VALUE1 block number data has transfer. Software need to polling TC bit (bit1 in Interrupt Status Register) to determine first transfer is end. Also software need to polling BGE bit (bit2 in Interrupt Status Register) to determine if first transfer stop at block gap.
10. When TC, BGE bit is 1, . SW can analyzes the first code of VALUE1 block, initializes the new memory device, if required, and sets the third pair of descriptors to define the start address and length of the remaining part of boot code(VALUE3 the remain boot code block). Remeber set the last descriptor with END.
11. Software needs to configure MMC Boot Register (offset 0xc4) again. Set bit 6 to 1(enable boot); and bit 5 to 0(normal fast boot), to 1(alternative boot); and bit 4 to select the ack mode or not. In DMA mode, configure bit 7 to 1 for enable automatically stop at block gap feature. Also configure bit31-bit16 to set the *(BLK_CNT - (VALUE1+1+VALUE3))*, that host will stop at block gap when *the uSDHC controller gets (VALUE1+1+VALUE3)) blocks from device totally include the blocks received in step 9*. And need to configure bit 3-bit0 to select the ack

timeout value according to the sd clk frequence. Please note, Software doesn't need to configure the *BLK_CNT* again, because it's counted down automatically by the uSDHC controller.

12. Software needs to clear TC and BGE bit. And software needs to clear SABGREQ(bit 16 in Protocol control register), and set CREQ(bit17 Protocol control register) to 1 to resume the data transfer. Host will transfer the VALUE2 and VALUE3 data to the destination that is set by descriptor.
13. Software need to polling BGE bit to determine if the fast boot is over.

Note:

1. When ADMA boot flow is started, for uSDHC, it is like a normal ADMA read operation. So setting ADMA2 descriptor as the normal ADMA2 transfer.
2. Need a few words length memory to keep descriptor.
3. For the 1~2 word data in second descriptor setting, it is the useful data, so software need to deal the data due to the application case.

10.3.6 Commands for MMC/SD/SDIO

A table containing the list of commands for the MMC/SD/SDIO cards can be found here.

Refer to the corresponding specifications for more details about the command information.

There are four kinds of commands defined to control the MultiMediaCard:

1. broadcast commands (bc), no response.
2. broadcast commands with response (bcr), response from all cards simultaneously.
3. addressed (point-to-point) commands (ac), no data transfer on the DATA.
4. addressed (point-to-point) data transfer commands (adtc).

Response: a response is a token which is sent from the card to the host as an answner to a previously received command. A response is transferred serially on the CMD line.

Table 10-42. Commands for MMC/SD/SDIO Cards

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all MMC and SD memory cards to idle state.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all MMC and SD Memory cards in idle state to send their operation conditions register contents in the response on the CMD line.

Table continues on the next page...

Table 10-42. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line.
CMD3[1]	ac	[31:6] RCA [15:0] stuff bits	R1 R6 (SDIO)	SET/ SEND_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:0] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards.
CMD5	bc	[31:0] OCR without busy	R4	IO_SEND_OP_COND	Asks all SDIO cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD6[2]	adtc	[31] Mode 0: Check function 1: Switch function [30:8] Reserved for function groups 6 ~ 3 (All 0 or 0xFFFF) [7:4] Function group1 for command system [3:0] Function group2 for access mode	R1	SWITCH_FUNC	Checks switch ability (mode 0) and switch card function (mode 1). Refer to "SD Physical Specification V1.1" for more details.
CMD6[3]	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Switches the mode of operation of the selected card or modifies the EXT_CSD registers. Refer to "The MultiMediaCard System Specification Version 4.0 Final draft 2" for more details.
CMD7	ac	[31:6] RCA [15:0] stuff bits	R1b	SELECT/ DESELECT_CARD	Toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address. Address 0 deselects all.
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data, with a block size of 512 bytes.
CMD9	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card-identification (CID) on the CMD line.

Table continues on the next page...

Table 10-42. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission.
CMD13	ac	[31:6] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	Reserved				
CMD15	ac	[31:6] RCA [15:0] stuff bits	-	GO_INACTIVE_STATE	Sets the card to inactive state in order to protect the card stack against communication breakdowns.
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command.
CMD19	Reserved				
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	Writes data stream from the host, starting at the given address, until a STOP_TRANSMISION follows.
CMD21-23	Reserved				
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command shall be issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).

Table continues on the next page...

Table 10-42. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.
CMD31	Reserved				
CMD32	ac	[31:0] data address	R1	TAG_SECTOR_START	Sets the address of the first sector of the erase group.
CMD33	ac	[31:0] data address	R1	TAG_SECTOR_END	Sets the address of the last sector in a continuous range within the selection of a single sector to be selected for erase.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection.
CMD35	ac	[31:0] data address	R1	TAG_ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	TAG_ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection.
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erase all previously selected sectors.
CMD39	ac	[31:0] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command addresses a card, and a register, and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This command accesses application dependent registers which are not defined in the MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode.
CMD41	Reserved				
CDM42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43~51	Reserved				

Table continues on the next page...

Table 10-42. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD52	ac	[31:0] stuff bits	R5	IO_RW_DIRECT	Access a single register within the total 128k of register space in any I/O function.
CMD53	ac	[31:0] stuff bits	R5	IO_RW_EXTENDED	Accesses a multiple I/O register with a single command. Allows the reading or writing of a large number of I/O registers.
CMD54	Reserved				
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather that a standard command.
CMD56	adtc	[31:1] stuff bits [0]: RD/WR	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block is set by the SET_BLOCK_LEN command.
CMD57-59	Reserved				
CMD60	adtc	[31] WR [30:24] stuff bits [23:16] address [15:8] stuff bits [7:0] byte count	R1b	RW_MULTIPLE_REGISTER	These registers are used to control the behavior of the device and to retrieve status information regarding the operation of the device. All Status and Control registers are WORD (32-bit) in size and are WORD aligned. CMD60 shall be used to read and write these registers.
CMD61	adtc	[31] WR [30:16] stuff bits [15:0] data unit count	R1b	RW_MULTIPLE_BLOCK	The host issues a RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer.
CMD62-63	Reserved				
ACMD6[4]	ac	[31:2] stuff bits [1:0] bus width	R1	SET_BUS_WIDTH	Defines the data bus width ('00'=1bit or '10'=4bit bus) to be used for data transfer. The allowed data bus widths are given in SCR register.
ACMD13[4]	adtc	[31:0] stuff bits	R1	SD_STATUS	Send the SD Memory Card status.
ACMD22[4]	adtc	[31:0] stuff bits	R1	SEND_NUM_WR_SECTORS	Send the number of the written sectors (without errors). Responds with 32-bit plus the CRC data block.
ACMD23[4]	ac	[31:23] stuff bits [22:0] Number of blocks	R1	SET_WR_BLK_ERASE_COUNT	Set the number of write blocks to be pre-erased before writing (to be used for fast Multiple Block WR command). "1"=default(one write block).

Table continues on the next page...

Table 10-42. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
ACMD41[4]	bcr	[31:0] OCR	R3	SD_APP_OP_COND	Asks the accessed card to send its operating condition register (OCR) contents in the response on the CMD line.
ACMD42[4]	ac	[31:1] stuff bits [0] set_cd	R1	SET_CLR_CARD_DETECT	Connect(1)/Disconnect(0) the 50KOhm pull-up resistor on CD_B/DATA3 of the card.
ACMD51[4]	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register (SCR).

1. CMD3 differs for MMC and SD cards. For MMC cards, it is referred to as SET_RELATIVE_ADDR, with a response type of R1. For SD cards, it is referred to as SEND_RELATIVE_ADDR, with a response type of R6 (with RCA inside).
2. CMD6 differs completely between high speed MMC cards and high speed SD cards. Command SWITCH_FUNC is for high speed SD cards.
3. Command SWITCH is for high speed MMC cards . The Index field can contain any value from 0-255, but only values 0-191 are valid. If the Index value is in the 192-255 range the card does not perform any modification and the SWITCH_ERROR status bit in the EXT_CSD register is set. The Access Bits are shown in Table 2.
4. ACMDs shall be preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this module).

The Access Bits for the EXT_CSD Access Modes are shown below.

Table 10-43. EXT_CSD Access Modes

Bits	Access Name	Operation
00	Command Set	The command set is changed according to the Cmd Set field of the argument
01	Set Bits	The bits in the pointed byte are set, according to the 1 bits in the Value field.
10	Clear Bits	The bits in the pointed byte are cleared, according to the 1 bits in the Value field.
11	Write Byte	The Value field is written into the pointed byte.

10.3.7 Software Restrictions

10.3.7.1 Initialization Active

The driver cannot set INITA bit in System Control register when any of the command line or data lines is active, so the driver must ensure both CDIHB and CIHB bits are cleared.

10.3.7.2 Software Polling Procedure

For polling read or write, once the software begins a buffer read or write, it must access exactly the number of times as the values set in the Watermark Level Register; moreover, if the block size is not a multiple of the value in Watermark Level Register (read and write respectively), the software must access exactly the remaining number of words at the end of each block.

For example, for a read operation, if the RD_WML is 4, indicating the watermark level is 16 bytes, block size is 40 bytes, and the block number is 2, then the access times for the burst sequence in the whole transfer process must be 4, 4, 2, 4, 4, 2.

10.3.7.3 Suspend Operation

In order to suspend the data transfer, the software must inform uSDHC that the suspend command is successfully accepted. To achieve this, after the Suspend command is accepted by the SDIO card, software must send another normal command marked as suspend command (CMDTYP bits set as '01') to inform uSDHC that the transfer is suspended.

If software needs to resume the suspended trasnfer, it should read the value in BLKCNT register to save the remaining number of blocks before sending the normal command marked as suspend, otherwise on sending such 'suspend' command, uSDHC will regard the current transfer is aborted and change BLKCNT register to its original value, instead of keeping the remained number of blocks.

10.3.7.4 Data Length Setting

For either ADMA (ADMA1 or ADMA2) transfer, the data in the data buffer must be word aligned, so the data length set in the descriptor must be a multiple of 4.

10.3.7.5 (A)DMA Address Setting

To configure ADMA1/ADMA2/DMA address register, when TC bit is set, the register will always update itself with the internal address value to support dynamic address synchronization, so the software must ensure that the TC bit is cleared prior to configuring ADMA1/ADMA2/DMA address register.

10.3.7.6 Data Port Access

Data Port does not support parallel access. For example, during an external DMA access, it is not allowed to write any data to the Data Port by CPU; or during a CPU read operation, it is also prohibited to write any data to the Data Port, by either CPU or external DMA. Otherwise the data would be corrupted inside the uSDHC buffer.

10.3.7.7 Change Clock Frequency

uSDHC does not automatically gate off the card clock when the Host Driver changes the clock frequency. To prevent possible glitch on the card clock, clear the FRC_SDCLK_ON bit when changing clock divisor value(SDCLKFS or DVS in System Control Register) or setting RSTA bit.

Also before changing the clock divisor value, Host Driver should make sure the SDSTB bit is high.

10.3.7.8 Multi-block Read

For pre-defined multi-block read operation, i.e., the number of blocks to read has been defined by previous CMD23 for MMC, or pre-defined number of blocks in CMD53 for SDIO/SDCombo, or whatever multi-block read without abort command at card side, an abort command, either automatic or manual CMD12/CMD52, is still required by uSDHC after the pre-defined number of blocks are done, to drive the internal state machine to idle mode.

In this case, the card may not respond to this extra abort command and uSDHC will get Response Timeout. It is recommended to manually send an abort command with RSPTYP[1:0] both bits cleared.

10.3.8 uSDHC Memory Map/Register Definition

This section includes the module memory map and detailed descriptions of all registers.

See the table below for the register memory map for the uSDHC. All these registers only support 32-bit accesses.

NOTE

The uSDHC registers are 32-bit wide and only support 32-bit access.

uSDHC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30B4_0000	DMA System Address (uSDHC1_DS_ADDR)	32	R/W	0000_0000h	10.3.8.1/ 2777
30B4_0004	Block Attributes (uSDHC1_BLK_ATT)	32	R/W	0000_0000h	10.3.8.2/ 2778
30B4_0008	Command Argument (uSDHC1_CMD_ARG)	32	R/W	0000_0000h	10.3.8.3/ 2779
30B4_000C	Command Transfer Type (uSDHC1_CMD_XFR_TYP)	32	R/W	0000_0000h	10.3.8.4/ 2780
30B4_0010	Command Response0 (uSDHC1_CMD_RSP0)	32	R	0000_0000h	10.3.8.5/ 2783
30B4_0014	Command Response1 (uSDHC1_CMD_RSP1)	32	R	0000_0000h	10.3.8.6/ 2784
30B4_0018	Command Response2 (uSDHC1_CMD_RSP2)	32	R	0000_0000h	10.3.8.7/ 2784
30B4_001C	Command Response3 (uSDHC1_CMD_RSP3)	32	R	0000_0000h	10.3.8.8/ 2785
30B4_0020	Data Buffer Access Port (uSDHC1_DATA_BUFF_ACC_PORT)	32	R/W	0000_0000h	10.3.8.9/ 2786
30B4_0024	Present State (uSDHC1_PRES_STATE)	32	R	0000_8080h	10.3.8.10/ 2786
30B4_0028	Protocol Control (uSDHC1_PROT_CTRL)	32	R/W	0880_0020h	10.3.8.11/ 2792
30B4_002C	System Control (uSDHC1_SYS_CTRL)	32	R/W	8080_800Fh	10.3.8.12/ 2797
30B4_0030	Interrupt Status (uSDHC1_INT_STATUS)	32	w1c	0000_0000h	10.3.8.13/ 2800
30B4_0034	Interrupt Status Enable (uSDHC1_INT_STATUS_EN)	32	R/W	0000_0000h	10.3.8.14/ 2806

Table continues on the next page...

uSDHC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30B4_0038	Interrupt Signal Enable (uSDHC1_INT_SIGNAL_EN)	32	R/W	0000_0000h	10.3.8.15/ 2809
30B4_003C	Auto CMD12 Error Status (uSDHC1_AUTOCMD12_ERR_STATUS)	32	R	0000_0000h	10.3.8.16/ 2812
30B4_0040	Host Controller Capabilities (uSDHC1_HOST_CTRL_CAP)	32	R	07F3_B407h	10.3.8.17/ 2815
30B4_0044	Watermark Level (uSDHC1_WTMK_LVL)	32	R/W	0810_0810h	10.3.8.18/ 2818
30B4_0048	Mixer Control (uSDHC1_MIX_CTRL)	32	R/W	8000_0000h	10.3.8.19/ 2819
30B4_0050	Force Event (uSDHC1_FORCE_EVENT)	32	W (always reads 0)	0000_0000h	10.3.8.20/ 2821
30B4_0054	ADMA Error Status Register (uSDHC1_ADMA_ERR_STATUS)	32	R	0000_0000h	10.3.8.21/ 2824
30B4_0058	ADMA System Address (uSDHC1_ADMA_SYS_ADDR)	32	R/W	0000_0000h	10.3.8.22/ 2826
30B4_0060	DLL (Delay Line) Control (uSDHC1_DLL_CTRL)	32	R/W	0000_0200h	10.3.8.23/ 2827
30B4_0064	DLL Status (uSDHC1_DLL_STATUS)	32	R	0000_0000h	10.3.8.24/ 2829
30B4_0068	CLK Tuning Control and Status (uSDHC1_CLK_TUNE_CTRL_STATUS)	32	R/W	0000_0000h	10.3.8.25/ 2830
30B4_0070	Strobe DLL Control (uSDHC1_STROBE_DLL_CTRL)	32	R/W	0000_0000h	10.3.8.26/ 2832
30B4_0074	Strobe DLL Status (uSDHC1_STROBE_DLL_STATUS)	32	R	0000_0000h	10.3.8.27/ 2834
30B4_00C0	Vendor Specific Register (uSDHC1_VEND_SPEC)	32	R/W	2000_7809h	10.3.8.28/ 2836
30B4_00C4	MMC Boot Register (uSDHC1_MMC_BOOT)	32	R/W	0000_0000h	10.3.8.29/ 2839
30B4_00C8	Vendor Specific 2 Register (uSDHC1_VEND_SPEC2)	32	R/W	0000_0006h	10.3.8.30/ 2840
30B4_00CC	Tuning Control Register (uSDHC1_TUNING_CTRL)	32	R/W	0021_2800h	10.3.8.31/ 2842
30B5_0000	DMA System Address (uSDHC2_DS_ADDR)	32	R/W	0000_0000h	10.3.8.1/ 2777
30B5_0004	Block Attributes (uSDHC2_BLK_ATT)	32	R/W	0000_0000h	10.3.8.2/ 2778
30B5_0008	Command Argument (uSDHC2_CMD_ARG)	32	R/W	0000_0000h	10.3.8.3/ 2779
30B5_000C	Command Transfer Type (uSDHC2_CMD_XFR_TYP)	32	R/W	0000_0000h	10.3.8.4/ 2780
30B5_0010	Command Response0 (uSDHC2_CMD_RSP0)	32	R	0000_0000h	10.3.8.5/ 2783

Table continues on the next page...

uSDHC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30B5_0014	Command Response1 (uSDHC2_CMD_RSP1)	32	R	0000_0000h	10.3.8.6/ 2784
30B5_0018	Command Response2 (uSDHC2_CMD_RSP2)	32	R	0000_0000h	10.3.8.7/ 2784
30B5_001C	Command Response3 (uSDHC2_CMD_RSP3)	32	R	0000_0000h	10.3.8.8/ 2785
30B5_0020	Data Buffer Access Port (uSDHC2_DATA_BUFF_ACC_PORT)	32	R/W	0000_0000h	10.3.8.9/ 2786
30B5_0024	Present State (uSDHC2_PRES_STATE)	32	R	0000_8080h	10.3.8.10/ 2786
30B5_0028	Protocol Control (uSDHC2_PROT_CTRL)	32	R/W	0880_0020h	10.3.8.11/ 2792
30B5_002C	System Control (uSDHC2_SYS_CTRL)	32	R/W	8080_800Fh	10.3.8.12/ 2797
30B5_0030	Interrupt Status (uSDHC2_INT_STATUS)	32	w1c	0000_0000h	10.3.8.13/ 2800
30B5_0034	Interrupt Status Enable (uSDHC2_INT_STATUS_EN)	32	R/W	0000_0000h	10.3.8.14/ 2806
30B5_0038	Interrupt Signal Enable (uSDHC2_INT_SIGNAL_EN)	32	R/W	0000_0000h	10.3.8.15/ 2809
30B5_003C	Auto CMD12 Error Status (uSDHC2_AUTOCMD12_ERR_STATUS)	32	R	0000_0000h	10.3.8.16/ 2812
30B5_0040	Host Controller Capabilities (uSDHC2_HOST_CTRL_CAP)	32	R	07F3_B407h	10.3.8.17/ 2815
30B5_0044	Watermark Level (uSDHC2_WTMK_LVL)	32	R/W	0810_0810h	10.3.8.18/ 2818
30B5_0048	Mixer Control (uSDHC2_MIX_CTRL)	32	R/W	8000_0000h	10.3.8.19/ 2819
30B5_0050	Force Event (uSDHC2_FORCE_EVENT)	32	W (always reads 0)	0000_0000h	10.3.8.20/ 2821
30B5_0054	ADMA Error Status Register (uSDHC2_ADMA_ERR_STATUS)	32	R	0000_0000h	10.3.8.21/ 2824
30B5_0058	ADMA System Address (uSDHC2_ADMA_SYS_ADDR)	32	R/W	0000_0000h	10.3.8.22/ 2826
30B5_0060	DLL (Delay Line) Control (uSDHC2_DLL_CTRL)	32	R/W	0000_0200h	10.3.8.23/ 2827
30B5_0064	DLL Status (uSDHC2_DLL_STATUS)	32	R	0000_0000h	10.3.8.24/ 2829
30B5_0068	CLK Tuning Control and Status (uSDHC2_CLK_TUNE_CTRL_STATUS)	32	R/W	0000_0000h	10.3.8.25/ 2830
30B5_0070	Strobe DLL Control (uSDHC2_STROBE_DLL_CTRL)	32	R/W	0000_0000h	10.3.8.26/ 2832
30B5_0074	Strobe DLL Status (uSDHC2_STROBE_DLL_STATUS)	32	R	0000_0000h	10.3.8.27/ 2834

Table continues on the next page...

uSDHC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30B5_00C0	Vendor Specific Register (uSDHC2_VEND_SPEC)	32	R/W	2000_7809h	10.3.8.28/ 2836
30B5_00C4	MMC Boot Register (uSDHC2_MMC_BOOT)	32	R/W	0000_0000h	10.3.8.29/ 2839
30B5_00C8	Vendor Specific 2 Register (uSDHC2_VEND_SPEC2)	32	R/W	0000_0006h	10.3.8.30/ 2840
30B5_00CC	Tuning Control Register (uSDHC2_TUNING_CTRL)	32	R/W	0021_2800h	10.3.8.31/ 2842
30B6_0000	DMA System Address (uSDHC3_DS_ADDR)	32	R/W	0000_0000h	10.3.8.1/ 2777
30B6_0004	Block Attributes (uSDHC3_BLK_ATT)	32	R/W	0000_0000h	10.3.8.2/ 2778
30B6_0008	Command Argument (uSDHC3_CMD_ARG)	32	R/W	0000_0000h	10.3.8.3/ 2779
30B6_000C	Command Transfer Type (uSDHC3_CMD_XFR_TYP)	32	R/W	0000_0000h	10.3.8.4/ 2780
30B6_0010	Command Response0 (uSDHC3_CMD_RSP0)	32	R	0000_0000h	10.3.8.5/ 2783
30B6_0014	Command Response1 (uSDHC3_CMD_RSP1)	32	R	0000_0000h	10.3.8.6/ 2784
30B6_0018	Command Response2 (uSDHC3_CMD_RSP2)	32	R	0000_0000h	10.3.8.7/ 2784
30B6_001C	Command Response3 (uSDHC3_CMD_RSP3)	32	R	0000_0000h	10.3.8.8/ 2785
30B6_0020	Data Buffer Access Port (uSDHC3_DATA_BUFF_ACC_PORT)	32	R/W	0000_0000h	10.3.8.9/ 2786
30B6_0024	Present State (uSDHC3_PRES_STATE)	32	R	0000_8080h	10.3.8.10/ 2786
30B6_0028	Protocol Control (uSDHC3_PROT_CTRL)	32	R/W	0880_0020h	10.3.8.11/ 2792
30B6_002C	System Control (uSDHC3_SYS_CTRL)	32	R/W	8080_800Fh	10.3.8.12/ 2797
30B6_0030	Interrupt Status (uSDHC3_INT_STATUS)	32	w1c	0000_0000h	10.3.8.13/ 2800
30B6_0034	Interrupt Status Enable (uSDHC3_INT_STATUS_EN)	32	R/W	0000_0000h	10.3.8.14/ 2806
30B6_0038	Interrupt Signal Enable (uSDHC3_INT_SIGNAL_EN)	32	R/W	0000_0000h	10.3.8.15/ 2809
30B6_003C	Auto CMD12 Error Status (uSDHC3_AUTOCMD12_ERR_STATUS)	32	R	0000_0000h	10.3.8.16/ 2812
30B6_0040	Host Controller Capabilities (uSDHC3_HOST_CTRL_CAP)	32	R	07F3_B407h	10.3.8.17/ 2815
30B6_0044	Watermark Level (uSDHC3_WTMK_LVL)	32	R/W	0810_0810h	10.3.8.18/ 2818

Table continues on the next page...

uSDHC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30B6_0048	Mixer Control (uSDHC3_MIX_CTRL)	32	R/W	8000_0000h	10.3.8.19/ 2819
30B6_0050	Force Event (uSDHC3_FORCE_EVENT)	32	W (always reads 0)	0000_0000h	10.3.8.20/ 2821
30B6_0054	ADMA Error Status Register (uSDHC3_ADMA_ERR_STATUS)	32	R	0000_0000h	10.3.8.21/ 2824
30B6_0058	ADMA System Address (uSDHC3_ADMA_SYS_ADDR)	32	R/W	0000_0000h	10.3.8.22/ 2826
30B6_0060	DLL (Delay Line) Control (uSDHC3_DLL_CTRL)	32	R/W	0000_0200h	10.3.8.23/ 2827
30B6_0064	DLL Status (uSDHC3_DLL_STATUS)	32	R	0000_0000h	10.3.8.24/ 2829
30B6_0068	CLK Tuning Control and Status (uSDHC3_CLK_TUNE_CTRL_STATUS)	32	R/W	0000_0000h	10.3.8.25/ 2830
30B6_0070	Strobe DLL Control (uSDHC3_STROBE_DLL_CTRL)	32	R/W	0000_0000h	10.3.8.26/ 2832
30B6_0074	Strobe DLL Status (uSDHC3_STROBE_DLL_STATUS)	32	R	0000_0000h	10.3.8.27/ 2834
30B6_00C0	Vendor Specific Register (uSDHC3_VEND_SPEC)	32	R/W	2000_7809h	10.3.8.28/ 2836
30B6_00C4	MMC Boot Register (uSDHC3_MMC_BOOT)	32	R/W	0000_0000h	10.3.8.29/ 2839
30B6_00C8	Vendor Specific 2 Register (uSDHC3_VEND_SPEC2)	32	R/W	0000_0006h	10.3.8.30/ 2840
30B6_00CC	Tuning Control Register (uSDHC3_TUNING_CTRL)	32	R/W	0021_2800h	10.3.8.31/ 2842

10.3.8.1 DMA System Address (uSDHCx_DS_ADDR)

This register contains the physical system memory address used for DMA transfers.

Address: Base address + 0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DS_ADDR																														0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_DS_ADDR field descriptions

Field	Description
31–2 DS_ADDR	DMA System Address: This register contains the 32-bit system memory address for a DMA transfer. Since the address must be word (4 bytes) aligned, the least 2 bits are reserved, always 0. When the uSDHC stops a DMA transfer, this register points to the system address of the next contiguous data position. It can be accessed only when no transaction is executing (i.e. after a transaction has stopped). Read operation during transfers may return an invalid value. The Host Driver shall initialize this register before starting a DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register. This register is protected during a data transfer. When data lines are active, write to this register is ignored. The Host driver shall wait, until the DLA bit in the Present State register is cleared, before writing to this register. The uSDHC internal DMA does not support a virtual memory system. It only supports continuous physical memory access. And due to AHB burst limitations, if the burst must cross the 1 KB boundary, uSDHC will automatically change SEQ burst type to NSEQ. Since this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so SW cannot change this register when TC bit is set. Such restriction is also listed in Software Restrictions .
Reserved	This read-only field is reserved and always has the value 0.

10.3.8.2 Block Attributes (uSDHCx_BLK_ATT)

This register is used to configure the number of data blocks and the number of bytes in each block.

Address: Base address + 4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BLKCNT																0			BLKSIZE[12:0]											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_BLK_ATT field descriptions

Field	Description
31–16 BLKCNT	Blocks Count For Current Transfer: This register is enabled when the Block Count Enable bit in the Transfer Mode register is set to 1 and is valid only for multiple block transfers. For single block transfer, this register will always read as 1. The Host Driver shall set this register to a value between 1 and the maximum block count. The uSDHC decrements the block count after each block transfer and stops when the count reaches zero. Setting the block count to 0 results in no data blocks being transferred. This register should be accessed only when no transaction is executing (i.e. after transactions are stopped). During data transfer, read operations on this register may return an invalid value and write operations are ignored.

Table continues on the next page...

uSDHCx_BLK_ATT field descriptions (continued)

Field	Description
	When saving transfer content as a result of a Suspend command, the number of blocks yet to be transferred can be determined by reading this register. The reading of this register should be applied after transfer is paused by stop at block gap operation and before sending the command marked as suspend. This is because when Suspend command is sent out, uSDHC will regard the current transfer is aborted and change BLKCNT register back to its original value instead of keeping the dynamical indicator of remained block count. When restoring transfer content prior to issuing a Resume command, the Host Driver shall restore the previously saved block count. **NOTE:** Although the BLKCNT field is 0 after reset, the read of reset value is 0x1. This is because when MSBSEL bit is indicating a single block transfer, the read value of BLKCNT is always 1. FFFF 65535 blocks 0002 2 blocks 0001 1 block 0000 Stop Count
15–13 Reserved	This read-only field is reserved and always has the value 0.
BLKSIZE[12:0]	Transfer Block Size: This register specifies the block size for block data transfers. Values ranging from 1 byte up to the maximum buffer size can be set. It can be accessed only when no transaction is executing (i.e. after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations will be ignored. 1000 4096 Bytes 800 2048 Bytes 200 512 Bytes 1FF 511 Bytes 004 4 Bytes 003 3 Bytes 002 2 Bytes 001 1 Byte 000 No data transfer

10.3.8.3 Command Argument (uSDHCx_CMD_ARG)

This register contains the SD / MMC Command Argument.

.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMDARG[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_CMD_ARG field descriptions

Field	Description
CMDARG[31:0]	Command Argument The SD / MMC Command Argument is specified as bits 39-8 of the Command Format in the SD or MMC Specification.This register is write protected when the Command Inhibit (CMD) bit in the Present State register is set.

10.3.8.4 Command Transfer Type (uSDHCx_CMD_XFR_TYP)

This register is used to control the operation of data transfers. The Host Driver shall set this register before issuing a command followed by a data transfer, or before issuing a Resume command. To prevent data loss, the uSDHC prevents writing to the bits, that are involved in the data transfer of this register, when data transfer is active. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN and DMAEN.

The Host Driver shall check the Command Inhibit DAT bit (CDIHB) and the Command Inhibit CMD bit (CIHB) in the Present State register before writing to this register. When the CDIHB bit in the Present State register is set, any attempt to send a command with data by writing to this register is ignored; when the CIHB bit is set, any write to this register is ignored.

On sending commands with data transfer invovled, it is mandatory that the block size is non-zero. Block count must also be non-zero, or indicated as single block transfer (bit 5 of this register is '0' when written), or block count is disabled (bit 1 of this register is '0' when written), otherwise uSDHC will ignore the sending of this command and do nothing. For write command, with all above restrictions, it is also mandatory that the write protect switch is not active (WPSPL bit of Present State Register is '1'), otherwise uSDHC will also ignore the command.

If the commands with data transfer does not receive the response in 64 clock cycles, i.e., response time-out, uSDHC will regard the external device does not accept the command and abort the data transfer. In this scenario, the driver should issue the command again to re-try the transfer. It is also possible that for some reason the card responds the command but uSDHC does not receive the response, and if it is internal DMA (either simple DMA or ADMA) read operation, the external system memory is over-written by the internal DMA with data sent back from the card.

The table below shows the summary of how register settings determine the type of data transfer.

Table 10-44. Transfer Type Register Setting for Various Transfer Types

Multi/Single Block Select	Block Count Enable	Block Count	Function
0	Don't Care	Don't Care	Single Transfer
1	0	Don't Care	Infinite Transfer
1	1	Positive Number	Multiple Transfer
1	1	Zero	No Data Transfer

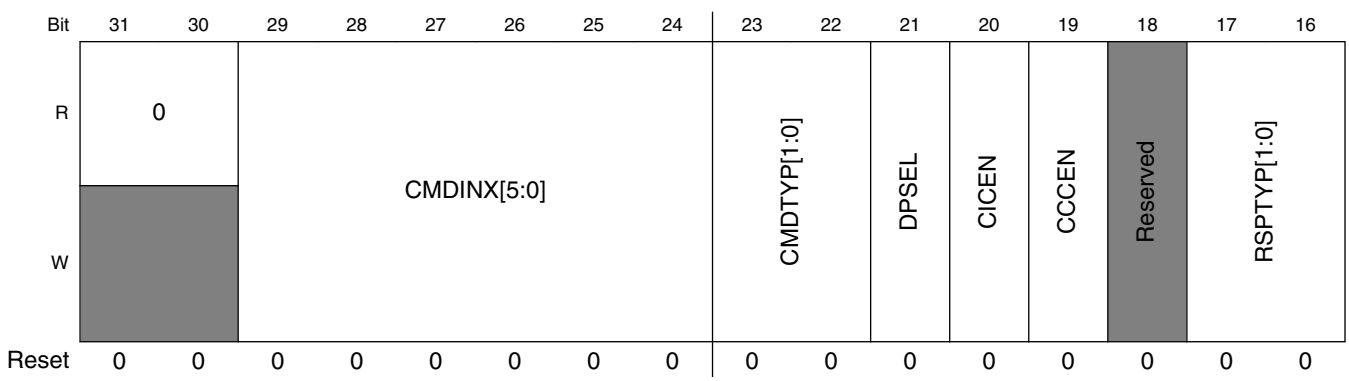
The table below shows the relationship between the Command Index Check Enable and the Command CRC Check Enable, in regards to the Response Type bits as well as the name of the response type.

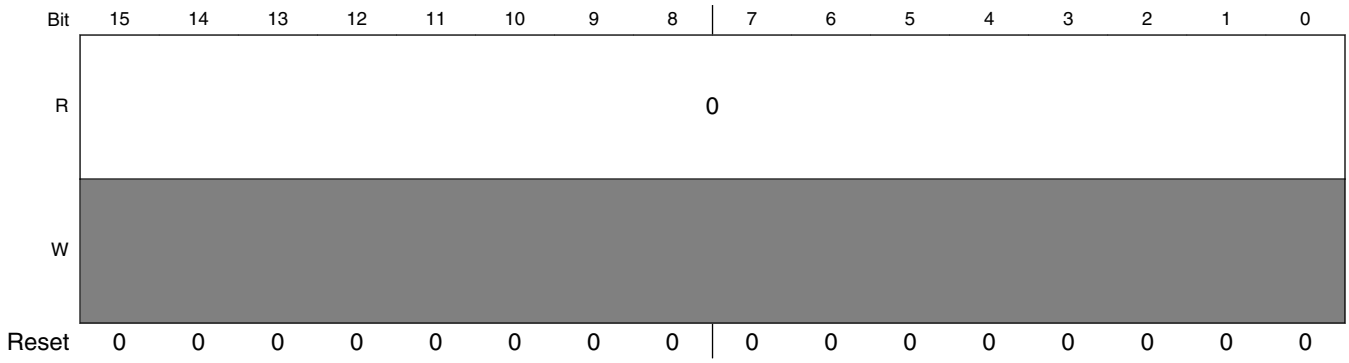
Table 10-45. Relationship Between Parameters and the Name of the Response Type

Response Type	Index Check Enable	CRC Check Enable	Name of Response Type
00	0	0	No Response
01	0	1	R2
10	0	0	R3,R4
10	1	1	R1,R5,R6
11	1	1	R1b,R5b

- In the SDIO specification, response type notation for R5b is not defined. R5 includes R5b in the SDIO specification. But R5b is defined in this specification to specify that the uSDHC will check the busy status after receiving a response. For example, usually CMD52 is used with R5, but the I/O abort command shall be used with R5b.
- The CRC field for R3 and R4 is expected to be all 1 bits. The CRC check shall be disabled for these response types.

Address: Base address + Ch offset





uSDHCx_CMD_XFR_TYP field descriptions

Field	Description
31–30 Reserved	This read-only field is reserved and always has the value 0.
29–24 CMDINX[5:0]	Command Index These bits shall be set to the command number that is specified in bits 45-40 of the Command-Format in the SD Memory Card Physical Layer Specification and SDIO Card Specification.
23–22 CMDTYP[1:0]	Command Type There are three types of special commands: Suspend, Resume and Abort. These bits shall be set to 00b for all other commands. • Suspend Command: If the Suspend command succeeds, the uSDHC shall assume that the card bus has been released and that it is possible to issue the next command which uses the DATA line. Since the uSDHC does not monitor the content of command response, it does not know if the Suspend command succeeded or not. It is the Host Driver's responsibility to check the status of the Suspend command and send another command marked as Suspend to inform the uSDHC that a Suspend command was successfully issued. Refer to Suspend Resume for more details. After the end bit of command is sent, the uSDHC de-asserts Read Wait for read transactions and stops checking busy for write transactions. In 4-bit mode, the interrupt cycle starts. If the Suspend command fails, the uSDHC will maintain its current state, and the Host Driver shall restart the transfer by setting the Continue Request bit in the Protocol Control register. • Resume Command: The Host Driver re-starts the data transfer by restoring the registers saved before sending the Suspend Command and then sends the Resume Command. The uSDHC will check for a pending busy state before starting write transfers. • Abort Command: If this command is set when executing a read transfer, the uSDHC will stop reads to the buffer. If this command is set when executing a write transfer, the uSDHC will stop driving the DATA line. After issuing the Abort command, the Host Driver should issue a software reset (Abort Transaction). 11 Abort CMD12, CMD52 for writing I/O Abort in CCCR 10 Resume CMD52 for writing Function Select in CCCR 01 Suspend CMD52 for writing Bus Suspend in CCCR 00 Normal Other commands
21 DPSEL	Data Present Select This bit is set to 1 to indicate that data is present and shall be transferred using the DATA line. It is set to 0 for the following: • Commands using only the CMD line (e.g. CMD52). • Commands with no data transfer, but using the busy signal on DATA0 line (R1b or R5b e.g. CMD38)

Table continues on the next page...

uSDHCx_CMD_XFR_TYP field descriptions (continued)

Field	Description
	NOTE: In resume command, this bit shall be set, and other bits in this register shall be set the same as when the transfer was initially launched. When the Write Protect switch is on, (i.e. the WPSPL bit is active as '0'), any command with a write operation will be ignored. That is to say, when this bit is set, while the DTDSEL bit is 0, writes to the register Transfer Type are ignored. 1 Data Present 0 No Data Present
20 CICEN	Command Index Check Enable If this bit is set to 1, the uSDHC will check the Index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this bit is set to 0, the Index field is not checked. 1 Enable 0 Disable
19 CCCEN	Command CRC Check Enable If this bit is set to 1, the uSDHC shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked. The number of bits checked by the CRC field value changes according to the length of the response. (Refer to RSPTYP[1:0] and Command Transfer Type (uSDHC_CMD_XFR_TYP) .) 1 Enable 0 Disable
18 -	This field is reserved. Reserved
17–16 RSPTYP[1:0]	Response Type Select 00 No Response 01 Response Length 136 10 Response Length 48 11 Response Length 48, check Busy after response
Reserved	This read-only field is reserved and always has the value 0.

10.3.8.5 Command Response0 (uSDHCx_CMD_RSP0)

This register is used to store part 0 of the response bits from the card.

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP0[31:0]																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_CMD_RSP0 field descriptions

Field	Description
CMDRSP0[31:0]	Command Response 0 Refer to Command Response3 (uSDHC_CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.

10.3.8.6 Command Response1 (uSDHCx_CMD_RSP1)

This register is used to store part 1 of the response bits from the card.

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP1[31:0]																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_CMD_RSP1 field descriptions

Field	Description
CMDRSP1[31:0]	Command Response 1 Refer to Command Response3 (uSDHC_CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.

10.3.8.7 Command Response2 (uSDHCx_CMD_RSP2)

This register is used to store part 2 of the response bits from the card.

Address: Base address + 18h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP2[31:0]																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_CMD_RSP2 field descriptions

Field	Description
CMDRSP2[31:0]	Command Response 2 Refer to Command Response3 (uSDHC_CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.

10.3.8.8 Command Response3 (uSDHCx_CMD_RSP3)

This register is used to store part 3 of the response bits from the card.

The table below describes the mapping of command responses from the SD Bus to Command Response registers for each response type. In the table, R[ ] refers to a bit range within the response data as transmitted on the SD Bus.

Table 10-46. Response Bit Definition for Each Response Type

Response Type	Meaning of Response	Response Field	Response Register
R1,R1b (normal response)	Card Status	R[39:8]	CMDRSP0
R1b (Auto CMD12 response)	Card Status for Auto CMD12	R[39:8]	CMDRSP3
R2 (CID, CSD register)	CID/CSD register [127:8]	R[127:8]	{CMDRSP3[23:0], CMDRSP2, CMDRSP1, CMDRSP0}
R3 (OCR register)	OCR register for memory	R[39:8]	CMDRSP0
R4 (OCR register)	OCR register for I/O etc.	R[39:8]	CMDRSP0
R5, R5b	SDIO response	R[39:8]	CMDRSP0
R6 (Publish RCA)	New Published RCA[31:16] and card status[15:0]	R[39:9]	CMDRSP0

This table shows that most responses with a length of 48 (R[47:0]) have 32-bits of the response data (R[39:8]) stored in the CMDRSP0 register. Responses of type R1b (Auto CMD12 responses) have response data bits (R[39:8]) stored in the CMDRSP3 register. Responses with length 136 (R[135:0]) have 120-bits of the response data (R[127:8]) stored in the CMDRSP0, 1, 2, and 3 registers.

To be able to read the response status efficiently, the uSDHC only stores part of the response data in the Command Response registers. This enables the Host Driver to efficiently read 32-bits of response data in one read cycle on a 32-bit bus system. Parts of the response, the Index field and the CRC, are checked by the uSDHC (as specified by the Command Index Check Enable and the Command CRC Check Enable bits in the Transfer Type register) and generate an error interrupt if any error is detected. The bit range for the CRC check depends on the response length. If the response length is 48, the uSDHC will check R[47:1], and if the response length is 136 the uSDHC will check R[119:1].

Since the uSDHC may have a multiple block data transfer executing concurrently with a CMD_wo_DAT command, the uSDHC stores the Auto CMD12 response in the CMDRSP3 register. The CMD_wo_DAT response is stored in CMDRSP0. This allows

the uSDHC to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa. When the uSDHC modifies part of the Command Response registers, as shown in the table above, it preserves the unmodified bits.

Address: Base address + 1Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP3[31:0]																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_CMD_RSP3 field descriptions

Field	Description
CMDRSP3[31:0]	Command Response 3 Refer to Command Response3 (uSDHC_CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.

10.3.8.9 Data Buffer Access Port (uSDHCx_DATA_BUFF_ACC_PORT)

This is a 32-bit data port register used to access the internal buffer.

Address: Base address + 20h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATCONT[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_DATA_BUFF_ACC_PORT field descriptions

Field	Description
DATCONT[31:0]	Data Content The Buffer Data Port register is for 32-bit data access by the ARM platform or the external DMA. When the internal DMA is enabled, any write to this register is ignored, and any read from this register will always yield 0s.

10.3.8.10 Present State (uSDHCx_PRES_STATE)

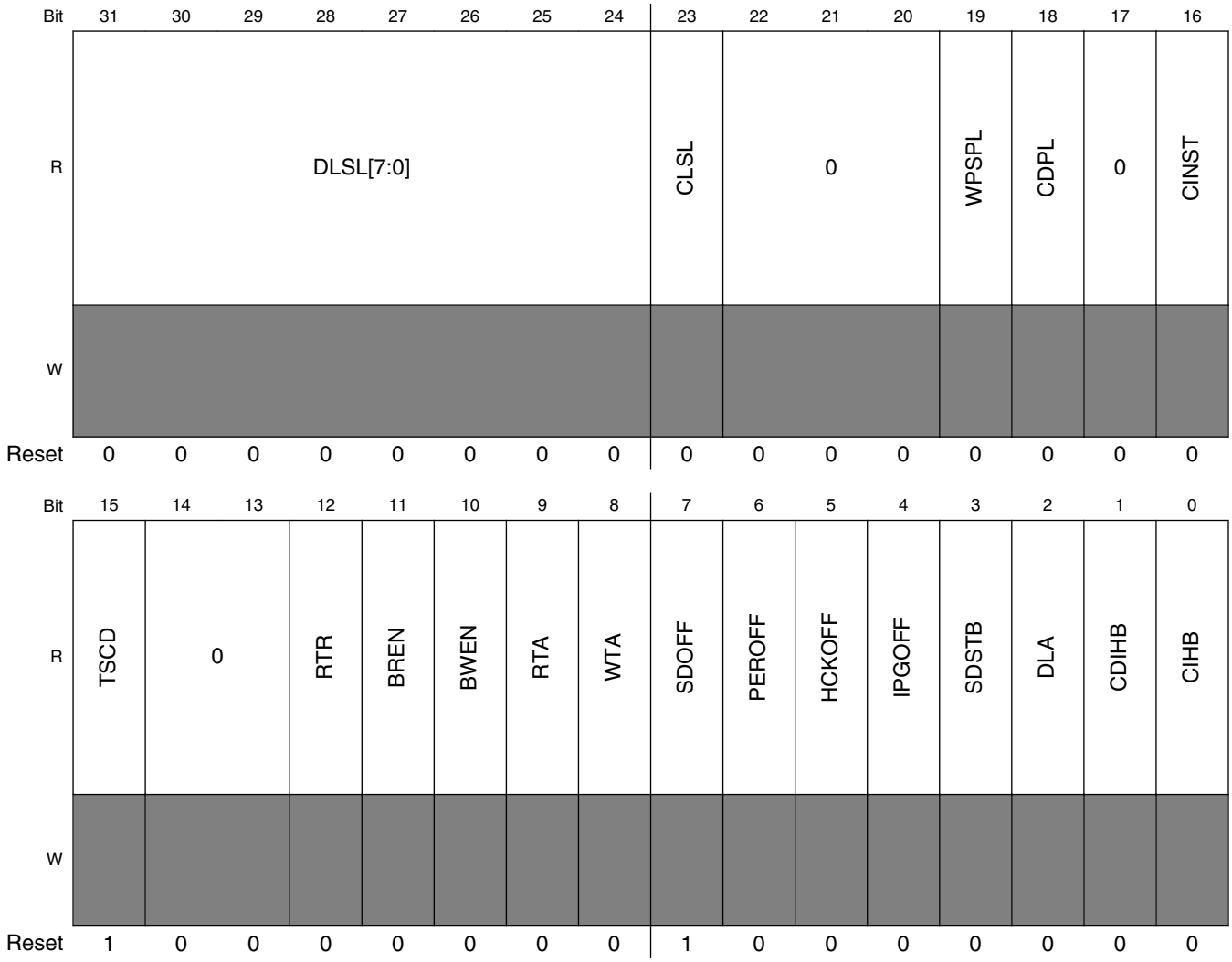
The Host Driver can get status of the uSDHC from this 32-bit read only register.

- The Host Driver can issue CMD0, CMD12, CMD13 (for memory) and CMD52 (for SDIO) when the DATA lines are busy during a data transfer. These commands can be issued when Command Inhibit (CMD) is set to zero. Other commands shall be

issued when Command Inhibit (DATA) is set to zero. Possible changes to the SD Physical Specification may add other commands to this list in the future.

- Note: the reset value of Present State Register depend on testbench connectivity.

Address: Base address + 24h offset



uSDHCx_PRES_STATE field descriptions

Field	Description
31–24 DLSL[7:0]	DATA[7:0] Line Signal Level This status is used to check the DATA line level to recover from errors, and for debugging.This is especially useful in detecting the busy signal level from DATA0. The reset value is affected by the external pull-up / pull-down resistors. By default, the read value of this bit field after reset is 8'b11110111, when DATA3 is pulled down and the other lines are pulled up. DATA7 Data 7 line signal level DATA6 Data 6 line signal level DATA5 Data 5 line signal level DATA4 Data 4 line signal level DATA3 Data 3 line signal level

Table continues on the next page...

uSDHCx_PRES_STATE field descriptions (continued)

Field	Description
	DATA2 Data 2 line signal level DATA1 Data 1 line signal level DATA0 Data 0 line signal level
23 CLSL	CMD Line Signal Level This status is used to check the CMD line level to recover from errors, and for debugging. The reset value is affected by the external pull-up / pull-down resistor, by default, the read value of this bit after reset is 1'b1, when the command line is pulled up.
22–20 Reserved	This read-only field is reserved and always has the value 0.
19 WPSPL	Write Protect Switch Pin Level The Write Protect Switch is supported for memory and combo cards.This bit reflects the inverted value of the WP pin of the card socket. A software reset does not affect this bit. The reset value is effected by the external write protect switch. If the WP pin is not used, it should be tied low, so that the reset value of this bit is high and write is enabled. 1 Write enabled (WP = 0) 0 Write protected (WP = 1)
18 CDPL	Card Detect Pin Level This bit reflects the inverse value of the CD_B pin for the card socket. Debouncing is not performed on this bit. This bit may be valid, but is not guaranteed, because of propagation delay. Use of this bit is limited to testing since it must be debounced by software. A software reset does not effect this bit. A write to the Force Event Register does not effect this bit. The reset value is effected by the external card detection pin. This bit shows the value on the CD_B pin (i.e. when a card is inserted in the socket, it is 0 on the CD_B input, and consequently the CDPL reads 1.) 1 Card present (CD_B = 0) 0 No card present (CD_B = 1)
17 Reserved	This read-only field is reserved and always has the value 0.
16 CINST	Card Inserted This bit indicates whether a card has been inserted. The uSDHC debounces this signal so that the Host Driver will not need to wait for it to stabilize. Changing from a 0 to 1 generates a Card Insertion interrupt in the Interrupt Status register. Changing from a 1 to 0 generates a Card Removal interrupt in the Interrupt Status register. A write to the Force Event Register does not effect this bit. The Software Reset For All in the System Control register does not effect this bit. A software reset does not effect this bit. 1 Card Inserted 0 Power on Reset or No Card
15 TSCD	Tape Select Change Done This bit indicates the dealy setting is effective after write CLK_TUNE_CTRL_STATUS register. 1 Delay cell select change is finished. 0 Delay cell select change is not finished.
14–13 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

uSDHCx_PRES_STATE field descriptions (continued)

Field	Description
12 RTR	Re-Tuning Request (only for SD3.0 SDR104 mode) Host Controller may request Host Driver to execute re-tuning sequence by setting this bit when the data window is shifted by temperature drift and a tuned sampling point does not have a good margin to receive correct data. This bit is cleared when a command is issued with setting Execute Tuning bit in MIXER_CTRL register. Changing of this bit from 0 to 1 generates Re-Tuning Event. Refer to Interrupt status registers for more detail. This bit isn't set to 1 if Sampling Clock Select in the MIXER_CTRL register is set to 0 (using fixed sampling clock). 1 Sampling clock needs re-tuning 0 Fixed or well tuned sampling clock
11 BREN	Buffer Read Enable This status bit is used for non-DMA read transfers. The uSDHC implements an internal buffer to transfer data efficiently. This read only flag indicates that valid data exists in the host side buffer. If this bit is high, valid data greater than the watermark level exist in the buffer. A change of this bit from 1 to 0 occurs when some reads from the buffer(read DATPORT (Base + 0x20)) are made and the buffer hasn't valid data greater than the watermark level. A change of this bit from 0 to1 occurs when there is enough valid data ready in the buffer and the Buffer Read Ready interrupt has been generated and enabled. 1 Read enable 0 Read disable
10 BWEN	Buffer Write Enable This status bit is used for non-DMA write transfers. The uSDHC implements an internal buffer to transfer data efficiently. This read only flag indicates if space is available for write data. If this bit is 1, valid data greater than the watermark level can be written to the buffer. A change of this bit from 1 to 0 occurs when some writes to the buffer(write DATPORT(Base + 0x20)) are made and the buffer hasn't valid space greater than the watermark level. A change of this bit from 0 to 1 occurs when the buffer can hold valid data greater than the write watermark level and the Buffer Write Ready interrupt is generated and enabled. 1 Write enable 0 Write disable
9 RTA	Read Transfer Active This status bit is used for detecting completion of a read transfer. This bit is set for either of the following conditions: • After the end bit of the read command. • When writing a 1 to the Continue Request bit in the Protocol Control register to restart a read transfer. A Transfer Complete interrupt is generated when this bit changes to 0. This bit is cleared for either of the following conditions: • When the last data block as specified by block length is transferred to the System, i.e. all data are read away from uSDHC internal buffer. • When all valid data blocks have been transferred from uSDHC internal buffer to the System and no current block transfers are being sent as a result of the Stop At Block Gap Request being set to 1. 1 Transferring data 0 No valid data

Table continues on the next page...

uSDHCx_PRES_STATE field descriptions (continued)

Field	Description
8 WTA	Write Transfer Active This status bit indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the uSDHC. This bit is set in either of the following cases: • After the end bit of the write command. • When writing 1 to the Continue Request bit in the Protocol Control register to restart a write transfer. This bit is cleared in either of the following cases: • After getting the CRC status of the last data block as specified by the transfer count (Single and Multiple). • After getting the CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request. During a write transaction, a Block Gap Event interrupt is generated when this bit is changed to 0, as result of the Stop At Block Gap Request being set. This status is useful for the Host Driver in determining when to issue commands during Write Busy state. 1 Transferring data 0 No valid data
7 SDOFF	SD Clock Gated Off Internally This status bit indicates that the SD Clock is internally gated off, because of buffer over / under-run or read pause without read wait assertion, or the driver set FRC_SDCLK_ON bit is 0 to stop the SD clock in idle status. Set IPG_PERCLK_SOFT_EN and CARD_CLK_SOFT_EN to 0 also gate off SD clock. This bit is for the Host Driver to debug data transaction on the SD bus. 1 SD Clock is gated off. 0 SD Clock is active.
6 PEROFF	IPG_PERCLK Gated Off Internally This status bit indicates that the IPG_PERCLK is internally gated off. This bit is for the Host Driver to debug transaction on the SD bus. When IPG_CLK_SOFT_EN is cleared, IPG_PERCLK will be gated off, otherwise IPG_PERCLK will be always active. 1 IPG_PERCLK is gated off. 0 IPG_PERCLK is active.
5 HCKOFF	HCLK Gated Off Internally This status bit indicates that the HCLK is internally gated off. This bit is for the Host Driver to debug during a data transfer. 1 HCLK is gated off. 0 HCLK is active.
4 IPGOFF	IPG_CLK Gated Off Internally This status bit indicates that the ipg_clk is internally gated off. This bit is for the Host Driver to debug. 1 IPG_CLK is gated off. 0 IPG_CLK is active.
3 SDSTB	SD Clock Stable

Table continues on the next page...

uSDHCx_PRES_STATE field descriptions (continued)

Field	Description
	This status bit indicates that the internal card clock is stable. This bit is for the Host Driver to poll clock status when changing the clock frequency. It is recommended to clear FRC_SDCLK_ON bit in System Control register to remove glitches on the card clock when the frequency is changing. *Before changing clock divisor value(SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high.* 1 Clock is stable. 0 Clock is changing frequency and not stable.
2 DLA	Data Line Active This status bit indicates whether one of the DATA lines on the SD Bus is in use. In the case of read transactions: This status indicates if a read transfer is executing on the SD Bus. Changes in this value from 1 to 0, between data blocks, generates a Block Gap Event interrupt in the Interrupt Status register. This bit will be set in either of the following cases: • After the end bit of the read command. • When writing a 1 to the Continue Request bit in the Protocol Control register to restart a read transfer. This bit will be cleared in either of the following cases: (1) When the end bit of the last data block is sent from the SD Bus to the uSDHC. (2) When the Read Wait state is stopped by a Suspend command and the DATA2 line is released. The uSDHC will wait at the next block gap by driving Read Wait at the start of the interrupt cycle. If the Read Wait signal is already driven (data buffer cannot receive data), the uSDHC can wait for a current block gap by continuing to drive the Read Wait signal. It is necessary to support Read Wait in order to use the suspend / resume function. This bit will remain 1 during Read Wait. In the case of write transactions: This status indicates that a write transfer is executing on the SD Bus. Changes in this value from 1 to 0 generate a Transfer Complete interrupt in the Interrupt Status register. This bit will be set in either of the following cases: • After the end bit of the write command. • When writing to 1 to the Continue Request bit in the Protocol Control register to continue a write transfer. This bit will be cleared in either of the following cases: • When the SD card releases Write Busy of the last data block, the uSDHC will also detect if the output is not busy. If the SD card does not drive the busy signal after the CRC status is received, the uSDHC shall assume the card drive "Not Busy". • When the SD card releases write busy, prior to waiting for write transfer, and as a result of a Stop At Block Gap Request. In the case of command with busy pending: This status indicates that a busy state follows the command and the data line is in use. This bit will be cleared when the DATA0 line is released. 1 DATA Line Active 0 DATA Line Inactive

Table continues on the next page...

uSDHCx_PRES_STATE field descriptions (continued)

Field	Description
1 CDIHB	Command Inhibit (DATA) This status bit is generated if either the DAT Line Active or the Read Transfer Active is set to 1. If this bit is 0, it indicates that the uSDHC can issue the next SD / MMC Command. Commands with a busy signal belong to Command Inhibit (DATA) (for example. R1b, R5b type). Changing from 1 to 0 generates a Transfer Complete interrupt in the Interrupt Status register. **NOTE:** The SD Host Driver can save registers for a suspend transaction after this bit has changed from 1 to 0. 1 Cannot issue command which uses the DATA line 0 Can issue command which uses the DATA line
0 CIHB	Command Inhibit (CMD) If this status bit is 0, it indicates that the CMD line is not in use and the uSDHC can issue a SD / MMC Command using the CMD line. This bit is set also immediately after the Transfer Type register is written. This bit is cleared when the command response is received. Even if the Command Inhibit (DATA) is set to 1, Commands using only the CMD line can be issued if this bit is 0. Changing from 1 to 0 generates a Command Complete interrupt in the Interrupt Status register. If the uSDHC cannot issue the command because of a command conflict error (Refer to Command CRC Error) or because of a Command Not Issued By Auto CMD12 Error, this bit will remain 1 and the Command Complete is not set. The Status of issuing an Auto CMD12 does not show on this bit. 1 Cannot issue command 0 Can issue command using only CMD line

10.3.8.11 Protocol Control (uSDHCx_PROT_CTRL)

There are three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether the uSDHC issues a Suspend command or the SD card accepts the Suspend command.

1. If the Host Driver does not issue a Suspend command, the Continue Request shall be used to restart the transfer.
2. If the Host Driver issues a Suspend command and the SD card accepts it, a Resume command shall be used to restart the transfer.
3. If the Host Driver issues a Suspend command and the SD card does not accept it, the Continue Request shall be used to restart the transfer.

Any time Stop At Block Gap Request stops the data transfer, the Host Driver shall wait for a Transfer Complete (in the Interrupt Status register), before attempting to restart the transfer. When restarting the data transfer by Continue Request, the Host Driver shall clear the Stop At Block Gap Request before or simultaneously.

Address: Base address + 28h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-	NON_EXACT_BLK_RD	BURST_LEN_EN			WECRM	WECINS	WECINT	-			RD_DONE_NO_8CLK	IABG	RWCTL	CREQ	SABGREQ
Reset	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						DMASEL		CDSS	CDTL	EMODE		D3CD	DTW[1:0]		LCTL
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

uSDHCx_PROT_CTRL field descriptions

Field	Description
31 -	Reserved. Always write as 0
30 NON_EXACT_BLK_RD	Current block read is non-exact block read. It is only used for SDIO. 1 The block read is non-exact block read. Host driver needs to issue abort command to terminate this multi-block read. 0 The block read is exact block read. Host driver doesn't need to issue abort command to terminate this multi-block read.
29–27 BURST_LEN_EN	BURST length enable for INCR, INCR4 / INCR8 / INCR16, INCR4-WRAP / INCR8-WRAP / INCR16-WRAP This is used to enable / disable the burst length for the external AHB2AXI bridge. It is useful especially for INCR transfer because without burst length indicator, the AHB2AXI bridge does not know the burst length in advance. Without burst length indicator, AHB INCR transfers can only be converted to SINGLEs on the AXI side. xx1 Burst length is enabled for INCR x1x Burst length is enabled for INCR4 / INCR8 / INCR16 1xx Burst length is enabled for INCR4-WRAP / INCR8-WRAP / INCR16-WRAP
26 WECRM	Wakeup Event Enable On SD Card Removal This bit enables a wakeup event, via a Card Removal, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, the Card Removal Status and the uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active in order to assert the Card Removal Status and the uSDHC interrupt. 1 Enable 0 Disable
25 WECINS	Wakeup Event Enable On SD Card Insertion This bit enables a wakeup event, via a Card Insertion, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, the Card Insertion Status and the uSDHC

Table continues on the next page...

uSDHCx_PROT_CTRL field descriptions (continued)

Field	Description
	interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active in order to assert the Card Insertion Status and the uSDHC interrupt. 1 Enable 0 Disable
24 WECINT	Wakeup Event Enable On Card Interrupt This bit enables a wakeup event, via a Card Interrupt, in the Interrupt Status register. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. When this bit is set, the Card Interrupt Status and the uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active in order to assert the Card Interrupt Status and the uSDHC interrupt. 1 Enable 0 Disable
23–21 -	Reserved. Always write as 3'b100
20 RD_DONE_NO_8CLK	*Read done no 8 clock:* *According to the SD/MMC spec, for read data transaction, 8 clocks are needed after the end bit of the last data block. So, by default(RD_DONE_NO_8CLK=0), 8 clocks will be active after the end bit of the last read data transaction.* *However, this 8 clocks should not be active if user wants to use stop at block gap(include the auto stop at block gap in boot mode) feature for read and the RWCTL bit(bit18) is not enabled. In this case, software should set RD_DONE_NO_8CLK to avoid this 8 clocks. Otherwise, the device may send extra data to uSDHC while uSDHC ignores these data.* *In a summary, this bit should be set only if the use case needs to use stop at block gap feature while the device can't support the read wait feature.*
19 IABG	Interrupt At Block Gap This bit is valid only in 4-bit mode, of the SDIO card, and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. Setting to 0 disables interrupt detection during a multiple block transfer. If the SDIO card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0 to avoid an inadvertent interrupt. When the Host Driver detects an SDIO card insertion, it shall set this bit according to the CCCR of the card. 1 Enabled 0 Disabled
18 RWCTL	Read Wait Control The read wait function is optional for SDIO cards. If the card supports read wait, set this bit to enable use of the read wait protocol to stop read data using the DATA2 line. Otherwise the uSDHC has to stop the SD Clock to hold read data, which restricts commands generation. When the Host Driver detects an SDIO card insertion, it shall set this bit according to the CCCR of the card. If the card does not support read wait, this bit shall never be set to 1, otherwise DATA line conflicts may occur. If this bit is set to 0, stop at block gap during read operation is also supported, but the uSDHC will stop the SD Clock to pause reading operation. 1 Enable Read Wait Control, and assert Read Wait without stopping SD Clock at block gap when SABGREQ bit is set 0 Disable Read Wait Control, and stop SD Clock at block gap when SABGREQ bit is set
17 CREQ	Continue Request

Table continues on the next page...

uSDHCx_PROT_CTRL field descriptions (continued)

Field	Description
	This bit is used to restart a transaction which was stopped using the Stop At Block Gap Request. When a Suspend operation is not accepted by the card, it is also by setting this bit to restart the paused transfer. To cancel stop at the block gap, set Stop At Block Gap Request to 0 and set this bit to 1 to restart the transfer. The uSDHC automatically clears this bit, therefore it is not necessary for the Host Driver to set this bit to 0. If both Stop At Block Gap Request and this bit are 1, the continue request is ignored. 1 Restart 0 No effect
16 SABGREQ	Stop At Block Gap Request This bit is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the Transfer Complete is set to 1, indicating a transfer completion, the Host Driver shall leave this bit set to 1. Clearing both the Stop At Block Gap Request and Continue Request does not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The uSDHC will honor the Stop At Block Gap Request for write transfers, but for read transfers it requires that the SDIO card support Read Wait. Therefore, the Host Driver shall not set this bit during read transfers unless the SDIO card supports Read Wait and has set the Read Wait Control to 1, otherwise the uSDHC will stop the SD bus clock to pause the read operation during block gap. In the case of write transfers in which the Host Driver writes data to the Data Port register, the Host Driver shall set this bit after all block data is written. If this bit is set to 1, the Host Driver shall not write data to the Data Port register after a block is sent. Once this bit is set, the Host Driver shall not clear this bit before the Transfer Complete bit in Interrupt Status Register is set, otherwise the uSDHCs behavior is undefined. This bit effects Read Transfer Active, Write Transfer Active, DATA Line Active and Command Inhibit (DATA) in the Present State register. 1 Stop 0 Transfer
15–10 Reserved	This read-only field is reserved and always has the value 0.
9–8 DMASEL	DMA Select This field is valid while DMA (SDMA or ADMA) is enabled and selects the DMA operation. 00 No DMA or Simple DMA is selected 01 ADMA1 is selected 10 ADMA2 is selected 11 reserved
7 CDSS	Card Detect Signal Selection This bit selects the source for the card detection. 1 Card Detection Test Level is selected (for test purpose). 0 Card Detection Level is selected (for normal purpose).
6 CDTL	Card Detect Test Level This is bit is enabled while the Card Detection Signal Selection is set to 1 and it indicates card insertion. 1 Card Detect Test Level is 1, card inserted 0 Card Detect Test Level is 0, no card inserted
5–4 EMODE	Endian Mode

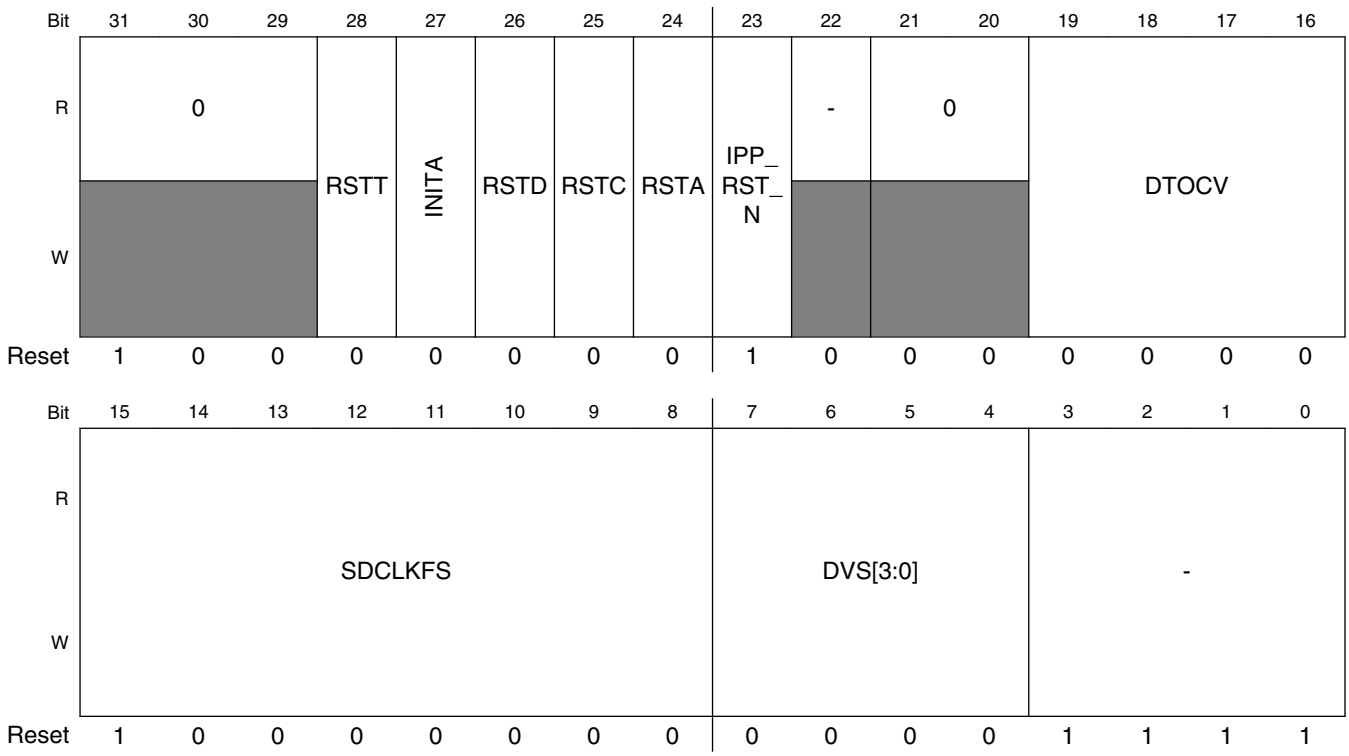
Table continues on the next page...

uSDHCx_PROT_CTRL field descriptions (continued)

Field	Description
	The uSDHC supports all three endian modes in data transfer. Refer to Data Buffer for more details. 00 Big Endian Mode 01 Half Word Big Endian Mode 10 Little Endian Mode 11 Reserved
3 D3CD	DATA3 as Card Detection Pin If this bit is set, DATA3 should be pulled down to act as a card detection pin. Be cautious when using this feature, because DATA3 is also a chip-select for the SPI mode. A pull-down on this pin and CMD0 may set the card into the SPI mode, which the uSDHC does not support. 1 DATA3 as Card Detection Pin 0 DATA3 does not monitor Card Insertion
2–1 DTW[1:0]	Data Transfer Width This bit selects the data width of the SD bus for a data transfer. The Host Driver shall set it to match the data width of the card. Possible Data transfer Width is 1-bit, 4-bits or 8-bits. 10 8-bit mode 01 4-bit mode 00 1-bit mode 11 Reserved
0 LCTL	LED Control This bit, fully controlled by the Host Driver, is used to caution the user not to remove the card while the card is being accessed. If the software is going to issue multiple SD commands, this bit can be set during all these transactions. It is not necessary to change for each transaction. When the software issues multiple SD commands, setting the bit once before the first command is sufficient: it is not necessary to reset the bit between commands. 1 LED on 0 LED off

10.3.8.12 System Control (uSDHC*x*_SYS_CTRL)

Address: Base address + 2Ch offset



uSDHC*x*_SYS_CTRL field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28 RSTT	Reset Tuning When set this bit to 1, it will reset tuning circuit. After tuning circuits are reset, bit value is 0. Clearing execute_tuning bit in AUTOCMD12_ERR_STATUS will also set this bit to 1 to reset tuning circuit
27 INITA	Initialization Active When this bit is set, 80 SD-Clocks are sent to the card. After the 80 clocks are sent, this bit is self cleared. This bit is very useful during the card power-up period when 74 SD-Clocks are needed and the clock auto gating feature is enabled. Writing 1 to this bit when this bit is already 1 has no effect. Writing 0 to this bit at any time has no effect. When either of the CIHB and CDIHB bits in the Present State Register are set, writing 1 to this bit is ignored (i.e. when command line or data lines are active, write to this bit is not allowed). On the otherhand, when this bit is set, i.e., during intialization active period, it is allowed to issue command, and the command bit stream will appear on the CMD pad after all 80 clock cycles are done. So when this command ends, the driver can make sure the 80 clock cycles are sent out. This is very useful when the driver needs send 80 cycles to the card and does not want to wait till this bit is self cleared.
26 RSTD	Software Reset For DATA Line

Table continues on the next page...

uSDHC*x*_SYS_CTRL field descriptions (continued)

Field	Description
	Only part of the data circuit is reset. DMA circuit is also reset. After this bit is set, SW waits for self-clear. The following registers and bits are cleared by this bit: • Data Port register • Buffer is cleared and initialized. • Present State register • Buffer Read Enable • Buffer Write Enable • Read Transfer Active • Write Transfer Active • DATA Line Active • Command Inhibit (DATA) Protocol Control register • Continue Request • Stop At Block Gap Request Interrupt Status register • Buffer Read Ready • Buffer Write Ready • DMA Interrupt • Block Gap Event • Transfer Complete **NOTE:** When reset, SW must make sure there is no incomplete data transferring. If there is data transfer going on, SW need wait TC or DC uSDHC_INT_STATUS register is set. 1 Reset 0 No Reset
25 RSTC	Software Reset For CMD Line Only part of the command circuit is reset. After this bit is set, SW waits for self-clear. The following registers and bits are cleared by this bit: • Present State register Command Inhibit (CMD) • Interrupt Status register Command Complete 1 Reset 0 No Reset
24 RSTA	Software Reset For ALL This reset effects the entire Host Controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared. During its initialization, the Host Driver shall set this bit to 1 to reset the uSDHC. The uSDHC shall reset this bit to 0 when the capabilities registers are valid and the Host Driver can read them. Additional use of Software Reset For All does not affect the value of the Capabilities registers. After this bit is set, it is recommended that the Host Driver reset the external card and re-initialize it. After this bit is set, SW should wait for self-clear. In tuning process, after every CMD19 is finished, this bit will be set to retest the uSDHC. **NOTE:** When reset, SW must make sure there is no incomplete data transferring. If there is data transfer going on, SW need wait TC or DC uSDHC_INT_STATUS register is set. 1 Reset 0 No Reset
23 IPP_RST_N	This register's value will be output to CARD from pad directly for hardware reset of the card if the card supports this feature.
22 -	Reserved

Table continues on the next page...

uSDHCx_SYS_CTRL field descriptions (continued)

Field	Description
21–20 Reserved	This read-only field is reserved and always has the value 0.
19–16 DTOCV	Data Timeout Counter Value This value determines the interval by which DAT line timeouts are detected. Refer to the Data Timeout Error bit in the Interrupt Status register for information on factors that dictate time-out generation. Time-out clock frequency will be generated by dividing the base clock SDCLK value by this value. The Host Driver can clear the Data Timeout Error Status Enable (in the Interrupt Status Enable register) to prevent inadvertent time-out events. 1111 SDCLK x 2^{28} 1110 SDCLK x 2^{27} 0001 SDCLK x 2^{14} 0000 SDCLK x 2^{13}
15–8 SDCLKFS	SDCLK Frequency Select This register is used to select the frequency of the SDCLK pin. The frequency is not programmed directly, rather this register holds the prescaler (this register) and divisor (next register) of the Base Clock Frequency register. *In Single Data Rate mode(DDR_EN bit of MIXERCTRL is '0')* Only the following settings are allowed: 80h) Base clock divided by 256 40h) Base clock divided by 128 20h) Base clock divided by 64 10h) Base clock divided by 32 08h) Base clock divided by 16 04h) Base clock divided by 8 02h) Base clock divided by 4 01h) Base clock divided by 2 00h) Base clock divided by 1 *While in Dual Data Rate mode(DDR_EN bit of MIXERCTRL is '1')* *Only the following settings are allowed:* *80h) Base clock divided by 512* *40h) Base clock divided by 256* *20h) Base clock divided by 128* *10h) Base clock divided by 64* *08h) Base clock divided by 32* *04h) Base clock divided by 16* *02h) Base clock divided by 8* *01h) Base clock divided by 4* *00h) Base clock divided by 2* *When S/W changes the DDR_EN bit, SDCLKFS may need to be changed also!*

Table continues on the next page...

uSDHCx_SYS_CTRL field descriptions (continued)

Field	Description
	In Single Data Rate mode, setting 00h bypasses the frequency prescaler of the SD Clock. Multiple bits must not be set, or the behavior of this prescaler is undefined. The two default divider values can be calculated by the frequency of ipg_perclk and the following Divisor bits. The frequency of SDCLK is set by the following formula: Clock Frequency = (Base Clock) / (prescaler x divisor) For example, in Single Data Rate mode, if the Base Clock Frequency is 96 MHz, and the target frequency is 25 MHz, then choosing the prescaler value of 01h and divisor value of 1h will yield 24 MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400 kHz, the prescaler value of 08h and divisor value of eh yields the exact clock value of 400 kHz. The reset value of this bit field is 80h, so if the input Base Clock (ipg_perclk) is about 96 MHz, the default SD Clock after reset is 375 kHz. According to the SD Physical Specification Version 1.1 and the SDIO Card Specification Version 1.2, the maximum SD Clock frequency is 50 MHz and shall never exceed this limit. *Before changing clock divisor value(SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high.* *If setting SDCLKFS and DVS can generate same clock frequency,(For example, in SDR mode, SDCLKFS = 01h is same as DVS = 01h.) SDCLKFS is highly recommended.*
7–4 DVS[3:0]	Divisor This register is used to provide a more exact divisor to generate the desired SD clock frequency. Note the divider can even support odd divisors without deterioration of duty cycle. *Before changing clock divisor value(SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high.* The setting are as following: 0000 Divide-by-1 0001 Divide-by-2 ..... 1110 Divide-by-15 1111 Divide-by-16
-	Reserved. Always write as 1.

10.3.8.13 Interrupt Status (uSDHCx_INT_STATUS)

An interrupt is generated when the Normal Interrupt Signal Enable is enabled and at least one of the status bits is set to 1. For all bits, writing 1 to a bit clears it; writing to 0 keeps the bit unchanged. More than one status can be cleared with a single register write. For Card Interrupt, before writing 1 to clear, it is required that the card stops asserting the interrupt, meaning that when the Card Driver services the interrupt condition, otherwise the CINT bit will be asserted again.

The table below shows the relationship between the Command Timeout Error and the Command Complete.

Table 10-47. uSDHC Status for Command Timeout Error/Command Complete Bit Combinations

Command Complete	Command Timeout Error	Meaning of the Status
0	0	X
X	1	Response not received within 64 SDCLK cycles
1	0	Response received

The table below shows the relationship between the Transfer Complete and the Data Timeout Error.

Table 10-48. uSDHC Status for Data Timeout Error/Transfer Complete Bit Combinations

Transfer Complete	Data Timeout Error	Meaning of the Status
0	0	X
0	1	Timeout occurred during transfer
1	X	Data Transfer Complete

The table below shows the relationship between the Command CRC Error and Command Timeout Error.

Table 10-49. uSDHC Status for Command CRC Error/Command Timeout Error Bit Combinations

Command Complete	Command Timeout Error	Meaning of the Status
0	0	No error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD line conflict

Address: Base address + 30h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			DMAE	0	TNE	0	AC12E	0	DEBE	DCE	DTOE	CIE	CEBE	CCE	CTOE
W				w1c		w1c		w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TP	0	RTE	0			CINT	CRM	CINS	BRR	BWR	DINT	BGE	TC	CC
W		w1c		w1c				w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_INT_STATUS field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28 DMAE	DMA Error Occurs when an Internal DMA transfer has failed. This bit is set to 1, when some error occurs in the data transfer. This error can be caused by either Simple DMA or ADMA, depending on which DMA is in use. The value in DMA System Address register is the next fetch address where the error occurs. Since any error corrupts the whole data block, the Host Driver shall re-start the transfer from the corrupted block boundary. The address of the block boundary can be calculated either from the current DS_ADDR value or from the remaining number of blocks and the block size. 1 Error 0 No Error
27 Reserved	This read-only field is reserved and always has the value 0.
26 TNE	Tuning Error: (only for SD3.0 SDR104 mode) This bit is set when an unrecoverable error is detected in a tuning circuit. By detecting Tuning Error, Host Driver needs to abort a command executing and perform tuning.
25 Reserved	This read-only field is reserved and always has the value 0.
24 AC12E	Auto CMD12 Error Occurs when detecting that one of the bits in the Auto CMD12 Error Status register has changed from 0 to 1. This bit is set to 1, not only when the errors in Auto CMD12 occur, but also when the Auto CMD12 is not executed due to the previous command error. 1 Error 0 No Error
23 Reserved	This read-only field is reserved and always has the value 0.
22 DEBE	Data End Bit Error Occurs either when detecting 0 at the end bit position of read data, which uses the DATA line, or at the end bit position of the CRC. This bit will be not asserted in tuning process. 1 Error 0 No Error

Table continues on the next page...

uSDHCx_INT_STATUS field descriptions (continued)

Field	Description
21 DCE	Data CRC Error Occurs when detecting a CRC error when transferring read data, which uses the DATA line, or when detecting the Write CRC status having a value other than 010. This bit will be not asserted in tuning process. 1 Error 0 No Error
20 DTOE	Data Timeout Error Occurs when detecting one of following time-out conditions. • Busy time-out for R1b, R5b type • Busy time-out after Write CRC status • Read Data time-out. This bit will be not asserted in tuning process. 1 Time out 0 No Error
19 CIE	Command Index Error Occurs if a Command Index error occurs in the command response. This bit will be not asserted in tuning process. 1 Error 0 No Error
18 CEBE	Command End Bit Error Occurs when detecting that the end bit of a command response is 0. This bit will be not asserted in tuning process. 1 End Bit Error Generated 0 No Error
17 CCE	Command CRC Error Command CRC Error is generated in two cases. • If a response is returned and the Command Timeout Error is set to 0 (indicating no time-out), this bit is set when detecting a CRC error in the command response. • The uSDHC detects a CMD line conflict by monitoring the CMD line when a command is issued. If the uSDHC drives the CMD line to 1, but detects 0 on the CMD line at the next SDCLK edge, then the uSDHC shall abort the command (Stop driving CMD line) and set this bit to 1. The Command Timeout Error shall also be set to 1 to distinguish CMD line conflict. This bit will be not asserted in tuning process. 1 CRC Error Generated. 0 No Error
16 CTOE	Command Timeout Error Occurs only if no response is returned within 64 SDCLK cycles from the end bit of the command. If the uSDHC detects a CMD line conflict, in which case a Command CRC Error shall also be set (as shown in Interrupt Status (uSDHC_INT_STATUS)), this bit shall be set without waiting for 64 SDCLK cycles. This is because the command will be aborted by the uSDHC.

Table continues on the next page...

uSDHCx_INT_STATUS field descriptions (continued)

Field	Description
	This bit will be not asserted in tuning process. 1 Time out 0 No Error
15 Reserved	This read-only field is reserved and always has the value 0.
14 TP	Tuning Pass:(only for SD3.0 SDR104 mode) Current CMD19 transfer is done successfully. That is, current sampling point is correct.
13 Reserved	This read-only field is reserved and always has the value 0.
12 RTE	Re-Tuning Event: (only for SD3.0 SDR104 mode) This status is set if Re-Tuning Request in the Present State register changes from 0 to 1. Host Controller requests Host Driver to perform re-tuning for next data transfer. Current data transfer (not large block count) can be completed without re-tuning. 1 Re-Tuning should be performed 0 Re-Tuning is not required
11–9 Reserved	This read-only field is reserved and always has the value 0.
8 CINT	Card Interrupt This status bit is set when an interrupt signal is detected from the external card. In 1-bit mode, the uSDHC will detect the Card Interrupt without the SD Clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so the interrupt from card can only be sampled during interrupt cycle, introducing some delay between the interrupt signal from the SDIO card and the interrupt to the Host System. Writing this bit to 1 can clear this bit, but as the interrupt source from the SDIO card does not clear, this bit is set again. In order to clear this bit, it is required to reset the interrupt source from the external card followed by a writing 1 to this bit. When this status has been set, and the Host Driver needs to service this interrupt, the Card Interrupt Signal Enable in the Interrupt Signal Enable register should be 0 to stop driving the interrupt signal to the Host System. After completion of the card interrupt service (It should reset the interrupt sources in the SDIO card and the interrupt signal may not be asserted), write 1 to clear this bit, set the Card Interrupt Signal Enable to 1, and start sampling the interrupt signal again. 1 Generate Card Interrupt 0 No Card Interrupt
7 CRM	Card Removal This status bit is set if the Card Inserted bit in the Present State register changes from 1 to 0. When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state may possibly be changed when the Host Driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if no card is inserted. In order to leave it cleared, clear the Card Removal Status Enable bit in Interrupt Status Enable register. 1 Card removed 0 Card state unstable or inserted
6 CINS	Card Insertion This status bit is set if the Card Inserted bit in the Present State register changes from 0 to 1. When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State

Table continues on the next page...

uSDHCx_INT_STATUS field descriptions (continued)

Field	Description
	register should be confirmed. Because the card state may possibly be changed when the Host Driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if a card is inserted. In order to leave it cleared, clear the Card Inserted Status Enable bit in Interrupt Status Enable register. 1 Card inserted 0 Card state unstable or removed
5 BRR	Buffer Read Ready This status bit is set if the Buffer Read Enable bit, in the Present State register, changes from 0 to 1. Refer to the Buffer Read Enable bit in the Present State register for additional information. This bit indicates that cmd19 is finished in tuning process. 1 Ready to read buffer 0 Not ready to read buffer
4 BWR	Buffer Write Ready This status bit is set if the Buffer Write Enable bit, in the Present State register, changes from 0 to 1. Refer to the Buffer Write Enable bit in the Present State register for additional information. 1 Ready to write buffer: 0 Not ready to write buffer
3 DINT	DMA Interrupt Occurs only when the internal DMA finishes the data transfer successfully. Whenever errors occur during data transfer, this bit will not be set. Instead, the DMAE bit will be set. Either Simple DMA or ADMA finishes data transferring, this bit will be set. 1 DMA Interrupt is generated 0 No DMA Interrupt
2 BGE	Block Gap Event If the Stop At Block Gap Request bit in the Protocol Control register is set, this bit is set when a read or write transaction is stopped at a block gap. If Stop At Block Gap Request is not set to 1, this bit is not set to 1. In the case of a Read Transaction: This bit is set at the falling edge of the DATA Line Active Status (When the transaction is stopped at SD Bus timing). The Read Wait must be supported in order to use this function. In the case of Write Transaction: This bit is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing). 1 Transaction stopped at block gap 0 No block gap event
1 TC	Transfer Complete This bit is set when a read or write transfer is completed. In the case of a Read Transaction: This bit is set at the falling edge of the Read Transfer Active Status. There are two cases in which this interrupt is generated. The first is when a data transfer is completed as specified by the data length (after the last data has been read to the Host System). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request bit in the Protocol Control register (after valid data has been read to the Host System).

Table continues on the next page...

uSDHCx_INT_STATUS field descriptions (continued)

Field	Description
	In the case of a Write Transaction: This bit is set at the falling edge of the DATA Line Active Status. There are two cases in which this interrupt is generated. The first is when the last data is written to the SD card as specified by the data length and the busy signal is released. The second is when data transfers are stopped at the block gap, by setting the Stop At Block Gap Request bit in the Protocol Control register, and the data transfers are completed. (after valid data is written to the SD card and the busy signal released). In the case of a command with busy, this bit is set when busy is deasserted. This bit will be not asserted in tuning process. 1 Transfer complete 0 Transfer not complete
0 CC	Command Complete This bit is set when you receive the end bit of the command response (except Auto CMD12). Refer to the Command Inhibit (CMD) in the Present State register. This bit will be not asserted in tuning process. 1 Command complete 0 Command not complete

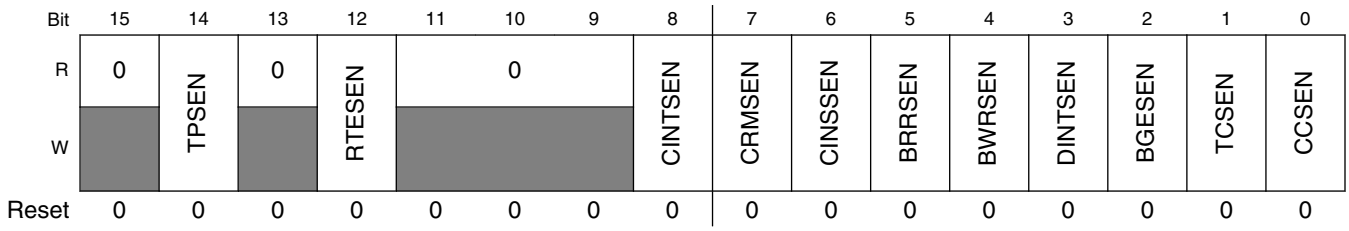
10.3.8.14 Interrupt Status Enable (uSDHCx_INT_STATUS_EN)

Setting the bits in this register to 1 enables the corresponding Interrupt Status to be set by the specified event. If any bit is cleared, the corresponding Interrupt Status bit is also cleared (i.e. when the bit in this register is cleared, the corresponding bit in Interrupt Status Register is always 0).

- Depending on IABG bit setting, uSDHC may be programmed to sample the card interrupt signal during the interrupt period and hold its value in the flip-flop. There will be some delays on the Card Interrupt, asserted from the card, to the time the Host System is informed.
- To detect a CMD line conflict, the Host Driver must set both Command Timeout Error Status Enable and Command CRC Error Status Enable to 1.

Address: Base address + 34h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			DMAESEN	0	TNESEN	0	AC12ESEN	0	DEBESEN	DCESEN	DTOESEN	CIESEN	CEBESEN	CCESEN	CTOESEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



uSDHCx_INT_STATUS_EN field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28 DMAESEN	DMA Error Status Enable 1 Enabled 0 Masked
27 Reserved	This read-only field is reserved and always has the value 0.
26 TNESEN	Tuning Error Status Enable 1 Enabled 0 Masked
25 Reserved	This read-only field is reserved and always has the value 0.
24 AC12ESEN	Auto CMD12 Error Status Enable 1 Enabled 0 Masked
23 Reserved	This read-only field is reserved and always has the value 0.
22 DEBESEN	Data End Bit Error Status Enable 1 Enabled 0 Masked
21 DCESEN	Data CRC Error Status Enable 1 Enabled 0 Masked
20 DTOESEN	Data Timeout Error Status Enable 1 Enabled 0 Masked
19 CIESEN	Command Index Error Status Enable 1 Enabled 0 Masked
18 CEBESEN	Command End Bit Error Status Enable 1 Enabled 0 Masked

Table continues on the next page...

uSDHCx_INT_STATUS_EN field descriptions (continued)

Field	Description
17 CCESEN	Command CRC Error Status Enable 1 Enabled 0 Masked
16 CTOESEN	Command Timeout Error Status Enable 1 Enabled 0 Masked
15 Reserved	This read-only field is reserved and always has the value 0.
14 TPSEN	Tuning Pass Status Enable 1 Enabled 0 Masked
13 Reserved	This read-only field is reserved and always has the value 0.
12 RTESEN	Re-Tuning Event Status Enable 1 Enabled 0 Masked
11–9 Reserved	This read-only field is reserved and always has the value 0.
8 CINTSEN	Card Interrupt Status Enable If this bit is set to 0, the uSDHC will clear the interrupt request to the system. The Card Interrupt detection is stopped when this bit is cleared and restarted when this bit is set to 1. The Host Driver should clear the Card Interrupt Status Enable before servicing the Card Interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts. 1 Enabled 0 Masked
7 CRMSEN	Card Removal Status Enable 1 Enabled 0 Masked
6 CINSSEN	Card Insertion Status Enable 1 Enabled 0 Masked
5 BRRSEN	Buffer Read Ready Status Enable 1 Enabled 0 Masked
4 BWRSEN	Buffer Write Ready Status Enable 1 Enabled 0 Masked
3 DINTSEN	DMA Interrupt Status Enable

Table continues on the next page...

uSDHCx_INT_STATUS_EN field descriptions (continued)

Field	Description
	1 Enabled 0 Masked
2 BGESEN	Block Gap Event Status Enable 1 Enabled 0 Masked
1 TCSEN	Transfer Complete Status Enable 1 Enabled 0 Masked
0 CCSEN	Command Complete Status Enable 1 Enabled 0 Masked

10.3.8.15 Interrupt Signal Enable (uSDHCx_INT_SIGNAL_EN)

This register is used to select which interrupt status is indicated to the Host System as the interrupt. These status bits all share the same interrupt line. Setting any of these bits to 1 enables interrupt generation. The corresponding Status register bit will generate an interrupt when the corresponding interrupt signal enable bit is set.

Address: Base address + 38h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				0		0		0							
W				DMAEIEN		TNEIEN		AC12EIEN		DEBEIEN	DCEIEN	DTOEIEN	CIEIEN	CEBEIEN	CCEIEN	CTOEIEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		0		0											
W		TPIEN		RTEIEN				CINTIEN	CRMIEN	CINSIEN	BRRIEN	BWRIEN	DINTIEN	BGEIEN	TCIEN	CCIEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_INT_SIGNAL_EN field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

uSDHCx_INT_SIGNAL_EN field descriptions (continued)

Field	Description
28 DMAEIEN	DMA Error Interrupt Enable 1 Enable 0 Masked
27 Reserved	This read-only field is reserved and always has the value 0.
26 TNEIEN	Tuning Error Interrupt Enable 1 Enabled 0 Masked
25 Reserved	This read-only field is reserved and always has the value 0.
24 AC12EIEN	Auto CMD12 Error Interrupt Enable 1 Enabled 0 Masked
23 Reserved	This read-only field is reserved and always has the value 0.
22 DEBEIEN	Data End Bit Error Interrupt Enable 1 Enabled 0 Masked
21 DCEIEN	Data CRC Error Interrupt Enable 1 Enabled 0 Masked
20 DTOEIEN	Data Timeout Error Interrupt Enable 1 Enabled 0 Masked
19 CIEIEN	Command Index Error Interrupt Enable 1 Enabled 0 Masked
18 CEBEIEN	Command End Bit Error Interrupt Enable 1 Enabled 0 Masked
17 CCEIEN	Command CRC Error Interrupt Enable 1 Enabled 0 Masked
16 CTOEIEN	Command Timeout Error Interrupt Enable 1 Enabled 0 Masked
15 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

uSDHCx_INT_SIGNAL_EN field descriptions (continued)

Field	Description
14 TPIEN	Tuning Pass Interrupt Enable 1 Enabled 0 Masked
13 Reserved	This read-only field is reserved and always has the value 0.
12 RTEIEN	Re-Tuning Event Interrupt Enable 1 Enabled 0 Masked
11–9 Reserved	This read-only field is reserved and always has the value 0.
8 CINTIEN	Card Interrupt Interrupt Enable 1 Enabled 0 Masked
7 CRMIEN	Card Removal Interrupt Enable 1 Enabled 0 Masked
6 CINSIEN	Card Insertion Interrupt Enable 1 Enabled 0 Masked
5 BRRIEN	Buffer Read Ready Interrupt Enable 1 Enabled 0 Masked
4 BWRIEN	Buffer Write Ready Interrupt Enable 1 Enabled 0 Masked
3 DINTIEN	DMA Interrupt Enable 1 Enabled 0 Masked
2 BGEIEN	Block Gap Event Interrupt Enable 1 Enabled 0 Masked
1 TCIEN	Transfer Complete Interrupt Enable 1 Enabled 0 Masked
0 CCIEN	Command Complete Interrupt Enable 1 Enabled 0 Masked

10.3.8.16 Auto CMD12 Error Status (uSDHCx_AUTOCMD12_ERR_STATUS)

When the Auto CMD12 Error Status bit in the Status register is set, the Host Driver shall check this register to identify what kind of error the Auto CMD12 / CMD 23 indicated. Auto CMD23 errors are indicated in bit 04-01.This register is valid only when the Auto CMD12 Error status bit is set.

The table bwlow shows the relationship between the Auto CMGD12 CRC Error and the Auto CMD12 Command Timeout Error.

Table 10-50. Relationship Between Command CRC Error and Command Timeout Error for Auto CMD12

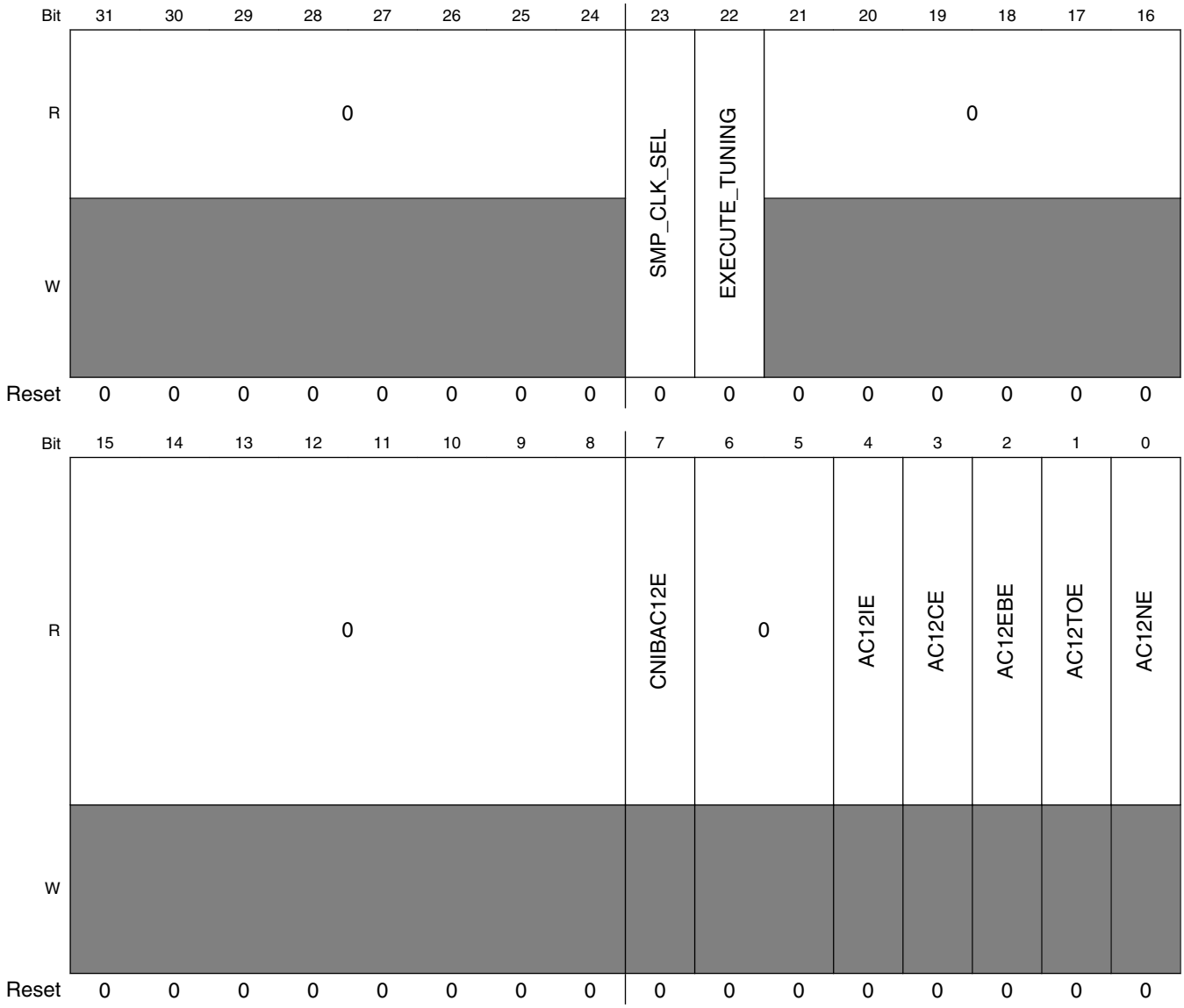
Auto CMD12 CRC Error	Auto CMD12 Timeout Error	Type of Error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD line conflict

Changes in Auto CMD12 Error Status register can be classified in three scenarios:

1. When the uSDHC is going to issue an Auto CMD12.
   - Set bit 0 to 1 if the Auto CMD12 can't be issued due to an error in the previous command
   - Set bit 0 to 0 if the Auto CMD12 is issued
2. At the end bit of an Auto CMD12 response.
   - Check errors correspond to bits 1-4.
   - Set bits 1-4 corresponding to detected errors.
   - Clear bits 1-4 corresponding to detected errors
3. Before reading the Auto CMD12 Error Status bit 7.
   - Set bit 7 to 1 if there is a command that can't be issued
   - Clear bit 7 if there is no command to issue

The timing for generating the Auto CMD12 Error and writing to the Command register are asynchronous. After that, bit 7 shall be sampled when the driver is not writing to the Command register. So it is suggested to read this register only when the AC12E bit in Interrupt Status register is set. An Auto CMD12 Error Interrupt is generated when one of the error bits (0-4) is set to 1. The Command Not Issued By Auto CMD12 Error does not generate an interrupt.

Address: Base address + 3Ch offset



uSDHC*x*_AUTOCMD12_ERR_STATUS field descriptions

Field	Description
31–24 Reserved	This read-only field is reserved and always has the value 0.
23 SMP_CLK_SEL	Sample Clock Select When std_tuning_en bit is set, this bit is used to select sampling clock to receive CMD and DATA. Otherwise, this bit is reserved. This bit is set by ty tuning procedure and valid after the completion of tuning(When Execute Tuning is cleared). Setting 1 means that tuning is completed successfully and setting 0 means that tuning is failed. Writing 1 to this bit is meaningless and ignored. A tuning circuit is reset by writing to 0. This bit can be cleared with setting Execute Tuning. Once the tuning circuit is reset, it will take time to complete tuning sequence. Therefore, Host Driver should keep this bit to 1 to perform re-tuning sequence to complete re-tuning sequence in a short time. Change of this bit is not allowed while the Host controller us receiving response or a read data block.

Table continues on the next page...

uSDHCx_AUTOCMD12_ERR_STATUS field descriptions (continued)

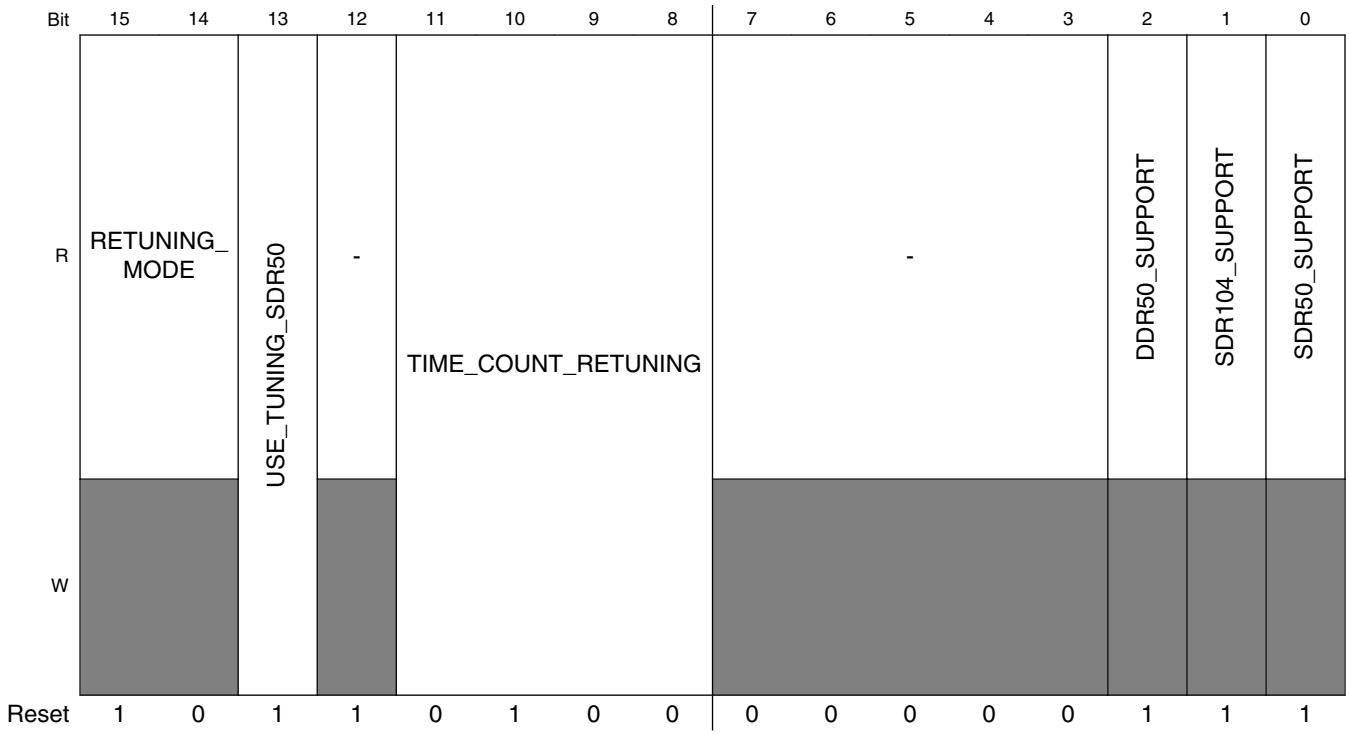
Field	Description
	1 Tuned clock is used to sample data 0 Fixed clock is used to sample data
22 EXECUTE_ TUNING	Execute Tuning When std_tuning_en bit is set, this bit is used to start tuning procedure. Otherwise, this bit is reserved. This bit is set to start tuning procedure and automatically cleared when runing procedure is completed. The result of tuning is indicated to sam_clk_sel bit. Tuning procedure is aborted by writing 0.
21–8 Reserved	This read-only field is reserved and always has the value 0.
7 CNIBAC12E	Command Not Issued By Auto CMD12 Error Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 Error (D04-D01) in this register. 1 Not Issued 0 No error
6–5 Reserved	This read-only field is reserved and always has the value 0.
4 AC12IE	Auto CMD12 / 23 Index Error Occurs if the Command Index error occurs in response to a command. 1 Error, the CMD index in response is not CMD12/23 0 No error
3 AC12CE	Auto CMD12 / 23 CRC Error Occurs when detecting a CRC error in the command response. 1 CRC Error Met in Auto CMD12/23 Response 0 No CRC error
2 AC12EBE	Auto CMD12 / 23 End Bit Error Occurs when detecting that the end bit of command response is 0 which should be 1. 1 End Bit Error Generated 0 No error
1 AC12TOE	Auto CMD12 / 23 Timeout Error Occurs if no response is returned within 64 SDCLK cycles from the end bit of the command. If this bit is set to1, the other error status bits (2-4) have no meaning. 1 Time out 0 No error
0 AC12NE	Auto CMD12 Not Executed If memory multiple block data transfer is not started, due to a command error, this bit is not set because it is not necessary to issue an Auto CMD12. Setting this bit to 1 means the uSDHC cannot issue the Auto CMD12 to stop a memory multiple block data transfer due to some error. If this bit is set to 1, other error status bits (1-4) have no meaning. 1 Not executed 0 Executed

10.3.8.17 Host Controller Capabilities (uSDHCx_HOST_CTRL_CAP)

This register provides the Host Driver with information specific to the uSDHC implementation. The value in this register is the power-on-reset value, and does not change with a software reset.

Address: Base address + 40h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0					VS18	VS30	VS33	SRS	DMAS	HSS	ADMAS	0	MBL[2:0]		
W																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1	1



uSDHCx_HOST_CTRL_CAP field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value 0.
26 VS18	Voltage Support 1.8 V This bit shall depend on the Host System ability. 1 1.8V supported 0 1.8V not supported
25 VS30	Voltage Support 3.0 V This bit shall depend on the Host System ability. 1 3.0V supported 0 3.0V not supported
24 VS33	Voltage Support 3.3V This bit shall depend on the Host System ability. 1 3.3V supported 0 3.3V not supported
23 SRS	Suspend / Resume Support This bit indicates whether the uSDHC supports Suspend / Resume functionality. If this bit is 0, the Suspend and Resume mechanism, as well as the Read Wait, are not supported, and the Host Driver shall not issue either Suspend or Resume commands. 1 Supported 0 Not supported

Table continues on the next page...

uSDHCx_HOST_CTRL_CAP field descriptions (continued)

Field	Description
22 DMAS	DMA Support This bit indicates whether the uSDHC is capable of using the internal DMA to transfer data between system memory and the data buffer directly. 1 DMA Supported 0 DMA not supported
21 HSS	High Speed Support This bit indicates whether the uSDHC supports High Speed mode and the Host System can supply a SD Clock frequency from 25 MHz to 50 MHz. 1 High Speed Supported 0 High Speed Not Supported
20 ADMAS	ADMA Support This bit indicates whether the uSDHC supports the ADMA feature. 1 Advanced DMA Supported 0 Advanced DMA Not supported
19 Reserved	This read-only field is reserved and always has the value 0.
18–16 MBL[2:0]	Max Block Length This value indicates the maximum block size that the Host Driver can read and write to the buffer in the uSDHC. The buffer shall transfer block size without wait cycles. 000 512 bytes 001 1024 bytes 010 2048 bytes 011 4096 bytes
15–14 RETUNING_ MODE	Retuning Mode This bit selects retuning method. 00 Mode 1 01 Mode 2 10 Mode 3 11 Reserved
13 USE_TUNING_ SDR50	Use Tuning for SDR50 This bit is set to 1. Host controller requires tuning to operate SDR50 1 SDR50 requires tuning 0 SDR does not require tuning
12 -	Reserved
11–8 TIME_COUNT_ RETUNING	Time Counter for Retuning This bit indicates an initial value of the Retuning Timer for Re-Tuning Mode1 and 3.Setting to 0 disables Retuning Timer.

Table continues on the next page...

uSDHCx_HOST_CTRL_CAP field descriptions (continued)

Field	Description
7–3 -	
2 DDR50_ SUPPORT	DDR50 support This bit indicates support of DDR50 mode.
1 SDR104_ SUPPORT	SDR104 support This bit indicates support of SDR104 mode.
0 SDR50_ SUPPORT	SDR50 support This bit indicates support of SDR50 mode.

10.3.8.18 Watermark Level (uSDHCx_WTMK_LVL)

Both write and read watermark levels (FIFO threshold) are configurable. There value can range from 1 to 128 words. Both write and read burst lengths are also Configurable. There value can range from 1 to 31 words.

Address: Base address + 44h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			WR_BRST_LEN[4:0]					WR_WML[7:0]								0			RD_BRST_LEN[4:0]					RD_WML[7:0]							
W																																
Reset	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0

uSDHCx_WTMK_LVL field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28–24 WR_BRST_ LEN[4:0]	Write Burst Length [1] The number of words the uSDHC writes in a single burst. The write burst length must be less than or equal to the write watermark level, and all bursts within a watermark level transfer will be in back-to-back mode. On reset, this field will be 8. Writing 0 to this field will result in '01000' (i.e. it is not able to clear this field).
23–16 WR_WML[7:0]	Write Watermark Level The number of words used as the watermark level (FIFO threshold) in a DMA write operation. Also the number of words as a sequence of write bursts in back-to-back mode. The maximum legal value for the write watermark level is 128.
15–13 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

uSDHCx_WTMK_LVL field descriptions (continued)

Field	Description
12–8 RD_BRST_ LEN[4:0]	Read Burst Length [2] The number of words the uSDHC reads in a single burst. The read burst length must be less than or equal to the read watermark level, and all bursts within a watermark level transfer will be in back-to-back mode. On reset, this field will be 8. Writing 0 to this field will result in '01000' (i.e. it is not able to clear this field).
RD_WML[7:0]	Read Watermark Level The number of words used as the watermark level (FIFO threshold) in a DMA read operation. Also the number of words as a sequence of read bursts in back-to-back mode. The maximum legal value for the read water mark level is 128.

1. Due to system restriction, the actual burst length may not exceed 16.

2. Due to system restriction, the actual burst length may not exceed 16.

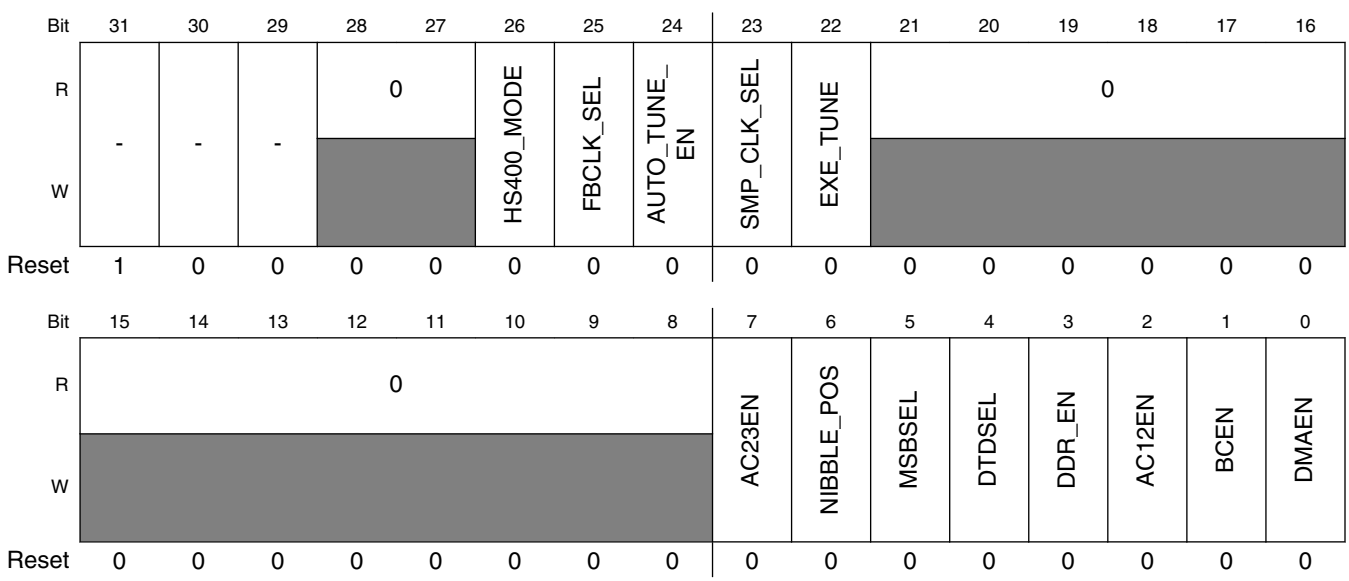
10.3.8.19 Mixer Control (uSDHCx_MIX_CTRL)

This register is used to DMA and data transfer. To prevent data loss, The software should check if data transfer is active before writing this register. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN, and DMAEN.

Table 10-51. Transfer Type Register Setting for Various Transfer Types

Multi/Single Block Select	Block Count Enable	Block Count	Function
0	Don't Care	Don't Care	Single Transfer
1	0	Don't Care	Infinite Transfer
1	1	Positive Number	Multiple Transfer
1	1	Zero	No Data Transfer

Address: Base address + 48h offset



uSDHCx_MIX_CTRL field descriptions

Field	Description
31 -	Reserved. Always write as 1
30 -	Reserved. Always write as 0.
29 -	Reserved. Always write as 0.
28–27 Reserved	This read-only field is reserved and always has the value 0.
26 HS400_MODE	Enable HS400 Mode HS400 Enable
25 FBCLK_SEL	Feedback Clock Source Selection (Only used for SD3.0, SDR104 mode) 1 Feedback clock comes from the ipp_card_clk_out 0 Feedback clock comes from the loopback CLK
24 AUTO_TUNE_EN	Auto Tuning Enable (Only used for SD3.0, SDR104 mode) 1 Enable auto tuning 0 Disable auto tuning
23 SMP_CLK_SEL	When STD_TUNING_EN is 0, this bit is used to select Tuned clock or Fixed clock to sample data / cmd (Only used for SD3.0, SDR104 mode) 1 Tuned clock is used to sample data / cmd 0 Fixed clock is used to sample data / cmd
22 EXE_TUNE	Execute Tuning: (Only used for SD3.0, SDR104 mode) When STD_TUNING_EN is 0, this bit is set to 1 to indicate the Host Driver is starting tuning procedure. Tuning procedure is aborted by writing 0. 1 Execute Tuning 0 Not Tuned or Tuning Completed
21–8 Reserved	This read-only field is reserved and always has the value 0.
7 AC23EN	Auto CMD23 Enable When this bit is set to 1, the Host Controller issues a CMD23 automatically before issuing a command specified in the Command Register.
6 NIBBLE_POS	In DDR 4-bit mode nibble position indictation. 0- the sequence is 'odd high nibble -> even high nibble -> odd low nibble -> even low nibble'; 1- the sequence is 'odd high nibble -> odd low nibble -> even high nibble -> even low nibble'.
5 MSBSEL	Multi / Single Block Select This bit enables multiple block DATA line data transfers. For any other commands, this bit can be set to 0. If this bit is 0, it is not necessary to set the Block Count register. (Refer to Command Transfer Type (uSDHC_CMD_XFR_TYP)). 1 Multiple Blocks 0 Single Block
4 DTDSEL	Data Transfer Direction Select

Table continues on the next page...

uSDHCx_MIX_CTRL field descriptions (continued)

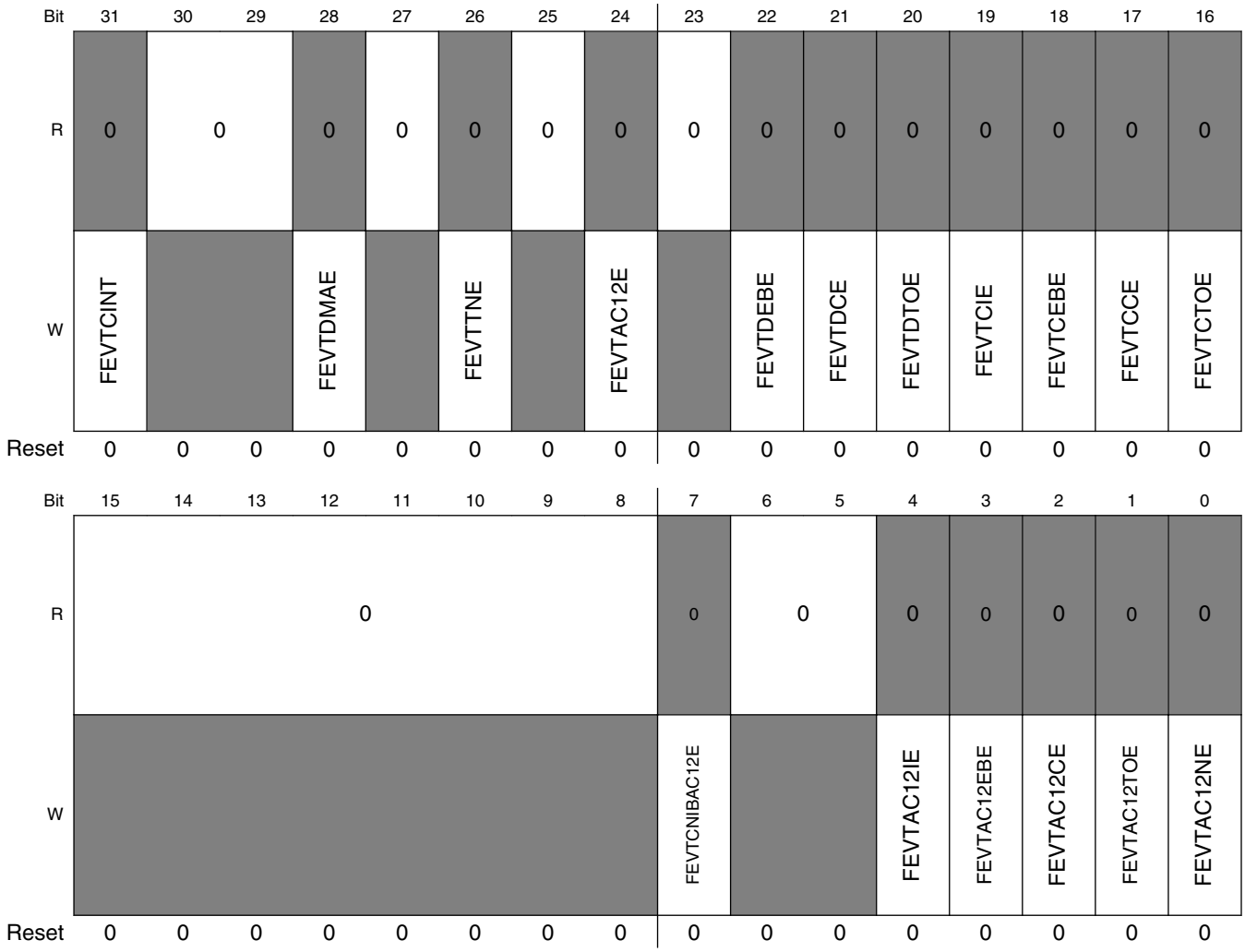
Field	Description
	This bit defines the direction of DATA line data transfers. The bit is set to 1 by the Host Driver to transfer data from the SD card to the uSDHC and is set to 0 for all other commands. 1 Read (Card to Host) 0 Write (Host to Card)
3 DDR_EN	Dual Data Rate mode selection
2 AC12EN	Auto CMD12 Enable Multiple block transfers for memory require a CMD12 to stop the transaction. When this bit is set to 1, the uSDHC will issue a CMD12 automatically when the last block transfer has completed. The Host Driver shall not set this bit to issue commands that do not require CMD12 to stop a multiple block data transfer. In particular, secure commands defined in File Security Specification (see reference list) do not require CMD12. In single block transfer, the uSDHC will ignore this bit no matter it is set or not. 1 Enable 0 Disable
1 BCEN	Block Count Enable This bit is used to enable the Block Count register, which is only relevant for multiple block transfers. When this bit is 0, the internal counter for block is disabled, which is useful in executing an infinite transfer. 1 Enable 0 Disable
0 DMAEN	DMA Enable This bit enables DMA functionality. If this bit is set to 1, a DMA operation shall begin when the Host Driver sets the DPSEL bit of this register. Whether the Simple DMA or the Advanced DMA is active depends on the DMA Select field of the Protocol Control register. 1 Enable 0 Disable

10.3.8.20 Force Event (uSDHCx_FORCE_EVENT)

The Force Event Register is not a physically implemented register. Rather, it is an address at which the Interrupt Status Register can be written if the corresponding bit of the Interrupt Status Enable Register is set. This register is a write only register and writing 0 to it has no effect. Writing 1 to this register actually sets the corresponding bit of Interrupt Status Register. A read from this register always results in 0's. In order to change the corresponding status bits in the Interrupt Status Register, make sure to set IPGEN bit in System Control Register so that IPG_CLK is always active.

Forcing a card interrupt will generate a short pulse on the DATA1 line, and the driver may treat this interrupt as a normal interrupt. The interrupt service routine may skip polling the card interrupt factor as the interrupt is self cleared.

Address: Base address + 50h offset



uSDHC*x*_FORCE_EVENT field descriptions

Field	Description
31 FEVTCINT	Force Event Card Interrupt Writing 1 to this bit generates a short low-level pulse on the internal DATA1 line, as if a self clearing interrupt was received from the external card. If enabled, the CINT bit will be set and the interrupt service routine may treat this interrupt as a normal interrupt from the external card.
30–29 Reserved	This read-only field is reserved and always has the value 0.
28 FEVTDMAE	Force Event DMA Error Forces the DMAE bit of Interrupt Status Register to be set.
27 Reserved	This read-only field is reserved and always has the value 0.
26 FEVTTNE	Force Tuning Error Forces the TNE bit of Interrupt Status Register to be set.
25 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

uSDHCx_FORCE_EVENT field descriptions (continued)

Field	Description
24 FEVTAC12E	Force Event Auto Command 12 Error Forces the AC12E bit of Interrupt Status Register to be set.
23 Reserved	This read-only field is reserved and always has the value 0.
22 FEVTDEBE	Force Event Data End Bit Error Forces the DEBE bit of Interrupt Status Register to be set.
21 FEVTDCE	Force Event Data CRC Error Forces the DCE bit of Interrupt Status Register to be set.
20 FEVTDTOE	Force Event Data Time Out Error Force the DTOE bit of Interrupt Status Register to be set.
19 FEVTCIE	Force Event Command Index Error Forces the CCE bit of Interrupt Status Register to be set.
18 FEVTCEBE	Force Event Command End Bit Error Forces the CEBE bit of Interrupt Status Register to be set.
17 FEVTCCE	Force Event Command CRC Error Forces the CCE bit of Interrupt Status Register to be set.
16 FEVTCTOE	Force Event Command Time Out Error Forces the CTOE bit of Interrupt Status Register to be set.
15–8 Reserved	This read-only field is reserved and always has the value 0.
7 FEVTCNIBAC12E	Force Event Command Not Executed By Auto Command 12 Error Forces the CNIBAC12E bit in the Auto Command12 Error Status Register to be set.
6–5 Reserved	This read-only field is reserved and always has the value 0.
4 FEVTAC12IE	Force Event Auto Command 12 Index Error Forces the AC12IE bit in the Auto Command12 Error Status Register to be set.
3 FEVTAC12EBE	Force Event Auto Command 12 End Bit Error Forces the AC12EBE bit in the Auto Command12 Error Status Register to be set.
2 FEVTAC12CE	Force Event Auto Command 12 CRC Error Forces the AC12CE bit in the Auto Command12 Error Status Register to be set.
1 FEVTAC12TOE	Force Event Auto Command 12 Time Out Error Forces the AC12TOE bit in the Auto Command12 Error Status Register to be set.
0 FEVTAC12NE	Force Event Auto Command 12 Not Executed Forces the AC12NE bit in the Auto Command12 Error Status Register to be set.

10.3.8.21 ADMA Error Status Register (uSDHCx_ADMA_ERR_STATUS)

When an ADMA Error Interrupt has occurred, the ADMA Error States field in this register holds the ADMA state and the ADMA System Address register holds the address around the error descriptor.

For recovering from this error, the Host Driver requires the ADMA state to identify the error descriptor address as follows:

- ST_STOP: Previous location set in the ADMA System Address register is the error descriptor address.
- ST_FDS: Current location set in the ADMA System Address register is the error descriptor address.
- ST_CADR: This state is never set because it only increments the descriptor pointer and doesn't generate an ADMA error.
- ST_TFR: Previous location set in the ADMA System Address register is the error descriptor address.

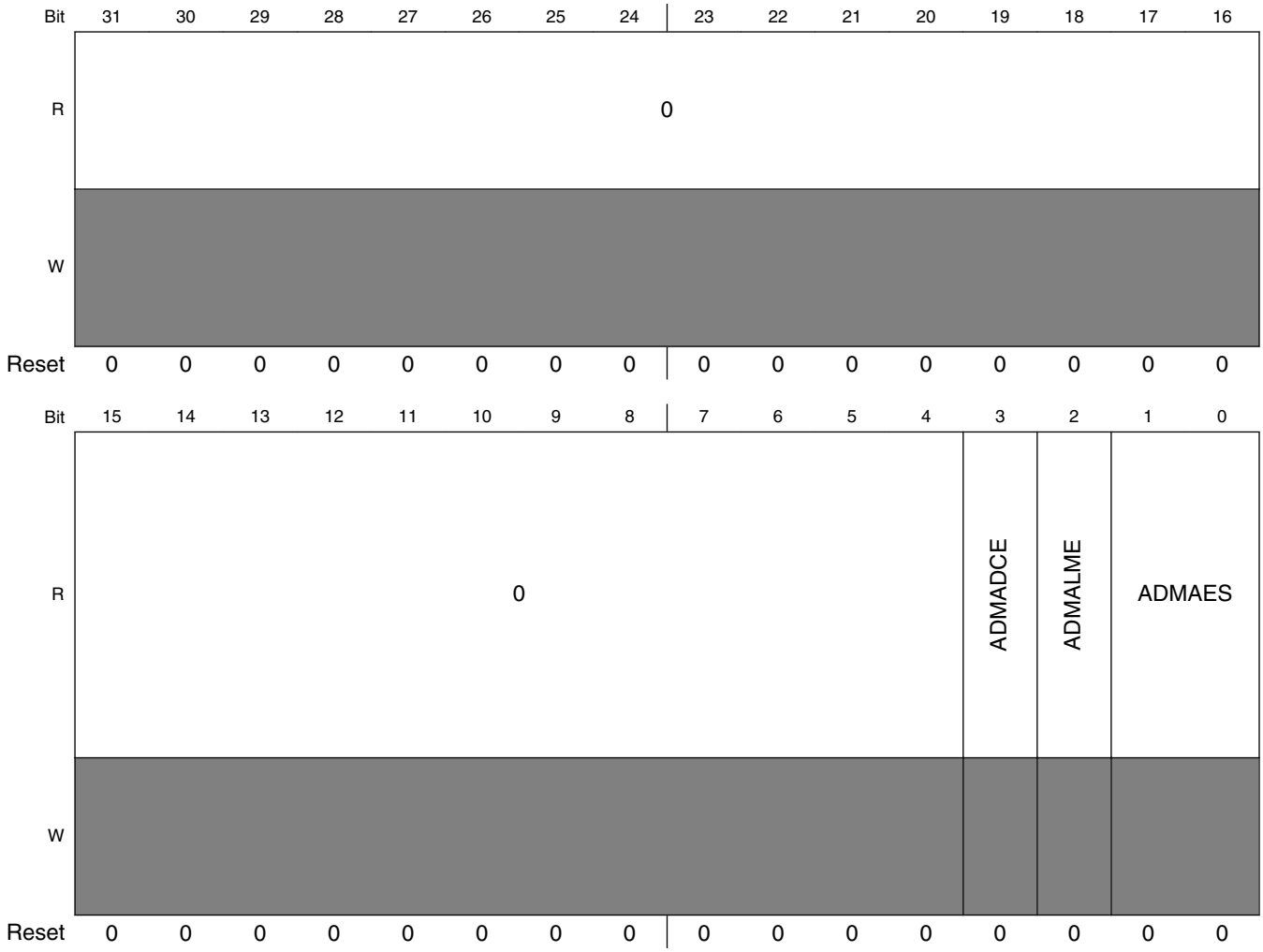
In case of a write operation, the Host Driver should use the ACMD22 to get the number of the written block, rather than using this information, since unwritten data may exist in the Host Controller.

The Host Controller generates the ADMA Error Interrupt when it detects invalid descriptor data (Valid=0) in the ST_FDS state. The Host Driver can distinguish this error by reading the Valid bit of the error descriptor.

Table 10-52. ADMA Error State Coding

D01-D00	ADMA Error State (when error has occurred)	Contents of ADMA System Address Register
00	ST_STOP (Stop DMA)	Holds the address of the next executable Descriptor command
01	ST_FDS (Fetch Descriptor)	Holds the valid Descriptor address
10	ST_CADR (Change Address)	No ADMA Error is generated
11	ST_TFR (Transfer Data)	Holds the address of the next executable Descriptor command

Address: Base address + 54h offset



uSDHCx_ADMA_ERR_STATUS field descriptions

Field	Description
31–4 Reserved	This read-only field is reserved and always has the value 0.
3 ADMADCE	ADMA Descritor Error This error occurs when invalid descriptor fetched by ADMA. 1 Error 0 No Error
2 ADMALME	ADMA Length Mismatch Error This error occurs in the following 2 cases: • While the Block Count Enable is being set, the total data length specified by the Descriptor table is different from that specified by the Block Count and Block Length. • Total data length cannot be divided by the block length. 1 Error 0 No Error
ADMAES	ADMA Error State (when ADMA Error is occurred)

Table continues on the next page...

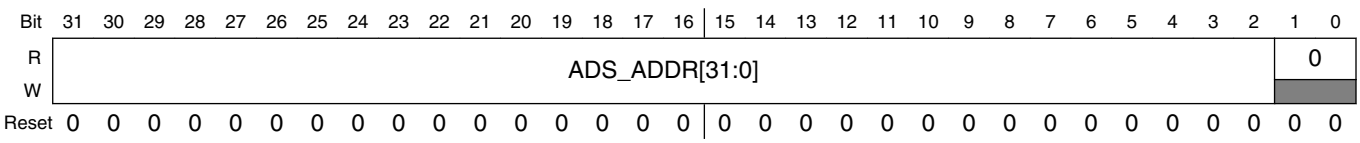
uSDHCx_ADMA_ERR_STATUS field descriptions (continued)

Field	Description
	This field indicates the state of the ADMA when an error has occurred during an ADMA data transfer. Refer to ADMA Error Status Register (uSDHC_ADMA_ERR_STATUS) for more details.

10.3.8.22 ADMA System Address (uSDHCx_ADMA_SYS_ADDR)

This register contains the physical system memory address used for ADMA transfers.

Address: Base address + 58h offset



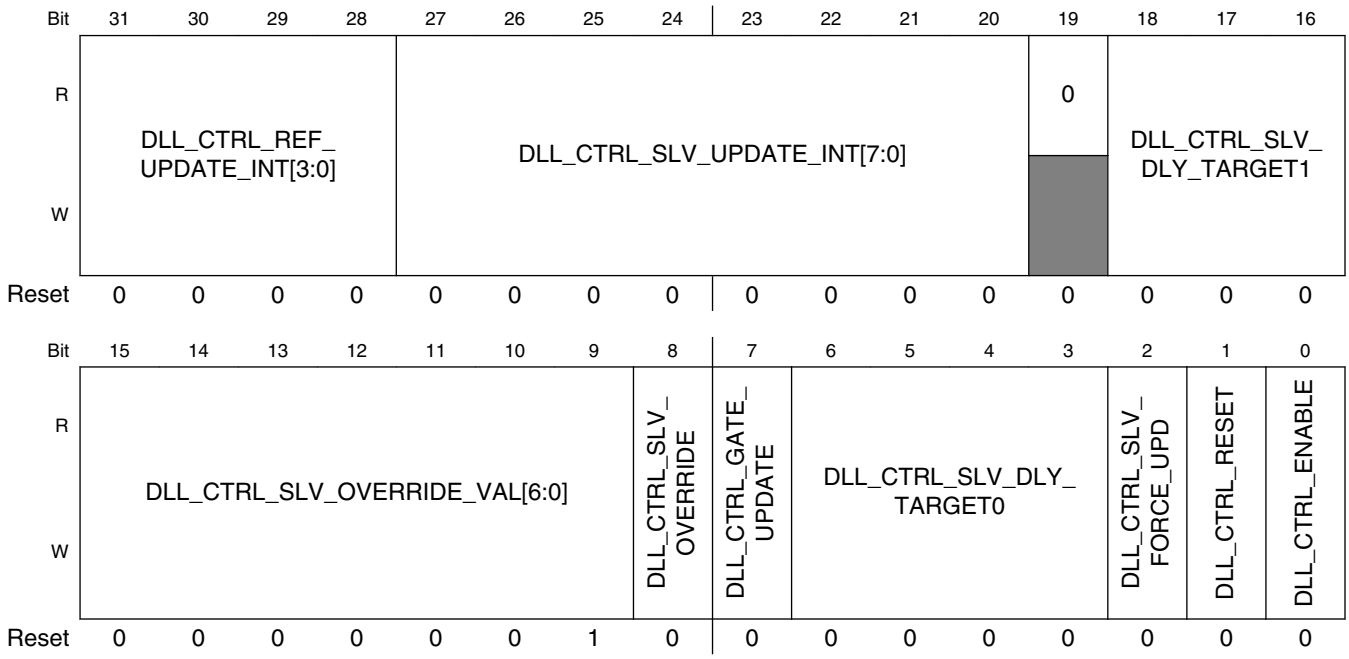
uSDHCx_ADMA_SYS_ADDR field descriptions

Field	Description
31–2 ADS_ ADDR[31:0]	ADMA System Address This register holds the word address of the executing command in the Descriptor table. At the start of ADMA, the Host Driver shall set the start address of the Descriptor table. The ADMA engine increments this register address whenever fetching a Descriptor command. When the ADMA is stopped at the Block Gap, this register indicates the address of the next executable Descriptor command. When the ADMA Error Interrupt is generated, this register shall hold the valid Descriptor address depending on the ADMA state. The lower 2 bits of this register is tied to '0' so the ADMA address is always word aligned. Since this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so SW cannot change this register when TC bit is set. Such restriction is also listed in Software Restrictions .
Reserved	This read-only field is reserved and always has the value 0.

10.3.8.23 DLL (Delay Line) Control (uSDHCx_DLL_CTRL)

This register contains control bits for DLL.

Address: Base address + 60h offset



uSDHCx_DLL_CTRL field descriptions

Field	Description
31–28 DLL_CTRL_REF_UPDATE_INT[3:0]	DLL control loop update interval. The interval cycle is (2 + REF_UPDATE_INT) * REF_CLOCK. By default, the DLL control loop shall update every two REF_CLOCK cycles. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature)
27–20 DLL_CTRL_SLV_UPDATE_INT[7:0]	Slave delay line update interval. If default 0 is used, it means 256 cycles of REF_CLOCK. A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line will also update automatically when the reference DLL transitions to a locked state (from an un-locked state).
19 Reserved	This read-only field is reserved and always has the value 0.
18–16 DLL_CTRL_SLV_DLY_TARGET1	Refer to DLL_CTRL_SLV_DLY_TARGET0 below.
15–9 DLL_CTRL_SLV_OVERRIDE_VAL[6:0]	When SLV_OVERRIDE = 1 This field is used to select 1 of 128 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 128.

Table continues on the next page...

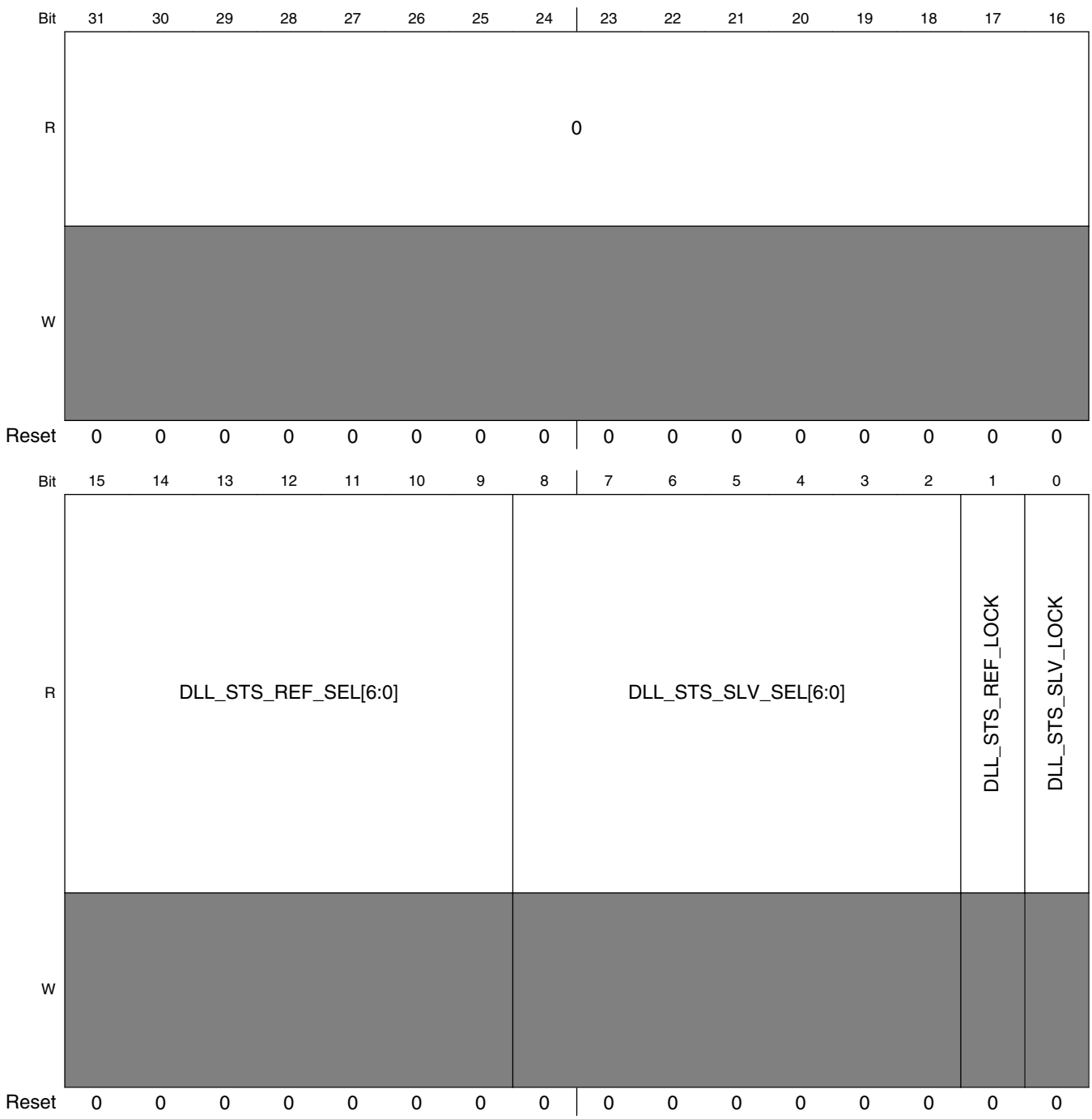
uSDHC*x*_DLL_CTRL field descriptions (continued)

Field	Description
8 DLL_CTRL_ SLV_OVERRIDE	Set this bit to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to be enabled using the ENABLE bit. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE = 0
7 DLL_CTRL_ GATE_UPDATE	Set this bit to 1 to prevent the DLL from updating (since when clock_in exists, glitches may appear during DLL updates). This bit may be used by software if such a condition occurs. Clear the bit to 0 to allow the DLL to update automatically.
6–3 DLL_CTRL_ SLV_DLY_ TARGET0	The delay target for the uSDHC loopback read clock can be programmed in 1/16th increments of an ref_clock half-period. The delay is ((DLL_CTRL_SLV_DLY_TARGET1 +1)* REF_CLOCK / 2) / 16 So the input read-clock can be delayed relative input data from (REF_CLOCK / 2) / 16 to REF_CLOCK * 4.
2 DLL_CTRL_ SLV_FORCE_ UPD	Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line shall update automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered). Be sure to use it when uSDHC is idle. This function may not work when uSDHC is working on data / cmd / response.
1 DLL_CTRL_ RESET	Setting this bit to 1 force a reset on DLL. This will cause the DLL to lose lock and re-calibrate to detect an REF_CLOCK half period phase shift. This signal is used by the DLL as edge-sensitive, so in order to create a subsequent reset, RESET must be taken low and then asserted again.
0 DLL_CTRL_ ENABLE	Set this bit to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled.

10.3.8.24 DLL Status (uSDHC*x*_DLL_STATUS)

This register contains the DLL status information. All bits are read only and will read the same as the power-reset value.

Address: Base address + 64h offset



uSDHC*x*_DLL_STATUS field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–9 DLL_STS_REF_SEL[6:0]	Reference delay line select taps. This is encoded by 7 bits for 127 taps.
8–2 DLL_STS_SLV_SEL[6:0]	Slave delay line select status. This is the instant value generated from reference chain. Since the reference chain can only be updated when REF_CLOCK is detected, this value should be the right value to be updated when the reference is locked.
1 DLL_STS_REF_LOCK	Reference DLL lock status. This signifies that the DLL has detected and locked to a half-phase ref_clock shift, allowing the slave delay-line to perform programmed clock delays
0 DLL_STS_SLV_LOCK	Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value

10.3.8.25 CLK Tuning Control and Status (uSDHC*x*_CLK_TUNE_CTRL_STATUS)

This register contains the Clock Tuning Control status information. All bits are read only and will read the same as the power-reset value. This register is added to support SD3.0 UHS-I SDR104 mode.

Address: Base address + 68h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRE_ERR	TAP_SEL_PRE[6:0]							TAP_SEL_OUT[3:0]				TAP_SEL_POST[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NXT_ERR	DLY_CELL_SET_PRE[6:0]							DLY_CELL_SET_OUT[6:0]				DLY_CELL_SET_POST[6:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHC*x*_CLK_TUNE_CTRL_STATUS field descriptions

Field	Description
31 PRE_ERR	PRE error which means the number of delay cells added on the feedback clock is too small. It is valid only when SMP_CLK_SEL of Mix control register (bit23 of 0x48) is enabled.

Table continues on the next page...

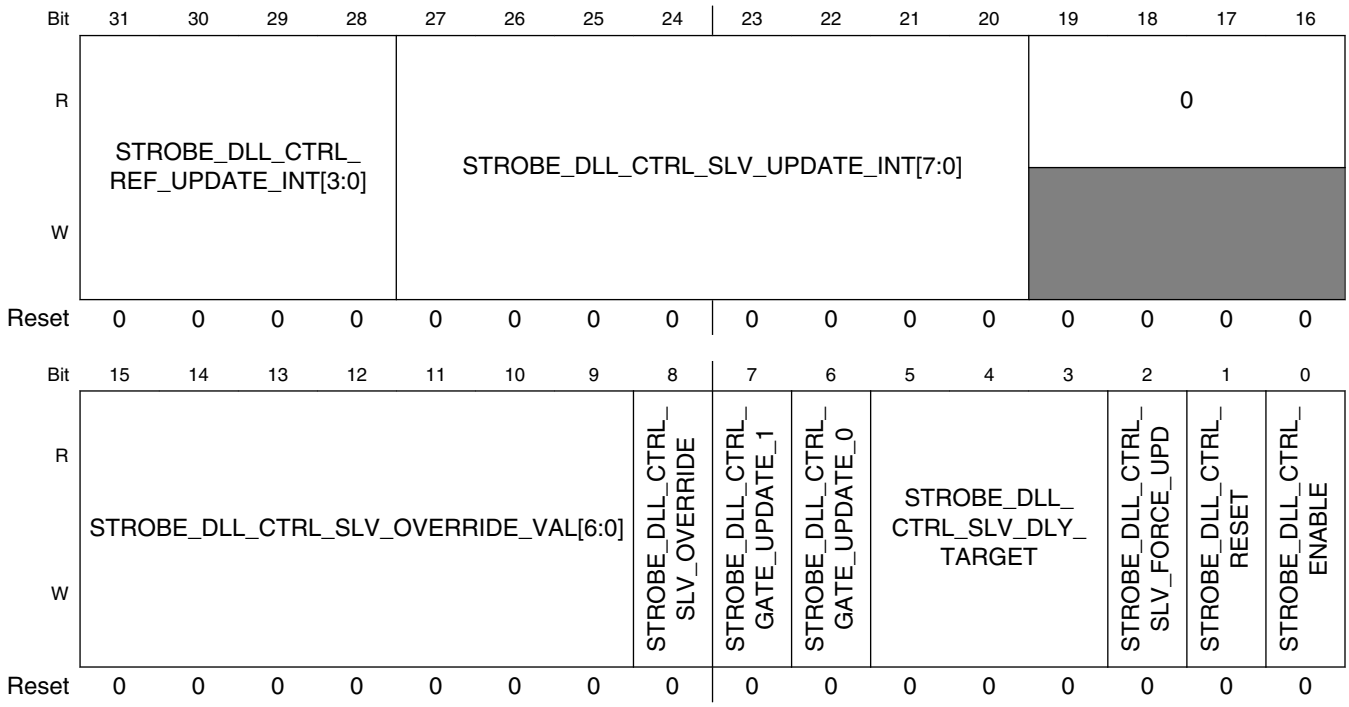
uSDHCx_CLK_TUNE_CTRL_STATUS field descriptions (continued)

Field	Description
30–24 TAP_SEL_ PRE[6:0]	Reflects the number of delay cells added on the feedback clock between the feedback clock and CLK_PRE. When AUTO_TUNE_EN (bit24 of 0x48) is disabled, TAP_SEL_PRE is always equal to DLY_CELL_SET_PRE. When AUTO_TUNE_EN (bit24 of 0x48) is enabled, TAP_SEL_PRE will be updated automatically according to the status of the auto tuning circuit to adjust the sample clock phase.
23–20 TAP_SEL_ OUT[3:0]	Reflect the number of delay cells added on the feedback clock between CLK_PRE and CLK_OUT.
19–16 TAP_SEL_ POST[3:0]	Reflect the number of delay cells added on the feedback clock between CLK_OUT and CLK_POST.
15 NXT_ERR	NXT error which means the number of delay cells added on the feedback clock is too large. It's valid only when SMP_CLK_SEL of Mix control register (bit23 of 0x48) is enabled.
14–8 DLY_CELL_ SET_PRE[6:0]	Set the number of delay cells on the feedback clock between the feedback clock and CLK_PRE.
7–4 DLY_CELL_ SET_OUT[6:0]	Set the number of delay cells on the feedback clock between CLK_PRE and CLK_OUT.
DLY_CELL_ SET_POST[6:0]	Set the number of delay cells on the feedback clock between CLK_OUT and CLK_POST.

10.3.8.26 Strobe DLL Control (uSDHCx_STROBE_DLL_CTRL)

This register contains the strobe DLL Control information.

Address: Base address + 70h offset



uSDHCx_STROBE_DLL_CTRL field descriptions

Field	Description
31–28 STROBE_DLL_ CTRL_REF_ UPDATE_ INT[3:0]	Strobe DLL Control Reference Update Interval The interval cycle is (2 + STROBE_REF_UPDATE_INT) * STROBE_REF_CLOCK. By default, the DLL control loop shall update every two STROBE_REF_CLOCK cycles. **NOTE:** Increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature)
27–20 STROBE_DLL_ CTRL_SLV_ UPDATE_ INT[7:0]	Strobe DLL Control Slave Update Interval Slave delay line update interval. If default 0 is used, it means 256 cycles of STROBE_REF_CLOCK. A value of 0x0F results in 15 cycles and so on. **NOTE:** software can always cause an update of the slave-delay line using the STROBE_SLV_FORCE_UPDATE register. The slave delay line will also update automatically when the reference DLL transitions to a locked state (from an un-locked state).
19–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

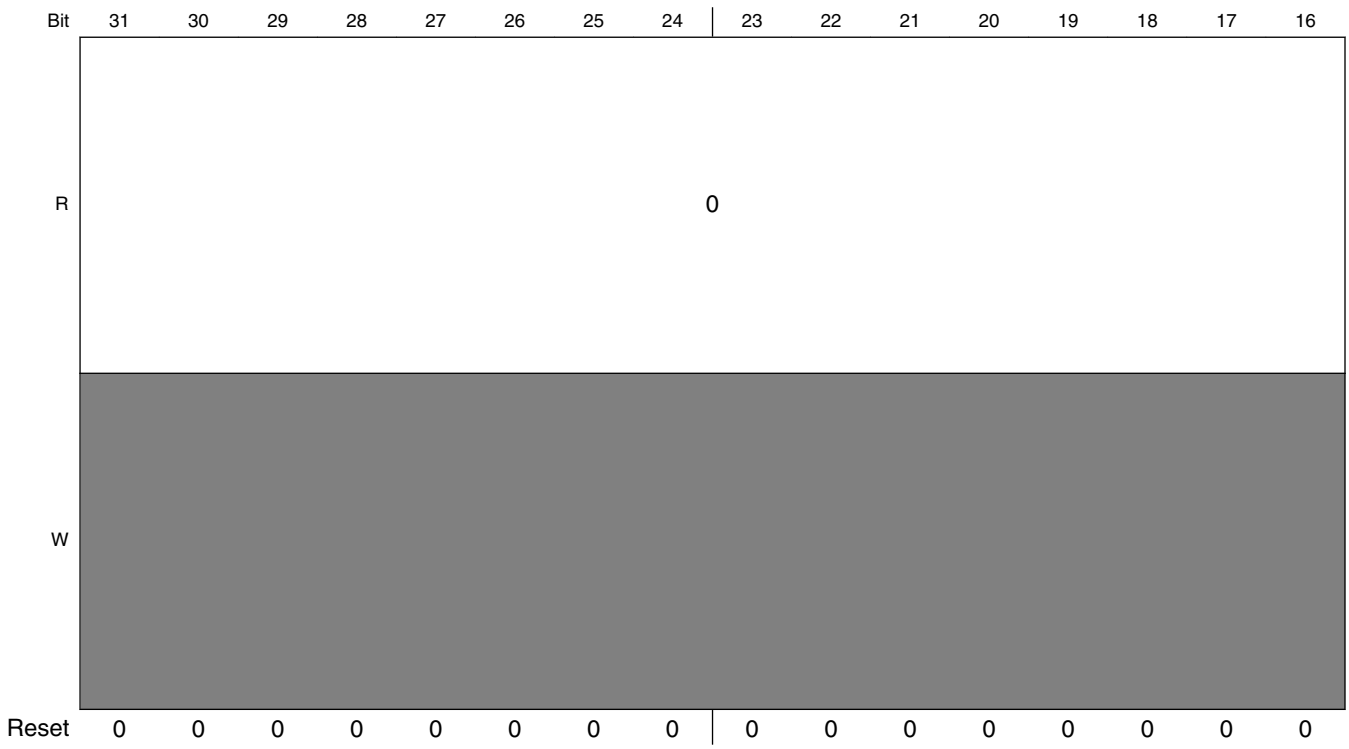
uSDHCx_STROBE_DLL_CTRL field descriptions (continued)

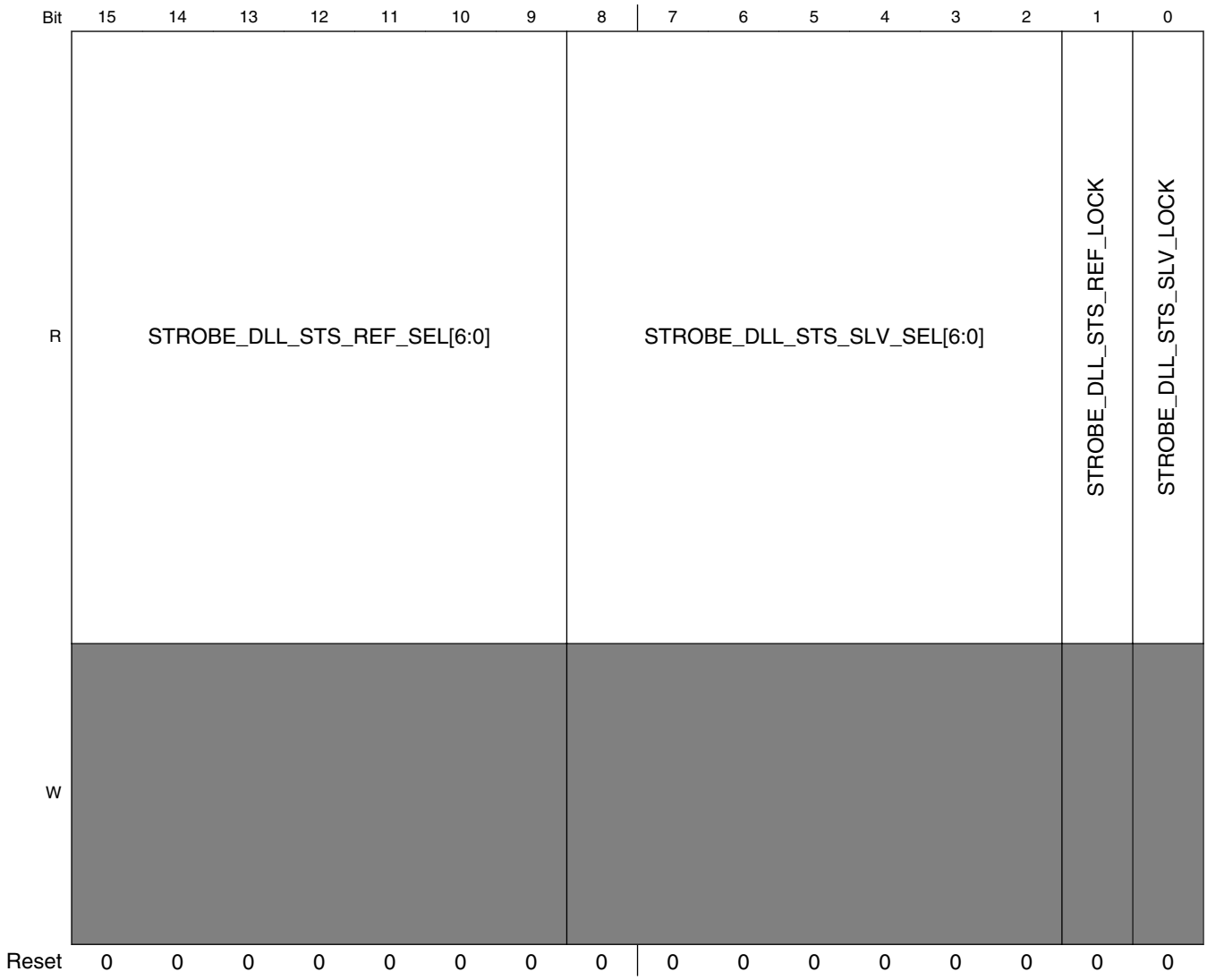
Field	Description
15–9 STROBE_DLL_CTRL_SLV_OVERRIDE_VAL[6:0]	Strobe DLL Control Slave Override Value When STROBE_SLV_OVERRIDE = 1, this field is used to manually select one of 128 physical taps. A value of 0 selects tap 1, and a value of 0x7F selects tap 128.
8 STROBE_DLL_CTRL_SLV_OVERRIDE	Strobe DLL Control Slave Override Set this bit to 1 to Enable manual override for slave delay chain using STROBE_SLV_OVERRIDE_VAL; set this bit to 0 to disable manual override. This feature does not require the DLL to be enabled using the ENABLE bit. In fact to reduce power, if STROBE_SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE = 0.
7 STROBE_DLL_CTRL_GATE_UPDATE_1	Strobe DLL Control Gate Update Set this bit to 1 to prevent the DLL from updating (since when STROBE_CLOCK_IN exists, glitches may appear during DLL updates). This bit can be used by software if such a condition occurs. Clear the bit to 0 to allow the DLL to update automatically.
6 STROBE_DLL_CTRL_GATE_UPDATE_0	Strobe DLL Control Gate Update Set this bit to 1 to prevent the DLL from updating (since when STROBE_CLOCK_IN exists, glitches may appear during DLL updates). This bit can be used by software if such a condition occurs. Clear the bit to 0 to allow the DLL to update automatically.
5–3 STROBE_DLL_CTRL_SLV_DLY_TARGET	Strobe DLL Control Slave Delay Target The delay target for the uSDHC loopback read clock can be programmed in 1/16th increments of an STROBE_REF_CLOCK half-period. The delay is ((STROBE_DLL_CTRL_SLV_DLY_TARGET + 1) * STROBE_REF_CLOCK / 2) / 16, So the input read-clock can be delayed relative input data from (STROBE_REF_CLOCK / 2) / 16 to (STROBE_REF_CLOCK *4)/16.
2 STROBE_DLL_CTRL_SLV_FORCE_UPD	Strobe DLL Control Slave Force Updated Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line should automatically update the STROBE_SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if STROBE_SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered). Be sure to use it when uSDHC is idle. This function may not work when uSDHC is working on data / cmd / response.
1 STROBE_DLL_CTRL_RESET	Strobe DLL Control Reset Setting this bit to 1 to force a reset on DLL. This will cause the DLL to lose lock and re-calibrate to detect an REF_CLOCK half period phase shift. This signal is used by the DLL as edge-sensitive, in order to create a subsequent reset, RESET must be taken low and then asserted again.
0 STROBE_DLL_CTRL_ENABLE	Strobe DLL Control Enable Set this bit to 1 to enable the DLL and delay chain; otherwise, set to 0 to bypasses DLL. **NOTE:** Using the slave delay line override feature with STROBE_SLV_OVERRIDE and STROBE_SLV_OVERRIDE VAL, the DLL does not need to be enabled.

10.3.8.27 Strobe DLL Status (uSDHCx_STROBE_DLL_STATUS)

This register contains the strobe DLL status information. All bits are read only and read the same as the power-reset value.

Address: Base address + 74h offset





uSDHCx_STROBE_DLL_STATUS field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–9 STROBE_DLL_STS_REF_SEL[6:0]	Strobe DLL Status Reference Select Reference delay line select taps. This is encoded by 7 bits for 127 taps.
8–2 STROBE_DLL_STS_SLV_SEL[6:0]	Strobe DLL Status Slave Select Slave delay line select status. This is the instant value generated from reference chain. Since the reference chain can only be updated when STROBE_REF_CLOCK is detected, this value can be updated with the right value when the reference is locked.
1 STROBE_DLL_STS_REF_LOCK	Strobe DLL Status Reference Lock This signifies that the DLL has detected and locked to a half-phase REF_CLOCK shift, it allows the slave delay-line to perform programmed clock delays.

Table continues on the next page...

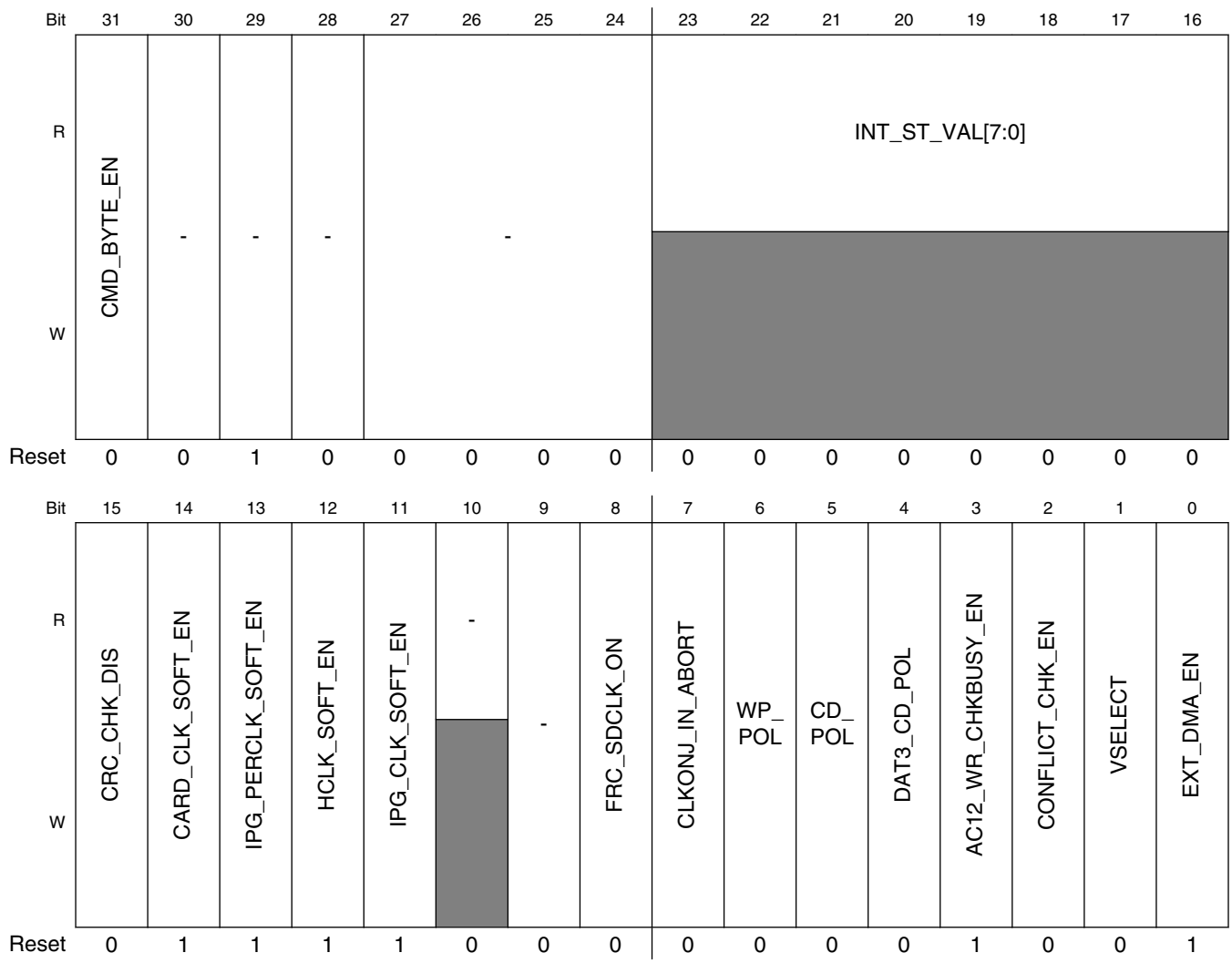
uSDHCx_STROBE_DLL_STATUS field descriptions (continued)

Field	Description
0 STROBE_DLL_ STS_SLV_LOCK	Strobe DLL Status Slave Lock Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line, and the slave-delay line is implementing the programmed delay value.

10.3.8.28 Vendor Specific Register (uSDHCx_VEND_SPEC)

This register contains the vendor specific control / status register.

Address: Base address + C0h offset



uSDHCx_VEND_SPEC field descriptions

Field	Description
31 CMD_BYTE_EN	Byte access 0 Disable 1 Enable
30 -	Reserved. Always write as 0.
29 -	Reserved. Always write as 1.
28 -	Reserved. Always write as 0.
27–24 -	Reserved. Always write as 4'b0000.
23–16 INT_ST_VAL[7:0]	Internal State Value Internal state value, reflecting the corresponding state value selected by Debug Select field. This field is read-only and write to this field does not have effect.
15 CRC_CHK_DIS	CRC Check Disable 0 Check CRC16 for every read data packet and check CRC bits for every write data packet 1 Ignore CRC16 check for every read data packet and ignore CRC bits check for every write data packet
14 CARD_CLK_ SOFT_EN	Card Clock Software Enable 0 Gate off the sd_clk 1 Enable the sd_clk
13 IPG_PERCLK_ SOFT_EN	IPG_PERCLK Software Enable 0 Gate off the IPG_PERCLK 1 Enable the IPG_PERCLK
12 HCLK_SOFT_EN	AHB Clock Software Enable **NOTE:** Hardware auto-enables the AHB clock when the internal DMA is enabled even if HCLK_SOFT_EN is 0. 0 Gate off the AHB clock. 1 Enable the AHB clock.
11 IPG_CLK_ SOFT_EN	IPG_CLK Software Enable 0 Gate off the IPG_CLK 1 Enable the IPG_CLK
10 -	Reserved. Always write as 0.
9 -	Reserved. Always write as 0.
8 FRC_SDCLK_ ON	Force CLK output active 0 CLK active or inactive is fully controlled by the hardware. 1 Force CLK active.

Table continues on the next page...

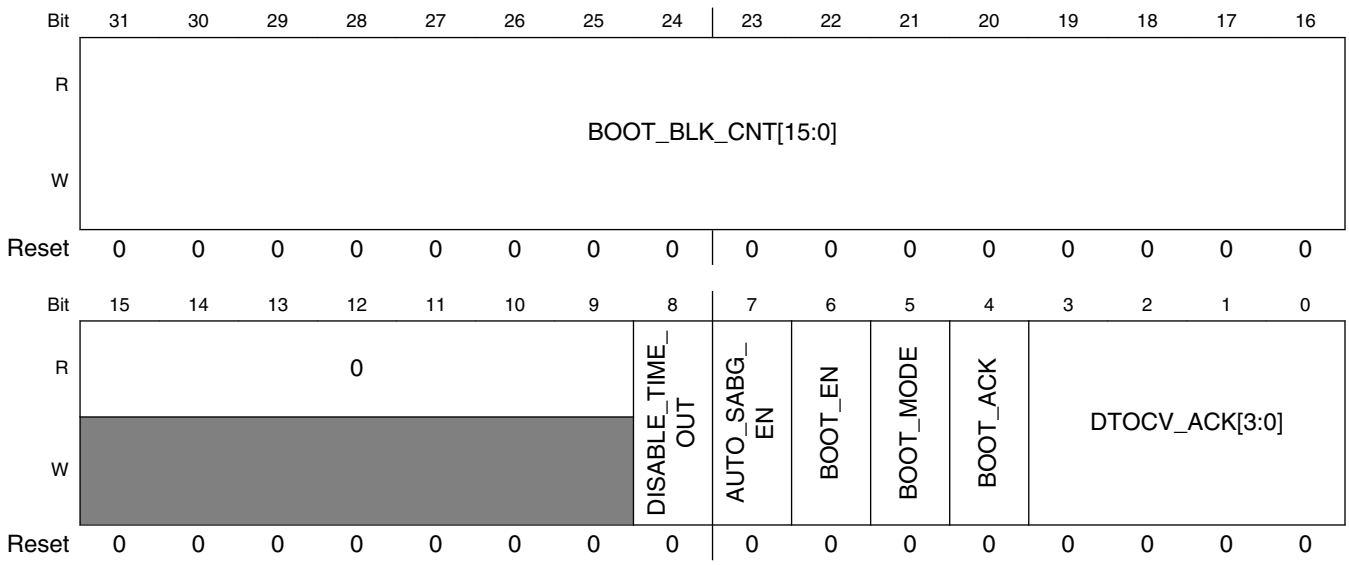
uSDHC*x*_VEND_SPEC field descriptions (continued)

Field	Description
7 CLKONJ_IN_ABORT	Only for debug. Force CLK output active when sending Abort command: 0 The CLK output is active when sending abort command while data is transmitting even if the internal FIFO is full (for read) or empty (for write). 1 The CLK output is inactive when sending abort command while data is transmitting if the internal FIFO is full (for read) or empty (for write).
6 WP_POL	Only for debug. Polarity of the WP pin: 0 WP pin is high active. 1 WP pin is low active.
5 CD_POL	Only for debug. Polarity of the CD_B pin: 0 CD_B pin is low active. 1 CD_B pin is high active.
4 DAT3_CD_POL	Only for debug. Polarity of DATA3 pin when it is used as card detection. 0 Card detected when DATA3 is high. 1 Card detected when DATA3 is low.
3 AC12_WR_CHKBUSY_EN	Check busy enable after auto CMD12 for write data packet 0 Do not check busy after auto CMD12 for write data packet 1 Check busy after auto CMD12 for write data packet
2 CONFLICT_CHK_EN	Conflict check enable. It is not implemented in uSDHC IP. 0 Conflict check disable 1 Conflict check enable
1 VSELECT	Voltage Selection Change the value of output signal VSELECT, to control the voltage on pads for external card. There must be a control circuit out of uSDHC to change the voltage on pads. 1 Change the voltage to low voltage range, around 1.8 V 0 Change the voltage to high voltage range, around 3.0 V
0 EXT_DMA_EN	External DMA Request Enable Enable the request to external DMA. When the internal DMA (either Simple DMA or Advanced DMA) is not in use, and this bit is set, uSDHC will send out DMA request when the internal buffer is ready. This bit is particularly useful when transferring data by ARM platform polling mode, and it is not allowed to send out the external DMA request. By default, this bit is set. 0 In any scenario, uSDHC does not send out external DMA request. 1 When internal DMA is not active, the external DMA request will be sent out.

10.3.8.29 MMC Boot Register (uSDHCx_MMC_BOOT)

This register contains the MMC Fast Boot control register.

Address: Base address + C4h offset



uSDHCx_MMC_BOOT field descriptions

Field	Description
31–16 BOOT_BLK_CNT[15:0]	The value defines the Stop At Block Gap value of automatic mode. When received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT) and AUTO_SABG_EN is 1, then Stop At Block Gap. Here, BLK_CNT is defined in the Block Atrributes Register, bit31 - 16 of 0x04.
15–9 Reserved	This read-only field is reserved and always has the value 0.
8 DISABLE_TIME_OUT	Disable Time Out **NOTE:** When this bit is set, there is no timeout check no matter whether BOOT_EN is set or not. 0 Enable time out 1 Disable time out
7 AUTO_SABG_EN	During boot, enable auto stop at block gap function. This function will be triggered, and host will stop at block gap when received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT).
6 BOOT_EN	Boot mode enable 0 Fast boot disable 1 Fast boot enable
5 BOOT_MODE	Boot mode select

Table continues on the next page...

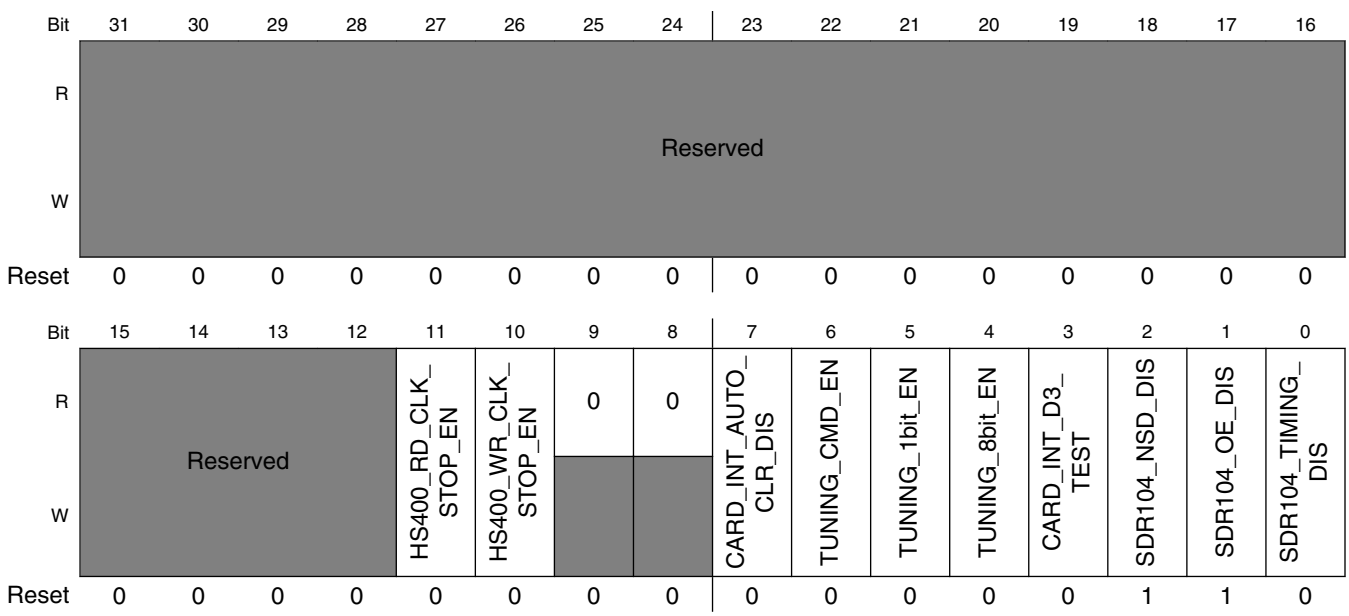
uSDHCx_MMC_BOOT field descriptions (continued)

Field	Description
	0 Normal boot 1 Alternative boot
4 BOOT_ACK	Boot ACK mode select 0 No ack 1 Ack
DTOCV_ ACK[3:0]	Boot ACK time out counter value. 0000 SDCLK x 2^13 0001 SDCLK x 2^14 0010 SDCLK x 2^15 0011 SDCLK x 2^16 0100 SDCLK x 2^17 0101 SDCLK x 2^18 0110 SDCLK x 2^19 0111 SDCLK x 2^20 1110 SDCLK x 2^27 1111 SDCLK x 2^28

10.3.8.30 Vendor Specific 2 Register (uSDHCx_VEND_SPEC2)

This register contains the vendor specific control 2 register.

Address: Base address + C8h offset



uSDHCx_VEND_SPEC2 field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11 HS400_RD_CLK_STOP_EN	HS400 Read Clock Stop Enable Only stop clock at read block gap.
10 HS400_WR_CLK_STOP_EN	HS400 Write Clock Stop Enable Only stop clock at write block gap.
9 Reserved	This read-only field is reserved and always has the value 0.
8 Reserved	This read-only field is reserved and always has the value 0.
7 CARD_INT_AUTO_CLR_DIS	Disable the feature to clear the Card interrupt status bit when Card Interrupt status enable bit is cleared. Only for debug. 0 Card interrupt status bit (CINT) can be cleared when Card Interrupt status enable bit is 0. 1 Card interrupt status bit (CINT) can only be cleared by writting a 1 to CINT bit.
6 TUNING_CMD_EN	Enable the auto tuning circuit to check the CMD line. 0 Auto tuning circuit does not check the CMD line. 1 Auto tuning circuit checks the CMD line.
5 TUNING_1bit_EN	Enable the auto tuning circuit to check the DATA0 only. It is used with the TUNING_8bit_EN together.
4 TUNING_8bit_EN	Enable the auto tuning circuit to check the DATA[7:0]. It is used with the TUNING_1bit_EN together. **NOTE:** The format of these two bits are [TUNNING_8bit_EN:TUNNING_1bit_EN]. 00 Tuning circuit only checks the DATA[3:0]. 01 Tuning circuit only checks the DATA0. 10 Tuning circuit checks the whole DATA[7:0]. 11 Invalid.
3 CARD_INT_D3_TEST	Card Interrupt Detection Test This bit only uses for debugging. 0 Check the card interrupt only when DATA3 is high. 1 Check the card interrupt by ignoring the status of DATA3.
2 SDR104_NSD_DIS	Interrupt window after abort command is sent. This bit only uses for debugging. 0 Enable the interrupt window 9 cycles later after the end of the I/O abort command (or CMD12) is sent. 1 Enable the interrupt window 5 cycles later after the end of the I/O abort command (or CMD12) is sent.
1 SDR104_OE_DIS	CMD_OE / DATA_OE logic generation test. This bit only uses for debugging. 0 Drive the CMD_OE / DATA_OE for one more clock cycle after the end bit. 1 Stop to drive the CMD_OE / DATA_OE at once after driving the end bit.

Table continues on the next page...

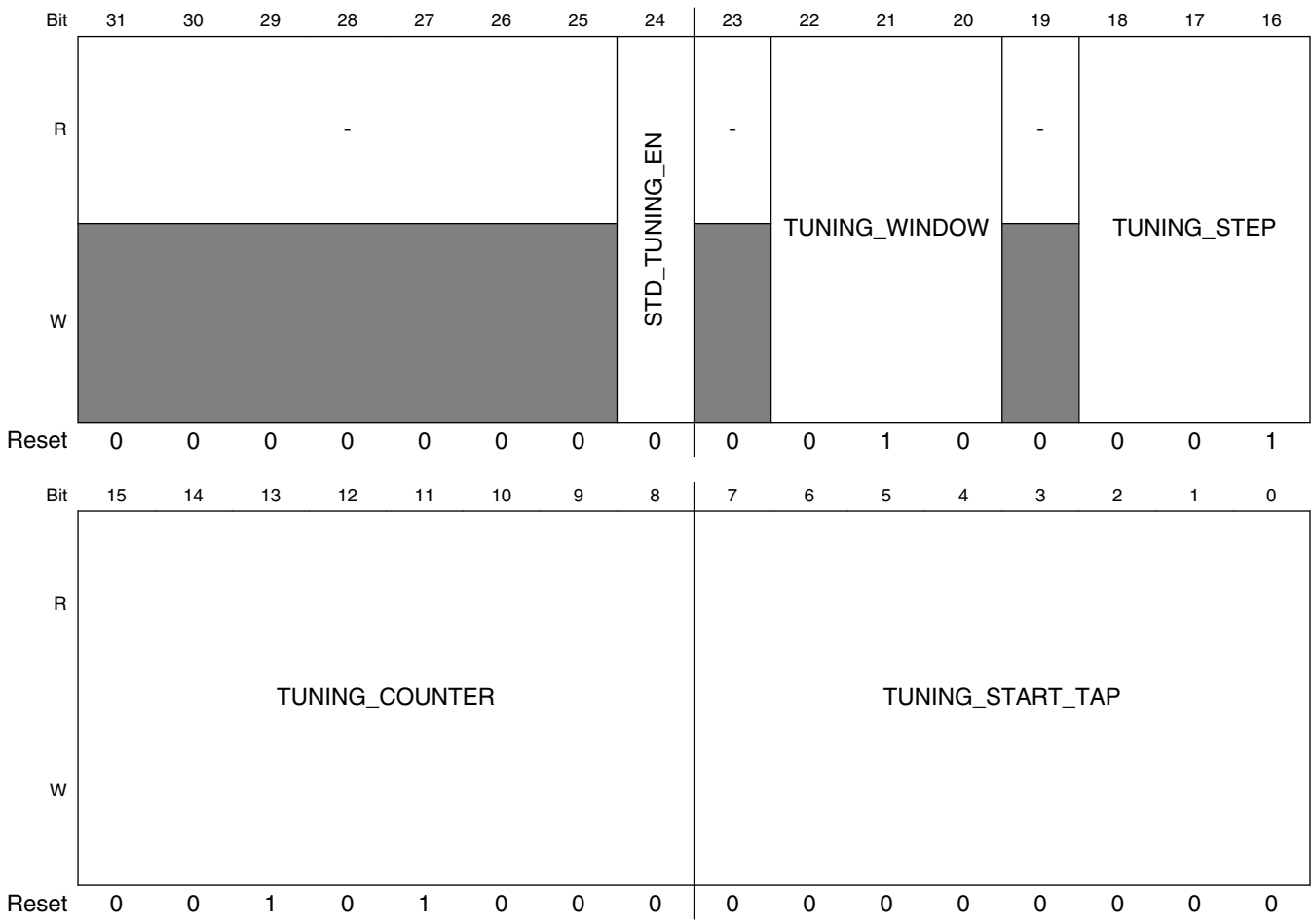
uSDHCx_VEND_SPEC2 field descriptions (continued)

Field	Description
0 SDR104_ TIMING_DIS	Timeout counter test. This bit only uses for debugging. 0 The timeout counter for Ncr changes to 80, Ncrc changes to 21. 1 The timeout counter for Ncr changes to 72, Ncrc changes to 15.

10.3.8.31 Tuning Control Register (uSDHCx_TUNING_CTRL)

The register contains configuration of tuning circuit.

Address: Base address + CCh offset



uSDHCx_TUNING_CTRL field descriptions

Field	Description
31–25 -	Reserved

Table continues on the next page...

uSDHCx_TUNING_CTRL field descriptions (continued)

Field	Description
24 STD_TUNING_EN	Standard tuning circuit and procedure enable: This bit is used to enable standard tuning circuit and procedure.
23 -	Reserved
22–20 TUNING_WINDOW	Select data window value for auto tuning
19 -	Reserved
18–16 TUNING_STEP	The increasing delay cell steps in tuning procedure.
15–8 TUNING_COUNTER	The MAX repeat CMD19 times in tuning procedure.
TUNING_START_TAP	The start dealy cell point when send first CMD19 in tuning procedure.

Chapter 11
Connectivity

11.1 Ethernet MAC (ENET)

11.1.1 Introduction

The MAC-NET core, in conjunction with a 10/100/1000-Mbit/s MAC, implements layer 3 network acceleration functions. These functions are designed to accelerate the processing of various common networking protocols, such as IP, TCP, UDP, and ICMP, providing wire speed services to client applications.

11.1.2 Overview

The core implements a triple-speed 10/100/1000-Mbit/s Ethernet MAC compliant with the IEEE802.3-2002 standard. The MAC layer provides compatibility with half- or full-duplex 10/100-Mbit/s and full-duplex gigabit Ethernet LANs.

The MAC operation is fully programmable and can be used in Network Interface Card (NIC), bridging, or switching applications. The core implements the remote network monitoring (RMON) counters according to IETF RFC 2819.

The core also implements a hardware acceleration block to optimize the performance of network controllers providing TCP/IP, UDP, and ICMP protocol services. The acceleration block performs critical functions in hardware, which are typically implemented with large software overhead.

The core implements programmable embedded FIFOs that can provide buffering on the receive path for lossless flow control.

Advanced power management features are available with magic packet detection and programmable power-down modes.

A unified DMA (uDMA), internal to the ENET module, optimizes data transfer between the ENET core and the SoC, and supports an enhanced buffer descriptor programming model to support IEEE 1588 functionality.

The programmable Ethernet MAC with IEEE 1588 integrates a standard IEEE 802.3 Ethernet MAC with a time-stamping module. The IEEE 1588 standard provides accurate clock synchronization for distributed control nodes for industrial automation applications.

11.1.2.1 Features

The MAC-NET core includes the following features.

11.1.2.1.1 Ethernet MAC features

- Implements the full 802.3 specification with preamble/SFD generation, frame padding generation, CRC generation and checking
- Supports zero-length preamble
- Dynamically configurable to support 10/100-Mbit/s and gigabit operation
- Supports 10/100 Mbit/s full-duplex and configurable half-duplex operation
- Supports gigabit full-duplex operation
- Compliant with the AMD magic packet detection with interrupt for node remote power management
- Seamless interface to commercial ethernet PHY devices via one of the following:
  - a 4-bit Media Independent Interface (MII) operating at 2.5/25 MHz.
  - a 4-bit non-standard MII-Lite (MII without the CRS and COL signals) operating at 2.5/25 MHz.
  - a 2-bit Reduced MII (RMII) operating at 50 MHz.
  - a (double data rate) 4-bit Reduced GMII (RGMII) operating at 125 MHz.
- Simple 64-Bit FIFO user-application interface
- CRC-32 checking at full speed with optional forwarding of the frame check sequence (FCS) field to the client
- CRC-32 generation and append on transmit or forwarding of user application provided FCS selectable on a per-frame basis
- In full-duplex mode:
  - Implements automated pause frame (802.3 x31A) generation and termination, providing flow control without user application intervention
  - Pause quanta used to form pause frames — dynamically programmable
  - Pause frame generation additionally controllable by user application offering flexible traffic flow control
  - Optional forwarding of received pause frames to the user application
  - Implements standard flow-control mechanism

- In half-duplex mode: provides full collision support, including jamming, backoff, and automatic retransmission
- Supports VLAN-tagged frames according to IEEE 802.1Q
- Programmable MAC address: Insertion on transmit; discards frames with mismatching destination address on receive (except broadcast and pause frames)
- Programmable promiscuous mode support to omit MAC destination address checking on receive
- Multicast and unicast address filtering on receive based on 64-entry hash table, reducing higher layer processing load
- Programmable frame maximum length providing support for any standard or proprietary frame length
- Statistics indicators for frame traffic and errors (alignment, CRC, length) and pause frames providing for IEEE 802.3 basic and mandatory management information database (MIB) package and remote network monitoring (RFC 2819)
- Simple handshake user application FIFO interface with fully programmable depth and threshold levels
- Provides separate status word for each received frame on the user interface providing information such as frame length, frame type, VLAN tag, and error information
- Multiple internal loopback options
- MDIO master interface for PHY device configuration and management supports two programmable MDIO base addresses, and standard (IEEE 802.3 Clause 22) and extended (Clause 45) MDIO frame formats
- Supports legacy FEC buffer descriptors
- Interrupt coalescing reduces the number of interrupts generated by the MAC, reducing CPU loading
- Traffic-shaping bandwidth distribution supports credit-based and round-robin-based policies. Either policy can be combined with time-based shaping.
- AVB (Audio Video Bridging, IEEE 802.1Qav) features:
  - Credit-based bandwidth distribution policy can be combined with time-based shaping
  - AVB endpoint talker and listener support
  - Support for arbitration between different priority traffic (for example, AVB class A, AVB class B, and non-AVB)

11.1.2.1.2 IP protocol performance optimization features

- Operates on TCP/IP and UDP/IP and ICMP/IP protocol data or IP header only
- Enables wire-speed processing
- Supports IPv4 and IPv6
- Transparent passing of frames of other types and protocols

- Supports VLAN tagged frames according to IEEE 802.1q with transparent forwarding of VLAN tag and control field
- Automatic IP-header and payload (protocol specific) checksum calculation and verification on receive
- Automatic IP-header and payload (protocol specific) checksum generation and automatic insertion on transmit configurable on a per-frame basis
- Supports IP and TCP, UDP, ICMP data for checksum generation and checking
- Supports full header options for IPv4 and TCP protocol headers
- Provides IPv6 support to datagrams with base header only — datagrams with extension headers are passed transparently unmodifed/unchecked
- Provides statistics information for received IP and protocol errors
- Configurable automatic discard of erroneous frames
- Configurable automatic host-to-network (RX) and network-to-host (TX) byte order conversion for IP and TCP/UDP/ICMP headers within the frame
- Configurable padding remove for short IP datagrams on receive
- Configurable Ethernet payload alignment to allow for 32-bit word-aligned header and payload processing
- Programmable store-and-forward operation with clock and rate decoupling FIFOs

11.1.2.1.3 IEEE 1588 features

- Supports all IEEE 1588 frames.
- Allows reference clock to be chosen independently of network speed.
- Software-programmable precise time-stamping of ingress and egress frames
- Timer monitoring capabilities for system calibration and timing accuracy management
- Precise time-stamping of external events with programmable interrupt generation
- Programmable event and interrupt generation for external system control
- Supports hardware- and software-controllable timer synchronization.
- Provides a 4-channel IEEE 1588 timer. Each channel supports input capture and output compare using the 1588 counter.

11.1.2.2 Block diagram

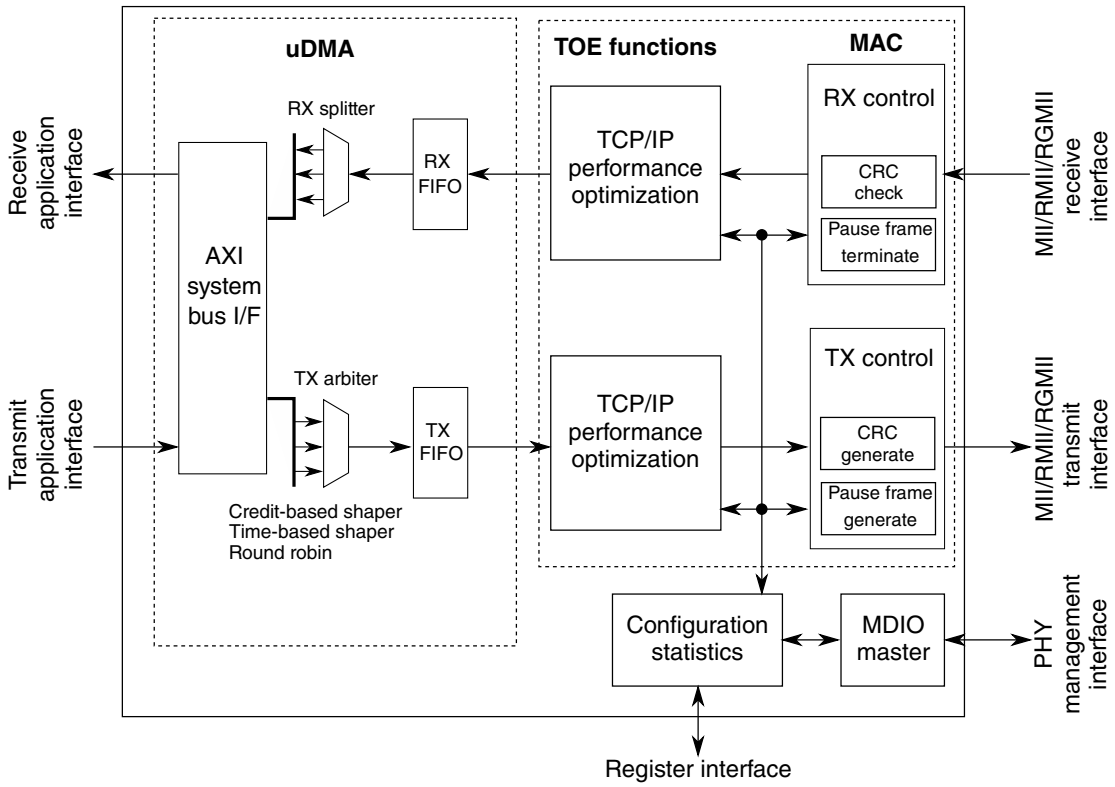


Figure 11-1. Ethernet MAC-NET core block diagram

11.1.3 External Signals

The table found here describes the external signals of ENET.

Table 11-1. ENET External Signals

Signal	Description	Mode	Pad	Alt Mode	Direction
ENET1_1588_EVENT0_IN	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII/RMII/RGMII	UART3_RXD	ALT4	I

Table continues on the next page...

Table 11-1. ENET External Signals (continued)

Signal	Description	Mode	Pad	Alt Mode	Direction
ENET1_1588_EVENT0_OUT	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set. **NOTE:** ENET_1588_EVENT0_OUT has a programmable output width, see IOMUXC_GPR0[CLK_STRETCH] delayed one clock cycle in relation to all other EVENTx_OUT signals	MII/RMII/RGMII	UART3_TXD	ALT4	O
ENET1_1588_EVENT1_IN	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII/RMII/RGMII	UART3_RTS	ALT4	I
ENET1_1588_EVENT1_OUT	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII/RMII/RGMII	UART3_CTS	ALT4	O
ENET1_1588_EVENT2_IN	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare,	MII/RMII/RGMII	LCD1_CLK	ALT2	I

Table continues on the next page...

Table 11-1. ENET External Signals (continued)

Signal	Description	Mode	Pad	Alt Mode	Direction
	the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.				
ENET1_1588_EVENT2_OUT	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII/RMII/ RGMII	LCD1_DATA20	ALT2	O
ENET1_1588_EVENT3_IN	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII/RMII/ RGMII	LCD1_ENABLE	ALT2	I
ENET1_1588_EVENT3_OUT	Capture/compare block input/output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII/RMII/ RGMII	LCD1_DATA21	ALT2	O
ENET1_COL	Asserted upon detection of a collision and remains asserted while the collision persists. This signal is not defined for full duplex mode.	MII	ENET1_COL	ALT0	I

Table continues on the next page...

Table 11-1. ENET External Signals (continued)

Signal	Description	Mode	Pad	Alt Mode	Direction
ENET1_CRS	Carrier sense. When asserted, indicates transmit or receive medium is not idle.	MII	ENET1_CRS	ALT0	I
ENET1_MDC	Output clock provides a timing reference to the PHY for data transfers on the MDIO signal.	MII/RMII/RGMII	GPIO1_IO11	ALT2	O
			SD2_WP	ALT1	
			UART1_TXD	ALT6	
ENET1_MDIO	Transfers control information between the external PHY and the mediaaccess controller. Data is synchronous to MDC. This signal is an input after reset.	MII/RMII/RGMII	GPIO1_IO10	ALT2	IO
			SD2_CD_B	ALT1	
			UART1_RXD	ALT6	
ENET1_RX_CLK	In MII mode, provides a timing reference for RX_EN, RX_DATA[3:0], and RX_ER.	MII	ENET1_RX_CLK	ALT0	I
ENET1_RX_ER	When asserted with RXDV, indicates the PHY detects an error in the current frame.	MII/RMII	ENET1_RXC	ALT1	I
ENET1_TX_CLK	Input clock, which provides a timing reference for TX_EN, TX_DATA[3:0], and TX_ER.	MII	ENET1_TX_CLK	ALT0	O
ENET1_TX_ER	When asserted for one or more clock cycles while TXEN is also asserted, PHY sends one or more illegal symbols.	MII	ENET1_TXC	ALT1	O
RGMII1_RD0	Contains the Ethernet input data transferred from the PHY to the media-access controller when RX_EN is asserted.	RGMII/RMII/MII	ENET1_RDATA0	ALT0	I
RGMII1_RD1	Contains the Ethernet input data transferred from the PHY to the media-access controller when RX_EN is asserted.	RGMII/RMII/MII	ENET1_RDATA1	ALT0	I
RGMII1_RD2	Contains the Ethernet input data transferred from the PHY to the media-access controller when RX_EN is asserted.	RGMII/MII	ENET1_RDATA2	ALT0	I
RGMII1_RD3	Contains the Ethernet input data transferred from the PHY to the media-access controller when RX_EN is asserted.	RGMII/MII	ENET1_RDATA3	ALT0	I
RGMII1_RX_CTL	In RGMII mode, contains RX_EN on the rising edge of RGMII_RXC, and a logical derivative of RX_EN and RX_ER (RX_EN XOR RX_ER) on the falling edge of RGMII_RXC.	RGMII	ENET1_RX_CTL	ALT0	I
RGMII1_RXC	In RGMII mode, provides a timing reference for RX_DATA[3:0] and RX_CTL.	RGMII/MII	ENET1_RXC	ALT0	I

Table continues on the next page...

Table 11-1. ENET External Signals (continued)

Signal	Description	Mode	Pad	Alt Mode	Direction
RGMII1_TD0	Serial output Ethernet data. Only valid during TX_EN assertion.	RGMII/ RMII/MII	ENET1_TD ATA0	ALT0	O
RGMII1_TD1	Serial output Ethernet data. Only valid during TX_EN assertion.	RGMII/ RMII/MII	ENET1_TD ATA1	ALT0	O
RGMII1_TD2	Serial output Ethernet data. Only valid during TX_EN assertion.	RGMII/MII	ENET1_TD ATA2	ALT0	O
RGMII1_TD3	Serial output Ethernet data. Only valid during TX_EN assertion.	RGMII/MII	ENET1_TD ATA3	ALT0	O
RGMII1_TX_CTL	In RGMII mode, contains TX_EN on the rising edge of RGMII_TXC, and a logical derivative of TX_EN and TX_ER (TX_EN XOR TX_ER) on the falling edge of RGMII_TXC.	RGMII	ENET1_TX_ CTL	ALT0	O
RGMII1_TXC	In RGMII mode, provides a timing reference for TX_DATA[3:0] and TX_CTL.	RGMII	ENET1_TX C	ALT0	O

11.1.4 Clocks

The table found here describes the clock sources for ENET. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 11-2. ENET Clocks

Clock name	Clock Root	Description
ipg_clk	ENET_AXI_CLK_ROO T	Module clock
ipg_clk_mac0	ENET_AXI_CLK_ROO T	MAC peripheral clock
ipg_clk_mac0_s	ENET_AXI_CLK_ROO T	MAC peripheral access clock
ipg_clk_s	ENET_AXI_CLK_ROO T	Peripheral access clock
ipg_clk_time	ENET1_TIME_CLK_R OOT	Peripheral clock
mac0_rxmem_clk	ENET_AXI_CLK_ROO T	MAC receive memory clock
mac0_txmem_clk	ENET_AXI_CLK_ROO T	MAC transmit memory clock

11.1.5 Memory map/register definition

ENET registers must be read or written with 32-bit accesses. Non-32 bit accesses will terminate with an error.

Reserved bits should be written with 0 and ignored on read. Unused registers read zero and a write has no effect.

This table shows Ethernet registers organization.

Table 11-3. Register map summary

Offset Address	Section	Description
0x0000 – 0x01FF	Configuration	Core control and status registers
0x0200 – 0x03FF	Statistics counters	MIB and Remote Network Monitoring (RFC 2819) registers
0x0400 – 0x0430	1588 control	1588 adjustable timer (TSM) and 1588 frame control
0x0600 – 0x07FC	Capture/Compare block	Registers for the Capture/Compare block

ENET memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BE_0004	Interrupt Event Register (ENET_EIR)	32	w1c	0000_0000h	11.1.5.1/ 2860
30BE_0008	Interrupt Mask Register (ENET_EIMR)	32	R/W	0000_0000h	11.1.5.2/ 2863
30BE_0010	Receive Descriptor Active Register - Ring 0 (ENET_RDAR)	32	R/W	0000_0000h	11.1.5.3/ 2867
30BE_0014	Transmit Descriptor Active Register - Ring 0 (ENET_TDAR)	32	R/W	0000_0000h	11.1.5.4/ 2868
30BE_0024	Ethernet Control Register (ENET_ECR)	32	R/W	See section	11.1.5.5/ 2869
30BE_0040	MII Management Frame Register (ENET_MMFR)	32	R/W	0000_0000h	11.1.5.6/ 2871
30BE_0044	MII Speed Control Register (ENET_MSCR)	32	R/W	0000_0000h	11.1.5.7/ 2872
30BE_0064	MIB Control Register (ENET_MIBC)	32	R/W	C000_0000h	11.1.5.8/ 2874
30BE_0084	Receive Control Register (ENET_RCR)	32	R/W	05EE_0001h	11.1.5.9/ 2875
30BE_00C4	Transmit Control Register (ENET_TCR)	32	R/W	0000_0000h	11.1.5.10/ 2878
30BE_00E4	Physical Address Lower Register (ENET_PALR)	32	R/W	0000_0000h	11.1.5.11/ 2880

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BE_00E8	Physical Address Upper Register (ENET_PAUR)	32	R/W	0000_8808h	11.1.5.12/ 2880
30BE_00EC	Opcode/Pause Duration Register (ENET_OPD)	32	R/W	0001_0000h	11.1.5.13/ 2881
30BE_00F0	Transmit Interrupt Coalescing Register (ENET_TXIC0)	32	R/W	0000_0000h	11.1.5.14/ 2881
30BE_00F4	Transmit Interrupt Coalescing Register (ENET_TXIC1)	32	R/W	0000_0000h	11.1.5.14/ 2881
30BE_00F8	Transmit Interrupt Coalescing Register (ENET_TXIC2)	32	R/W	0000_0000h	11.1.5.14/ 2881
30BE_0100	Receive Interrupt Coalescing Register (ENET_RXIC0)	32	R/W	0000_0000h	11.1.5.15/ 2882
30BE_0104	Receive Interrupt Coalescing Register (ENET_RXIC1)	32	R/W	0000_0000h	11.1.5.15/ 2882
30BE_0108	Receive Interrupt Coalescing Register (ENET_RXIC2)	32	R/W	0000_0000h	11.1.5.15/ 2882
30BE_0118	Descriptor Individual Upper Address Register (ENET_IAUR)	32	R/W	0000_0000h	11.1.5.16/ 2883
30BE_011C	Descriptor Individual Lower Address Register (ENET_IALR)	32	R/W	0000_0000h	11.1.5.17/ 2884
30BE_0120	Descriptor Group Upper Address Register (ENET_GAUR)	32	R/W	0000_0000h	11.1.5.18/ 2884
30BE_0124	Descriptor Group Lower Address Register (ENET_GALR)	32	R/W	0000_0000h	11.1.5.19/ 2885
30BE_0144	Transmit FIFO Watermark Register (ENET_TFWR)	32	R/W	0000_0000h	11.1.5.20/ 2885
30BE_0160	Receive Descriptor Ring 1 Start Register (ENET_RDSR1)	32	R/W	0000_0000h	11.1.5.21/ 2886
30BE_0164	Transmit Buffer Descriptor Ring 1 Start Register (ENET_TDSR1)	32	R/W	0000_0000h	11.1.5.22/ 2887
30BE_0168	Maximum Receive Buffer Size Register - Ring 1 (ENET_MRBR1)	32	R/W	0000_0000h	11.1.5.23/ 2888
30BE_016C	Receive Descriptor Ring 2 Start Register (ENET_RDSR2)	32	R/W	0000_0000h	11.1.5.24/ 2889
30BE_0170	Transmit Buffer Descriptor Ring 2 Start Register (ENET_TDSR2)	32	R/W	0000_0000h	11.1.5.25/ 2889
30BE_0174	Maximum Receive Buffer Size Register - Ring 2 (ENET_MRBR2)	32	R/W	0000_0000h	11.1.5.26/ 2890
30BE_0180	Receive Descriptor Ring 0 Start Register (ENET_RDSR)	32	R/W	0000_0000h	11.1.5.27/ 2891
30BE_0184	Transmit Buffer Descriptor Ring 0 Start Register (ENET_TDSR)	32	R/W	0000_0000h	11.1.5.28/ 2892
30BE_0188	Maximum Receive Buffer Size Register - Ring 0 (ENET_MRBR)	32	R/W	0000_0000h	11.1.5.29/ 2892

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BE_0190	Receive FIFO Section Full Threshold (ENET_RSFL)	32	R/W	0000_0000h	11.1.5.30/ 2893
30BE_0194	Receive FIFO Section Empty Threshold (ENET_RSEM)	32	R/W	0000_0000h	11.1.5.31/ 2894
30BE_0198	Receive FIFO Almost Empty Threshold (ENET_RAEM)	32	R/W	0000_0004h	11.1.5.32/ 2894
30BE_019C	Receive FIFO Almost Full Threshold (ENET_RAFL)	32	R/W	0000_0004h	11.1.5.33/ 2895
30BE_01A0	Transmit FIFO Section Empty Threshold (ENET_TSEM)	32	R/W	0000_0000h	11.1.5.34/ 2895
30BE_01A4	Transmit FIFO Almost Empty Threshold (ENET_TAEM)	32	R/W	0000_0004h	11.1.5.35/ 2896
30BE_01A8	Transmit FIFO Almost Full Threshold (ENET_TAFL)	32	R/W	0000_0008h	11.1.5.36/ 2896
30BE_01AC	Transmit Inter-Packet Gap (ENET_TIPG)	32	R/W	0000_000Ch	11.1.5.37/ 2897
30BE_01B0	Frame Truncation Length (ENET_FTRL)	32	R/W	0000_07FFh	11.1.5.38/ 2897
30BE_01C0	Transmit Accelerator Function Configuration (ENET_TACC)	32	R/W	0000_0000h	11.1.5.39/ 2898
30BE_01C4	Receive Accelerator Function Configuration (ENET_RACC)	32	R/W	0000_0000h	11.1.5.40/ 2899
30BE_01C8	Receive Classification Match Register for Class n (ENET_RCMR1)	32	R/W	0000_0000h	11.1.5.41/ 2901
30BE_01CC	Receive Classification Match Register for Class n (ENET_RCMR2)	32	R/W	0000_0000h	11.1.5.41/ 2901
30BE_01D8	DMA Class Based Configuration (ENET_DMA1CFG)	32	R/W	0000_0000h	11.1.5.42/ 2902
30BE_01DC	DMA Class Based Configuration (ENET_DMA2CFG)	32	R/W	0000_0000h	11.1.5.42/ 2902
30BE_01E0	Receive Descriptor Active Register - Ring 1 (ENET_RDAR1)	32	R/W	0000_0000h	11.1.5.43/ 2904
30BE_01E4	Transmit Descriptor Active Register - Ring 1 (ENET_TDAR1)	32	R/W	0000_0000h	11.1.5.44/ 2904
30BE_01E8	Receive Descriptor Active Register - Ring 2 (ENET_RDAR2)	32	R/W	0000_0000h	11.1.5.45/ 2906
30BE_01EC	Transmit Descriptor Active Register - Ring 2 (ENET_TDAR2)	32	R/W	0000_0000h	11.1.5.46/ 2907
30BE_01F0	QOS Scheme (ENET_QOS)	32	R/W	0000_0000h	11.1.5.47/ 2907
30BE_0200	Reserved Statistic Register (ENET_RMON_T_DROP)	32	R	0000_0000h	11.1.5.48/ 2909
30BE_0204	Tx Packet Count Statistic Register (ENET_RMON_T_PACKETS)	32	R	0000_0000h	11.1.5.49/ 2909

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BE_0208	Tx Broadcast Packets Statistic Register (ENET_RMON_T_BC_PKT)	32	R	0000_0000h	11.1.5.50/ 2910
30BE_020C	Tx Multicast Packets Statistic Register (ENET_RMON_T_MC_PKT)	32	R	0000_0000h	11.1.5.51/ 2910
30BE_0210	Tx Packets with CRC/Align Error Statistic Register (ENET_RMON_T_CRC_ALIGN)	32	R	0000_0000h	11.1.5.52/ 2911
30BE_0214	Tx Packets Less Than Bytes and Good CRC Statistic Register (ENET_RMON_T_UNDERSIZE)	32	R	0000_0000h	11.1.5.53/ 2911
30BE_0218	Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (ENET_RMON_T_OVERSIZE)	32	R	0000_0000h	11.1.5.54/ 2912
30BE_021C	Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_T_FRAG)	32	R	0000_0000h	11.1.5.55/ 2912
30BE_0220	Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (ENET_RMON_T_JAB)	32	R	0000_0000h	11.1.5.56/ 2913
30BE_0224	Tx Collision Count Statistic Register (ENET_RMON_T_COL)	32	R	0000_0000h	11.1.5.57/ 2913
30BE_0228	Tx 64-Byte Packets Statistic Register (ENET_RMON_T_P64)	32	R	0000_0000h	11.1.5.58/ 2914
30BE_022C	Tx 65- to 127-byte Packets Statistic Register (ENET_RMON_T_P65TO127)	32	R	0000_0000h	11.1.5.59/ 2914
30BE_0230	Tx 128- to 255-byte Packets Statistic Register (ENET_RMON_T_P128TO255)	32	R	0000_0000h	11.1.5.60/ 2915
30BE_0234	Tx 256- to 511-byte Packets Statistic Register (ENET_RMON_T_P256TO511)	32	R	0000_0000h	11.1.5.61/ 2915
30BE_0238	Tx 512- to 1023-byte Packets Statistic Register (ENET_RMON_T_P512TO1023)	32	R	0000_0000h	11.1.5.62/ 2916
30BE_023C	Tx 1024- to 2047-byte Packets Statistic Register (ENET_RMON_T_P1024TO2047)	32	R	0000_0000h	11.1.5.63/ 2916
30BE_0240	Tx Packets Greater Than 2048 Bytes Statistic Register (ENET_RMON_T_P_GTE2048)	32	R	0000_0000h	11.1.5.64/ 2917
30BE_0244	Tx Octets Statistic Register (ENET_RMON_T_OCTETS)	32	R	0000_0000h	11.1.5.65/ 2917
30BE_0248	Reserved Statistic Register (ENET_IEEE_T_DROP)	32	R	0000_0000h	11.1.5.66/ 2917
30BE_024C	Frames Transmitted OK Statistic Register (ENET_IEEE_T_FRAME_OK)	32	R	0000_0000h	11.1.5.67/ 2918
30BE_0250	Frames Transmitted with Single Collision Statistic Register (ENET_IEEE_T_1COL)	32	R	0000_0000h	11.1.5.68/ 2918
30BE_0254	Frames Transmitted with Multiple Collisions Statistic Register (ENET_IEEE_T_MCOL)	32	R	0000_0000h	11.1.5.69/ 2919
30BE_0258	Frames Transmitted after Deferral Delay Statistic Register (ENET_IEEE_T_DEF)	32	R	0000_0000h	11.1.5.70/ 2919
30BE_025C	Frames Transmitted with Late Collision Statistic Register (ENET_IEEE_T_LCOL)	32	R	0000_0000h	11.1.5.71/ 2920

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BE_0260	Frames Transmitted with Excessive Collisions Statistic Register (ENET_IEEE_T_EXCOL)	32	R	0000_0000h	11.1.5.72/ 2920
30BE_0264	Frames Transmitted with Tx FIFO Underrun Statistic Register (ENET_IEEE_T_MACERR)	32	R	0000_0000h	11.1.5.73/ 2921
30BE_0268	Frames Transmitted with Carrier Sense Error Statistic Register (ENET_IEEE_T_CSERR)	32	R	0000_0000h	11.1.5.74/ 2921
30BE_026C	Reserved Statistic Register (ENET_IEEE_T_SQE)	32	R (reads 0)	0000_0000h	11.1.5.75/ 2922
30BE_0270	Flow Control Pause Frames Transmitted Statistic Register (ENET_IEEE_T_FDXFC)	32	R	0000_0000h	11.1.5.76/ 2922
30BE_0274	Octet Count for Frames Transmitted w/o Error Statistic Register (ENET_IEEE_T_OCTETS_OK)	32	R	0000_0000h	11.1.5.77/ 2923
30BE_0284	Rx Packet Count Statistic Register (ENET_RMON_R_PACKETS)	32	R	0000_0000h	11.1.5.78/ 2923
30BE_0288	Rx Broadcast Packets Statistic Register (ENET_RMON_R_BC_PKT)	32	R	0000_0000h	11.1.5.79/ 2924
30BE_028C	Rx Multicast Packets Statistic Register (ENET_RMON_R_MC_PKT)	32	R	0000_0000h	11.1.5.80/ 2924
30BE_0290	Rx Packets with CRC/Align Error Statistic Register (ENET_RMON_R_CRC_ALIGN)	32	R	0000_0000h	11.1.5.81/ 2925
30BE_0294	Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (ENET_RMON_R_UNDERSIZE)	32	R	0000_0000h	11.1.5.82/ 2925
30BE_0298	Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (ENET_RMON_R_OVERSIZE)	32	R	0000_0000h	11.1.5.83/ 2926
30BE_029C	Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_R_FRAG)	32	R	0000_0000h	11.1.5.84/ 2926
30BE_02A0	Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (ENET_RMON_R_JAB)	32	R	0000_0000h	11.1.5.85/ 2927
30BE_02A4	Reserved Statistic Register (ENET_RMON_R_RESVD_0)	32	R (reads 0)	0000_0000h	11.1.5.86/ 2927
30BE_02A8	Rx 64-Byte Packets Statistic Register (ENET_RMON_R_P64)	32	R	0000_0000h	11.1.5.87/ 2927
30BE_02AC	Rx 65- to 127-Byte Packets Statistic Register (ENET_RMON_R_P65TO127)	32	R	0000_0000h	11.1.5.88/ 2928
30BE_02B0	Rx 128- to 255-Byte Packets Statistic Register (ENET_RMON_R_P128TO255)	32	R	0000_0000h	11.1.5.89/ 2928
30BE_02B4	Rx 256- to 511-Byte Packets Statistic Register (ENET_RMON_R_P256TO511)	32	R	0000_0000h	11.1.5.90/ 2929
30BE_02B8	Rx 512- to 1023-Byte Packets Statistic Register (ENET_RMON_R_P512TO1023)	32	R	0000_0000h	11.1.5.91/ 2929
30BE_02BC	Rx 1024- to 2047-Byte Packets Statistic Register (ENET_RMON_R_P1024TO2047)	32	R	0000_0000h	11.1.5.92/ 2930
30BE_02C0	Rx Packets Greater than 2048 Bytes Statistic Register (ENET_RMON_R_P_GTE2048)	32	R	0000_0000h	11.1.5.93/ 2930

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BE_02C4	Rx Octets Statistic Register (ENET_RMON_R_OCTETS)	32	R	0000_0000h	11.1.5.94/ 2931
30BE_02C8	Frames not Counted Correctly Statistic Register (ENET_IEEE_R_DROP)	32	R	0000_0000h	11.1.5.95/ 2931
30BE_02CC	Frames Received OK Statistic Register (ENET_IEEE_R_FRAME_OK)	32	R	0000_0000h	11.1.5.96/ 2932
30BE_02D0	Frames Received with CRC Error Statistic Register (ENET_IEEE_R_CRC)	32	R	0000_0000h	11.1.5.97/ 2932
30BE_02D4	Frames Received with Alignment Error Statistic Register (ENET_IEEE_R_ALIGN)	32	R	0000_0000h	11.1.5.98/ 2933
30BE_02D8	Receive FIFO Overflow Count Statistic Register (ENET_IEEE_R_MACERR)	32	R	0000_0000h	11.1.5.99/ 2933
30BE_02DC	Flow Control Pause Frames Received Statistic Register (ENET_IEEE_R_FDXFC)	32	R	0000_0000h	11.1.5.100/ 2934
30BE_02E0	Octet Count for Frames Received without Error Statistic Register (ENET_IEEE_R_OCTETS_OK)	32	R	0000_0000h	11.1.5.101/ 2934
30BE_0400	Adjustable Timer Control Register (ENET_ATCR)	32	R/W	0000_0000h	11.1.5.102/ 2935
30BE_0404	Timer Value Register (ENET_ATVR)	32	R/W	0000_0000h	11.1.5.103/ 2937
30BE_0408	Timer Offset Register (ENET_ATOFF)	32	R/W	0000_0000h	11.1.5.104/ 2937
30BE_040C	Timer Period Register (ENET_ATPER)	32	R/W	3B9A_CA00h	11.1.5.105/ 2937
30BE_0410	Timer Correction Register (ENET_ATCOR)	32	R/W	0000_0000h	11.1.5.106/ 2938
30BE_0414	Time-Stamping Clock Period Register (ENET_ATINC)	32	R/W	0000_0000h	11.1.5.107/ 2938
30BE_0418	Timestamp of Last Transmitted Frame (ENET_ATSTMP)	32	R	0000_0000h	11.1.5.108/ 2939
30BE_0604	Timer Global Status Register (ENET_TGSR)	32	R/W	0000_0000h	11.1.5.109/ 2939
30BE_0608	Timer Control Status Register (ENET_TCSR0)	32	R/W	0000_0000h	11.1.5.110/ 2940
30BE_060C	Timer Compare Capture Register (ENET_TCCR0)	32	R/W	0000_0000h	11.1.5.111/ 2941
30BE_0610	Timer Control Status Register (ENET_TCSR1)	32	R/W	0000_0000h	11.1.5.110/ 2940
30BE_0614	Timer Compare Capture Register (ENET_TCCR1)	32	R/W	0000_0000h	11.1.5.111/ 2941
30BE_0618	Timer Control Status Register (ENET_TCSR2)	32	R/W	0000_0000h	11.1.5.110/ 2940
30BE_061C	Timer Compare Capture Register (ENET_TCCR2)	32	R/W	0000_0000h	11.1.5.111/ 2941

Table continues on the next page...

ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BE_0620	Timer Control Status Register (ENET_TCSR3)	32	R/W	0000_0000h	11.1.5.110/ 2940
30BE_0624	Timer Compare Capture Register (ENET_TCCR3)	32	R/W	0000_0000h	11.1.5.111/ 2941

11.1.5.1 Interrupt Event Register (ENET_EIR)

When an event occurs that sets a bit in EIR, an interrupt occurs if the corresponding bit in the interrupt mask register (EIMR) is also set. Writing a 1 to an EIR bit clears it; writing 0 has no effect. This register is cleared upon hardware reset.

NOTE

TxBD[INT] and RxBD[INT] must be set to 1 to allow setting the corresponding EIR register flags in enhanced mode, ENET_ECR[EN1588] = 1. Legacy mode does not require these flags to be enabled.

Address: 30BE_0000h base + 4h offset = 30BE_0004h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	PLR	WAKEUP	TS_AVAIL
W		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS_TIMER	RXFLUSH_2	RXFLUSH_1	RXFLUSH_0					TXF2	TXB2	RXF2	RXB2	TXF1	TXB1	RXF1	RXB1
W	w1c	w1c	w1c	w1c	0			0	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_EIR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 BABR	Babbling Receive Error Indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
2 BABT	Babbling Transmit Error Indicates the transmitted frame length exceeds RCR[MAX_FL] bytes. Usually this condition is caused when a frame that is too long is placed into the transmit data buffer(s). Truncation does not occur.
3 GRA	Graceful Stop Complete This interrupt is asserted after the transmitter is put into a pause state after completion of the frame currently being transmitted. See Graceful Transmit Stop (GTS) for conditions that lead to graceful stop. **NOTE:** The GRA interrupt is asserted only when the TX transitions into the stopped state. If this bit is cleared by writing 1 and the TX is still stopped, the bit is not set again.
4 TXF	Transmit Frame Interrupt Indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated.
5 TXB	Transmit Buffer Interrupt Indicates a transmit buffer descriptor has been updated.
6 RXF	Receive Frame Interrupt Indicates a frame has been received and the last corresponding buffer descriptor has been updated.
7 RXB	Receive Buffer Interrupt Indicates a receive buffer descriptor is not the last in the frame has been updated.
8 MII	MII Interrupt. Indicates that the MII has completed the data transfer requested.
9 EBERR	Ethernet Bus Error Indicates a system bus error occurred when a uDMA transaction is underway. When this bit is set, ECR[ETHEREN] is cleared, halting frame processing by the MAC. When this occurs, software must ensure proper actions, possibly resetting the system, to resume normal operation.
10 LC	Late Collision Indicates a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame truncates with a bad CRC and the remainder of the frame is discarded.
11 RL	Collision Retry Limit Indicates a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This error can only occur in half-duplex mode.
12 UN	Transmit FIFO Underrun Indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
13 PLR	Payload Receive Error

Table continues on the next page...

ENET_EIR field descriptions (continued)

Field	Description
	Indicates a frame was received with a payload length error. See Frame Length/Type Verification: Payload Length Check for more information.
14 WAKEUP	Node Wakeup Request Indication Read-only status bit to indicate that a magic packet has been detected. Will act only if ECR[MAGICEN] is set.
15 TS_AVAIL	Transmit Timestamp Available Indicates that the timestamp of the last transmitted timing frame is available in the ATSTMP register.
16 TS_TIMER	Timestamp Timer The adjustable timer reached the period event. A period event interrupt can be generated if ATCR[PEREN] is set and the timer wraps according to the periodic setting in the ATPER register. Set the timer period value before setting ATCR[PEREN].
17 RXFLUSH_2	RX DMA Ring 2 flush indication. This ring's RX frame has been flushed due to either RDAR2[RDAR] or RxBD[E] being clear and only if QOS[RXFLUSH_2] is enabled.
18 RXFLUSH_1	RX DMA Ring 1 flush indication. This ring's RX frame has been flushed due to either RDAR1[RDAR] or RxBD[E] being clear and only if QOS[RXFLUSH_1] is enabled.
19 RXFLUSH_0	RX DMA Ring 0 flush indication. . This ring's RX frame has been flushed due to either RDAR[RDAR] or RxBD[E] being clear and only if QOS[RXFLUSH_0] is enabled.
20–22 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
23 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
24 TXF2	Transmit frame interrupt, class 2 This bit indicates that a frame has been transmitted and the last corresponding buffer descriptor has been updated (ring/class 2).
25 TXB2	Transmit buffer interrupt, class 2 This field indicates that a transmit buffer descriptor has been updated (ring/class 2).
26 RXF2	Receive frame interrupt, class 2 This field indicates that a frame has been received and the last corresponding buffer descriptor has been updated (ring/class 2).
27 RXB2	Receive buffer interrupt, class 2 This field indicates that a receive buffer descriptor, that not the last in the frame, has been updated (ring/class 2).
28 TXF1	Transmit frame interrupt, class 1 This bit indicates that a frame has been transmitted and the last corresponding buffer descriptor has been updated (ring/class 1).
29 TXB1	Transmit buffer interrupt, class 1 This field indicates that a transmit buffer descriptor has been updated (ring/class 1).
30 RXF1	Receive frame interrupt, class 1 This field indicates that a frame has been received and the last corresponding buffer descriptor has been updated (ring/class 1).

Table continues on the next page...

ENET_EIR field descriptions (continued)

Field	Description
31 RXB1	Receive buffer interrupt, class 1 This field indicates that a receive buffer descriptor, that not the last in the frame, has been updated (ring/ class 1).

11.1.5.2 Interrupt Mask Register (ENET_EIMR)

EIMR controls which interrupt events are allowed to generate actual interrupts. A hardware reset clears this register. If the corresponding bits in the EIR and EIMR registers are set, an interrupt is generated. The interrupt signal remains asserted until a 1 is written to the EIR field (write 1 to clear) or a 0 is written to the EIMR field.

Address: 30BE_0000h base + 8h offset = 30BE_0008h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	PLR	WAKEUP	TS_AVAIL
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS_TIMER	RXFLUSH_2	RXFLUSH_1	RXFLUSH_0					TXF2	TXB2	RXF2	RXB2	TXF1	TXB1	RXF1	RXB1
W					0			0								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_EIMR field descriptions

Field	Description
0 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
1 BABR	BABR Interrupt Mask Corresponds to interrupt source EIR[BABR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR BABR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
2 BABT	BABT Interrupt Mask

Table continues on the next page...

ENET_EIMR field descriptions (continued)

Field	Description
	Corresponds to interrupt source EIR[BABT] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR BABT field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
3 GRA	GRA Interrupt Mask Corresponds to interrupt source EIR[GRA] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR GRA field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
4 TXF	TXF Interrupt Mask Corresponds to interrupt source EIR[TXF] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
5 TXB	TXB Interrupt Mask Corresponds to interrupt source EIR[TXB] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
6 RXF	RXF Interrupt Mask Corresponds to interrupt source EIR[RXF] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
7 RXB	RXB Interrupt Mask Corresponds to interrupt source EIR[RXB] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RXB field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
8 MII	MII Interrupt Mask Corresponds to interrupt source EIR[MII] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR MII field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.

Table continues on the next page...

ENET_EIMR field descriptions (continued)

Field	Description
9 EBERR	EBERR Interrupt Mask Corresponds to interrupt source EIR[EBERR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR EBERR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
10 LC	LC Interrupt Mask Corresponds to interrupt source EIR[LC] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR LC field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
11 RL	RL Interrupt Mask Corresponds to interrupt source EIR[RL] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RL field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
12 UN	UN Interrupt Mask Corresponds to interrupt source EIR[UN] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR UN field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
13 PLR	PLR Interrupt Mask Corresponds to interrupt source EIR[PLR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR PLR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
14 WAKEUP	WAKEUP Interrupt Mask Corresponds to interrupt source EIR[WAKEUP] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR WAKEUP field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
15 TS_AVAIL	TS_AVAIL Interrupt Mask Corresponds to interrupt source EIR[TS_AVAIL] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TS_AVAIL field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
16 TS_TIMER	TS_TIMER Interrupt Mask Corresponds to interrupt source EIR[TS_TIMER] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TS_TIMER field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
17 RXFLUSH_2	Corresponds to interrupt source EIR[RXFLUSH_2] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[RXFLUSH_2] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.

Table continues on the next page...

ENET_EIMR field descriptions (continued)

Field	Description
18 RXFLUSH_1	Corresponds to interrupt source EIR[RXFLUSH_1] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[RXFLUSH_1] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
19 RXFLUSH_0	Corresponds to interrupt source EIR[RXFLUSH_0] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[RXFLUSH_0] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
20–22 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
23 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
24 TXF2	Transmit frame interrupt, class 2 Corresponds to interrupt source EIR[TXF2] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[TXF2] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
25 TXB2	Transmit buffer interrupt, class 2 Corresponds to interrupt source EIR[TXB2] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[TXB2] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
26 RXF2	Receive frame interrupt, class 2 Corresponds to interrupt source EIR[RXF2] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[RXF2] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
27 RXB2	Receive buffer interrupt, class 2 Corresponds to interrupt source EIR[RXB2] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[RXB2] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
28 TXF1	Transmit frame interrupt, class 1 Corresponds to interrupt source EIR[TXF1] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[TXF1] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
29 TXB1	Transmit buffer interrupt, class 1 Corresponds to interrupt source EIR[TXB1] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[TXB1] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
30 RXF1	Receive frame interrupt, class 1 Corresponds to interrupt source EIR[RXF1] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[RXF1] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
31 RXB1	Receive buffer interrupt, class 1

Table continues on the next page...

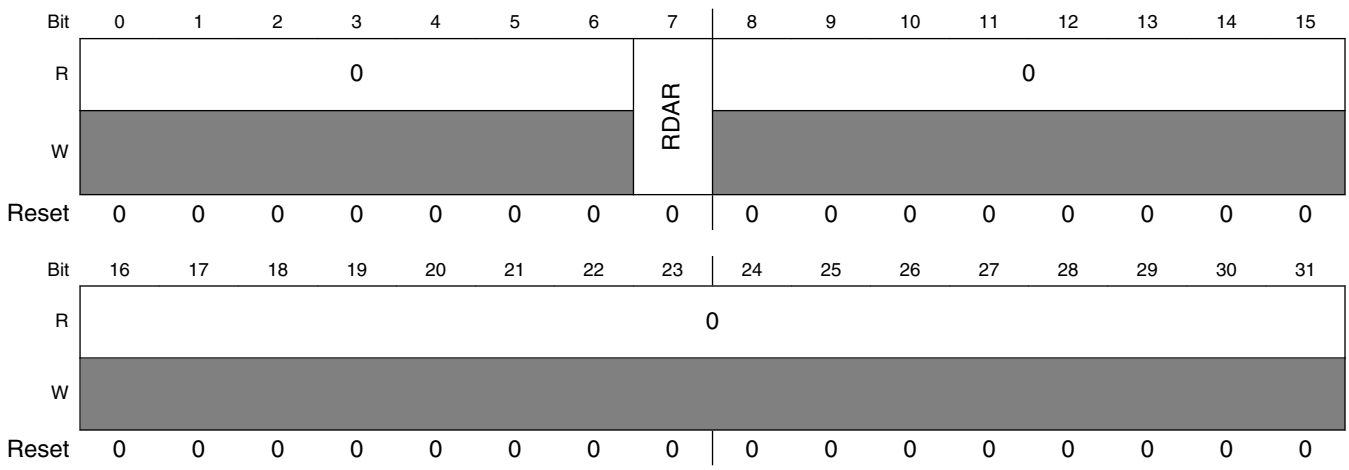
ENET_EIMR field descriptions (continued)

Field	Description
	Corresponds to interrupt source EIR[RXB1] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[RXB1] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.

11.1.5.3 Receive Descriptor Active Register - Ring 0 (ENET_RDAR)

RDAR is a command register, written by the user, to indicate that the receive descriptor ring 0 has been updated, that is, that the driver produced empty receive buffers with the empty bit set.

Address: 30BE_0000h base + 10h offset = 30BE_0010h



ENET_RDAR field descriptions

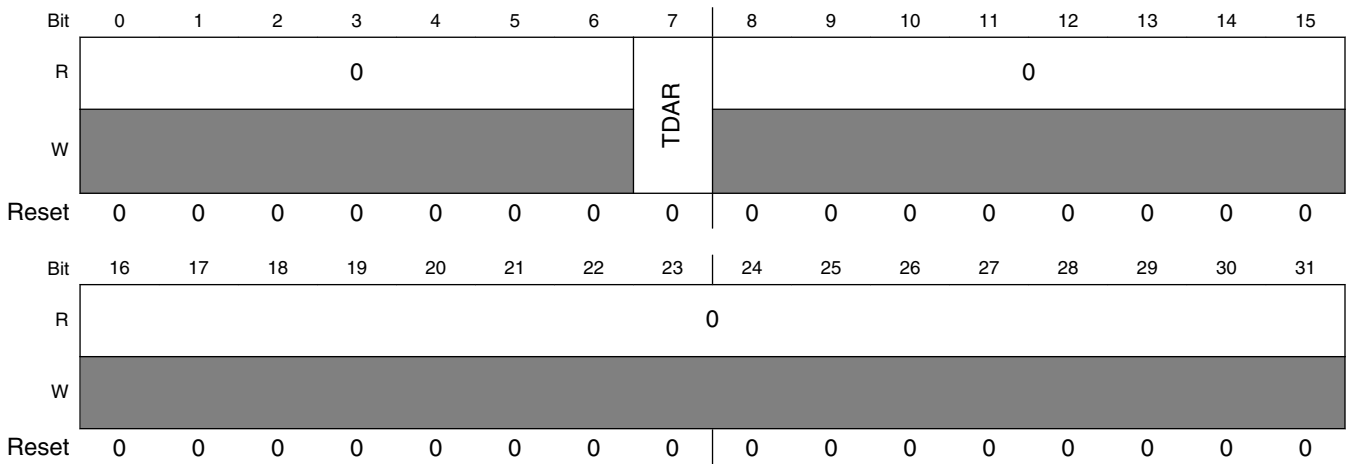
Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 RDAR	Receive Descriptor Active Always set to 1 when this register is written, regardless of the value written. This field is cleared by the MAC device when no additional empty descriptors remain in the receive ring. It is also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.4 Transmit Descriptor Active Register - Ring 0 (ENET_TDAR)

The TDAR is a command register that the user writes to indicate that the transmit descriptor ring 0 has been updated, that is, that transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor.

The TDAR register is cleared at reset, when ECR[ETHEREN] transitions from set to cleared, or when ECR[RESET] is set.

Address: 30BE_0000h base + 14h offset = 30BE_0014h



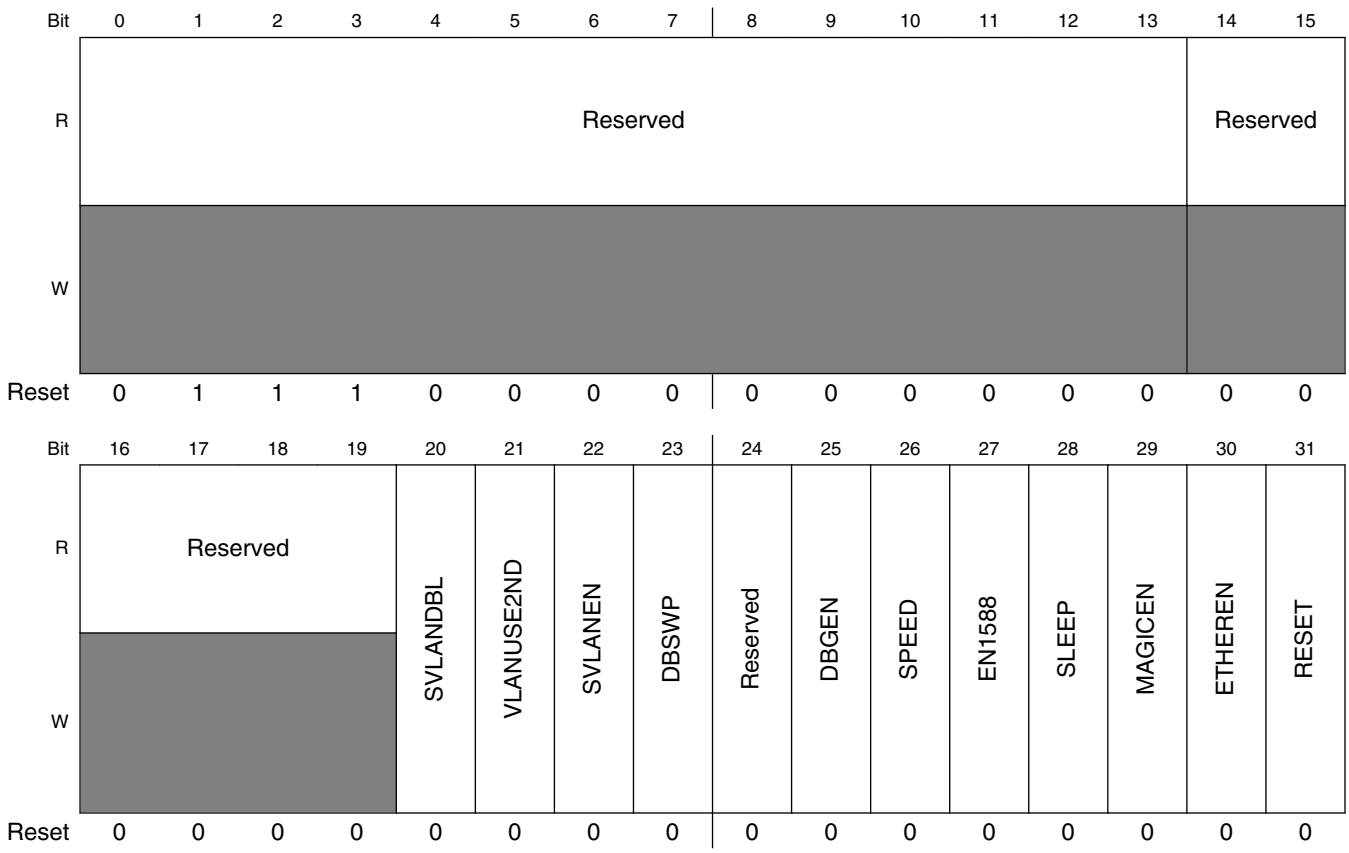
ENET_TDAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TDAR	Transmit Descriptor Active Always set to 1 when this register is written, regardless of the value written. This bit is cleared by the MAC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.5 Ethernet Control Register (ENET_ECR)

ECR is a read/write user register, though hardware may also alter fields in this register. It controls many of the high level features of the Ethernet MAC, including legacy FEC support through the EN1588 field.

Address: 30BE_0000h base + 24h offset = 30BE_0024h



ENET_ECR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This field must be set to 01110000000000b.
14–19 Reserved	This field is reserved. This field must be set to 0.
20 SVLANDBL	S-VLAN double tag If enabled, S-VLAN detection requires a double-tagged frame to define a frame as being a VLAN frame. The following rules apply: • If the first tag is the S-VLAN type, it must be followed by a second tag with the C-VLAN type to declare the frame as VLAN frame. • If the first tag is the S-VLAN type but no 2nd tag follows, it is not considered a VLAN frame but instead treated as an untagged frame. • If the first tag is the C-VLAN type, it is considered a VLAN frame as normal.

Table continues on the next page...

ENET_ECR field descriptions (continued)

Field	Description
	NOTE: VLANUSE2ND can be used to determine from which tag the data should be extracted. This applies only if SVLAN_EN = 1, ignored otherwise.
21 VLANUSE2ND	VLAN use second tag 0 Always extract data from the first VLAN tag if it exists. 1 When a double-tagged frame is detected, the data of the second tag is extracted for further processing. A double-tagged frame is defined as: • The first tag can be a C-VLAN or a S-VLAN (if SVLAN_ENA = 1) • The second tag must be a C-VLAN
22 SVLANEN	S-VLAN enable Enable additional detection of S-VLAN tag according to IEEE802.1Q. 0 Only the EtherType 0x8100 will be considered for VLAN detection. 1 The EtherType 0x88a8 will be considered in addition to 0x8100 (C-VLAN) to identify a VLAN frame in receive. When a VLAN frame is identified, the two bytes following the VLAN type are extracted and used by the classification match comparators, RCMRn.
23 DBSWP	Descriptor Byte Swapping Enable Swaps the byte locations of the buffer descriptors. **NOTE:** This field must be written to 1 after reset. 0 The buffer descriptor bytes are not swapped to support big-endian devices. 1 The buffer descriptor bytes are swapped to support little-endian devices.
24 Reserved	This field is reserved. This field must be set to zero.
25 DBGEN	Debug Enable Enables the MAC to enter hardware freeze mode when the device enters debug mode. 0 MAC continues operation in debug mode. 1 MAC enters hardware freeze mode when the processor is in debug mode.
26 SPEED	Selects between 10/100-Mbit/s and 1000-Mbit/s modes of operation. 0 10/100-Mbit/s mode 1 1000-Mbit/s mode
27 EN1588	EN1588 Enable Enables enhanced functionality of the MAC. 0 Legacy FEC buffer descriptors and functions enabled. 1 Enhanced frame time-stamping functions enabled.
28 SLEEP	Sleep Mode Enable 0 Normal operating mode. 1 Sleep mode.
29 MAGICEN	Magic Packet Detection Enable Enables/disables magic packet detection.

Table continues on the next page...

ENET_ECR field descriptions (continued)

Field	Description
	NOTE: MAGICEN is relevant only if the SLEEP field is set. If MAGICEN is set, changing the SLEEP field enables/disables sleep mode and magic packet detection. 0 Magic detection logic disabled. 1 The MAC core detects magic packets and asserts EIR[WAKEUP] when a frame is detected.
30 ETHEREN	Ethernet Enable Enables/disables the Ethernet MAC. When the MAC is disabled, the buffer descriptors for an aborted transmit frame are not updated. The uDMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. Hardware clears this field under the following conditions: • RESET is set by software • An error condition causes the EBERR field to set. **NOTE:** • ETHEREN must be set at the very last step during ENET configuration/setup/initialization, only *after* all other ENET-related registers have been configured. • If ETHEREN is cleared to 0 by software then next time ETHEREN is set, the EIR interrupts must cleared to 0 due to previous pending interrupts. 0 Reception immediately stops and transmission stops after a bad CRC is appended to any currently transmitted frame. 1 MAC is enabled, and reception and transmission are possible.
31 RESET	Ethernet MAC Reset When this field is set, it clears the ETHEREN field.

11.1.5.6 MII Management Frame Register (ENET_MMFR)

Writing to MMFR triggers a management frame transaction to the PHY device unless MSCR is programmed to zero.

If MSCR is changed from zero to non-zero during a write to MMFR, an MII frame is generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero.

If the MMFR register is written while frame generation is in progress, the frame contents are altered. Software must use the EIR[MII] interrupt indication to avoid writing to the MMFR register while frame generation is in progress.

Address: 30BE_0000h base + 40h offset = 30BE_0040h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST		OP		PA					RA					TA		DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_MMFR field descriptions

Field	Description
0–1 ST	Start Of Frame Delimiter See Table 11-42 (Clause 22) or Table 11-44 (Clause 45) for correct value.
2–3 OP	Operation Code See Table 11-42 (Clause 22) or Table 11-44 (Clause 45) for correct value.
4–8 PA	PHY Address See Table 11-42 (Clause 22) or Table 11-44 (Clause 45) for correct value.
9–13 RA	Register Address See Table 11-42 (Clause 22) or Table 11-44 (Clause 45) for correct value.
14–15 TA	Turn Around This field must be programmed to 10 to generate a valid MII management frame.
16–31 DATA	Management Frame Data This is the field for data to be written to or read from the PHY register.

11.1.5.7 MII Speed Control Register (ENET_MSCR)

MSCR provides control of the MII clock (MDC pin) frequency and allows a preamble drop on the MII management frame.

The MII_SPEED field must be programmed with a value to provide an MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value to source a read or write management frame. After the management frame is complete, the MSCR register may optionally be cleared to turn off MDC. The MDC signal generated has a 50% duty cycle except when MII_SPEED changes during operation. This change takes effect following a rising or falling edge of MDC.

For example, if the internal module clock (i.e., IPS bus clock) is 25 MHz, programming MII_SPEED to 0x4 results in an MDC as given in the following equation:

MII clock frequency = 25 MHz / ((4 + 1) x 2) = 2.5 MHz

The following table shows the optimum values for MII_SPEED as a function of IPS bus clock frequency.

Table 11-4. Programming Examples for MSCR

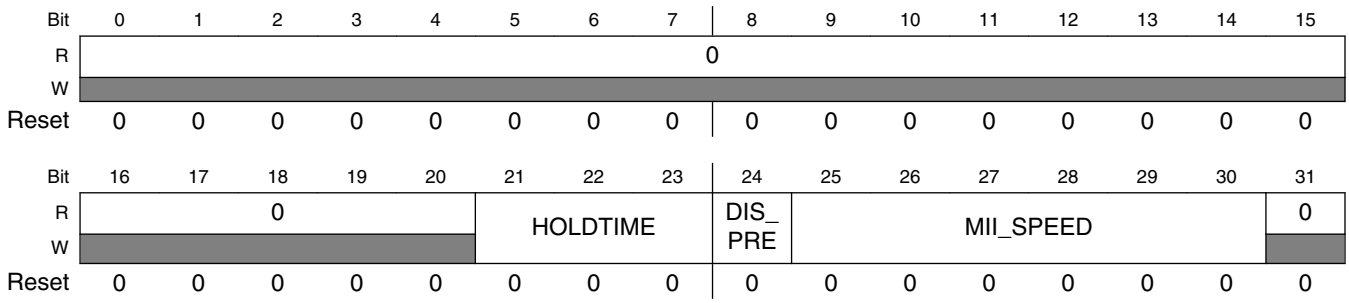
Internal module clock frequency	MSCR [MII_SPEED]	MDC frequency
25 MHz	0x4	2.50 MHz

Table continues on the next page...

Table 11-4. Programming Examples for MSCR (continued)

Internal module clock frequency	MSCR [MII_SPEED]	MDC frequency
33 MHz	0x6	2.36 MHz
40 MHz	0x7	2.50 MHz
50 MHz	0x9	2.50 MHz
66 MHz	0xD	2.36 MHz

Address: 30BE_0000h base + 44h offset = 30BE_0044h



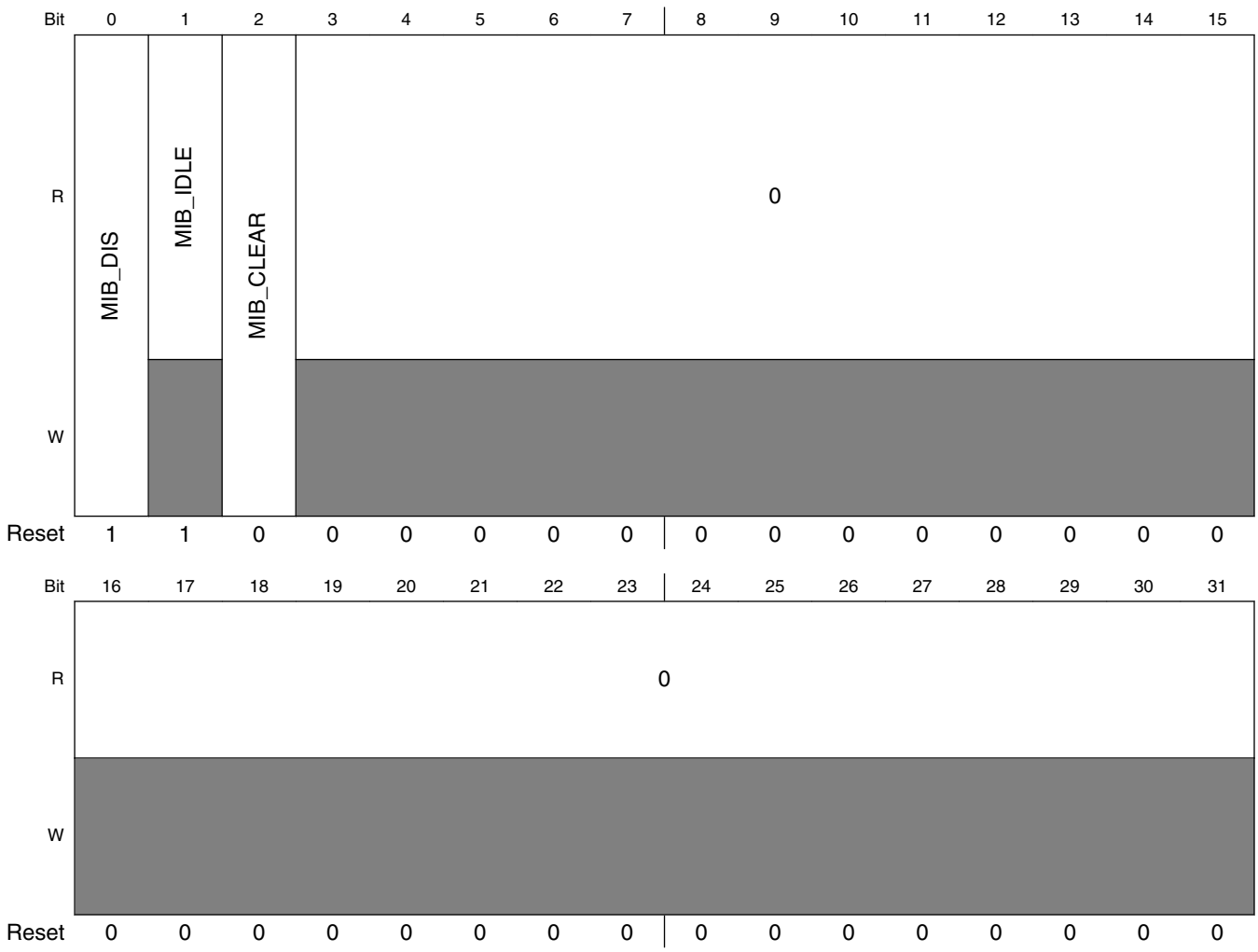
ENET_MSCR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 HOLDTIME	Hold time On MDIO Output IEEE802.3 clause 22 defines a minimum of 10 ns for the hold time on the MDIO output. Depending on the host bus frequency, the setting may need to be increased. 000 1 internal module clock cycle 001 2 internal module clock cycles 010 3 internal module clock cycles 111 8 internal module clock cycles
24 DIS_PRE	Disable Preamble Enables/disables prepending a preamble to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY devices do not require it. 0 Preamble enabled. 1 Preamble (32 ones) is not prepended to the MII management frame.
25–30 MII_SPEED	MII Speed Controls the frequency of the MII management interface clock (MDC) relative to the internal module clock. A value of 0 in this field turns off MDC and leaves it in low voltage state. Any non-zero value results in the MDC frequency of: 1/((MII_SPEED + 1) x 2) of the internal module clock frequency
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.8 MIB Control Register (ENET_MIBC)

MIBC is a read/write register controlling and observing the state of the MIB block. Access this register to disable the MIB block operation or clear the MIB counters. The MIB_DIS field resets to 1.

Address: 30BE_0000h base + 64h offset = 30BE_0064h



ENET_MIBC field descriptions

Field	Description
0 MIB_DIS	Disable MIB Logic If this control field is set, 0 MIB logic is enabled. 1 MIB logic is disabled. The MIB logic halts and does not update any MIB counters.

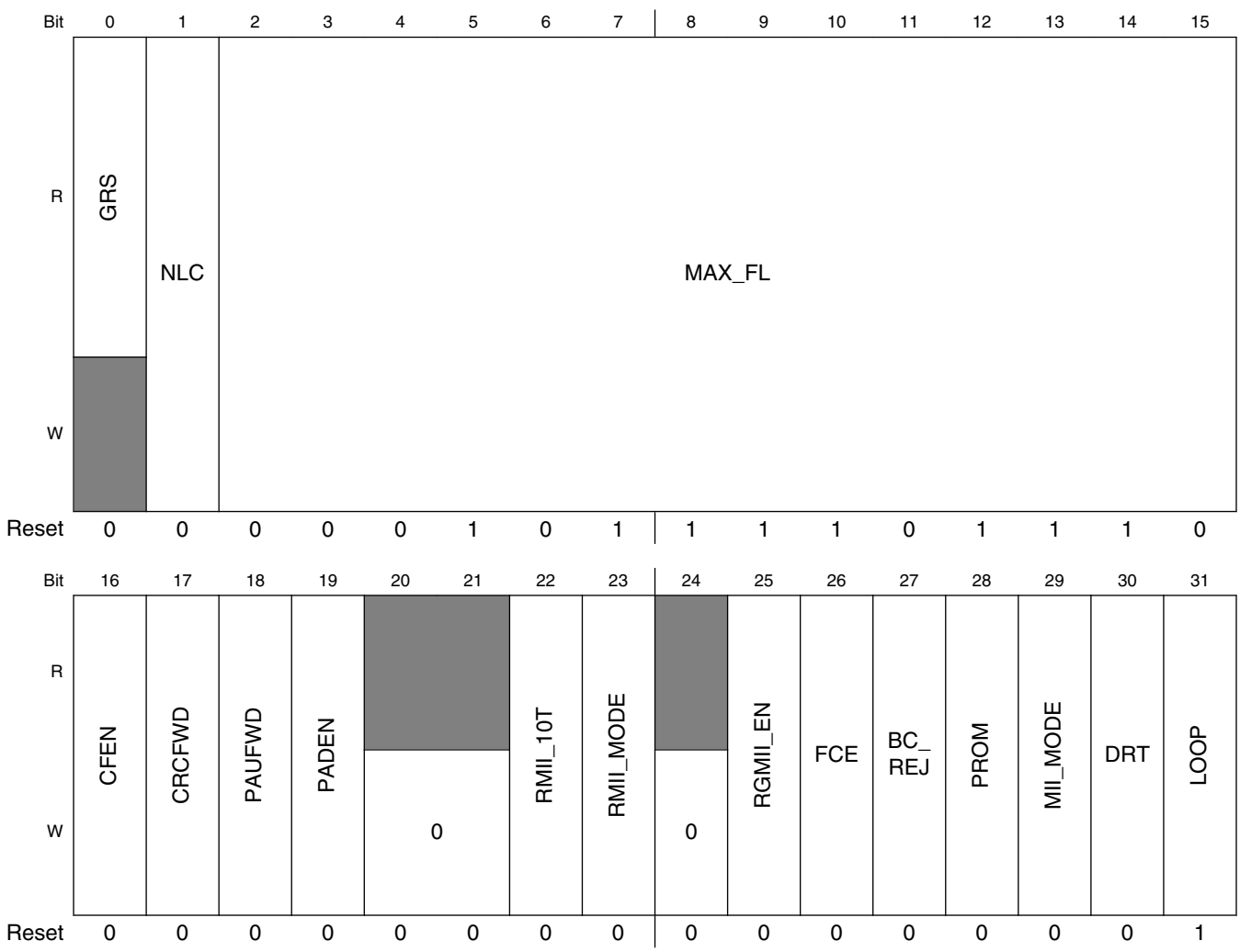
Table continues on the next page...

ENET_MIBC field descriptions (continued)

Field	Description
1 MIB_IDLE	MIB Idle 0 The MIB block is updating MIB counters. 1 The MIB block is not currently updating any MIB counters.
2 MIB_CLEAR	MIB Clear **NOTE:** This field is not self-clearing. To clear the MIB counters set and then clear this field. 0 See note above. 1 All statistics counters are reset to 0.
3–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.9 Receive Control Register (ENET_RCR)

Address: 30BE_0000h base + 84h offset = 30BE_0084h



ENET_RCR field descriptions

Field	Description
0 GRS	Graceful Receive Stopped Read-only status indicating that the MAC receive datapath is stopped.
1 NLC	Payload Length Check Disable Enables/disables a payload length check. 0 The payload length check is disabled. 1 The core checks the frame's payload length with the frame length/type field. Errors are indicated in the EIR[PLC] field.
2–15 MAX_FL	Maximum Frame Length Resets to decimal 1518. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL cause the BABT interrupt to occur. Receive frames longer than MAX_FL cause the BABR interrupt to occur and set the LG field in the end of frame receive buffer descriptor. The recommended default value to be programmed is 1518 or 1522 if VLAN tags are supported.
16 CFEN	MAC Control Frame Enable Enables/disables the MAC control frame. 0 MAC control frames with any opcode other than 0x0001 (pause frame) are accepted and forwarded to the client interface. 1 MAC control frames with any opcode other than 0x0001 (pause frame) are silently discarded.
17 CRCFWD	Terminate/Forward Received CRC Specifies whether the CRC field of received frames is transmitted or stripped. **NOTE:** If padding function is enabled (PADEN = 1), CRCFWD is ignored and the CRC field is checked and always terminated and removed. 0 The CRC field of received frames is transmitted to the user application. 1 The CRC field is stripped from the frame.
18 PAUFWD	Terminate/Forward Pause Frames Specifies whether pause frames are terminated or forwarded. 0 Pause frames are terminated and discarded in the MAC. 1 Pause frames are forwarded to the user application.
19 PADEN	Enable Frame Padding Remove On Receive Specifies whether the MAC removes padding from received frames. 0 No padding is removed on receive by the MAC. 1 Padding is removed from received frames.
20–21 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
22 RMII_10T	Enables 10-Mbit/s mode of the RMII or RGMII . 0 100-Mbit/s operation. 1 10-Mbit/s operation.

Table continues on the next page...

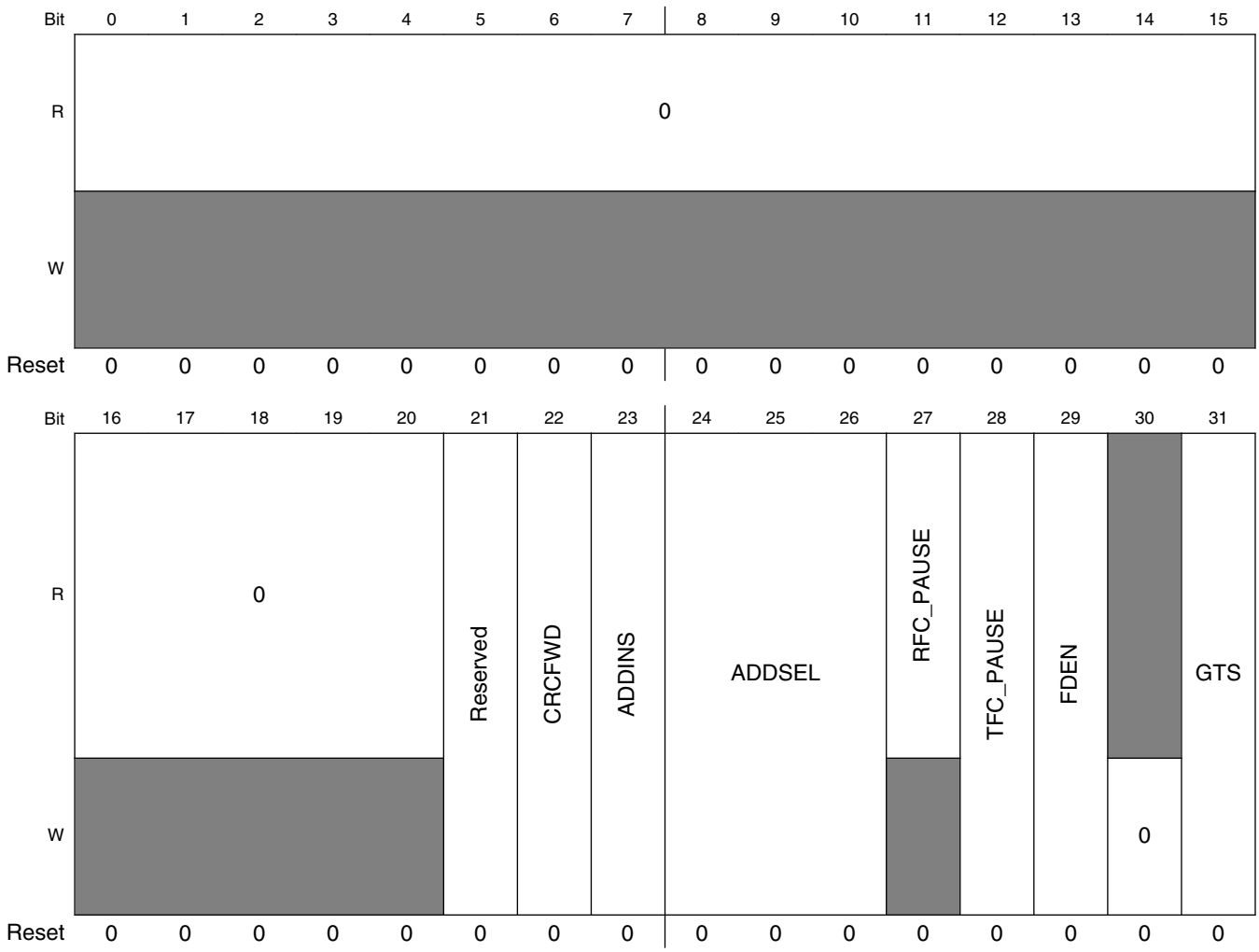
ENET_RCR field descriptions (continued)

Field	Description
23 RMII_MODE	RMII Mode Enable Specifies whether the MAC is configured for MII mode or RMII operation , when ECR[SPEED] is cleared . **NOTE:** Do not set both RCR[RGMII_EN] and RCR[RMII_MODE]. 0 MAC configured for MII mode. 1 MAC configured for RMII operation.
24 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
25 RGMII_EN	RGMII Mode Enable **NOTE:** Do not set both RCR[RGMII_EN] and RCR[RMII_MODE]. 0 MAC configured for non-RGMII operation 1 MAC configured for RGMII operation. If ECR[SPEED] is set, the MAC is in RGMII 1000-Mbit/s mode. If ECR[SPEED] is cleared, the MAC is in RGMII 10/100-Mbit/s mode.
26 FCE	Flow Control Enable If set, the receiver detects PAUSE frames. Upon PAUSE frame detection, the transmitter stops transmitting data frames for a given duration.
27 BC_REJ	Broadcast Frame Reject If set, frames with destination address (DA) equal to 0xFFFF_FFFF_FFFF are rejected unless the PROM field is set. If BC_REJ and PROM are set, frames with broadcast DA are accepted and the MISS (M) is set in the receive buffer descriptor.
28 PROM	Promiscuous Mode All frames are accepted regardless of address matching. 0 Disabled. 1 Enabled.
29 MII_MODE	Media Independent Interface Mode This field must always be set. 0 Reserved. 1 MII or RMII mode, as indicated by the RMII_MODE field.
30 DRT	Disable Receive On Transmit 0 Receive path operates independently of transmit (i.e., full-duplex mode). Can also be used to monitor transmit activity in half-duplex mode. 1 Disable reception of frames while transmitting. (Normally used for half-duplex mode.)
31 LOOP	Internal Loopback This is an MII internal loopback, therefore MII_MODE must be written to 1 and RMII_MODE must be written to 0. 0 Loopback disabled. 1 Transmitted frames are looped back internal to the device and transmit MII output signals are not asserted. DRT must be cleared.

11.1.5.10 Transmit Control Register (ENET_TCR)

TCR is read/write and configures the transmit block. This register is cleared at system reset. FDEN can only be modified when ECR[ETHEREN] is cleared.

Address: 30BE_0000h base + C4h offset = 30BE_00C4h



ENET_TCR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This field is read/write and must be set to 0.
22 CRCFWD	Forward Frame From Application With CRC

Table continues on the next page...

ENET_TCR field descriptions (continued)

Field	Description
	0 TxBD[TC] controls whether the frame has a CRC from the application. 1 The transmitter does not append any CRC to transmitted frames, as it is expecting a frame with CRC from the application.
23 ADDINS	Set MAC Address On Transmit 0 The source MAC address is not modified by the MAC. 1 The MAC overwrites the source MAC address with the programmed MAC address according to ADDSEL.
24–26 ADDSEL	Source MAC Address Select On Transmit If ADDINS is set, indicates the MAC address that overwrites the source MAC address. 000 Node MAC address programmed on PADDR1/2 registers. 100 Reserved. 101 Reserved. 110 Reserved.
27 RFC_PAUSE	Receive Frame Control Pause This status field is set when a full-duplex flow control pause frame is received and the transmitter pauses for the duration defined in this pause frame. This field automatically clears when the pause duration is complete.
28 TFC_PAUSE	Transmit Frame Control Pause Pauses frame transmission. When this field is set, EIR[GRA] is set. With transmission of data frames stopped, the MAC transmits a MAC control PAUSE frame. Next, the MAC clears TFC_PAUSE and resumes transmitting data frames. If the transmitter pauses due to user assertion of GTS or reception of a PAUSE frame, the MAC may continue transmitting a MAC control PAUSE frame. 0 No PAUSE frame transmitted. 1 The MAC stops transmission of data frames after the current transmission is complete.
29 FDEN	Full-Duplex Enable If this field is set, frames transmit independent of carrier sense and collision inputs. Only modify this bit when ECR[ETHEREN] is cleared.
30 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
31 GTS	Graceful Transmit Stop When this field is set, MAC stops transmission after any frame currently transmitted is complete and EIR[GRA] is set. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission finishes, clear GTS to restart. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS is set, transmission stops after the collision. The frame is transmitted again after GTS is cleared. There may be old frames in the transmit FIFO that transmit when GTS is reasserted. To avoid this, clear ECR[ETHEREN] following the GRA interrupt.

11.1.5.11 Physical Address Lower Register (ENET_PALR)

PALR contains the lower 32 bits (bytes 0, 1, 2, 3) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in bytes 0 through 3 of the six-byte source address field when transmitting PAUSE frames.

Address: 30BE_0000h base + E4h offset = 30BE_00E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_PALR field descriptions

Field	Description
0–31 PADDR1	Pause Address Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match and the source address field in PAUSE frames.

11.1.5.12 Physical Address Upper Register (ENET_PAUR)

PAUR contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the six-byte source address field when transmitting PAUSE frames. Bits 15:0 of PAUR contain a constant type field (0x8808) for transmission of PAUSE frames.

Address: 30BE_0000h base + E8h offset = 30BE_00E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PADDR2																TYPE															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

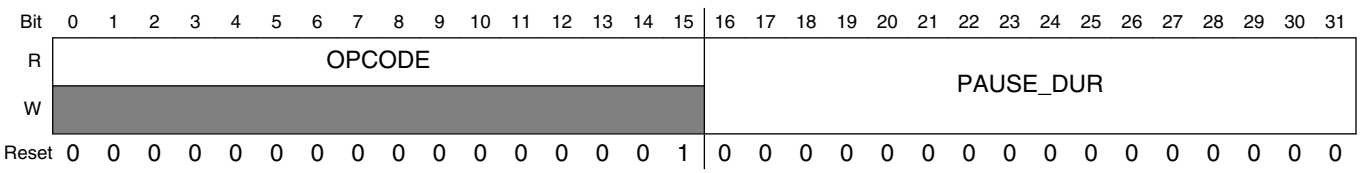
ENET_PAUR field descriptions

Field	Description
0–15 PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match, and the source address field in PAUSE frames.
16–31 TYPE	Type Field In PAUSE Frames These fields have a constant value of 0x8808.

11.1.5.13 Opcode/Pause Duration Register (ENET_OPD)

OPD is read/write accessible. This register contains the 16-bit opcode and 16-bit pause duration fields used in transmission of a PAUSE frame. The opcode field is a constant value, 0x0001. When another node detects a PAUSE frame, that node pauses transmission for the duration specified in the pause duration field. The lower 16 bits of this register are not reset and you must initialize it.

Address: 30BE_0000h base + ECh offset = 30BE_00ECh



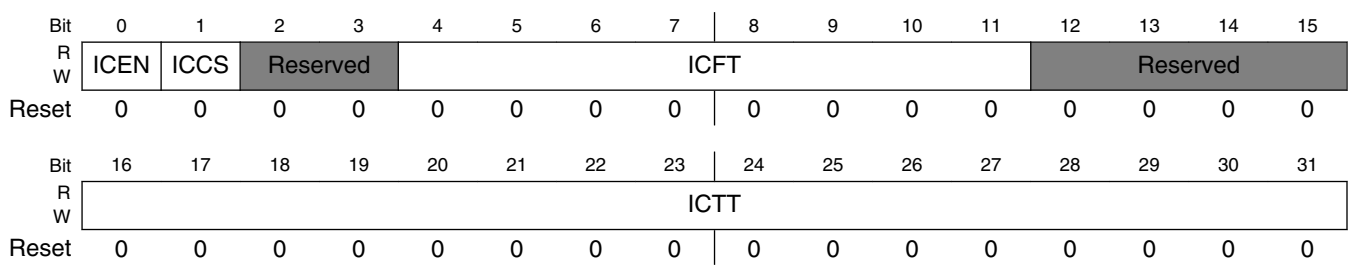
ENET_OPD field descriptions

Field	Description
0–15 OPCODE	Opcode Field In PAUSE Frames These fields have a constant value of 0x0001.
16–31 PAUSE_DUR	Pause Duration Pause duration field used in PAUSE frames.

11.1.5.14 Transmit Interrupt Coalescing Register (ENET_TXIC*n*)

See Interrupt coalescence for more information.

Address: 30BE_0000h base + F0h offset + (4d × i), where i=0d to 2d



ENET_TXIC*n* field descriptions

Field	Description
0 ICEN	Interrupt Coalescing Enable

Table continues on the next page...

ENET_TXIC*n* field descriptions (continued)

Field	Description
	0 Disable Interrupt coalescing. 1 Enable Interrupt coalescing.
1 ICCS	Interrupt Coalescing Timer Clock Source Select 0 Use MII/GMII TX clocks. 1 Use ENET system clock.
2–3 Reserved	This field must be set to 0. This field is reserved.
4–11 ICFT	Interrupt coalescing frame count threshold This value determines the number of frames needed to be transmitted for raising an interrupt. Frame counter restarts after reaching this threshold value or after the expiring of the coalescing timer. Must be greater than zero to avoid unpredictable behavior.
12–15 Reserved	This field must be set to 0. This field is reserved.
16–31 ICTT	Interrupt coalescing timer threshold Interrupt coalescing timer threshold in units of 64 clock periods. This value determines the maximum amount of time after transmitting a frame before raising an interrupt. The threshold timer is disabled after expiring or number of frame transmission defined by ICFT and starts again upon transmission of the next first frame. Must be greater than zero to avoid unpredictable behavior.

11.1.5.15 Receive Interrupt Coalescing Register (ENET_RXIC*n*)

See Interrupt coalescence for more information.

Address: 30BE_0000h base + 100h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ICEN	ICCS	Reserved		ICFT								Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ICTT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RXIC*n* field descriptions

Field	Description
0 ICEN	Interrupt Coalescing Enable 0 Disable Interrupt coalescing. 1 Enable Interrupt coalescing.

Table continues on the next page...

ENET_RXIC*n* field descriptions (continued)

Field	Description
1 ICCS	Interrupt Coalescing Timer Clock Source Select 0 Use MII/GMII TX clocks. 1 Use ENET system clock.
2–3 Reserved	This field must be set to 0. This field is reserved.
4–11 ICFT	Interrupt coalescing frame count threshold This value determines the number of frames needed to be received for raising an interrupt. Frame counter restarts after reaching this threshold value or after the expiring of the coalescing timer. Must be greater than zero to avoid unpredictable behavior.
12–15 Reserved	This field must be set to 0. This field is reserved.
16–31 ICTT	Interrupt coalescing timer threshold Interrupt coalescing timer threshold in units of 64 clock periods. This value determines the maximum amount of time after receiving a frame before raising an interrupt. The threshold timer is disabled after expiring or number of frame reception defined by ICFT and starts again upon reception of the next first frame. Must be greater than zero to avoid unpredictable behavior.

11.1.5.16 Descriptor Individual Upper Address Register (ENET_IAUR)

IAUR contains the upper 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the destination address (DA) field of receive frames with an individual DA. This register is not reset and you must initialize it.

Address: 30BE_0000h base + 118h offset = 30BE_0118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

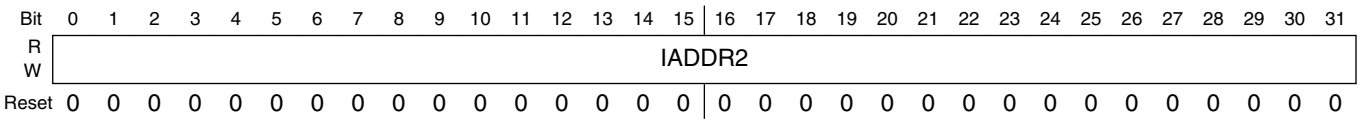
ENET_IAUR field descriptions

Field	Description
0–31 IADDR1	Contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

11.1.5.17 Descriptor Individual Lower Address Register (ENET_IALR)

IALR contains the lower 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the DA field of receive frames with an individual DA. This register is not reset and you must initialize it.

Address: 30BE_0000h base + 11Ch offset = 30BE_011Ch



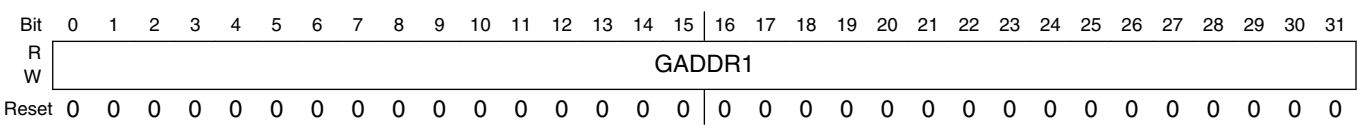
ENET_IALR field descriptions

Field	Description
0–31 IADDR2	Contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

11.1.5.18 Descriptor Group Upper Address Register (ENET_GAUR)

GAUR contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. You must initialize this register.

Address: 30BE_0000h base + 120h offset = 30BE_0120h



ENET_GAUR field descriptions

Field	Description
0–31 GADDR1	Contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

11.1.5.19 Descriptor Group Lower Address Register (ENET_GALR)

GALR contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. You must initialize this register.

Address: 30BE_0000h base + 124h offset = 30BE_0124h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GADDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_GALR field descriptions

Field	Description
0–31 GADDR2	Contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

11.1.5.20 Transmit FIFO Watermark Register (ENET_TFWR)

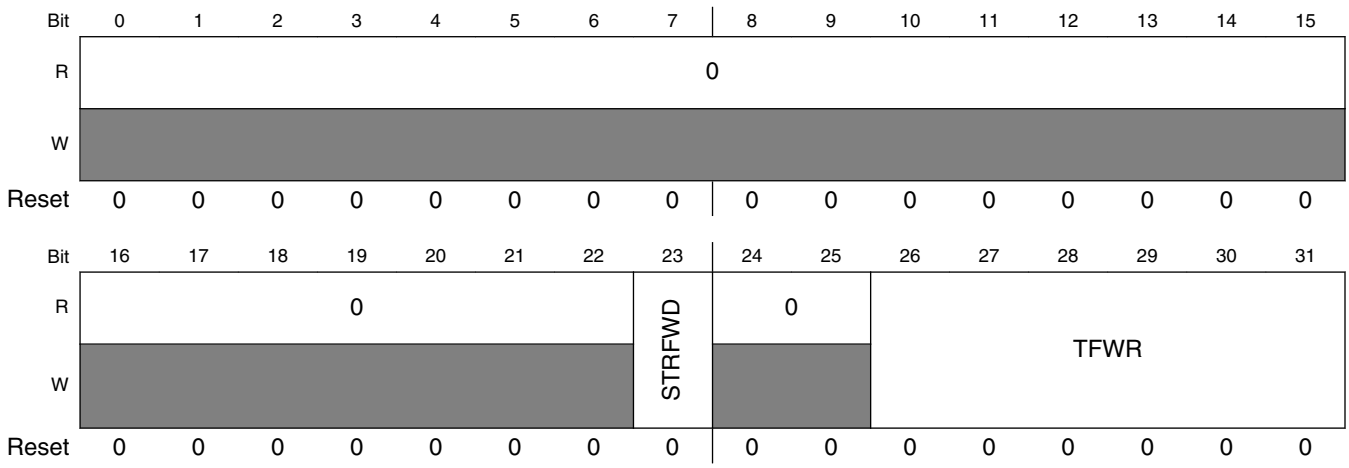
If TFWR[STRFWD] is cleared, TFWR[TFWR] controls the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows you to minimize transmit latency (TFWR = 00 or 01) or allow for larger bus access latency (TFWR = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the TFWR field may need to be modified to match a given system requirement, for example, worst-case bus access latency by the transmit data uDMA channel.

When the FIFO level reaches the value the TFWR field and when the STR_FWD is set to ‘0’, the MAC transmit control logic starts frame transmission even before the end-of-frame is available in the FIFO (cut-through operation).

If a complete frame has a size smaller than the threshold programmed with TFWR, the MAC also transmits the Frame to the line.

To enable store and forward on the Transmit path, set STR_FWD to ‘1’. In this case, the MAC starts to transmit data only when a complete frame is stored in the Transmit FIFO.

Address: 30BE_0000h base + 144h offset = 30BE_0144h



ENET_TFWR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 STRFWD	Store And Forward Enable 0 Reset. The transmission start threshold is programmed in TFWR[TFWR]. 1 Enabled.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 TFWR	Transmit FIFO Write If TFWR[STRFWD] is cleared, this field indicates the number of bytes, in steps of 64 bytes, written to the transmit FIFO before transmission of a frame begins. **NOTE:** If a frame with less than the threshold is written, it is still sent independently of this threshold setting. The threshold is relevant only if the frame is larger than the threshold given. 000000 64 bytes written. 000001 64 bytes written. 000010 128 bytes written. 000011 192 bytes written. ... ... 111111 4032 bytes written.

11.1.5.21 Receive Descriptor Ring 1 Start Register (ENET_RDSR1)

RDSR1 points to the beginning of circular receive buffer descriptor queue 1 in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation.

Address: 30BE_0000h base + 160h offset = 30BE_0160h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	R_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	R_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RDSR1 field descriptions

Field	Description
0–28 R_DES_START	Pointer to the beginning of the receive buffer descriptor queue 1.
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.22 Transmit Buffer Descriptor Ring 1 Start Register (ENET_TDSR1)

TDSR1 provides a pointer to the beginning of the circular transmit buffer descriptor queue 1 in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation.

Address: 30BE_0000h base + 164h offset = 30BE_0164h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	X_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	X_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TDSR1 field descriptions

Field	Description
0–28 X_DES_START	Pointer to the beginning of transmit buffer descriptor queue 1.

Table continues on the next page...

ENET_TDSR1 field descriptions (continued)

Field	Description
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.23 Maximum Receive Buffer Size Register - Ring 1 (ENET_MRBR1)

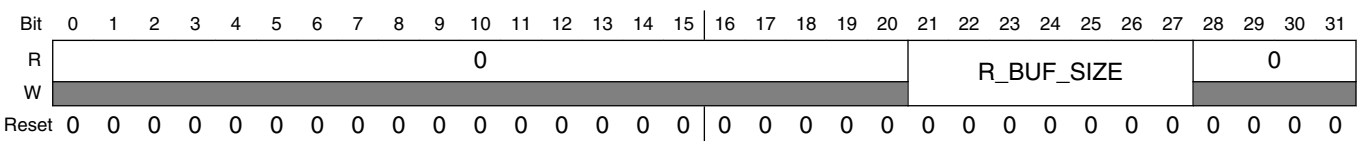
MRBR1 is a user-programmable register that dictates the maximum size of all ring-1 receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer.

- To allow one maximum size frame per buffer, MRBR1 must be set to RCR[MAX_FL] or larger.
- R_BUF_SIZE is concatentated with the four least-significant bits of this register and are used as the maximum receive buffer size.
- To properly align the buffer, MRBR1 must be evenly divisible by 64. To ensure this, set the lower two bits of R_BUF_ZIZE to zero. The lower four bits are already set to zero by the device.
- To minimize bus usage (descriptor fetches), set MRBR1 greater than or equal to 256 bytes.

NOTE

This register must be initialized before operation.

Address: 30BE_0000h base + 168h offset = 30BE_0168h



ENET_MRBR1 field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–27 R_BUF_SIZE	Receive buffer size in bytes. This value, concatenated with the four least-significant bits of this register (which are always zero), is the effective maximum receive buffer size.
28–31 Reserved	This field, which is always zero, is the four least-significant bits of the maximum receive buffer size. This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.24 Receive Descriptor Ring 2 Start Register (ENET_RDSR2)

RDSR points to the beginning of circular receive buffer descriptor queue 2 in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation

Address: 30BE_0000h base + 16Ch offset = 30BE_016Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	R_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	R_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RDSR2 field descriptions

Field	Description
0–28 R_DES_START	Pointer to the beginning of receive buffer descriptor queue 2.
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.25 Transmit Buffer Descriptor Ring 2 Start Register (ENET_TDSR2)

TDSR2 provides a pointer to the beginning of circular transmit buffer descriptor queue 2 in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation

Address: 30BE_0000h base + 170h offset = 30BE_0170h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	X_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	X_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TDSR2 field descriptions

Field	Description
0–28 X_DES_START	Pointer to the beginning of transmit buffer descriptor queue 2.
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.26 Maximum Receive Buffer Size Register - Ring 2 (ENET_MRBR2)

MRBR2 is a user-programmable register that dictates the maximum size of all ring-2 receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer.

- To allow one maximum size frame per buffer, MRBR2 must be set to RCR[MAX_FL] or larger.
- R_BUF_SIZE is concatentated with the four least-significant bits of this register and are used as the maximum receive buffer size.
- To properly align the buffer, MRBR2 must be evenly divisible by 64. To ensure this, set the lower two bits of R_BUF_SIZE to zero. The lower four bits are already set to zero by the device.
- To minimize bus usage (descriptor fetches), set MRBR2 greater than or equal to 256 bytes.

NOTE

This register must be initialized prior to operation

Address: 30BE_0000h base + 174h offset = 30BE_0174h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																R_BUF_SIZE												0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_MRBR2 field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–27 R_BUF_SIZE	Receive buffer size in bytes. This value, concatenated with the four least-significant bits of this register (which are always zero), is the effective maximum receive buffer size.
28–31 Reserved	This field, which is always zero, is the four least-significant bits of the maximum receive buffer size. This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.27 Receive Descriptor Ring 0 Start Register (ENET_RDSR)

RDSR points to the beginning of the circular receive buffer descriptor queue in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation

Address: 30BE_0000h base + 180h offset = 30BE_0180h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	R_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	R_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RDSR field descriptions

Field	Description
0–28 R_DES_START	Pointer to the beginning of the receive buffer descriptor queue. 0
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.28 Transmit Buffer Descriptor Ring 0 Start Register (ENET_TDSR)

TDSR provides a pointer to the beginning of the circular transmit buffer descriptor queue 0 in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation.

Address: 30BE_0000h base + 184h offset = 30BE_0184h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	X_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	X_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TDSR field descriptions

Field	Description
0–28 X_DES_START	Pointer to the beginning of the transmit buffer descriptor queue.
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.29 Maximum Receive Buffer Size Register - Ring 0 (ENET_MRBR)

The MRBR is a user-programmable register that dictates the maximum size of all ring-0 receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer.

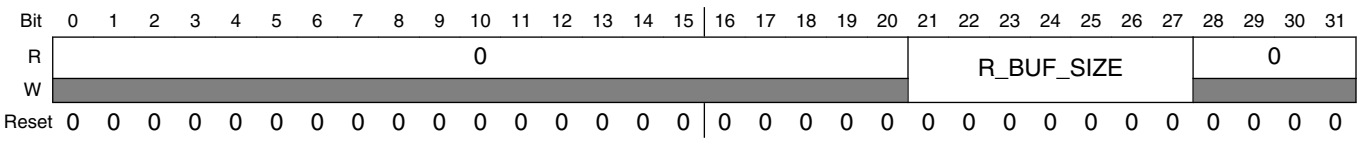
- R_BUF_SIZE is concatentated with the four least-significant bits of this register and are used as the maximum receive buffer size.
- To allow one maximum size frame per buffer, MRBR must be set to RCR[MAX_FL] or larger.

- To properly align the buffer, MRBR must be evenly divisible by 64. To ensure this, set the lower two bits of R_BUF_SIZE to zero. The lower four bits of this register are already set to zero by the device.
- To minimize bus usage (descriptor fetches), set MRBR greater than or equal to 256 bytes.

NOTE

This register must be initialized before operation.

Address: 30BE_0000h base + 188h offset = 30BE_0188h

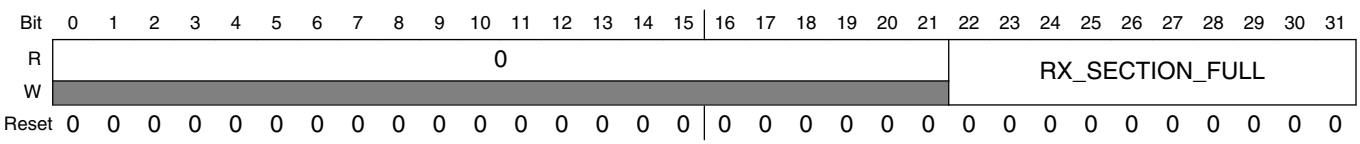


ENET_MRBR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–27 R_BUF_SIZE	Receive buffer size in bytes. This value, concatenated with the four least-significant bits of this register (which are always zero), is the effective maximum receive buffer size.
28–31 Reserved	This field, which is always zero, is the four least-significant bits of the maximum receive buffer size. This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.30 Receive FIFO Section Full Threshold (ENET_RSFL)

Address: 30BE_0000h base + 190h offset = 30BE_0190h

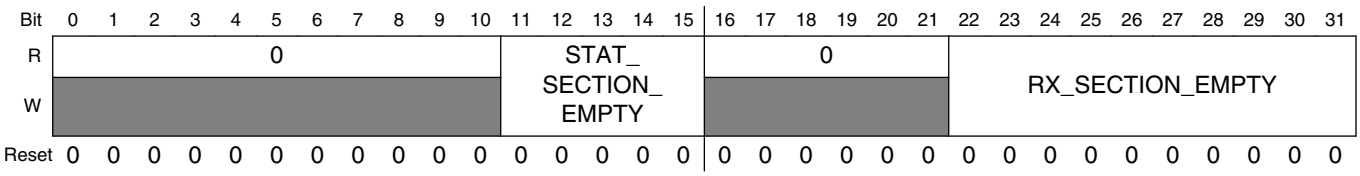


ENET_RSFL field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–31 RX_SECTION_ FULL	Value Of Receive FIFO Section Full Threshold Value, in 64-bit words, of the receive FIFO section full threshold. Clear this field to enable store and forward on the RX FIFO. When programming a value greater than 0 (cut-through operation), it must be greater than RAEM[RX_ALMOST_EMPTY]. When the FIFO level reaches the value in this field, data is available in the Receive FIFO (cut-through operation).

11.1.5.31 Receive FIFO Section Empty Threshold (ENET_RSEM)

Address: 30BE_0000h base + 194h offset = 30BE_0194h

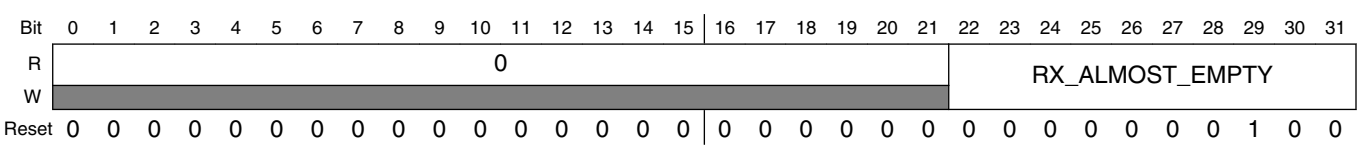


ENET_RSEM field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 STAT_ SECTION_ EMPTY	RX Status FIFO Section Empty Threshold Defines number of frames in the receive FIFO, independent of its size, that can be accepted. If the limit is reached, reception will continue normally, however a pause frame will be triggered to indicate a possible congestion to the remote device to avoid FIFO overflow. A value of 0 disables automatic pause frame generation
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–31 RX_SECTION_ EMPTY	Value Of The Receive FIFO Section Empty Threshold Value, in 64-bit words, of the receive FIFO section empty threshold. When the FIFO has reached this level, a pause frame will be issued. A value of 0 disables automatic pause frame generation. When the FIFO level goes below the value programmed in this field, an XON pause frame is issued to indicate the FIFO congestion is cleared to the remote Ethernet client. **NOTE:** The section-empty threshold indications from both FIFOs are OR'ed to cause XOFF pause frame generation.

11.1.5.32 Receive FIFO Almost Empty Threshold (ENET_RAEM)

Address: 30BE_0000h base + 198h offset = 30BE_0198h



ENET_RAEM field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

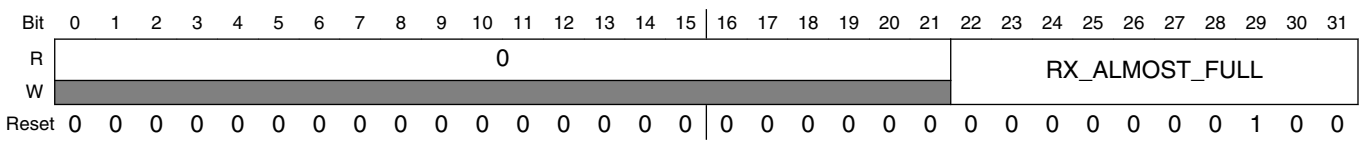
Table continues on the next page...

ENET_RAEM field descriptions (continued)

Field	Description
22–31 RX_ALMOST_ EMPTY	Value Of The Receive FIFO Almost Empty Threshold Value, in 64-bit words, of the receive FIFO almost empty threshold. When the FIFO level reaches the value programmed in this field and the end-of-frame has not been received for the frame yet, the core receive read control stops FIFO read (and subsequently stops transferring data to the MAC client application). It continues to deliver the frame, if again more data than the threshold or the end-of-frame is available in the FIFO. A minimum value of 4 should be set.

11.1.5.33 Receive FIFO Almost Full Threshold (ENET_RAFL)

Address: 30BE_0000h base + 19Ch offset = 30BE_019Ch

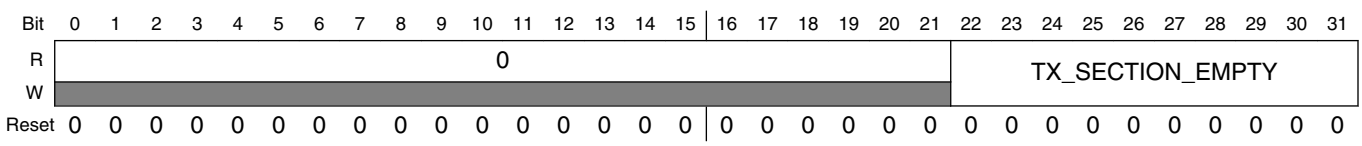


ENET_RAFL field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–31 RX_ALMOST_ FULL	Value Of The Receive FIFO Almost Full Threshold Value, in 64-bit words, of the receive FIFO almost full threshold. When the FIFO level comes close to the maximum, so that there is no more space for at least RX_ALMOST_FULL number of words, the MAC stops writing data in the FIFO and truncates the received frame to avoid FIFO overflow. The corresponding error status will be set when the frame is delivered to the application. A minimum value of 4 should be set.

11.1.5.34 Transmit FIFO Section Empty Threshold (ENET_TSEM)

Address: 30BE_0000h base + 1A0h offset = 30BE_01A0h



ENET_TSEM field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

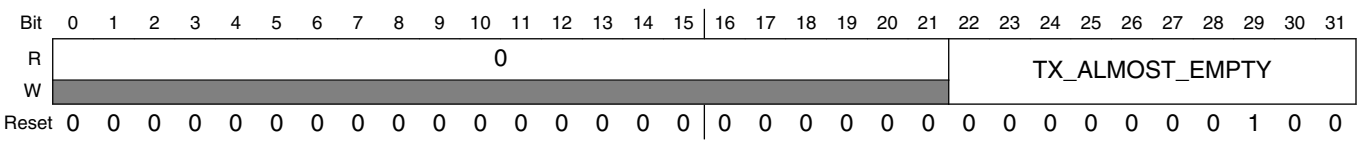
Table continues on the next page...

ENET_TSEM field descriptions (continued)

Field	Description
22–31 TX_SECTION_EMPTY	Value Of The Transmit FIFO Section Empty Threshold Value, in 64-bit words, of the transmit FIFO section empty threshold. See Transmit FIFO for more information.

11.1.5.35 Transmit FIFO Almost Empty Threshold (ENET_TAEM)

Address: 30BE_0000h base + 1A4h offset = 30BE_01A4h

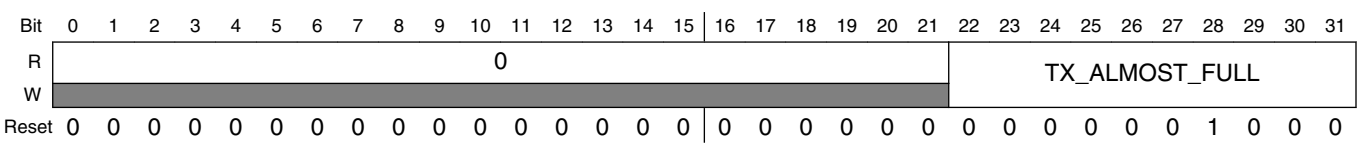


ENET_TAEM field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–31 TX_ALMOST_EMPTY	Value of Transmit FIFO Almost Empty Threshold Value, in 64-bit words, of the transmit FIFO almost empty threshold. When the FIFO level reaches the value programmed in this field, and no end-of-frame is available for the frame, the MAC transmit logic, to avoid FIFO underflow, stops reading the FIFO and transmits a frame with an MII error indication. See Transmit FIFO for more information. A minimum value of 4 should be set.

11.1.5.36 Transmit FIFO Almost Full Threshold (ENET_TAFL)

Address: 30BE_0000h base + 1A8h offset = 30BE_01A8h



ENET_TAFL field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–31 TX_ALMOST_FULL	Value Of The Transmit FIFO Almost Full Threshold

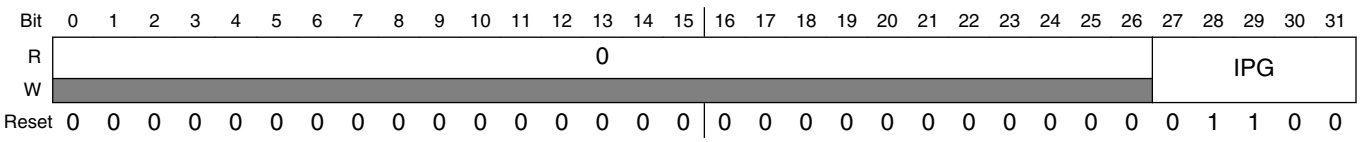
Table continues on the next page...

ENET_TAFL field descriptions (continued)

Field	Description
	Value, in 64-bit words, of the transmit FIFO almost full threshold. A minimum value of six is required . A recommended value of at least 8 should be set allowing a latency of two clock cycles to the application. If more latency is required the value can be increased as necessary (latency = TAFL - 5). When the FIFO level comes close to the maximum, so that there is no more space for at least TX_ALMOST_FULL number of words, the pin ff_tx_rdy is deasserted. If the application does not react on this signal, the FIFO write control logic, to avoid FIFO overflow, truncates the current frame and sets the error status. As a result, the frame will be transmitted with an GMII/MII error indication. See Transmit FIFO for more information. **NOTE:** A FIFO overflow is a fatal error and requires a global reset on the transmit datapath or at least deassertion of ETHEREN.

11.1.5.37 Transmit Inter-Packet Gap (ENET_TIPG)

Address: 30BE_0000h base + 1ACh offset = 30BE_01ACh

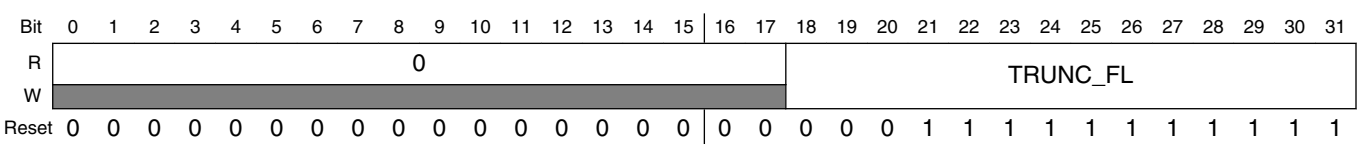


ENET_TIPG field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 IPG	Transmit Inter-Packet Gap Indicates the IPG, in bytes, between transmitted frames. Valid values range from 8 to 26. If the written value is less than 8 or greater than 26, the internal (effective) IPG is 12. **NOTE:** The IPG value read will be the value that was written, even if it is out of range.

11.1.5.38 Frame Truncation Length (ENET_FTRL)

Address: 30BE_0000h base + 1B0h offset = 30BE_01B0h



ENET_FTRL field descriptions

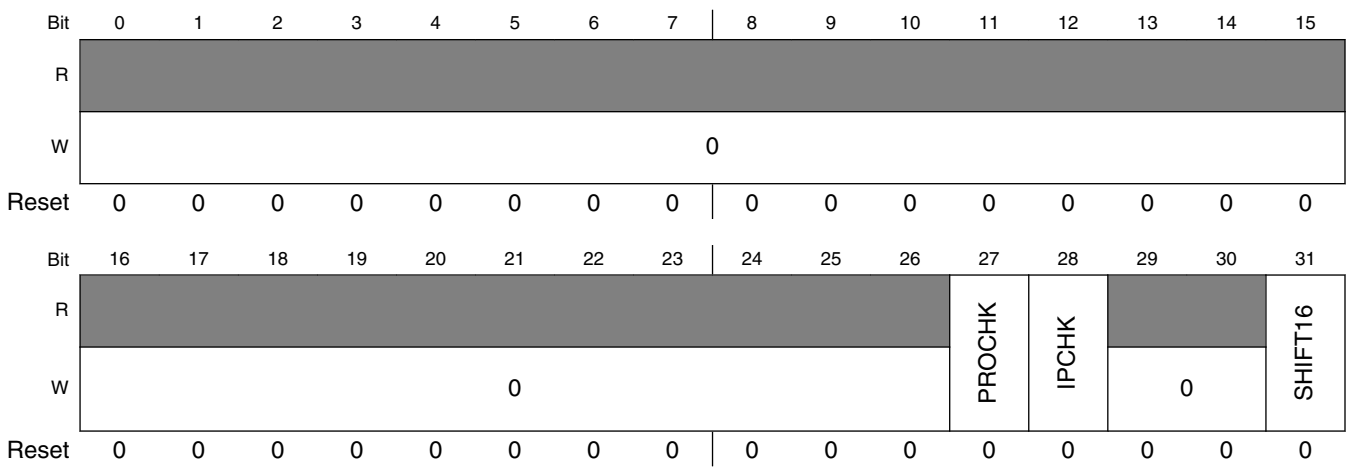
Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–31 TRUNC_FL	Frame Truncation Length Indicates the value a receive frame is truncated, if it is greater than this value. Must be greater than or equal to RCR[MAX_FL]. **NOTE:** Truncation happens at TRUNC_FL. However, when truncation occurs, the application (FIFO) may receive less data, guaranteeing that it never receives more than the set limit.

11.1.5.39 Transmit Accelerator Function Configuration (ENET_TACC)

TACC controls accelerator actions when sending frames. The register can be changed before or after each frame, but it must remain unmodified during frame writes into the transmit FIFO.

The TFWR[STRFWD] field must be set to use the checksum feature.

Address: 30BE_0000h base + 1C0h offset = 30BE_01C0h



ENET_TACC field descriptions

Field	Description
0–26 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
27 PROCHK	Enables insertion of protocol checksum. 0 Checksum not inserted. 1 If an IP frame with a known protocol is transmitted, the checksum is inserted automatically into the frame. The checksum field must be cleared. The other frames are not modified.
28 IPCHK	Enables insertion of IP header checksum.

Table continues on the next page...

ENET_TACC field descriptions (continued)

Field	Description
	0 Checksum is not inserted. 1 If an IP frame is transmitted, the checksum is inserted automatically. The IP header checksum field must be cleared. If a non-IP frame is transmitted the frame is not modified.
29–30 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
31 SHIFT16	TX FIFO Shift-16 0 Disabled. 1 Indicates to the transmit data FIFO that the written frames contain two additional octets before the frame data. This means the actual frame begins at bit 16 of the first word written into the FIFO. This function allows putting the frame payload on a 32-bit boundary in memory, as the 14-byte Ethernet header is extended to a 16-byte header.

11.1.5.40 Receive Accelerator Function Configuration (ENET_RACC)

Address: 30BE_0000h base + 1C4h offset = 30BE_01C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									SHIFT16	LINEDIS				PRODIS	IPDIS	PADREM
W	0										0					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RACC field descriptions

Field	Description
0–23 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
24 SHIFT16	RX FIFO Shift-16 When this field is set, the actual frame data starts at bit 16 of the first word read from the RX FIFO aligning the Ethernet payload on a 32-bit boundary. **NOTE:** This function only affects the FIFO storage and has no influence on the statistics, which use the actual length of the frame received. 0 Disabled. 1 Instructs the MAC to write two additional bytes in front of each frame received into the RX FIFO.

Table continues on the next page...

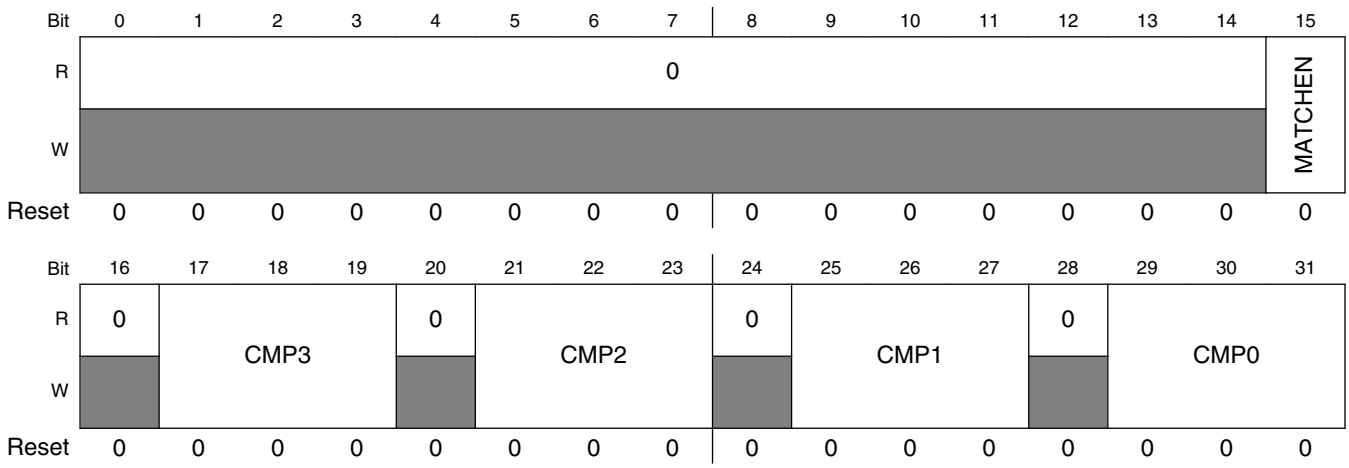
ENET_RACC field descriptions (continued)

Field	Description
25 LINEDIS	Enable Discard Of Frames With MAC Layer Errors 0 Frames with errors are not discarded. 1 Any frame received with a CRC, length, or PHY error is automatically discarded and not forwarded to the user application interface.
26–28 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
29 PRODIS	Enable Discard Of Frames With Wrong Protocol Checksum 0 Frames with wrong checksum are not discarded. 1 If a TCP/IP, UDP/IP, or ICMP/IP frame is received that has a wrong TCP, UDP, or ICMP checksum, the frame is discarded. Discarding is only available when the RX FIFO operates in store and forward mode (RSFL cleared).
30 IPDIS	Enable Discard Of Frames With Wrong IPv4 Header Checksum 0 Frames with wrong IPv4 header checksum are not discarded. 1 If an IPv4 frame is received with a mismatching header checksum, the frame is discarded. IPv6 has no header checksum and is not affected by this setting. Discarding is only available when the RX FIFO operates in store and forward mode (RSFL cleared).
31 PADREM	Enable Padding Removal For Short IP Frames 0 Padding not removed. 1 Any bytes following the IP payload section of the frame are removed from the frame.

11.1.5.41 Receive Classification Match Register for Class n (ENET_RCMR*n*)

This match register allows specifying up to four priorities, which are tested (OR'ed) simultaneously. The match detection uses the extracted VLAN field according to the rules for VLAN detection configured through the ECR register. If both match registers, RCMR1 and RCMR2, report a match at the same time, only the class 1 match is indicated as the final result.

Address: 30BE_0000h base + 1C8h offset + (4d × i), where i=0d to 1d



ENET_RCMR*n* field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 MATCHEN	Match Enable **NOTE:** A comparison is done only on incoming VLAN frames. If no VLAN frame is received no match will occur. If both match registers have overlapping compare values and hence can match both on the same frame, only class 1 will be indicated and the class 2 match is ignored. 0 Disabled (default): no compares will occur and the classification indicator for this class will never assert. 1 The register contents are valid and a comparison with all compare values is done when a VLAN frame is received.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 CMP3	Compare 3 Fourth value to compare against. If unused it must be set to the same value as CMP0.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ENET_RCMR*n* field descriptions (continued)

Field	Description
21–23 CMP2	Compare 2 Third value to compare against. If unused it must be set to the same value as CMP0.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 CMP1	Compare 1 Second value to compare against. If unused it must be set to the same value as CMP0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 CMP0	Compare 0 A three-bit value that will be compared with the frame's VLAN priority field (if a VLAN frame is received). All four compare values, CMP0..3, will be used in parallel. If any of the values match, a match for the class is reported (if MATCHEN is 1). **NOTE:** To implement a single priority match, all four compare values must be set to the same value.

11.1.5.42 DMA Class Based Configuration (ENET_DMA*n*CFG)

The DMA class based configuration registers are used to configure the DMA controller interface to support the additional class 1 (buffer descriptor ring 1) and class 2 (buffer descriptor ring 2) traffic and define configuration options such as bandwidth allocation as needed.

NOTE

The registers are cleared when ECR[ETHEREN] becomes 0.

Address: 30BE_0000h base + 1D8h offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								Reserved						CALC_NOIPG	DMA_CLASS_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IDLE_SLOPE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

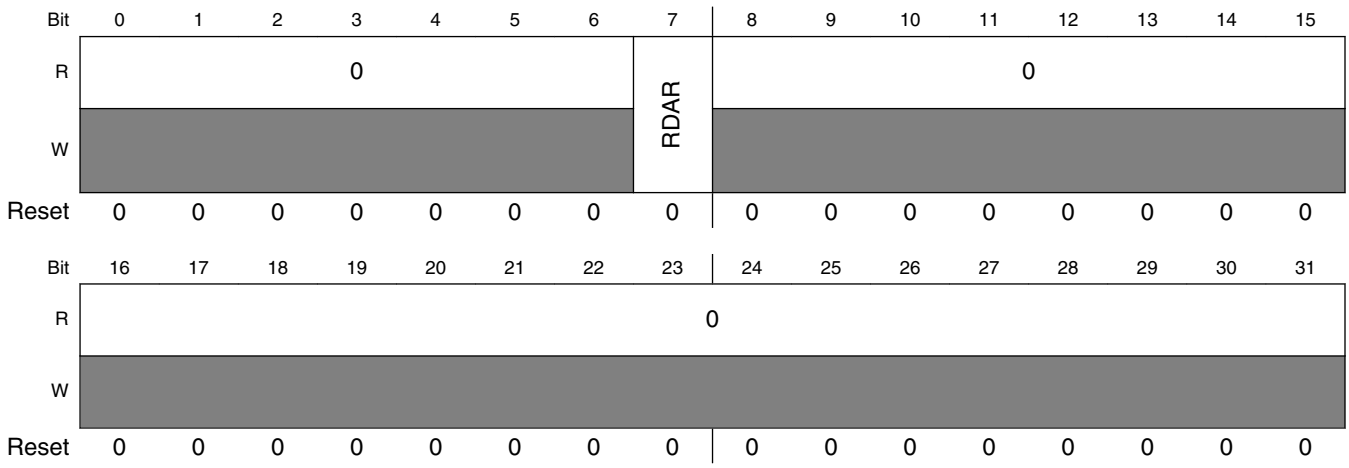
ENET_DMA*n*CFG field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–13 Reserved	This field is reserved. **NOTE:** Write only zeroes to this field.
14 CALC_NOIPG	Calculate no IPG Disable inclusion of IPG bytes for bandwidth calculations. 0 The traffic shaper function should consider 12 octets of IPG in addition to the frame data transferred for a frame when doing bandwidth calculations. This is the default. 1 Addition of 12 bytes for the IPG should be omitted when calculating the bandwidth (for traffic shaping, when writing a frame into the transmit FIFO, the shaper will usually consider 12 bytes of IPG for every frame as part of the bandwidth allocated by the frame. This addition can be suppressed, meaning short frames will become more bandwidth than large frames due to the relation of data to IPG overhead).
15 DMA_CLASS_EN	DMA class enable 0 The DMA controller's channel for the class is not used. **NOTE:** Disabling the DMA controller of a class also requires disabling the class match comparator for the class (see registers RCMRn). When class 1 and class 2 queues are disabled then their frames will be placed in queue 0. 1 Enable the DMA controller to support the corresponding descriptor ring for this class of traffic.
16–31 IDLE_SLOPE	Idle slope 16-bit value to define the per class idle slope setting used by the credit based shaper defining allocated bandwidth for the class. This value is used to calculate the BW (bandwidth) fraction, given by the equation, BW fraction = 1/(1+512/IDLE_SLOPE). Idle slope is restricted to certain values. For values less than 128, idle slope = 2^n, where n = 0, 1, 2, ...6. For values equal to or greater than 128, idle slope = 128×m, where m = 1, 2, 3, ...12. Example 1. BW fraction = 0.20 = 1/(1+(512/128)); therefore idleslope = 128. Example 2. BW fraction = 0.33 = 1/(1+(512/256)); therefore idleslope = 256. Example 3. BW fraction = 0.75 = 1/(1+(512/1536)); therefore idleslope = 1536. **NOTE:** For AVB applications, the BW fraction of class 1 and class 2 combined must not exceed .75.

11.1.5.43 Receive Descriptor Active Register - Ring 1 (ENET_RDAR1)

RDAR1 is a command register, written by the user, to indicate that the receive descriptor ring 1 has been updated, that is, that the driver produced empty receive buffers with the empty bit set.

Address: 30BE_0000h base + 1E0h offset = 30BE_01E0h



ENET_RDAR1 field descriptions

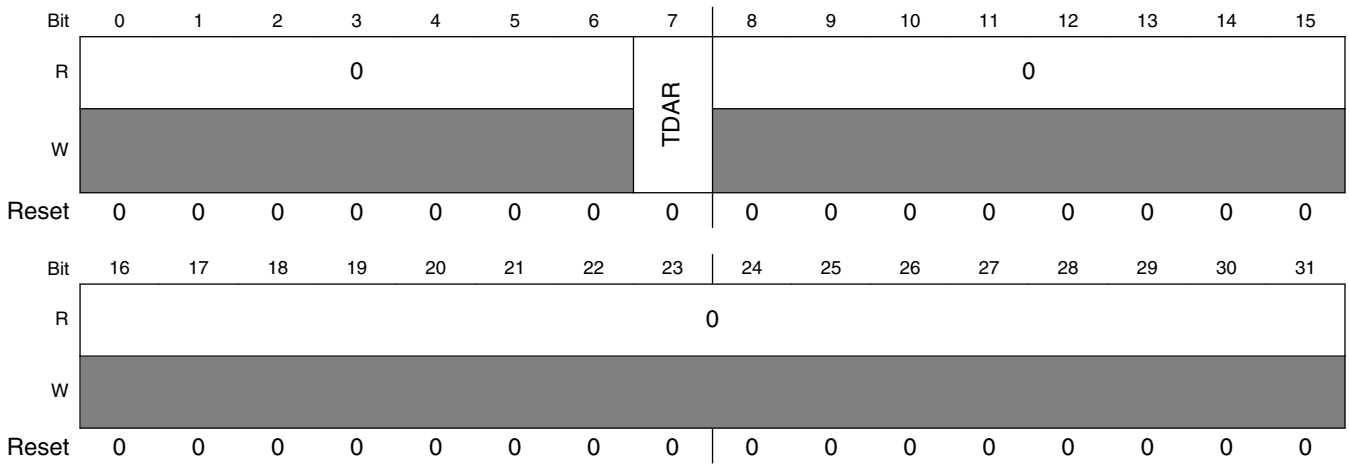
Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 RDAR	Receive Descriptor Active Always set to 1 when this register is written, regardless of the value written. This field is cleared by the MAC device when no additional empty descriptors remain in the receive ring. It is also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.44 Transmit Descriptor Active Register - Ring 1 (ENET_TDAR1)

TDAR1 is a command register that the user writes to indicate that transmit descriptor ring 1 has been updated, that is, that transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor.

The TDAR register is cleared at reset, when ECR[ETHEREN] transitions from set to cleared, or when ECR[RESET] is set.

Address: 30BE_0000h base + 1E4h offset = 30BE_01E4h



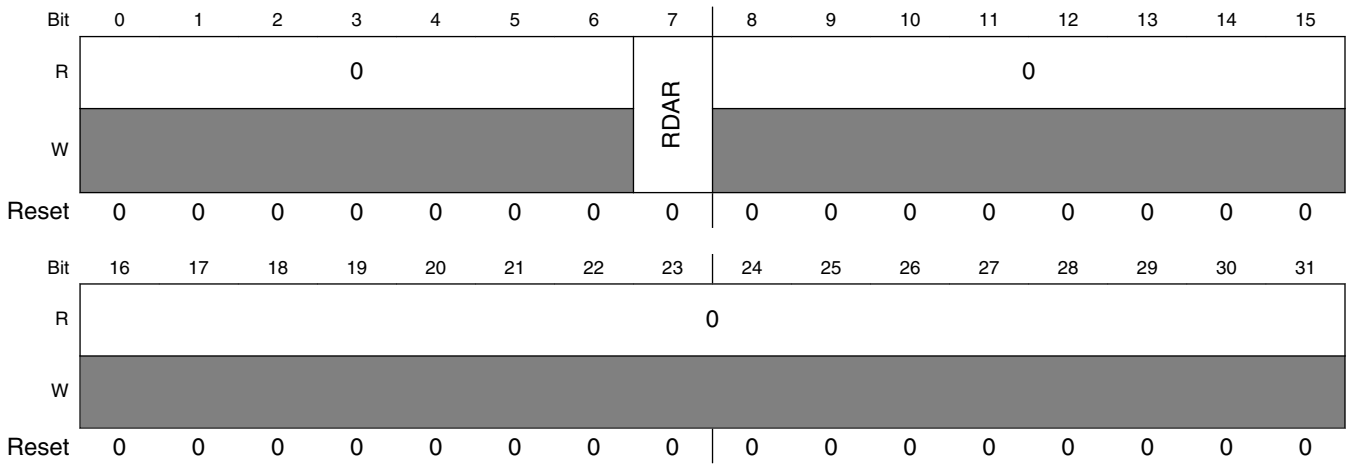
ENET_TDAR1 field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TDAR	Transmit Descriptor Active Always set to 1 when this register is written, regardless of the value written. This bit is cleared by the MAC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.45 Receive Descriptor Active Register - Ring 2 (ENET_RDAR2)

RDAR2 is a command register, written by the user, to indicate that the receive descriptor ring 2 has been updated, that is, that the driver produced empty receive buffers with the empty bit set.

Address: 30BE_0000h base + 1E8h offset = 30BE_01E8h



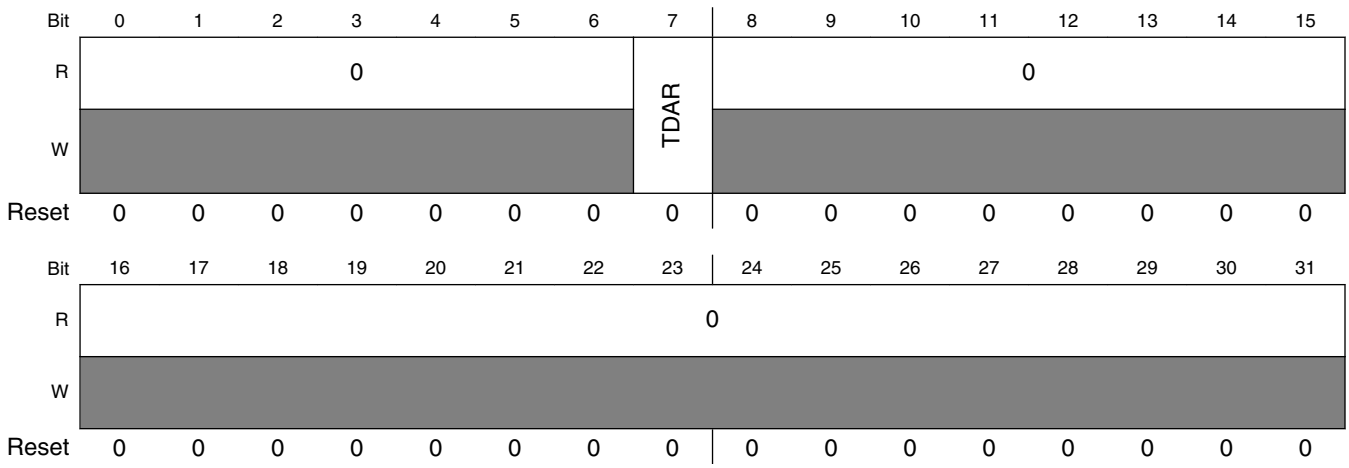
ENET_RDAR2 field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 RDAR	Receive Descriptor Active Always set to 1 when this register is written, regardless of the value written. This field is cleared by the MAC device when no additional empty descriptors remain in the receive ring. It is also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.46 Transmit Descriptor Active Register - Ring 2 (ENET_TDAR2)

TDAR2 is a command register that the user writes to indicate that transmit descriptor ring 2 has been updated, that is, that transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor.

Address: 30BE_0000h base + 1ECh offset = 30BE_01ECh



ENET_TDAR2 field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TDAR	Transmit Descriptor Active Always set to 1 when this register is written, regardless of the value written. This bit is cleared by the MAC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

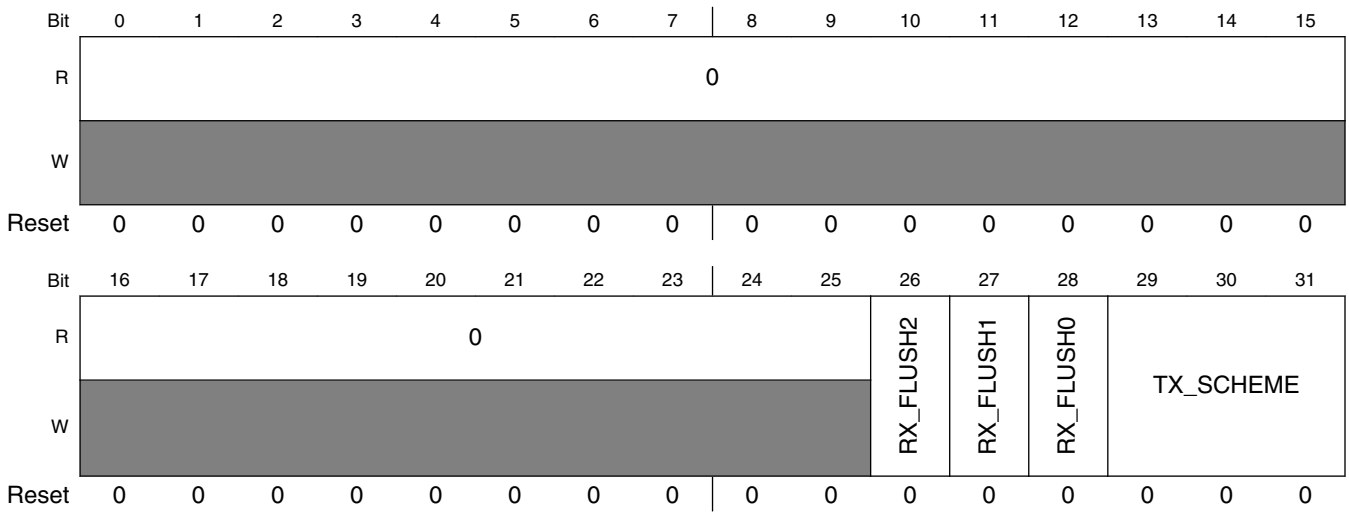
11.1.5.47 QOS Scheme (ENET_QOS)

This register sets the QOS scheme.

NOTE

When both class 1 and class 2 are disabled, RX flushing for these rings must also be disabled.

Address: 30BE_0000h base + 1F0h offset = 30BE_01F0h

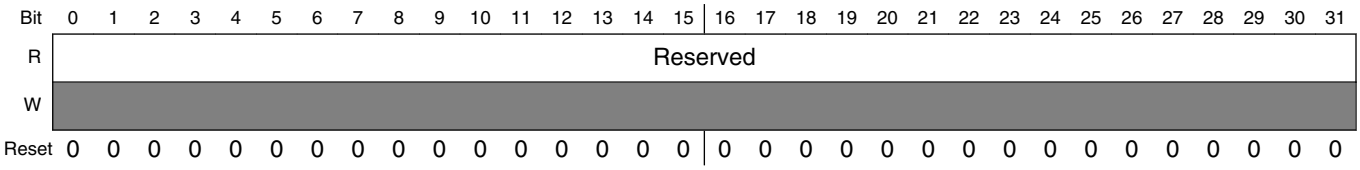


ENET_QOS field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 RX_FLUSH2	RX Flush Ring 2 Enable or disable RX Flush for ring 2. See Receive flushing. 0 Disable 1 Enable
27 RX_FLUSH1	RX Flush Ring 1 Enable or disable RX Flush for ring 1. See Receive flushing. 0 Disable 1 Enable
28 RX_FLUSH0	RX Flush Ring 0 Enable or disable RX Flush for ring 0. See Receive flushing. 0 Disable 1 Enable
29–31 TX_SCHEME	TX scheme configuration Configuration information for DMA to select transmitter queue selection/arbitration scheme. 000 Credit-based scheme 001 Round-robin scheme 010-111 Reserved

11.1.5.48 Reserved Statistic Register (ENET_RMON_T_DROP)

Address: 30BE_0000h base + 200h offset = 30BE_0200h

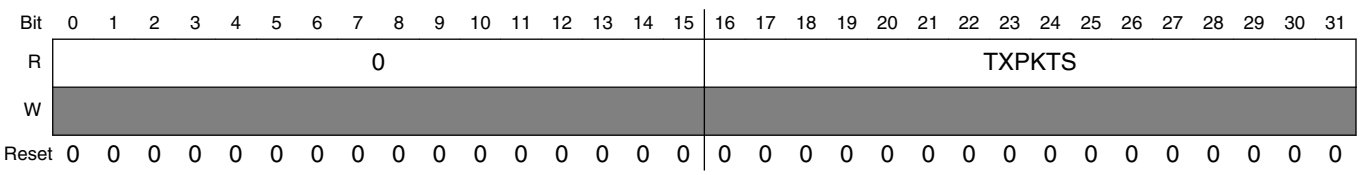


ENET_RMON_T_DROP field descriptions

Field	Description
0–31 Reserved	This read-only field always has the value 0. This field is reserved.

11.1.5.49 Tx Packet Count Statistic Register (ENET_RMON_T_PACKETS)

Address: 30BE_0000h base + 204h offset = 30BE_0204h



ENET_RMON_T_PACKETS field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Packet count Transmit packet count

11.1.5.50 Tx Broadcast Packets Statistic Register (ENET_RMON_T_BC_PKT)

RMON Tx Broadcast Packets

Address: 30BE_0000h base + 208h offset = 30BE_0208h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_BC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Broadcast packets

11.1.5.51 Tx Multicast Packets Statistic Register (ENET_RMON_T_MC_PKT)

Address: 30BE_0000h base + 20Ch offset = 30BE_020Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_MC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Multicast packets

11.1.5.52 Tx Packets with CRC/Align Error Statistic Register (ENET_RMON_T_CRC_ALIGN)

Address: 30BE_0000h base + 210h offset = 30BE_0210h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_CRC_ALIGN field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Packets with CRC/align error

11.1.5.53 Tx Packets Less Than Bytes and Good CRC Statistic Register (ENET_RMON_T_UNDERSIZE)

Address: 30BE_0000h base + 214h offset = 30BE_0214h

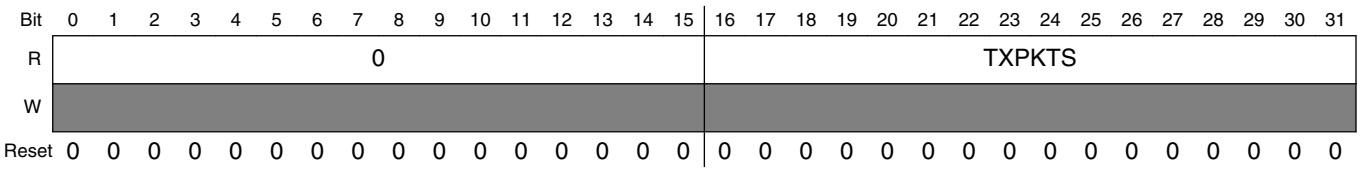
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_UNDERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets less than 64 bytes with good CRC

11.1.5.54 Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (ENET_RMON_T_OVERSIZE)

Address: 30BE_0000h base + 218h offset = 30BE_0218h



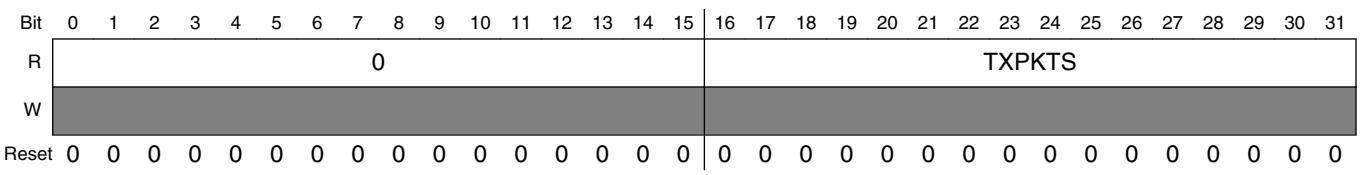
ENET_RMON_T_OVERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets greater than MAX_FL bytes with good CRC

11.1.5.55 Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_T_FRAG)

.

Address: 30BE_0000h base + 21Ch offset = 30BE_021Ch

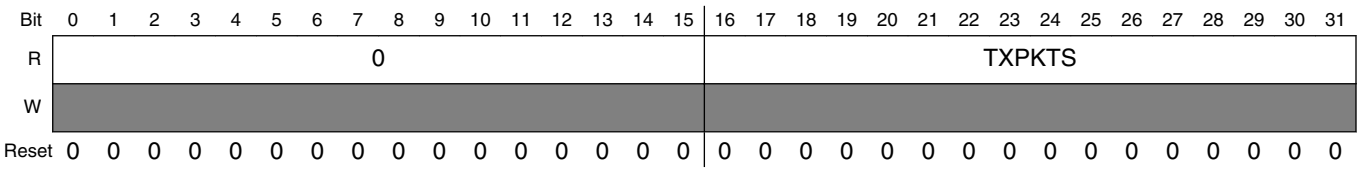


ENET_RMON_T_FRAG field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of packets less than 64 bytes with bad CRC

11.1.5.56 Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (ENET_RMON_T_JAB)

Address: 30BE_0000h base + 220h offset = 30BE_0220h

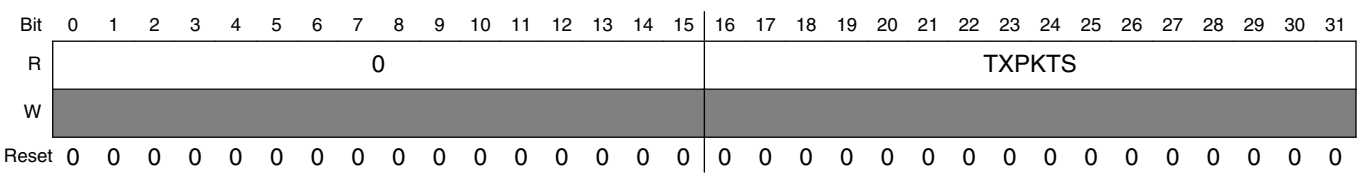


ENET_RMON_T_JAB field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets greater than MAX_FL bytes and bad CRC

11.1.5.57 Tx Collision Count Statistic Register (ENET_RMON_T_COL)

Address: 30BE_0000h base + 224h offset = 30BE_0224h



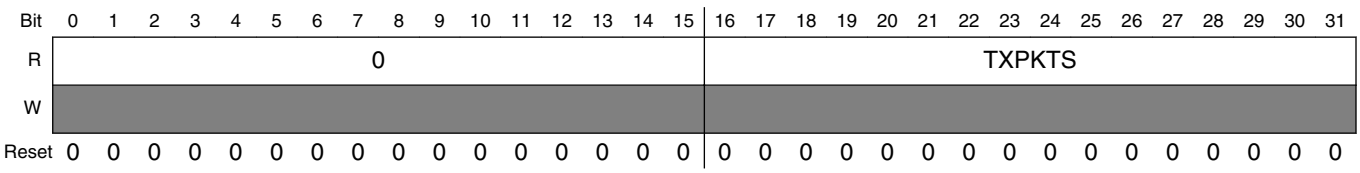
ENET_RMON_T_COL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit collisions

11.1.5.58 Tx 64-Byte Packets Statistic Register (ENET_RMON_T_P64)

.

Address: 30BE_0000h base + 228h offset = 30BE_0228h

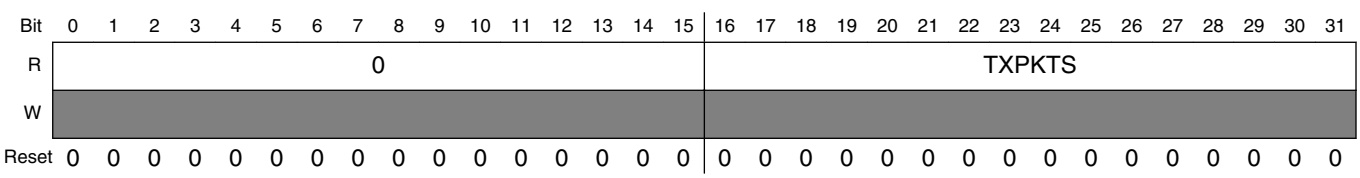


ENET_RMON_T_P64 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 64-byte transmit packets

11.1.5.59 Tx 65- to 127-byte Packets Statistic Register (ENET_RMON_T_P65TO127)

Address: 30BE_0000h base + 22Ch offset = 30BE_022Ch



ENET_RMON_T_P65TO127 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 65- to 127-byte transmit packets

11.1.5.60 Tx 128- to 255-byte Packets Statistic Register (ENET_RMON_T_P128TO255)

Address: 30BE_0000h base + 230h offset = 30BE_0230h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P128TO255 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 128- to 255-byte transmit packets

11.1.5.61 Tx 256- to 511-byte Packets Statistic Register (ENET_RMON_T_P256TO511)

Address: 30BE_0000h base + 234h offset = 30BE_0234h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P256TO511 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 256- to 511-byte transmit packets

11.1.5.62 Tx 512- to 1023-byte Packets Statistic Register (ENET_RMON_T_P512TO1023)

.

Address: 30BE_0000h base + 238h offset = 30BE_0238h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P512TO1023 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 512- to 1023-byte transmit packets

11.1.5.63 Tx 1024- to 2047-byte Packets Statistic Register (ENET_RMON_T_P1024TO2047)

Address: 30BE_0000h base + 23Ch offset = 30BE_023Ch

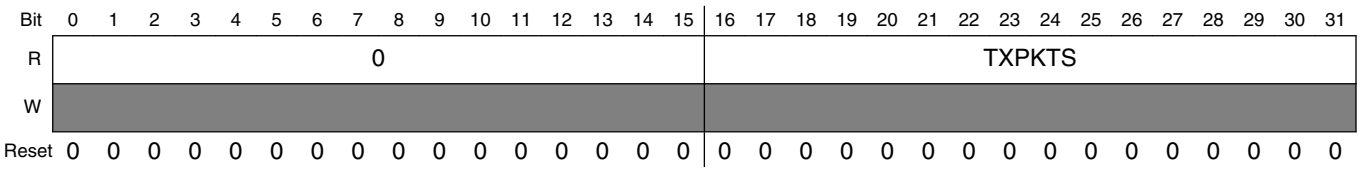
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P1024TO2047 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 1024- to 2047-byte transmit packets

11.1.5.64 Tx Packets Greater Than 2048 Bytes Statistic Register (ENET_RMON_T_P_GTE2048)

Address: 30BE_0000h base + 240h offset = 30BE_0240h

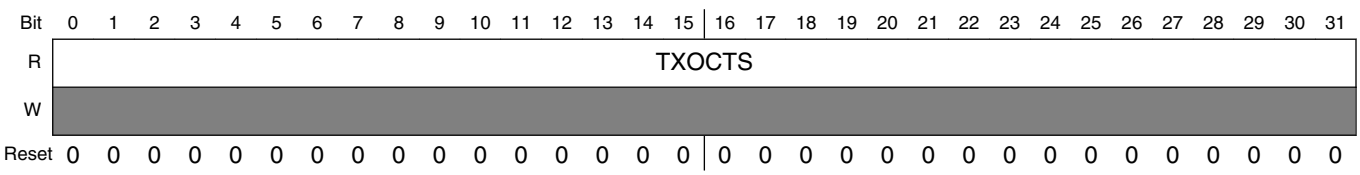


ENET_RMON_T_P_GTE2048 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets greater than 2048 bytes

11.1.5.65 Tx Octets Statistic Register (ENET_RMON_T_OCTETS)

Address: 30BE_0000h base + 244h offset = 30BE_0244h

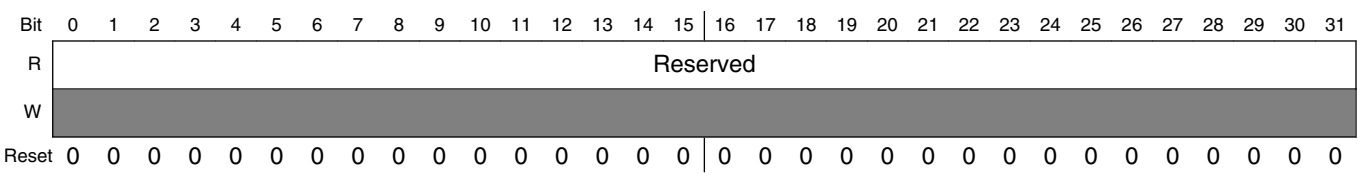


ENET_RMON_T_OCTETS field descriptions

Field	Description
0–31 TXOCTS	Number of transmit octets

11.1.5.66 Reserved Statistic Register (ENET_IEEE_T_DROP)

Address: 30BE_0000h base + 248h offset = 30BE_0248h

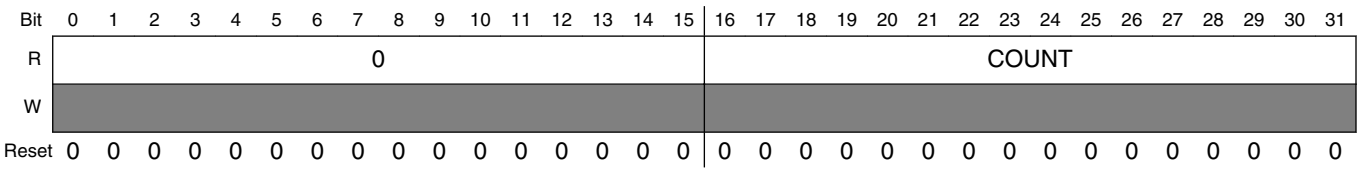


ENET_IEEE_T_DROP field descriptions

Field	Description
0–31 Reserved	This read-only field always has the value 0. This field is reserved.

11.1.5.67 Frames Transmitted OK Statistic Register (ENET_IEEE_T_FRAME_OK)

Address: 30BE_0000h base + 24Ch offset = 30BE_024Ch

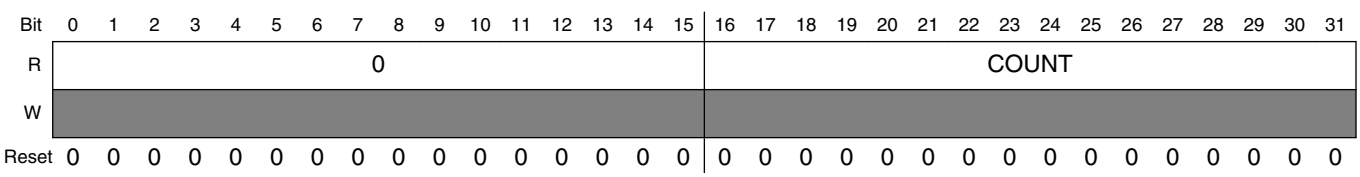


ENET_IEEE_T_FRAME_OK field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted OK **NOTE:** Does not increment for the broadcast frames when broadcast reject is enabled and promiscuous mode is disabled within the receive control register (RCR).

11.1.5.68 Frames Transmitted with Single Collision Statistic Register (ENET_IEEE_T_1COL)

Address: 30BE_0000h base + 250h offset = 30BE_0250h

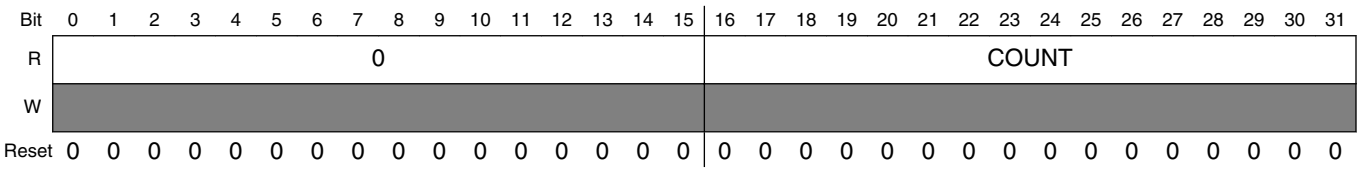


ENET_IEEE_T_1COL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with one collision

11.1.5.69 Frames Transmitted with Multiple Collisions Statistic Register (ENET_IEEE_T_MCOL)

Address: 30BE_0000h base + 254h offset = 30BE_0254h

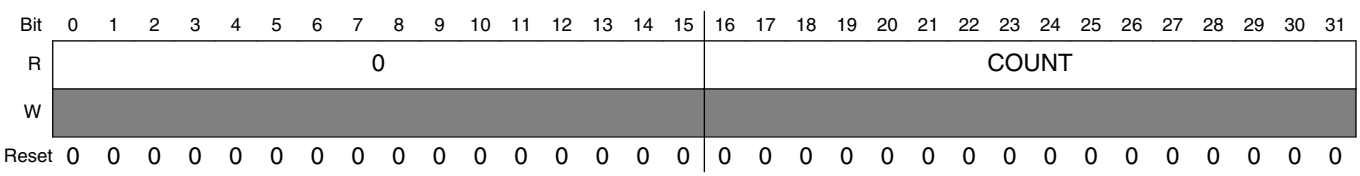


ENET_IEEE_T_MCOL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with multiple collisions

11.1.5.70 Frames Transmitted after Deferral Delay Statistic Register (ENET_IEEE_T_DEF)

Address: 30BE_0000h base + 258h offset = 30BE_0258h



ENET_IEEE_T_DEF field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with deferral delay

11.1.5.71 Frames Transmitted with Late Collision Statistic Register (ENET_IEEE_T_LCOL)

Address: 30BE_0000h base + 25Ch offset = 30BE_025Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_LCOL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with late collision

11.1.5.72 Frames Transmitted with Excessive Collisions Statistic Register (ENET_IEEE_T_EXCOL)

Address: 30BE_0000h base + 260h offset = 30BE_0260h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_EXCOL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with excessive collisions

11.1.5.73 Frames Transmitted with Tx FIFO Underrun Statistic Register (ENET_IEEE_T_MACERR)

Address: 30BE_0000h base + 264h offset = 30BE_0264h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_MACERR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with transmit FIFO underrun

11.1.5.74 Frames Transmitted with Carrier Sense Error Statistic Register (ENET_IEEE_T_CSERR)

Address: 30BE_0000h base + 268h offset = 30BE_0268h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_CSERR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with carrier sense error

11.1.5.75 Reserved Statistic Register (ENET_IEEE_T_SQE)

Address: 30BE_0000h base + 26Ch offset = 30BE_026Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_SQE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	This read-only field is reserved and always has the value 0. **NOTE:** Counter not implemented as no SQE information is available.

11.1.5.76 Flow Control Pause Frames Transmitted Statistic Register (ENET_IEEE_T_FDXFC)

Address: 30BE_0000h base + 270h offset = 30BE_0270h

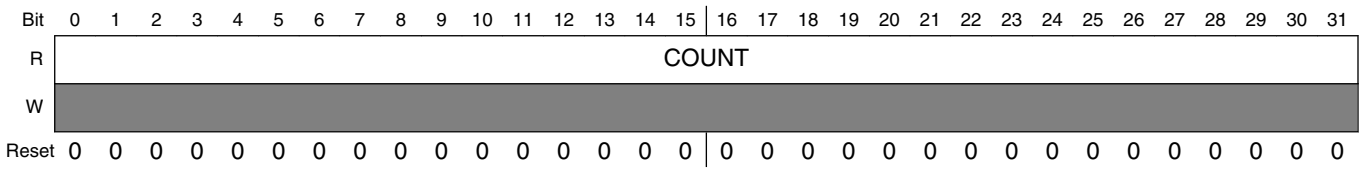
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_FDXFC field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of flow-control pause frames transmitted

11.1.5.77 Octet Count for Frames Transmitted w/o Error Statistic Register (ENET_IEEE_T_OCTETS_OK)

Address: 30BE_0000h base + 274h offset = 30BE_0274h

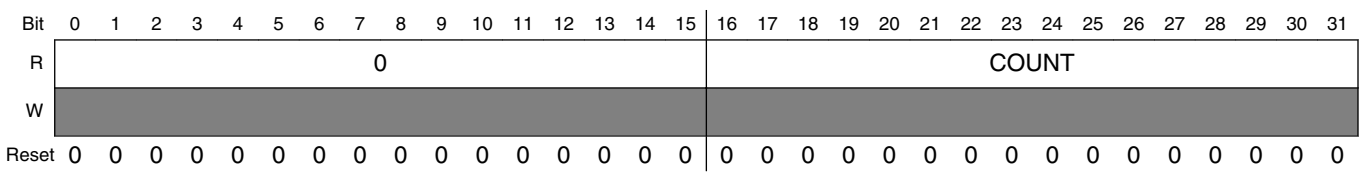


ENET_IEEE_T_OCTETS_OK field descriptions

Field	Description
0–31 COUNT	Octet count for frames transmitted without error **NOTE** Counts total octets (includes header and FCS fields).

11.1.5.78 Rx Packet Count Statistic Register (ENET_RMON_R_PACKETS)

Address: 30BE_0000h base + 284h offset = 30BE_0284h



ENET_RMON_R_PACKETS field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of packets received

11.1.5.79 Rx Broadcast Packets Statistic Register (ENET_RMON_R_BC_PKT)

Address: 30BE_0000h base + 288h offset = 30BE_0288h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_BC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive broadcast packets

11.1.5.80 Rx Multicast Packets Statistic Register (ENET_RMON_R_MC_PKT)

Address: 30BE_0000h base + 28Ch offset = 30BE_028Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_MC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive multicast packets

11.1.5.81 Rx Packets with CRC/Align Error Statistic Register (ENET_RMON_R_CRC_ALIGN)

Address: 30BE_0000h base + 290h offset = 30BE_0290h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_CRC_ALIGN field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets with CRC or align error

11.1.5.82 Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (ENET_RMON_R_UNDERSIZE)

Address: 30BE_0000h base + 294h offset = 30BE_0294h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_UNDERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets with less than 64 bytes and good CRC

11.1.5.83 Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (ENET_RMON_R_OVERSIZE)

Address: 30BE_0000h base + 298h offset = 30BE_0298h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_OVERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets greater than MAX_FL and good CRC

11.1.5.84 Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_R_FRAG)

Address: 30BE_0000h base + 29Ch offset = 30BE_029Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_FRAG field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets with less than 64 bytes and bad CRC

11.1.5.85 Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (ENET_RMON_R_JAB)

Address: 30BE_0000h base + 2A0h offset = 30BE_02A0h

Bit	0–15	16–31
R	0	COUNT
W		
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ENET_RMON_R_JAB field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets greater than MAX_FL and bad CRC

11.1.5.86 Reserved Statistic Register (ENET_RMON_R_RESVD_0)

Address: 30BE_0000h base + 2A4h offset = 30BE_02A4h

Bit	0–15	16–31
R	0	0
W		
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ENET_RMON_R_RESVD_0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.1.5.87 Rx 64-Byte Packets Statistic Register (ENET_RMON_R_P64)

Address: 30BE_0000h base + 2A8h offset = 30BE_02A8h

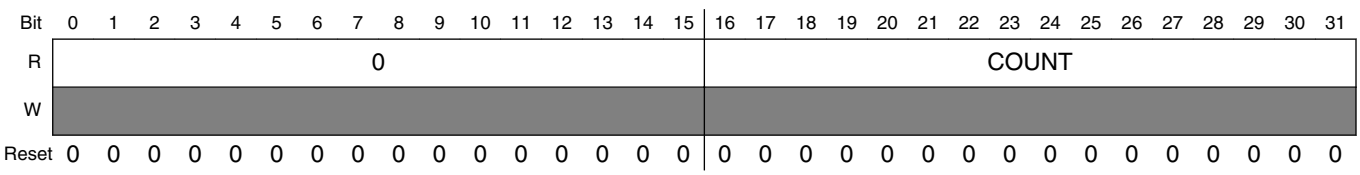
Bit	0–15	16–31
R	0	COUNT
W		
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ENET_RMON_R_P64 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 64-byte receive packets

11.1.5.88 Rx 65- to 127-Byte Packets Statistic Register (ENET_RMON_R_P65TO127)

Address: 30BE_0000h base + 2ACh offset = 30BE_02ACh

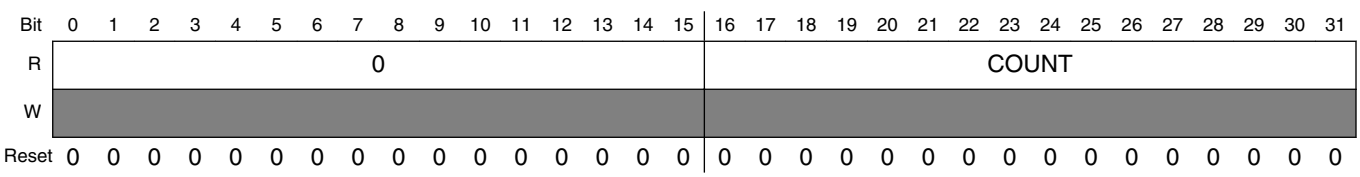


ENET_RMON_R_P65TO127 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 65- to 127-byte recieve packets

11.1.5.89 Rx 128- to 255-Byte Packets Statistic Register (ENET_RMON_R_P128TO255)

Address: 30BE_0000h base + 2B0h offset = 30BE_02B0h

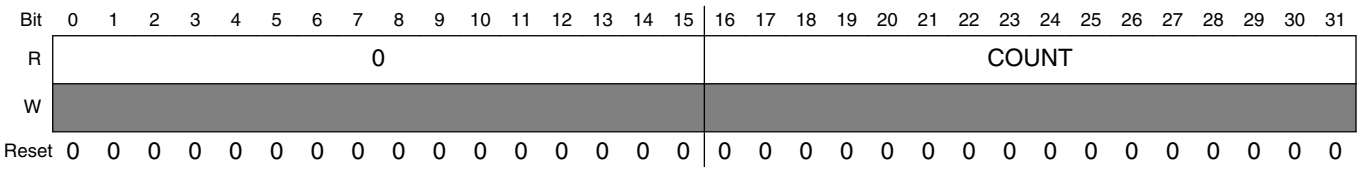


ENET_RMON_R_P128TO255 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 128- to 255-byte recieve packets

11.1.5.90 Rx 256- to 511-Byte Packets Statistic Register (ENET_RMON_R_P256TO511)

Address: 30BE_0000h base + 2B4h offset = 30BE_02B4h

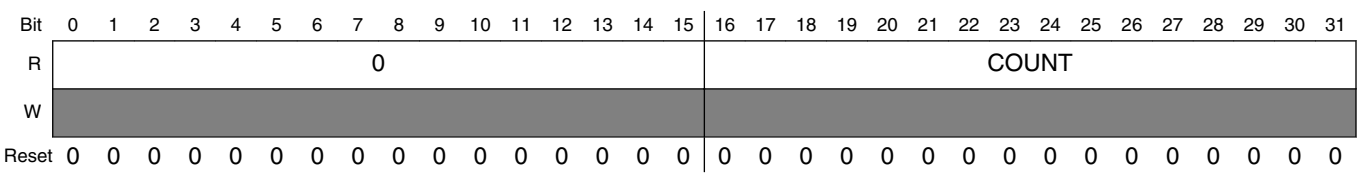


ENET_RMON_R_P256TO511 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 256- to 511-byte recieve packets

11.1.5.91 Rx 512- to 1023-Byte Packets Statistic Register (ENET_RMON_R_P512TO1023)

Address: 30BE_0000h base + 2B8h offset = 30BE_02B8h



ENET_RMON_R_P512TO1023 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 512- to 1023-byte recieve packets

11.1.5.92 Rx 1024- to 2047-Byte Packets Statistic Register (ENET_RMON_R_P1024TO2047)

Address: 30BE_0000h base + 2BCh offset = 30BE_02BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P1024TO2047 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 1024- to 2047-byte recieve packets

11.1.5.93 Rx Packets Greater than 2048 Bytes Statistic Register (ENET_RMON_R_P_GTE2048)

Address: 30BE_0000h base + 2C0h offset = 30BE_02C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P_GTE2048 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of greater-than-2048-byte recieve packets

11.1.5.94 Rx Octets Statistic Register (ENET_RMON_R_OCTETS)

Address: 30BE_0000h base + 2C4h offset = 30BE_02C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_OCTETS field descriptions

Field	Description
0–31 COUNT	Number of receive octets

11.1.5.95 Frames not Counted Correctly Statistic Register (ENET_IEEE_R_DROP)

Counter increments if a frame with invalid or missing SFD character is detected and has been dropped. None of the other counters increments if this counter increments.

Address: 30BE_0000h base + 2C8h offset = 30BE_02C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_DROP field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Frame count

11.1.5.96 Frames Received OK Statistic Register (ENET_IEEE_R_FRAME_OK)

Address: 30BE_0000h base + 2CCh offset = 30BE_02CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_FRAME_OK field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames received OK

11.1.5.97 Frames Received with CRC Error Statistic Register (ENET_IEEE_R_CRC)

Address: 30BE_0000h base + 2D0h offset = 30BE_02D0h

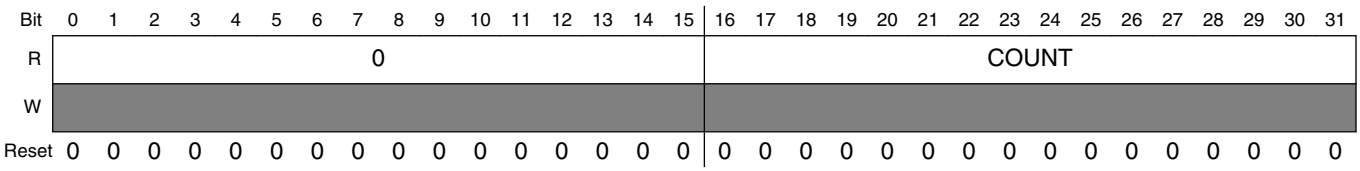
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_CRC field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames received with CRC error

11.1.5.98 Frames Received with Alignment Error Statistic Register (ENET_IEEE_R_ALIGN)

Address: 30BE_0000h base + 2D4h offset = 30BE_02D4h

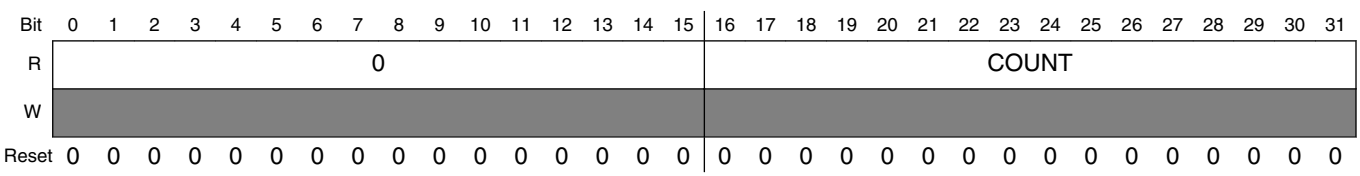


ENET_IEEE_R_ALIGN field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames received with alignment error

11.1.5.99 Receive FIFO Overflow Count Statistic Register (ENET_IEEE_R_MACERR)

Address: 30BE_0000h base + 2D8h offset = 30BE_02D8h

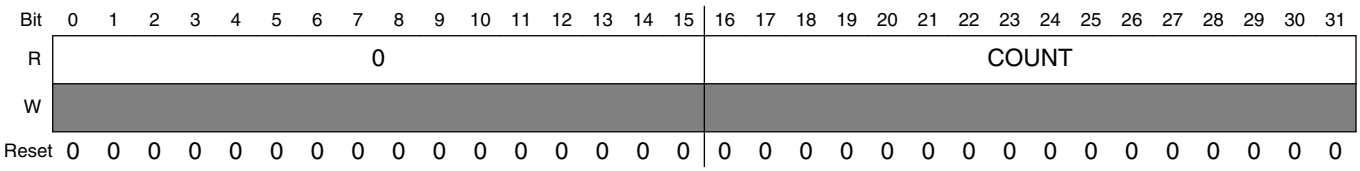


ENET_IEEE_R_MACERR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Receive FIFO overflow count

11.1.5.100 Flow Control Pause Frames Received Statistic Register (ENET_IEEE_R_FDXFC)

Address: 30BE_0000h base + 2DCh offset = 30BE_02DCh

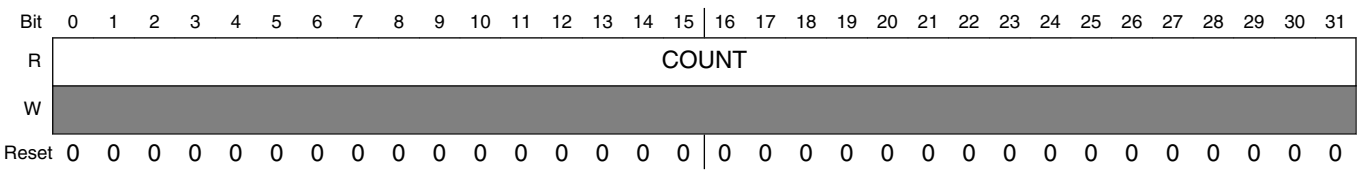


ENET_IEEE_R_FDXFC field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of flow-control pause frames received

11.1.5.101 Octet Count for Frames Received without Error Statistic Register (ENET_IEEE_R_OCTETS_OK)

Address: 30BE_0000h base + 2E0h offset = 30BE_02E0h



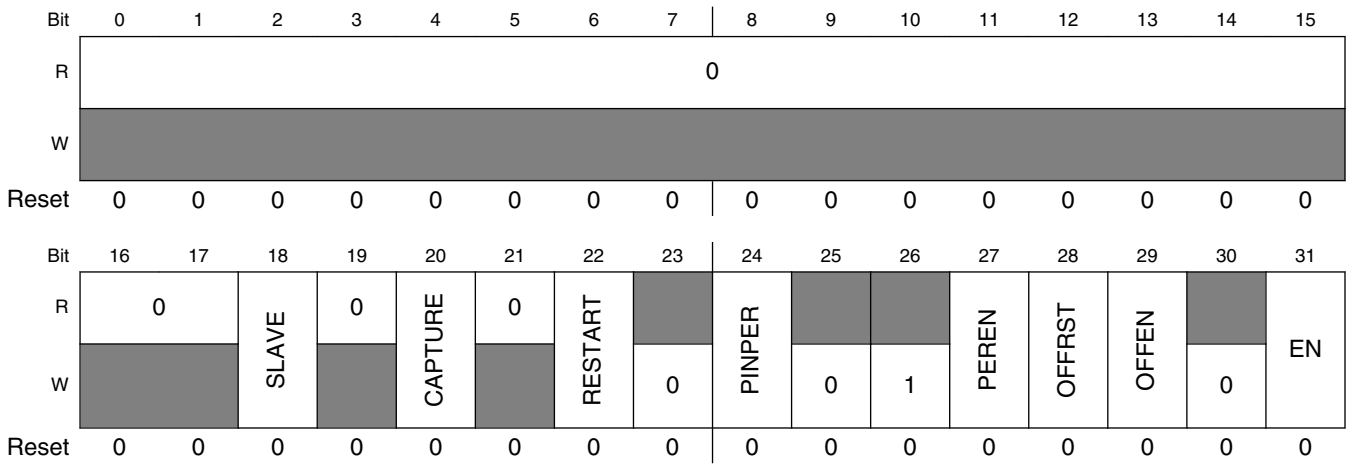
ENET_IEEE_R_OCTETS_OK field descriptions

Field	Description
0–31 COUNT	Number of octets for frames received without error **NOTE:** Counts total octets (includes header and FCS fields). Does not increment for the broadcast frames when broadcast reject is enabled and promiscuous mode is disabled within the receive control register (RCR).

11.1.5.102 Adjustable Timer Control Register (ENET_ATCR)

ATCR command fields can trigger the corresponding events directly. It is not necessary to preserve any of the configuration fields when a command field is set in the register, that is, no read-modify-write is required.

Address: 30BE_0000h base + 400h offset = 30BE_0400h



ENET_ATCR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 SLAVE	Enable Timer Slave Mode 0 The timer is active and all configuration fields in this register are relevant. 1 The internal timer is disabled and the externally provided timer value is used. All other fields, except CAPTURE, in this register have no effect. CAPTURE can still be used to capture the current timer value.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 CAPTURE	Capture Timer Value When this field is set, all other fields are ignored during a write. This field automatically clears to 0 after the command completes. **NOTE:** To ensure that the correct time value is read from the ATVR register, a minimum amount of time must elapse from issuing this command to reading the ATVR register. This minimum time is defined by the greater of either six register clock cycles or six 1588/timestamp clock cycles. 0 No effect. 1 The current time is captured and can be read from the ATVR register.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ENET_ATCR field descriptions (continued)

Field	Description
22 RESTART	Reset Timer Resets the timer to zero. This has no effect on the counter enable. If the counter is enabled when this field is set, the timer is reset to zero and starts counting from there. When set, all other fields are ignored during a write. This field automatically clears to 0 after the command completes. **NOTE:** The Reset Timer command requires at least 6 clock cycles of either the register clock or the 1588/timestamp clock, whichever is greater, to complete.
23 Reserved	This field is reserved.
24 PINPER	Enables event signal output assertion on period event. **NOTE:** Not all devices contain the event signal output. See the chip configuration details. 0 Disable. 1 Enable.
25 Reserved	This field is reserved.
26 Reserved	This field is reserved. **NOTE:** This field must be written always with one.
27 PEREN	Enable Periodical Event 0 Disable. 1 A period event interrupt can be generated (EIR[TS_TIMER]) and the event signal output is asserted when the timer wraps around according to the periodic setting ATPER. The timer period value must be set before setting this bit. **NOTE:** Not all devices contain the event signal output. See the chip configuration details.
28 OFFRST	Reset Timer On Offset Event 0 The timer is not affected and no action occurs, besides clearing OFFEN, when the offset is reached. 1 If OFFEN is set, the timer resets to zero when the offset setting is reached. The offset event does not cause a timer interrupt.
29 OFFEN	Enable One-Shot Offset Event 0 Disable. 1 The timer can be reset to zero when the given offset time is reached (offset event). The field is cleared when the offset event is reached, so no further event occurs until the field is set again. The timer offset value must be set before setting this field.
30 Reserved	This field is reserved.
31 EN	Enable Timer 0 The timer stops at the current value. 1 The timer starts incrementing.

11.1.5.103 Timer Value Register (ENET_ATVR)

Address: 30BE_0000h base + 404h offset = 30BE_0404h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ATIME																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATVR field descriptions

Field	Description
0–31 ATIME	A write sets the timer. A read returns the last captured value. To read the current value, issue a capture command (i.e., set ATCR[CAPTURE]) prior to reading this register.

11.1.5.104 Timer Offset Register (ENET_ATOFF)

Address: 30BE_0000h base + 408h offset = 30BE_0408h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	OFFSET																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATOFF field descriptions

Field	Description
0–31 OFFSET	Offset value for one-shot event generation. When the timer reaches the value, an event can be generated to reset the counter. If the increment value in ATINC is given in true nanoseconds, this value is also given in true nanoseconds.

11.1.5.105 Timer Period Register (ENET_ATPER)

Address: 30BE_0000h base + 40Ch offset = 30BE_040Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PERIOD																															
Reset	0	0	1	1	1	0	1	1	1	0	0	1	1	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0

ENET_ATPER field descriptions

Field	Description
0–31 PERIOD	Value for generating periodic events. Each instance the timer reaches this value, the period event occurs and the timer restarts. If the increment value in ATINC is given in true nanoseconds, this value is also given in true nanoseconds. The value should be initialized to 1,000,000,000 (1×10^9) to represent a timer wrap around of one second. The increment value set in ATINC should be set to the true nanoseconds of the period of clock ts_clk, hence implementing a true 1 second counter.

ENET_ATPER field descriptions (continued)

Field	Description
	NOTE: The value of PERIOD has the following constraint: 2^{32} – ENET_ATINC[INC_COR] – 3×ENET_ATINC[INC] ≥ PERIOD > 0.

11.1.5.106 Timer Correction Register (ENET_ATCOR)

Address: 30BE_0000h base + 410h offset = 30BE_0410h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	COR														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATCOR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–31 COR	Correction Counter Wrap-Around Value Defines after how many timer clock cycles (ts_clk) the correction counter should be reset and trigger a correction increment on the timer. The amount of correction is defined in ATINC[INC_CORR]. A value of 0 disables the correction counter and no corrections occur. **NOTE:** This value is given in clock cycles, not in nanoseconds as all other values.

11.1.5.107 Time-Stamping Clock Period Register (ENET_ATINC)

Address: 30BE_0000h base + 414h offset = 30BE_0414h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	INC_CORR							0	INC						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATINC field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–23 INC_CORR	Correction Increment Value This value is added every time the correction timer expires (every clock cycle given in ATCOR). A value less than INC slows down the timer. A value greater than INC speeds up the timer.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 INC	Clock Period Of The Timestamping Clock (ts_clk) In Nanoseconds The timer increments by this amount each clock cycle. For example, set to 10 for 100 MHz, 8 for 125 MHz, 5 for 200 MHz. **NOTE:** For highest precision, use a value that is an integer fraction of the period set in ATPER.

11.1.5.108 Timestamp of Last Transmitted Frame (ENET_ATSTMP)

Address: 30BE_0000h base + 418h offset = 30BE_0418h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TIMESTAMP																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATSTMP field descriptions

Field	Description
0–31 TIMESTAMP	Timestamp of the last frame transmitted by the core that had TxBD[TS] set . This register is only valid when EIR[TS_AVAIL] is set.

11.1.5.109 Timer Global Status Register (ENET_TGSR)

Address: 30BE_0000h base + 604h offset = 30BE_0604h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												TF3	TF2	TF1	TF0
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TGSR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 TF3	Copy Of Timer Flag For Channel 3 0 Timer Flag for Channel 3 is clear 1 Timer Flag for Channel 3 is set
29 TF2	Copy Of Timer Flag For Channel 2 0 Timer Flag for Channel 2 is clear 1 Timer Flag for Channel 2 is set
30 TF1	Copy Of Timer Flag For Channel 1 0 Timer Flag for Channel 1 is clear 1 Timer Flag for Channel 1 is set
31 TF0	Copy Of Timer Flag For Channel 0 0 Timer Flag for Channel 0 is clear 1 Timer Flag for Channel 0 is set

11.1.5.110 Timer Control Status Register (ENET_TCSR*n*)

Address: 30BE_0000h base + 608h offset + (8d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								TF	TIE	TMODE				0	TDRE
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TCSR*n* field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 TF	Timer Flag Sets when input capture or output compare occurs. This flag is double buffered between the module clock and 1588 clock domains. When this field is 1, it can be cleared to 0 by writing 1 to it. 0 Input Capture or Output Compare has not occurred. 1 Input Capture or Output Compare has occurred.

Table continues on the next page...

ENET_TCSR*n* field descriptions (continued)

Field	Description
25 TIE	Timer Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
26–29 TMODE	Timer Mode Updating the Timer Mode field takes a few cycles to register because it is synchronized to the 1588 clock. The version of Timer Mode returned on a read is from the 1588 clock domain. When changing Timer Mode, always disable the channel and read this register to verify the channel is disabled first. 0000 Timer Channel is disabled. 0001 Timer Channel is configured for Input Capture on rising edge. 0010 Timer Channel is configured for Input Capture on falling edge. 0011 Timer Channel is configured for Input Capture on both edges. 0100 Timer Channel is configured for Output Compare - software only. 0101 Timer Channel is configured for Output Compare - toggle output on compare. 0110 Timer Channel is configured for Output Compare - clear output on compare. 0111 Timer Channel is configured for Output Compare - set output on compare. 1000 Reserved 1010 Timer Channel is configured for Output Compare - clear output on compare, set output on overflow. 10X1 Timer Channel is configured for Output Compare - set output on compare, clear output on overflow. 110X Reserved 1110 Timer Channel is configured for Output Compare - pulse output low on compare for one 1588-clock cycle. 1111 Timer Channel is configured for Output Compare - pulse output high on compare for one 1588-clock cycle.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TDRE	Timer DMA Request Enable 0 DMA request is disabled 1 DMA request is enabled

11.1.5.111 Timer Compare Capture Register (ENET_TCCR*n*)

Address: 30BE_0000h base + 60Ch offset + (8d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TCC																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TCCR*n* field descriptions

Field	Description
0–31 TCC	Timer Capture Compare

ENET_TCCR*n* field descriptions (continued)

Field	Description
	This register is double buffered between the module clock and 1588 clock domains. When configured for compare, the 1588 clock domain updates with the value in the module clock domain whenever the Timer Channel is first enabled and on each subsequent compare. Write to this register with the first compare value before enabling the Timer Channel. When the Timer Channel is enabled, write the second compare value either immediately, or at least before the first compare occurs. After each compare, write the next compare value before the previous compare occurs and before clearing the Timer Flag. The compare occurs one 1588 clock cycle after the IEEE 1588 Counter increments past the compare value in the 1588 clock domain. If the compare value is less than the value of the 1588 Counter when the Timer Channel is first enabled, then the compare does not occur until following the next overflow of the 1588 Counter. If the compare value is greater than the IEEE 1588 Counter when the 1588 Counter overflows, or the compare value is less than the value of the IEEE 1588 Counter after the overflow, then the compare occurs one 1588 clock cycle following the overflow. When configured for capture, the value of the IEEE 1588 Counter is captured into the 1588 clock domain and then updated into the module clock domain, provided the Timer Flag is clear. Always read the capture value before clearing the Timer Flag.

11.1.6 Functional description

This section provides a complete functional description of the MAC-NET core.

11.1.6.1 Ethernet MAC frame formats

The IEEE 802.3 standard defines the Ethernet frame format as follows:

- Minimum length of 64 bytes
- Maximum length of 1518 bytes excluding the preamble and the start frame delimiter (SFD) bytes

An Ethernet frame consists of the following fields:

- Seven bytes preamble
- Start frame delimiter (SFD)
- Two address fields
- Length or type field
- Data field

- Frame check sequence (CRC value)
- Extension field is defined only for Gigabit Ethernet half-duplex implementations and is not supported by the MAC core

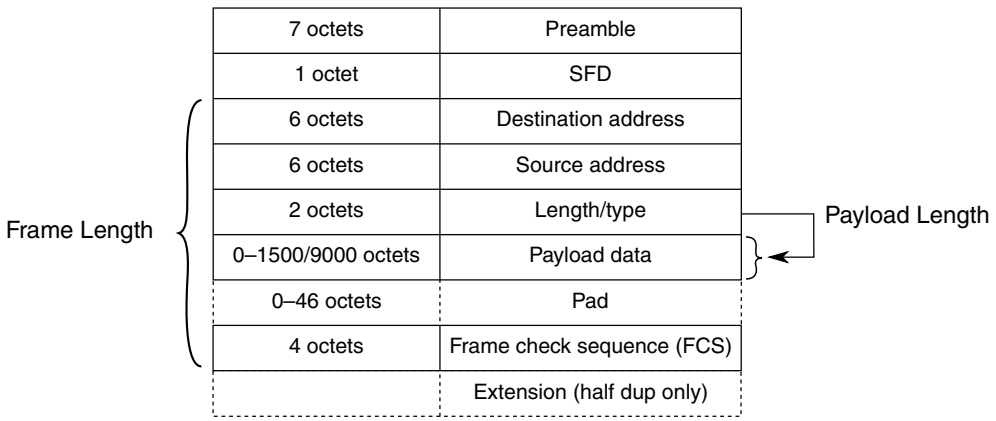


Figure 11-2. MAC frame format overview

Optionally, MAC frames can be VLAN-tagged with an additional four-byte field inserted between the MAC source address and the type/length field. VLAN tagging is defined by the IEEE P802.1q specification. VLAN-tagged frames have a maximum length of 1522 bytes, excluding the preamble and the SFD bytes.

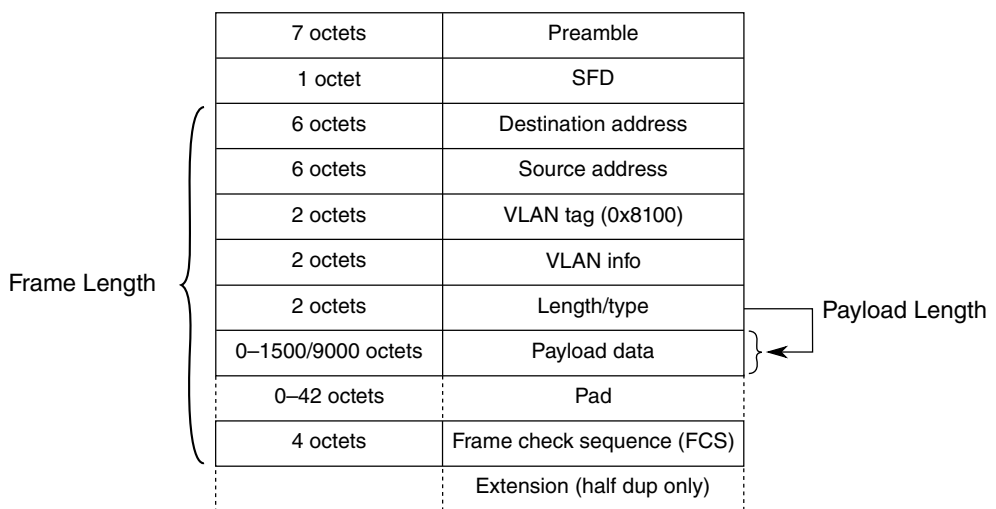


Figure 11-3. VLAN-tagged MAC frame format overview

Table 11-5. MAC frame definition

Term	Description
Frame length	Defines the length, in octets, of the complete frame without preamble and SFD. A frame has a valid length if it contains at least 64 octets and does not exceed the programmed maximum length.
Payload length	The length/type field indicates the length of the frame's payload section. The most significant byte is sent/received first.

Table continues on the next page...

Table 11-5. MAC frame definition (continued)

Term	Description
	• If the length/type field is set to a value less than 46, the payload is padded so that the minimum frame length requirement (64 bytes) is met. For VLAN-tagged frames, a value less than 42 indicates a padded frame. • If the length/type field is set to a value larger than the programmed frame maximum length (e.g. 1518) it is interpreted as a type field.
Destination and source address	48-bit MAC addresses. The least significant byte is sent/received first and the first two least significant bits of the MAC address distinguish MAC frames, as detailed in MAC address check.

Note

Although the IEEE specification defines a maximum frame length, the MAC core provides the flexibility to program any value for the frame maximum length.

11.1.6.1.1 Pause Frames

The receiving device generates a pause frame to indicate a congestion to the emitting device, which should stop sending data.

Pause frames are indicated by the length/type set to 0x8808. The two first bytes of a pause frame following the type, defines a 16-bit opcode field set to 0x0001 always. A 16-bit pause quanta is defined in the frame payload bytes 2 (P1) and 3 (P2) as defined in the following table. The P1 pause quanta byte is the most significant.

Table 11-6. Pause Frame Format (Values in Hex)

1	2	3	4	5	6	7	8	9	10	11	12	13	14
55	55	55	55	55	55	55	D5	01	80	C2	00	00	01
Preamble							SFD	Multicast Destination Address					
15	16	17	18	19	20	21	22	23	24	25	26	27 –68	
00	00	00	00	00	00	88	08	00	01	hi	lo	00	
Source Address						Type		Opcode		P1	P2	pad (42)	
69	70	71	72										
26	6B	AE	0A										
CRC-32													

There is no payload length field found within a pause frame and a pause frame is always padded with 42 bytes (0x00).

If a pause frame with a pause value greater than zero (XOFF condition) is received, the MAC stops transmitting data as soon the current frame transfer is completed. The MAC stops transmitting data for the value defined in pause quanta. One pause quanta fraction refers to 512 bit times.

If a pause frame with a pause value of zero (XON condition) is received, the transmitter is allowed to send data immediately (see Full-duplex flow control operation for details).

11.1.6.1.2 Magic packets

A magic packet is a unicast, multicast, or broadcast packet, which carries a defined sequence in the payload section.

Magic packets are received and inspected only under specific conditions as described in Magic packet detection.

The defined sequence to decode a magic packet is formed with a synchronization stream which consists of six consecutive 0xFF bytes, and is followed by sequence of sixteen consecutive unicast MAC addresses of the node to be awakened.

This sequence can be located anywhere in the magic packet payload. The magic packet is formed with a standard Ethernet header, optional padding, and CRC.

11.1.6.2 IP and higher layers frame format

The following sections use the term datagram to describe the protocol specific data unit that is found within the payload section of its container entity.

For example, an IP datagram specifies the payload section of an Ethernet frame. A TCP datagram specifies the payload section within an IP datagram.

11.1.6.2.1 Ethernet types

IP datagrams are carried in the payload section of an Ethernet frame. The Ethernet frame type/length field discriminates several datagram types.

The following table lists the types of interest:

Table 11-7. Ethernet type value examples

Type	Description
0x8100	VLAN-tagged frame. The actual type is found 4 octets later in the frame.
0x0800	IPv4
0x0806	ARP
0x86DD	IPv6

11.1.6.2.2 IPv4 datagram format

The following figure shows the IP Version 4 (IPv4) header, which is located at the beginning of an IP datagram. It is organized in 32-bit words. The first byte sent/received is the leftmost byte of the first word (in other words, version/IHL field).

The IP header can contain further options, which are always padded if necessary to guarantee the payload following the header is aligned to a 32-bit boundary.

The IP header is immediately followed by the payload, which can contain further protocol headers (for example, TCP or UDP, as indicated by the protocol field value). The complete IP datagram is transported in the payload section of an Ethernet frame.

Table 11-8. IPv4 header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Version	IHL	TOS		Length			
Fragment ID				Flags	Fragment offset		
TTL		Protocol		Header checksum			
Source address							
Destination address							
Options							

Table 11-9. IPv4 header fields

Field name	Description
Version	4-bit IP version information. 0x4 for IPv4 frames.
IHL	4-bit Internet header length information. Determines number of 32-bit words found within the IP header. If no options are present, the default value is 0x5.
TOS	Type of service/DiffServ field.
Length	Total length of the datagram in bytes, including all octets of header and payload.
Fragment ID, flags, fragment offset	Fields used for IP fragmentation.
TTL	Time-to-live. In effect, is decremented at each router arrival. If zero, datagram must be discarded.
Protocol	Identifier of protocol that follows in the datagram.
Header checksum	Checksum of IP header. For computational purposes, this field's value is zero.
Source address	Source IP address.
Destination address	Destination IP address.

11.1.6.2.3 IPv6 datagram format

The following figure shows the IP version 6 (IPv6) header, which is located at the beginning of an IP datagram. It is organized in 32-bit words and has a fixed length of ten words (40 bytes). The next header field identifies the type of the header that follows the IPv6 header. It is defined similar to the protocol identifier within IPv4, with new definitions for identifying extension headers. These headers can be inserted between the IPv6 header and the protocol header, which will shift the protocol header accordingly.The accelerator currently only supports IPv6 without extension headers (in other words, the next header specifies TCP, UDP, or IMCP).

The first byte sent/received is the leftmost byte of the first word (in other words, version/ traffic class fields).

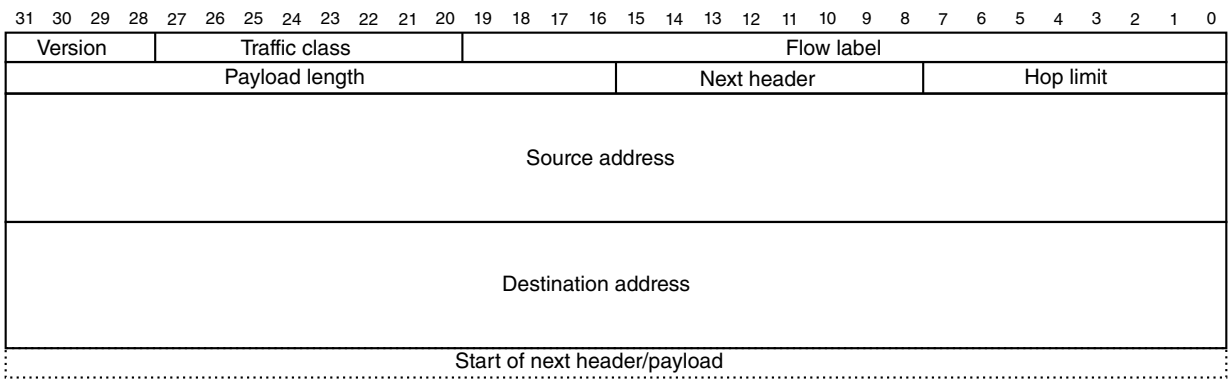


Figure 11-4. IPv6 header format

Table 11-10. IPv6 header fields

Field name	Description
Version	4-bit IP version information. 0x6 for all IPv6 frames.
Traffic class	8-bit field defining the traffic class.
Flow label	20-bit flow label identifying frames of the same flow.
Payload length	16-bit length of the datagram payload in bytes. It includes all octets following the IPv6 header.
Next header	Identifies the header that follows the IPv6 header. This can be the protocol header or any IPv6 defined extension header.
Hop limit	Hop counter, decremented by one by each station that forwards the frame. If hop limit is 0 the frame must be discarded.
Source address	128-bit IPv6 source address.
Destination address	128-bit IPv6 destination address.

11.1.6.2.4 Internet Control Message Protocol (ICMP) datagram format

An internet control message protocol (ICMP) is found following the IP header, if the protocol identifier is 1. The ICMP datagram has a four-octet header followed by additional message data.

Table 11-11. ICMP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Type		Code		Checksum			
ICMP message data							

Table 11-12. IP header fields

Field name	Description
Type	8-bit type information
Code	8-bit code that is related to the message type
Checksum	16-bit one's complement checksum over the complete ICMP datagram

11.1.6.2.5 User Datagram Protocol (UDP) datagram format

A user datagram protocol header is found after the IP header, when the protocol identifier is 17.

The payload of the datagram is after the UDP header. The header byte order follows the conventions given for the IP header above.

Table 11-13. UDP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source port				Destination port			
Length				Checksum			

Table 11-14. UDP header fields

Field name	Description
Source port	Source application port
Destination port	Destination application port
Length	Length of user data which immediately follows the header, including the UDP header (that is, minimum value is 8)
Checksum	Checksum over the complete datagram and some IP header information

11.1.6.2.6 TCP datagram format

A TCP header is found following the IP header, when the protocol identifier has a value of 6.

The TCP payload immediately follows the TCP header.

Table 11-15. TCP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source port				Destination port			
Sequence number							
Acknowledgement number							
Offset	Reserved		Flags	Window			
Checksum				Urgent pointer			
Options							

Table 11-16. TCP header fields

Field name	Description
Source port	Source application port
Destination port	Destination application port
Sequence number	Transmit sequence number
Ack. number	Receive sequence number
Offset	Data offset, which is number of 32-bit words within TCP header — if no options selected, defaults to value of 5
Flags	URG, ACK, PSH, RST, SYN, FIN flags
Window	TCP receive window size information
Checksum	Checksum over the complete datagram (TCP header and data) and IP header information
Options	Additional 32-bit words for protocol options

11.1.6.3 IEEE 1588 message formats

The following sections describe the IEEE 1588 message formats.

11.1.6.3.1 Transport encapsulation

The precision time protocol (PTP) datagrams are encapsulated in Ethernet frames using the UDP/IP transport mechanism, or optionally, with the newer 1588v2 directly in Ethernet frames (layer 2).

Typically, multicast addresses are used to allow efficient distribution of the synchronization messages.

11.1.6.3.1.1 UDP/IP

The 1588 messages (v1 and v2) can be transported using UDP/IP multicast messages.

Table 11-17 shows IP multicast groups defined for PTP. The table also shows their respective MAC layer multicast address mapping according to RFC 1112 (last three octets of IP follow the fixed value of 01-00-5E).

Table 11-17. UDP/IP multicast domains

Name	IP Address	MAC Address mapping
DefaultPTPdomain	224.0.1.129	01-00-5E-00-01-81
AlternatePTPdomain1	224.0.1.130	01-00-5E-00-01-82
AlternatePTPdomain2	224.0.1.131	01-00-5E-00-01-83
AlternatePTPdomain3	224.0.1.132	01-00-5E-00-01-84

Table 11-18. UDP port numbers

Message type	UDP port	Note
Event	319	Used for SYNC and DELAY_REQUEST messages
General	320	All other messages (for example, follow-up, delay-response)

11.1.6.3.1.2 Native Ethernet (PTPv2)

In addition to using UDP/IP frames, IEEE 1588v2 defines a native Ethernet frame format that uses ethertype = 0x88F7. The payload of the Ethernet frame immediately contains the PTP datagram, starting with the PTPv2 header.

Besides others, version 2 adds a peer delay mechanism to allow delay measurements between individual point-to-point links along a path over multiple nodes. The following multicast domains are also defined in PTPv2.

Table 11-19. PTPv2 multicast domains

Name	MAC address
Normal messages	01-1B-19-00-00-00
Peer delay messages	01-80-C2-00-00-0E

11.1.6.3.2 PTP header

All PTP frames contain a common header that determines the protocol version and the type of message, which defines the remaining content of the message.

All multi-octet fields are transmitted in big-endian order (the most significant byte is transmitted/received first).

The last four bits of versionPTP are at the same position (second byte) for PTPv1 and PTPv2 headers. This allows accurate identification by inspecting the first two bytes of the message.

11.1.6.3.2.1 PTPv1 header

Table 11-20. Common PTPv1 message header

Offset	Octets	Bits							
		7	6	5	4	3	2	1	0
0	2	versionPTP = 0x0001							
2	2	versionNetwork							
4	16	subdomain							
20	1	messageType							
21	1	sourceCommunicationTechnology							
22	6	sourceUuid							
28	2	sourcePortId							
30	2	sequenceId							
32	1	control							
33	1	0x00							
34	2	flags							
36	4	reserved							

The type of message is encoded in the messageType and control fields as shown in Table 11-21 :

Table 11-21. PTPv1 message type identification

messageType	control	Message Name	Message
0x01	0x0	SYNC	Event message
0x01	0x1	DELAY_REQ	Event message
0x02	0x2	FOLLOW_UP	General message
0x02	0x3	DELAY_RESP	General message
0x02	0x4	MANAGEMENT	General message
other	other	—	Reserved

The field sequenceId is used to non-ambiguously identify a message.

11.1.6.3.2.2 PTPv2 header

Table 11-22. Common PTPv2 message header

Offset	Octets	Bits							
		7	6	5	4	3	2	1	0
0	1	transportSpecific				messageId			
1	1	reserved				versionPTP = 0x2			
2	2	messageLength							
4	1	domainNumber							
5	1	reserved							
6	2	flags							
8	8	correctionField							
16	4	reserved							
20	10	sourcePortIdentity							
30	2	sequenceId							
32	1	control							
33	1	logMeanMessageInterval							

The type of message is encoded in the field messageId as follows:

Table 11-23. PTPv2 message type identification

messageId	Message name	Message
0x0	SYNC	Event message
0x1	DELAY_REQ	Event message
0x2	PATH_DELAY_REQ	Event message
0x3	PATH_DELAY_RESP	Event message
0x4–0x7	—	Reserved
0x8	FOLLOW_UP	General message
0x9	DELAY_RESP	General message
0xa	PATH_DELAY_FOLLOW_UP	General message
0xb	ANNOUNCE	General message
0xc	SIGNALING	General message
0xd	MANAGEMENT	General message

The PTPv2 flags field contains further details on the type of message, especially if one-step or two-step implementations are used. The one- or two-step implementation is controlled by the TWO_STEP bit in the first octet of the flags field as shown below. Reserved bits are cleared.

Table 11-24. PTPv2 message flags field definitions

Bit	Name	Description
0	ALTERNATE_MASTER	See IEEE 1588 Clause 17.4
1	TWO_STEP	1 Two-step clock 0 One-step clock
2	UNICAST	1 Transport layer address uses a unicast destination address 0 Multicast is used
3	—	Reserved
4	—	Reserved
5	Profile specific	
6	Profile specific	
7	—	Reserved

11.1.6.4 MAC receive

The MAC receive engine performs the following tasks:

- Check frame framing
- Remove frame preamble and frame SFD field
- Discard frame based on frame destination address field
- Terminate pause frames
- Check frame length
- Remove payload padding if it exists
- Calculate and verify CRC-32
- Write received frames in the core receive FIFO

If the MAC is programmed to operate in half-duplex mode, it will also check if the frame is received with a collision.

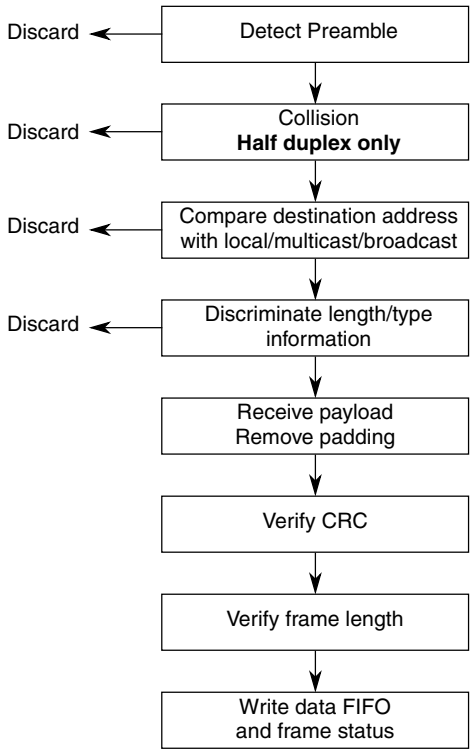


Figure 11-5. MAC receive flow

11.1.6.4.1 Collision detection in half-duplex mode

If the packet is received with a collision detected during reception of the first 64 bytes, the packet is discarded (if frame size was less than ~14 octets) or transmitted to the user application with an error and RxBD[CE] set.

11.1.6.4.2 Preamble processing

The IEEE 802.3 standard allows a maximum size of 56 bits (seven bytes) for the preamble, while the MAC core allows any preamble length, including zero length preamble.

The MAC core checks for the start frame delimiter (SFD) byte. If the next byte of the preamble, which is different from 0x55, is not 0xD5, the frame is discarded.

Although the IEEE specification dictates that the inner-packet gap should be at least 96 bits, the MAC core is designed to accept frames separated by only 64 10/100-Mbit/s operation (MII) bits.

The MAC core removes the preamble and SFD bytes.

11.1.6.4.3 MAC address check

The destination address bit 0 differentiates between multicast and unicast addresses.

- If bit 0 is 0, the MAC address is an individual (unicast) address.
- If bit 0 is 1, the MAC address defines a group (multicast) address.
- If all 48 bits of the MAC address are set, it indicates a broadcast address.

11.1.6.4.3.1 Unicast address check

If a unicast address is received, the destination MAC address is compared to the node MAC address programmed by the host in the PADDR1/2 registers.

If the destination address matches any of the programmed MAC addresses, the frame is accepted.

If Promiscuous mode is enabled (RCR[PROM] = 1) no address checking is performed and all unicast frames are accepted.

11.1.6.4.3.2 Multicast and unicast address resolution

The hash table algorithm used in the group and individual hash filtering operates as follows.

- The 48-bit destination address is mapped into one of 64 bits, represented by 64 bits in ENET*n*_GAUR/GALR (group address hash match) or ENET*n*_IAUR/IALR (individual address hash match).
- This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63.
- The msb of the CRC result selects ENET*n*_GAUR (msb = 1) or ENET*n*_GALR (msb = 0).
- The five lsbs of the hash result select the bit within the selected register.
- If the CRC generator selects a bit set in the hash table, the frame is accepted; else, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (or 87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The user must initialize the hash table registers. Use this CRC32 polynomial to compute the hash:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

If Promiscuous mode is enabled (ENET*n*_RCR[PROM] = 1) all unicast and multicast frames are accepted regardless of ENET*n*_GAUR/GALR and ENET*n*_IAUR/IALR settings.

11.1.6.4.3.3 Broadcast address reject

All broadcast frames are accepted if BC_REJ is cleared or ENET*n*_RCR[PROM] is set. If PROM is cleared when ENET*n*_RCR[BC_REJ] is set, all broadcast frames are rejected.

Table 11-25. Broadcast address reject programming

PROM	BC_REJ	Broadcast frames
0	0	Accepted
0	1	Rejected
1	0	Accepted
1	1	Accepted

11.1.6.4.3.4 Miss-bit implementation

For higher layer filtering purposes, RxBD[M] indicates an address miss when the MAC operates in promiscuous mode and accepts a frame that would otherwise be rejected.

If a group/individual hash or exact match does not occur and Promiscuous mode is enabled (RCR[PROM] = 1), the frame is accepted and the M bit is set in the buffer descriptor; otherwise, the frame is rejected.

This means the status bit is set in any of the following conditions during Promiscuous mode:

- A broadcast frame is received when BC_REJ is set
- A unicast is received that does not match either:

- Node address (PALR[PADDR1] and PAUR[PADDR2])
- Hash table for unicast (IAUR[IADDR1] and IALR[IADDR2])

- A multicast is received that does not match the GAUR[GADDR1] and GALR[GADDR2] hash table entries

11.1.6.4.4 Frame length/type verification: payload length check

If the length/type is less than 0x600 and NLC is set, the MAC checks the payload length and reports any error in the frame status word and interrupt bit PLR.

If the length/type is greater than or equal to 0x600, the MAC interprets the field as a type and no payload length check is performed.

The length check is performed on VLAN and stacked VLAN frames. If a padded frame is received, no length check can be performed due to the extended frame payload because padded frames can never have a payload length error.

11.1.6.4.5 Frame length/type verification: frame length check

When the receive frame length exceeds MAX_FL bytes, the BABR interrupt is generated and the RxBD[LG] bit is set.

The frame is not truncated unless the frame length exceeds the value programmed in ENET*n*_FTRL[TRUNC_FL]. If the frame is truncated, RxBD[TR] is set. In addition, a truncated frame always has the CRC error indication set (RxBD[CR]).

11.1.6.4.6 VLAN frames processing

VLAN frames have a length/type field set to 0x8100 immediately followed by a 16-Bit VLAN control information field.

VLAN-tagged frames are received as normal frames because the VLAN tag is not interpreted by the MAC function, and are pushed complete with the VLAN tag to the user application. If the length/type field of the VLAN-tagged frame, which is found four octets later in the frame, is less than 42, the padding is removed. In addition, the frame status word (RxBD[NO]) indicates that the current frame is VLAN tagged.

11.1.6.4.7 Pause frame termination

The receive engine terminates pause frames and does not transfer them to the receive FIFO. The quanta is extracted and sent to the MAC transmit path via a small internal clock rate decoupling asynchronous FIFO.

The quanta is written only if a correct CRC and frame length are detected by the control state machine. If not, the quanta is discarded and the MAC transmit path is not paused.

Good pause frames are ignored if ENET*n*_RCR[FCE] is cleared and are forwarded to the client interface when ENET*n*_RCR[PAUFWD] is set.

11.1.6.4.8 CRC check

The CRC-32 field is checked and forwarded to the core FIFO interface if ENET*n*_RCR[CRCFWD] is cleared and ENET*n*_RCR[PADEN] is set.

When CRCFWD is set (regardless of PADEN), the CRC-32 field is checked and terminated (not transmitted to the FIFO).

The CRC polynomial, as specified in the 802.3 standard, is:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

The 32 bits of the CRC value are placed in the frame check sequence (FCS) field with the x^{31} term as right-most bit of the first octet. The CRC bits are thus received in the following order: x^{31}, x^{30},..., x^{1}, x^{0}.

If a CRC error is detected, the frame is marked invalid and RxBD[CR] is set.

11.1.6.4.9 Frame padding removal

When a frame is received with a payload length field set to less than 46 (42 for VLAN-tagged frames and 38 for frames with stacked VLANs), the zero padding can be removed before the frame is written into the data FIFO depending on the setting of ENET*n*_RCR[PADEN].

Note

If a frame is received with excess padding (in other words, the length field is set as mentioned above, but the frame has more than 64 octets) and padding removal is enabled, then the padding is removed as normal and no error is reported if the frame is otherwise correct (for example: good CRC, less than maximum length, and no other error).

11.1.6.4.10 Frame classification (AVB)

To support protocols such as Audio Video Bridging (AVB, IEEE 802.1Qav), normal traffic is separated from time-sensitive traffic immediately at the application interface to allow storing them in different queues for further processing. For every frame received, a

classification based on VLAN priority can be performed. When a frame is received, and it contains a VLAN tag, the priority is compared against the values set in the classification match registers (see Receive Classification Match Register for Class n (ENET_RCMR*n*)) and the following figure.

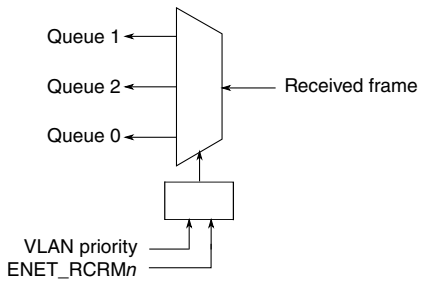


Figure 11-6. AVB frame classification

11.1.6.4.11 Receive flushing

RX flushing prevents frames in the RX FIFO from being blocked. Blocking can occur if the frame at the head of the RX FIFO cannot be forwarded because the ring it is associated with cannot accept it. This situation occurs when either the ring's RxBD[EMPTY] is not set or ENET_RDAR*n* is not set. When RX flushing is enabled, via QOS Scheme (ENET_QOS), the blocking frame will be flushed (discarded).

11.1.6.5 MAC transmit

Frame transmission starts when the transmit FIFO holds enough data.

After a transfer starts, the MAC transmit function performs the following tasks:

- Generates preamble and SFD field before frame transmission
- Generates XOFF pause frames if the receive FIFO reports a congestion or if ENET*n*_TCR[TFC_PAUSE] is set with ENET*n*_OPD[PAUSE_DUR] set to a non-zero value
- Generates XON pause frames if the receive FIFO congestion condition is cleared or if TFC_PAUSE is set with PAUSE_DUR cleared
- Suspends Ethernet frame transfer (XOFF) if a non-zero pause quanta is received from the MAC receive path
- Adds padding to the frame if required

- Calculates and appends CRC-32 to the transmitted frame
- Sends the frame with correct inter-packet gap (IPG) (deferring)

When the MAC is configured to operate in half-duplex mode, the following additional tasks are performed:

- Collision detection
- Frame retransmit after back-off timer expires

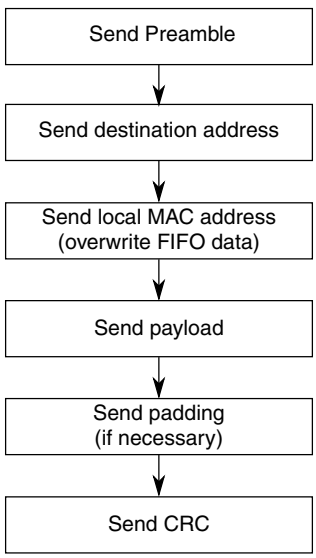


Figure 11-7. Frame transmit overview

11.1.6.5.1 Frame payload padding

The IEEE specification defines a minimum frame length of 64 bytes.

If the frame sent to the MAC from the user application has a size smaller than 60 bytes, the MAC automatically adds padding bytes (0x00) to comply with the Ethernet minimum frame length specification. Transmit padding is always performed and cannot be disabled.

If the MAC is not allowed to append a CRC (TxBD[TC] = 1), the user application is responsible for providing frames with a minimum length of 64 octets.

11.1.6.5.2 MAC address insertion

On each frame received from the core transmit FIFO interface, the source MAC address is either:

- Replaced by the address programmed in the PADDR1/2 fields (ENET*n*_TCR[ADDINS] = 1)
- Transparently forwarded to the Ethernet line (ENET*n*_TCR[ADDINS] = 0)

11.1.6.5.3 CRC-32 generation

The CRC-32 field is optionally generated and appended at the end of a frame.

The CRC polynomial, as specified in the 802.3 standard, is:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

The 32 bits of the CRC value are placed in the FCS field so that the x^{31} term is the right-most bit of the first octet. The CRC bits are thus transmitted in the following order: x^{31}, x^{30},..., x^{1}, x^{0}.

11.1.6.5.4 Inter-packet gap (IPG)

In full-duplex mode, after frame transmission and before transmission of a new frame, an IPG (programmed in ENET*n*_TIPG) is maintained. The minimum IPG can be programmed between 8 and 26 byte-times (64 and 208 bit-times).

In half-duplex mode, the core constantly monitors the line. Actual transmission of the data onto the network occurs only if it has been idle for a 96-bit time period, and any back-off time requirements have been satisfied. In accordance with the standard, the core begins to measure the IPG from CRS/GMII_CRS de-assertion.

11.1.6.5.5 Collision detection and handling — half-duplex operation only

A collision occurs on a half-duplex network when concurrent transmissions from two or more nodes take place. During transmission, the core monitors the line condition and detects a collision when the PHY device asserts COL.

When the core detects a collision while transmitting, it stops transmission of the data and transmits a 32-bit jam pattern. If the collision is detected during the preamble or the SFD transmission, the jam pattern is transmitted after completing the SFD, which results in a minimum 96-bit fragment. The jam pattern is a fixed pattern that is not compared to the actual frame CRC, and has a very low probability (0.532) of having a jam pattern identical to the CRC.

If a collision occurs before transmission of 64 bytes (including preamble and SFD), the MAC core waits for the backoff period and retransmits the packet data (stored in a 64-byte re-transmit buffer) that has already been sent on the line. The backoff period is generated from a pseudo-random process (truncated binary exponential backoff).

If a collision occurs after transmission of 64 bytes (including preamble and SFD), the MAC discards the remainder of the frame, optionally sets the LC interrupt bit, and sets TxBD[LCE].

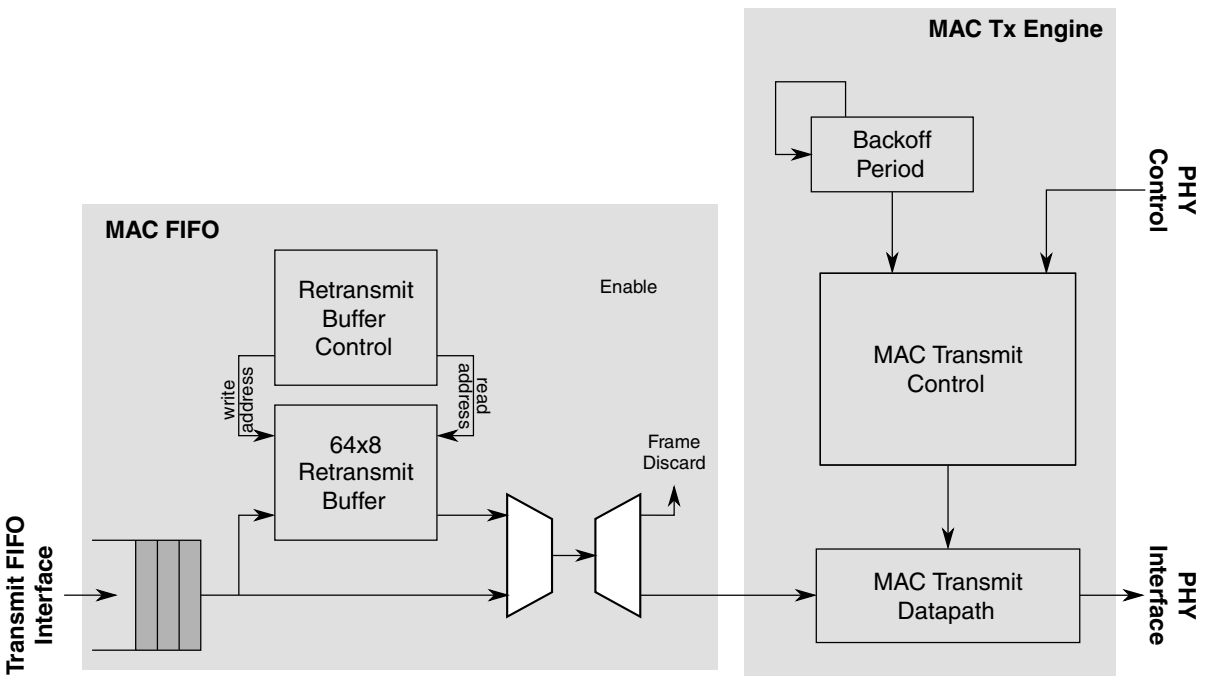


Figure 11-8. Packet re-transmit overview

The backoff time is represented by an integer multiple of slot times. One slot is equal to a 512-bit time period. The number of the delay slot times, before the n^{th} re-transmission attempt, is chosen as a uniformly-distributed random integer in the range:

- $0 < r < 2^k$
- $k = \min(n, N)$; where n is the number of retransmissions and $N = 10$

For example, after the first collision, the backoff period is 0 or 1 slot time. If a collision occurs on the first retransmission, the backoff period is 0, 1, 2, or 3, and so on.

The maximum backoff time (in 512-bit time slots) is limited by N = 10 as specified in the IEEE 802.3 standard.

If a collision occurs after 16 consecutive retransmissions, the core reports an excessive collision condition (ENET*n*_EIR[RL] interrupt field and TxBD[EE]) and discards the current packet from the FIFO.

In networks violating the standard requirements, a collision may occur after transmission of the first 64 bytes. In this case, the core stops the current packet transmission and discards the rest of the packet from the transmit FIFO. The core resumes transmission with the next packet available in the core transmit FIFO.

warning

Ethernet PHYs that support the SQE Test, or "heartbeat," feature must disable this feature. When this feature is enabled, the PHY asserts the collision signal after a frame is transmitted to indicate to the ENET that the PHY's collision logic is working. This may cause data corruption in the next frame from the ENET. This corrupted frame contains up to 21 zero bytes which start somewhere within the MAC destination address field. The ENET, however, will still generate a good FCS (CRC-32) but with corrupted data.

11.1.6.5.6 Rate limiting / traffic shaping support

The MAC-NET supports two methods to optimize frame traffic for either time-sensitive AVB frames (Class A and Class B) or best-effort non-AVB frames:

- Round-robin scheme
- Credit-based traffic shaping

To ensure that sufficient bandwidth is allocated for the AVB frames, the credit-based shaper must be used. Either method can be combined with a time-based shaper to ensure that a frame is always transmitted in its correct time slot.

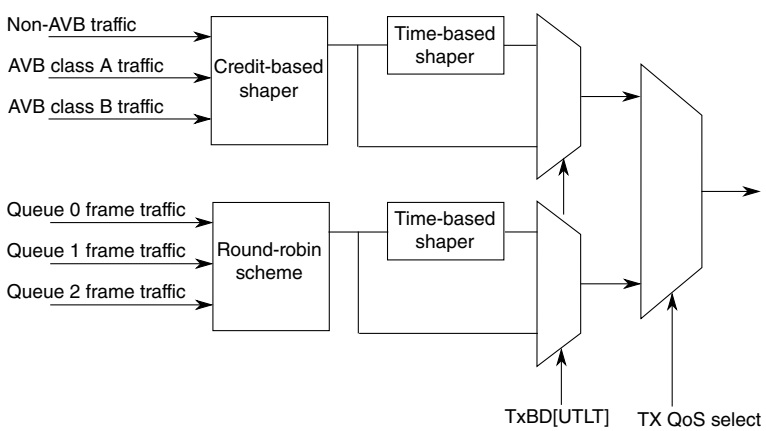


Figure 11-9. Transmit arbitration schemes

11.1.6.5.6.1 Round-robin policy

In the round-robin QoS scheme, each queue is given an equal opportunity to transmit one frame. For example, if queue *n* has a frame to transmit, the queue transmits its frame. After queue *n* has transmitted its frame, or if queue *n* does not have a frame to transmit, queue *n*+1 is then allowed to transmit its frame, and so on.

11.1.6.5.6.2 Credit-based shaper

The AVB credit based shaper acts independently, per class, to control the bandwidth distribution between normal traffic and time-sensitive traffic with respect to the total link bandwidth available. As per the IEEE 802.1Q, the maximum bandwidth distribution that can be allocated for the time-sensitive frames is 75%. The following example uses 70% for AVB frames and 30% for non-AVB frames:

Table 11-26. Bandwidth allocation example

AVB Class A	50%
AVB Class B	20%
non-AVB frames	30%
Total	100%

See DMA Class Based Configuration (ENET_DMA*n*CFG) for information on how to allocate bandwidth. The following figure shows how frame traffic is controlled within the MAC-NET core.

11.1.6.5.6.3 Time-based shaper

The time-based shaper enables the user to specify when a frame can be transmitted. It is always used in combination with either the round-robin scheme or the credit-based shaper. Use of the time-based shaper can ensure that frames are always transmitted in the correct time slot.

11.1.6.5.6.3.1 Time-based shaper example

Time-based shaping involves the following procedure:

1. Read the current value of the timer. See Timer Value Register (ENET_ATVR) and Adjustable Timer Control Register (ENET_ATCR).
2. Calculate the frame launch time and write this value to the TLT field of the frame's TxBD. See Enhanced transmit buffer descriptor.
3. Instruct the ENET to use the TLT by setting TxBD[UTLT].

The frame will be fetched and transmitted only if TxBD[TLT] is less than the current value of the timer. In other words, the timer must be past, that is, be greater than, the transmit launch time before the frame will be fetched and transmitted.

The transmit launch time must be not be greater than the current value of ENET_ATVR + (0.5 × ENET_ATPER). This means the application can not prepare frames with launch times beyond ENET_ATVR + (0.5 × ENET_ATPER). Because ENET_ATVR wraps, calculate TLT using the equation

(ENET_ATVR + (0.5 × ENET_ATPER)) mod ENET_ATPER

For example, if ENET_ATPER is set to the recommended value of 1,000,000,000 ns (one second), then the user must not prepare future frames with launch times greater than (500,000,000 + ENET_ATVR).

As a specific example, if the current value of ENET_ATVR is 100,000, then you can prepare the following frames:

1. Frame 1 TxBD[TLT] = 100,000ns + 125,000ns
2. Frame 2 TxBD[TLT] = 100,000ns + 250,000ns
3. And so on

11.1.6.6 Full-duplex flow control operation

Three conditions are handled by the core's flow control engine:

- Remote device congestion — The remote device connected to the same Ethernet segment as the core reports congestion and requests that the core stop sending data.
- Core FIFO congestion — When the core's receive FIFO reaches a user-programmable threshold (RX section empty), the core sends a pause frame back to the remote device requesting the data transfer to stop.
- Local device congestion — Any device connected to the core can request (typically, via the host processor) the remote device to stop transmitting data.

11.1.6.6.1 Remote device congestion

When the MAC transmit control gets a valid pause quanta from the receive path and if ENET*n*_RCR[FCE] is set, the MAC transmit logic:

- Completes the transfer of the current frame.

- Stops sending data for the amount of time specified by the pause quanta in 512 bit time increments.
- Sets ENET*n*_TCR[RFC_PAUSE].

Frame transfer resumes when the time specified by the quanta expires and if no new quanta value is received, or if a new pause frame with a quanta value set to 0x0000 is received. The MAC also resets RFC_PAUSE to zero.

If ENET*n*_RCR[FCE] cleared, the MAC ignores received pause frames.

Optionally and independent of ENET*n*_RCR[FCE], pause frames are forwarded to the client interface if PAUFWD is set.

11.1.6.6.2 Local device/FIFO congestion

The MAC transmit engine generates pause frames when the local receive FIFO is not able to receive more than a pre-defined number of words (FIFO programmable threshold) or when pause frame generation is requested by the local host processor.

- To generate a pause frame, the host processor sets ENET*n*_TCR[TFC_PAUSE]. A single pause frame is generated when the current frame transfer is completed and TFC_PAUSE is automatically cleared. Optionally, an interrupt is generated.
- An XOFF pause frame is generated when the receive FIFO asserts its section empty flag (internal). An XOFF pause frame is generated automatically, when the current frame transfer completes.
- An XON pause frame is generated when the receive FIFO deasserts its section empty flag (internal). An XON pause frame is generated automatically, when the current frame transfer completes.

When an XOFF pause frame is generated, the pause quanta (payload byte P1 and P2) is filled with the value programmed in ENET*n*_OPD[PAUSE_DUR].

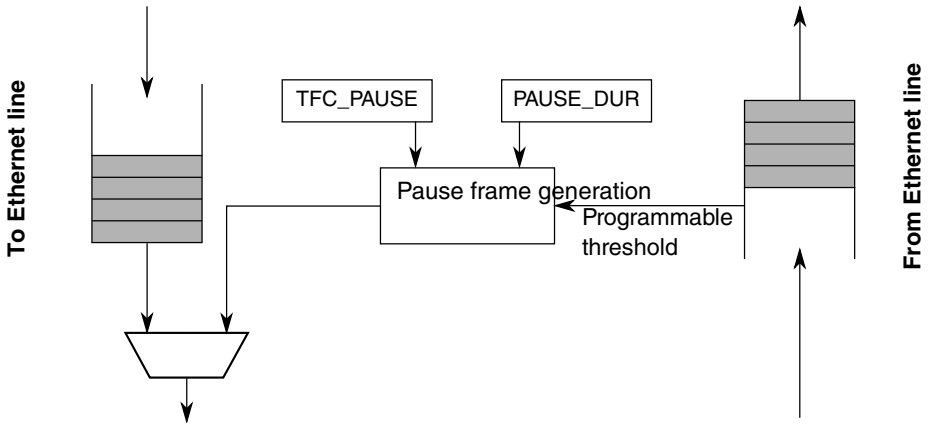


Figure 11-10. Pause frame generation overview

Note

Although the flow control mechanism should prevent any FIFO overflow on the MAC core receive path, the core receive FIFO is protected. When an overflow is detected on the receive FIFO, the current frame is truncated with an error indication set in the frame status word. The frame should subsequently be discarded by the user application.

11.1.6.7 Magic packet detection

Magic packet detection wakes a node that is put in power-down mode by the node management agent. Magic packet detection is supported only if the MAC is configured in sleep mode.

11.1.6.7.1 Sleep mode

To put the MAC in Sleep mode, set ENET*n*_ECR[SLEEP]. At the same time ENET*n*_ECR[MAGICEN] should also be set to enable magic packet detection.

In addition, when the processor is in Stop mode, Sleep mode is entered, without affecting the ENET*n*_ECR register bits.

When the MAC is in Sleep mode:

- The transmit logic is disabled.
- The FIFO receive/transmit functions are disabled.
- The receive logic is kept in Normal mode, but it ignores all traffic from the line except magic packets. They are detected so that a remote agent can wake the node.

11.1.6.7.2 Magic packet detection

The core is designed to detect magic packets (see Magic packets) with the destination address set to:

- Any multicast address
- The broadcast address
- The unicast address programmed in PADDR1/2

When a magic packet is detected, EIR[WAKEUP] is set and none of the statistic registers are incremented.

11.1.6.7.3 Wakeup

When a magic packet is detected, indicated by ENET*n*_EIR[WAKEUP], ENET*n*_ECR[SLEEP] should be cleared to resume normal operation of the MAC. Clearing the SLEEP bit automatically masks ENET*n*_ECR[MAGICEN], disabling magic packet detection.

11.1.6.8 IP accelerator functions

The following sections describe the IP accelerator functions.

11.1.6.8.1 Checksum calculation

The IP and ICMP, TCP, UDP checksums are calculated with one's complement arithmetic summing up 16-bit values.

- For ICMP, the checksum is calculated over the complete ICMP datagram, in other words without IP header.
- For TCP and UDP, the checksums contain the header and data sections and values from the IP header, which can be seen as a pseudo-header that is not actually present in the data stream.

Table 11-27. IPv4 pseudo-header for checksum calculation

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source address							
Destination address							
Zero		Protocol		TCP/UDP length			

Table 11-28. IPv6 pseudo-header for checksum calculation

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source address							
Destination address							
TCP/UDP length							
Zero						Next header	

The TCP/UDP length value is the length of the TCP or UDP datagram, which is equal to the payload of an IP datagram. It is derived by subtracting the IP header length from the complete IP datagram length that is given in the IP header (IPv4), or directly taken from the IP header (IPv6). The protocol field is the corresponding value from the IP header. The Zero fields are all zeroes.

For IPv6, the complete 128-bit addresses are considered. The next header value identifies the upper layer protocol as either TCP or UDP. It may differ from the next header value of the IPv6 header if extension headers are inserted before the protocol header.

The checksum calculation uses 16-bit words in network byte order: The first byte sent/received is the MSB, and the second byte sent/received is the LSB of the 16-bit value to add to the checksum. If the frame ends on an odd number of bytes, a zero byte is appended for checksum calculation only, and is not actually transmitted.

11.1.6.8.2 Additional padding processing

According to IEEE 802.3, any Ethernet frame must have a minimum length of 64 octets.

The MAC usually removes padding on receive when a frame with length information is received. Because IP frames have a type value instead of length, the MAC does not remove padding for short IP frames, as it is not aware of the frame contents.

The IP accelerator function can be configured to remove the Ethernet padding bytes that might follow the IP datagram.

On transmit, the MAC automatically adds padding as necessary to fill any frame to a 64-byte length.

11.1.6.8.3 32-bit Ethernet payload alignment

The data FIFOs allow inserting two additional arbitrary bytes in front of a frame. This extends the 14-byte Ethernet header to a 16-byte header, which leads to alignment of the Ethernet payload, following the Ethernet header, on a 32-bit boundary.

This function can be enabled for transmit and receive independently with the corresponding SHIFT16 bits in the ENET*n*_TACC and ENET*n*_RACC registers.

When enabled, the valid frame data is arranged as shown in Table 11-29.

Table 11-29. 64-bit interface data structure with SHIFT16 enabled

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	Any value	Any value
Byte 13	Byte 12	Byte 11	Byte 10	Byte 9	Byte 8	Byte 7	Byte 6
...							

11.1.6.8.3.1 Receive processing

When ENET*n*_RACC[SHIFT16] is set, each frame is received with two additional bytes in front of the frame.

The user application must ignore these first two bytes and find the first byte of the frame in bits 23–16 of the first word from the RX FIFO.

Note

SHIFT16 must be set during initialization and kept set during the complete operation, because it influences the FIFO write behavior.

11.1.6.8.3.2 Transmit processing

When ENET*n*_TACC[SHIFT16] is set, the first two bytes of the first word written (bits 15–0) are discarded immediately by the FIFO write logic.

The SHIFT16 bit can be enabled/disabled for each frame individually if required, but can be changed only between frames.

11.1.6.8.4 Received frame discard

Because the receive FIFO must be operated in store and forward mode (ENET*n*_RSFL cleared), received frames can be discarded based on the following errors:

- The MAC function receives the frame with an error:
  - The frame has an invalid payload length
  - Frame length is greater than MAX_FL

- Frame received with a CRC-32 error
- Frame truncated due to receive FIFO overflow
- Frame is corrupted as PHY signaled an error (RX_ERR asserted during reception)

- An IP frame is detected and the IP header checksum is wrong
- An IP frame with a valid IP header and a valid IP header checksum is detected, the protocol is known but the protocol-specific checksum is wrong

If one of the errors occurs and the IP accelerator function is configured to discard frames (ENET*n*_RACC), the frame is automatically discarded. Statistics are maintained normally and are not affected by this discard function.

11.1.6.8.5 IPv4 fragments

When an IPv4 IP fragment frame is received, only the IP header is inspected and its checksum verified. 32-bit alignment operates the same way on fragments as it does on normal IP frames, as specified above.

The IP fragment frame payload is not inspected for any protocol headers. As such, a protocol header would only exist in the very first fragment. To assist in protocol-specific checksum verification, the one's-complement sum is calculated on the IP payload (all bytes following the IP header) and provided with the frame status word.

The frame fragment status field (RxBD[FRAG]) is set to indicate a fragment reception, and the one's-complement sum of the IP payload is available in RxBD[Payload checksum].

Note

After all fragments have been received and reassembled, the application software can take advantage of the payload checksum delivered with the frame's status word to calculate the protocol-specific checksum of the datagram.

For example, if a TCP payload is delivered by multiple IP fragments, the application software can calculate the pseudo-header checksum value from the first fragment, and add the payload checksums delivered with the status for all fragments to verify the TCP datagram checksum.

11.1.6.8.6 IPv6 support

The following sections describe the IPv6 support.

11.1.6.8.6.1 Receive processing

An Ethernet frame of type 0x86DD identifies an IP Version 6 frame (IPv6) frame. If an IPv6 frame is received, the first IP header is inspected (first ten words), which is available in every IPv6 frame.

If the receive SHIFT16 function is enabled, the IP header is aligned on a 32-bit boundary allowing more efficient processing (see 32-bit Ethernet payload alignment).

For TCP and UDP datagrams, the pseudo-header checksum calculation is performed and verified.

To assist in protocol-specific checksum verification, the one's-complement sum is always calculated on the IP payload (all bytes following the IP header) and provided with the frame status word. For example, if extension headers were present, their sums can be subtracted in software from the checksum to isolate the TCP/UDP datagram checksum, if required.

11.1.6.8.6.2 Transmit processing

For IPv6 transmission, the SHIFT16 function is supported to process 32-bit aligned datagrams.

IPv6 has no IP header checksum; therefore, the IP checksum insertion configuration is ignored.

The protocol checksum is inserted only if the next header of the IP header is a known protocol (TCP, UDP, or ICMP). If a known protocol is detected, the checksum over all bytes following the IP header is calculated and inserted in the correct position.

The pseudo-header checksum calculation is performed for TCP and UDP datagrams accordingly.

11.1.6.9 Resets and stop controls

The following sections describe the resets and stop controls.

11.1.6.9.1 Hardware reset

To reset the Ethernet module, set ENET*n*_ECR[RESET].

11.1.6.9.2 Soft reset

When ENET*n*_ECR[ETHER_EN] is cleared during operation, the following occurs:

- uDMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers.
- A currently ongoing transmit is terminated by asserting GMII/TXER to the PHY.
- A currently ongoing transmit FIFO write from the application is terminated by stopping the write to the FIFO, and all further data from the application is ignored. All subsequent writes are ignored until re-enabled.
- A currently ongoing receive FIFO read is terminated. The RxBD has arbitrary values in this case.

11.1.6.9.3 Hardware freeze

When the processor enters debug mode and ECR[DBGEN] is set, the MAC enters a freeze state where it stops all transmit and receive activities gracefully.

The following happens when the MAC enters hardware freeze:

- A currently ongoing receive transaction on the receive application interface is completed as normal. No further frames are read from the FIFO.
- A currently ongoing transmit transaction on the transmit application interface is completed as normal (in other words, until writing end-of-packet (EOP)).
- A currently ongoing frame receive is completed normally, after which no further frames are accepted from the MII/GMII.
- A currently ongoing frame transmit is completed normally, after which no further frames are transmitted.

11.1.6.9.4 Graceful stop

During a graceful stop, any currently ongoing transactions are completed normally and no further frames are accepted. The MAC can resume from a graceful stop without the need for a reset (for example, clearing ETHER_EN is not required).

The following conditions lead to a graceful stop of the MAC transmit or receive datapaths.

11.1.6.9.4.1 Graceful transmit stop (GTS)

When gracefully stopped, the MAC is no longer reading frame data from the transmit FIFO and has completed any ongoing transmission.

In any of the following conditions, the transmit datapath stops after an ongoing frame transmission has been completed normally.

- ENET*n*_TCR[GTS] is set by software.
- ENET*n*_TCR[TFC_PAUSE] is set by software requesting a pause frame transmission. The status (and register bit) is cleared after the pause frame has been sent.
- A pause frame was received stopping the transmitter. The stopped situation is terminated when the pause timer expires or a pause frame with zero quanta is received.
- MAC is placed in Sleep mode by software or the processor entering Stop mode (see Sleep mode).
- The MAC is in Hardware Freeze mode.

When the transmitter has reached its stopped state, the following events occur:

- The GRA interrupt is asserted, when transitioned into stopped.
- In Hardware Freeze mode, the GRA interrupt does not wait for the application write completion and asserts when the transmit state machine (in other words, line side of TX FIFO) reaches its stopped state.

11.1.6.9.4.2 Graceful receive stop (GRS)

When gracefully stopped, the MAC is no longer writing frames into the receive FIFO.

The receive datapath stops after any ongoing frame reception has been completed normally, if any of the following conditions occur:

- MAC is placed in Sleep mode either by the software or the processor is in Stop mode). The MAC continues to receive frames and search for magic packets if enabled (see Magic packet detection). However, no frames are written into the receive FIFO, and therefore are not forwarded to the application.
- The MAC is in Hardware Freeze mode. The MAC does not accept any frames from the MII/GMII.

When the receive datapath is stopped, the following events occur:

- If the RX is in the stopped state, RCR[GRS] is set
- The GRA interrupt is asserted when the transmitter and receiver are stopped
- Any ongoing receive transaction to the application (RX FIFO read) continues normally until the frame end of package (EOP) is reached. After this, the following occurs:
  - When Sleep mode is active, all further frames are discarded, flushing the RX FIFO
  - In Hardware Freeze mode, no further frames are delivered to the application and they stay in the receive FIFO.

Note

The assertion of GRS does not wait for an ongoing FIFO read transaction on the application side of the FIFO (FIFO read).

11.1.6.9.4.3 Graceful stop interrupt (GRA)

The graceful stopped interrupt (GRA) is asserted for the following conditions:

- In Sleep mode, the interrupt asserts only after both TX and RX datapaths are stopped.
- In Hardware Freeze mode, the interrupt asserts only after both TX and RX datapaths are stopped.
- The MAC transmit datapath is stopped for any other condition (GTS, TFC_PAUSE, pause received).

The GRA interrupt is triggered only once when the stopped state is entered. If the interrupt is cleared while the stop condition persists, no further interrupt is triggered.

11.1.6.10 IEEE 1588 functions

To allow for IEEE 1588 or similar time synchronization protocol implementations, the MAC is combined with a time-stamping module to support precise time-stamping of incoming and outgoing frames. Set ENET*n*_ECR[EN1588] to enable 1588 support.

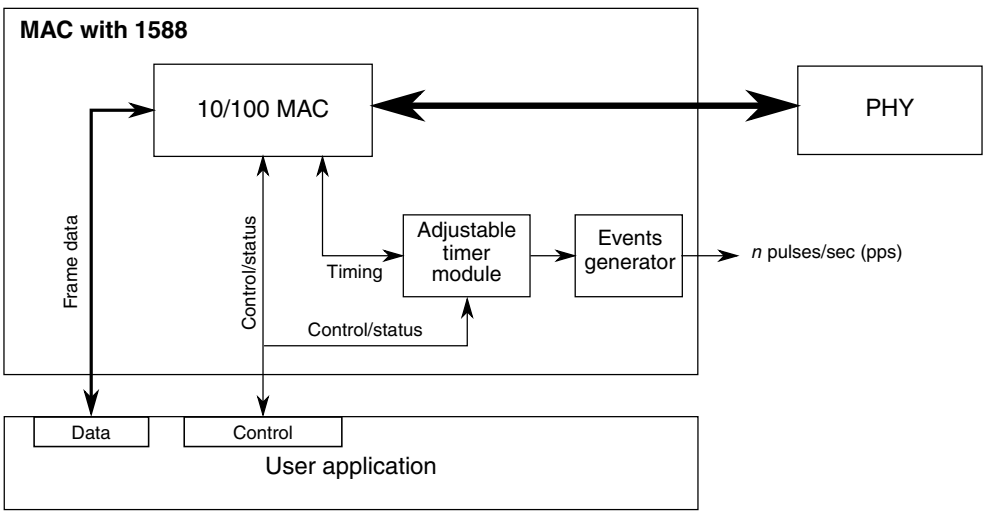


Figure 11-11. IEEE 1588 functions overview

11.1.6.10.1 Adjustable timer module

The adjustable timer module (TSM) implements the free-running counter (FRC), which generates the timestamps. The FRC operates with the time-stamping clock, which can be set to any value depending on your system requirements.

Through dedicated correction logic, the timer can be adjusted to allow synchronization to a remote master and provide a synchronized timing reference to the local system. The timer can be configured to cause an interrupt after a fixed time period, to allow synchronization of software timers or perform other synchronized system functions.

The timer is typically used to implement a period of one second; hence, its value ranges from 0 to (1×10^9)-1. The period event can trigger an interrupt, and software can maintain the seconds and hours time values as necessary.

11.1.6.10.1.1 Adjustable timer implementation

The adjustable timer consists of a programmable counter/accumulator and a correction counter. The periods of both counters and their increment rates are freely configurable, allowing very fine tuning of the timer.

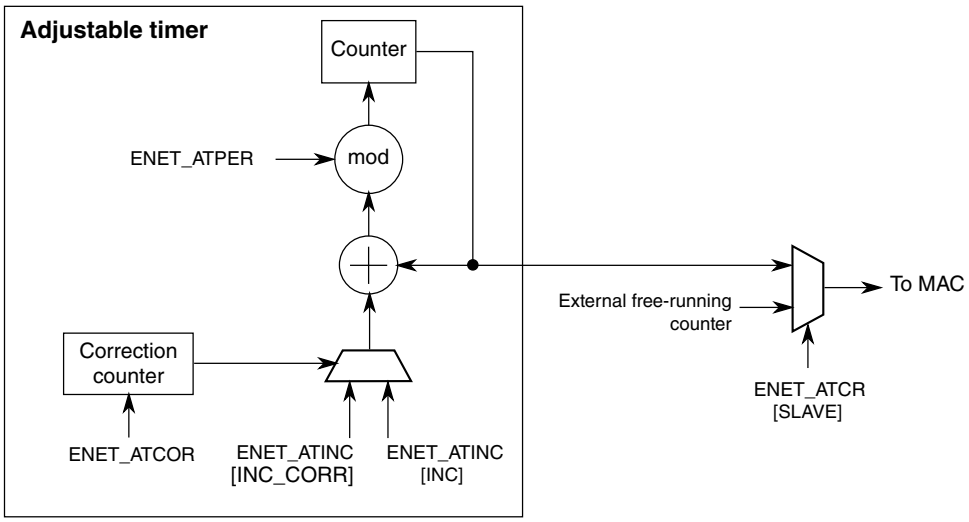


Figure 11-12. Adjustable timer implementation detail

The counter produces the current time. During each time-stamping clock cycle, a constant value is added to the current time as programmed in ENET*n*_ATINC. The value depends on the chosen time-stamping clock frequency. For example, if it operates at 125 MHz, setting the increment to eight represents 8 ns.

The period, configured in ENET*n*_ATPER, defines the modulo when the counter wraps. In a typical implementation, the period is set to 1×10^9 so that the counter wraps every second, and hence all timestamps represent the absolute nanoseconds within the one second period. When the period is reached, the counter wraps to start again respecting the period modulo. This means it does not necessarily start from zero, but instead the counter is loaded with the value (Current + Inc $-(1 \times 10^9)$), assuming the period is set to 1×10^9.

The correction counter operates fully independently, and increments by one with each time-stamping clock cycle. When it reaches the value configured in ENET*n*_ATCOR, it restarts and instructs the timer once to increment by the correction value, instead of the normal value.

The normal and correction increments are configured in ENET*n*_ATINC. To speed up the timer, set the correction increment more than the normal increment value. To slow down the timer, set the correction increment less than the normal increment value.

The correction counter only defines the distance of the corrective actions, not the amount. This allows very fine corrections and low jitter (in the range of 1 ns) independent of the chosen clock frequency.

By enabling slave mode (ENET*n*_ATCR[SLAVE] = 1), the timer is ignored and the current time is externally provided from one of the external modules. See the Chip Configuration details for which clock source is used. This is useful if multiple modules

within the system must operate from a single timer. When slave mode is enabled, you still must set ENET*n*_ATINC[INC] to the value of the master, since it is used for internal comparisons.

11.1.6.10.2 Transmit timestamping

Only 1588 event frames need to be time-stamped on transmit. The client application (for example, the MAC driver) should detect 1588 event frames and set TxBD[TS] together with the frame.

If TxBD[TS] is set, the MAC records the timestamp for the frame in ENET*n*_ATSTMP. ENET*n*_EIR[TS_AVAIL] is set to indicate that a new timestamp is available.

Software implements a handshaking procedure by setting TxBD[TS] when it transmits the frame for which a timestamp is needed, and then waits for ENET*n*_EIR[TS_AVAIL] to determine when the timestamp is available. The timestamp is then read from ENET*n*_ATSTMP. This is done for all event frames. Other frames do not use TxBD[TS] and, therefore, do not interfere with the timestamp capture.

11.1.6.10.3 Receive timestamping

When a frame is received, the MAC latches the value of the timer when the frame's start of frame delimiter (SFD) field is detected, and provides the captured timestamp on RxBD[1588 timestamp]. This is done for all received frames.

11.1.6.10.4 Time synchronization

The adjustable timer module is available to synchronize the local clock of a node to a remote master. It implements a free running 32-bit counter, and also contains an additional correction counter.

The correction counter increases or decreases the rate of the free running counter, enabling very fine granular changes of the timer for synchronization, yet adding only very low jitter when performing corrections.

The application software implements, in a slave scenario, the required control algorithm, setting the correction to compensate for local oscillator drifts and locking the timer to the remote master clock on the network.

The timer and all timestamp-related information should be configured to show the true nanoseconds value of a second (in other words, the timer is configured to have a period of one second). Hence, the values range from 0 to (1×10^9)–1. In this application, the seconds counter is implemented in software using an interrupt function that is executed when the nanoseconds counter wraps at 1×10^9.

11.1.6.10.5 Input Capture and Output Compare

The Input Capture Output Compare block can be used to provide precise hardware timing for input and output events.

11.1.6.10.5.1 Input capture

The TCCR*n* capture registers latch the time value when the corresponding external event occurs. An event can be a rising-, falling-, or either-edge of one of the 1588_TMR*n* signals. An event will cause the corresponding TCSR*n*[TF] timer flag to be set, indicating that an input capture has occurred. If the corresponding interrupt is enabled with the TCSR*n*[TIE] field, an interrupt can be generated.

11.1.6.10.5.2 Output compare

The TCCR*n* compare registers are loaded with the time at which the corresponding event should occur. When the ENET free-running counter value matches the output compare reference value in the TCCR*n* register, the corresponding flag, TCSR*n*[TF], is set, indicating that an output compare has occurred. The corresponding interrupt, if enabled by TCSR*n*[TIE], will be generated.The corresponding 1588_TMR*n* output signal will be asserted according to TCSR*n*[TMODE].

11.1.6.10.5.3 DMA requests

A DMA request can be enabled by setting TCSR*n*[TDRE]. The corresponding DMA request is generated when the TCSR*n*[TF] timer flag is set. When the DMA has completed, the corresponding TCSRn[TF] flag is cleared.

11.1.6.11 FIFO thresholds

The core FIFO thresholds are fully programmable to dynamically change the FIFO operation.

For example, store and forward transfer can be enabled by a simple change in the FIFO threshold registers.

The thresholds are defined in 64-bit words.

The receive and transmit FIFOs both have a depth of 1024 words.

11.1.6.11.1 Receive FIFO

Four programmable thresholds are available, which can be set to any value to control the core operation as follows.

Table 11-30. Receive FIFO thresholds definition

Register	Description
ENET*n*_RSFL [RX_SECTION_FULL]	When the FIFO level reaches the ENET*n*_RSFL value, the MAC status signal is asserted to indicate that data is available in the receive FIFO (cut-through operation). Once asserted, if the FIFO empties below the threshold set with ENET*n*_RAEM and if the end-of-frame is not yet stored in the FIFO, the status signal is deasserted again. If a frame has a size smaller than the threshold (in other words, an end-of-frame is available for the frame), the status is also asserted. To enable store and forward on the receive path, clear ENET*n*_RSFL. The MAC status signal is asserted only when a complete frame is stored in the receive FIFO. When programming a non-zero value to ENET*n*_RSFL (cut-through operation) it should be greater than ENET*n*_RAEM.
ENET*n*_RAEM [RX_ALMOST_EMPTY]	When the FIFO level reaches the ENET*n*_RAEM value, and the end-of-frame has not been received, the core receive read control stops the FIFO read (and subsequently stops transferring data to the MAC client application). It continues to deliver the frame, if again more data than the threshold or the end-of-frame is available in the FIFO. Set ENET*n*_RAEM to a minimum of six.
ENET*n*_RAFL [RX_ALMOST_FULL]	When the FIFO level approaches the maximum and there is no more space remaining for at least ENETn_RAFL number of words, the MAC control logic stops writing data in the FIFO and truncates the receive frame to avoid FIFO overflow. The corresponding error status is set when the frame is delivered to the application. Set ENET*n*_RAFL to a minimum of 4.
ENET*n*_RSEM [RX_SECTION_EMPTY]	When the FIFO level reaches the ENET*n*_RSEM value, an indication is sent to the MAC transmit logic, which generates an XOFF pause frame. This indicates FIFO congestion to the remote Ethernet client. When the FIFO level goes below the value programmed in ENET*n*_RSEM, an indication is sent to the MAC transmit logic, which generates an XON pause frame. This indicates the FIFO congestion is cleared to the remote Ethernet client. Clearing ENET*n*_RSEM disables any pause frame generation.

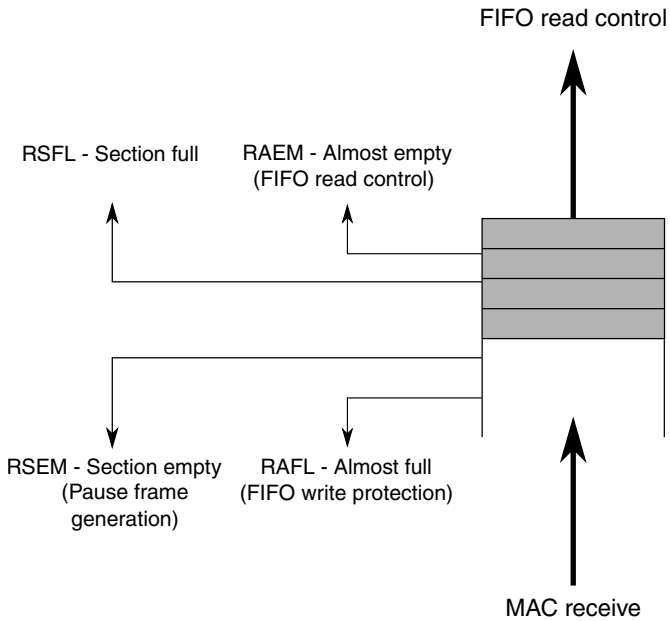


Figure 11-13. Receive FIFO overview

11.1.6.11.2 Transmit FIFO

Four programmable thresholds are available which control the core operation as described below.

Table 11-31. Transmit FIFO thresholds definition

Register	Description
ENET*n*_TAEM [TX_ALMOST _EMPTY]	When the FIFO level reaches the ENET*n*_TAEM value and no end-of-frame is available for the frame, the MAC transmit logic avoids a FIFO underflow by stopping FIFO reads and transmitting the Ethernet frame with an MII error indication. Set ENET*n*_TAEM to a minimum of 4.
ENET*n*_TAFL [TX_ALMOST _FULL]	When the FIFO level approaches the maximum, so that there is no more space for at least ENET*n*_TAFL number of words, the MAC deasserts its control signal to the application. If the application does not react on this signal, the FIFO write control logic avoids FIFO overflow by truncating the current frame and setting the error status. As a result, the frame is transmitted with an GMII/MII error indication. Set ENET*n*_TAFL to a minimum of 4. Larger values allow more latency for the application to react on the MAC control signal deassertion, before the frame is truncated. A typical setting is 8, which offers 3–4 clock cycles of latency to the application to react on the MAC control signal deassertion.
ENET*n*_TSEM [TX_SECTION _EMPTY]	When the FIFO level reaches the ENET*n*_TSEM value, a MAC status signal is deasserted to indicate that the transmit FIFO is getting full. This gives the ENET module an indication to slow or stop its write transaction to avoid a buffer overflow. This is a pure indication function to the application. It has no effect within the MAC. When ENET*n*_TSEM is 0, the signal is never deasserted.
ENET*n*_TFWR	When the FIFO level reaches the ENET*n*_TFWR value and when STRFWD is cleared, the MAC transmit control logic starts frame transmission before the end-of-frame is available in the FIFO (cut-through operation).

Table 11-31. Transmit FIFO thresholds definition

Register	Description
	If a complete frame has a size smaller than the ENET*n*_TFWR threshold, the MAC also transmits the frame to the line. To enable store and forward on the transmit path, set STRFWD. In this case, the MAC starts to transmit data only when a complete frame is stored in the transmit FIFO.

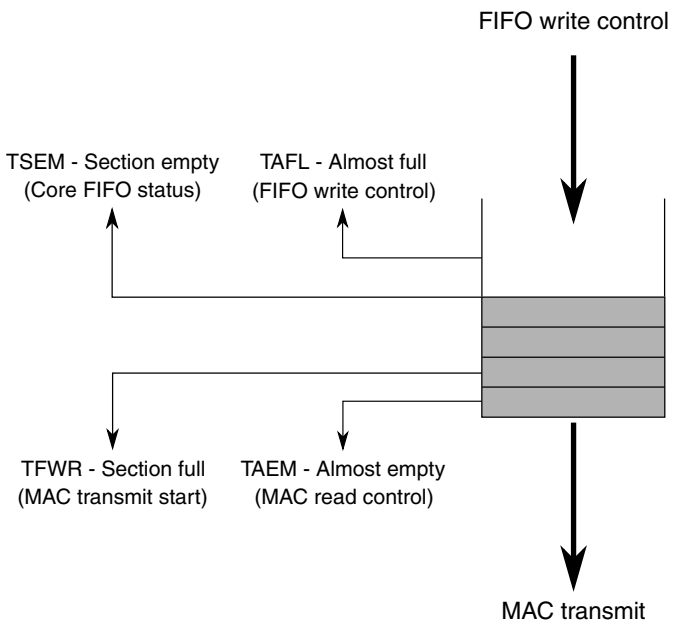


Figure 11-14. Transmit FIFO overview

11.1.6.12 Loopback options

The core implements external and internal loopback options, which are controlled by the ENET*n*_RCR register fields found here.

The core implements external and internal loopback options, which are controlled by the following ENET*n*_RCR register fields:

Table 11-32. Loopback options

Register field	Description
LOOP	Internal MII loopback. The MAC transmit is returned to the MAC receive. No data is transmitted to the external interfaces. In MII internal loopback, MII_TXCLK and MII_RXCLK must be provided with a clock signal (2.5 MHz for 10 Mbit/s, and 25 MHz for 100 Mbit/s))

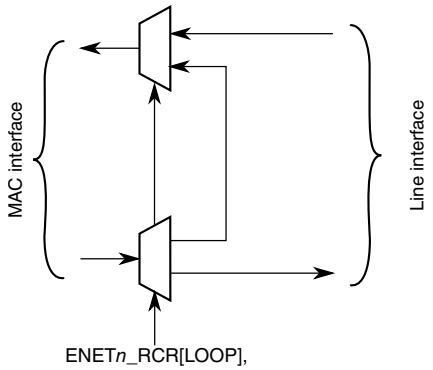


Figure 11-15. Loopback options

11.1.6.13 Legacy buffer descriptors

To support the Ethernet controller on previous Freescale devices, legacy FEC buffer descriptors are available. To enable legacy support, clear ENET*n*_ECR[1588EN].

NOTE

- Legacy buffer descriptors are used in single-ring mode, that is, when DMA*n*CFG[DMA_CLASS_EN] are zero. In multi-ring mode, legacy TxBDs are used only with the round-robin scheme.
- The legacy buffer descriptor tables show the byte order for little-endian chips. DBSWP must be set to 1 after reset to enable little-endian mode.

11.1.6.13.1 Legacy receive buffer descriptor

The following table shows the legacy FEC receive buffer descriptor. Table 11-36 contains the descriptions for each field.

Table 11-33. Legacy FEC receive buffer descriptor (RxBD)

	Byte 1								Byte 0							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	Data length															
Offset + 2	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 4	Rx data buffer pointer — low halfword															
Offset + 6	Rx data buffer pointer — high halfword															

11.1.6.13.2 Legacy transmit buffer descriptor

The following table shows the legacy FEC transmit buffer descriptor. Table 11-38 contains the descriptions for each field.

Table 11-34. Legacy FEC transmit buffer descriptor (TxBD)

	Byte 1								Byte 0							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	Data Length															
Offset + 2	R	TO1	W	TO2	L	TC	ABC[1]	—	—	—	—	—	—	—	—	—
Offset + 4	Tx Data Buffer Pointer — low halfword															
Offset + 6	Tx Data Buffer Pointer — high halfword															

1. This field is not supported by the uDMA.

11.1.6.14 Enhanced buffer descriptors

This section provides a description of the enhanced operation of the driver/uDMA via the buffer descriptors.

It is followed by a detailed description of the receive and transmit descriptor fields. To enable the enhanced features, set ENET*n*_ECR[1588EN].

NOTE

The enhanced buffer descriptor tables show the byte order for little-endian chips. DBSWP must be set to 1 after reset to enable little-endian mode.

11.1.6.14.1 Enhanced receive buffer descriptor

The following table shows the enhanced uDMA receive buffer descriptor. Table 11-36 contains the descriptions for each field.

Table 11-35. Enhanced uDMA receive buffer descriptor (RxBD)

	Byte 1								Byte 0							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	Data length															
Offset + 2	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 4	Rx data buffer pointer – low halfword															
Offset + 6	Rx data buffer pointer – high halfword															
Offset + 8	VPCP			—	—	—	—	—	—	—	ICE	PCR	—	VLAN	IPV6	FRAG
Offset + A	ME	—	—	—	—	PE	CE	UC	INT	—	—	—	—	—	—	—

Table continues on the next page...

Table 11-35. Enhanced uDMA receive buffer descriptor (RxBD) (continued)

Offset + C	Payload checksum															
Offset + E	Header length					—	—	—	Protocol type							
Offset + 10	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 12	BDU	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 14	1588 timestamp – low halfword															
Offset + 16	1588 timestamp – high halfword															
Offset + 18	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 11-36. Receive buffer descriptor field definitions

Word	Field	Description
Offset + 0	0–15 Data Length	Data length. Written by the MAC. Data length is the number of octets written by the MAC into this BD's data buffer if L is cleared (the value is equal to EMRBR), or the length of the frame including CRC if L is set. It is written by the MAC once as the BD is closed.
Offset + 2	0 E	Empty. Written by the MAC (= 0) and user (= 1). 0 The data buffer associated with this BD is filled with received data, or data reception has aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 2	2 RO1	Receive software ownership. This field is reserved for use by software. This read/write field is not modified by hardware, nor does its value affect hardware.
Offset + 2	2 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in ENET*n*_RDSR.
Offset + 2	3 RO2	Receive software ownership. This field is reserved for use by software. This read/write field is not modified by hardware, nor does its value affect hardware.
Offset + 2	4 L	Last in frame. Written by the uDMA. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
Offset + 2	5–6	Reserved, must be cleared.
Offset + 2	7 M	Miss. Written by the MAC. This field is set by the MAC for frames accepted in promiscuous mode, but flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, you can use the this field to quickly determine whether the frame was destined to this station. This field is valid only if the L and PROM bits are set. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode. The information needed for this field comes from the promiscuous_miss(ff_rx_err_stat[26]) sideband signal.
Offset + 2	8	Set if the DA is broadcast (FFFF_FFFF_FFFF).

Table continues on the next page...

Table 11-36. Receive buffer descriptor field definitions (continued)

Word	Field	Description
	BC	
Offset + 2	9 MC	Set if the DA is multicast and not BC.
Offset + 2	10 LG	Receive frame length violation. Written by the MAC. A frame length greater than RCR[MAX_FL] was recognized. This field is valid only if the L field is set. The receive data is not altered in any way unless the length exceeds TRUNC_FL bytes.
Offset + 2	11 NO	Receive non-octet aligned frame. Written by the MAC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error or a PHY error occurred. This field is valid only if the L field is set. If this field is set, the CR field is not set.
Offset + 2	12	Reserved, must be cleared.
Offset + 2	13 CR	Receive CRC or frame error. Written by the MAC. This frame contains a PHY or CRC error and is an integral number of octets in length. This field is valid only if the L field is set.
Offset + 2	14 OV	Overrun. Written by the MAC. A receive FIFO overrun occurred during frame reception. If this field is set, the other status fields, M, LG, NO, and CR, lose their normal meaning and are zero. This field is valid only if the L field is set.
Offset + 2	15 TR	Set if the receive frame is truncated (frame length >TRUNC_FL). If the TR field is set, the frame must be discarded and the other error fields must be ignored because they may be incorrect.
Offset + 4	0–15 Data buffer pointer low	Receive data buffer pointer, low halfword
0ffset + 6	0–15 Data buffer pointer high	Receive data buffer pointer, high halfword[1]
Offset + 8	0–2 VPCP	VLAN priority code point. This field is written by the uDMA to indicate the frame priority level. Valid values are from 0 (best effort) to 7 (highest). This value can be used to prioritize different classes of traffic (e.g., voice, video, data). This field is only valid if the L field is set.
Offset + 8	3–9	Reserved, must be cleared.
Offset + 8	10 ICE	IP header checksum error. This is an accelerator option. This field is written by the uDMA. Set when either a non-IP frame is received or the IP header checksum was invalid. An IP frame with less than 3 bytes of payload is considered to be an invalid IP frame. This field is only valid if the L field is set.
Offset + 8	11 PCR	Protocol checksum error. This is an accelerator option. This field is written by the uDMA. Set when the checksum of the protocol is invalid or an unknown protocol is found and checksumming could not be performed. This field is only valid if the L field is set.
Offset + 8	12	Reserved, must be cleared.
Offset + 8	13 VLAN	VLAN. This is an accelerator option. This field is written by the uDMA. It means that the frame has a VLAN tag. This field is valid only if the L field is set.
Offset + 8	14 IPV6	IPV6 Frame. This field is written by the uDMA. This field indicates that the frame has an IPv6 frame type. If this field is not set it means that an IPv4 or other protocol frame was received. This field is valid only if the L field is set.
Offset + 8	15 FRAG	IPv4 Fragment.This is an accelerator option.This field is written by the uDMA. It indicates that the frame is an IPv4 fragment frame. This field is only valid when the L field is set.

Table continues on the next page...

Table 11-36. Receive buffer descriptor field definitions (continued)

Word	Field	Description
Offset + A	0 ME	MAC error. This field is written by the uDMA. This field means that the frame stored in the system memory was received with an error (typically, a receive FIFO overflow). This field is only valid when the L field is set.
Offset + A	1–4	Reserved, must be cleared.
Offset + A	5 PE	PHY Error. This field is written by the uDMA. Set to "1"when the frame was received with an Error character on the PHY interface. The frame is invalid. This field is valid only when the L field is set.
Offset + A	6 CE	Collision. This field is written by the uDMA. Set when the frame was received with a collision detected during reception. The frame is invalid and sent to the user application. This field is valid only when the L field is set.
Offset + A	7 UC	Unicast. This field is written by the uDMA, and means that the frame is unicast. This field is valid regardless of whether the L field is set.
Offset + A	8 INT	Generate RXB/RXF interrupt. This field is set by the user to indicate that the uDMA is to generate an interrupt on the *dma_int_rxb* / *dma_int_rxfevent*.
Offset + A	9–15	Reserved, must be cleared.
Offset + C	0–15 Payload checksum	Internet payload checksum. This is an accelerator option. It is the one's complement sum of the payload section of the IP frame. The sum is calculated over all data following the IP header until the end of the IP payload. This field is valid only when the L field is set.
Offset + E	0–4 Header length	Header length. This is an accelerator option. This field is written by the uDMA. This field is the sum of 32-bit words found within the IP and its following protocol headers. If an IP datagram with an unknown protocol is found, then the value is the length of the IP header. If no IP frame or an erroneous IP header is found, the value is 0. The following values are minimum values if no header options exist in the respective headers: • ICMP/IP: 6 (5 IP header, 1 ICMP header) • UDP/IP: 7 (5 IP header, 2 UDP header) • TCP/IP: 10 (5 IP header, 5 TCP header) This field is only valid if the L field is set.
Offset + E	5–7	Reserved, must be cleared.
Offset + E	8–15 Protocol type	Protocol type. This is an accelerator option. The 8-bit protocol field found within the IP header of the frame. It is valid only when ICE is cleared. This field is valid only when the L field is set.
Offset + 10	0–15	Reserved, must be cleared.
Offset + 12	0 BDU	Last buffer descriptor update done. Indicates that the last BD data has been updated by uDMA. This field is written by the user (=0) and uDMA (=1).
Offset + 12	1–15	Reserved, must be cleared.
Offset + 14	0–15 1588 timestamp	This value is written by the uDMA. It is only valid if the L field is set.
Offset + 16		
Offset + 18 – Offset + 1E	0–15	Reserved, must be cleared.

1. The receive buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 64. The buffer must reside in memory external to the MAC. The Ethernet controller never modifies this value.

11.1.6.14.2 Enhanced transmit buffer descriptor

Table 11-37. Enhanced transmit buffer descriptor (TxBD)

	Byte 1								Byte 0							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	Data length															
Offset + 2	R	TO1	W	TO2	L	TC	—	—	—	—	—	—	—	—	—	—
Offset + 4	Tx Data Buffer Pointer – low halfword															
Offset + 6	Tx Data Buffer Pointer – high halfword															
Offset + 8	TXE	—	UE	EE	FE	LCE	OE	TSE	—	—	—	—	—	—	—	—
Offset + A	—	INT	TS	PINS	IINS	—	—	UTLT	FTYPE				—	—	—	—
Offset + C	TLT – low halfword															
Offset + E	TLT – high halfword															
Offset + 10	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 12	BDU	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 14	1588 timestamp – low halfword															
Offset + 16	1588 timestamp – high halfword															
Offset + 18	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 11-38. Enhanced transmit buffer descriptor field definitions

Word	Field	Description
Offset + 0	0–15 Data Length	Data length, written by user. Data length is the number of octets the MAC should transmit from this BD's data buffer. It is never modified by the MAC.
Offset + 2	0 R	Ready. Written by the MAC and you. 0 The data buffer associated with this BD is not ready for transmission. You are free to manipulate this BD or its associated data buffer. The MAC clears this field after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, prepared for transmission by you, has not been transmitted or currently transmits. You may write no fields of this BD after this field is set.
Offset + 2	1 TO1	Transmit software ownership. This field is reserved for software use. This read/write field is not modified by hardware and its value does not affect hardware.
Offset + 2	2 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ETDSR.

Table continues on the next page...

Table 11-38. Enhanced transmit buffer descriptor field definitions (continued)

Word	Field	Description
Offset + 2	3 TO2	Transmit software ownership. This field is reserved for use by software. This read/write field is not modified by hardware and its value does not affect hardware.
Offset + 2	4 L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame 1 The buffer is the last in the transmit frame
Offset + 2	5 TC	Transmit CRC. Written by user, and valid only when L is set. 0 End transmission immediately after the last data byte 1 Transmit the CRC sequence after the last data byte This field is valid only when the L field is set.
Offset + 2	6 ABC	Append bad CRC. **Note:** This field is not supported by the uDMA and is ignored.
Offset + 2	7–15	Reserved, must be cleared.
Offset + 4	0–15 Data buffer pointer low	Tx data buffer pointer, low halfword
Offset + 6	0–15 Data buffer pointer high	Tx data buffer pointer, high halfword. The buffer must reside in memory external to the MAC. This value is never modified by the Ethernet controller. **NOTE:** For optimal performance, make the transmit buffer pointer evenly divisible by 64.
Offset + 8	0 TXE	Transmit error occurred. This field is written by the uDMA. This field indicates that there was a transmit error of some sort reported with the frame. Effectively this field is an OR of the other error fields including UE, EE, FE, LCE, OE, and TSE. This field is valid only when the L field is set.
Offset + 8	1	Reserved, must be cleared.
Offset + 8	2 UE	Underflow error. This field is written by the uDMA. This field indicates that the MAC reported an underflow error on transmit. This field is valid only when the L field is set.
Offset + 8	3 EE	Excess Collision error. This field is written by the uDMA. This field indicates that the MAC reported an excess collision error on transmit. This field is valid only when the L field is set.
Offset + 8	4 FE	Frame with error. This field is written by the uDMA. This field indicates that the MAC reported that the uDMA reported an error when providing the packet. This field is valid only when the L field is set.
Offset + 8	5 LCE	Late collision error. This field is written by the uDMA. This field indicates that the MAC reported that there was a Late Collision on transmit. This field is valid only when the L field is set.
Offset + 8	6 OE	Overflow error. This field is written by the uDMA. This field indicates that the MAC reported that there was a FIFO overflow condition on transmit. This field is only valid when the L field is set.
Offset + 8	7 TSE	Timestamp error. This field is written by the uDMA. This field indicates that the MAC reported a different frame type then a timestamp frame. This field is valid only when the L field is set.
Offset + 8	8–15	Reserved, must be cleared.
Offset + A	0	Reserved, must be cleared.

Table continues on the next page...

Table 11-38. Enhanced transmit buffer descriptor field definitions (continued)

Word	Field	Description
Offset + A	1 INT	Generate interrupt flags. This field is written by the user. This field is valid regardless of the L field and must be the same for all EBD for a given frame. The uDMA does not update this value.
Offset + A	2 TS	Timestamp. This field is written by the user. This indicates that the uDMA is to generate a timestamp frame to the MAC. This field is valid regardless of the L field and must be the same for all EBD for the given frame. The uDMA does not update this value.
Offset + A	3 PINS	Insert protocol specific checksum. This field is written by the user. If set, the MAC's IP accelerator calculates the protocol checksum and overwrites the corresponding checksum field with the calculated value. The checksum field must be cleared by the application generating the frame. The uDMA does not update this value. This field is valid regardless of the L field and must be the same for all EBD for a given frame.
Offset + A	4 IINS	Insert IP header checksum. This field is written by the user. If set, the MAC's IP accelerator calculates the IP header checksum and overwrites the corresponding header field with the calculated value. The checksum field must be cleared by the application generating the frame. The uDMA does not update this value. This field is valid regardless of the L field and must be the same for all EBD for a given frame.
Offset + A	5–6	Reserved, must be cleared.
Offset + A	7 UTLT	Use transmit launch time. If set to 1, TLT high and low values are used to determine if frame can be transmitted. This field must only be set in the *first* BD of a frame. TLT is always used in combination with either the round-robin scheme or the credit-based shaper. TLT can be used with a single BD queue. However, at least one DMA class, DMA*n*CFG[DMA_CLASS_ EN] must be enabled. For example, to use TLT with only queue 0, enable one of the DMA*n*CFG[DMA_CLASS_EN] fields and clear both TDAR1 and TDAR2. Although this enables multi-queue, only the single queue 0 is used.
Offset + A	8–11 FTYPE	Type of frame to be transmitted. If a credit-based scheme is used, this field must match the BD ring queue. 0x0 – Non-AVB. Corresponds to ENET_TDSR. 0x1 – AVB Class A. Corresponds to ENET_TDSR1. 0x2 – AVB Class B. Corresponds to ENET_TDSR2. All other values are reserved.
Offset + A	12–15	Reserved, must be cleared.
Offset + C	0–15 TLT low	Transmit launch time low. Low two bytes of time value that specifies when frame can be transmitted.
Offset + E	0–15 TLT high	Transmit launch time high. High two bytes of time value that specifies when frame can be transmitted.
Offset + 10	0–15	Reserved, must be cleared.
Offset + 12	0 BDU	Last buffer descriptor update done. Indicates that the last BD data has been updated by uDMA. This field is written by the user (=0) and uDMA (=1).
Offset + 12	1–15	Reserved, must be cleared.
Offset + 14	0–15 1588 timestamp	This value is written by the uDMA . It is valid only when the L field is set.
Offset + 16		

Table continues on the next page...

Table 11-38. Enhanced transmit buffer descriptor field definitions (continued)

Word	Field	Description
Offset + 18–Offset + 1E	0–15	Reserved, must be cleared.

11.1.6.15 Client FIFO application interface

The FIFO interface is completely asynchronous from the Ethernet line, and the transmit and receive interface can operate at a different clock rate.

All transfers to/from the user application are handled independently of the core operation, and the core provides a simple interface to user applications based on a two-signal handshake.

11.1.6.15.1 Data structure description

The data structure defined in the following tables for the FIFO interface must be respected to ensure proper data transmission on the Ethernet line. Byte 0 is sent to and received from the line first.

Table 11-39. FIFO interface data structure

	63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Word 0	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
Word 1	Byte 15	Byte 14	Byte 13	Byte 12	Byte 11	Byte 10	Byte 9	Byte 8
...	...							

The size of a frame on the FIFO interface may not be a modulo of 64-bit.

The user application may not care about the Ethernet frame formats in full detail. It needs to provide and receive an Ethernet frame with the following structure:

- Ethernet MAC destination address
- Ethernet MAC source address
- Optional 802.1q VLAN tag (VLAN type and info field)
- Ethernet length/type field
- Payload

Frames on the FIFO interface do not contain preamble and SFD fields, which are inserted and discarded by the MAC on transmit and receive, respectively.

- On receive, CRC and frame padding can be stripped or passed through transparently.
- On transmit, padding and CRC can be provided by the user application, or appended automatically by the MAC independently for each frame. No size restrictions apply.

Note

On transmit, if ENET*n*_TCR[ADDINS] is set, bytes 6–11 of each frame can be set to any value, since the MAC overwrites the bytes with the MAC address programmed in the ENET*n*_PAUR and ENET*n*_PALR registers.

Table 11-40. FIFO interface frame format

Byte number	Field
0–5	Destination MAC address
6–11	Source MAC address
12–13	Length/type field
14–N	Payload data

VLAN-tagged frames are supported on both transmit and receive, and implement additional information (VLAN type and info).

Table 11-41. FIFO interface VLAN frame format

Byte number	Field
0–5	Destination MAC address
6–11	Source MAC address
12–15	VLAN tag and info
16–17	Length/type field
18–N	Payload data

Note

The standard defines that the LSB of the MAC address is sent/received first, while for all the other header fields — in other words, length/type, VLAN tag, VLAN info, and pause quanta — the MSB is sent/received first.

11.1.6.15.2 Data structure examples

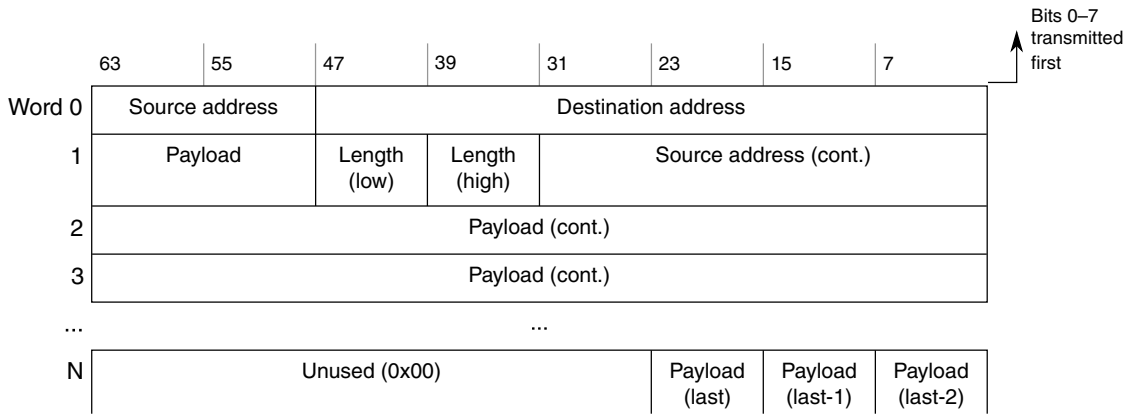


Figure 11-16. Normal Ethernet frame 64-bit mapping example

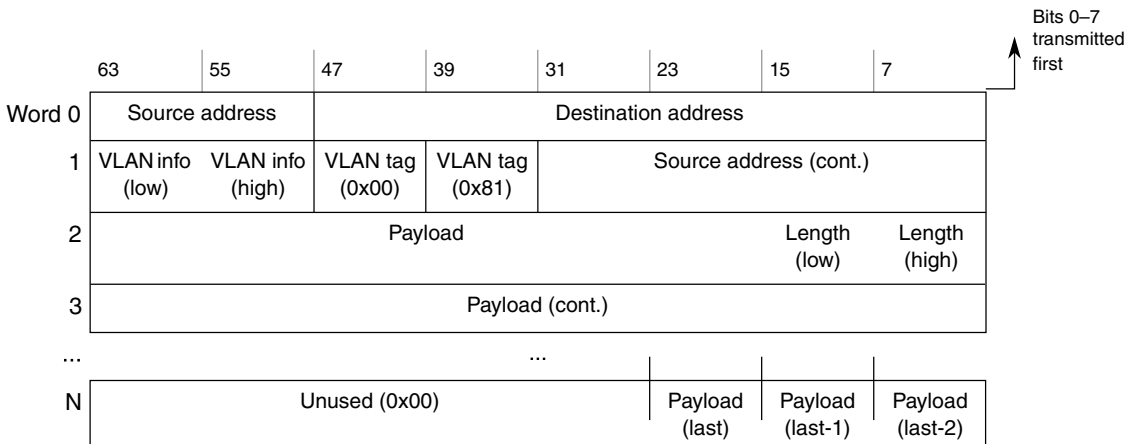


Figure 11-17. VLAN-tagged frame 64-bit mapping example

If CRC forwarding is enabled (CRCFWD = 0), the last four valid octets of the frame contain the FCS field. The non-significant bytes of the last word can have any value.

11.1.6.15.3 Frame status

A MAC layer status word and an accelerator status word is available in the receive buffer descriptor.

See Enhanced buffer descriptors for details.

The status is available with each frame with the last data of the frame.

If the frame status contains a MAC layer error (for example, CRC or length error), RxBD[ME] is also set with the last data of the frame.

11.1.6.16 FIFO protection

The following sections describe the FIFO protection mechanisms.

11.1.6.16.1 Transmit FIFO underflow

During a frame transfer, when the transmit FIFO reaches the almost empty threshold with no end-of-frame indication stored in the FIFO, the MAC logic:

- Stops reading data from the FIFO
- Asserts the MII error signal (MII_TXER) (1 in Figure 11-18) to indicate that the fragment already transferred is not valid
- Deasserts the MII transmit enable signal (MII_TXEN) to terminate the frame transfer (2)

After an underflow, when the application completes the frame transfer (3), the MAC transmit logic discards any new data available in the FIFO until the end of packet is reached (4) and sets the enhanced TxBD[UE] field.

The MAC starts to transfer data on the MII interface when the application sends a new frame with a start of frame indication (5).

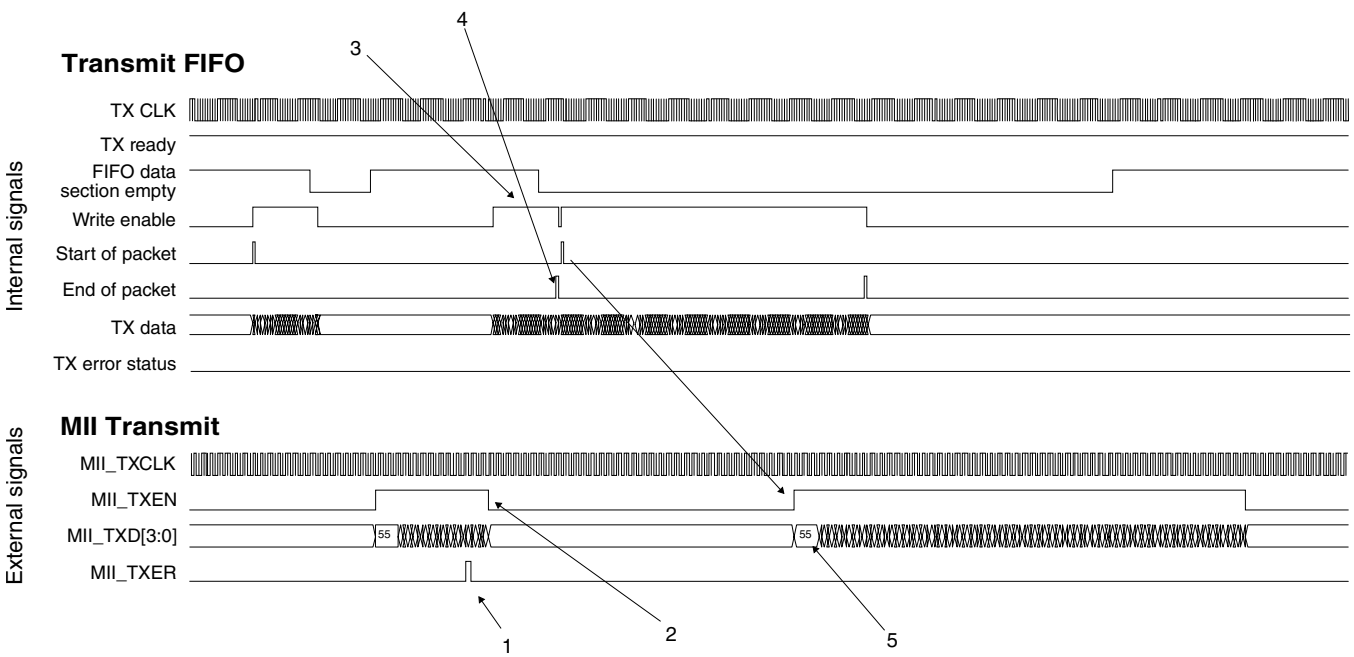


Figure 11-18. Transmit FIFO underflow protection

11.1.6.16.2 Transmit FIFO overflow

On the transmit path, when the FIFO reaches the programmable almost full threshold, the internal MAC ready signal is deasserted. The application should stop sending new data.

However, if the application keeps sending data, the transmit FIFO overflows, corrupting contents that were previously stored. The core logic sets the enhanced TxBD[OE] field for the next frame transmitted to indicate this overflow occurence.

Note

Overflow is a fatal error and must be addressed by resetting the core or clearing ENET*n*_ECR[ETHER_EN], to clear the FIFOs and prepare for normal operation again.

11.1.6.16.3 Receive FIFO overflow

During a frame reception, if the client application is not able to receive data (1), the MAC receive control truncates the incoming frame when the FIFO reaches the programmable almost-full threshold to avoid an overflow.

The frame is subsequently received on the FIFO interface with an error indication (enhanced RxBD[ME] field set together with receive end-of-packet) (2) with the truncation error status field set (3).

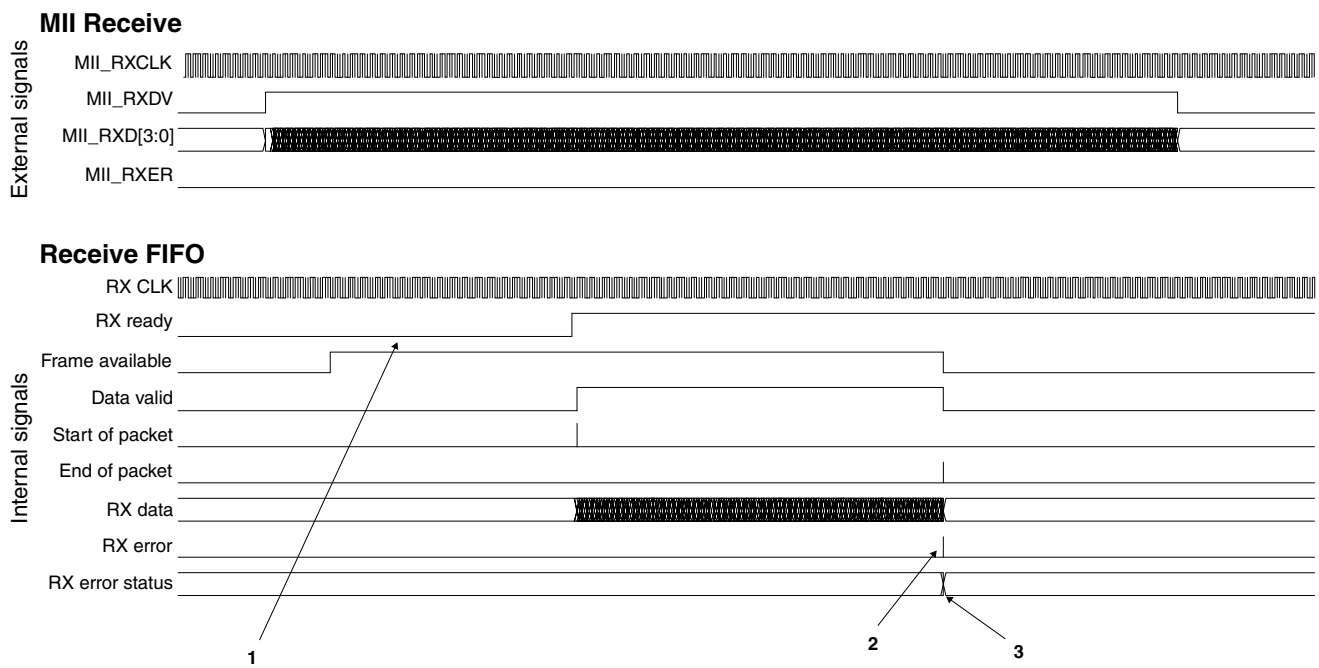


Figure 11-19. Receive FIFO overflow protection

11.1.6.17 PHY management interface

The MDIO interface is a two-wire management interface. The MDIO management interface implements a standardized method to access the PHY device management registers.

The core implements a master MDIO interface, which can be connected to up to 32 PHY devices.

11.1.6.17.1 MDIO clause 22 frame format

The core MDIO master controller communicates with the slave (PHY device) using frames that are defined in the following table.

A complete frame has a length of 64 bits made up of an optional 32-bit preamble, 14-bit command, 2-bit bus direction change, and 16-bit data. Each bit is transferred on the rising edge of the MDIO clock (MDC signal). The MDIO data signal is tri-stated between frames.

The core PHY management interface supports the standard MDIO specification (IEEE 802.3 Clause 22).

Table 11-42. MDIO clause 22 frame structure

ST	OP	PHYADR	REGADR	TA	DATA

Table 11-43. MDIO frame field descriptions

Field	Description
ST (2 bits)	Start indication field, programmed with ENET*n*_MMFR[ST] and equal to 01 for Standard MDIO (Clause 22).
OP (2 bits)	Opcode defines type of operation. Programmed with ENET*n*_MMFR[OP]. 01 Write operation 10 Read operation
PHYADR (5 bits)	Five-bit PHY device address, programmed with ENET*n*_MMFR[PA]. Up to 32 devices can be addressed.
REGADR (5 bits)	Five-bit register address, programmed with ENET*n*_MMFR[RA]. Each PHY can implement up to 32 registers.
TA (2 bits)	Turnaround time, programmed with ENET*n*_MMFR[TA]. Two bit-times are reserved for read operations to switch the data bus from write to read. The PHY device presents its register contents in the data phase and drives the bus from the second bit of the turnaround phase.
Data (16 bits)	Data, set by ENET*n*_MMFR[DATA]. Written to or read from the PHY

11.1.6.17.2 MDIO clause 45 frame format

The extended MDIO frame structure defined in IEEE 802.3 Clause 45 introduces indirect addressing. First, a write transaction to an address register is done, followed by a write or read transaction which will put the 16-bit data in the register or retrieve the register contents respectively. A preamble of 32 bits of logical ones is sent prior to every transaction. The MDIO data signal is tri-stated between frames.

The extended MDIO defines four transactions, which are determined by the two-bit opcode field.

Table 11-44. MDIO clause 45 frame structure

ST	OP	PRTAD	DEVAD	TA	ADDR/DATA

All bits are transmitted from left to right (Preamble bits first) and all fields have their Most-Significant bit sent first (leftmost in above table). The complete frame has a length of 64 bits (32-bit preamble, 14-bit command, 2-bit bus direction change, 16-bit data). Each bit is transferred with the rising edge of the MDIO clock (MDC).

The fields and transactions are summarized in the following tables.

Table 11-45. MDIO clause 45 frame field descriptions

Field	Description
ST	Start indication. Indicates the end of the preamble and start of the frame. This value is 00 for extended MDIO (Clause 45) frames.
OP	Opcode defines if a read or write operation is performed and is programmed with ENET*n*_MMFR[OP]. See Table 11-46 for more information. 00 Address write 01 Write operation 10 Read inc. operation 11 Read operation
PRTAD	The port address specifies a MDIO port. Each Port can have up to 32 devices which each can have a separate set of registers.
DEVAD	Device address. Up to 32 devices can be addressed (within a port).
TA	Turnaround time, programmed with ENET*n*_MMFR[TA]. Two bit-times are reserved for read operations to switch the data bus from write to read. The PHY device presents its register contents in the data phase and drives the bus from the second bit of the turnaround phase.
ADDR/DATA	16-bit address (for address write) or data, set by ENET*n*_MMFR[DATA], written to or read from the PHY.

Table 11-46. MDIO Clause 45 Transactions

Transaction Type	Description
Address	A write transaction to the internal address register of the device/port. The data section of the frame contains the value to be stored in the device's internal address "pointer" register for further transactions.
Write	Data write to a register. The 16 bit data will be written to the register identified by the device-internal address.
Read	Data is read from the register identified by the device-internal address.
Read inc.	Read with address postincrement. The register identified by the device-internal address is read. After this, the device-internal address is incremented. If the address register is all '1' (0xFFFF) no increment is done (i.e. increment does not wrap around).

11.1.6.17.3 MDIO clock generation

The MDC clock is generated from the internal bus clock (i.e., IPS bus clock) divided by the value programmed in ENET*n*_MSCR[MII_SPEED].

11.1.6.17.4 MDIO operation

To perform an MDIO access, set the MDIO command register (ENET*n*_MMFR) according to the description provided in MII Management Frame Register (ENETn_MMFR).

To check when the programmed access completes, read the ENET*n*_EIR[MII] field.

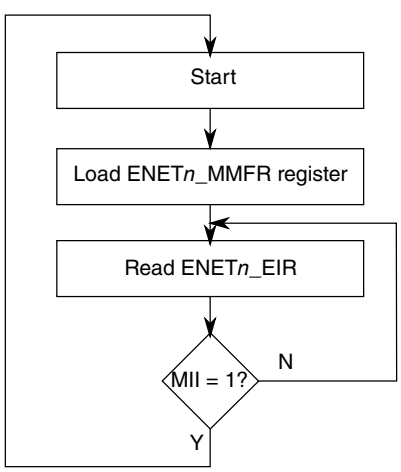


Figure 11-20. MDIO access overview

11.1.6.18 Ethernet interfaces

The following Ethernet interfaces are implemented:

- Fast Ethernet MII (Media Independent Interface)
- RMII 10/100 using interface converters/gaskets
- RGMII 10/100/1000 by way of interface converters/gaskets

The following table shows how to configure ENET registers to select each interface.

Mode	ECR[SPEED]	RCR[RMII_10T]	RCR[RMII_MODE]	RCR[RGMII_EN]
MII - 10 Mbit/s	0	—	0	0
MII - 100 Mbit/s	0	—	0	0
RMII - 10 Mbit/s	0	1	1	0
RMII - 100 Mbit/s	0	0	1	0
RGMII - 10 Mbit/s	0	1	0	1
RGMII - 100 Mbit/s	0	0	0	1
RGMII - 1000 Mbit/s	1	—	0	1

11.1.6.18.1 RMII interface

In RMII receive mode, for normal reception following assertion of CRS_DV, RXD[1:0] is 00 until the receiver determines that the receive event has a proper start-of-stream delimiter (SSD).

The preamble appears (RXD[1:0]=01) and the MACs begin capturing data following detection of SFD.

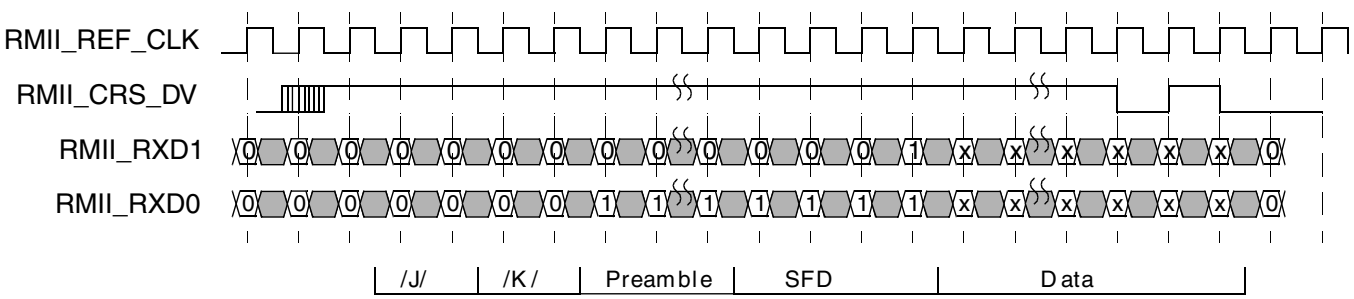


Figure 11-21. RMII receive operation

If a false carrier is detected (bad SSD), then RXD[1:0] is 10 until the end of the receive event. This is a unique pattern since a false carrier can only occur at the beginning of a packet where the preamble is decoded (RXD[1:0] = 01).

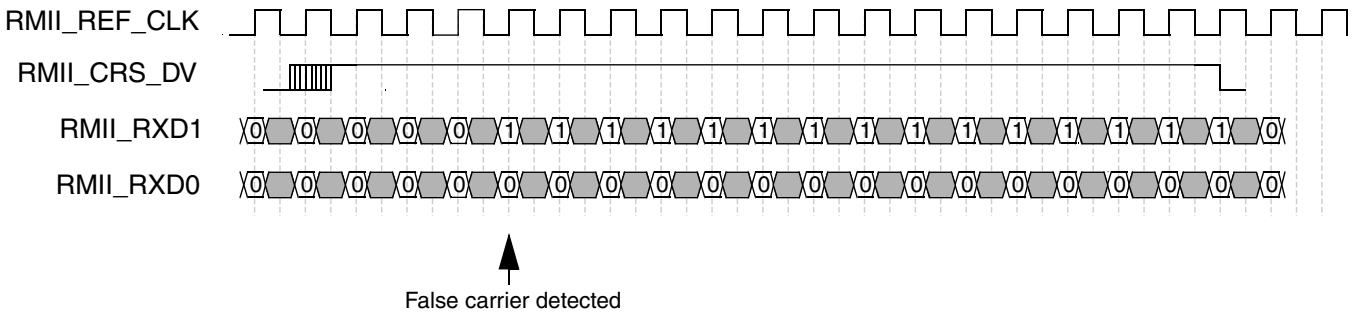


Figure 11-22. RMII receive operation with false carrier

In RMII transmit mode, TXD[1:0] provides valid data for each REF_CLK period while TXEN is asserted.

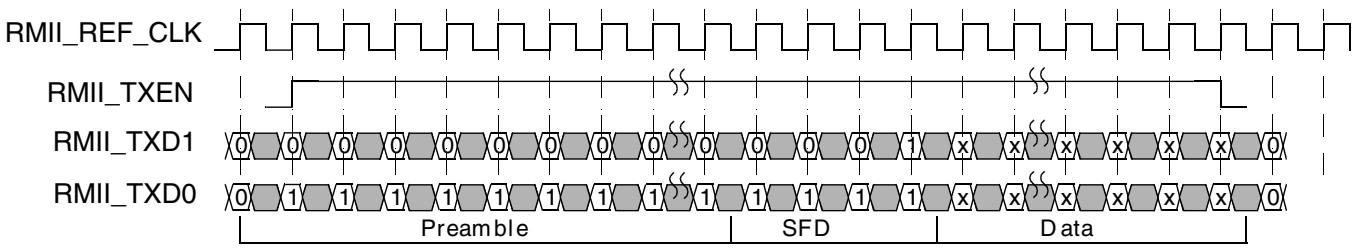


Figure 11-23. RMII transmit operation

11.1.6.18.2 RGMII interface

In RGMII modes, the data and control information is multiplexed by taking advantage of both edges of the reference clocks.

The data signals contain the lower four data bits on the rising edge and the upper four bits on the falling edge. The control signals are multiplexed into a single clock cycle using the same technique.

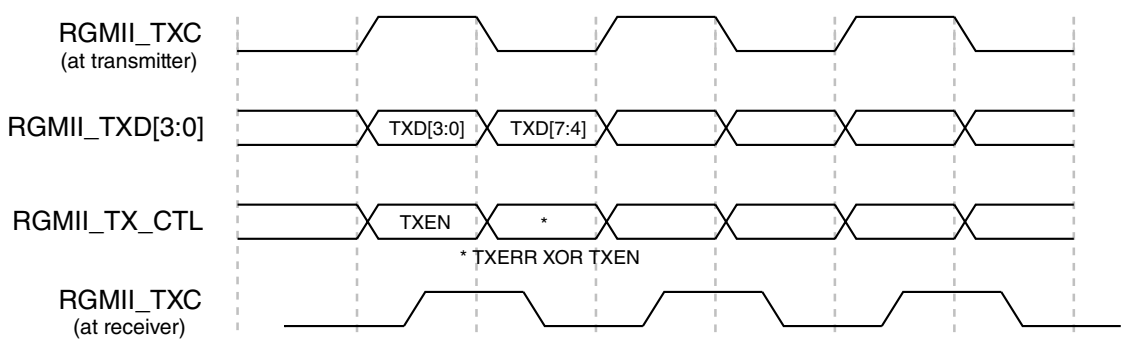


Figure 11-24. RGMII transmit operation

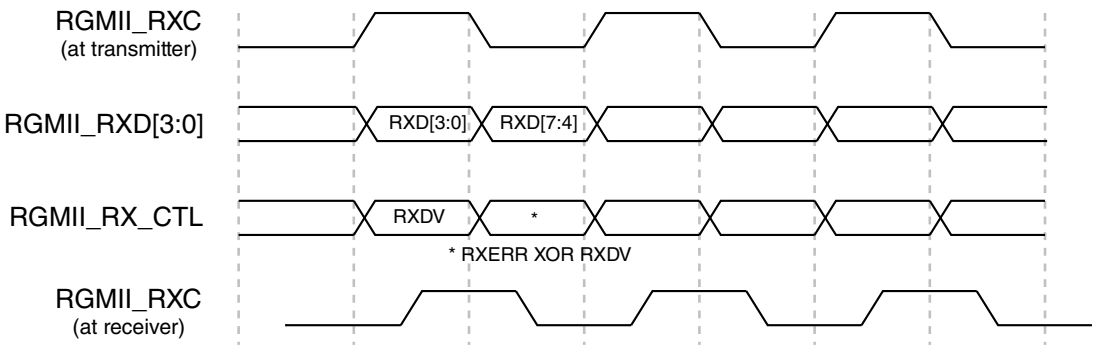


Figure 11-25. RGMII receive operation

11.1.6.18.3 MII Interface — transmit

On transmit, all data transfers are synchronous to MII_TXCLK rising edge. The MII data enable signal MII_TXEN is asserted to indicate the start of a new frame, and remains asserted until the last byte of the frame is present on the MII_TXD[3:0] bus.

Between frames, MII_TXEN remains deasserted.

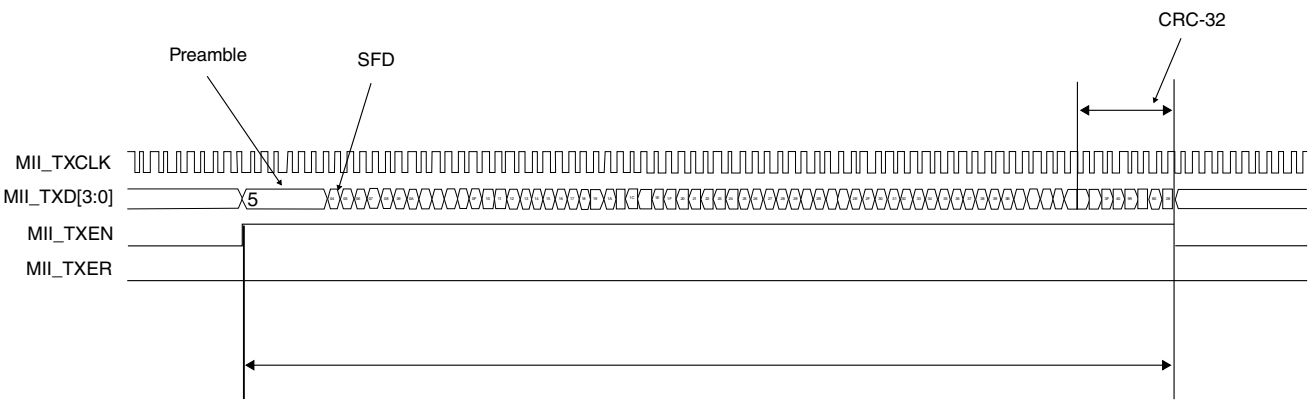


Figure 11-26. MII transmit operation

If a frame is received on the FIFO interface with an error (for example, RxBD[ME] set) the frame is subsequently transmitted with the MII_TXER error signal for one clock cycle at any time during the packet transfer.

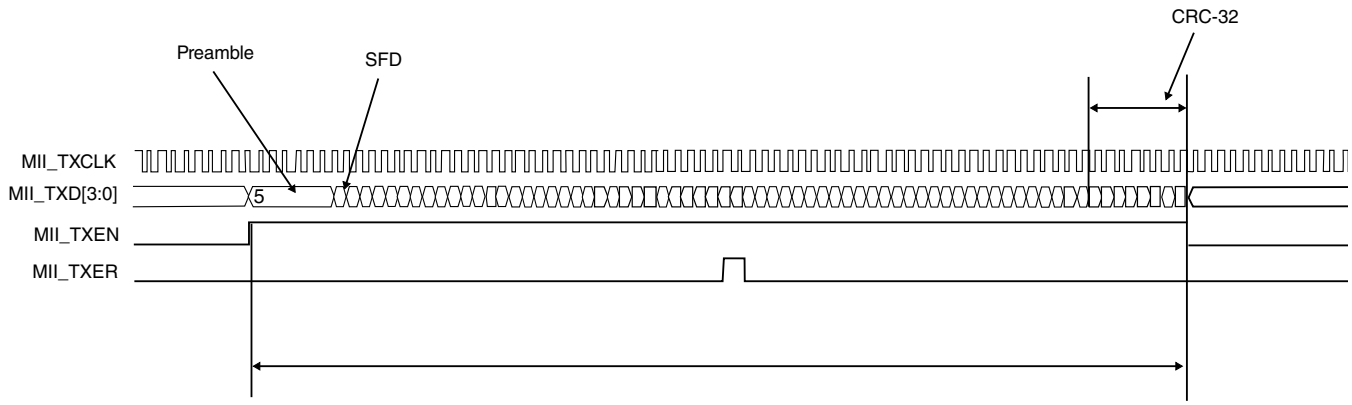


Figure 11-27. MII transmit operation — errored frame

11.1.6.18.3.1 Transmit with collision — half-duplex

When a collision is detected during a frame transmission (MII_COL asserted), the MAC stops the current transmission, sends a 32-bit jam pattern, and re-transmits the current frame.

(See Collision detection in half-duplex mode for details)

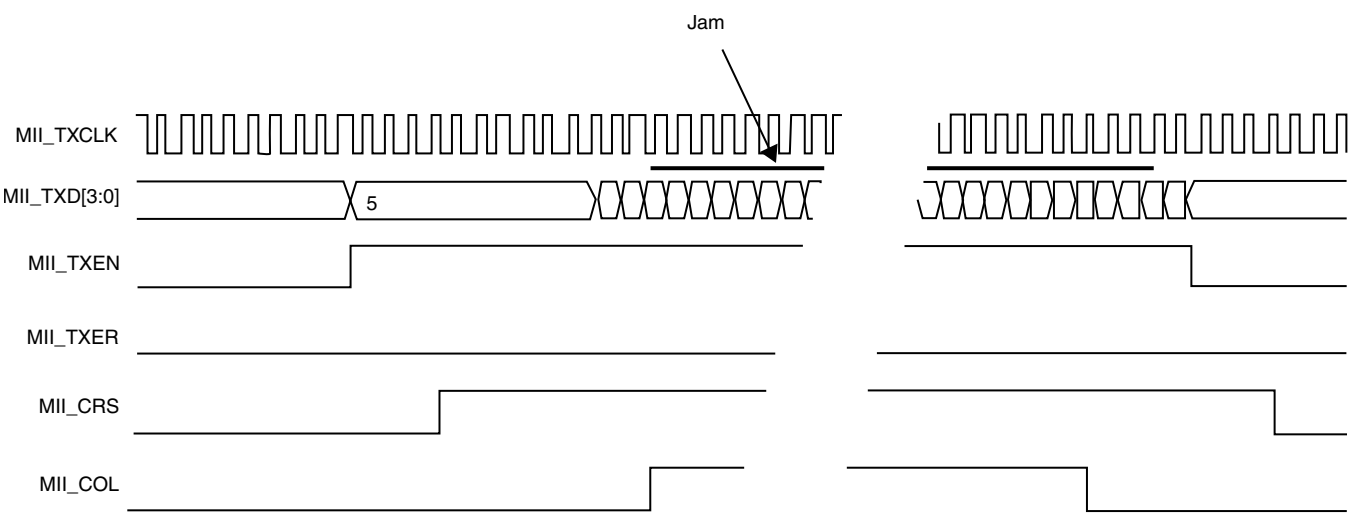


Figure 11-28. MII transmit operation — transmission with collision

11.1.6.18.4 MII interface — receive

On receive, all signals are sampled on the MII_RXCLK rising edge. The MII data enable signal, MII_RXDV, is asserted by the PHY to indicate the start of a new frame and remains asserted until the last byte of the frame is present on MII_RXD[3:0] bus.

Between frames, MII_RXDV remains deasserted.

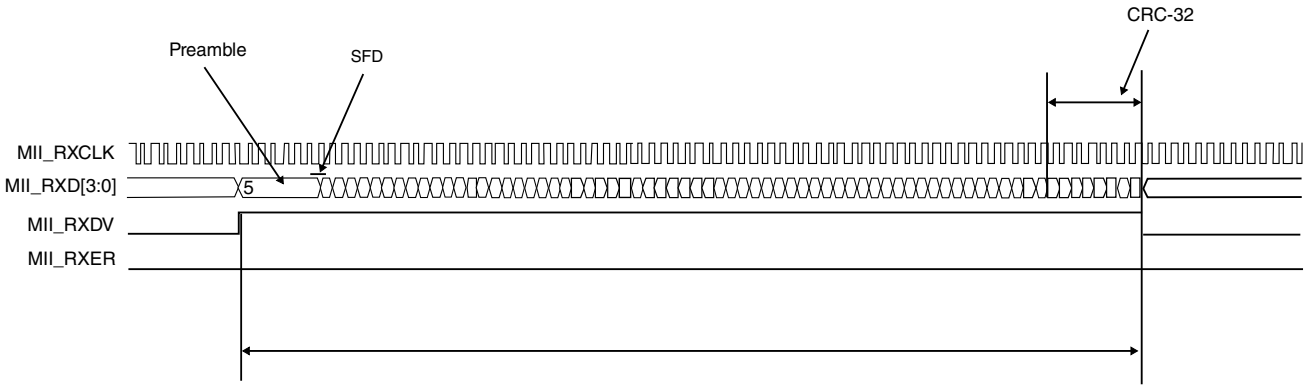


Figure 11-29. MII receive operation

If the PHY detects an error on the frame received from the line, the PHY asserts the MII error signal, MII_RXER, for at least one clock cycle at any time during the packet transfer.

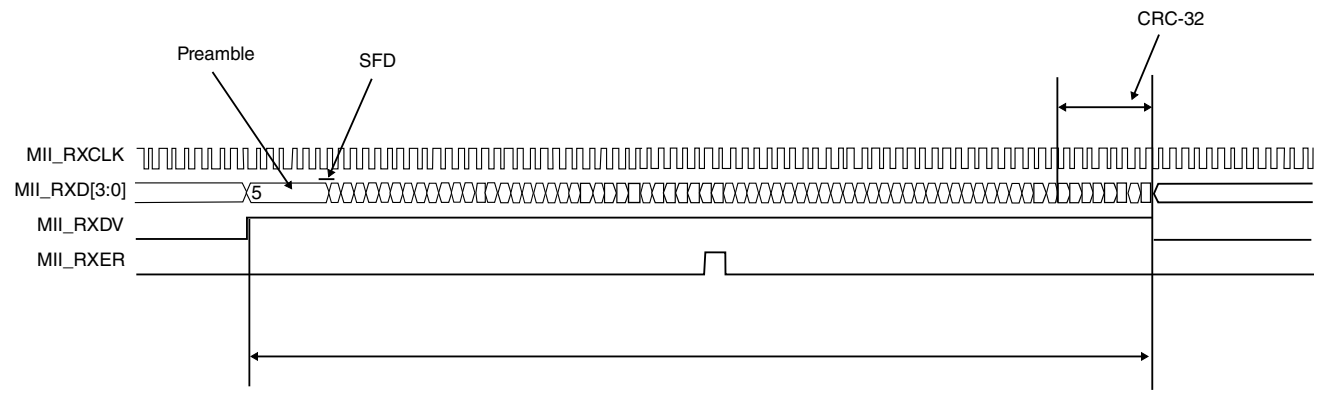


Figure 11-30. MII receive operation — errored frame

A frame received on the MII interface with a PHY error indication is subsequently transferred on the FIFO interface with RxBD[ME] set.

11.1.6.19 AVB configuration

The following steps give an example of how to initialize the ENET module for AVB.

1. Set up ENET_QOS:
   - Set TX_SCHEME to 000b, credit-based scheme.
   - Set RX_FLUSH0 to 1, enable RX flush for ring 0.
2. Set up TX BD ring and RX BD ring for each queue, 0-2.
   - Program ENET_MRBR, ENET_TDSR, and ENET_RDSR.
   - Program ENET_MRBR1, ENET_TDSR1, and ENET_RDSR1.

- Program ENET_MRBR2, ENET_TDSR2, and ENET_RDSR2.
- Program each TX and RX BD ring queue for all classes used in memory.

NOTE

If using credit-based scheme, ensure that enhanced transmit buffer descriptor FTYPE field matches BD ring queue, for example:
- FTYPE = 0h corresponds to ENET_TDSR
- FTYPE = 1h corresponds to ENET_TDSR1
- FTYPE = 2h corresponds to ENET_TDSR2

3. Program ENET_DMA1CFG and ENET_DMA2CFG for class 1 and class 2 corresponding to BD ring queue 1 and 2. DMA_CLASS_EN must be set to one for that class to be enabled. See DMA Class Based Configuration (ENET_DMA*n*CFG) for information on how to program IDLE_SLOPE.
4. Program ENET_RCMR1 and ENET_RCMR2 for class 1 and class 2 matching for receive.

NOTE

Even if a match occurs, if ENET_DMA*n*CFG[DMA_CLASS_EN] is zero for the corresponding class, the RX frame will be automatically forwarded to BD ring queue 0.

5. Program the other ENET registers according to application requirements.
6. Program ENET_RDAR, ENET_RDAR1, ENET_RDAR2, ENET_TDAR, ENET_TDAR1, and ENET_TDAR2 to 0100_0000h according to the classes used.
7. Set ENET_ECR[ETHEREN] to one.

11.1.6.20 Interrupt coalescence

The purpose of the interrupt coalescing is to reduce the number of interrupts generated by the MAC so as to reduce the CPU loading.

To facilitate this interrupt coalescing for each queue, these registers are available with the same control and configuration fields.

- Transmit Interrupt Coalescing Register (ENET_TXIC*n*) where n=0,1,2 for queue/class 0,1,2.
- Receive Interrupt Coalescing Register (ENET_RXIC*n*) where n=0,1,2 for queue/class 0,1,2.

When coalescing is enabled by asserting the corresponding ICEN field and such interrupt is also enabled by the corresponding interrupt mask of the EIMR register, the MAC generates an interrupt when the threshold number of frames is reached (defined by ICFT) or when the threshold timer expires (defined by ICTT).

When coalescing is disabled by de-asserting ICEN, but interrupt is enabled by the corresponding interrupt mask of the EIMR register, the MAC generates an interrupt as they are received without using coalescing. Interrupt coalescing is done for each transmit and receive queue/class independently.

11.1.6.20.1 Interrupt coalescence setup

Interrupt coalescence supports both legacy and enhanced BDs. The following guidelines are recommended when setting up interrupt coalescence.

- When the MAC is configured for enhanced (IEEE 1588) mode, that is, enhanced BDs:
  - Set the INT bit in the enhanced received buffer descriptor to one.
  - Set the INT bit in the enhanced transmit buffer descriptor(s) to one.
- Clear the RXB, RXB1, and RXB2 fields in the EIMR register.
- Clear the TXB, TXB1, and TXB2 fields in the EIMR register.

11.1.6.20.2 Updating the frame count threshold on-the-fly

To update the ICFT field in the RXIC*n* and TXIC*n* registers:

1. Disable interrupt coalescence by clearing the appropriate ICEN field. This will allow the internal interrupt coalescence counter to reset to zero.

NOTE

When disabling interrupt coalescence, if an interrupt event is pending, that is, the interrupt counter is not zero, then an interrupt will occur.

2. Write the new threshold value to the ICFT field.
3. Set ICEN to one.

NOTE

The ICFT field can be updated on-the-fly without disabling the ICEN field. The hardware interrupt will continue and there is a possibility that an interrupt will occur depending on the state of the hardware counter and the previous ICFT value.

11.1.6.20.3 Updating the timer threshold on-the-fly

To update the ICTT field in the RXIC*n* and TXIC*n* registers:

1. Disable interrupt coalescence by clearing the appropriate ICEN field. This will allow the internal interrupt coalescence counter to reset to zero.

NOTE

When disabling interrupt coalescence, if an interrupt event is pending, that is, the interrupt counter is not zero, then an interrupt will occur.

2. Write the new timer value to the ICTT field.
3. Set ICEN to one.

11.2 Subscriber Identification Module (SIM)

11.2.1 SIM Overview

The Subscriber Identification Module (SIM) is designed to facilitate communication to SIM cards or Eurochip pre-paid phone cards, and compatible with ISO/IEC 7816-3 standards. See the SIM block diagram in the figure below. The SIM module has one port that can be used to interface with the various cards. The interface with the Micro Controller Unit (MCU) is a 32-bit connection as described in the reference document IP Bus Specification.

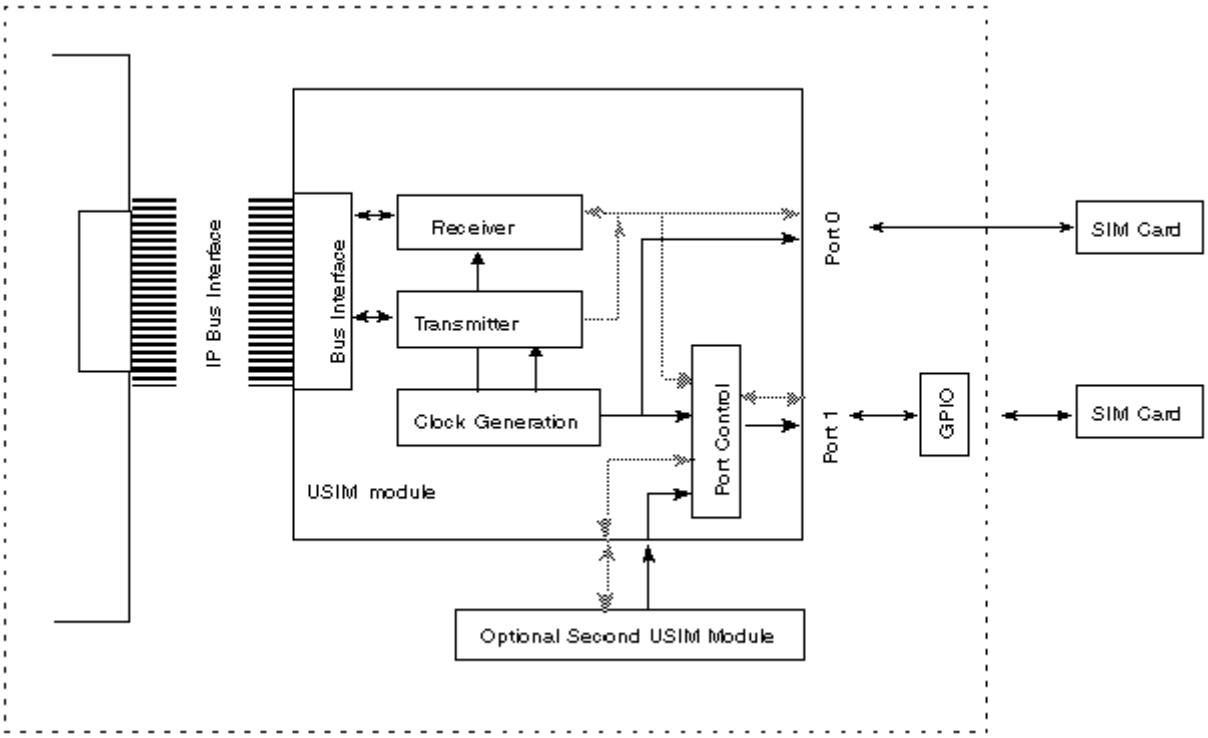


Figure 11-31. SIM Block Diagram

11.2.1.1 Features

The SIM design is summarized in the following sections:

- SIM Bus Interface Overview
- SIM Clock Generator Overview
- SIM Transmitter Overview
- SIM Receiver Overview
- SIM Port Control Overview
- SIM General Purpose Counter Overview
- SIM LRC Block Overview
- SIM CRC Block Overview

11.2.1.2 Modes of Operation

The SIM module I/O interface can be operated in one of three modes of operation summarized below.

- Two wire interface. In this mode both the IC pin RX and TX are used to interface to the SmartCard. This is activated by resetting the 3VOLTx bit in the port control register to a "0".
- External one wire interface. In this mode the IC pins RX and TX are tied together external to the IC and routed to the SmartCard. The 3VOLT bit in the port control register is reset to a "0" and the OD_Px bit in the OD_CONFIG register is set to a "1". For this interface to work properly the IC pin (RX-TX) must be pulled high by a resistor. The value should be selected small enough to give a fast enough rise time.
- Internal one wire interface. In this mode the IC pin TX is routed to the SmartCard. The receive pin RX is connected to the TX pin internal to the IC. The 3VOLTx bit in the port control register is reset to a "1" and the OD_Px bit in the OD_CONFIG register is set to a "1". For this interface to work properly the IC pin TX must be pulled high by a resistor. The value should be selected small enough to give a fast enough rise time.

NOTE

For the internal one wire interface to work, the bond pad must be capable of simultaneous input and output. (The input path should not be disabled when the output is enabled.)

11.2.1.3 SIM Bus Interface Overview

The SIM module is designed with a separate block that encompasses the interface to the IP bus. The bus interface is built in such a way as to be easily modified to other bus definitions. The SIM module is to be considered a 32-bit peripheral. To avoid endian issues with register access, all read/writes should be done as word access (32-bit). The bus interface has the following features:

- Peripheral Address/chip select decoding
- Illegal Address generation
- Dynamic wait state generation (tied inactive)
- Register organization
- Register bit implementations

See the table below for a summary of SIM top level interrupts.

Table 11-47. SIM Top Level Interrupt Summary

Flag	Interrupt Sources	Interrupt Masks	Description
SIMIRQ_N	XTE, OEF, SDI0, SDI1, TFO, GPCNT, BWT, BGT, CWT, RTE	XTM, OIM, SDIM0, SDIM1, TFOM, GPCNTM, BWTM, BGTM, CWTM, RTM	SIM General Interrupt

Table continues on the next page...

Table 11-47. SIM Top Level Interrupt Summary (continued)

Flag	Interrupt Sources	Interrupt Masks	Description
SIMDAT_N	TC, ETC, TFE, RDRF, TDTF, RFE	TCIM, ETCIM, TFEIM, RIM, TDTFM,RFEM	SIM Data Interrupt

11.2.1.4 SIM Clock Generator Overview

The SIM module contains a block designed specifically for generating the clocks used internal to the SIM module, and the clocks provided to the SIM cards. The clock generator has the following features:

- Programmable Divisor (CLK_PRESCALER[7:0]) for SIM card clock generation (45-55% duty cycle except for divider by 3, 5, 7, 9)
- Receive clock generation (/1, /2, /4, /8, /16, /32, /31) from first stage
- Transmit clock generation (/12, /8) from second stage
- Programmable divisor for receive clock generation
- There are no interrupt sources generated by the SIM clock generator block

11.2.1.5 SIM Transmitter Overview

The SIM transmitter block contains the transmit state machine, transmit shift register, and transmit FIFO. The transmitter has the following features.

- 16 bytes deep transmit FIFO
- Automatic NACK generation on parity and overrun errors
- Hardware data format support (inverse convention or direct convention)
- Retransmission of data upon SIM card NACK request with programmable maximum threshold of retransmissions
- Programmable guard time between transmitted bytes
- Six interrupt sources (see the table below)

Table 11-48. SIM Transmitter Interrupt Summary

Flag	Flag Register	Mask	Mask Register	Description
TC	XMT_STATUS	TCIM	INT_MASK	Transmit Complete
ETC	XMT_STATUS	ETCIM	INT_MASK	Early Transmit Complete
TFE	XMT_STATUS	TFEIM	INT_MASK	Transmit FIFO Empty
XTE	XMT_STATUS	XTM	INT_MASK	Transmit Threshold Error
TFO	XMT_STATUS	TFOM	INT_MASK	Transmit FIFO Overfill Error
TDTF	XMT_STATUS	TDTFM	INT_MASK	Transmit Data Threshold Flag

11.2.1.6 SIM Receiver Overview

The SIM receiver block contains the receive state machine, receive FIFO, and control logic. The receiver has the following features.

- 288 bytes deep receive FIFO
- Decoding of initial character mode for setting data format
- Hardware data format support (inverse convention or direct convention)
- NACK detection
- Eleven ETU character support
- Character Wait Time Counter
- Six interrupt sources (see the table below)

Table 11-49. SIM Receiver Interrupt Summary

Flag	Flag register	Mask	Mask register	Description
BGT	RCV_STATUS	BGTM	INT_MASK	Block Guard Time flag
BWT	RCV_STATUS	BWTM	INT_MASK	Block Wait Time flag
RTE	RCV_STATUS	RTM	INT_MASK	Receive Nack threshold
CWT	RCV_STATUS	CWTM	INT_MASK	Character Wait Time flag
OEF	RCV_STATUS	OIM	INT_MASK	Overrun Error Flag
RDRF	RCV_STATUS	RIM	INT_MASK	Receive Data Register Full
RFE	RCV_STATUS	RFEM	INT_MASK	Receive Fifo Empty

11.2.1.7 SIM Port Control Overview

The SIM module supports numerous port control functions necessary for SIM card interaction. Each of the two SIM card ports has the following I/O: CLK, $\overline{\text{RST}}$, DATA_RCV, DATA_XMT, SIMPD (SIM presence detect), and SVEN (SIM V_{CC} enable). The following list gives a number of the important features of the port logic.

- Open-drain or push pull DATA_XMT pin
- Port 1, port 0, ore alternate port selection
- SIM card presence detect with interrupt capability
- Sleep mode wake-up via SIM card presence detect interrupt
- Manual control of all SIM card interface signals
- Automatic power down of port logic on SIM card presence detect
- Two interrupt sources. See Table 11-50.

Table 11-50. SIM Card Detect Interrupts

Flag	Flag register	Mask	Mask Register	Description
SDI1	PORT1_DETECT	SDIM1	PORT1_DETECT	SIM Detect Interrupt for port 1
SDI0	PORT0_DETECT	SDIM0	PORT0_DETECT	SIM Detect Interrupt for port 0

11.2.1.8 SIM General Purpose Counter Overview

The SIM module provides a 16-bit counter that can be configured to count using clock sources from the card clock, receive clock, or transmitter clock (ETU Clock). A programmable 16-bit register is provided to compare with the counter for interrupt generation:

- Selectable clock source
- 16-bit counter and comparator
- One Interrupt source (see the table below)

Table 11-51. SIM General Purpose Counter Interrupts

Flag	Flag Register	Mask	Mask Register	Description
GPCNT	XMT_STATUS	GPCNTM	INT_MASK	GPCNT Comparator Interrupt

11.2.1.9 SIM LRC Block Overview

The SIM module contains a block designed to generate Linear Redundancy Checking (LRC) information for both received and transmitted characters. The LRC block has the following features:

- 8-bit LRC value
- Valid LRC Detector
- There are no interrupt sources generated by the SIM LRC block

11.2.1.10 SIM CRC Block Overview

The SIM module contains a block designed to generate Cyclic Redundancy Checking (CRC) information for both received and transmitted characters. The CRC block has the following features:

- 16-bit CRC value
- 16-bit CRC Polynomial Generator

- Valid CRC Detector
- There are no interrupt sources generated by the >SIM CRC block

11.2.2 External Signal Description

11.2.2.1 External Signals Overview

See the table below for summary of the SIM module signals.

Table 11-52. Signal Properties

Name	Port	Function	Reset State	Pull-up
External Signals				
SIM_CLK	O	Clock for the smartcard on port0	0	Active
SIM_RST_B	O	Reset signal for port0	0	Active
SIM_SVEN	O	Vcc enable signal for port0	0	Active
SIM_TRXD	O	Transmit/receive data signal for port0	0	Active/Passive
SIM_PD	I	Card insertion detect signal for port0	-	-
sim_pin_sclk1	O	Clock for the smartcard on port1	0	Active
sim_pin_srst1	O	Reset signal for port1	0	Active
sim_pin_sven1	O	Vcc enable signal for port1	0	Active
sim_pin_data1_tx_out	O	Transmit/receive data signal for port1	0	Active/Passive
pin_sim_simpd1	I	Card insertion detect signal for port1	-	-
sim_rcvd1_io	I	I/O Receive data signal for port1	-	-
Module Interrupts				
ipi_int_sim_ipb_int	O	Active high SIM Interrupt. Composed of OEF, XTE, SDI1, and SDI0.	0	-
ipi_int_sim_data_irq	O	Active high SIM Data Interrupt. Composed of TC, ETC, TFE, and RDRF.	0	-
Module DMA Requests				
ipd_sim_tx_dmareq_b	O	Active low. Transmitter DMA request.	1	-
ipd_sim_rx_dmareq_b	O	Active low. Receiver DMA request.	1	-

11.2.2.2 Detailed Signal Descriptions

11.2.2.2.1 SIM_CLK

The SIM_CLK is an output from the chip to the SmartCard. This signal is the clock that the SIM module provides for the SmartCard. Typical frequencies are 1 MHz to 5 MHz. This clock is 372 times the data rate that is on pin SIM_TRXD. There is no required timing relationship between this clock signal and any of the other data signals. This is because of the asynchronous nature of the protocol.

11.2.2.2.2 SIM_RST_B

The SIM_RST_B is the reset signal from the SIM to the SmartCard.

11.2.2.2.3 SIM_SVEN

The SIM_SVEN is the SmartCard power supply enable control signal.

11.2.2.2.4 SIM_TRXD

The SIM_TRXD is the data transmitted/received from the SIM module to the SmartCard. In a 1-wire mode of operating, this output port must be bidirectional with a pull-up resister off chip.

11.2.2.2.5 SIM_PD

The SIM_PD is the SmartCard insertion detect.

11.2.3 SIM Functional Description

To best describe the organization of the SIM module from a user's point of view, it is instructive to view the SIM at a number of different levels of hierarchy. See Figure 11-31 for illustration of the organization of SIM and connection of the signals to the available serial port.

The SIM module is essentially a standard UART with some special provisions made for SIM card communication. The SIM consists of the following main parts:

- IP bus interface
- Bus interface
- Clock generator
- Transmitter
- Receiver
- General purpose counter

- LRC blocks
- CRC blocks

See the figure below for an illustration of the block diagram in detail, specifically the main parts.

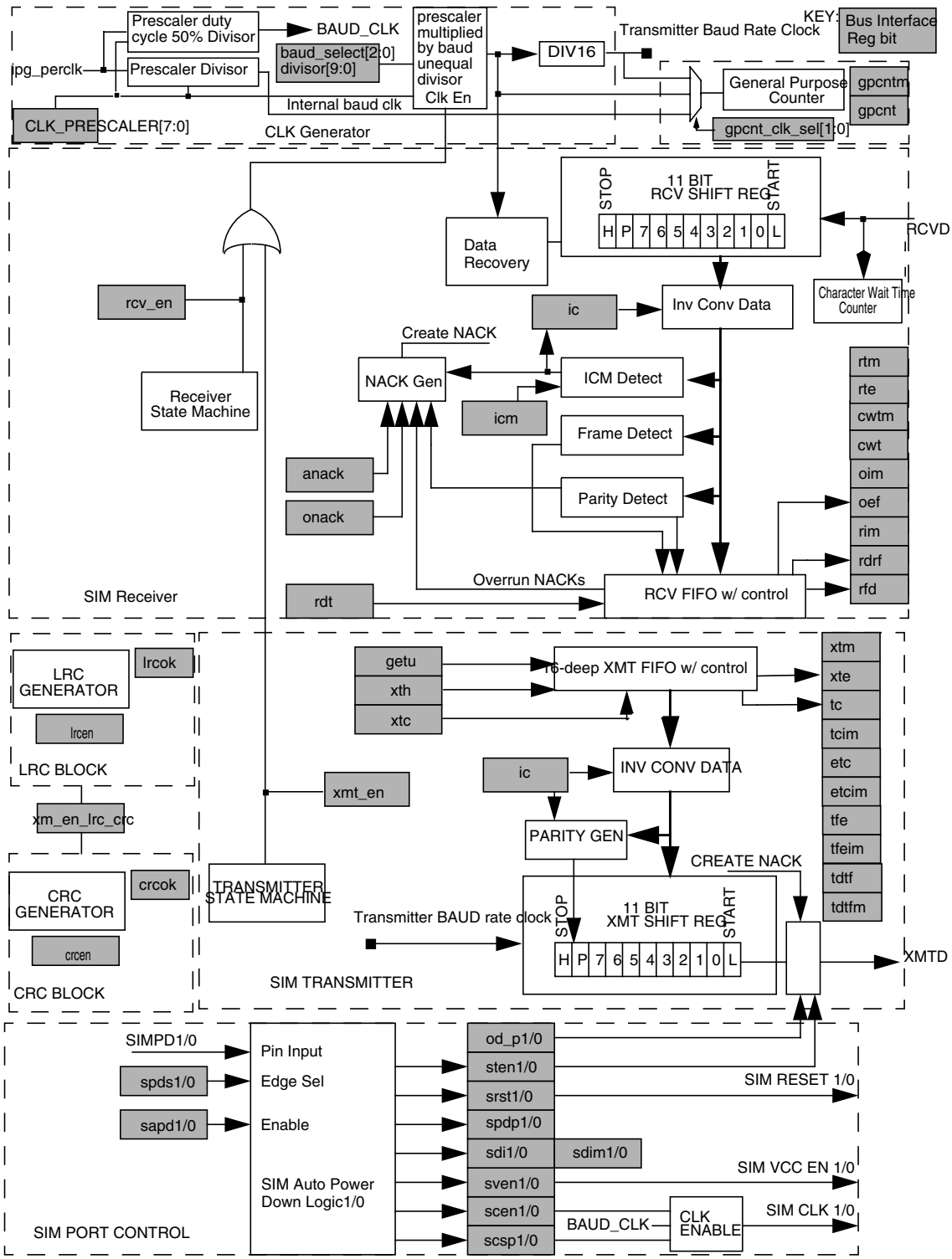


Figure 11-32. SIM Detailed Block Diagram

11.2.3.1 SIM Bus Interface

The IM Bus interface block has been designed to enable the SIM module to be easily ported to other MCU cores. The bus interface block contains all of the logic that uses the bus interface signals. See the figure below.

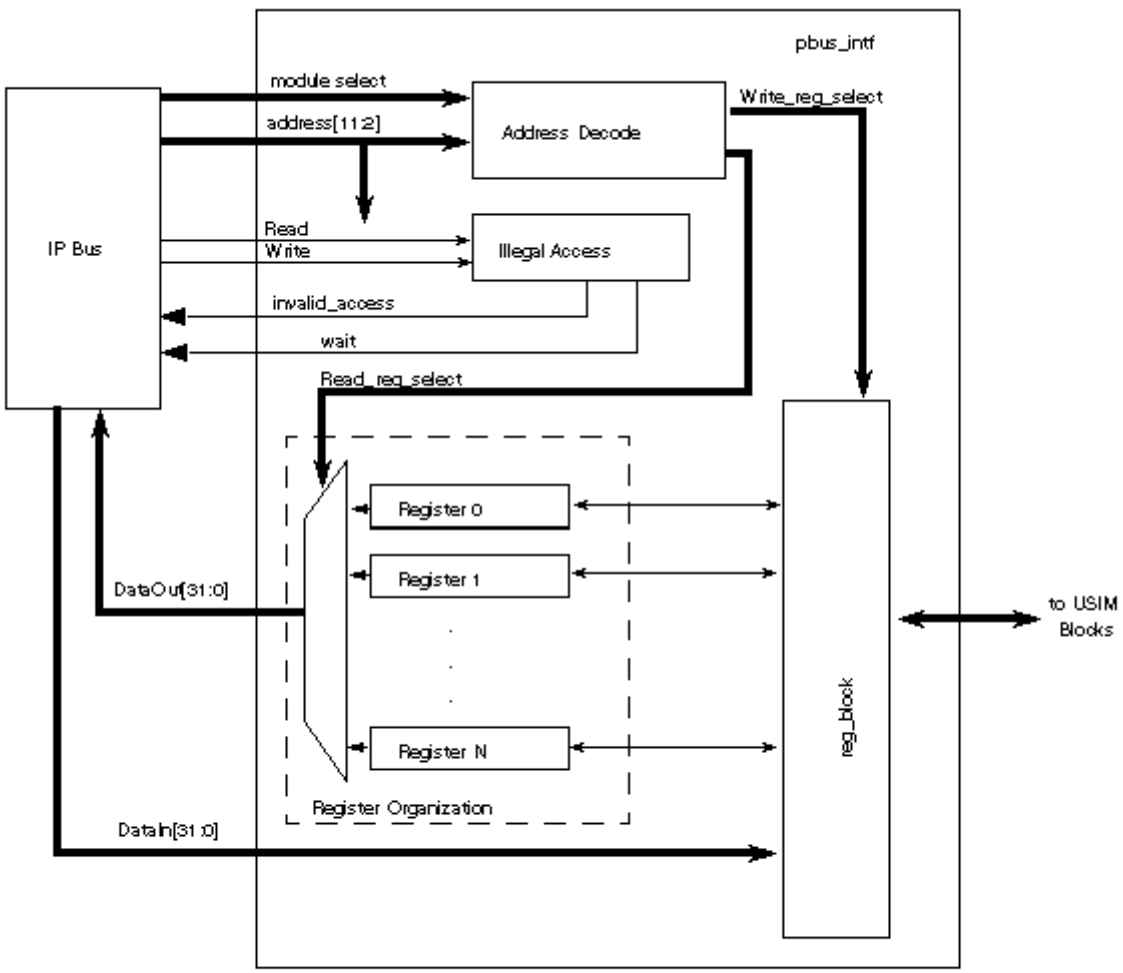


Figure 11-33. SIM Bus Interface

The bus interface block is responsible for peripheral bus address decoding, illegal access detection, dynamic wait-state requests, register organization, and register bit implementations. The address decoding uses the module address bus input along with the module select output of the IP bus to decode if the SIM module is being addressed. This decoding of these signals is done asynchronously. The output of the address decode is used with the read/write signals from the IP bus to determine the action requested by the bus master. If the read signal is active, the data_out bus will be driven with the register

selected by the address decoding. If the write signal is active, the data_in bus will be latched into the register selected by the address decoder at the rising edge of the data_strobe signal.

The illegal access detection is implemented similarly to the address decoding. If the module_select signal becomes active while the address inputs are pointing to an unimplemented address, the invalid signal is asserted asynchronously to the bus master. The invalid signal can also be generated under certain register access conditions such as writing to a full transmit FIFO, or writing to a read-only register. See the figure below.

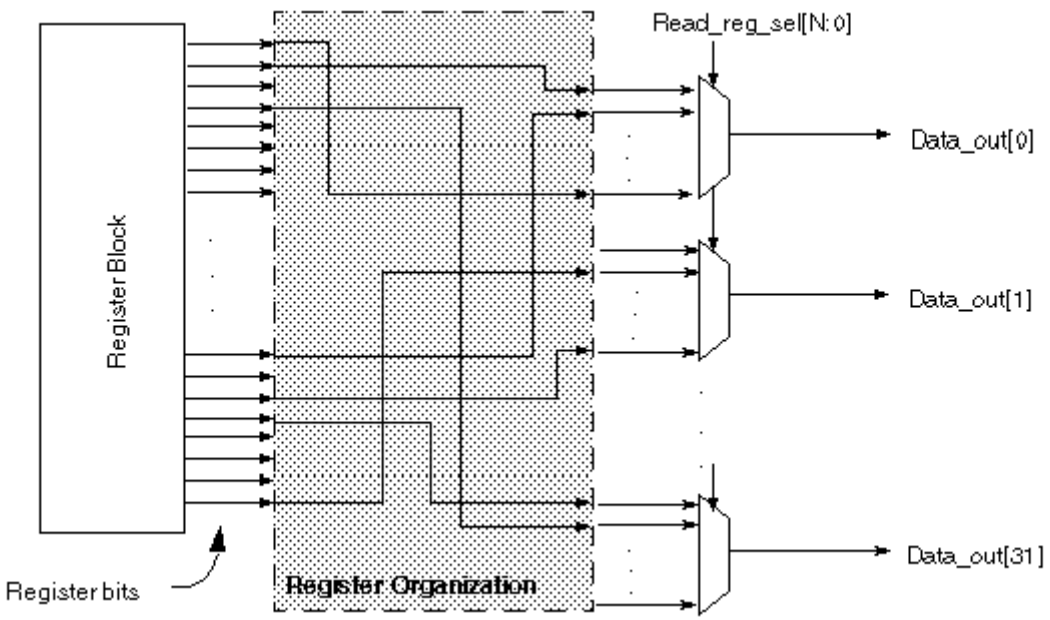


Figure 11-34. SIM Read Registers

The register bit implementations are done inside the reg_block sub-module. This block accepts the decoded read and write signals for the register bits generated in the address decoder. The data_strobe output is used as the clock input to the register bits in order to clock in the new data during a write access. The write_reg_sel signals are used to gate the write action with the address decoding. The clearing mechanisms for status bits are implemented as write-one-to-clear. See the figure below.

.

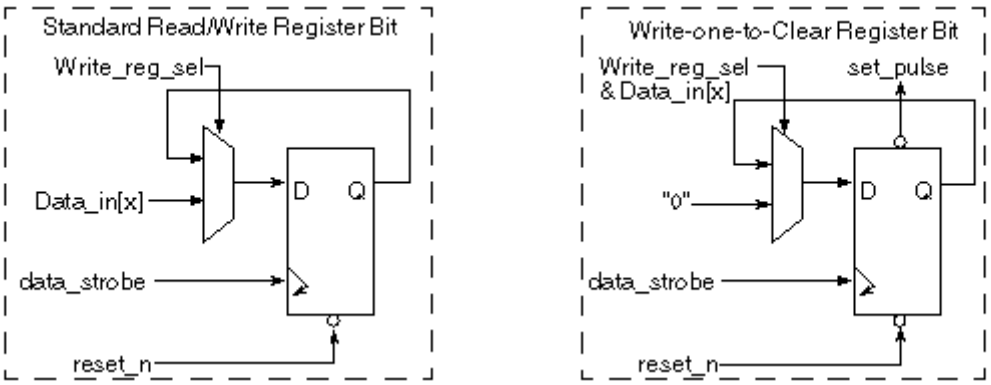


Figure 11-35. Register Bit Diagram

In the "write-one-to-clear" register bit example, "set_pulse" represents a pulsed signal generated in the data_strobe clock domain to set the register output.

11.2.3.2 SIM Clocking

The SIM has three clock inputs:

- ipg_clk-read/write clock; the frequency is half of platform_clk
- ipg_per_clk-reference frequency input clock to generate the SIM card clock and other internal clocks; the frequency is platform_clk divided by 8
- ckil-low frequency clock (32 KHz) used by the auto power down hardware circuit in the SIM. It is generated by dividing the platform_clk by a division factor which is programmed in the SIMCLKCR register in the global utilities. The default value of the divider is 0x3D09, which gives a 32 KHz clock from a platform_clk of 500 MHz.

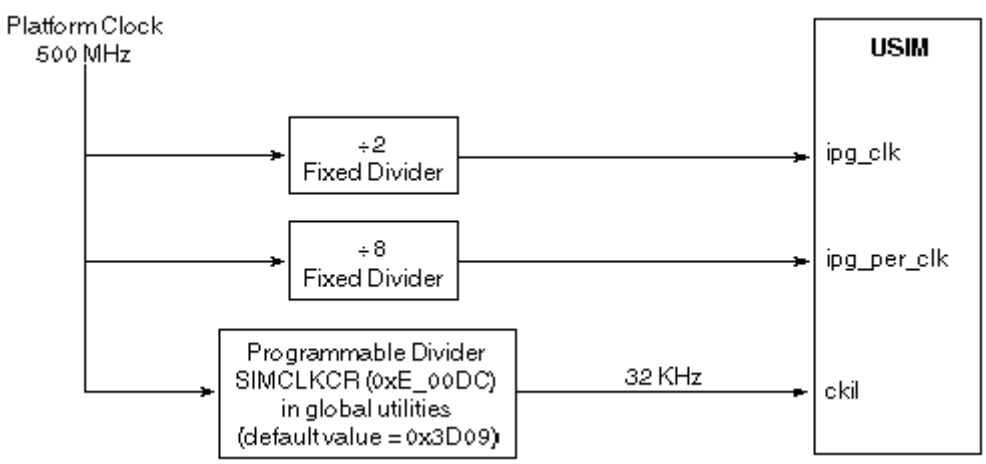


Figure 11-36. SIM Clock Generator

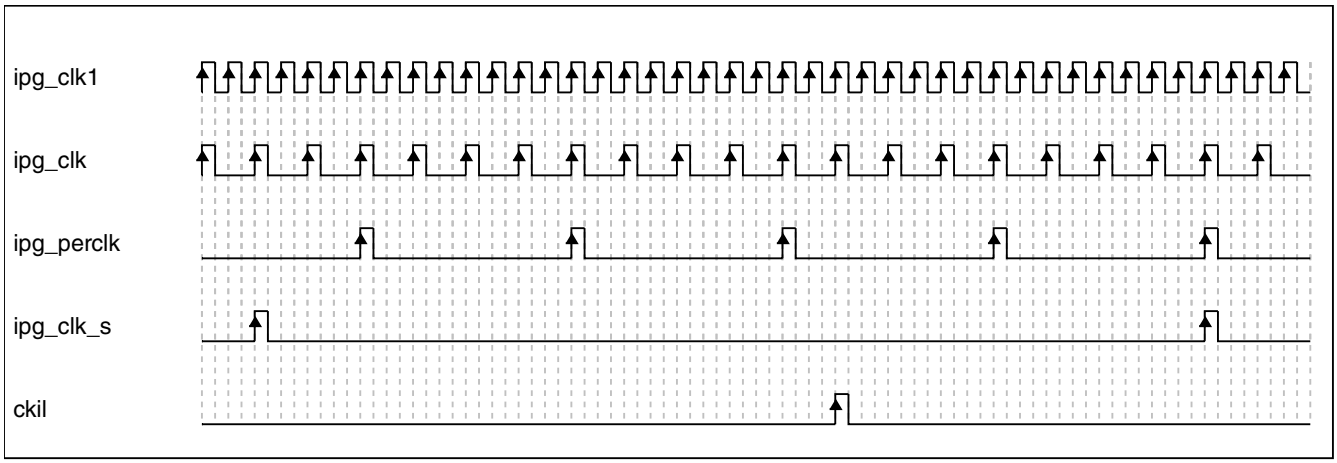


Figure 11-37. SIM Clocking Diagram

11.2.3.3 SIM Clock Generator

The clock generator is responsible for implementing clock tree synthesis constructs, scan clock muxing, baud rate clock (BAUD_CLK) generation, and providing clocks to the transmitter, receiver, and port controller sections of the SIM module. See the figure below for the schematic of the SIM clock generator. The dividers outlined in bold, generate a pulsed (gated) clock of the desired frequency. The pulse is equal in duration to one half the ipg_per_clk period. This clock is used internal to the SIM module to drive the transmit and receive sections of the module.

The dividers outline in normal, generate a clock of 45-55% duty cycle to drive the clock pin of the SmartCard. This clock is not used internal to the SIM module.

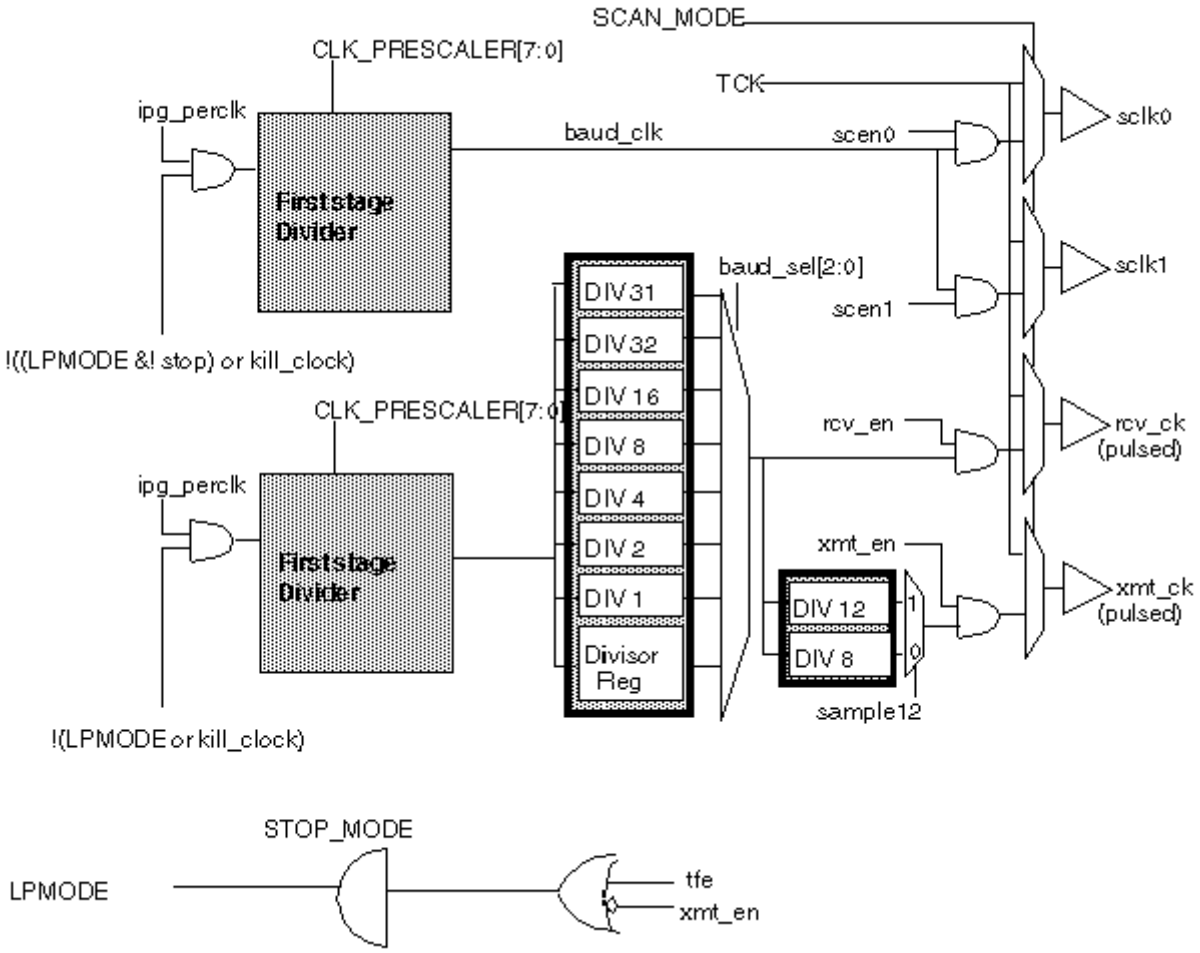


Figure 11-38. SIM Clock Generator

11.2.3.3.1 Scan Test

The scan clock muxing is implemented according to the design for testability rules.

11.2.3.3.2 Baud Clock Generation

The baud rate clock generation performed by the clock generator results in different frequencies. The default frequency is a divide by two of the ipg_perclk input. The baud rate can be programmed to be ipg_perclk divided by the even value using the CLK_PRESCALER[7:0] bits as shown in the CLK_PRESCALER register. The baud rate clock is generated in two forms in the design. The BAUD_CLK that is used by the SIM cards (SCLK0) is generated so that it must be 45-55% duty at the divide values. This is necessary to meet the requirements of the ISO 7816 specification. The BAUD_CLK that is used internal to the SIM module is generated as a gated version of the ipg_perclk clock. The resultant clock will be a pulse of one-half ipg_perclk period in width with the

expected frequency. The gated baud clock complies with the clock methodology initiative. The 50% duty cycle clock does not comply with the clock methodology initiative but this clock is not used as a clock internal to the SIM module. It is only used as a data path.

11.2.3.3.3 Transmitter Clock Generation

The transmitter clock (xmt_clk) is generated by the clock generator based on the values passed to it for the baud rate select (baud_sel[2:0]) and sample12. The transmit clock is gated by the transmit enable (XMT_EN) register bit. When the transmitter is enabled, the clock generator counts the appropriate number of receive clock (rcv_clk) positive edges to determine when to toggle the transmitter clock output. The transmitter clock is always based upon the receive clock.

11.2.3.3.4 Receiver Clock Generation

The receiver clock (rcv_clk) is generated by the clock generator based on the value passed to it for the baud rate select (baud_sel[2:0]). The receiver clock is gated by the receiver enable (RCV_EN) register bit. When the receiver is enabled, the clock generator counts the appropriate number of BAUD_CLK positive edges to determine when to toggle the receiver clock output. The number of BAUD_CLK positive edges is determined by the value of the baud rate select input (baud_sel[2:0]) and is programmable to these divisors: 31 (slowest because used with the 372/1 Fi/Di rate), 32, 16, 8, 4, 2, 1, and divisor[7:0]. The programmable divisor (divisor[7:0]) is programmable via the DIVISOR_REG register.

11.2.3.3.5 Port Control Clock Generation

The port controller clocks are provided by the clock generator for use on the SIM card ports. These clocks are equivalent in frequency to the BAUD_CLK and are gated by the SIM clock enable (scen0, scen1) signals. The level at which the card clocks (SCLK0, SCLK1) are stopped when disabled is determined by the SIM clock select polarity (SCSP0, SCSP1) inputs to the clock generator. Synchronizers are implemented to ensure glitch free operation of the card clocks when enabling/disabling, or changing the clock stopped polarity.

11.2.3.3.6 Low Power Mode Clock Control

The clock generator block is responsible for gating the clocks to the SIM module appropriately whenever a low power mode instruction is decoded. The IP interface provides two signals that encode the two available low-power states of the processor. These states are: STOP and DOZE. The response to the DOZE instruction is controlled

through the use of the DOZE bit in the RESET_CNTL register. See the RESET_CNTL register description. The response to the STOP instruction is controlled through the use of the STOP bit in the RESET_CNTL register. See the RESET_CNTL description

11.2.3.4 SIM Transmitter

The SIM Transmitter block has been designed to localize all transmit related circuitry into one section of hierarchy. The Transmitter block comprises the following sections of logic: transmit state machine, transmit shift register, transmit FIFO, guard time generator, transmit NACK control, and transmit data convention.

11.2.3.4.1 Transmit State Machine

The Transmit state machine is the heart of the transmitter block. The state machine is responsible for sequencing through a transmit operation while reacting to inputs from the receiver, the transmit FIFO, and the guard time circuit. See the figure below for flow diagram of the transmit state machine.

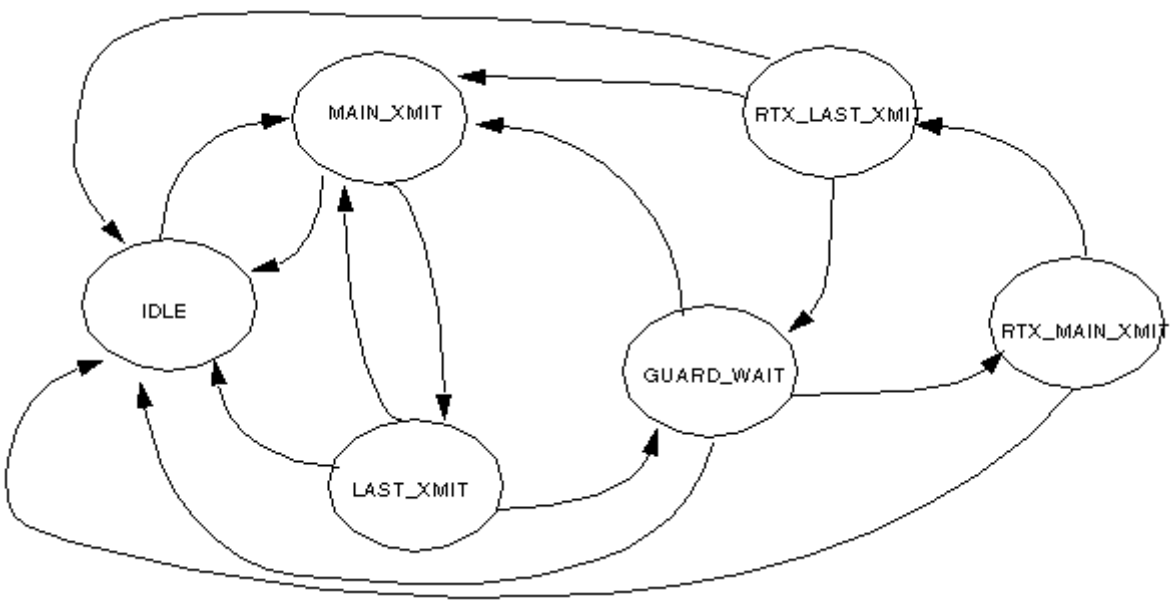


Figure 11-39. Transmit State Machine

The functions performed by each state are:

- IDLE
  - This is the initial state. The state machine waits here until it detects XMT_EN is set and a write to the transmit FIFO has occurred. The data pointed to by the transmit read pointer is loaded into the shift register, and the state machine

transitions to the MAIN_XMIT state. Any time XMT_EN is cleared, the state machine returns to this state.

- MAIN_XMIT
  - The transmitter is operating normally in this state. The data in the shift register is shifting once every transmit clock cycle. When the second to last bit of the current transmission is about to be sent, the state machine transitions to the LAST_XMIT state.
- LAST_XMIT
  - This state transmits the last bit of the current transmission and determines the next operation. One of the following will occur.
    - If GETU[7:0] is non-zero, jump to the GUARD_WAIT state.
    - If a transmit NACK error occurred, with a zero in GETU[7:0], jump to MAIN_XMIT state to retransmit the current byte.
    - If no transmit NACK, and GETU[7:0] is zero, load the shift register, jump to MAIN_XMIT to transmit the next byte
    - If no transmit NACK, GETU[7:0] is zero, and the FIFO is empty, jump to IDLE state; set the transmit complete (TC) flag.
- GUARD_WAIT
  - The state machine remains in this state until the Guard time counter has expired.
    - If a transmit NACK error occurred on last transmission, jump to RTX_MAIN_XMIT and re-transmit
    - If no transmit NACK and the FIFO is not empty, load the shift register, jump to MAIN_XMIT to transmit the next byte.
    - If no transmit NACK and the FIFO is empty, return to the IDLE state.
    - If transmit NACK threshold is detected, stop transmitter, set XTE flag, and jump to the IDLE state.
- RTX_MAIN_XMIT
  - The transmitter is operating normally in this state. The data in the shift register is shifting once every transmit clock cycle. When the second to last bit of the current transmission is about to be sent, the state machine transitions to the RTX_LAST_XMIT state. This state is identical to the MAIN_XMIT state except that it retransmits the previously NACKed byte.
- RTX_LAST_XMIT
  - This state transmits the last bit of the current re-transmission and determines the next operation. One of the following will occur.
    - If GETU[7:0] is non-zero, jump to the GUARD_WAIT state.
    - If a transmit NACK error occurred, with a zero in GETU[7:0], jump to GUARD_WAIT state to check Transmit NACK threshold.

- If no transmit NACK, GETU[7:0] is zero, and the FIFO is not empty, load the shift register, jump to MAIN_XMIT to transmit the next byte
- If no transmit NACK, GETU[7:0] is zero, and the FIFO is empty, jump to IDLE state; set the transmit complete (TC) flag.

11.2.3.4.2 Transmit Shift Register

The transmit shift register is 11 bits wide and controlled by the transmit state machine described previously. The shift register shifts out data at the transmit clock frequency (xmt_ck).

11.2.3.4.3 Transmit FIFO

The transmit FIFO is implemented inside the transmitter block. The FIFO depth is 16 bytes. The FIFO block is shared by both SIM module ports. The transmit FIFO cannot be accessed by the alternate SIM module through the alternate port. Each write to the transmit FIFO increases the transmit FIFO write pointer. Each time the transmit shift register is loaded from the transmit FIFO, the transmit FIFO read pointer is increased. When the read and write pointers are equal, the transmit FIFO empty flag (TFE) is set. Software has no visibility of the transmit FIFO pointers, but a transmit FIFO threshold value can be set to alert the software when the number of bytes in the FIFO has reached a specified level. A read of the transmit FIFO register (XMT_BUF) will return the last byte written to the FIFO. Writes to the transmit FIFO can occur at any time.

The transmit FIFO can be flushed by setting the XMT_FLUSH bit in the RESET_CNTL register. A transmit NACK threshold error (XTE) will halt the transmitter, and flush the transmit FIFO. The flush operation resets the transmit read and write pointers to equal values. Everything in the transmitter block is reset by the transmit flush operation. This does not include the control registers associated with the transmitter. The Transmit data threshold flag (TDTF) does not get cleared by the XMT_FLUSH operation.

11.2.3.4.4 Transmit Guard Time Generator

The guard time generator is simply a counter that is clocked by the transmit clock in order to delay the beginning of the next transmission and the setting of the transmit complete interrupt flag (TC) by a programmable amount of transmit bit widths (ETU's). The duration of the count is controlled by the GUARD_CNTL register (GETU[7:0]). See the figure below for depiction of three transmit operations in order to show the effect of the guard time generator logic

.

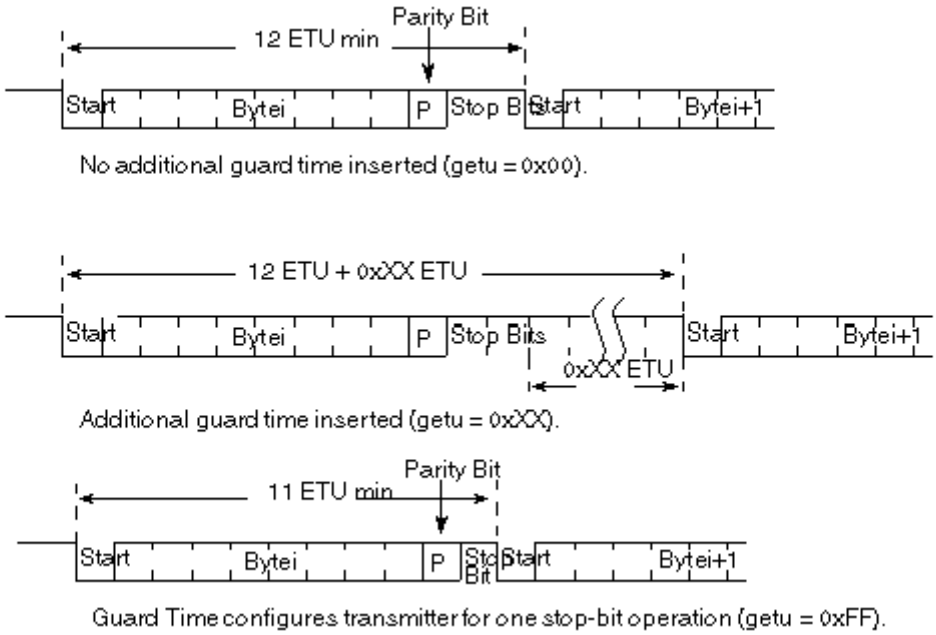


Figure 11-40. Transmit Guard Time

11.2.3.4.5 Transmit NACK Generator

The transmit NACK generator is responsible for driving the transmitter output low during the STOP bit time to signify an error was detected in the received data from the SIM card. This logic responds to a NACK request generated by the receiver block. The figure below shows a typical SIM transaction with the NACK pulse inserted.

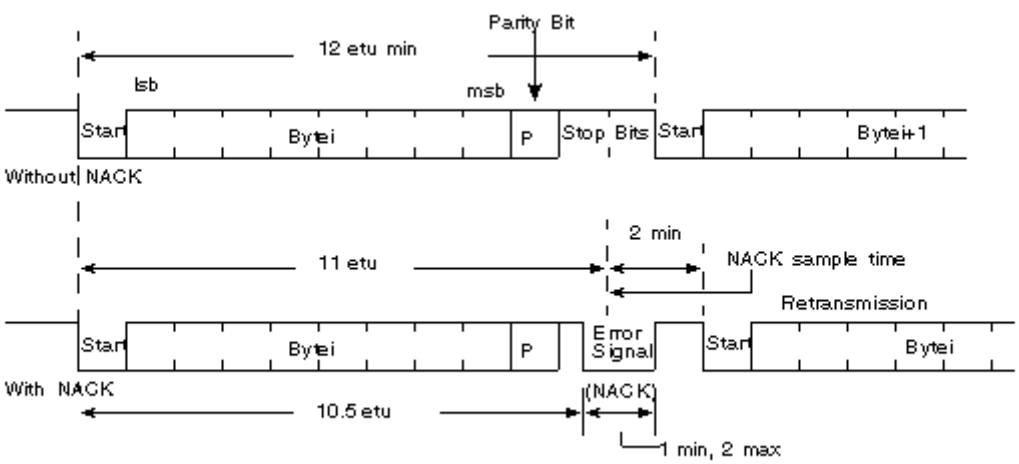
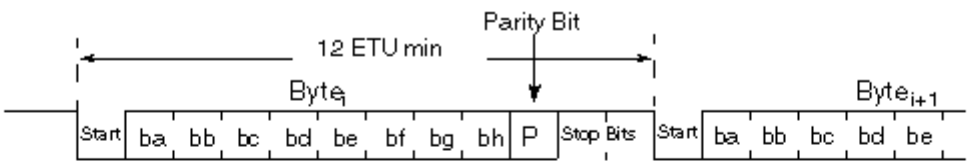


Figure 11-41. Transmit NACK Generator

The transmit NACK generator is also responsible for keeping track of the number of NACKs received during a transmit operation. The SIM receive state machine detects NACKs generated by the SIM card, and reports them to the transmit NACK logic. Once the number of detected NACKs has reached the programmed threshold (XTH[3:0]), an interrupt flag is generated, the transmit FIFO is flushed, and the transmitter is disabled.

11.2.3.4.6 Transmit Data Convention Logic

The transmit data convention logic provides the support for the two different data conventions available in SIM cards. See the figure below for an illustration of SIM data conventions.



Parity Bit: if configured for even parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be even
if configured for odd parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be odd
When configured for inverse convention, the parity bit is inverted by USIM before being transmitted.

Direct Convention: *ba* is lsb of data byte to be sent. *bh* is msb.
Neither the data bits nor parity bit is logically inverted.

Inverse Convention: *ba* is msb of data byte to be sent. *bh* is lsb.
Both the data bits and parity bit are logically inverted by hardware.

Figure 11-42. SIM Data Conventions

The direct data convention is the default. If the inverse convention bit (IC) in the DATA_FORMAT register is set, then the transmit data convention logic will convert the output of the transmit FIFO to the inverse convention before sending it to the transmit shift register.

11.2.3.5 SIM Receiver

The SIM Receiver block has been designed to localize all receive related circuitry into one section of hierarchy. The receiver block is comprised of the following functions: receive state machine and the receive FIFO.

11.2.3.5.1 Receive State Machine

The receive state machine is responsible for sampling the receive data pin and capturing the bit value into the receive shift register. Additionally, the receive state machine can detect the START bit, parity errors, framing errors, and initial character when operating in initial character mode.

Once enabled by the RCV_EN bit in the ENABLE register, the receive state machine sequences through the states as shown in the figure below.

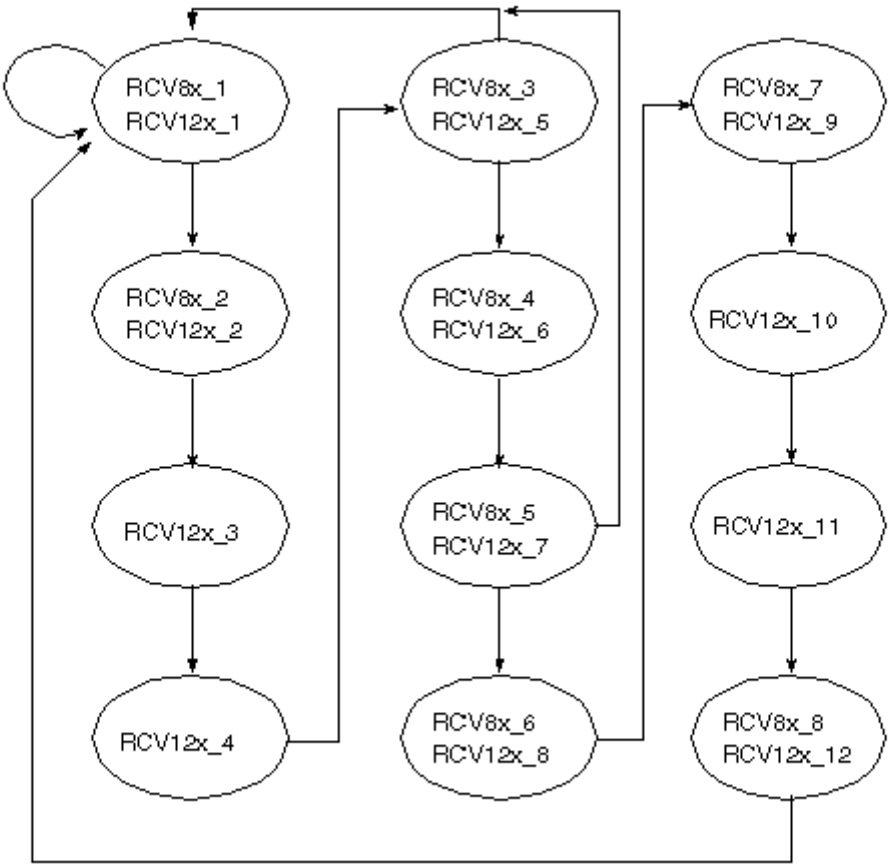


Figure 11-43. Receive State Machine

The states identified with a "8x" are used when operating in a 8x oversampling mode. The states identified with a "12x" are used when operating in a 12x oversampling mode. This mode is only used when sample12 is set to "1". The number following the oversampling mode identifier represents the state number in the current mode. There are 12 states in "12x" mode, and 8 states in "8x" mode. Some states simply implement a one RCV_CK delay. States that perform additional functions are:

- RCV8x_1, RCV12x_1

    - This is the initial state of the receive state machine. If the first bit has not been received, the state machine remains in this state until a valid START bit is detected. For every subsequent bit, this state is simply a one RCV_CK cycle delay.
- RCV8x_2, RCV12x_2
    - This state captures the first sample of the current receive data input.
- RCV12x_3
    - This state captures the second sample of the current receive data input.
- RCV12x_4
    - This state captures the third sample of the current receive data input.
- RCV8x_3, RCV12x_5
    - This state checks if we are receiving a correct START Bit. If not, Then we return to state 1.
- RCV8x_4, RCV12x_6
    - If we are in the 11th Bit, We check for Parity or Initial Character errors and send NACK if needed.
- RCV8x_5, RCV12x_7
    - Store current Bit value in Shift Register. If this is the first bit and the value is not 0, restart the State Machine.
- RCV8x_6, RCV12x_8
    - If the current bit is the last bit of the transfer, Set flag to transfer Shift Register to Receive Buffer.
- RCV8x_7, RCV12x_9
    - Clear flag for transferring Shift Register to Receive Buffer.
- RCV12x_10
    - If the current bit is the last bit of the transfer (first stop bit), this state takes the sample of the NACK window.
- RCV12x_11
    - If the current bit is the last bit of the transfer (first stop bit), this state takes the sample of the NACK window.
- RCV8x_8, RCV12x_12
    - This state represents the end of the current receive input bit. Several operations occur during this state, including:
        - Increment bit counter
        - Perform a majority vote on the NACK samples and notify the transmitter if a NACK pulse was detected.

11.2.3.5.2 Data Sampling / Voting

The receive state machine runs at the receive clock rate (RCV_CK). This clock is used to oversample the received data at either a 8X or 12X sample rate. For each input bit, the receive state machine captures 3 samples. A majority voting algorithm is then applied to determine the value of the bit received. The value common to 2 or more of the samples is determined to be the bit value in the receive shift register.

11.2.3.5.3 Start Bit Detection

The SIM receive input is defined to be high when not active. The data transmission is defined to begin with a low pulse for a bit duration. This is called the START bit. The receive state machine is responsible for detecting and validating the START bit. The receive state machine samples the START bit three times using a majority voting scheme to determine if the START bit is valid. This will effectively filter out any low receive inputs shorter than one RCV_CK period. See the figure below for an illustration of a typical SIM data transaction with the START bit identified.

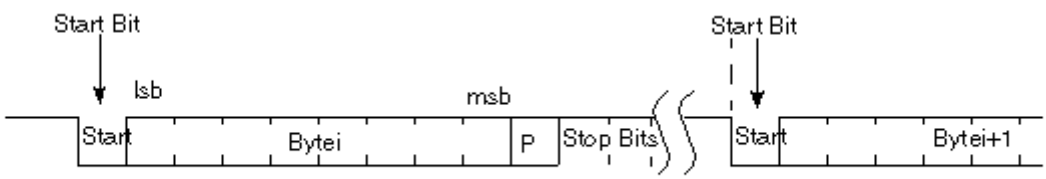


Figure 11-44. Start Bit Diagram

11.2.3.5.4 Parity Error Detection

The receive state machine is responsible for detecting parity errors in the received data. Data is always transmitted with even parity, except when in inverse convention mode. In inverse convention mode, all data bits and the parity bit are complemented making the data appear to be odd parity. The parity bit is defined as the 10th bit of the received data. The parity of the 2nd through 10th received bits is calculated by the receiver parity logic. This logic determines if the parity of the 9 received bits is correct. See the figure below for an illustration of a typical SIM data transaction with the parity bit identified.

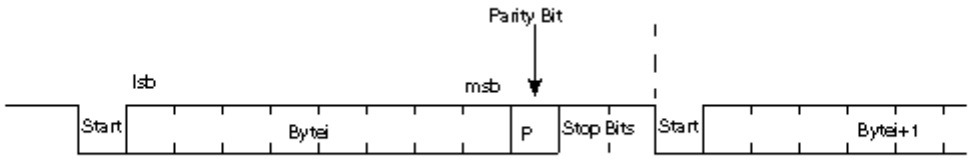


Figure 11-45. Parity Bit Diagram

When a parity error is detected on a given byte, the RCV_PF bit for that byte is set in the receive FIFO if NACK generation on errors is disabled (ANACK = 0 in the CNTL register). A parity error cannot cause an interrupt, but when NACK generation is enabled (ANACK = 1), it can signal the SIM transmitter to create a NACK pulse to the SIM card asking for a retransmission of the corrupted data.

11.2.3.5.5 Framing Error Detection

The receive state machine is responsible for detecting framing errors in the received data. A SIM data transaction is defined as 11 or 12 bits long consisting of the START bit, 8 data bits, 1 parity bit and 1 or 2 STOP bits. A framing error occurs when the STOP bit is not detected during the 11th bit time of a data transaction. The STOP bit is generally defined as two bit times (ETU's) of a high pulse following the parity bit. When the GUARD_CNTL register is programmed to 0xFF, the STOP bit is defined as one bit time. A framing error can only occur when the parity bit of the current byte is low, and the STOP bit arrives late. See the figure below for an illustration of a typical SIM data transaction with the STOP bits identified. Also, shown is a SIM data transaction with a late arriving STOP bit indicating a framing error.

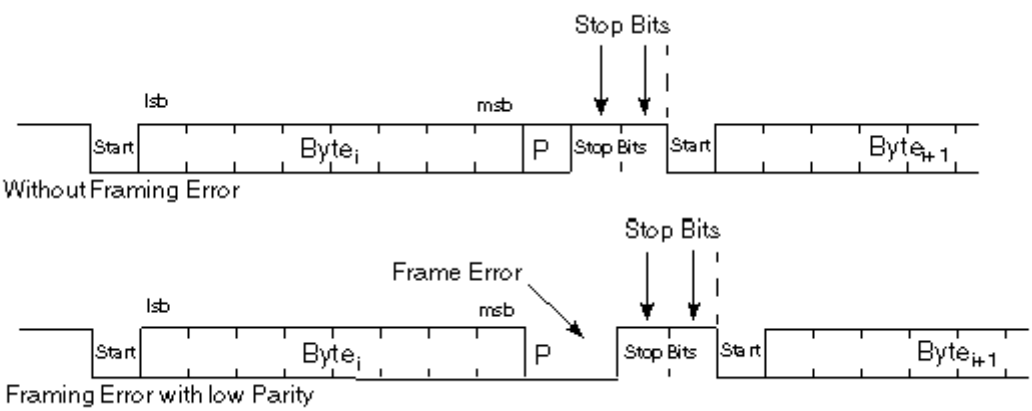


Figure 11-46. Framing Error Diagram

When a framing error is detected on a given byte, the RCV_FE bit for that byte is set in the receive FIFO. A framing error cannot cause an interrupt, nor can it create a NACK pulse to the SIM card asking for a retransmission of the corrupted data.

11.2.3.5.6 NACK Detection

The existence of the NACK pulse is sampled by the receive state machine at 11 Elementary Time Units (ETUs) after the falling edge of the START bit. An ETU is equivalent in time to 1 transmit clock period. Once the receiver detects a NACK, it signals the transmitter that an error occurred. The transmitter will not initiate retransmission for at least another two ETU times as required by the ISO 7816 specification.

11.2.3.5.7 Initial Character Detection

The SIM receive state machine supports the detection of special characters that allow it to determine what data format is being used by the connected SIM card. When placed in initial character mode, the SIM expects to receive one of two potential characters that it will use to set the data format control bit, IC, in the DATA_FORMAT register.

The two possible data formats are inverse convention and direct convention. The following figures illustrate the differences between the two formats. Essentially, inverse convention differs from direct convention in that the order of the data is flipped msb for lsb and the data bits and parity bit are logically inverted. When receiving inverse convention data, the transformation of the data back to direct convention format is done in hardware, including the inversion of the data and parity bits.

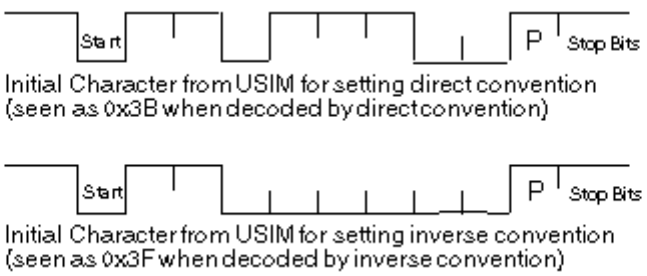
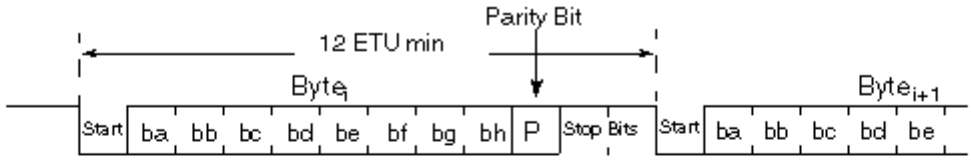


Figure 11-47. Valid Initial Characters



Parity Bit: if configured for even parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be even
if configured for odd parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be odd
When configured for inverse convention, the parity bit is inverted by SIM card before being transmitted.

Direct Convention: *ba* is lsb of data byte to be sent. *bh* is msb.
Neither the data bits nor parity bit is logically inverted.

Inverse Convention: *ba* is msb of data byte to be sent. *bh* is lsb.
Both the data bits and parity bit are logically inverted by SIM card.

Figure 11-48. Inverse Convention versus Direct Convention

11.2.3.5.8 Receive FIFO

The receive FIFO is implemented inside a sub-block of the receiver block. The FIFO depth is 288 bytes. The FIFO block is shared by both SIM module ports. The receive FIFO cannot be accessed by another SIM module through the alternate port. Only the port1 IO pins can be accessed through the alternate port. The secondary SIM would then uses its own internal FIFO while using the primary SIM IO ports. This is only done if it is required to control two SIM cards at the same time. It the two SIM card access only occurs one at a time, then a single SIM module can control both SIM cards (one on port0 and one on port1). The receive FIFO is accessed through the RCV_BUF register.

The receive FIFO is loaded from the receive shift register after the final bit of the current SIM card transmission has been received. The FIFO contains 10 bits per transmission. The lower eight bits contain the received data byte. Bits 8 and 9 contain the parity and framing status for the received byte.

Each read from the receive FIFO increases the receive FIFO read pointer. Each time the receive shift register is transferred to the receive FIFO, the receive FIFO write pointer is increased. When the difference between the read and write pointers equals the programmed threshold value (RDT), the receive data register full flag (RDRF) will be set. An interrupt can be generated by the RDRF flag if the RIM bit in the INT_MASK register is cleared. Software has no visibility of the receive FIFO pointers. A write to the receive FIFO register (RCV_BUF) will generate an invalid access exception to the processor.

The receive FIFO can be flushed by setting the RCV_FLUSH bit in the RESET_CNTL register. The flush operation resets the receive read and write pointers to equal values. All logic associated with the receiver will be reset by the flush operation except for receiver control registers.

11.2.3.5.9 Overrun Detection

The receive FIFO logic is responsible for detecting an overrun condition. When a received byte is transferred from the receive shift register to a receive FIFO that contains 288 unread bytes, the SIM receiver will flag an overrun condition. This condition will always set the overrun error flag, OEF in the RCV_STATUS register. The received byte will be discarded leaving the 288 unread bytes in the FIFO unaltered. The SIM module will generate a NACK to the SIM card on an overrun condition if the ONACK bit in the CNTL register is set. The SIM module will continually NACK SIM card transmissions until a read of the receive FIFO occurs.

11.2.3.5.10 Character Wait Time Counter

The SIM receiver block includes a 16-bit counter that counts the number of bit times (ETUs) between received characters. When enabled, the Character Wait Time Counter (CWT) will not start counting until after the START bit(s) of a valid character has been received. The counter is synchronized to the receive character bit positions so that an accurate count of the number of ETUs between characters can be made. The Character Wait Time Counter has a 16-bit programmable comparator. Software can write the expected number of ETUs between characters to the comparator. If the time between characters exceeds this value, an interrupt flag will be set and an interrupt generated if the mask is clear.

11.2.3.6 SIM Port Control

The SIM Port Control block has been designed to localize all port related circuitry into one section of hierarchy. The port control block comprises the following functions: SIM card interface, SIM Card presence detect, and SIM card auto-power down.

11.2.3.6.1 SIM Card Interface

The SIM module allows for direct control of two separate SIM cards. The SIM module does not support simultaneous communication with two SIM cards. To achieve simultaneous communication with two SIM cards, a second SIM module must be used. The SIM module can relinquish control of the inactive port to a secondary SIM module by setting the AMODE bit in the SETUP register. When alternate SIM Card Mode enable, output signals in port1 is directly from the secondary SIM module and input signals in port1 will pass through to the secondary SIM module.

The SIM card clock is generated in the SIM clock generator. The SIM card reset is controlled by software through the PORT0_CNTL register. Since the SIM module does not support a separate transmit data and receive data port, the 3VOLT0 bit should be set to 1.

11.2.3.6.2 SIM Card Presence Detect

The SIMPDx input allows for detection of the insertion or removal of a SIM card. The SPDSx control bit in the PORTx_DETECT register allows the software to configure which edge of the SIMPDx pin will cause a presence detect event. A maskable interrupt can be generated when a SIMPDx event occurs.

An internal pull-down device is present on the SIMPDx pins. This will provide for a high to low transition on the SIMPDx pin when a SIM card is removed.

11.2.3.6.3 SIM Card Automatic Power Down

When interfacing to the SIM cards, it is necessary to follow a particular sequence when powering them up and down. The SIM port control block contains hardware that provides the correct sequence to power down a SIM card (see the figure below). The power up sequence must be done manually by the software using the pin control bits supplied in the PORTx_CNTL registers.

The power down sequence is specified in ISO 7816 as:

1. RST transitions from high to low
2. CLK is turned off to a low
3. I/O transitions from high impedance to low
4. SIM V_{CC} is turned off

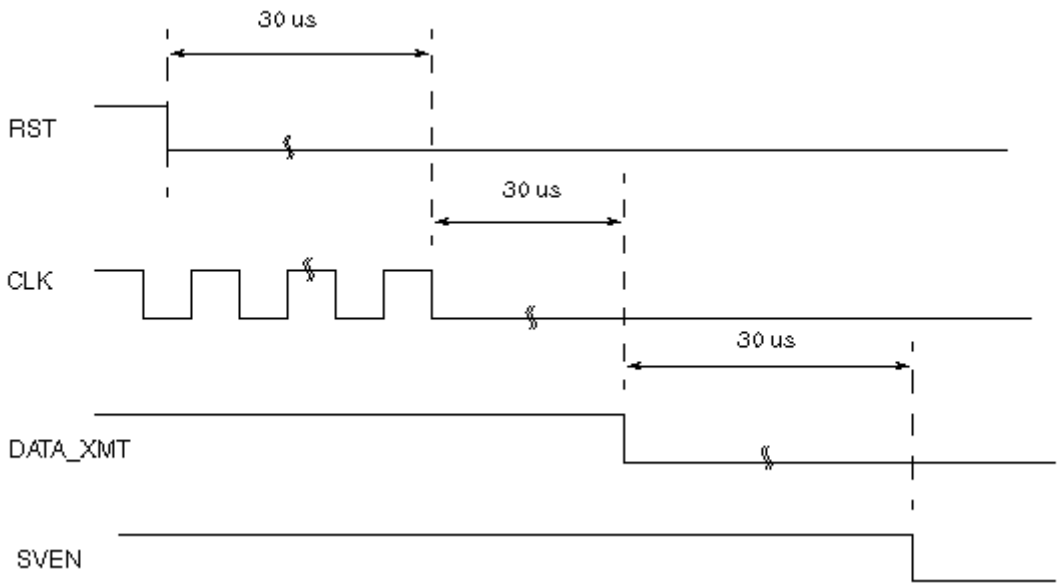


Figure 11-49. Auto Power Down Sequence

11.2.3.7 SIM General Purpose Counter

The SIM module provides a 16-bit counter for use when timing events during SIM card communication. The clock source for the counter is selectable between three sources: BAUD_CLK, RCV_CLK, or XMT_CK (ETU clock). The GPCNT_CLKSEL[1:0] bits in the CNTL register are used to select the clock input. The counter is enabled as soon as the input clock is selected. The starting of the counter is immediate once the input clock is running. Software can control the three input clock sources by using the KILL_CLOCK, RCV_EN and XMT_EN bits provided in the RESET_CNTL and ENABLE registers.

To run the counter from the card clock source the following conditions must be met:

1. 'KILL_CLOCK = 0' in the RESET_CNTL register.
2. 'GPCNT_CLKSET[1:0] = 01' in the CNTL register

The counter will begin to count at the card clock rate as soon as these conditions are met.

To run the counter from the receive clock source the following conditions must be met:

1. 'KILL_CLOCK = 0' in the RESET_CNTL register.
2. 'RCV_EN = 1' or XMT_EN = 1 in the ENABLE register
3. 'GPCNT_CLKSET[1:0] = 10' in the CNTL register

The counter will begin to count at the receive clock rate as soon as these conditions are met.

To run the counter from the ETU (transmit) clock source the following conditions must be met:

1. 'KILL_CLOCK = 0' in the RESET_CNTL register.
2. Either one of the following:
3. "RCV_EN = 1' in the ENABLE register and 'CWT_EN = 1' in the CNTL register

   'XMT_EN = 1' in the ENABLE register

4. 'GPCNT_CLKSET[1:0] = 11' in the CNTL register

The counter will begin to count at the ETU clock rate as soon as these conditions are met.

The counter can be reset by setting GPCNT_CLKSEL[1:0] to 00. A 16-bit comparator value is provided that allows the software to select a count value at which an interrupt flag can be set and an interrupt generated if the mask is clear.

11.2.3.8 SIM LRC Block

The SIM module provides an 8-bit Linear Redundancy Check (LRC) generator / checker. The block is provided for use with T=1 SIM cards that support LRC. This block can be enabled through the LRCEN bit in the CNTL register. This block performs an 8-bit exclusive-OR on all received or transmitted characters. At the end of the reception of a block of characters, the result is expected to be 00. If so, the LRCOK bit is set in the RCV_STAT register. During transmission, the LRC block Exclusive-ORs each character that is transmitted with the current value of the LRC. If the XMT_CRC_LRC bit in the CNTL register is set, the LRC value will automatically be sent by the SIM transmitter as the final character when the transmit FIFO empties.

The LRC value can be reset in multiple ways. Clearing the LCREN bit in the CNTL register will reset the LRC value. At the end of a transmission (either after the LRC byte is transmitted, or after the last character in the transmit FIFO is sent when XMT_CRC_LRC is clear), the LRC value is automatically reset by the SIM hardware. Finally, when setting the XMT_EN bit, the SIM hardware resets the LRC value.

11.2.3.9 SIM CRC Block

The SIM module provides an 16-bit Cyclic Redundancy Check (CRC) generator / checker. The block is provided for use with T=1 SIM cards that support CRC. This block can be enabled through the CRCEN bit in the CNTL register. This block performs a polynomial based check on all received or transmitted characters. The polynomial description is shown in Figure 4-17. The polynomial used is the standard CRC-CCITT

where g(x) = x^16 + x^12 + x^5 + 1. The CRC register is initialized to all "1" before the data is shifted in. Before transmission the resulting CRC is inverted. For example, transmitting a 0xFA results in the following:

- Data = 0x5F (bit reversal of 0xFA)
- crc = 0x4AEA
- invert the crc = 0xB515 (bit reversal of 0xADA8
- data transmitted = 0xFA, 0xAD, 0xA8

See the figure below for an illustration of the SIM CRC block.

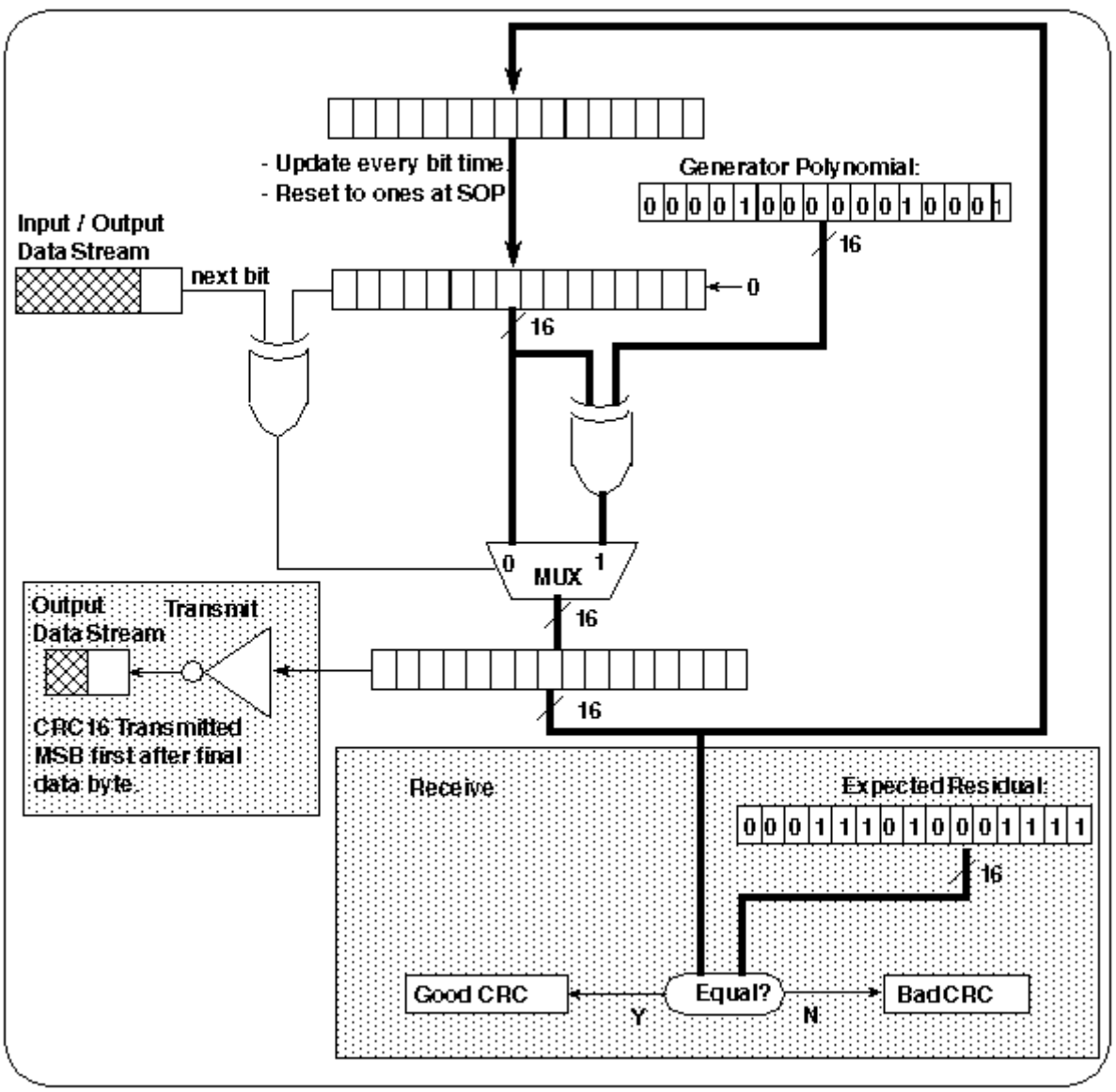


Figure 11-50. CRC Block Diagram

At the end of the reception of a block of characters, the residual from the CRC calculation is compared to 0x1D0F. If so, the CRCOK bit is set in the RCV_STAT register. During transmission, the CRC block updates the current value of the CRC residual using each character. If the XMT_CRC_LRC bit in the CNTL register is set, the CRC value will automatically be inverted and sent by the SIM transmitter as the final two characters when the transmit FIFO empties.

The CRC value can be reset in multiple ways. Clearing the CRCEN bit in the CNTL register will reset the CRC value. At the end of a transmission (either after the CRC characters are transmitted, or after the last character in the transmit FIFO is sent when XMT_CRC_LRC is clear), the CRC value is automatically reset by the SIM hardware. Finally, when setting the XMT_EN bit, the SIM hardware resets the CRC value.

11.2.3.10 Module Interrupts

See the table below for the list of all possible interrupt sources and their corresponding mask bits. All SIM interrupts are logically ORed to create the active low irq_n and data_irq_n signals that go to the ITC module. All mask bits are such that a logic 1 implies that the corresponding interrupt is disabled (masked), whereas a 0 implies that the interrupt enabled (unmasked). All of these mask bits are logic 1 out of reset (all interrupts are masked).

Table 11-53. SIM Module Interrupts

Flag	Flag Register	Mask	Mask Register	Description
TC	XMT_STATUS	TCIM	INT_MASK	Transmit Complete
ETC	XMT_STATUS	ETCIM	INT_MASK	Early Transmit Complete
TFE	XMT_STATUS)	TFEIM	INT_MASK	Transmit FIFO Empty
XTE	XMT_STATUS	XTM	INT_MASK	Transmit Threshold Error
TFO	XMT_STATUS	TFOM	INT_MASK	Transmit FIFO Overfill error
TDTF	XMT_STATUS	TDTFM	INT_MASK	Transmit Data Threshold Flag
GPCNT	XMT_STATUS	GPCNTM	INT_MASK	General Purpose Counter Comparator Flag
RDRF	RCV_STATUS	RIM	INT_MASK	Receive data register full (FIFO threshold level reached)
OEF	RCV_STATUS	OIM	INT_MASK	Overrun Error Flag
CWT	RCV_STATUS	CWTM	INT_MASK	Character Wait Time Counter Comparator Flag
BWT	RCV_STATUS	BWTM	INT_MASK	Block Wait Time Counter Comparator Flag
BGT	RCV_STATUS	BGTM	INT_MASK	Block Guard Time Counter Comparator Flag
RTE	RCV_STATUS	RTM	INT_MASK	Receive NACK Threshold Error
SDI1	PORT1_DETECT	SDIM1	PORT1_DETECT	SIM Detect Interrupt for port 1
SDI0	PORT0_DETECT	SDIM0	PORT0_DETECT)	SIM Detect Interrupt for port 0

11.2.4 Initialization/Application Information

This section describes the intended programming model for using the SIM module. The section describes how to begin a typical mode of operation using the registers.

11.2.4.1 Configuring SIM for Operation

The following is a list of items that need to be performed in order to configure the SIM module for operation:

1. Port selection in the SETUP register.
   a. Select which SIM module port is active by writing the SPS bit.
   b. Select whether the second SIM module port is active at the same time under the control of an alternate SIM module by writing the AMODE bit.
2. Enable the selected port (for port 1, see PORT1_CNTL register; for port 0, see PORT0_CNTL) following the SIM power up procedure specified in ISO 7816.
   a. Program 3VOLT0 pin of PORT0_CNTRL register to 1.
   b. Enable power to the SIM card using the SVEN1 or SVEN0 bit.
   c. Enable SIM module transmit data output using the STEN1 or STEN0 bit. This is required to allow the SIM receiver to create NACK pulses.
   d. Enable SIM card clock using the SCEN1 or SCEN0 bit.
   e. Release the SIM card reset using the SRST1 or SRST0 bit (assert SRSTx high).
3. Select XMT port driver type (open-drain or push-pull) in the OD_CONFIG register.
   a. Configure the selected port to be open drain or push-pull by using the OD_P1 or OD_P0 bit.
4. Choose the port baud rate in the CNTL register.
   a. Select the SIM card clock rate by using the CLK_PRESCALER[7:0] register.
   b. Select the SIM card baud rate by using the BAUD_SEL[2:0] bits and SAMPLE12.

NOTE

Follow the ISO 7816 spec with regards to card clock frequencies to ensure the maximum frequency specification is not violated.

5. Select Data format type, or place SIM receiver in initial character mode.
   a. Select inverse convention or direct convention by using the IC bit in the DATA_FORMAT register, or
   b. Enable initial character mode by using the ICM bit in the CNTL register.

11.2.4.1.1 Configuring SIM Receiver

The following is a list of items that need to be performed in order to configure the SIM receiver for operation:

1. Enable NACK capability in CNTL register.
   a. Select NACK generation on parity errors, or invalid initial character by using the ANACK bit.
   b. Select NACK generation on overrun conditions by using the ONACK bit.
2. Select the desired Receive NACK threshold and receive FIFO threshold in RCV_THRESHOLD register.
   a. Program the threshold at which the RDRF flag will be set by using the RDT bits.
   b. Program the threshold at which the RTE flag will be set by writing to the RTH[3:0] bits. If an auto power down after RTE flag is set, is desired, then enable the SIM card auto power down by setting the SAPD0,1 bits in the port0,1_cntl register.
3. Configure the Character Wait Time Counter, the Block Wait Timer Counter and the Block Guard Time Counter.
4. Enable interrupts in the INT_MASK register.
   a. Enable the receive data register full interrupt by using the RIM bit
   b. Enable the receive threshold error interrupt by using the RTM bit.
   c. Enable the overrun condition interrupt by using the OIM bit.
   d. Enable the Character Wait Time interrupt by using the CWTM bit
5. Enable enforce reception early stop function in ENABLE register.
   a. Enable the enforce reception early stop enable by using the ESTOP_EN bit.
   b. Enable the enforce reception early stop execution by using the ESTOP_EXE bit.

11.2.4.1.2 Configuring SIM Transmitter

The following is a list of items that need to be performed to configure the SIM transmitter for operation:

1. Select desired re-transmission threshold for NACKed characters in XMT_THRESHOLD register.
   a. Program the threshold at which the XTE flag will be set by using the XTH[3:0] bits.
2. Select the guard time between transmissions in the GUARD_CNTL register.
   a. Program the desired guard time between characters transmitted by the SIM module by using the GETU[7:0] bits.
3. Select the desired Transmit FIFO threshold level in the XMT_THRESHOLD register.
   a. Program the desired threshold using the TDT[3:0] bits.
4. Enable interrupts in the INT_MASK register.

a. Enable the transmit complete interrupt by using the TCIM BIT.
b. Enable the early transmit complete interrupt by using the ETCIM bit.
c. Enable the transmit FIFO empty interrupt by using the TFEIM bit.
d. Enable the transmit threshold error interrupt by using the XTM bit.
e. Enable the transmit FIFO threshold interrupt by using the TDTFM bit.
f. Enable the transmit FIFO overfill interrupt by using the TFOM bit.

5. Enable NACK delay detection in ENABLE register.
   a. Enable the NACK delay detection enable by using the NACK_DD_EN bit.

11.2.4.1.3 Configuring SIM General Purpose Counter

The following is a list of items that need to be performed in order to configure the SIM General Purpose Counter for operation:

1. Select desired clock source for the General Purpose Counter using the CNTL register.
   a. Use the GPCNT_CLKSEL[1:0] bits to select the desired clock source for the counter
2. Program counter comparator using the GPCNT register.
   a. Use the GPCNT[15:0] bits to select the desired count value at which the GPCNT interrupt flag will be set.
3. Enable the selected clock source for the General Purpose Counter using either the RESET_CNTL or ENABLE registers.
   a. If the GP Counter is configured for the card clock, enable the clock by clearing the KILL_CLOCK bit in the RESET_CNTL register.
   b. If the GP Counter is configured for the receive oversample clock, enable this clock by setting the RCV_EN bit or XMT_EN bit in the ENABLE register.
   c. If the GP Counter is configured for the transmit oversample clock, enable this clock by setting the XMT_EN bit in the ENABLE register.
4. Enable interrupts in the INT_MASK register.
   a. Enable the general purpose counter interrupt by using the GPCNTM bit.

11.2.4.1.4 Configuring SIM to Measure WWT (Work Wait Time) for Type=0 Smartcards

The following is a list of items that need to be performed in order to configure the SIM to measure work wait time. This is intended for type=0 SmartCards. Work wait time is a combination of BWT and CWT. The CWT timer is to measure the time between the start bits of two consecutive characters in the block received from the SmartCard. If this time is larger that the value programmed in the cwt[15:0] register, then a flag will be set and an interrupt generated. This timer can be used for both type=0 and type=1 SmartCards. The BWT and BGT timer are used to measure the delay between the leading edge of the

last character of the block received by the card and the leading edge of the first character of the next block sent by the card. If you want the SIM to enforce a WWT of 100 bits. Then, you must activate both the CWT and BWT, and program both of them for a value of 100 bits. When measuring WWT, the BGT timer will not be used so it will be masked (inactive) by the BGTM bit. Below is the sequence to program the SIM to send and receive data while checking WWT.

1. Program both the CWT and the BWT (low and high) registers to the WWT (work wait time) value that needs to be enforced. Set BGT register to 0 (this is the default value).
2. Activate both the CWT and BWT functions by setting the CWTEN and BWTEN bit in the CNTL register.
3. Enable the CWT and BWT interrupts by clearing the CWTM and BWTM bits in the INT_MASK register.
4. Program the data to send to the SmartCard by writing data to XMT_BUFFER
5. Activate the SIM transmit and receive by setting bits XMT_EN and RCV_EN in the ENABLE register.
6. If the software has more data to send, then as the FIFO starts to get near empty, it should write more data to the XMT_BUFFER. If the software does not have any more data to send then proceed to the next step.
7. When SmartCard has completed its transmission to the SIM module, disable the BWT by clearing the bit BWTEN in the CNTL register. This will prepare SIM to measure the next block wait time.
8. Disable the SIM transmitter by clearing the XMT_EN bit in the ENABLE register.
9. Program the next data to send by writing data to the XMT_BUFFER.
10. Enable the BWT function by setting the BWTEN bit in the CNTL register.
11. Enable the SIM transmitter by setting the XMT_EN bit in the ENABLE register.
12. Steps 6 to 10 can be repeated for each transmission to the SmartCard.
13. If a CWT or a BWT interrupt occurs, then the software should consider that a WWT violation. The software should clear the CWT or BWT interrupt by writing a "1" to the BWT or CWT bit in the RCV_STATUS register.

11.2.4.1.5 Configuring SIM to measure CWT, BWT, BGT for type=1 SmartCards

The following is a list of items that need to be performed in order to configure the SIM to measure CWT, BWT, BGT. This is intended for type=1 SmartCards. The CWT timer is to measure the time between start bits in the consecutive characters received from the SmartCard. If this time is larger that the value programmed in the CWT[15:0] register, then a flag will be set and an interrupt generated. This timer can be used for both type=0 and type=1 SmartCards. The BWT, BGT timer is used to measure the delay between the leading edge of the last character of the block received by the card and the leading edge

of the first character of the next block sent by the card. If this time is larger than the value in the both BWT registers (BWT and BWT_H), then the BWT flag will be set and an interrupt generated. If the time is less than the value in the BGT register, then the BGT flag will be set and an interrupt generated.

1. Program the CWT, BWT, BWT_H and BGT registers to the value that needs to be enforced.
2. Activate both the CWT and BWT functions by setting the CWTEN and BWTEN bit in the CNTL register.
3. Enable the CWT, BWT, BGT interrupts by clearing the CWTM, BWTM, BGTM bits in the INT_MASK register.
4. Program the data to send to the SmartCard by writing data to XMT_BUFFER
5. Activate the SIM transmit and receive by setting bits XMT_EN and RCV_EN in the ENABLE register.
6. If the software has more data to send, then as the FIFO starts to get near empty, it should write more data to the XMT_BUFFER. If the software does not have any more data to send then proceed to the next step.
7. When SmartCard has completed its transmission to the SIM module, disable the BWT by clearing the bit BWTEN in the CNTL register. This will prepare SIM to measure the next block wait time.
8. Disable the SIM transmitter by clearing the XMT_EN bit in the ENABLE register.
9. Program the next data to send by writing data to the XMT_BUFFER.
10. Enable the BWT function by setting the BWTEN bit in the CNTL register.
11. Enable the SIM transmitter by setting the XMT_EN bit in the ENABLE register.
12. Steps 6 to 10 can be repeated for each transmission to the SmartCard.
13. If a CWT or a BWT interrupt occurs, then the software should read the RCV_STATUS register to determine if the error was caused by a CWT, BWT, or a BGT violation. The software should clear the CWT, BWT, BGT interrupt by writing a "1" to the CWT, BWT, BGT bit in the RCV_STATUS register.

11.2.4.1.6 Configuring SIM Linear Redundancy Check (LRC) Block

The following is a list of items that need to be performed in order to configure the SIM Linear Redundancy Check Block for operation:

1. Enable the LRC block by using the CNTL register.
   a. Use the LRCEN bit to enable the LRC block.
   b. Use the XMT_CRC_LRC bit to enable the transmission of the LRC Character after the last character in the Transmit FIFO is sent. Refer to the T=1 programming model for more details.

11.2.4.1.7 Configuring SIM Cyclic Redundancy Check (CRC) Block

The following is a list of items that must be performed to configure the SIM Cyclic Redundancy Check Block for operation:

1. Enable the CRC block by using the CNTL register.
   a. Use the CRCEN bit to enable the CRC block.
   b. Use the XMT_CRC_LRC bit to enable the transmission of the CRC Characters after the last character in the Transmit FIFO is sent. Refer to the T=1 Programming model for more details.

11.2.4.2 Using the SIM Receiver

Once the SIM has been properly configured (such as correct baud rate and correct data format), SIM receptions can be enabled by setting the receive enable bit, RCV_EN, in the ENABLE register. As bytes are received, they will be placed in the 288 byte deep receive data FIFO. Unread bytes can be accessed from this FIFO at any time. There is no need to disable the receiver to access the FIFO. The FIFO should only be read when the receive FIFO data flag, RFD, in the RCV_STATUS register is set. The RFD flag, which cannot create an interrupt, is high any time there is at least one unread byte in the receive FIFO. If the receive FIFO is read when RFD is low, it will simply produce the last byte read.

The receive data register full flag, RDRF, in the RCV_STATUS register can be used to determine when the receive FIFO has reached a given threshold value. This flag will create an interrupt if the RIM bit in the INT_MASK register is clear. To control at which point RDRF is set, program the receive data threshold, RDT, in the RCV_THRESHOLD register. If the number of unread bytes in the receive FIFO is equal to or greater than the value set by RDT, RDRF will be set.

NOTE

A value of 0x0 in RDT implies that there must be 288 unread bytes in the receive FIFO to trigger RDRF.

The standard flow for receiving bytes from the SIM card is to set RDT to the appropriate value, wait for RDRF to cause an interrupt (RIM clear), and then read bytes out of the receive FIFO as long as RFD is high. In addition to checking RFD between every byte, it is also recommended that software check for the existence of a set OEF flag as well.

11.2.4.2.1 Receive Parity Errors and Parity NACK Generation

The SIM receiver checks every byte received for proper parity. The IC control bit in the DATA_FORMAT register controls whether it checks for odd parity or even parity. When checking for odd parity, the number of logic ones contained in the 9 received bits (8 data bits and 1 parity bit) should be odd. Likewise, when checking for even parity, the number of logic ones contained in the 9 received bits (8 data bits and 1 parity bit) should be even.

If NACK generation on errors is disabled (ANACK = 0 in the CNTL register), the PORTx_PE bit will be set when the SIM receiver detects a parity error, the received data byte and its PORTx_PE bit are placed into FIFO. There is no need to clear the parity error flag in the FIFO. It is simply overwritten the next time a byte is received into that position of the FIFO. A parity error cannot cause an interrupt, and it is up to the software to discard data bytes with parity errors.

If NACK generation on errors is enabled (ANACK =1), the SIM can automatically request the SIM card to re-send a byte that has a parity error by generating a NACK pulse on the SIM XMT pin. In this case, Bytes with parity errors are discarded and will not placed into the FIFO.

To control NACK generation by the SIM receiver use the RTH[3:0], Receive NACK threshold, in the RCV_THRESHOLD register. This set of bits specify the number of consecutive NACK's generated by the SIM module on a received byte, before generating an RTE (receive threshold interrupt flag) in the RCV_STATUS register. The RTE flag would also force the SIM port to power down the card if the SAPD0,1 bit is set in the PORT0,1_CNTL register.

NOTE

The ANACK bit must be set in the CNTL register to enable this feature. The control bit ANACK is also used in initial character mode to enable the retransmission of initial characters in the event that an invalid initial character is received.

When a valid character has been received by the SIM, the internal counter keeping track of the number of NACK's transmitted on the current byte resets to zero. Clearing the receive threshold error (RTE) bit would also clear that counter.

When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETU.

11.2.4.2.2 Receive Frame Errors

The SIM receiver checks every byte received for a proper stop bit. A stop bit should exist during at least the first half of the 11th ETU time after the start of the character. If this is not true, a frame error is flagged. When a frame error is detected on a given byte, the

PORTx_FE bit for that byte is set in the FIFO. The PORTx_FE flag for each byte is read out of the FIFO when the data itself is read. There is no need to clear the frame error flag in the FIFO. It is simply overwritten the next time a byte is received into that position of the FIFO. A frame error cannot cause an interrupt, nor can it create a NACK pulse to the receiver asking for a retransmission of the corrupted data.

11.2.4.2.3 Receive Overrun Errors and Overrun NACK Generation

When there already exists 288 unread bytes in the FIFO, a received character will cause the SIM receiver to flag an overrun condition. This condition will always set the overrun error flag, OEF in the RCV_STATUS register. The received byte will be discarded leaving the 288 unread bytes in the FIFO unaltered.

It is possible for the SIM to automatically request that the SIM card re-send the byte that caused the overrun condition by generating a NACK pulse on the SIM XMT pin. The SIM will generate a NACK pulse for the byte that caused the overrun condition if the control bit ONACK in the CNTL register is set. In this case, the existence of an OEF flag does not indicate the loss of data, but rather a NACK (that is, retransmission request) due to a full receive FIFO. As opposed to transmit NACK generation, there is no limit to the number of times an overrun condition will cause a NACK other than to disable ONACK itself. When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETU.

If the control bit ONACK is clear, the existence of a high OEF flag indicates the loss of data. The OEF flag can create an interrupt if the OIM bit in the INT_MASK register is clear.

To clear the OEF flag, software must simply write a "one" to the OEF bit position in the RCV_STATUS register. A high OEF flag has no effect on the operation of the SIM receiver other than to create an interrupt if OIM is clear.

11.2.4.2.4 Using Initial Character Mode and Resulting Receive Data Formats

The SIM receiver supports the detection of special characters that allow it to determine what data format is being used by the connected SIM card. When placed in initial character mode, the SIM expects to receive one of two potential values that it will use to set the data format control bit, IC, in the DATA_FORMAT register.

The two possible data formats are inverse convention and direct convention. Essentially, inverse convention differs from direct convention in that the order of the data is flipped msb for lsb, and the data bits and parity bit are logically inverted. When receiving inverse convention data, the transformation of the data back to direct convention format is done in hardware, including the inversion of the data and parity bits.

To place the SIM into initial character mode, set the ICM bit in the CNTL register. Once a valid initial character is received, the IC bit in the DATA_FORMAT register will be set accordingly by the hardware, and the ICM bit will be cleared. Software can read the state of the IC bit to determine which mode the SIM is currently using.

The 0x3B (as decoded by direct convention) with parity bit high will cause direct convention to be used (IC set to logic 0), whereas a 0x3F (as decoded by inverse convention) with parity bit high will cause inverse convention to be used (IC set to logic 1).

When the receiver is in initial character mode, all received bytes will continue to be placed into the receive FIFO whether they be valid initial characters or not. If a valid initial character is received that causes the data format being used to change, all subsequent bytes will be decoded with that format before being placed into the FIFO, including the initial character byte itself. That is, if the IC bit is low, and the correct initial character for setting inverse convention is received, that character and all subsequently received characters will be stored in the FIFO after having been decoded using inverse convention (for example, the initial character will be stored as 0x3F).

If the receiver is in initial character mode (ICM is high), and an invalid initial character is received, the SIM can be configured to automatically request that the initial character be retransmitted by setting the ANACK control bit in the CNTL register. When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETUs. The invalid initial character will be placed into the receive FIFO and marked with a parity error to signify that this is an invalid initial character.

11.2.4.2.5 Initial Character Mode Programming Notes

The usage of the initial character mode requires close attention to the programming model. There is a condition where a parity error in the initial byte for direct convention (0x3B) could be decoded as what appears to be a valid initial character for inverse convention (that is, 0x3F). The SIM module will not recognize this as a valid initial character for inverse convention and will mark the character by setting the parity error flag. The software must look for the existence of a parity error before recognizing a character as a valid initial character.

11.2.4.2.6 Automatic Receiver Mode

The SIM module has an automatic receive mode that inhibits the data being transmitted by the SIM module from entering the SIM receive buffer through the feedback path of the SIM data pin. The SIM module receiver should normally be enabled while the transmitter is operational. Automatic receive mode saves the software from having to actively manage the transition from transmitter to receiver. The auto receive mode is always active whenever the receiver is enabled.

11.2.4.2.7 Using the SIM Receiver with "T=1" SIM Cards

The SIM module provides hardware support for "T=1" type SIM cards. These type of cards present several requirements above and beyond the standard "T=0" cards. The features provided to meet the requirements that pertain to the SIM receiver are as follows:

- 11 ETU Characters
  - "T=1" cards can transmit with character lengths of 11 ETUs (that is, one STOP bit). The SIM module provides the RCVR11 bit in the GUARD_CNTL register in order to configure the receiver state machine to accept 11 ETU characters.
- Character Wait Time Counter
  - The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters received from the SmartCard. The value of CWT can range from 12 ETU to 32779 ETU. The SIM module provides a 16-bit counter with programmable comparator clocked at the ETU bit rate to identify when the CWT has been exceeded by the SIM card.
- Block Waiting Time
  - The block waiting time (BWT) is defined as the maximum time between the start bits of the last character of a transmitted block and the first character of the next received block. The block wait time must not exceed a value that is programmable. The SIM module provides a 32-bit Block Wait Timer that can be used to identify when the BWT has been violated (divided in two registers).
- Block Guard Time
  - The block guard time (BGT) is defined as the minimum delay between the start bits of the last character of a transmitted block and the first character of the next received block. The block guard must be greater than a value that is programmable. The SIM module provides a 16-bit Block Guard Timer that can be used to identify when the BGT has been violated.
- Error Detection Code
  - "T=1" cards can specify LRC or CRC error detection codes to be used. The SIM module provides hardware support for both the LRC and CRC operations.

11.2.4.3 Using the SIM Transmitter

Once the SIM has been properly configured (such as data XMT enabled, correct baud rate, and correct data format), the transmitter can be enabled by setting the XMT_EN bit in the ENABLE register. If data had been previously written to the transmit FIFO, the transmitter will begin to send the first character. If no data is written to the transmit FIFO before enabling the transmitter, then the transmitter will wait until the first character is written before beginning transmission. Clearing the XMT_EN bit while the transmitter is in operation, will halt any transmission in progress, flush the transmit FIFO, and reset the transmit state machine.

Data can be written to the transmit FIFO at any time.

The transmit data threshold flag, TDTF, in the XMT_STATUS register can be used to determine when the transmit FIFO has reached a given threshold value. This flag will create an interrupt if the TDTFM bit in the INT_MASK register is clear. To control at which point TDTF is set, program the transmit data threshold, TDT[3:0], in the XMT_THRESHOLD register. If the number of bytes remaining in the transmit FIFO is equal to or less than the value set by TDT[4:0], TDTF will be set.

NOTE

A value of 0x0 in TDT[3:0] implies that the transmit FIFO must be empty to trigger TDTF.

The value in TDT[3:0] can be changed at any time to alter this threshold level. The comparison between the number of remaining bytes in the transmit FIFO and the value set by TDT[3:0] is continuously updated so that any change in either will be immediately reflected in the state of TDTF. Unlike the RDRF flag for the receive FIFO, TDTF is latched and remains set until the software writes a 1 to the TDTF bit location in the XMT_STATUS register. For instance, if TDT[3:0] is set to 5, and there are 6 bytes remaining in the FIFO, changing TDT[3:0] to 6 will immediately cause TDTF to be set. However, setting TDT[3:0] back to 5 will not cause TDTF to clear.

The standard flow for transmitting bytes from the SIM card is to set TDT[3:0] to the appropriate value, write up to 16 bytes to the transmit FIFO, enable the transmitter, wait for TDTF to cause an interrupt (TDTFM clear), and then write additional bytes to the transmit FIFO.

NOTE

For the SIM module to transmit successfully, the transmit pin must be connected to the receive pin of the same port. This connection is necessary for the SIM to decode transmit NACKs sent to it by the SIM card. Without this connection, all transmissions will appear to have a NACK thereby causing the first byte to be transmitted continuously. When operating in

external one wire interface mode (3VOLT0 or 3VOLT1), this connection is made internal to the SoC.

11.2.4.3.1 Transmit Data Formats

There are two possible data formats that the SIM module uses when transferring data to the SIM: card - inverse convention or direct convention. The format used depends on the state of inverse convention bit (IC in the DATA_FORMAT register). Software can set the IC bit, or it can be set automatically by hardware when using the initial character mode.

11.2.4.3.2 Transmit NACK

The SIM transmitter can respond to NACKs created by the SIM card. A NACK will be decoded if the SIM card creates a logic low level on the SIM receive pin during the STOP bit time at the end of a transmitted byte. To prevent a situation where the SIM interface is stalled by an infinite number of NACK pulses on a given byte, the SIM module can be configured to limit the number of times it will respond to NACKs. The XTH[3:0] control field in the XMT_THRESHOLD register allows software to set a threshold on the number of times a given byte will be retransmitted. If the threshold is reached, a transmit threshold error, XTE in the XMT_STATUS register, is asserted.

When XTE is set, the SIM transmitter is halted, and all pending transfers are aborted, and the TC, ETC, AND TFE flags are set. All bytes remaining in the transmit FIFO are lost. There is no way to restart the transmission on the next byte in the FIFO. The transmitter remains frozen until XTE is cleared by software. The only way to clear XTE is to write a 1 to the XTE bit in the XMT_THRESHOLD register. The XTE flag can create an interrupt if the XTM mask in the XMT_THRESHOLD register is clear.

It is possible to disable the detection of NACKs from the SIM card by setting XTH[3:0] to 0x0. By setting XTH[3:0] to 0x1, it is possible to disable all retransmissions while still setting XTE on the first NACK received. In general, XTE is set on the NACK that causes the threshold set by XTH[3:0] to be reached. This final NACK will not cause a retransmission, whereas all previous NACKs will cause a retransmission.

11.2.4.3.3 Transmit Guard Time

The time between data bytes sent from the SIM transmitter can be altered using the transmit guard time control. By default, the minimum time between start bits of successive transmitted bytes is 12 ETUs (Elementary Time Units). An ETU is equivalent in time to 1 bit time. Therefore, the amount of time an ETU consumes is determined by the baud rate chosen. The 12 ETUs that form the default character transmission time consist of a start bit, 8 data bits, a parity bit and two stop bits. The number of stop bits (idle bits) can be extended by an integer number of ETUs. The number of additional

ETUs can be programmed directly into the GETU[7:0] bits of the GUARD_CNTL register. Programming the GETU[7:0] bits to 0xFF will configure the SIM transmitter to use only one stop bit for each character transmission.

11.2.4.3.4 Using SIM Transmit with "T=1" SIM Cards

The SIM module provides hardware support for "T=1" type SIM cards. These type of cards present several requirements above and beyond the standard "T=0" cards. The features provided to meet the requirements that pertain to the SIM transmitter are as follows:

- 11 ETU Characters
  - The SIM module transmitter has a programmable guard time register that allows the programmer to specify the number of ETUs between character transmissions. Programming a value of 255 (0xFF) in the GETU[7:0] bits in the GUARD_CNTL register will set the number of ETUs per character transmitted to 11.
- Character Waiting Time
  - The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters. The value of CWT can range from 12 ETU to 32779 ETU. The time between transmitted characters is controlled by the programmable guard time in the GUARD_CNTL register. However, the time between the last byte in the transmit FIFO, and the next transmitted byte can be largely affected by software response time to the transmit interrupts. The SIM transmitter provides a Transmit FIFO threshold (TDTF) interrupt to signal the system when the expected number of characters have been transmitted from the transmit FIFO. The minimum CWT is achieved only if the software can respond to the TDTF interrupt and write new data to the transmit FIFO before the last character in the Transmit FIFO has been sent.
- Block Waiting Time
  - The block waiting time (BWT) is defined as the maximum time between the start bits of the last character of a transmitted block and the first character of the next received block. The value of BWT is always greater than 1800 ETU. The SIM transmitter provides a General Purpose Counter that can be used to track the BWT. The BWT is purely determined by software response time to the transmit interrupts.
- Block Guard Time
  - The block guard time (BGT) is defined as the minimum delay between the start bits of the last character of a transmitted block and the first character of the next received block. The value of BGT is 22 ETU. The SIM module supports the BGT by providing the ability to generate an interrupt when the last byte is received, and transmitting within 2 ETU after the XMT_EN bit is set. The BGT

will be determined by the speed at which the software can react to an interrupt and enable the transmitter.

- Error Detection Code
  - "T=1" cards can specify LRC or CRC error detection codes to be used. The SIM module provides hardware support for both the LRC and CRC operation.

11.2.4.4 Suggested "T=1" Compliant Programming Model

This section describes the suggested programming model for supporting "T=1", "T=0", and known "special" cards using the SIM module on the device. This should be used as a rough guide for how to configure the SIM module for use with SIM cards specified by ISO 7816-3 and EMV. Some details are not addressed. Other uses for some of the SIM features are not included (for example, GP Counter uses for some ISO timing requirements).

11.2.4.4.1 Answer To Reset (ATR) Detection

The first step to communicating with a SIM card is to provide power and a clock signal to the card. Once the card is detected as present (using the presence detect features, or some other method), the SIM card should be powered up according to the power up sequence specified in the ISO 7816-3 specification.

1. Apply voltage to the SIM card by setting the SVEN0 or SVEN1 bit in the PORTx_CNTL register
2. Select the appropriate clock frequency for the SIM card by programming the CLK_PRESCALER[7:0] register.
3. Enable the clock to the SIM card by setting the SCEN0 or SCEN1 bit in the PORTx_CNTL register.
4. Remove the card from reset by setting the SRST0 or SRST1 bit in the PORTx_CNTL register.

   The first communication between the SIM card and the SIM module will be a block of data sent from the SIM card to the SIM module after the card is powered and the card reset is removed. This block is called the Answer To Reset (ATR). To receive the ATR, the SIM module should be configured for 12 ETU character reception. According to the ISO 7816-3 spec, both "T=0" and "T=1" cards will communicate initially using 12 ETU character durations.

NOTE

We are aware of some card manufacturers that communicate at 11.5 ETU character durations (Geldkarte).

This will complicate the initial card detection sequence shown below.

5. Clear RCVR11 bit in the GUARD_CNTL register

   The next item to configure for ATR reception is the NACK capability of the SIM module. The ISO 7816-3 spec allows the SIM module to NACK any communication errors that occur during the initial communication at 12 ETU.

NOTE

The Europay Mastercard and VISA (EMV) cards are similar to "T=1", but do not allow the SIM module to NACK during the initial communication. This will again complicate the initial card detection sequence shown below.

6. Set ANACK bit in the CNTL register to enable NACK generation

   In order for the SIM module to notify when characters are received, the receive interrupt(s) can be enabled. A threshold can be set for the number of characters to receive before generating an interrupt.

7. Enable RDRF and OEF interrupts by setting RIM and OIM bits in INT_MASK register
8. Set desired threshold for received characters by writing RDT in RCV_THRESHOLD register.

   The SIM should be setup to perform in initial character mode. This will cause the hardware to identify the first valid character sent during the ATR as an initial character. This character will automatically configure the hardware for the data convention used by the SIM card.

9. Set Initial Character Mode by setting the ICM bit in the CNTL register

   The ISO 7816-3 spec requires that SIM cards meet certain timing restrictions. One of these is the time from the deassertion of the card reset to the beginning of the ATR sequence. The SIM module General Purpose Counter can be used to verify that the SIM card begins its ATR within the 400 to 40,000 clock cycle range.

1. Set General Purpose Counter Comparator to 0x9C40 using GPCNT register.
2. Enable General Purpose Counter Interrupt by clearing GPCNTM in INT_MASK register.
3. Enable General Purpose Counter by programming the GPCNT_CLKSEL[1:0] bits to 01 so the card clock is used for counting.

The ISO7816-3 spec states that the maximum allowed time between two characters during the ATR is 9600 ETUs (Initial Waiting Time). The Character Wait Time Counter should be setup to detect any errors for this condition.

1. Set Character Wait Time Counter Comparator to 9600 using the CHAR_WAIT register
2. Enable the Character Wait Time Counter Interrupt by clearing CWTM in INT_MASK register
3. Enable the Character Wait Time Counter by setting the CWTEN bit in the CNTL register

   The last step in preparing for ATR reception is to enable the receiver.

4. Set RCV_EN bit in ENABLE register

The SIM module will generate interrupts once a threshold number of characters is received. The software should react to these interrupts and read the characters from the receive FIFO (RCV_BUF) until the complete ATR has been received. If a General Purpose Counter interrupt occurs before the final ATR character is received, then the card should be deactivated according to the ISO 7816-3 spec. Otherwise, once a valid ATR is received, the software will know from the ATR information the specific characteristics for this card (refer to the ISO 7816-3 spec for details).

11.2.4.4.2 Programming Considerations for Geldkarte Cards

There is at least one manufacturer of SIM cards (Geldkarte) that we know of that does not send the ATR in 12 ETU mode or support NACKs. This creates an issue with detecting a valid ATR from a SIM card. Since any kind of card can be attached to the SIM module, the software and hardware do not know how to begin communication. Basically, if the card fails to send a valid ATR on the first try, disable NACKs, configure the SIM module for 11 ETU and try again. The software should toggle between 12 and 11 ETU modes with and without NACKs enabled until a valid ATR is received, or the number of attempts to communicate passes a predetermined error threshold. The figure below shows the flow chart for the suggested Geldkarte Compliant SIM Initialization.

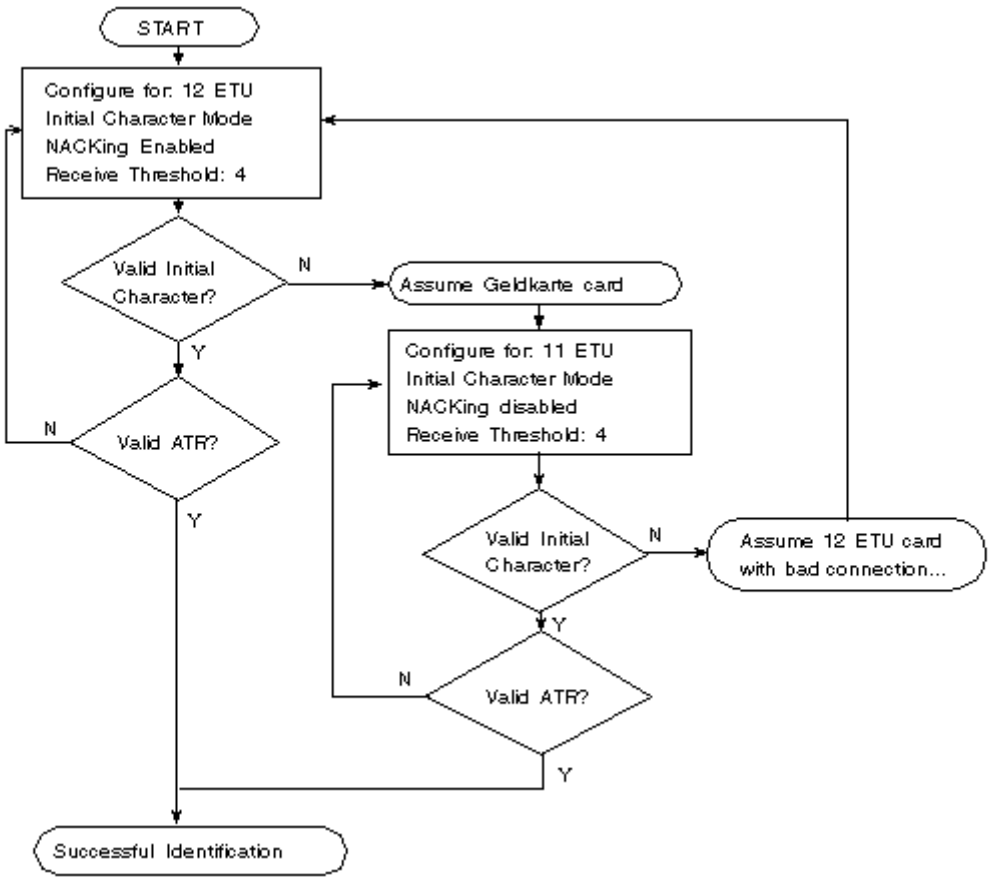


Figure 11-51. Suggested "T=1", EMV, Geldkarte Compliant SIM Initialization

11.2.4.4.3 Programming Considerations for T=0 SIM Cards

If the card is of type T=0, the software should adjust the following parameters according to the information in the ATR:

1. Adjust the baud rate by changing the values of BAUD_SEL[2:0] and SAMPLE12 in the CNTL register
2. Adjust the guard time between characters by changing the value of GETU[7:0] in the GUARD_CNTL register
3. Adjust NACK capability by modifying the values of the ONACK and ANACK bits in the CNTL register
4. Adjust the stop clock polarity by modifying the values of the SCSP0 or SCSP1 bits in the PORTx_CNTL register
5. Adjust the level of transmit NACK re-transmissions allowed by modifying the value of the XTH[3:0] bits in the XMT_THRESHOLD register
6. Adjust the level for the Receive NACK threshold by modifying the RTH[3:0] bits in the RCV_THRESHOLD register.

If a negotiation with the SIM card is desired, the software sends a PPS response to the SIM card. To send the response, the following steps should be performed:

1. Set the desired transmit FIFO threshold level by writing the TDT[3:0] bits in the XMT_THRESHOLD register
2. Write the characters to be sent as response (max 16) to the transmit FIFO using the XMT_BUF register
3. Clear all transmit interrupt flags in the XMT_STAT register by writing a one to them
4. Enable the transmit interrupts desired by clearing the mask bits in the INT_MASK register. If more than 16 character are to be sent, it is suggested that the TDTF interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the SIM card.
5. Enable the transmitter by setting the XMT_EN bit in the ENABLE register

At this point, the SIM module will transmit the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt will be set when the threshold number of characters are remaining in the FIFO. The software can then write an additional number of characters to be sent without interrupting transmission to the SIM card.

Once the transmission is complete, the SIM module should be completely configured for standard operation with the T=0 SIM card. The software can continue to service RDRF interrupts for received characters, and TDTF interrupts for transmitted characters.

11.2.4.4.4 Programming Considerations for T=1 SIM Cards

If the card is of type T=1, the software should adjust the following parameters according to the information in the ATR:

1. Adjust the baud rate by changing the values of BAUD_SEL[2:0] and SAMPLE12 in the CNTL register
2. Adjust the guard time between characters by changing the value of GETU[7:0] in the GUARD_CNTL register. Setting GETU[7:0] to 0xFF configures the SIM transmitter for 11 ETU transmissions
3. Disable NACK capability by clearing the ONACK and ANACK bits in the CNTL register. T=1 cards do not allow NACKs.
4. Adjust the stop clock polarity by modifying the values of the SCSP0 or SCSP1 bits in the PORTx_CNTL register
5. Set Character Wait Time Counter Comparator to value specified in the ATR by using the CHAR_WAIT register
6. Enable the Character Wait Time Counter Interrupt by clearing CWTM in INT_MASK register

7. Enable the Character Wait Time Counter by setting the CWTEN bit in the CNTL register
8. Enable CRC or LRC error checking according to the ATR information by setting either the CRCEN or LRCEN bit in the CNTL register. These bits will never be set at the same time!

   For T=1 cards, the ATR is sent using a T=0 type of structure (12 ETU, no LRC or CRC). If a negotiation with the SIM card is desired, the software will send a PPS response to the SIM card. Otherwise, the protocol is initiated with a block transfer from the SIM module. In order to send the response or the first block, the following steps should be performed:

9. Set the desired transmit FIFO threshold level by writing the TDT[3:0] bits in the XMT_THRESHOLD register
10. Write the characters to be sent as response (max 16) to the transmit FIFO using the XMT_BUF register
11. Clear all transmit interrupt flags in the XMT_STAT register by writing a 1 to them
12. Enable the transmit interrupts desired by clearing the mask bits in the INT_MASK register. If more than 16 character are to be sent, it is suggested that the TDTF interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the SIM card.
13. Enable the transmission of the error checking characters (LRC or CRC) by setting the XMT_CRC_LRC bit in the CNTL register.

NOTE

If the card supports PPS, the software may not be allowed to send the LRC/CRC information until the PPS exchange is completed. If so, do not set the XMT_CRC_LRC bit during the PPS exchange.

14. Enable the transmitter by setting the XMT_EN bit in the ENABLE register

At this point, the SIM module will transmit the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt will be set when the threshold number of characters are remaining in the FIFO. The software can then write an additional number of characters to be sent without interrupting transmission to the SIM card.

Once the transmission is complete, the SIM module should be completely configured for standard operation with the T=1 SIM card. The software can continue to service RDRF interrupts for received characters, and TDTF interrupts for transmitted characters.

11.2.5 SIM Memory Map/Register Definition

SIM memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30B9_0000	SIM Port1 Control Register (SIM1_PORT1_CNTL)	32	R/W	0000_0000h	11.2.5.1/ 3061
30B9_0004	SIM Setup Register (SIM1_SETUP)	32	R/W	0000_0000h	11.2.5.2/ 3063
30B9_0008	SIM Port 1 Detect Register (SIM1_PORT1_DETECT)	32	R/W	0000_0001h	11.2.5.3/ 3063
30B9_000C	SIM Transmit Buffer Register (SIM1_XMT_BUF)	32	R/W	0000_0000h	11.2.5.4/ 3065
30B9_0010	SIM Receive Buffer Register (SIM1_RCV_BUF)	32	R	0000_0000h	11.2.5.5/ 3066
30B9_0014	SIM Port0 Control Register (SIM1_PORT0_CNTL)	32	R/W	0000_0000h	11.2.5.6/ 3067
30B9_0018	SIM Control Register (SIM1_CNTL)	32	R/W	0000_0006h	11.2.5.7/ 3069
30B9_001C	SIM Clock Prescaler Register (SIM1_CLK_PRESCALER)	32	R/W	0000_0002h	11.2.5.8/ 3071
30B9_0020	SIM Receive Threshold Register (SIM1_RCV_THRESHOLD)	32	R/W	0000_0001h	11.2.5.9/ 3072
30B9_0024	SIM Enable Register (SIM1_ENABLE)	32	R/W	0000_0000h	11.2.5.10/ 3073
30B9_0028	SIM Transmit Status Register (SIM1_XMT_STATUS)	32	w1c	0000_00B8h	11.2.5.11/ 3075
30B9_002C	SIM Receive Status Register (SIM1_RCV_STATUS)	32	w1c	0000_0042h	11.2.5.12/ 3077
30B9_0030	SIM Interrupt Mask Register (SIM1_INT_MASK)	32	R/W	0000_3FFFh	11.2.5.13/ 3079
30B9_003C	SIM Port0 Detect Register (SIM1_PORT0_DETECT)	32	R/W	0000_0001h	11.2.5.14/ 3081
30B9_0040	SIM Data Format Register (SIM1_DATA_FORMAT)	32	R/W	0000_0000h	11.2.5.15/ 3083
30B9_0044	SIM Transmit Threshold Register (SIM1_XMT_THRESHOLD)	32	R/W	0000_0000h	11.2.5.16/ 3083
30B9_0048	SIM Transmit Guard Control Register (SIM1_GUARD_CNTL)	32	R/W	0000_0000h	11.2.5.17/ 3084
30B9_004C	SIM Open Drain Configuration Control Register (SIM1_OD_CONFIG)	32	R/W	0000_0000h	11.2.5.18/ 3085
30B9_0050	SIM Reset Control Register (SIM1_RESET_CNTL)	32	R/W	0000_0000h	11.2.5.19/ 3086
30B9_0054	SIM Character Wait Time Register (SIM1_CHAR_WAIT)	32	R/W	0000_FFFFh	11.2.5.20/ 3087
30B9_0058	SIM General Purpose Counter Register (SIM1_GPCNT)	32	R/W	0000_FFFFh	11.2.5.21/ 3088
30B9_005C	SIM Divisor Register (SIM1_DIVISOR)	32	R/W	0000_00FFh	11.2.5.22/ 3088

Table continues on the next page...

SIM memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30B9_0060	SIM Block Wait Time Register (SIM1_BWT)	32	R/W	0000_FFFFh	11.2.5.23/ 3089
30B9_0064	SIM Block Guard Time Register (SIM1_BGT)	32	R/W	0000_0000h	11.2.5.24/ 3089
30B9_0068	SIM Block Wait Time Register HIGH (SIM1_BWT_H)	32	R/W	0000_FFFFh	11.2.5.25/ 3090
30B9_006C	SIM Transmit FIFO Status Register (SIM1_XMT_FIFO_STAT)	32	R	0000_0000h	11.2.5.26/ 3090
30B9_0070	SIM Receive FIFO Counter Register (SIM1_RCV_FIFO_CNT)	32	R	0000_0000h	11.2.5.27/ 3091
30B9_0074	SIM Receive FIFO Write Pointer Register (SIM1_RCV_FIFO_WPTR)	32	R	0000_0000h	11.2.5.28/ 3092
30B9_0078	SIM Receive FIFO Read Pointer Register (SIM1_RCV_FIFO_RPTR)	32	R	0000_0000h	11.2.5.29/ 3092
30BA_0000	SIM Port1 Control Register (SIM2_PORT1_CNTL)	32	R/W	0000_0000h	11.2.5.1/ 3061
30BA_0004	SIM Setup Register (SIM2_SETUP)	32	R/W	0000_0000h	11.2.5.2/ 3063
30BA_0008	SIM Port 1 Detect Register (SIM2_PORT1_DETECT)	32	R/W	0000_0001h	11.2.5.3/ 3063
30BA_000C	SIM Transmit Buffer Register (SIM2_XMT_BUF)	32	R/W	0000_0000h	11.2.5.4/ 3065
30BA_0010	SIM Receive Buffer Register (SIM2_RCV_BUF)	32	R	0000_0000h	11.2.5.5/ 3066
30BA_0014	SIM Port0 Control Register (SIM2_PORT0_CNTL)	32	R/W	0000_0000h	11.2.5.6/ 3067
30BA_0018	SIM Control Register (SIM2_CNTL)	32	R/W	0000_0006h	11.2.5.7/ 3069
30BA_001C	SIM Clock Prescaler Register (SIM2_CLK_PRESCALER)	32	R/W	0000_0002h	11.2.5.8/ 3071
30BA_0020	SIM Receive Threshold Register (SIM2_RCV_THRESHOLD)	32	R/W	0000_0001h	11.2.5.9/ 3072
30BA_0024	SIM Enable Register (SIM2_ENABLE)	32	R/W	0000_0000h	11.2.5.10/ 3073
30BA_0028	SIM Transmit Status Register (SIM2_XMT_STATUS)	32	w1c	0000_00B8h	11.2.5.11/ 3075
30BA_002C	SIM Receive Status Register (SIM2_RCV_STATUS)	32	w1c	0000_0042h	11.2.5.12/ 3077
30BA_0030	SIM Interrupt Mask Register (SIM2_INT_MASK)	32	R/W	0000_3FFFh	11.2.5.13/ 3079
30BA_003C	SIM Port0 Detect Register (SIM2_PORT0_DETECT)	32	R/W	0000_0001h	11.2.5.14/ 3081
30BA_0040	SIM Data Format Register (SIM2_DATA_FORMAT)	32	R/W	0000_0000h	11.2.5.15/ 3083

Table continues on the next page...

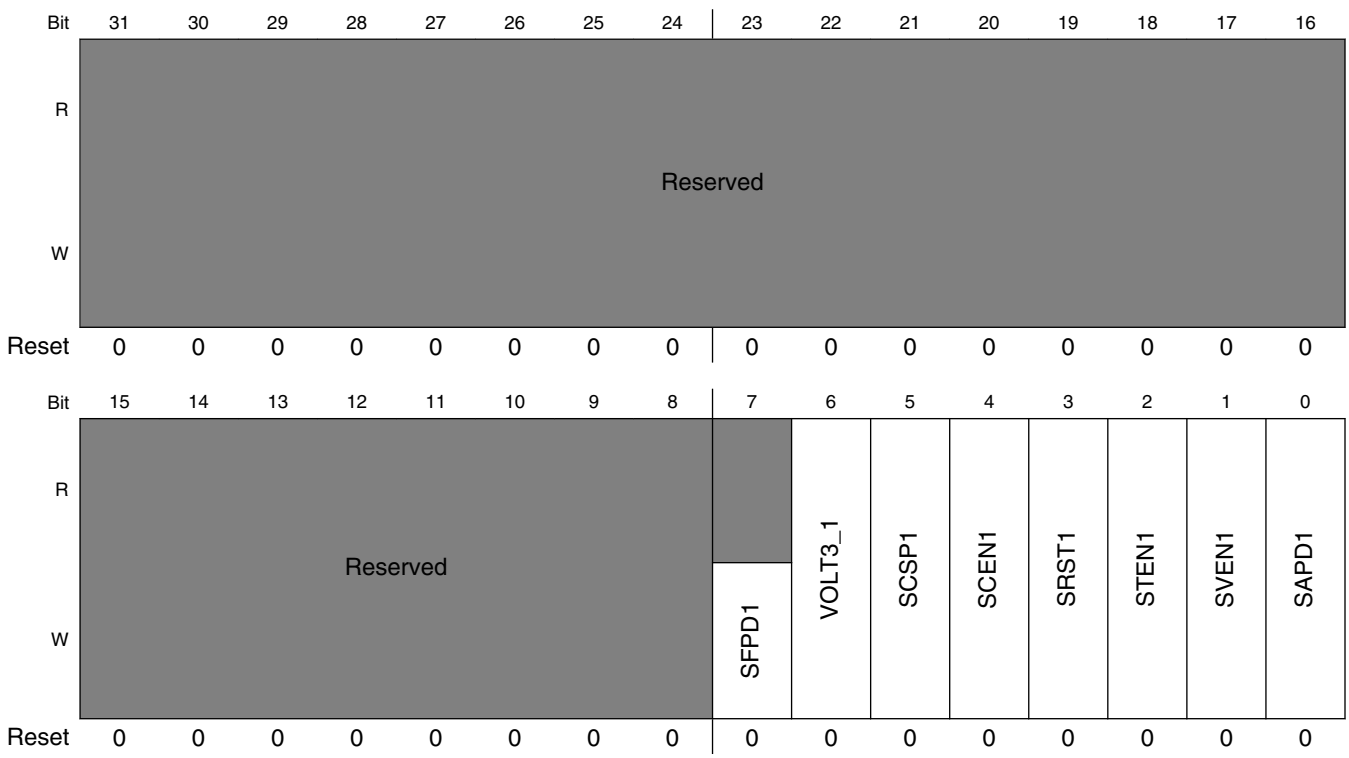
SIM memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30BA_0044	SIM Transmit Threshold Register (SIM2_XMT_THRESHOLD)	32	R/W	0000_0000h	11.2.5.16/ 3083
30BA_0048	SIM Transmit Guard Control Register (SIM2_GUARD_CNTL)	32	R/W	0000_0000h	11.2.5.17/ 3084
30BA_004C	SIM Open Drain Configuration Control Register (SIM2_OD_CONFIG)	32	R/W	0000_0000h	11.2.5.18/ 3085
30BA_0050	SIM Reset Control Register (SIM2_RESET_CNTL)	32	R/W	0000_0000h	11.2.5.19/ 3086
30BA_0054	SIM Character Wait Time Register (SIM2_CHAR_WAIT)	32	R/W	0000_FFFFh	11.2.5.20/ 3087
30BA_0058	SIM General Purpose Counter Register (SIM2_GPCNT)	32	R/W	0000_FFFFh	11.2.5.21/ 3088
30BA_005C	SIM Divisor Register (SIM2_DIVISOR)	32	R/W	0000_00FFh	11.2.5.22/ 3088
30BA_0060	SIM Block Wait Time Register (SIM2_BWT)	32	R/W	0000_FFFFh	11.2.5.23/ 3089
30BA_0064	SIM Block Guard Time Register (SIM2_BGT)	32	R/W	0000_0000h	11.2.5.24/ 3089
30BA_0068	SIM Block Wait Time Register HIGH (SIM2_BWT_H)	32	R/W	0000_FFFFh	11.2.5.25/ 3090
30BA_006C	SIM Transmit FIFO Status Register (SIM2_XMT_FIFO_STAT)	32	R	0000_0000h	11.2.5.26/ 3090
30BA_0070	SIM Receive FIFO Counter Register (SIM2_RCV_FIFO_CNT)	32	R	0000_0000h	11.2.5.27/ 3091
30BA_0074	SIM Receive FIFO Write Pointer Register (SIM2_RCV_FIFO_WPTR)	32	R	0000_0000h	11.2.5.28/ 3092
30BA_0078	SIM Receive FIFO Read Pointer Register (SIM2_RCV_FIFO_RPTR)	32	R	0000_0000h	11.2.5.29/ 3092

11.2.5.1 SIM Port1 Control Register (SIMx_PORT1_CNTL)

See the figure below for illustration of valid bits in the SIM Port1 Control Register and the table below for description of the bit fields.

Address: Base address + 0h offset



SIMx_PORT1_CNTL field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
7 SFPD1	Auto Power Down port1. Writing a "1" to this location will start the auto power down sequence for port 1. This bit will autoclear. A read of this bit will give a "0". 0 No effect 1 Start Auto Power down
6 VOLT3_1	External one wire interface for SIM Card port1. Used to configure the Port 1 transmit pin as bi-directional. This allows the Port 1 Receive pin to be re-used as General Purpose I/O. This operation is restricted to bi-directional data SIM cards only. This bit should not be changed while the receiver or transmitter is enabled! 0 Port1 uses both rcv and xmt pins 1 Port1 XMT pin bidirectional. Port1 RCV PIN unused

Table continues on the next page...

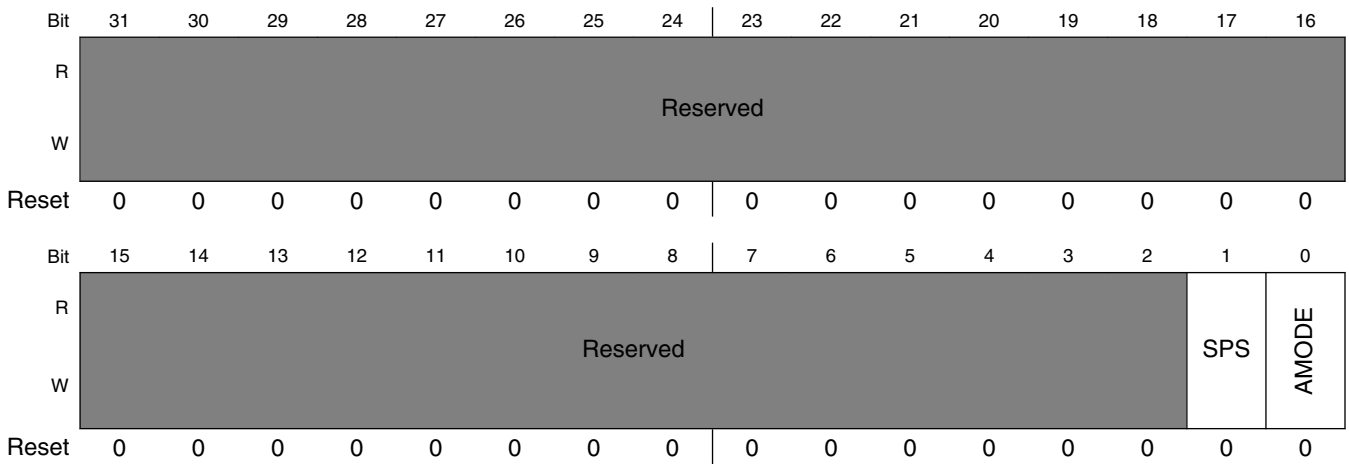
SIMx_PORT1_CNTL field descriptions (continued)

Field	Description
5 SCSP1	SIM Card Clock Stop Polarity Port1. Used to control the polarity of the idle SIM clock when the clock is disabled by SCEN1. It will be forced low by hardware during the auto power down sequence. This forces the clock be a logic 0 when stopped by auto power down as required by ISO 7816 spec. 0 Clock is logic 0 when stopped by SCEN1 1 Clock is logic 1 when stopped by SCEN1
4 SCEN1	SIM card Clock Enable Port 1. Used to enable/disable the clock to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 SIM Card Clock Disabled Port1 1 SIM Card Clock Enabled Port1
3 SRST1	SIM card Reset. Used to control state of reset line to the SIM card. It can be forced low by hardware during the auto power down sequence. SIM card reset signals are active low. 0 SIM Card reset Port1 inactive (default) 1 SIM Card reset Port1 active
2 STEN1	SIM card Transmit Enable Port 1. Used to enable/disable the XMT data to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 Port1 Transmit Data is forced to zero (default) 1 Port 1 Transmit Data controlled by SIM module
1 SVEN1	SIM card Vcc Enable Port 1. Used to control the state of the SVEN1 pin on SIM card port 1. The SVEN1 pin controls the SIM card V_{CC} enable in the power management chip. It can be forced low by hardware during the auto power down sequence. 0 SIM card Voltage Port 1 disabled (default) 1 SIM card Voltage Port 1 enabled
0 SAPD1	SIM card Auto Power Down Port 1. Used to enable/disable the auto power down function for port 1. It will be forced low at the end of the auto power down sequence. 0 Auto power down Port 1 disabled (default) 1 Auto power down Port 1 enabled

11.2.5.2 SIM Setup Register (SIMx_SETUP)

See the figure below for illustration of valid bits in the SIM Setup Register and the table below for description of the bit fields.

Address: Base address + 4h offset



SIMx_SETUP field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
1 SPS	**SIM card Port Select.** Controls which port the SIM interface uses. **NOTE:** The AMODE bit must be zero when the SPS bit is set to 1. 0 Port 0 Enabled (default) 1 Port 1 Enabled
0 AMODE	**Alternate SIM Card Mode enable.** Enables an alternate SIM module to control SIM card Port 1. **NOTE:** The SPS bit must be 0 to give the alternate SIM module control. 0 Alternate Port Disabled (default) 1 Alternate Port Enabled

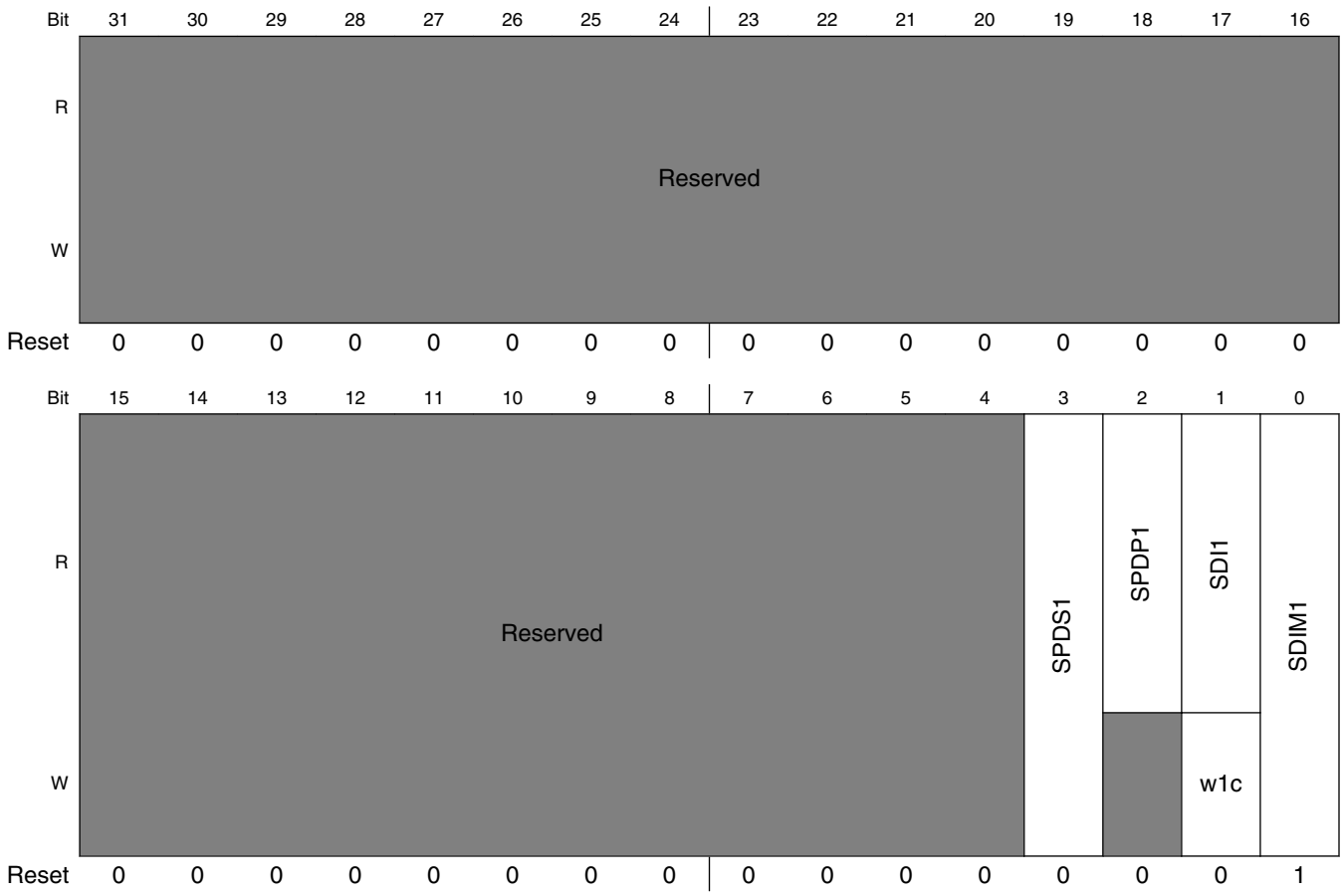
11.2.5.3 SIM Port 1 Detect Register (SIMx_PORT1_DETECT)

See the figure below for illustration of valid bits in the SIM Port1 Detect Register and the table below for description of the bit fields.

Summary:

spds1 determines which edge transition of the SIMPD1 pin is used for SIM card presence detection. Presence detection can be used to determine if the card has been inserted or removed. The occurrence of the SIMPD1 edge specified by spds1 will cause the following: sdi1 to be set; if the sdim1 mask is clear, an interrupt on SIMIRQ_N; and if sapd1 in the PORT1_CNTL register is set, an auto power down sequence to begin. If SIM card insertion is expected, sapd1 can be set low to avoid the auto power down sequence. There is no auto power up sequence. The bit spdp1 can be used to determine the current state of the SIMPD1 pin.

Address: Base address + 8h offset



SIMx_PORT1_DETECT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved
3 SPDS1	SIM **Presence Detect Select Port 1.** Controls which edge of the SIMPD1 pin is used to detect the presence of the SIM card. 0 Falling edge of SIMPD1 Input (default) 1 Rising edge of SIMPD1 Input
2 SPDP1	**SIMPD1 input pin status.** This bit reflects the state of the SIMPD1 pin. It is not a latched register bit, but instead a synchronized version of the state of the SIMPD1 pin itself.

Table continues on the next page...

SIMx_PORT1_DETECT field descriptions (continued)

Field	Description
	0 SIMPD1 pin is logic low 1 SIMPD1 pin is logic high
1 SDI1	**SIM Detect Interrupt Flag Port 1.** Status flag to indicate the insertion or removal of a SIM card has been detected on port 1. Can create an interrupt to the MCU if SDIM1 is low. Write a "1" to this bit to clear. 0 No insertion or removal of SIM card detected on Port 1(default) 1 Insertion or removal of SIM card detected on Port 1
0 SDIM1	**SIM Detect Interrupt Mask Port 1.** Interrupt mask for the sdi1 interrupt flag. 0 SDI1 enabled 1 SDI1 masked (default)

11.2.5.4 SIM Transmit Buffer Register (SIMx_XMT_BUF)

See the figure below for illustration of valid bits in the SIM Transmit Buffer Register and the table below for description of the bit fields.

Address: Base address + Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																								XMT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

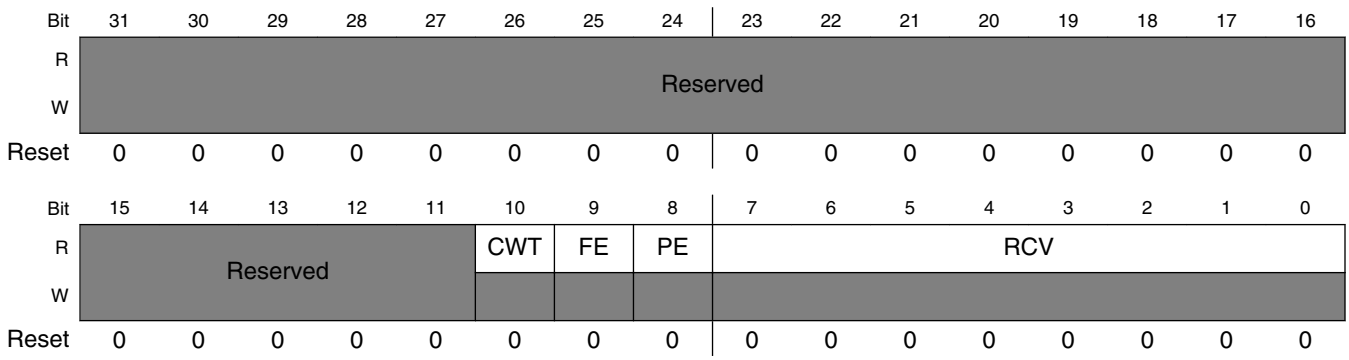
SIMx_XMT_BUF field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
XMT	**Transmit Buffer.** Write to the next available location in the transmit buffer.

11.2.5.5 SIM Receive Buffer Register (SIMx_RCV_BUF)

See the figure below for illustration of valid bits in the SIM Receive Buffer Register and the table below for description of the bit fields.

Address: Base address + 10h offset



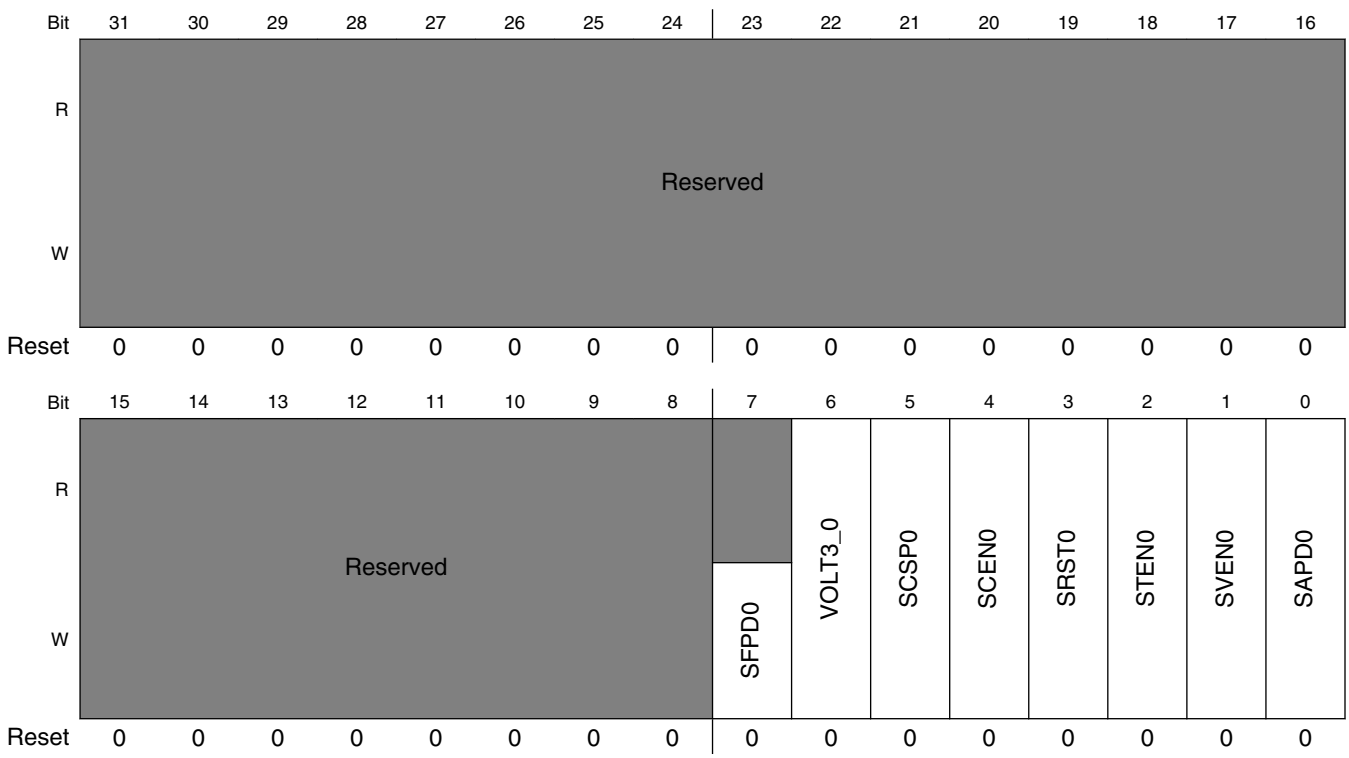
SIMx_RCV_BUF field descriptions

Field	Description
31–11 -	This field is reserved. Reserved
10 CWT	**CWT flag.** The CWT indicates that this byte was late. It is not necessary to clear the byte since it is overwritten by the next byte received into that location of the FIFO. 0 Byte was on time 1 Byte was late
9 FE	**Frame Error flag.** The FE flag indicates whether a frame error was detected during the reception of the corresponding byte read in the RCV field. The FE flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. 0 Byte contains no framing error (default) 1 Byte contains a framing error
8 PE	**Parity Error flag.** The PE flag indicates whether a parity error was detected during the reception of the corresponding byte read in the RCV field. The PE flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. A parity error can create a NACK pulse. 0 Byte contains no parity error (default) 1 Byte contains a parity error
RCV	**Receive buffer.** Read from the next location in the receive buffer.

11.2.5.6 SIM Port0 Control Register (SIMx_PORT0_CNTL)

See the figure below for illustration of valid bits in the SIM Port0 Control Register and the table below for description of the bit fields.

Address: Base address + 14h offset



SIMx_PORT0_CNTL field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
7 SFPD0	Auto Power Down port0. Writing a "1" to this location will start the auto power down sequence for port 0. This bit will autoclear.A read of this bit will give a "0". 0 No effect 1 Start Auto Power down
6 VOLT3_0	External one wire interface for SIM Card port0. Used to configure the Port 0 transmit pin as bi-directional. Only bi-directional data SIM cards are supported. Hence user should program this bit to 1. This bit should not be changed while the receiver or transmitter is enabled! Port0 uses both RCV and XMT pins 1 Port0 XMT pin bidirectional. Port0 rcv pin unused

Table continues on the next page...

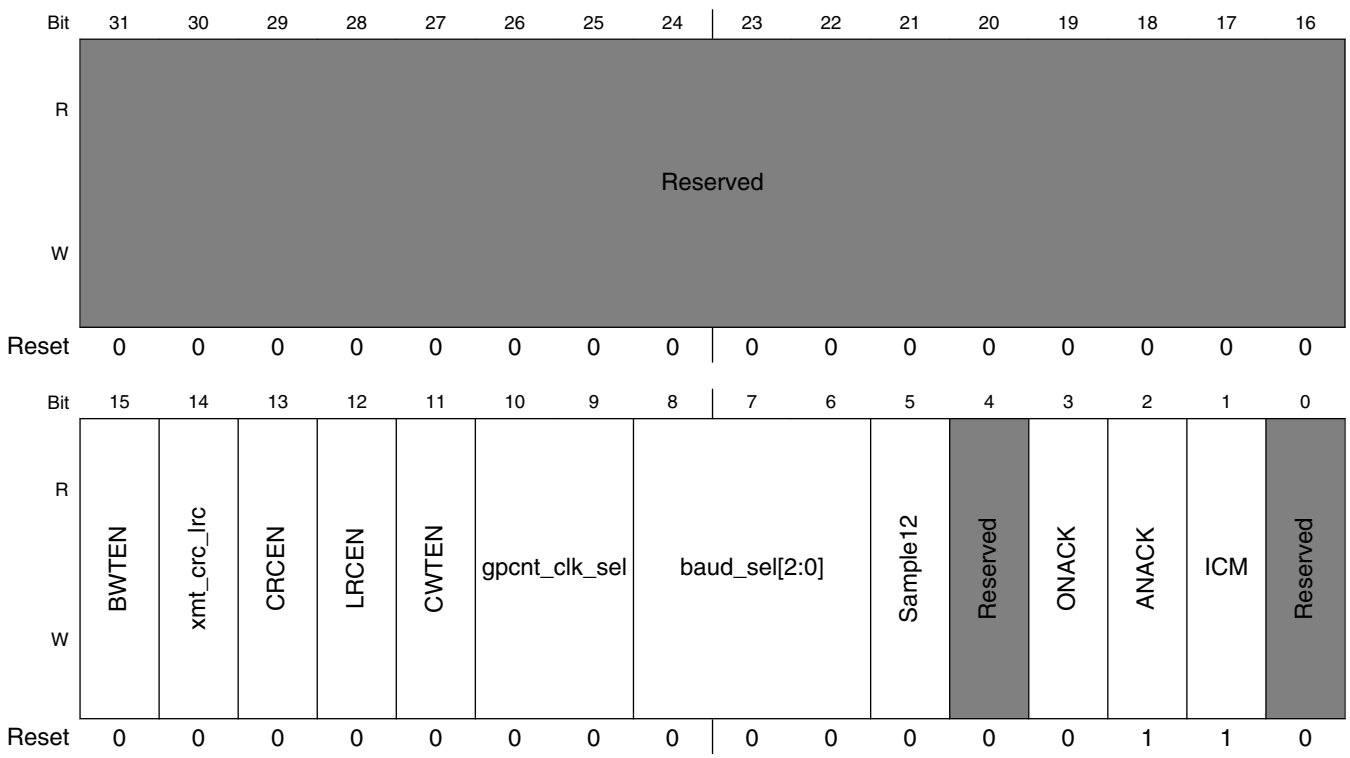
SIMx_PORT0_CNTL field descriptions (continued)

Field	Description
5 SCSP0	SIM Card Clock Stop Polarity port0. Used to control the polarity of the idle SIM clock when the clock is disabled by SCEN0. It will be forced low by hardware during the auto power down sequence. This forces the clock to be a logic 0 when stopped by auto power down as required by ISO 7816 spec. 0 Clock is logic 0 when stopped by scen0 1 Clock is logic 1 when stopped by scen0
4 SCEN0	SIM card Clock Enable Port 0. Used to enable/disable the clock to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 SIM Card Clock Disabled Port 0 1 SIM Card Clock Enabled Port 0
3 SRST0	SIM card Reset. Used to control state of reset line to the SIM card. It can be forced low by hardware during the auto power down sequence. SIM card reset signals are active low. 0 SIM Card reset Port0 inactive (default) 1 SIM Card reset Port0 active
2 STEN0	SIM card Transmit Enable Port 0. Used to enable/disable the XMT data to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 Port0 Transmit Data is forced to zero (default) 1 Port 0 Transmit Data controlled by SIM module
1 SVEN0	SIM card V_{CC} Enable Port 0. Used to control the state of the SVEN0 pin on SIM card port 0. The SVEN0 pin controls the SIM card V_{CC} enable in the power management chip. It can be forced low by hardware during the auto power down sequence. 0 SIM card Voltage Port 0 disabled (default) 1 SIM card Voltage Port 0 enabled
0 SAPD0	SIM card Auto Power Down Port 0. Used to enable/disable the auto power down function for port 0. It will be forced low at the end of the auto power down sequence. 0 Auto power down Port 0 disabled (default) 1 Auto power down Port 0 enabled

11.2.5.7 SIM Control Register (SIMx_CNTL)

See the figure below for illustration of valid bits in the SIM Control Register and the table below for description of the bit fields.

Address: Base address + 18h offset



SIMx_CNTL field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 BWTEN	**Block wait time enable.** Writing a "1" to this bit will enable the bwt and bgt functions. The bwt and bgt functions can then be individually selected using the interrupt mask. 0 Disable BWT, BGT 1 Enable BWT, BGT
14 xmt_crc_lrc	**Transmit CRC or LRC.** This bit specifies whether or not to transmit the redundancy checking data at the end of a transmission (that is, when the FIFO becomes empty). 0 No Redundancy check info transmitted (default) 1 Transmit LRC or CRC info when FIFO empties (whichever is enabled)
13 CRCEN	**CRC Enable.** This bit enables the calculation of the 16-bit CRC value for both receiver and transmitter. The result of the calculation is continuously compared to the expected remainder and reflected in the

Table continues on the next page...

SIMx_CNTL field descriptions (continued)

Field	Description
	crcok bit in the RCV_STAT register. Clearing this bit resets the current CRC residual value in the SIM hardware. 0 16-bit Cyclic Redundancy Checking disabled (default) 1 16-bit Cyclic Redundancy Checking enabled
12 LRCEN	**LRC Enable.** This bit enables the calculation of the 8-bit LRC value for both receiver and transmitter. The result of the calculation is continuously compared to zero and reflected in the lrcok bit in the RCV_STAT register. Clearing this bit resets the current LRC value in the SIM hardware. 0 8-bit Linear Redundancy Checking disabled (default) 1 8-bit Linear Redundancy Checking enabled
11 CWTEN	**Character Wait Time Counter Enable.** Enables the character wait time counter. Clearing this bit resets the counter to zero. 0 Character Wait time Counter off (default) 1 Character Wait time counter on
10–9 gpcnt_clk_sel	**General Purpose Counter Clock Select.** Selects which clock source is used by SIM Module general purpose counter. The only way to reset the counter is to set these bits to zero. The counter will begin counting as soon as the clock input is selected and the clocks are enabled. These input clocks are enabled through other register bits of the SIM module (KILL_CLOCK, RCV_EN, and XMT_EN respectively). 00 Disabled / Reset 01 Card Clock 10 Receive Clock 11 ETU Clock (transmit clock)
8–6 baud_sel[2:0]	SIM **Baud Rate Select.** Selects the asynchronous baud rate divisor of the clock When set to "111", the divisor is set to the value programmed in the DIVISOR register. This allows for more flexible baud rate determination. 000 31 (512/1 Fi/Di) 001 32 (512/2 Fi/Di) 010 16 (512/4 Fi/Di) 011 8 (512/8 Fi/Di) 100 4 (512/16 Fi/Di) 101 2 (512/32 Fi/Di) 110 1 (512/64 Fi/Di) 111 DIVISOR Reg
5 Sample12	**Sample12.** Set the third stage divider. This sets the corresponding sample rate which is the number of times a bit being received is sampled. 0 divide by 8(default) 1 divide by 12
4 -	This field is reserved. Reserved
3 ONACK	**Overrun NACK Enable.** Enables overrun NACK generation. 0 NACK generation on overrun is disabled (default) 1 NACK generation on overrun is enabled

Table continues on the next page...

SIMx_CNTL field descriptions (continued)

Field	Description
2 ANACK	**Automatic NACK Enable.** Enables nack generation for parity errors or invalid initial characters when in ICM mode. 0 NACK generation on errors disabled 1 NACK generation on errors enabled (default)
1 ICM	**Initial Character Mode.** Enables initial character mode. Will be automatically cleared by hardware once a valid initial character is received. 0 Initial Character Mode disabled 1 Initial Character Mode enabled (default)
0 -	This field is reserved. Reserved

11.2.5.8 SIM Clock Prescaler Register (SIMx_CLK_PRESCALER)

See the figure below for illustration of valid bits in the SIM Clock Prescaler Register and the table below for description of the bit fields.

Address: Base address + 1Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																								CLK_PRESCALER							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

SIMx_CLK_PRESCALER field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
CLK_ PRESCALER	**Clock prescaler divisor register.** The value written to this register will determine the first stage divider setting. If the ipg_perclk is 66Mhz, a typical setting would be 0x0e. This would set the SIM card clock to 66Mhz/14 = 4.7Mhz. The duty cycle of divided clock will be between 45% and 55% according to ISO7816 requirement. For the odd divider factor (2K+1), the duty cycle will be K/(2K+1) and (K+1)/(2k+1). So for 0x03, the duty cycle is 33%-66%. For 0x05, the duty cycle is 40%-60%. For 0x07, the duty cycle is 43%-57%, and for 0x09, the duty cycle is 44%-56%. For all other values, the clock duty cycle can meet the ISO7816 requirement. **NOTE:** Not all values are listed 0x00 ~0x02 ipg_perclk/ 2 0x03 ipg_perclk / 3 0x04 ipg_perclk / 4 0x05 ipg_perclk / 5 0x06 ipg_perclk / 6 0xFD ipg_perclk / 253

Table continues on the next page...

SIMx_CLK_PRESCALER field descriptions (continued)

Field	Description
	0xFE ipg_perclk / 254 0xFF ipg_perclk / 255

11.2.5.9 SIM Receive Threshold Register (SIMx_RCV_THRESHOLD)

See the figure below for illustration of valid bits in the SIM Receive Threshold Register and the table below for description of the bit fields.

Address: Base address + 20h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																			RTH[3:0]				RDT[8:0]								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

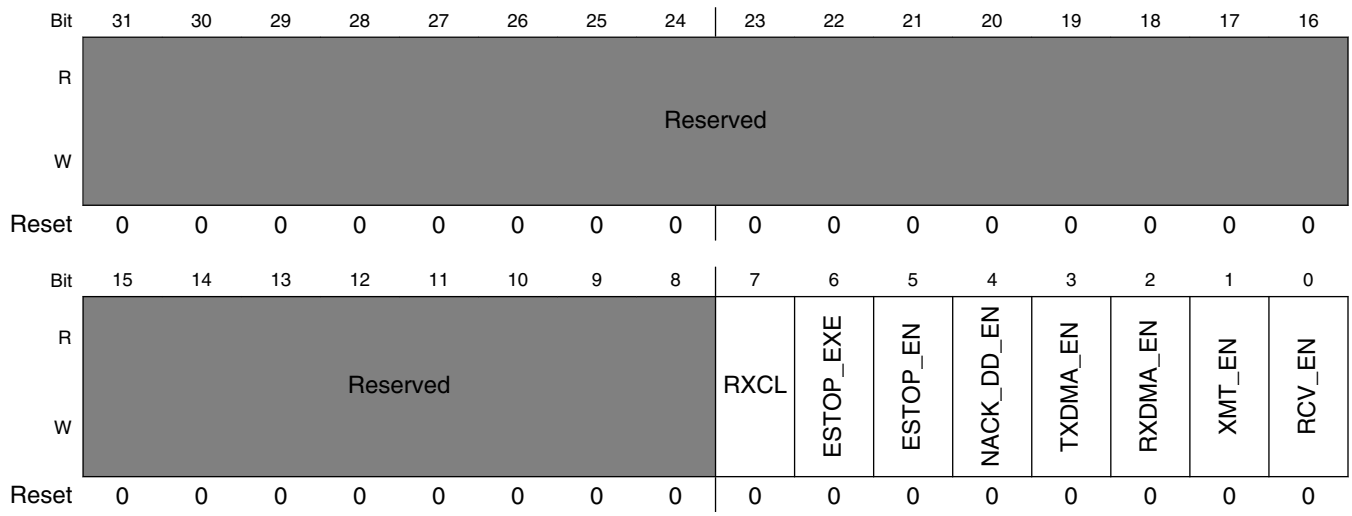
SIMx_RCV_THRESHOLD field descriptions

Field	Description
31–13 -	This field is reserved. Reserved
12–9 RTH[3:0]	**Receive Nack Threshold.** Used to specify the number of consecutive NACK's transmitted by the SIM module, for a given character, before the receive threshold error (RTE) flag is triggered. A value of 0 indicates that RTE is never set. When a valid character is received by the SIM, the internal counter keeping track of the NACK count resets to zero for the subsequent byte being received. If the ANACK bit is clear in the CNTL register, RTH has no effect.
RDT[8:0]	**Receive Data Threshold.** Determines the number of unread bytes that must exist in the FIFO to trigger the receive data register full (RDRF) interrupt flag. If the number of unread bytes in the receive FIFO is greater than or equal to the value in RDT, the RDRF flag in the RCV_STATUS register will be set. A value of zero indicates that there must be 288 unread bytes in the FIFO to trigger RDRF. If a value of more than 288 is entered in this register, the register value will remain 288.

11.2.5.10 SIM Enable Register (SIMx_ENABLE)

See the figure below for illustration of valid bits in the SIM Enable Register and the table below for description of the bit fields.

Address: Base address + 24h offset



SIMx_ENABLE field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
7 RXCL	Reception data latch disable. Used to disable/enable the cross clock domain synchronization operation of reception data in SIM receive state machine. 0 Enable the synchronization operation. 1 Disable the synchronization operation.
6 ESTOP_EXE	Enforce reception early stop execution. Used to execute the function of enforce reception early stop when ESTOP_EN bit set to 1. If this bit is 0 the early stop function can't be executed when NACK signal occur. 0 Enforce reception early stop function will be executed without NACK. 1 Enforce reception early stop function will be executed at all times.
5 ESTOP_EN	Enforce reception early stop enable. Used to early stop the SIM receive state machine. The state machine will stop earliey than 10.8 ETUs when 11 ETUs mode or earlier than 11.8 ETUs when non 11 ETUs mode, if enable this bit. This function is used for the test case tramsmission stop at 10.8 ETUs from the falling edge of start signal. **NOTE:** This bit should be set to 1 when receive data, and set to 0 when transmit data. 0 Disable the enforce reception early stop function. 1 Enable the enforce reception early stop function.
4 NACK_DD_EN	NACK delay detection enable. Used to enable/disable the function of NACK delay detection. The SIM receive state machine will detect the NACK signal at the 16th phase and use the values of12th, 14th and

Table continues on the next page...

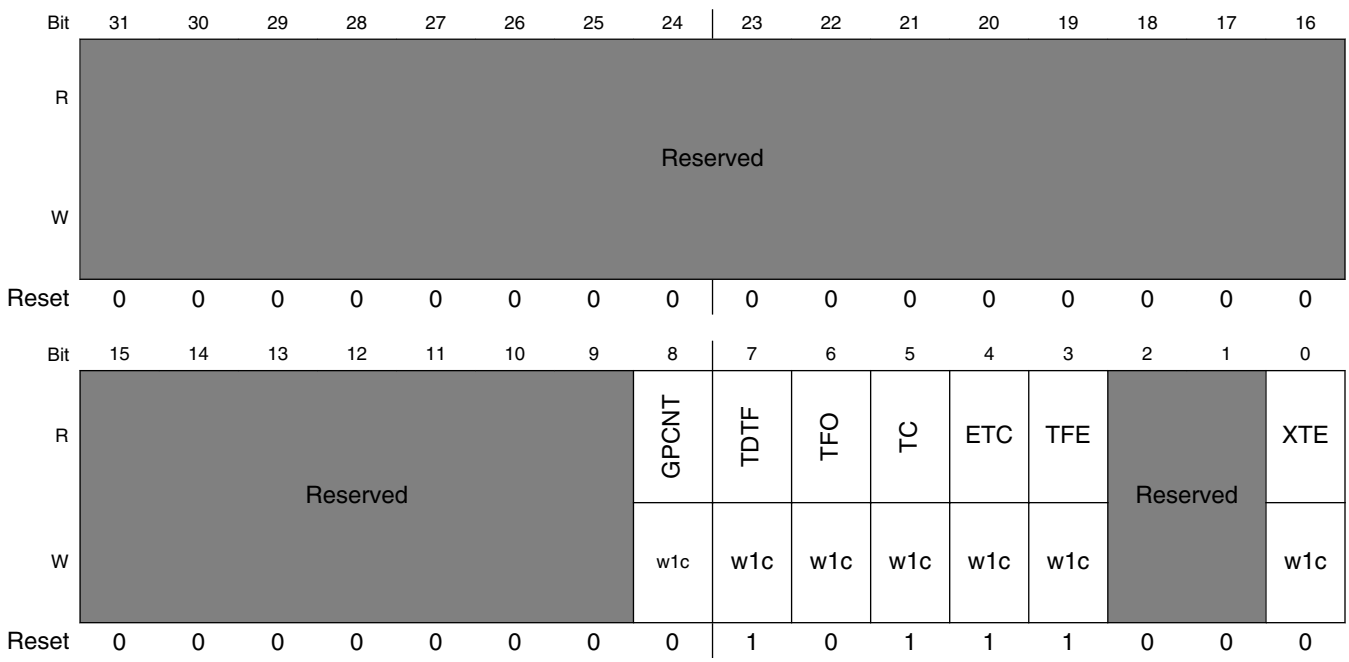
SIMx_ENABLE field descriptions (continued)

Field	Description
	15th sample point, if enable this bit. This function is used for the test case NACK signal send at 10.7 ETUs from the falling edge of start signal. 0 Disable the delay detection function. 1 Enable the delay detection function.
3 TXDMA_EN	Transmitter DMA Enable. This bit allows SIM to request for DMA transfers. When enabled, DMA requests are generated when the number of bytes in the transmit FIFO is less than or equal to the value programmed in the tdt[3:0] bits in the XMT_THRESHOLD register and XMT_EN is set. 0 SIM Transmitter DMA requests disabled. 1 SIM Transmitter DMA requests enabled.
2 RXDMA_EN	Receiver DMA Enable. This bit allows SIM to request for DMA transfers. When enabled, DMA requests are generated when when the number of bytes in the receive FIFO is more than or equal to the value programed in the RDT[8:0] in the RCV_THRESHOLD register and RCV_EN is set 0 SIM Receiver DMA request disabled. 1 SIM Receiver DMA request enabled.
1 XMT_EN	SIM **Transmit Enable.** Used to enable/disable the SIM transmit state machine. When the SIM is being used to receive data, XMT_EN should be deasserted. This bit also enables the xmt_clk and rcv_clk inputs to the General Purpose Counter. **NOTE:** Setting this bit (transition from 0 to 1) will reset the CRC and LRC values. 0 SIM Transmitter disabled (default) 1 SIM Transmitter enabled
0 RCV_EN	SIM **Receiver Enable.** Used to enable/disable the SIM receive state machine. The RCV_EN bit should be left high whenever the SIM module is in use. The SIM module has an automatic receive mode operation that disables the reception of characters when the transmitter is operational. Once the transmitter has completed sending the last character, the receiver is automatically enabled. This bit also enables the RCV_CLK input to the General Purpose Counter. 0 SIM Receiver disabled (default) 1 SIM Receiver enabled

11.2.5.11 SIM Transmit Status Register (SIMx_XMT_STATUS)

See the figure below for illustration of valid bits in the SIM Transmit Status Register and the table below for description of the bit fields.

Address: Base address + 28h offset



SIMx_XMT_STATUS field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 GPCNT	**General purpose Counter Flag.** Used to indicate when the General purpose counter has reached the value in the GPCNT register. 0 GPCNT time not reached, or bit has been cleared. (default). 1 General Purpose counter has reached the GPCNT value.
7 TDTF	**Transmit Data Threshold Flag.** Used to indicate when the number of bytes in the transmit FIFO is less than or equal to the value programmed in the tdt[3:0] bits in the XMT_THRESHOLD register. 0 Number of bytes in FIFO is greater than tdt[3:0], or bit has been cleared. 1 Number of bytes in FIFO is less than or equal to tdt[3:0] (default)
6 TFO	**Transmit FIFO Overfill Error.** Used to indicate when the Transmit FIFO has been written with more than 16 bytes. The tfo bit can only be cleared by setting the flush_xmt or soft_reset bit in the RESET_CNTL register. 0 No transmit FIFO overfill error has occurred (default). 1 A Transmit FIFO overfill error has occurred.

Table continues on the next page...

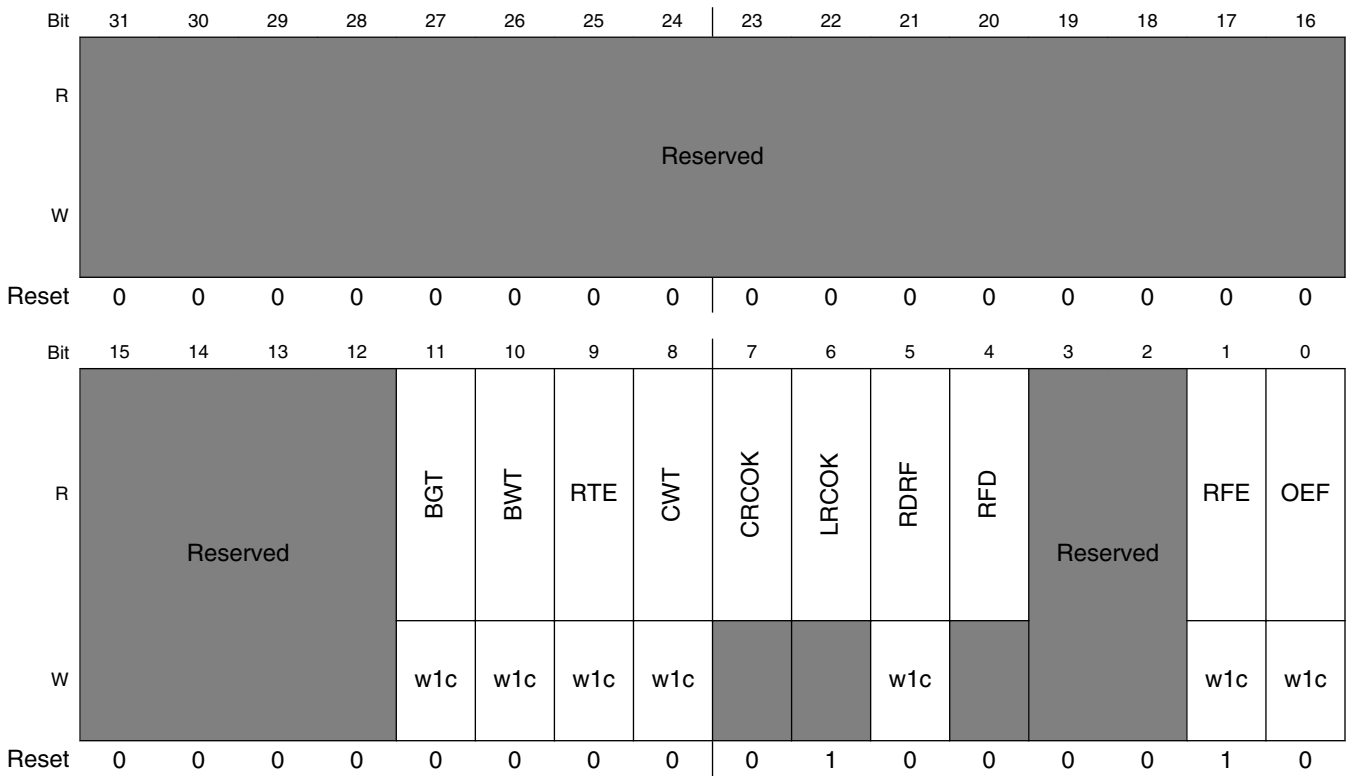
SIMx_XMT_STATUS field descriptions (continued)

Field	Description
5 TC	**Transmit Complete.** Used to indicate whether the SIM transmitter is ready for a new transmission. The tc flag becomes set after the guard time has expired for the last byte in the transmit FIFO. The tc flag will create an interrupt if tcim in the INT_MASK register is low. The tc bit is a write-one-to-clear bit. 0 Transmit pending or in progress 1 Transmit complete (default)
4 ETC	**Early Transmit Complete.** Used to indicate that the SIM transmitter has finished sending the current byte and the transmit FIFO is empty. This bit differs from the tc bit in that it is set before the guard time of the last byte has elapsed. The ETC flag will create an interrupt if ETCIM in the INT_MASK register is low. The ETC bit is a write-one-to-clear bit. 0 Transmit pending or in progress 1 Transmit complete (default)
3 TFE	**Transmit FIFO Empty.** Used to indicate that the SIM transmit FIFO has emptied. This bit will be set when the last byte in the transmit FIFO has been transferred to the SIM transmitter shift register. The TFE flag will create an interrupt if TFEIM in the INT_MASK register is low. The TFE bit is a write-one-to-clear bit. 0 Transmit FIFO is not empty 1 Transmit FIFO is empty (default)
2–1 -	This field is reserved. Reserved
0 XTE	**Transmit NACK Threshold Error.** Used to indicate the transmit NACK threshold has been reached. When XTE is high, no further transmissions will be done until the XTE flag is cleared. Any data transmissions still pending in the transmit FIFO will be aborted, and the TC, ETC, and TFE flags will be set. The XTE flag will create an interrupt if XTM in the IMT_MASK register is low. The XTE bit is a write-one-to-clear bit. 0 Transmit NACK threshold has not been reached (default) 1 Transmit NACK threshold reached; transmitter frozen

11.2.5.12 SIM Receive Status Register (SIMx_RCV_STATUS)

See the figure below for illustration of valid bits in the SIM Receive Status Register and the table below for description of the bit fields.

Address: Base address + 2Ch offset



SIMx_RCV_STATUS field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11 BGT	**Block guard time error flag.** Used to indicate if the block guard time was too small. The threshold is set by the block guard time register. 0 Block guard time was sufficient. 1 Block guard time was too small.
10 BWT	**Block wait time error flag.** Used to indicate if the block wait time has been exceeded. The threshold is set by the block wait time registers. 0 Block wait time not exceeded. 1 Block wait time was exceeded.
9 RTE	**Receive NACK threshold error flag.** Used to indicate whether the number of consecutive NACKs generated by the SIM module in response to receive parity errors, for the byte being received, equals the

Table continues on the next page...

SIMx_RCV_STATUS field descriptions (continued)

Field	Description
	value programmed in the RTH[3:0] in the RCV_THRESHOLD register. This bit would never be set unless the ANACK bit is set in the CNTL register. The SAPDx bit in the PORTx_CNTL resister must be set to enable the threshold error to trigger the auto power down sequence. RTE is a write-one-to-clear bit. Clearing this bit also resets the internal counter for consecutive NACKs being transmitted for a given byte. 0 Number of NACKs generated by the receiver is less than the value programmed in RTH[3:0] 1 Number of NACKs generated by the receiver is equal to the value programmed in RTH[3:0]
8 CWT	**Character Wait Time Counter Flag.** Used to indicate when the time between received characters is equal to or greater than the value programmed in the CHAR_WAIT register. 0 No CWT violation has occurred (default). 1 Time between two consecutive characters exceeded the value in CHAR_WAIT.
7 CRCOK	**Cyclic Redundancy Check Okay flag.** Used to indicate when the calculated 16-bit CRC value matches the expected value for the current input data stream. The value is calculated across all received characters from the point the CRCEN bit is set in the CNTL register. The current CRC residual can be reset by three mechanisms: Clear CRCEN bit in CNTL register Set XMT_EN bit in ENABLE register Automatically by hardware when ETC flag is set at the end of a transmission. 0 Current CRC value does not match remainder. 1 Current calculated CRC value matches the expected result.
6 LRCOK	Linear Redundancy Check Okay flag. Used to indicate when the calculated 8-bit LRC value is zero value for the current input data stream. The value is calculated across all received characters from the point the LRCEN bit is set in the CNTL register. The current LRC residual can be reset by three mechanisms: Clear LRCEN bit in CNTL register Set XMT_EN bit in ENABLE register Automatically by hardware when ETC flag is set at the end of a transmission. 0 Current LRC value does not match remainder. 1 Current calculated LRC value matches the expected result (that is, zero).
5 RDRF	Receive Data Register Full. Used to indicate when the number of bytes in the receive FIFO is more than or equal to the value programed in the RDT[8:0] in the RCV_THRESHOLD register. 0 Number of unread bytes in receive buffer < value set by RDT[8:0] (default), or bit has been cleared. 1 Number of unread bytes in receive buffer >= value set by RDT[8:0].
4 RFD	**Receive FIFO has unread Data.** Used to indicate that there is at least one unread byte in the receive data FIFO. Can only be cleared by reading all bytes out of the receive FIFO. The RFD bit cannot be used to create an interrupt. Normally, the SIM triggers the interrupt with RDRF and software uses rfd to read all of the bytes out of the receive FIFO. 0 There are no unread bytes in the receive FIFO (default). 1 There is at least one unread byte in the receiver FIFO.
3–2 -	This field is reserved. Reserved
1 RFE	**Receive FIFO Empty.** Used to indicate that the SIM receive FIFO has emptied. This bit will be set when the last byte in the receive FIFO has been transferred to the IPS bus. The RFE flag will create an interrupt if RFEIM in the INT_MASK register is low. The RFE bit is a write-one-to-clear bit.

Table continues on the next page...

SIMx_RCV_STATUS field descriptions (continued)

Field	Description
	0 Receive FIFO is not empty 1 Receive FIFO is empty (default)
0 OEF	**Overrun Error Flag.** Used to indicate that the SIM was unable to store received data due to already having 288 unread bytes in the FIFO. It does not necessarily indicate that data has been lost. If the ONACK control bit in the CNTL register is set, there will be a NACK pulse generated on bytes that would otherwise cause a loss of data due to a full FIFO. These bytes should be retransmitted by the SIM card which implies that no data has actually been lost. In this case, the OEF flag is just an indicator that this situation has occurred which may be helpful in system debug. For the case where ONACK is not set, a set OEF flag does indicate a loss of data since all bytes received with the OEF flag set will indeed be lost (including the byte that caused the bit to be set). The OEF flag will cause an interrupt if the oim bit in the INT_MASK register. The OEF flag is a write-one-to-clear bit. 0 No overrun error has occurred (default). 1 A byte was received when the received FIFO was already full.

11.2.5.13 SIM Interrupt Mask Register (SIMx_INT_MASK)

See the figure below for illustration of valid bits in the SIM Interrupt Mask Register and the table below for description of the bit fields.

Address: Base address + 30h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		RFEM	BGTM	BWTM	RTM	CWTM	GPCNTM	TDTFM	TFOM	XTM	TFEIM	ETCIM	OIM	TCIM	RIM
Reset	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

SIMx_INT_MASK field descriptions

Field	Description
31–14 -	This field is reserved. Reserved
13 RFEM	**Receive fifo empty interrupt mask.** Used to enable/disable the ability of the RFE flag in the RCV_STATUS register to generate SIM interrupts.

Table continues on the next page...

SIMx_INT_MASK field descriptions (continued)

Field	Description
	0 RFE interrupt enabled 1 RFE interrupt masked (default)
12 BGTM	**Block guard time interrupt mask.** Used to enable/disable the ability of the bgt flag in the RCV_STATUS register to generate SIM interrupts. 0 BGT interrupt enabled 1 BGT interrupt masked (default)
11 BWTM	**Block wait time interrupt mask.** Used to enable/disable the ability of the bwt flag in the RCV_STATUS register to generate SIM interrupts. 0 BWT interrupt enabled 1 BWT interrupt masked (default)
10 RTM	**Receive Nack threshold interrupt mask.** Used to enable/disable the ability of the RTE flag in the RCV_STATUS register to generate SIM interrupts. 0 RTE interrupt enabled 1 RTE interrupt masked (default)
9 CWTM	**Character Wait Time Interrupt Mask.** Used to enable/disable the ability of the CWT flag in the RCV_STATUS register to generate SIM interrupts. 0 CWT interrupt enabled 1 CWT interrupt masked (default)
8 GPCNTM	**General Purpose Counter Interrupt Mask.** Used to enable/disable the ability of the GPCNT flag in the XMT_STATUS register to generate SIM interrupts. 0 GPCNT interrupt enabled 1 GPCNT interrupt masked (default)
7 TDTFM	**Transmit Data Threshold Interrupt Mask.** Used to enable/disable the ability of the TDTF flag in the XMT_STATUS register to generate SIM interrupts. 0 TDTF interrupt enabled 1 TDTF interrupt masked (default)
6 TFOM	**Transmit FIFO Overfill Error Interrupt Mask.** Used to enable/disable the ability of the TFO flag in the XMT_STATUS register to generate SIM interrupts. 0 TFO interrupt enabled 1 TFO interrupt masked (default)
5 XTM	**Transmit Threshold Interrupt Mask.** Used to enable/disable the ability of the XTE flag in the XMT_STATUS register to generate SIM interrupts 0 XTE interrupt enabled 1 XTE interrupt masked (default)
4 TFEIM	**Transmit FIFO Empty Interrupt Mask.** Used to enable/disable the ability of the TFE flag in the XMT_STATUS register to generate SIM interrupts. 0 TFE interrupt enabled 1 TFE interrupt masked (default)
3 ETCIM	**Early Transmit Complete Interrupt Mask.** Used to enable/disable the ability of the ETC flag in the XMT_STATUS register to generate SIM interrupts.

Table continues on the next page...

SIMx_INT_MASK field descriptions (continued)

Field	Description
	0 ETC interrupt enabled 1 ETC interrupt masked (default)
2 OIM	**Overrun Interrupt Mask.** Used to enable/disable the ability of the OEF flag in the RCV_STATUS register to generate SIM interrupts. 0 OEF interrupt enabled 1 OEF interrupt masked (default)
1 TCIM	**Transmit Complete Interrupt Mask.** Used to enable/disable the ability of the TC flag in the XMT_STATUS register to generate SIM interrupts. 0 TC interrupt enabled 1 TC interrupt masked (default)
0 RIM	**Receive Interrupt Mask.** Used to enable/disable the ability of the RDRF flag in the RCV_STATUS register to generate SIM interrupts. 0 RDRF interrupt enabled 1 RDRF interrupt masked (default)

11.2.5.14 SIM Port0 Detect Register (SIMx_PORT0_DETECT)

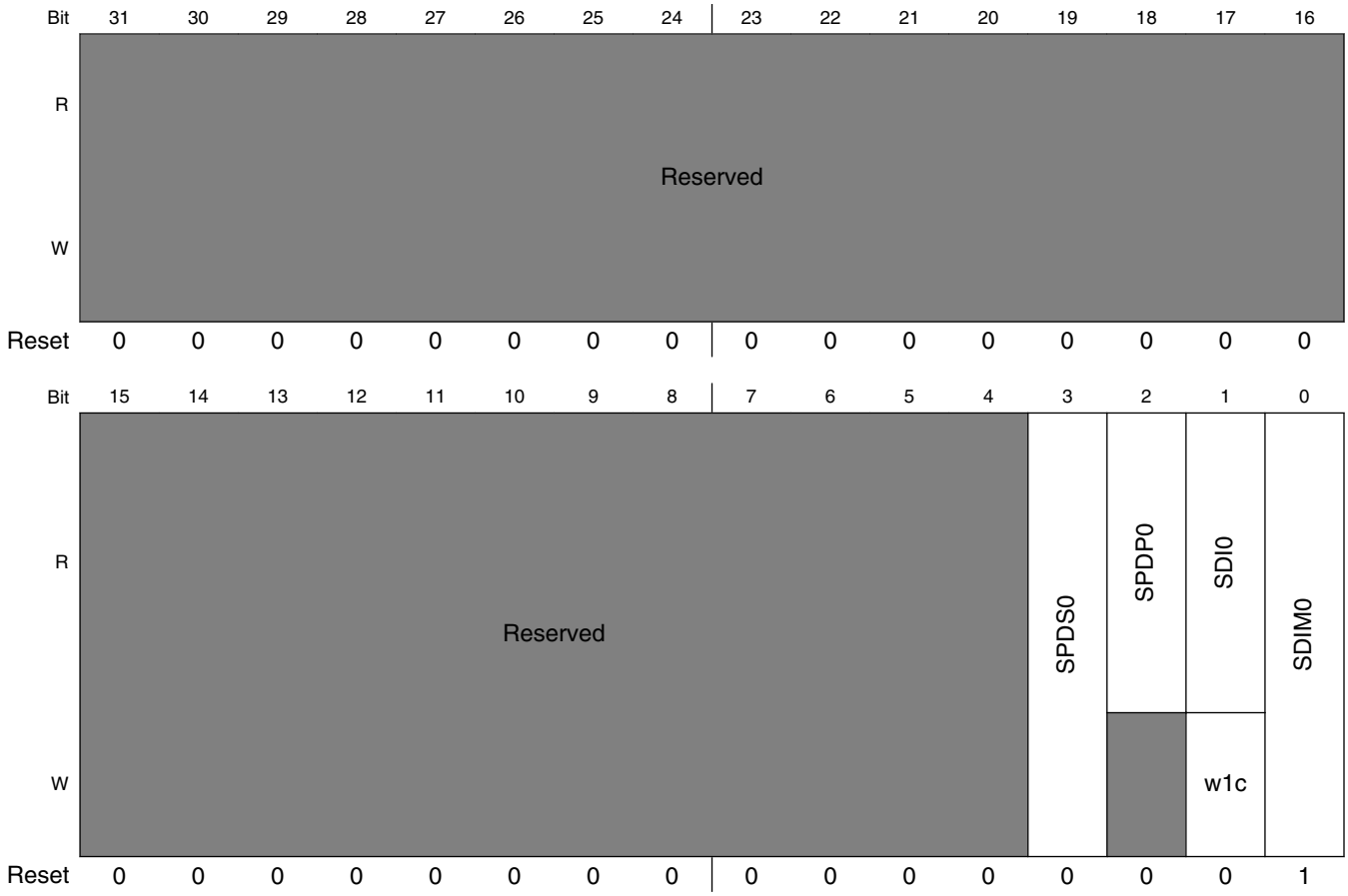
See the figure below for illustration of valid bits in the SIM Port0 Detect Register and the table below for description of the bit fields.

SPDS0 determines which edge transition of the SIMPD0 pin is used for SIM card presence detection. Presence detection can be used to determine if the card has been inserted or removed. The occurrence of the SIMPD0 edge, specified by SPDS0, will cause the following:

- SDI0 to be set
- If the SDIM0 mask is clear, an interrupt to occur on SIMIRQ_N
- If SAPD0 in the PORT0_CNTL register is set, auto power down sequence will begin

If a SIM card insertion is expected, SAPD0 can be set low to avoid the auto power down sequence. There is no auto power up sequence. SPDP0 can be used to determine the current state of the SIMPD0 pin.

Address: Base address + 3Ch offset



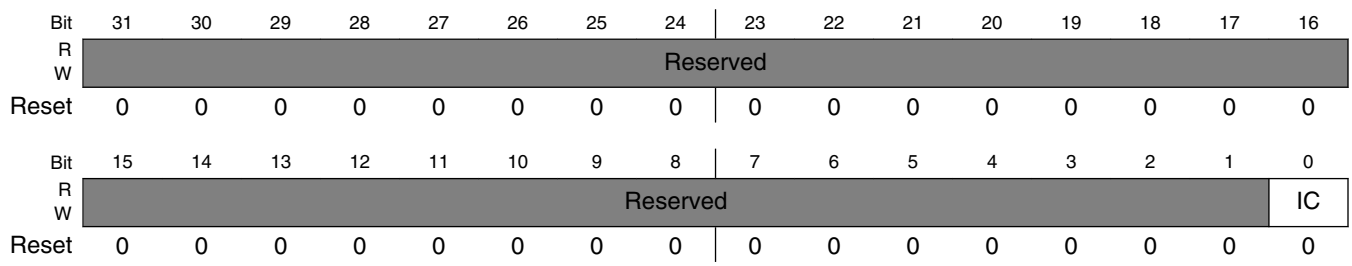
SIMx_PORT0_DETECT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved
3 SPDS0	**SIM Presence Detect Select Port 0.** Controls which edge of the SIMPD0 pin is used to detect the presence of the SIM card. 0 Falling edge of SIMPD0 Input (default) 1 Rising edge of SIMPD0 Input
2 SPDP0	**SIMPD0 input pin status.** This bit reflects the state of the SIMPD0 pin. It is not a latched register bit, but instead a synchronized version of the state of the SIMPD0 pin itself. 0 SIMPD0 pin is logic low 1 SIMPD0 pin is logic high
1 SDI0	**SIM Detect Interrupt flag Port 0.** Status flag to indicate the insertion or removal of a SIM card has been detected on port 0. Can create an interrupt to the MCU if SDIM0 is low. Write a "1" to this bit to clear. 0 No insertion or removal of SIM card detected on Port 0 (default) 1 Insertion or removal of SIM card detected on Port 0
0 SDIM0	**SIM Detect Interrupt Mask Port 0.** Interrupt mask for the SDI0 interrupt flag. 0 sdi0 enabled 1 sdi0 masked (default)

11.2.5.15 SIM Data Format Register (SIMx_DATA_FORMAT)

See the figure below for illustration of valid bits in the SIM Data Format Register and the table below for description of the bit fields.

Address: Base address + 40h offset



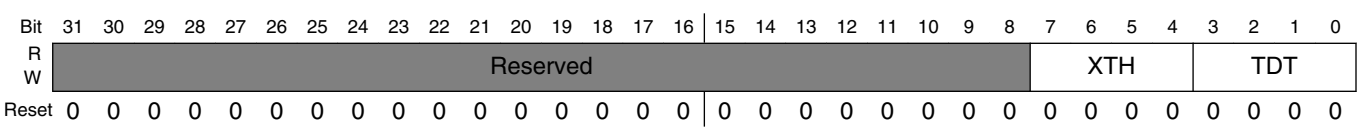
SIMx_DATA_FORMAT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 IC	**Inverse Convention.** Used to configure the SIM to use either inverse convention or direct convention for its data format. The IC bit can be controlled by software, but it is normally set by hardware as a result of the interpretation of the initial character when in ICM mode. 0 Direction convention transfers enabled (default). 1 Inverse convention transfers enabled.

11.2.5.16 SIM Transmit Threshold Register (SIMx_XMT_THRESHOLD)

See the figure below for illustration of valid bits in the SIM Transmit Threshold Register and the table below for description of the bit fields.

Address: Base address + 44h offset



SIMx_XMT_THRESHOLD field descriptions

Field	Description
31–8 -	This field is reserved. Reserved

Table continues on the next page...

SIMx_XMT_THRESHOLD field descriptions (continued)

Field	Description
7–4 XTH	XTH[3:0] **Transmit NACK Threshold.** Used to set the NACK threshold for the transmitter. Once the threshold number set by XTH has been reached for the current byte being transmitted, the error flag XTE in the XMT_STATUS register will be set. Setting of XTE causes the remaining transmissions queued in the transmit FIFO to be aborted and no more transmissions to occur until software clears XTE. To trigger XTE, a given byte being transmitted must reach the XTH threshold itself. Transmit NACKs accumulated on one byte are not carried over to the next. 0x0 XTE will never be set; retransmission after NACK reception is disabled. 0x1 XTE will be set after 1 nack is received; 0 retransmissions occurs. 0x2 XTE will be set after 2 nacks are received; at most 1 retransmission occurs. 0x3 XTE will be set after 3 nacks are received; at most 2 retransmissions occurs. 0xX XTE will be set after X nacks are received; at most (X - 1) retransmissions occurs. 0xF XTE will be set after 15 nacks are received; at most 14 retransmissions occurs.
TDT	**Transmit Data Threshold.** Used to set the threshold value for the Transmit FIFO at which the TDTF bit in the XMT_STATUS register will be set. When the number of bytes in the Transmit FIFO is less than or equal to TDT[3:0], TDTF will be set.

11.2.5.17 SIM Transmit Guard Control Register (SIMx_GUARD_CNTL)

See the figure below for illustration of valid bits in the SIM Transmit Guard Control Register and the table below for description of the bit fields.

Address: Base address + 48h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved							RCVR11	GETU							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIMx_GUARD_CNTL field descriptions

Field	Description
31–9 -	This field is reserved. Reserved

Table continues on the next page...

SIMx_GUARD_CNTL field descriptions (continued)

Field	Description
8 RCVR11	**Receiver use 11 ETUs.** Used to configure the SIM module receiver for 11 ETU operation (that is, 1 Stop bit). This bit is provided for support of T=1 cards. 0 Receiver configured for 12 ETU operation (default) 1 Receiver configured for 11 ETU operation
GETU	**Transmit Guard ETUs.** Used to control the number of additional Elementary Time Units (ETUs) inserted between bytes transmitted by the SIM transmitter. An ETU is equivalent to one bit time at the given baud rate (for example, the length of a START bit). The guard time has no effect on the SIM receiver. A value of 0x00 inserts no additional ETUs, while a value of 0xFE inserts 254 additional ETUs. A value of 0xFF subtracts one ETU by reducing the number of STOP bits from two to one.

11.2.5.18 SIM Open Drain Configuration Control Register (SIMx_OD_CONFIG)

See the figure below for illustration of valid bits in the SIM Open Drain Configuration Control Register and the table below for description of the bit fields.

Address: Base address + 4Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														OD_P1	OD_P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIMx_OD_CONFIG field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
1 OD_P1	**Open Drain control for Port 1.** Used to control whether the XMT data line on port 1 is open-drain. If AMODE bit in SETUP register is set, this bit will have no effect. 0 XMT pin on port 1 is push-pull (default). 1 XMT pin on port 1 is open-drain.

Table continues on the next page...

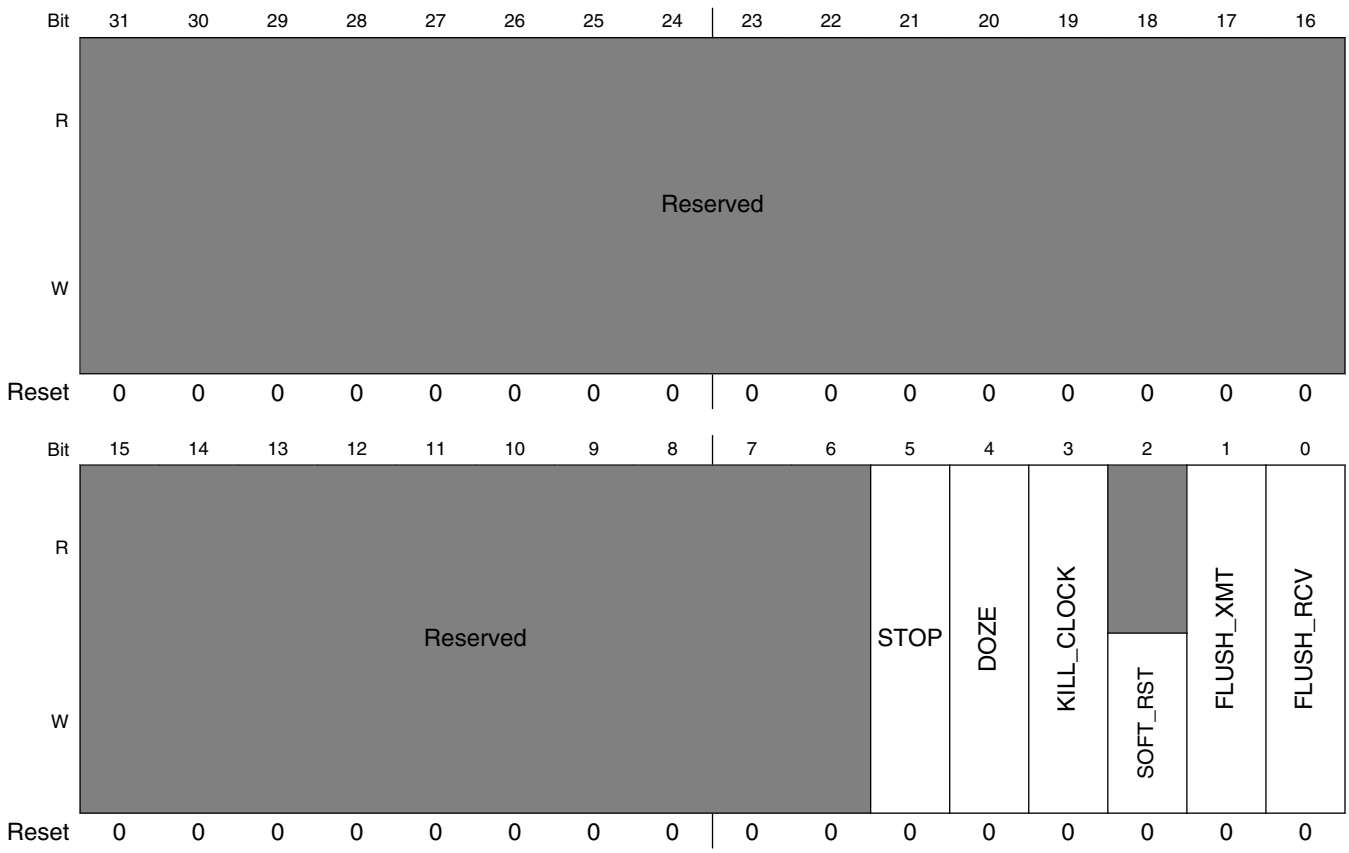
SIMx_OD_CONFIG field descriptions (continued)

Field	Description
0 OD_P0	**Open Drain control for Port 0.** Used to control whether the XMT data line on port 0 is open-drain. 0 XMT pin on port 0 is push-pull (default). 1 XMT pin on port 0 is open-drain.

11.2.5.19 SIM Reset Control Register (SIMx_RESET_CNTL)

See the figure below for illustration of valid bits in the SIM Reset Control Register and the table below for description of the bit fields.

Address: Base address + 50h offset



SIMx_RESET_CNTL field descriptions

Field	Description
31–6 -	This field is reserved. Reserved

Table continues on the next page...

SIMx_RESET_CNTL field descriptions (continued)

Field	Description
5 STOP	**STOP.** Used to configure the operation of the SIM module when a processor STOP instruction is executed. This bit is added to provide support for SIM cards that do not allow the SIM Card clock to be stopped while power is applied. 0 STOP instruction shuts down all SIM clocks (default). 1 STOP instruction shuts down all clocks except for the BAUD_CLK (clock provided to SIM Card).
4 DOZE	DOZE. Used to configure the operation of the SIM module when a processor DOZE instruction is executed. 0 DOZE instruction has no effect on SIM module (default). 1 DOZE instruction will cause SIM module to gate SIM clocks when the transmit FIFO is empty.
3 KILL_CLOCK	**Kill** SIM **Clock.** Used to enable/disable the SIM clock input to the SIM module. This bit will gate all SIM clocks including the SIM card clock regardless of the state of the STOP bit described above. 0 SIM input clock enabled (default). 1 SIM input clock disabled.
2 SOFT_RST	**Software Reset.** Used to reset the entire SIM module. This acts the same as a hardware reset for the SIM module. This bit is self-clearing. **NOTE:** Software should allow a minimum of 4 reference clock cycles (CKIH) before attempting to access the SIM module after a software reset. 0 SIM Normal operation (default). 1 SIM held in Reset.
1 FLUSH_XMT	**Flush Transmitter.** This bit operates as a SIM transmitter reset. The receive portion of the SIM module is not affected. The software must clear this bit before the SIM transmitter can operate. 0 SIM Transmitter normal operation (default). 1 SIM Transmitter held in Reset.
0 FLUSH_RCV	**Flush Receiver.** This bit operates as a SIM receiver reset. The transmit portion of the SIM module is not affected. The software must clear this bit before the SIM receiver can operate. 0 SIM Receiver normal operation (default). 1 SIM Receiver held in Reset.

11.2.5.20 SIM Character Wait Time Register (SIMx_CHAR_WAIT)

See the figure below for illustration of valid bits in the SIM Character Wait Time Register and the table below for description of the bit fields.

Address: Base address + 54h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																CWT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

SIMx_CHAR_WAIT field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
CWT	**Character Wait Time.** The value written to this register will specify the number of ETU times allowed between characters. Default is 0xFFFF

11.2.5.21 SIM General Purpose Counter Register (SIMx_GPCNT)

See the figure below for illustration of valid bits in the SIM General Purpose Counter Register and the table below for description of the bit fields.

Address: Base address + 58h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																GPCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

SIMx_GPCNT field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
GPCNT	**General Purpose Counter.** The value written to this register will be used to compare to the general purpose counter in the SIM module. Once the General purpose counter reaches this value, the GPCNT flag in the XMT_STATUS register will be set. This counter is intended to be used for any events that must be monitored for duration based on the card clock, receiver sample rate, or ETU rate (transmit clock). Example: ATR arrival time and ATR duration. Default is 0xFFFF

11.2.5.22 SIM Divisor Register (SIMx_DIVISOR)

See the figure below for illustration of valid bits in the SIM Divisor Register and the table below for description of the bit fields.

Address: Base address + 5Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																								DIVISOR							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

SIMx_DIVISOR field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
DIVISOR	**DIVISOR Register.** The value written to this register will be used to generate the SIM Receive clock. The BAUD_SEL[2:0] bits in the CNTL register must be set to '111' in order to control the divisor value using the DIVISOR register. Default is 0xFF

11.2.5.23 SIM Block Wait Time Register (SIMx_BWT)

See the figure below for illustration of valid bits in the SIM Block Wait Time Register and the table below for description of the bit fields.

Address: Base address + 60h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																BWT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

SIMx_BWT field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
BWT	**BWT Register 16 LSB.** The value in this register is the block wait time 16 LSB. The time from START bit of last byte sent from the SIM module to the START bit of the first byte sent from the SmartCard must be less than the value formed by the 32-bit register composed by BWT_H and BWT registers. If it is not, then the BWT flag will be set. Default is 0xFFFF

11.2.5.24 SIM Block Guard Time Register (SIMx_BGT)

See the figure below for illustration of valid bits in the SIM Block Guard Time Register and the table below for description of the bit fields.

Address: Base address + 64h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																BGT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIMx_BGT field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
BGT	**BGT Register.** The value in this register is the block guard time. Time from START bit of last byte sent from the SIM module to the START bit of the first byte sent from the SmartCard must be greater than this value. If it is not, then the BGT flag will be set. Default is 0x0000

11.2.5.25 SIM Block Wait Time Register HIGH (SIMx_BWT_H)

See the figure below for illustration of valid bits in the SIM Block Wait Time Register HIGH and the table below for description of the bit fields.

Address: Base address + 68h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																BWT_H																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

SIMx_BWT_H field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
BWT_H	**BWT Register 16 MSB.** The value in this register is the block wait time 16 MSB. The time from START bit of last byte sent from the SIM module to the START bit of the first byte sent from the SmartCard must be less than the value formed by the 32-bit register composed by BWT_H and BWT registers. If it is not, then the BWT flag is set. Default is 0xFFFF

11.2.5.26 SIM Transmit FIFO Status Register (SIMx_XMT_FIFO_STAT)

See the figure below for illustration of valid bits in the SIM Transmit FIFO Status Register and the table below for description of the bit fields.

Address: Base address + 6Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																				XMT_CNT				XMT_WPTR				XMT_RPTR			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

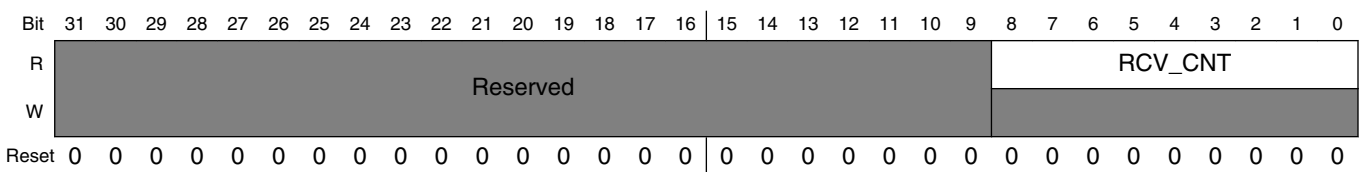
SIMx_XMT_FIFO_STAT field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11–8 XMT_CNT	These bits indicate the number of Bytes in the transmit FIFO. '0' value means FIFO is empty or full.
7–4 XMT_WPTR	These bits indicate the transmit FIFO Write Pointer.
XMT_RPTR	These bits indicate the transmit FIFO Read Pointer.

11.2.5.27 SIM Receive FIFO Counter Register (SIMx_RCV_FIFO_CNT)

See the figure below for illustration of valid bits in the SIM Receiver FIFO Counter Register and the table below for description of the bit fields.

Address: Base address + 70h offset



SIMx_RCV_FIFO_CNT field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
RCV_CNT	These bits indicate the number of Byte can be written into the receive FIFO. 0 value means FIFO is empty or full.

11.2.5.28 SIM Receive FIFO Write Pointer Register (SIMx_RCV_FIFO_WPTR)

See the figure below for illustration of valid bits in the SIM Receive FIFO Write Pointer Register and the table below for description of the bit fields.

Address: Base address + 74h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																							RCV_WPTR								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIMx_RCV_FIFO_WPTR field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
RCV_WPTR	These bits indicate the receive FIFO Write pointer.

11.2.5.29 SIM Receive FIFO Read Pointer Register (SIMx_RCV_FIFO_RPTR)

See the figure below for illustration of valid bits in the SIM Receive FIFO Read Pointer Register and the table below for description of the bit fields.

Address: Base address + 78h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																							RCV_RPTR								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIMx_RCV_FIFO_RPTR field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
RCV_RPTR	These bits indicate the receiver FIFO read pointer.

11.3 Universal Serial Bus Controller (USB)

11.3.1 Overview

The USB controller block provides high performance USB functionality that conforms to the *Universal Serial Bus Specification*, Rev. 2.0 (Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips; 2000), and the *On-The-Go and Embedded Host Supplement to the USB Revision* 2.0 Specification (Hewlett-Packard Company, Intel Corporation, LSI Corporation, Microsoft Corporation, Renesas Electronics Corporation, ST-Ericsson; 2012).

The USB controller consists of three independent USB controller cores: two On-The-Go (OTG) controller cores, and one Host-only controller core. Each controller core connects to its dedicated on-chip USB PHY instance using a UTMI interface. See Features for more details. All three controller cores are single-port cores.

The following figure is a block diagram of USB.

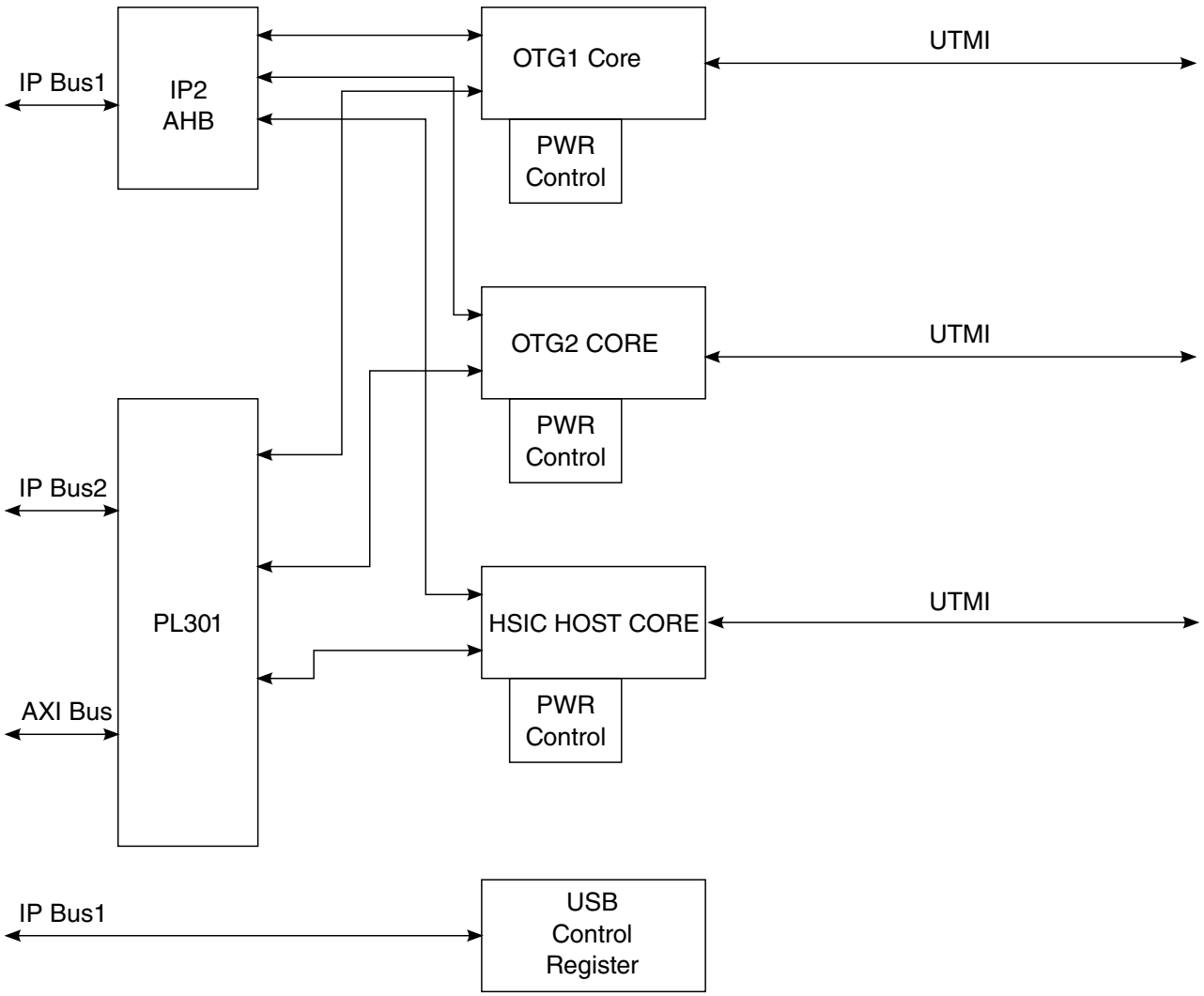


Figure 11-52. USB block diagram

11.3.1.1 Features

There are three USB 2.0 controller cores in this chip:

- Controller Core 0 is also named 'OTG1 Core'; its connected port is named 'OTG1 port'.
- Controller Core 1 is also named 'OTG2 Core'; its connected port is named 'OTG2 port'.
- Controller Core 2 is also named 'HSIC'.

The following list provides features of each of the controller cores.

- USB 2.0 Controller Core 0
  - High-Speed/Full-Speed/Low-Speed OTG core
  - HS/FS/LS UTMI compliant interface connected to on-chip UTMI PHY
  - High Speed, Full Speed and Low Speed operation in Host mode (with UTMI transceiver)
  - High Speed, and Full Speed operation in Peripheral mode (with UTMI transceiver)
  - Hardware support for OTG signaling, Session Request Protocol (SRP), Host Negotiation Protocol (HNP), and Attach Detection Protocol (ADP). ADP support includes dedicated timer hardware and register interface.
  - Up to 8 bidirectional endpoints
  - Supports charger detection with USB_OTG1_CHD_B pin and register interface
- USB 2.0 Controller Core 1
  - High-Speed/Full-Speed/Low-Speed OTG core
  - HS/FS/LS UTMI compliant interface connected to on-chip UTMI PHY
  - High Speed, Full Speed and Low Speed operation in Host mode (with UTMI transceiver)
  - High Speed, and Full Speed operation in Peripheral mode (with UTMI transceiver)
  - Hardware support for OTG signaling, Session Request Protocol (SRP), Host Negotiation Protocol (HNP), and Attach Detection Protocol (ADP). ADP support includes basic register interface only.
  - Up to 8 bidirectional endpoints
  - Supports charger detection with register interface only
- USB 2.0 Controller Core 2
  - High-Speed Host-Only core
  - HS UTMI interface connectes to on-chip HSIC PHY
- Low-power mode with local and remote wake-up capability
- Embedded DMA controller in each core

11.3.1.2 Modes of Operation

The USB has two main modes of operation: normal mode and low power mode.

Each USB OTG controller core can operate in High Speed operation (480 Mbps), Full Speed operation (12 Mbps) and Low Speed operation (1.5 Mbps, Host mode only).

The HSIC controller core supports High Speed operation (other USB speeds are not defined for HSIC).

This chapter explains the operation modes.

11.3.1.2.1 Normal Mode

The OTG controller core can operate in Host mode and Device (Peripheral) mode. The Host-only HSIC controller core can operate in Host mode only.

Each USB controller core has its corresponding port, which can work in one or more interface modes.

NOTE

Each controller supports only the interface type listed below. Selecting a different interface type in the PORTSC.PTS field results in unpredictable behavior and may cause the system to hang.

- OTG1 port
  - This port supports on-chip UTMI transceiver only.
- OTG2 port
  - This port supports on-chip UTMI transceiver only.
- Host1 Port
  - This port supports on-chip HSIC transceiver only.

    Interface for on-board HSIC compatible USB peripherals.

11.3.1.2.2 Low-Power Mode

Each USB controller core has a low-power mode (Suspend mode) to save power consumption.

As described in the USB 2.0 specification, the device can go into the Suspend state after it sees a constant Idle state on the upstream facing port. The OTG controller core enters Suspend mode after 3 ms of inactivity on the port when it is in Device Operation mode. Host controllers, including the OTG controller in Host mode, do not suspend automatically but can be placed in Suspend mode by software.

Either the local ARM platform or the remote USB Host/Peripheral can initiate a wake-up sequence to resume USB communication. For details about Suspend/Resume, see USB Power Control.

11.3.2 External Signals

The table found here describes the external signals of USB.

Table 11-54. USB External Signals

Signal	Description	Pad	Mode	Direction
USB_OTG1_ID	OTG1 ID Signal	GPIO1_IO02	ALT7	I
		GPIO1_IO12	ALT7	
		I2C4_SCL	ALT4	
		SD2_WP	ALT4	
USB_OTG1_OC	External Input for VBUS overcurrent detectoin	GPIO1_IO04	ALT1	I
		UART3_RXD	ALT1	
USB_OTG1_PWR	To control PMIC to supply VBUS voltage	GPIO1_IO05	ALT1	O
		UART3_TXD	ALT1	
USB_H_DATA	Data Signal	USB_H_DATA	No muxing	IO
USB_H_STROBE	Strobe Signal	USB_H_STROBE	No muxing	IO
USB_OTG1_CHD_B	Charge Detect Signal	USB_OTG1_CHD_B	No muxing	IO
USB_OTG1_DN	OTG1 DN Signal	USB_OTG1_DN	No muxing	O
USB_OTG1_DP	OTG1 DP Signal	USB_OTG1_DP	No muxing	IO
USB_OTG1_ID	OTG1 ID Signal	USB_OTG1_ID	No muxing	I
USB_OTG1_REXT	External reference resistor input.	USB_OTG1_REXT	No muxing	IO

11.3.3 Functional Description

These sections describe the functionality of the various building blocks of the USB.

11.3.3.1 USB 2.0 Controller Core 1

The USB 2.0 Controller 1 is an instantiation of an EHCI-compatible core which supports high-, full-, and low-speed operation.

In Host mode, this controller core supports high-, full-, and low-speed operation. In Device mode, it supports high- and full-speed operation.

11.3.3.1.1 Host Mode

The controller supports direct connection of a HS/FS/LS device with on-chip UTMI transceiver.

Although there is no separate Transaction Translator block in the system, the transaction translator function normally associated with a USB 2.0 high speed hub has been implemented within the DMA and protocol engine blocks to support connection to full and low speed devices.

11.3.3.1.2 Peripheral (Device) Mode

- Up to eight bidirectional endpoints
- High/full-speed operation
- Support of HNP, ADP, and SRP
- Remote wake-up capability

11.3.3.2 USB 2.0 Controller Core 1

USB 2.0 Controller Core 1 is an instantiation of EHCI-compatible core which supports High Speed / Full Speed / Low Speed operation.

11.3.3.3 USB Power Control

The USB controller supports suspend and wake-up functionality.

The power control block allows for placing the transceiver in USB low power mode when USB bus is IDLE, and supports local and remote wake-up to bring the transceiver out of USB low power mode when needed. Additionally, the power control block can wake-up the ARM platform from core sleep mode by generating an interrupt.

11.3.3.3.1 Entering Low Power Suspend Mode

In Host operation mode, low power suspend mode is entered as follows:

1. Clear the ASE and PSE bits in USB_USBCMD, and wait until the AS and PS bits in USB_USBSTS become "0".
2. Set the "SUSPEND" bit in USB_PORTSC1.
3. Set the "PHCD" bit in USB_PORTSC1.

For device operation mode, low power suspend mode is entered as follows:

1. After Host drive is IDLE for 3ms, an SLI interrupt is issued (the "DCSUSPEND" or "SLI" bit in USB_USBSTS).
2. Set the "PHCD" bit on USB_PORTSC1

11.3.3.3.2 Wake-Up Events

The power control block monitors the USB bus when the USB core is in the USB suspend state.

Depending on whether the core is on Host or Device mode, a number of wake-up conditions are monitored. Upon detection of a wake-up condition, an interrupt (asynchronous) will be generated to ARM platform if the related wake-up interrupt enable bit is set.

USB wake-up interrupt also re-activates the ARM platform clocks if they were stopped during the suspend.

11.3.3.3.2.1 Host Mode Events

The host controller wakes up on the following events:

- Remote Wake-up Request

A peripheral can request the host to reactivate the bus by driving wake-up signaling on the DM/DP lines. The power control block sends a wake-up request to the USB core when a J-K transition on DM/DP line is detected.

- Wake-Up On Overcurrent

If Wake-Up On Overcurrent is enabled (WKOC bit in the USB core register PORTSC1 is set '1'), the power control block sends a wake-up request to the USB core when an overcurrent event is detected.

- Wake-Up On Disconnect

If Wake-Up On Disconnect is enabled (WKDC bit in the USB core register PORTSC1 is set '1'), the power control block sends a wake-up request to the USB core when a disconnection event is detected (J-SE0/K-SE0 transition on DM/DP line).

- Wake-Up On Connect

If a Wake-Up On Connect is enabled (WKCN bit in the USB core register PORTSC1 is set '1'), the power control sub-block sends a wake-up request to the USB core when the connection event is detected (SE0-J/SE0-K transition on DM/DP line).

For a detailed description of register bits WKOC, WKDC, WKCN, please see Port Status & Control (USB_PORTSC1).

11.3.3.3.2.2 USB OTG Power Domains

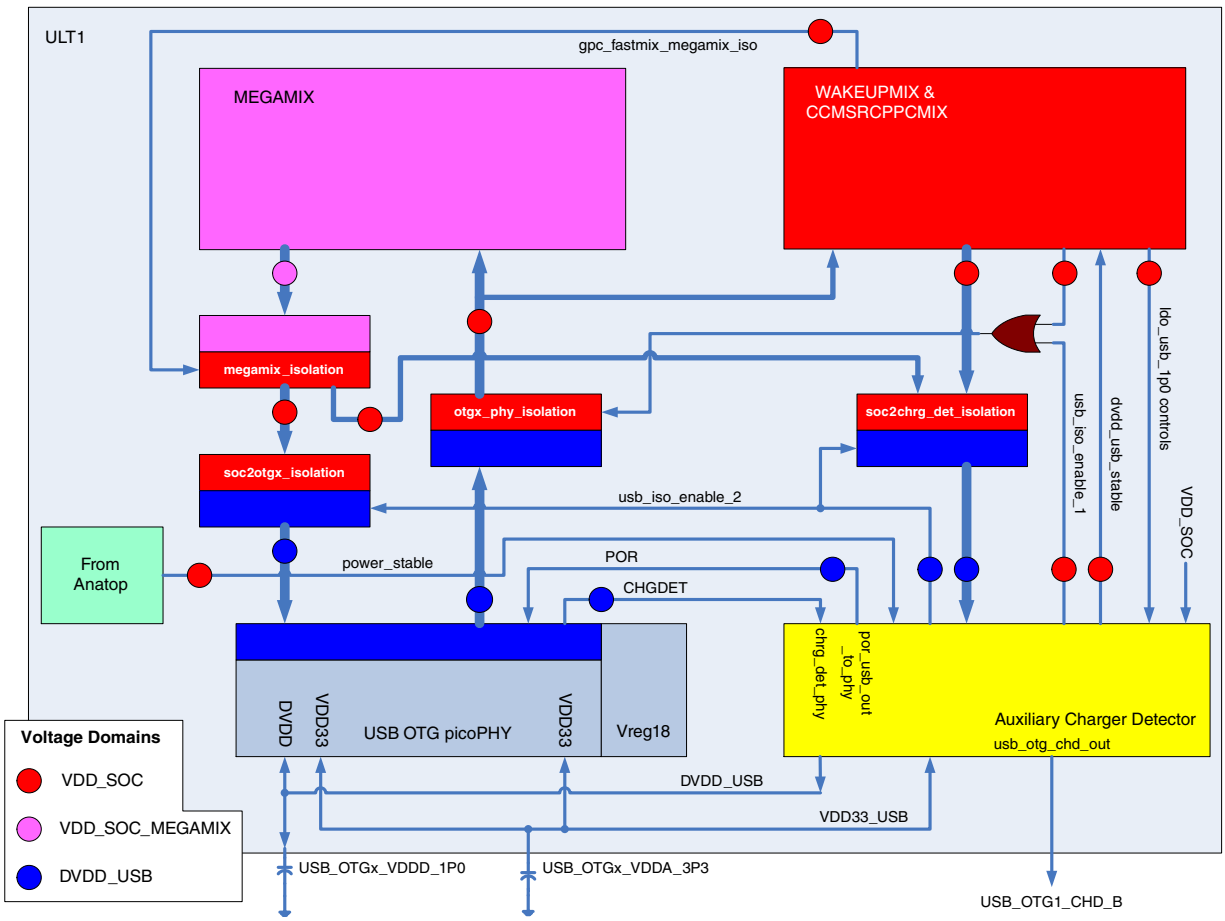


Figure 11-53. USB OTG Power Domains

11.3.3.3.2.2.1 Power Overview

USB controller and USB OTG PHY are in different power domains. USB controller locates in megamix power domain. USB OTG PHY has three powers: VDD33, VDD18, and DVDD (1.0 V).

- VDD33 comes from external of the chip, which may be from VBUS or 3.3 V power supply depend on the board design.
- VDD18 is generated by on-chip regulator vreg18 from VDD33.
- DVDD is generated by on-chip regulator inside the Auxiliary Charger Detector from VDD33.

All three OTG PHY power supplies turn on/off at same time.

Auxiliary Charger Detector can detect each power to control related power isolations and generate reset signal for the PHY.

11.3.3.3.2.2.2 Wake-up in Power Down Mode

NOTE

USB controller or OTG PHY can only power down during USB disconnect mode, power down any part when device/host is connected will cause unknown result.

11.3.3.3.2.2.2.1 Power Down USB Controller

When USB controller is power down, it can be wake-up by changes on PHY port. This is completed by the wake-up logic in the wakeupmix (it is always on power domain).

NOTE

This wake-up does not generate any interrupts, the wake-up event is only used to power up the system (includes USB controller), software re-initials the USB controller like a system power up event.

The wake-up source in the USB controller power down mode includes following:

- ID
- VBUS/SESS_VLD
- DP/DM

The related enable signals come from USB_x_CTRL1, software set related enable signals before USB controller is power down. The latch based isolation cells keep these enable values.

11.3.3.4 Interrupts

11.3.3.4.1 USB Core Interrupts

Each USB core uses one dedicated vector in the Interrupt Table. The vector numbers associated with each of the cores can be found in the Interrupt section.

With the exception of the wake-up interrupts, all of the interrupt sources are controlled in the USB Cores. Refer to the Interrupt Enable Register (USB_USBINTR) for details.

11.3.3.4.2 USB Wake-Up Interrupts

Each USB Core has an associated wake-up interrupt. The wake-up interrupts are generated outside of the USB controller cores, but using the same vector as the corresponding USB controller cores interrupt.

These interrupts are generated by the Power Control blocks which run on the 32 KHz standby clock. The wake-up interrupt is designed to work even when the USB and ARM platform clocks are disabled, such that a wake-up condition on the USB bus can re-activate the ARM platform clocks.

Because the wake-up interrupt is generated and cleared on a 32 KHz clock, this interrupt request responds very slowly to clear actions. For this reason, the software must disable the wake-up interrupt to clear the request flag. Disabling the interrupt masks the request instantaneously as this is clocked by the ARM platform clock. The software should wait for at least three 32 KHz clock cycles before re-enabling this interrupt to allow sufficient time for the request flag to clear. Because this interrupt is only used during low power modes of the USB, it is sufficient to enable the wake-up interrupt just prior to entering the USB suspend mode.

11.3.4 USB Operation Model

11.3.4.1 Register Interface

Configuration, control and status registers are divided into three categories, identification, capability and operational registers.

NOTE

USB controller registers support only DWORD (32-bit) access.

- Identification registers are used to declare the slave interface presence along with the complete set of the hardware configuration parameters.
- Static, read only capability registers define the software limits, restrictions, and capabilities of the host/device controller.
- Operational registers are dynamic control or status registers that may be read only, read/write, or read/write-to-clear. The following sections define the use of these registers.

EHCI registers are listed alongside device registers to show the complementary nature of host and device control.

The following table describes the Interface register sets.

Table 11-55. Interface Register Sets

Offset	Register Set	Explanation
000h-07Ch	Identification Registers	Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.
100h-124h	Capability Registers	Capability registers specify the limits, restrictions, and capabilities of a host/device controller implementation. These values are used as parameters to the host/device controller driver.
080h-0FCh 140h-1FCh	Operational Registers	Operational registers are used by the system software to control and monitor the operational state of the host/device controller.

11.3.4.1.1 Configuration, Control and Status Register Set

The following table describes the Device/Host capability registers.

.

Table 11-56. Device/Host Capability Registers

Offset	Size (Bytes)	Mnemonic	Register Name	Device Mode	Host Mode
000h	4	USB_*x*_ID	Identification Register	O	O
004h	4	USB_*x*_HWGENERAL	General Hardware Parameters	O	O
008h	4	USB_*x*_HWHOST	Host Hardware Parameters	X	O
00Ch	4	USB_*x*_HWDEVICE	Device Hardware Parameters	O	X
010h	4	USB_*x*_HWTXBUF	TX Buffer Hardware Parameters	O	O
014h	4	USB_*x*_HWRXBUF	RX Buffer Hardware Parameters	O	O
018-07Fh		-	Reserved		
080h	4	USB_*x*_GPTIMER0LD	General Purpose Timer #0 Load Register	O	O
084h	4	USB_*x*_GPTIMER0CTRL	General Purpose Timer #0 Control Register	O	O
088h	4	USB_*x*_GPTIMER1LD	General Purpose Timer #1 Load Register	O	O
08Ch	4	USB_*x*_GPTIMER1CTRL	General Purpose Timer #1 Control Register	O	O
090h	4	USB_*x*_SBUSCFG	System Bus Interface Configuration Register	O	O
094-09Fh		-	Reserved		
100h	1	USB_*x*_CAPLENGTH	Capability Register Length	O	O
101h		-	Reserved		
102h	2	USB_*x*_HCIVERSION	Host Controller Interface Version Number	X	O
104h	4	USB_*x*_HCSPARAMS	Host Controller Structural Parameters	X	O
108h	4	USB_*x*_HCCPARAMS	Host Controller Capability Parameters	X	O
10C-11Fh		-	Reserved		
120h	2	USB_*x*_DCIVERSION	Device Controller Interface Version Number	O	X
122h	2	-	Reserved		

Table continues on the next page...

Table 11-56. Device/Host Capability Registers (continued)

Offset	Size (Bytes)	Mnemonic	Register Name	Device Mode	Host Mode
124h	4	USB_*x*_DCCPARAMS	Device Controller Capability Parameters	O	X
128-13Fh		-	Reserved		
140h	4	USB_*x*_USBCMD	USB Command Register	O	O
144h	4	USB_*x*_USBSTS	USB Status Register	O	O
148h	4	USB_*x*_USBINTR	USB Interrupt Enable Register	O	O
14Ch	4	USB_*x*_FRINDEX	USB Frame Index	O	O
150h	4	-	Reserved		
154h	4	USB_*x*_PERIODICLISTBASE	Frame List Base Address	X	O
		USB_*x*_DEVICEADDR	USB Device Address	O	X
158h	4	USB_*x*_ASYNCLISTADDR	Next Asynchronous List Address	X	O
	4	USB_*x*_ENDPOINTLISTADDR	Address at Endpoint list in memory	O	X
15Ch	4	-	Reserved		
160h	4	USB_*x*_BURSTSIZE	Programmable Burst Size	O	O
164h	4	USB_*x*_TXFILLTUNING	Host Transmit Pre-Buffer Packet Tuning	X	O
180h	4	USB_*x*_CONFIGFLAG	Configured Flag Register	X	O
184h	4	USB_*x*_PORTSC1	Port Status/Control Register 1	O	O
188-1A3h		-	Reserved		
1A4h	4	USB_*x*_OTGSC	On-The-Go Status/Control Register	O	O
1A8h	4	USB_*x*_USBMODE	USB Controller Operating Mode	O	O
1ACh	4	USB_*x*_ENDPTSETUPSTAT	Endpoint Setup Status	O	X
1B0h	4	USB_*x*_ENDPTPRIME	Endpoint Initialization	O	X
1B4h	4	USB_*x*_ENDPTFLUSH	Endpoint De-Initialization	O	X
1B8h	4	USB_*x*_ENDPTSTATUS	Endpoint Status	O	X
1BCh	4	USB_*x*_ENDPTCOMPLETE	Endpoint Complete	O	X
1C0 1C4 ... 1DCh	64	USB_*x*_ENDPTCTRL0 USB_*x*_ENDPTCTRL1 .... USB_*x*_ENDPTCTRL7	Endpoint Control Register 0-7	O	X

NOTE

"O" means the register is available in host/device operation mode;

"X" means the register is reserved in host/device operation mode

11.3.4.1.2 Identification Registers

Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.

11.3.4.1.3 OTG Operations

11.3.4.1.3.1 Register Bits

In the previous section, the Register interface has behaviors described for device mode and behaviors described for host mode. However, for OTG operations it is necessary to perform tasks independent of the controller mode.

The following figure shows the controller mode.

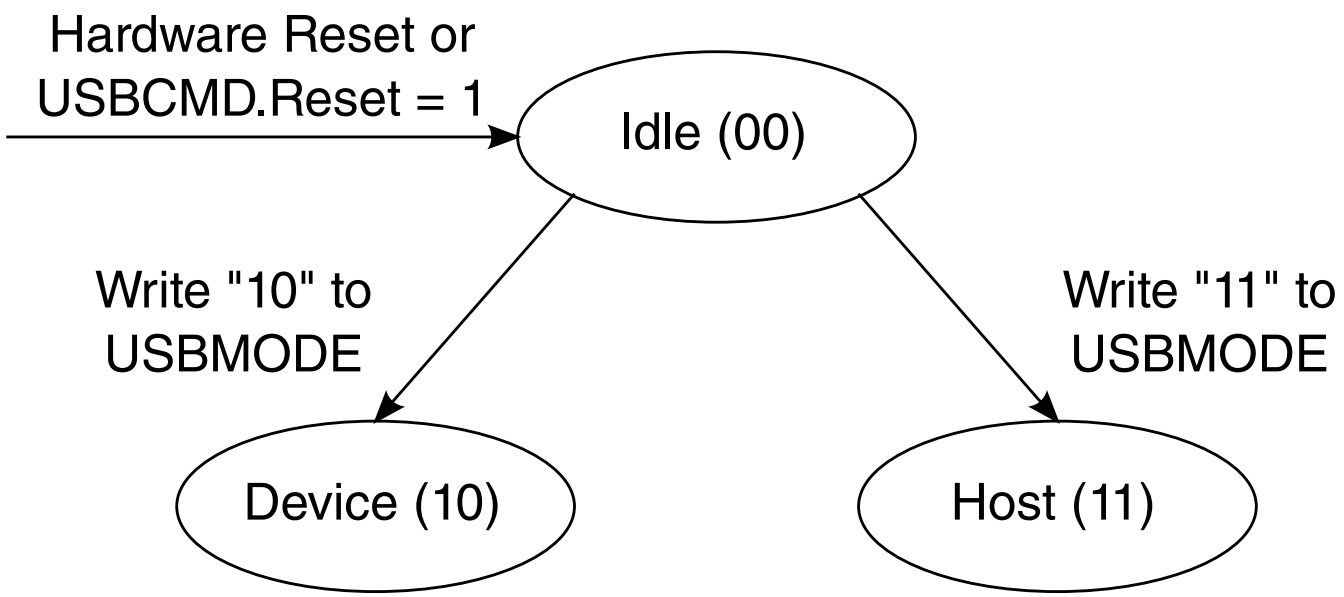


Figure 11-54. Controller Mode

To this end, listed below are the register bits that are used for OTG operations, which are independent of the controller mode and are also not affected by a write to the reset bit in the USBCMD register:

All Identification Registers

All Device/Host Capability Registers

OTGSC: All bits

PORTSC1:

Physical Interface Select

Physical Interface Serial Select

Physical Interface Data Width

Physical Interface Low Power

Physical Interface Wake Signals

Port Indicators

Port Power

11.3.4.2 Host Data Structures

This section defines the interface data structures used to communicate control, status, and data between HCD (software) and the Enhanced Host Controller (hardware).

The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of a Periodic Schedule, Periodic Frame List, Asynchronous Schedule, Isochronous Transaction Descriptors, Split-transaction Isochronous Transfer Descriptors, Queue Heads, and Queue Element Transfer Descriptors.

The periodic frame list is the root of all periodic (isochronous and interrupt transfer type) transfers for the host controller. The asynchronous list is the root for all the bulk and control transfers. Isochronous data streams are managed using Isochronous Transaction Descriptors. Isochronous split-transaction data streams are managed with Split-transaction Isochronous Transfer Descriptors. All Interrupt, Control, and Bulk data streams are managed via queue heads and Queue Element Transfer Descriptors. These data structures are optimized to reduce the total memory footprint of the schedule and to reduce (on average) the number of memory accesses needed to execute a USB transaction.

Note that software must ensure that no interface data structure reachable by the EHCI host controller spans a 4 K-page boundary.

The data structures defined in this section are (from the host controller's perspective) a mix of read-only and read/writeable fields. The host controller must preserve the read-only fields on all data structure writes.

11.3.4.2.1 Periodic Frame List

This schedule is for all periodic transfers (isochronous and interrupt). The periodic schedule is referenced from the operational registers space using the USB_PERIODICLISTBASE address register and the USB_FRINDEX register.

The periodic schedule is based on an array of pointers called the Periodic Frame List.

The USB_PERIODICLISTBASE address register is combined with the USB_FRINDEX register to produce a memory pointer into the frame list. The Periodic Frame List implements a sliding window of work over time.

The following figure shows the organization of periodic schedule.

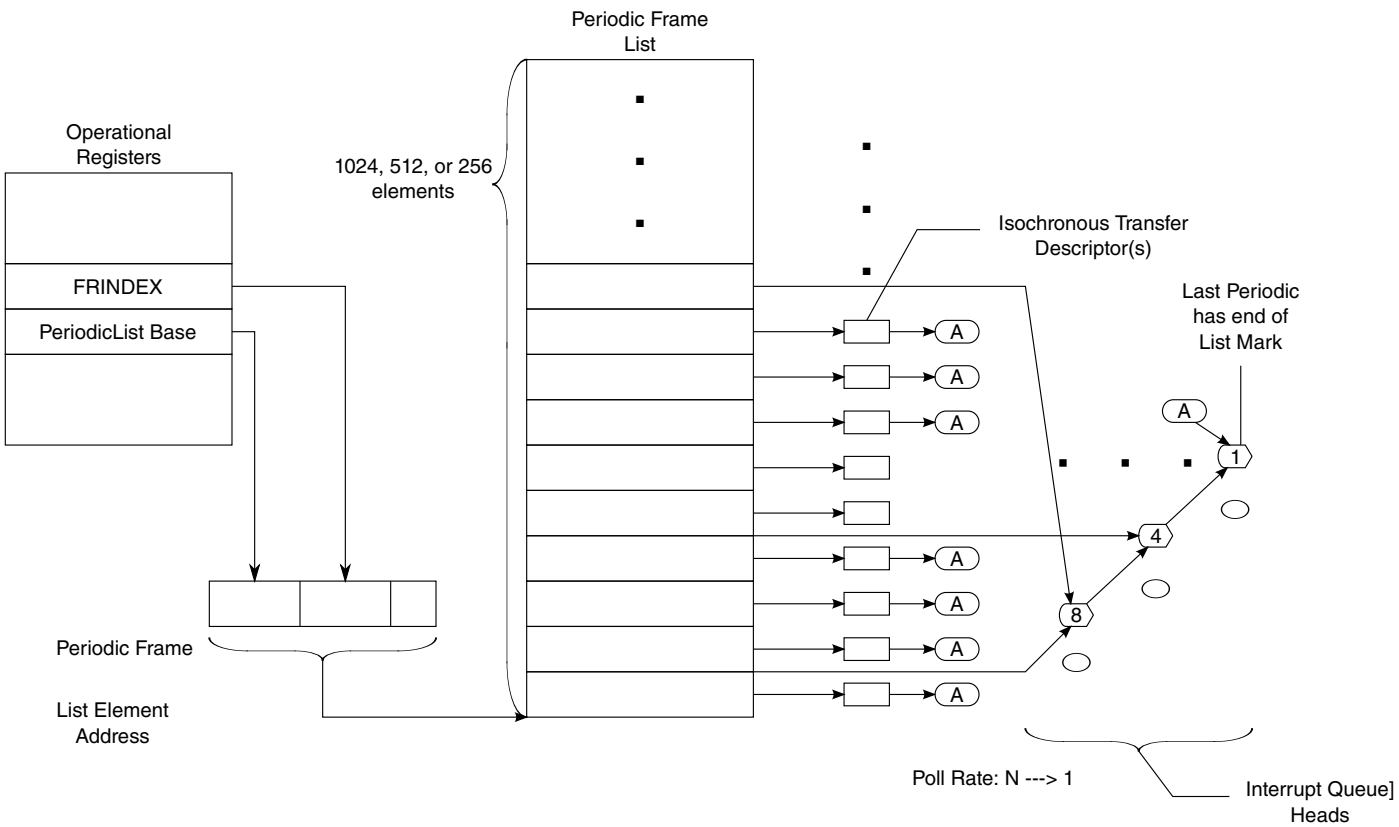


Figure 11-55. Periodic Schedule Organization

Split transaction Interrupt, Bulk and Control are also managed using queue heads and queue element transfer descriptors.

The periodic frame list is a 4 K-page aligned array of Frame List Link pointers. The length of the frame list may be programmable. The programmability of the periodic frame list is exported to system software via the USB_HCCPARAMS register. If non-programmable, the length is 1024 elements. If programmable, the length can be selected by system software as one of 256, 512, or 1024 elements. An implementation must

support all three sizes. Programming the size (that is, the number of elements) is accomplished by system software writing the appropriate value into Frame List Size field in the USB_USBCMD register.

Frame List Link pointers direct the host controller to the first work item in the frame's periodic schedule for the current micro-frame. The link pointers are aligned on DWord boundaries within the Frame List.

The table below illustrates the format of the Frame list element pointer.

Table 11-57. Format of Frame List Element Pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Frame List Link Pointer																											0		Typ			03-00H

Frame List Link pointers always reference memory objects that are 32-byte aligned. The referenced object may be an isochronous transfer descriptor for high-speed devices, a split-transaction isochronous transfer descriptor (for full-speed isochronous endpoints), or a queue head (used to support high-, full- and low-speed interrupt). System software should not place non-periodic schedule items into the periodic schedule. The least significant bits in a frame list pointer are used to key the host controller as to the type of object the pointer is referencing.

The least significant bit is the T-Bit (bit 0). When this bit is set to a one, the host controller never uses the value of the frame list pointer as a physical memory pointer. The Typ field is used to indicate the exact type of data structure being referenced by this pointer. The value encodings are.

Table 11-58. Typ Field Value Definitions

Value	Meaning
00b	Isochronous Transfer Descriptor
01b	Queue Head
10b	Split Transaction Isochronous Transfer Descriptor.
11b	Frame Span Traversal Node.

11.3.4.2.2 Asynchronous List Queue Head Pointer

The Asynchronous Transfer List (based at the USB_ASYNCLISTADDR register) is where all of the control and bulk transfers are managed.

Host controllers use this list only when it reaches the end of the periodic list, the periodic list is disabled, or the periodic list is empty.

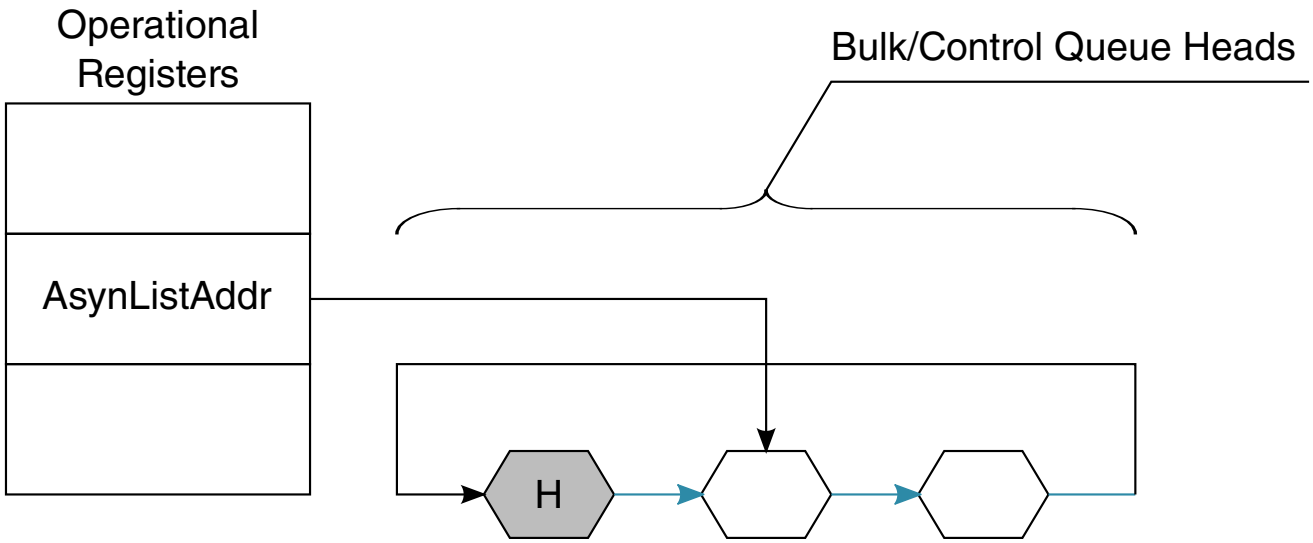


Figure 11-56. Asynchronous Schedule Organization

The Asynchronous list is a simple circular list of queue heads. The USB_ASYNCLISTADDR register is simply a pointer to the next queue head. This implements a pure round-robin service for all queue heads linked into the asynchronous list.

11.3.4.2.3 Isochronous (High-Speed) Transfer Descriptor (iTD)

The format of an isochronous transfer descriptor is shown in the table below.

This structure is used only for high-speed isochronous endpoints. All other transfer types should use queue structures. Isochronous TDs must be aligned on a 32-byte boundary.

Table 11-59. Isochronous Transaction Descriptor (iTD)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Next Link Pointer																											0		Typ		T	03-00 H
Status				Transaction 0 Length												IOC	PG*			Transaction 0 Offset*													07-04 H
Status				Transaction 1 Length												IOC	PG*			Transaction 1 Offset*													0B-08H
Status				Transaction 2 Length												IOC	PG*			Transaction 2 Offset*													0F-0CH
Status				Transaction 3 Length												IOC	PG*			Transaction 3 Offset*													13-10 H

Table continues on the next page...

Table 11-59. Isochronous Transaction Descriptor (iTD) (continued)

Status	Transaction 4 Length	IOC	PG*	Transaction 4 Offset*				17-14H
Status	Transaction 5 Length	IOC	PG*	Transaction 5 Offset*				1B-18H
Status	Transaction 6 Length	IOC	PG*	Transaction 6 Offset*				1F-1CH
Status	Transaction 7 Length	IOC	PG*	Transaction 7 Offset*				23-20H
Buffer Pointer (Page 0)				EndPt	R	Device Address		27-24H
Buffer Pointer (Page 1)				I/O	Maximum Packet Size			2B-28H
Buffer Pointer (Page 2)				-			Mult	2F-2CH
Buffer Pointer (Page 3)				-				33-30H
Buffer Pointer (Page 4)				-				37-34H
Buffer Pointer (Page 5)				-				3B-38H
Buffer Pointer (Page 6)				-				3F-3CH

Host Controller Read/Write

Host Controller Read Only

These fields may be modified by the host controller if the I/O field indicates an OUT.

11.3.4.2.3.1 Next Link Pointer

The first DWord of an iTD is a pointer to the next schedule data structure.

The following table describes the Next Schedule Element pointer field.

Table 11-60. Next Schedule Element Pointer

Bit	Description
31-5 Link Pointer (LP)	These bits correspond to memory address signals [31:5], respectively. This field points to another Isochronous Transaction Descriptor (iTD/siTD) or Queue Head (QH).

Table continues on the next page...

Table 11-60. Next Schedule Element Pointer (continued)

4-3 Reserved	These bits are reserved and their value has no effect on operation. Software should initialize this field to zero.
2-1 QH/(s)iTD Select (Typ)	This field indicates to the Host Controller whether the item referenced is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor 11b FSTN (frame span traversal node)
0 Terminate (T)	1= Link Pointer field is not valid. 0= Link Pointer field is valid.

11.3.4.2.3.2 iTD Transaction Status and Control List

DWords 1 through 8 are eight slots of transaction control and status.

Each transaction description includes:

- Status results field
- Transaction length (bytes to send for OUT transactions and bytes received for IN transactions).
- Buffer offset. The PG and Transaction X Offset fields are used with the buffer pointer list to construct the starting buffer address for the transaction.

The host controller uses the information in each transaction description plus the endpoint information contained in the first three DWords of the Buffer Page Pointer list, to execute a transaction on the USB.

The following table describes iTD Transaction Status and Control fields.

Table 11-61. iTD Transaction Status and Control

Bit	Description	
31-28 Status	This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:	
	Bit	Definition
	31	Active. Set to one by software to enable the execution of an isochronous transaction by the Host Controller. When the transaction associated with this descriptor is completed, the Host Controller sets this bit to zero indicating that a transaction for this element should not be executed when it is next encountered in the schedule.
	30	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, no action is necessary.
	29	Babble Detected. Set to one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor.

Table continues on the next page...

Table 11-61. iTD Transaction Status and Control (continued)

Bit	Description	
	28	Transaction Error (XactErr). Set to one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit may only be set for isochronous IN transactions.
27-16 Transaction X Length	For an OUT, this field is the number of data bytes the host controller sends during the transaction. The host controller is not required to update this field to reflect the actual number of bytes transferred during the transfer. For an IN, the initial value of the endpoint to deliver. During the status update, the host controller writes back the field is the number of bytes the host expects the number of bytes successfully received. The value in this register is the actual byte count (0‡zero length data, 1‡one byte, 2‡two bytes, etc.). The maximum value this field may contain is 0xC00 (3072).	
15 Interrupt On Complete (IOC)	If this bit is set to one, it specifies that when this transaction completes, the Host Controller should issue an interrupt at the next interrupt threshold.	
14-12 Page Select (PG)	These bits are set by software to indicate which of the buffer page pointers the offset field in this slot should be concatenated to produce the starting memory address for this transaction. The valid range of values for this field is 0 to 6.	
11-0 Transaction X Offset	This field is a value that is an offset, expressed in bytes, from the beginning of a buffer. This field is concatenated onto the buffer page pointer indicated in the adjacent *PG* field to produce the starting buffer address for this transaction.	

11.3.4.2.3.3 iTD Buffer Page Pointer List (Plus)

DWords 9-15 of an isochronous transaction descriptor are nominally page pointers (4 K aligned) to the data buffer for this transfer descriptor. This data structure requires the associated data buffer to be contiguous (relative to virtual memory), but allows the physical memory pages to be non-contiguous.

Seven page pointers are provided to support the expression of eight isochronous transfers. The seven pointers allow for 3 (transactions) * 1024 (maximum packet size) * 8 (transaction records) (24576 bytes) to be moved with this data structure, regardless of the alignment offset of the first page.

Because each pointer is a 4 K aligned page pointer, the least significant 12 bits in several of the page pointers are used for other purposes.

The tables below illustrate the field descriptions.

Table 11-62. iTD Buffer Pointer Page 0 (Plus)

Bit	Description
31-12 Buffer Pointer (Page 0)	This is a 4 K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11-8	This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.

Table continues on the next page...

Table 11-62. iTD Buffer Pointer Page 0 (Plus) (continued)

Bit	Description
Endpoint Number (Endpt)	
7 Reserved	Bit reserved for future use and should be initialized by software to zero.
6-0 Device Address	This field selects the specific device serving as the data source or sink.

Table 11-63. iTD Buffer Pointer Page 1 (Plus)

Bit	Description
31-12 Buffer Pointer (Page 1)	This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11 Direction (I/O)	0 = OUT; 1 = IN. This field encodes whether the high-speed transaction should use an IN or OUT PID.
10-0 Maximum Packet Size	This directly corresponds to the maximum packet size of the associated endpoint *(wMaxPacketSize)*. This field is used for high-bandwidth endpoints where more than one transaction is issued per transaction description (per micro-frame). This field is used with the *Multi* field to support high-bandwidth pipes. This field is also used for all IN transfers to detect packet babble. Software should not set a value larger than 1024 (400h). Any value larger yields undefined results.

Table 11-64. iTD Buffer Pointer Page 2 (Plus)

Bit	Description
31-12 Buffer Pointer	This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11-2 Reserved	This bit reserved for future use and should be set to zero.
1-0 Multi	This field is used to indicate to the host controller the number of transactions that should be executed per transaction description (per micro-frame). The valid values are: Value Meaning 00b Reserved. A zero in this field yields undefined results. 01b One transaction to be issued for this endpoint per micro- frame. 10b Two transactions to be issued for this endpoint per micro- frame. 11b Three transactions to be issued for this endpoint per micro- frame.

Table 11-65. iTD Buffer Pointer Page 3-6

Bit	Description
31-12 Buffer Pointer	This is a 4 K aligned pointer to physical memory. Corresponds to memory address bits [31:12].

Table continues on the next page...

Table 11-65. iTD Buffer Pointer Page 3-6 (continued)

Bit	Description
11-0 Reserved	These bits reserved for future use and should be set to zero.

11.3.4.2.4 Split Transaction Isochronous Transfer Descriptor (siTD)

All Full-speed isochronous transfers through the internal transaction translator are managed using the siTD data structure. This data structure satisfies the operational requirements for managing the split transaction protocol.

The following table shows the Split Transaction Isochronous Transfer Descriptor (siTD).

Table 11-66. Split Transaction Isochronous Transfer Descriptor

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Next Link Pointer																											0		Typ		T	03-00
I/O	Port Number							-	Hub Addr							Reserved				EndPt				-	Device Address							07-04[1]
Reserved																µFrame C-mask				µFrame S-mask												0B-08[1]
ioc	P	Reserved				Total Bytes to Transfer										µFrame C-prog-mask				Status												0F-0C[2]
Buffer Pointer (Page 0)																				Current Offset												13-10[2]
Buffer Pointer (Page 1)																				Reserved							TP		T-count			17-14[2]
Back Pointer																											0				T	1B-18

1. 04-0B: Static Endpoint State
2. 0C-13: Transfer results

Host Controller Read/Write

Host Controller Read Only

11.3.4.2.4.1 Next Link Pointer

DWord0 of a siTD is a pointer to the next schedule data structure.

The following table describes the Next Link Pointer fields.

Table 11-67. Next Link Pointer

Bit	Description
31-5	Next Link Pointer (LP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4-3	Reserved. These bits must be written as zeros.
2-1	QH/(s)iTD Select (Typ). This field indicates to the Host Controller whether the item referenced is an iTD/siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor 11b FSTN (frame span traversal node)
0	Terminate (T). 1 = Link Pointer field is not valid. 0 = Link Pointer is valid.

11.3.4.2.4.2 siTD Endpoint Capabilities/Characteristics

DWords 1 and 2 specify static information about the full-speed endpoint, the addressing of the parent Companion Controller, and micro-frame scheduling control.

The tables below describe the Endpoint and transaction translator characteristics and micro-frame schedule control fields.

Table 11-68. Endpoint and Transaction Translator Characteristics

Bit	Description
31	Direction (I/O).0 = OUT; 1 = IN. This field encodes whether the full-speed transaction should be an IN or OUT.
30-24	Port Number. This field is the port number of the recipient Transaction Translator.
23	Reserved. Bit reserved and should be set to zero.
22-16	Hub Address. This field holds the device address of the Companion Controllers' hub.
15-12	Reserved. Field reserved and should be set to zero.
11-8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Reserved. Bit is reserved for future use. It should be set to zero.
6-0	Device Address. This field selects the specific device serving as the data source or sink.

Table 11-69. Micro-frame Schedule Control

Bit	Description
31-16	Reserved. This field reserved for future use. It should be set to zero.

Table continues on the next page...

Table 11-69. Micro-frame Schedule Control (continued)

Bit	Description
15-8	Split Completion Mask (mFrame C-Mask). This field (along with the *Active* and *SplitX- state* fields in the *Status* byte) is used to determine during which micro-frames the host controller should execute complete-split transactions. When the criteria for using this field is met, an all zeros value has undefined behavior. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the m*Frame C-Mask* field is a one, then this siTD is a candidate for transaction execution. There may be more than one bit in this mask set.
7-0	Split Start Mask (mFrame S-mask). This field (along with the *Active* and *SplitX-state* fields in the *Status* byte) is used to determine during which micro-frames the host controller should execute start-split transactions. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the m*Frame S-mask* field is a one, then this siTD is a candidate for transaction execution. An all zeros value in this field, in combination with existing periodic frame list has undefined results.

11.3.4.2.4.3 siTD Transfer State

DWords 3-6 are used to manage the state of the transfer.

The following table describes siTD transfer state fields.

Table 11-70. siTD Transfer Status and Control

Bit	Description
31	Interrupt On Complete (ioc). 0 = Do not interrupt when transaction is complete. 1 = Do interrupt when transaction is complete. When the host controller determines that the split transaction has completed it asserts a hardware interrupt at the next interrupt threshold.
30	Page Select (P). Used to indicate which data page pointer should be concatenated with the *CurrentOffset* field to construct a data buffer pointer (0 selects *Page 0* pointer and 1 selects *Page 1*). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero).
29-26	Reserved. This field reserved for future use and should be set to zero.
25-16	Total Bytes To Transfer. This field is initialized by software to the total number of bytes expected in this transfer. Maximum value is 1023 (3FFh)
15-8	µFrame Complete-split Progress Mask (C-prog-Mask). This field is used by the host controller to record which split-completes has been executed.
7-0: Status—This field records the status of the transaction executed by the host controller for this slot. It is a bit vector with the encoding shown in the following rows.	
7	Active. Set to one by software to enable the execution of an isochronous split transaction by the Host Controller.
6	ERR. Set to a one by the Host Controller when an ERR response is received from the Companion Controller.
5	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). In the case of an under run, the Host Controller transmits an incorrect CRC (thus invalidating the data at the endpoint). If an overrun condition occurs, no action is necessary.
4	Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor.

Table continues on the next page...

Table 11-70. siTD Transfer Status and Control (continued)

Bit	Description
3	Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit is set only for IN transactions.
2	Missed Micro-Frame. The host controller detected that a host-induced hold- off caused the host controller to miss a required complete-split transaction.
1	Split Transaction State (SplitXstate). The bit encodings are: Value Meaning 00b Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint when a match is encountered in the S-mask. 01b Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint when a match is encountered in the C-mask.
0	Reserved. Bit reserved for future use and should be set to zero.

11.3.4.2.4.4 siTD Buffer Pointer List (plus)

DWords 4 and 5 are the data buffer page pointers for the transfer. This structure supports one physical page cross. The most significant 20 bits of each DWord in this section are the 4 K (page) aligned buffer pointers.

The least significant 12 bits of each DWord are used as additional transfer state. The following table describes the siTD buffer pointer fields.

Table 11-71. Buffer Page Pointer List (plus)

Bit	Description
31-12	Buffer Pointer List. Bits [31:12] of DWords 4 and 5 are 4 K page aligned physical memory addresses. These bits correspond to physical address bits [31:12] respectively. The lower 12 bits in each pointer are defined and used as specified below. The field *P* (see siTD Transfer State) specifies the *current* active pointer.
Bits 11-0 (Page 0)	Current Offset—The 12 least significant bits of the Page 0 pointer are the current byte offset for the current page pointer (as selected with the page indicator bit (*P* field). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero).
Bits 11-0 (Page 1)—The least significant bits of the Page 1 pointer are split into three subfields as shown in the following rows.	
11-5 (Page 1)	Reserved
4-3 (Page 1)	Transaction position (TP). This field is used with T-count to determine whether to send *all*, *first*, *middle*, or *last* with each outbound transaction payload. System software must initialize this field with the appropriate starting value. The host controller must correctly manage this state during the lifetime of the transfer. The bit encodings are: Value Meaning

Table continues on the next page...

Table 11-71. Buffer Page Pointer List (plus) (continued)

Bit	Description
	00b All. The entire full-speed transaction data payload is in this transaction (that is, less than or equal to 188 bytes). 01b Begin. This is the first data payload for a full- speed that is greater than 188 bytes. 10B Mid. This is the *middle* payload for a full-speed OUT transaction that is larger than 188 bytes. 11b End. This is the *last* payload for a full-speed OUT transaction that was larger than 188 bytes.
2-0 (Page 1)	Transaction count (T-Count). Software initializes this field with the number of OUT start-splits this transfer requires. Any value larger than 6 is undefined.

11.3.4.2.4.5 siTD Back Link Pointer

DWord 6 of a siTD is simply another schedule link pointer. This pointer is always zero, or references a siTD, and it cannot reference any other schedule data structure.

The following table describes the siTD back link pointer fields.

Table 11-72. siTD Back Link Pointer

Bit	Description
31-5	siTD Back Pointer. This field is a physical memory pointer to a siTD.
4-1	Reserved. This field is reserved for future use. It should be set to zero.
0	Terminate (T). 1 = siTD Back Pointer field is not valid. 0 = siTD Back Pointer field is valid.

11.3.4.2.5 Queue element transfer descriptor (qTD)

This data structure is only used with a queue head. It describes one or more USB transactions to transfer up to 20480 (5*4096) bytes.

The structure contains two structure pointers used for queue advancement, a DWord of transfer state, and a five-element array of data buffer pointers.

It is 32 bytes and must be physically contiguous.

The buffer associated with this transfer must be virtually contiguous. The buffer may start on any byte boundary; however, for optimal utilization of on-chip busses it is recommended to align the buffers on a 32-byte boundary. A separate buffer pointer list element must be used for each physical page in the buffer, regardless of whether the buffer is physically contiguous.

Host controller updates (host controller writes) to stand-alone qTDs only occur during transfer retirement. References in the following bit field definitions of updates to the qTD are to the qTD portion of a queue head.

The following table shows the queue element transfer descriptor data structure.

Table 11-73. Queue element transfer descriptor data structure

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Next qTD Pointer																											0				T	03-00
Alternate Next qTD Pointer																											0				T	07-04
dt	Total Bytes to Transfer															ioc	C_Page			Cerr		PID Code		Status								0B-08[1]
Buffer Pointer (page 0)																				Current Offset												0F-0C[1]
Buffer Pointer (page 1)																				Reserved												13-10
Buffer Pointer (page 2)																				Reserved												17-14
Buffer Pointer (page 3)																				Reserved												1B-18
Buffer Pointer (page 4)																				Reserved												1F-1C

1. 08-0F: Transfer Results

Host Controller Read/Write

Host Controller Read Only

Queue Element Transfer Descriptors must be aligned on 32-byte boundaries.

11.3.4.2.5.1 Next qTD Pointer

The first DWord of an element transfer descriptor is a pointer to another transfer element descriptor.

The following table describes Next qTD pointer fields.

Table 11-74. qTD Next Element Transfer Pointer (DWord 0)

Bit	Description
31-5	Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed. The field corresponds to memory address signals[31:5], respectively.
4-1	Reserved
0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

11.3.4.2.5.2 Alternate Next qTD Pointer

The second DWord of a queue element transfer descriptor is used to support hardware-only advance of the data stream to the next transfer descriptor on short packet. To be more explicit the host controller always uses this pointer when the current qTD is retired due to short packet.

The following table describes the TD Alternate Next Element Transfer Pointer field descriptions.

Table 11-75. TD Alternate Next Element Transfer Pointer (DWord 1)

Bit	Description
31-5	Alternate Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed in the event that the current qTD execution encounters a short packet (for an IN transaction). The field corresponds to memory address signals [31:5], respectively.
4-1	Reserved
0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

11.3.4.2.5.3 qTD Token

The third DWord of a queue element transfer descriptor contains most of the information the host controller requires to execute a USB transaction (the remaining endpoint-addressing information is specified in the queue head).

NOTE

The field descriptions forward reference fields defined in the queue head. Where necessary, these forward references are preceded with a QH notation.

The following table describes the TD Token fields.

Table 11-76. TD Token (DWord 2)

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31
Data Toggle</td><td>This is the data toggle sequence bit. The use of this bit depends on the setting of the Data Toggle Control bit in the queue head.</td></tr>
<tr><td>30-16
Total Bytes to Transfer</td><td>This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction, only on the successful completion of the transaction. The maximum value software may store in this field is 5 * 4K (5000H). This is the maximum number of bytes 5 page pointers can access. If the value of this field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the host controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for OUT transfers that Total Bytes To Transfer be an even multiple of QHD.Maximum Packet Length. If software builds such a transfer descriptor for an OUT transfer, the last transaction is always less than QHD.Maximum Packet Length.

Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16 K(4000H).</td></tr>
<tr><td>15
Interrupt On Complete (IOC)</td><td>If this bit is set to a one, it specifies that when this qTD is completed, the Host Controller should issue an interrupt at the next interrupt threshold.</td></tr>
<tr><td>14-12
Current Page (C_Page)</td><td>This field is used as an index into the qTD buffer pointer list. Valid values are in the range 0H to 4H. The host controller is not required to write this field back when the qTD is retired.</td></tr>
<tr><td>11-10
Error Counter (CERR)</td><td>This field is a 2-bit down counter that keeps track of the number of consecutive Errors detected while executing this qTD. If this field is programmed with a non-zero value during set-up, the Host Controller decrements the count and writes it back to the qTD if the transaction fails. If the counter counts from one to zero, the Host Controller marks the qTD inactive, sets the Halted bit to a one, and error status bit for the error that caused CERR to decrement to zero. An interrupt is generated if the USB Error Interrupt Enable bit in the USBINTR register is set to a one. If HCD programs this field to zero during set-up, the Host Controller does not count errors for this qTD and there is no limit on the retries of this qTD. Note that write-backs of intermediate execution state are to the queue head overlay area, not the qTD.

Transaction Error - Decrement

Data Buffer Error - No Decrement[3]

Stalled - No Decrement[1]

Babble Detected - No Decrement[1]

No Error - No Decrement[2]
<table>
<tr><td>Error</td><td>Decrement Counter</td></tr>
<tr><td>1</td><td>Detection of Babble or Stall automatically halts the queue head. Thus, count is not decremented</td></tr>
<tr><td>2</td><td>If the EPS field indicates a HS device or the queue head is in the Asynchronous Schedule (and PIDCode indicates an IN or OUT) and a bus transaction completes and the host controller does not detect a transaction error, then the host controller should reset CERR to extend the total number of errors for this transaction. For example, CERR should be reset with maximum value (3) on each successful completion of a transaction. The host controller must never reset this field if the value at the start of the transaction is 00b.</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

Table 11-76. TD Token (DWord 2) (continued)

Bit	Description	
		See Split Transaction Interrupt for CERR adjustment rules when the EPS field indicates a FS or LS device and the queue head is in the Periodic Schedule. See Asynchronous - Do Complete Split for CERR adjustment rules when the EPS field indicates a FS or LS device, the queue head is in the Asynchronous schedule and the *PIDCode* indicates a SETUP.
	3	Data buffer errors are host problems. They don't count against the device's retries.
	NOTE: Software must not program CERR to a value of zero when the EPS field is programmed with a value indicating a Full- or Low-speed device. This combination could result in undefined behavior.	
9-8 PID Code	This field is an encoding of the token, which should be used for transactions associated with this transfer descriptor. Encodings are:	
	00b	OUT Token generates token (E1H)
	01b	IN Token generates token (69H)
	10b	SETUP Token generates token (2DH) (undefined if endpoint is an interrupt, the queue head is non-zero) transfer type, for example, µ*Frame S-mask* field in.
	11b	Reserved
7-0 Status	This field is used by the Host Controller to communicate individual command execution states back to HCD. This field contains the status of the last transaction performed on this qTD. The bit encodings are:	
	Bit	Status Field Description
	7	Active. Set to one by software to enable the execution of transactions by the Host Controller.
	6	Halted. Set to one by the Host Controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this qTD. This can be caused by babble, the error counter counting down to zero, or reception of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set to a one, the Active bit is also set to zero.
	5	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, the Host Controller forces a timeout condition on the USB, invalidating the transaction at the source. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	4	Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction. In addition to setting this bit, the Host Controller also sets the *Halted* bit to a one. Because "babble" is considered a fatal error for the transfer, setting the Halted bit to a one insures that no more transactions occur because of this descriptor.
	3	Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.

Table continues on the next page...

Table 11-76. TD Token (DWord 2) (continued)

Bit	Description	
	2	Missed Micro-Frame. This bit is ignored unless the *QH.EPS* field indicates a full- or low-speed endpoint and the queue head is in the periodic list. This bit is set when the host controller detected that a host-induced hold-off caused the host controller to miss a required complete-split transaction. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	1	Split Transaction State (SplitXstate). This bit is ignored by the host controller unless the *QH.EPS* field indicates a full- or low-speed endpoint. When a Full- or Low-speed device, the host controller uses this bit to track the state of the split-transaction. The functional requirements of the host controller for managing this state bit and the split transaction protocol depends on whether the endpoint is in the periodic or asynchronous schedule. The bit encodings are: Value Meaning 0b Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint. 1b Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint.
	0	Ping State (P)/ERR. If the *QH.EPS* field indicates a High-speed device and the *PID_Code* indicates an OUT endpoint, then this is the state bit for the Ping protocol. The bit encodings are: Value Meaning 0b Do OUT. This value directs the host controller to issue an OUT PID to the endpoint. 1b Do Ping. This value directs the host controller to issue a PING PID to the endpoint. If the *QH.EPS* field does not indicate a High-speed device, then this field is used as an error indicator bit. It is set to a one by the host controller whenever a periodic split-transaction receives an ERR handshake.

11.3.4.2.5.4 qTD Buffer Page Pointer List

The last five DWords of a queue element transfer descriptor is an array of physical memory address pointers. These pointers reference the individual pages of a data buffer.

System software initializes Current Offset field to the starting offset into the current page, where current page is selected through the value in the C_Page field.

The following table describes the qTD Buffer Pointer(s) (DWords 3-7) fields.

Table 11-77. qTD Buffer Pointer(s) (DWords 3-7)

Bit	Description
31-12	Buffer Pointer List. Each element in the list is a 4 K page aligned physical memory address. The lower 12 bits in each pointer are reserved (except for the first one), as each memory pointer must reference the start of a 4 K page. The field C_Page specifies the current active pointer. When the transfer element descriptor is fetched, the starting buffer address is selected using C_Page (similar to an array index to

Table continues on the next page...

Table 11-77. qTD Buffer Pointer(s) (DWords 3-7) (continued)

	select an array element). If a transaction spans a 4K buffer boundary, the host controller must detect the page-span boundary in the data stream, increment C_Page and advance to the next buffer pointer in the list, and conclude the transaction through the new buffer pointer.
11-0	Current Offset (Reserved). This field is reserved in all pointers except the first one (for example Page 0). The host controller should ignore all reserved bits. For the page 0 current offset interpretation, this field is the byte offset into the current page (as selected by C_Page). The host controller is not required to write this field back when the qTD is retired. Software should ensure the Reserved fields are initialized to zero.

11.3.4.2.6 Queue Head

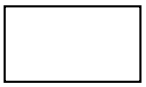
The following table shows the queue head structure layout.

Table 11-78. Queue Head Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Queue Head Horizontal Link Pointer																											0		Typ		T	03-00
RL				C	Maximum Packet Length											H	dtc	EP		EndPt				I	Device Address							07-04[1]
Mult		Port Number[2]							Hub Addr[2]							μFrame C-mask[2]								μFrame S-mask								0B-08[1]
Current qTD Pointer																											0					0F-0C
Next qTD Pointer																											0				T	13-10[3]
Alternate Next qTD pointer																											NakCnt				T	17-14[4]
dt	Total Bytes to Transfer															ioc	C_Page			Cerr		PID Code		Status								1B-18
Buffer Pointer (Page 0)																				Current Offset												1F-1C
Buffer Pointer (Page 1)																				Reserved				C-prog-mask[2]								23-20
Buffer Pointer (Page 2)																				S-bytes[2]										FrameTag[2]		27-24[4]
Buffer Pointer (Page 3)																				Reserved												2B-28
Buffer Pointer (Page 4)																				Reserved												2F-2C[3]

1. 04-0B: Static endpoint state.
2. These fields are used exclusively to support split transactions to USB 2.0 hubs
3. 10-2F: Transfer overlay.
4. 14-27: Transfer results.


Host Controller Read/Write


Host Controller Read Only

11.3.4.2.6.1 Queue Head Horizontal Link Pointer

The first DWord of a Queue Head contains a link pointer to the next data object to be processed after any required processing in this queue has been completed, as well as the control bits defined below.

This pointer may reference a queue head or one of the isochronous transfer descriptors. It must not reference a queue element transfer descriptor.

The following table describes the Queue head DWord 0 fields.

Table 11-79. Queue Head DWord 0

Bit	Description
31-5	Queue Head Horizontal Link Pointer (QHLP). This field contains the address of the next data object to be processed in the horizontal list and corresponds to memory address signals [31:5], respectively.
4-3	Reserved
2-1	QH/(s)iTD Select (Typ). This field indicates to the hardware whether the item referenced by the link pointer is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor) 11b FSTN (frame span traversal node)
0	Terminate (T). 1=Last QH (pointer is invalid). 0=Pointer is valid. If the queue head is in the context of the periodic list, a one bit in this field indicates to the host controller that this is the end of the periodic list. This bit is ignored by the host controller when the queue head is in the Asynchronous schedule. Software must ensure that queue heads reachable by the host controller always have valid horizontal link pointers.

11.3.4.2.6.2 Queue Head Endpoint Capabilities/Characteristics

The second and third DWords of a Queue Head specifies static information about the endpoint. This information does not change over the lifetime of the endpoint.

There are three types of information in this region:

- Endpoint Characteristics. These are the USB endpoint characteristics including addressing, maximum packet size, and endpoint speed.

- Endpoint Capabilities. These are adjustable parameters of the endpoint. They effect how the endpoint data stream is managed by the host controller.
- Split Transaction Characteristics. This data structure is used to manage full- and low-speed data streams for bulk, control, and interrupt via split transactions to USB2.0 Hub Transaction Translator. There are additional fields used for addressing the hub and scheduling the protocol transactions (for periodic).

The host controller must not modify the bits in this region.

The following table describes the Endpoint characteristics: Queue head DWord 1 fields.

Table 11-80. Endpoint Characteristics: Queue Head DWord 1

Bit	Description	
31-28	Nak Count Reload (RL). This field contains a value, which is used by the host controller to reload the Nak Counter field.	
27	Control Endpoint Flag (C). If the *QH.EPS* field indicates the endpoint is not a high-speed device, and the endpoint is a control endpoint, then software must set this bit to a one. Otherwise, it should always set this bit to zero.	
26-16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (*wMaxPacketSize*). The maximum value this field may contain is 0x400 (1024).	
15	Head of Reclamation List Flag (H). This bit is set by System Software to mark a queue head as being the head of the reclamation list.	
14	Data Toggle Control (DTC). This bit specifies where the host controller should get the initial data toggle on an overlay transition. 0b Ignore DT bit from incoming qTD. Host controller preserves DT bit in the queue head. 1b Initial data toggle comes from incoming qTD DT bit. Host controller replaces DT bit in the queue head from the DT bit in the qTD.	
13-12	Endpoint Speed (EPS). This is the speed of the associated endpoint. Bit combinations are:	
	Value	Meaning
	00b	Full-Speed (12 Mbits/sec)
	01b	Low-Speed (1.5 Mbits/sec)
	10b	High-Speed (480 Mbits/sec)
	11b	Reserved
	This field must not be modified by the host controller.	
11-8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.	
7	Inactivate on Next Transaction (I). This bit is used by system software to request that the host controller set the Active bit to zero. See Rebalancing the periodic schedule , for full operational details. This field is only valid when the queue head is in the Periodic Schedule and the *EPS* field indicates a Full or Low-speed endpoint. Setting this bit to one when the queue head is in the Asynchronous Schedule or the *EPS* field indicates a high-speed device yields undefined results.	
6-0	Device Address. This field selects the specific device serving as the data source or sink.	

The table below describes the Endpoint capabilities: Queue head DWord 2 field descriptions.

Table 11-81. Endpoint Capabilities: Queue Head DWord 2

Bit	Description
31-30	High-Bandwidth Pipe Multiplier (Mult). This field is a multiplier used to key the host controller as the number of successive packets the host controller may submit to the endpoint in the current execution. The host controller makes the simplifying assumption that software properly initializes this field (regardless of location of queue head in the schedules or other run time parameters). The valid values are: Value Meaning 00b Reserved. A zero in this field yields undefined results. 01b One transaction to be issued for this endpoint per micro-frame. 10b Two transactions to be issued for this endpoint per micro-frame. 11b Three transactions to be issued for this endpoint per micro-frame.
29-23	Port Number. This field is ignored by the host controller unless the *EPS* field indicates a full- or low-speed device. The value is the port number identifier on the USB 2.0 Hub (for hub at device address *Hub Addr* below), below which the full- or low-speed device associated with this endpoint is attached. This information is used in the split-transaction protocol.
22-16	Hub Addr. This field is ignored by the host controller unless the *EPS* field indicates a full-or low-speed device. The value is the USB device address of the USB 2.0 Hub below which the full- or low-speed device associated with this endpoint is attached. This field is used in the split-transaction protocol.
15-8	Split Completion Mask (µFrame C-Mask). This field is ignored by the host controller unless the *EPS* field indicates this device is a low- or full-speed device and this queue head is in the periodic list. This field (along with the *Active* and *SplitX-state* fields) is used to determine during which micro-frames the host controller should execute a complete-split transaction. When the criteria for using this field are met, a zero value in this field has undefined behavior. This field is used by the host controller to match against the three low-order bits of the FRINDEX register. If the FRINDEX register bits decode to a position where the µ*Frame C- Mask* field is a one, then this queue head is a candidate for transaction execution. There may be more than one bit in this mask set.
7-0	Interrupt Schedule Mask (µFrame S-mask). This field is used for all endpoint speeds. Software should set this field to a zero when the queue head is on the asynchronous schedule. A non-zero value in this field indicates an interrupt endpoint. The host controller uses the value of the three low-order bits of the FRINDEX register as an index into a bit position in this bit vector. If the µ*Frame S-mask* field has a one at the indexed bit position then this queue head is a candidate for transaction execution. If the *EPS* field indicates the endpoint is a high-speed endpoint, then the transaction executed is determined by the *PID_Code* field contained in the execution area. This field is also used to support split transaction types: Interrupt (IN/OUT). This condition is true when this field is non-zero and the *EPS* field indicates this is either a full- or low-speed device. A zero value in this field, in combination with existing in the periodic frame list has undefined results.

11.3.4.2.6.3 Transfer Overlay-Queue Head

The nine DWords in this area represent a transaction working space for the host controller. The general operational model is that the host controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it follows the Queue Head Horizontal Link Pointer to the next queue head. The host controller will never follow the Next Transfer Queue Element or Alternate Queue Element pointers unless it is actively attempting to advance the queue. For the duration of the transfer, the host controller keeps the incremental status of the transfer in the overlay area. When the transfer is complete, the results are written back to the original queue element.

The DWord3 of a Queue Head contains a pointer to the source qTD currently associated with the overlay. The host controller uses this pointer to write back the overlay area into the source qTD after the transfer is complete.

The following table describes the current qTD link pointer field descriptions.

Table 11-82. Current qTD Link Pointer

Bit	Description
31-5	Current Element Transaction Descriptor Link Pointer. This field contains the address Of the current transaction being processed in this queue and corresponds to memory address signals [31:5], respectively.
4-0	Reserved (R). These bits are ignored by the host controller when using the value as an address to write data. The actual value may vary depending on the usage.

The DWords 4-11 of a queue head are the transaction overlay area. This area has the same base structure as a Queue Element Transfer Descriptor. The queue head utilizes the reserved fields of the page pointers to implement tracking the state of split transactions.

This area is characterized as an overlay because when the queue is advanced to the next queue element, the source queue element is merged onto this area. This area serves as execution cache for the transfer.

The table below describes the Host-controller rules for bits in overlay.

Table 11-83. Host-Controller Rules for Bits in Overlay (DWords 5, 6, 8 and 9)

DWord	Bit	Description
5	4-1	Nak Counter (NakCnt)µRW. This field is a counter the host controller decrements whenever a transaction for the endpoint associated with this queue head results in a Nak or Nyet response. This counter is reloaded from *RL* before a transaction is executed during the first pass of the reclamation list (relative to an Asynchronous List Restart condition). It is also loaded from RL during an overlay.
6	31	Data Toggle. The *Data Toggle Control* controls whether the host controller preserves this bit when an overlay operation is performed.
6	15	Interrupt On Complete (IOC). The IOC control bit is always inherited from the source qTD when the overlay operation is performed.
6	11-10	Error Counter (C_ERR). This two-bit field is copied from the qTD during the overlay and written back during queue advancement.
6	0	Ping State (P)/ERR. If the *EPS* field indicates a high-speed endpoint, then this field should be preserved during the overlay operation.
8	7-0	Split-transaction Complete-split Progress (C-prog-mask). This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	4-0	Split-transaction Frame Tag (Frame Tag). This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	11-5	S-bytes. Software must ensure that the *S-bytes* field in a *qTD* is zero before activating the *qTD*. This field is used to keep track of the number of bytes sent or received during an IN or OUT split transaction.

11.3.4.2.7 Periodic Frame Span Traversal Node (FSTN)

This data structure is to be used only for managing Full- and Low-speed transactions that span a Host-frame boundary.

See Host Controller Operational Model for FSTNs for full operational details. Software must not use an FSTN in the Asynchronous Schedule. An FSTN in the Asynchronous schedule results in undefined behavior. Software must not use the FSTN feature with a host controller whose USB_HCIVERSION register indicates a revision implementation below 0096h. FSTNs are not defined for implementations before 0.96 and their use yields undefined results.

Table 11-84. Frame Span Traversal Node Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Normal Path Link Pointer																											0		Typ		T	03-00
Back Path Link Pointer																											0		Typ[1]		T	07-04

1. Must be set to indicate a queue head

Host Controller Read/Write

Host Controller Read Only

11.3.4.2.7.1 FSTN Normal Path Pointer

The first DWord of an FSTN contains a link pointer to the next schedule object. This object can be of any valid periodic schedule data type.

The following table describes the FSTN normal path pointer fields.

Table 11-85. FSTN Normal Path Pointer Field Descriptions

Bit	Description
31-5	Normal Path Link Pointer (NPLP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4-3	Reserved
2-1	QH/(s)iTD/FSTN Select (Typ). This field indicates to the Host Controller whether the item referenced is a iTD/siTD, a QH or an FSTN. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor)

Table continues on the next page...

Table 11-85. FSTN Normal Path Pointer Field Descriptions (continued)

	11b FSTN (Frame Span Traversal Node)
0	Terminate (T). 1 = Link Pointer field is not valid. 0 = Link Pointer is valid.

11.3.4.2.7.2 FSTN Back Path Link Pointer

The second DWord of an FTSN node contains a link pointer to a queue head.

If the T-bit in this pointer is zero, then this FSTN is a Save-Place indicator. Its Typ field must be set by software to indicate the target data structure is a queue head. If the T-bit in this pointer is set to one, then this FSTN is the Restore indicator. When the T-bit is one, the host controller ignores the Typ field.

The following table describes the FSTN back path link pointer fields.

Table 11-86. FSTN Back Path Link Pointer Field Descriptions

Bit	Description
31-5	Back Path Link Pointer (BPLP). This field contains the address of a Queue Head. This field corresponds to memory address signals [31:5], respectively.
4-3	Reserved
2-1	Typ. Software must ensure this field is set to indicate the target data structure is a Queue Head. Any other value in this field yields undefined results.
0	Terminate (T). 1=Link Pointer field is not valid (that is the host controller must not use bits [31:5] as a valid memory address). This value also indicates that this FSTN is a Restore indicator. 0=Link Pointer is valid (that is the host controller may use bits [31:5] (in combination with the CTRLDSSEGMENT register if applicable) as a valid memory address). This value also indicates that this FSTN is a Save-Place indicator.

11.3.4.3 Host Operational Model

The general operational model is for the enhanced interface host controller hardware and enhanced interface host controller driver (generally referred to as system software).

Each significant operational feature of the EHCI host controller is discussed in a separate section. Each section presents the operational model requirements for the host controller hardware. Where appropriate, recommended system software operational models for features are also presented.

11.3.4.3.1 Host Controller Initialization

After initial power-on or HCReset (hardware or through HCReset bit in the USB_USBCMD register), all of the operational registers are at their default values. After a hardware reset, only the operational registers not contained in the Auxiliary power well are at their default values.

The following table describes the default values of operational registers.

.

Table 11-87. Default Values of Operational Register Space

Operational Register	Default Value (after Reset)
USB_USBCMD	00080000h (00080B00h, if *Asynchronous Schedule Park Capability is one*)
USB_USBSTS	00001000h
USB_USBINTR	00000000h
USB_FRINDEX	00000000h
USB_CTRLDSSEGMENT	00000000h
USB_PERIODICLISTBASE	Undefined
USB_ASYNCLISTADDR	Undefined
USB_CONFIGFLAG	00000000h
USB_PORTSC1	00002000h (w/*PPC* set to one); 00003000h (w/*PPC* set to zero)

To initialize the host controller, software should perform the following steps:

- Write the appropriate value to the USB_USBINTR register to enable the appropriate interrupts.
- Write the base address of the Periodic Frame List to the USB_PERIODICLIST BASE register. If no work items are in the periodic schedule, all elements of the Periodic Frame List should have their T-Bits set to one.
- Write the USB_USBCMD register to set the desired interrupt threshold, frame list size (if applicable) and turn the host controller ON through setting the Run/Stop bit.

At this point, the host controller is up and running and the port registers begin reporting device connects, and so on. System software can enumerate a port through the reset process (where the port is in the enabled state). At this point, the port is active with SOFs occurring down the enabled ports, but the schedules have not enabled. To communicate with devices through the asynchronous schedule, system software must write the USB_ASYNCLISTADDR register with the address of a control or bulk queue head. Software must then enable the asynchronous schedule by writing one to the Asynchronous Schedule Enable bit in the USB_USBCMD register. To communicate with devices through the periodic schedule, system software must enable the periodic schedule by writing one to the Periodic Schedule Enable bit in the USB_USBCMD register.

NOTE

The schedules can be turned on before the first port is reset (and enabled).

When the USB_USBCMD register is written, system software must ensure the appropriate bits are preserved, depending on the intended operation.

11.3.4.3.2 Port Routing and Control

The EHCI specification defines that a USB 2.0 Host controller is comprised of one high-speed host controller, which implements the EHCI programming interface and 0 to N USB 1.1 companion host controllers.

Companion host controllers (cHCs) may be implementations of either Universal or Open host controller specifications. This configuration is used to deliver the required full USB 2.0-defined port capability; for example, Low-, Full-, and High-speed capability for every port.

NOTE

The USB controllers on i.MX parts do not require nor support companion controllers to support Full and Low Speed device. Full and Low Speed devices are supported within the USB controller by emulating the functionality of a high-speed HUB. Therefore, no port routing is present in the controller. Please refer to Embedded Transaction Translator Function for detail!

The following figure illustrates a simple block diagram of the port routing logic and its relationship to the high-speed and companion host controllers within a USB 2.0 host controller.

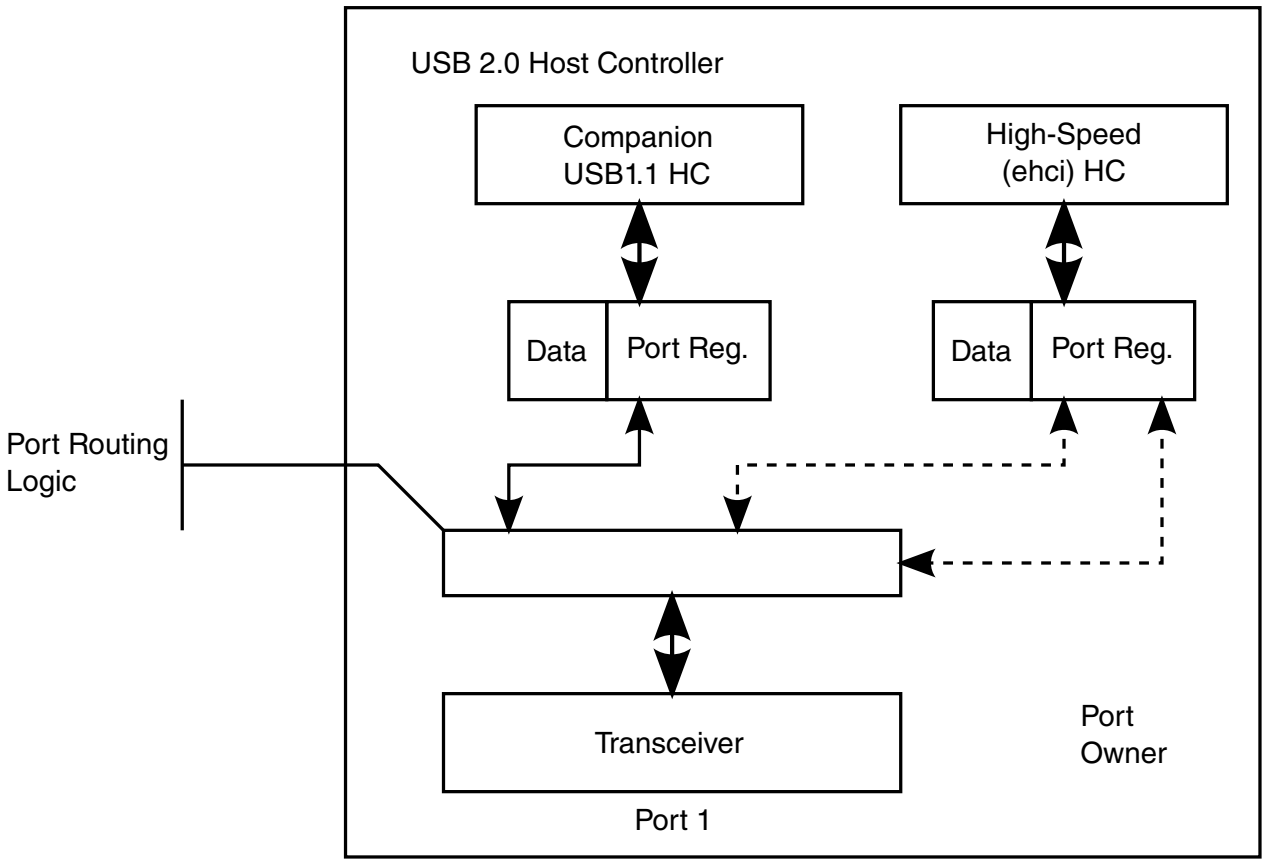


Figure 11-57. Example USB 2.0 Host Controller Port Routing Block Diagram

There exists one transceiver per physical port and each host controller block has its own port status and control registers. The EHCI controller has port status and control registers for every port. Each companion host controller has only the port control and status registers it is required to operate. Either the EHCI host controller or one companion host controller controls each transceiver. Routing logic lies between the transceiver, the port status and control registers.[1]

The port routing logic is controlled from signals originating in the EHCI host controller. The EHCI host controller has a global routing policy control field and per-port ownership control fields. The Configured Flag (CF) bit is the global routing policy control. At power-on or reset, the default routing policy is to the companion controllers (if they exist). If the system does not include a driver for the EHCI host controller and the host controller includes Companion Controllers, then the ports still work in Full- and Low-speed mode (assuming the system includes a driver for the companion controllers). In general, when the EHCI owns the ports, the companion host controllers' port registers do not see a connect indication from the transceiver. Similarly, when a companion host controller owns a port, the EHCI controller's port registers do not see a connect indication

1. The routing logic should not be implemented in the 480 MHz clock domain of the transceiver.

from the transceiver. The details on the rules for the port routing logic are described in the following sections. The USB 2.0 host controller must be implemented as a multi-function PCI device if the implementation includes companion controllers. The companion host controllers' function numbers must be less than the EHCI host controller function number. The EHCI host controller must be a larger function number with respect to the companion host controllers associated with this EHCI host controller. If a PCI device implementation contains only an EHCI controller (that is no companion controllers or other PCI functions), then the EHCI host controller must be function zero, in accordance with the PCI Specification. The N_CC field in the Structural Parameter register (HCSPARAMS) indicates whether the controller implementation includes companion host controllers. When N_CC has a non-zero value there exists companion host controllers. If N_CC has a value of zero, then the host controller implementation does not include companion host controllers. If the host controller root ports are exposed to attachment of full- or low-speed devices, the ports always fails the high-speed chirp during reset and the ports are not enabled. System software can notify the user of the illegal condition. This type of implementation requires a USB 2.0 hub be connected to a root port to provide full and low-speed device connectivity.

System software uses information in the host controller capability registers to determine how the ports are routed to the companion host controllers. See .

11.3.4.3.2.1 Port Routing Control through EHCI Configured (CF) Bit

Each port in the USB 2.0 host controller are routed either to a single companion host controller or to the EHCI host controller.

The port routing logic is controlled by two mechanisms in the EHCI HC: a host controller global flag and per-port control. The Configured Flag (CF) bit, is used to globally set the policy of the routing logic. Each port register has a Port Owner control bit which allows the EHCI Driver to explicitly control the routing of individual ports. Whenever the CF bit transitions from zero to one (this transition is only available under program control) the port routing unconditionally routes all of the port registers to the EHCI HC (all Port Owner bits go to zero). While the CF-bit is one, the EHCI Driver controls individual ports' routing through the Port Owner control bit. Likewise, whenever the CF bit transitions from one to zero (as a result of Aux power application, HCRESET, or software writing zero to CF-bit), the port routing unconditionally routes all of the port registers to the appropriate companion HC. The default value for the EHCI HC's CF bit (after Aux power application or HCRESET) is zero.

The *view* of the port depends on the current owner. A Universal or Open companion host controller will see port register bits consistent with the appropriate specification. Port bit definitions that are required for EHCI host controllers are not visible to companion host controllers.

The following table summarizes the default routing for all the ports, based on the value of the EHCI HC's CF bit.

Table 11-88. Default Port Routing Depending on EHCI HC CF Bit

HS CF Bit	Default Port Ownership	Explanation
0B	Companion HCs	The companion host controllers own the ports and only Full- and Low-speed devices are supported in the system. The exact port assignments are implementation dependent. The ports behave only as Full- and Low-speed ports in this configuration
1B	EHCI HC	The EHCI host controller has default ownership over all of the ports. The routing logic inhibits device connect events from reaching the companion HCs' port status and control registers when the port owner is the EHCI HC.The EHCI HC has access to the additional port status and control bits defined in this specification (see). The EHCI HC can temporarily release control of the port to a companion HC by setting the *PortOwner* bit in the PORTSC1 register to one.

11.3.4.3.2.2 Port Routing Control through PortOwner and Disconnect Event

Manipulating the port routing through the CF-bit is an extreme process and not intended to be used during normal operation.

The normal mode of port ownership transferal is on the granularity of individual ports using the Port Owner bit in the EHCI HC's USB_PORTSC1 register (for hand-offs from EHCI to companion host controllers). Individual port ownership is returned to the EHCI controller when the port registers a device disconnect. When the disconnect is detected, the port routing logic immediately returns the port ownership to the EHCI controller. The companion host controller port register detects the device disconnect and operates normally.

Under normal operating conditions (assuming all HC drivers loaded and operational and the EHCI *CF-bit* is set to one), the typical port enumeration sequence proceeds as illustrated below:

- Initial condition is that EHCI is port owner. A device is connected causing the port to detect a connect, set the port connect change bit and issue a port-change interrupt (if enabled).
- EHCI Driver identifies the port with the new connect change bit asserted and sends a change report to the hub driver. Hub driver issues a GetPortStatus() request and

identifies the connect change. It then issues a request to clear the connect change, followed by a request to reset and enable the port.

- When the EHCI Driver receives the request to reset and enable the port, it first checks the value reported by the LineStatus bits in the USB_PORTSC1 register. If they indicate the attached device is a full-speed device (for example, D+ is asserted), then the EHCI Driver sets the PortReset control bit to one (and sets the PortEnable bit to zero) which begins the reset-process. Software times the duration of the reset, then terminates reset signaling by writing zero to the port reset bit. The reset process is actually complete when software reads zero in the PortReset bit. The EHCI Driver checks the PortOwner bit in the USB_PORTSC1 register. If set to one, the connected device is a high-speed device and EHCI Driver (root hub emulator) issues a change report to the hub driver and the hub driver continues to enumerate the attached device.
- At the time the EHCI Driver receives the port reset and enable request the LineStatus bits might indicate a low-speed device. Additionally, when the port reset process is complete, the PortEnable field may indicate that a full-speed device is attached. In either case the EHCI driver sets the PortOwner bit in the USB_PORTSC1 register to one to release port ownership to a companion host controller.
- When the EHCI Driver sets PortOwner bit to one, the port routing logic makes the connection state of the transceiver available to the companion host controller port register and removes the connection state from the EHCI HC port. The EHCI USB_PORTSC1 register observes and reports a disconnect event through the disconnect change bit. The EHCI Driver detects the connection status change (either by polling or by port change interrupt) and then sends a change report to the hub driver. When the hub driver requests that port-state, the EHCI Driver responds with a reset complete change set to one, a connect change set to one and a connect status set to zero. This information is derived directly from the EHCI port register. This allows the hub driver to assume the device was disconnected during reset. It acknowledges the change bits and wait for the next change event. While the EHCI controller does not own the port, it simply remains in a state where the port reports no device connected. The device-connect evaluation circuitry of the companion HC activates and detects the device, the companion Driver detects the connection and enumerates the port.

When a port is routed to a companion HC, it remains under the control of the companion HC until the device is disconnected from the root port (ignoring for now the scenario where EHCI's CF-bit transitions from1b to 0b). When a disconnect occurs, the disconnect event is detected by both the companion HC port control and the EHCI port ownership control. On the event, the port ownership is returned immediately to the EHCI controller. The companion HC stack detects the disconnect and acknowledges as it would in an ordinary standalone implementation. Subsequent connects is detected by the EHCI port register and the process repeats.

11.3.4.3.2.3 Example Port Routing State Machine

The following figure illustrates an example of how the port ownership should be managed. The following sections describe the entry conditions to each state.

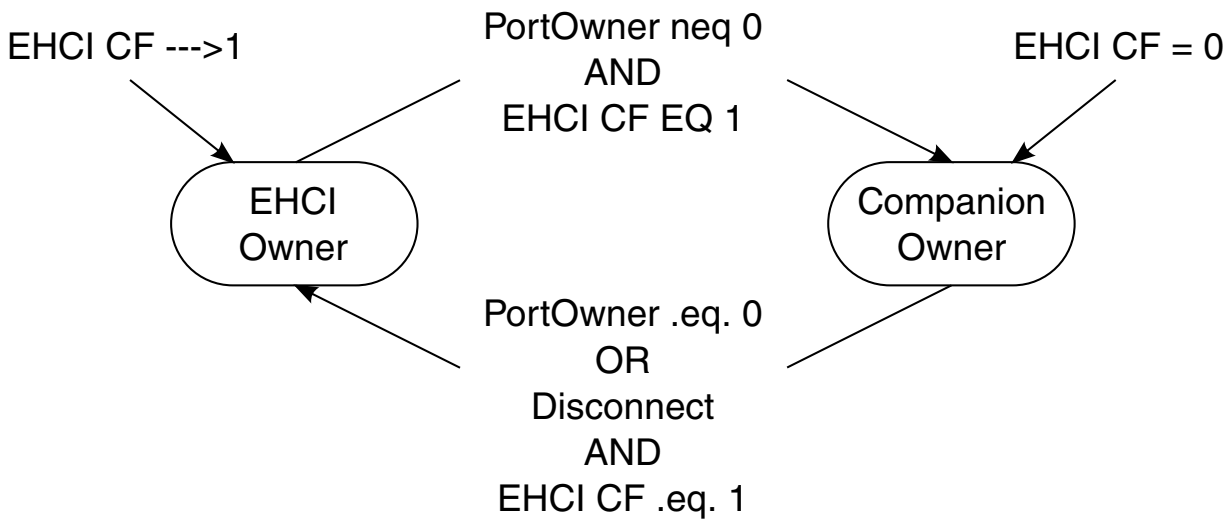


Figure 11-58. Port Owner Handoff State Machine

11.3.4.3.2.3.1 EHCI HC Owner

Entry to this state occurs when one of the following events occur:

- When the EHCI HC's Configure Flag (CF) bit in the USB_CONFIGFLAG register transitions from zero to one. This signals the fact that the system has a host controller driver for the EHCI HC and that all ports in the USB 2.0 host controller must default route to the EHCI controller.
- When the port is owned by a companion HC and the device is disconnected from the port. The EHCI port routing control logic is notified of the disconnect, and returns port routing to the EHCI controller. The connection state of the companion HC goes immediately to the disconnected state (with appropriate side effect to connect change, enable and enable change). The companion HC driver acknowledges the disconnect by setting the connect status change bit to zero. This allows the companion HC's driver to interact with the port completely through the disconnect process.
- When system software writes zero to the PortOwner bit in the USB_PORTSC1 register. This allows software to take ownership of a port from a companion host controller. When this occurs, the routing logic to the companion HC effectively signals a disconnect to the companion HC's port status and control register.

11.3.4.3.2.3.2 Companion HC Owner

Entry to this state occurs whenever one of the following events occur:

- When the PortOwner field transitions from zero to one.
- When the HS-mode HC's Configure Flag (CF) is equal to zero.

On entry to this state, the routing logic allows the companion HC port register to detect a device connect. Normal port enumeration proceeds.

11.3.4.3.2.4 Port Power

The Port Power Control (PPC) bit in the USB_HCSPARAMS register indicates whether the USB 2.0 host controller has port power control (see).

When this bit is zero, then the host controller does not support software control of port power switches. When in this configuration, the port power is always available and the companion host controllers must implement functionality consistent with port power always on. When the *PPC* bit is one, then the host controller implementation includes port power switches. Each available switch has an output enable, which is referred to in this discussion as PortPowerOutputEnable (PPE). PPE is controlled based on the state of the combination bits PPC bit, EHCI Configured (CF)-bit and individual Port Power (PP) bits.

The following table describes the summary behavioral model.

Table 11-89. Port Power Enable Control Rules

CF	CHC[1] (PP)	EHC[2] (PP)	Owner	PPE[3]	Description
0	0	X	CHC	0	When the EHCI controller is not configured, the port is owned by the companion host controller. When the companion HC's port power select is off, then the port power is off.
0	1	X	CHC	1	Similar to previous entry. When the companion HC's port power select is on, then the port power is on.
1	0	0	CHC	0	Port owner has port power turned off, the power to port is off.
1	0	0	EHC	0	Port owner has port power turned off, the power to port is off.
1	0	1	EHC	1	Port owner has port power on, so power to port is on.
1	0	1	CHC	1	If either HC has port power turned on, the power to the port is on.

Table continues on the next page...

Table 11-89. Port Power Enable Control Rules (continued)

1	1	0	EHC	1	If either HC has port power turned on, the power to the port is on.
1	1	0	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	EHC	1	Port owner has port power on, so power to port is on.

1. CHC (Companion Host Controller).
2. EHC (EHCI Host Controller).
3. PPE (Port Power Enable). This bit actually turns on the port power switch (if one exists).

11.3.4.4.3.2.5 Port Reporting Over-Current

Host controllers are by definition power providers on USB. Whether the ports are considered high- or low-powered is a platform implementation issue. Each EHCI USB_PORTSC1 register has an over-current status and over-current change bit.

The functionality of these bits are specified in the USB Specification Revision 2.0.

The over current detection and limiting logic usually resides outside the host controller logic. This logic may be associated with one or more ports. When this logic detects an over-current condition it is made available to both the companion and EHCI ports. The effect of an over-current status on a companion host controller port is beyond the scope of this document.

The over-current condition effects the following bits in the USB_PORTSC1 register on the EHCI port:

- Over-current Active bits are set to one. When the over-current condition goes away, the Over-current Active bit transitions from one to zero.
- Over-current Change bits are set to one. On every transition of the Over-current Active bit the host controller sets the Over-current Change bit to one. Software sets the Over-current Change bit to zero by writing one to this bit.
- Port Enabled/Disabled bit is set to zero. When this change bit gets set to one, then the Port Change Detect bit in the USB_USBSTS register is set to one.
- Port Power (PP) bits may optionally be set to zero. There is no requirement in USB that a power provider shut off power in an over current condition. It is sufficient to limit the current and leave power applied. When the Over-current Change bit transitions from zero to one, the host controller also sets the Port Change Detect bit in the USB_USBSTS register to one. In addition, if the Port Change Interrupt Enable bit in the USB_USBINTR register is one, then the host controller issues an interrupt to the system. Refer to Table 11-90 for summary behavior for over-current detection

when the host controller is halted (suspended from a device component point of view).

11.3.4.4.3.3 Suspend/Resume-Host Operational Model

The EHCI host controller provides an equivalent suspend and resume model as that defined for individual ports in a USB 2.0 Hub.

Control mechanisms are provided to allow system software to suspend and resume individual ports. The mechanisms allow the individual ports to be resumed completely through software initiation. Other control mechanisms are provided to parameterize the host controller's response (or sensitivity) to external resume events. In this discussion, host-initiated, or software initiated resumes are called Resume Events/Actions. Bus-initiated resume events are called wake-up events. The classes of wake-up events are:

- Remote-wake-up enabled device asserts resume signaling. In similar kind to USB 2.0 Hubs, EHCI controllers must always respond to explicit device resume signaling and wake-up the system (if necessary).
- Port connect and disconnect and over-current events. Sensitivity to these events can be turned on or off by using the per-port control bits in the USB_PORTSC1 registers.

Selective suspend is a feature supported by every USB_PORTSC1 register. It is used to place specific ports into a suspend mode. This feature is used as a functional component for implementing the appropriate power management policy implemented in a particular operating system. When system software intends to suspend the entire bus, it should selectively suspend all enabled ports, then shut off the host controller by setting the Run/Stop bit in the USB_USBCMD register to zero. The EHCI sub-block can then be placed into a lower device state through the PCI power management interface (see Appendix A, Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com).

When a wake event occurs, the system resumes operation and system software eventually set the Run/Stop bit to one and resume the suspended ports. Software must not set the Run/Stop bit to one until it is confirmed that the clock to the host controller is stable. This is usually confirmed in a system implementation in that all of the clocks in the system are stable before the ARM platform is restarted. So, by definition, if software is running, clocks in the system are stable and the Run/Stop bit in the USB_USBCMD register can be set to one. Minimum system software delays are also defined in the PCI Power Management Specification. Refer to PCI Power Management Specification for more information.

11.3.4.4.3.3.1 Port Suspend/Resume

System software places individual ports into suspend mode by writing one into the appropriate USB_PORTSC1 Suspend bit. Software must only set the Suspend bit when the port is in the enabled state (Port Enabled bit is one) and the EHCI is the port owner (PortOwner bit is zero).

The host controller may evaluate the Suspend bit immediately or wait until a micro-frame or frame boundary occurs. If evaluated immediately, the port is not suspended until the current transaction (if one is executing) completes. Therefore, there may be several micro-frames of activity on the port until the host controller evaluates the Suspend bit. The host controller must evaluate the Suspend bit at least every frame boundary.

System software can initiate a resume on a selectively suspended port by writing one to the Force Port Resume bit. Software should not attempt to resume a port unless the port reports that it is in the suspended state (see). If system software sets Force Port Resume bit to one when the port is not in the suspended state, the resulting behavior is undefined. In order to assure proper USB device operation, software must wait for at least 10 ms after a port indicates that it is suspended (Suspend bit is one) before initiating a port resume through the Force Port Resume bit. When Force Port Resume bit is one, the host controller sends resume signaling down the port. System software times the duration of the resume (nominally 20 ms) then sets the Force Port Resume bit to zero. When the host controller receives the write to transition Force Port Resume to zero, it completes the resume sequence as defined in the USB specification, and sets both the Force Port Resume and Suspend bits to zero. Software-initiated port resumes do not affect the Port Change Detect bit in the USB_USBSTS register nor do they cause an interrupt if the Port Change Interrupt Enable bit in the USB_USBINTR register is one. An external USB event may also initiate a resume. The wake events are defined above. When a wake event occurs on a suspended port, the resume signaling is detected by the port and the resume is reflected downstream within 100 μsec. The port's Force Port Resume bit is set to one and the Port Change Detect bit in the USB_USBSTS register is set to one. If the Port Change Interrupt Enable bit in the USB_USBINTR register is one the host controller issues a hardware interrupt.

System software observes the resume event on the port, delays a port resume time (nominally 20 ms), then terminates the resume sequence by writing zero to the Force Port Resume bit in the port. The host controller receives the write of zero to Force Port Resume, terminates the resume sequence and sets Force Port Resume and Suspend port bits to zero. Software can determine that the port is enabled (not suspended) by sampling the USB_PORTSC1 register and observing that the Suspend and Force Port Resume bits are zero. Software must ensure that the host controller is running (that is HCHalted bit in the USB_USBSTS register is zero), before terminating a resume by writing zero to a

port's Force Port Resume bit. If HCHalted is one when Force Port Resume is set to zero, then SOFs do not occur down the enabled port and the device returns to suspend mode in a maximum of 10 msec.

The table below summarizes the wake-up events. Whenever a resume event is detected, the Port Change Detect bit in the USB_USBSTS register is set to one. If the Port Change Interrupt Enable bit is one in the USB_USBINTR register, the host controller generates an interrupt on the resume event. Software acknowledges the resume event interrupt by clearing the Port Change Detect status bit in the USB_USBSTS register.

Table 11-90. Behavior During Wake-up Events

Port Status and Signaling Type	Signaled Port Response	Device State	
		D0	Not D0
Port disabled, resume K-State received	No Effect	N/A	N/A
Port suspended, resume K-State received	Resume reflected downstream on signaled port. Force Port Resume status bit in USB_PORTSC1 register is set to one. Port Change Detect bit in USB_USBSTS register set to one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is one. A disconnect is detected.	Depending in the initial port state, theUSB_ PORTSC1 Connected Enable status bits are set to zero, and the Connect Change status bit is set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is zero. A disconnect is detected.	Depending on the initial port state, the USB_PORTSC1 Connect and Enable status bits are set to zero, and the Connect Change status bit is set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [3]	[3]
Port is not connected and the port's WKCNNT_E bit is one. A connect is detected.	USB_PORTSC1 Connect Status and Connect Status Change bits are set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [2]	[2]
Port is not connected and the port's WKCNNT_E bit is zero. A connect is detected.	USB_PORTSC1 Connect Status and Connect Status Change bits are set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [3]	[3]
Port is connected and the port's WKOC_E bit is one. An over-current condition occurs.	USB_PORTSC1 Over-current Active, Over-current Change bits are set to one. If Port Enable/Disable bit is one, it is set to zero. Port Change Detect bit in the USB_USBSTS register is set to one	[1], [2]	[2]
Port is connected and the port's WKOC_E bit is zero. An over-current condition occurs.	USB_PORTSC1 Over-current Active, Over-current Change bits are set to one. If Port Enable/Disable bit is one, it is set to zero. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [3]	[3]

[1] Hardware interrupt issued if Port Change Interrupt Enable bit in the USB_USBINTR register is one.

[2] PME# asserted if enabled (Note: PME Status must always be set to one).

[3] PME# not asserted.

11.3.4.3.4 Schedule Traversal Rules

The host controller executes transactions for devices using a simple, shared-memory schedule.

The schedule is comprised of a few data structures, organized into two distinct lists. The data structures are designed to provide the maximum flexibility required by USB, minimize memory traffic and hardware / software complexity.

System software maintains two schedules for the host controller: a periodic schedule and an asynchronous schedule. The root of the periodic schedule is the USB_PERIODICLISTBASE register (see)/ . The USB_PERIODICLISTBASE register is the physical memory base address of the periodic frame list. The periodic frame list is an array of physical memory pointers. The objects referenced from the frame list must be valid schedule data structures as defined in Host Data Structures. In each micro-frame, if the periodic schedule is enabled (see Periodic scheduling threshold) then the host controller must execute from the periodic schedule before executing from the asynchronous schedule. It only executes from the asynchronous schedule after it encounters the end of the periodic schedule. The host controller traverses the periodic schedule by constructing an array offset reference from the USB_PERIODICLISTBASE and the USB_FRINDEX registers (see the following figure). It fetches the element and begins traversing the graph of linked schedule data structures.

The end of the periodic schedule is identified by a next link pointer of a schedule data structure having its T-bit set to one. When the host controller encounters a T-Bit set to one during a horizontal traversal of the periodic list, it interprets this as an End-Of-Periodic-List mark. This causes the host controller to cease working on the periodic schedule and transitions immediately to traversing the asynchronous schedule. After the transition, the host controller executes from the asynchronous schedule until the end of the micro-frame.

The following figure illustrates the derivation of pointer into frame list array.

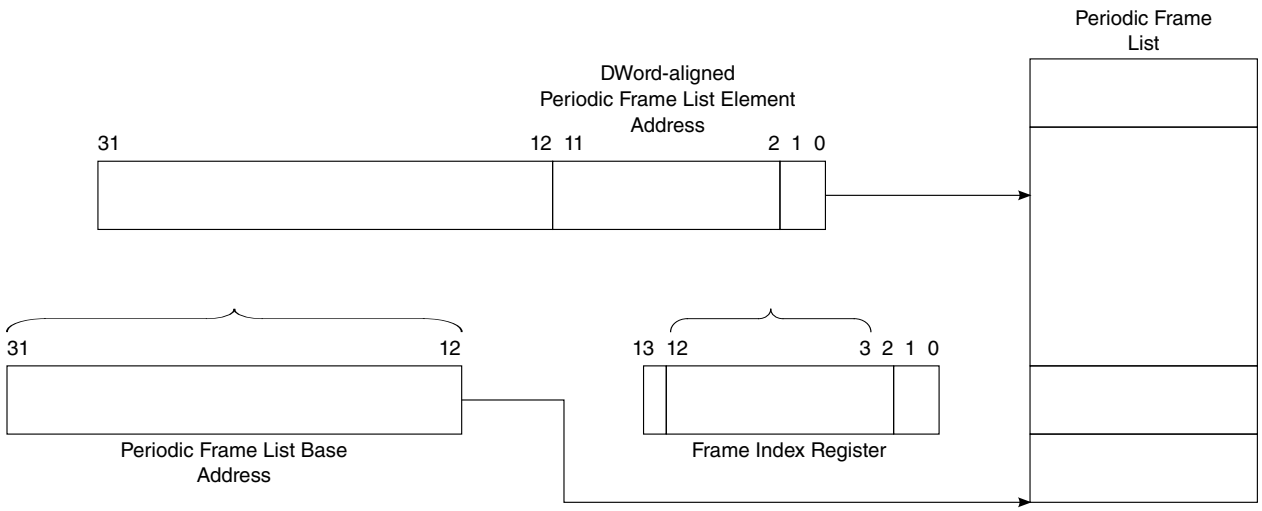


Figure 11-59. Derivation of Pointer into Frame List Array

When the host controller determines that it is the time to execute from the asynchronous list, it uses the operational register USB_ASYNCLISTADDR to access the asynchronous schedule, see the figure below.

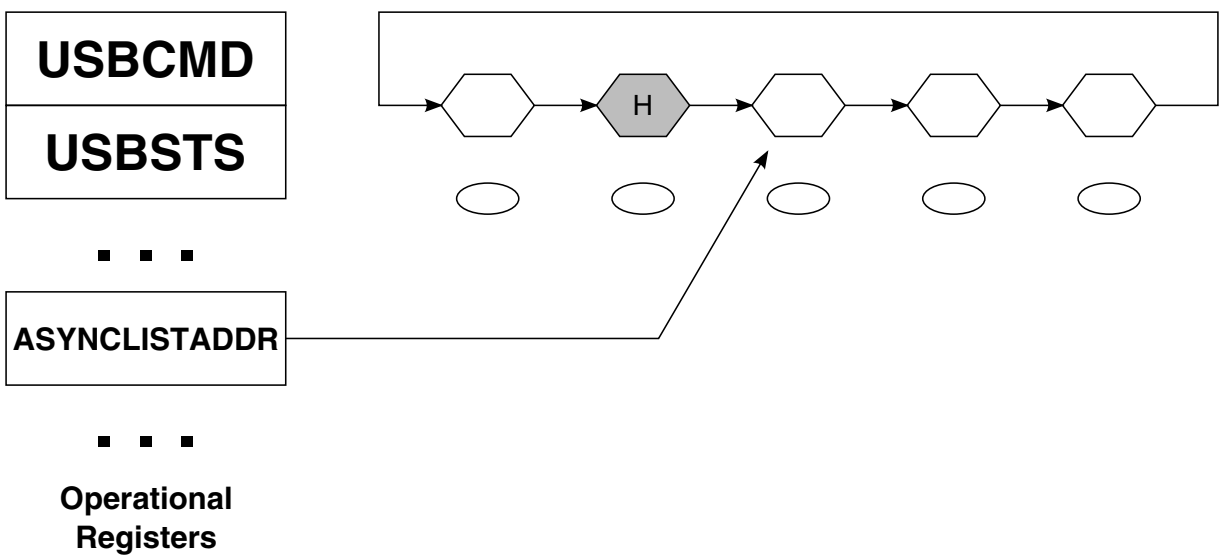


Figure 11-60. General Format of Asynchronous Schedule List

The USB_ASYNCLISTADDR register contains a physical memory pointer to the next queue head. When the host controller makes a transition to executing the asynchronous schedule, it begins by reading the queue head referenced by the USB_ASYNCLISTADDR register. Software must set queue head horizontal pointer T-bits to zero for queue heads in the asynchronous schedule. See Asynchronous Schedule for complete operational details.

11.3.4.3.4.1 Example - Preserving Micro-Frame Integrity

One of the requirements of a USB host controller is to maintain Frame Integrity. This means that the HC must preserve the micro-frame boundaries.

For example, SOF packets must be generated on time (within the specified allowable jitter), and High-speed EOF1,2 thresholds must be enforced. The end of micro-frame timing points EOF1 and EOF2 are clearly defined in the USB Specification Revision 2.0. One implication of this responsibility is that the HC must ensure that it does not start transactions that do not complete before the end of the micro-frame. More precisely, no transactions should be started by the host controller, which do not complete in their entirety before the EOF1 point. In order to enforce this rule, the host controller must check each transaction before it starts to ensure that it completes before the end of the micro-frame.

So, what exactly needs to be involved in this check? Fundamentally, the transaction data payload, plus bit stuffing, plus transaction overhead must be taken into consideration. It is possible to be extremely accurate on how much time the next transaction takes. Take OUTs for an example. The host controller must fetch all of the OUT data from memory in order to send it onto the USB bus. A host controller implementation could pre-fetch all of the OUT data, and pre-compute the actual number of bits in the token and data packets. In addition, the system knows the depth of the target endpoint, so it could closely estimate turnaround time for handshake. In addition, the host controller knows the size of a handshake packet. Pre-computing effects of bit stuffing and summing up the other overhead numbers can allow the host controller to know exactly whether there is enough bus time, before EOF1 to complete the OUT transaction. To accomplish this particular approach takes an inordinate amount of time and hardware complexity.

The alternative is to make a reasonable guess whether the next transaction can be started. An example approximation algorithm is described below. This example algorithm relies on the EHCI policy that periodic transactions are scheduled first in the micro-frame. It is a reasonable assumption that software never over-commits the micro-frame to periodic transactions greater than the specification allowable 80%. In the available remaining 20% bandwidth, the host controller has some ability (in this example) to decide whether or not to execute a transaction. The result of this algorithm is that sometimes, under some circumstances a transaction is not executed that could have been executed. However, under all circumstances, a transaction is never started unless there is enough time in the frame to complete the transaction.

11.3.4.3.4.1.1 Transaction Fit - A Best-Fit Approximation Algorithm

A curve is calculated which represents the latest start time for every packet size, at which software schedules the start of a periodic transaction.

This curve is the 80% bandwidth curve. Another curve is calculated which is the absolute, latest permitted start time for every packet size. This curve represents the absolute latest time, that a transaction of each packet size can be started and completed, in the micro-frame. A plot of these two curves are illustrated in Figure 11-61. The plot Y-axis represents the number of byte-times left in a frame.

The space between the 80% and the Last Start plots is bandwidth reclamation area. In this algorithm the host controller may skip transactions during this time if it is prudent.

The Best-Fit Approximation method plots a function ($f(x)$) between the 80% and Last Start curves. The function $f(x)$ adds a constant to every transaction's maximum packet size and the result compared with the number of bytes left in the frame. The constant represents an approximation of the effects of bit stuffing and protocol overhead. The host controller starts transactions whose results land above the function curve. The host controller will not start transactions whose results land below the function curve.

The following figure illustrates the Best-Fit Approximation.

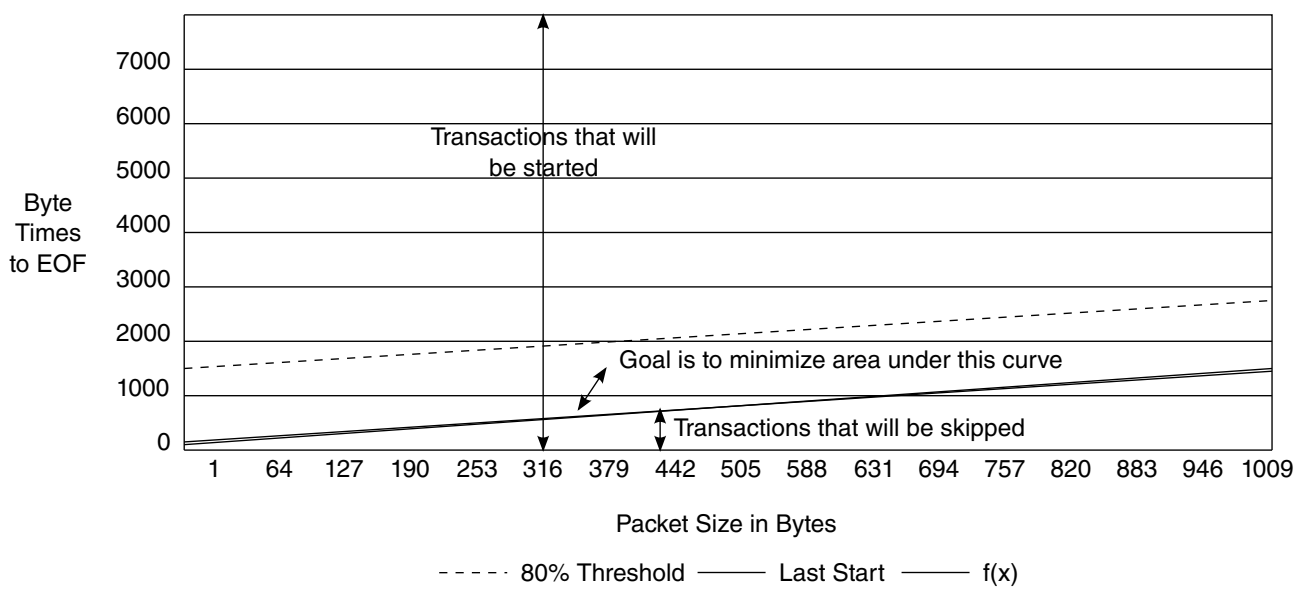


Figure 11-61. Best Fit Approximation

The LastStart line was calculated in this example to assume the absolute worst-case bus overhead per transaction. The particular transaction used is a start-split, zero-length OUT transaction with a handshake. Summaries of the component parts are listed in the table below. The component times were derived from the protocol timings defined in the USB Specification Revision 2.0.

Table 11-91. Example Worse-case Transaction Timing Components

Component	Bit time	Byte Time	Explanation
Split Token	76	9.5	Split token as defined in USB core specification. Includes sync, token, eop, and so on.
Host 2 Host IPG	88	11	Number of bit times required between consecutive host packets.
Token	67	8.375	Token as defined in USB core specification. Includes sync, token, eop, and so on.
Host 2 Host IPG	88	11	Token as defined in USB core specification. Includessync, token, eop, and so on.
Data Packet (0 data bytes)	66.7	8.34	Zero-length data packet. Includes sync, PID, crc16, eop, and so on.
Turnaround time	721	90.125	Time for packet initiator (Host) to see the beginning of a response to a transmitted packet.
Handshake packet	48	6	Handshake packet as defined in USB core specification. Includes sync, PID, eop, and so on.
		144	Total

The exact details of the function (f(x)) are up to the particular implementation. However, it should be obvious that the goal is to minimize the area under the curve between the approximation function and the Last Start curve, without dipping below the LastStart line, while at the same time keeping the check as simple as possible for hardware implementation. The f(x) in Figure 11-61 was constructed using the following pseudo-code test on each transaction size data point. This algorithm assumes that the host controller keeps track of the remaining bits in the frame.

```
Alorithm CheckTransactionWillFit (MaximumPacketSize, HC_BytesLeftInFrame)
Begin
Local Temp = MaximumPacketSize + 192
Local rvalue = TRUE
If MaximumPacketSize >= 128 then
     Temp += 128
End If
If Temp > HC_BytesLeftInFrame then
     Rvalue = FALSE
End If
Return rvalue
End
```

This algorithm takes two inputs, the current maximum packet size of the transaction and the hardware counter of the number of bytes left in the current micro-frame. It unconditionally adds a simple constant of 192 to the maximum packet size to account for a first-order effect of transaction overhead and bit stuffing. If the transaction size is greater than or equal to 128 bytes, then an additional constant of 128 is added to the

running sum to account for the additional worst-case bit stuffing of payloads larger than 128. An inflection point was inserted at 128 because the $f(x)$ plot was getting close to the LastStart line.

11.3.4.3.5 Periodic Schedule Frame Boundaries vs Bus Frame Boundaries

The USB Specification Revision 2.0 requires that the frame boundaries (SOF frame number changes) of the high-speed bus and the full- and low-speed bus(s) below USB 2.0 Hubs be strictly aligned.

Super-imposed on this requirement is that USB 2.0 Hubs manage full- and low-speed transactions through a micro-frame pipeline (see start- (SS) and complete- (CS) splits illustrated in the following figure). A simple, direct projection of the frame boundary model into the host controller interface schedule architecture creates tension (complexity for both hardware and software) between the frame boundaries and the scheduling mechanisms required to service the full- and low-speed transaction translator periodic pipelines.

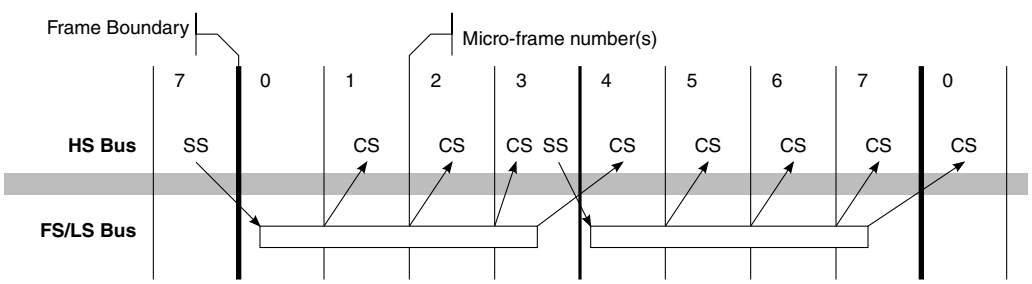


Figure 11-62. Frame Boundary Relationship between HS bus and FS/LS Bus

The simple projection, as the above figure illustrates, introduces frame-boundary wrap conditions for scheduling on both the beginning and end of a frame. In order to reduce the complexity for hardware and software, the host controller is required to implement one micro-frame phase shift for its view of frame boundaries. The phase shift eliminates the beginning of frame and frame-wrap scheduling boundary conditions.

The implementation of this phase shift requires that the host controller use one register value for accessing the periodic frame list and another value for the frame number value included in the SOF token. These two values are separate, but tightly coupled. The periodic frame list is accessed through the Frame List Index Register (USB_FRINDEX) documented in and initially illustrated in Schedule Traversal Rules . Bits FRINDEX[2:0], represent the micro-frame number. The SOF value is coupled to the value of FRINDEX[13:3]. Both FRINDEX[13:3] and the SOF value are increment based on FRINDEX[2:0]. It is required that the SOF value be delayed from the FRINDEX value by one micro-frame. The one micro-frame delay yields host controller periodic schedule

and bus frame boundary relationship as illustrated in the following figure. This adjustment allows software to trivially schedule the periodic start and complete-split transactions for full-and low-speed periodic endpoints, using the natural alignment of the periodic schedule interface. The reasons for selecting this phase-shift are beyond the scope of this specification.

The following figure illustrates how periodic schedule data structures relate to schedule frame boundaries and bus frame boundaries. To aid the presentation, two terms are defined: The host controller's view of the 1 msec boundaries is called H-Frames. The high-speed bus's view of the 1 msec boundaries is called B-Frames.

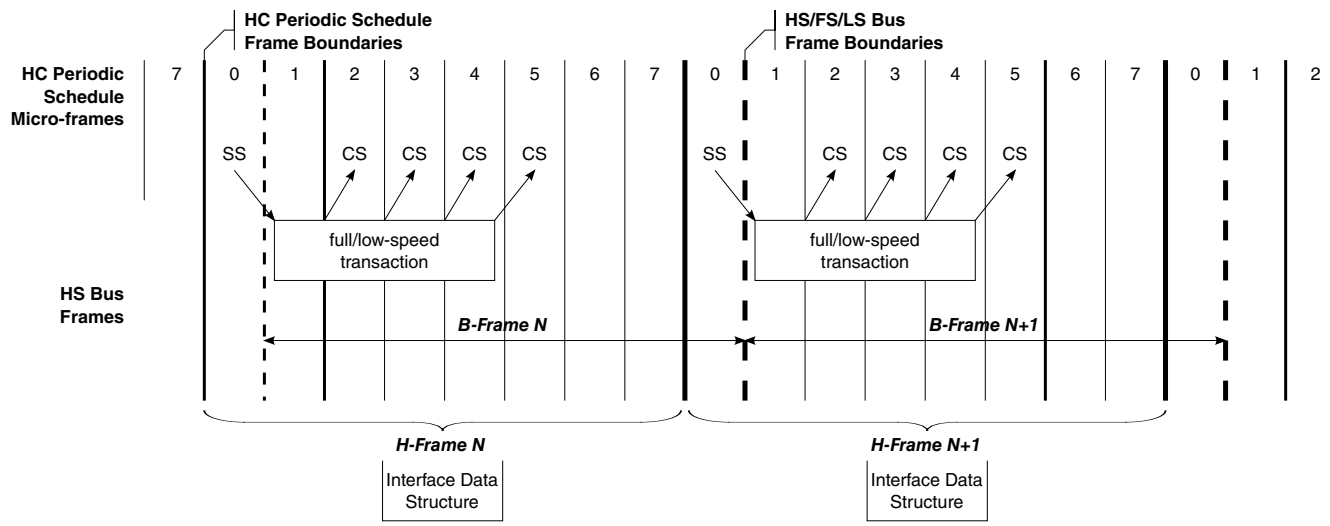


Figure 11-63. Relationship of Periodic Schedule Frame Boundaries to Bus Frame Boundaries

H-Frame boundaries for the host controller correspond to increments of FRINDEX[13:3]. Micro-frame numbers for the H-Frame are tracked by FRINDEX[2:0]. B-Frame boundaries are visible on the high-speed bus through changes in the SOF token's frame number. Micro-frame numbers on the high-speed bus are only derived from the SOF token's frame number (that is the high-speed bus sees eight SOFs with the same frame number value). H-Frames and B-Frames have the fixed relationship (that is B-Frames lag H-Frames by one micro-frame time) illustrated in the figure above. The host controller's periodic schedule is naturally aligned to H-Frames. Software schedules transactions for full- and low-speed periodic endpoints relative the H-Frames. The result is these transactions execute on the high-speed bus at exactly the right time for the USB 2.0 Hub periodic pipeline. As described in , the SOF Value can be implemented as a shadow register (in this example, called SOFV), which lags the FRINDEX register bits [13:3] by one micro-frame count. This lag behavior can be accomplished by incrementing FRINDEX[13:3] based on carry-out on the 7 to 0 increment of FRINDEX[2:0] and incrementing SOFV based on the transition of 0 to 1 of FRINDEX[2:0].

Software is allowed to write to FRINDEX. provides the requirements that software should adhere when writing a new value in FRINDEX.

The table below illustrates the required relationship between the value of FRINDEX and the value of SOFV.

Table 11-92. Operation of FRINDEX and SOFV (SOF Value Register)

Current			Next		
FRINDEX[F]	SOFV	FRINDEX[mF]	FRINDEX[F]	SOFV	FRINDEX[mF]
N	N	111b	N+1	N	000b
N+1	N	000b	N+1	N+1	001b
N+1	N+1	001b	N+1	N+1	010b
N+1	N+1	010b	N+1	N+1	011b
N+1	N+1	011b	N+1	N+1	100b
N+1	N+1	100b	N+1	N+1	101b
N+1	N+1	101b	N+1	N+1	110b
N+1	N+1	110b	N+1	N+1	111b

NOTE

Where [F] = [13:3]; [µF] = [2:0]

11.3.4.3.6 Periodic Schedule

The periodic schedule traversal is enabled or disabled through the Periodic Schedule Enable bit in the USB_USBCMD register.

If the Periodic Schedule Enable bit is set to zero, then the host controller simply does not try to access the periodic frame list through the USB_PERIODICLISTBASE register. Likewise, when the Periodic Schedule Enable bit is one, then the host controller does use the USB_PERIODICLISTBASE register to traverse the periodic schedule. The host controller will not react to modifications to the Periodic Schedule Enable immediately. In order to eliminate conflicts with split transactions, the host controller evaluates the Periodic Schedule Enable bit only when FRINDEX[2:0] is zero. System software must not disable the periodic schedule if the schedule contains an active split transaction work item that spans the 000b micro-frame. These work items must be removed from the schedule before the Periodic Schedule Enable bit is written to zero. The Periodic Schedule Status bit in the USB_USBSTS register indicates status of the periodic schedule. System software enables (or disables) the periodic schedule by writing one (or zero) to the Periodic Schedule Enable bit in the USB_USBCMD register. Software then can poll the Periodic Schedule Status bit to determine when the periodic schedule has

made the desired transition. Software must not modify the Periodic Schedule Enable bit unless the value of the Periodic Schedule Enable bit equals that of the Periodic Schedule Status bit.

The periodic schedule is used to manage all isochronous and interrupt transfer streams. The base of the periodic schedule is the periodic frame list. Software links schedule data structures to the periodic frame list to produce a graph of scheduled data structures. The graph represents an appropriate sequence of transactions on the

The following figure illustrates isochronous transfers (using iTDs and siTDs) with a period of one are linked directly to the periodic frame list. Interrupt transfers (are managed with queue heads) and isochronous streams with periods other than one are linked following the period-one iTD/siTDs. Interrupt queue heads are linked into the frame list ordered by poll rate. Longer poll rates are linked first (for example, closest to the periodic frame list), followed by shorter poll rates, with queue heads with a poll rate of one, on the very end.

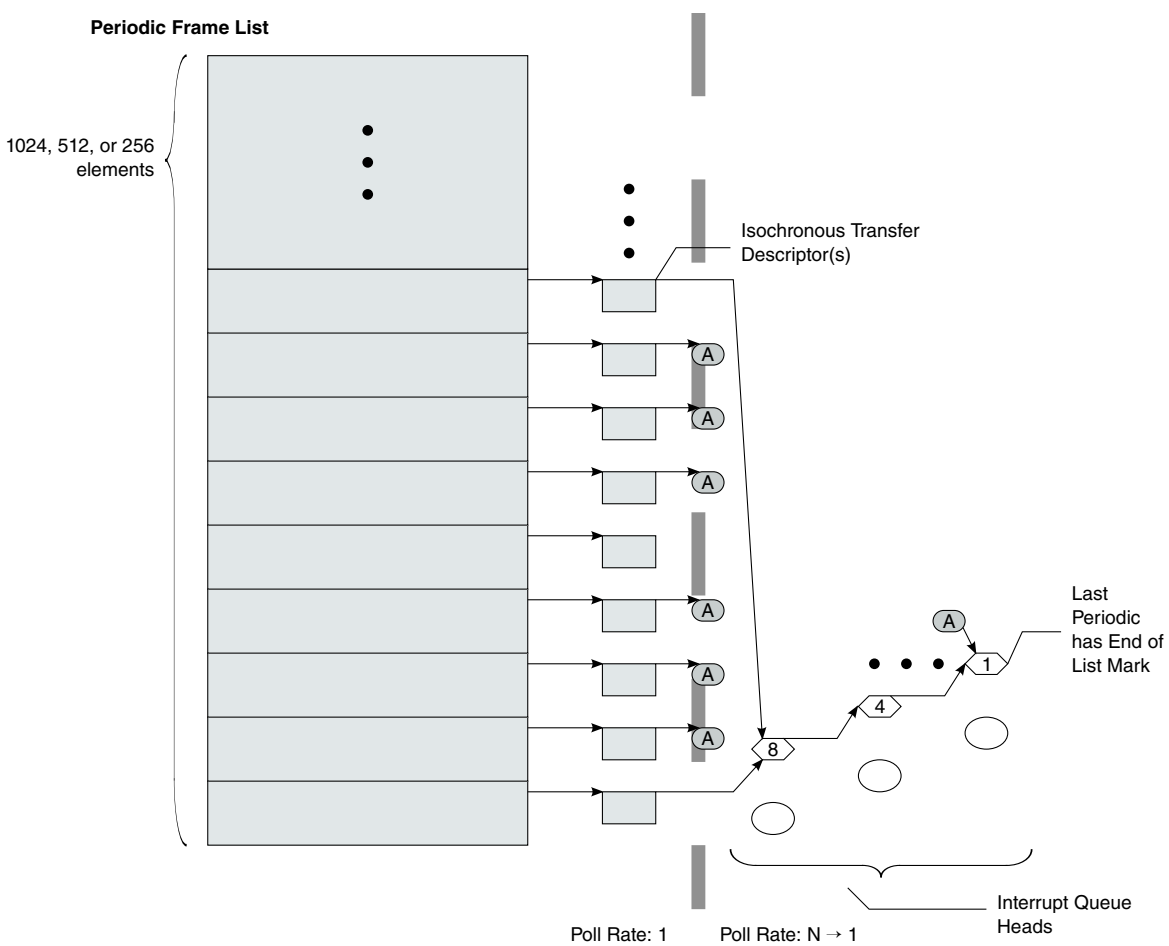


Figure 11-64. Example Periodic Schedule

11.3.4.3.7 Managing Isochronous Transfers Using iTDs

The structure of an iTD is presented in Isochronous (High-Speed) Transfer Descriptor (iTD). The four distinct sections to an iTD:

- The first field is the Next Link Pointer. This field is for schedule linkage purposes only.
- Transaction description array. This area is an eight-element array. Each element represents control and status information for one micro-frame's worth of transactions for a single high-speed isochronous endpoint.
- The buffer page pointer array is a 7-element array of physical memory pointers to data buffers. These are 4 K aligned pointers to physical memory.
- Endpoint capabilities. This area utilizes the unused low-order 12 bits of the buffer page pointer array. The fields in this area are used across all transactions executed for this iTD, including endpoint addressing, transfer direction, maximum packet size and high-bandwidth multiplier.

11.3.4.3.7.1 Host Controller Operational Model for iTDs

The host controller uses FRINDEX register bits [12:3] to index into the periodic frame list. This means that the host controller visits each frame list element eight consecutive times before incrementing to the next periodic frame list element. Each iTD contains eight transaction descriptions, which map directly to FRINDEX register bits [2:0]. Therefore, each transaction descriptor corresponds to one micro-frame. Each iTD can span 8 micro-frames worth of transactions.

When the host controller fetches an iTD, it uses FRINDEX register bits [2:0] to index into the transaction description array.

If the active bit in the Status field of the indexed transaction description is set to zero, the host controller ignores the iTD and follows the Next pointer to the next schedule data structure.

When the indexed active bit is one, the host controller continues to parse the iTD. It stores the indexed transaction description and the general endpoint information (device address, endpoint number, maximum packet size, and so on.). It also uses the Page Select (PG) field to index the buffer pointer array, storing the selected buffer pointer and the next sequential buffer pointer. For example, if PG field is 0, then the host controller stores Page 0 and Page 1.

The host controller constructs a physical data buffer address by concatenating the current buffer pointer (as selected using the current transaction description's PG field) and the transaction description's Transaction Offset field. The host controller uses the endpoint addressing information and I/O-bit to execute a transaction to the appropriate endpoint.

When the transaction is complete, the host controller clears the active bit and writes back any additional status information to the Status field in the currently selected transaction description.

The data buffer associated with the iTD must be virtually contiguous memory. Seven page pointers are provided to support eight high-bandwidth transactions regardless of the starting packet's offset alignment into the first page. A starting buffer pointer (physical memory address) is constructed by concatenating the page pointer (for example, page 0 pointer) selected by the active transaction descriptions' PG (for example, value: 00B) field with the transaction offset field. As the transaction moves data, the host controller must detect when an increment of the current buffer pointer crosses a page boundary. When this occurs the host controller simply replaces the current buffer pointer's page portion with the next page pointer (for example, page 1 pointer) and continues to move data. The size of each bus transaction is determined by the value in the Maximum Packet Size field. An iTD supports high-bandwidth pipes through the Mult (multiplier) field. When the Mult field is 1, 2, or 3, the host controller executes the specified number of Maximum Packet sized bus transactions for the endpoint in the current micro-frame. In other words, the Mult field represents a transaction count for the endpoint in the current micro-frame. If the Mult field is zero, the operation of the host controller is undefined. The transfer description is used to service all transactions indicated by the Mult field.

For OUT transfers, the value of the Transaction X Length field represents the total bytes to be sent during the micro-frame. The Mult field must be set by software to be consistent with Transaction X Length and Maximum Packet Sixe. The host controller sends the bytes in Maximum Packet Size'd portions. After each transaction, the host controller decrements its local copy of Transaction X Length by Maximum Packet Size. The number of bytes the host controller sends is always Maximum Packet Size or Transaction X Length, whichever is less. The host controller advances the transfer state in the transfer description, updates the appropriate record in the iTD and moves to the next schedule data structure. The maximum sized transaction supported is 3 x 1024 bytes.

For IN transfers, the host controller issues Mult transactions. It is assumed that software has properly initialized the iTD to accommodate all of the possible data. During each IN transaction, the host controller must use Maximum Packet Size to detect packet babble errors. The host controller keeps the sum of bytes received in the Transaction X Length field. After all transactions for the endpoint have completed for the micro-frame, Transaction X Length contains the total bytes received. If the final value of Transaction X Length is less than the value of Maximum Packet Size, then less data than was allowed for was received from the associated endpoint. This short packet condition does not set the USBINT bit in the USB_USBSTS register to one. The host controller will not detect this condition. If the device sends more than Transaction X Length or Maximum Packet Size bytes (whichever is less), then the host controller sets the Babble Detected bit to one and set the Active bit to zero. Note, that the host controller is not required to update the

iTD field Transaction X Length in this error scenario. If the Mult field is greater than one, then the host controller automatically executes the value of Mult transactions. The host controller will not execute all Mult transactions if:

- The endpoint is an OUT and Transaction X Length goes to zero before all the Mult transactions have executed (ran out of data), or
- The endpoint is an IN and the endpoint delivers a short packet, or an error occurs on a transaction before Mult transactions have been executed. The end of micro-frame may occur before all of the transaction opportunities have been executed. When this happens, the transfer state of the transfer description is advanced to reflect the progress that was made, the result written back to the iTD and the host controller proceeds to processing the next micro-frame. Refer to Appendix D for a table summary of the host controller required behavior for all the high-bandwidth transaction cases.

11.3.4.3.7.2 Software Operational Model for iTDs

A client buffer request to an isochronous endpoint may span 1 to N micro-frames. When N is larger than one, system software may have to use multiple iTDs to read or write data with the buffer (if N is larger than eight, it must use more than one iTD).

The following figure illustrates the simple model of how a client buffer is mapped by system software to the periodic schedule (that is the periodic frame list and a set of iTDs). On the right is the client description of its request. The description includes a buffer base address plus additional annotations to identify which portions of the buffer should be used with each bus transaction. In the middle is the iTD data structures used by the system software to service the client request. Each iTD can be initialized to service up to 24 transactions, organized into eight groups of up to three transactions each. Each group maps to one micro-frame's worth of transactions. The EHCI controller does not provide per-transaction results within a micro-frame. It treats the per-micro-frame transactions as a single logical transfer. On the left is the host controller's frame list. System software establishes references from the appropriate locations in the frame list to each of the appropriate iTDs. If the buffer is large, then system software can use a small set of iTDs to service the entire buffer. System software can activate the transaction description records (contained in each iTD) in any pattern required for the particular data stream.

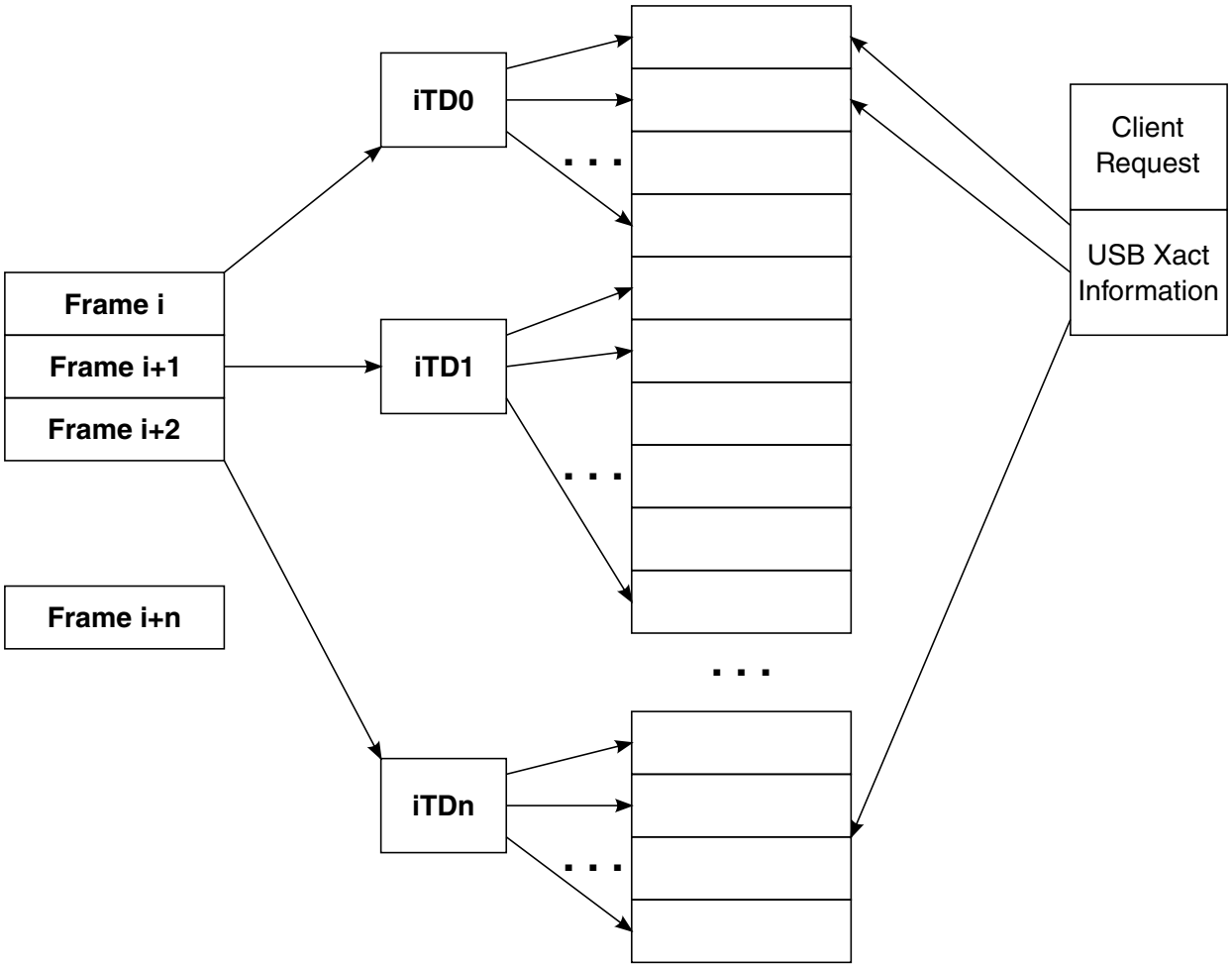


Figure 11-65. Example Association of iTDs to Client Request Buffer

As noted above, the client request includes a pointer to the base of the buffer and offsets into the buffer to annotate which buffer sections are to be used on each bus transaction that occurs on this endpoint. System software must initialize each transaction description in an iTD to ensure it uses the correct portion of the client buffer. For example, for each transaction description, the PG field is set to index the correct physical buffer page pointer and the Transaction Offset field is set relative to the correct buffer pointer page (for example, the same one referenced by the PG field). When the host controller executes a transaction it selects a transaction description record based on FRINDEX[2:0]. It then uses the current Page Buffer Pointer (as selected by the PG field) and concatenates to the transaction offset field. The result is a starting buffer address for the transaction. As the host controller moves data for the transaction, it must watch for a page wrap condition and properly advance to the next available Page Buffer Pointer. System software must not use the Page 6 buffer pointer in a transaction description where the length of the transfer wraps a page boundary. Doing so yields undefined behavior. The host controller hardware is not required to 'alias' the page selector to Page zero. USB 2.0 isochronous

endpoints can specify a period greater than one. Software can achieve the appropriate scheduling by linking iTDs into the appropriate frames (relative to the frame list) and by setting appropriate transaction description elements active bits to one.

11.3.4.3.7.2.1 Periodic scheduling threshold

The Isochronous Scheduling Threshold field in the USB_HCCPARAMS capability register is an indicator to system software as to how the host controller pre-fetches and effectively caches schedule data structures.

It is used by system software when adding isochronous work items to the periodic schedule. The value of this field indicates to system software the minimum distance it can update isochronous data (relative to the current location of the host controller execution in the periodic list) and still have the host controller process them.

The iTD and siTD data structures each describe 8 micro-frames worth of transactions. The host controller is allowed to cache one (or more) of these data structures in order to reduce memory traffic. Three basic caching models that account for the fact the isochronous data structures span 8 micro-frames. The three caching models are: no caching, micro-frame caching and frame caching.

When software is adding new isochronous transactions to the schedule, it always performs a read of the USB_FRINDEX register to determine the current frame and micro-frame the host controller is currently executing. Of course, there is no information about where in the micro-frame the host controller is, so a constant uncertainty-factor of one micro-frame has to be assumed. Combining the knowledge of where the host controller is executing with the knowledge of the caching model allows the definition of simple algorithms for how closely software can reliably work to the executing host controller.

No caching is indicated with a value of zero in the Isochronous Scheduling Threshold field. The host controller may pre-fetch data structures during a periodic schedule traversal (per micro-frame) but always dumps any accumulated schedule state at the end of the micro-frame. At the appropriate time relative to the beginning of every micro-frame, the host controller always begins schedule traversal from the frame list. Software can use the value of the USB_FRINDEX register (plus the constant 1 uncertainty-factor) to determine the approximate position of the executing host controller. When no caching is selected, software can add an isochronous transaction as near as 2 micro-frames in front of the current executing position of the host controller.

Frame caching is indicated with a non-zero value in bit [7] of the Isochronous Scheduling Threshold field. In the frame-caching model, system software assumes that the host controller caches one (or more) isochronous data structures for an entire frame (8 micro-frames). Software uses the value of the USB_FRINDEX register (plus the constant 1

uncertainty) to determine the current micro-frame/frame (assume modulo 8 arithmetic in adding the constant 1 to the micro-frame number). For any current frame N, if the current micro-frame is 0 to 6, then software can safely add isochronous transactions to Frame N + 1. If the current micro-frame is 7, then software can add isochronous transactions to Frame N + 2.

Micro-frame caching is indicated with a non-zero value in the least-significant 3 bits of the Isochronous Scheduling Threshold field. System software assumes the host controller caches one or more periodic data structures for the number of micro-frames indicated in the Isochronous Scheduling Threshold field. For example, if the count value were 2, then the host controller keeps a window of 2 micro-frames worth of state (current micro-frame, plus the next) on-chip. On each micro-frame boundary, the host controller releases the current micro-frame state and begins accumulating the next micro-frame state.

11.3.4.3.8 Asynchronous Schedule

The Asynchronous schedule traversal is enabled or disabled through the Asynchronous Schedule Enable bit in the USB_USBCMD register.

If the Asynchronous Schedule Enable bit is set to zero, then the host controller simply does not try to access the asynchronous schedule through the USB_ASYNCLISTADDR register. Likewise, when the Asynchronous Schedule Enable bit is one, then the host controller does use the USB_ASYNCLISTADDR register to traverse the asynchronous schedule. Modifications to the Asynchronous Schedule Enable bit are not necessarily immediate. Rather the new value of the bit is taken into consideration the next time the host controller needs to use the value of the USB_ASYNCLISTADDR register to get the next queue head.

The Asynchronous Schedule Status bit in the USB_USBSTS register indicates status of the asynchronous schedule. System software enables (or disables) the asynchronous schedule by writing one (or zero) to the Asynchronous Schedule Enable bit in the USB_USBCMD register. Software then can poll the Asynchronous Schedule Status bit to determine when the asynchronous schedule has made the desired transition. Software must not modify the Asynchronous Schedule Enable bit unless the value of the Asynchronous Schedule Enable bit equals that of the Asynchronous Schedule Status bit.

The asynchronous schedule is used to manage all Control and Bulk transfers. Control and Bulk transfers are managed using queue head data structures. The asynchronous schedule is based at the USB_ASYNCLISTADDR register. The default value of the USB_ASYNCLISTADDR register after reset is undefined and the schedule is disabled when the Asynchronous Schedule Enable bit is zero.

Software may only write this register with defined results when the schedule is disabled. For example, Asynchronous Schedule Enable bit in the USB_USBCMD and the Asynchronous Schedule Status bit in the USB_USBSTS register are zero. System software enables execution from the asynchronous schedule by writing a valid memory address (of a queue head) into this register. Then software enables the asynchronous schedule by setting the Asynchronous Schedule Enable bit to one. The asynchronous schedule is actually enabled when the Asynchronous Schedule Status bit is one.

When the host controller begins servicing the asynchronous schedule, it begins by using the value of the USB_ASYNCLISTADDR register. It reads the first referenced data structure and begins executing transactions and traversing the linked list as appropriate. When the host controller completes processing the asynchronous schedule, it retains the value of the last accessed queue head's horizontal pointer in the USB_ASYNCLISTADDR register. Next time the asynchronous schedule is accessed, this is the first data structure that is serviced. This provides round-robin fairness for processing the asynchronous schedule.

A host controller completes processing the asynchronous schedule when one of the following events occur:

- The end of a micro-frame occurs.
- The host controller detects an empty list condition (see Empty Asynchronous Schedule Detection)
- The schedule has been disabled through the Asynchronous Schedule Enable bit in the USB_USBCMD register.

The queue heads in the asynchronous list are linked into a simple circular list as shown in Figure 11-60. Queue head data structures are the only valid data structures that may be linked into the asynchronous schedule. An isochronous transfer descriptor (iTD or siTD) in the asynchronous schedule yields undefined results.

The maximum packet size field in a queue head is sized to accommodate the use of this data structure for all non-isochronous transfer types. The USB Specification, Revision 2.0 specifies the maximum packet sizes for all transfer types and transfer speeds. System software should always parameterize the queue head data structures according to the core specification requirements.

11.3.4.3.8.1 Adding Queue Heads to Asynchronous Schedule

This is a software requirement section.

There are two independent events for adding queue heads to the asynchronous schedule. The first is the initial activation of the asynchronous list. The second is inserting a new queue head into an activated asynchronous list.

Activation of the list is simple. System software writes the physical memory address of a queue head into the USB_ASYNCLISTADDR register, then enables the list by setting the Asynchronous Schedule Enable bit in the USB_USBCMD register to one.

When inserting a queue head into an active list, software must ensure that the schedule is always coherent from the host controllers' point of view. This means that the system software must ensure that all queue head pointer fields are valid. For example, qTD pointers have T-Bits set to one or reference valid qTDs and the Horizontal Pointer references a valid queue head data structure. The following algorithm represents the functional requirements:

```
InsertQueueHead (pQHeadCurrent, pQueueHeadNew)
          --
          -- Requirement: all inputs must be properly initialized.
          --
          -- pQHeadCurrent is a pointer to a queue head that is
          -- already in the active list
          -- pQHeadNew is a pointer to the queue head to be added
          --
          -- This algorithm links a new queue head into a existing
          -- list
          --
          pQueueHeadNew.HorizontalPointer  =  pQueueHeadCurrent.HorizontalPointer
          pQueueHeadCurrent.HorizontalPointer =  physicalAddressOf(pQueueHeadNew)
End InsertQueueHead
```

11.3.4.3.8.2 Removing Queue Heads from Asynchronous Schedule

This is a software requirement section.

There are two independent events for removing queue heads from the asynchronous schedule. The first is shutting down (deactivating) the asynchronous list. The second is extracting a single queue head from an activated list.

Software deactivates the asynchronous schedule by setting the Asynchronous Schedule Enable bit in the USB_USBCMD register to zero. Software can determine when the list is idle when the Asynchronous Schedule Status bit in the USB_USBSTS register is zero. The normal mode of operation is that software removes queue heads from the asynchronous schedule without shutting it down. Software must not remove an active queue head from the schedule. Software should first deactivate all active qTDs, wait for the queue head to go inactive, then remove the queue head from the asynchronous list. Software removes a queue head from the asynchronous list through the following algorithm. As illustrated, the unlinking is quite easy. Software merely must ensure all of the link pointers reachable by the host controller are kept consistent.

```
UnlinkQueueHead (pQHeadPrevious, pQueueHeadToUnlink, pQHeadNext)
          --
          -- Requirement: all inputs must be properly initialized.
          --
          -- pQHeadPrevious is a pointer to a queue head that
          -- references the queue head to remove
```

```
          -- pQHeadToUnlink is a pointer to the queue head to be
          -- removed
          -- pQheadNext is a pointer to a queue head still in the
          -- schedule. Software provides this pointer with the
          -- following strict rules:
          --          if the host software is one queue head, then
          --          pQHeadNext must be the same as
          --          QueueheadToUnlink.HorizontalPointer. If the host
          --          software is unlinking a consecutive series of
          --          queue heads, QHeadNext must be set by software to
          --          the  queue head remaining in the schedule.
          --
          -- This algorithm unlinks a queue head from a circular list
          --
          pQueueHeadPrevious.HorizontalPointer =  pQueueHeadToUnlink.HorizontalPointer
                                  pQueueHeadToUnlink.HorizontalPointer = pQHeadNext
End UnlinkQueueHead
```

If software removes the queue head with the H-bit set to one, it must select another queue head still linked into the schedule and set its H-bit to one. This should be completed before removing the queue head. The requirement is that software keep one queue head in the asynchronous schedule, with its H-bit set to one. At the point software has removed one or more queue heads from the asynchronous schedule, it is unknown whether the host controller has a cached pointer to them. Similarly, it is unknown how long the host controller might retain the cached information, as it is implementation dependent and may be affected by the actual dynamics of the schedule load. Therefore, once software has removed a queue head from the asynchronous list, it must retain the coherency of the queue head (link pointers, and so on). It cannot disturb the removed queue heads until it knows that the host controller does not have a local copy of a pointer to any of the removed data structures.

The method software uses to determine when it is safe to modify a removed queue head is to handshake with the host controller. The handshake mechanism allows software to remove items from the asynchronous schedule, then execute a simple, lightweight handshake that is used by software as a key that it can free (or reuse) the memory associated the data structures it has removed from the asynchronous schedule.

The handshake is implemented with three bits in the host controller. The first bit is a command bit (Interrupt on Async Advance Doorbell bit in the USB_USBCMD register) that allows software to inform the host controller that something has been removed from its asynchronous schedule. The second bit is a status bit (Interrupt on Async Advance bit in the USB_USBSTS register) that the host controller sets after it has released all on-chip state that may potentially reference one of the data structures just removed. When the host controller sets this status bit to one, it also sets the command bit to zero. The third bit is an interrupt enable (Interrupt on Async Advance bit in the USB_USBINTR register) that is matched with the status bit. If the status bit is one and the interrupt enable bit is one, then the host controller asserts a hardware interrupt.

The figure below illustrates a general example. In this example, consecutive queue heads (B and C) are unlinked from the schedule using the algorithm above. Before the unlink operation, the host controller has a copy of queue head A.

The unlink algorithm requires that as software unlinks each queue head, the unlinked queue head is loaded with the address of a queue head that remains in the asynchronous schedule.

When the host controller observes that doorbell bit being set to one, it makes a note of the local reachable schedule information. In this example, the local reachable schedule information includes both queue heads (A and B). It is sufficient that the host controller can set the status bit (and clear the doorbell bit) as soon as it has traversed beyond current reachable schedule information (that is traversed beyond queue head (B) in this example). The following figure illustrates the generic queue head unlink scenario.

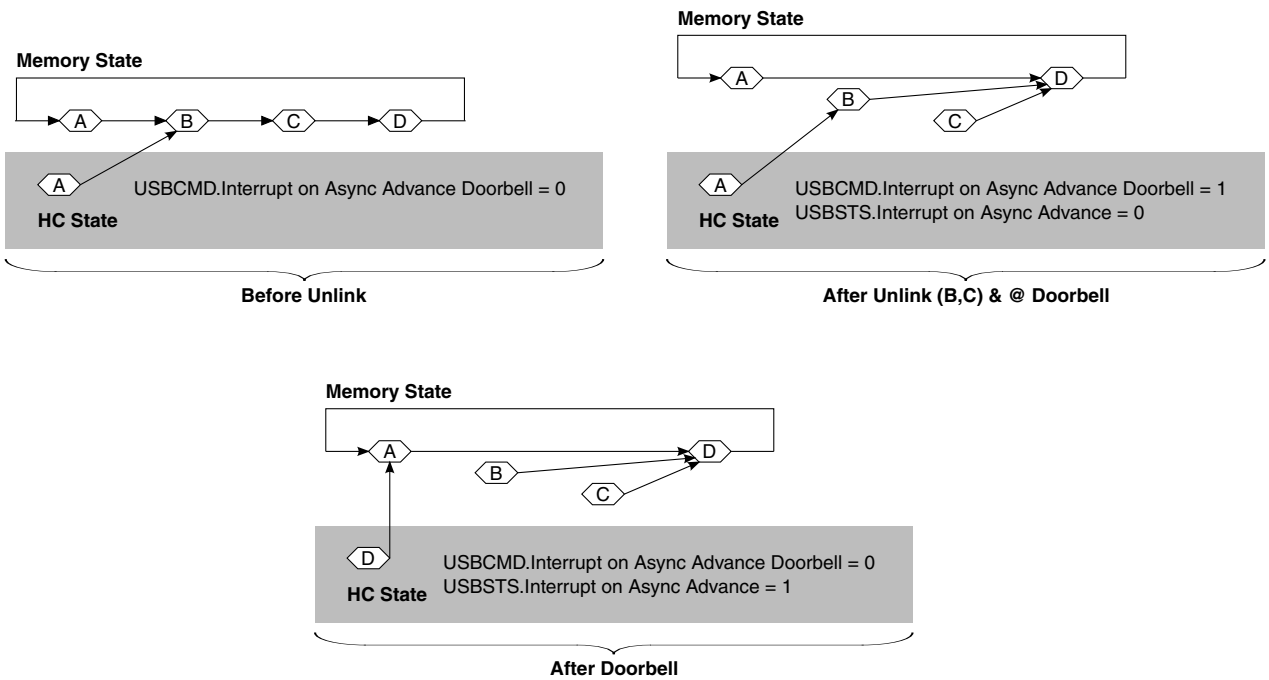


Figure 11-66. Generic Queue Head Unlink Scenario

Alternatively, a host controller implementation is allowed to traverse the entire asynchronous schedule list (for example, observed the head of the queue (twice)) before setting the Advance on Async status bit to one.

Software may re-use the memory associated with the removed queue heads after it observes the Interrupt on Async Advance status bit is set to one, following assertion of the doorbell. Software should acknowledge the Interrupt on Async Advance status as indicated in the USB_USBSTS register, before using the doorbell handshake again.

11.3.4.3.8.3 Empty Asynchronous Schedule Detection

The Enhanced Host Controller Interface uses two bits to detect when the asynchronous schedule is empty.

The queue head data structure (see Table 11-78) defines an *H-bit* in the queue head, which allows software to mark a queue head as being the *head* of the reclaim list. The Enhanced Host Controller Interface also keeps a 1-bit flag in the USB_USBSTS register (*Reclamation*) that is set to zero when the Enhanced Interface Host Controller observes a queue head with the H-bit set to one. The reclamation flag in the status register is set to one when any USB transaction from the asynchronous schedule is executed (or whenever the asynchronous schedule starts, see Asynchronous schedule traversal: *Start Event*).

If the Enhanced Host Controller Interface ever encounters an *H-bit* of one and a *Reclamation* bit of zero, the EHCI controller simply stops traversal of the asynchronous schedule.

An example illustrating the H-bit in a schedule is shown in the following figure.

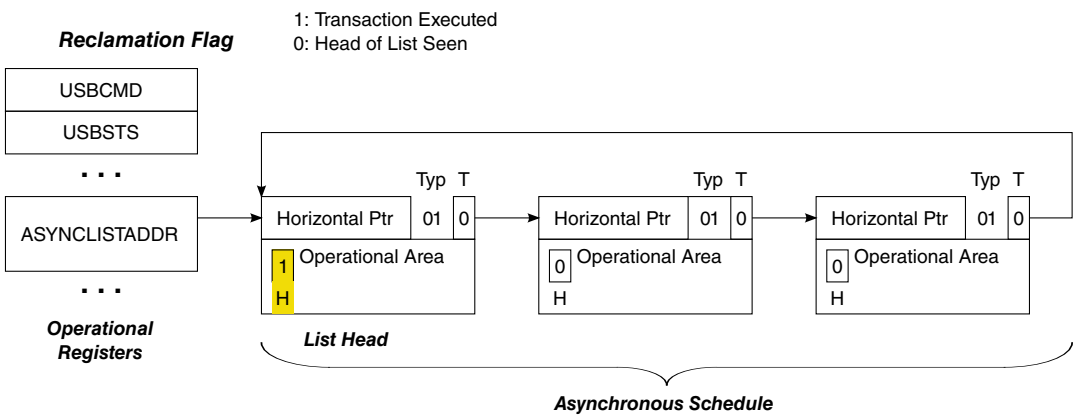


Figure 11-67. Asynchronous Schedule List w/Annotation to Mark Head of List

Software must ensure there is at most one queue head with the *H-bit* set to one, and that it is always coherent with respect to the schedule.

11.3.4.3.8.4 Restarting Asynchronous Schedule Before EOF

There are many situations where the host controller will detect an empty list *long* before the end of the micro-frame.

It is important to remember that under many circumstances the schedule traversal has stopped due to Nak/Nyet responses from all endpoints.

An example of particular interest is when a start-split for a bulk endpoint occurs early in the micro-frame. Given the EHCI simple traversal rules, the complete-split for that transaction may Nak/Nyet out very quickly. If it is the only item in the schedule, then the host controller ceases traversal of the Asynchronous schedule very early in the micro-frame. In order to provide reasonable service to this endpoint, the host controller should issue the complete-split before the end of the current micro-frame, instead of waiting until the next micro-frame. When the reason for host controller idling asynchronous schedule traversal is because of empty list detection, it is mandatory the host controller implement a 'waking' method to resume traversal of the asynchronous schedule. An example method is described below.

11.3.4.3.8.4.1 Example Method for Restarting Asynchronous Schedule Traversal

The reason for idling the host controller when the list is empty is to keep the host controller from unnecessarily occupying too much memory bandwidth. The question is: *how long should the host controller stay idle before restarting*?

The answer in this example is based on deriving a manifest constant, which is the amount of time the host controller will stay idle before restarting traversal. In this example, the manifest constant is called *AsyncSchedSleepTime*, and has a value of 10 µsec. The value is derived based on the analysis in Example Derivation for *AsyncSchedSleepTime*, The traversal algorithm is simple:

- Traverse the Asynchronous schedule until the either an End-Of-micro-Frame event occurs, or an empty list is detected. If the event is an End-of-micro-Frame, go attempt to traverse the Periodic schedule. If the event is an empty list, then set a sleep timer and go to a *schedule sleep* state.
- When the sleep timer expires, set working context to the Asynchronous Schedule start condition and go to *schedule active* state. The start context allows the HC to reload *Nakcnt* fields, and so on. So the HC has a chance to run for more than one iteration through the schedule.

This process simply repeats itself each micro-frame. The figure below illustrates a sample state machine to manage the active and sleep states of the Asynchronous Schedule traversal policy. There are three states: Actively traversing the Asynchronous schedule, Sleeping, and Not Active. The last two are similar in terms of interaction with the Asynchronous schedule, but the Not Active state means that the host controller is busy with the Periodic schedule or the Asynchronous schedule is not enabled. The Sleeping state is specifically a special state where the host controller is just waiting for a period of time before resuming execution of the Asynchronous schedule.

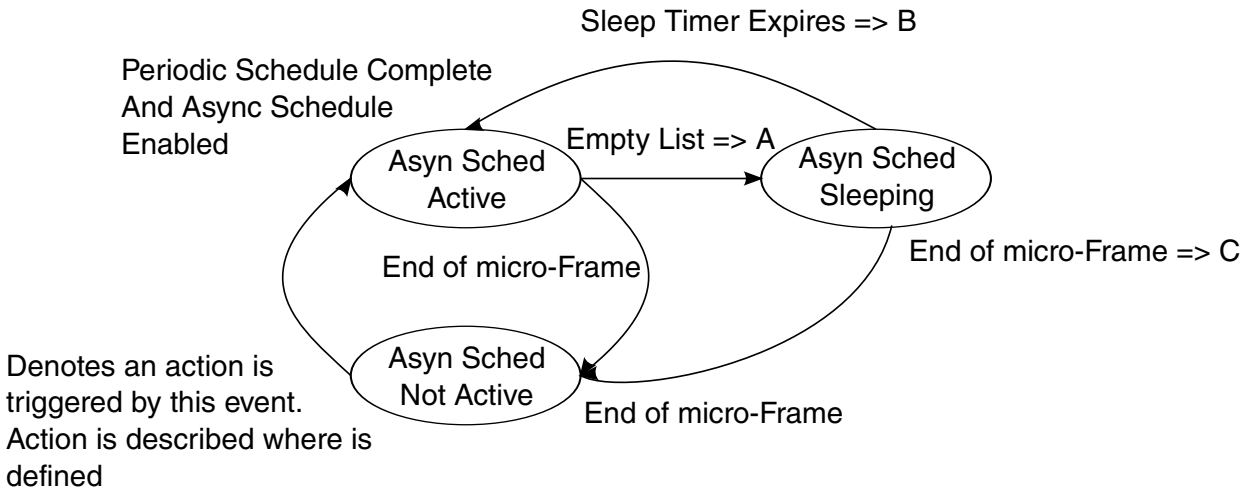


Figure 11-68. Example State Machine for Managing Asynchronous Schedule Traversal

The actions referred to in the figure above are defined in the following table.

Table 11-93. Asynchronous Schedule SM Transition Actions

Action	Action Description Label
A	On detection of the empty list, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.
B	When the *AsynchronousTraversalSleepTimer* expires, the host controller sets the *Reclamation* bit in the USBSTS register to one and moves the Nak Counter reload state machine to WaitForListHead (see Nak Count Reload Control).
C	The host controller cancels the sleep timer (*AsynchronousTraversalSleepTimer*).

11.3.4.3.8.4.2 Async Sched Not Active

This is the initial state of the traversal state machine after a host controller reset. The traversal state machine does not leave this state when the *Asynchronous Schedule Enable* bit in the USB_USBCMD register is zero.

This state is entered from Async Sched Active or Async Sched Sleeping states when the end-of-micro-frame event is detected.

11.3.4.3.8.4.3 Async Sched Active

This state is entered from the Async Sched Not Active state when the periodic schedule is not active. It is also entered from the Async Sched Sleeping states when the *AsyncrhonousTraversalSleepTimer* expires. On every transition into this state, the host controller sets the *Reclamation* bit in the USB_USBSTS register to one.

While in this state, the host controller continually traverses the asynchronous schedule until either the end of micro-frame or an empty list condition is detected.

11.3.4.3.8.4.4 Async Sched Sleeping

The state is entered from the Async Sched Active state when a schedule empty condition is detected. On entry to this state, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.

11.3.4.3.8.4.5 Example Derivation for AsyncSchedSleepTime

The derivation is based on analysis of what work the host controller could be doing next.

It assumes the host controller does not keep any state about what work is possibly pending in the asynchronous schedule. The schedule could contain any mix of the possible combinations of high- full- or low-speed control and bulk requests.

The table below summarizes some of the typical 'next transactions' that could be in the schedule, and the amount of time (for example *footprint*, or *wall clock*) the transaction takes to complete.

Table 11-94. Typical Low-/Full-speed Transaction Times

Transaction Attributes		Footprint (time)	Description
Speed	HS	11.9 ms	Maximum foot print for a worst-case, full-sized bulk data transaction.
Size	512	9.45 ms	Maximum footprint for an approximate best-case, full-sized bulk data transaction.
Type	Bulk		
Speed	FS	~50 ms	Approximate typical for full-sized bulk data. An 8-byte low-speed is about 2x, or between 90 and 100 ms.
Size	64		
Type	Bulk		
Speed	FS	~12 ms	Approximate typical for 8-byte bulk/control (that is setup)
Size	8		
Type	Cntrl		

A *AsyncSchedSleepTime* value of 10 µs provides a reasonable relaxation of the system memory load and still provides a good level of service for the various transfer types and payload sizes. For example, say we detect an empty list after issuing a start-split for a 64-byte full-speed bulk request. Assuming this is the only thing in the list, the host controller gets the results of the full-speed transaction from the hub during the fifth complete-split request. If the full-speed transaction was an IN and it nak'd, the 10 µs sleep period would allow the host controller to get the NAK results on the first complete-split.

11.3.4.3.8.5 Asynchronous schedule traversal: *Start Event*

Once the HC has *idled* itself through the empty schedule detection (Section 0), it will naturally *activate* and begin processing from the Periodic Schedule at the beginning of each micro-frame.

In addition, it may have idled itself early in a micro-frame. When this occurs (idles early in the micro-frame) the HC must occasionally *re-activate* during the micro-frame and traverse the asynchronous schedule to determine whether any progress can be made. The requirements and method for this restart are described in Restarting Asynchronous Schedule Before EOF . Asynchronous schedule *Start Event*s are defined to be:

- Whenever the host controller transitions from the periodic schedule to the asynchronous schedule. If the periodic schedule is disabled and the asynchronous schedule is enabled, then the beginning of the micro-frame is equivalent to the transition from the periodic schedule, or
- The asynchronous schedule traversal restarts from a sleeping state (see Restarting Asynchronous Schedule Before EOF).

11.3.4.3.8.6 Reclamation Status Bit (USBSTS Register)

The operation of the empty asynchronous schedule detection feature (see Empty Asynchronous Schedule Detection) depends on the proper management of the *Reclamation* bit in the USB_USBSTS register.

The host controller tests for an empty schedule just after it fetches a new queue head while traversing the asynchronous schedule (see Fetch Queue Head).

It is required that the host controller sets the *Reclamation* bit to one whenever an asynchronous schedule traversal *Start Event*, as documented inAsynchronous schedule traversal: *Start Event*, occurs. The *Reclamation* bit is also set to one whenever the host controller executes a transaction while traversing the asynchronous schedule (see Execute Transaction). The host controller sets the *Reclamation* bit to zero whenever it finds a queue head with its *H-bit* set to one. Software should only set a queue head's *H-bit* if the queue head is in the asynchronous schedule. If software sets the *H-bit* in an interrupt queue head to one, the resulting behavior is undefined. The host controller may set the *Reclamation* bit to zero when executing from the periodic schedule.

11.3.4.3.9 Operational Model for Nak Counter

This section describes the operational model for the *NakCnt* field defined in a queue head.

See Queue Head Initialization for more information. Software should not use this feature for interrupt queue heads. This rule is not required to be enforced by the host controller.

USB protocol has built-in flow control through the Nak response by a device. There are several scenarios, beyond the Ping feature, where an endpoint may naturally Nak or Nyet the majority of the time. An example is the host controller management of the split transaction protocol for control and bulk endpoints. All bulk endpoints (High- or Full-speed) are serviced through the same asynchronous schedule. The time between the *Start-split* transaction and the first *Complete-split* transaction could be very short (that is like when the endpoint is the only one in the asynchronous schedule). The hub NYETs (effectively Naks) the *Complete-split* transaction until the classic transaction is complete. This could result in the host controller thrashing memory, repeatedly fetching the queue head and executing the transaction to the Hub, which does not complete until after the transaction on the classic bus completes.

The two component fields in a queue head to support the throttling feature: a counter field (*NakCnt*), and a counter reload field (*RL*). *NakCnt* is used by the host controller as one of the criteria to determine whether or not to execute a transaction to the endpoint. The two operational modes associated with this counter:

- Not Used- This mode is set when the *RL* field is zero. The host controller ignores the *NakCnt* field for any execution of transactions through a queue head with an *RL* field of zero. Software must use this selection for interrupt endpoints.
- Nak Throttle Mode- This mode is selected when the *RL* field is non-zero. In this mode, the value in the *NakCnt* field represents the maximum number of Nak or Nyet responses the host controller tolerates on each endpoint. In this mode, the HC decrements the *NakCnt* field based on the token/handshake criteria listed in the table below. The host controller must reload *NakCnt* when the endpoint successfully moves data (for example, policy to reward device for moving data).

The following table describes the NakCnt field adjustment rules.

Table 11-95. NakCnt Field Adjustment Rules

Token	Handshake	
	Handshake NAK	NYET
IN/PING	decrement *NakCnt*	N/A (protocol error)
OUT	decrement *NakCnt*	No Action[1] Start
Split	decrement *NakCnt*	N/A (protocol error)
Complete Split	No Action	Decrement *NakCnt*

1. Recommended behavior on this response is to reload *NakCnt*

In summary, system software enables the counter by setting the reload field (*RL*) to a non-zero value. The host controller may execute a transaction if *NakCnt* is non-zero. The host controller does not execute a transaction if *NakCnt* is zero. The reload mechanism is described in detail in Nak Count Reload Control .

NOTE

When all queue heads in the Asynchronous Schedule either exhausts all transfers or all NakCnt's go to zero, then the host controller detects an empty Asynchronous Schedule and idle schedule traversal (see Empty Asynchronous Schedule Detection).

Any time the host controller begins a new traversal of the Asynchronous Schedule, a *Start Event* is assumed, see Asynchronous schedule traversal: *Start Event*. Every time a Start-Event occurs, the Nak Count reload procedure is enabled.

11.3.4.3.9.1 Nak Count Reload Control

When the host controller reaches the *Execute Transaction* state for a queue head (meaning that it has an active operational state), it checks to determine whether the *NakCnt* field should be reloaded from *RL* (see Execute Transaction). If the answer is yes, then *RL* is copied into *NakCnt*. After the reload or if the reload is not active, the host controller evaluates whether to execute the transaction.

The host controller must reload nak counters (*NakCnt* see Table 11-78) in queue heads during the first pass through the reclamation list after an asynchronous schedule Start Event (see Asynchronous schedule traversal: *Start Event* for the definition of the Start Event). The Asynchronous Schedule should have at most one queue head marked as the head (see Figure 11-67).

The following figure illustrates an example state machine that satisfies the operational requirements of the host controller detecting the first pass through the Asynchronous Schedule. This state machine is maintained internal to the host controller and is only used to gate reloading of the nak counter during the queue head traversal state: Execute Transaction (see the figure below). The host controller does not perform the nak counter reload operation if the RL field (see Table 11-78) is set to zero.

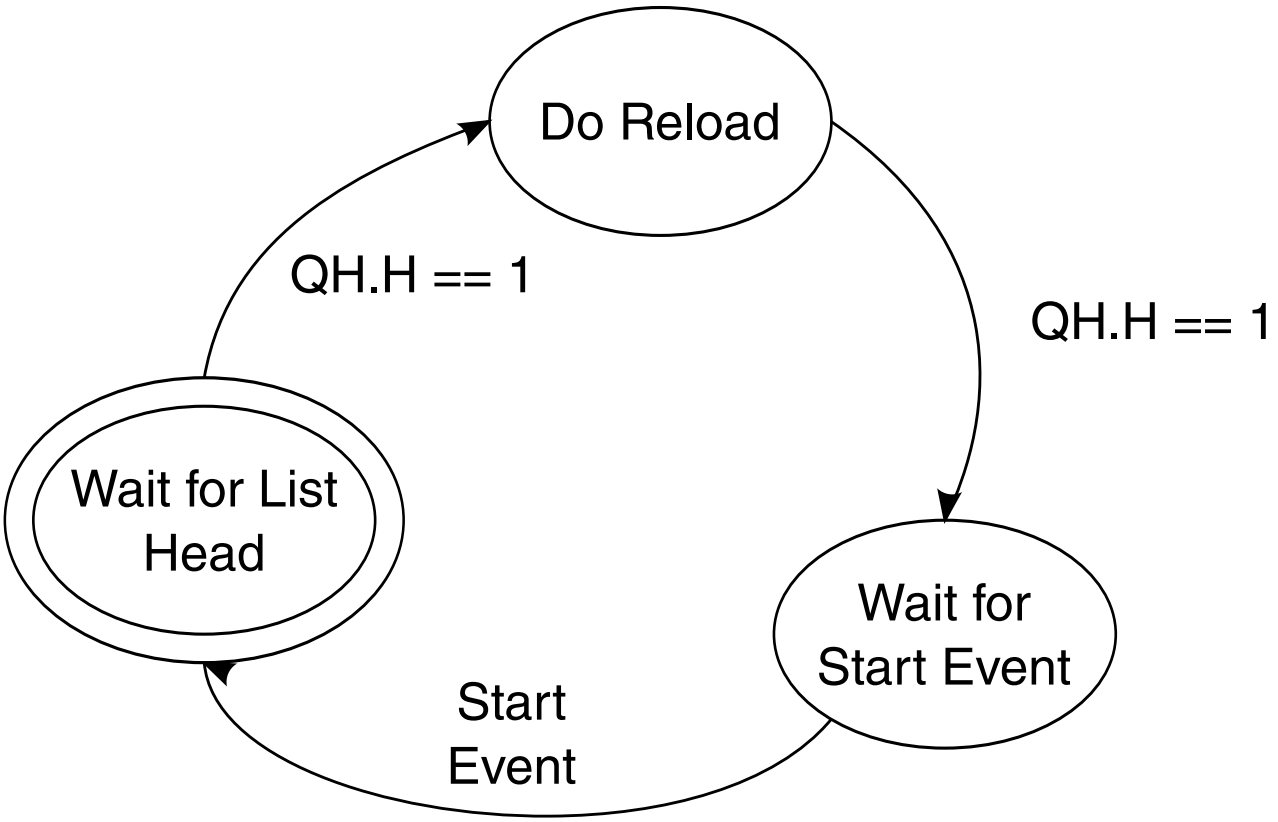


Figure 11-69. Example HC State Machine for Controlling Nak Counter Reloads

11.3.4.3.9.1.1 Wait for List Head

This is the initial state.

The state machine enters this state from Wait for Start Event when a start event as defined in Asynchronous schedule traversal: *Start Event*occurs.

The purpose of this state is to wait for the first observation of the head of the Asynchronous Schedule.

This occurs when the host controller fetches a queue head whose *H-bit* is set to one.

11.3.4.3.9.1.2 Do Reload

This state is entered from the Wait for List Head state when the host controller fetches a queue head with the *H-bit* set to one. While in this state, the host controller performs nak counter reloads for every queue head visited that has a non-zero nak reload value (*RL*) field.

11.3.4.3.9.1.3 Wait for Start Event

This state is entered from the *Do Reload* state when a queue head with the *H-bit* set to one is fetched. While in this state, the host controller does not perform nak counter reloads.

11.3.4.3.10 Managing Control/Bulk/Interrupt Transfers through Queue Heads

This section presents an overview of how the host controller interacts with queuing data structures.

Queue heads use the Queue Element Transfer Descriptor (qTD) structure. One queue head is used to manage the data stream for one endpoint. The queue head structure contains static endpoint characteristics and capabilities. It also contains a working area from where individual bus transactions for an endpoint are executed (see Overlay area defined in Table 11-78). Each qTD represents one or more bus transactions, which is defined in the context of this specification as a *transfer*.

The general processing model for the host controller's use of a queue head is simple:

- read a queue head,
- execute a transaction from the overlay area,
- write back the results of the transaction to the overlay area,
- move to the next queue head.

If the host controller encounters errors during a transaction, the host controller sets one (or more) of the error reporting bits in the queue head's *Status* field. The *Status* field accumulates all errors encountered during the execution of a qTD (for example, the error bits in the queue head *Status* field are 'sticky' until the transfer (qTD) has completed). This state is always written back to the source qTD when the transfer is complete. On transfer (for example, buffer or halt conditions) boundaries, the host controller must auto-advance (without software intervention) to the next qTD. Additionally, the hardware must be able to halt the queue so no additional bus transactions occurs for the endpoint and the host controller does not advance the queue.

An example host controller operational state machine of a queue head traversal is illustrated in the following figure. This state machine is a model for how a host controller should traverse a queue head. The host controller must be able to advance the queue from the *Fetch QH* state in order to avoid all hardware/software race conditions. This simple mechanism allows software to simply link qTDs to the queue head and *activate* them, then the host controller always *find* them if/when they are reachable. The figure below illustrates the Host Controller Queue Head Traversal State Machine.

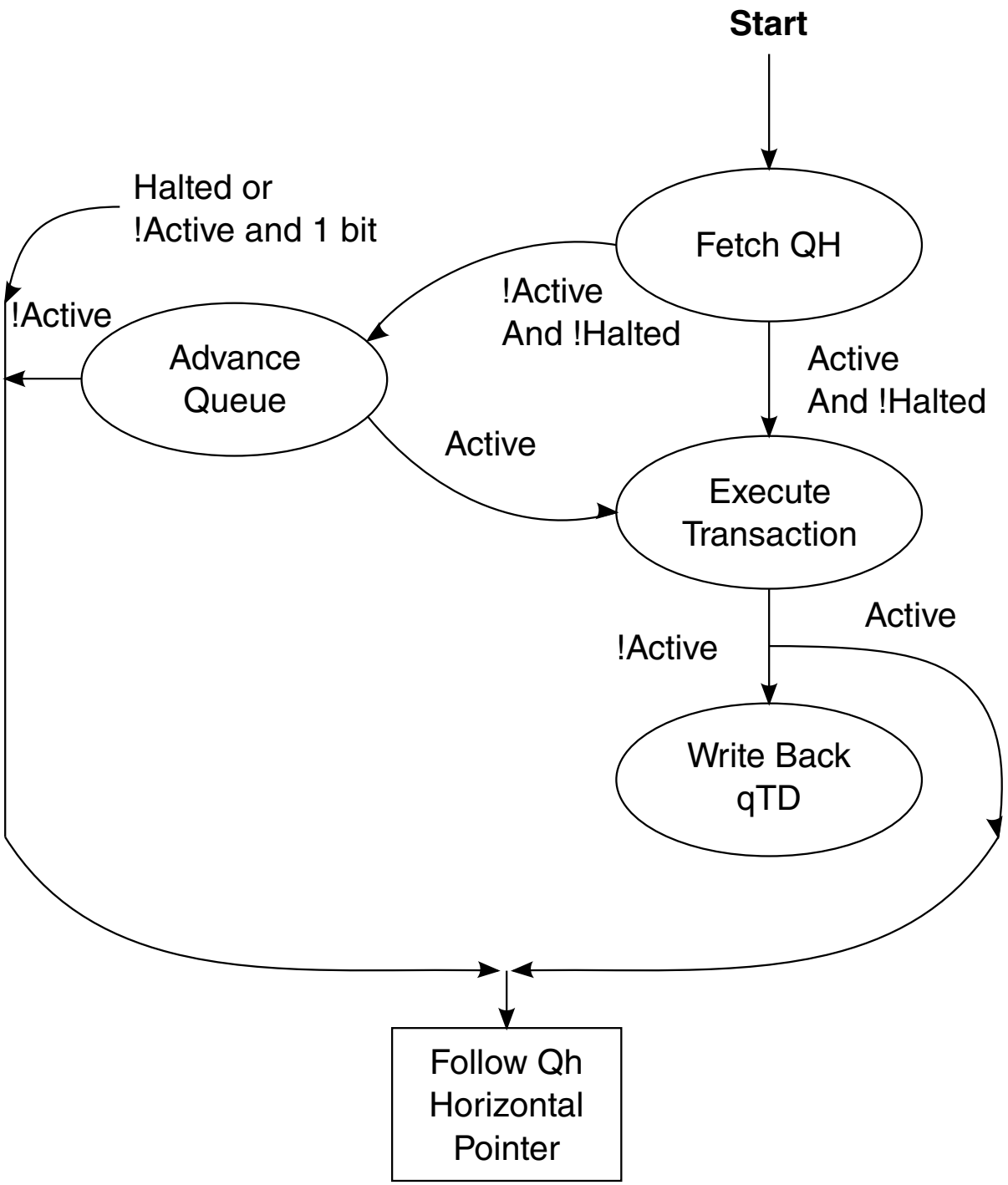


Figure 11-70. Host Controller Queue Head Traversal State Machine

This traversal state machine applies to all queue heads, regardless of transfer type or whether split transactions are required. The following sections describe each state. Each state description describes the entry criteria. The Execute Transaction state (see Execute

Transaction) describes the basic requirements for all endpoints. Split Transactions for Asynchronous Transfers and Split Transaction Interrupt describe details of the required extensions to the Execute Transaction state for endpoints requiring split transactions.

NOTE

Prior to software placing a queue head into either the periodic or asynchronous list, software must ensure the queue head is properly initialized. Minimally, the queue head should be initialized to the following (see Section Queue Head for layout of a queue head):

Valid static endpoint state.

- For the very first use of a queue head, software may zero-out the queue head transfer overlay, then set the *Next qTD Pointer* field value to reference a valid qTD.

11.3.4.3.10.1 Fetch Queue Head

A queue head can be referenced from the physical address stored in the ASYNCLISTADDR Register (see Next Asynch. Address (USB_ASYNCLISTADDR))/ Endpoint List Address (USB_ENDPTLISTADDR) Additionally, it may be referenced from the *Next LinkPointer* field of an iTD, siTD, FSTN or another Queue Head. If the referencing link pointer has the *Typ* field set to indicate a queue head, it is assumed to reference a queue head structure as defined in Table 11-78.

While in this state, the host controller performs operations to implement empty schedule detection (see Empty Asynchronous Schedule Detection) and Nak Counter reloads (see Operational Model for Nak Counter). After the queue head has been fetched, the host controller conducts the following queries for empty schedule detection:

- If queue head is not an interrupt queue head (that is *S-mask* is zero), and
- The *H-bit* is one, and
- The *Reclamation* bit in the USBSTS register is zero.

When these criteria are met, the host controller stops traversing the asynchronous list (as described in Empty Asynchronous Schedule Detection). When the criteria are not met, the host controller continues schedule traversal. If the queue head is not an interrupt and the *H-bit* is one and the *Reclamation* bit is one, then the host controller sets the *Reclamation* bit in the USBSTS register to zero before completing this state. The operations for reloading of the Nak Counter are described in detail in Operational Model for Nak Counter.

This state is complete when the queue head has been read on-chip.

11.3.4.3.10.2 Advance Queue

To advance the queue, the host controller must find the next qTD, adjust pointers, perform the overlay and write back the results to the queue head.

This state is entered from the FetchQHD state if the overlay *Active* and *Halt* bits are set to zero. On entry to this state, the host controller determines which next pointer to use to fetch a qTD, fetches a qTD and determines whether or not to perform an overlay.

NOTE

If the *I-bit* is one and the *Active* bit is zero, the host controller immediately skips processing of this queue head, exits this state and uses the horizontal pointer to the next schedule data structure. If the field *Bytes to Transfer* is not zero and the *T-bit* in the *Alternate Next qTD Pointer* is set to zero, then the host controller uses the *Alternate Next qTD Pointer*. Otherwise, the host controller uses the *NextqTD Pointer*. If *NextqTD Pointer's T-bit* is set to one, then the host controller exits this state and uses the horizontal pointer to the next schedule data structure.

Using the selected pointer the host controller fetches the referenced qTD. If the fetched qTD has its *Active* bit set to one, the host controller moves the pointer value used to reach the qTD (*Next* or *Alternate Next*) to the *Current qTD Pointer* field, then performs the overlay. If the fetched qTD has its *Active* bit set to zero, the host controller aborts the queue advance and follows the queue head's horizontal pointer to the next schedule data structure.

The host controller performs the overlay based on the following rules:

- The value of the data toggle (*dt*) field in the overlay area depends on the value of the *data toggle control* (*dtc*) bit (see Table 11-80).
- If the *EPS* field indicates the endpoint is a high-speed endpoint, the *Ping* state field is preserved by the host controller. The value of this field is not changed as a result of the overlay.
- *C-prog-mask* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *Frame Tag* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *NakCnt* field in the overlay area is loaded from the *RL* field in the queue head's Static Endpoint State.
- All other areas of the overlay are set by the incoming qTD.

The host controller exits this state when it has committed the write to the queue head.

11.3.4.3.10.3 Execute Transaction

The host controller enters this state from the Fetch Queue Head state only if the *Active* bit in *Status* field of the queue head is set to one.

On entry to this state, the host controller executes a few pre-operations, then checks some pre-condition criteria before committing to executing a transaction for the queue head.

The pre-operations performed and the pre-condition criteria depend on whether the queue head is an interrupt endpoint. The host controller can determine that a queue head is an interrupt queue head when the queue head's *S-mask* field contains a non-zero value. It is the responsibility of software to ensure the *S-mask* field is appropriately initialized based on the transfer type. There are other criteria that must be met if the *EPS* field indicates that the endpoint is a low- or full-speed endpoint, see Split Transactions for Asynchronous Transfers and Split Transaction Interrupt .

11.3.4.3.10.3.1 Interrupt Transfer Pre-condition Criteria

If the queue head is for an interrupt endpoint (for example, non-zero *S-mask* field), then the FRINDEX[2:0] field must identify a bit in the *S-mask* field that has one in it.

For example, an *S-mask* value of 00100000b would evaluate to true only when FRINDEX[2:0] is equal to 101b. If this condition is met then the host controller considers this queue head for a transaction.

11.3.4.3.10.3.2 Asynchronous Transfer Pre-operations and Pre-condition Criteria

If the queue head is not for an interrupt endpoint (for example, zero *S-mask* field), then the host controller performs one pre-operation and then evaluates one pre-condition criteria.

The pre-operation is:

Checks the Nak counter reload state (Operational Model for Nak Counter). It may be necessary for the host controller to reload the Nak Counter field. The reload is performed at this time.

The pre-condition evaluated is:

- Whether or not the *NakCnt* field has been reloaded, the host controller checks the value of the *NakCnt* field in the queue head. If *NakCnt* is non-zero, or if the *Reload Nak Counter* field is zero, then the host controller considers this queue head for a transaction.

11.3.4.3.10.3.3 Transfer Type Independent Pre-operations

Regardless of the transfer type, the host controller always performs at least one pre-operation and evaluates one pre-condition. The pre-operation is:

- A host controller internal transaction (down) counter *qHTransactionCounter* is loaded from the queue head's *Mult* field. A host controller implementation is allowed to ignore this for queue heads on the asynchronous list. It is mandatory for interrupt queue heads. Software should ensure that the *Mult* field is set appropriately for the transfer type.

The pre-conditions evaluated are:

- The host controller determines whether there is enough time in the micro-frame to complete this transaction (see Transaction Fit - A Best-Fit Approximation Algorithm for an example evaluation method). If there is not enough time to complete the transaction, the host controller exits this state.
- If the value of *qHTransactionCounter* for an interrupt endpoint is zero, then the host controller exits this state.

When the pre-operations are complete and pre-conditions are met, the host controller sets the *Reclamation* bit in the USBSTS register to one and then begins executing one or more transactions using the endpoint information in the queue head. The host controller iterates *qHTransactionCounter* times in this state executing transactions. After each transaction is executed, *qHTransactionCounter* is decremented by one. The host controller exits this state when one of the following events occurs:

- The *qHTransactionCounter* decrements to zero, or
- The endpoint responds to the transaction with any handshake other than an ACK,[4] or
- The transaction experiences a transaction error, or
- The *Active* bit in the queue head goes to zero, or
- There is not enough time in the micro-frame left to execute the next transaction(see Transaction Fit - A Best-Fit Approximation Algorithm) for example method for implementing the frame boundary test).

NOTE

For a high-bandwidth interrupt OUT endpoint, the host controller may optionally immediately retry the transaction if it fails.

The results of each transaction is recorded in the on-chip overlay area. If data was successfully moved during the transaction, the transfer state in the overlay area is advanced. To advance queue head's transfer state, the *Total Bytes to Transfer* field is decremented by the number of bytes moved in the transaction, the data toggle bit (*dt*) is toggled, the current page offset is advanced to the next appropriate value (for example,

advanced by the number of bytes successfully moved), and the *C_Page* field is updated to the appropriate value (if necessary). See Buffer Pointer List Use for Data Streaming with qTDs .

NOTE

The *Total Bytes To Transfer* field may be zero when all the other criteria for executing a transaction are met. When this occurs, the host controller executes zero-length transaction to the endpoint. If the *PID_Code* field indicates an IN transaction and the device delivers data, the host controller detects a packet babble condition, set the *babble* and *halted* bits in the *Status* field, set the *Active* bit to zero, write back the results to the source qTD, then exit this state.

In the event an IN token receives a data PID mismatch response, the host controller must ignore the received data (for example not advance the transfer state for the bytes received). Additionally, if the endpoint is an interrupt IN, then the host controller must record that the transaction occurred (for example, decrement *qHTransactionCounter*). It is recommended (but not required) the host controller continue executing transactions for this endpoint if the resultant value of *qHTransactionCounter* is greater than one.

If the response to the IN bus transaction is a Nak (or Nyet) and *RL* is non-zero, *NakCnt* is decremented by one. If *RL* is zero, then no write-back by the host controller is required (for a transaction receiving a Nak or Nyet response and the value of *CErr* did not change). Software should set the *RL* field to zero if the queue head is an interrupt endpoint. Host controller hardware is not required to enforce this rule or operation.

After the transaction has finished and the host controller has completed the post processing of the results (advancing the transfer state and possibly *NakCnt*, the host controller writes back the results of the transaction to the queue head's overlay area in main memory).

The number of bytes moved during an IN transaction depends on how much data the device endpoint delivers. The maximum number of bytes a device can send is *MaximumPacket Size*. The number of bytes moved during an OUT transaction is either *Maximum Packet Length* bytes or *Total Bytes to Transfer*, whichever is less.

If there was a transaction error during the transaction, the transfer state (as defined above) is not advanced by the host controller. The *CErr* field is decremented by one and the status field is updated to reflect the type of error observed. Transaction errors are summarized in Transaction Error .

The following events causes the host controller to clear the *Active* bit in the queue head's overlay status field. When the *Active* bit transitions from one to zero, the transfer in the overlay is considered complete. The reason for the transfer completion (clearing the *Active* bit) determines the next state.

- *CErr* field decrements to zero. When this occurs the *Halted* bit is set to one and *Active* is set to zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The device responds to the transaction with a STALL PID. When this occurs, the *Halted* bit is set to one and the *Active* bit is set to zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The *Total Bytes to Transfer* field is zero after the transaction completes.
  - For a zero length transaction, it was zero before the transaction was started. When this condition occurs, the *Active* bit is set to zero.
- The PID code is an IN, and the number of bytes moved during the transaction is less than the *Maximum Packet Length*. When this occurs, the *Active* bit is set to zero and a short packet condition exists. The short-packet condition is detected during the Advance Queue state. Refer to Split Transactions for additional rules for managing low- and full-speed transactions.

With the exception of a NAK response (when *RL* field is zero), the host controller always writes the results of the transaction back to the overlay area in main memory. This includes when the transfer completes. For a high-speed endpoint, the queue head information written back includes minimally the following fields: The *PID Code* field indicates an IN and the device sends more than the expected number of bytes (for example *Maximum Packet Length* or *Total Bytes to Transfer* bytes, whichever is less) (for example a packet babble). This results in the host controller setting the *Halted* bit to one.

- NakCnt, dt, Total Bytes to Transfer, C_Page, Status, CERR, and Current Offset

For a low- or full-speed device the queue head information written back also includes the fields:

- C-prog-mask, FrameTag and S-bytes.

The duration of this state depends on the time it takes to complete the transaction(s) and the status write to the overlay is committed.

11.3.4.3.10.3.4 Halting a Queue Head

A halted endpoint is defined only for the transfer types that are managed through queue heads (control, bulk and interrupt).

The following events indicate that the endpoint has reached a condition where no more activity can occur without intervention from the driver:

- An endpoint may return a STALL handshake during a transaction,
- A transaction had three consecutive error conditions, or
- A Packet Babble error occurs on the endpoint.

When any of these events occur (for a queue head) the Host Controller halts the queue head and set the USBERRINT status bit in the USB_n_USBSTS register to one. To halt the queue head, the *Active* bit is set to zero and the *Halted* bit is set to one. There may be other error status bits that are set when a queue is halted. The host controller always writes back the overlay area to the source qTD when the transfer is complete, regardless of the reason (normal completion, short packet or halt). The host controller does not advance the transfer state on a transaction that results in a *Halt* condition (for example no updates necessary for *Total Bytes to Transfer*, *C_Page*, *Current Offset,* and *dt*). The host controller must update *CErr* as appropriate. When a queue head is halted, the *USB Error Interrupt* bit in the USB_n_USBSTS register is set to one. If the *USB Error Interrupt Enable* bit in the USB_n_USBINTR register is set to one, a hardware interrupt is generated at the next interrupt threshold.

11.3.4.3.10.3.5 Asynchronous Schedule Park Mode

Asynchronous Schedule Park mode is a special execution mode that can be enabled by system software, where the host controller is permitted to execute more than one bus transaction from a high-speed queue head in the Asynchronous schedule before continuing horizontal traversal of the Asynchronous schedule.

This feature has no effect on queue heads or other data structures in the Periodic schedule. This feature is similar in intent as the *Mult* feature that is used in the Periodic schedule. Where-as the *Mult* feature is a characteristic that is tunable for each endpoint; park-mode is a policy that is applied to all high-speed queue heads in the asynchronous schedule. It is essentially the specification of an iterator for consecutive bus transactions to the same endpoint. All of the rules for managing bus transactions and the results of those as defined in Execute Transaction apply. This feature merely specifies how many consecutive times the host controller is permitted to execute from the same queue head before moving to the next queue head in the Asynchronous List. This feature should allow the host controller to attain better bus utilization for those devices that are capable of moving data at maximum rate, while at the same time providing a fair service to all endpoints.

A host controller exports its capability to support this feature to system software by setting the *Asynchronous Schedule Park Capability* bit in the USB_n_HCCPARAMs register to one. This information keys system software that the *Asynchronous Schedule Park Mode Enable* and *Asynchronous Schedule Park Mode Count* fields in the USB_n_USBCMD register are modifiable. System software enables the feature by writing a one to the *Asynchronous Schedule Park Mode Enable* bit.

When park-mode is not enabled (for example *Asynchronous Schedule Park Mode Enable* bit in the USB_n_USBCMD register is zero), the host controller must not execute more than one bus transaction per high-speed queue head, per traversal of the asynchronous schedule. When park-mode is enabled, the host controller must not apply the feature to a queue head whose *EPS* field indicates a Low/Full-speed device (for example only one bus transaction is allowed from each Low/Full-speed queue head per traversal of the asynchronous schedule). Park-mode may only be applied to queue heads in the Asynchronous schedule whose *EPS* field indicates that it is a high-speed device.

The host controller must apply park mode to queue heads whose *EPS* field indicates a high-speed endpoint. The maximum number of consecutive bus transactions a host controller may execute on a high-speed queue head is determined by the value in the *Asynchronous Schedule Park Mode Count* field in the USB_n_USBCMD register. Software must not set *Asynchronous Schedule Park Mode Enable* bit to one and also set *Asynchronous Schedule Park Mode Count* field to zero. The resulting behavior is not defined. An example behavioral example describes the operational requirements for the host controller implementing park-mode. This feature does not affect how the host controller handles the bus transaction as defined in Execute Transaction . It only effects how many consecutive bus transactions for the current queue head can be executed. All boundary conditions, error detection and reporting applies as usual. This feature is similar in concept to the use of the *Mult* field for high-bandwidth Interrupt for queue heads in the Periodic Schedule.

The host controller effectively loads an internal down-counter *PM-Count* from *Asynchronous Schedule Park Mode Count* when *Asyncrhonous Schedule Park Mode Enable* bit is one, and a high-speed queue head is first fetched and meets all the criteria for executing a bus transaction. After the bus transaction, *PM-Count* is decremented. The host controller may continue to execute bus transactions from the current queue head until *PM-Count* goes to zero, an error is detected, the buffer for the current transfer is exhausted or the endpoint responds with a flow-control or STALL handshake.

The following table summarizes the responses that effect whether the host controller continues with another bus transaction for the current queue head.

Table 11-96. Actions for Park Mode, based on Endpoint Response and Residual Transfer State

PID	Endpoint Response	Transfer State after Transaction		Action
		PM-Count	Bytes to Transfer	
IN	DATA[0,1] w/Maximum Packet sized data	Not zero	Not Zero	Allowed to perform another bus transaction.[1, 2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't care	Move to next QH.

Table continues on the next page...

Table 11-96. Actions for Park Mode, based on Endpoint Response and Residual Transfer State (continued)

	DATA[0,1] w/short packet	Don't care	Don't care	Retire qTD and move to next QH.
	NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH.
OUT	ACK	Not zero	Not Zero	Allowed to perform another bus transaction.[2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't' care	Move to next QH.
	NYET, NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH
PING	ACK	Not Zero	Not Zero	Allowed to perform another bus transaction. [2]
	NAK	Don't care	Don't care	Move to next QH
	STALL, XactErr	Don't care	Don't care	Move to next QH

1. The host controller may continue to execute bus transactions from the current high-speed queue head (if *PM-Count* is not equal to zero), if a PID mismatch is detected (for example expected DATA1 and received DATA0, or visa-versa).
2. This specification does not *require* that the host controller execute another bus transaction when *PM-Count* is non-zero. Implementations are encouraged to make appropriate complexity and performance trade-offs.

11.3.4.3.10.4 Write Back qTD

This state is entered from the Execute Transaction state when the *Active* bit is set to zero.

The source data for the write-back is the transfer results area of the queue head overlay area (see Table 11-96).

The host controller uses the *Current qTD Pointer* field as the target address for the qTD.

The queue head transfer result area is written back to the transfer result area of the target qTD. This state is also referred to as: qTD retirement. The fields that must be written back to the source qTD include *Total Bytes to Transfer*, *Cerr*, and *Status*.

The duration of this state depends on when the qTD write-back is committed.

11.3.4.3.10.5 Follow Queue Head Horizontal Pointer

The host controller must use the horizontal pointer in the queue head to the next schedule data structure when any of the following conditions exist:

- If the *Active* bit is one on exit from the Execute Transaction state, or
- When the host controller exits the Write Back qTD state, or

- If the Advance Queue state fails to advance the queue because the target qTD is not active, or
- If the *Halted* bit is one on exit from the Fetch QH state.

There is no functional requirement that the host controller wait until the current transaction is complete before using the horizontal pointer to read the next linked data structure. However, it must wait until the current transaction is complete before executing the next data structure.

11.3.4.3.10.6 Buffer Pointer List Use for Data Streaming with qTDs

A qTD has an array of buffer pointers, which is used to reference the data buffer for a transfer. This specification requires that the buffer associated with the transfer be *virtually contiguous*.

This means: if the buffer spans more than one physical page, it must obey the following rules (the figure below illustrates an example):

- The first portion of the buffer must begin at some offset in a page and extend through the end of the page.
- The remaining buffer cannot be allocated in small chunks scattered around memory. For each 4 K chunk beyond the first page, each buffer portion matches to a full 4 K page. The final portion, which may only be large enough to occupy a portion of a page, must start at the top of the page and be contiguous within that page.

The buffer pointer list in the qTD is long enough to support a maximum transfer size of 20 K bytes. This case occurs when all five buffer pointers are used and the first offset is zero. A qTD handles a 16 Kbyte buffer with any starting buffer alignment.

The host controller uses the field *C_Page* field as an index value to determine which buffer pointer in the list should be used to start the current transaction. The host controller uses a different buffer pointer for each physical page of the buffer. This is always true, even if the buffer is physically contiguous.

The host controller must detect when the current transaction spans a page boundary and automatically move to the next available buffer pointer in the page pointer list. The next available pointer is reached by incrementing *C_Page* and pulling the next page pointer from the list. Software must ensure there are sufficient buffer pointers to move the amount of data specified in the *Bytes to Transfer* field.

The following figure illustrates a nominal example of how System software would initialize the buffer pointers list and the *C_Page* field for a transfer size of 16383 bytes. *C_Page* is set to zero. The upper 20-bits of Page 0 references the start of the physical

page. *Current Offset* (the lower 12-bits of queue head Dword 7) holds the offset in the page for example 2049 (for example 4096-2047). The remaining page pointers are set to reference the beginning of each subsequent 4 K page.

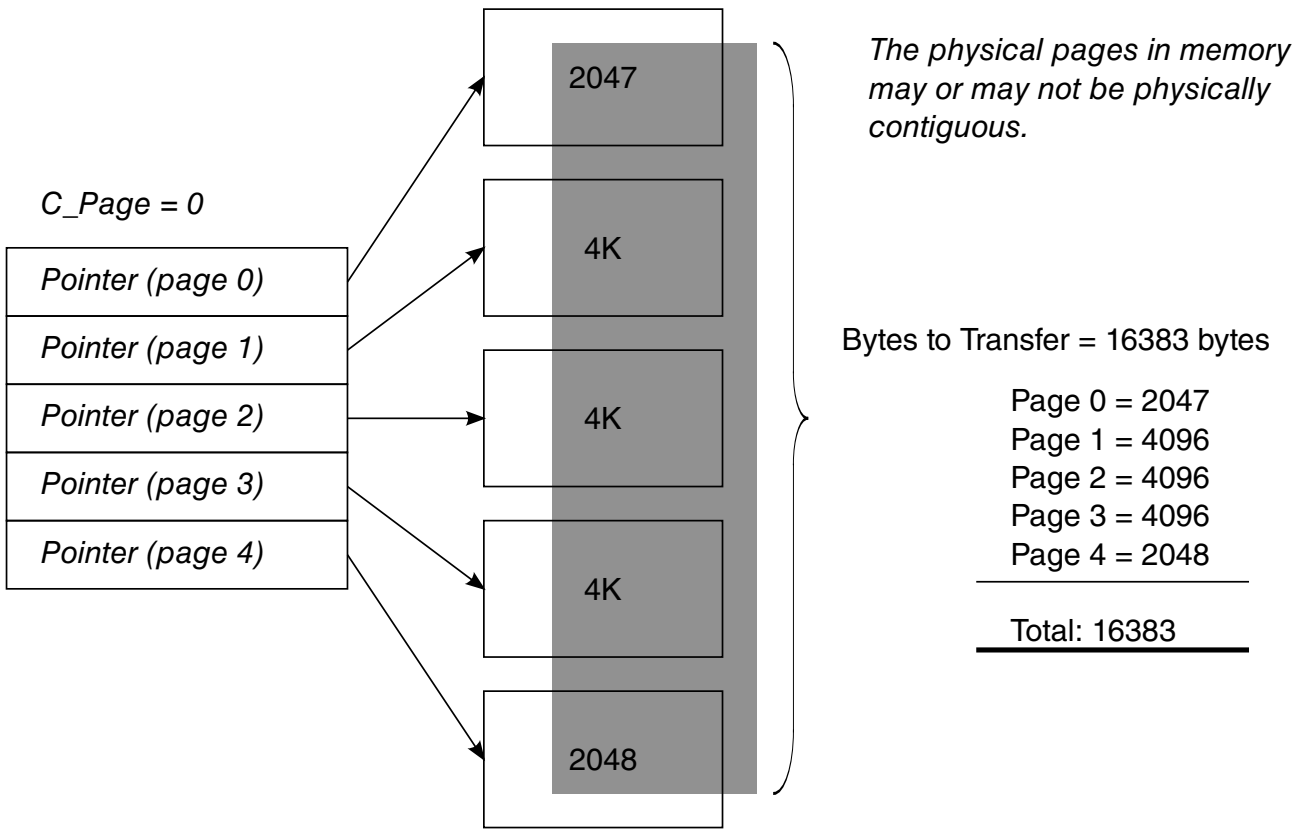


Figure 11-71. Example Mapping of qTD Buffer Pointers to Buffer Pages

For the first transaction on the qTD (assuming a 512-byte transaction), the host controller uses the first buffer pointer (page 0 because *C_Page* is set to zero) and concatenates the *Current Offset* field. The 512 bytes are moved during the transaction, the *Current Offset* and *Total Bytes to Transfer* are adjusted by 512 and written back to the queue head working area.

During the 4th transaction, the host controller needs 511 bytes in page 0 and one byte in page 1. The host controller increments *C_Page* (to 1) and use the page 1 pointer to move the final byte of the transaction. After the 4th transaction, the active page pointer is the page 1 pointer and *Current Offset* has rolled to one, and both are written back to the overlay area. The transactions continue for the rest of the buffer, with the host controller automatically moving to the next page pointer (that is *C_Page*) when necessary. The three conditions for how the host controller handles *C_Page*:

- The current transaction does not span a page boundary. The value of *C_Page* is not adjusted by the host controller.

- The current transaction does span a page boundary. The host controller must detect the page cross condition and advance to the next buffer while streaming data to/from the USB.
- The current transaction completes on a page boundary (that is the last byte moved for the current transaction is the last byte in the page for the current page pointer). The host controller must increment *C_Page* before writing back status for the transaction.

NOTE

The only valid adjustment the host controller may make to *C_Page* is to increment by one.

11.3.4.3.10.7 Adding Interrupt Queue Heads to the Periodic Schedule

The link path(s) from the periodic frame list to a queue head establishes in which frames a transaction can be executed for the queue head. Queue heads are linked into the periodic schedule so they are polled at the appropriate rate.

System software sets a bit in a queue head's *S-Mask* to indicate which micro-frame within 1 msec period a transaction should be executed for the queue head. Software must ensure that all queue heads in the periodic schedule have *S-Mask* set to a non-zero value. An *S-mask* with zero value in the context of the periodic schedule yields undefined results.

If the desired poll rate is greater than one frame, system software can use a combination of queue head linking and *S-Mask* values to spread interrupts of equal poll rates through the schedule so that the periodic bandwidth is allocated and managed in the most efficient manner possible. Some examples are illustrated in the following table.

Table 11-97. Example Periodic Reference Patterns for Interrupt Transfers with 2ms Poll Rate

Frame # Reference Sequence	Description
0, 2, 4, 6, 8, and so on *S-Mask* = 01h	A queue head for the *bInterval* of 2 msec (16 micro-frames) is linked into the periodic schedule so that it is reachable from the periodic frame list locations indicated in the previous column. In addition, the *S-Mask* field in the queue head is set to 01h, indicating that the transaction for the endpoint should be executed on the bus during micro-frame 0 of the frame.
0, 2, 4, 6, 8, and so on *S-Mask* = 02h	Another example of a queue head with a *bInterval* of 2 msec is linked into the periodic frame list at exactly the same interval as the previous example. However, the *S-Mask* is set to 02h indicating that the transaction for the endpoint should be executed on the bus during micro-frame 1 of the frame.

11.3.4.3.10.8 Managing Transfer Complete Interrupts from Queue Heads

The host controller sets an interrupt to be signaled at the next interrupt threshold when the completed transfer (qTD) has an *Interrupt on Complete* (*IOC*) bit set to one, or whenever a transfer (qTD) completes with a short packet.

If system software needs multiple qTDs to complete a client request (that is like a control transfer) the intermediate qTDs do not require interrupts. System software may only need a single interrupt to notify it that the complete buffer has been transferred. System software may set IOC's to occur more frequently. A motivation for this may be that it wants early notification so that interface data structures can be re-used in a timely manner.

11.3.4.3.11 Ping Control

USB 2.0 defines an addition to the protocol for high-speed devices called Ping. Ping is required for all USB 2.0 High-speed bulk and control endpoints.

Ping is not allowed for a split-transaction stream. This extension to the protocol eliminates the bad side-effects of Naking OUT endpoints. The *Status* field has a *Ping State* bit, which the host controller uses to determine the *next* actual PID it uses in the next transaction to the endpoint (see the table below).

The Ping State bit is only managed by the host controller for queue heads that meet the following criteria:

- Queue head is not an interrupt and
- *EPS* field equals High-Speed and
- *PIDCode* field equals OUT

The following table illustrates the state transition table for the host controller's responsibility for maintaining the PING protocol. Refer to Chapter 8 in the USB Specification Revision 2.0 for detailed description on the Ping protocol.

Table 11-98. Ping Control State Transition Table

Event			
Current	Host	Device	Next
Do Ping	PING	Nak	Do Ping
Do Ping	PING	Ack	Do OUT
Do Ping	PING	XactErr [1]	Do Ping
Do Ping	PING	Stall	N/C [2] Do
OUT	OUT	Nak	Do Ping
Do OUT	OUT	Nyet	Do Ping
Do OUT	OUT	Ack	Do OUT

Table continues on the next page...

Table 11-98. Ping Control State Transition Table (continued)

Do OUT	OUT	XactErr[1]	Do Ping
Do OUT	OUT	Stall	N/C [2]

1. Transaction Error (XactErr) is any time the host misses the handshake.
2. No transition change required for the Ping State bit. The Stall handshake results in the endpoint being halted (for example Active set to zero and Halt set to one). Software intervention is required to restart queue. 3 A Nyet response to an OUT means that the device has accepted the data, but cannot receive any more at this time. Host must advance the transfer state and additionally, transition the Ping State bit to Do Ping. The Ping State bit has the following encoding:

Table 11-99. Ping State bit Encoding

Value	Meaning
0B	Do OUT The host controller uses an OUT PID during the next bus transaction to this endpoint.
1B	Do Ping The host controller uses a PING PID during the next bus transaction to this endpoint.

The defined ping protocol (see USB 2.0 Specification, Chapter 8) allows the host to be *imprecise* on the initialization of the ping protocol (that is start in *Do OUT* when we don't know whether there is space on the device or not). The host controller manages the *Ping State* bit. System software sets the initial value in the queue head when it initializes a queue head. The host controller preserves the *Ping State* bit across all queue advancements. This means that when a new qTD is written into the queue head overlay area, the previous value of the *Ping State* bit is preserved.

11.3.4.3.12 Split Transactions

USB 2.0 defines extensions to the bus protocol for managing USB 1.x data streams through USB 2.0 Hubs.

This section describes how the host controller uses the interface data structures to manage data streams with full- and low-speed devices, connected below USB 2.0 hub, utilizing the split transaction protocol.

Refer to USB 2.0 Specification for the complete definition of the split transaction protocol. Full- and Low-speed devices are enumerated identically as high-speed devices, but the transactions to the Full- and Low-speed endpoints use the split-transaction protocol on the high-speed bus. The split transaction protocol is an encapsulation of (or wrapper around) the Full- or Low-speed transaction. The high-speed wrapper portion of the protocol is addressed to the USB 2.0 Hub and Transaction Translator below which the Full- or Low-speed device is attached.

The EHCI interface uses dedicated data structures for managing full-speed isochronous data streams (see Split Transaction Isochronous Transfer Descriptor (siTD)). Control, Bulk and Interrupt are managed using the queuing data structures (see Queue Head). The interface data structures need to be programmed with the device address and the

Transaction Translator number of the USB 2.0 Hub operating as the Low-/Full-speed host controller for this link. The following sections describe the details of how the host controller must process and manage the split transaction protocol.

11.3.4.3.12.1 Split Transactions for Asynchronous Transfers

A queue head in the asynchronous schedule with an *EPS* field indicating a full-or low-speed device indicates to the host controller that it must use split transactions to stream data for this queue head.

All full-speed bulk and full-, low-speed control are managed through queue heads in the asynchronous schedule.

Software must initialize the queue head with the appropriate device address and port number for the transaction translator that is serving as the full/low-speed host controller for the links connecting the endpoint. Software must also initialize the split transaction state bit (*SplitXState*) to Do-Start-Split. Finally, if the endpoint is a control endpoint, then system software must set the *Control Transfer Type* (*C*) bit in the queue head to one. If this is not a control transfer type endpoint, the *C* bit must be initialized by software to be zero. This information is used by the host controller to properly set the Endpoint Type (ET) field in the split transaction bus token. When the *C* bit is zero, the split transaction token's ET field is set to indicate a bulk endpoint. When the *C* bit is one, the split transaction token's ET field is set to indicate a control endpoint. Refer to Chapter 8 of USB Specification Revision 2.0 for details.

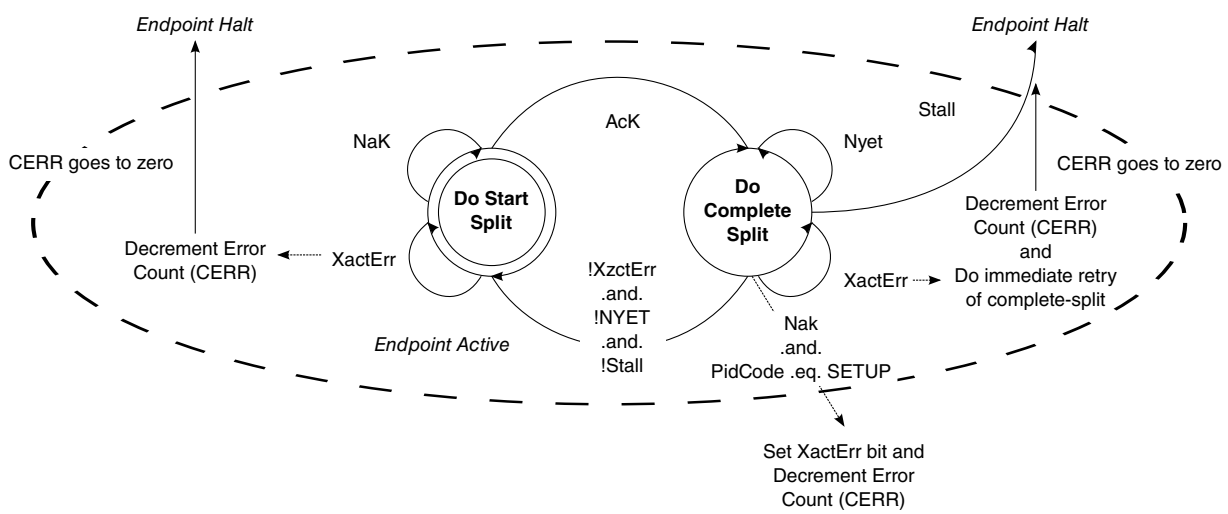


Figure 11-72. Host Controller Asynchronous Schedule Split-Transaction State Machine

11.3.4.3.12.1.1 Asynchronous - Do Start Split

This is the state which software must initialize a full- or low-speed asynchronous queue head. This state is entered from the Do Complete Split state only after a complete-split transaction receives a valid response from the transaction translator that is not a Nyet handshake.

For queue heads in this state, the host controller executes a start-split transaction to the appropriate transaction translator. If the bus transaction completes without an error and *PidCode* indicates an IN or OUT transaction, then the host controller reloads the error counter (*CErr*). If it is a successful bus transaction and the *PidCode* indicates a SETUP, the host controller does not reload the error counter. If the transaction translator responds with a Nak, the queue head is left in this state, and the host controller proceeds to the next queue head in the asynchronous schedule.

If the host controller times out the transaction (no response, or bad response) the host controller decrements *Cerr* and proceeds to the next queue head in the asynchronous schedule.

11.3.4.3.12.1.2 Asynchronous - Do Complete Split

This state is entered from the Do Start Split state only after a start-split transaction receives an Ack handshake from the transaction translator.

For queue heads in this state, the host controller executes a complete-split transaction to the appropriate transaction translator. If the transaction translator responds with a Nyet handshake, the queue head is left in this state, the error counter is reset and the host controller proceeds to the next queue head in the asynchronous schedule. When a Nyet handshake is received for a bus transaction where the queue head's *PidCode* indicates an IN or OUT, the host controller reloads the error counter (*CErr*). When a Nyet handshake is received for a complete-split bus transaction where the queue head's *PidCode* indicates a SETUP, the host controller must not adjust the value of *CErr*.

Independent of *PIDCode*, the following responses have the effects:

- Transaction Error (XactErr). Timeout or data CRC failure, and so on. The error counter (*Cerr*) is decremented by one and the complete split transaction is *immediately* retried (if possible). If there is not enough time in the micro-frame to execute the retry, the host controller MUST ensure that the next time the host controller begins executing from the Asynchronous schedule, it must begin executing from this queue head. If another start-split (for some other endpoint) is sent to the transaction translator before the complete-split is really completed, the transaction translator could dump the results (which were never delivered to the host). This is why the core specification states the retries must be immediate. A method to accomplish this behavior is to not advance the asynchronous schedule. When the host

controller returns to the asynchronous schedule in the next micro-frame, the first transaction from the schedule is the retry for this endpoint.

If *Cerr* went to zero, the host controller must halt the queue.

- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced and the state is exited. If the *PidCode* is a SETUP, then the Nak response is a protocol error. The *XactErr* status bit is set to one and the *CErr* field is decremented.
- STALL. The target endpoint responded with a STALL handshake. The host controller sets the *halt* bit in the status byte, retires the qTD but does not attempt to advance the queue.

If the *PidCode* indicates an IN, then any of following responses are expected:

- DATA0/1. On reception of data, the host controller ensures the PID matches the expected data toggle and checks CRC. If the packet is *good*, the host controller advances the state of the transfer, for example move the data pointer by the number of bytes received, decrement *BytesToTransfer* field by the number of bytes received, and toggle the *dt* bit. The host controller then exit this state. The response and advancement of transfer may trigger other processing events, such as retirement of the qTD and advancement of the queue.

If the data sequence PID does not match the expected, the data is ignored, the transfer state is not advanced and this state is exited. If the *PidCode* indicates an OUT/SETUP, then any of following responses are expected:

- ACK. The target endpoint accepted the data, so the host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount and the data toggle bit (*dt*) is toggled. The host controller then exit this state.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Managing Control/Bulk/Interrupt Transfers through Queue Heads).

11.3.4.3.12.2 Split Transaction Interrupt

Split-transaction Interrupt-IN/OUT endpoints are managed through the same data structures used for high-speed interrupt endpoints. They both co-exist in the periodic schedule.

Queue heads/qTDs offer the set of features required for reliable data delivery, which is characteristic to interrupt transfer types. The split-transaction protocol is managed completely within this defined functional transfer framework. For example, for a high-speed endpoint, the host controller visits a queue head, execute a high-speed transaction (if criteria are met) and advance the transfer state (or not) depending on the results of the entire transaction. For low- and full-speed endpoints, the details of the *execution* phase are different (that is takes more than one bus transaction to complete), but the remainder of the operational framework is intact. This means that the transfer advancement, and so on, occurs as defined in Managing Control/Bulk/Interrupt Transfers through Queue Heads, but only occurs on the completion of a split transaction.

11.3.4.3.12.2.1 Split Transaction Scheduling Mechanisms for Interrupt

Full- and low-speed Interrupt queue heads have an *EPS* field indicating full- or low-speed and have a non-zero *S-mask* field.

The host controller can detect this combination of parameters and assume the endpoint is a periodic endpoint. Low- and full-speed interrupt queue heads require the use of the split transaction protocol. The host controller sets the Endpoint Type (ET) field in the split token to indicate the transaction is an interrupt. These transactions are managed through a transaction translator's periodic pipeline. Software should not set these fields to indicate the queue head is an interrupt unless the queue head is used in the periodic schedule.

System software manages the per/transaction translator periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each endpoint occurs. The characteristics of the transaction translator are such that the high-speed transaction protocol must execute during explicit micro-frames, or the data or response information in the pipeline is lost.

The following figure illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule and queue head data structure. The S and ^{C}X labels indicate micro-frames where software can schedule start-splits and complete splits (respectively).

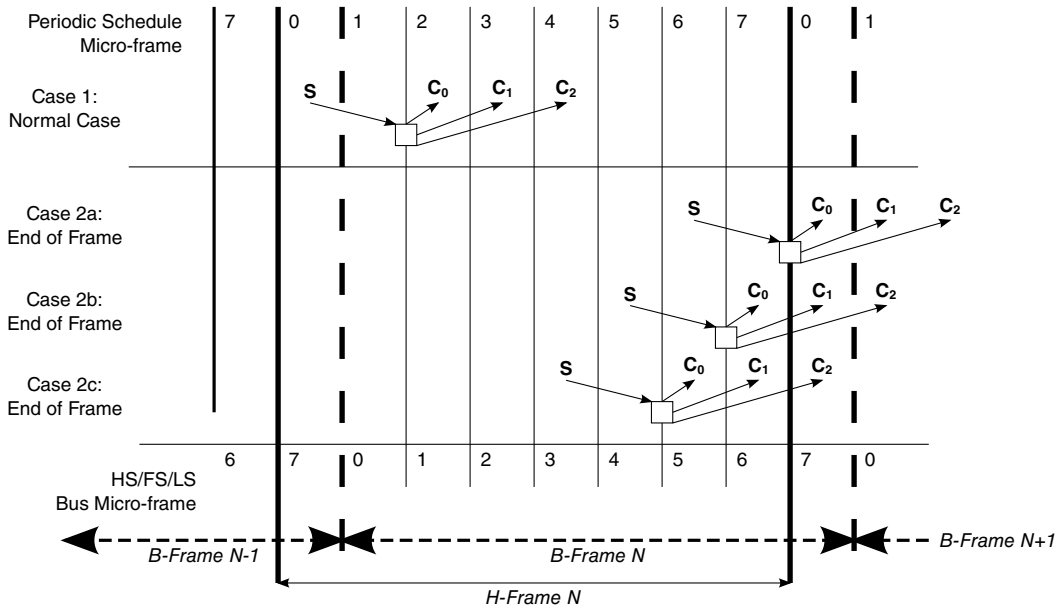


Figure 11-73. Split Transaction, Interrupt Scheduling Boundary Conditions

The scheduling cases are:

- Case 1: The normal scheduling case is where the entire split transaction is completely bounded by a frame (*H-Frame* in this case).
- Case 2a through Case 2c: The USB 2.0 Hub pipeline rules states clearly, when and how many complete-splits must be scheduled to account for earliest to latest execution on the full/low-speed link. The complete-splits may span the *H-Frame* boundary when the start-split is in micro-frame 4 or later. When this occurs, the *H-Frame* to *B-Frame* alignment requires that the queue head be reachable from consecutive periodic frame list locations. System software cannot build an efficient schedule that satisfies this requirement unless it uses FSTNs.

The figure below illustrates the general layout of the periodic schedule.

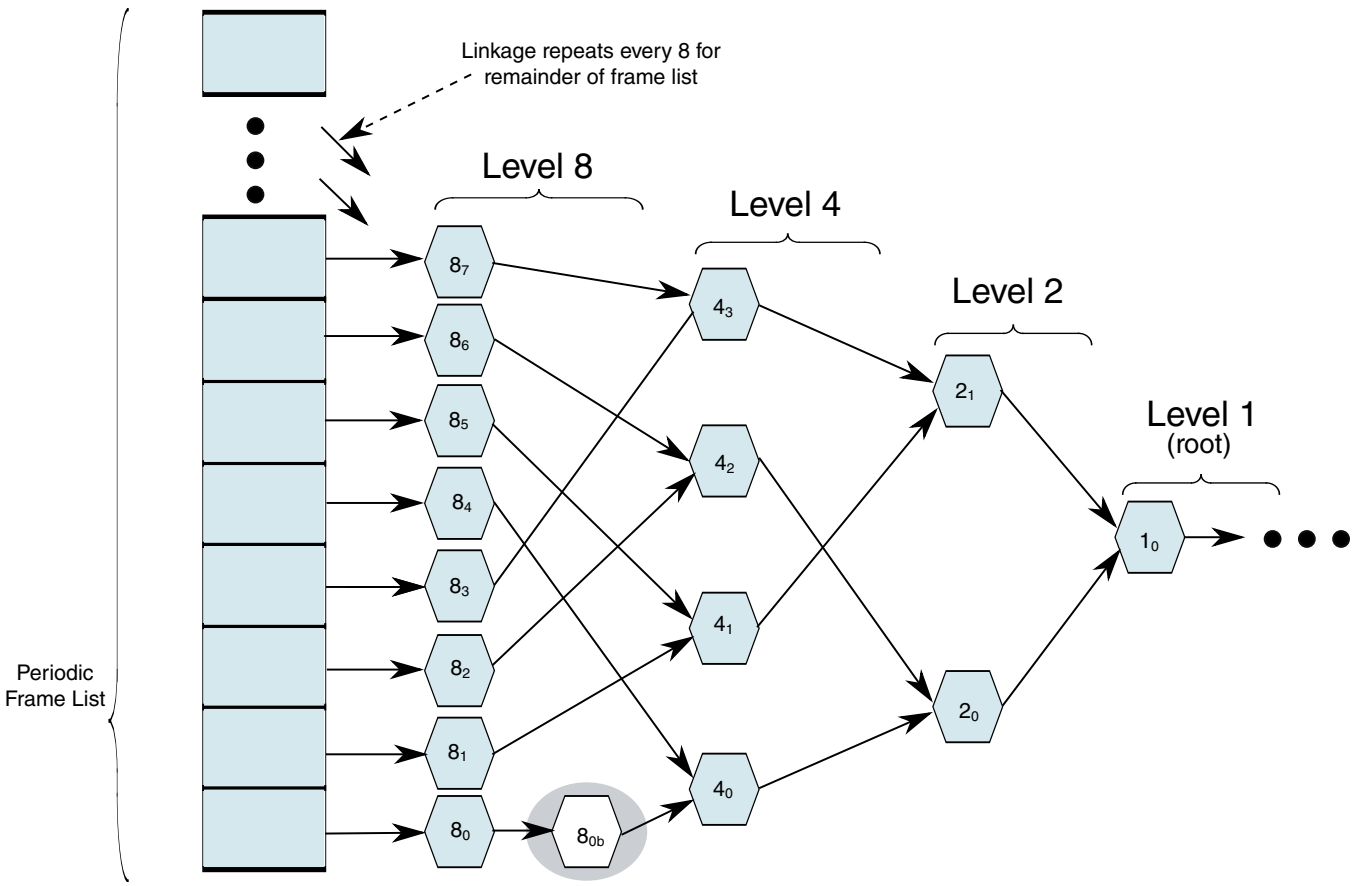


Figure 11-74. General Structure of EHCI Periodic Schedule Utilizing Interrupt Spreading

The periodic frame list is effectively the leaf level a binary tree, which is always traversed leaf to root. Each level in the tree corresponds to a 2^N poll rate. Software can efficiently manage periodic bandwidth on the USB by *spreading* interrupt queue heads that have the same poll rate requirement across all the available paths from the frame list. For example, system software can schedule eight poll rate 8 queue heads and account for them once in the high-speed bus bandwidth allocation.

When an endpoint is allocated an execution footprint that spans a frame boundary, the queue head for the endpoint must be reachable from consecutive locations in the frame list. An example would be if 8_{0b} where such an endpoint. Without additional support on the interface, to get 8_{0b} reachable at the correct time, software would have to link 8_1 to 8_{0b}. It would then have to move 4_1 and everything linked after into the same path as 4_0. This upsets the integrity of the binary tree and disallows the use of the spreading technique.

FSTN data structures are used to preserve the integrity of the binary-tree structure and enable the use of the spreading technique. Host Controller Operational Model for FSTNs defines the hardware and software operational model requirements for using FSTNs.

The following queue head fields are initialized by system software to instruct the host controller when to execute portions of the split-transaction protocol:

- *SplitXState*. This is single bit residing in the *Status* field of a queue head (see Table 11-76). This bit is used to track the current state of the split transaction.
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure 11-73, case one, the *S-mask* would have a value of 00000001b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates Do_Start, and the current micro-frame as indicated by FRINDEX[2:0] is 0, then execute a start-split transaction.
- *Frame C-mask.* This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure 11-73, case one, the *C-mask* would have a value of 00011100b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates Do_Complete, and the current micro-frame as indicated by FRINDEX[2:0] is 2, 3, or 4, then execute a complete-split transaction. It is software's responsibility to ensure that the translation between *H-Frames* and *B-Frames* is correctly performed when setting bits in *S-mask* and *C-mask*

11.3.4.3.12.2.2 Host Controller Operational Model for FSTNs

The FSTN data structure is used to manage Low/Full-speed interrupt queue heads that need to be reached from consecutive frame list locations (that is boundary cases 2a through 2c).

An FSTN is essentially a *back pointer*, similar in intent to the back pointer field in the siTD data structure (see siTD Back Link Pointer).

This feature provides software a simple primitive to save a schedule position, redirect the host controller to traverse the necessary queue heads in the previous frame, then restore the original schedule position and complete normal traversal.

The four components to the use of FSTNs:

- FSTN data structure.
- A *Save Place* indicator. This is always an FSTN with its *Back Path Link Pointer.T-bit* set to zero.

- A *Restore* indicator. This is always an FSTN with its *Back Path Link Pointer.T-bit* set to one.
- Host controller FSTN traversal rules.

When the host controller encounters an FSTN during micro-frames 2 through 7 it simply follows the node's *Normal Path Link Pointer* to access the next schedule data structure.

NOTE

The FSTN's *Normal Path Link Pointer.T-bit* may set to one, which the host controller must interpret as the end of periodic list mark.

When the host controller encounters a *Save-Place* FSTN in micro-frames 0 or 1, it saves the value of the *Normal Path Link Pointer* and set an internal flag indicating that it is executing in *Recovery Path* mode. *Recovery Path* mode modifies the host controller's rules for how it traverses the schedule and limits which data structures is considered for execution of bus transactions. The host controller continues executing in *Recovery Path* mode until it encounters a *Restore* FSTN or it determines that it has reached the end of the micro-frame (see details in the list below).

The rules for schedule traversal and limited execution while in *Recovery Path* mode are:

- Always follow the *Normal Path Link Pointer* when it encounters an FSTN that is a *Save-Place* indicator. The host controller must not recursively follow *Save-Place* FSTNs. Therefore, while executing in *Recovery Path* mode, it must never follow an FSTN's *Back Path Link Pointer*.
- Do not process an siTD or, iTD data structure. Simply follow its *Next Link Pointer*.
- Do not process a QH (Queue Head) whose *EPS* field indicates a high-speed device. Simply follow its *Horizontal Link Pointer*.
- When a QH's *EPS* field indicates a Full/Low-speed device, the host controller considers only it for execution if its *SplitXState* is DoComplete (note: this applies whether the *PID Code* indicates an IN or an OUT). See Execute Transaction and Tracking Split Transaction Progress for Interrupt Transfers for a complete list of additional conditions that must be met in general for the host controller to issue a bus transaction.
  - The host controller must not execute a Start-split transaction while executing in *Recovery Path* mode. See Periodic Isochronous - Do Complete Split for special handling when in *Recovery Path* mode.
- Stop traversing the *recovery path* when it encounters an FSTN that is a *Restore* indicator. The host controller unconditionally uses the saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* when returning to the normal path traversal. The

host controller must clear the context of executing a *Recovery Path* when it restores schedule traversal to the *Save-Place* FSTN's *Normal Path Link Pointer*.

- If the host controller determines that there is not enough time left in the micro-frame to complete processing of the periodic schedule, it abandons traversal of the recovery path, and clears the context of executing a recovery path. The result is that at the start of the next consecutive micro-frame, the host controller starts traversal at the frame list.

An example traversal of a periodic schedule that includes FSTNs is illustrated in the following figure.

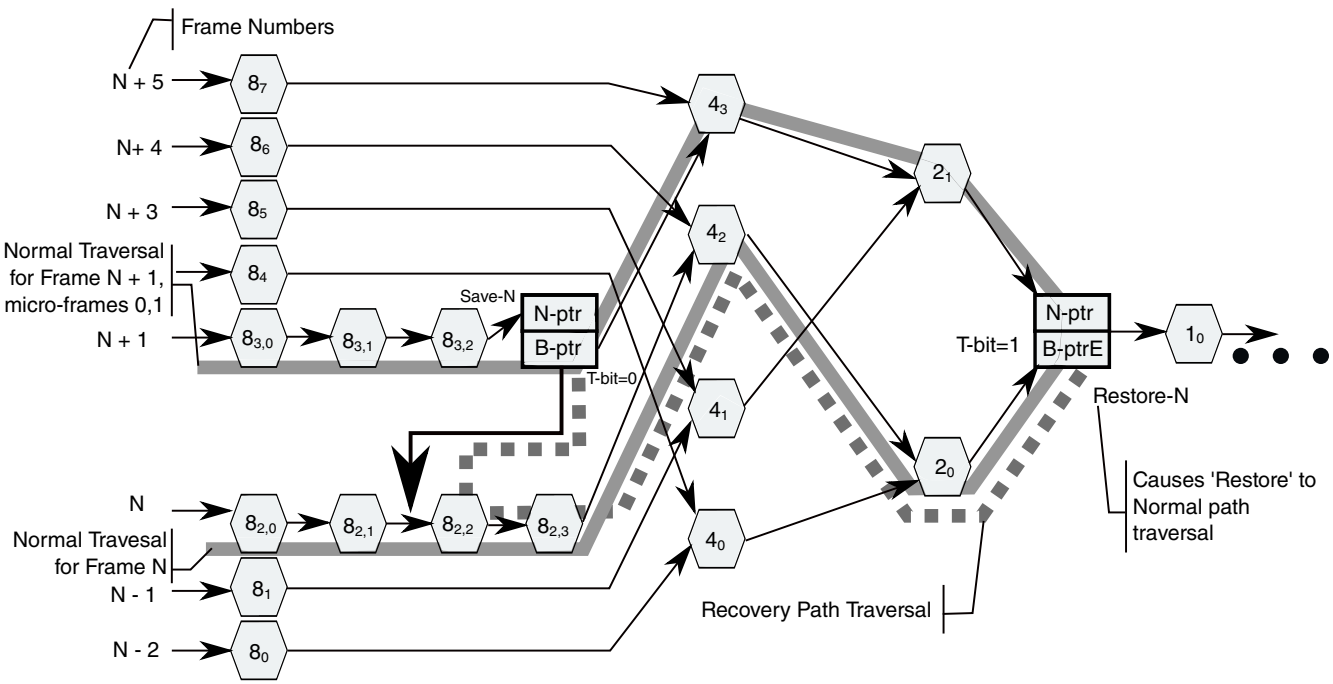


Figure 11-75. Example Host Controller Traversal of Recovery Path via FSTNs

In frame N+1 (micro-frames 0 and 1), when the host controller encounters Save-Path FSTN (Save-N), it observes that Save-N.Back Path Link Pointer.T-bit is zero (definition of a Save-Path indicator). The host controller saves the value of Save-N.Normal Path Link Pointer and follows Save-N.Back Path Link Pointer. At the same time, it sets an internal flag indicating that it is now in *Recovery Path* mode (the recovery path is annotated in the figure above with a large dashed line). The host controller continues traversing data structures on the recovery path and executing only those bus transactions as noted above, on the recovery path until it reaches Restore FSTN (Restore-N). Restore-N.Back Path Link Pointer.T-bit is set to one (definition of a Restore indicator), so the host controller exits *Recovery Path* mode by clearing the internal *Recovery Path* mode flag and commences (restores) schedule traversal using the saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* (for example Save-N.Normal Path Link Pointer). The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, $8_{2.2}$, $8_{2.3}$, 4_2,

2_0, Restore-N, 4_3, 2_1, Restore-N, 1_0 …}. The nodes on the recovery-path are in bold. In frame N (micro-frames 0-7), for this example, the host controller traverses all of the schedule data structures utilizing the *Normal Path Link Pointers* in any FSTNs it encounters. This is because the host controller has not yet encountered a *Save-Place* FSTN so it not executing in *Recovery Path* mode. When it encounters the *Restore* FSTN, (Restore-N), during micro-frames 0 and 1, it uses Restore-N.Normal Path Link Pointer to traverse to the next data structure (that is normal schedule traversal). This is because the host controller must use a Restore FSTN's *Normal Path Link Pointer* when not executing in a *Recovery-Path* mode. The nodes traversed during frame N include: {$8_{2.0}$, $8_{2.1}$, $8_{2.2}$, $8_{2.3}$, 4_2, 2_0, Restore-N, 1_0 …}.

In frame N+1 (micro-frames 2-7), when the host controller encounters Save-Path FSTN Save-N, it unconditionally follows Save-N.Normal Path Link Pointer. The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, 4_3, 2_1, Restore-N, 1_0 …}.

11.3.4.3.12.2.3 Software Operational Model for FSTNs

Software must create a consistent, coherent schedule for the host controller to traverse.

When using FSTNs, system software must adhere to the following rules:

- Each *Save-Place* indicator requires a matching *Restore* indicator.
  - The *Save-Place* indicator is an FSTN with a valid *Back Path Link Pointer and T-bit* equal to zero.
    - *Back Path Link Pointer.Typ* field must be set to indicate the referenced data structure is a queue head. The *Restore* indicator is an FSTN with its *Back Path Link Pointer.T-bit* set to one.
  - A *Restore* FSTN may be matched to one or more *Save-Place* FSTNs. For example, if the schedule includes a poll-rate 1 level, then system software only needs to place a *Restore* FSTN at the beginning of this list in order to match all possible *Save-Place* FSTNs.
- If the schedule does not have elements linked at a poll-rate level of one, and one or more *Save-Place* FSTNs are used, then System Software must ensure the *Restore* FSTN's *Normal Path Link Pointer*'s *T-bit* is set to one, as this is used to mark the end of the periodic list.
- When the schedule does have elements linked at a poll rate level of one, a *Restore* FSTN must be the first data structure on the poll rate one list. All traversal paths from the frame list converge on the poll-rate one list. System software must ensure that *Recovery Path* mode is exited before the host controller is allowed to traverse the poll rate level one list.
- A *Save-Place* FSTN's *Back Path Link Pointer* must reference a queue head data structure. The referenced queue head must be reachable from the previous frame list

location. In other words, if the *Save-Place* FSTN is reachable from frame list offset N, then the FSTN's *Back Path Link Pointer* must reference a queue head that is reachable from frame list offset N-1.

Software should make the schedule as efficient as possible. What this means in this context is that software should have no more than one *Save-Place* FSTN reachable in any single frame. Note there is times when two (or more, depending on the implementation) could exist as full/low-speed footprints change with bandwidth adjustments. This could occur, for example when a bandwidth re-balance causes system software to move the *Save-Place* FSTN from one poll rate level to another. During the transition, software must preserve the integrity of the previous schedule until the new schedule is in place.

11.3.4.3.12.2.4 Tracking Split Transaction Progress for Interrupt Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where data is lost.

For interrupt-IN transfers, data is lost when it makes it into the USB 2.0 hub, but the USB 2.0 host system is unable to get it from the USB 2.0 Hub and into the system before it expires from the transaction translator pipeline.

When a lost data condition is detected, the queue must be halted, thus signaling system software to recover from the error. A data-loss condition exists whenever a start-split is issued, accepted and successfully executed by the USB 2.0 Hub, but the complete-splits get unrecoverable errors on the high-speed link, or the complete-splits do not occur at the correct times. One reason complete-splits might not occur at the right time would be due to host-induced system hold-offs that cause the host controller to miss bus transactions because it cannot get timely access to the schedule in system memory.

The same condition can occur for an interrupt-OUT, but the result is not an endpoint halt condition, but rather effects only the progress of the transfer. The queue head has the following fields to track the progress of each split transaction. These fields are used to keep incremental state about which (and when) portions have been executed.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets one of the *C-prog*-mask bits for each complete-split executed. The bit position is determined by the micro-frame number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a

complete-split transaction. If the previous complete-splits have not been executed then it means one (or more) have been skipped and data has potentially been lost.
- *FrameTag*. This field is used by the host controller during the complete-split portion of the split transaction to tag the queue head with the frame number (*H-Frame* number) when the next complete split must be executed.
- *S-bytes*. This field can be used to store the number of data payload bytes sent during the start-split (if the transaction was an OUT). The *S-bytes* field must be used to accumulate the data payload bytes received during the complete-splits (for an IN).

11.3.4.3.12.2.5 Split Transaction Execution State Machine for Interrupt

In the following presentation, all references to micro-frame are in the context of a micro-frame within an *H-Frame.*

>As with asynchronous Full- and Low-speed endpoints, a split-transaction state machine is used to manage the split transaction sequence.

Aside from the fields defined in the queue head for scheduling and tracking the split transaction, the host controller calculates one internal mechanism that is also used to manage the split transaction. The internal calculated mechanism is:

- *cMicroFrameBit*. This is a single-bit encoding of the current micro-frame number. It is an eight-bit value calculated by the host controller at the beginning of every micro-frame. It is calculated from the three least significant bits of the *FRINDEX* register (that is, *cMicroFrameBit* = (1 shifted-left(*FRINDEX*[2:0])). The *cMicroFrameBit* has at most one bit asserted, which always corresponds to the current micro-frame number. For example, if the current micro-frame is 0, then *cMicroFrameBit* will equal 00000001b. The variable *cMicroFrameBit* is used to compare against the *S-mask* and *C-mask* fields to determine whether the queue head is marked for a start- or complete-splt transaction for the current micro-frame.

The following figure illustrates the state machine for managing a complete interrupt split transaction. There are two phases to each split transaction. The first is a single start-split transaction, which occurs when the *SplitXState* is at Do_Start and the single bit in *cMicroFrameBit* has a corresponding bit active in *QH.S-mask*. The transaction translator does not acknowledge the receipt of the periodic start-split, so the host controller unconditionally transitions the state to Do_Complete. Due to the available jitter in the transaction translator pipeline, there will be more than one complete-split transaction scheduled by software for the Do_Complete state. This translates simply to the fact that there are multiple bits set to a one in the *QH.C-mask* field.

The host controller keeps the queue head in the Do_Complete state until the split transaction is complete (see definition below), or an error condition triggers the *three-strikes-rule* (for example, after the host tries the same transaction three times, and each

encounters an error, the host controller will stop retrying the bus transaction and halt the endpoint, thus requiring system software to detect the condition and perform system-dependent recovery).

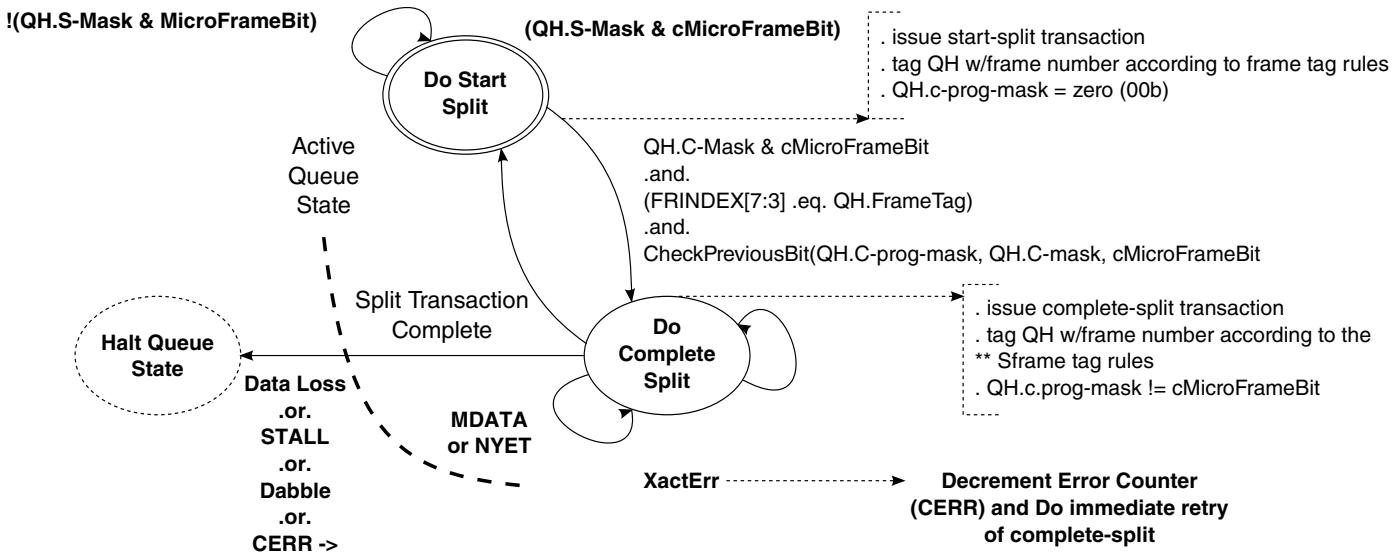


Figure 11-76. Split Transaction State Machine for Interrupt

See Previous Section for the frame tag management rules.

Periodic Interrupt - Do Start Split

This is the state software must initialize a full- or low-speed interrupt queue head *StartXState* bit. This state is entered from the Do_Complete Split state only after the split transaction is complete. This occurs when one of the following events occur: The transaction translator responds to a complete-split transaction with one of the following:

- NAK. A NAK response is a propagation of the full- or low-speed endpoint's NAK response.
- ACK. An ACK response is a propagation of the full- or low-speed endpoint's ACK response. Only occurs on an OUT endpoint.
- DATA 0/1. Only occurs for INs. Indicates that this is the last of the data from the endpoint for this split transaction.
- ERR. The transaction on the low-/full-speed link below the transaction translator had a failure (for example, timeout, bad CRC, etc.).
- NYET (and Last). The host controller issued the last complete-split and the transaction translator responded with a NYET handshake. This means that the start-split was not correctly received by the transaction translator, so it never executed a transaction to the full- or low-speed endpoint, see Section Periodic Isochronous - Do Complete Split for the definition of 'Last'.

Each time the host controller visits a queue head in this state (once within the Execute Transaction state), it performs the following test to determine whether to execute a start-split.

- *QH.S-mask* is bit-wise anded with *cMicroFrameBit*.

If the result is non-zero, then the host controller will issue a start-split transaction. If the *PIDCode* field indicates an IN transaction, the host controller must zero-out the *QH.S-bytes* field. After the split-transaction has been executed, the host controller sets up state in the queue head to track the progress of the complete-split phase of the split transaction. Specifically, it records the expected frame number into *QH.FrameTag* field (see Section), set *C-prog-mask* to zero (00h), and exits this state. Note that the host controller must not adjust the value of *CErr* as a result of completion of a start-split transaction.

Periodic Interrupt - Do Complete Split

This state is entered unconditionally from the Do Start Split state after a start-split transaction is executed on the bus. Each time the host controller visits a queue head in this state (once within the Execute Transaction state), it checks to determine whether a complete-split transaction should be executed now.

There are four tests to determine whether a complete-split transaction should be executed.

- Test A. *cMicroFrameBit* is bit-wise anded with *QH.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame.
- Test B. *QH.FrameTag* is compared with the current contents of *FRINDEX*[7:3]. An equal indicates a match.
- Test C. The complete-split progress bit vector is checked to determine whether the previous bit is set, indicating that the previous complete-split was appropriately executed. An example algorithm for this test is provided below:

```
Algorithm Boolean CheckPreviousBit(QH.C-prog-mask, QH.C-mask, cMicroFrameBit)
Begin
-- Return values:
-- TRUE - no error
-- FALSE - error
--
Boolean rvalue = TRUE;
previousBit = cMicroframeBit logical-rotate-right(1)
-- Bit-wise anding previousBit with C-mask indicates
-- whether there was an intent
-- to send a complete split in the previous micro-frame. So,
-- if the
-- 'previous bit' is set in C-mask, check C-prog-mask to
-- make sure it
-- happened.
If (previousBit bitAND QH.C-mask)then
                                If not(previousBit bitAND QH.C-prog-mask) then
                      rvalue = FALSE;
          End if
End If
-- If the C-prog-mask already has a one in this bit position,
```

```
-- then an aliasing
-- error has occurred. It will probably get caught by the
-- FrameTag Test, but
-- at any rate it is an error condition that as detectable here
-- should not allow
-- a transaction to be executed.
If (cMicroFrameBit bitAND QH.C-prog-mask) then
 rvalue = FALSE;
End if
return (rvalue)
End Algorithm
```

- Test D. Check to see if a start-split should be executed in this micro-frame. Note this is the same test performed in the Do Start Split state (see Section Periodic Isochronous - Do Start Split). Whenever it evaluates to TRUE and the controller is NOT processing in the context of a *Recovery Path* mode, it means a start-split should occur in this micro-frame. Test D and Test A evaluating to TRUE at the same time is a system software error. Behavior is undefined.

If (A .and. B .and. C .and. not(D)) then the host controller will execute a complete-split transaction. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. On completion of the complete-split transaction, the host controller records the result of the transaction in the queue head and sets *QH.FrameTag* to the expected *H-Frame* number (see Section). The effect to the state of the queue head and thus the state of the transfer depends on the response by the transaction translator to the complete-split transaction. The following responses have the effects (note that any responses that result in decrementing of the *CErr* will result in the queue head being halted by the host controller if the result of the decrement is zero):

- NYET (and Last). On each NYET response, the host controller checks to determine whether this is the last complete-split for this split transaction. Last is defined in this context as the condition where all of the scheduled complete-splits have been executed. If it is the last complete-split (with a NYET response), then the transfer state of the queue head is not advanced (never received any data) and this state exited. The transaction translator must have responded to all the clompete-splits with NYETs, meaning that the start-split issued by the host controller was not received. The start-split should be retried at the next poll period.
- The test for whether this is the Last complete split can be performed by XOR *QH.C-mask* with *QH.C-prog-mask*. If the result is all zeros then all complete-splits have been executed. When this condition occurs, the *XactErr* status bit is set to a one and the *CErr* field is decremented.
- NYET (and not Last). See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask* and *FrameTag*) and stay in this state. The host controller must not adjust *CErr* on this response.

- Transaction Error (XactErr). Timeout, data CRC failure, etc. The *CErr* field is decremented and the *XactErr* bit in the *Status* field is set to a one. The complete split transaction is *immediately* retried (if *Cerr* is non-zero).If there is not enough time in the micro-frame to complete the retry and the endpoint is an IN, or *CErr* is decremented to a zero from a one, the queue is halted. If there is not enough time in the micro-frame to complete the retry and the endpoint is an OUT and *CErr* is not zero, then this state is exited (that is, return to Do Start Split). This results in a retry of the entire OUT split transaction, at the next poll period. Refer to Chapter 11 Hubs (specifically the section full- and low-speed Interrupts) in the USB Specification Revision 2.0 for detailed requirements on why these errors must be immediately retried.
- ACK. This can only occur if the target endpoint is an OUT. The target endpoint ACK'd the data and this response is a propagation of the endpoint ACK up to the host controller. The host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount. And the data toggle bit (*dt*) is toggled. The host controller will then exit this state for this queue head. The host controller must reload *CErr* with maximum value on this response. Advancing the transfer state may cause other process events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers through Queue Heads).
- MDATA. This response will only occur for an IN endpoint. The transaction translator responded with zero or more bytes of data and an MDATA PID. The incremental number of bytes received is accumulated in *QH.S-bytes*. The host controller must not adjust *CErr* on this response.
- DATA0/1. This response may only occur for an IN endpoint. The number of bytes received is added to the accumulated byte count in *QH.S-bytes*. The state of the transfer is advanced by the result and the host controller will exit this state for this queue head.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers through Queue Heads).
- If the data sequence PID does not match the expected, the entirety of the data received in this split transaction is ignored, the transfer state is not advanced and this state is exited.
- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced, and this state is exited. The host controller must reload *CErr* with maximum value on this response.

- ERR. There was an error during the full- or low-speed transaction. The ERR status bit is set to a one, *Cerr* is decremented, the state of the transfer is not advanced, and this state is exited.
- STALL. The queue is halted (an exit condition of the Execute Transaction state). The status field bits: *Active* bit is set to zero and the *Halted* bit is set to a one and the qTD is retired. Responses which are not enumerated in the list or which are received out of sequence are illegal and may result in undefined host controller behavior. The other possible combinations of tests A, B, C, and D may indicate that data or response was lost. The table below lists the possible combinations and the appropriate action.

Table 11-100. Interrupt IN/OUT Do Complete Split State Execution Criteria

Condition	Action	Description
not(A) not(D)	Ignore QHD	Neither a start nor complete-split is scheduled for the current micro-frame.Host controller should continue walking the schedule.
A not(C)	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	Progress bit check failed. These means a complete-split has been missed. There is the possibility of lost data. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one.
A not(B) C	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	*QH.FrameTag* test failed. This means that exactly one or more *H-Frames* have been skipped. This means complete-splits and have missed. There is the possibility of lost data. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one.
A B C not(D)	Execute complete-split	This is the non-error case where the host controller executes a complete-split transaction.
D	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	This is a degenerate case where the start-split was issued, but all of the complete-splits were skipped and all possible intervening opportunities to detect the missed data failed to fire. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one. Note: When executing in the context of a *Recovery Path* mode, the host controller is allowed to process the queue head and take the actions indicated above, or it may wait until the queue head is visited in the

Table 11-100. Interrupt IN/OUT Do Complete Split State Execution Criteria

		normal processing mode. Regardless, the host controller must not execute a start-split in the context of a executing in a *Recovery Path* mode.

Managing QH.FrameTag Field

The *QH.FrameTag* field in a queue head is completely managed by the host controller. The rules for setting *QH.FrameTag* are simple:

- Rule 1: If transitioning from Do Start Split to Do Complete Split and the current value of *FRINDEX*[2:0] is 6 *QH.FrameTag* is set to *FRINDEX*[7:3] + 1. This accommodates split transactions whose start-split and complete-splits are in different *H-Frames* (case 2a, see Figure 11-73).
- Rule 2: If the current value of *FRINDEX*[2:0] is 7, *QH.FrameTag* is set to *FRINDEX*[7:3] + 1. This accommodates staying in Do Complete Split for cases 2a, 2b, and 2c (Figure 11-73).
- Rule 3: If transitioning from Do_Start Split to Do Complete Split and the current value of *FRINDEX*[2:0] is not 6, or currently in Do Complete Split and the current value of (*FRINDEX*[2:0]) is not 7, *FrameTag* is set to *FRINDEX*[7:3]. This accommodates all other cases (Figure 11-73).

11.3.4.3.12.2.6 Rebalancing the periodic schedule

System software must occasionally adjust a periodic queue head's S-mask and C-mask fields during operation.

This need occurs when adjustments to the periodic schedule create a new bandwidth budget and one or more queue head's are assigned new execution footprints (that is, new S-mask and C-mask values).

It is imperative that System software must not update these masks to new values in the midst of a split transaction. In order to avoid any race conditions with the update, the EHCI host controller provides a simple assist to system software. System software sets the *Inactivate-on-next-Transaction* (*I*) bit to a one to signal the host controller that it intends to update the S-mask and C-mask on this queue head. System software will then wait for the host controller to observe the *I-bit* is a one and transition the *Active* bit to a zero. The rules for how and when the host controller sets the *Active* bit to zero are enumerated below:

- If the *Active* bit is a zero, no action is taken. The host controller does not attempt to advance the queue when the *I-bit* is a one.
- If the *Active* bit is a one and the *SplitXState* is DoStart (regardless of the value of *S-mask*), the host controller will simply set *Active* bit to a zero. The host controller is

not required to write the transfer state back to the *current* qTD. Note that if the *S-mask* indicates that a start-split is scheduled for the current micro-frame, the host controller must not issue the start-split bus transaction. It must set the *Active* bit to zero.

System software must save transfer state before setting the *I-bit* to a one. This is required so that it can correctly determine what transfer progress (if any) occurred after the *I-bit* was set to a one and the host controller executed its final bus-transaction and set *Active* to a zero.

After system software has updated the S-mask and C-mask, it must then reactivate the queue head. Because the *Active* bit and the *I-bit* cannot be updated with the same write, system software needs to use the following algorithm to coherently re-activate a queue head that has been stopped via the *I-bit*.

1. Set the *Halted* bit to a one, then
2. Set the *I-bit* to a zero, then
3. Set the *Active* bit to a one and the *Halted* bit to a zero in the same write.

Setting the *Halted* bit to a one inhibits the host controller from attempting to advance the queue between the time the *I-bit* goes to a zero and the *Active* bit goes to a one.

11.3.4.3.12.3 Split Transaction Isochronous

Full-speed isochronous transfers are managed using the split-transaction protocol through a USB 2.0 transaction translator in a USB2.0 Hub. The EHCI controller utilizes siTD data structure to support the special requirements of isochronous split-transactions.

This data structure uses the scheduling model of isochronous TDs (iTD, Section Isochronous (High-Speed) Transfer Descriptor (iTD)) (see Section Managing Isochronous Transfers Using iTDs for the operational model of iTDs) with the contiguous data feature provided by queue heads. This simple arrangement allows a single isochronous scheduling model and adds the additional feature that all data received from the endpoint (per split transaction) must land into a contiguous buffer.

11.3.4.3.12.3.1 Split Transaction Scheduling Mechanisms for Isochronous

Full-speed isochronous transactions are managed through a transaction translator's periodic pipeline. As with full- and low-speed interrupt, system software manages each transaction translator's periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each full-speed isochronous endpoint occur.

The requirements described in Section Split Transaction Scheduling Mechanisms for Interrupt apply. The following figure illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule. The ^{S}X and ^{C}X labels indicate micro-frames where software can schedule start- and complete-splits (respectively). The *H-Frame* boundaries are marked with a large, solid bold vertical line. The *B-Frame* boundaries are marked with a large, bold, dashed line. The bottom of the figure illustrates the relationship of an siTD to the *H-Frame*.

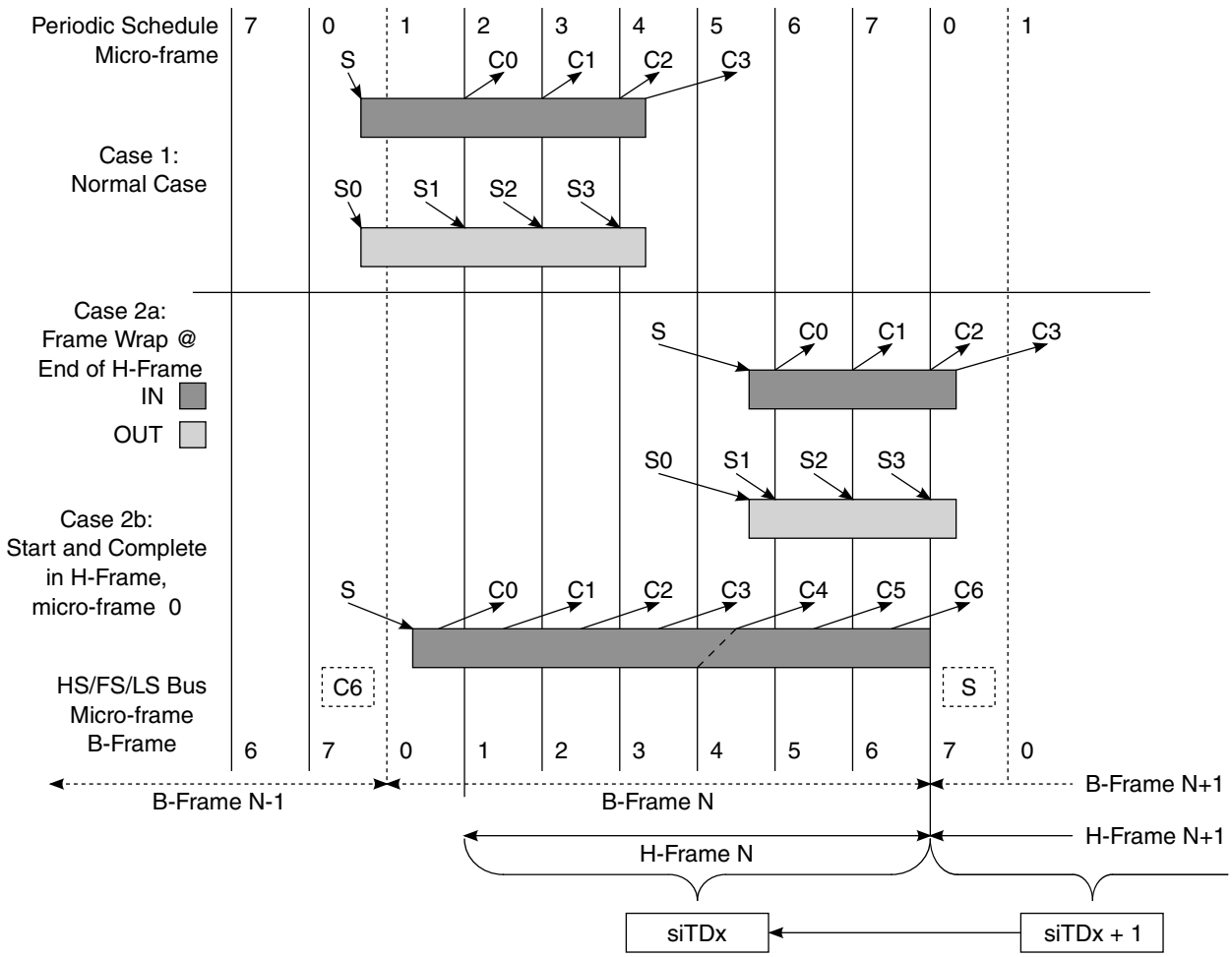


Figure 11-77. Split Transaction, Isochronous Scheduling Boundary Conditions

When the endpoint is an isochronous OUT, there are only start-splits, and no complete-splits. When the endpoint is an isochronous IN, there is at most one start-split and one to *N* complete-splits. The scheduling boundary cases are:

- *Case 1*: The entire split transaction is completely bounded by an *H-Frame*. For example: the start-splits and complete-splits are all scheduled to occur in the same *H-Frame*.

- *Case 2a*: This boundary case is where one or more (at most two) complete-splits of a split transaction IN are scheduled across an *H-Frame* boundary. This can only occur when the split transaction has the possibility of moving data in *B-Frame,* micro-frames 6 or 7 (*H-Frame* micro-frame 7 or 0). When an *H-Frame* boundary wrap condition occurs, the scheduling of the split transaction spans more than one location in the periodic list.(For example, it takes two siTDs in adjacent periodic frame list locations to fully describe the scheduling for the split transaction.)
- Although the scheduling of the split transaction may take two data structures, all of the complete-splits for each full-speed IN isochronous transaction must use only one data pointer. For this reason, siTDs contain a back pointer, the use of which is described below.
- Software must never schedule full-speed isochronous OUTs across an *H-Frame* boundary.
- *Case 2b*: This case can only occur for a very large isochronous IN. It is the only allowed scenario where a start-split and complete-split for the same endpoint can occur in the same micro-frame. Software must enforce this rule by scheduling the large transaction first. Large is defined to be anything larger than 579 byte maximum packet size.

A subset of the same mechanisms employed by full- and low-speed interrupt queue heads are employed in siTDs to schedule and track the portions of isochronous split transactions. The following fields are initialized by system software to instruct the host controller when to execute portions of the split transaction protocol.

- *SplitXState*. This is a single bit residing in the *Status* field of an siTD (see Figure 11-78). This bit is used to track the current state of the split transaction. The rules for managing this bit are described in Section Split Transaction Execution State Machine for Interrupt .
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 11-77, case one, the *S-mask* would have a value of 00000001b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates Do Start Split, and the current micro-frame as indicated by USB_n_FRINDEX[2:0] is 0, then execute a start-split transaction.
- *Frame C-mask.* This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 11-77, case one, the *C-mask* would have a value of 00111100b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates Do

Complete Split, and the current micro-frame as indicated by *USB_n_FRINDEX*[2:0] is 2, 3, 4, or 5, then execute a complete-split transaction.

- *Back Pointer*. This field in a siTD is used to complete an IN split-transaction using the previous *H-Frame*'s siTD. This is only used when the scheduling of the complete-splits span an *H-Frame* boundary.

There exists a one-to-one relationship between a high-speed isochronous split transaction (including all start- and complete-splits) and one full-speed isochronous transaction. An siTD contains (amongst other things) buffer state and split transaction scheduling information. An siTD's buffer state always maps to one full-speed isochronous data payload. This means that for any full-speed transaction payload, a single siTD's data buffer must be used. This rule applies to both IN an OUTs. An siTD's scheduling information usually also maps to one high-speed isochronous split transaction. The exception to this rule is the *H-Frame* boundary wrap cases mentioned above.

The siTD data structure describes at most, one frame's worth of high-speed transactions and that description is strictly bounded within a frame boundary. The figure below illustrates some examples. On the top are examples of the full-speed transaction footprints for the boundary scheduling cases described above. In the middle are time-frame references for both the *B-Frames* (HS/FS/LS Bus) and the *H-Frames*. On the bottom is illustrated the relationship between the scope of an siTD description and the time references. Each *H-Frame* corresponds to a single location in the periodic frame list. The implication is that each siTD is reachable from a single periodic frame list location at a time.

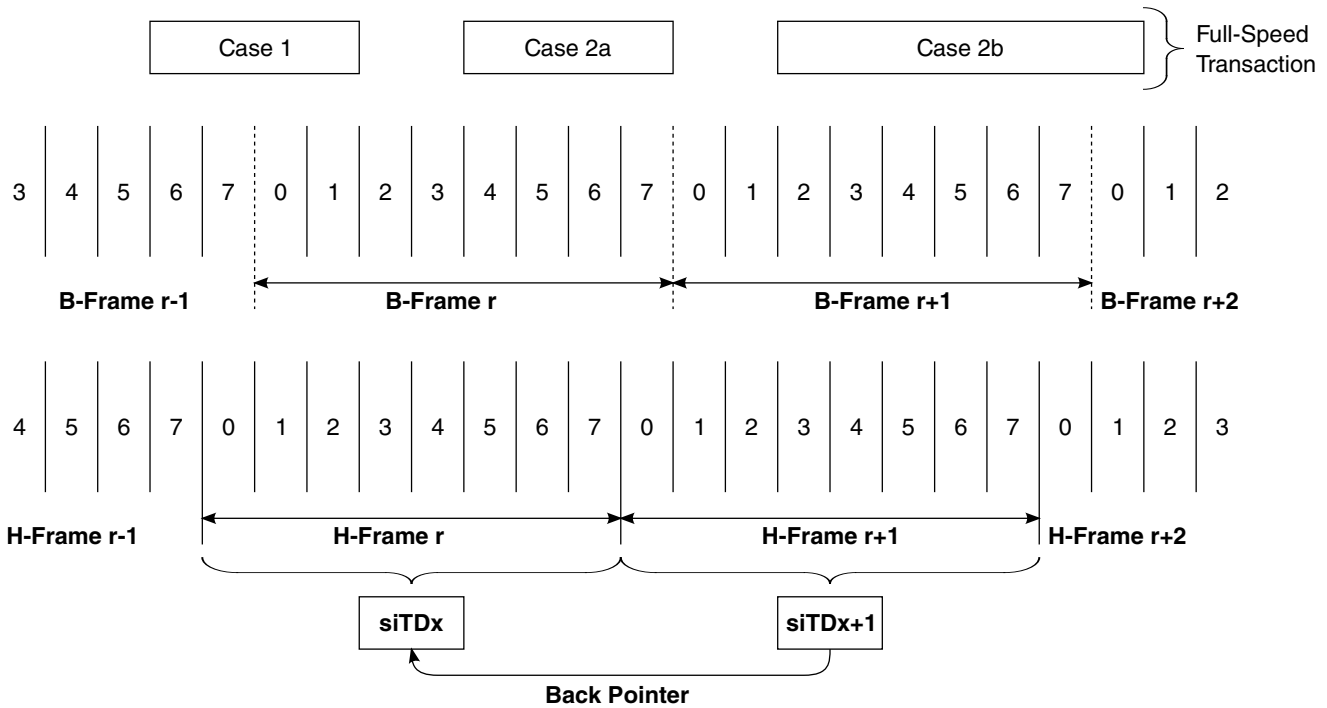


Figure 11-78. siTD Scheduling Boundary Examples

Each case is described below:

- *Case 1*: One siTD is sufficient to describe and complete the isochronous split transaction because the whole isochronous split transaction is tightly contained within a single *H-Frame*.
- *Case 2a, 2b*: Although both INs and OUTs can have these footprints, OUTs always take only one siTD to schedule. However, INs (for these boundary cases) require two siTDs to complete the scheduling of the isochronous split transaction $siTD_X$ is used to always issue the start-split and the first *N* complete-splits. The full-speed transaction (for these cases) can deliver data on the full-speed bus segment during micro-frame 7 of $H\text{-}Frame_{Y+1}$, or micro-frame 0 of $H\text{-}Frame_{Y+2}$. The complete splits are scheduled using $siTD_{X+2}$ (not shown). The complete-splits to extract this data must use the buffer pointer from $siTD_{X+1}$. The only way for the host controller to reach $siTD_{X+1}$ from $H\text{-}Frame_{Y+2}$is to use $siTD_{X+2}$'s back pointer. The host controller rules for when to use the back pointer are described is Section Periodic Isochronous - Do Complete Split .

Software must apply the following rules when calculating the schedule and linking the schedule data structures into the periodic schedule:

- Software must ensure that an isochronous split-transaction is started so that it will complete before the end of the *B-Frame*.
- Software must ensure that for a single full-speed isochronous endpoint, there is never a start-split and complete-split in *H-Frame, micro-frame 1*. This is mandated as a rule so that case 2a and case 2b can be discriminated. According to the core USB specification, the long isochronous transaction illustrated in Case 2b, could be scheduled so that the start-split was in micro-frame 1 of *H-Frame* N and the last complete-split would need to occur in micro-frame 1 of H-Frame N+1. However, it is impossible to discriminate between cases 2a and case 2b, which has significant impact on the complexity of the host controller.

11.3.4.3.12.3.2 Tracking Split Transaction Progress for Isochronous Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where device to host data is lost. Isochronous endpoints do not employ the concept of a halt on error, however the host is required to identify and report per-packet errors observed in the data stream. This includes schedule traversal problems (skipped micro-frames), timeouts and corrupted data received.

In similar kind to interrupt split-transactions, the portions of the split transaction protocol must execute in the micro-frames they are scheduled. The queue head data structure used to manage full- and low-speed interrupt has several mechanisms for tracking when portions of a transaction have occurred. Isochronous transfers use siTDs, for their transfers, and the data structures are only reachable via the schedule in the exact micro-frame in which they are required (so all the mechanism employed for tracking in queue heads is not required for siTDs). Software has the option of reusing siTD several times in the complete periodic schedule. However, it must ensure that the results of split transaction N are consumed and the siTD reinitialized (activated) before the host controller gets back to the siTD (in a future micro-frame).

Split-transaction isochronous OUTs utilize a low-level protocol to indicate which portions of the split transaction data have arrived. Control over the low-level protocol is exposed in an siTD via the fields *Transaction Position* (*TP*) and *Transaction Count* (*T-count*). If the entire data payload for the OUT split transaction is larger than 188 bytes, there will be more than one start-split transaction, each of which require proper annotation. If host hold-offs occur, then the sequence of annotations received from the host will not be complete, which is detected and handled by the transaction translator. See Section Periodic Isochronous - Do Start Split for a description on how these fields are used during a sequence of start-split transactions.

The fields *siTD.T-Count* and *siTD.TP* are used by the host controller to drive and sequence the transaction position annotations. It is the responsibility of system software to properly initialize these fields in each siTD. Once the budget for a split-transaction

isochronous endpoint is established, *S-mask*, *T-Count*, and *TP* initialization values for all the siTD associated with the endpoint are constant. They remain constant until the budget for the endpoint is recalculated by software and the periodic schedule adjusted.

For IN-endpoints, the transaction translator simply annotates the response data packets with enough information to allow the host controller to identify the last data. As with split transaction Interrupt, it is the host controller's responsibility to detect when it has missed an opportunity to execute a complete-split. The following field in the siTD is used to track and detect errors in the execution of a split transaction for an IN isochronous endpoint.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets a bit for each complete-split executed. The bit position is determined by the micro-frame (USB_n_FRINDEX[2:0]) number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a complete-split transaction. If the previous complete-splits have not been executed, then it means one (or more) have been skipped and data has potentially been lost. System software is required to initialize this field to zero before setting an siTD's *Active* bit to a one.

If a transaction translator returns with the final data before all of the complete-splits have been executed, the state of the transfer is advanced so that the remaining complete-splits are not executed. Refer to Section Asynchronous - Do Complete Split for a description on how the state of the transfer is advanced. It is important to note that an IN siTD is retired based solely on the responses from the Transaction Translator to the complete-split transactions. This means, for example, that it is possible for a transaction translator to respond to a complete-split with an MDATA PID. The number of bytes in the MDATA's data payload could cause the siTD field *Total Bytes to Transfer* to decrement to zero. This response can occur, before all of the scheduled complete-splits have been executed. In other interface, data structures (for example, high-speed data streams through queue heads), the transition of *Total Bytes to Transfer* to zero signals the end of the transfer and results in setting of the *Active* bit to zero. However, in this case, the result has not been delivered by the Transaction Translator and the host must continue with the next complete-split transaction to extract the residual transaction state. This scenario occurs because of the pipeline rules for a Transaction Translator (see Chapter 11 of the Universal Serial Bus Revision 2.0). In summary the periodic pipeline rules require that on a micro-frame boundary, the Transaction Translator will hold the final two bytes received (if it has not seen an End Of Packet (EOP)) in the full-speed bus pipe stage and give the

remaining bytes to the high-speed pipeline stage. At the micro-frame boundary, the Transaction Translator could have received the entire packet (including both CRC bytes) but not received the packet EOP. In the next micro-frame, the Transaction Translator will respond with an MDATA and send all of the data bytes (with the two CRC bytes being held in the full-speed pipeline stage). This could cause the siTD to decrement its *Total Bytes to Transfer* field to zero, indicating it has received all expected data. The host must still execute one more (scheduled) complete-split transaction in order to extract the results of the full-speed transaction from the Transaction Translator (for example, the Transaction Translator may have detected a CRC failure, and this result must be forwarded to the host).

If the host experiences hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous OUT, then the protocol to the transaction translator will not be consistent and the transaction translator will detect and react to the problem. Likewise, for host hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous IN, the *C-prog-mask* is used by the host controller to detect errors. However, if the host experiences a hold-off that causes it to skip all of an siTD, or an siTD expires during a host hold off (for example, a hold-off occurs and the siTD is no longer reachable by the host controller in order for it to report the hold-off event), then system software must detect that the siTDs have not been processed by the host controller (that is, state not advanced) and report the appropriate error to the client driver.

11.3.4.3.12.3.3 Split Transaction Execution State Machine for Isochronous

In the following presentation, all references to micro-frame are in the context of a micro-frame within an *H-Frame.*

If the *Active* bit in the *Status* byte is a zero, the host controller will ignore the siTD and continue traversing the periodic schedule. Otherwise the host controller will process the siTD as specified below. A split transaction state machine is used to manage the split-transaction protocol sequence. The host controller uses the fields defined in Section Tracking Split Transaction Progress for Interrupt Transfers , plus the variable *cMicroFrameBit* defined in Section Split Transaction Execution State Machine for Interrupt to track the progress of an isochronous split transaction. The figure below illustrates the state machine for managing an siTD through an isochronous split transaction. Bold, dotted circles denote the state of the *Active* bit in the *Status* field of a siTD. The Bold, dotted arcs denote the transitions between these states. Solid circles denote the states of the split transaction state machine and the solid arcs denote the transitions between these states. Dotted arcs and boxes reference actions that take place either as a result of a transition or from being in a state.

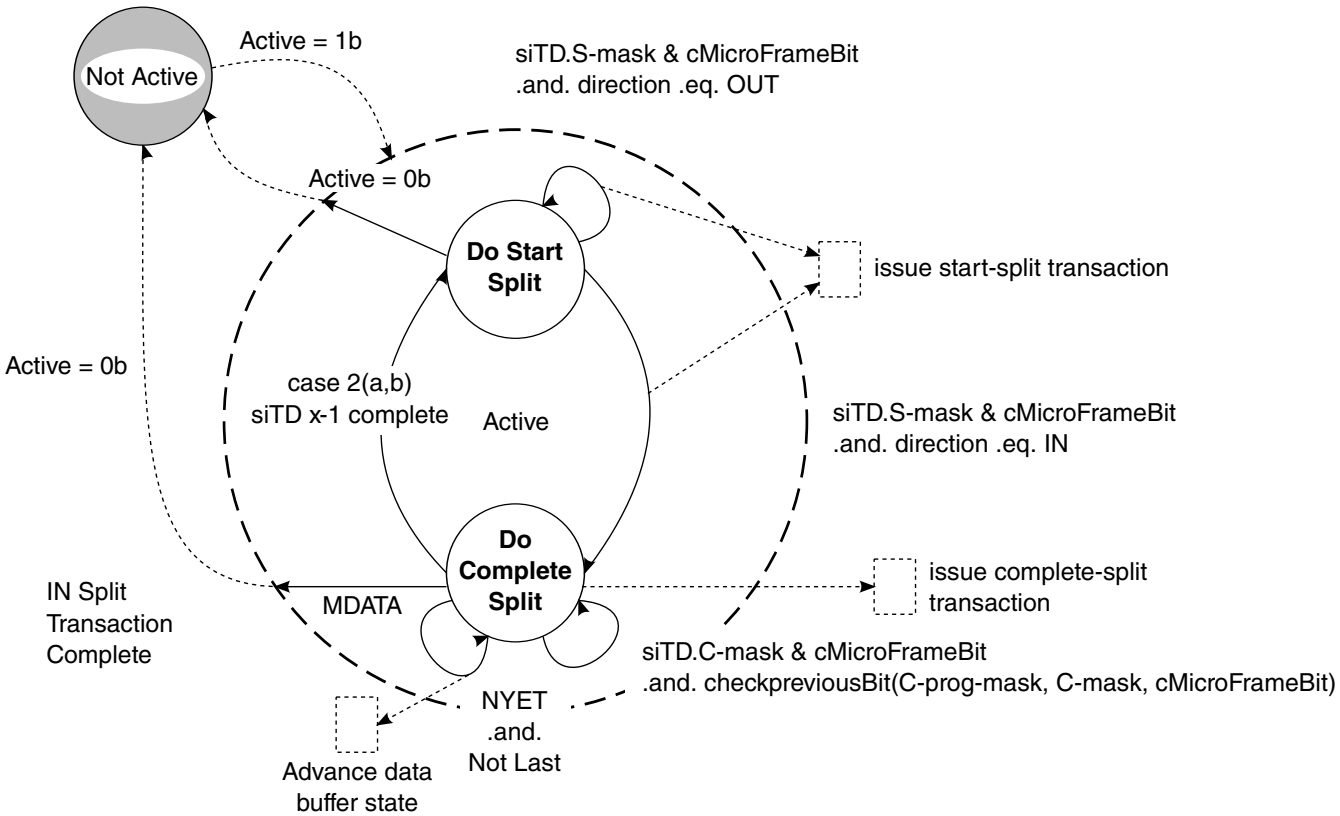


Figure 11-79. Split Transaction State Machine for Isochronous

11.3.4.3.12.3.4 Periodic Isochronous - Do Start Split

Isochronous split transaction OUTs use only this state.

An siTD for a split-transaction isochronous IN is either initialized to this state, or the siTD transitions to this state from Do Complete Split when a case 2a (IN) or 2b scheduling boundary isochronous split-transaction completes.

Each time the host controller reaches an active siTD in this state, it checks the *siTD.S-mask* against *cMicroFrameBit*. If there is a one in the appropriate position, the siTD will execute a start-split transaction. By definition, the host controller cannot *reach* an siTD at the wrong time. If the *I/O* field indicates an IN, then the start-split transaction includes only the extended token plus the full-speed token. Software must initialize the *siTD.Total Bytes To Transfer* field to the number of bytes expected. This is usually the maximum packet size for the full-speed endpoint. The host controller exits this state when the start-split transaction is complete.

The remainder of this section is specific to an isochronous OUT endpoint (that is, the *I/O* field indicates an OUT). When the host controller executes a start-split transaction for an isochronous OUT it includes a data payload in the start-split transaction. The memory

buffer address for the data payload is constructed by concatenating *siTD.Current Offset* with the page pointer indicated by the page selector field (*siTD.P*). A zero in this field selects Page 0 and a 1 selects Page 1. During the start-split for an OUT, if the data transfer crosses a page boundary during the transaction, the host controller must detect the page cross, update the *siTD.P*-bit from a zero to a one, and begin using the *siTD.Page 1* with *siTD.Current Offset* as the memory address pointer. The field *siTD.TP* is used to annotate each start-split transaction with the indication of which part of the split-transaction data the current payload represents (ALL, BEGIN, MID, END). In all cases the host controller simply uses the value in *siTD.TP* to mark the start-split with the correct transaction position code.

T-Count is always initialized to the number of start-splits for the current frame. *TP* is always initialized to the first required transaction position identifier. The scheduling boundary case (see Figure 11-78) is used to determine the initial value of *TP*. The initial cases are summarized in the following table.

Table 11-101. Initial Conditions for OUT siTD's TP and T-count Fields

Case	T-count	TP	Description
1, 2a	=1	ALL	When the OUT data payload is less than (or equal to) 188 bytes, only one start-split is required to move the data. The one start-split must be marked with an ALL.
1, 2a	!=1	BEGIN	When the OUT data payload is greater than 188 bytes more than one start-split must be used to move the data. The initial start-split must be marked with a BEGIN.

After each start-split transaction is complete, the host controller updates *T-Count* and *TP* appropriately so that the next start-split is correctly annotated.

The table below illustrates all of the *TP* and *T-count* transitions, which must be accomplished by the host controller.

.

Table 11-102. Transaction Position (TP)/Transaction Count (T-Count) Transition Table

TP	T-count next	TP next	Description
ALL	0	N/A	Transition from ALL, to done.
BEGIN	1	END	Transition from BEGIN to END. Occurs when *T-count* starts at 2.
BEGIN	!=1	MID	Transition from BEGIN to MID. Occurs when *T-count* starts at greater than 2.
MID	!=1	MID	*TP* stays at MID while *T-count* is not equal to 1 (that is, greater than 1). This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 3.
MID	1	END	Transition from MID to END. This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 2.

The start-split transactions do not receive a handshake from the transaction translator, so the host controller always advances the transfer state in the siTD after the bus transaction is complete. To advance the transfer state the following operations take place:

- The *siTD.Total Bytes To Transfer* and the *siTD.Current Offset* fields are adjusted to reflect the number of bytes transferred.
- The *siTD.P* (page selector) bit is updated appropriately.
- The *siTD.TP* and *siTD.T-count* fields are updated appropriately as defined in Table 11-102.

These fields are then written back to the memory based siTD. The *S-mask* is fixed for the life of the current budget. As mentioned above, *TP* and *T-count* are set specifically in each siTD to reflect the data to be sent from this siTD. Therefore, regardless of the value of *S-mask*, the actual number of start-split transactions depends on *T-count* (or equivalently, *Total Bytes to Transfer*). The host controller must set the *Active* bit to a zero when it detects that all of the schedule data has been sent to the bus. The preferred method is to detect when *T-Count* decrements to zero as a result of a start-split bus transaction. Equivalently, the host controller can detect when *Total Bytes to Transfer* decrements to zero. Either implementation must ensure that if the initial condition is *Total Bytes to Transfer* equal to zero and *T-count* is equal to a one, then the host controller will issue a single start-split, with a zero-length data payload. Software must ensure that *TP*, *T-count* and *Total Bytes to Transfer* are set to deliver the appropriate number of bus transactions from each siTD. An inconsistent combination will yield undefined behavior.

If the host experiences hold-offs that cause the host controller to skip start-split transactions for an OUT transfer, the state of the transfer will not progress appropriately. The transaction translator will observe protocol violations in the arrival of the start-splits for the OUT endpoint (that is, the transaction position annotation will be incorrect as received by the transaction translator).

Example scenarios are described in Section Split Transaction for Isochronous - Processing Examples .

A host controller implementation can optionally track the progress of an OUT split transaction by setting appropriate bits in the *siTD.C-prog-mask* as it executes each scheduled start-split. The *checkPreviousBit*() algorithm defined in Periodic Isochronous - Do Complete Split can be used prior to executing each start-split to determine whether start-splits were skipped. The host controller can use this mechanism to detect missed micro-frames. It can then set the siTD's *Active* bit to zero and stop execution of this siTD. This saves on both memory and high-speed bus bandwidth.

11.3.4.3.12.3.5 Periodic Isochronous - Do Complete Split

This state is only used by a split-transaction isochronous IN endpoint.

This state is entered unconditionally from the Do Start State after a start-split transaction is executed for an IN endpoint. Each time the host controller visits an siTD in this state, it conducts a number of tests to determine whether it should execute a complete-split transaction. The individual tests are listed below. The sequence they are applied depends on which micro-frame the host controller is currently executing which means that the tests might not be applied until after the siTD referenced from the back pointer has been fetched.

- Test A. *cMicroFrameBit* is bit-wise anded with *siTD.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame. This test is always applied to a newly fetched siTD that is in this state.
- Test B. The *siTD.C-prog-mask* bit vector is checked to determine whether the previous complete splits have been executed. An example algorithm is below (this is slightly different than the algorithm used in Section Periodic Isochronous - Do Complete Split). The sequence in which this test is applied depends on the current value of USB_n_FRINDEX[2:0]. If USB_n_FRINDEX[2:0] is 0 or 1, it is not applied until the back pointer has been used. Otherwise it is applied immediately.

```
Algorithm Boolean CheckPreviousBit(siTD.C-prog-mask, siTD.C-mask, cMicroFrameBit)
Begin
          Boolean rvalue = TRUE;
          previousBit = cMicroFrameBit rotate-right(1)
          -- Bit-wise anding previousBit with C-mask indicates whether there was an intent
          -- to send a complete split in the previous micro-frame. So, if the
          -- 'previous bit' is set in C-mask, check C-prog-mask to make sure it
          -- happened.
          if previousBit bitAND siTD.C-mask then
                    if not (previousBit bitAND siTD.C-prog-mask) then
                              rvalue = FALSE
                    End if
          End if
          Return rvalue
End Algorithm
```

If Test A is true and USB_n_FRINDEX[2:0] is zero or one, then this is a case 2a or 2b scheduling boundary (see Figure 11-77). See Section Periodic Isochronous - Do Complete Split for details in handling this condition.

If Test A and Test B evaluate to true, then the host controller will execute a complete-split transaction using the transfer state of the current siTD. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. The transfer state is advanced based on the completion status of the complete-split transaction. To advance the transfer state of an IN siTD, the host controller must:

- Decrement the number of bytes received from *siTD.Total Bytes To Transfer,*
- Adjust *siTD.Current Offset* by the number of bytes received,

- Adjust *siTD.P* (page selector) field if the transfer caused the host controller to use the next page pointer, and
- Set any appropriate bits in the *siTD.Status* field, depending on the results of the transaction.

Note that if the host controller encounters a condition where *siTD.Total Bytes To Transfer* is zero, and it receives more data, the host controller must not write the additional data to memory. The *siTD.Status.Active* bit must be set to zero and the *siTD.Status.Babble Detected* bit must be set to a one. The fields *siTD.Total Bytes To Transfer*, *siTD.Current Offset*, and *siTD.P* (page selector) are not required to be updated as a result of this transaction attempt.

The host controller must accept (assuming good data packet CRC and sufficient room in the buffer as indicated by the value of *siTD.Total Bytes To Transfer*) MDATA and DATA0/1 data payloads up to and including 192 bytes. A host controller implementation may optionally set *siTD.Status Active* to a zero and *siTD.Status.Babble Detected* to a one when it receives and MDATA or DATA0/1 with a data payload of more than 192 bytes. The following responses have the noted effects:

- ERR. The full-speed transaction completed with a time-out or bad CRC and this is a reflection of that error to the host. The host controller sets the *ERR* bit in the *siTD.Status* field and sets the *Active* bit to a zero.
- Transaction Error (XactErr). The complete-split transaction encounters a Timeout, CRC16 failure, etc. The *siTD.Status* field *XactErr* field is set to a one and the complete-split transaction must be retried immediately. The host controller must use an internal error counter to count the number of retries as a counter field is not provided in the siTD data structure. The host controller will not retry more than two times. If the host controller exhausts the retries or the end of the micro-frame occurs, the *Active* bit is set to zero.
- DATAx (0 or 1). This response signals that the final data for the split transaction has arrived. The transfer state of the siTD is advanced and the *Active* bit is set to a zero. If the *Bytes To Transfer* field has not decremented to zero (including the reception of the data payload in the DATAx response), then less data than was expected, or allowed for was actually received. This *short packet* event does not set the USBINT status bit in the USBSTS register to a one. The host controller will not detect this condition.
- NYET (and Last). On each NYET response, the host controller also checks to determine whether this is the last complete-split for this split transaction. Last was defined in Section Periodic Isochronous - Do Complete Split . If it is the last complete-split (with a NYET response), then the transfer state of the siTD is not advanced (never received any data) and the *Active* bit is set to a zero. No bits are set in the *Status* field because this is essentially a skipped transaction. The transaction translator must have responded to all the scheduled clompete-splits with NYETs,

meaning that the start-split issued by the host controller was not received. This result should be interpreted by system software as if the transaction was completely skipped. The test for whether this is the last complete split can be performed by XORing C-mask with C-prog-mask. A zero result indicates that all complete-splits have been executed.

- MDATA (and Last). See above description for testing for Last. This can only occur when there is an error condition. Either there has been a babble condition on the full-speed link, which delayed the completion of the full-speed transaction, or software set up the *S-mask* and/or *C-mask*s incorrectly. The host controller must set *XactErr* bit to a one and the *Active* bit is set to a zero.
- NYET (and not Last). See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask*) and stay in this state.
- MDATA (and not Last). The transaction translator responds with an MDATA when it has partial data for the split transaction. For example, the full-speed transaction data payload spans from micro-frame X to X+1 and during micro-frame X, the transaction translator will respond with an MDATA and the data accumulated up to the end of micro-frame X. The host controller advances the transfer state to reflect the number of bytes received.

If Test A succeeds, but Test B fails, it means that one or more of the complete-splits have been skipped. The host controller sets the *Missed Micro-Frame* status bit and sets the *Active* bit to a zero.

11.3.4.3.12.3.6 Complete-Split for Scheduling Boundary Cases 2a, 2b

Boundary cases 2a and 2b (INs only) (see Figure 11-77) require that the host controller use the transaction state context of the previous siTD to finish the split transaction. The table below enumerates the transaction state fields.

Table 11-103. Summary siTD Split Transaction State

Buffer State	Status	Execution Progress
Total Bytes To Transfer P (page select) Current Offset TP (transaction position) T-count (transaction count)	All bits in the status field	C-prog-mask

NOTE

TP and *T-count* are used only for Host to Device (OUT) endpoints.

If software has budgeted the schedule of this data stream with a frame wrap case, then it must initialize the *siTD.Back Pointer* field to reference a valid siTD and will have the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer*

field set to a zero. Otherwise, software must set the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer* field to a one. The host controller's rules for interpreting when to use the *siTD.Back Pointer* field are listed below. These rules apply only when the siTD's *Active* bit is a one and the *SplitXState* is Do Complete Split.

- When cMicroFrameBit is a 1h and the siTDX.Back Pointer.T-bit is a zero, or
- If cMicroFrameBit is a 2h and siTDX.S-mask[0] is a zero

When either of these conditions apply, then the host controller must use the transaction state from $siTD_{X-1}$.

In order to access $siTD_{X-1}$, the host controller reads on-chip the siTD referenced from $siTD_X$.*Back Pointer*.

The host controller must save the entire state from $siTD_X$ while processing $siTD_{X-1}$. This is to accommodate for case 2b processing. The host controller must not recursively walk the list of *siTD.Back Pointers*.

If $siTD_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is Do Complete Split), then both Test A and Test B are applied as described above. If these criteria to execute a complete-split are met, the host controller executes the complete split and evaluates the results as described above. The transaction state (see Table 11-103) of $siTD_{X-1}$ is appropriately advanced based on the results and written back to memory. If the resultant state of $siTD_{X-1}$'s *Active* bit is a one, then the host controller returns to the context of $siTD_X$, and follows its next pointer to the next schedule item. No updates to $siTD_X$ are necessary.

If $siTD_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is Do Start Split), then the host controller must set *Active* bit to a zero and *Missed Micro-Frame* status bit to a one and the resultant status written back to memory.

If $siTD_{X-1}$*'sActive* bit is a zero, (because it was zero when the host controller first visited $siTD_{X-1}$ via $siTD_X$*'s* back pointer, it transitioned to zero as a result of a detected error, or the results of $siTD_{X-1}$*'s* complete-split transaction transitioned it to zero), then the host controller returns to the context of $siTD_X$ and transitions its *SplitXState* to Do Start Split. The host controller then determines whether the case 2b start split boundary condition exists (that is, if *cMicroframeBit* is a 1b and $siTD_X$.*S-mask*[0] is a 1b). If this criterion is met the host controller immediately executes a start-split transaction and appropriately advances the transaction state of $siTD_X$, then follows $siTD_X$.*Next Pointer* to the next schedule item. If the criterion is not met, the host controller simply follows $siTD_X$.*Next Pointer* to the next schedule item. Note that in the case of a 2b boundary case, the split-transaction of $siTD_{X-1}$ will have its *Active* bit set to zero when the host controller returns

to the context of $siTD_X$. Also, note that software should not initialize an siTD with *C-mask* bits 0 and 1 set to a one and an *S-mask* with bit zero set to a one. This scheduling combination is not supported and the behavior of the host controller is undefined.

11.3.4.3.12.3.7 Split Transaction for Isochronous - Processing Examples

There is an important difference between how the hardware/software manages the isochronous split transaction state machine and how it manages the asynchronous and interrupt split transaction state machines.

The asynchronous and interrupt split transaction state machines are encapsulated within a single queue head. The progress of the data stream depends on the progress of each split transaction. In some respects, the split-transaction state machine is sequenced via the Execute Transaction queue head traversal state machine (see Figure 11-70).

Isochronous is a pure time-oriented transaction/data stream. The interface data structures are optimized to efficiently describe transactions that need to occur at specific times. The isochronous split-transaction state machine must be managed across these time-oriented data structures. This means that system software must correctly describe the scheduling of split-transactions across more than one data structure.

Then the host controller must make the appropriate state transitions at the appropriate times, in the correct data structures.

For example, the table below illustrates a couple of frames worth of scheduling required to schedule a case 2a full-speed isochronous data stream.

Table 11-104. Example Case 2a - Software Scheduling siTDs for an IN Endpoint

siTDX		Micro-Frames								Initial
#	Masks	0	1	2	3	4	5	6	7	SplitXState
X	S-Mask	-	-	-	-	1	-	-	-	Do Start Split
	C-Mask	1	1	-	-	-	-	1	1	
X+1	S-Mask	-	-	-	-	1	-	-	-	Do Complete Split
	C-Mask	1	1					1	1	
X+2	S-Mask	-	-	-	-	1	-	-	-	Do Complete Split
	C-Mask	1	1					1	1	
X+3	S-Mask	Repeats previous pattern								Do Complete Split
	C-Mask									

This example shows the first three siTDs for the transaction stream. Because this is the case-2a frame-wrap case, *S-masks* of all siTDs for this endpoint have a value of 10h (a one bit in micro-frame 4) and *C-mask* value of C3h (one-bits in micro-frames 0,1, 6 and 7). Additionally, sofware ensures that the *Back Pointer* field of each siTD references the appropriate siTD data structure (and the *Back PointerT-bits* are set to zero).

The initial *SplitXState* of the first siTD is Do Start Split. The host controller will visit the first siTD eight times during frame X. The C-mask bits in micro-frames 0 and 1 are ignored because the state is Do Start Split. During micro-frame 4, the host controller determines that it can run a start-split (and does) and changes *SplitXState* to Do Complete Split. During micro-frames 6 and 7, the host controller executes complete-splits. Notice the siTD for frame X+1 has its *SplitXState* initialized to Do Complete Split. As the host controller continues to traverse the schedule during *H-Frame* X+1, it will visit the second siTD eight times. During micro-frames 0 and 1 it will detect that it must execute complete-splits.

During *H-Frame* X+1, micro-frame 0, the host controller detects that $siTD_{X+1}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+1}$ and fetches $siTD_X$. It executes the complete split transaction using the transaction state of $siTD_X$. If the $siTD_X$ split transaction is complete, siTD's *Active* bit is set to zero and results written back to $siTD_X$. The host controller retains the fact that $siTD_X$ is retired and transitions the *SplitXState* in the $siTD_{X+1}$ to Do Start Split. At this point, the host controller is prepared to execute the start-split for $siTD_{X+1}$ when it reaches micro-frame 4. If the split-transaction completes early (transaction-complete is defined in Section Periodic Isochronous - Do Complete Split), that is, before all the scheduled complete-splits have been executed, the host controller will transition $siTD_X$.*SplitXState* to Do Start Split early and naturally skip the remaining scheduled complete-split transactions. For this example, $siTD_{X+1}$ does not receive a DATA0 response until *H-Frame* X+2, micro-frame 1.

During *H-Frame X+2,* micro-frame 0, the host controller detects that $siTD_{X+2}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. As described above, it executes another split transaction, receives an MDATA response, updates the transfer state, but does not modify the *Active* bit. The host controller returns to the context of $siTD_{X+2}$, and traverses its next pointer without any state change updates to $siTD_{X+2}$. S

During *H-Frame X+2*, micro-frame 1, the host controller detects $siTD_{X+2}$'s *S-mask[0]* is a zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. It executes another complete-split transaction, receives a DATA0 response, updates the transfer state and sets the *Active* bit to a zero. It returns to the state of $siTD_{X+2}$ and changes its *SplitXState* to Do Start Split. At this point, the host controller is prepared to execute start-splits for $siTD_{X+2}$ when it

reaches micro-frame 4. <TBD… describe how software detects that there was missing micro-frames (don't think we care about missing out micro-frames. There is enough residual state to identify than not all transactions were executed.).

11.3.4.3.13 Host Controller Pause

When the host controller's *HCHalted* bit in the USBSTS register is a zero, the host controller is sending SOF (Start OF Frame) packets down all enabled ports.

When the schedules are enabled, the EHCI host controller will access the schedules in main memory each micro-frame. This constant pinging of main memory is known to create ARM platform power management problems for mobile systems. Specifically, mobile systems aggressively manage the state of the ARM platform, based on recent history usage. In the more aggressive power saving modes, the ARM platform can disable its caches. Current PC architectures assume that bus-master accesses to main memory must be cache-coherent. So, when bus masters are busy touching memory, the ARM platform power management software can detect this activity over time and inhibit the transition of the ARM platform into its lowest power savings mode. USB controllers are bus-masters and the frequency at which they access their memory-based schedules keeps the ARM platform power management software from placing the ARM platform into its lowest power savings state.

USB Host controllers don't access main memory when they are suspended. However, there are a variety of reasons why placing the USB controllers into suspend won't work, but they are beyond the scope of this document. The base requirement is that the USB controller needs to be kept out of main memory, while at the same time, the USB bus is kept from going into suspend.

EHCI controllers provide a large-grained mechanism that can be manipulated by system software to change the memory access pattern of the host controller. System software can manipulate the schedule enable bits in the USBCMD register to turn on/off the scheduling traversal. A software heuristic can be applied to implement an on/off duty cycle that allows the USB to make reasonable progress and allow the ARM platform power management to get the ARM platform into its lowest power state. This method is not intended to be applied at all times to throttle USB, but should only be applied in very specific configurations and usage loads. For example, when only a keyboard or mouse is attached to the USB, the heuristic could detect times when the USB is attempting to move data only very infrequently and can adjust the duty cycle to allow the ARM platform to reach its low power state for longer periods of time. Similarly, it could detect increases in the USB load and adjust the duty cycle appropriately, even to the point where the schedules are never disabled. The assumption here is that the USB is moving data and the ARM platform will be required to process the data streams.

It is suggested that in order to provide a complete solution for the system, the companion host controllers should also provide a similar method to allow system software to inhibit the companion host controller from accessing its shared memory based data structures (schedule lists or otherwise).

11.3.4.3.14 Port Test Modes -Host Operational Model

EHCI host controllers must implement the port test modes Test J_State, Test K_State, Test_Packet, Test Force_Enable, and Test SE0_NAK as described in the USB Specification Revision 2.0.

The system is only allowed to test ports that are owned by the EHCI controller (for example, *CF-bit* is a one and *PortOwner* bit is a zero). System software is allowed to have at most one port in test mode at a time. Placing more than one port in test mode will yield undefined results. The required, per port test sequence is (assuming the *CF-bit* in the USB_n_CONFIGFLAG register is a one):

- Disable the periodic and asynchronous schedules by setting the *Asynchronous Schedule Enable* and *Periodic Schedule Enable* bits in the USBCMD register to a zero.
- Place all enabled root ports into the suspended state by setting the *Suspend* bit in each appropriate USB_n_PORTSC register to a one.
- Set the *Run/Stop* bit in the USBCMD register to a zero and wait for the *HCHalted* bit in the USBSTS register, to transition to a one. Note that an EHCI host controller implementation may optionally allow port testing with the *Run/Stop* bit set to a one. However, all host controllers must support port testing with *Run/Stop* set to a zero and *HCHalted* set to a one.
- Set the *Port Test Control* field in the port under test PORTSC register to the value corresponding to the desired test mode. If the selected test is Test_Force_Enable, then the *Run/Stop* bit in the USBCMD register must then be transitioned back to one, in order to enable transmission of SOFs out of the port under test.
- When the test is complete, system software must ensure the host controller is halted (*HCHalted* bit is a one) then it terminates and exits test mode by setting *HCReset* to a one.

11.3.4.3.15 Interrupts-Host Operational Model

The EHCI Host Controller hardware provides interrupt capability based on a number of sources.

There are several general groups of interrupt sources:

- Interrupts as a result of executing transactions from the schedule (success and error conditions),
- Host controller events (Port change events, etc.), and
- Host Controller error events

All transaction-based sources are maskable through the Host Controller's Interrupt Enable register (USBINTR, see Section). Additionally, individual transfer descriptors can be marked to generate an interrupt on completion. This section describes each interrupt source and the processing that occurs in response to the interrupt.

During normal operation, interrupts may be immediate or deferred until the next interrupt threshold occurs. The interrupt threshold is a tunable parameter via the *Interrupt Threshold Control* field in the USBCMD register. The value of this register controls when the host controller will generate an interrupt on behalf of normal transaction execution. When a transaction completes during an interrupt interval period, the interrupt signaling the completion of the transfer will not occur until the interrupt threshold occurs. For example, the default value is eight micro-frames. This means that the host controller will not generate interrupts any more frequently than once every eight micro-frames.

Section Host System Error details effects of a host system error.

If an interrupt has been scheduled to be generated for the current interrupt threshold interval, the interrupt is not signaled until after the status for the last complete transaction in the interval has been written back to host memory. This may sometimes result in the interrupt not being signaled until the next interrupt threshold.

Initial interrupt processing is the same, regardless of the reason for the interrupt. When an interrupt is signaled by the hardware, ARM platform control is transferred to host controller's USB interrupt handler. The precise mechanism to accomplish the transfer is OS specific. For this discussion it is just assumed that control is received. When the interrupt handler receives control, its first action is to reads the USBSTS (USB Status Register). It then acknowledges the interrupt by clearing all of the interrupt status bits by writing ones to these bit positions. The handler then determines whether the interrupt is due to schedule processing or some other event. After acknowledging the interrupt, the handler (via an OS-specific mechanism), schedules a deferred procedure call (DPC) which will execute later. The DPC routine processes the results of the schedule execution. The precise mechanisms used are beyond the scope of this document.

Note: the host controller is not required to de-assert a currently active interrupt condition when software sets the interrupt enables (in the USBINR register, see Section) to a zero. The only reliable method software should use for acknowledging an interrupt is by transitioning the appropriate status bits in the USBSTS register (Section) from a one to a zero.

11.3.4.3.15.1 Transfer/Transaction Based Interrupts

These interrupt sources are associated with transfer and transaction progress. They are all dependent on the next interrupt threshold.

11.3.4.3.15.1.1 Transaction Error

A transaction error is any error that caused the host controller to think that the transfer did not complete successfully.

The table below lists the events/responses that the host can observe as a result of a transaction. The effects of the error counter and interrupt status are summarized in the following paragraphs. Most of these errors set the *XactErr* status bit in the appropriate interface data structure.

There is a small set of protocol errors that relate only when executing a queue head and fit under the umbrella of a WRONG PID error that are significant to explicitly identify. When these errors occur, the *XactErr* status bit in the queue head is set and the *CErr* field is decremented. When the *PIDCode* indicates a SETUP, the following responses are protocol errors and result in *XactErr* bit being set to a one and the *CErr* field being decremented.

- *EPS* field indicates a high-speed device and it returns a Nak handshake to a SETUP.
- *EPS* field indicates a high-speed device and it returns a Nyet handshake to a SETUP.
- *EPS* field indicates a low- or full-speed device and the complete-split receives a Nak handshake.

Table 11-105. Summary of Transaction Errors

Event / Result	Queue Head/qTD/iTD/siTD Side-effects		USB Status Register (USBSTS)
	Cerr	Status Field	USBERRINT
CRC	-1	XactErr set to a one.	1[1]
Timeout	-1	XactErr set to a one.	1[1]
Bad PID[2]	-1	XactErr set to a one.	1[1]
Babble	N/A	Section Serial Bus Babble	1
Buffer Error	N/A	Section Data Buffer Error	

1. If occurs in a queue head, then *USBERRINT* is asserted only when *CErr* counts down from a one to a zero. In addition the queue is halted, see Halting a Queue Head.
2. The host controller received a response from the device, but it could not recognize the PID as a valid PID.

11.3.4.3.15.1.2 Serial Bus Babble

When a device transmits more data on the USB than the host controller is expecting for this transaction, it is defined to be babbling. In general, this is called a *Packet Babble*.

When a device sends more data than the *Maximum Length* number of bytes, the host controller sets the *Babble Detected* bit to a one and halts the endpoint if it is using a queue head (see Halting a Queue Head). *Maximum Length* is defined as the minimum of *Total Bytes to Transfer* and *Maximum Packet Size*. The *CErr* field is not decremented for a packet babble condition (only applies to queue heads). A babble condition also exists if IN transaction is in progress at High-speed EOF2 point. This is called a frame babble. A frame babble condition is recorded into the appropriate schedule data structure. In addition, the host controller must disable the port to which the frame babble is detected.

The *USBERRINT* bit in the USB_n_USBSTS register is set to a one and if the *USB Error Interrupt Enable* bit in the USB_n_USBINTR register is a one, then a hardware interrupt is signaled to the system at the next interrupt threshold. The host controller must never start an OUT transaction that will babble across a micro-frame EOF.

NOTE

When a host controller detects a data PID mismatch, it must either: disable the packet babble checking for the duration of the bus transaction or do packet babble checking based solely on *Maximum Packet Size*. The USB core specification defines the requirements on a data receiver when it receives a data PID mismatch (for example, expects a DATA0 and gets a DATA1 or visa-versa). In summary, it must ignore the received data and respond with an ACK handshake, in order to advance the transmitter's data sequence.

The EHCI interface allows System software to provide buffers for a Control, Bulk or Interrupt IN endpoint that are not an even multiple of the maximum packet size specified by the device. Whenever a device misses an ACK for an IN endpoint, the host and device are out of synchronization with respect to the progress of the data transfer. The host controller may have advanced the transfer to a buffer that is less than maximum packet size. The device will re-send its maximum packet size data packet, with the original data PID, in response to the next IN token. In order to properly manage the bus protocol, the host controller must disable the packet babble check when it observes the data PID mismatch.

11.3.4.3.15.1.3 Data Buffer Error

This event indicates that an overrun of incoming data or a underrun of outgoing data has occurred for this transaction.

This would generally be caused by the host controller not being able to access required data buffers in memory within necessary latency requirements. These conditions are not considered transaction errors, and do not effect the error count in the queue head. When these errors do occur, the host controller records the fact the error occurred by setting the *Data Buffer Error* bit in the queue head, iTD or siTD.

If the data buffer error occurs on a non-isochronous IN, the host controller will not issue a handshake to the endpoint. This will force the endpoint to resend the same data (and data toggle) in response to the next IN to the endpoint.

If the data buffer error occurs on an OUT, the host controller must corrupt the end of the packet so that it cannot be interpreted by the device as a good data packet. Simply truncating the packet is not considered acceptable. An acceptable implementation option is to 1's complement the CRC bytes and send them. There are other options suggested in the Transaction Translator section of the USB Specification Revision 2.0.

11.3.4.3.15.1.4 USB Interrupt (Interrupt on Completion (IOC))

Transfer Descriptors (iTDs, siTDs, and queue heads (qTDs)) contain a bit that can be set to cause an interrupt on their completion. The completion of the transfer associated with that schedule item causes the USB Interrupt (USBINT) bit in the USB_n_USBSTS register to be set to a one.

In addition, if a short packet is encountered on an IN transaction associated with a queue head, then this event also causes USBINT to be set to a one. If the USB Interrupt Enable bit in the USB_n_USBINTR register is set to a one, a hardware interrupt is signaled to the system at the next interrupt threshold. If the completion is because of errors, the *USBERRINT* bit in the USB_n_USBSTS register is also set to a one.

11.3.4.3.15.1.5 Short Packet

Reception of a data packet that is less than the endpoint's Max Packet size during Control, Bulk or Interrupt transfers signals the completion of the transfer. Whenever a short packet completion occurs during a queue head execution, the *USBINT* bit in the USB_n_USBSTS register is set to a one.

If the *USB Interrupt Enable* bit is set in the USB_n_USBINTR register, a hardware interrupt is signaled to the system at the next interrupt threshold.

11.3.4.3.15.2 Host Controller Event Interrupts

These interrupt sources are independent of the interrupt threshold (with the one exception being the Interrupt on Async Advance, see Section Interrupt on Async Advance).

11.3.4.3.15.2.1 Port Change Events

Port registers contain status and status change bits. When the status change bits are set to a one, the host controller sets the *Port Change Detect* bit in the USBSTS register to a one.

If the *Port Change Interrupt Enable* bit in the USB_n_USBINTR register is a one, then the host controller will issue a hardware interrupt. The port status change bits include:

- Connect Status Change
- Port Enable/Disable Change
- Over-current Change
- Force Port Resume

11.3.4.3.15.2.2 Frame List Rollover

This event indicates that the host controller has wrapped the frame list. The current programmed size of the frame list effects how often this interrupt occurs.

If the frame list size is 1024, then the interrupt will occur every 1024 milliseconds, if it is 512, then it will occur every 512 milliseconds, etc. When a frame list rollover is detected, the host controller sets the *Frame List Rollover* bit in the USB.USBSTS register to a one. If the *Frame List Rollover Enable* bit in the USB.USBINTR register is set to a one, the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

11.3.4.3.15.2.3 Interrupt on Async Advance

This event is used for deterministic removal of queue heads from the asynchronous schedule. Whenever the host controller advances the on-chip context of the asynchronous schedule, it evaluates the value of the *Interrupt on Async Advance Doorbell* bit in the USB.USBCMD register.

If it is a one, it sets the *Interrupt on Async Advance* bit in the USB.USBSTS register to a one. If the *Interrupton Async Advance Enable* bit in the USB.USBINTR register is a one, the host controller issues a hardware interrupt at the next interrupt threshold. A detailed explanation of this feature is described in Section Removing Queue Heads from Asynchronous Schedule .

11.3.4.3.15.2.4 Host System Error

The host controller is a bus master and any interaction between the host controller and the system may experience errors.

The type of host error may be catastrophic to the host controller (such as a Master Abort) making it impossible for the host controller to continue in a coherent fashion.In the presence of non-catastrophic host errors, such as parity errors, the host controller could potentially continue operation. The recommended behavior for these types of errors is to escalate it to a catastrophic error and halt the host controller. Host-based error must result in the following actions:

- The *Run/Stop* bit in the USB.USBCMD register is set to a zero.
- The following bits in the USB.USBSTS register are set:
  - *Host System Error* bit is to a one.
  - *HCHalted* bit is set to a one.
- If the *Host System Error Enable* bit in the USB.USBINTR register is a one, then the host controller will issue a hardware interrupt. This interrupt is not delayed to the next interrupt threshold. The following table summarizes the required actions taken on the various host errors.

Table 11-106. Summary Behavior of EHCI Host Controller on Host System Errors

Cycle Type	Master Abort	Target Abort	Data Phase Parity
Frame list pointer fetch (read)	Fatal	Fatal	Fatal [o]
siTD fetch (read)	Fatal	Fatal	Fatal [o]
siTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
iTD fetch (read)	Fatal	Fatal	Fatal [o]
iTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
qTD fetch (read)	Fatal	Fatal	Fatal [o]
qHD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
Data write	Fatal [o]	Fatal [o]	Fatal [o]
Data read	Fatal	Fatal	Fatal [o]

Potentially, a host controller implementation could continue operation without a halt. However, the recommended behavior is to halt the host controller.

NOTE

After a *Host System Error*, Software must reset the host controller through *HCReset* in the USB.USBCMD register before re-initializing and restarting the host controller.

11.3.4.4 EHCI Deviation

For the purposes a dual-role Host/Device controller with support for On-The-Go applications, it is necessary to deviate from the EHCI specification. Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com. Device operation & On-The-Go operation is not specified in the EHCI and thus the implementation supported in this core is proprietary.

The host mode operation of the core is near EHCI compatible with few minor differences documented in this section.

The particulars of the deviations occur in the areas summarized here:

- Embedded Transaction Translator - Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.
- Device operation - In host mode the device operational registers are generally disabled and thus device mode is mostly transparent when in host mode. However, there are a couple exceptions documented in the following sections.
- Embedded design interface - This core does not have a PCI Interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.
- On-The-Go Operation - This design includes an On-The-Go controller for Port #1.

11.3.4.4.1 Embedded Transaction Translator Function

The OTG controller supports directly connected full and low speed devices without requiring a companion controller by including the capabilities of a USB 2.0 high speed hub transaction translator.

Although there is no separate Transaction Translator block in the system, the transaction translator function normally associated with a high speed hub has been implemented within the DMA and Protocol engine blocks. The embedded transaction translator function is an extension to EHCI interface, but makes use of the standard data structures and operational models that exist in the EHCI specification to support full and low speed devices.

11.3.4.4.1.1 Capability Registers

The following additions have been added to the capability registers to support the embedded Transaction Translator Function:

- N_TT added to USB.HCSPARAMS - Host Control Structural Parameters
- N_PTT added to USB.HCSPARAMS - Host Control Structural Parameters

11.3.4.4.1.2 Operational Registers

The following additions have been added to the operational registers to support the embedded TT:

- Addition of two-bit Port Speed (PSPD) to the register.

11.3.4.4.1.3 Discovery-EHCI Deviation

In a standard EHCI controller design, the EHCI host controller driver detects a Full speed (FS) or Low speed (LS) device by noting if the port enable bit is set after the port reset operation.

The port enable will only be set in a standard EHCI controller implementation after the port reset operation and when the host and device negotiate a High-Speed connection (that is, Chirp completes successfully).

Because this controller has an embedded Transaction Translator, the port enable will always be set after the port reset operation regardless of the result of the host device chirp result and the resulting port speed will be indicated by the PSPD field in USB.PORTSCx.

Therefore, the standard EHCI host controller driver requires an alteration to handle directly connected Full and Low speed devices or hubs.

The change is a fundamental one in that is summarized in the following table.

Table 11-107. Summary of EHCI

Standard EHCI	EHCI with embedded Transaction Translator
After port enable bit is set following a connection and reset sequence, the device/hub is assumed to be HS.	After port enable bit is set following a connection and reset sequence, the device/hub speed is noted from USB.PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub thus, all port-level control is performed through the Hub Class to the nearest Hub.	FS and LS device can be either downstream from a HS hub or directly attached. When the FS/LS device is downstream from a HS hub, then port-level control is done using the Hub Class through the nearest Hub. When a FS/LS device is directly attached, then port-level control is accomplished using USB.PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub with HubAddr=X. [where HubAddr > 0 and HubAddr is the address of the Hub where the bus transitions from HS to FS/LS (ie. Split target hub)]	FS and LS device can be either downstream from a HS hub with HubAddr = X [HubAddr > 0] or directly attached [where HubAddr = 0 and HubAddr is the address of the Root Hub where the bus transitions from HS to FS/LS (ie. Split target hub is the root hub)]

11.3.4.4.1.4 Data Structures

The same data structures used for FS/LS transactions though a HS hub are also used for transactions through the Root Hub with sm embedded Transaction Translator.

Here it is demonstrated how the Hub Address and Endpoint Speed fields should be set for directly attached FS/LS devices and hubs:

1. QH (for direct attach FS/LS) - Async. (Bulk/Control Endpoints) Periodic (Interrupt)
   - Hub Address = 0
   - Transactions to direct attached device/hub.
     - QH.EPS = Port Speed
   - Transactions to a device downstream from direct attached FS hub.
     - QH.EPS = Downstream Device Speed

NOTE

When QH.EPS = 01 (LS) and PORTSCx.PSPD = 00 (FS), a LS-pre-pid will be sent before the transmitting LS traffic.

Maximum Packet Size must be less than or equal 64 or undefined behaviour may result.

2. siTD (for direct attach FS) - Periodic (ISO Endpoint)
   - All FS ISO transactions:
     - Hub Address = 0
     - siTD.EPS = 00 (full speed)
       - Maximum Packet Size must less than or equal to 1023 or undefined behaviour may result.

11.3.4.4.1.5 Operational Model

The operational models are well defined for the behavior of the Transaction Translator (see USB 2.0 specification. Universal Serial Bus Specification, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org) and for the EHCI controller moving packets between system memory and a USB-HS hub. Because the embedded Transaction Translator exists within the host controller there is no physical bus between EHCI host controller driver and the USB FS/LS bus. These sections will briefly discuss the operational model for how the EHCI and Transaction Translator operational models are combined without the physical bus between. The following sections assume the reader is familiar with both the EHCI and USB 2.0 Transaction Translator operational models.

11.3.4.4.1.5.1 Micro- frame Pipeline

The EHCI operational model uses the concept of H-frames and B-frames to describe the pipeline between the Host (H) and the Bus (B). The embedded Transaction Translator shall use the same pipeline algorithms specified in the USB 2.0 specification for a Hub-based Transaction Translator.

All periodic transfers always begin at B-frame 0 (after SOF) and continue until the stored periodic transfers are complete. As an example of the micro-frame pipeline implemented in the embedded Transaction Translator, all periodic transfers that are tagged in EHCI to execute in H-frame 0 will be ready to execute on the bus in B-frame 0.

It is important to note that when programming the S-mask and C-masks in the EHCI data structures to schedule periodic transfers for the embedded Transaction Translator, the EHCI host controller driver must follow the same rules specified in EHCI for programming the S-mask and C-mask for downstream Hub-based Transaction Translators.

Once periodic transfers are exhausted, any stored asynchronous transfer will be moved. Asynchronous transfers are opportunistic in that they shall execute whenever possible and their operation is not tied to H-frame and B-frame boundaries with the exception that an asynchronous transfer can not babble through the SOF (start of B-frame 0.)

11.3.4.4.1.5.2 Split State Machines

The start and complete split operational model differs from EHCI slightly because there is no bus medium between the EHCI controller and the embedded Transaction Translator.

Where a start or complete-split operation would occur by requesting the split to the HS hub, the start/complete split operation is simple an internal operation to the embedded Transaction Translator. The following table summarizes the conditions where handshakes are emulated from internal state instead of actual handshakes to HS split bus traffic.

Table 11-108. Summary of the Conditons of Handshakes[1]

Condition	Emulate TT Response
Start-Split: All asynchronous buffers full.	NAK
Start-Split: All periodic buffers full.	ERR
Start-Split: Success for start of Async. Transaction.	ACK
Start-Split: Start Periodic Transaction.	No Handshake (Ok)
Complete-Split: Failed to find transaction in queue.	Bus Time Out
Complete-Split: Transaction in Queue is Busy.	NYET
Complete-Split: Transaction in Queue is Complete.	[Actual Handshake from LS/FS device]

1. The un-shaded cells represent Start-Splits and the shaded cells represent Complete-Splits

11.3.4.4.1.5.3 Asynchronous Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

11.3.4.4.1.5.3.1 USB 2.0 - 11.17.3

- Sequencing is provided & a packet length estimator ensures no full-speed/low-speed packet babbles into SOF time.

11.3.4.4.1.5.3.2 USB 2.0 - 11.17.4

- Transaction tracking for 2 data pipes.

11.3.4.4.1.5.4 Periodic Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

11.3.4.4.1.5.4.1 USB 2.0 - 11.18.6.[1-2]

- Abort of pending start-splits
  - EOF (and not started in micro-frames 6)
  - Idle for more than 4 micro-frames
- Abort of pending complete-splits
  - EOF
  - Idle for more than 4 micro-frames

11.3.4.4.1.5.4.2 USB 2.0 - 11.18.[7-8]

- Transaction tracking for up to 16 data pipes.
- Complete-split transaction searching.

NOTE

There is no data schedule mechanism for these transactions other than the micro-frame pipeline. The embedded TT assumes the number of packets scheduled in a frame does not exceed the frame duration (1 ms) or else undefined behavior may result.

11.3.4.4.1.5.5 Multiple Transaction Translators

The maximum number of embedded Transaction Translators that is currently supported is one as indicated by the N_TT field in the register.

11.3.4.4.2 Device Operation

The co-existence of a device operational controller within the host controller has little effect on EHCI compatibility for host operation except as noted in this section.

11.3.4.4.2.1 USB_USBMODE Register

Given that the dual-role controller is initialized in neither host nor device mode, the USB Device Mode (USB_USBMODE) register must be programmed for host operation before the EHCI host controller driver can begin EHCI host operations.

11.3.4.4.2.2 Non-Zero Fields the Register File

Some of the reserved fields and reserved addresses in the capability registers and operational register have use in device mode, the following must be adhered to:

- Write operations to all EHCI reserved fields (some of which are device fields) with the operation registers should always be written to zero. This is an EHCI requirement of the device controller driver that must be adhered to.
- Read operations by the host controller must properly mask EHCI reserved fields (some of which are device fields) because fields that are used exclusive for device are undefined in host mode .

11.3.4.4.2.3 SOF Interrupt

This SOF Interrupt used for device mode is shared as a free running 125us interrupt for host mode.

EHCI does not specify this interrupt but it has been added for convenience and as a potential software time base. See and registers.

11.3.4.4.3 Embedded Design Interface

This is an Embedded USB Host Controller as defined by the EHCI specification and thus does not implement the PCI configuration registers.

11.3.4.4.3.1 Frame Adjust Register

Given that the optional PCI configuration registers are not included in this implementation, there is no corresponding bit level timing adjustments like is provided by the Frame Adjust register in the PCI configuration registers. Starts of micro-frames are timed precisely to 125 us using the transceiver clock as a reference clock. That is, a 60 Mhz transceiver clock for 8-bit physical interfaces & full-speed serial interfaces or 30 Mhz transceiver clock for 16-bit physical interfaces.

11.3.4.4.4 Miscellaneous variations from EHCI

11.3.4.4.4.1 Programmable Physical Interface Behaviour

This design supports multiple Physical interfaces which can operate in differing modes when the core is configured with software programmable Physical Interface Modes.

Software programmability allows the selection of the Physical interface part during the board design phase instead of during the chip design phase.

The control bits for selecting the Physical Interface operating mode have been added to the register providing a capability that is not defined by EHCI.

11.3.4.4.4.2 Discovery

11.3.4.4.4.2.1 Port Reset

The port connect methods specified by EHCI require setting the port reset bit in the register for a duration of 10ms. Due to the complexity required to support the attachment of devices that are not high speed there are counter already present in the design that can count the 10ms reset pulse to alleviate the requirement of the software to measure this duration. Therefore, the basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a '1' to reset the device.
- Software shall write a '0' to reset the device after 10 ms.
  - This step, which is necessary in a standard EHCI design, may be omitted with this implementation. Should the EHCI host controller driver attempt to write a '0' to the reset bit while a reset is in progress, the write will simple be ignored and the reset will continue until completion.
- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device in now operational and at this point the port speed has been determined.

11.3.4.4.4.2.2 Port Speed Detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation, which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non High-Speed devices.

Therefore, the following differences are important regarding port speed detection:

- Port Owner is read-only and always reads 0.
- A 2-bit Port Speed indicator has been added to PORTSC to provide the current operating speed of the port to the host controller driver.
- A 1-bit High Speed indicator has been added to PORTSC to signify that the port is in High-Speed vs. Full/Low Speed - *This information is redundant with the 2-bit Port Speed indicator above.*

11.3.4.4.4.3 Port Test Mode

Port Test Control mode behaves fully as described in EHCI. An alternate host controller driver procedure is not necessary or supported.

11.3.4.5 Device Data Structures

This section defines the interface data structures used to communicate control, status, and data between Device Controller Driver (DCD) Software and the Device Controller.

The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of device Queue Heads and Transfer Descriptors.

NOTE

Software must ensure that no interface data structure reachable by the Device Controller spans a 4K-page boundary.

The data structures defined in the chapter are (from the device controller's perspective) a mix of read-only and read/ writable fields. The device controller must preserve the read-only fields on all data structure writes.

The figure below shows the organization of the EndPoint Queue Head.

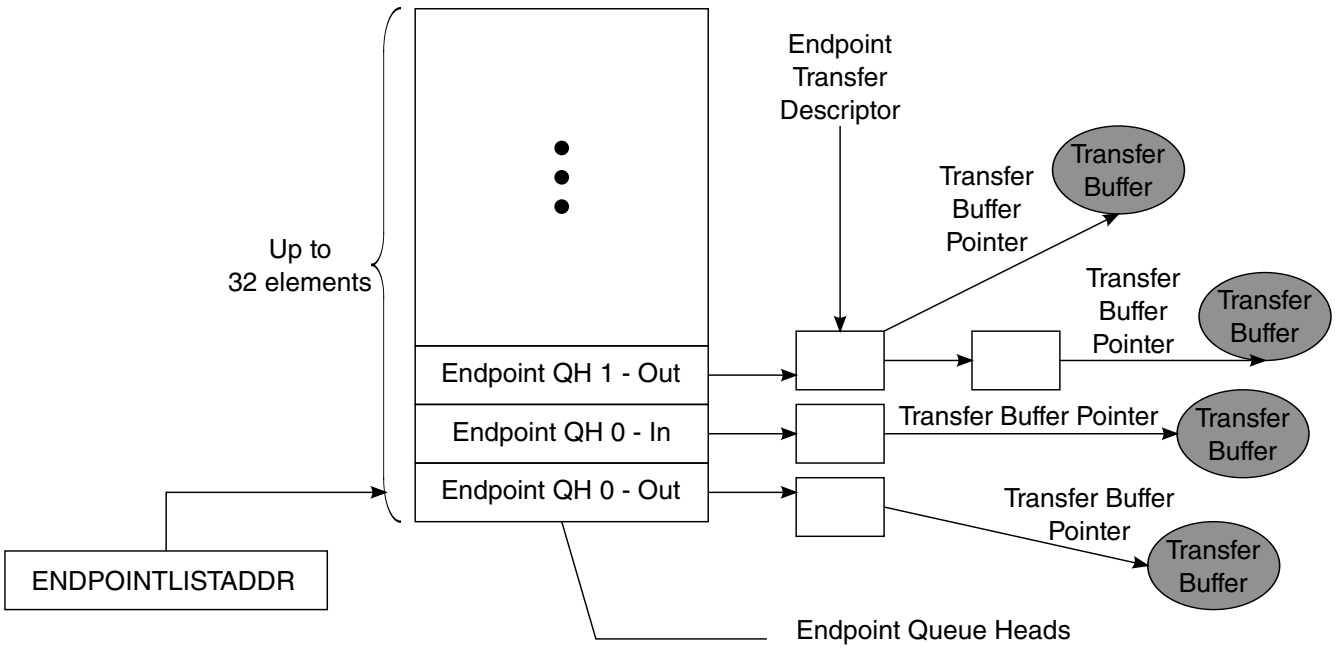


Figure 11-80. EndPoint Queue Head Organization

Endpoint queue heads are arranged in an array in a continuous area of memory pointed to by the USB.ENDPOINTLISTADDR pointer. The even -numbered device queue heads in the list support receive endpoints (OUT/SETUP) and the odd-numbered queue heads in the list are used for transmit endpoints (IN/INTERRUPT). The device controller will index into this array based upon the endpoint number received from the USB bus. All information necessary to respond to transactions for all primed transfers is contained in this list so the Device Controller can readily respond to incoming requests without having to traverse a linked list.

NOTE

The Endpoint Queue Head List must be aligned to a 2k boundary.

11.3.4.5.1 Endpoint Queue Head (dQH)

The device Endpoint Queue Head (dQH) is where all transfers for a given endpoint are managed. The dQH is a 48-byte data structure, but must be aligned on 64-byte boundaries.

During priming of an endpoint, the dTD (device transfer descriptor) is copied into the overlay area of the dQH, which starts at the nextTD pointer DWord and continues through the end of the buffer pointers DWords. After a transfer is complete, the dTD

status DWord is updated in the dTD pointed to by the currentTD pointer. While a packet is in progress, the overlay area of the dQH is used as a staging area for the dTD so that the Device Controller can access needed information with little minimal latency.

Table 11-109. Endpoint Queue Head (dQH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult		zlt	0		Maximum Packet Length											ios	0														
Current dTD Pointer																											0				
Next dTD Pointer																											0				T[1]
0	Total Bytes															ioc	0			MultO		0		Status							
Buffer Pointer (Page 0)																				Current Offset											
Buffer Pointer (Page 1)																				Reserved											
Buffer Pointer (Page 2)																				Reserved											
Buffer Pointer (Page 3)																				Reserved											
Buffer Pointer (Page 4)[1]																				Reserved											
Reserved																															
Set-up Buffer Bytes 3…0																															
Set-up Buffer Bytes 7…4																															

1. Transfer overlay starts at T and continues through Buffer Pointer (Page 4).

Host Controller Read/Write

Host Controller Read Only

11.3.4.5.1.1 Endpoint Capabilities/Characteristics

This DWord specifies static information about the endpoint, in other words, this information does not change over the lifetime of the endpoint. Device Controller software should not attempt to modify this information while the corresponding endpoint is enabled.

Table 11-110 describes the endpoint capabilities.

Table 11-110. Endpoint Capabilities/Characteristics

Bit	Description

Table continues on the next page...

Table 11-110. Endpoint Capabilities/Characteristics (continued)

31-30	Mult. This field is used to indicate the number of packets executed per transaction description as given by the following: 00 - Execute N Transactions as demonstrated by the USB variable length packet protocol where N is computed using the Maximum Packet Length (dQH) and the Total Bytes field (dTD) 01 Execute 1 Transaction. 10 Execute 2 Transactions. 11 Execute 3 Transactions. **NOTE:** Non-ISO endpoints must set Mult="00". ISO endpoints must set Mult="01", "10", or "11" as needed.
29	Zero Length Termination Select. This bit is used to indicate when a zero length packet is used to terminate transfers where to total transfer length is a multiple . This bit is not relevant for Isochronous 0 - Enable zero length packet to terminate transfers equal to a multiple of the Maximum Packet Length. (default). 1 - Disable the zero length packet on transfers that are equal in length to a multiple Maximum Packet Length.
28-27	Reserved. These bit reserved for future use and should be set to zero.
26-16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).
15	Interrupt On Setup (IOS). This bit is used on control type endpoints to indicate if USBINT is set in response to a setup being received.
14-0	Reserved. Bits reserved for future use and should be set to zero.

11.3.4.5.1.2 Transfer Overlay-Endpoint Queue Head

The seven DWords in the overlay area represent a transaction working space for the device controller.

The general operational model is that the device controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it will not read the associated endpoint.

After an endpoint is readied, the dTD will be copied into this queue head overlay area by the device controller. Until a transfer is expired, software must not write the queue head overlay area or the associated transfer descriptor. When the transfer is complete, the device controller will write the results back to the original transfer descriptor and advance the queue.

See dTD for a description of the overlay fields.

11.3.4.5.1.3 Current dTD Pointer

The current dTD pointer is used by the device controller to locate the transfer in progress. This word is for Device Controller (hardware) use only and should not be modified by DCD software.

The following table describes the dTD Pointer.

Table 11-111. Next dTD Pointer

Bit	Description
31-5	Current dTD. This field is a pointer to the dTD that is represented in the transfer overlay area. This field will be modified by the Device Controller to next dTD pointer during endpoint priming or queue advance.
4-0	Reserved. Bit reserved for future use and should be set to zero.

11.3.4.5.1.4 Set-up Buffer

The set-up buffer is dedicated storage for the 8-byte data that follows a set-up PID.

NOTE

Each endpoint has a TX and an RX dQH associated with it, and only the RX queue head is used for receiving setup data packets.

The following table describes the Multiple Mode Control.

Table 11-112. Multiple Mode Control (HCCPARAMS)

DWord	Bits	Description
1	31-0	Setup Buffer 0. This buffer contains bytes 3 to 0 of an incoming setup buffer packet and is written by the device controller to be read by software.
2	31-0	Setup Buffer 1. This buffer contains bytes 7 to 4 of an incoming setup buffer packet and is written by the device controller to be read by software.

11.3.4.5.2 Endpoint Transfer Descriptor (dTD)

The dTD describes to the device controller the location and quantity of data to be sent/ received for a given transfer.

The DCD should not attempt to modify any field in an active dTD except the Next Like Pointer, which should only be modified as described in section Managing Transfers with Transfer Descriptors.

Table below shows the Endpoint Transfer Descriptor (dTD).

Table 11-113. Endpoint Transfer Descriptor (dTD)

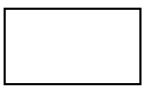
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Next Link Pointer																											0				T
0	Total Bytes															ioc	0			MultO		0		Status							
Buffer Pointer (Page 0)																				Current Offset											

Table continues on the next page...

Table 11-113. Endpoint Transfer Descriptor (dTD) (continued)

Buffer Pointer (Page 1)	0	Frame Number
Buffer Pointer (Page 2)	Reserved	
Buffer Pointer (Page 3)	Reserved	
Buffer Pointer (Page 4)	Reserved	

Host Controller Read/Write



Host Controller Read Only

The following table describes the dTD Pointer.

Table 11-114. Next dTD Pointer

Bit	Description
31-5	Next Transfer Element Pointer. This field contains the physical memory address of the next dTD to be processed. The field corresponds to memory address signals [31:5], respectively.
4-1	Reserved. Bits reserved for future use and should be set to zero.
0	Terminate (T). 1=pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Device Controller that there are no more valid entries in the queue.

The following table describes the dTD Token.

Table 11-115. dTD Token

Bit	Description
31	Reserved. Bit reserved for future use and should be set to zero.
30-16	Total Bytes. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction and only on the successful completion of the transaction. The maximum value software may store in the field is 5*4K (5000H). This is the maximum number of bytes 5 page pointers can access. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16K (4000H). If the value of the field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the device controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for IN transfers that Total Bytes To Transfer be an even multiple of *Maximum Packet Length*. If software builds such a transfer descriptor for an IN transfer, the last transaction will always be less that *Maximum Packet Length*.
15	Interrupt On Complete (IOC). This bit is used to indicate if USBINT is to be set in response to device controller being finished with this dTD.
14-12	Reserved. Bits reserved for future use and should be set to zero.
11-10	Multiplier Override (MultO). This field can be used for transmit ISO's (ie. ISO-IN) to override the multiplier in the QH. This field must be zero for all packet types that are not transmit-ISO. Example:

Table continues on the next page...

Table 11-115. dTD Token (continued)

	if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 0 [default] Three packets are sent: {Data2(8); Data1(7); Data0(0)} if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 2 Two packets are sent: {Data1(8); Data0(7)} For maximal efficiency, software should compute MultO = greatest integer of (Total Bytes / Max. Packet Size) except for the case when Total Bytes = 0; then MultO should be 1. Note: Non-ISO and Non-TX endpoints must set MultO = "00".
9-8	Reserved. Bits reserved for future use and should be set to zero.
7-0	Status. This field is used by the Device Controller to communicate individual command execution states back to the Device Controller software. This field contains the status of the last transaction performed on this qTD. The bit encodings are: Bit Status Field Description 7 Active. 6 Halted. 5 Data Buffer Error. 3 Transaction Error. 4, 2, 0 Reserved.

The table below describes the dTD Buffer Page Pointer List.

Table 11-116. dTD Buffer Page Pointer List

Bit	Description
31-12	Buffer Pointer. Selects the page offset in memory for the packet buffer. Non virtual memory systems will typically set the buffer pointers to a series of incrementing integers.
0,11-0	Current Offset. Offset into the 4kb buffer where the packet is to begin.
1,10-0	Frame Number. Written by the device controller to indicate the frame number in which a packet finishes. This is typically be used to correlate relative completion times of packets on an ISO endpoint.

11.3.4.6 Device Operational Model

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus.

Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers. The following sections explain the use of the device controller from the device controller driver (DCD) point-of-view and further describe how specific USB bus events relate to status changes in the device controller programmer's interface.

11.3.4.6.1 Device Controller Initialization

After hardware reset, the device is disabled until the Run/Stop bit is set to a '1'. In the disabled state, the pull-up on the USB D+ is not active which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs.

Shortly after the device is enabled, a USB reset will occur followed by setup packet arriving at endpoint 0. A Queue head must be prepared so that the device controller can store the incoming setup packet.

In order to initialize a device, the software should perform the following steps:

- Set Controller Mode in the USB.USBMODE register to device mode.

NOTE

Transitioning from host mode to device mode requires a device controller reset before modifying USB.USBMODE.

- Allocate and Initialize device queue heads in system memory.
  - Minimum: Initialize device queue heads 0 Tx & 0 Rx.

NOTE

All device queue heads for control endpoints must be initialized before the endpoint is enabled. Non-Control device queue heads before the endpoint can be used.

  - For information on device queue heads, refer to section Device Data Structures.
- Configure USB.ENDPOINTLISTADDR Pointer.
  - For additional information on USB.ENDPOINTLISTADDR, refer to the register table.
- Enable the microprocessor interrupt associated with the USB core.
  - Recommended: enable all device interrupts including: USBINT, USBERRINT, Port Change Detect, USB Reset Received, DCSuspend.
  - For a list of available interrupts refer to the and the register tables.
- Set Run/Stop bit to Run Mode.
  - After the Run bit is set and the device is connected to a host, a Bus Reset will be issued by host downstream port. The DCD must monitor the reset event and adjust the software state as described in the Bus Reset section of the Port State and Control section below.

NOTE

Endpoint 0 is designed as a control endpoint only and does not need to be configured using ENDPTCTRL0 register.

It is also not necessary to prime Endpoint 0 initially because the first packet received will always be a setup packet. The contents of the first setup packet will require a response in accordance with USB device framework (Chapter 9) command set.

11.3.4.6.2 Port State and Control

From a chip or system reset, the device controller enters the *powered* state. A transition from the *powered* state to the *attach* state occurs when the Run/Stop bit is set to a '1'.

After receiving a reset on the bus, the port will enter the *defaultFS or defaultHS* state in accordance with the reset protocol described in Appendix C.2 of the USB Specification Rev. 2.0.

The following state diagram depicts the state of a USB 2.0 device.

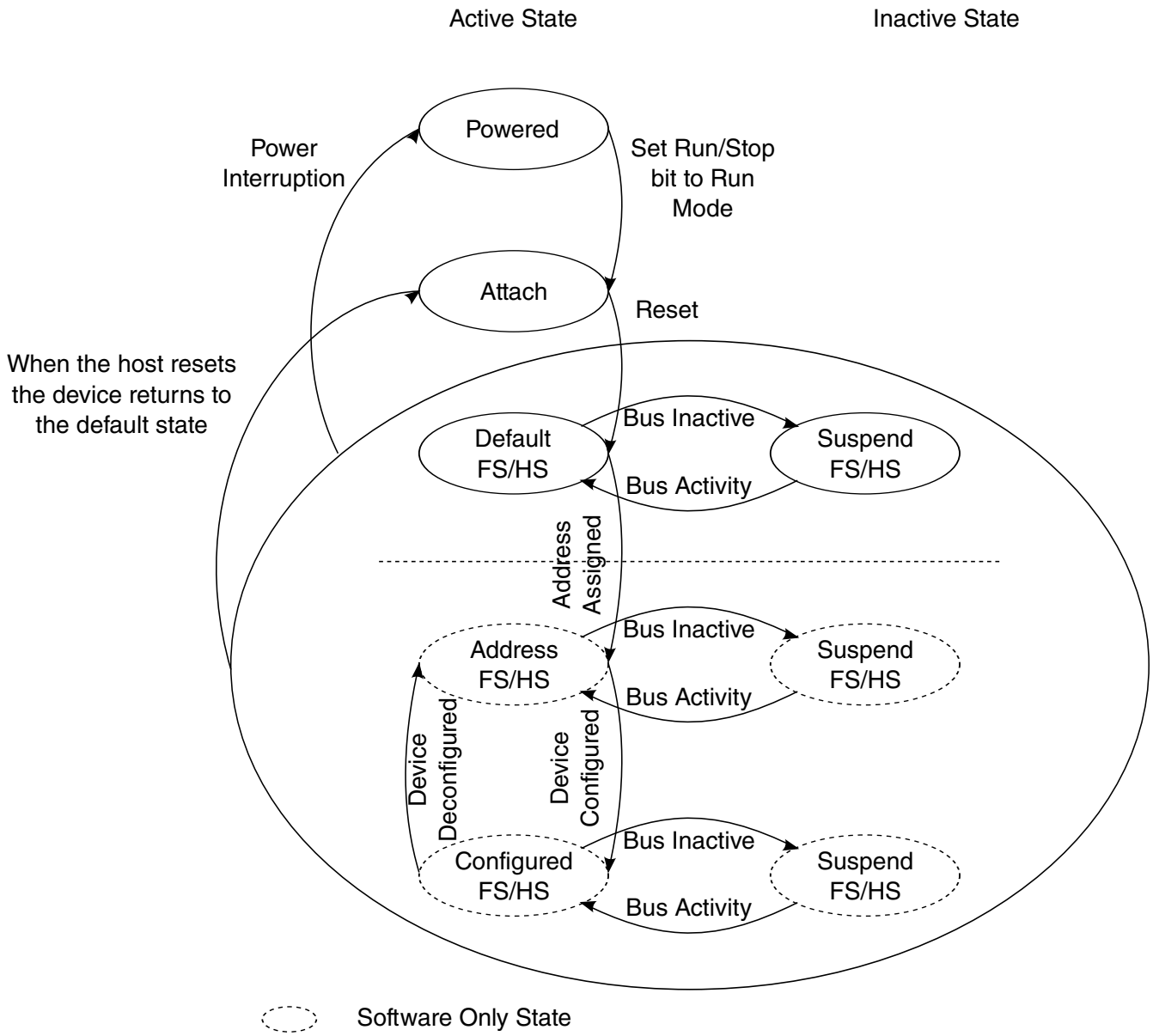


Figure 11-81. Device State Diagram

States *powered, attach, defaultFS/HS, suspendFS/HS* are implemented in the device controller and are communicated to the DCD using the following status bits:

The following table describes the Device Controller State Information Bits.

Table 11-117. Device Controller State Information Bits

Bit	Register
DCSuspend	
USB Reset Received	
Port Change Detect	
High-Speed Port	

It is the responsibility of the DCD to maintain a state variable to differentiate between the *DefaultFS/HS* state and the *Address/Configured* states. Change of state from *Default* to *Address* and the *Configured* states is part of the enumeration process described in the device framework section of the USB 2.0 Specification.

As a result of entering the *Address* state, the device address register (DEVICEADDR) must be programmed by the DCD.

Entry into the *Configured* indicates that all endpoints to be used in the operation of the device have been properly initialized by programming the USB_UOG_ENDPTCTRLx registers and initializing the associated queue heads.

11.3.4.6.2.1 Bus Reset

A bus reset is used by the host to initialize downstream devices.

When a bus reset is detected, the device controller will renegotiate its attachment speed, reset the device address to 0, and notify the DCD by interrupt (assuming the USB Reset Interrupt Enable is set). After a reset is received, all endpoints (except endpoint 0) are disabled and any primed transactions will be cancelled by the device controller. The concept of priming will be clarified below, but the DCD must perform the following tasks when a reset is received:

Clear all setup token semaphores by reading the register and writing the same value back to the register.

Clear all the endpoint complete status bits by reading the register and writing the same value back to the register.

Cancel all primed status by waiting until all bits in the are 0 and then writing 0xFFFFFFFF to Endpoint Flush (USB_ENDPTFLUSH).

Read the reset bit in the register and make sure that it is still active. A USB reset will occur for a minimum of 3 ms and the DCD must reach this point in the reset cleanup before end of the reset occurs, otherwise a hardware reset of the device controller is recommended (rare.)

- A hardware reset can be performed by writing a one to the device controller reset bit in the USBCMD reset. Note: a hardware reset will cause the device to detach from the bus by clearing the Run/Stop bit. Thus, the DCD must completely re-initialize the device controller after a hardware reset.

Free all allocated dTDs because they will no longer be executed by the device controller. If this is the first time the DCD is processing a USB reset event, then it is likely that no dTDs have been allocated.

At this time, the DCD may release control back to the OS because no further changes to the device controller are permitted until a Port Change Detect is indicated.

After a Port Change Detect, the device has reached the default state and the DCD can read the to determine if the device is operating in FS or HS mode. At this time, the device controller has reached normal operating mode and DCD can begin enumeration according to the USB Chapter 9 - Device Framework.

NOTE

The device DCD may use the FS/HS mode information to determine the bandwidth mode of the device

In some applications, it may not be possible to enable one or more pipes while in FS mode. *Beyond the data rate issue, there is no difference in DCD operation between FS and HS modes.*

11.3.4.6.2.2 Suspend/Resume

11.3.4.6.2.2.1 Suspend

Suspend Description

In order to conserve power, USB devices automatically enter the suspended state when the device has observed no bus traffic for a specified period. When suspended, the USB device maintains any internal status, including its address and configuration. Attached devices must be prepared to suspend at any time they are powered, regardless of if they have been assigned a non-default address, are configured, or neither. Bus activity may cease due to the host entering a suspend mode of its own. In addition, a USB device shall also enter the suspended state when the hub port it is attached to is disabled.

A USB device exits suspend mode when there is bus activity. A USB device may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wakeup. The ability of a device to signal remote wakeup is optional. If the USB device is capable of remote wakeup signaling, the device must support the ability of the host to enable and disable this capability. When the device is reset, remote wakeup signaling must be disabled.

Suspend Operational Model

The device controller moves into the suspend state when suspend signaling is detected or activity is missing on the upstream port for more than a specific period. After the device controller enters the suspend state, the DCD is notified by an interrupt (assuming *DC Suspend Interrupt* is enabled). When the *DCSuspend* bit in the is set to a '1', the device controller is suspended.

DCD response when the device controller is suspended is application specific and may involve switching to low power operation.

Information on the bus power limits in suspend state can be found in USB 2.0 specification.

NOTE

Review system level clocking issues defined in section (Ref: Signals-Clocking) for the clocking requirements of a suspended device controller.

11.3.4.6.2.2.2 Resume

If the device controller is suspended, its operation is resumed when any non-idle signaling is received on its upstream facing port. In addition, the device can signal the system to resume operation by forcing resume signaling to the upstream port.

Resume signaling is sent upstream by writing a '1' to the Resume bit in the in the Port Status & Control (USB_PORTSC1) while the device is in suspend state. Sending resume signal to an upstream port should cause the host to issue resume signaling and bring the suspended bus segment (one more devices) back to the active condition.

NOTE

Before resume signaling can be used, the host must enable it by using the Set Feature command defined in device framework (chapter 9) of the USB 2.0 Specification.

11.3.4.6.3 Managing Endpoints

The USB 2.0 specification defines an endpoint, also called a device endpoint or an address endpoint as a uniquely addressable portion of a USB device that can source or sink data in a communications channel between the host and the device.

The endpoint address is specified by the combination of the endpoint number and the endpoint direction.

The channel between the host and an endpoint at a specific device represents a data pipe. Endpoint 0 for a device is always a *control* type data channel used for device discovery and enumeration. Other types of endpoints support by USB include *bulk, interrupt*, and

isochronous. Each endpoint type has specific behavior related to packet response and error handling. More detail on endpoint operation can be found in the USB 2.0 specification.

The USB OTG device controller hardware supports up to 8 endpoint numbers.

Each endpoint direction is essentially independent and can be configured with differing behavior in each direction. For example, the DCD can configure endpoint 1-IN to be a bulk endpoint and endpoint 1-OUT to be an isochronous endpoint. This helps to conserve the total number of endpoints required for device operation. The only exception is that control endpoints must use both directions on a single endpoint number to function as a control endpoint. Endpoint 0 is, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a *queue head* allocated in memory. To support the 8 endpoint numbers, 16 *queue heads* are required. The operation of an endpoint and use of *queue heads* are described later in this document.

11.3.4.6.3.1 Endpoint Initialization

After hardware reset, all endpoints except endpoint zero are uninitialized and disabled. The DCD must configure and enable each endpoint by writing to configuration bit in the USB_UOG_ENDPTCTRLx register.

Each 32-bit USB_UOG_ENDPTCTRLx is split into an upper and lower half. The lower half of USB_UOG_ENDPTCTRLx is used to configure the receive or OUT endpoint and the upper half is likewise used to configure the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the USB_UOG_ENDPTCTRLx register otherwise the behavior is undefined. The following table shows how to construct a configuration word for endpoint initialization. The following table shows the fields and values for the Device Controller Endpoint initialization.

Table 11-118. Device Controller Endpoint Initialization

Field	Value
Data Toggle Reset	1
Data Toggle Inhibit	0
Endpoint Type	00 Control 01 Isochronous 10 Bulk 11 Interrupt
Endpoint Stall	0

11.3.4.6.3.2 Stalling

There are two occasions where the device controller may need to return to the host a STALL.

The first occasion is the functional stall, which is a condition set by the DCD as described in the USB 2.0 device framework. A functional stall is only used on non-control endpoints and can be enabled in the device controller by setting the endpoint stall bit in the USB_UOG_ENDPTCTRLx register associated with the given endpoint and the given direction. In a functional stall condition, the device controller will continue to return STALL responses to all transactions occurring on the respective endpoint and direction until the endpoint stall bit is cleared by the DCD.

A protocol stall, unlike a function stall, is used on control endpoints is automatically cleared by the device controller at the start of a new control transaction (setup phase). When enabling a protocol stall, the DCD should enable the stall bits (both directions) as a pair. A single write to the USB_UOG_ENDPTCTRLx register can ensure that both stall bits are set at the same instant.

NOTE

Any write to the USB_UOG_ENDPTCTRLx register during operational mode must preserve the endpoint type field (that is, perform a read-modify-write).

The following table shows the response matrix for the Device Controller Stall.

Table 11-119. Device Controller Stall Response Matrix

USB Packet	Endpoint Stall Bit.	Effect on STALL bit.	USB Response
SETUP packet received by a non-control endpoint.	N/A	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	'1'	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	'0'	None.	ACK/ NAK/ NYET
SETUP packet received by a control endpoint.	N/A	Cleared	ACK
IN/OUT/PING packet received by a control endpoint	'1'	None	STALL
IN/OUT/PING packet received by a control endpoint.	'0'	None.	ACK/ NAK/ NYET

11.3.4.6.3.3 Data Toggle

Data toggle is a mechanism to maintain data coherency between host and device for any given data pipe.

For more information on data toggle, refer to the USB 2.0 specification.

11.3.4.6.3.3.1 Data Toggle Reset

The DCD may reset the data toggle state bit and cause the data toggle sequence to reset in the device controller by writing a '1' to the data toggle reset bit in the USB_UOG_ENDPTCTRLx register.

This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

11.3.4.6.3.3.2 Data Toggle Inhibit

NOTE

This feature is for test purposes only and should never be used during normal device controller operation.

Setting the *data toggle Inhibit bit* active ('1') causes the device controller to ignore the data toggle pattern that is normally sent and accept all incoming data packets regardless of the data toggle state.

In normal operation, the device controller checks the DATA0/DATA1 bit against the data toggle to determine if the packet is valid. If Data PID does not match the data toggle state bit maintained by the device controller for that endpoint, the Data toggle is considered not valid. If the data toggle is not valid, the device controller assumes the packet was already received and discards the packet (not reporting it to the DCD). To prevent the host controller from re-sending the same packet, the device controller will respond to the error packet by acknowledging it with either an ACK or NYET response.

11.3.4.6.3.3.3 Priming Transmit Endpoints

Priming a transmit endpoint will cause the device controller to fetch the device transfer descriptor (dTD) for the transaction pointed to by the device queue head (dQH).

After the dTD is fetched, it will be stored in the dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to handle a request from the host without the need to follow the linked list, starting at the dQH when the host request is received.

After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint up to the maximum number of endpoints configured at device synthesis time.

After a priming request is complete, an endpoint state of primed is indicated in the USB_UOG_ENDPTSTATUS register. For a primed transmit endpoint, the device controller can respond to an IN request from the host and meet the stringent bus turnaround time of High Speed USB.

Because only the leading data is stored in the device controller FIFO, it is necessary for the device controller to begin filling in behind leading data after the transaction starts. The FIFO must be sized to account for the maximum latency that can be incurred by the system memory bus. More information about FIFO sizing is presented in section .

11.3.4.6.3.3.4 Priming Receive Endpoints

Priming receive endpoints is identical to priming of transmit endpoints from the point of view of the DCD. At the device controller the major difference in the operational model is that there is no data movement of the leading packet data simply because the data is to be received from the host.

Note as part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Thus, the size of the RX FIFO does not scale with the number of endpoints.

11.3.4.6.4 Operational Model For Packet Transfers

All transactions on the USB bus are initiated by the host and in turn, the device must respond to any request from the host within the turnaround time stated in the USB 2.0 Specification.

At USB 1.1 Full or Low Speed rates, this turnaround time was significant and the USB 1.1 device controllers were architected so that the device controller could access main memory or interrupt a host protocol processor in order to respond to the USB 1.1 transaction. The architecture of the USB 2.0 device controller must be different because same methods will not meet USB 2.0 High-speed turnaround time requirements by simply increasing clock rate.

A USB host will send requests to the device controller in an order that can not be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However, the order of packet requests is predictable when the endpoint number and direction is considered. For example, if endpoint 3 (transmit direction) is configured as a bulk pipe, then we can expect the host will send IN

requests to that endpoint. This device controller is architected in such a way that it can prepare packets for each endpoint/direction in anticipation of the host request. The process of preparing the device controller to send or receive data in response to host initiated transaction on the bus is referred to as "priming" the endpoint. This term will be used throughout the following documentation to describe the device controller operation so the DCD can be architected properly use priming. Further, note that the term "flushing" is used to describe the action of clearing a packet that was queued for execution.

11.3.4.6.4.1 Interrupt/Bulk Endpoint Operational Model

The behaviors of the device controller for interrupt and bulk endpoints are identical.

All valid IN and OUT transactions to bulk pipes will handshake with a NAK unless the endpoint had been primed. Once the endpoint has been primed, data delivery will commence.

A dTD will be retired by the device controller when the packets described in the transfer descriptor have been completed. Each dTD describes N packets to be transferred according to the USB Variable Length transfer protocol. The formula and table on the following page describe how the device controller computes the number and length of the packets to be sent/received by the USB vary according to the total number of bytes and maximum packet length.

With Zero Length Termination (ZLT) = 0

N = INT(Number Of Bytes/Max. Packet Length) + 1

With Zero Length Termination (ZLT) = 1

N = MAXINT(Number Of Bytes/Max. Packet Length)

Table 11-120. Variable Length Transfer Protocol Example (ZLT = 0)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	3	256	256	0
512	512	2	512	0	

Table 11-121. Variable Length Transfer Protocol Example (ZLT = 1)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	2	256	256	
512	512	1	512		

NOTE

The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.

TX-dTD is complete when:

- All packets described dTD were successfully transmitted. *** Total bytes in dTD will equal zero when this occurs.

RX-dTD is complete when:

- All packets described in dTD were successfully received. *** Total bytes in dTD will equal zero when this occurs.
- A short packet (number of bytes < maximum packet length) was received. *** This is a successful transfer completion; DCD must check Total Bytes in dTD to determine the number of bytes that are remaining. From the total bytes remaining in the dTD, the DCD can compute the actual bytes received.
- A long packet was received (number of bytes > maximum packet size) OR (total bytes received > total bytes specified). *** This is an error condition. The device controller will discard the remaining packet, and set the Buffer Error bit in the dTD. In addition, the endpoint will be flushed and the USBERR interrupt will become active.

On the successful completion of the packet(s) described by the dTD, the active bit in the dTD will be cleared and the next pointer will be followed when the Terminate bit is clear. When the Terminate bit is set, the device controller will flush the endpoint/direction and cease operations for that endpoint/direction.

On the unsuccessful completion of a packet (see long packet above), the dQH will be left pointing to the dTD that was in error. In order to recover from this error condition, the DCD must properly reinitialize the dQH by clearing the active bit and update the nextTD pointer before attempting to re-prime the endpoint.

NOTE

All packet level errors such as a missing handshake or CRC error will be retried automatically by the device controller.

There is no required interaction with the DCD for handling such errors.

11.3.4.6.4.1.1 Interrupt/Bulk Endpoint Bus Response Matrix

The table below shows the response matrix for Interrput/Bulk Endpoint Bus.

Table 11-122. Interrupt/Bulk Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	Ignore	Ignore	Ignore	N/A	N/A
In	STALL	NAK	Transmit	BS Error	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK
Ping	STALL	NAK	ACK	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

NOTE

BS Error = Force Bit Stuff Error

NYET/ACK - NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR - System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

11.3.4.6.4.2 Control Endpoint Operation Model

11.3.4.6.4.2.1 Setup Phase

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase. The device controller will always accept the setup phase unless the setup lockout is engaged.

The setup lockout will engage so that future setup packets are ignored. Lockout of setup packets ensures that while software is reading the setup packet stored in the queue head, that data is not written as it is being read potentially causing an invalid setup packet.

The setup lockout mechanism can be disabled and a new tripwire type semaphore will ensure that the setup packet payload is extracted from the queue head without being corrupted be an incoming setup packet. This is the preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

- Disable Setup Lockout by writing 1 to Setup Lockout Mode (SLOM) in . (once at initialization). Setup lockout is not necessary when using the tripwire as described below.

NOTE

Leaving the Setup Lockout Mode As 0 will result in pre-2.3 hardware behavior.

- After receiving an interrupt and inspecting to determine that a setup packet was received on a particular pipe:
  a. Write 1 to clear corresponding bit .
  b. Write 1 to Setup Tripwire (SUTW) in register.
  c. Duplicate contents of dQH.SetupBuffer into local software byte array.
  d. Read Setup TripWire (SUTW) in register. (if set - continue; if cleared - goto 2)
  e. Write 0 to clear Setup Tripwire (SUTW) in register.
  f. Process setup packet using local software byte array copy and execute status/ handshake phases.

NOTE

After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed & deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

11.3.4.6.4.2.2 Data Phase

Following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer.

After priming the packet, the DCD must verify a new setup packet has not been received by reading the USB.ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime will complete when the associated bit in the register is zero and the associated bit in the register is a one. If a prime fails, ie. The bit goes to zero and the Endpoint Status (USB_ENDPTSTAT) bit is not set, then the prime has failed. This can only be due to improper setup of the dQH, dTD or a setup arriving during the prime operation. If a new setup packet is indicated after the ENDPTPRIME bit is cleared, then the transfer descriptor can be freed and the DCD must reinterpret the setup packet.

Should a setup arrive after the data stage is primed, the device controller will automatically clear the prime status () to enforce data coherency with the setup packet.

NOTE

- The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.
- Error handling of data phase packets is the same as bulk packets described previously.

11.3.4.6.4.2.3 Status Phase

Similar to the data phase, the DCD must create a transfer descriptor (with byte length equal zero) and prime the endpoint for the status phase.

The DCD must also perform the same checks of the USB.ENDPTSETUPSTAT as described above in the data phase.

NOTE

- The MULT field in the dQH must be set to 00 for bulk, interrupt, and control endpoints.
- Error handling of data phase packets is the same as bulk packets described previously.

11.3.4.6.4.2.4 Control Endpoint Bus Response Matrix

Shown in the following table is the device controller response to packets on a control endpoint according to the device controller state.

The table below shows the response matrix for the Control Endpoint Bus.

Table 11-123. Control Endpoint Bus Response Matrix

Token Type	Endpoint State					Setup Lockout
	Stall	Not Primed	Primed	Underflow	Overflow	
Setup	ACK	ACK	ACK	N/A	SYSERR	
In	STALL	NAK	Transmit	BS Error	N/A	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK	N/A
Ping	STALL	NAK	ACK	N/A	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NYET/ACK - NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR - System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

11.3.4.6.4.3 Isochronous Endpoint Operational Model

Isochronous endpoints are used for real-time scheduled delivery of data and their operational model is significantly different than the host throttled Bulk, Interrupt, and Control data pipes.

Real time delivery by the device controller will be accomplished by the following:

- Exactly MULT Packets per (micro) Frame are transmitted/received. Note: MULT is a two-bit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.
- NAK responses are not used. Instead, zero length packets are sent in response to an IN request to an unprimed endpoints. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (micro) frame. If the ISO-dTD is still active after that frame, then the ISO-dTD will be held ready until executed or canceled by the DCD.

An EHCI compatible host controller uses the periodic frame list to schedule data exchanges to Isochronous endpoints. The operational model for device mode does not use such a data structure. Instead, the same dTD used for Control/Bulk/Interrupt endpoints is also used for isochronous endpoints. The difference is in the handling of the dTD.

The first difference between bulk and ISO-endpoints is that priming an ISO-endpoint is a delayed operation such that an endpoint will become primed only after a SOF is received. After the DCD writes the prime bit, the prime bit will be cleared as usual to indicate to software that the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD but occurs so that the device controller can match the dTD to a specific (micro)frame.

Another difference with isochronous endpoints is that the transaction must wholly complete in a (micro)frame. Once an ISO transaction is started in a (micro)frame it will retire the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition.

The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device controller will force retire the ISO-dTD and move to the next ISO-dTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction will stay primed indefinitely. This means it is up to software discard transmit ISO-dTDs that pile up from a failure of the host to move the data.

Finally, the last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is noted by setting the *Transaction Error* bit and the data is stored as usual for the application software to sort out.

- TX Packet Retired
  - MULT counter reaches zero.
  - Fulfillment Error [*Transaction Error* bit is set]

- # Packets Occurred > 0 AND # Packets Occurred < MULT

NOTE

For TX-ISO, MULT Counter can be loaded with a lesser value in the dTD Multiplier Override field in hardware versions 2.3 and later. If the Multiplier Override is zero, the MULT Counter is initialized to the Multiplier in the QH.

- RX Packet Retired:
  - MULT counter reaches zero.
  - Non-MDATA Data PID is received**
    - ** Exit criteria only valid in hardware version 2.3 or later. Previous to hardware version 2.3, any PID sequence that did not match the MULT field exactly would be flagged as a transaction error due to PID mismatch or fulfillment error.
  - Overflow Error:
    - Packet received is > maximum packet length. [*Buffer Error* bit is set]
    - Packet received exceeds total bytes allocated in dTD. [*Buffer Error* bit is set]
  - Fulfillment Error [*Transaction Error* bit is set]
    - # Packets Occurred > 0 AND # Packets Occurred < MULT
  - CRC Error [*Transaction Error* bit is set]

NOTE

For ISO, when a dTD is retired, the next dTD is primed for the next frame. For continuous (micro)frame to (micro)frame operation the DCD should ensure that the dTD linked-list is out ahead of the device controller by at least two (micro)frames.

11.3.4.6.4.3.1 Isochronous Pipe Synchronization

When it is necessary to synchronize an isochronous data pipe to the host, the (micro) frame number (USB_UOG_FRINDEX register) can be used as a marker.

To cause a packet transfer to occur at a specific (micro) frame number [N], the DCD should interrupt on SOF during frame N-1. When the USB_UOG_FRINDEX=N-1, the DCD must write the prime bit. The device controller will prime the isochronous endpoint in (micro) frame N-1 so that the device controller will execute delivery during (micro) frame N.

NOTE

Priming an endpoint towards the end of (micro) frame N-1 will not guarantee delivery in (micro) frame N. The delivery may actually occur in (micro) frame N+1 if device controller does

not have enough time to complete the prime before the SOF for packet N is received.

11.3.4.6.4.3.2 Isochronous Endpoint Bus Response Matrix

The following table shows the response matrix for the Isochronous Endpoint Bus.

Table 11-124. Isochronous Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	STALL	STALL	STALL	N/A	N/A
In	NULL Packet	NULL Packet	Transmit	BS Error	N/A
Out	Ignore	Ignore	Receive	N/A	Drop Packet
Ping	Ignore	Ignore	Ignore	Ignore	Ignore
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

1. BS Error = Force Bit Stuff Error

   NULL Packet = Zero Length Packet

11.3.4.6.5 Managing Queue Heads

The following figure shows the End Point Queue Head.

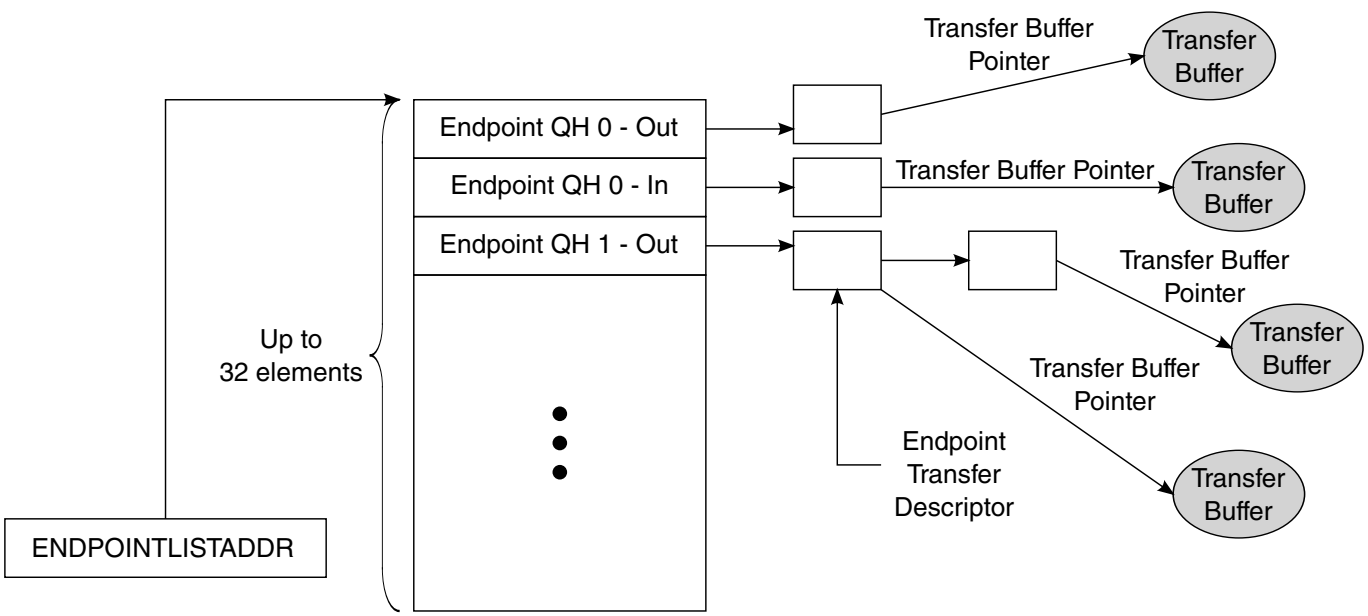


Figure 11-82. End Point Queue Head Diagram

The device queue head (dQH) points to the linked list of transfer tasks, each depicted by the device Transfer Descriptor (dTD). An area of memory pointed to by USB.ENDPOINTLISTADDR contains a group of all dQH's in a sequential list as shown in Figure 11-82. The even elements in the list of dQH's are used for receive endpoints (OUT/SETUP) and the odd elements are used for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. Once the dTD has been retired, it will no longer be part of the linked list from the queue head. Therefore, software is required to track all transfer descriptors because pointers will no longer exist within the queue head once the dTD is retired (see section Software Link Pointers).

In addition to the current and next pointers and the dTD overlay examined in section Operational Model For Packet Transfers, the dQH also contains the following parameters for the associated endpoint: Multipler, Maximum Packet Length, Interrupt On Setup. The complete initialization of the dQH including these fields is demonstrated in the next section.

11.3.4.6.5.1 Queue Head Initialization

One device queue head must be initialized for each active endpoint.

To initialize a device queue head:

- Write the wMaxPacketSize field as required by the USB Chapter 9 or application specific protocol.
- Write the multiplier field to 0 for control, bulk, and interrupt endpoints. For ISO endpoints, set the multiplier to 1,2, or 3 as required bandwidth and in conjunction with the USB Chapter 9 protocol.

NOTE

In FS mode, the multiplier field can only be 1 for ISO endpoints.

- Write the next dTD Terminate bit field to 1.
- Write the Active bit in the status field to 0.
- Write the Halt bit in the status field to 0.

NOTE

The DCD must only modify dQH if the associated endpoint is not primed and there are no outstanding dTD's.

11.3.4.6.5.2 Operational Model For Setup Transfers

As discussed in section Control Endpoint Operation Model, setup transfer requires special treatment by the DCD. A setup transfer does not use a dTD but instead stores the incoming data from a setup packet in an 8-byte buffer within the dQH.

Upon receiving notification of the setup packet, the DCD should handle the setup transfer as demonstrated here:

1. Copy setup buffer contents from dQH - RX to software buffer.
2. Acknowledge setup backup by writing a "1" to the corresponding bit in ENDPTSETUPSTAT.

NOTE

- The acknowledge must occur before continuing to process the setup packet.
- After the acknowledge has occurred, the DCD must not attempt to access the setup buffer in the dQH - RX. Only the local software copy should be examined.

3. Check for pending data or status dTD's from previous control transfers and flush if any exist as discussed in section Flushing/De-priming an Endpoint.
4. Decode setup packet and prepare data phase [optional] and status phase transfer as required by the USB Chapter 9 or application specific protocol.

NOTE

It is possible for the device controller to receive setup packets before previous control transfers complete. Existing control packets in progress must be flushed and the new control packet completed.

11.3.4.6.6 Managing Transfers with Transfer Descriptors

11.3.4.6.6.1 Software Link Pointers

It is necessary for the DCD software to maintain head and tail pointers to the for the linked list of dTDs for each respective queue head.

This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list. The following figure shows the Software Link Pointers.

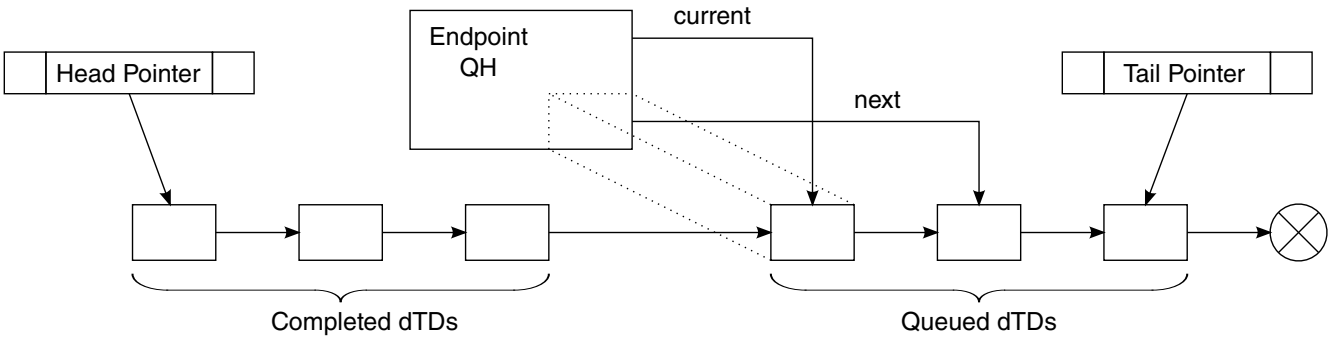


Figure 11-83. Software Link Pointers

NOTE

To conserve memory, the reserved fields at the end of the dQH can be used to store the Head & Tail pointers, but it still remains the responsibility of the DCD to maintain the pointers.

11.3.4.6.6.2 Building a Transfer Descriptor

Before a transfer can be executed from the linked list, a dTD must be built to describe the transfer.

Use the following procedure for building dTDs.

Allocate 8-DWord dTD block of memory aligned to 8-DWord boundaries. Example: bit address 4:0 would be equal to "00000"

Write the following fields:

1. Initialize first 7 DWords to 0.
2. Set the terminate bit to 1.
3. Fill in total bytes with transfer size.
4. Set the interrupt on complete if desired.
5. Initialize the status field with the active bit set to 1 and all remaining status bits set to 0.
6. Fill in buffer pointer page 0 and the current offset to point to the start of the data buffer.
7. Initialize buffer pointer page 1 through page 4 to be one greater than each of the previous buffer pointer.

11.3.4.6.6.3 Executing A Transfer Descriptor

To safely add a dTD, the DCD must be follow this procedure which will handle the event where the device controller reaches the end of the dTD list at the same time a new dTD is being added to the end of the list.

Determine whether the link list is empty: Check DCD driver to see if pipe is empty (internal representation of linked-list should indicate if any packets are outstanding).

- Case 1: Link list is empty
  a. Write dQH next pointer AND dQH terminate bit to 0 as a single DWord operation.
  b. Clear active & halt bit in dQH (in case set from a previous error).
  c. Prime endpoint by writing 1 to correct bit position in .
- Case 2: Link list is not empty
  a. Add dTD to end of linked list.
  b. Read correct prime bit in - if 1 DONE.
  c. Set ATDTW bit in USBCMD register to 1.
  d. Read correct status bit in . (store in tmp. variable for later)
  e. Read ATDTW bit in USBCMD register.
     - If 0 goto 3.
     - If 1 continue to 6.
  f. Write ATDTW bit in USBCMD register to 0.
  g. If status bit read in (3) is 1 DONE.
  h. If status bit read in (3) is 0 then Goto Case 1: Step 1.

11.3.4.6.6.4 Transfer Completion

After a dTD has been initialized and the associated endpoint primed the device controller will execute the transfer upon the host-initiated request. The DCD will be notified with a USB interrupt if the Interrupt On Complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure.

NOTE

Multiple dTD can be completed in a single endpoint complete notification. After clearing the notification, DCD must search the dTD linked list and retire all dTDs that have finished (Active bit cleared).

By reading the status fields of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following combination of status bits:

- Active = 0
- Halted = 0
- Transaction Error = 0
- Data Buffer Error = 0

Should any combination other than the one shown above exist, the DCD must take proper action. Transfer failure mechanisms are indicated in the Device Error Matrix.

In addition to checking the status bit the DCD must read the Transfer Bytes field to determine the actual bytes transferred. When a transfer is complete, the Total Bytes transferred is by decremented by the actual bytes transferred. For Transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according the USB variable length packet protocol.

11.3.4.6.6.5 Flushing/De-priming an Endpoint

It is necessary for the DCD to flush to de-prime one more endpoints on a USB device reset or during a broken control transfer.

There may also be application specific requirements to stop transfers in progress. The following procedure can be used by the DCD to stop a transfer in progress:

1. Write a '1' to the corresponding bit(s) in .
2. Wait until all bits in are '0'.
   - Software note: this operation may take a large amount of time depending on the USB bus activity. It is not desirable to have this wait loop within an interrupt service routine.
3. Read to ensure that for all endpoints commanded to be flushed, that the corresponding bits are now '0'. If the corresponding bits are '1' after step #2 has finished, then the flush failed as described in the following:
   - Explanation: In very rare cases, a packet is in progress to the particular endpoint when commanded flush using . A safeguard is in place to refuse the flush to ensure that the packet in progress completes successfully. The DCD may need to repeatedly flush any endpoints that fail to flush be repeating steps 1-3 until each endpoint is successfully flushed.

11.3.4.6.6.6 Device Error Matrix

The following table summarizes packet errors that are not automatically handled by the Device Controller.

Table 11-125. Device Error Matrix

Error	Direction	Packet Type	Data Buffer Error Bit	Transaction Error Bit
Overflow **	RX	Any	1	0
ISO Packet Error	RX	ISO	0	1
ISO Fulfillment Error	Both	ISO	0	1

Notice that the device controller handles all errors on Bulk/Control/Interrupt Endpoints except for a data buffer overflow. However, for ISO endpoints, errors packets are not retried and errors are tagged as indicated. The table below describes the errors.

Table 11-126. Error Descriptions

Error	Description
Overflow	Number of bytes received exceeded max. packet size or total buffer length. ** This error will also set the Halt bit in the dQH and if there are dTDs remaining in the linked list for the endpoint, then those will not be executed.
ISO Packet Error	CRC Error on received ISO packet. Contents not guaranteed to be correct.
ISO Fulfillment Error	Host failed to complete the number of packets defined in the dQH mult field within the given (micro)frame. For scheduled data delivery the DCD may need to readjust the data queue because a fulfillment error will cause Device Controller to cease data transfers on the pipe for one (micro)frame. During the "dead" (micro)frame, the Device Controller reports error on the pipe and primes for the following frame.

11.3.4.6.7 Servicing Interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

11.3.4.6.7.1 High-Frequency Interrupts

High frequency interrupts in particular should be handled in the order below. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

The table below describes the High frequency interrupt events.

Table 11-127. High Frequency Interrupt Events

Execution Order	Interrupt	Action
1a	USB Interrupt - USB.ENDPTSETUPSTATUS	Copy contents of setup buffer and acknowledge setup packet (as indicated in Figure 11-82 shows the End Point Queue Head). Process setup packet according to USB 2.0 Chapter 9 or application specific protocol.

Table continues on the next page...

Table 11-127. High Frequency Interrupt Events (continued)

1b	USB Interrupt[1] - USB.ENDPTCOMPLETE	Handle completion of dTD as indicated in Figure 11-82 shows the End Point Queue Head.
2	SOF Interrupt	Action as deemed necessary by application. This interrupt may not have a use in all applications.

1. It is likely that multiple interrupts to stack up on any call to the Interrupt Service Routine AND during the Interrupt Service Routine.

11.3.4.6.7.2 Low-Frequency Interrupts

The low frequency interrupts can be handled in any order because they do not occur often in comparison to the high-frequency interrupts.

The table below shows the Low frequency interrupt events.

Table 11-128. Low Frequency Interrupt Events

Interrupt	Action
Port Change	Change software state information.
Sleep Enable (Suspend)	Change software state information. Low power handling as necessary.
Reset Received	Change software state information. Abort pending transfers.

11.3.4.6.7.3 Error Interrupts

Error interrupts will be least frequent and should be placed last in the interrupt service routine.

The following table shows the error interrput events.

Table 11-129. Error Interrupt Events

Interrupt	Action
USB Error Interrupt	This error is redundant because it combines USB Interrupt and an error status in the dTD. The DCD will more aptly handle packet-level errors by checking dTD status field upon receipt of USB Interrupt (w/ USB.ENDPTCOMPLETE).
System Error	Unrecoverable error. Immediate Reset of core; free transfers buffers in progress and restart the DCD.

11.3.5 USB Non-Core Memory Map/Register Definition

There are two kinds of registers in the USB module: USB core registers and USB non-core registers.

USB core registers are used to control USB core functions, and more independent of USB features. Each USB controller core has its own core registers.

USB non-core registera are additional to USB core registers, and more dependent on USB features. i.MX series products vary in non-core registers.

This section describes only the USB non-core registers. For detailed descriptions of USB core registers, please refer to Register Interface .

NOTE

- For reserved bits, please preserve the value when writing (read its reset value, then write this value back)
- USBNC_USB_" prefix in register name indicates it is a USB non-core register.

USBNC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30B1_0400	USBNC_OTG1_CTRL1	32	R/W	3000_1000h	11.3.5.1/ 3270
30B1_0404	USBNC_OTG1_CTRL2	32	R/W	0F00_0000h	11.3.5.2/ 3272
30B1_0430	USBNC_OTG1_PHY_CFG1	32	R/W	1533_71CBh	11.3.5.3/ 3275
30B1_0434	USBNC_OTG1_PHY_CFG2	32	R/W	F001_0020h	11.3.5.4/ 3279
30B1_043C	USBNC_OTG1_PHY_STATUS	32	R	0000_0000h	11.3.5.5/ 3282
30B1_0450	USBNC_ADP_CFG1	32	R/W	0000_C000h	11.3.5.6/ 3286
30B1_0454	USBNC_ADP_CFG2	32	R/W	0046_4010h	11.3.5.7/ 3288
30B1_0458	USBNC_ADP_STATUS	32	R	0000_0000h	11.3.5.8/ 3289
30B2_0400	USBNC_OTG2_CTRL1	32	R/W	3000_1000h	11.3.5.1/ 3270
30B2_0404	USBNC_OTG2_CTRL2	32	R/W	0F00_0000h	11.3.5.2/ 3272
30B2_0430	USBNC_OTG2_PHY_CFG1	32	R/W	1533_71CBh	11.3.5.3/ 3275
30B2_0434	USBNC_OTG2_PHY_CFG2	32	R/W	F001_0020h	11.3.5.4/ 3279

Table continues on the next page...

USBNC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30B2_043C	USBNC_OTG2_PHY_STATUS	32	R	0000_0000h	11.3.5.5/ 3282
30B3_0400	USBNC_HSIC_CTRL1	32	R/W	3000_1000h	11.3.5.1/ 3270
30B3_0404	USBNC_HSIC_CTRL2	32	R/W	0F00_0000h	11.3.5.2/ 3272
30B3_0440	USBNC_UH_HSICPHY_CFG1	32	R/W	C224_3A1Dh	11.3.5.9/ 3290

11.3.5.1 USBNC_*n*_CTRL1

The USB_n_PHY_CTRL1 register controls the integration specific features of the USB PHY modules. These features are not directly related to basic USB functionality, but control special features, interfacing on the USB ports, as well as power control and wake-up functionality.

Address: 30B1_0200h base + 200h offset + (65536d × i), where i=0d to 2d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	WIR	Reserved	WKUP_DPDM_EN	Reserved											WKUP_VBUS_EN	WKUP_ID_EN
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WKUP_SW	WKUP_SW_EN	Reserved			WIE	PWR_POL	OVER_CUR_POL	OVER_CUR_DIS	Reserved						
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

USBNC_*n*_CTRL1 field descriptions

Field	Description
31 WIR	Wake-up Interrupt Request This bit indicates that a wake-up interrupt request is received on the OTG1 port. This bit is cleared by disabling the wake-up interrupt (clearing bit "OWIE"). 1 Wake-up Interrupt Request received 0 No wake-up interrupt request received
30 Reserved	This field is reserved.
29 WKUP_DPDM_EN	Wake-up on DPDM change enable 1 (Default) DPDM changes wake-up to be enabled, it is for device only. 0 DPDM changes wake-up to be disabled only when VBUS is 0.
28–18 Reserved	This field is reserved.
17 WKUP_VBUS_EN	Wake-up on VBUS change enable 1 Enable 0 Disable
16 WKUP_ID_EN	Wake-up on ID change enable 1 Enable 0 Disable
15 WKUP_SW	Software Wake-up 1 Force wake-up 0 Inactive
14 WKUP_SW_EN	Software Wake-up Enable 1 Enable 0 Disable
13–11 Reserved	This field is reserved. Reserved. Do not write to.
10 WIE	Wake-up Interrupt Enable This bit enables or disables the OTG1 wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 1 Interrupt Enabled 0 Interrupt Disabled
9 PWR_POL	Power Polarity This bit should be set according to PMIC Power Pin polarity. 1 PMIC Power Pin is High active. 0 PMIC Power Pin is Low active.
8 OVER_CUR_POL	Polarity of Overcurrent The polarity of OTG1/OTG2 port overcurrent event

Table continues on the next page...

USBNC_*n*_CTRL1 field descriptions (continued)

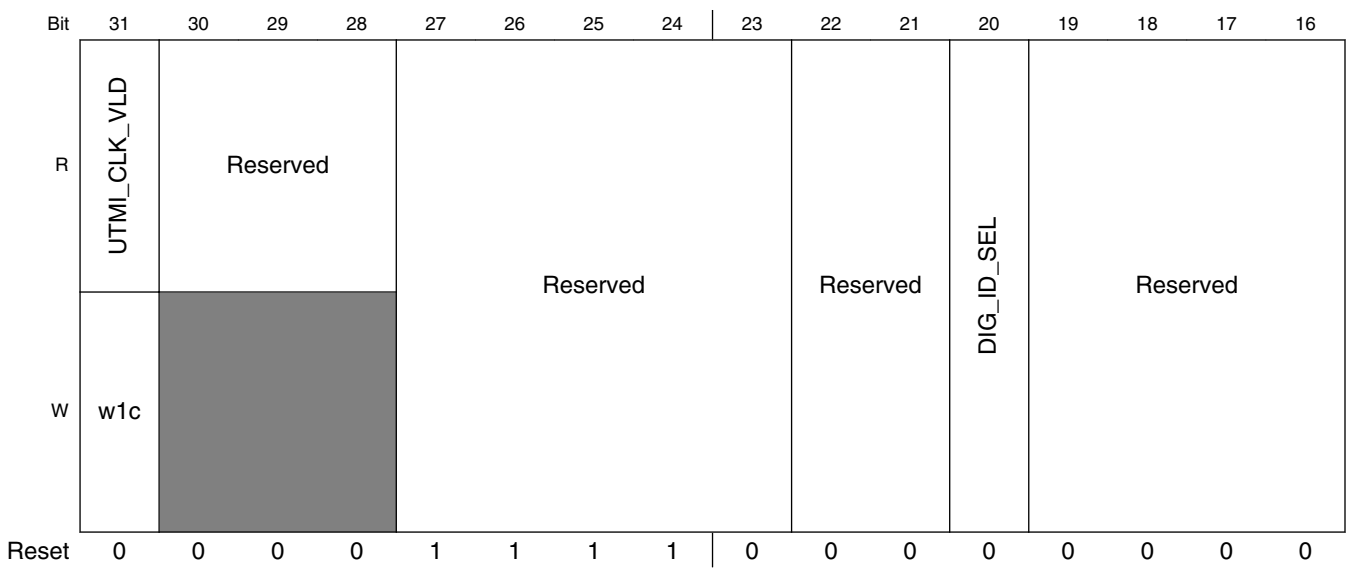
Field	Description
	1 Low active (low on this signal represents an overcurrent condition) 0 High active (high on this signal represents an overcurrent condition)
7 OVER_CUR_DIS	Disable Overcurrent Detection 1 Disables overcurrent detection 0 Enables overcurrent detection
Reserved	This field is reserved.

11.3.5.2 USBNC_*n*_CTRL2

The USB_OTGx_PHY_CTRL2 register allows observation of the UTMI Clock Valid status along with control of selected inputs to the USB OTG PHY transceiver.

Some of these input signals may be used for USB out-of-band signaling. Customers should use caution when overriding any UTMI signals since that will interfere with normal USB data communication.

Address: 30B1_0200h base + 204h offset + (65536d × i), where i=0d to 2d



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMPULLDOWN_OVERRIDEEN	DMPULLDOWN_OVERRIDE	DPPULLDOWN_OVERRIDEEN	DPPULLDOWN_OVERRIDE	XCVRSEL_OVERRIDEEN	XCVRSEL_OVERRIDE		OPMODE_OVERRIDEEN	OPMODE_OVERRIDE		TERMSEL_OVERRIDEEN	TERMSEL_OVERRIDE	LOWSPEED_EN	AUTURESUME_EN	VBUS_SOURCE_SEL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBNC_*n*_CTRL2 field descriptions

Field	Description
31 UTMI_CLK_VLD	Indicate whether PHY clock is valid. Write 1 to clear 0 UTMI clock to USB PHY is not toggling (Default) 1 UTMI clock to USB PHY has toggled several times
30–28 Reserved	This field is reserved.
27–23 Reserved	This field is reserved. 0B11110 Default
22–21 Reserved	This field is reserved. Customers should not change the value of this bit field.
20 DIG_ID_SEL	Selects whether the USB OTG ID pin function is connected to the PHY's USB_OTG*_ID pin or a GPIO pin. The ID pin state is observed on USBNC_n_PHY_STATUS[ID_DIG]. 0 Use the USB_OTG*_ID pin for the USB OTG ID pin detection function(default) 1 Use the pin configured by the IOMUXC_USB_OTG*_ID_SELECT_INPUT register for the USB OTG ID pin detection function
19–16 Reserved	This field is reserved. Customers should not change the value of this bit field.
15 DMPULLDOWN_ OVERRIDEEN	Enables override of control of the DM pulldown resistor instead of the state set by the USB controller. 0 USB controller enables/disables the DM pulldown resistor in the USB PHY. 1 Use the value set by the USBNC_n_CTRL2[DMPULLDOWN_OVERRIDE] bit field to enable/disable the DM pulldown resistor in the USB PHY.
14 DMPULLDOWN_ OVERRIDE	Controls state of DM pulldown resistor if USBNC_n_CTRL2[DMPULLDOWN_OVERRIDEEN] is set to 1'b1. 0 DM pulldown resistor disabled 1 DM pulldown resistor enabled

Table continues on the next page...

USBNC_*n*_CTRL2 field descriptions (continued)

Field	Description
13 DPPULLDOWN_ OVERRIDEEN	Enables override of control of the DP pulldown resistor instead of using the state set by the USB controller. 0 USB controller enables/disables the DP pulldown resistor in the USB PHY. 1 Use the value set by the USBNC_n_CTRL2[DPPULLDOWN_OVERRIDE] bit field to enable/disable the DP pulldown resistor in the USB PHY.
12 DPPULLDOWN_ OVERRIDE	Controls state of DP pulldown resistor if USBNC_n_CTRL2[DPPULLDOWN_OVERRIDEEN] is set to 1'b1. 0 DP pulldown resistor disabled 1 DP pulldown resistor enabled
11 XCVRSEL_ OVERRIDEEN	Enables override of control of the UTMI XcvrSelect signals to the USB OTG PHY instead of using the state normally set by the USB controller. 0 The state of the UTMI XcvrSelect signals to the USB PHY is set by the USB controller. 1 The state of the UTMI XcvrSelect signals to the USB PHY is set by the values in the USBNC_x_CTRL2[XCVRSEL_OVERRIDE] bit field.
10–9 XCVRSEL_ OVERRIDE	Controls the state of the UTMI XcvrSelect signals to the USB OTG PHY if the value of USBNC_x_CTRL2[XCVRSEL_OVERRIDEEN] is set to 1'b1. See the *UTMI+ Specification* for descriptions of the functions of the XcvrSelect signals.
8 OPMODE_ OVERRIDEEN	Enables override of control of the UTMI OpMode signals to the USB OTG PHY instead of using the state normally set by the USB controller. 0 The state of the UTMI OpMode signals to the USB PHY is set by the USB controller. 1 The state of the UTMI OpMode signals to the USB PHY is set by the values in the USBNC_x_CTRL2[OPMODE_OVERRIDE] bit field.
7–6 OPMODE_ OVERRIDE	Controls the state of the UTMI XcvrSelect signals to the USB OTG PHY if the value of USBNC_x_CTRL2[OPMODE_OVERRIDEEN] is set to 1'b1. See the *UTMI+ Specification* for descriptions of the functions of the OpMode signals.
5 TERMSEL_ OVERRIDEEN	Enables override of control of the UTMI TermSelect signal to the USB OTG PHY instead of using the state normally set by the USB controller. 0 The state of the UTMI TermSelect signal to the USB PHY is set by the USB controller. 1 The state of the UTMI TermSelect signal to the USB PHY is set by the value in the USBNC_x_CTRL2[TERMSEL_OVERRIDE] bit field.
4 TERMSEL_ OVERRIDE	Controls the state of the UTMI TermSelect signal to the USB OTG PHY if the value of USBNC_x_CTRL2[TERMSEL_OVERRIDEEN] is set to 1'b1. See the *UTMI+ Specification* for descriptions of the function of the TermSelect signal.
3 LOWSPEED_EN	Set if AUTURESUME_EN is set and works on low speed. 0 Default
2 AUTURESUME_ EN	Auto Resume Enable This bit field is for UTMI OTG PHY host mode only (UH core does not have this feature since HSIC PHY does not support auto resume). To set USB, it will send resume with 32 KHz clock when it detects remote wakeup from device. This feature is useful if device drives a very short remote wakeup (minimal value is 1 ms for USB2 spec) and system 24 MHz is off during suspend mode. 0 Default

Table continues on the next page...

USBNC_*n*_CTRL2 field descriptions (continued)

Field	Description
VBUS_SOURCE_SEL	VBUS source select used to detect a VBUS wakeup event. This bit field is for UTMI OTG PHY only (UH core and HSIC PHY have no such feature). 2'b00 vbus_valid others sess_valid

11.3.5.3 USB OTG PHY Configuration Register 1 (USBNC_*n*_PHY_CFG1)

The USB_OTGx_PHY_CFG1 register allows control of selected inputs to the USB OTG PHY.

Most of these signals are used for parametric tuning of the USB transceiver functions.

One signal is used to allow persistent control of the USB_OTG1_CHD_B external pin.

Address: 30B1_0200h base + 230h offset + (65536d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CHRGDET_Megamix	TXPREEMPPULSETUNE0	TXPREEMPAMPTUNE0		TXRESTUNE0		TXRISETUNE0		TXVREFTUNE0				TXFSLSTUNE0			
Reset	0	0	0	1	0	1	0	1	0	0	1	1	0	0	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	0	TXHSXVTUNE0		OTGTUNE0			SQRXTUNE0		COMPDISTUNE0				FSEL			COMMONONN
Reset	0	1	1	1	0	0	0	1	1	1	0	0	1	0	1	1

USBNC_*n*_PHY_CFG1 field descriptions

Field	Description
31 CHRGDET_Megamix	USB_OTG1_CHD_B output control This signal controls the state of the USB_OTG1_CHD_B external pin together with the CHRGDET (USBNC_USB_OTG1_PHY_STATUS[29] signal during periods when the SOC is fully powered. If either CHRGDET_Megamix or CHRGDET are set to 1'b1, then USB_OTG1_CHD_B will be pulled low. If both CHRGDET_Megamix and CHRGDET are set to 1'b0, then USB_OTG1_CHD_B will be pulled high by its external open-drain pull-up resistor.

Table continues on the next page...

USBNC_*n*_PHY_CFG1 field descriptions (continued)

Field	Description
	Note: The instance of this bit field for the USB OTG2 PHY does not produce results on an external pin. The R/W register bit may still be used as a memory location to store results of Battery Charger detection tests for USB OTG2 PHY. 0 The external state of USB_OTG1_CHD_B is only controlled by the state of the CHRGDET signal 1 The external state of USB_OTG1_CHD_B is forced low
30 TXPREEMPPULSETUNE0	HS Transmitter Pre-Emphasis Duration Control This signal controls the duration for which the HS pre-emphais current is sourced on the the USB_OTG*_DP and USB_OTG*DN pins. The HS Transmitter pre-emphasis duration is defined in terms of unit amounts. One unit of pre-emphasis duration is approximately 580ps and is defined as 1X pre-emphasis duration. This signal is valid only if TXPREEMPAMPTUNE (USBNC_USB_OTGx_PHY_CFG1[29:28]) is NOT set to 2'b00. 0 2X, long pre-emphasis current duration (design default) 1 1X, short pre-emphasis current duration
29–28 TXPREEMPAMPTUNE0	HS Treansmitter Pre-Emphasis Current Control This signal controls the amount of current sourced to the USB_OTG*_DP and USB_OTG*_DN pins after a J-to-K or K-to-J transition. The HS Transmitter pre-emphasis current is defined in terms of unit amounts. One unit amount is approximately 600uA and is defined as 1X pre-emphasis current. 00 HS Transmitter pre-emphasis is disabled 01 HS Transmitter pre-emphasis circuit sources 1X pre-emphasis current (design default) 10 HS Transmitter pre-emphasis circuit sources 2X pre-emphasis current 11 HS Transmitter pre-emphasis circuit sources 3X pre-emphasis current
27–26 TXRESTUNE0	USB Source Impedance Adjustment In some applications, there can be significant series resistance on the USB DP/DN path between the USB_OTG*_DP/USB_OTG*_DN pins tns and the USB cable. This bus adjusts the driver source impedance to compensate for that added resistance. **Note:** Any setting other than the design default can cause HS/FS signal integrity changes along with variation across process, voltage, and temperature conditions that affect compliance with USB 2.0 specification limits. 00 Source impedance is increased by approximately 1.5 Ω 01 Design default 10 Source impedance is decreased by approximately 2 Ω 11 Source impedance is decreased by approximately 4 Ω
25–24 TXRISETUNE0	HS Transmitter Rise/Fall Time Adjustment This bus adjust the rise/fall times of the high-speed transmitter waveform. 00 -10% 01 Design default 10 +15% 11 +20%
23–20 TXVREFTUNE0	HS DC Voltage Level Adjustment This bus adjust the high-speed transmitter DC level voltage.

Table continues on the next page...

USBNC_*n*_PHY_CFG1 field descriptions (continued)

Field	Description
	0000 -6% 0001 -4% 0010 -2% 0011 Design default 0100 +2% 0101 +4% 0110 +6% 0111 +8% 1000 +10% 1001 +12% 1010 +14% 1011 +16% 1100 +18% 1101 +20% 1110 +22% 1111 +24%
19–16 TXFSLSTUNE0	FS/LS Source Impedance Adjustment This bus adjusts the low- and full-speed single-ended source impedance while driving high. The adjustment values listed are based on nominal process, voltage, and temperature conditions. 0000 +5% 0001 +2.5% 0011 Design default 0111 -2.5% 1111 -5% All other bit settings are reserved and should not be used.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–13 TXHSXVTUNE0	Transmitter High-Speed Crossover Adjustment This bus adjusts the voltage at which the USB_OTG*_DP and USB_OTG*_DN signals cross while transmitting in HS mode. 00 Reserved 01 -15mV 10 +15mV 11 Design default
12–10 OTGTUNE0	VBUS Valid Threshold Adjustment This bus adjust the voltage level for the VBUS VALID threshold. 000 -6% 001 -4.5% 010 -3% 011 -1.5% 100 Design default 101 +1.5%

Table continues on the next page...

USBNC_*n*_PHY_CFG1 field descriptions (continued)

Field	Description
	110 +3% 111 +4.5%
9–7 SQRXTUNE0	Squelch Threshold Adjustment This bus adjusts the voltage level for the receiver threshold used to detect valid high-speed data. 000 +15% 001 +10% 010 +5% 011 Design default 100 -5% 101 -10% 110 -15% 111 -20%
6–4 COMPDISTUNE0	Disconnect Threshold Adjustment This bus adjusts the voltage level for the receiver threshold used to detect a disconnect event at the host. 000 -6% 001 -4.5% 010 -3% 011 -1.5% 100 Design default 101 +1.5% 110 +3% 111 +4.5%
3–1 FSEL	Reference Clock Frequency Select This signal selects the USB OTG PHY reference clock frequency for the USB PLL. **Note:** This SOC is only configured for a 24MHz clock to be routed to the USB OTG PHY PLL clock input. Customers should not change the value of this bit field. 000 9.6 MHz 001 10 MHz 010 12 MHz 011 19.2 MHz 100 20 MHz 101 24 MHz (only valid setting for this SOC) 110 Reserved 111 50 MHz
0 COMMONONN	Common Block Power-Down Control This signal controls the power-down signals in the Bias and PLL blocks when the USB OTG PHY is in Suspend or Sleep modes. 0 In Suspend or Sleep modes, the Bias and PLL blocks remain powered 1 In Suspend or Sleep modes, the Bias and PLL blocks are powered down 0

11.3.5.4 USB OTG PHY Configuration Register 2 (USBNC_*n*_PHY_CFG2)

The USB_OTGx_PHY_CFG2 register allows control of selected inputs to the USB OTG PHY.

Most of these signals are used for configuration of VBUS detection, ADP (Attach Detection Protocol), and Battery Charging ACA (Accesory Charging Adaptor) functions.

Address: 30B1_0200h base + 234h offset + (65536d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															DRVVBUS0
W																
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VBUSVLDEXT	VBUSVLDEXTSEL0	ADPPRBENB0	ADPDISCHRG0	ADPCHRG0	OTGDISABLE0	TXBITSTUFFENH0	TXBITSTUFFEN0	0	LOOPBACKENB0	SLEEPM0	ACAENB0	DCDENB	VDATSRCENB0	VDATDETENB0	CHRGSEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

USBNC_*n*_PHY_CFG2 field descriptions

Field	Description
31–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 DRVVBUS0	VBUS Valid Comparator Enable This signal controls the USB OTG PHY VBUS Valid comparator which indicates whether the voltage on the USB_OTG*_VBUS pin is below the VBUS Valid threshold. The VBUS Valid threshold is nominally 4.75V on this USB PHY. The VBUS Valid threshold can be adjusted using the USBNC_OTGn_PHY_CFG1[OTGTUNE0] bit field. Status of the VBUS Valid comparator, when it is enabled, is reported on the USBNC_OTGn_PHY_STATUS[VBUS_VLD] bit. When OTGDISABLE0 (USBNC_USB_OTGx_PHY_CFG2[10])is set to 1'b0 and DRVVBUS0 is set to 1'b1, the Bandgap circuitry and VBUS Valid comparator are powered, even in Suspend or Sleep mode. DRVVBUS0 should be reset to 1'b0 when the internal VBUS Valid comparator is not required, to reduce quiescent current in Suspend or Sleep mode. 0 The VBUS Valid comparator is disabled 1 The VBUS Valid comparator is enabled
15 VBUSVLDEXT	External VBUS Valid Indicator

Table continues on the next page...

USBNC_*n*_PHY_CFG2 field descriptions (continued)

Field	Description
	This signal is used if a VBUS Valid signal generated from a circuit outside the USB OTG PHY is desired. If the USB OTG PHY is configured for Device mode and VBUSVLDEXTSEL0 (USBNC_USB_OTGx_PHY_CFG2[14] is set to 1'b1, VBUSVLD indicated whether the VBUS signal on the USB cable is valid and enables the pull-up resistor on USB_OTG1_DP. This signal has no effect when the USB OTG PHY is configured for Host mode. 0 The VBUS signal sensed outside the USB OTG PHY is not valid, and the pull-up resistor on USB_OTG*_DP is disabled 1 The VBUS signal sensed outside the USB OTG PHY is valid, and the pull-up resistor on USB_OTG*_DP is enabled
14 VBUSVLDEXTSEL0	External VBUS Valid Select This signal selects whether the VBUSVLDEXT input (USBNC_USB_OTGx_PHY_CFG2[15]) or the internal Session Valid comparator in the USB OTG PHY is used to check if the VBUS signal on USB_OTG*_VBUS is valid and to enable the pull-up resistor on the USB_OTG*_DP pin. The activation of the pull-up resistor on the USB_OTG*_DP pin also depends on the state of the XCVRSEL, OPMODE, TERMSEL, DPPULLDOWN, and DMPULLDOWN signals in the UTMI bus per the UTMI+ specification. 0 The USB OTG PHY internal Session Valid comparator is used to enable the pull-up resistor on the USB_OTG*_DP pin 1 The VBUSVLDEXT signal is used to enable the pull-up resistor on the USB_OTG*_DP pin
13 ADPPRBENB0	ADP Probe Enable This signal determines whether the ADP (Attach Detection Protocol) Probe comparator on the USB_OTG*_VBUS pin is enabled. The ADP Probe threshold is > 0.6V and < 0.75V on this USB PHY. 0 ADP Probe comparator is disabled 1 ADP Probe comparator is enabled
12 ADPDISCHRG0	VBUS Input ADP Discharge Enable This signal controls discharging the USB_OTG*_VBUS pin during ADP. The I_{ADP_SINK} current is >1.1mA and <2.0mA. 0 Disables discharging USB_OTG*_VBUS during ADP 1 Enables discharging USB_OTG*_VBUS during ADP
11 ADPCHRG0	VBUS Input ADP Charge Enable This signal controls charging the USB_OTG*_VBUS pin during ADP. The I_{ADP_SRC} current is >1.1mA and <1.65mA. 0 Disables charging USB_OTG*_VBUS during ADP 1 Disables charging USB_OTG*_VBUS during ADP
10 OTGDISABLE0	OTG Block Disable This signal powers down the VBUS Valid comparator. It does not control power to the Session Valid comparator, ADP Probe and Sense comparators, or the ID detection circuitry.

Table continues on the next page...

USBNC_*n*_PHY_CFG2 field descriptions (continued)

Field	Description
	0 The OTG block is powered up 1 The OTG block is powered down
9 TXBITSTUFFENH0	High-Byte Transmit Bit-Stuffing Enable This controller signal controls bit stuffing on DATAINH[7:0] when OPMODE[1:0] = 2'b11. For more information on the use of this setting see UTMI+ Specification Section 2.2.1.12 . 0 Bit stuffing is disabled 1 Bit stuffing is enabled
8 TXBITSTUFFEN0	Low-Byte Transmit Bit-Stuffing Enable This controller signal controls bit stuffing on DATAIN[7:0] when OPMODE[1:0] = 2'b11. For more information on the use of this setting see UTMI+ Specification Section 2.2.1.12 . 0 Bit stuffing is disabled 1 Bit stuffing is enabled
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 LOOPBACKENB0	Loopback Test Enable This signal places the USB OTG PHY in Loopback mode, which enables the receive and transmit logic concurrently. Loopback mode is for test purposes only; it cannot be used for normal operation. 0 During data transmission, the receive logic is disabled 1 During data transmission, the receive logic is enabled
5 SLEEPM0	Sleep Mode Assertion Asserting this signal places the USB OTG PHY in Sleep mode accourding to the USB 2.0 Link Power Management (LPM) addendum to the USB 2.0 specification. In Sleep mode, the transmitter is tri-stated and the USB OTG PHY circuits used in this SOC are powered down. **Note:**The USB controller used in this SOC does not support LPM operation. Leave this bit set to 1'b1 for normal operation. 0 Sleep mode 1 Normal operating mode
4 ACAENB0	ACA USB_OTG*_ID Pin Resistance Detection Enable This signal enables the circuitry to detect resistance on the USB_OTG*_ID pin if the USB cable is connected to an ACA (Acessory Charger Adaptor). When ACAENB0 is set to 1'b1, IDPULLUP0 (USB_OTGSC[26]) is overridden and set to 1'b0 internally to the USB OTG PHY. The RID* results are reported in USBNC_USB_OTGx_PHY_STATUS[28:24]. 0 Disables detection of resistance on the USB_OTG*_ID pin 1 Enables detection of resistance on the USB_OTG*_ID pin
3 DCDENB	Data Contact Detection Enable This signal enables current sourcing on the USB_OTG*_DP pin and pull-down resistaince on the USB_OTG*_DN pin for Data Contact Detect (DCD) per the USB Battery Charging Specification revision 1.2.

Table continues on the next page...

USBNC_*n*_PHY_CFG2 field descriptions (continued)

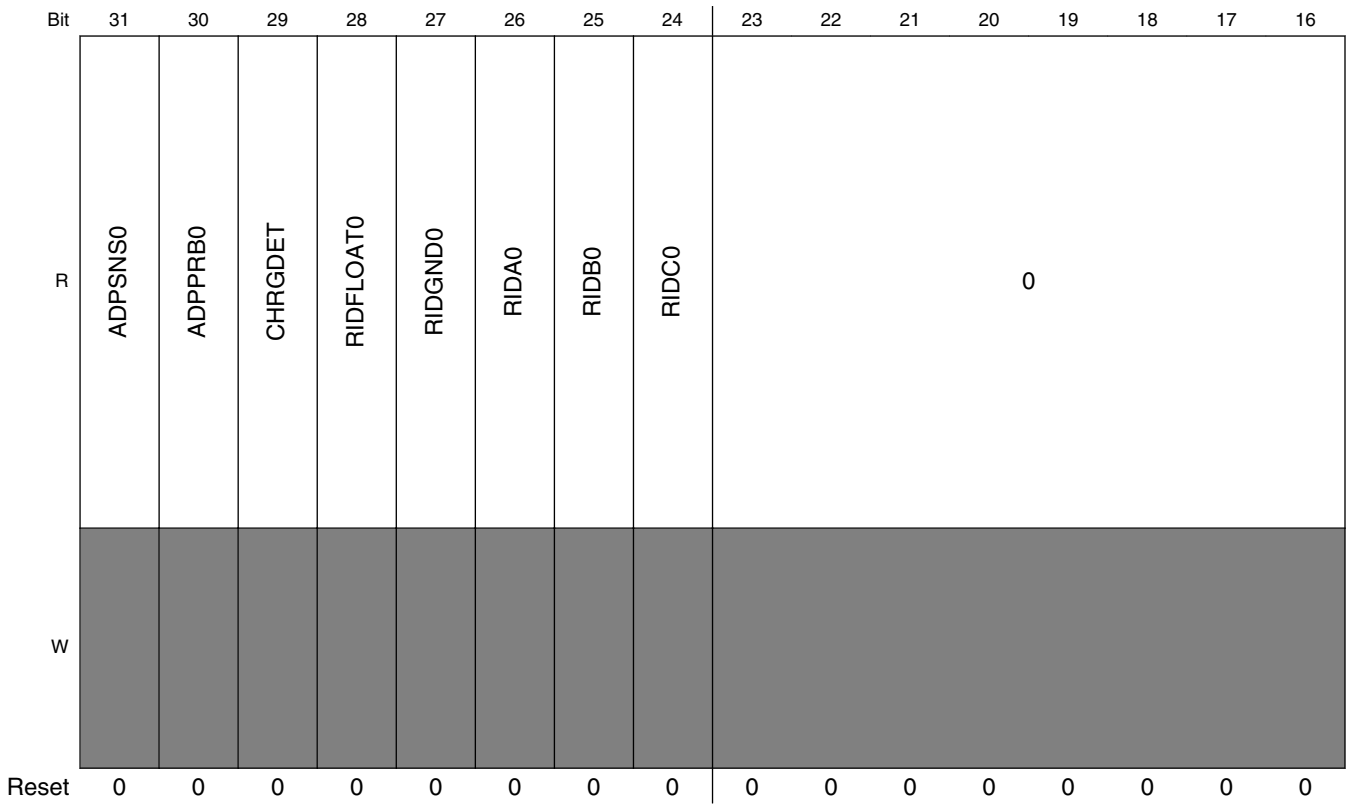
Field	Description
	Note: During normal USB operation of the USB OTG PHY, set this signal to 1'b0. 0 I_{DP_SRC} current and R_{DM_DWN} pull-down resistance are disabled 1 I_{DP_SRC} current and R_{DM_DWN} pull-down resistance are enabled
2 VDATSRCENB0	Battery Charging Source Select This signal is used to enable the VD*_SRC voltage source and ID*SINK current sink per the USB Battery Charging Specification revision 1.2. Pin selection for the voltage source and the current sink are controlled by USBNC_USB_OTGx_PHY_CFG2[0]. **Note:** During normal USB operation of the USB OTG PHY, set this signal to 1'b0. 0 VD*_SRC and ID*_SINK are disabled 1 VD*_SRC and ID*_SINK are enabled
1 VDATDETENB0	Battery Charging Attach / Connect Detection Enable This signal is used to enable attach/connect detection per the USB Battery Charging Specification revision 1.2. Results are reported by using the single-ended receiver status in USBNC_USB_OTGx_PHY_STATUS[1:0]. **Note:** During normal USB operation of the USB OTG PHY, set this signal to 1'b0. 0 IDP_SRC current source compliant to VLGC_HI connected to USB_OTG*_DP pin is disabled 1 IDP_SRC current source compliant to VLGC_HI connected to USB_OTG*_DP pin is enabled
0 CHRGSEL	Battery Charging Source Select This signal is used to select the Battery Charging detection connections per the USB Battery Charging Specification revision 1.2. It determines which of USB_OTG*_DP / USB_OTG*_DN has the VD*_SRC voltage source and which has the ID*_SINK current sink. Results are reported in USBNC_USB_OTGx_PHY_STATUS[29]. **Note:** During normal USB operation of the USB OTG PHY, set this signal to 1'b0. 0 VDP_SRC is connected to USB_OTG*_DP and IDM_SINK is connected to USB_OTG*_DN. Used for Primary Detection. 1 VDM_SRC is connected to USB_OTG*_DN and IDP_SINK is connected to USB_OTG*_DP. Used for Secondary Detection.

11.3.5.5 USB OTG PHY Status Register (USBNC_*n*_PHY_STATUS)

The USB_OTGx_PHY_STATUS register allows visibility of selected outputs of the USB OTG PHY.

These signals are used for some VBUS detection, ADP (Attach Detection Protocol), and Battery Charging ACA (Accesory Charging Adaptor) functions.

Address: 30B1_0200h base + 23Ch offset + (65536d × i), where i=0d to 1d



USBNC_*n*_PHY_STATUS field descriptions

Field	Description
31 ADPSNS0	ADP Sense Indicator This signal is output from the USB OTG PHY ADP Sense comparator and indicates whether the voltage on the USB_OTG*_VBUS pin is below the ADP sensing voltage. 0 The voltage on USB_OTG*_VBUS is below the ADP sensing voltage 1 The voltage on USB_OTG*_VBUS is above the ADP sensing voltage
30 ADPPRB0	ADP Probe Indicator This signal is output from the USB OTG PHY ADP Probe comparator and indicates whether the voltage on the USB_OTG*_VBUS pin is below the ADP probing voltage. 0 The voltage on USB_OTG*_VBUS is below the ADP probing voltage 1 The voltage on USB_OTG*_VBUS is above the ADP probing voltage
29 CHRGDET	Battery Charger Detection Output This signal indicates whether the voltage level on USB_OTG*_DP or USB_OTG*_DN is greater than VDAT_REF as defined in the USB Battery Charger, Revision 1.2 Specification. The selection of which USB data pins have the VD*_SRC and the ID*_SINK sources is controlled by register bit USBNC_USB_OTGx_CFG2[0]. The CHRGDET register bit is only valid when USBNC_USB_OTGx_CFG2[2:1] is set to 2'b11. 0 VD* < VDAT_REF 1 VD* > VDAT_REF
28 RIDFLOAT0	ACA USB_OTG*_ID Pin Resistance Indicator For USB Charger Detection, indicates whether the USB_OTG*_ID pin connected to an ACA is floating (has resistance > R_{ID_FLOAT}). 0 ACA OTG_ID pin resistance is ≤ R_{ID_A} (max) 1 ACA OTG_ID pin resistance is ≥ R_{ID_FLOAT} (min)
27 RIDGND0	ACA USB_OTG*_ID Pin Resistance Indicator For USB Charger Detection, indicates whether the USB_OTG*_ID pin connected to an ACA is grounded (has resistance < R_{ID_GND}). 0 ACA OTG_ID pin resistance is ≥ R_{ID_C} (min) 1 ACA OTG_ID pin resistance is ≤ R_{ID_GND} (max)
26 RIDA0	ACA USB_OTG*_ID Pin Resistance Indicator For USB Charger Detection, indicates whether the USB_OTG*_ID pin connected to an ACA is within the R_{ID_A}) range. 0 ACA OTG_ID pin resistance is ≥ R_{ID_FLOAT} (min) and ≤ R_{ID_B} max 1 ACA OTG_ID pin resistance is ≥ R_{ID_A} (min) and ≤ R_{ID_A} max
25 RIDB0	ACA USB_OTG*_ID Pin Resistance Indicator For USB Charger Detection, indicates whether the USB_OTG*_ID pin connected to an ACA is within the R_{ID_B}) range. 0 ACA OTG_ID pin resistance is ≥ R_{ID_A} (min) and ≤ R_{ID_C} max 1 ACA OTG_ID pin resistance is ≥ R_{ID_B} (min) and ≤ R_{ID_B} max

Table continues on the next page...

USBNC_*n*_PHY_STATUS field descriptions (continued)

Field	Description
24 RIDC0	ACA USB_OTG*_ID Pin Resistance Indicator For USB Charger Detection, indicates whether the USB_OTG*_ID pin connected to an ACA is within the R_{ID_C}) range. 0 ACA OTG_ID pin resistance is ≥ R_{ID_B} (min) and ≤ R_{ID_GND} max 1 ACA OTG_ID pin resistance is ≥ R_{ID_C} (min) and ≤ R_{ID_C} max
23–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 HOST_ DISCONNECT	Peripheral Disconnect Indicator This signal indicates to the USB_OTGx controller that a peripheral has been disconnected from or connected to the USB cable. In high-speed host operation, HOST_DISCONNECT is updated only at the end of SOF packet transmission. In FS and LS host operations, HOST_DISCONNECT is updated according to UTMI+ Specification, Section 2.2.1.10. This output is valid only when the USB OTG PHY and associated USB controller are configured to Host mode by setting USBC_n_USBMODE[1:0] to 2'b11. 0 Peripheral is connected 1 No peripheral is connected
4 ID_DIG	Micro- or Mini- A/B Plug Indicator This signal is the output from the ID pin A/B detector, which determines whether the resistance from the USB_OTG*_ID pin to VSS is grounded (10 Ω maximum for a Micro- or Mini-A plug) or floating (1 MegΩ minimum for a Micro- or Mini- B plug). The USB OTG PHY internal ID pin A/B detector circuit is enabled or disabled depending on the states of register bits USBC_n_OTGSC[5] and USBNC_USB_OTGx_PHY_CFG2[4]. Selection of whether to use the USB OTG PHY ID pin A/B detector or a GPIO pin ID detector is determined by the state of the USBNC_n_CTRL2[DIG_ID_SEL] bit field. 0 The connnected plug is a Micro- or Mini-A plug 1 The connnected plug is a Micro- or Mini-B plug
3 VBUS_VLD	VBUS Valid Indicator from USB OTG PHY This signal is the output from the USB OTG PHY VBUS Valid comparator and indicates whether the voltage on the USB_OTG*_VBUS pin is below the VBUS Valid threshold. The threshold is 4.75V but can be adjusted with OTGTUNE0 (USBNC_USB_PHY_CFG1[12:10]). The USB OTG PHY VBUS Valid comparator is enabled or disabled depending on the states of register bits USBNC_USB_OTGx_PHY_CFG2[10] and USBNC_USB_OTGx_PHY_CFG2[16]. 0 The voltage on USB_OTG*_VBUS is below the VBUS Valid threshold 1 The voltage on USB_OTG*_VBUS is above the VBUS Valid threshold
2 SESS_VLD	OTG Device Session Valid Indicator from USB OTG PHY This signal is the output from the USB OTG PHY Session Valid comparator and indicates whether the voltage on the USB_OTG*_VBUS pin is below the OTG Device Session Valid threshold. The Session Valid threshold is > 0.8V and < 2.0V on this USB PHY. 0 The voltage on USB_OTG*_VBUS is below the OTG Device Session Valid threshold 1 The voltage on USB_OTG*_VBUS is above the OTG Device Session Valid threshold

Table continues on the next page...

USBNC_*n*_PHY_STATUS field descriptions (continued)

Field	Description
LINE_STATE	Line State Indicator outputs from USB OTG PHY These signals reflect the state of the single-ended receivers as defined by the UTMI+ Specification. In Suspend or Sleep mode, this bus is a combinatatorial output (directly reflecting the current state of DN and DP, respectively. During normal high-speed packet transfers, the line indicates a high-speed J state. 00 SE0 (DP low, DN low) 01 J state for high-speed and full-speed USB traffic; K state for low-speed USB traffic (DP high, DN low) 10 K state for high-speed and full-speed USB traffic; J state for low-speed USB traffic (DP low, DN high) 11 SE1 (DP high, DN high)

11.3.5.6 USBNC_ADP_CFG1

Address: 30B1_0200h base + 250h offset = 30B1_0450h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								ADP_PRB_EN	ADP_PRB_INT_EN	ADP_SNS_INT_EN	TIMER_EN	Reserved		ADP_WAIT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

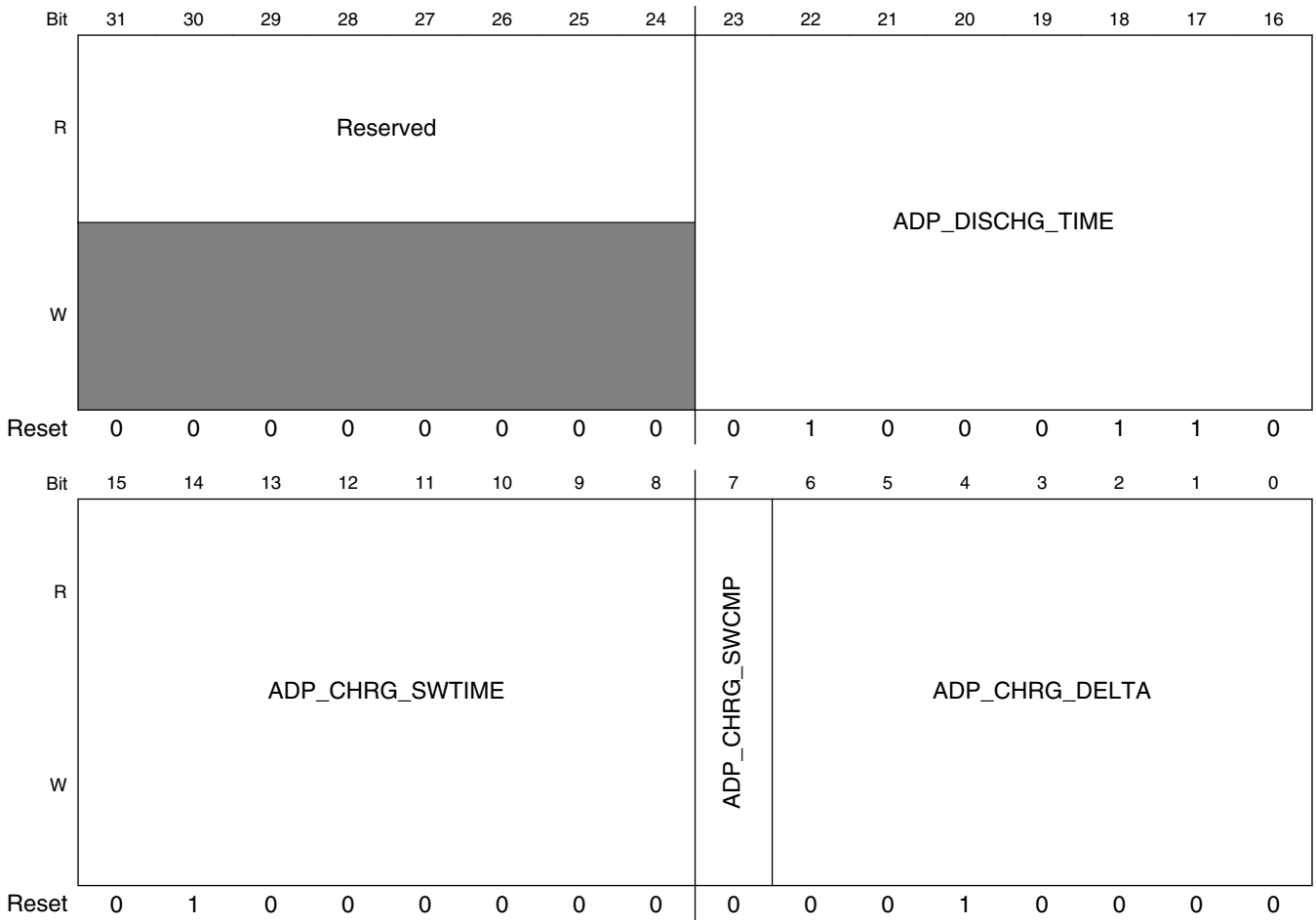
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADP_WAIT															
W																
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBNC_ADP_CFG1 field descriptions

Field	Description
31–24 Reserved	This field is reserved. Normal read/write able register 8'b0 Default
23 ADP_PRB_EN	Set to enable the ADP probe sequence 0 Default
22 ADP_PRB_INT_EN	ADP probe interrupt enable 0 Default
21 ADP_SNS_INT_EN	ADP Sense Interrupt Enable 0 Default
20 TIMER_EN	ADP Timer Test Enable Set will start the internal 18-bit counter. 0 Default
19–18 Reserved	This field is reserved.
ADP_WAIT	Delay between 2 ADP probe, default value is about 1.5 s in 32 KHz. 18'hc000 Default

11.3.5.7 USBNC_ADP_CFG2

Address: 30B1_0200h base + 254h offset = 30B1_0454h



USBNC_ADP_CFG2 field descriptions

Field	Description
31–24 Reserved	This field is reserved. Normal read/write able register 8'b0 Default
23–16 ADP_DISCHG_TIME	ADP Discharge time Default value is about 2.2 ms in 32 KHz. 8'h46 Default
15–8 ADP_CHRG_SWTIME	ADP charge time assigned by software 8'h40 Default
7 ADP_CHRG_SWCMP	If set, HW uses ADP_CHRG_SWTIME to compare. If clear, HW compares current charge time with n-2 (see otg2 spec).

Table continues on the next page...

USBNC_ADP_CFG2 field descriptions (continued)

Field	Description
	0 Default
ADP_CHRG_ DELTA	ADP charge time compare If diffence is larger than ADP_CHG_DELTA, it generates ADP_PROBE interrupt. 7'h10 Default

11.3.5.8 USBNC_ADP_STATUS

Address: 30B1_0200h base + 258h offset = 30B1_0458h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				ADP_PRB_INT	ADP_SNS_INT	ADP_CNT									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADP_CNT								ADP_PRB_TIMR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBNC_ADP_STATUS field descriptions

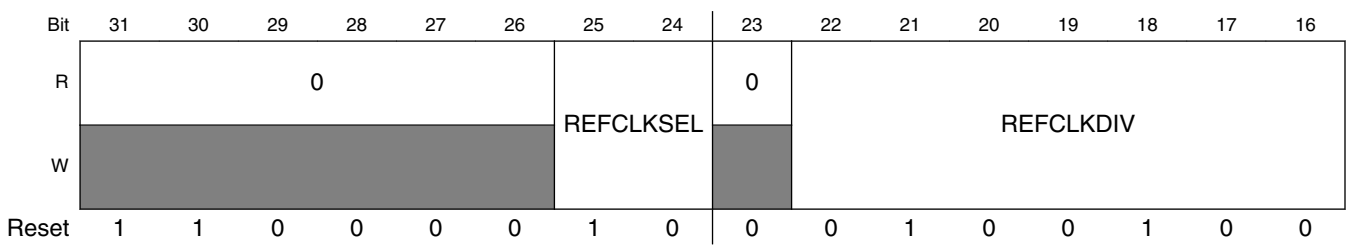
Field	Description
31–28 Reserved	This field is reserved.
27 ADP_PRB_INT	ADP Probe Interrupt Status Set when ADP charge time exceed [ADP_CHG_MIN, ADP_CHG_MAX] cleared by CLEAR ADP_PRB_INT_EN 0 Default
26 ADP_SNS_INT	ADP Sense Interrupt Status Set when PHY detect ADP sense cleared by clear ADP_SNS_INT_EN 0 Default
25–8 ADP_CNT	ADP Internal 18-bit Counter It counts when ADP_PRB_EN or TIMER_TEST_EN is set cleared if both ADP_PRB_EN and TIMER_TEST_EN are 0, or internal state machine change internal state machine: IDLE When ADP_PRB_EN is 0, change to DSCH when ADP_PRB_EN is set. DSCH ADP is discharging, change to CHRG when timer reach ADP_DSCHG_TIME. CHRG ADP is charging, change to WAIT when PHY indicate ADP_PROBE. WAIT Wait for next probe sequence. Change to DSCH when timer reach ADP_WAIT_TIME.
ADP_PRB_TIMR	ADP Probe Time It latchs the internal counter when PHY indicates ADP_PROBE. 0 Default

11.3.5.9 USB Host HSIC PHY Configuration Register (USBNC_UH_HSICPHY_CFG1)

This register provides functional and parametric settings for the USB UH HSIC PHY.

Customers should not need to alter the settings on this register but it will be used for characterization of the USB_H_DATA and USB_H_STROBE pins.

Address: 30B1_0200h base + 2_0240h offset = 30B3_0440h



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXSRTUNE				TXRPDTUNE		TXRPUTUNE		0	TXBITSTUFFENH	TXBITSTUFFEN	SLEEPM	DMPULLDOWN	DPPULLDOWN	LOOPBACKENB	COMMONONN
W																
Reset	0	0	1	1	1	0	1	0	0	0	0	1	1	1	0	1

USBNC_UH_HSICPHY_CFG1 field descriptions

Field	Description
31–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–24 REFCLKSEL	Reference clock select. Leave this field set to 2'b10. Operation with other settings is undefined.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–16 REFCLKDIV	Reference clock frequency select. This bus selects the HSIC PHY reference clock frequency. The value must remain static during normal operation. 0100100 12 MHz (Default) 0011100 15 MHz 0011010 16 MHz 0010101 19.2 MHz 0010100 20 MHz
15–12 TXSRTUNE	Driver slew rate adjustment for USB_H_DATA and USB_H_STROBE pins. 0000 -20% 0001 -10% 0011 Design default 0111 +10% 1111 +20%
11–10 TXRPDTUNE	Driver pull-down single-ended source impedance adjustment for USB_H_DATA and USB_H_STROBE pins when driving low. 00 +11% 01 +5% 10 Design default 11 -5%
9–8 TXRPUTUNE	Driver pull-up single-ended source impedance adjustment for USB_H_DATA and USB_H_STROBE pins when driving high. 00 +11% 01 +5% 10 Design default 11 -5%
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

USBNC_UH_HSICPHY_CFG1 field descriptions (continued)

Field	Description
6 TXBITSTUFFENH	High-Byte Transmit Bit-Stuffing Enable This controller signal controls bit stuffing on DATAINH[7:0] when OPMODE[1:0] = 2'b11. For more information on the use of this setting see UTMI+ Specification Section 2.2.1.12 . 0 Bit stuffing is disabled 1 Bit stuffing is enabled
5 TXBITSTUFFEN	Low-Byte Transmit Bit-Stuffing Enable This controller signal controls bit stuffing on DATAIN[7:0] when OPMODE[1:0] = 2'b11. For more information on the use of this setting see UTMI+ Specification Section 2.2.1.12 . 0 Bit stuffing is disabled 1 Bit stuffing is enabled
4 SLEEPM	Sleeep mode assertion. Reseting this signal laces the USB UH HSIC PHY in Sleep mode. Since the controller used for the HSB UH HSIC port does not support USB 2.0 Link Power Management (LPM) this bit field should remain at 1'b1. 0 Sleep mode 1 Normal operating mode
3 DMPULLDOWN	Bus keeper resistors enable. If both DPPULLDOWN and DMPULLDOWN are set to 1'b1, the HSIC PHY is configured to operate as a Host. Otherwise, the HSIC PHY is configured to operate as a Device. Since the controller for the USB UH HSIC port only operates as a USB Host, these signals should not be reset to 1'b0 . 0 Disable bus keeper resistors 1 Enable bus keeper resistors
2 DPPULLDOWN	Bus keeper resistors enable. If both DPPULLDOWN and DMPULLDOWN are set to 1'b1, the HSIC PHY is configured to operate as a Host. Otherwise, the HSIC PHY is configured to operate as a Device. Since the controller for the USB UH HSIC port only operates as a USB Host, these signals should not be reset to 1'b0 . 0 Disable bus keeper resistors 1 Enable bus keeper resistors
1 LOOPBACKENB	Loopback test enable. This controller signal places the HSIC PHY in Loopback mode, which enables the receive and transmit logic concurrently. **Note:** Loopback mode is for test purposes only; it cannot be used for normal operation. 0 During data transmission, the receive logic is disabled 1 During data transmission, the receive logic is enabled
0 COMMONONN	This clock configuration setting should not be changed and must remain at 1'b1.

11.3.6 USB Core Memory Map/Register Definition

USB memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30B1_0000	Identification register (USB_OTG1_ID)	32	R	E401_FA05h	11.3.6.1/ 3297
30B1_0004	Hardware General (USB_OTG1_HWGENERAL)	32	R	See section	11.3.6.2/ 3298
30B1_0008	Host Hardware Parameters (USB_OTG1_HWHOST)	32	R	1002_0001h	11.3.6.3/ 3299
30B1_000C	Device Hardware Parameters (USB_OTG1_HWDEVICE)	32	R	0000_0011h	11.3.6.4/ 3300
30B1_0010	TX Buffer Hardware Parameters (USB_OTG1_HWTXBUF)	32	R	8008_0B08h	11.3.6.5/ 3300
30B1_0014	RX Buffer Hardware Parameters (USB_OTG1_HWRXBUF)	32	R	0000_0808h	11.3.6.6/ 3301
30B1_0080	General Purpose Timer #0 Load (USB_OTG1_GPTIMER0LD)	32	R/W	0000_0000h	11.3.6.7/ 3302
30B1_0084	General Purpose Timer #0 Controller (USB_OTG1_GPTIMER0CTRL)	32	R/W	0000_0000h	11.3.6.8/ 3302
30B1_0088	General Purpose Timer #1 Load (USB_OTG1_GPTIMER1LD)	32	R/W	0000_0000h	11.3.6.9/ 3303
30B1_008C	General Purpose Timer #1 Controller (USB_OTG1_GPTIMER1CTRL)	32	R/W	0000_0000h	11.3.6.10/ 3304
30B1_0090	System Bus Config (USB_OTG1_SBUSCFG)	32	R/W	0000_0002h	11.3.6.11/ 3305
30B1_0100	Capability Registers Length (USB_OTG1_CAPLENGTH)	8	R	40h	11.3.6.12/ 3306
30B1_0102	Host Controller Interface Version (USB_OTG1_HCIVERSION)	16	R	0100h	11.3.6.13/ 3306
30B1_0104	Host Controller Structural Parameters (USB_OTG1_HCSPARAMS)	32	R	0001_0011h	11.3.6.14/ 3307
30B1_0108	Host Controller Capability Parameters (USB_OTG1_HCCPARAMS)	32	R	0000_0006h	11.3.6.15/ 3309
30B1_0120	Device Controller Interface Version (USB_OTG1_DCIVERSION)	16	R	0001h	11.3.6.16/ 3311
30B1_0124	Device Controller Capability Parameters (USB_OTG1_DCCPARAMS)	32	R	0000_0188h	11.3.6.17/ 3311
30B1_0140	USB Command Register (USB_OTG1_USBCMD)	32	R/W	0008_0000h	11.3.6.18/ 3313
30B1_0144	USB Status Register (USB_OTG1_USBSTS)	32	R/W	0000_0000h	11.3.6.19/ 3317
30B1_0148	Interrupt Enable Register (USB_OTG1_USBINTR)	32	R/W	0000_0000h	11.3.6.20/ 3321

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30B1_014C	USB Frame Index (USB_OTG1_FRINDEX)	32	R/W	0000_0000h	11.3.6.21/ 3323
30B1_0154	Frame List Base Address (USB_OTG1_PERIODICLISTBASE)	32	R/W	0000_0000h	11.3.6.22/ 3324
30B1_0154	Device Address (USB_OTG1_DEVICEADDR)	32	R/W	0000_0000h	11.3.6.23/ 3324
30B1_0158	Next Asynch. Address (USB_OTG1_ASYNCLISTADDR)	32	R/W	0000_0000h	11.3.6.24/ 3325
30B1_0158	Endpoint List Address (USB_OTG1_ENDPTLISTADDR)	32	R/W	0000_0000h	11.3.6.25/ 3326
30B1_0160	Programmable Burst Size (USB_OTG1_BURSTSIZE)	32	R/W	0000_0000h	11.3.6.26/ 3326
30B1_0164	TX FIFO Fill Tuning (USB_OTG1_TXFILLTUNING)	32	R/W	0000_0808h	11.3.6.27/ 3327
30B1_0178	Endpoint NAK (USB_OTG1_ENDPTNAK)	32	R/W	0000_0000h	11.3.6.28/ 3329
30B1_017C	Endpoint NAK Enable (USB_OTG1_ENDPTNAKEN)	32	R/W	0000_0000h	11.3.6.29/ 3329
30B1_0180	Configure Flag Register (USB_OTG1_CONFIGFLAG)	32	R/W	0000_0001h	11.3.6.30/ 3330
30B1_0184	Port Status & Control (USB_OTG1_PORTSC1)	32	R/W	1000_0000h	11.3.6.31/ 3330
30B1_01A4	On-The-Go Status & control (USB_OTG1_OTGSC)	32	R/W	0000_0120h	11.3.6.32/ 3337
30B1_01A8	USB Device Mode (USB_OTG1_USBMODE)	32	R/W	0000_0000h	11.3.6.33/ 3341
30B1_01AC	Endpoint Setup Status (USB_OTG1_ENDPTSETUPSTAT)	32	R/W	0000_0000h	11.3.6.34/ 3342
30B1_01B0	Endpoint Prime (USB_OTG1_ENDPTPRIME)	32	R/W	0000_0000h	11.3.6.35/ 3343
30B1_01B4	Endpoint Flush (USB_OTG1_ENDPTFLUSH)	32	R/W	0000_0000h	11.3.6.36/ 3344
30B1_01B8	Endpoint Status (USB_OTG1_ENDPTSTAT)	32	R	0000_0000h	11.3.6.37/ 3344
30B1_01BC	Endpoint Complete (USB_OTG1_ENDPTCOMPLETE)	32	R/W	0000_0000h	11.3.6.38/ 3345
30B1_01C0	Endpoint Control0 (USB_OTG1_ENDPTCTRL0)	32	R/W	0080_0080h	11.3.6.39/ 3346
30B1_01C4	Endpoint Control 1 (USB_OTG1_ENDPTCTRL1)	32	R/W	0000_0000h	11.3.6.40/ 3348
30B1_01C8	Endpoint Control 2 (USB_OTG1_ENDPTCTRL2)	32	R/W	0000_0000h	11.3.6.41/ 3351
30B1_01CC	Endpoint Control 3 (USB_OTG1_ENDPTCTRL3)	32	R/W	0000_0000h	11.3.6.42/ 3353

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30B1_01D0	Endpoint Control 4 (USB_OTG1_ENDPTCTRL4)	32	R/W	0000_0000h	11.3.6.43/ 3356
30B1_01D4	Endpoint Control 5 (USB_OTG1_ENDPTCTRL5)	32	R/W	0000_0000h	11.3.6.44/ 3359
30B1_01D8	Endpoint Control 6 (USB_OTG1_ENDPTCTRL6)	32	R/W	0000_0000h	11.3.6.45/ 3362
30B1_01DC	Endpoint Control 7 (USB_OTG1_ENDPTCTRL7)	32	R/W	0000_0000h	11.3.6.46/ 3365
30B3_0000	Identification register (USB_HSIC_ID)	32	R	E401_FA05h	11.3.6.1/ 3297
30B3_0004	Hardware General (USB_HSIC_HWGENERAL)	32	R	See section	11.3.6.2/ 3298
30B3_0008	Host Hardware Parameters (USB_HSIC_HWHOST)	32	R	1002_0001h	11.3.6.3/ 3299
30B3_000C	Device Hardware Parameters (USB_HSIC_HWDEVICE)	32	R	0000_0011h	11.3.6.4/ 3300
30B3_0010	TX Buffer Hardware Parameters (USB_HSIC_HWTXBUF)	32	R	8008_0B08h	11.3.6.5/ 3300
30B3_0014	RX Buffer Hardware Parameters (USB_HSIC_HWRXBUF)	32	R	0000_0808h	11.3.6.6/ 3301
30B3_0080	General Purpose Timer #0 Load (USB_HSIC_GPTIMER0LD)	32	R/W	0000_0000h	11.3.6.7/ 3302
30B3_0084	General Purpose Timer #0 Controller (USB_HSIC_GPTIMER0CTRL)	32	R/W	0000_0000h	11.3.6.8/ 3302
30B3_0088	General Purpose Timer #1 Load (USB_HSIC_GPTIMER1LD)	32	R/W	0000_0000h	11.3.6.9/ 3303
30B3_008C	General Purpose Timer #1 Controller (USB_HSIC_GPTIMER1CTRL)	32	R/W	0000_0000h	11.3.6.10/ 3304
30B3_0090	System Bus Config (USB_HSIC_SBUSCFG)	32	R/W	0000_0002h	11.3.6.11/ 3305
30B3_0100	Capability Registers Length (USB_HSIC_CAPLENGTH)	8	R	40h	11.3.6.12/ 3306
30B3_0102	Host Controller Interface Version (USB_HSIC_HCIVERSION)	16	R	0100h	11.3.6.13/ 3306
30B3_0104	Host Controller Structural Parameters (USB_HSIC_HCSPARAMS)	32	R	0001_0011h	11.3.6.14/ 3307
30B3_0108	Host Controller Capability Parameters (USB_HSIC_HCCPARAMS)	32	R	0000_0006h	11.3.6.15/ 3309
30B3_0120	Device Controller Interface Version (USB_HSIC_DCIVERSION)	16	R	0001h	11.3.6.16/ 3311
30B3_0124	Device Controller Capability Parameters (USB_HSIC_DCCPARAMS)	32	R	0000_0188h	11.3.6.17/ 3311
30B3_0140	USB Command Register (USB_HSIC_USBCMD)	32	R/W	0008_0000h	11.3.6.18/ 3313

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30B3_0144	USB Status Register (USB_HSIC_USBSTS)	32	R/W	0000_0000h	11.3.6.19/ 3317
30B3_0148	Interrupt Enable Register (USB_HSIC_USBINTR)	32	R/W	0000_0000h	11.3.6.20/ 3321
30B3_014C	USB Frame Index (USB_HSIC_FRINDEX)	32	R/W	0000_0000h	11.3.6.21/ 3323
30B3_0154	Frame List Base Address (USB_HSIC_PERIODICLISTBASE)	32	R/W	0000_0000h	11.3.6.22/ 3324
30B3_0154	Device Address (USB_HSIC_DEVICEADDR)	32	R/W	0000_0000h	11.3.6.23/ 3324
30B3_0158	Next Asynch. Address (USB_HSIC_ASYNCLISTADDR)	32	R/W	0000_0000h	11.3.6.24/ 3325
30B3_0158	Endpoint List Address (USB_HSIC_ENDPTLISTADDR)	32	R/W	0000_0000h	11.3.6.25/ 3326
30B3_0160	Programmable Burst Size (USB_HSIC_BURSTSIZE)	32	R/W	0000_0000h	11.3.6.26/ 3326
30B3_0164	TX FIFO Fill Tuning (USB_HSIC_TXFILLTUNING)	32	R/W	0000_0808h	11.3.6.27/ 3327
30B3_0178	Endpoint NAK (USB_HSIC_ENDPTNAK)	32	R/W	0000_0000h	11.3.6.28/ 3329
30B3_017C	Endpoint NAK Enable (USB_HSIC_ENDPTNAKEN)	32	R/W	0000_0000h	11.3.6.29/ 3329
30B3_0180	Configure Flag Register (USB_HSIC_CONFIGFLAG)	32	R/W	0000_0001h	11.3.6.30/ 3330
30B3_0184	Port Status & Control (USB_HSIC_PORTSC1)	32	R/W	1000_0000h	11.3.6.31/ 3330
30B3_01A4	On-The-Go Status & control (USB_HSIC_OTGSC)	32	R/W	0000_0120h	11.3.6.32/ 3337
30B3_01A8	USB Device Mode (USB_HSIC_USBMODE)	32	R/W	0000_0000h	11.3.6.33/ 3341
30B3_01AC	Endpoint Setup Status (USB_HSIC_ENDPTSETUPSTAT)	32	R/W	0000_0000h	11.3.6.34/ 3342
30B3_01B0	Endpoint Prime (USB_HSIC_ENDPTPRIME)	32	R/W	0000_0000h	11.3.6.35/ 3343
30B3_01B4	Endpoint Flush (USB_HSIC_ENDPTFLUSH)	32	R/W	0000_0000h	11.3.6.36/ 3344
30B3_01B8	Endpoint Status (USB_HSIC_ENDPTSTAT)	32	R	0000_0000h	11.3.6.37/ 3344
30B3_01BC	Endpoint Complete (USB_HSIC_ENDPTCOMPLETE)	32	R/W	0000_0000h	11.3.6.38/ 3345
30B3_01C0	Endpoint Control0 (USB_HSIC_ENDPTCTRL0)	32	R/W	0080_0080h	11.3.6.39/ 3346
30B3_01C4	Endpoint Control 1 (USB_HSIC_ENDPTCTRL1)	32	R/W	0000_0000h	11.3.6.40/ 3348

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30B3_01C8	Endpoint Control 2 (USB_HSIC_ENDPTCTRL2)	32	R/W	0000_0000h	11.3.6.41/ 3351
30B3_01CC	Endpoint Control 3 (USB_HSIC_ENDPTCTRL3)	32	R/W	0000_0000h	11.3.6.42/ 3353
30B3_01D0	Endpoint Control 4 (USB_HSIC_ENDPTCTRL4)	32	R/W	0000_0000h	11.3.6.43/ 3356
30B3_01D4	Endpoint Control 5 (USB_HSIC_ENDPTCTRL5)	32	R/W	0000_0000h	11.3.6.44/ 3359
30B3_01D8	Endpoint Control 6 (USB_HSIC_ENDPTCTRL6)	32	R/W	0000_0000h	11.3.6.45/ 3362
30B3_01DC	Endpoint Control 7 (USB_HSIC_ENDPTCTRL7)	32	R/W	0000_0000h	11.3.6.46/ 3365

11.3.6.1 Identification register (USBx_ID)

The ID register identifies the USB 2.0 High-Speed core and its revision.

Address: Base address + 0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								REVISION							
W																
Reset	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		NID						Reserved		ID					
W																
Reset	1	1	1	1	1	0	1	0	0	0	0	0	0	1	0	1

USBx_ID field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 REVISION	Revision number of the controller core.
15–14 -	This field is reserved. Reserved
13–8 NID	Complement version of ID

Table continues on the next page...

USBx_ID field descriptions (continued)

Field	Description
7–6 -	This field is reserved. Reserved
ID	Configuration number. This number is set to 0x05 and indicates that the peripheral is USB 2.0 High-Speed core.

11.3.6.2 Hardware General (USBx_HWGENERAL)

General hardware parameters as defined in System Level Issues and Core Configuration.

NOTE

The reset value could vary from instance to instance. Please see the detail in bit field description and ignore reset value in summary table in this case!

Address: Base address + 4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				SM		PHYM				PHYW		Reserved			
W																
Reset	0	0	0	0	0	0	x*	x*	x*	x*	1	1	0	1	0	1

* Notes:
- x = Undefined at reset.

USBx_HWGENERAL field descriptions

Field	Description
31–12 -	This field is reserved.
11–10 SM	Serial interface mode capability SM bit reset value is '00b' 00 No Serial Engine, always use parallel signalling.
9–6 PHYM	Transceiver type PHYM bit reset value: '0000b' for OTG1/OTG2 controller core, and '1011b' for Host-only HSIC controller core.

Table continues on the next page...

USBx_HWGENERAL field descriptions (continued)

Field	Description
	0000 UTMI/UMTI+ 1011 Software programmable - reset to HSIC
5–4 PHYW	Data width of the transciever connected to the controller core. PHYW bit reset value is PHYW bit reset value is '11b'. 11 Reset to 16 bit wide data bus Software programmable
-	This field is reserved. Reserved

11.3.6.3 Host Hardware Parameters (USBx_HWHOST)

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												NPORT			HC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

USBx_HWHOST field descriptions

Field	Description
31–4 -	This field is reserved. Reserved
3–1 NPORT	The Nmber of downstream ports supported by the host controller is (NPORT+1). **NOTE:** When these bits value is '000', it indicates a single-port host controller.
0 HC	Host Capable. Indicating whether host operation mode is supported or not. 1 Supported 0 Not supported

11.3.6.4 Device Hardware Parameters (USBx_HWDEVICE)

NOTE

This register is only available in OTG core.

Address: Base address + Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved										DEVEP					DC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

USBx_HWDEVICE field descriptions

Field	Description
31–6 -	This field is reserved. Reserved
5–1 DEVEP	Device Endpoint Number
0 DC	Device Capable. Indicating whether device operation mode is supported or not. 1 Supported 0 Not supported

11.3.6.5 TX Buffer Hardware Parameters (USBx_HWTXBUF)

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								TXCHANADD								Reserved								TXBURST							
W																																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	0	0	0

USBx_HWTXBUF field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 TXCHANADD	TX FIFO Buffer size is: (2^TXCHANADD) * 4 Bytes. These bits are set to '08h', so buffer size is 256*4 Bytes. For the OTG controller operating in device mode, this is the FIFO buffer size per endpoint. As the OTG controller has 8 TX endpoint, there are 8 of these buffers. For the OTG controller operating in host mode, or for Host-only controller, the entire buffer memory is used as a single TX buffer. Therefore, there is only 1 of this buffer
15–8 -	This field is reserved. Reserved
TXBURST	Default burst size for memory to TX buffer transfer. This is reset value of TXPBURST bits in USB core regsiter USB_n_BURSTSIZE. Please see Programmable Burst Size (USB_BURSTSIZE) .

11.3.6.6 RX Buffer Hardware Parameters (USBx_HWRXBUF)

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																RXADD								RXBURST							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

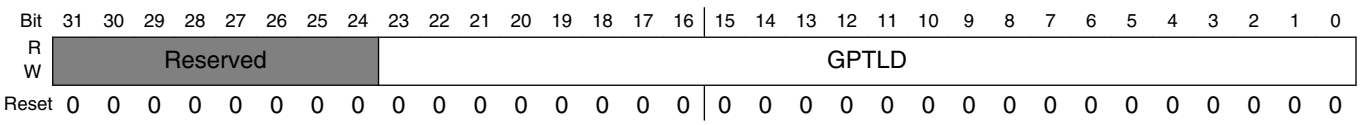
USBx_HWRXBUF field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 RXADD	Buffer total size for all receive endpoints is (2^RXADD). RX Buffer size is: (2^RXADD) * 4 Bytes. These bits are set to '08h', so buffer size is 256*4 Bytes. There is a single Receive FIFO buffer in the USB controller. The buffer is shared for all endpoints for the OTG controller in device mode.
RXBURST	Default burst size for memory to RX buffer transfer. This is reset value of RXPBURST bits in USB core regsiter USB_n_BURSTSIZE. Please see Programmable Burst Size (USB_BURSTSIZE) .

11.3.6.7 General Purpose Timer #0 Load (USBx_GPTIMER0LD)

This register controls load value of the count timer in register n_GPTIMER0CTRL. Please see General Purpose Timer #0 Controller (USB_GPTIMER0CTRL) .

Address: Base address + 80h offset



USBx_GPTIMER0LD field descriptions

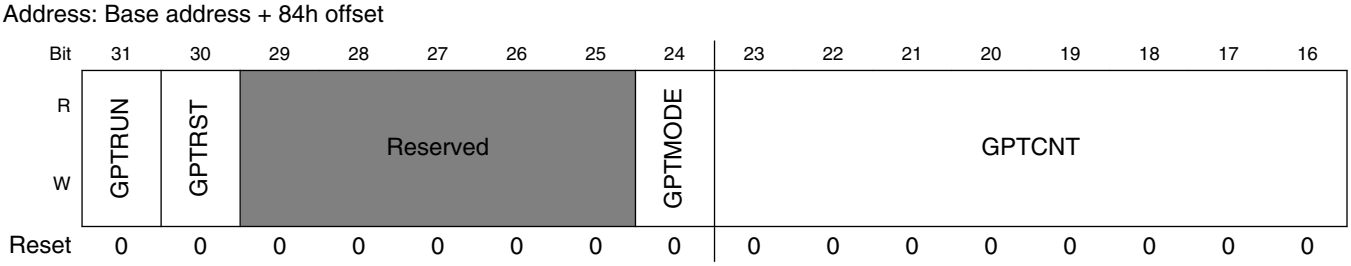
Field	Description
31–24 -	This field is reserved. Reserved
GPTLD	General Purpose Timer Load Value These bit fields are loaded to GPTCNT bits when GPTRST bit is set '1b'. This value represents the time in microseconds minus 1 for the timer duration. Example: for a one millisecond timer, load 1000-1=999 or 0x0003E7. **NOTE:** Max value is 0xFFFFFF or 16.777215 seconds.

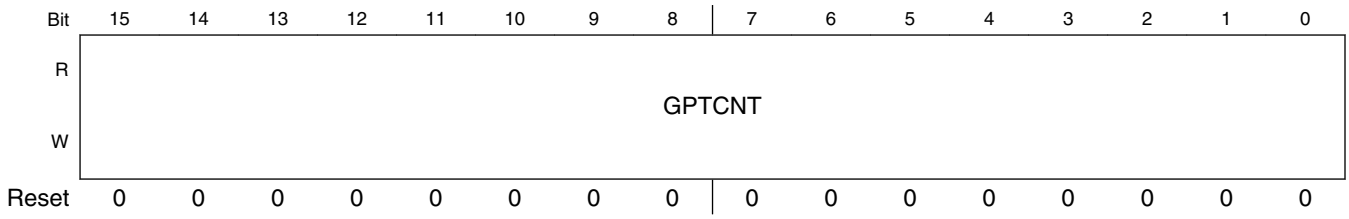
11.3.6.8 General Purpose Timer #0 Controller (USBx_GPTIMER0CTRL)

This register contains the control for this countdown timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two counter modes which are described in the register table below. When the timer counter value transitions to zero, an interrupt could be generated if enable.

Interrupt status bit is TI0 bit in n_USBSTS register (See USB Status Register (USB_USBSTS)), interrupt enable bit is TIE0 bit in n_USBINTR register. (See Interrupt Enable Register (USB_USBINTR) .)

Address: Base address + 84h offset





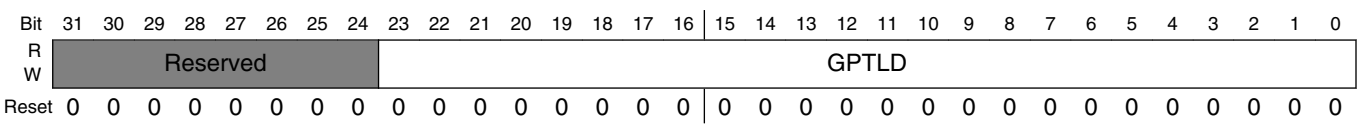
USBx_GPTIMER0CTRL field descriptions

Field	Description
31 GPTRUN	General Purpose Timer Run GPTCNT bits are not effected when setting or clearing this bit. 0 Stop counting 1 Run
30 GPTRST	General Purpose Timer Reset 0 No action 1 Load counter value from GPTLD bits in n_GPTIMER0LD
29–25 -	This field is reserved. Reserved
24 GPTMODE	General Purpose Timer Mode In one shot mode, the timer will count down to zero, generate an interrupt, and stop until the counter is reset by software; In repeat mode, the timer will count down to zero, generate an interrupt and automatically reload the counter value from GPTLD bits to start again. 0 One Shot Mode 1 Repeat Mode
GPTCNT	General Purpose Timer Counter. This field is the count value of the countdown timer.

11.3.6.9 General Purpose Timer #1 Load (USBx_GPTIMER1LD)

This register controls load value of the count timer in register n_GPTIMER1CTRL. Please see General Purpose Timer #1 Controller (USB_GPTIMER1CTRL) .

Address: Base address + 88h offset



USBx_GPTIMER1LD field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
GPTLD	General Purpose Timer Load Value These bit fields are loaded to GPTCNT bits when GPTRST bit is set '1b'. This value represents the time in microseconds minus 1 for the timer duration. Example: for a one millisecond timer, load 1000-1=999 or 0x0003E7. **NOTE:** Max value is 0xFFFFFF or 16.777215 seconds.

11.3.6.10 General Purpose Timer #1 Controller (USBx_GPTIMER1CTRL)

This register contains the control for this countdown timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two counter modes which are described in the register table below. When the timer counter value transitions to zero, an interrupt could be generated if enable.

Interrupt status bit is TI1 bit in USB_n_USBSTS register (See USB Status Register (USB_USBSTS)), interrupt enable bit is TIE1 bit in n_USBINTR register (See Interrupt Enable Register (USB_USBINTR)).

Address: Base address + 8Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPTRUN	GPTRST	Reserved					GPTMODE	GPTCNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPTCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBx_GPTIMER1CTRL field descriptions

Field	Description
31 GPTRUN	General Purpose Timer Run GPTCNT bits are not effected when setting or clearing this bit.

Table continues on the next page...

USBx_GPTIMER1CTRL field descriptions (continued)

Field	Description
	0 Stop counting 1 Run
30 GPTRST	General Purpose Timer Reset 0 No action 1 Load counter value from GPTLD bits in USB_n_GPTIMER0LD
29–25 -	This field is reserved. Reserved
24 GPTMODE	General Purpose Timer Mode In one shot mode, the timer will count down to zero, generate an interrupt, and stop until the counter is reset by software. In repeat mode, the timer will count down to zero, generate an interrupt and automatically reload the counter value from GPTLD bits to start again. 0 One Shot Mode 1 Repeat Mode
GPTCNT	General Purpose Timer Counter. This field is the count value of the countdown timer.

11.3.6.11 System Bus Config (USBx_SBUSCFG)

Address: Base address + 90h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																													AHBBRST		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

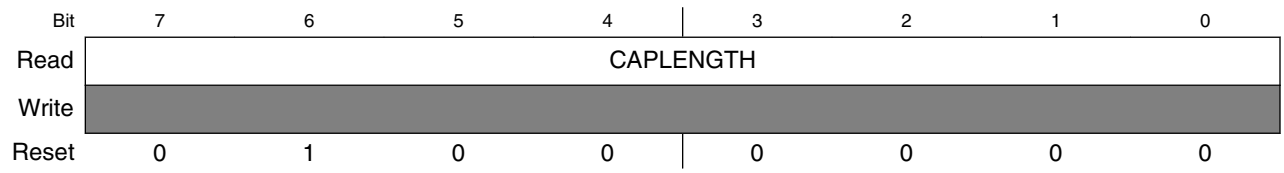
USBx_SBUSCFG field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
AHBBRST	AHB master interface Burst configuration These bits control AHB master transfer type sequence (or priority). **NOTE:** This register overrides n_BURSTSIZE register when its value is not zero. 000 Incremental burst of unspecified length only 001 INCR4 burst, then single transfer 010 INCR8 burst, INCR4 burst, then single transfer 011 INCR16 burst, INCR8 burst, INCR4 burst, then single transfer 100 Reserved, don't use 101 INCR4 burst, then incremental burst of unspecified length 110 INCR8 burst, INCR4 burst, then incremental burst of unspecified length 111 INCR16 burst, INCR8 burst, INCR4 burst, then incremental burst of unspecified length

11.3.6.12 Capability Registers Length (USBx_CAPLENGTH)

The Capability Registers Length register contains the address offset to the Operational registers relative to the CAPLENGTH register.

Address: Base address + 100h offset



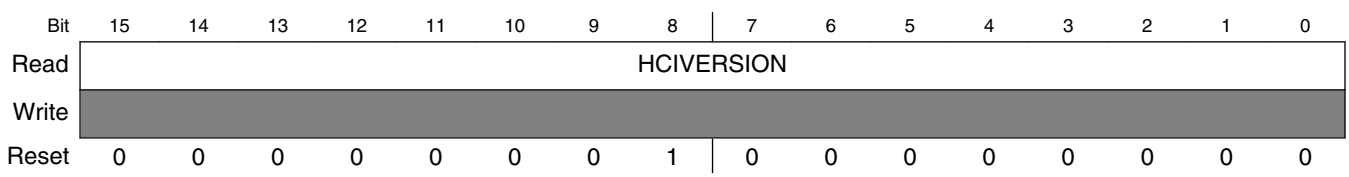
USBx_CAPLENGTH field descriptions

Field	Description
CAPLENGTH	These bits are used as an offset to add to register base to find the beginning of the Operational Register. Default value is '40h'.

11.3.6.13 Host Controller Interface Version (USBx_HCIVERSION)

This is a 2-byte register containing a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision.

Address: Base address + 102h offset



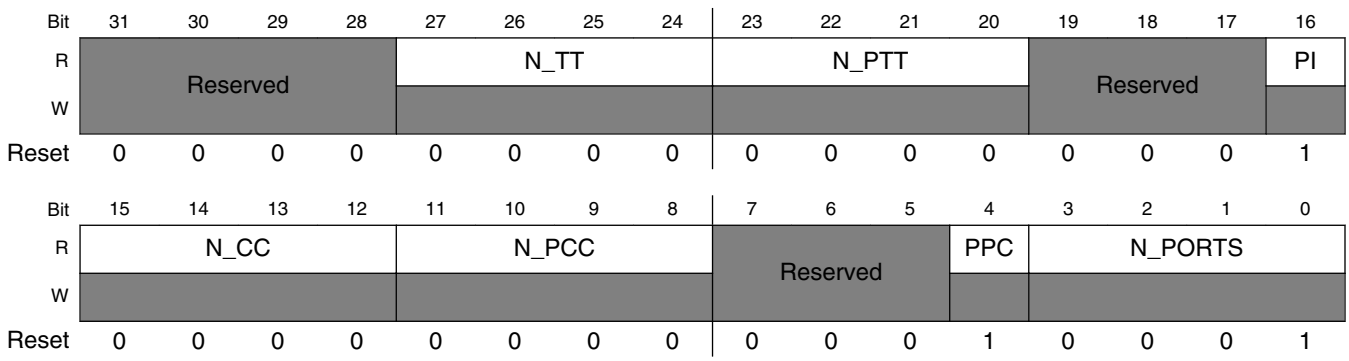
USBx_HCIVERSION field descriptions

Field	Description
HCIVERSION	Host Controller Interface Version Number Default value is '10h', which means EHCI rev1.0.

11.3.6.14 Host Controller Structural Parameters (USBx_HCSPARAMS)

The following figure shows the port steering logic capabilities of Host Control Structural Parameters (n_HCSPARAMS).

Address: Base address + 104h offset



USBx_HCSPARAMS field descriptions

Field	Description
31–28 -	This field is reserved. Reserved
27–24 N_TT	Number of Transaction Translators (N_TT). Default value '0000b' This field indicates the number of embedded transaction translators associated with the USB2.0 host controller. These bits would be set to '0001b' for Multi-Port Host, and '0000b' for Single-Port Host.
23–20 N_PTT	Number of Ports per Transaction Translator (N_PTT). Default value '0000b' This field indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. These bits would be set to equal N_PORTS for Multi-Port Host, and '0000b' for Single-Port Host.
19–17 -	This field is reserved. Reserved
16 PI	Port Indicators (P INDICATOR) This bit indicates whether the ports support port indicator control. When set to one, the port status and control registers include a read/writeable field for controlling the state of the port indicator This bit is "1b" in all controller core.
15–12 N_CC	Number of Companion Controller (N_CC). This field indicates the number of companion controllers associated with this USB2.0 host controller. These bits are '0000b' in all controller core. 0 There is no internal Companion Controller and port-ownership hand-off is not supported. 1 There are internal companion controller(s) and port-ownership hand-offs is supported.

Table continues on the next page...

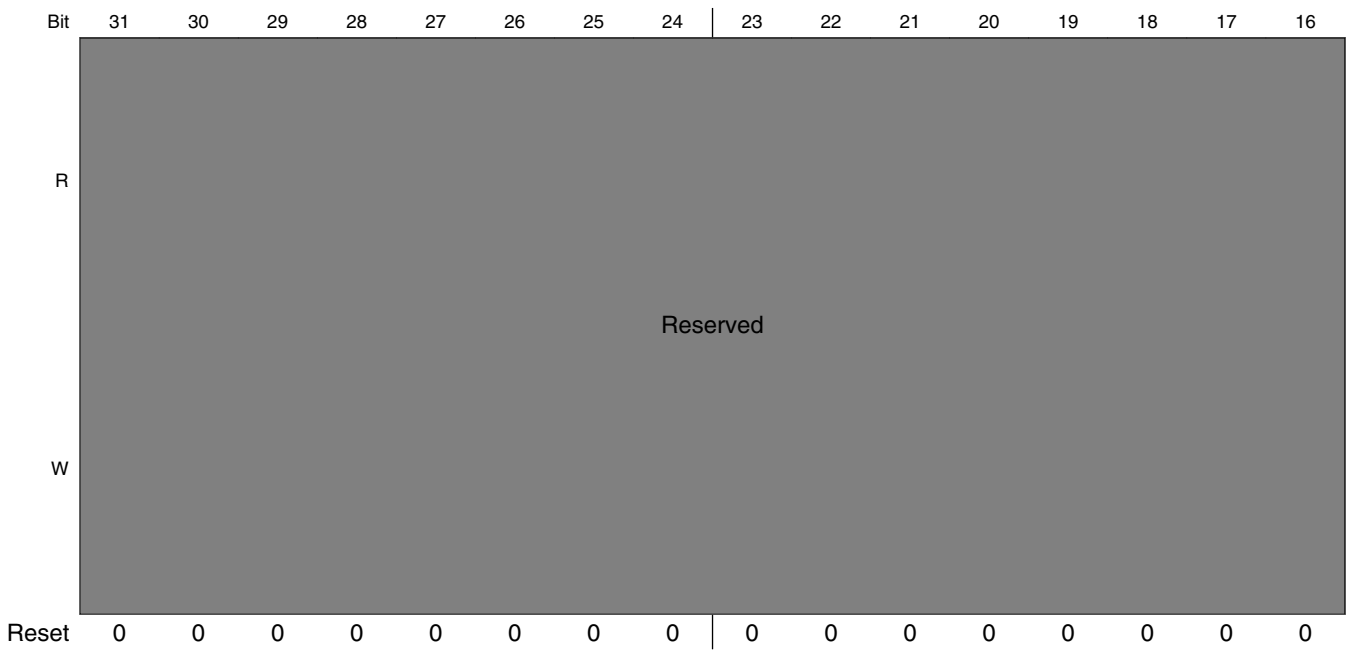
USBx_HCSPARAMS field descriptions (continued)

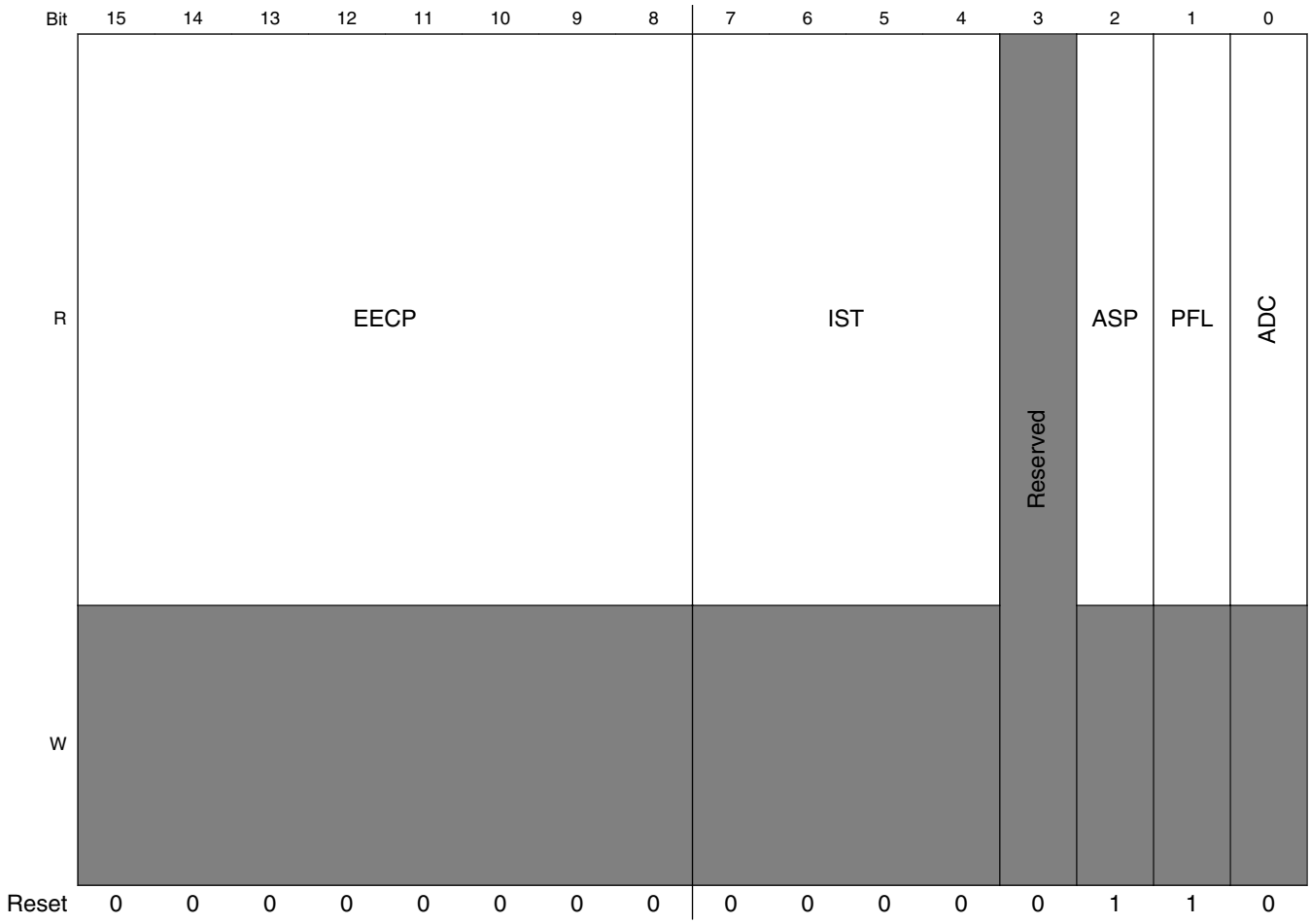
Field	Description
11–8 N_PCC	Number of Ports per Companion Controller This field indicates the number of ports supported per internal Companion Controller. It is used to indicate the port routing configuration to the system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2. The number in this field must be consistent with N_PORTS and N_CC. These bits are '0000b' in all controller core.
7–5 -	This field is reserved. Reserved
4 PPC	Port Power Control This field indicates whether the host controller implementation includes port power control. A one indicates the ports have port power switches. A zero indicates the ports do not have port power switches. The value of this field affects the functionality of the Port Power field in each port status and control register
N_PORTS	Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register. Valid values are in the range of 1h to Fh. A zero in this field is undefined. These bits are always set to '0001b' because all controller cores are Single-Port Host.

11.3.6.15 Host Controller Capability Parameters (USBx_HCCPARAMS)

This register identifies multiple mode control (time-base bit functionality), addressing capability.

Address: Base address + 108h offset





USBx_HCCPARAMS field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 EECP	EHCI Extended Capabilities Pointer. This field indicates the existence of a capabilities list. A value of 00h indicates no extended capabilities are implemented. A non-zero value in this register indicates the offset in PCI configuration space of the first EHCI extended capability. The pointer value must be 40h or greater if implemented to maintain the consistency of the PCI header defined for this class of device. **NOTE:** These bits are set '00h' in all controller core.
7–4 IST	Isochronous Scheduling Threshold. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. These bits are set '00h' in all controller core.
3 -	This field is reserved. Reserved

Table continues on the next page...

USBx_HCCPARAMS field descriptions (continued)

Field	Description
2 ASP	Asynchronous Schedule Park Capability If this bit is set to a one, then the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the *Asynchronous Schedule Park Mode Enable* and *Asynchronous Schedule Park Mode Count* fields in the USBCMD register. **NOTE:** ASP bit reset value: '00b' for OTG controller core, '11b' for Host-only controller core.
1 PFL	Programmable Frame List Flag If this bit is set to zero, then the system software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero. If set to a one, then the system software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous. This bit is set '1b' in all controller core.
0 ADC	64-bit Addressing Capability This bit is set '0b' in all controller core, no 64-bit addressing capability is supported.

11.3.6.16 Device Controller Interface Version (USBx_DCIVERSION)

This register indicates the two-byte BCD encoding of the device controller interface version number.

Address: Base address + 120h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DCIVERSION															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

USBx_DCIVERSION field descriptions

Field	Description
DCIVERSION	Device Controller Interface Version Number Default value is '01h', which means rev0.1.

11.3.6.17 Device Controller Capability Parameters (USBx_DCCPARAMS)

These fields describe the overall device capability of the controller.

NOTE

This register is only available in OTG controller core.

Address: Base address + 124h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved							HC	DC	Reserved		DEN				
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0

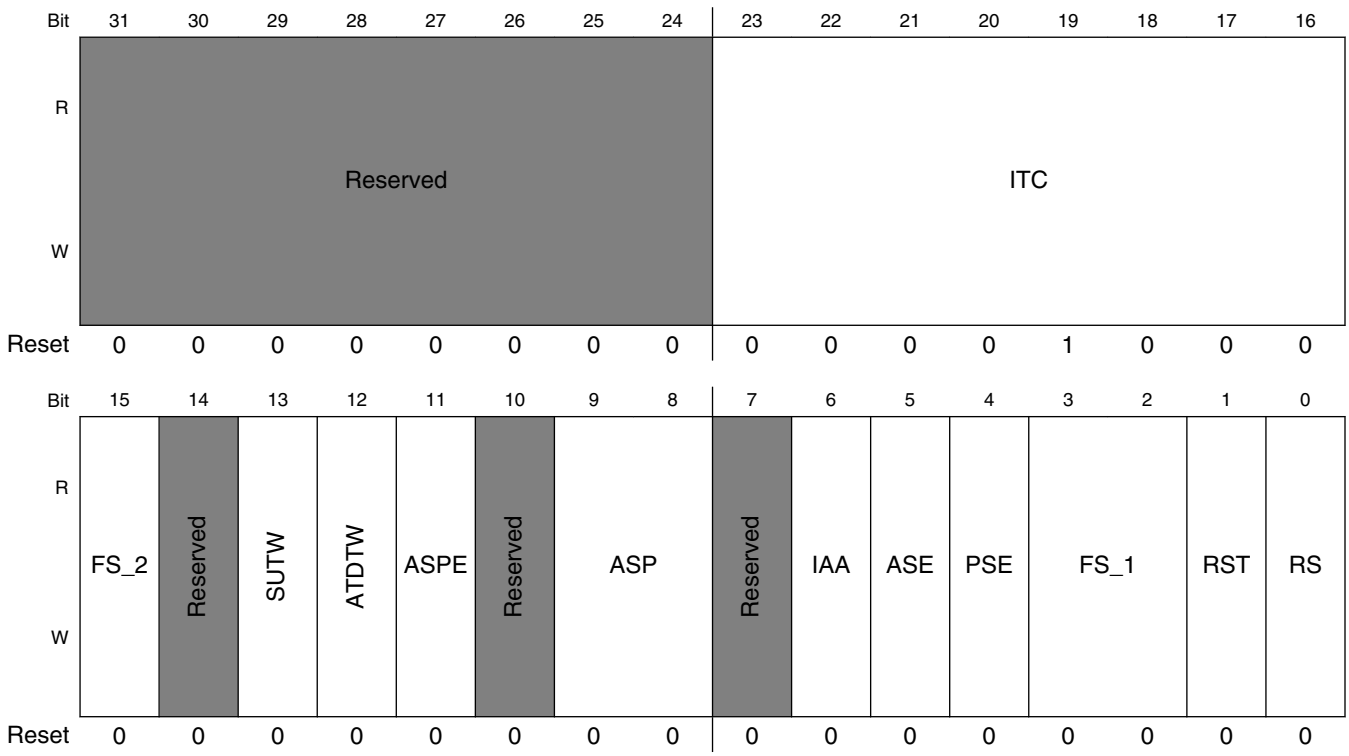
USBx_DCCPARAMS field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 HC	Host Capable When this bit is 1, this controller is capable of operating as an EHCI compatible USB 2.0 host controller.
7 DC	Device Capable When this bit is 1, this controller is capable of operating as a USB 2.0 device.
6–5 -	This field is reserved. Reserved
DEN	Device Endpoint Number This field indicates the number of endpoints built into the device controller. If this controller is not device capable, then this field will be zero. Valid values are 0 - 15.

11.3.6.18 USB Command Register (USB*x*_USBCMD)

The Command Register indicates the command to be executed by the serial bus host/device controller. Writing to the register causes a command to be executed.

Address: Base address + 140h offset



USB*x*_USBCMD field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 ITC	Interrupt Threshold Control -Read/Write. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval 0x00 Immediate (no threshold) 0x01 1 micro-frame 0x02 2 micro-frames 0x04 4 micro-frames 0x08 8 micro-frames 0x10 16 micro-frames

Table continues on the next page...

USBx_USBCMD field descriptions (continued)

Field	Description
	0x20 32 micro-frames 0x40 64 micro-frames
15 FS_2	See also bits 3-2 Frame List Size - (Read/Write or Read Only). [host mode only] This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. **NOTE:** This field is made up from USBCMD bits 15, 3 and 2. Value Meaning 000 1024 elements (4096 bytes) Default value 001 512 elements (2048 bytes) 010 256 elements (1024 bytes) 011 128 elements (512 bytes) 100 64 elements (256 bytes) 101 32 elements (128 bytes) 110 16 elements (64 bytes) 111 8 elements (32 bytes)
14 -	This field is reserved. Reserved
13 SUTW	Setup TripWire - Read/Write. [device mode only] This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off (SLOM bit in USB core register n_USBMODE, see USB Device Mode (USB_USBMODE)) then there is a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software. This bit would also be cleared by hardware when a hazard detected.
12 ATDTW	Add dTD TripWire - Read/Write. [device mode only] This bit is used as a semaphore to ensure proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software. This bit would also be cleared by hardware when state machine is hazard region for which adding a dTD to a primed endpoint may go unrecognized.
11 ASPE	Asynchronous Schedule Park Mode Enable - Read/Write. If the *Asynchronous Park Capability* bit in the HCCPARAMS register is a one, then this bit defaults to a 1h and is R/W. Otherwise the bit must be a zero and is RO. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. **NOTE:** ASPE bit reset value: '0b' for OTG controller core, '1b' for Host-only controller core.
10 -	This field is reserved. Reserved
9–8 ASP	Asynchronous Schedule Park Mode Count - Read/Write. If the *Asynchronous Park Capability* bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is Read-Only. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the

Table continues on the next page...

USBx_USBCMD field descriptions (continued)

Field	Description
	Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 1h to 3h. Software must not write a zero to this bit when *Park Mode Enable* is a one as this will result in undefined behavior. This field is set to 3h in all controller core.
7 -	This field is reserved. Reserved
6 IAA	Interrupt on Async Advance Doorbell - Read/Write. This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a one to this bit when device mode is selected will have undefined results.
5 ASE	Asynchronous Schedule Enable - Read/Write. Default 0b. This bit controls whether the host controller skips processing the Asynchronous Schedule. Only the host controller uses this bit. Values Meaning 0 Do not process the Asynchronous Schedule. 1 Use the ASYNCLISTADDR register to access the Asynchronous Schedule.
4 PSE	Periodic Schedule Enable- Read/Write. Default 0b. This bit controls whether the host controller skips processing the Periodic Schedule. Only the host controller uses this bit. Values Meaning 0 Do not process the Periodic Schedule 1 Use the PERIODICLISTBASE register to access the Periodic Schedule.
3–2 FS_1	See description at bit 15
1 RST	Controller Reset (RESET) - Read/Write. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Host operation mode: When software writes a one to this bit, the Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior. Device operation mode:

Table continues on the next page...

USBx_USBCMD field descriptions (continued)

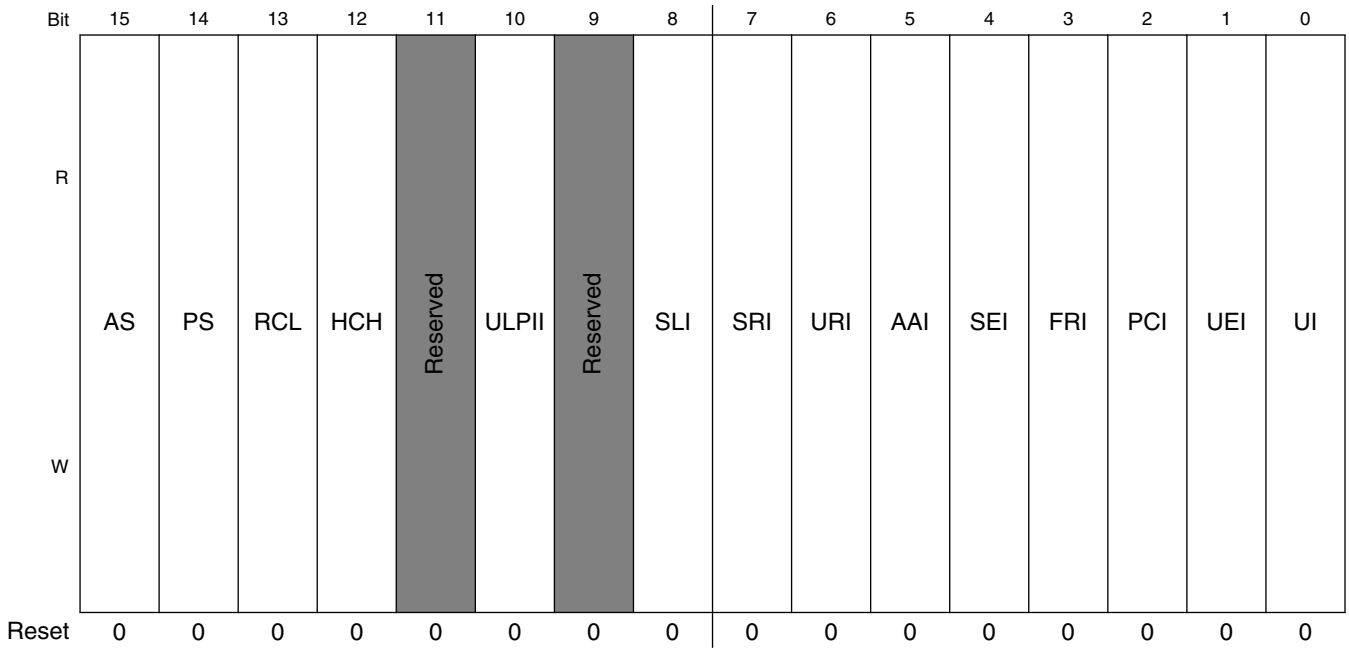
Field	Description
	When software writes a one to this bit, the Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Writing a one to this bit when the device is in the attached state is not recommended, because the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the USBCMD Run/Stop bit should be set to 0.
0 RS	Run/Stop (RS) - Read/Write. Default 0b. 1=Run. 0=Stop. Host operation mode: When set to '1b', the Controller proceeds with the execution of the schedule. The Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the controller is in the Halted state (that is, HCHalted in the USBSTS register is a one). Device operation mode: Writing a one to this bit will cause the controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the controller has been properly initialized. Writing a 0 to this will cause a detach event.

11.3.6.19 USB Status Register (USB*x*_USBSTS)

This register indicates various states of the Host/Device Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus.

Address: Base address + 144h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved						TI1	TI0	Reserved							NAKI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USBx_USBSTS field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25 TI1	General Purpose Timer Interrupt 1(GPTINT1)--R/WC. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero, writing a one to this bit will clear it.
24 TI0	General Purpose Timer Interrupt 0(GPTINT0)--R/WC. This bit is set when the counter in the GPTIMER0CTRL register transitions to zero, writing a one to this bit clears it.
23–17 -	This field is reserved. Reserved
16 NAKI	NAK Interrupt Bit--RO. This bit is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when all Enabled TX/RX Endpoint NAK bits are cleared.
15 AS	Asynchronous Schedule Status - Read Only. This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0). Only used in the host operation mode.
14 PS	Periodic Schedule Status - Read Only.

Table continues on the next page...

USBx_USBSTS field descriptions (continued)

Field	Description
	This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). Only used in the host operation mode.
13 RCL	Reclamation - Read Only. This is a read-only status bit used to detect an empty asynchronous schedule. Only used in the host operation mode.
12 HCH	HCHalted - Read Only. This bit is a zero whenever the Run/Stop bit is a one. The Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Controller hardware (for example, an internal error). Only used in the host operation mode. Default value is '0b' for OTG core, and '1b' for Host1/Host2/Host3 core. This is because OTG core is not operating as host in default. Please see CM bit in USB_n_USBMODE register. **NOTE:** HCH bit reset value: '0b' for OTG controller core, '1b' for Host-only controller core.
11 -	This field is reserved. Reserved
10 ULPII	ULPI Interrupt - R/WC. This bit will be set '1b' by hardware when there is an event completion in ULPI viewport. This bit is usable only if the controller support UPLI interface mode.
9 -	This field is reserved. Reserved
8 SLI	DCSuspend - R/WC. When a controller enters a suspend state from an active state, this bit will be set to a one. The device controller clears the bit upon exiting from a suspend state. Only used in device operation mode.
7 SRI	SOF Received - R/WC. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125ms in HS mode and will be synchronized to the actual SOF that is received. Because the device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to connect and chirp. In host mode, this bit will be set every 125us and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it.
6 URI	USB Reset Received - R/WC. When the device controller detects a USB Reset and enters the default state, this bit will be set to a one. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used in device operation mode.

Table continues on the next page...

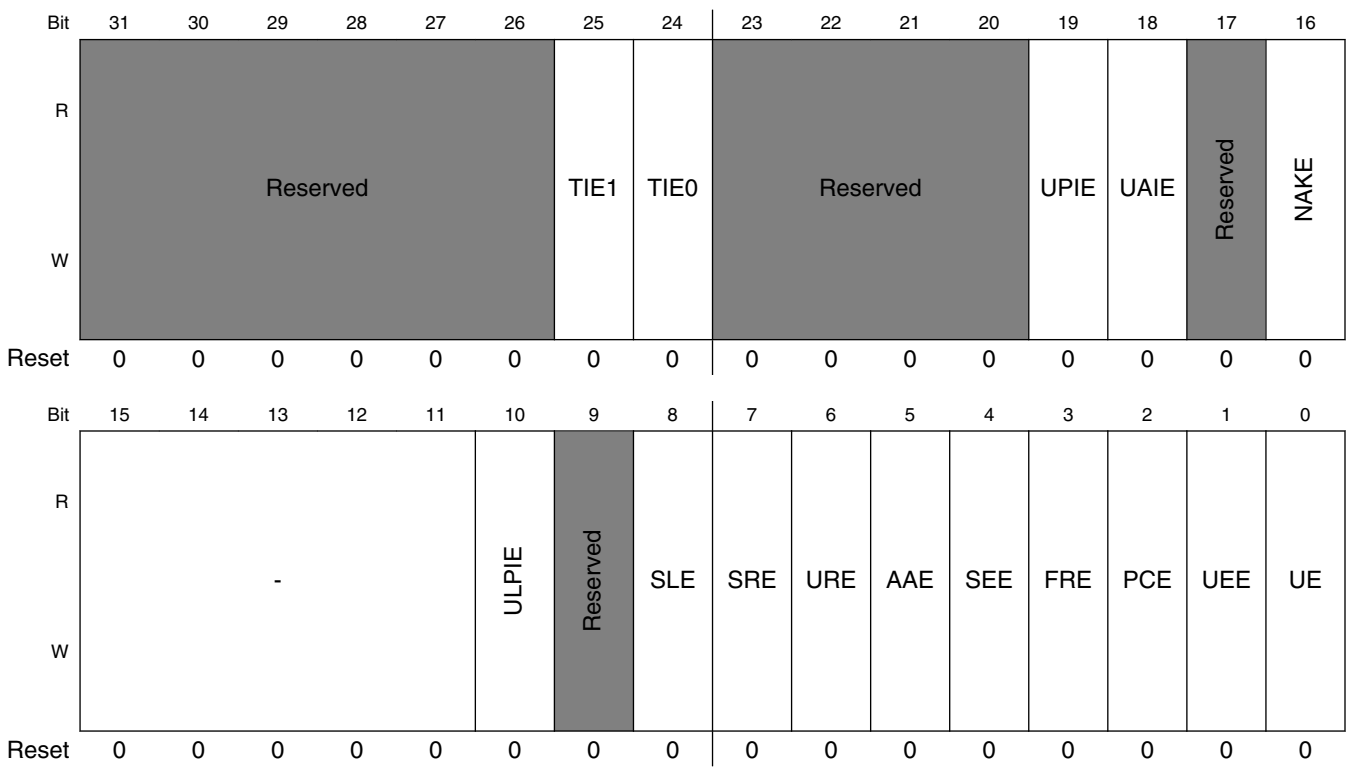
USBx_USBSTS field descriptions (continued)

Field	Description
5 AAI	Interrupt on Async Advance - R/WC. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the n_USBCMD register. This status bit indicates the assertion of that interrupt source. Only used in host operation mode.
4 SEI	System Error- R/WC. This bit is will be set to '1b' when an Error response is seen to a read on the system interface.
3 FRI	Frame List Rollover - R/WC. The Host Controller sets this bit to a one when the Frame List Index rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USB_n_USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FHINDEX [12] toggles. Only used in host operation mode.
2 PCI	Port Change Detect - R/WC. The Host Controller sets this bit to a one when on any port a Connect Status occurs, a Port Enable/ Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively.
1 UEI	USB Error Interrupt (USBERRINT) - R/WC. When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set. The device controller detects resume signaling only.
0 UI	USB Interrupt (USBINT) - R/WC. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.

11.3.6.20 Interrupt Enable Register (USBx_USBINTR)

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt source is active. The USB Status register (n_USBSTS) still shows interrupt sources even if they are disabled by the n_USBINTR register, allowing polling of interrupt events by the software.

Address: Base address + 148h offset



USBx_USBINTR field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25 TIE1	General Purpose Timer #1 Interrupt Enable When this bit is one and the TI1 bit in n_USBSTS register is a one the controller will issue an interrupt.
24 TIE0	General Purpose Timer #0 Interrupt Enable When this bit is one and the TI0 bit in n_USBSTS register is a one the controller will issue an interrupt.
23–20 -	This field is reserved. Reserved
19 UPIE	USB Host Periodic Interrupt Enable

Table continues on the next page...

USBx_USBINTR field descriptions (continued)

Field	Description
	When this bit is one, and the UPI bit in the n_USBSTS register is one, host controller will issue an interrupt at the next interrupt threshold.
18 UAIE	USB Host Asynchronous Interrupt Enable When this bit is one, and the UAI bit in the n_USBSTS register is one, host controller will issue an interrupt at the next interrupt threshold.
17 -	This field is reserved. Reserved
16 NAKE	NAK Interrupt Enable When this bit is one and the NAKI bit in n_USBSTS register is a one the controller will issue an interrupt.
15–11 -	These bits are reserved and should be set to zero.
10 ULPIE	ULPI Interrupt Enable When this bit is one and the UPLII bit in n_USBSTS register is a one the controller will issue an interrupt. This bit is usable only if the controller support UPLI interface mode.
9 -	This field is reserved. Reserved
8 SLE	Sleep Interrupt Enable When this bit is one and the SLI bit in n_n_USBSTS register is a one the controller will issue an interrupt. Only used in device operation mode.
7 SRE	SOF Received Interrupt Enable When this bit is one and the SRI bit in n_USBSTS register is a one the controller will issue an interrupt.
6 URE	USB Reset Interrupt Enable When this bit is one and the URI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in device operation mode.
5 AAE	Async Advance Interrupt Enable When this bit is one and the AAI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in host operation mode.
4 SEE	System Error Interrupt Enable When this bit is one and the SEI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in host operation mode.
3 FRE	Frame List Rollover Interrupt Enable When this bit is one and the FRI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in host operation mode.
2 PCE	Port Change Detect Interrupt Enable When this bit is one and the PCI bit in n_USBSTS register is a one the controller will issue an interrupt.
1 UEE	USB Error Interrupt Enable When this bit is one and the UEI bit in n_USBSTS register is a one the controller will issue an interrupt.
0 UE	USB Interrupt Enable When this bit is one and the UI bit in n_USBSTS register is a one the controller will issue an interrupt.

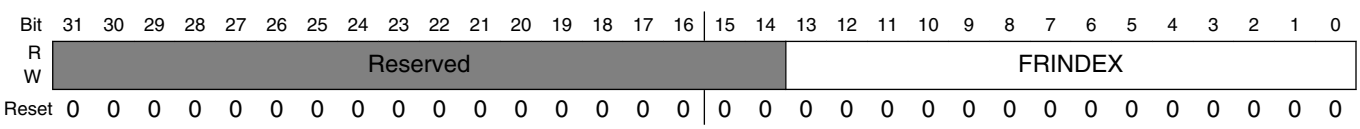
11.3.6.21 USB Frame Index (USBx_FRINDEX)

This register is used by the host controller to index the periodic frame list. The register updates every 125 microseconds (once each micro-frame). Bits [N: 3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in the Frame List Size field in the n_USBCMD register.

This register must be written as a DWord. Byte writes produce undefined results. This register cannot be written unless the Host Controller is in the 'Halted' state as indicated by the HCHalted bit. A write to this register while the Run/Stop hit is set to a one produces undefined results. Writes to this register also affect the SOF value.

In device mode this register is read only and, the device controller updates the FRINDEX [13:3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX [13:3] will be checked against the SOF marker. If FRINDEX [13:3] is different from the SOF marker, FRINDEX [13:3] will be set to the SOF value and FRINDEX [2:0] will be set to zero (that is, SOF for 1 ms frame). If FRINDEX [13:3] is equal to the SOF value, FRINDEX [2:0] will be increment (that is, SOF for 125 us micro-frame.).

Address: Base address + 14Ch offset



USBx_FRINDEX field descriptions

Field	Description
31–14 -	This field is reserved. Reserved
FRINDEX	Frame Index. The value, in this register, increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD [Frame List Size] Number Elements N In device mode the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode bits 2:0 indicate the current microframe. 000 (1024) 12 001 (512) 11

Table continues on the next page...

USBx_FRINDEX field descriptions (continued)

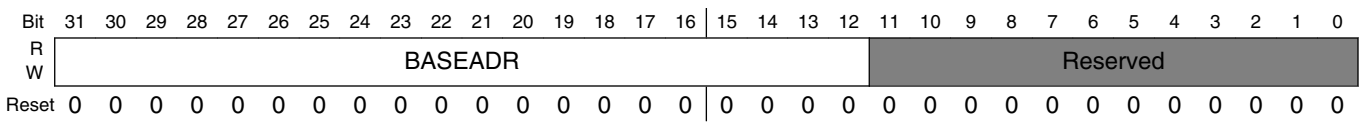
Field	Description
	010 (256) 10 011 (128) 9 100 (64) 8 101 (32) 7 110 (16) 6 111 (8) 5

11.3.6.22 Frame List Base Address (USBx_PERIODICLISTBASE)

Host Controller only

This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (USB_n_FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence.

Address: Base address + 154h offset



USBx_PERIODICLISTBASE field descriptions

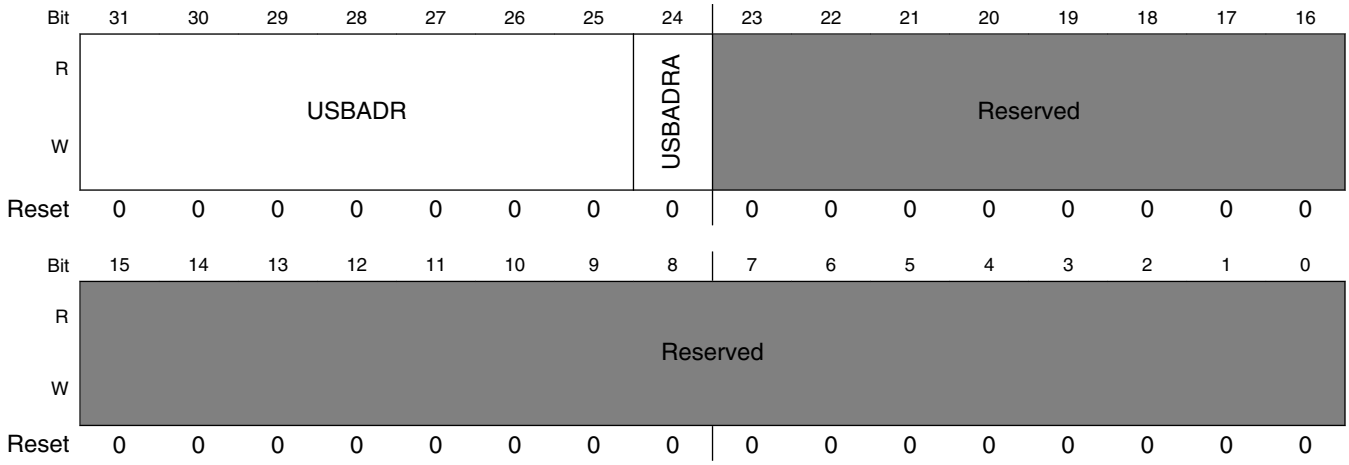
Field	Description
31–12 BASEADR	Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.
-	This field is reserved. Reserved

11.3.6.23 Device Address (USBx_DEVICEADDR)

Device Controller only

The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS descriptor.

Address: Base address + 154h offset



USBx_DEVICEADDR field descriptions

Field	Description
31–25 USBADR	Device Address. These bits correspond to the USB device address
24 USBADRA	Device Address Advance. Default=0. When this bit is '0', any writes to USBADR are instantaneous. When this bit is written to a '1' at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). **NOTE:** After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD can not write of the device address within 2ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
-	This field is reserved. Reserved

11.3.6.24 Next Asynch. Address (USBx_ASYNCLISTADDR)

Host Controller only

This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Bits [4:0] of this register cannot be modified by the system software and will always return a zero when read.

Address: Base address + 158h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ASYBASE																Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBx_ASYNCLISTADDR field descriptions

Field	Description
31–5 ASYBASE	Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH). Only used by the host controller.
-	This field is reserved. Reserved

11.3.6.25 Endpoint List Address (USBx_ENDPTLISTADDR)

Device Controller only

In device mode, this register contains the address of the top of the endpoint list in system memory. Bits [10:0] of this register cannot be modified by the system software and will always return a zero when read.

The memory structure referenced by this physical memory pointer is assumed 64-byte.

Address: Base address + 158h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EPBASE																					Reserved										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

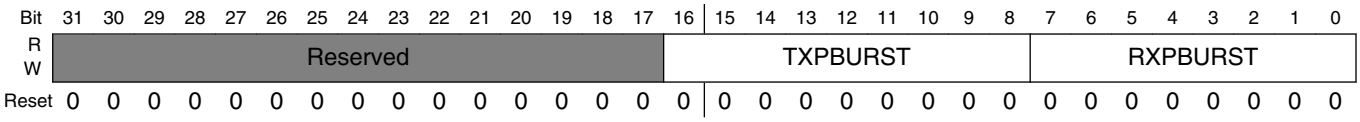
USBx_ENDPTLISTADDR field descriptions

Field	Description
31–11 EPBASE	Endpoint List Pointer(Low). These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Head (QH) (that is, one queue head per endpoint & direction).
-	This field is reserved. Reserved

11.3.6.26 Programmable Burst Size (USBx_BURSTSIZE)

This register is used to control the burst size used during data movement on the AHB master interface. This register is ignored if AHBBRST bits in SBUSCFG register is non-zero value.

Address: Base address + 160h offset



USBx_BURSTSIZE field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16–8 TXPBURST	Programmable TX Burst Size. Default value is determined by TXBURST bits in n_HWTXBUF. This register represents the maximum length of a the burst in 32-bit words while moving data from system memory to the USB bus.
RXPBURST	Programmable RX Burst Size. Default value is determined by TXBURST bits in n_HWRXBUF. This register represents the maximum length of a the burst in 32-bit words while moving data from the USB bus to system memory.

11.3.6.27 TX FIFO Fill Tuning (USBx_TXFILLTUNING)

The fields in this register control performance tuning associated with how the host controller posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include the how much data to post into the FIFO and an estimate for how long that operation should take in the target system.

Definitions:

T_0 = Standard packet overhead

T_1 = Time to send data payload

T_{ff} = Time to fetch packet into TX FIFO up to specified level.

T_s = Total Packet Flight Time (send-only) packet

$T_s = T_0 + T_1$

T_p = Total Packet Time (fetch and send) packet

$T_p = T_{ff} + T_0 + T_1$

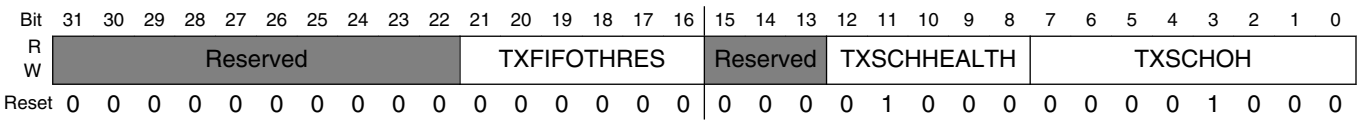
Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure T_p remains before the end of the [micro]frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the [micro]frame is < T_s then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the host controller will eventually recover, a

mark will be made the scheduler health counter to note the occurrence of a "back-off" event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Back-offs can be minimized with use of the n_TSCHHEALTH (T_{ff}) described below.

NOTE

The reset value could vary from instance to instance. Please see the detail in bit field description and ignore reset value in summary table in this case!

Address: Base address + 164h offset



USBx_TXFILLTUNING field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–16 TXFIFOTHRES	FIFO Burst Threshold. (Read/Write) This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USB_n_USBMODE register is set. Default value is '00h' for OTG controller core, and '02h' for Host-only controller core.
15–13 -	This field is reserved. Reserved
12–8 TXSCHHEALTH	Scheduler Health Counter. (Read/Write To Clear) This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31. Default value is '08h' for OTG controller core, and '00h' for Host-only controller core.
TXSCHOH	Scheduler Overhead. (Read/Write) [Default = 0] This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267us when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode. Default value is '08h' for OTG controller core, and '00h' for Host-only controller core.

11.3.6.28 Endpoint NAK (USBx_ENDPTNAK)

Address: Base address + 178h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								EPTN								Reserved								EPRN							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBx_ENDPTNAK field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 EPTN	TX Endpoint NAK - R/WC. Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. Bit [N] - Endpoint #[N], N is 0-7
15–8 -	This field is reserved. Reserved
EPRN	RX Endpoint NAK - R/WC. Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. Bit [N] - Endpoint #[N], N is 0-7

11.3.6.29 Endpoint NAK Enable (USBx_ENDPTNAKEN)

Address: Base address + 17Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								EPTNE								Reserved								EPRNE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBx_ENDPTNAKEN field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 EPTNE	TX Endpoint NAK Enable - R/W. Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. Bit [N] - Endpoint #[N], N is 0-7
15–8 -	This field is reserved. Reserved
EPRNE	RX Endpoint NAK Enable - R/W.

Table continues on the next page...

USBx_ENDPTNAKEN field descriptions (continued)

Field	Description
	Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. Bit [N] - Endpoint #[N], N is 0-7

11.3.6.30 Configure Flag Register (USBx_CONFIGFLAG)

Address: Base address + 180h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															CF
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

USBx_CONFIGFLAG field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 CF	Configure Flag Host software sets this bit as the last action in its process of configuring the Host Controller. This bit controls the default port-routing control logic. 0 Port routing control logic default-routes each port to an implementation dependent classic host controller. 1 Port routing control logic default-routes all ports to this host controller.

11.3.6.31 Port Status & Control (USBx_PORTSC1)

Host Controller

A host controller could implement one to eight port status and control registers. The number is determined by N_PORTs bits in HWSPARAMs register (please see Host Controller Structural Parameters (USB_HCSPARAMS)). Software could read this parameter register to determine how many ports need service.

All controller cores on this product are Single-Port, so there is only one port status and control register for each controller core.

This register is only reset by power on reset or controller core reset. The initial conditions of a port are:

- No device connected
- Port disabled

If the port supports power control, this state remains until port power is supplied (by software).

Device Controller

A controller operating in device mode has only port register one (PORTSC1) and it does not support power control in that mode. Port control in device mode is only used for status port reset, suspend, and current connect status. It is also used to initiate test mode or force signaling and allows software to put the PHY into low power suspend mode and disable the PHY clock.

Address: Base address + 184h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PTS_1		STS	PTW	PSPD		PTS_2	PFSC	PHCD	WKOC	WKDC	WKCN	PTC			
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBx_PORTSC1 field descriptions

Field	Description
31–30 PTS_1	Bit field {bit25, bit31, bit30}: "000b" UTMI/UTMI+ "001b" Reserved "010b" ULPI "011b" Serial/USB 1.1 PHY/IC-USB (FS Only) "100b" HSIC Parallel Transceiver Select (bit25, bit31, bi30). For OTG core, it is Read-Only. Reset value is 000b. **NOTE:** All USB port interface modes are listed in this field description, but not all are supported. For detail feature of each controller core, please see Features . The behaviour is unknown when unsupported interface mode is selected.
29 STS	Serial Transceiver Select - Read Only Serial Transceiver Select 1 Serial Interface Engine is selected 0 Parallel Interface signals is selected Serial Interface Engine can be used in combination with UTMI+/ULPI physical interface to provide FS/LS signaling instead of the parallel interface signals. When this bit is set '1b', serial interface engine will be used instead of parallel interface signals. This bit has no effect unless PTS bits is set to select UTMI+/ULPI interface. The Serial/USB1.1 PHY/IC-USB will use the serial interface engine for FS/LS signaling regardless of this bit value.
28 PTW	Parallel Transceiver Width This bit has no effect if serial interface engine is used. For OTG1/OTG2/Host1 core, it is Read-Only. Reset value is '1b'. 0 Select the 8-bit UTMI interface [60MHz] 1 Select the 16-bit UTMI interface [30MHz]
27–26 PSPD	Port Speed - Read Only. This register field indicates the speed at which the port is operating. 00 Full Speed 01 Low Speed 10 High Speed 11 Undefined
25 PTS_2	See description at bits 31-30
24 PFSC	Port Force Full Speed Connect - Read/Write. Default = 0b. When this bit is set to '1b', the port will be forced to only connect at Full Speed, It disables the chirp sequence that allows the port to identify itself as High Speed. 1 Forced to full speed 0 Normal operation

Table continues on the next page...

USBx_PORTSC1 field descriptions (continued)

Field	Description
23 PHCD	PHY Low Power Suspend - Clock Disable (PLPSCD) - Read/Write. Default = 0b. When this bit is set to '1b', the PHY clock is disabled. Reading this bit will indicate the status of the PHY clock. **NOTE:** The PHY clock cannot be disabled if it is being used as the system clock. In device mode, The PHY can be put into Low Power Suspend when the device is not running (USBCMD Run/Stop=0b) or the host has signalled suspend (PORTSC1 SUSPEND=1b). PHY Low power suspend will be cleared automatically when the host initials resume. Before forcing a resume from the device, the device controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend when the downstream device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software. 1 Disable PHY clock 0 Enable PHY clock
22 WKOC	Wake on Over-current Enable (WKOC_E) - Read/Write. Default = 0b. Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if *Port Power(*Port Status & Control (USB_PORTSC1)*)* is zero.
21 WKDC	Wake on Disconnect Enable (WKDSCNNT_E) - Read/Write. Default=0b. Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if *Port Power(*Port Status & Control (USB_PORTSC1)*)* is zero or in device mode.
20 WKCN	Wake on Connect Enable (WKCNNT_E) - Read/Write. Default=0b. Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if *Port Power(*Port Status & Control (USB_PORTSC1)*)* is zero or in device mode.
19–16 PTC	Port Test Control - Read/Write. Default = 0000b. Refer to Port Test Mode for the operational model for using these test modes and the USB Specification Revision 2.0, Chapter 7 for details on each test mode. The FORCE_ENABLE_FS and FORCE ENABLE_LS are extensions to the test mode support specified in the EHCI specification. Writing the PTC field to any of the FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_MODE_DISABLE will allow the port state machines to progress normally from that point. **NOTE:** *Low speed operations are not supported as a peripheral device.* Any other value than zero indicates that the port is operating in test mode. Value Specific Test 0000 TEST_MODE_DISABLE 0001 J_STATE 0010 K_STATE 0011 SE0 (host) / NAK (device) 0100 Packet 0101 FORCE_ENABLE_HS 0110 FORCE_ENABLE_FS 0111 FORCE_ENABLE_LS 1000-1111 Reserved

Table continues on the next page...

USBx_PORTSC1 field descriptions (continued)

Field	Description
15–14 PIC	Port Indicator Control - Read/Write. Default = Ob. Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is a zero. Refer to the USB Specification Revision 2.0 for a description on how these bits are to be used. This field is zero if *Port Power* is zero. Bit Value Meaning 00 Port indicators are off 01 Amber 10 Green 11 Undefined
13 PO	Port Owner-Read/Write. Default = 0. This bit unconditionally goes to a 0 when the configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is zero System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a one to this bit when the attached device is not a high-speed device. A one in this bit means that an internal companion controller owns and controls the port. Port owner handoff is not supported in all controller cores, therefore this bit will always be 0.
12 PP	Port Power (PP)-Read/Write or Read Only. The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation 0 1b *Read Only - Host controller does not have port power control switches. Each port is hard-wired to power.* 1 1b/0b - *Read/Write. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (that is, PP equals a 0), the port is non-functional and will not report attaches, detaches, etc.* When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitional by the host controller driver from a one to a zero (removing power from the port). This feature is implemented in all controller cores (PPC = 1).
11–10 LS	Line Status-Read Only. These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. In host mode, the use of linestate by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of linestate by the device controller driver is not necessary. The encoding of the bits are: Bits [11:10] Meaning 00 SE0 10 J-state

Table continues on the next page...

USBx_PORTSC1 field descriptions (continued)

Field	Description
	01 K-state 11 Undefined
9 HSP	High-Speed Port - Read Only. Default = 0b. When the bit is one, the host/device connected to the port is in high-speed mode and if set to zero, the host/device connected to the port is not in a high-speed mode. **NOTE:** HSP is redundant with PSPD(bit 27, 26) but remained for compatibility.
8 PR	Port Reset - Read/Write or Read Only. Default = 0b. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. Default 0. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. *This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver.* In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register. This field is zero if *Port Power(*Port Status & Control (USB_PORTSC1)*)* is zero.
7 SUSP	Suspend - Read/Write or Read Only. Default = 0b. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/Write. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the *Force Port Resume* bit to zero. The host controller ignores a write of zero to this bit. If host software sets this bit to a one when the port is not enabled (that is, *Port enabled* bit is a zero) the results are undefined. This field is zero if *Port Power(*Port Status & Control (USB_PORTSC1)*)* is zero in host mode. In Device Mode: Read Only. In device mode this bit is a read only status bit.
6 FPR	Force Port Resume -Read/Write. 1= Resume detected/driven on port. 0=No resume (K-state) detected/driven on port. Default = 0. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the *Port Change Detect* bit in the USBSTS register is also set to one. *This bit*

Table continues on the next page...

USBx_PORTSC1 field descriptions (continued)

Field	Description
	will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit will remain a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time the resume operation, clear the bit the port control state switches to HS or FS idle. This field is zero if *Port Power(*Port Status & Control (USB_PORTSC1)*)* is zero in host mode. This bit is not-EHCI compatible. In Device mode: After the device has been in Suspend State for 5ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit wil be cleared because a K-to-J transition detected, the *Port Change Detect* bit in the USBSTS register is also set to one.
5 OCC	Over-current Change-R/WC. Default=0. This bit is set '1b' by hardware when there is a change to Over-current Active. Software can clear this bit by writing a one to this bit position.
4 OCA	Over-current Active-Read Only. Default 0. This bit will automatically transition from one to zero when the over current condition is removed. 1 This port currently has an over-current condition 0 This port does not have an over-current condition.
3 PEC	Port Enable/Disable Change-R/WC. 1=Port enabled/disabled status has changed. 0=No change. Default = 0. In Host Mode: For the root hub, this bit is set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if *Port Power(*Port Status & Control (USB_PORTSC1)*)* is zero. In Device mode: The device port is always enabled, so this bit is always '0b'.
2 PE	Port Enabled/Disabled-Read/Write. 1=Enable. 0=Disable. Default 0. In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if *Port Power(*Port Status & Control (USB_PORTSC1)*)* is zero in host mode. In Device Mode: The device port is always enabled, so this bit is always '1b'.

Table continues on the next page...

USBx_PORTSC1 field descriptions (continued)

Field	Description
1 CSC	Connect Status Change-R/WC. 1 =Change in Current Connect Status. 0=No change. Default 0. In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (that is, the bit will remain set). Software clears this bit by writing a one to it. This field is zero if *Port Power(*Port Status & Control (USB_PORTSC1)*)* is zero in host mode. In Device Mode: This bit is undefined in device controller mode.
0 CCS	Current Connect Status-Read Only. In Host Mode: 1=Device is present on port. 0=No device is present. Default = 0. This value reflects the current state of the port, and may not correspond directly to the event that caused the *Connect Status Change* bit (Bit 1) to be set. This field is zero if *Port Power(*Port Status & Control (USB_PORTSC1)*)* is zero in host mode. In Device Mode: 1=Attached. 0=Not Attached. Default=0. A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended.

11.3.6.32 On-The-Go Status & control (USBx_OTGSC)

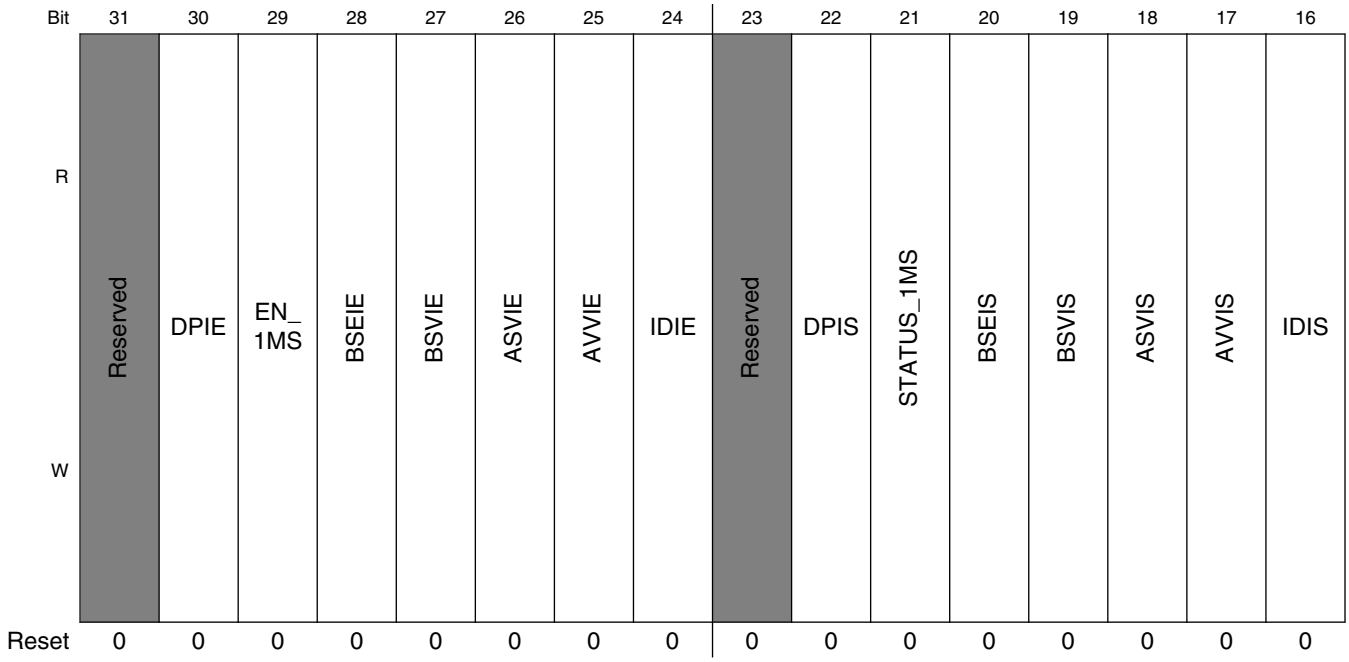
This register is availabe only in OTG controller core. It has four sections:

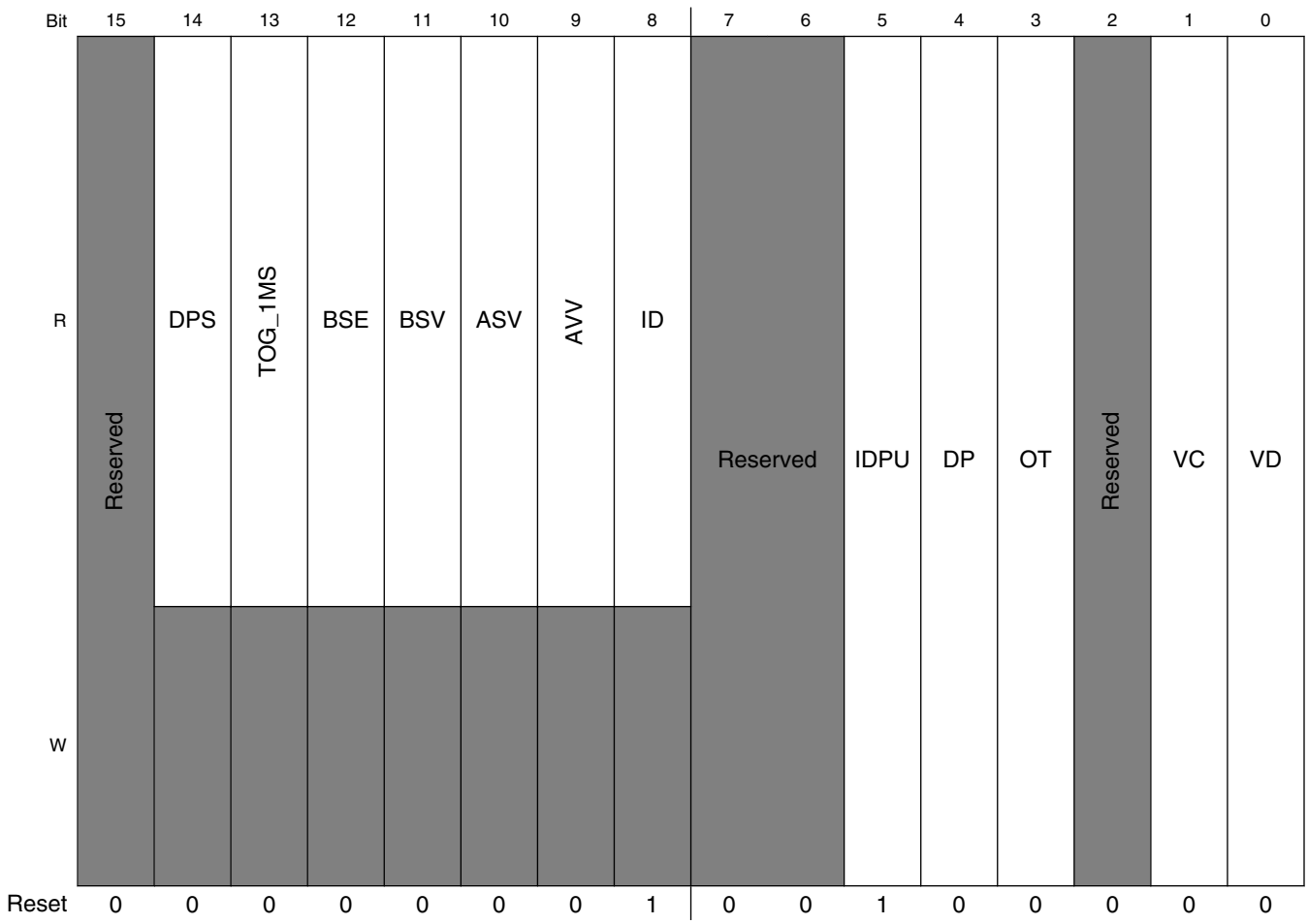
- OTG Interrupt enables (Read/Write)
- OTG Interrupt status (Read/Write to Clear)
- OTG Status inputs (Read Only)
- OTG Controls (Read/Write)

The status inputs are debounced using a 1 ms time constant. Values on the status inputs that do not persist for more than 1 ms does not cause an update of the status input register, or cause an OTG interrupt.

See also USB Device Mode (USB_USBMODE) register.

Address: Base address + 1A4h offset





USBx_OTGSC field descriptions

Field	Description
31 -	This field is reserved. Reserved
30 DPIE	Data Pulse Interrupt Enable
29 EN_1MS	1 millisecond timer Interrupt Enable - Read/Write
28 BSEIE	B Session End Interrupt Enable - Read/Write. Setting this bit enables the B session end interrupt.
27 BSVIE	B Session Valid Interrupt Enable - Read/Write. Setting this bit enables the B session valid interrupt.
26 ASVIE	A Session Valid Interrupt Enable - Read/Write. Setting this bit enables the A session valid interrupt.
25 AVVIE	A VBus Valid Interrupt Enable - Read/Write. Setting this bit enables the A VBus valid interrupt.
24 IDIE	USB ID Interrupt Enable - Read/Write. Setting this bit enables the USB ID interrupt.
23 -	This field is reserved. Reserved
22 DPIS	Data Pulse Interrupt Status - Read/Write to Clear. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC1(0)[PP] = 0. Software must write a one to clear this bit.
21 STATUS_1MS	1 millisecond timer Interrupt Status - Read/Write to Clear. This bit is set once every millisecond. Software must write a one to clear this bit.
20 BSEIS	B Session End Interrupt Status - Read/Write to Clear. This bit is set when VBus has fallen below the B session end threshold. Software must write a one to clear this bit.
19 BSVIS	B Session Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the B session valid threshold. Software must write a one to clear this bit.
18 ASVIS	A Session Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the A session valid threshold. Software must write a one to clear this bit.
17 AVVIS	A VBus Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the VBus valid threshold on an A device. Software must write a one to clear this bit.
16 IDIS	USB ID Interrupt Status - Read/Write. This bit is set when a change on the ID input has been detected. Software must write a one to clear this bit.

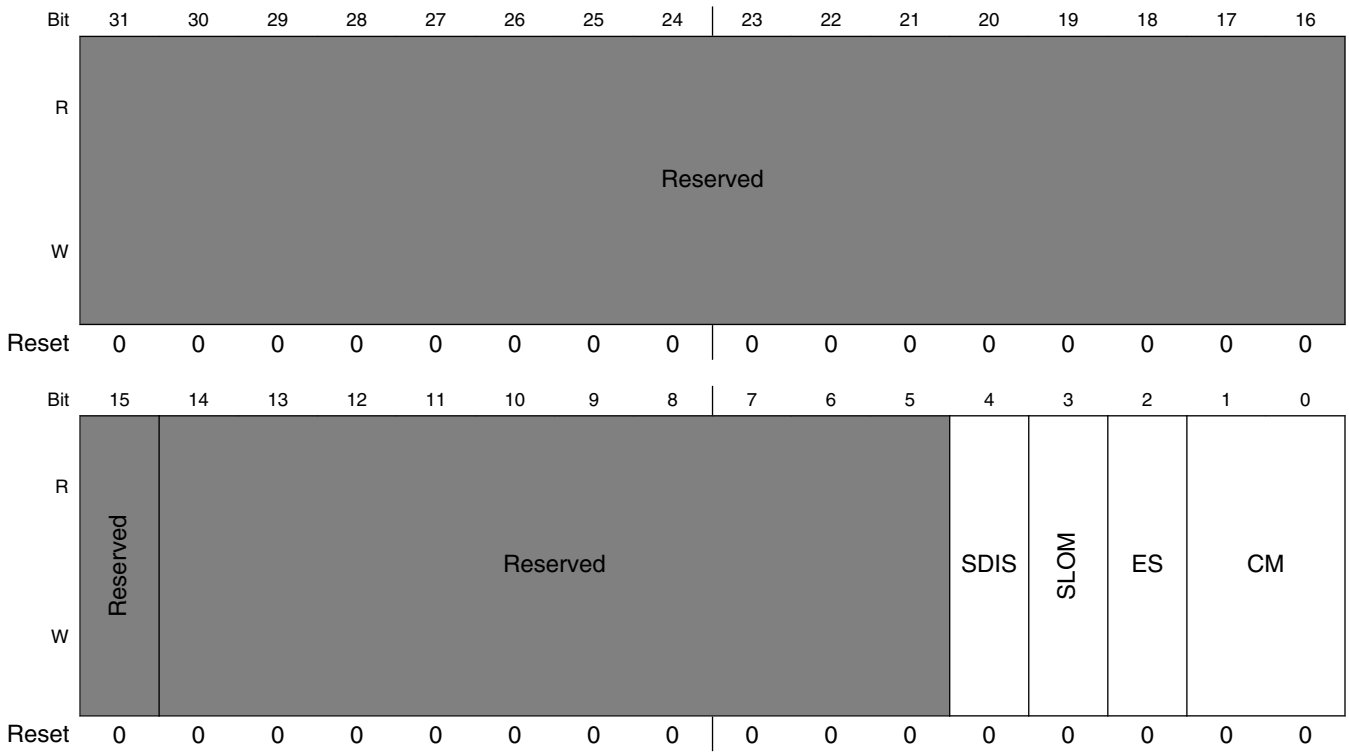
Table continues on the next page...

USBx_OTGSC field descriptions (continued)

Field	Description
15 -	This field is reserved. Reserved
14 DPS	Data Bus Pulsing Status - Read Only. A '1' indicates data bus pulsing is being detected on the port.
13 TOG_1MS	1 millisecond timer toggle - Read Only. This bit toggles once per millisecond.
12 BSE	B Session End - Read Only. Indicates VBus is below the B session end threshold.
11 BSV	B Session Valid - Read Only. Indicates VBus is above the B session valid threshold.
10 ASV	A Session Valid - Read Only. Indicates VBus is above the A session valid threshold.
9 AVV	A VBus Valid - Read Only. Indicates VBus is above the A VBus valid threshold.
8 ID	USB ID - Read Only. 0 = A device, 1 = B device
7–6 -	This field is reserved. Reserved
5 IDPU	ID Pullup - Read/Write This bit provide control over the ID pull-up resistor; 0 = off, 1 = on [default]. When this bit is 0, the ID input will not be sampled.
4 DP	Data Pulsing - Read/Write. Setting this bit causes the pullup on DP to be asserted for data pulsing during SRP.
3 OT	OTG Termination - Read/Write. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM.
2 -	This field is reserved. Reserved
1 VC	VBUS Charge - Read/Write. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP.
0 VD	VBUS_Discharge - Read/Write. Setting this bit causes VBus to discharge through a resistor.

11.3.6.33 USB Device Mode (USBx_USBMODE)

Address: Base address + 1A8h offset



USBx_USBMODE field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14–5 -	This field is reserved. Reserved
4 SDIS	Stream Disable Mode. (0 - Inactive [default]; 1 - Active) Device Mode: Setting to a '1' disables double priming on both RX and TX for low bandwidth systems. This mode ensures that when the RX and TX buffers are sufficient to contain an entire packet that the standard double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. Note: In High Speed Mode, all packets received are responded to with a NYET handshake when stream disable is active. Host Mode: Setting to a '1' ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB. **NOTE:** Time duration to pre-fill the FIFO becomes significant when stream disable is active. See TX FIFO Fill Tuning (USB_TXFILLTUNING) and TXTTFILLTUNING [MPH Only] to characterize the adjustments needed for the scheduler when using this feature.

Table continues on the next page...

USBx_USBMODE field descriptions (continued)

Field	Description
	NOTE: The use of this feature substantially limits of the overall USB performance that can be achieved.
3 SLOM	Setup Lockout Mode. In device mode, this bit controls behavior of the setup lock mechanism. See Control Endpoint Operation Model . 0 Setup Lockouts On (default); 1 Setup Lockouts Off (DCD requires use of Setup Data Buffer Tripwire in USB Command Register (USB_USBCMD) .
2 ES	Endian Select - Read/Write. This bit can change the byte alignment of the transfer buffers to match the host microprocessor. The bit fields in the microprocessor interface and the data structures are unaffected by the value of this bit because they are based upon the 32-bit word. Bit Meaning 0 Little Endian [Default] 1 Big Endian
CM	Controller Mode - R/WO. Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host & device capability, the controller defaults to an idle state and needs to be initialized to the desired operating mode after reset. For combination host/device controllers, this register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the *RESET* bit in the USBCMD register before reprogramming this register. For OTG controller core, reset value is '00b'. For Host-only controller core, reset value is '11b'. 00 Idle [Default for combination host/device] 01 Reserved 10 Device Controller [Default for device only controller] 11 Host Controller [Default for host only controller]

11.3.6.34 Endpoint Setup Status (USBx_ENDPTSETUPSTAT)

Address: Base address + 1ACh offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																ENDPTSETUPSTAT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBx_ENDPTSETUPSTAT field descriptions

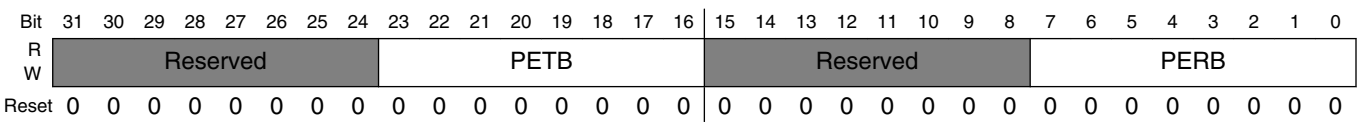
Field	Description
31–16 -	This field is reserved. Reserved
ENDPTSETUPSTAT	Setup Endpoint Status. For every setup transaction that is received, a corresponding bit in this register is set to one. Software must clear or acknowledge the setup transfer by writing a one to a respective bit after it has read the setup data from Queue head. The response to a setup packet as in the order of operations and total response time is crucial to limit bus time outs while the setup lock our mechanism is engaged. See Managing Endpoints in the Device Operational Model. This register is only used in device mode.

11.3.6.35 Endpoint Prime (USBx_ENDPTPRIME)

This register is only used in device mode.

When software sets the prime bit for a given endpoint, the device controller loads the transfer descriptor, pointed to by the queue head, such that the endpoint is ready to transmit or receive when the host sends a request (IN/OUT token). The endpoint will NAK all requests from the host until the endpoint is primed. The controller will automatically re-prime the endpoint with a new transfer descriptor when one is found via the next_dtd pointer of the current transfer descriptor. Hence, the prime bit must only be set by software when a descriptor is added to the queue head.

Address: Base address + 1B0h offset



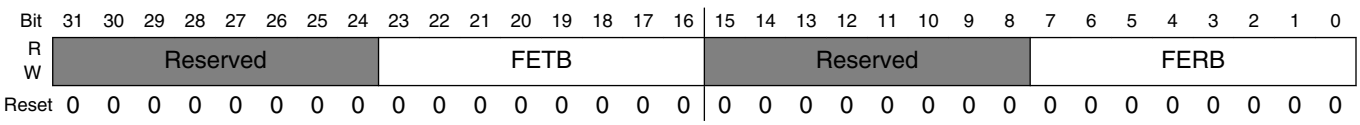
USBx_ENDPTPRIME field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 PETB	Prime Endpoint Transmit Buffer - R/WS. For each endpoint a corresponding bit is used to request that a buffer is prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a one to the corresponding bit when posting a new transfer descriptor to an endpoint queue head. Hardware automatically uses this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware clears this bit when the associated endpoint(s) is (are) successfully primed. **NOTE:** These bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PETB[N] - Endpoint #N, N is in 0..7
15–8 -	This field is reserved. Reserved
PERB	Prime Endpoint Receive Buffer - R/WS. For each endpoint, a corresponding bit is used to request a buffer prepare for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a one to the corresponding bit whenever posting a new transfer descriptor to an endpoint queue head. Hardware automatically uses this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware clears this bit when the associated endpoint(s) is (are) successfully primed. **NOTE:** These bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PERB[N] - Endpoint #N, N is in 0..7

11.3.6.36 Endpoint Flush (USBx_ENDPTFLUSH)

This register is only used in device mode.

Address: Base address + 1B4h offset



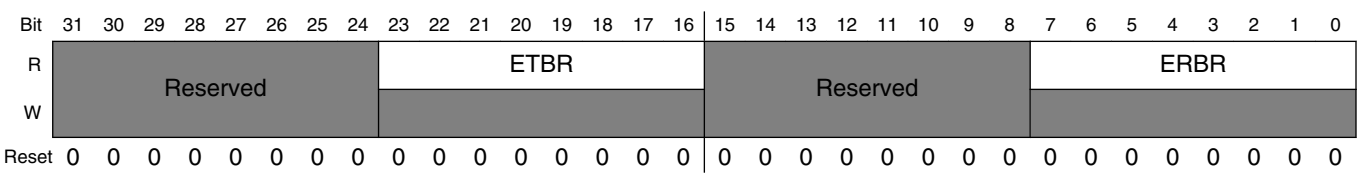
USBx_ENDPTFLUSH field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 FETB	Flush Endpoint Transmit Buffer - R/WS. Writing one to a bit(s) in this register causes the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer continues until completion. Hardware clears this register after the endpoint flush operation is successful. FETB[N] - Endpoint #N, N is in 0..7
15–8 -	This field is reserved. Reserved
FERB	Flush Endpoint Receive Buffer - R/WS. Writing one to a bit(s) causes the assocUOGiated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer continues until completion. Hardware clears this register after the endpoint flush operation is successful. FERB[N] - Endpoint #N, N is in 0..7\

11.3.6.37 Endpoint Status (USBx_ENDPTSTAT)

This register is only used in device mode.

Address: Base address + 1B8h offset



USBx_ENDPTSTAT field descriptions

Field	Description
31–24 -	This field is reserved. Reserved

Table continues on the next page...

USBx_ENDPTSTAT field descriptions (continued)

Field	Description
23–16 ETBR	Endpoint Transmit Buffer Ready -- Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There is always a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. **NOTE:** These bits are momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ETBR[N] - Endpoint #N, N is in 0..7
15–8 -	This field is reserved. Reserved
ERBR	Endpoint Receive Buffer Ready -- Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPRIME register. There is always a delay between setting a bit in the ENDPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. **NOTE:** These bits are momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ERBR[N] - Endpoint #N, N is in 0..7

11.3.6.38 Endpoint Complete (USBx_ENDPTCOMPLETE)

This register is only used in device mode.

Address: Base address + 1BCh offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								ETCE								Reserved								ERCE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBx_ENDPTCOMPLETE field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 ETCE	Endpoint Transmit Complete Event - R/WC. Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit is set simultaneously with the *USBINT* . Writing one clears the corresponding bit in this register. ETCE[N] - Endpoint #N, N is in 0..7
15–8 -	This field is reserved. Reserved

Table continues on the next page...

USBx_ENDPTCOMPLETE field descriptions (continued)

Field	Description
ERCE	Endpoint Receive Complete Event - RW/C. Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit is set simultaneously with the *USBINT* . Writing one clears the corresponding bit in this register. ERCE[N] - Endpoint #N, N is in 0..7

11.3.6.39 Endpoint Control0 (USBx_ENDPTCTRL0)

Every Device implements Endpoint 0 as a control endpoint.

Address: Base address + 1C0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	Reserved			TXT		Reserved	TXS
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								RXE	Reserved			RXT		Reserved	RXS
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

USBx_ENDPTCTRL0 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 1 Enabled Endpoint0 is always enabled.

Table continues on the next page...

USBx_ENDPTCTRL0 field descriptions (continued)

Field	Description
22–20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 - Control Endpoint0 is fixed as a Control End Point.
17 -	This field is reserved. Reserved
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK [Default] 1 End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues returning STALL until the bit is cleared by software or it is automatically cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. **NOTE:** There is a slight delay (50 clocks max.) between the endptsetupstat being cleared and hardware continuing to clear this bit. In most systems it is unlikely the DCD software will observe this delay. However, should the dcd observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a newsetup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 1 Enabled Endpoint0 is always enabled.
6–4 -	This field is reserved. Reserved
3–2 RXT	RX Endpoint Type - Read/Write 00 Control Endpoint0 is fixed as a Control End Point.
1 -	This field is reserved. Reserved
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues returning STALL until the bit is cleared by software or it is automatically cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared.

Table continues on the next page...

USBx_ENDPTCTRL0 field descriptions (continued)

Field	Description
	NOTE: There is a slight delay (50 clocks max.) between the endptsetupstat being cleared and hardware continuing to clear this bit. In most systems it is unlikely the dcd software will observe this delay. However, should the dcd observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a newsetup has been received by checking the associated endptsetupstat bit.

11.3.6.40 Endpoint Control 1 (USBx_ENDPTCTRL1)

This is endpoint control register for endpoint 1 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: Base address + 1C4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBx_ENDPTCTRL1 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved

Table continues on the next page...

USBx_ENDPTCTRL1 field descriptions (continued)

Field	Description
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Reserved
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

11.3.6.41 Endpoint Control 2 (USBx_ENDPTCTRL2)

This is endpoint control register for endpoint 2 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: Base address + 1C8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBx_ENDPTCTRL2 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.

Table continues on the next page...

USBx_ENDPTCTRL2 field descriptions (continued)

Field	Description
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence

Table continues on the next page...

USBx_ENDPTCTRL2 field descriptions (continued)

Field	Description
	Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Reserved
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

11.3.6.42 Endpoint Control 3 (USBx_ENDPTCTRL3)

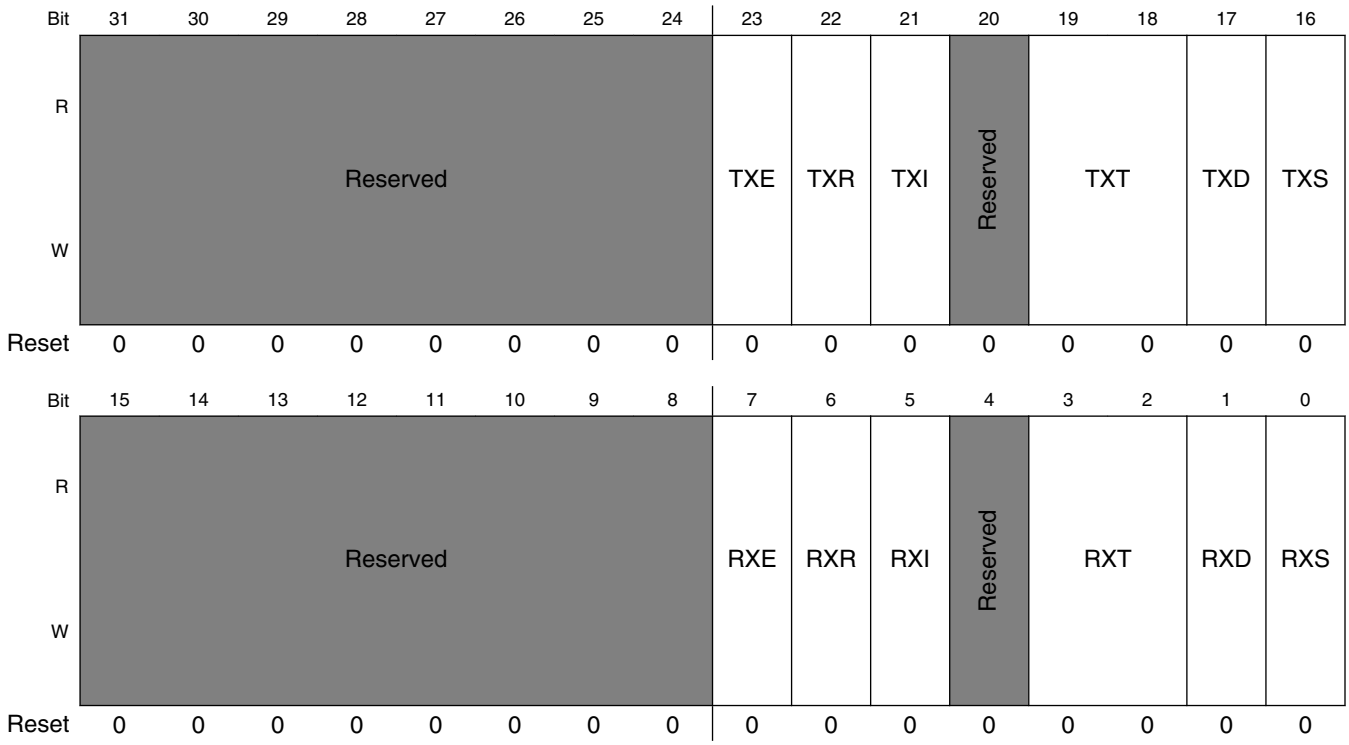
This is endpoint control register for endpoint 3 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control

causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: Base address + 1CCh offset



USBx_ENDPTCTRL3 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved

Table continues on the next page...

USBx_ENDPTCTRL3 field descriptions (continued)

Field	Description
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control

Table continues on the next page...

USBx_ENDPTCTRL3 field descriptions (continued)

Field	Description
	01 Isochronous 10 Bulk 11 Reserved
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

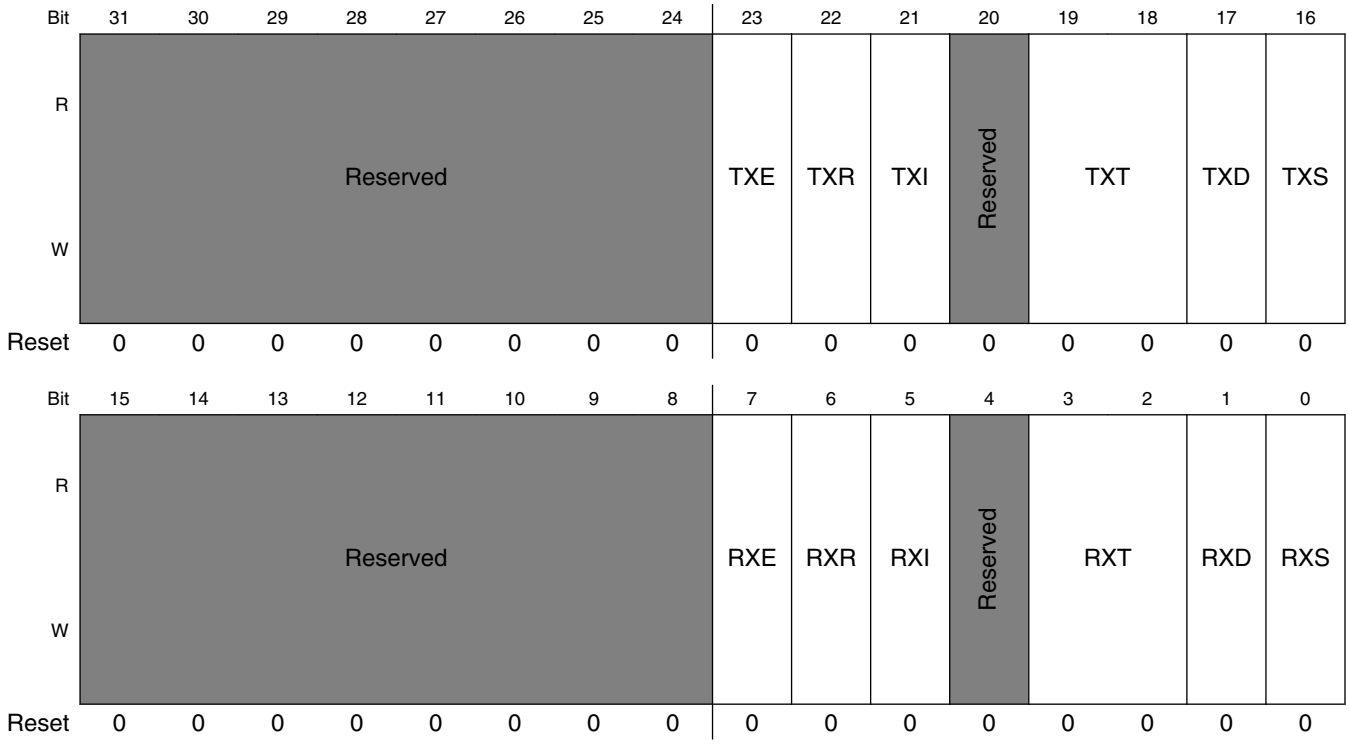
11.3.6.43 Endpoint Control 4 (USBx_ENDPTCTRL4)

This is endpoint control register for endpoint 4 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: Base address + 1D0h offset



USBx_ENDPTCTRL4 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous

Table continues on the next page...

USBx_ENDPTCTRL4 field descriptions (continued)

Field	Description
	10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Reserved

Table continues on the next page...

USBx_ENDPTCTRL4 field descriptions (continued)

Field	Description
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

11.3.6.44 Endpoint Control 5 (USBx_ENDPTCTRL5)

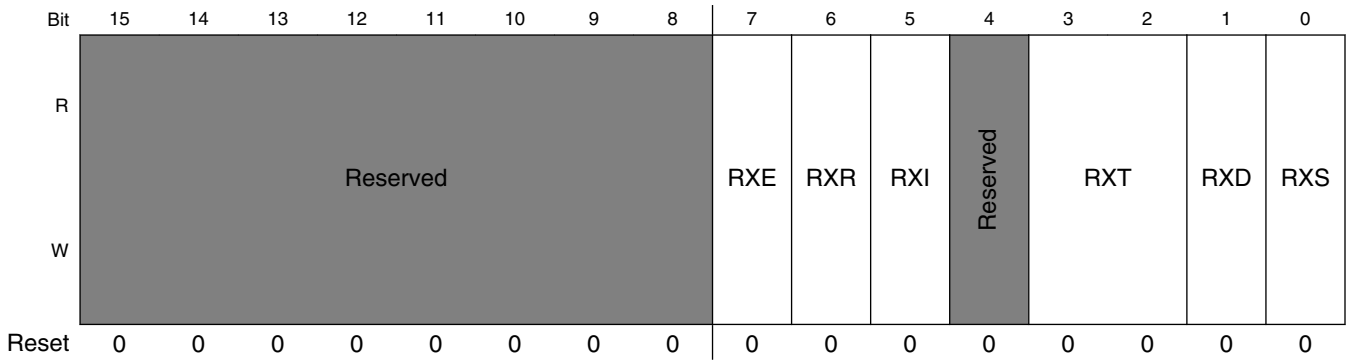
This is endpoint control register for endpoint 5 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: Base address + 1D4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USBx_ENDPTCTRL5 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled

Table continues on the next page...

USBx_ENDPTCTRL5 field descriptions (continued)

Field	Description
	This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Reserved
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled

Table continues on the next page...

USBx_ENDPTCTRL5 field descriptions (continued)

Field	Description
	This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

11.3.6.45 Endpoint Control 6 (USBx_ENDPTCTRL6)

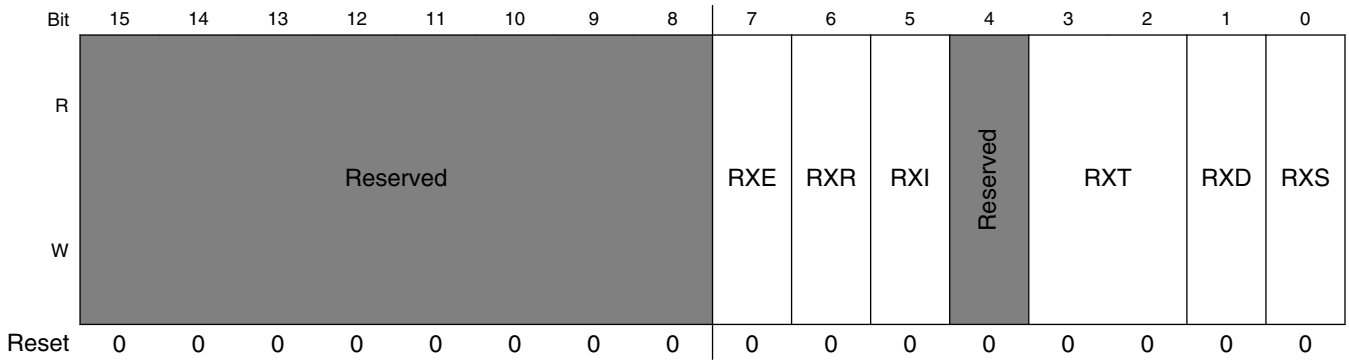
This is endpoint control register for endpoint 6 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: Base address + 1D8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USBx_ENDPTCTRL6 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled

Table continues on the next page...

USBx_ENDPTCTRL6 field descriptions (continued)

Field	Description
	This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Reserved
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled

Table continues on the next page...

USBx_ENDPTCTRL6 field descriptions (continued)

Field	Description
	This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

11.3.6.46 Endpoint Control 7 (USBx_ENDPTCTRL7)

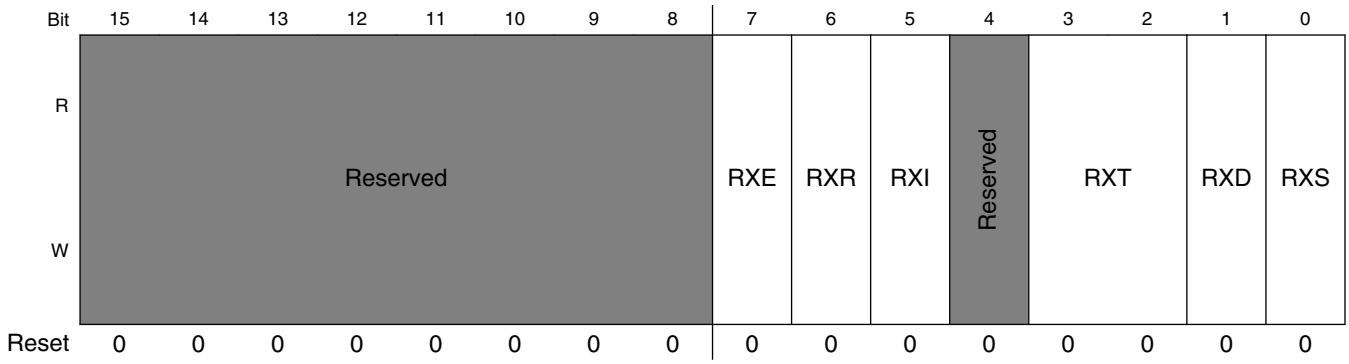
This is endpoint control register for endpoint 7 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: Base address + 1DCh offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USBx_ENDPTCTRL7 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled

Table continues on the next page...

USBx_ENDPTCTRL7 field descriptions (continued)

Field	Description
	This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Reserved
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled

Table continues on the next page...

USBx_ENDPTCTRL7 field descriptions (continued)

Field	Description
	This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

11.4 Universal Serial Bus 2.0 Integrated PHY (USB-PHY)

11.4.1 USB PHY Overview

The chip contains 2 integrated USB 2.0 PHY macrocells capable of connecting to USB host/device systems at the USB low-speed (LS) rate of 1.5 Mbits/s, full-speed (FS) rate of 12 Mbits/s or at the USB 2.0 high-speed (HS) rate of 480 Mbits/s.

The integrated PHY provides a standard UTM interface. The USB_*n*_DN and USB_*n*_DP pins connect directly to a USB connector.

The following subsections describe the external interfaces, internal interfaces, major blocks, and programmable registers that comprise the integrated USB 2.0 PHY.

11.4.2 Operation

The UTM provides a 16-bit interface to the USB controller. This interface is clocked at 30 MHz.

- The digital portions of the USBPHY block include the UTMI, digital transmitter, digital receiver, and the programmable registers.
- The analog transceiver section comprises an analog receiver and an analog transmitter, as shown in Figure 11-84.

11.4.2.1 UTMI

The UTMI block handles the line_state bits, reset buffering, suspend distribution, transceiver speed selection, transceiver termination selection, and clock generation logic for the clocks derived from the PHY's 480 MHz USB PLL.

11.4.2.2 Digital Transmitter

The digital transmitter receives the 16-bit transmit data from the USB controller and handles the tx_valid, tx_validh and tx_ready handshake.

In addition, it contains the transmit serializer that converts the 16-bit parallel words at 30 MHz to a single bitstream at 480 Mbit for high-speed or 12 Mbit for full-speed or 1.5 Mbit for low-speed. It does this while implementing the bit-stuffing algorithm and the NRZI encoder that are used to remove the DC component from the serial bitstream. The output of this encoder is sent to the low-speed (LS), full-speed (FS) or high-speed (HS) drivers in the analog transceiver section's transmitter block.

11.4.2.3 Digital Receiver

The digital receiver receives the raw serial bitstream from the low speed (LS) differential transceiver, full speed (FS) differential transceiver, and multiphase 480 MHz sampled data from the high speed (HS) differential transceiver.

As the phase of the USB host transmitter shifts relative to the local PLL, the receiver section's HS DLL tracks these changes to give a reliable sample of the incoming 480 Mbit/s bitstream. Since this sample point shifts relative to the PLL phase used by the digital logic, a rate-matching elastic buffer is provided to cross this clock domain boundary. Once the bitstream is in the local clock domain, an NRZI decoder and bit unstuffer restore the original payload data bitstream and pass it to a deserializer and holding register. The receive state machine handles the rx_valid, rx_validh, and handshake with the USB controller. The handshake is not interlocked, in that there is no rx_ready signal coming from the controller. The controller must take each 16-bit value as presented by the PHY. The receive state machine provides an rx_active signal to the controller that indicates when it is inside a valid packet (SYNC detected, and so on).

11.4.2.4 Analog Receiver

The analog receiver comprises five differential receivers, two single-ended receivers, and a 9X, 480 MHz HS data sampling module

, as shown in the figure below and described further in this section.

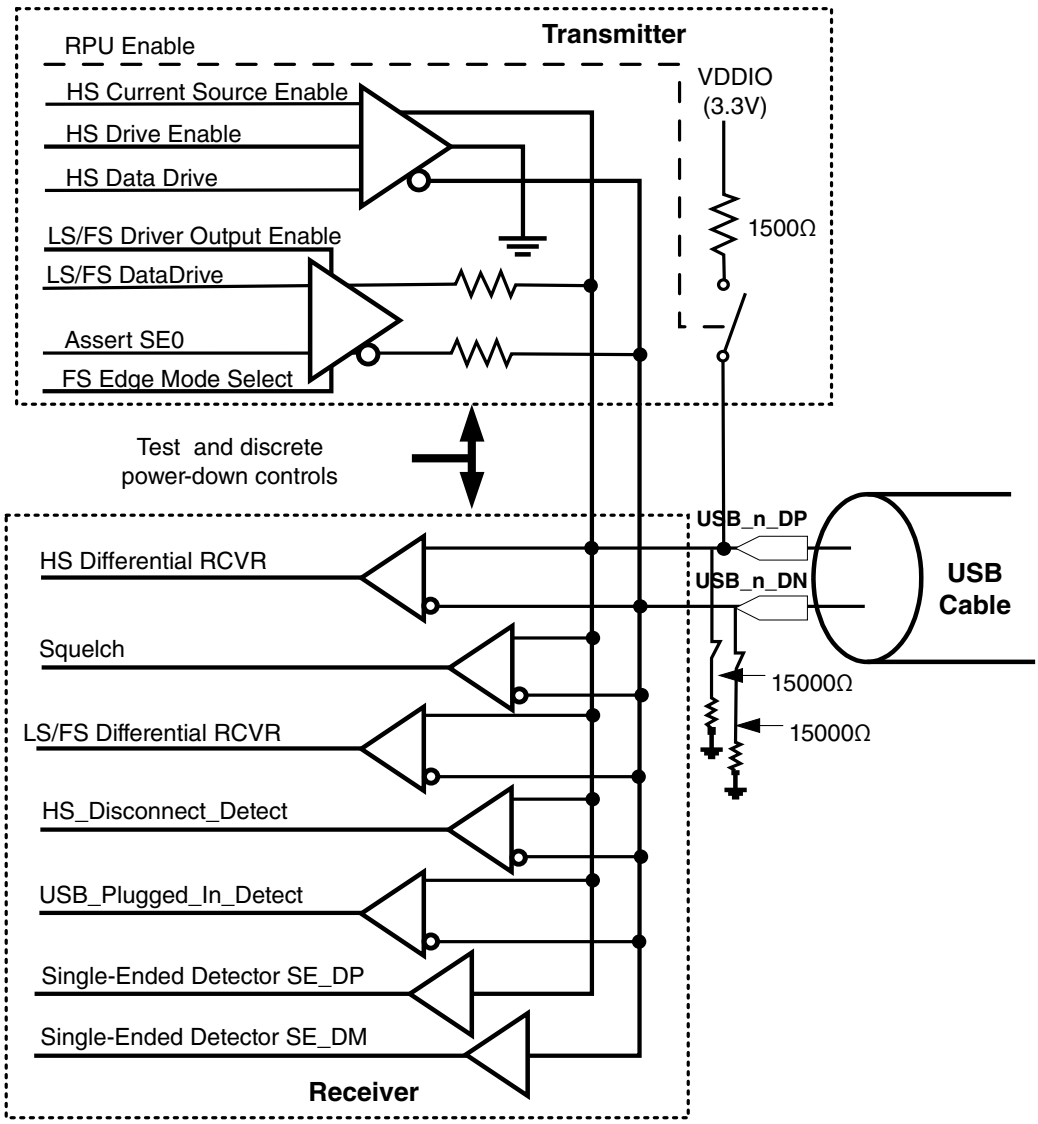


Figure 11-84. USB 2.0 PHY Analog Transceiver Block Diagram

11.4.2.4.1 HS Differential Receiver

The high-speed differential receiver is both a differential analog receiver and threshold comparator. Its output is a one if the differential signal is greater than a 0-V threshold.

Otherwise, its output is 0. Its purpose is to discriminate the ± 400-mV differential voltage resulting from the high-speed drivers current flow into the dual 45Ω terminations found on each pin of the differential pair. The envelope or squelch detector, described below, ensures that the differential signal has sufficient magnitude to be valid. The HS differential receiver tolerates up to 500 mV of common mode offset.

11.4.2.4.2 Squelch Detector

The squelch detector is a differential analog receiver and threshold comparator.

Its output is 1, if the differential magnitude is less than a nominal 100 mV threshold. Otherwise, its output is 0.

Its purpose is to invalidate the HS differential receiver when the incoming signal is simply too low to receive reliably.

11.4.2.4.3 LS/FS Differential Receiver

The low-speed/full-speed differential receiver is both a differential analog receiver and threshold comparator.

The crossover voltage falls between 1.3 V and 2.0 V. Its output is 1, when the USB_*n*_DP line is above the crossover point and the USB_*n*_DN line is below the crossover point. The digital receiver section decodes the receiver data into J or K state according to the speed.

11.4.2.4.4 HS Disconnect Detector

It is a differential analog receiver and threshold comparator. It outputs high when differential magnitude is greater than a nominal 575-mV threshold. Otherwise, it outputs low.

11.4.2.4.5 Single-Ended USB_DP Receiver

The single-ended USB_*n*_DP receiver output is high whenever the USB_*n*_DP input is above its nominal 1.8 V threshold.

11.4.2.4.6 Single-Ended USB_DN Receiver

The single-ended USB_*n*_DN receiver output is high whenever the USB_*n*_DN input is above its nominal 1.8 V threshold.

11.4.2.5 Analog Transmitter

The analog transmitter comprises two differential drivers: one for high-speed signaling and one for full-speed signaling. It also contains the switchable 1.5 KΩ pullup resistor.

See Figure 11-84.

11.4.2.5.1 Switchable High-Speed 45Ω Termination Resistors

High-speed current mode differential signaling requires good 90 Ω differential termination at each end of the USB cable. This results from switching in 45 Ω terminating resistors from each signal line to ground at each end of the cable.

Because each signal is parallel terminated with 45 Ω at each end, each driver sees a 22.5 Ω load. This load impedance is much too low for full-speed signaling levels—hence the need for switchable high-speed terminating resistors. Switchable trimming resistors are provided to tune the actual termination resistance of each device, as shown in Figure 11-85. The HW_USBPHY_TX_TXCAL45DP bit field, for example, allows one of 16 trimming resistor values to be placed in parallel with the 45Ω terminator on the USB_*n*_DP signal.

11.4.2.5.2 Low-Speed/Full-Speed Differential Driver

The low-speed/full-speed differential drivers are essentially low-impedance pulldown devices that are switched in a differential mode for low-speed or full-speed signaling, that is, either one or the other device is turned on to signal the "J" state or the "K" state.

11.4.2.5.3 High-Speed Differential Driver

The high-speed differential driver receives a 17.78 mA current from the constant current source (Iref) and essentially steers it down either the USB_DP signal or the USB_DN signal or alternatively to ground.

This current will produce approximately a 400 mV drop across the 22.5 Ω termination seen by the driver when it is steered onto one of the signal lines. The approximately 17.78 mA current source is referenced back to the integrated voltage-band-gap (Vbg) circuit. The Iref, Ibias, and V to I circuits are shared with the integrated battery charger.

11.4.2.5.4 Switchable 1.5KΩ USB_DP Pullup Resistor

This product contains a switchable 1.5 KΩ pullup resistor on the USB_*n*_DP signal.

This resistor is switched on to indicate to the host/hub controller that a full-speed-capable device is on the USB cable, powered on, and ready. This resistor is switched off at power-on reset so the host does not recognize a USB device until the processor software enables the announcement of a full-speed device.

11.4.2.5.5 Switchable 15KΩ USB_DP Pulldown Resistor

This product contains a switchable 15 KΩ pulldown resistor on both USB_*n*_DP and USB_*n*_DN signals. This is used in host mode to indicate to the device controller that a host is present.

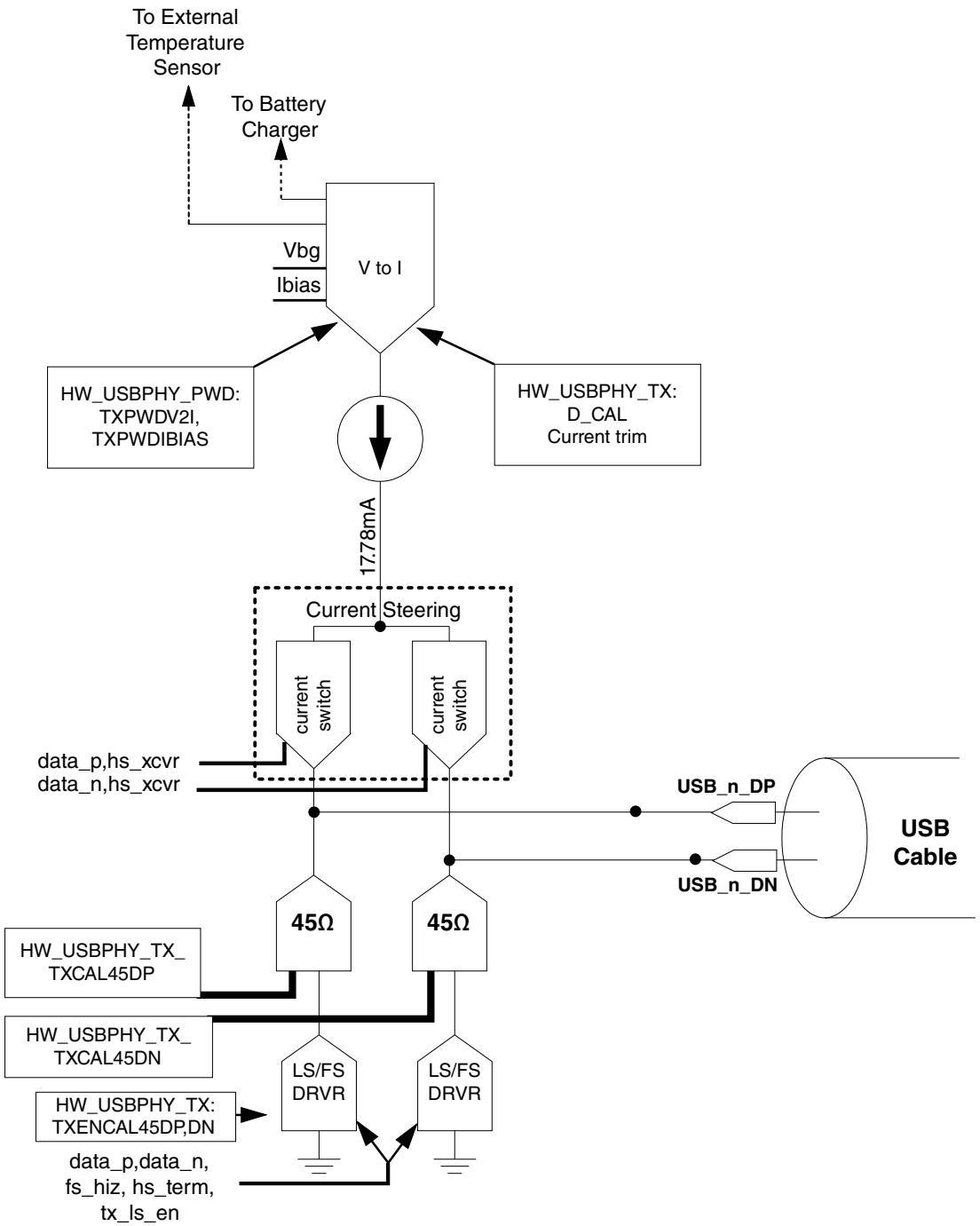


Figure 11-85. USB 2.0 PHY Transmitter Block Diagram

11.4.2.6 Recommended Register Configuration for USB Certification

The register settings in this section are recommended for passing USB certification.

The following settings lower the J/K levels to certifiable limits:

HW_USBPHY_TX_TXCAL45DP = 0x0
HW_USBPHY_TX_TXCAL45DN = 0x0
HW_USBPHY_TX_D_CAL = 0x7

Chapter 12
Timers

12.1 General Purpose Timer (GPT)

12.1.1 Overview

This chapter describes the General Purpose Timer (GPT) module interface. It is also a reference for software driver programming.

The GPT has a 32-bit up-counter. The timer counter value can be captured in a register using an event on an external pin. The capture trigger can be programmed to be a rising or/and falling edge. The GPT can also generate an event on the DO_CMPOUT*n* pins and an interrupt when the timer reaches a programmed value. The GPT has a 12-bit prescaler, which provides a programmable clock frequency derived from multiple clock sources.

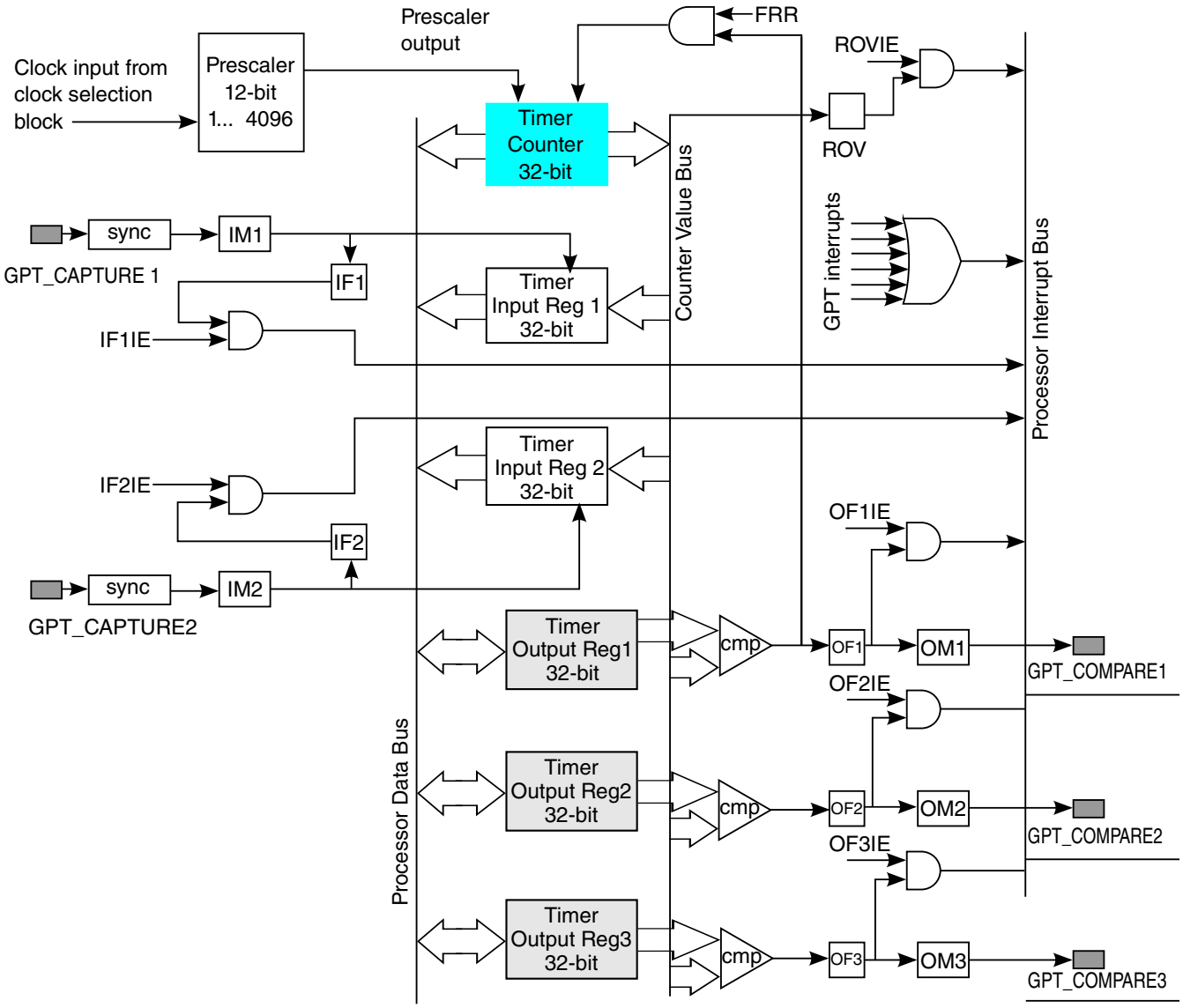


Figure 12-1. GPT Block Diagram

The following figure shows the GPT functional clocking scheme.

12.1.1.1 Features

- One 32-bit up-counter with clock source selection, including external clock.
- Two input capture channels with a programmable trigger edge.
- Three output compare channels with a programmable output mode. A "forced compare" feature is also available.
- Can be programmed to be *active* in low power and debug modes.
- Interrupt generation at capture, compare, and rollover events.
- Restart or free-run modes for counter operations.

12.1.1.2 Modes and Operation

The GPT supports the modes described in the indicated sections:

- Operating Modes
  - Restart Mode
  - Free-Run Mode

12.1.2 External Signals

The GPT follows the IP Bus protocol for interfacing with the processor core. The GPT does not have *any interface signals with any other module inside the chip,* except for the clock and reset inputs (from the clock and reset controller module) and for the interrupt signals *to* the processor interrupt handler. There are functional and clock inputs, and functional output signals going outside the chip boundary.

The following table describes all block signals that connect off-chip.

Table 12-1. GPT External Signals

Signal	Description	Pad	Mode	Direction
GPT1_CAPTURE1	Input pin for a capture event for Input Capture Channel 1.	LCD1_DATA03	ALT1	I
GPT1_CAPTURE2	Input pin for a capture event for Input Capture Channel 2.	LCD1_DATA04	ALT1	I
GPT1_CLK	Input pin for an external clock that the counter can be operated at.	LCD1_DATA02	ALT1	I
GPT1_COMPARE1	Output pin that indicates a "compare event" occurrence in Output Compare Channel 1.	LCD1_RESET	ALT1	O
GPT1_COMPARE2	Output pin that indicates a "compare event" occurrence in Output Compare Channel 2.	LCD1_DATA00	ALT1	O
GPT1_COMPARE3	Output pin that indicates a "compare event" occurrence in Output Compare Channel 3.	LCD1_DATA01	ALT1	O
GPT2_CAPTURE1	Input pin for a capture event for Input Capture Channel 1.	ENET1_CRS	ALT3	I
GPT2_CAPTURE2	Input pin for a capture event for Input Capture Channel 2.	ENET1_COL	ALT3	I
GPT2_CLK	Input pin for an external clock that the counter can be operated at.	ENET1_RX_CLK	ALT3	I
GPT2_COMPARE1	Output pin that indicates a "compare event" occurrence in Output Compare Channel 1.	ENET1_TX_CTL	ALT3	O

Table continues on the next page...

Table 12-1. GPT External Signals (continued)

Signal	Description	Pad	Mode	Direction
GPT2_COMPARE2	Output pin that indicates a "compare event" occurrence in Output Compare Channel 2.	ENET1_TXC	ALT3	O
GPT2_COMPARE3	Output pin that indicates a "compare event" occurrence in Output Compare Channel 3.	ENET1_TX_CLK	ALT3	O
GPT3_CAPTURE1	Input pin for a capture event for Input Capture Channel 1.	SD3_CMD	ALT4	I
GPT3_CAPTURE2	Input pin for a capture event for Input Capture Channel 2.	SD3_DATA0	ALT4	I
GPT3_CLK	Input pin for an external clock that the counter can be operated at.	SD3_CLK	ALT4	I
GPT3_COMPARE1	Output pin that indicates a "compare event" occurrence in Output Compare Channel 1.	SD3_DATA1	ALT4	O
GPT3_COMPARE2	Output pin that indicates a "compare event" occurrence in Output Compare Channel 2.	SD3_DATA2	ALT4	O
GPT3_COMPARE3	Output pin that indicates a "compare event" occurrence in Output Compare Channel 3.	SD3_DATA3	ALT4	O
GPT4_CAPTURE1	Input pin for a capture event for Input Capture Channel 1.	SD2_CMD	ALT3	I
GPT4_CAPTURE2	Input pin for a capture event for Input Capture Channel 2.	SD2_DATA0	ALT3	I
GPT4_CLK	Input pin for an external clock that the counter can be operated at.	SD2_CLK	ALT3	I
GPT4_COMPARE1	Output pin that indicates a "compare event" occurrence in Output Compare Channel 1.	SD2_DATA1	ALT3	O
GPT4_COMPARE2	Output pin that indicates a "compare event" occurrence in Output Compare Channel 2.	SD2_DATA2	ALT3	O
GPT4_COMPARE3	Output pin that indicates a "compare event" occurrence in Output Compare Channel 3.	SD2_DATA3	ALT3	O

There are six signals (three input, three output) in the GPT module that *can be* connected to the chip pads.

12.1.2.1 External Clock Input

The GPT counter can be operated using an external clock from outside the device, and this is the input pin used for that purpose.

The external clock input GPT_CLK is treated as asynchronous to the peripheral clock. To ensure proper operations of GPT, the external clock input frequency should be less than 1/4 of frequency of the peripheral clock. Hysteresis characteristics on this pad will be required because this is a clock input.

12.1.2.2 Input Capture Trigger Signals

The GPT counter value can be stored in a register, triggered by an event from *outside the device*.

A positive or/and negative edge on these signals GPT_CAPTURE1 , GPT_CAPTURE2 can trigger this capture event. These signals are treated as asynchronous to the peripheral clock. Only those transitions which occur *at least a single clock cycle* (the clock selected to run the counter) *after the previous recorded transition* are guaranteed to trigger a capture event.

12.1.2.3 Output Compare Signals

The output compare signals: GPT_COMPARE1, GPT_COMPARE2, GPT_COMPARE3, indicate that output compare events have gone through a specified transition.

12.1.3 Signals

12.1.3.1 External Clock Input

The GPT counter can be operated using an external clock from outside the device, and this is the input pin used for that purpose.

The external clock input GPT_CLK is treated as asynchronous to the peripheral clock. To ensure proper operations of GPT, the external clock input frequency should be less than 1/4 of frequency of the peripheral clock. Hysteresis characteristics on this pad will be required because this is a clock input.

12.1.3.2 Input Capture Trigger Signals

The GPT counter value can be stored in a register, triggered by an event from *outside the device*.

A positive or/and negative edge on these signals GPT_CAPTURE1 , GPT_CAPTURE2 can trigger this capture event. These signals are treated as asynchronous to the peripheral clock. Only those transitions which occur *at least a single clock cycle* (the clock selected to run the counter) *after the previous recorded transition* are guaranteed to trigger a capture event.

12.1.3.3 Output Compare Signals

The output compare signals: GPT_COMPARE1, GPT_COMPARE2, GPT_COMPARE3, indicate that output compare events have gone through a specified transition.

12.1.4 Clocks

The clock that is input to the prescaler can be selected from 4 clock sources:

The following table describes the clock sources for GPT. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 12-2. GPT Clocks

Clock name	Clock Root	Description
ipg_clk	GPT_CLK_ROOT	Peripheral clock
ipg_clk_32k	CKIL_SYNC_CLK_ROOT	Low-frequency reference clock (32 kHz)
ipg_clk_highfreq	GPT_CLK_ROOT	High-frequency reference clock
ipg_clk_s	GPT_CLK_ROOT	Peripheral access clock

- High-Frequency Clock (ipg_clk_highfreq)

  Provided by the Clock Controller Module (CCM), the High Frequency Clock is intended to be ON in Normal Power mode when the Peripheral Clock is turned OFF, thereby enabling the GPT to be operated using the High Frequency Clock *in Normal Power mode*. The CCM is expected to provide this clock *after* synchronizing it to the System Bus Clock in Normal functional mode; the CCM is also expected to switch to the *unsynchronized* version of the High Frequency Clock in a Low Power mode.

- Low-Reference Clock (ipg_clk_32k)

  This 32 kHz Low Reference Clock (provided by the CCM) is intended to be ON in Low Power mode when the Peripheral Clock is turned OFF, thereby enabling the GPT to be operated using the Low Reference Clock in Low Power mode. The CCM

is expected to provide the Low Reference Clock *after* synchronizing it to the System Bus Clock in Normal functional mode; the CCM is also expected to switch to the *unsynchronized* version of the Low Reference Clock in a Low Power mode.

- Peripheral Clock (ipg_clk)

  If the Peripheral Clock or the External Clock is selected(CLKSRC=001 or 011) as Clock Source, then the Peripheral Clock will be ON in normal GPT operations. In Low Power modes, if the GPT is programmed to be disabled (STOPEN or WAITEN or DOZEN=0), then the Peripheral Clock can be switched OFF.

- External Clock

  The External Clock comes from *outside the device* and can be selected to run the GPT counter. The External Clock is treated as *asynchronous to the Peripheral Clock,* and is synchronized to the Peripheral Clock, *inside* the module. Therefore, the External Clock frequency is limited to < 1/4 frequency of the Peripheral Clock, for proper GPT operations. Note that in Low Power modes, *if* the Peripheral Clock is not available, then the External Clock *cannot be used* to run the counter.

- Crystal Oscillator Clock

  This 24 MHz Crystal Oscillator Clock (provided by the CCM) is intended to be used against frequency change of Peripheral Clock changes to provide a more accurate timer clock for operation system. The CCM is expected to provide the 24 MHz Crystal Oscillator Clock *without* synchronizing it to the System Bus Clock in Normal functional mode. Synchronization is done in GPT module. Before synchronization, the 24 MHz Crystal Oscillator Clock is divided by a 24 MHz clock prescaler, to make sure the clock frequency less than half of System Bus Clock .

The clock input source is configured using the clock source field (CLKSRC, in the GPT_CR control register). The clock input to the prescaler can be disabled by programming the CLKSRC bits (of the GPT_CR control register) to 000. **The CLKSRC field value should be changed** *only after disabling the GPT* (by setting the EN bit in the GPT_CR to 0).

The PRESCALER field selects the divide ratio of the input clock that drives the main counter. The prescaler can divide the input clock by a value (from 1 to 4096) and can be changed *at any time*. A change in the value of the PRESCALER field *immediately affects* the output clock frequency.

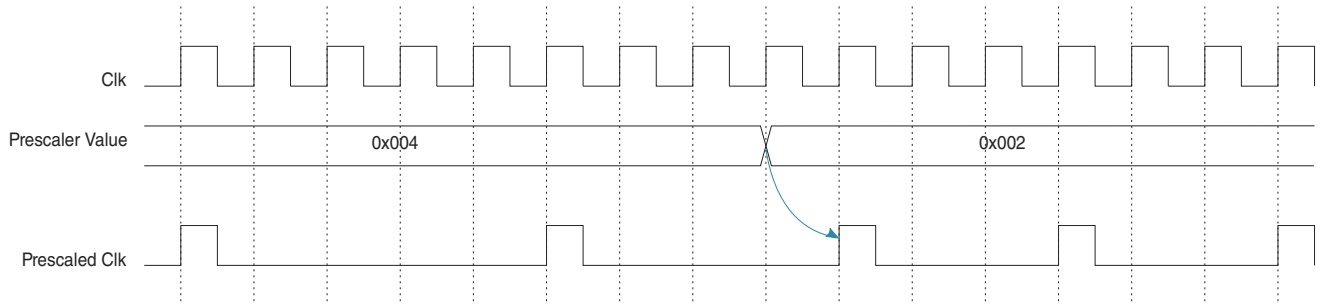


Figure 12-2. Prescaler Value Change Timing Diagram

12.1.5 Functional Description

This section provides a complete functional description of the GPT.

12.1.5.1 Operating Modes

The GPT counter can be programmed to work in either of two modes: Restart mode or Free-Run mode.

12.1.5.1.1 Restart Mode

In Restart mode (selectable through the GPT Control Register GPT_CR), when the counter reaches the compared value, the counter resets and starts again from 0x00000000. The Restart feature is associated only with Compare Channel 1.

Any write access to the Compare register of Channel 1 will reset the GPT counter. This is done to avoid possibly missing a compare event when compare value is changed from a higher value to lower value while counting is proceeding.

For the other two compare channels, when the compare event occurs the counter is *not reset*.

12.1.5.1.2 Free-Run Mode

In Free-Run mode, when compare events occur for all 3 channels, the counter is *not reset*; instead the counter continues to count until 0xffffffff, and then rolls over (to 0x00000000).

12.1.5.2 Operation

The General Purpose Timer (GPT) has a single counter (GPT_CNT) that is a 32-bit free-running *up-counter,* which starts counting *after it is enabled by software* (EN=1).

The counter's clock source is the output of the prescaler labelled "Prescaler output" in Figure 12-1.

- If the GPT timer is disabled (EN=0), then the Main Counter *and* Prescaler Counter freeze their current count values. The ENMOD bit determines the value of the GPT counter when the EN bit is set and the Counter is enabled again.
  - If the ENMOD bit is set (=1), then the Main Counter and Prescaler Counter values are reset to 0, when GPT is enabled (EN=1).
  - If ENMOD bit is programmed to 0, then the Main Counter and Prescaler Counter restart counting from their frozen values, when GPT is enabled again (EN=1).
- If GPT is programmed to be disabled in a low power mode (STOP/WAIT), then the Main Counter and Prescaler Counter freeze at their current count values *when* GPT enters low power mode. When GPT exits a low power mode, the Main Counter and Prescaler Counter start counting from their frozen values *regardless* of the ENMOD bit value. Note that the GPT_CNT can be read *at any time* by the processor, and that *both* Input Capture Channels use the *same* counter (GPT_CNT).
- A hardware reset resets all the GPT registers to their respective reset values. All registers except the Output Compare Registers (OCR1, OCR2, OCR3) obtain a value of 0x0. The Compare registers are reset to 0xffffffff.
- The software reset (SWR bit in the GPT_CR control register) resets *all* of the register bits *except* the EN, ENMOD, STOPEN, WAITEN, and DBGEN bits. The state of these bits is not affected by a software reset. Note that a software reset can be given *while the GPT is disabled*.

12.1.5.2.1 Input Capture

There are two Input Capture Channels, and each Input Capture Channel has a dedicated capture pin, capture register and input edge detection/selection logic. Each input capture function has an associated status flag, and can cause the processor to make an interrupt service request.

When a selected edge transition occurs on an Input Capture pin, the contents of the GPT_CNT is captured on the corresponding capture register and the appropriate interrupt status flag is set. An interrupt request can be generated when the transition is detected *if* its corresponding enable bit is set (in the Interrupt Register). The capture can be programmed to occur on the input pin's rising edge, falling edge, on both rising and falling edges, or the capture can be disabled. The events are synchronized with the clock that was selected to run the counter. Only those transitions that occur at least one clock

cycle (clock selected to run the counter) *after* the previous recorded transition will be guaranteed to trigger a capture event. There can be up to one clock cycle of uncertainty in the latching of the input transition. The Input Capture registers can be read *at any time* without affecting their values.

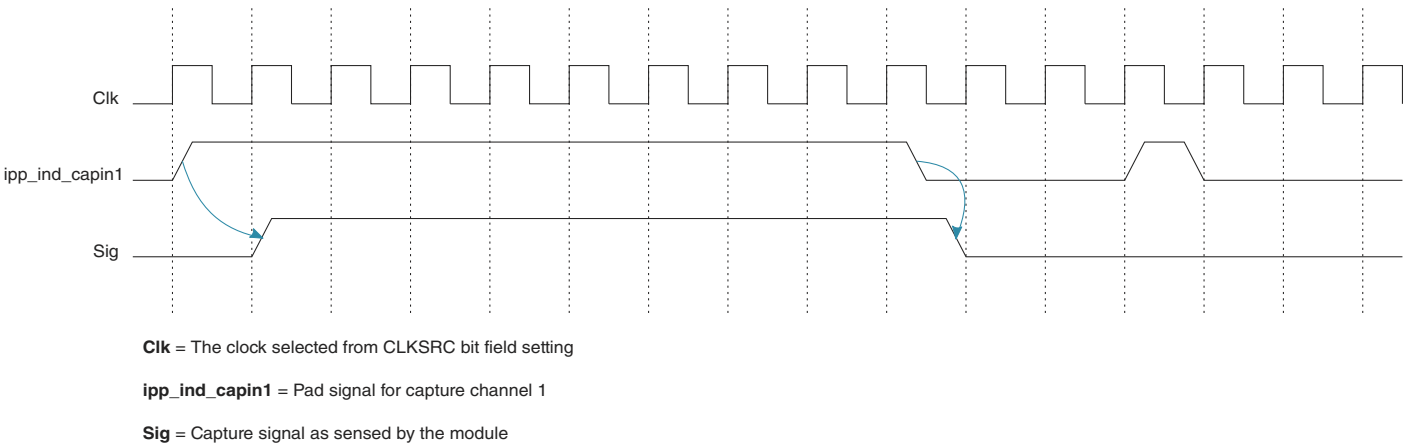


Figure 12-3. Input Capture Event Timing

12.1.5.2.2 Output Compare

The three Output Compare Channels *use the same counter* (GPT_CNT) as the Input Capture Channels. When the programmed content of an Output Compare register matches the value in GPT_CNT, an output compare status flag is set and an interrupt is generated (if the corresponding bit is set in the interrupt register). Consequently, the Output Compare timer pin will be set, cleared, toggled, not affected at all or provide an active-low pulse for one input clock period (subject to the restriction on the maximum frequency allowed on the pad) according to the mode bits (that were programmed).

There is also a "forced-compare" feature that allows the software to generate a compare event when required, *without the condition of the counter value that is equal to the compare value*. The action taken as a result of a forced compare is the same as when an output compare match occurs, *except that the status flags are not set and no interrupt can be generated*. Forced channels take programmed action immediately after the write to the force-compare bits. These bits are self-negating and always read as zeros.

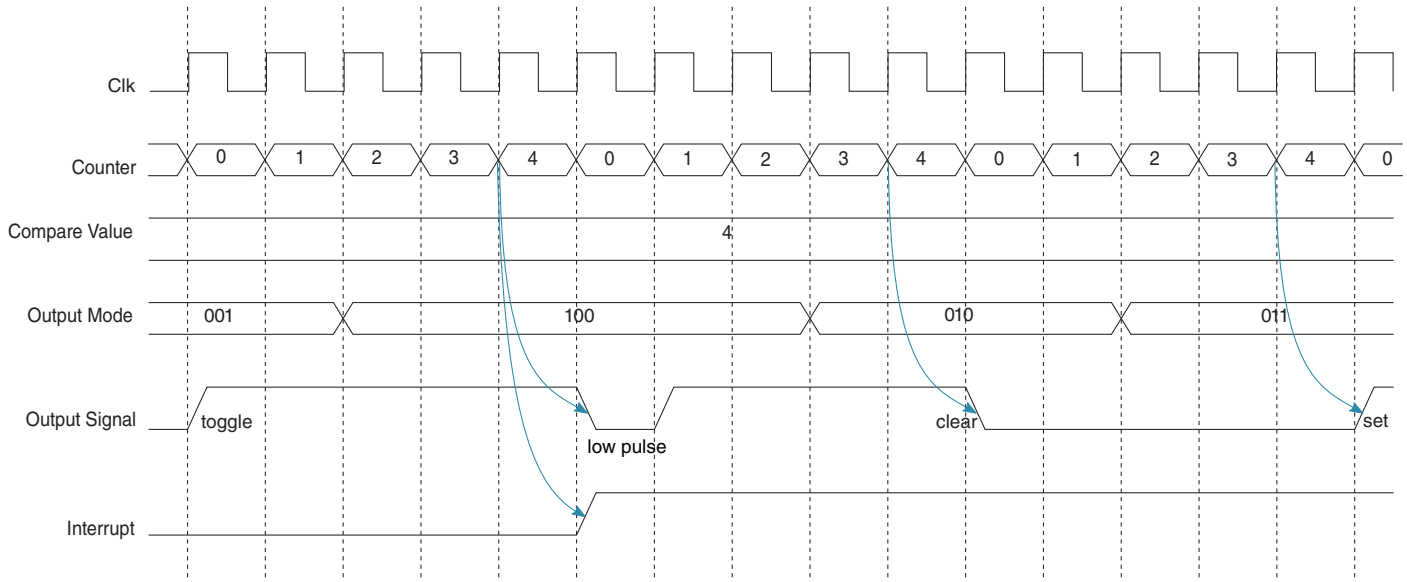


Figure 12-4. Output Compare and Interrupt Timing

12.1.5.2.3 Interrupts

There are 6 different interrupts that are generated by the GPT. If the selected clock for running the counter is available, then *all interrupts can be generated in Low Power and Debug modes*.

- Rollover Interrupt

  The Rollover Interrupt is generated when the GPT counter reaches 0xffffffff, then resets to 0x00000000 and continues counting. The Rollover Interrupt is enabled by the ROVIE bit in the GPT_IR register; the associated status bit is the ROV bit in the GPT_SR register.

- Input Capture Interrupt 1, 2

  After a capture event occurs, the associated Input Capture Channel generates an interrupt. The "capture event" interrupts are enabled by the IF2IE and IF1IE bits (in the GPT_IR register); the associated status bits are IF2 and IF1 (in the GPT_SR register). The capture of the counter value because of a capture event is *not affected by a pending capture interrupt*. The Capture register is updated with a new counter value when a capture event occurs, regardless of whether that Capture Channels' interrupt has been serviced or not.

- Output Compare Interrupt 1, 2, 3

After a compare event occurs, the associated Output Compare Channel generates an interrupt. The "compare event" interrupts are enabled by the OF3IE, OF2IE, and OF1IE bits (in the GPT_IR register); the associated status bits are OF3, OF2, and OF1 (in the GPT_SR register). A "forced compare" does not generate an interrupt.

A *cumulative* interrupt line is also present, which is asserted whenever any of the above interrupts are posted. The cumulative interrupt line has *no* associated enables or status bits.

12.1.5.2.4 Low Power Mode Behavior

In Low Power modes, if the clock from the selected clock source is available (except for the External Clock, which can be used *only if* the Peripheral Clock is available), the counter will continue to run depending on whether the control bit for that mode is set. If the clock is not present or if the corresponding low power bit in the GPT_CR control register is 0, the Main Counter and the Prescaler Counter freeze at their current values and resume counting (from their frozen values) when the Low Power mode is exited.

12.1.5.2.5 Debug Mode Behavior

In Debug mode, the modules in the device have the option of continuing to run or be halted.

- If the DBGEN bit is set, then the GPT timer will continue to run in Debug mode.
- If the DBGEN bit is not set (in the GPT_CR control register), then the GPT timer is halted.

12.1.6 Initialization/ Application Information

12.1.6.1 Selecting the Clock Source

The CLKSRC field in the GPT_CR register selects the clock source. The CLKSRC field value should be changed only after disabling the GPT (EN=0).

The software sequence to be followed while changing clock source is:

1. Disable GPT by setting EN=0 in GPT_CR register.
2. Disable GPT interrupt register (GPT_IR).
3. Configure Output Mode to unconnected/ disconnected—Write zeros in OM3, OM2, and OM1 in GPT_CR
4. Disable Input Capture Modes—Write zeros in IM1 and IM2 in GPT_CR

5. Change clock source CLKSRC to the desired value in GPT_CR register.
6. Assert the SWR bit in GPT_CR register.
7. Clear GPT status register (GPT_SR) (i.e., w1c).
8. Set ENMOD=1 in GPT_CR register, to bring GPT counter to 0x00000000.
9. Enable GPT (EN=1) in GPT_CR register.
10. Enable GPT interrupt register (GPT_IR).

12.1.7 GPT Memory Map/Register Definition

The GPT has 10 user-accessible 32-bit registers, which are used to configure, operate, and monitor the state of the GPT.

An IP bus write access to the GPT Control Register (GPT_CR) and the GPT Output Compare Register1 (GPT_OCR1) results in *one cycle of wait state*, while other valid IP bus accesses incur 0 wait states.

Irrespective of the Response Select signal value, a Write access to the GPT Status Registers (Read-only registers GPT_ICR1, GPT_ICR2, GPT_CNT) will generate a bus exception.

- If the Response Select signal is driven Low, then the Read/Write access to the *unimplemented* address space of GPT (*ips_addr* is greater than or equal to $BASE + $028) will generate a bus exception.
- If the Response Select is driven High, then the Read/Write access to the unimplemented address space of GPT will *not* generate any error response (like a bus exception).

GPT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
302D_0000	GPT Control Register (GPT1_CR)	32	R/W	0000_0000h	12.1.7.1/ 3392
302D_0004	GPT Prescaler Register (GPT1_PR)	32	R/W	0000_0000h	12.1.7.2/ 3396
302D_0008	GPT Status Register (GPT1_SR)	32	R/W	0000_0000h	12.1.7.3/ 3397
302D_000C	GPT Interrupt Register (GPT1_IR)	32	R/W	0000_0000h	12.1.7.4/ 3398
302D_0010	GPT Output Compare Register 1 (GPT1_OCR1)	32	R/W	FFFF_FFFFh	12.1.7.5/ 3399

Table continues on the next page...

GPT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
302D_0014	GPT Output Compare Register 2 (GPT1_OCR2)	32	R/W	FFFF_FFFFh	12.1.7.6/ 3400
302D_0018	GPT Output Compare Register 3 (GPT1_OCR3)	32	R/W	FFFF_FFFFh	12.1.7.7/ 3400
302D_001C	GPT Input Capture Register 1 (GPT1_ICR1)	32	R	0000_0000h	12.1.7.8/ 3401
302D_0020	GPT Input Capture Register 2 (GPT1_ICR2)	32	R	0000_0000h	12.1.7.9/ 3401
302D_0024	GPT Counter Register (GPT1_CNT)	32	R	0000_0000h	12.1.7.10/ 3402
302E_0000	GPT Control Register (GPT2_CR)	32	R/W	0000_0000h	12.1.7.1/ 3392
302E_0004	GPT Prescaler Register (GPT2_PR)	32	R/W	0000_0000h	12.1.7.2/ 3396
302E_0008	GPT Status Register (GPT2_SR)	32	R/W	0000_0000h	12.1.7.3/ 3397
302E_000C	GPT Interrupt Register (GPT2_IR)	32	R/W	0000_0000h	12.1.7.4/ 3398
302E_0010	GPT Output Compare Register 1 (GPT2_OCR1)	32	R/W	FFFF_FFFFh	12.1.7.5/ 3399
302E_0014	GPT Output Compare Register 2 (GPT2_OCR2)	32	R/W	FFFF_FFFFh	12.1.7.6/ 3400
302E_0018	GPT Output Compare Register 3 (GPT2_OCR3)	32	R/W	FFFF_FFFFh	12.1.7.7/ 3400
302E_001C	GPT Input Capture Register 1 (GPT2_ICR1)	32	R	0000_0000h	12.1.7.8/ 3401
302E_0020	GPT Input Capture Register 2 (GPT2_ICR2)	32	R	0000_0000h	12.1.7.9/ 3401
302E_0024	GPT Counter Register (GPT2_CNT)	32	R	0000_0000h	12.1.7.10/ 3402
302F_0000	GPT Control Register (GPT3_CR)	32	R/W	0000_0000h	12.1.7.1/ 3392
302F_0004	GPT Prescaler Register (GPT3_PR)	32	R/W	0000_0000h	12.1.7.2/ 3396
302F_0008	GPT Status Register (GPT3_SR)	32	R/W	0000_0000h	12.1.7.3/ 3397
302F_000C	GPT Interrupt Register (GPT3_IR)	32	R/W	0000_0000h	12.1.7.4/ 3398
302F_0010	GPT Output Compare Register 1 (GPT3_OCR1)	32	R/W	FFFF_FFFFh	12.1.7.5/ 3399
302F_0014	GPT Output Compare Register 2 (GPT3_OCR2)	32	R/W	FFFF_FFFFh	12.1.7.6/ 3400
302F_0018	GPT Output Compare Register 3 (GPT3_OCR3)	32	R/W	FFFF_FFFFh	12.1.7.7/ 3400

Table continues on the next page...

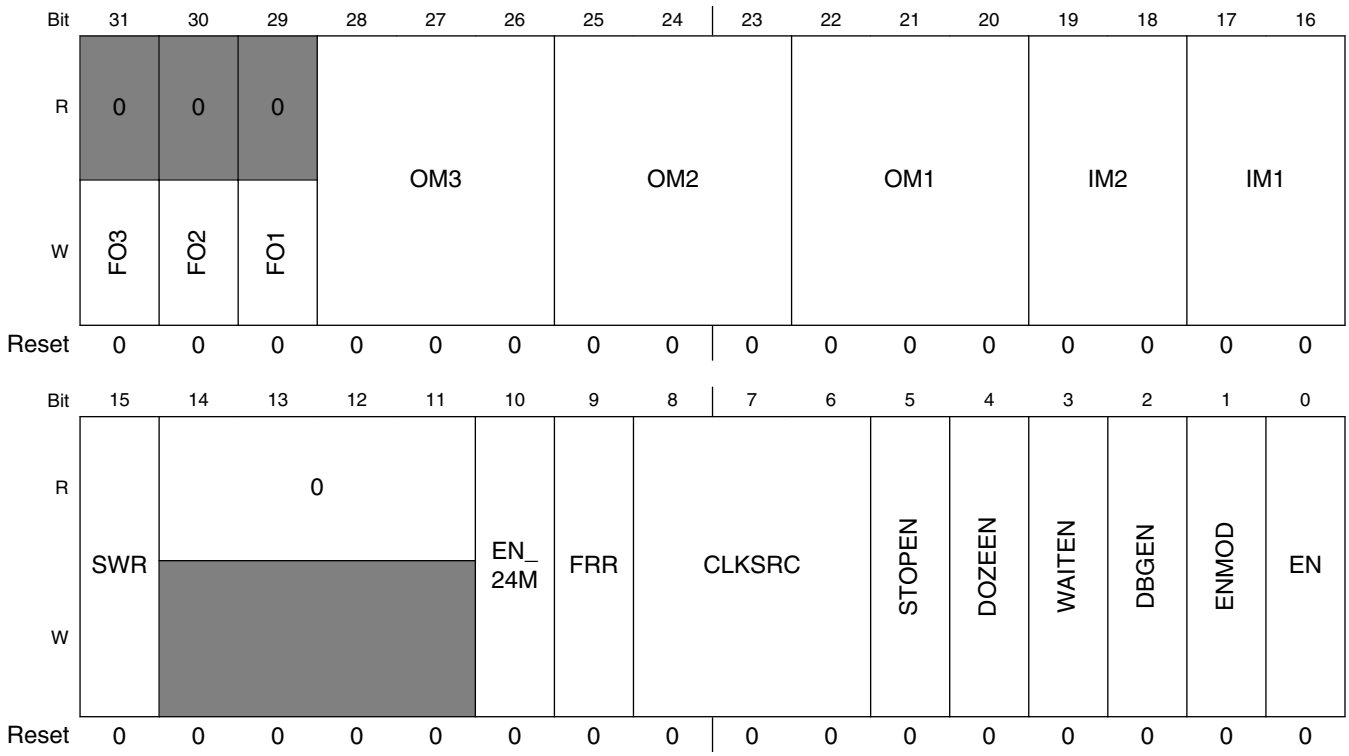
GPT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
302F_001C	GPT Input Capture Register 1 (GPT3_ICR1)	32	R	0000_0000h	12.1.7.8/ 3401
302F_0020	GPT Input Capture Register 2 (GPT3_ICR2)	32	R	0000_0000h	12.1.7.9/ 3401
302F_0024	GPT Counter Register (GPT3_CNT)	32	R	0000_0000h	12.1.7.10/ 3402
3030_0000	GPT Control Register (GPT4_CR)	32	R/W	0000_0000h	12.1.7.1/ 3392
3030_0004	GPT Prescaler Register (GPT4_PR)	32	R/W	0000_0000h	12.1.7.2/ 3396
3030_0008	GPT Status Register (GPT4_SR)	32	R/W	0000_0000h	12.1.7.3/ 3397
3030_000C	GPT Interrupt Register (GPT4_IR)	32	R/W	0000_0000h	12.1.7.4/ 3398
3030_0010	GPT Output Compare Register 1 (GPT4_OCR1)	32	R/W	FFFF_FFFFh	12.1.7.5/ 3399
3030_0014	GPT Output Compare Register 2 (GPT4_OCR2)	32	R/W	FFFF_FFFFh	12.1.7.6/ 3400
3030_0018	GPT Output Compare Register 3 (GPT4_OCR3)	32	R/W	FFFF_FFFFh	12.1.7.7/ 3400
3030_001C	GPT Input Capture Register 1 (GPT4_ICR1)	32	R	0000_0000h	12.1.7.8/ 3401
3030_0020	GPT Input Capture Register 2 (GPT4_ICR2)	32	R	0000_0000h	12.1.7.9/ 3401
3030_0024	GPT Counter Register (GPT4_CNT)	32	R	0000_0000h	12.1.7.10/ 3402

12.1.7.1 GPT Control Register (GPTx_CR)

The GPT Control Register (GPT_CR) is used to program and configure GPT operations. An IP Bus Write to the GPT Control Register occurs after one cycle of wait state, while an IP Bus Read occurs after 0 wait states.

Address: Base address + 0h offset



GPTx_CR field descriptions

Field	Description
31 FO3	FO3 Force Output Compare Channel 3 FO2 Force Output Compare Channel 2 FO1 Force Output Compare Channel 1 The FO*n* bit causes the pin action *programmed* for the timer Output Compare *n* pin (according to the OM*n* bits in this register). • The OF*n* flag (OF3, OF2, OF1) in the status register is **not affected**. • This bit is self-negating and always read as zero. 0 Writing a 0 has no effect. 1 Causes the programmed pin action on the timer Output Compare *n* pin; the OF*n* flag is not set.
30 FO2	See F03

Table continues on the next page...

GPTx_CR field descriptions (continued)

Field	Description
29 FO1	See F03
28–26 OM3	OM3 (bits 28-26) controls the Output Compare Channel 3 operating mode. OM2 (bits 25-23) controls the Output Compare Channel 2 operating mode. OM1 (bits 22-20) controls the Output Compare Channel 1 operating mode. The OM*n* bits specify the response that a compare event will generate on the output pin of Output Compare Channel *n*. • The toggle, clear, and set options cause a change on the output pin *only* if a compare event occurs. • When OM*n* is programmed as 1xx (active low pulse), the output pin is set to one immediately on the next input clock; a low pulse (that is an input clock in width) occurs when there is a compare event. Note that here, "input clock" refers to the clock selected by the CLKSRC bits of the GPT Control Register. 000 Output disconnected. No response on pin. 001 Toggle output pin 010 Clear output pin 011 Set output pin 1xx Generate an active low pulse (that is one input clock wide) on the output pin.
25–23 OM2	See OM3
22–20 OM1	See OM3
19–18 IM2	IM2 (bits 19-18, Input Capture Channel 2 operating mode) IM1 (bits 17-16, Input Capture Channel 1 operating mode) The IM*n* bit field determines the transition on the input pin (for Input capture channel *n*), which will trigger a capture event. 00 capture disabled 01 capture on rising edge only 10 capture on falling edge only 11 capture on both edges
17–16 IM1	See IM2
15 SWR	Software reset. This is the software reset of the GPT module. It is a self-clearing bit. • The SWR bit is set when the module is in reset state. • The SWR bit is cleared when the reset procedure finishes. • Setting the SWR bit resets **all of the registers** to their default reset values, except for the CLKSRC, EN, ENMOD, STOPEN, WAITEN, and DBGEN bits in the GPT Control Register (this control register). 0 GPT is not in reset state 1 GPT is in reset state
14–11 Reserved	This read-only field is reserved and always has the value 0.
10 EN_24M	Enable 24 MHz clock input from crystal.

Table continues on the next page...

GPTx_CR field descriptions (continued)

Field	Description
	• A hardware reset resets the EN_24M bit. • A software reset *does not affect* the EN_24M bit. 0 24M clock disabled 1 24M clock enabled
9 FRR	Free-Run or Restart mode. The FFR bit determines the behavior of the GPT when a compare event in channel 1 occurs. • In Restart mode, after a compare event, the counter resets to 0x00000000 and resumes counting (after the occurrence of a compare event). • In Free-Run mode, after a compare event, the counter continues counting until 0xFFFFFFFF and then rolls over to 0. 0 Restart mode 1 Free-Run mode
8–6 CLKSRC	Clock Source select. The CLKSRC bits select which clock will go to the prescaler (and subsequently be used to run the GPT counter). • The CLKSRC bit field value should only be changed after disabling the GPT by clearing the EN bit in this register (GPT_CR). • A software reset does not affect the CLKSRC bit. 000 No clock 001 Peripheral Clock 010 High Frequency Reference Clock 011 External Clock (CLKIN) 100 Low Frequency Reference Clock 101 Crystal oscillator as Reference Clock others Reserved
5 STOPEN	GPT Stop Mode enable. The STOPEN read/write control bit enables GPT operation *during Stop mode*. • A hardware reset resets the STOPEN bit. • A software reset *does not affect* the STOPEN bit. 0 GPT is disabled in Stop mode. 1 GPT is enabled in Stop mode.
4 DOZEEN	GPT Doze Mode Enable. • A hardware reset resets the DOZEEN bit. • A software reset *does not affect* the DOZEEN bit. 0 GPT is disabled in doze mode. 1 GPT is enabled in doze mode.
3 WAITEN	GPT Wait Mode enable. The WAITEN read/write control bit enables GPT operation *during Wait mode*. • A hardware reset resets the WAITEN bit. • A software reset *does not affect* the WAITEN bit. 0 GPT is disabled in wait mode. 1 GPT is enabled in wait mode.

Table continues on the next page...

GPTx_CR field descriptions (continued)

Field	Description
2 DBGEN	GPT debug mode enable. The DBGEN read/write control bit enables GPT operation *during Debug mode*. • A hardware reset resets the DBGEN bit. • A software reset *does not affect* the DBGEN bit. 0 GPT is disabled in debug mode. 1 GPT is enabled in debug mode.
1 ENMOD	GPT Enable mode. When the GPT is disabled (EN=0), then both the Main Counter and Prescaler Counter *freeze their current count values*. The ENMOD bit determines the value of the GPT counter when Counter is enabled again (if the EN bit is set). • If the ENMOD bit is 1, then the Main Counter and Prescaler Counter values are reset to 0 after GPT is enabled (EN=1). • If the ENMOD bit is 0, then the Main Counter and Prescaler Counter restart counting *from their frozen values* after GPT is enabled (EN=1). • If GPT is programmed to be disabled in a low power mode (STOP/WAIT), then the Main Counter and Prescaler Counter *freeze at their current count values* when the GPT enters low power mode. • When GPT exits low power mode, the Main Counter and Prescaler Counter start counting from their frozen values, regardless of the ENMOD bit value. • Setting the SWR bit will clear the Main Counter and Prescaler Counter values, regardless of the value of EN or ENMOD bits. • A hardware reset resets the ENMOD bit. • A software reset *does not affect* the ENMOD bit. 0 GPT counter will retain its value when it is disabled. 1 GPT counter value is reset to 0 when it is disabled.
0 EN	GPT Enable. The EN bit is the GPT module enable bit. **Before setting the EN bit**, we recommend that *all registers be properly programmed*. • A hardware reset resets the EN bit. • A software reset *does not affect* the EN bit. 0 GPT is disabled. 1 GPT is enabled.

12.1.7.2 GPT Prescaler Register (GPTx_PR)

The GPT Prescaler Register (GPT_PR) contains bits that determine the *divide value* of the clock that runs the counter.

Address: Base address + 4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRESCALER24M				PRESCALER											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPTx_PR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–12 PRESCALER24M	Prescaler bits. 24M crystal clock is divided by [PRESCALER24M + 1] before selected by the CLKSRC field. If 24M crystal clock is not selected, this feild takes no effect. 0x0 Divide by 1 0x1 Divide by 2 ... ... 0xF Divide by 16
PRESCALER	Prescaler bits. The clock selected by the CLKSRC field is divided by [PRESCALER + 1], and then used to run the counter. • A change in the value of the PRESCALER bits cause the Prescaler counter to reset and a new count period to start immediately. • See Figure 12-2 for the timing diagram. 0x000 Divide by 1 0x001 Divide by 2 ... ... 0xFFF Divide by 4096

12.1.7.3 GPT Status Register (GPTx_SR)

The GPT Status Register (GPT_SR) contains bits that indicate that a counter has rolled over, and if any event has occurred on the Input Capture and Output Compare channels. The bits are cleared by writing a 1 to them.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0										ROV	IF2	IF1	OF3	OF2	OF1
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPTx_SR field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value 0.
5 ROV	Rollover Flag. The ROV bit indicates that the counter has reached its *maximum possible value* and *rolled over* to 0 (from which the counter continues counting). The ROV bit is only set if the counter has reached 0xFFFFFFFF in both Restart and Free-Run modes. 0 Rollover has not occurred. 1 Rollover has occurred.
4 IF2	IF2 Input capture 2 Flag IF1 Input capture 1 Flag The IF*n* bit indicates that a capture event has occurred on Input Capture channel *n*. 0 Capture event has not occurred. 1 Capture event has occurred.
3 IF1	See IF2
2 OF3	OF3 Output Compare 3 Flag OF2 Output Compare 2 Flag OF1 Output Compare 1 Flag The OF*n* bit indicates that a compare event has occurred on Output Compare channel *n*. 0 Compare event has not occurred. 1 Compare event has occurred.

Table continues on the next page...

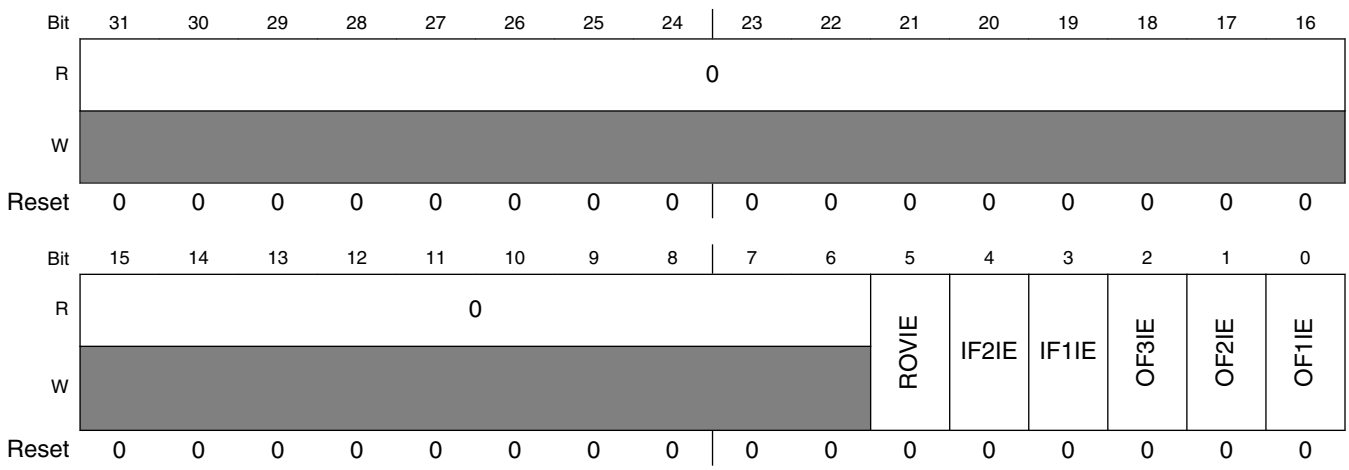
GPTx_SR field descriptions (continued)

Field	Description
1 OF2	See OF3
0 OF1	See OF3

12.1.7.4 GPT Interrupt Register (GPTx_IR)

The GPT Interrupt Register (GPT_IR) contains bits that control whether interrupts are generated after rollover, input capture and output compare events.

Address: Base address + Ch offset



GPTx_IR field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value 0.
5 ROVIE	Rollover Interrupt Enable. The ROVIE bit controls the Rollover interrupt. 0 Rollover interrupt is disabled. 1 Rollover interrupt enabled.
4 IF2IE	IF2IE Input capture 2 Interrupt Enable IF1IE Input capture 1 Interrupt Enable The IF*n*IE bit controls the IF*n*IE Input Capture *n* Interrupt Enable. 0 IF2IE Input Capture *n* Interrupt Enable is disabled. 1 IF2IE Input Capture *n* Interrupt Enable is enabled.

Table continues on the next page...

GPTx_IR field descriptions (continued)

Field	Description
3 IF1IE	See IF2IE
2 OF3IE	OF3IE Output Compare 3 Interrupt Enable OF2IE Output Compare 2 Interrupt Enable OF1IE Output Compare 1 Interrupt Enable The OF*n*IE bit controls the Output Compare Channel *n* interrupt. 0 Output Compare Channel *n* interrupt is disabled. 1 Output Compare Channel *n* interrupt is enabled.
1 OF2IE	See OF3IE
0 OF1IE	See OF3IE

12.1.7.5 GPT Output Compare Register 1 (GPTx_OCR1)

The GPT Compare Register 1 (GPT_OCR1) holds the value that determines when a compare event will be generated on Output Compare Channel 1. Any write access to the Compare register of Channel 1 while in Restart mode (FRR=0) will reset the GPT counter.

An IP Bus Write access to the GPT Output Compare Register1 (GPT_OCR1) occurs *after* one cycle of wait state; an IP Bus Read access occurs *immediately* (0 wait states).

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMP																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

GPTx_OCR1 field descriptions

Field	Description
COMP	Compare Value. When the counter value equals the COMP bit field value, a compare event is generated on Output Compare Channel 1.

12.1.7.6 GPT Output Compare Register 2 (GPTx_OCR2)

The GPT Compare Register 2 (GPT_OCR2) holds the value that determines when a compare event will be generated on Output Compare Channel 2.

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMP																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

GPTx_OCR2 field descriptions

Field	Description
COMP	Compare Value. When the counter value equals the COMP bit field value, a compare event is generated on Output Compare Channel 2.

12.1.7.7 GPT Output Compare Register 3 (GPTx_OCR3)

The GPT Compare Register 3 (GPT_OCR3) holds the value that determines when a compare event will be generated on Output Compare Channel 3.

Address: Base address + 18h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMP																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

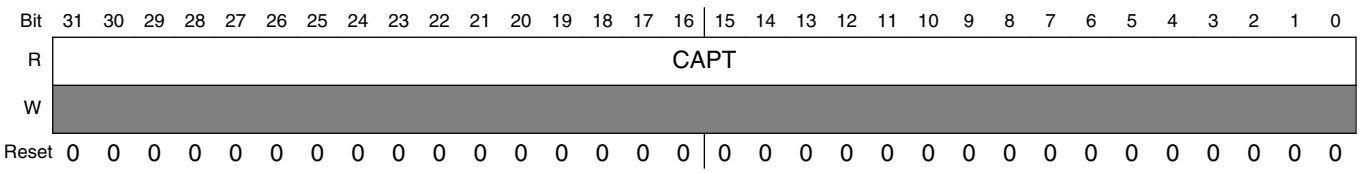
GPTx_OCR3 field descriptions

Field	Description
COMP	Compare Value. When the counter value equals the COMP bit field value, a compare event is generated on Output Compare Channel 3.

12.1.7.8 GPT Input Capture Register 1 (GPTx_ICR1)

The GPT Input Capture Register 1 (GPT_ICR1) is a read-only register that holds the value *that was in the counter during the last capture event* on Input Capture Channel 1.

Address: Base address + 1Ch offset



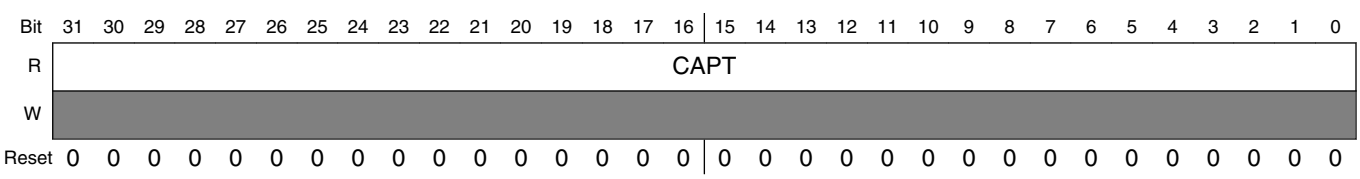
GPTx_ICR1 field descriptions

Field	Description
CAPT	Capture Value. After a capture event on Input Capture Channel 1 occurs, the current value of the counter is loaded into GPT Input Capture Register 1.

12.1.7.9 GPT Input Capture Register 2 (GPTx_ICR2)

The GPT Input capture Register 2 (GPT_ICR2) is a read-only register which holds the value that was in the counter during the last capture event on input capture channel 2.

Address: Base address + 20h offset



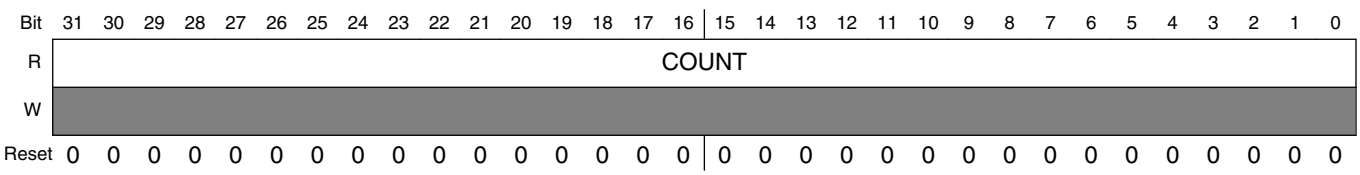
GPTx_ICR2 field descriptions

Field	Description
CAPT	Capture Value. After a capture event on Input Capture Channel 2 occurs, the current value of the counter is loaded into GPT Input Capture Register 2.

12.1.7.10 GPT Counter Register (GPTx_CNT)

The GPT Counter Register (GPT_CNT) is the main counter's register. GPT_CNT is a read-only register and can be read *without affecting the counting process* of the GPT.

Address: Base address + 24h offset



GPTx_CNT field descriptions

Field	Description
COUNT	Counter Value. The COUNT bits show the current count value of the GPT counter.

12.2 Flextimer (FTM)

12.2.1 Introduction

The FlexTimer module (FTM) is a two-to-eight channel timer that supports input capture, output compare, and the generation of PWM signals to control electric motor and power management applications. The FTM time reference is a 16-bit counter that can be used as an unsigned or signed counter.

NOTE

The number of channels supported can vary for each instance of the FTM module on a chip. See the chip-specific FTM information to see how many channels are supported for each module instance. For example, if a module instance supports only six channels, references to channel numbers 6 and 7 do not apply for that instance.

12.2.1.1 FlexTimer philosophy

The FlexTimer is built upon a simple timer, the HCS08 Timer PWM Module – TPM, used for many years on Freescale's 8-bit microcontrollers. The FlexTimer extends the functionality to meet the demands of motor control, digital lighting solutions, and power conversion, while providing low cost and backwards compatibility with the TPM module.

Several key enhancements are made:

- Signed up counter
- Deadtime insertion hardware
- Enhanced triggering functionality
- Initialization and polarity control

All of the features common with the TPM have fully backwards compatible register assignments. The FlexTimer can also use code on the same core platform without change to perform the same functions.

Motor control and power conversion features have been added through a dedicated set of registers and defaults turn off all new features. The new features, such as hardware deadtime insertion, polarity, and output forcing and masking, greatly reduce loading on the execution software and are usually each controlled by a group of registers.

FlexTimer input triggers can be from comparators, ADC, or other submodules to initiate timer functions automatically. These triggers can be linked in a variety of ways during integration of the sub modules so please note the options available for used FlexTimer configuration.

More than one FlexTimers may be synchronized to provide a larger timer with their counters incrementing in unison, assuming the initialization, the input clocks, the initial and final counting values are the same in each FlexTimer.

All main user access registers are buffered to ease the load on the executing software. A number of trigger options exist to determine which registers are updated with this user defined data.

12.2.1.2 Features

The FTM features include:

- FTM source clock is the system clock.
- Prescaler divide-by 1, 2, 4, 8, 16, 32, 64, or 128
- 16-bit counter

    - It can be a free-running counter or a counter with initial and final value
    - The counting can be up or up-down
- Each channel can be configured for input capture, output compare, or edge-aligned PWM mode
- In Input Capture mode:
    - The capture can occur on rising edges, falling edges or both edges
    - An input filter can be selected for some channels
- In Output Compare mode the output signal can be set, cleared, or toggled on match
- All channels can be configured for center-aligned PWM mode
- Each pair of channels can be combined to generate a PWM signal with independent control of both edges of PWM signal
- The FTM channels can operate as pairs with equal outputs, pairs with complementary outputs, or independent channels with independent outputs
- The deadtime insertion is available for each complementary pair
- Generation of match triggers
- Initialization trigger
- Software control of PWM outputs
- The polarity of each channel is configurable
- The generation of an interrupt per channel
- The generation of an interrupt when the counter overflows
- Synchronized loading of write buffered FTM registers
- Write protection for critical registers
- Backwards compatible with TPM
- Testing of input captures for a stuck at zero and one conditions
- Dual edge capture for pulse and period width measurement
- Quadrature decoder with input filters, relative position counting, and interrupt on position count or capture of position count on external event

12.2.1.3 Modes of operation

When the chip is in an active BDM mode, the FTM temporarily suspends all counting until the chip returns to normal user operating mode. During Stop mode, all FTM input clocks are stopped, so the FTM is effectively disabled until clocks resume. During Wait mode, the FTM continues to operate normally. If the FTM does not need to produce a real time reference or provide the interrupt sources needed to wake the chip from Wait mode, the power can then be saved by disabling FTM functions before entering Wait mode.

12.2.1.4 Block diagram

The FTM uses one input/output (I/O) pin per channel, CHn (FTM channel (n)) where n is the channel number (0–7).

The following figure shows the FTM structure. The central component of the FTM is the 16-bit counter with programmable initial and final values and its counting can be up or up-down.

NOTE

The number of channels supported can vary for each instance of the FTM module on a chip. See the chip-specific FTM information to see how many channels are supported for each module instance. For example, if a module instance supports only six channels, references to channel numbers 6 and 7 do not apply for that instance.

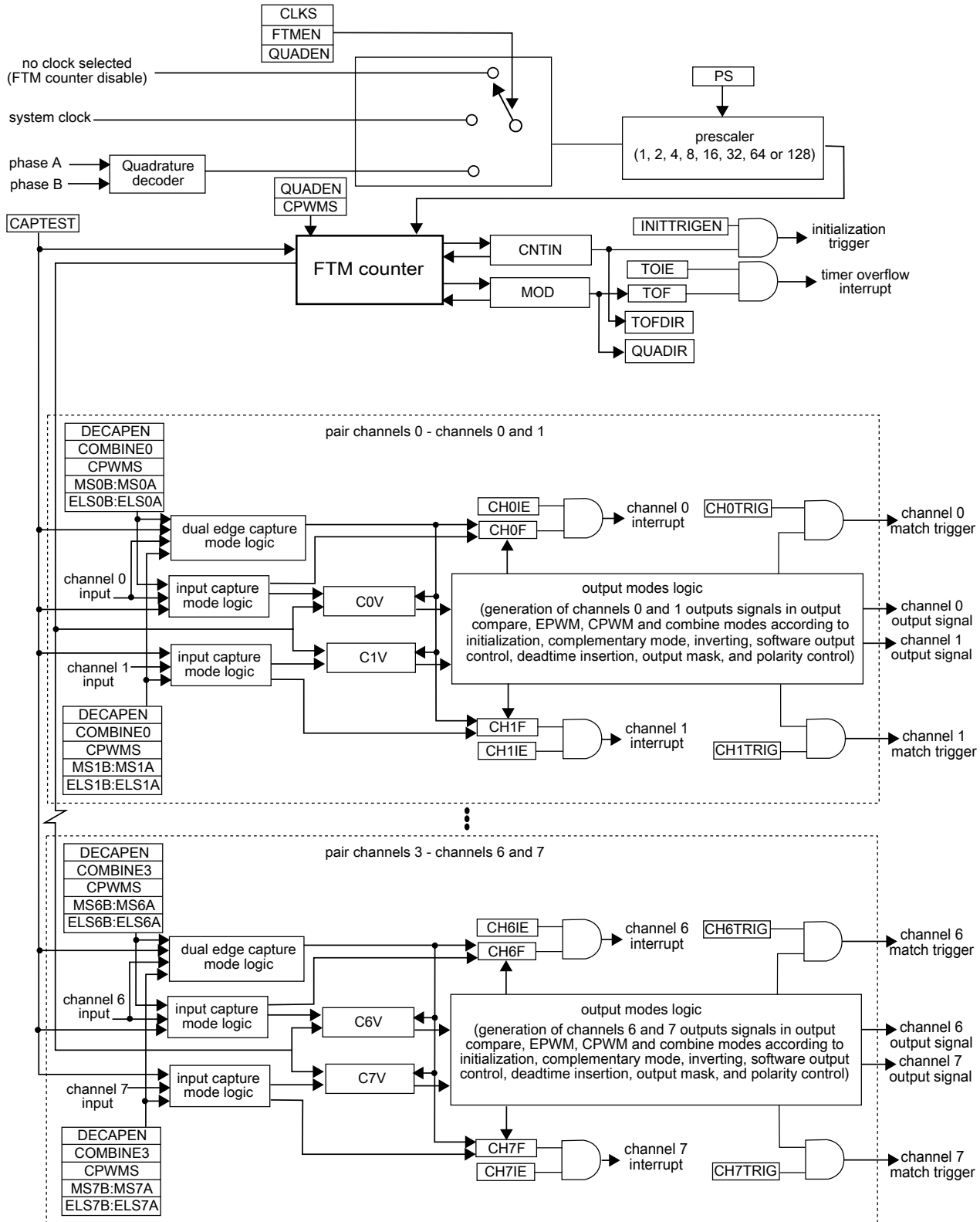


Figure 12-5. FTM block diagram

12.2.2 FTM signal descriptions

Table 12-3 shows the user-accessible signals for the FTM.

Table 12-3. FTM signal descriptions

Signal	Description	I/O	Function
CHn	FTM channel (n), where n can be 7-0	I/O	Each FTM channel can be configured to operate either as input or output. The direction associated with each channel, input or output, is selected according to the mode assigned for that channel.
PHA	Quadrature decoder phase A input. Input pin associated with quadrature decoder phase A.	I	The quadrature decoder phase A input is used as the Quadrature Decoder mode is selected. The phase A input signal is one of the signals that control the FTM counter increment or decrement in the Quadrature Decoder mode.
PHB	Quadrature decoder phase B input. Input pin associated with quadrature decoder phase B.	I	The quadrature decoder phase B input is used as the Quadrature Decoder mode is selected. The phase B input signal is one of the signals that control the FTM counter increment or decrement in the Quadrature Decoder mode.

12.2.3 Memory map and register definition

12.2.3.1 Memory map

This section presents a high-level summary of the FTM registers and how they are mapped.

The registers and bits of an unavailable function in the FTM remain in the memory map and in the reset value, but they have no active function.

Note

Do not write in the region from the CNTIN register through the PWMLOAD register when FTMEN = 0.

NOTE

The number of channels supported can vary for each instance of the FTM module on a chip. See the chip-specific FTM information to see how many channels are supported for each module instance. For example, if a module instance supports only six channels, references to channel numbers 6 and 7 do not apply for that instance.

12.2.3.2 Register descriptions

Accesses to reserved addresses result in transfer errors. Registers for absent channels are considered reserved.

FTM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3064_0000	Status And Control (FTM1_SC)	32	R/W	0000_0000h	12.2.3.3/ 3412
3064_0004	Counter (FTM1_CNT)	32	R/W	0000_0000h	12.2.3.4/ 3413
3064_0008	Modulo (FTM1_MOD)	32	R/W	0000_0000h	12.2.3.5/ 3414
3064_000C	Channel (n) Status And Control (FTM1_C0SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3064_0010	Channel (n) Value (FTM1_C0V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3064_0014	Channel (n) Status And Control (FTM1_C1SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3064_0018	Channel (n) Value (FTM1_C1V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3064_001C	Channel (n) Status And Control (FTM1_C2SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3064_0020	Channel (n) Value (FTM1_C2V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3064_0024	Channel (n) Status And Control (FTM1_C3SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3064_0028	Channel (n) Value (FTM1_C3V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3064_002C	Channel (n) Status And Control (FTM1_C4SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3064_0030	Channel (n) Value (FTM1_C4V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3064_0034	Channel (n) Status And Control (FTM1_C5SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3064_0038	Channel (n) Value (FTM1_C5V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3064_003C	Channel (n) Status And Control (FTM1_C6SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3064_0040	Channel (n) Value (FTM1_C6V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3064_0044	Channel (n) Status And Control (FTM1_C7SC)	32	R/W	0000_0000h	12.2.3.6/ 3415

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3064_0048	Channel (n) Value (FTM1_C7V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3064_004C	Counter Initial Value (FTM1_CNTIN)	32	R/W	0000_0000h	12.2.3.8/ 3418
3064_0050	Capture And Compare Status (FTM1_STATUS)	32	R/W	0000_0000h	12.2.3.9/ 3419
3064_0054	Features Mode Selection (FTM1_MODE)	32	R/W	0000_0004h	12.2.3.10/ 3421
3064_0058	Synchronization (FTM1_SYNC)	32	R/W	0000_0000h	12.2.3.11/ 3422
3064_005C	Initial State For Channels Output (FTM1_OUTINIT)	32	R/W	0000_0000h	12.2.3.12/ 3425
3064_0060	Output Mask (FTM1_OUTMASK)	32	R/W	0000_0000h	12.2.3.13/ 3426
3064_0064	Function For Linked Channels (FTM1_COMBINE)	32	R/W	0000_0000h	12.2.3.14/ 3428
3064_0068	Deadtime Insertion Control (FTM1_DEADTIME)	32	R/W	0000_0000h	12.2.3.15/ 3432
3064_006C	FTM External Trigger (FTM1_EXTTRIG)	32	R/W	0000_0000h	12.2.3.16/ 3433
3064_0070	Channels Polarity (FTM1_POL)	32	R/W	0000_0000h	12.2.3.17/ 3435
3064_0074	Fault Mode Status (FTM1_FMS)	32	R/W	0000_0000h	12.2.3.18/ 3437
3064_0078	Input Capture Filter Control (FTM1_FILTER)	32	R/W	0000_0000h	12.2.3.19/ 3438
3064_0080	Quadrature Decoder Control And Status (FTM1_QDCTRL)	32	R/W	0000_0000h	12.2.3.20/ 3439
3064_0084	Configuration (FTM1_CONF)	32	R/W	0000_0000h	12.2.3.21/ 3441
3064_008C	Synchronization Configuration (FTM1_SYNCONF)	32	R/W	0000_0000h	12.2.3.22/ 3442
3064_0090	FTM Inverting Control (FTM1_INVCTRL)	32	R/W	0000_0000h	12.2.3.23/ 3444
3064_0094	FTM Software Output Control (FTM1_SWOCTRL)	32	R/W	0000_0000h	12.2.3.24/ 3445
3064_0098	FTM PWM Load (FTM1_PWMLOAD)	32	R/W	0000_0000h	12.2.3.25/ 3448
3065_0000	Status And Control (FTM2_SC)	32	R/W	0000_0000h	12.2.3.3/ 3412
3065_0004	Counter (FTM2_CNT)	32	R/W	0000_0000h	12.2.3.4/ 3413
3065_0008	Modulo (FTM2_MOD)	32	R/W	0000_0000h	12.2.3.5/ 3414

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3065_000C	Channel (n) Status And Control (FTM2_C0SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3065_0010	Channel (n) Value (FTM2_C0V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3065_0014	Channel (n) Status And Control (FTM2_C1SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3065_0018	Channel (n) Value (FTM2_C1V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3065_001C	Channel (n) Status And Control (FTM2_C2SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3065_0020	Channel (n) Value (FTM2_C2V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3065_0024	Channel (n) Status And Control (FTM2_C3SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3065_0028	Channel (n) Value (FTM2_C3V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3065_002C	Channel (n) Status And Control (FTM2_C4SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3065_0030	Channel (n) Value (FTM2_C4V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3065_0034	Channel (n) Status And Control (FTM2_C5SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3065_0038	Channel (n) Value (FTM2_C5V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3065_003C	Channel (n) Status And Control (FTM2_C6SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3065_0040	Channel (n) Value (FTM2_C6V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3065_0044	Channel (n) Status And Control (FTM2_C7SC)	32	R/W	0000_0000h	12.2.3.6/ 3415
3065_0048	Channel (n) Value (FTM2_C7V)	32	R/W	0000_0000h	12.2.3.7/ 3418
3065_004C	Counter Initial Value (FTM2_CNTIN)	32	R/W	0000_0000h	12.2.3.8/ 3418
3065_0050	Capture And Compare Status (FTM2_STATUS)	32	R/W	0000_0000h	12.2.3.9/ 3419
3065_0054	Features Mode Selection (FTM2_MODE)	32	R/W	0000_0004h	12.2.3.10/ 3421
3065_0058	Synchronization (FTM2_SYNC)	32	R/W	0000_0000h	12.2.3.11/ 3422
3065_005C	Initial State For Channels Output (FTM2_OUTINIT)	32	R/W	0000_0000h	12.2.3.12/ 3425
3065_0060	Output Mask (FTM2_OUTMASK)	32	R/W	0000_0000h	12.2.3.13/ 3426

Table continues on the next page...

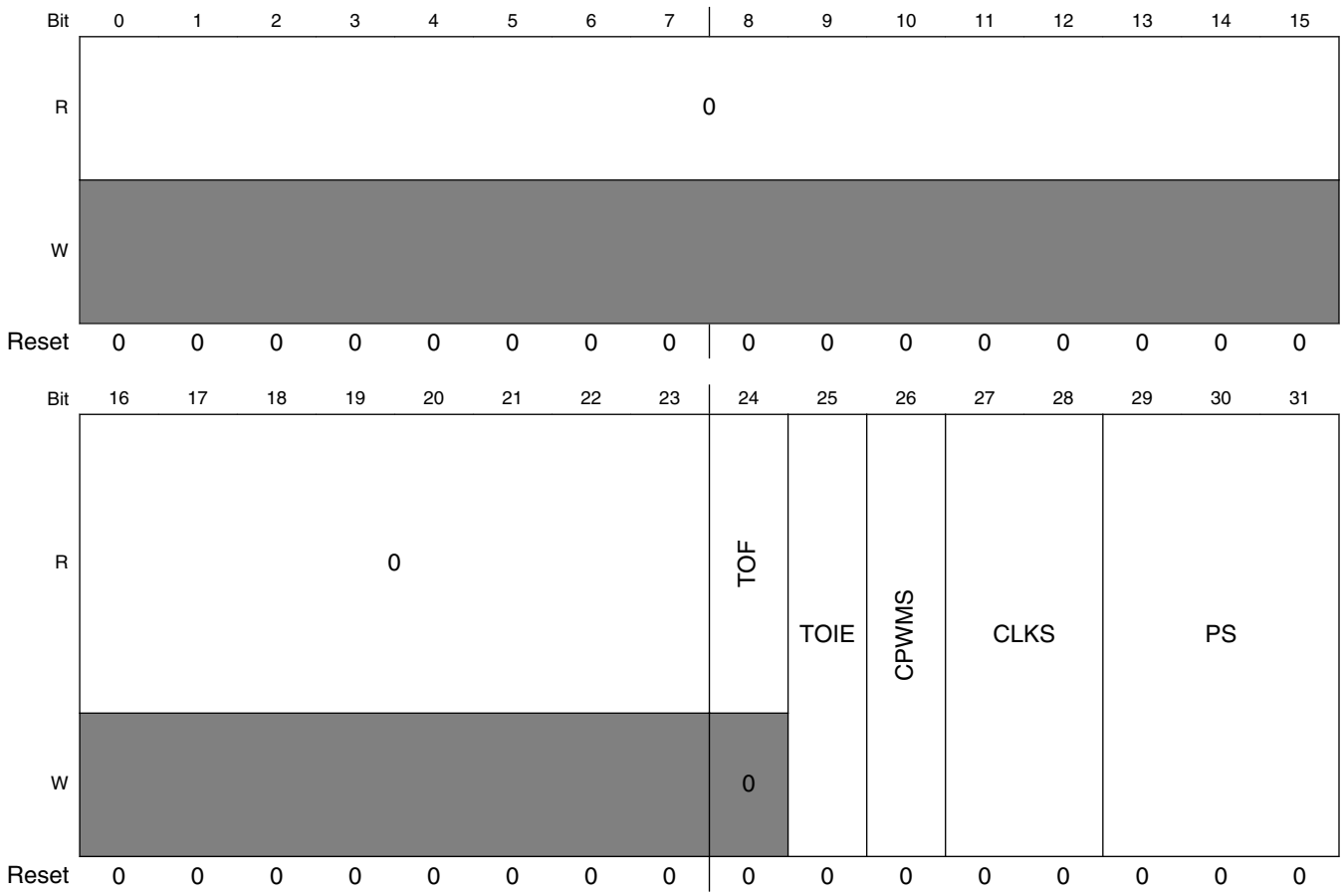
FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3065_0064	Function For Linked Channels (FTM2_COMBINE)	32	R/W	0000_0000h	12.2.3.14/ 3428
3065_0068	Deadtime Insertion Control (FTM2_DEADTIME)	32	R/W	0000_0000h	12.2.3.15/ 3432
3065_006C	FTM External Trigger (FTM2_EXTTRIG)	32	R/W	0000_0000h	12.2.3.16/ 3433
3065_0070	Channels Polarity (FTM2_POL)	32	R/W	0000_0000h	12.2.3.17/ 3435
3065_0074	Fault Mode Status (FTM2_FMS)	32	R/W	0000_0000h	12.2.3.18/ 3437
3065_0078	Input Capture Filter Control (FTM2_FILTER)	32	R/W	0000_0000h	12.2.3.19/ 3438
3065_0080	Quadrature Decoder Control And Status (FTM2_QDCTRL)	32	R/W	0000_0000h	12.2.3.20/ 3439
3065_0084	Configuration (FTM2_CONF)	32	R/W	0000_0000h	12.2.3.21/ 3441
3065_008C	Synchronization Configuration (FTM2_SYNCONF)	32	R/W	0000_0000h	12.2.3.22/ 3442
3065_0090	FTM Inverting Control (FTM2_INVCTRL)	32	R/W	0000_0000h	12.2.3.23/ 3444
3065_0094	FTM Software Output Control (FTM2_SWOCTRL)	32	R/W	0000_0000h	12.2.3.24/ 3445
3065_0098	FTM PWM Load (FTM2_PWMLOAD)	32	R/W	0000_0000h	12.2.3.25/ 3448

12.2.3.3 Status And Control (FTMx_SC)

SC contains the overflow status flag and control bits used to configure the interrupt enable, FTM configuration, clock source, and prescaler factor. These controls relate to all channels within this module.

Address: Base address + 0h offset



FTMx_SC field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 TOF	Timer Overflow Flag Set by hardware when the FTM counter passes the value in the MOD register. The TOF bit is cleared by reading the SC register while TOF is set and then writing a 0 to TOF bit. Writing a 1 to TOF has no effect. If another FTM overflow occurs between the read and write operations, the write operation has no effect; therefore, TOF remains set indicating an overflow has occurred. In this case, a TOF interrupt request is not lost due to the clearing sequence for a previous TOF. 0 FTM counter has not overflowed. 1 FTM counter has overflowed.

Table continues on the next page...

FTMx_SC field descriptions (continued)

Field	Description
25 TOIE	Timer Overflow Interrupt Enable Enables FTM overflow interrupts. 0 Disable TOF interrupts. Use software polling. 1 Enable TOF interrupts. An interrupt is generated when TOF equals one.
26 CPWMS	Center-Aligned PWM Select Selects CPWM mode. This mode configures the FTM to operate in Up-Down Counting mode. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 FTM counter operates in Up Counting mode. 1 FTM counter operates in Up-Down Counting mode.
27–28 CLKS	Clock Source Selection Selects FTM counter clock sources. This field is write protected. It can be written only when MODE[WPDIS] = 1. 00 No clock selected. This in effect disables the FTM counter. 01 System clock 10 Reserved
29–31 PS	Prescale Factor Selection Selects one of 8 division factors for the clock source selected by CLKS. The new prescaler factor affects the clock source on the next system clock cycle after the new value is updated into the register bits. This field is write protected. It can be written only when MODE[WPDIS] = 1. 000 Divide by 1 001 Divide by 2 010 Divide by 4 011 Divide by 8 100 Divide by 16 101 Divide by 32 110 Divide by 64 111 Divide by 128

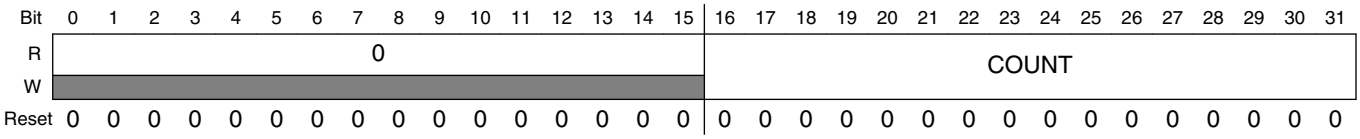
12.2.3.4 Counter (FTMx_CNT)

The CNT register contains the FTM counter value.

Reset clears the CNT register. Writing any value to COUNT updates the counter with its initial value, CNTIN.

When BDM is active, the FTM counter is frozen. This is the value that you may read.

Address: Base address + 4h offset



FTMx_CNT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Counter Value

12.2.3.5 Modulo (FTMx_MOD)

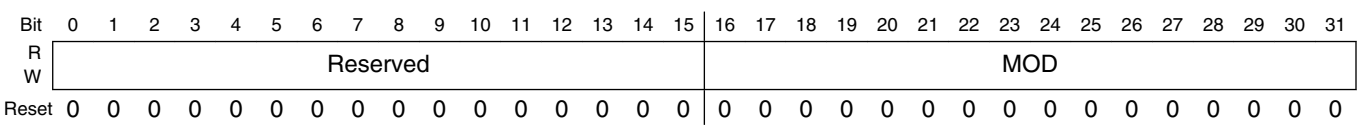
The Modulo register contains the modulo value for the FTM counter. After the FTM counter reaches the modulo value, the overflow flag (TOF) becomes set at the next clock, and the next value of FTM counter depends on the selected counting method; see Counter.

Writing to the MOD register latches the value into a buffer. The MOD register is updated with the value of its write buffer according to Registers updated from write buffers.

If FTMEN = 0, this write coherency mechanism may be manually reset by writing to the SC register whether BDM is active or not.

Initialize the FTM counter, by writing to CNT, before writing to the MOD register to avoid confusion about when the first counter overflow will occur.

Address: Base address + 8h offset



FTMx_MOD field descriptions

Field	Description
0–15 Reserved	This field is reserved.
16–31 MOD	Modulo Value

12.2.3.6 Channel (n) Status And Control (FTM*x*_C*n*SC)

CnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

Table 12-4. Mode, edge, and level selection

DECAPEN	COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	X	X	XX	00	Pin not used for FTM—revert the channel pin to general purpose I/O or other peripheral control	
0	0	0	00	01	Input Capture	Capture on Rising Edge Only
				10		Capture on Falling Edge Only
				11		Capture on Rising or Falling Edge
			01	01	Output Compare	Toggle Output on match
				10		Clear Output on match
				11		Set Output on match
			1X	10	Edge-Aligned PWM	High-true pulses (clear Output on match)
				X1		Low-true pulses (set Output on match)
		1	XX	10	Center-Aligned PWM	High-true pulses (clear Output on match-up)
				X1		Low-true pulses (set Output on match-up)
	1	0	XX	10	Combine PWM	High-true pulses (set on channel (n) match, and clear on channel (n+1) match)
				X1		Low-true pulses (clear on channel (n) match, and set on channel (n +1) match)

Table continues on the next page...

Table 12-4. Mode, edge, and level selection (continued)

DECAPEN	COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
1	0	0	X0	See the following table (Table 12-5).	Dual Edge Capture	One-Shot Capture mode
			X1			Continuous Capture mode

Table 12-5. Dual Edge Capture mode — edge polarity selection

ELSnB	ELSnA	Channel Port Enable	Detected Edges
0	0	Disabled	No edge
0	1	Enabled	Rising edge
1	0	Enabled	Falling edge
1	1	Enabled	Rising and falling edges

Address: Base address + Ch offset + (8d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								CHF	CHIE	MSB	MSA	ELSB	ELSA	ICRST	DMA
W									0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_C*n*SC field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 CHF	Channel Flag Set by hardware when an event occurs on the channel. CHF is cleared by reading the CSC register while CHnF is set and then writing a 0 to the CHF bit. Writing a 1 to CHF has no effect. If another event occurs between the read and write operations, the write operation has no effect; therefore, CHF remains set indicating an event has occurred. In this case a CHF interrupt request is not lost due to the clearing sequence for a previous CHF. 0 No channel event has occurred. 1 A channel event has occurred.
25 CHIE	Channel Interrupt Enable Enables channel interrupts. 0 Disable channel interrupts. Use software polling. 1 Enable channel interrupts.
26 MSB	Channel Mode Select Used for further selections in the channel logic. Its functionality is dependent on the channel mode. See Table 12-4 . This field is write protected. It can be written only when MODE[WPDIS] = 1.
27 MSA	Channel Mode Select Used for further selections in the channel logic. Its functionality is dependent on the channel mode. See Table 12-4 . This field is write protected. It can be written only when MODE[WPDIS] = 1.
28 ELSB	Edge or Level Select The functionality of ELSB and ELSA depends on the channel mode. See Table 12-4 . This field is write protected. It can be written only when MODE[WPDIS] = 1.
29 ELSA	Edge or Level Select The functionality of ELSB and ELSA depends on the channel mode. See Table 12-4 . This field is write protected. It can be written only when MODE[WPDIS] = 1.
30 ICRST	FTM counter reset by the selected input capture event. FTM counter reset is driven by the selected event of the channel (n) in the Input Capture mode. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 FTM counter is not reset when the selected channel (n) input event is detected. 1 FTM counter is reset when the selected channel (n) input event is detected.
31 DMA	DMA Enable Enables DMA transfers for the channel. 0 Disable DMA transfers. 1 Enable DMA transfers.

12.2.3.7 Channel (n) Value (FTM*x*_C*n*V)

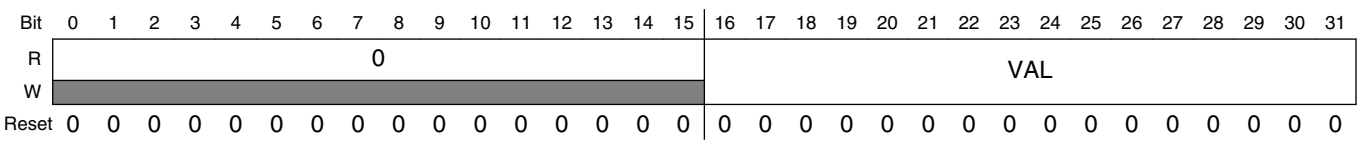
These registers contain the captured FTM counter value for the input modes or the match value for the output modes.

In Input Capture, Capture Test, and Dual Edge Capture modes, any write to a CnV register is ignored.

In output modes, writing to a CnV register latches the value into a buffer. A CnV register is updated with the value of its write buffer according to Registers updated from write buffers.

If FTMEN = 0, this write coherency mechanism may be manually reset by writing to the CnSC register whether BDM mode is active or not.

Address: Base address + 10h offset + (8d × i), where i=0d to 7d



FTM*x*_C*n*V field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 VAL	Channel Value Captured FTM counter value of the input modes or the match value for the output modes

12.2.3.8 Counter Initial Value (FTM*x*_CNTIN)

The Counter Initial Value register contains the initial value for the FTM counter.

Writing to the CNTIN register latches the value into a buffer. The CNTIN register is updated with the value of its write buffer according to Registers updated from write buffers.

When the FTM clock is initially selected, by writing a non-zero value to the CLKS bits, the FTM counter starts with the value 0x0000. To avoid this behavior, before the first write to select the FTM clock, write the new value to the the CNTIN register and then initialize the FTM counter by writing any value to the CNT register.

Address: Base address + 4Ch offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved																INIT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_CNTIN field descriptions

Field	Description
0–15 Reserved	This field is reserved.
16–31 INIT	Initial Value Of The FTM Counter

12.2.3.9 Capture And Compare Status (FTMx_STATUS)

The STATUS register contains a copy of the status flag CHnF bit in CnSC for each FTM channel for software convenience.

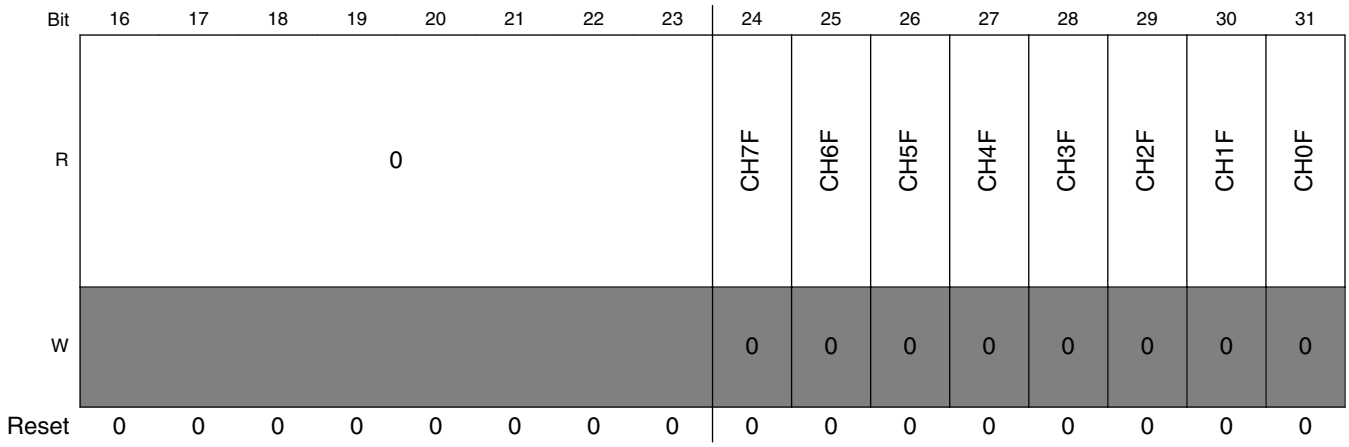
Each CHnF bit in STATUS is a mirror of CHnF bit in CnSC. All CHnF bits can be checked using only one read of STATUS. All CHnF bits can be cleared by reading STATUS followed by writing 0x00 to STATUS.

Hardware sets the individual channel flags when an event occurs on the channel. CHnF is cleared by reading STATUS while CHnF is set and then writing a 0 to the CHnF bit. Writing a 1 to CHnF has no effect.

If another event occurs between the read and write operations, the write operation has no effect; therefore, CHnF remains set indicating an event has occurred. In this case, a CHnF interrupt request is not lost due to the clearing sequence for a previous CHnF.

Address: Base address + 50h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



FTMx_STATUS field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 CH7F	Channel 7 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
25 CH6F	Channel 6 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
26 CH5F	Channel 5 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
27 CH4F	Channel 4 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
28 CH3F	Channel 3 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
29 CH2F	Channel 2 Flag See the register description.

Table continues on the next page...

FTMx_STATUS field descriptions (continued)

Field	Description
	0 No channel event has occurred. 1 A channel event has occurred.
30 CH1F	Channel 1 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
31 CH0F	Channel 0 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.

12.2.3.10 Features Mode Selection (FTMx_MODE)

This register contains the global enable bit for FTM-specific features and the control bits used to configure:

- Capture Test mode
- PWM synchronization
- Write protection
- Channel output initialization

These controls relate to all channels within this module.

Address: Base address + 54h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								0			CAPTEST	PWMSYNC	WPDIS	INIT	FTMEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

FTMx_MODE field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 CAPTEST	Capture Test Mode Enable Enables the capture test mode. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Capture test mode is disabled. 1 Capture test mode is enabled.
28 PWMSYNC	PWM Synchronization Mode Selects which triggers can be used by MOD, CnV, OUTMASK, and FTM counter synchronization. See PWM synchronization. The PWMSYNC bit configures the synchronization when SYNCMODE is 0. 0 No restrictions. Software and hardware triggers can be used by MOD, CnV, OUTMASK, and FTM counter synchronization. 1 Software trigger can only be used by MOD and CnV synchronization, and hardware triggers can only be used by OUTMASK and FTM counter synchronization.
29 WPDIS	Write Protection Disable When write protection is enabled (WPDIS = 0), write protected bits cannot be written. When write protection is disabled (WPDIS = 1), write protected bits can be written. The WPDIS bit is the negation of the WPEN bit. WPDIS is cleared when 1 is written to WPEN. WPDIS is set when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPDIS has no effect. 0 Write protection is enabled. 1 Write protection is disabled.
30 INIT	Initialize The Channels Output When a 1 is written to INIT bit the channels output is initialized according to the state of their corresponding bit in the OUTINIT register. Writing a 0 to INIT bit has no effect. The INIT bit is always read as 0.
31 FTMEN	FTM Enable This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 TPM compatibility. Free running counter and synchronization compatible with TPM. 1 Free running counter and synchronization are different from TPM behavior.

12.2.3.11 Synchronization (FTMx_SYNC)

This register configures the PWM synchronization.

A synchronization event can perform the synchronized update of MOD, CV, and OUTMASK registers with the value of their write buffer and the FTM counter initialization.

NOTE

The software trigger, SWSYNC bit, and hardware triggers TRIG0, TRIG1, and TRIG2 bits have a potential conflict if used together when SYNCMODE = 0. Use only hardware or software triggers but not both at the same time, otherwise unpredictable behavior is likely to happen.

The selection of the loading point, CNTMAX and CNTMIN bits, is intended to provide the update of MOD, CNTIN, and CnV registers across all enabled channels simultaneously. The use of the loading point selection together with SYNCMODE = 0 and hardware trigger selection, TRIG0, TRIG1, or TRIG2 bits, is likely to result in unpredictable behavior.

The synchronization event selection also depends on the PWMSYNC (MODE register) and SYNCMODE (SYNCONF register) bits. See PWM synchronization.

Address: Base address + 58h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								SWSYNC	TRIG2	TRIG1	TRIG0	SYNCHOM	REINIT	CNTMAX	CNTMIN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_SYNC field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 SWSYNC	PWM Synchronization Software Trigger Selects the software trigger as the PWM synchronization trigger. The software trigger happens when a 1 is written to SWSYNC bit. 0 Software trigger is not selected. 1 Software trigger is selected.
25 TRIG2	PWM Synchronization Hardware Trigger 2

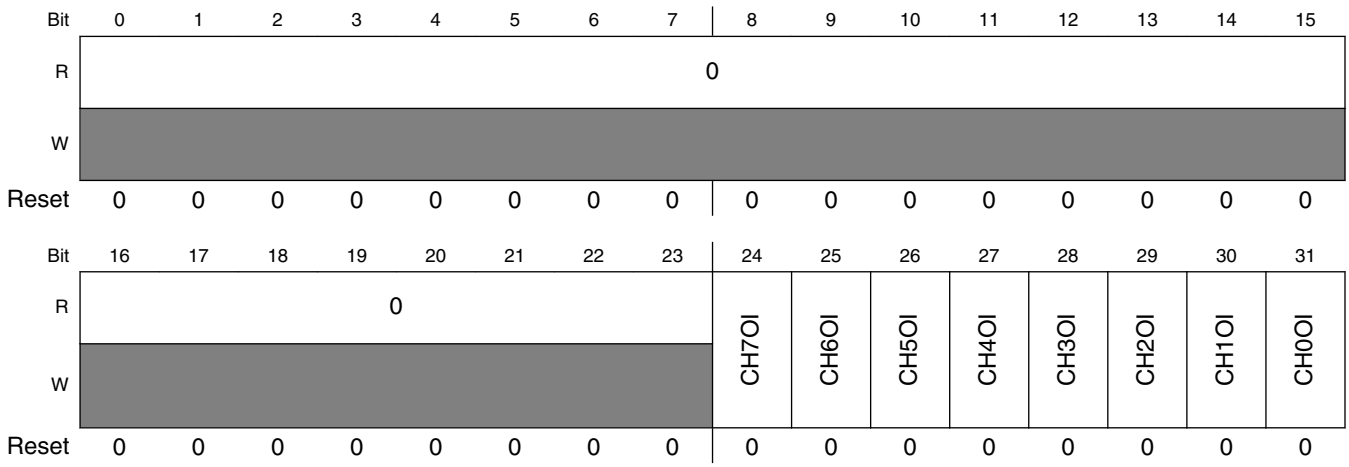
Table continues on the next page...

FTMx_SYNC field descriptions (continued)

Field	Description
	Enables hardware trigger 2 to the PWM synchronization. Hardware trigger 2 happens when a rising edge is detected at the trigger 2 input signal. 0 Trigger is disabled. 1 Trigger is enabled.
26 TRIG1	PWM Synchronization Hardware Trigger 1 Enables hardware trigger 1 to the PWM synchronization. Hardware trigger 1 happens when a rising edge is detected at the trigger 1 input signal. 0 Trigger is disabled. 1 Trigger is enabled.
27 TRIG0	PWM Synchronization Hardware Trigger 0 Enables hardware trigger 0 to the PWM synchronization. Hardware trigger 0 occurs when a rising edge is detected at the trigger 0 input signal. 0 Trigger is disabled. 1 Trigger is enabled.
28 SYNCHOM	Output Mask Synchronization Selects when the OUTMASK register is updated with the value of its buffer. 0 OUTMASK register is updated with the value of its buffer in all rising edges of the system clock. 1 OUTMASK register is updated with the value of its buffer only by the PWM synchronization.
29 REINIT	FTM Counter Reinitialization By Synchronization (FTM counter synchronization) Determines if the FTM counter is reinitialized when the selected trigger for the synchronization is detected. The REINIT bit configures the synchronization when SYNCMODE is zero. 0 FTM counter continues to count normally. 1 FTM counter is updated with its initial value when the selected trigger is detected.
30 CNTMAX	Maximum Loading Point Enable Selects the maximum loading point to PWM synchronization. See Boundary cycle and loading points. If CNTMAX is 1, the selected loading point is when the FTM counter reaches its maximum value (MOD register). 0 The maximum loading point is disabled. 1 The maximum loading point is enabled.
31 CNTMIN	Minimum Loading Point Enable Selects the minimum loading point to PWM synchronization. See Boundary cycle and loading points. If CNTMIN is one, the selected loading point is when the FTM counter reaches its minimum value (CNTIN register). 0 The minimum loading point is disabled. 1 The minimum loading point is enabled.

12.2.3.12 Initial State For Channels Output (FTMx_OUTINIT)

Address: Base address + 5Ch offset



FTMx_OUTINIT field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 CH7OI	Channel 7 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
25 CH6OI	Channel 6 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
26 CH5OI	Channel 5 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
27 CH4OI	Channel 4 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
28 CH3OI	Channel 3 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs.

Table continues on the next page...

FTMx_OUTINIT field descriptions (continued)

Field	Description
	0 The initialization value is 0. 1 The initialization value is 1.
29 CH2OI	Channel 2 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
30 CH1OI	Channel 1 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
31 CH0OI	Channel 0 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.

12.2.3.13 Output Mask (FTMx_OUTMASK)

This register provides a mask for each FTM channel. The mask of a channel determines if its output responds, that is, it is masked or not, when a match occurs. This feature is used for BLDC control where the PWM signal is presented to an electric motor at specific times to provide electronic commutation.

Any write to the OUTMASK register, stores the value in its write buffer. The register is updated with the value of its write buffer according to PWM synchronization.

Address: Base address + 60h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								CH7OM	CH6OM	CH5OM	CH4OM	CH3OM	CH2OM	CH1OM	CH0OM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_OUTMASK field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 CH7OM	Channel 7 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
25 CH6OM	Channel 6 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
26 CH5OM	Channel 5 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
27 CH4OM	Channel 4 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
28 CH3OM	Channel 3 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
29 CH2OM	Channel 2 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
30 CH1OM	Channel 1 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
31 CH0OM	Channel 0 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.

12.2.3.14 Function For Linked Channels (FTMx_COMBINE)

This register contains the control bits used to configure the synchronization, deadtime insertion, Dual Edge Capture mode, Complementary, and Combine mode for each pair of channels (n) and (n+1), where n equals 0, 2, 4, and 6.

Address: Base address + 64h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	SYNCEN3	DTEN3	DECAP3	DECAPEN3	COMP3	COMBINE3	0	0	SYNCEN2	DTEN2	DECAP2	DECAPEN2	COMP2	COMBINE2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	SYNCEN1	DTEN1	DECAP1	DECAPEN1	COMP1	COMBINE1	0	0	SYNCEN0	DTEN0	DECAP0	DECAPEN0	COMP0	COMBINE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_COMBINE field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 SYNCEN3	Synchronization Enable For n = 6 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
3 DTEN3	Deadtime Enable For n = 6 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
4 DECAP3	Dual Edge Capture Mode Captures For n = 6 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
	This field applies only when DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
5 DECAPEN3	Dual Edge Capture Mode Enable For n = 6 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 12-4. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
6 COMP3	Complement Of Channel (n) for n = 6 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
7 COMBINE3	Combine Channels For n = 6 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 SYNCEN2	Synchronization Enable For n = 4 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
11 DTEN2	Deadtime Enable For n = 4 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
12 DECAP2	Dual Edge Capture Mode Captures For n = 4 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
	This field applies only when DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
13 DECAPEN2	Dual Edge Capture Mode Enable For n = 4 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 12-4. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
14 COMP2	Complement Of Channel (n) For n = 4 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
15 COMBINE2	Combine Channels For n = 4 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 SYNCEN1	Synchronization Enable For n = 2 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
19 DTEN1	Deadtime Enable For n = 2 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
20 DECAP1	Dual Edge Capture Mode Captures For n = 2 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
	This field applies only when DECAPEN = 1. DECAP bit is cleared automatically by hardware if Dual Edge Capture – One-Shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
21 DECAPEN1	Dual Edge Capture Mode Enable For n = 2 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 12-4. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
22 COMP1	Complement Of Channel (n) For n = 2 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
23 COMBINE1	Combine Channels For n = 2 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 SYNCEN0	Synchronization Enable For n = 0 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
27 DTEN0	Deadtime Enable For n = 0 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
28 DECAP0	Dual Edge Capture Mode Captures For n = 0 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
	This field applies only when DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
29 DECAPEN0	Dual Edge Capture Mode Enable For n = 0 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 12-4. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
30 COMP0	Complement Of Channel (n) For n = 0 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
31 COMBINE0	Combine Channels For n = 0 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.

12.2.3.15 Deadtime Insertion Control (FTMx_DEADTIME)

This register selects the deadtime prescaler factor and deadtime value. All FTM channels use this clock prescaler and this deadtime value for the deadtime insertion.

Address: Base address + 68h offset

Bit	0–15	16–23	24	25	26–31
R	0		DTPS		DTVAL
W					
Reset	0	0	0	0	0

FTMx_DEADTIME field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FTMx_DEADTIME field descriptions (continued)

Field	Description
24–25 DTPS	Deadtime Prescaler Value Selects the division factor of the system clock. This prescaled clock is used by the deadtime counter. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0x Divide the system clock by 1. 10 Divide the system clock by 4. 11 Divide the system clock by 16.
26–31 DTVAL	Deadtime Value Selects the deadtime insertion value for the deadtime counter. The deadtime counter is clocked by a scaled version of the system clock. See the description of DTPS. Deadtime insert value = (DTPS × DTVAL). DTVAL selects the number of deadtime counts inserted as follows: When DTVAL is 0, no counts are inserted. When DTVAL is 1, 1 count is inserted. When DTVAL is 2, 2 counts are inserted. This pattern continues up to a possible 63 counts. This field is write protected. It can be written only when MODE[WPDIS] = 1.

12.2.3.16 FTM External Trigger (FTMx_EXTTRIG)

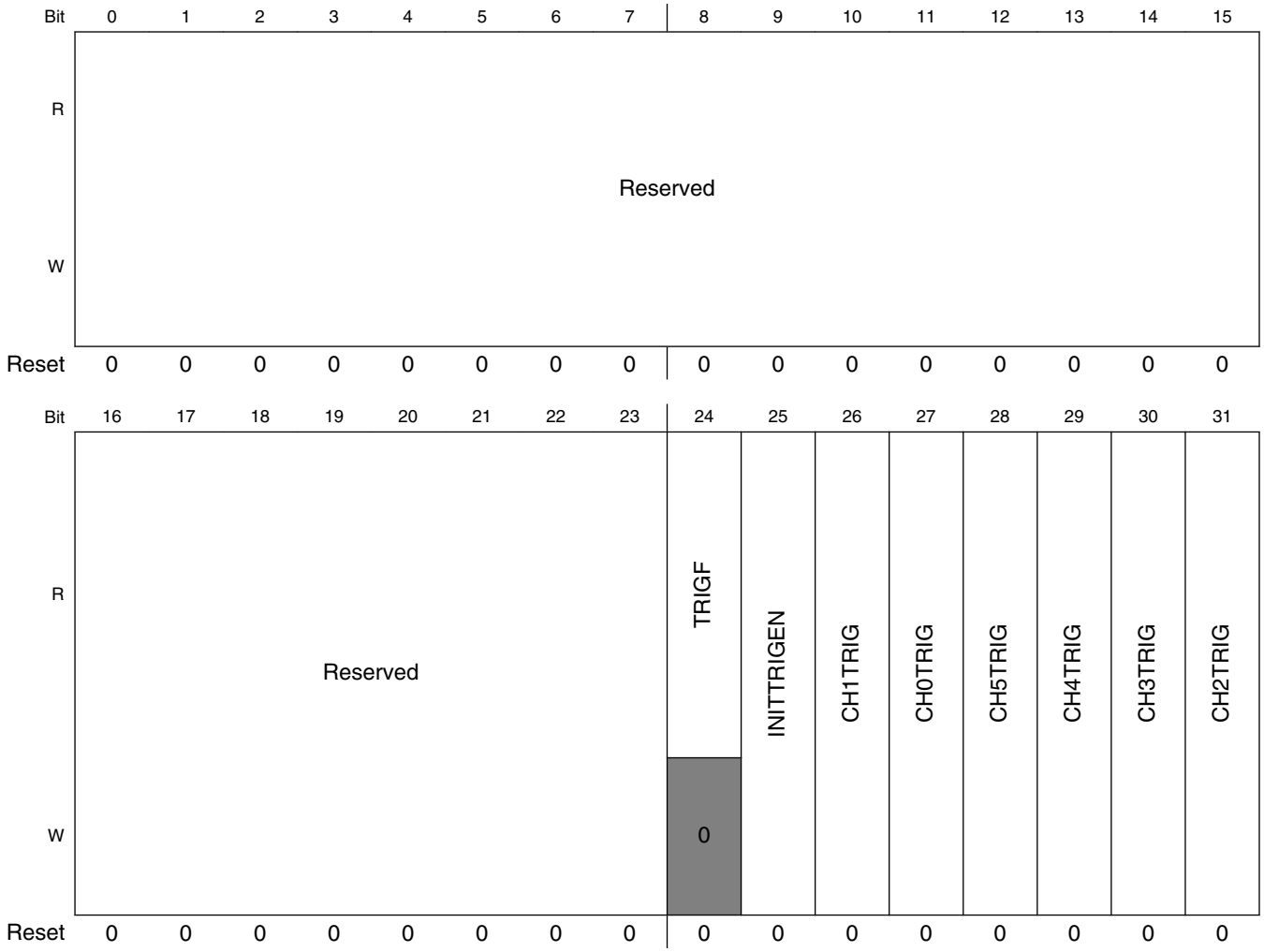
This register:

- Indicates when a channel trigger was generated
- Enables the generation of a trigger when the FTM counter is equal to its initial value
- Selects which channels are used in the generation of the channel triggers

Several channels can be selected to generate multiple triggers in one PWM period. See Channel trigger output and Initialization trigger.

Channels 6 and 7 are not used to generate channel triggers.

Address: Base address + 6Ch offset



FTMx_EXTTRIG field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 TRIGF	Channel Trigger Flag Set by hardware when a channel trigger is generated. Clear TRIGF by reading EXTTRIG while TRIGF is set and then writing a 0 to TRIGF. Writing a 1 to TRIGF has no effect. If another channel trigger is generated before the clearing sequence is completed, the sequence is reset so TRIGF remains set after the clear sequence is completed for the earlier TRIGF. 0 No channel trigger was generated. 1 A channel trigger was generated.
25 INITTRIGEN	Initialization Trigger Enable Enables the generation of the trigger when the FTM counter is equal to the CNTIN register. 0 The generation of initialization trigger is disabled. 1 The generation of initialization trigger is enabled.

Table continues on the next page...

FTMx_EXTTRIG field descriptions (continued)

Field	Description
26 CH1TRIG	Channel 1 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
27 CH0TRIG	Channel 0 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
28 CH5TRIG	Channel 5 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
29 CH4TRIG	Channel 4 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
30 CH3TRIG	Channel 3 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
31 CH2TRIG	Channel 2 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.

12.2.3.17 Channels Polarity (FTMx_POL)

This register defines the output polarity of the FTM channels.

NOTE

The safe value of a channel is the value of its POL bit.

Address: Base address + 70h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								POL7	POL6	POL5	POL4	POL3	POL2	POL1	POL0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_POL field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 POL7	Channel 7 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
25 POL6	Channel 6 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
26 POL5	Channel 5 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
27 POL4	Channel 4 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
28 POL3	Channel 3 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
29 POL2	Channel 2 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.

Table continues on the next page...

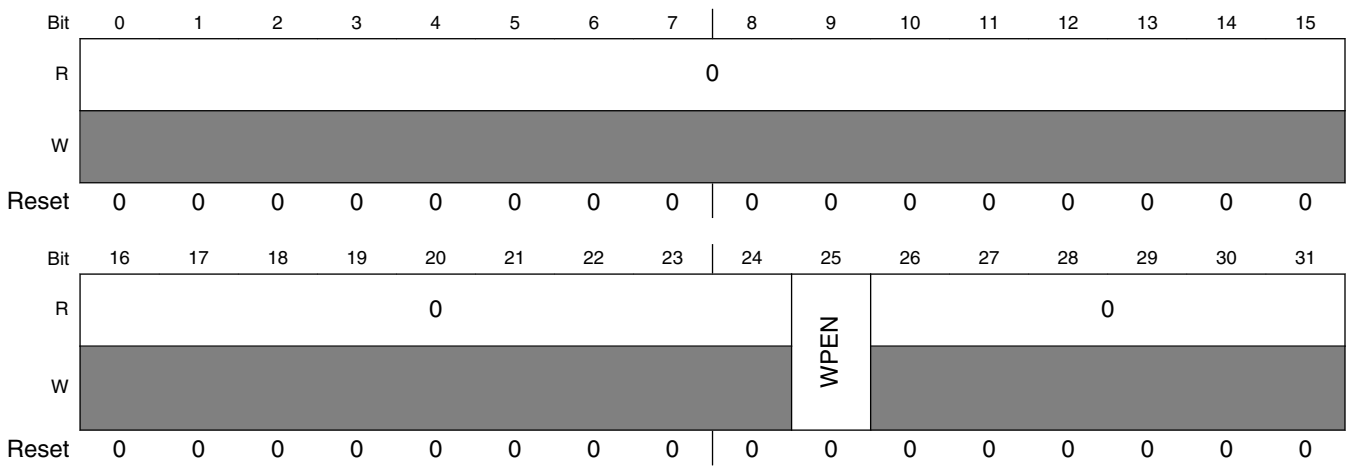
FTMx_POL field descriptions (continued)

Field	Description
30 POL1	Channel 1 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
31 POL0	Channel 0 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.

12.2.3.18 Fault Mode Status (FTMx_FMS)

This register contains the write protection enable bit.

Address: Base address + 74h offset



FTMx_FMS field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 WPEN	Write Protection Enable The WPEN bit is the negation of the WPDIS bit. WPEN is set when 1 is written to it. WPEN is cleared when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPEN has no effect. 0 Write protection is disabled. Write protected bits can be written. 1 Write protection is enabled. Write protected bits cannot be written.

Table continues on the next page...

FTMx_FMS field descriptions (continued)

Field	Description
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.2.3.19 Input Capture Filter Control (FTMx_FILTER)

This register selects the filter value for the inputs of channels.

Channels 4, 5, 6 and 7 do not have an input filter.

NOTE

Writing to the FILTER register has immediate effect and must be done only when the channels 0, 1, 2, and 3 are not in input modes. Failure to do this could result in a missing valid signal.

Address: Base address + 78h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved																CH3FVAL				CH2FVAL				CH1FVAL				CH0FVAL			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

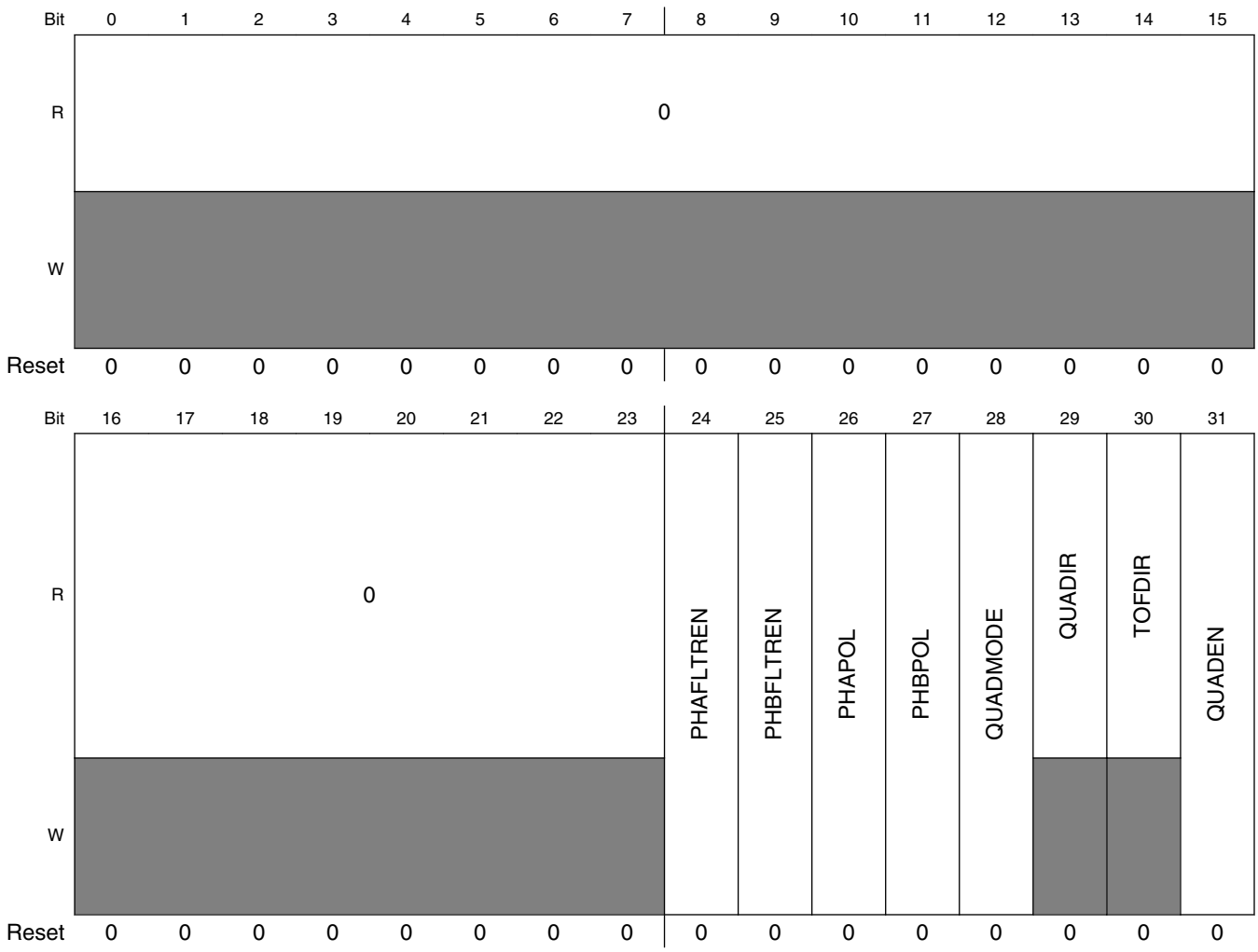
FTMx_FILTER field descriptions

Field	Description
0–15 Reserved	This field is reserved.
16–19 CH3FVAL	Channel 3 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.
20–23 CH2FVAL	Channel 2 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.
24–27 CH1FVAL	Channel 1 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.
28–31 CH0FVAL	Channel 0 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.

12.2.3.20 Quadrature Decoder Control And Status (FTMx_QDCTRL)

This register has the control and status bits for the Quadrature Decoder mode.

Address: Base address + 80h offset



FTMx_QDCTRL field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PHAFLTREN	Phase A Input Filter Enable Enables the filter for the quadrature decoder phase A input. The filter value for the phase A input is defined by the CH0FVAL field of FILTER. The phase A filter is also disabled when CH0FVAL is zero. 0 Phase A input filter is disabled. 1 Phase A input filter is enabled.

Table continues on the next page...

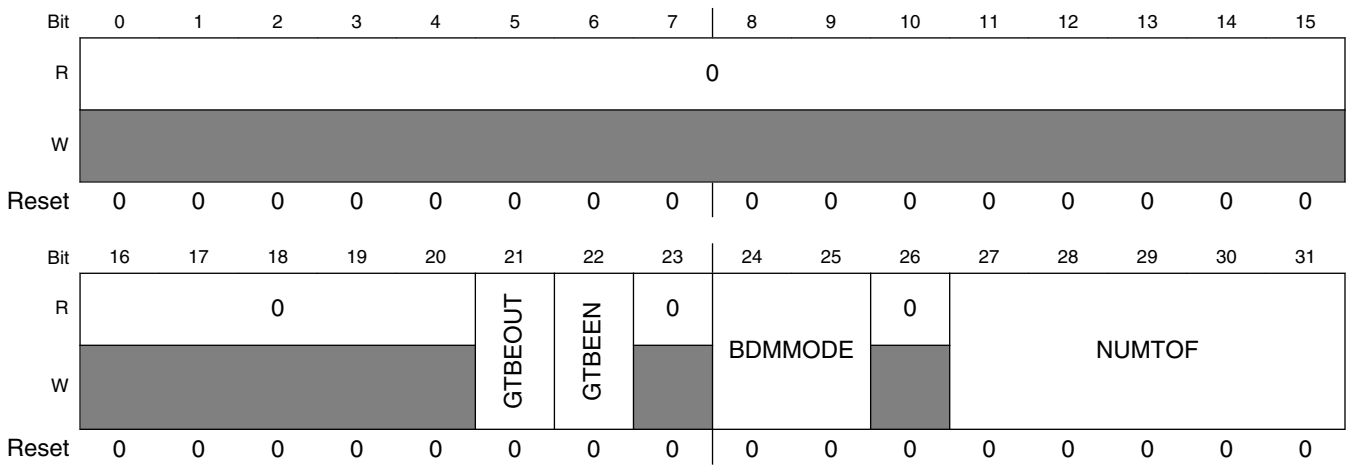
FTMx_QDCTRL field descriptions (continued)

Field	Description
25 PHBFLTREN	Phase B Input Filter Enable Enables the filter for the quadrature decoder phase B input. The filter value for the phase B input is defined by the CH1FVAL field of FILTER. The phase B filter is also disabled when CH1FVAL is zero. 0 Phase B input filter is disabled. 1 Phase B input filter is enabled.
26 PHAPOL	Phase A Input Polarity Selects the polarity for the quadrature decoder phase A input. 0 Normal polarity. Phase A input signal is not inverted before identifying the rising and falling edges of this signal. 1 Inverted polarity. Phase A input signal is inverted before identifying the rising and falling edges of this signal.
27 PHBPOL	Phase B Input Polarity Selects the polarity for the quadrature decoder phase B input. 0 Normal polarity. Phase B input signal is not inverted before identifying the rising and falling edges of this signal. 1 Inverted polarity. Phase B input signal is inverted before identifying the rising and falling edges of this signal.
28 QUADMODE	Quadrature Decoder Mode Selects the encoding mode used in the Quadrature Decoder mode. 0 Phase A and phase B encoding mode. 1 Count and direction encoding mode.
29 QUADIR	FTM Counter Direction In Quadrature Decoder Mode Indicates the counting direction. 0 Counting direction is decreasing (FTM counter decrement). 1 Counting direction is increasing (FTM counter increment).
30 TOFDIR	Timer Overflow Direction In Quadrature Decoder Mode Indicates if the TOF bit was set on the top or the bottom of counting. 0 TOF bit was set on the bottom of counting. There was an FTM counter decrement and FTM counter changes from its minimum value (CNTIN register) to its maximum value (MOD register). 1 TOF bit was set on the top of counting. There was an FTM counter increment and FTM counter changes from its maximum value (MOD register) to its minimum value (CNTIN register).
31 QUADEN	Quadrature Decoder Mode Enable Enables the Quadrature Decoder mode. In this mode, the phase A and B input signals control the FTM counter direction. The Quadrature Decoder mode has precedence over the other modes. See Table 12-4. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Quadrature Decoder mode is disabled. 1 Quadrature Decoder mode is enabled.

12.2.3.21 Configuration (FTMx_CONF)

This register selects the number of times that the FTM counter overflow should occur before the TOF bit to be set, the FTM behavior in BDM modes, the use of an external global time base, and the global time base signal generation.

Address: Base address + 84h offset



FTMx_CONF field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 GTBEOUT	Global Time Base Output Enables the global time base signal generation to other FTMs. 0 A global time base signal generation is disabled. 1 A global time base signal generation is enabled.
22 GTBEEN	Global Time Base Enable Configures the FTM to use an external global time base signal that is generated by another FTM. 0 Use of an external global time base is disabled. 1 Use of an external global time base is enabled.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 BDMMODE	BDM Mode Selects the FTM behavior in BDM mode. See BDM mode.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FTMx_CONF field descriptions (continued)

Field	Description
27–31 NUMTOF	TOF Frequency Selects the ratio between the number of counter overflows to the number of times the TOF bit is set. NUMTOF = 0: The TOF bit is set for each counter overflow. NUMTOF = 1: The TOF bit is set for the first counter overflow but not for the next overflow. NUMTOF = 2: The TOF bit is set for the first counter overflow but not for the next 2 overflows. NUMTOF = 3: The TOF bit is set for the first counter overflow but not for the next 3 overflows. This pattern continues up to a maximum of 31.

12.2.3.22 Synchronization Configuration (FTMx_SYNCONF)

This register selects the PWM synchronization configuration, SWOCTRL, INVCTRL and CNTIN registers synchronization, if FTM clears the TRIGj bit, where j = 0, 1, 2, when the hardware trigger j is detected.

Address: Base address + 8Ch offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0											HWSOC	HWINVC	HWOM	HWWRBUF	HWRSTCNT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			SWSOC	SWINVC	SWOM	SWWRBUF	SWRSTCNT	SYNCMODE	0	SWOC	INVC	0	CNTINC	0	HWTRIGMODE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_SYNCONF field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 HWSOC	Software output control synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the SWOCTRL register synchronization. 1 A hardware trigger activates the SWOCTRL register synchronization.
12 HWINVC	Inverting control synchronization is activated by a hardware trigger.

Table continues on the next page...

FTMx_SYNCONF field descriptions (continued)

Field	Description
	0 A hardware trigger does not activate the INVCTRL register synchronization. 1 A hardware trigger activates the INVCTRL register synchronization.
13 HWOM	Output mask synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the OUTMASK register synchronization. 1 A hardware trigger activates the OUTMASK register synchronization.
14 HWWRBUF	MOD, CNTIN, and CV registers synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate MOD, CNTIN, and CV registers synchronization. 1 A hardware trigger activates MOD, CNTIN, and CV registers synchronization.
15 HWRSTCNT	FTM counter synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the FTM counter synchronization. 1 A hardware trigger activates the FTM counter synchronization.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 SWSOC	Software output control synchronization is activated by the software trigger. 0 The software trigger does not activate the SWOCTRL register synchronization. 1 The software trigger activates the SWOCTRL register synchronization.
20 SWINVC	Inverting control synchronization is activated by the software trigger. 0 The software trigger does not activate the INVCTRL register synchronization. 1 The software trigger activates the INVCTRL register synchronization.
21 SWOM	Output mask synchronization is activated by the software trigger. 0 The software trigger does not activate the OUTMASK register synchronization. 1 The software trigger activates the OUTMASK register synchronization.
22 SWWRBUF	MOD, CNTIN, and CV registers synchronization is activated by the software trigger. 0 The software trigger does not activate MOD, CNTIN, and CV registers synchronization. 1 The software trigger activates MOD, CNTIN, and CV registers synchronization.
23 SWRSTCNT	FTM counter synchronization is activated by the software trigger. 0 The software trigger does not activate the FTM counter synchronization. 1 The software trigger activates the FTM counter synchronization.
24 SYNCMODE	Synchronization Mode Selects the PWM Synchronization mode. 0 Legacy PWM synchronization is selected. 1 Enhanced PWM synchronization is selected.
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 SWOC	SWOCTRL Register Synchronization 0 SWOCTRL register is updated with its buffer value at all rising edges of system clock. 1 SWOCTRL register is updated with its buffer value by the PWM synchronization.

Table continues on the next page...

FTMx_SYNCONF field descriptions (continued)

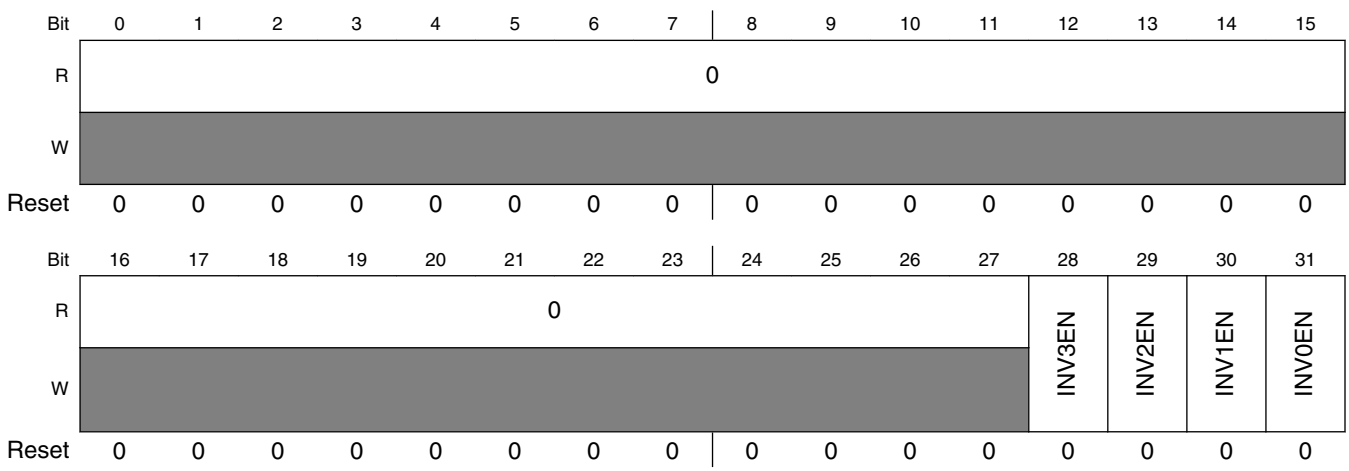
Field	Description
27 INVC	INVCTRL Register Synchronization 0 INVCTRL register is updated with its buffer value at all rising edges of system clock. 1 INVCTRL register is updated with its buffer value by the PWM synchronization.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 CNTINC	CNTIN Register Synchronization 0 CNTIN register is updated with its buffer value at all rising edges of system clock. 1 CNTIN register is updated with its buffer value by the PWM synchronization.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 HWTRIGMODE	Hardware Trigger Mode 0 FTM clears the TRIGj bit when the hardware trigger j is detected, where j = 0, 1,2. 1 FTM does not clear the TRIGj bit when the hardware trigger j is detected, where j = 0, 1,2.

12.2.3.23 FTM Inverting Control (FTMx_INVCTRL)

This register controls when the channel (n) output becomes the channel (n+1) output, and channel (n+1) output becomes the channel (n) output. Each INVmEN bit enables the inverting operation for the corresponding pair channels m.

This register has a write buffer. The INVmEN bit is updated by the INVCTRL register synchronization.

Address: Base address + 90h offset



FTMx_INVCTRL field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 INV3EN	Pair Channels 3 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.
29 INV2EN	Pair Channels 2 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.
30 INV1EN	Pair Channels 1 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.
31 INV0EN	Pair Channels 0 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.

12.2.3.24 FTM Software Output Control (FTMx_SWOCTRL)

This register enables software control of channel (n) output and defines the value forced to the channel (n) output:

- The CHnOC bits enable the control of the corresponding channel (n) output by software.
- The CHnOCV bits select the value that is forced at the corresponding channel (n) output.

This register has a write buffer. The fields are updated by the SWOCTRL register synchronization.

Address: Base address + 94h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH7OCV	CH6OCV	CH5OCV	CH4OCV	CH3OCV	CH2OCV	CH1OCV	CH0OCV	CH7OC	CH6OC	CH5OC	CH4OC	CH3OC	CH2OC	CH1OC	CH0OC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_SWOCTRL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CH7OCV	Channel 7 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
17 CH6OCV	Channel 6 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
18 CH5OCV	Channel 5 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
19 CH4OCV	Channel 4 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
20 CH3OCV	Channel 3 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
21 CH2OCV	Channel 2 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
22 CH1OCV	Channel 1 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
23 CH0OCV	Channel 0 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
24 CH7OC	Channel 7 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
25 CH6OC	Channel 6 Software Output Control Enable

Table continues on the next page...

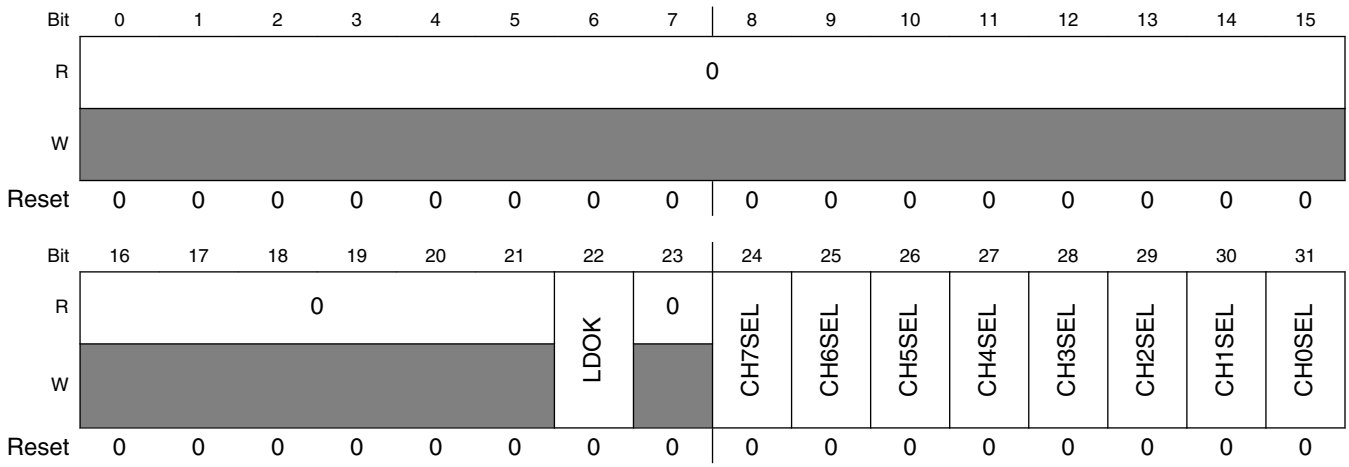
FTMx_SWOCTRL field descriptions (continued)

Field	Description
	0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
26 CH5OC	Channel 5 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
27 CH4OC	Channel 4 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
28 CH3OC	Channel 3 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
29 CH2OC	Channel 2 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
30 CH1OC	Channel 1 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
31 CH0OC	Channel 0 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.

12.2.3.25 FTM PWM Load (FTMx_PWMLOAD)

Enables the loading of the MOD, CNTIN, C(n)V, and C(n+1)V registers with the values of their write buffers when the FTM counter changes from the MOD register value to its next value or when a channel (j) match occurs. A match occurs for the channel (j) when FTM counter = C(j)V.

Address: Base address + 98h offset



FTMx_PWMLOAD field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 LDOK	Load Enable Enables the loading of the MOD, CNTIN, and CV registers with the values of their write buffers. 0 Loading updated values is disabled. 1 Loading updated values is enabled.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 CH7SEL	Channel 7 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
25 CH6SEL	Channel 6 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
26 CH5SEL	Channel 5 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.

Table continues on the next page...

FTMx_PWMLOAD field descriptions (continued)

Field	Description
27 CH4SEL	Channel 4 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
28 CH3SEL	Channel 3 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
29 CH2SEL	Channel 2 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
30 CH1SEL	Channel 1 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
31 CH0SEL	Channel 0 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.

12.2.4 Functional description

The notation used in this document to represent the counters and the generation of the signals is shown in the following figure.

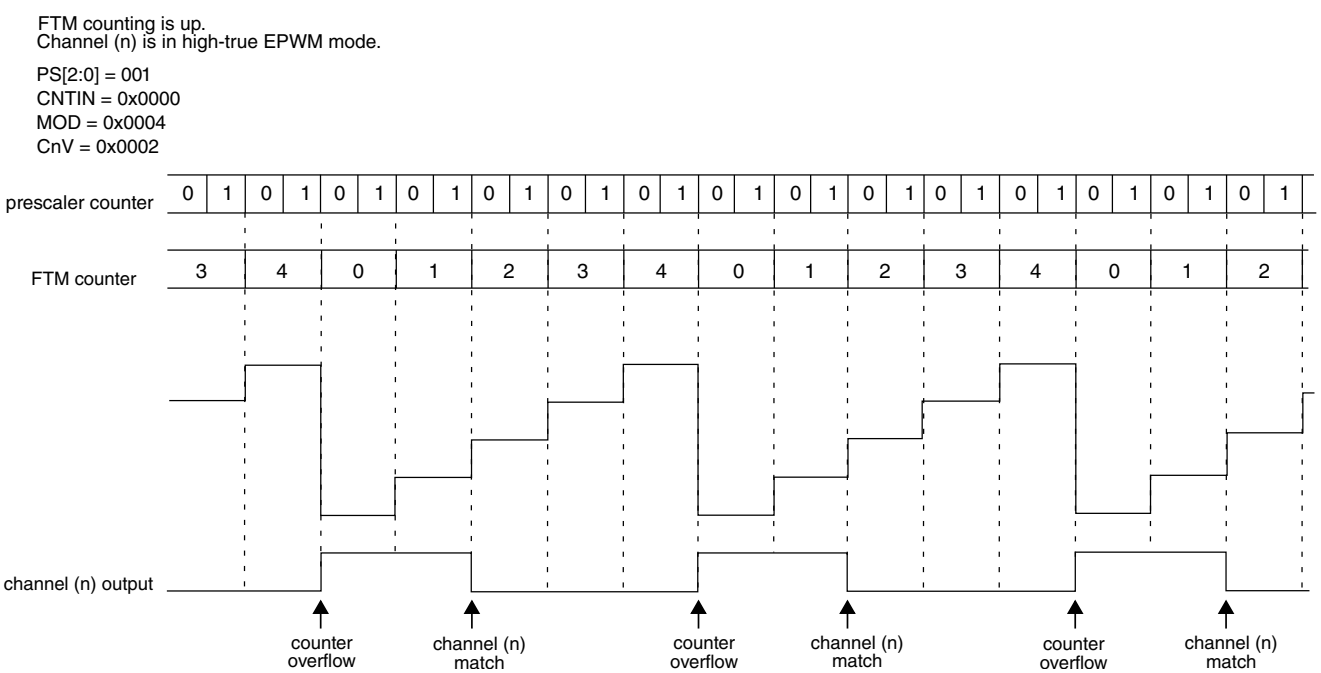


Figure 12-6. Notation used

12.2.4.1 Clock source

The FTM has only one clock domain: the system clock.

12.2.4.1.1 Counter clock source

The CLKS[1:0] bits in the SC register selects clock sources for the FTM counter or disables the FTM counter. After any chip reset, CLKS[1:0] = 0:0 so no clock source is selected.

The CLKS[1:0] bits may be read or written at any time. Disabling the FTM counter by writing 0:0 to the CLKS[1:0] bits does not affect the FTM counter value or other registers.

12.2.4.2 Prescaler

The selected counter clock source passes through a prescaler that is a 7-bit counter. The value of the prescaler is selected by the PS[2:0] bits. The following figure shows an example of the prescaler counter and FTM counter.

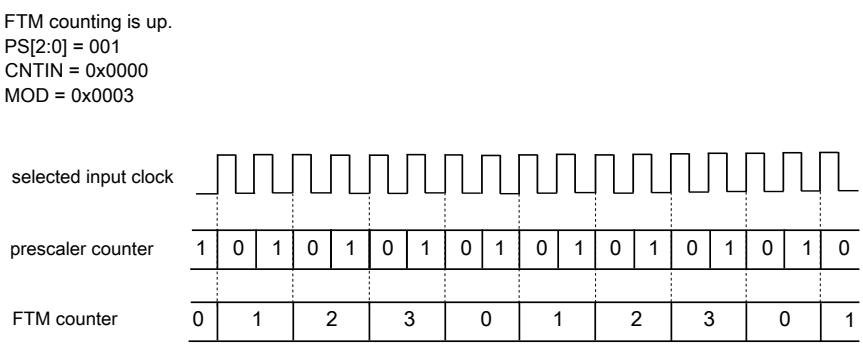


Figure 12-7. Example of the prescaler counter

12.2.4.3 Counter

The FTM has a 16-bit counter that is used by the channels either for input or output modes. The FTM counter clock is the selected clock divided by the prescaler.

The FTM counter has these modes of operation:

- Up counting
- Up-down counting
- Quadrature Decoder mode

12.2.4.3.1 Up counting

Up counting is selected when:

- QUADEN = 0, and
- CPWMS = 0

CNTIN defines the starting value of the count and MOD defines the final value of the count, see the following figure. The value of CNTIN is loaded into the FTM counter, and the counter increments until the value of MOD is reached, at which point the counter is reloaded with the value of CNTIN.

The FTM period when using up counting is (MOD – CNTIN + 0x0001) × period of the FTM counter clock.

The TOF bit is set when the FTM counter changes from MOD to CNTIN.

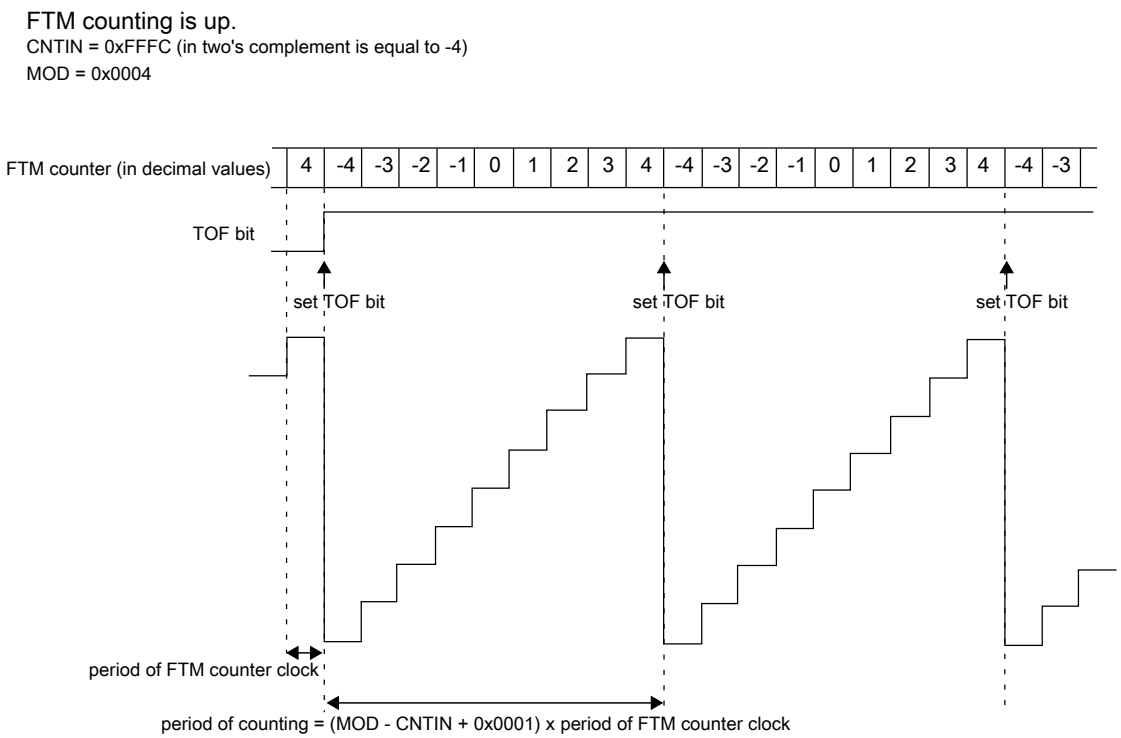


Figure 12-8. Example of FTM up and signed counting

Table 12-6. FTM counting based on CNTIN value

When	Then
CNTIN = 0x0000	The FTM counting is equivalent to TPM up counting, that is, up and unsigned counting. See the following figure.
CNTIN[15] = 1	The initial value of the FTM counter is a negative number in two's complement, so the FTM counting is up and signed.

Table continues on the next page...

Table 12-6. FTM counting based on CNTIN value (continued)

When	Then
CNTIN[15] = 0 and CNTIN ≠ 0x0000	The initial value of the FTM counter is a positive number, so the FTM counting is up and unsigned.

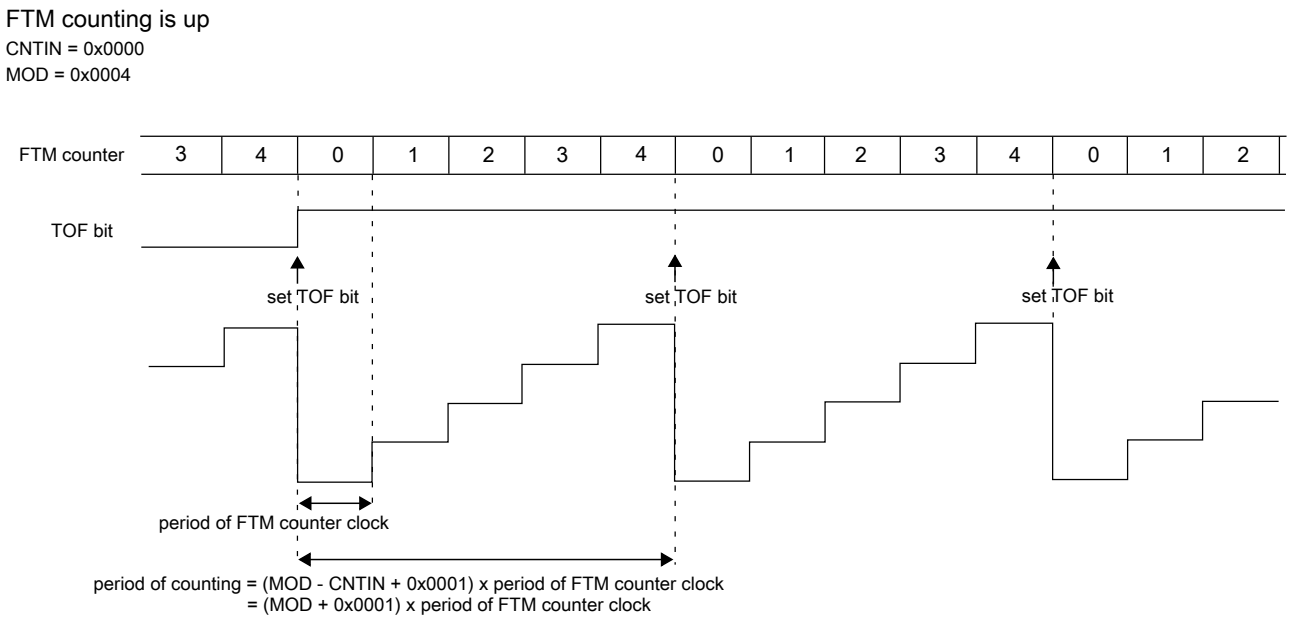


Figure 12-9. Example of FTM up counting with CNTIN = 0x0000

Note

- FTM operation is only valid when the value of the CNTIN register is less than the value of the MOD register, either in the unsigned counting or signed counting. It is the responsibility of the software to ensure that the values in the CNTIN and MOD registers meet this requirement. Any values of CNTIN and MOD that do not satisfy this criteria can result in unpredictable behavior.
- MOD = CNTIN is a redundant condition. In this case, the FTM counter is always equal to MOD and the TOF bit is set in each rising edge of the FTM counter clock.

- When MOD = 0x0000, CNTIN = 0x0000, for example after reset, and FTMEN = 1, the FTM counter remains stopped at 0x0000 until a non-zero value is written into the MOD or CNTIN registers.
- Setting CNTIN to be greater than the value of MOD is not recommended as this unusual setting may make the FTM operation difficult to comprehend. However, there is no restriction on this configuration, and an example is shown in the following figure.

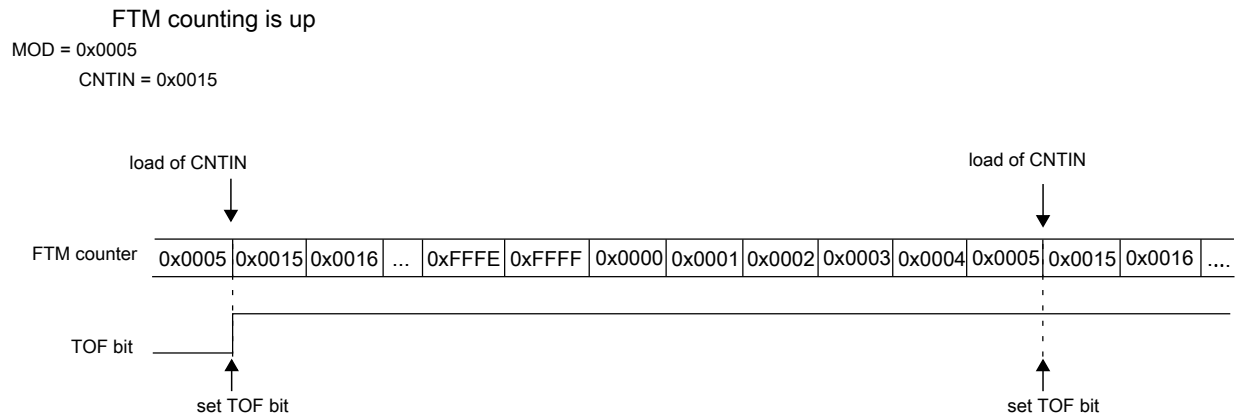


Figure 12-10. Example of up counting when the value of CNTIN is greater than the value of MOD

12.2.4.3.2 Up-down counting

Up-down counting is selected when:

- QUADEN = 0, and
- CPWMS = 1

CNTIN defines the starting value of the count and MOD defines the final value of the count. The value of CNTIN is loaded into the FTM counter, and the counter increments until the value of MOD is reached, at which point the counter is decremented until it returns to the value of CNTIN and the up-down counting restarts.

The FTM period when using up-down counting is 2 × (MOD – CNTIN) × period of the FTM counter clock.

The TOF bit is set when the FTM counter changes from MOD to (MOD – 1).

If (CNTIN = 0x0000), the FTM counting is equivalent to TPM up-down counting, that is, up-down and unsigned counting. See the following figure.

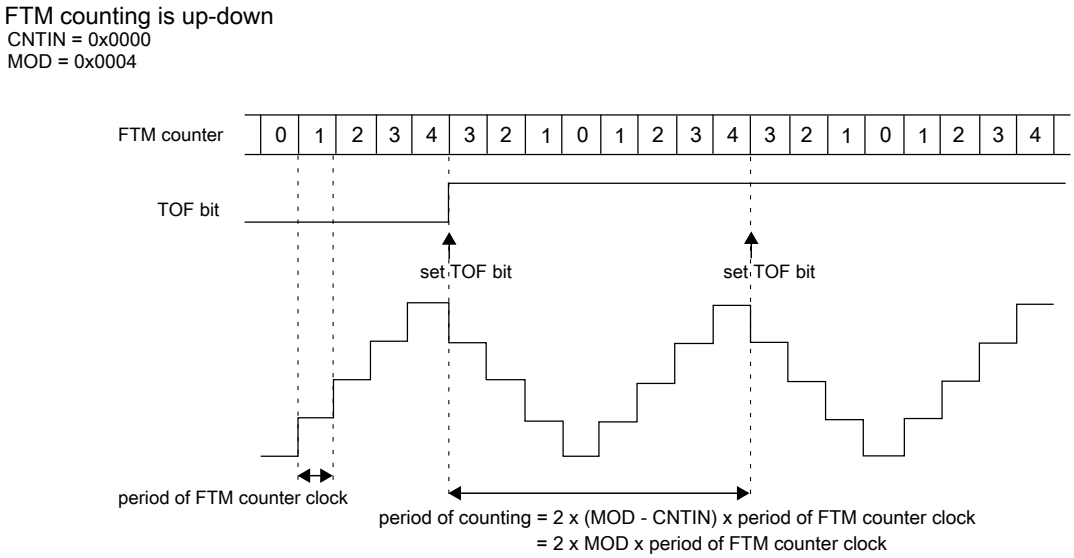


Figure 12-11. Example of up-down counting when CNTIN = 0x0000

Note

When CNTIN is different from zero in the up-down counting, a valid CPWM signal is generated:

- if CnV > CNTIN, or
- if CnV = 0 or if CnV[15] = 1. In this case, 0% CPWM is generated.

12.2.4.3.3 Free running counter

If (FTMEN = 0) and (MOD = 0x0000 or MOD = 0xFFFF), the FTM counter is a free running counter. In this case, the FTM counter runs free from 0x0000 through 0xFFFF and the TOF bit is set when the FTM counter changes from 0xFFFF to 0x0000. See the following figure.

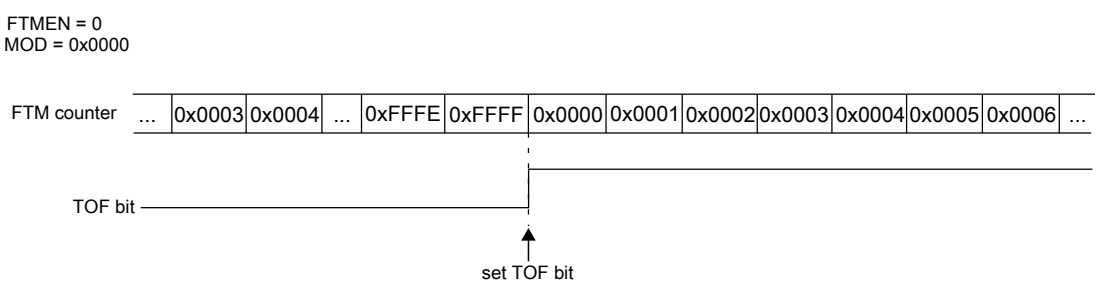


Figure 12-12. Example when the FTM counter is free running

The FTM counter is also a free running counter when:

- FTMEN = 1

- QUADEN = 0
- CPWMS = 0
- CNTIN = 0x0000, and
- MOD = 0xFFFF

12.2.4.3.4 Counter reset

Any one of the following cases resets the FTM counter to the value in the CNTIN register and the channels output to its initial value, except for channels in Output Compare mode.

- Any write to CNT.
- FTM counter synchronization.
- A channel in Input Capture mode with ICRST = 1 (FTM Counter Reset in Input Capture Mode).

12.2.4.3.5 When the TOF bit is set

The NUMTOF[4:0] bits define the number of times that the FTM counter overflow should occur before the TOF bit to be set. If NUMTOF[4:0] = 0x00, then the TOF bit is set at each FTM counter overflow.

Initialize the FTM counter, by writing to CNT, after writing to the NUMTOF[4:0] bits to avoid confusion about when the first counter overflow will occur.

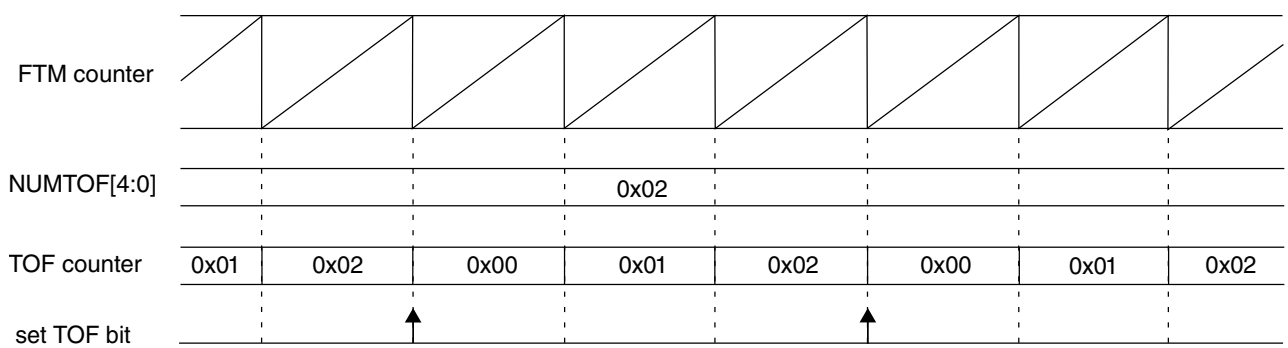


Figure 12-13. Periodic TOF when NUMTOF = 0x02

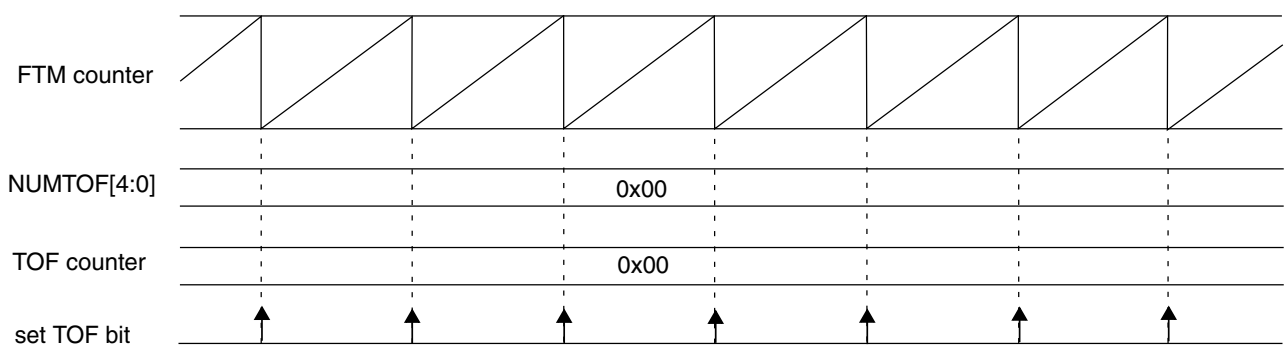


Figure 12-14. Periodic TOF when NUMTOF = 0x00

12.2.4.4 Input Capture mode

The Input Capture mode is selected when:

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0
- MSnB:MSnA = 0:0, and
- ELSnB:ELSnA ≠ 0:0

When a selected edge occurs on the channel input, the current value of the FTM counter is captured into the CnV register, at the same time the CHnF bit is set and the channel interrupt is generated if enabled by CHnIE = 1. See the following figure.

When a channel is configured for input capture, the FTMxCHn pin is an edge-sensitive input. ELSnB:ELSnA control bits determine which edge, falling or rising, triggers input-capture event. Note that the maximum frequency for the channel input signal to be detected correctly is system clock divided by 4, which is required to meet Nyquist criteria for signal sampling.

Writes to the CnV register is ignored in Input Capture mode.

While in BDM, the input capture function works as configured. When a selected edge event occurs, the FTM counter value, which is frozen because of BDM, is captured into the CnV register and the CHnF bit is set.

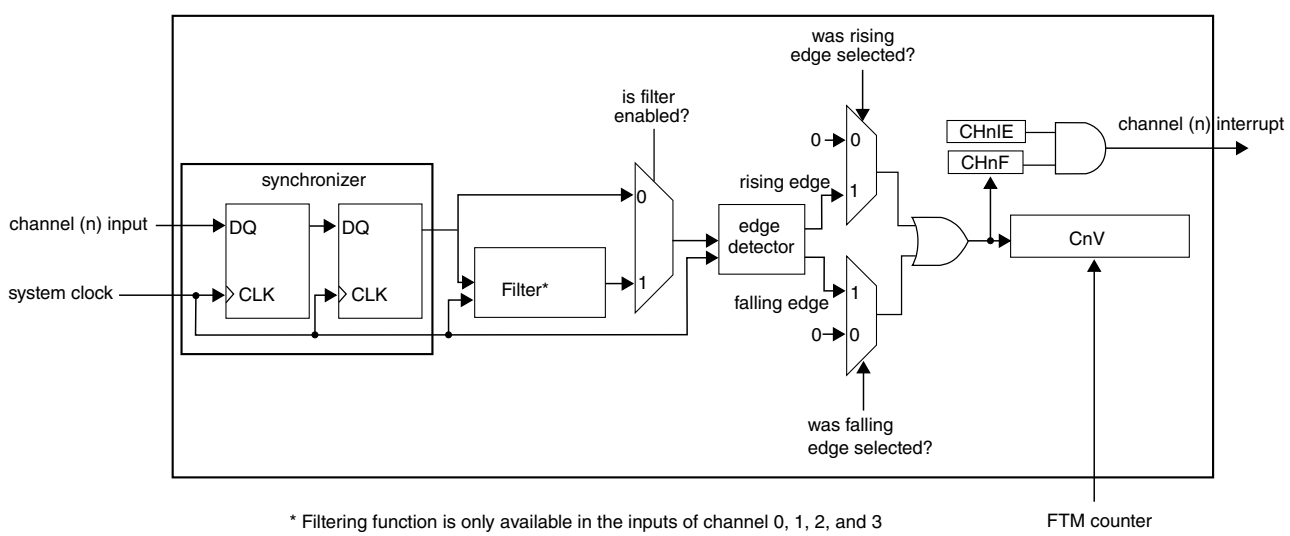


Figure 12-15. Input Capture mode

If the channel input does not have a filter enabled, then the input signal is always delayed 3 rising edges of the system clock, that is, two rising edges to the synchronizer plus one more rising edge to the edge detector. In other words, the CHnF bit is set on the third rising edge of the system clock after a valid edge occurs on the channel input.

12.2.4.4.1 Filter for Input Capture mode

The filter function is only available on channels 0, 1, 2, and 3.

First, the input signal is synchronized by the system clock. Following synchronization, the input signal enters the filter block. See the following figure.

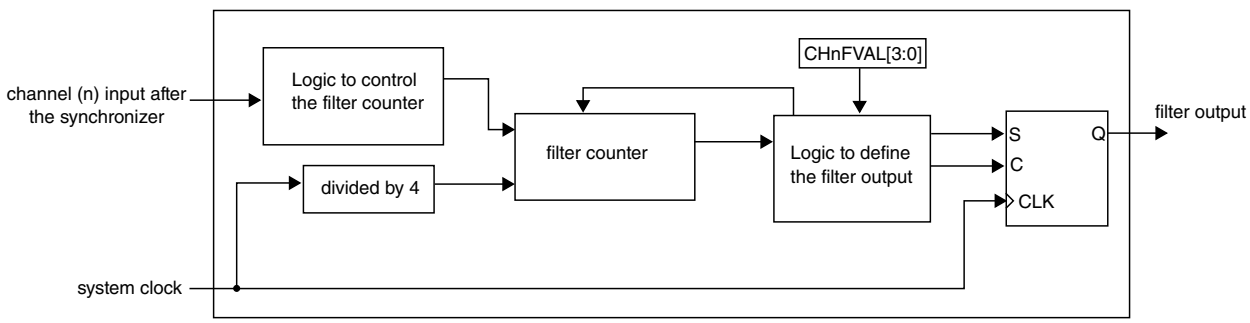


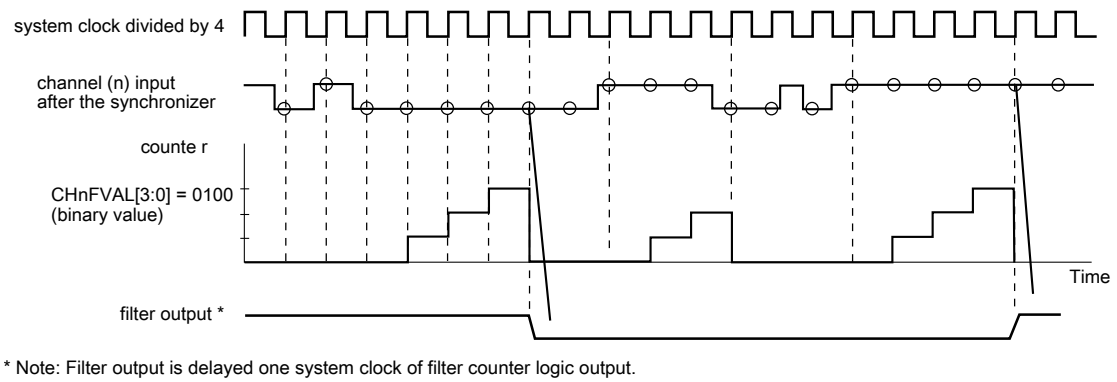
Figure 12-16. Channel input filter

When there is a state change in the input signal, the counter is reset and starts counting up. As long as the new state is stable on the input, the counter continues to increment. When the counter is equal to CHnFVAL[3:0], the state change of the input signal is validated. It is then transmitted as a pulse edge to the edge detector.

If the opposite edge appears on the input signal before it can be validated, the counter is reset. At the next input transition, the counter starts counting again. Any pulse that is shorter than the minimum value selected by CHnFVAL[3:0] (× 4 system clocks) is regarded as a glitch and is not passed on to the edge detector. A timing diagram of the input filter is shown in the following figure.

The filter function is disabled when CHnFVAL[3:0] bits are zero. In this case, the input signal is delayed 3 rising edges of the system clock. If (CHnFVAL[3:0] ≠ 0000), then the input signal is delayed by the minimum pulse width (CHnFVAL[3:0] × 4 system clocks) plus a further 4 rising edges of the system clock: two rising edges to the synchronizer, one rising edge to the filter output, plus one more to the edge detector. In other words, CHnF is set (4 + 4 × CHnFVAL[3:0]) system clock periods after a valid edge occurs on the channel input.

The clock for the counter in the channel input filter is the system clock divided by 4.



* Note: Filter output is delayed one system clock of filter counter logic output.

Figure 12-17. Channel input filter example

The figure below shows an example of input capture with filter enabled and the delay added by each part of the input capture logic. Note that the input signal is delayed only by the synchronizer and edge dector logic if the filter is disabled.

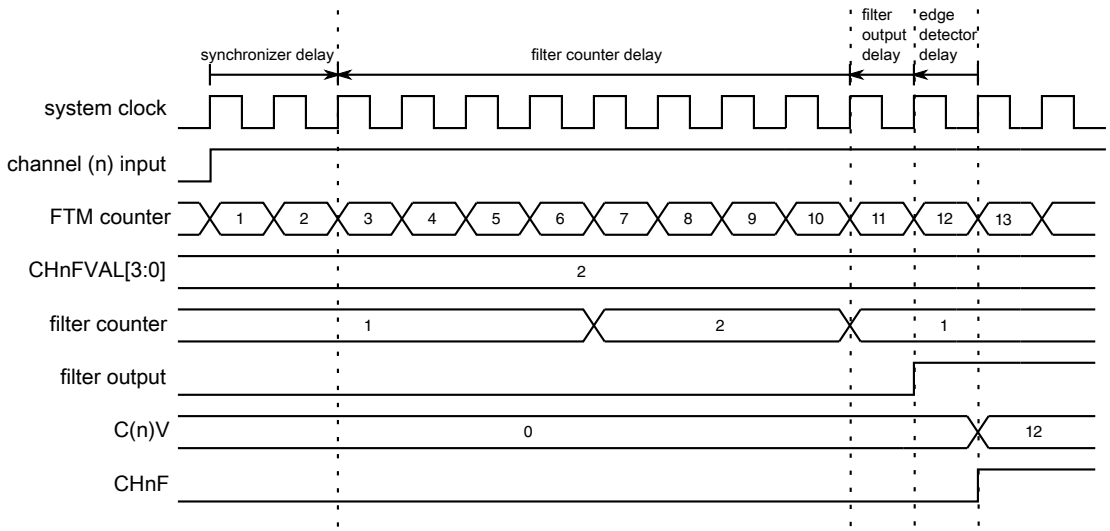


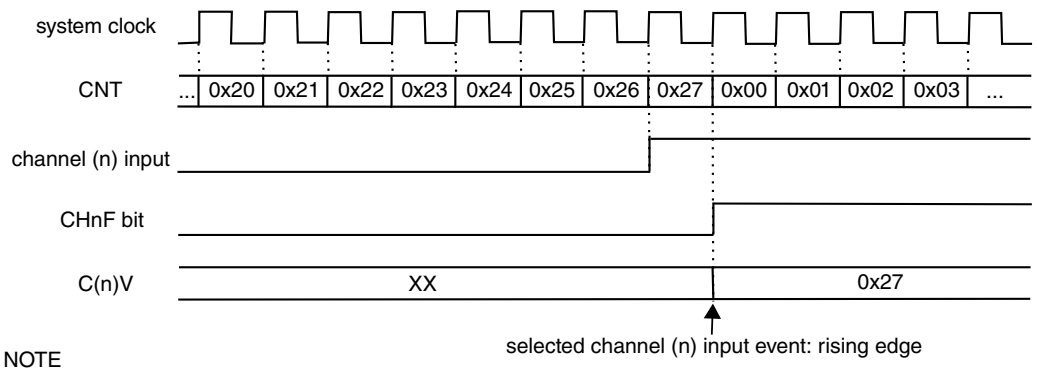
Figure 12-18. Input capture example

12.2.4.4.2 FTM Counter Reset in Input Capture Mode

If the channel (n) is in input capture mode and FTMx_CnSC [ICRST = 1], then when the selected input capture event occurs in the channel (n) input signal, the current value of the FTM counter is captured into the CnV register, the CHnF bit is set, the channel (n) interrupt is generated (if CHnIE = 1) and the FTM counter is reset to the CNTIN register value.

This allows the FTM to measure a period/pulse being applied to FTM_CHn (counts of the FTM clock input) without having to implement a subtraction calculation in software subsequent to the event occurring.

The figure below shows the FTM counter reset when the selected input capture event is detected in a channel in input capture mode with ICRST = 1.



NOTE
Channel (n) input after its synchronizer and filter
MOD = 0xFFFF
CNTIN = 0x0000
PS[2:0] = 3'b000
ICRST = 1'b1

Figure 12-19. Example of the Input Capture mode with ICRST = 1

NOTE

- It is expected that the ICRST bit be set only when the channel is in input capture mode.
- In this case, if the FTM counter is reset, then the prescaler counter (Prescaler) and the TOF counter (When the TOF bit is set) also are reset.

12.2.4.5 Output Compare mode

The Output Compare mode is selected when:

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0, and
- MSnB:MSnA = 0:1

In Output Compare mode, the FTM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CnV register of an output compare channel, the channel (n) output can be set, cleared, or toggled.

When a channel is initially configured to Toggle mode, the previous value of the channel output is held until the first output compare event occurs.

The CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1 at the channel (n) match (FTM counter = CnV).

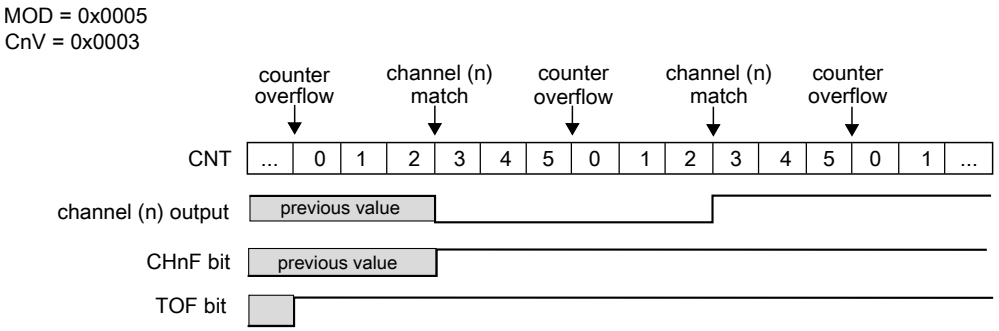


Figure 12-20. Example of the Output Compare mode when the match toggles the channel output

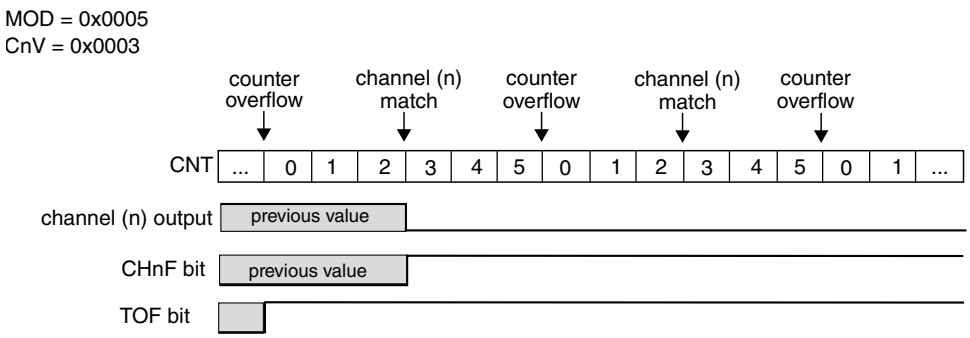


Figure 12-21. Example of the Output Compare mode when the match clears the channel output

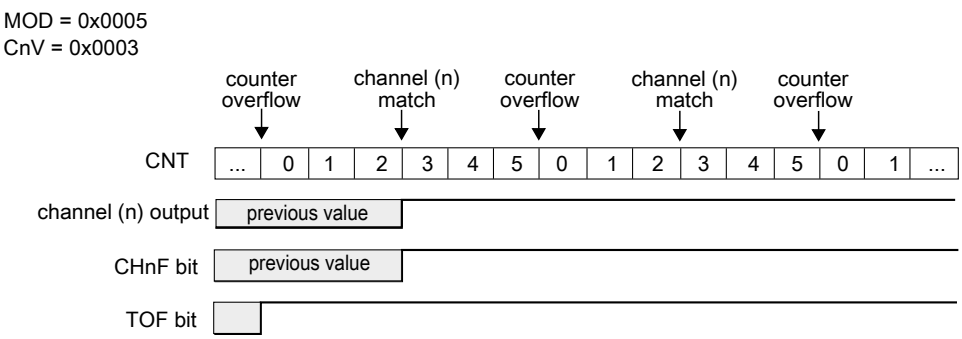


Figure 12-22. Example of the Output Compare mode when the match sets the channel output

If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1, however the channel (n) output is not modified and controlled by FTM.

12.2.4.6 Edge-Aligned PWM (EPWM) mode

The Edge-Aligned mode is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0, and
- MSnB = 1

The EPWM period is determined by (MOD − CNTIN + 0x0001) and the pulse width (duty cycle) is determined by (CnV − CNTIN).

The CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1 at the channel (n) match (FTM counter = CnV), that is, at the end of the pulse width.

This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within an FTM.

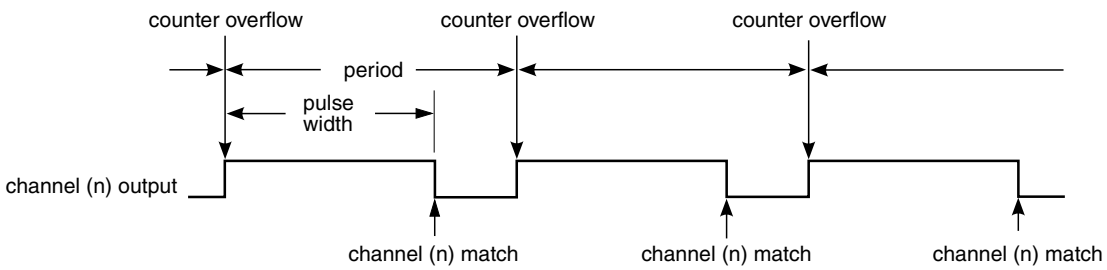


Figure 12-23. EPWM period and pulse width with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1, however the channel (n) output is not controlled by FTM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the counter overflow when the CNTIN register value is loaded into the FTM counter, and it is forced low at the channel (n) match (FTM counter = CnV). See the following figure.

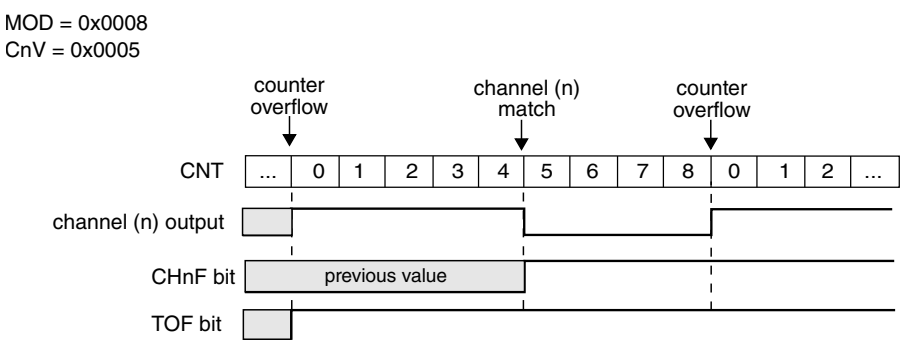


Figure 12-24. EPWM signal with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the counter overflow when the CNTIN register value is loaded into the FTM counter, and it is forced high at the channel (n) match (FTM counter = CnV). See the following figure.

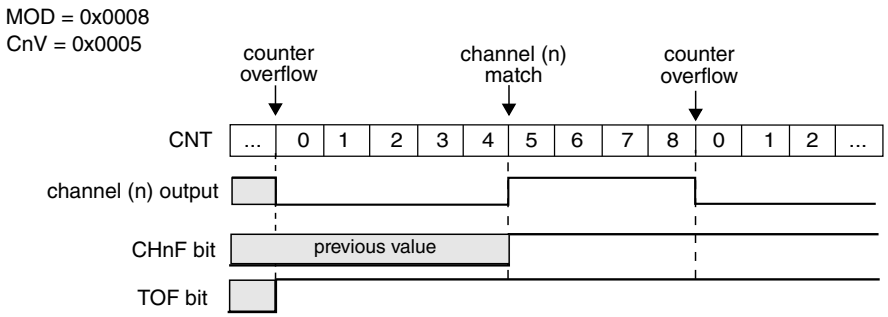


Figure 12-25. EPWM signal with ELSnB:ELSnA = X:1

If (CnV = 0x0000), then the channel (n) output is a 0% duty cycle EPWM signal and CHnF bit is not set even when there is the channel (n) match.

If (CnV > MOD), then the channel (n) output is a 100% duty cycle EPWM signal and CHnF bit is not set.Therefore, MOD must be less than 0xFFFF in order to get a 100% duty cycle EPWM signal.

Note

When CNTIN is different from zero the following EPWM signals can be generated:

- 0% EPWM signal if CnV = CNTIN,
- EPWM signal between 0% and 100% if CNTIN < CnV <= MOD,
- 100% EPWM signal when CNTIN > CnV or CnV > MOD.

12.2.4.7 Center-Aligned PWM (CPWM) mode

The Center-Aligned mode is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMBINE = 0, and
- CPWMS = 1

The CPWM pulse width (duty cycle) is determined by 2 × (CnV − CNTIN) and the period is determined by 2 × (MOD − CNTIN). See the following figure. MOD must be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results.

In the CPWM mode, the FTM counter counts up until it reaches MOD and then counts down until it reaches CNTIN.

The CHnF bit is set and channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = CnV) when the FTM counting is down (at the begin of the pulse width) and when the FTM counting is up (at the end of the pulse width).

This type of PWM signal is called center-aligned because the pulse width centers for all channels are aligned with the value of CNTIN.

The other channel modes are not compatible with the up-down counter (CPWMS = 1). Therefore, all FTM channels must be used in CPWM mode when (CPWMS = 1).

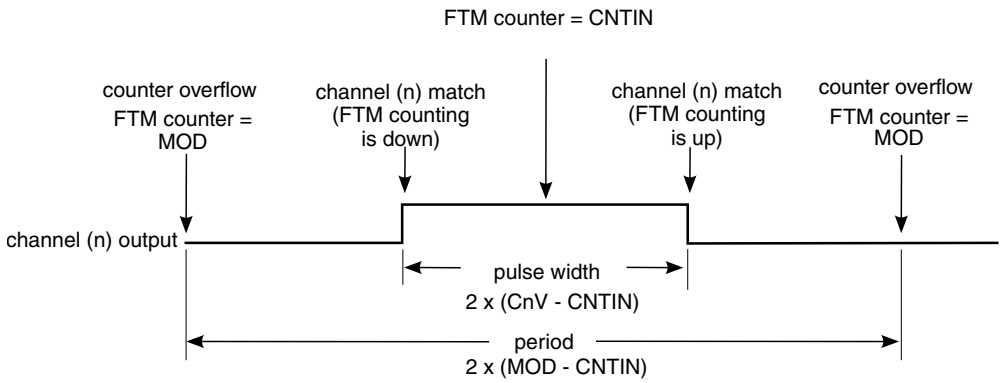


Figure 12-26. CPWM period and pulse width with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = 0:0) when the FTM counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1), however the channel (n) output is not controlled by FTM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the channel (n) match (FTM counter = CnV) when counting down, and it is forced low at the channel (n) match when counting up. See the following figure.

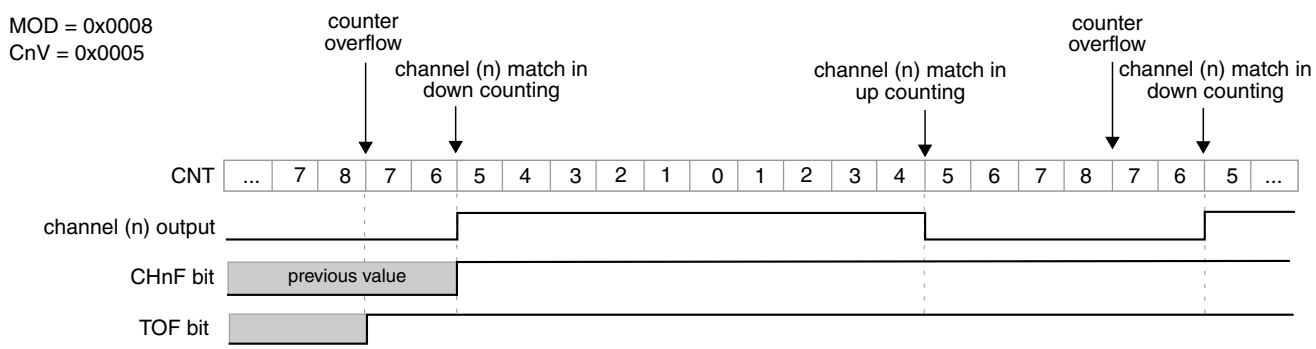


Figure 12-27. CPWM signal with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the channel (n) match (FTM counter = CnV) when counting down, and it is forced high at the channel (n) match when counting up. See the following figure.

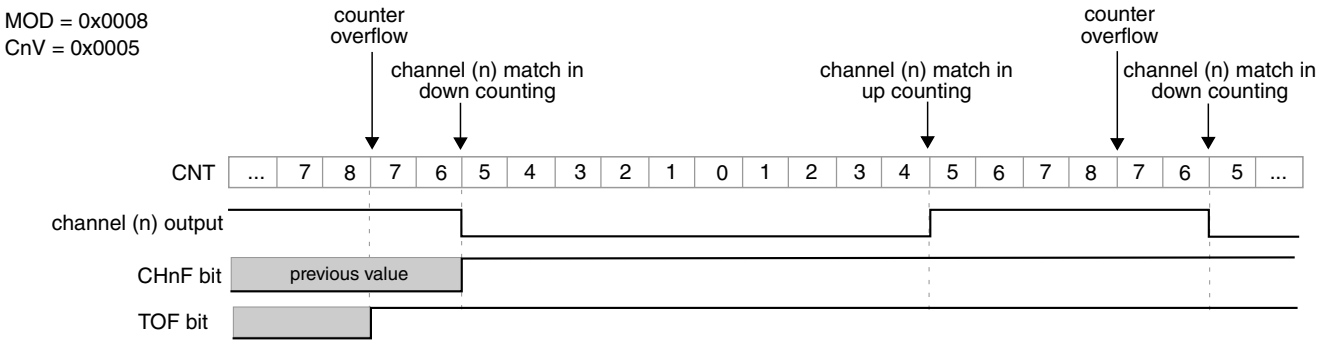


Figure 12-28. CPWM signal with ELSnB:ELSnA = X:1

If (CnV = 0x0000) or CnV is a negative value, that is (CnV[15] = 1), then the channel (n) output is a 0% duty cycle CPWM signal and CHnF bit is not set even when there is the channel (n) match.

If CnV is a positive value, that is (CnV[15] = 0), (CnV ≥ MOD), and (MOD ≠ 0x0000), then the channel (n) output is a 100% duty cycle CPWM signal and CHnF bit is not set even when there is the channel (n) match. This implies that the usable range of periods set by MOD is 0x0001 through 0x7FFE, 0x7FFF if you do not need to generate a 100% duty cycle CPWM signal. This is not a significant limitation because the resulting period is much longer than required for normal applications.

The CPWM mode must not be used when the FTM counter is a free running counter.

12.2.4.8 Combine mode

The Combine mode is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMBINE = 1, and
- CPWMS = 0

In Combine mode, an even channel (n) and adjacent odd channel (n+1) are combined to generate a PWM signal in the channel (n) output.

In the Combine mode, the PWM period is determined by (MOD − CNTIN + 0x0001) and the PWM pulse width (duty cycle) is determined by (|C(n+1)V − C(n)V|).

The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = C(n)V). The CH(n+1)F bit is set and the channel (n +1) interrupt is generated, if CH(n+1)IE = 1, at the channel (n+1) match (FTM counter = C(n+1)V).

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced low at the beginning of the period (FTM counter = CNTIN) and at the channel (n+1) match (FTM counter = C(n +1)V). It is forced high at the channel (n) match (FTM counter = C(n)V). See the following figure.

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced high at the beginning of the period (FTM counter = CNTIN) and at the channel (n+1) match (FTM counter = C(n +1)V). It is forced low at the channel (n) match (FTM counter = C(n)V). See the following figure.

In Combine mode, the ELS(n+1)B and ELS(n+1)A bits are not used in the generation of the channels (n) and (n+1) output. However, if (ELSnB:ELSnA = 0:0) then the channel (n) output is not controlled by FTM, and if (ELS(n+1)B:ELS(n+1)A = 0:0) then the channel (n+1) output is not controlled by FTM.

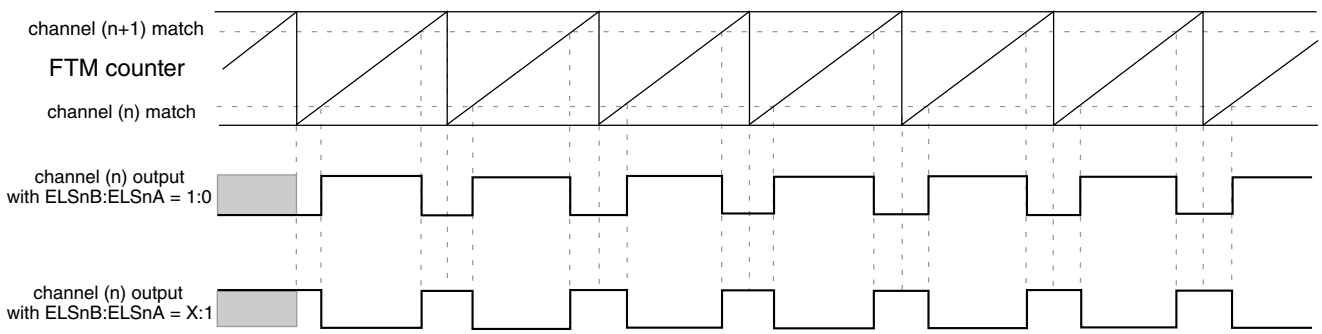


Figure 12-29. Combine mode

The following figures illustrate the PWM signals generation using Combine mode.

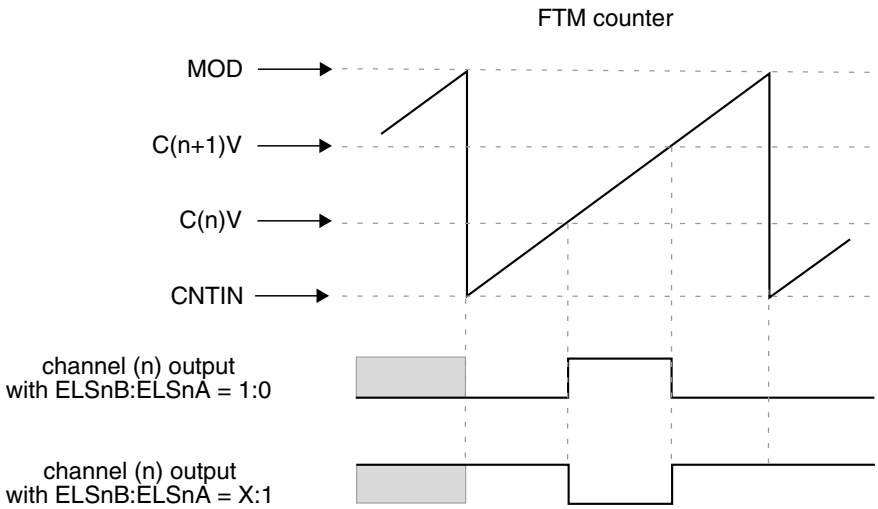


Figure 12-30. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V < C(n+1)V)

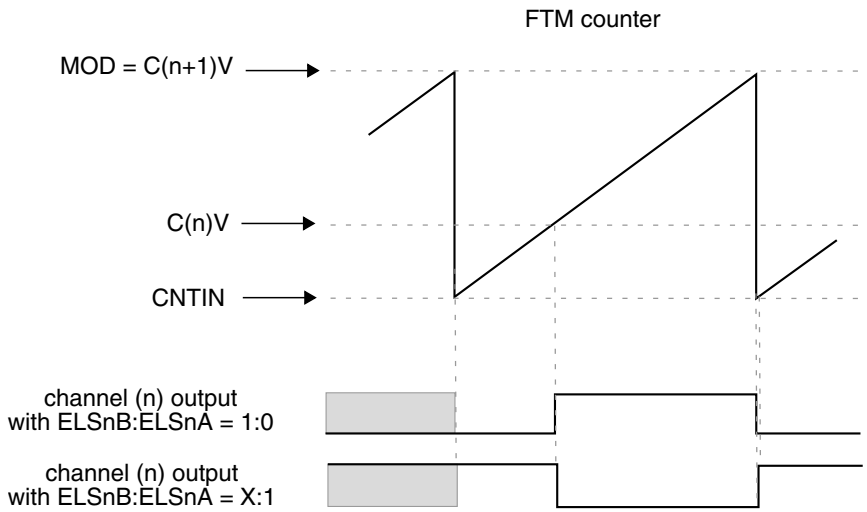


Figure 12-31. Channel (n) output if (CNTIN < C(n)V < MOD) and (C(n+1)V = MOD)

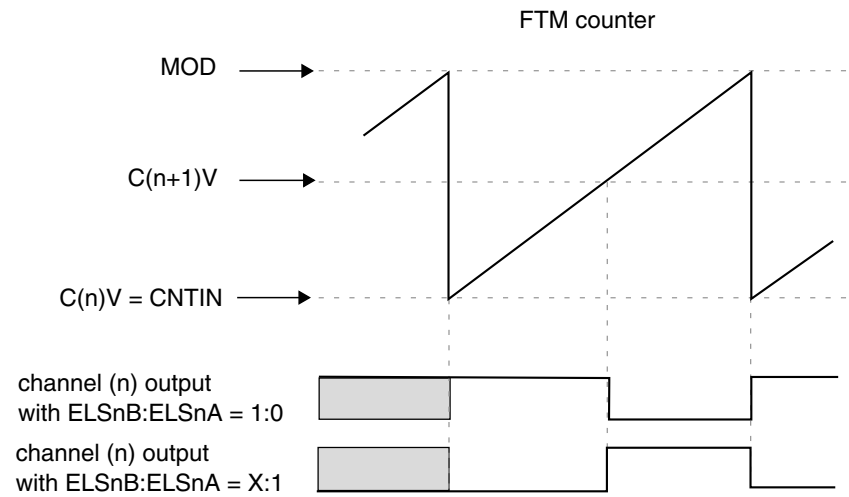


Figure 12-32. Channel (n) output if (C(n)V = CNTIN) and (CNTIN < C(n+1)V < MOD)

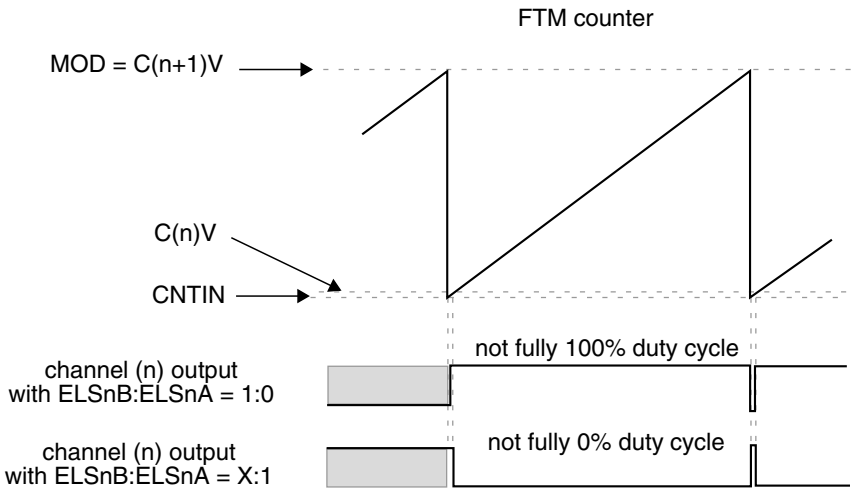


Figure 12-33. Channel (n) output if (CNTIN < C(n)V < MOD) and (C(n)V is Almost Equal to CNTIN) and (C(n+1)V = MOD)

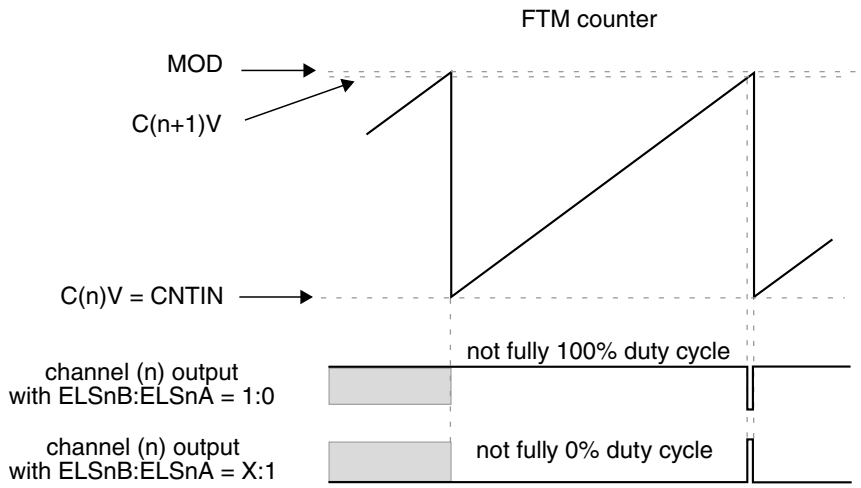


Figure 12-34. Channel (n) output if (C(n)V = CNTIN) and (CNTIN < C(n+1)V < MOD) and (C(n+1)V is Almost Equal to MOD)

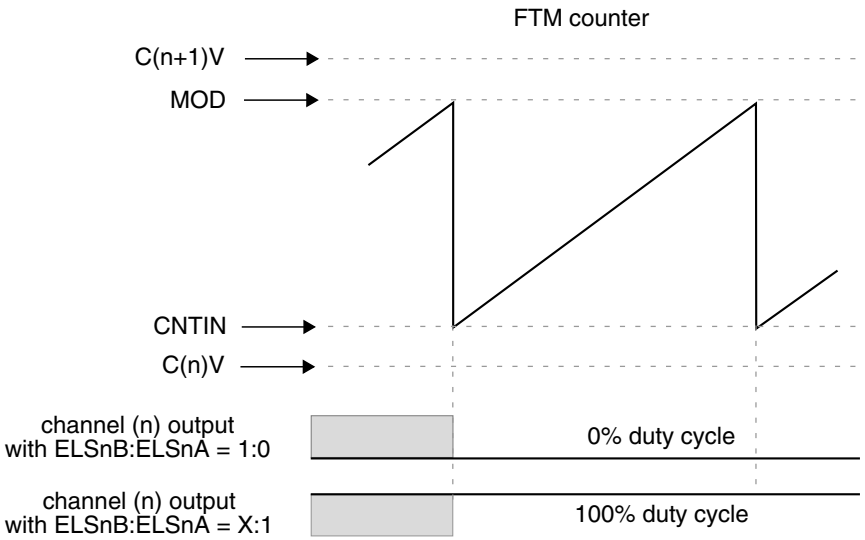


Figure 12-35. Channel (n) output if C(n)V and C(n+1)V are not between CNTIN and MOD

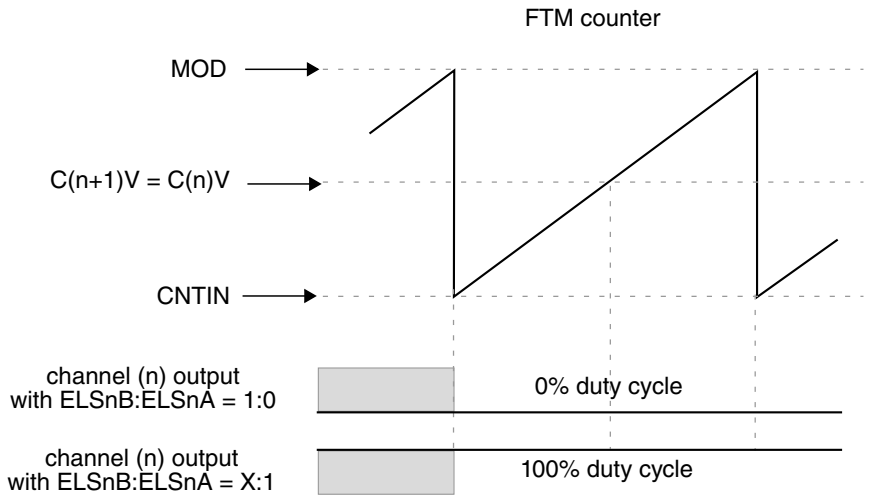


Figure 12-36. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V = C(n+1)V)

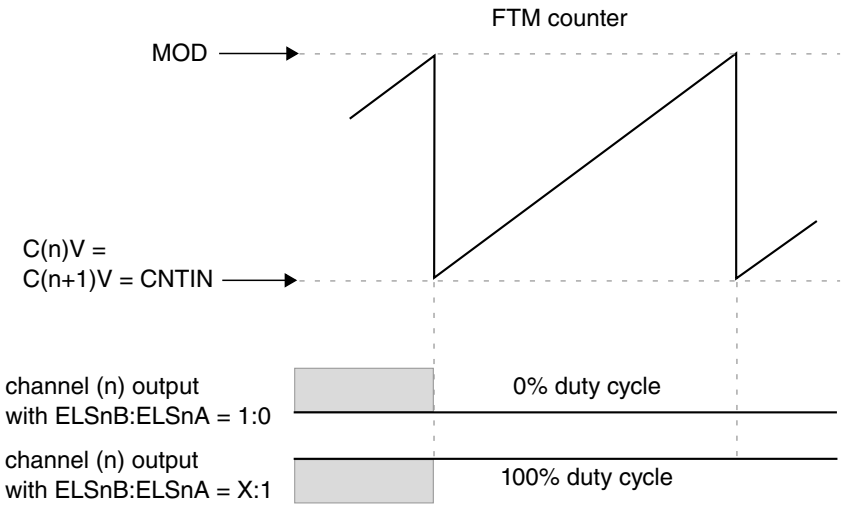


Figure 12-37. Channel (n) output if (C(n)V = C(n+1)V = CNTIN)

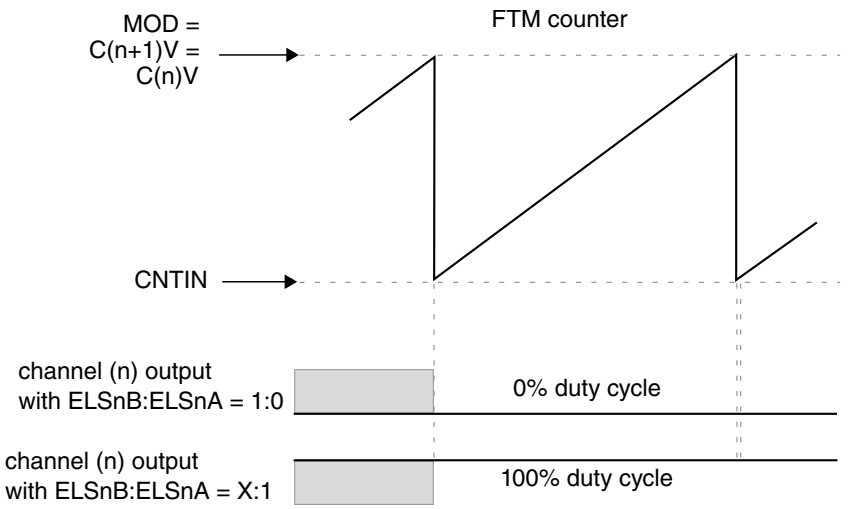


Figure 12-38. Channel (n) output if (C(n)V = C(n+1)V = MOD)

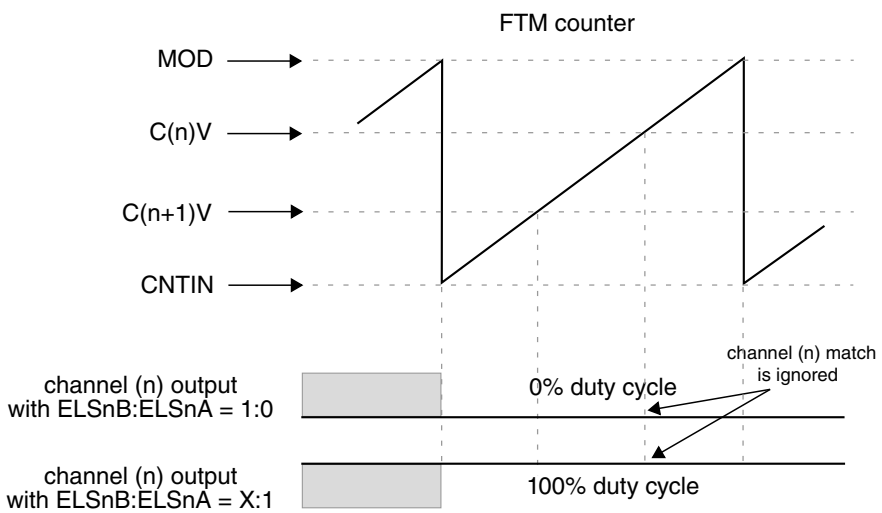


Figure 12-39. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V > C(n+1)V)

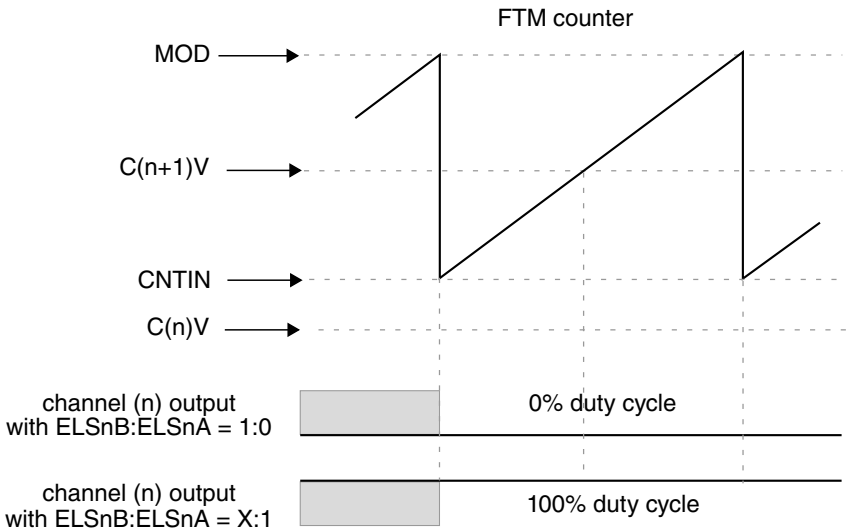


Figure 12-40. Channel (n) output if (C(n)V < CNTIN) and (CNTIN < C(n+1)V < MOD)

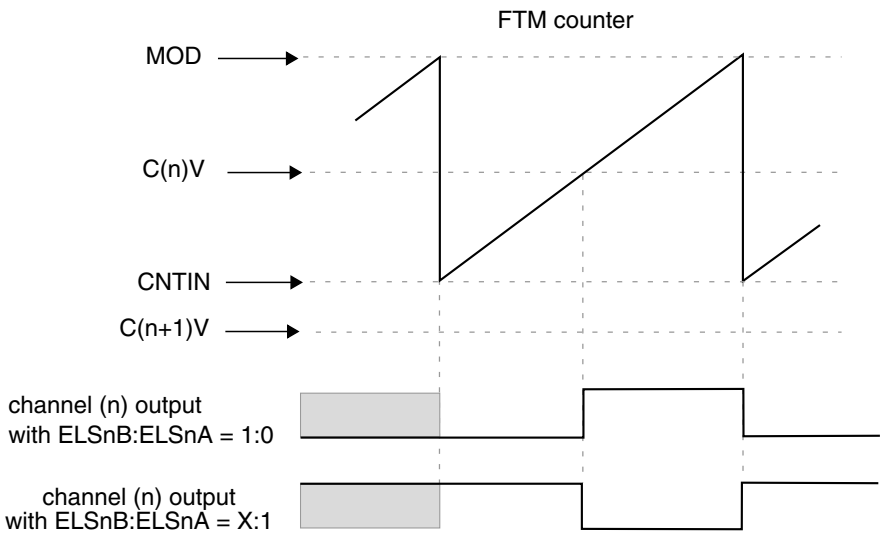


Figure 12-41. Channel (n) output if (C(n+1)V < CNTIN) and (CNTIN < C(n)V < MOD)

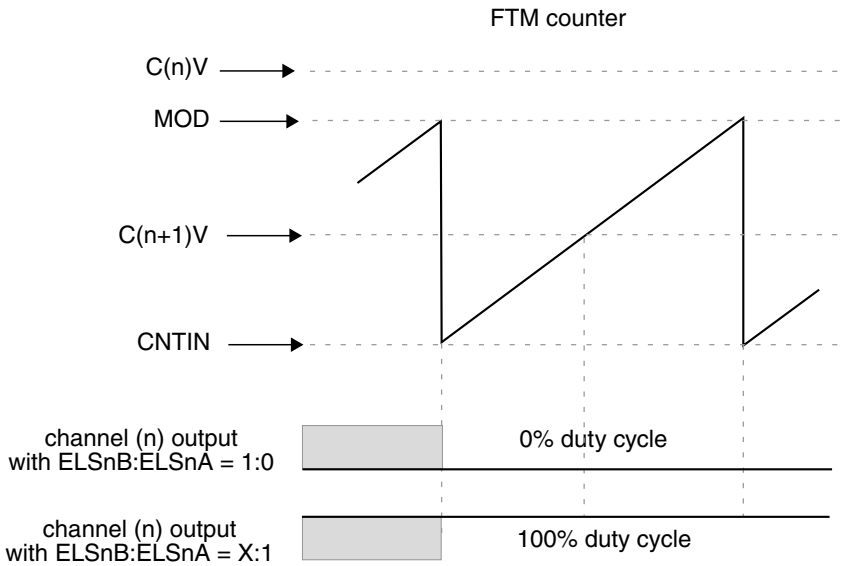


Figure 12-42. Channel (n) output if (C(n)V > MOD) and (CNTIN < C(n+1)V < MOD)

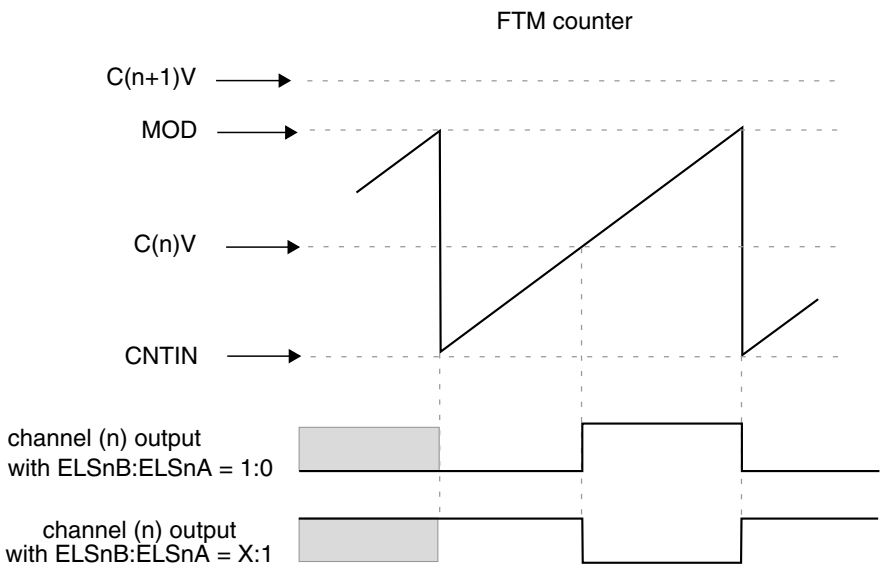


Figure 12-43. Channel (n) output if (C(n+1)V > MOD) and (CNTIN < C(n)V < MOD)

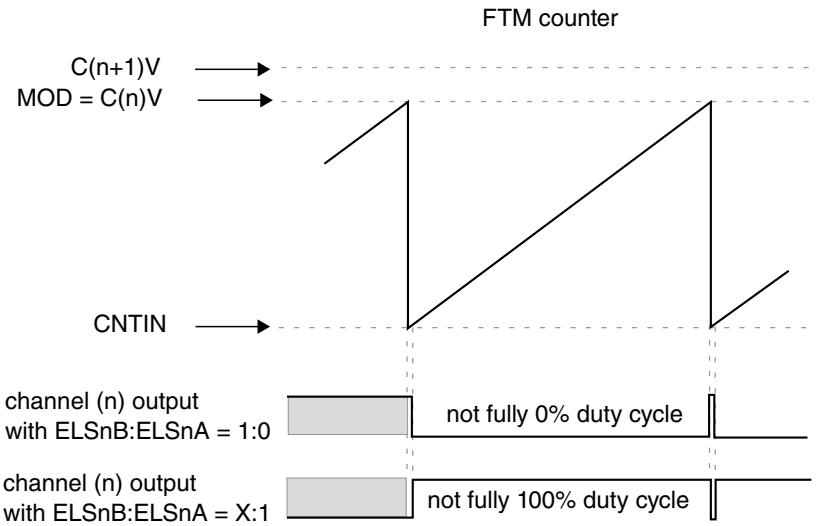


Figure 12-44. Channel (n) output if (C(n+1)V > MOD) and (CNTIN < C(n)V = MOD)

12.2.4.8.1 Asymmetrical PWM

In Combine mode, the control of the PWM signal first edge, when the channel (n) match occurs, that is, FTM counter = C(n)V, is independent of the control of the PWM signal second edge, when the channel (n+1) match occurs, that is, FTM counter = C(n+1)V. So, Combine mode allows the generation of asymmetrical PWM signals.

12.2.4.9 Complementary mode

The Complementary mode is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMP = 1

In Complementary mode, the channel (n+1) output is the inverse of the channel (n) output.

So, the channel (n+1) output is the same as the channel (n) output when:

- QUADEN = 0
- DECAPEN = 0
- COMP = 0

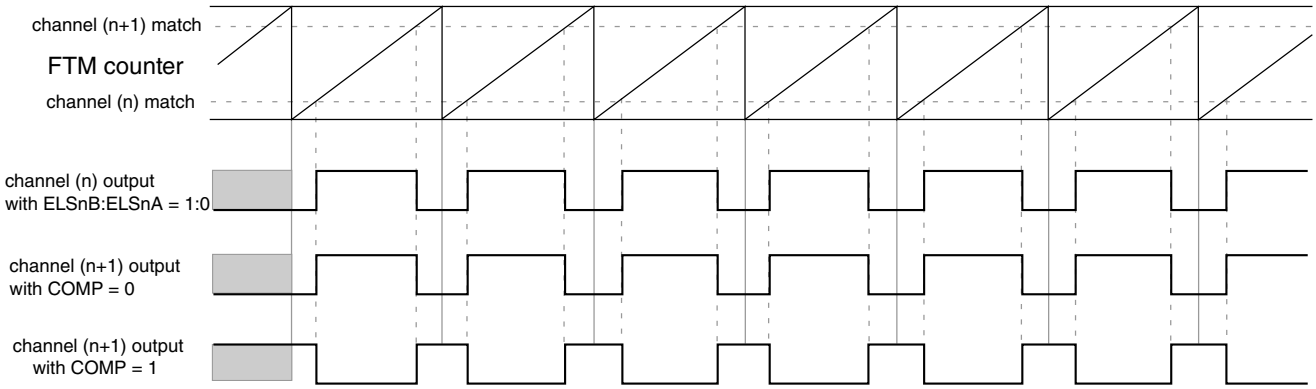


Figure 12-45. Channel (n+1) output in Complementary mode with (ELSnB:ELSnA = 1:0)

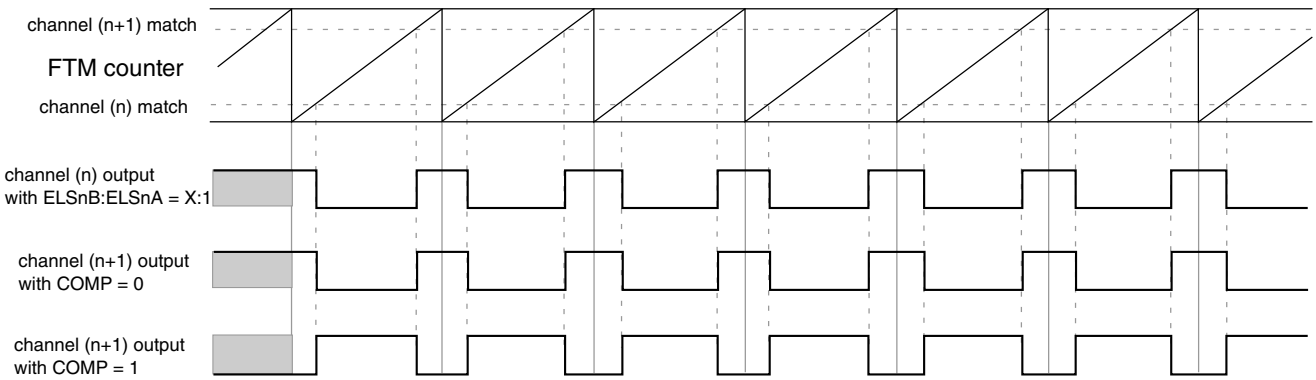


Figure 12-46. Channel (n+1) output in Complementary mode with (ELSnB:ELSnA = X:1)

NOTE

The complementary mode is not available in Output Compare mode.

12.2.4.10 Registers updated from write buffers

12.2.4.10.1 CNTIN register update

The following table describes when CNTIN register is updated:

Table 12-7. CNTIN register update

When	Then CNTIN register is updated
CLKS[1:0] = 0:0	When CNTIN register is written, independent of FTMEN bit.
• FTMEN = 0, or • CNTINC = 0	At the next system clock after CNTIN was written.
• FTMEN = 1, • SYNCMODE = 1, and • CNTINC = 1	By the CNTIN register synchronization.

12.2.4.10.2 MOD register update

The following table describes when MOD register is updated:

Table 12-8. MOD register update

When	Then MOD register is updated
CLKS[1:0] = 0:0	When MOD register is written, independent of FTMEN bit.
• CLKS[1:0] ≠ 0:0, and • FTMEN = 0	According to the CPWMS bit, that is: • If the selected mode is not CPWM then MOD register is updated after MOD register was written and the FTM counter changes from MOD to CNTIN. If the FTM counter is at free-running counter mode then this update occurs when the FTM counter changes from 0xFFFF to 0x0000. • If the selected mode is CPWM then MOD register is updated after MOD register was written and the FTM counter changes from MOD to (MOD – 0x0001).
• CLKS[1:0] ≠ 0:0, and • FTMEN = 1	By the MOD register synchronization.

12.2.4.10.3 CnV register update

The following table describes when CnV register is updated:

Table 12-9. CnV register update

When	Then CnV register is updated
CLKS[1:0] = 0:0	When CnV register is written, independent of FTMEN bit.
• CLKS[1:0] ≠ 0:0, and • FTMEN = 0	According to the selected mode, that is: • If the selected mode is Output Compare, then CnV register is updated on the next FTM counter change, end of the prescaler counting, after CnV register was written. • If the selected mode is EPWM, then CnV register is updated after CnV register was written and the FTM counter changes from MOD to CNTIN. If the FTM counter is at free-running counter mode then this update occurs when the FTM counter changes from 0xFFFF to 0x0000. • If the selected mode is CPWM, then CnV register is updated after CnV register was written and the FTM counter changes from MOD to (MOD – 0x0001).
• CLKS[1:0] ≠ 0:0, and • FTMEN = 1	According to the selected mode, that is: • If the selected mode is output compare then CnV register is updated according to the SYNCEN bit. If (SYNCEN = 0) then CnV register is updated after CnV register was written at the next change of the FTM counter, the end of the prescaler counting. If (SYNCEN = 1) then CnV register is updated by the C(n)V and C(n+1)V register synchronization. • If the selected mode is not output compare and (SYNCEN = 1) then CnV register is updated by the C(n)V and C(n+1)V register synchronization.

12.2.4.11 PWM synchronization

The PWM synchronization provides an opportunity to update the MOD, CNTIN, CnV, OUTMASK, INVCTRL and SWOCTRL registers with their buffered value and force the FTM counter to the CNTIN register value.

Note

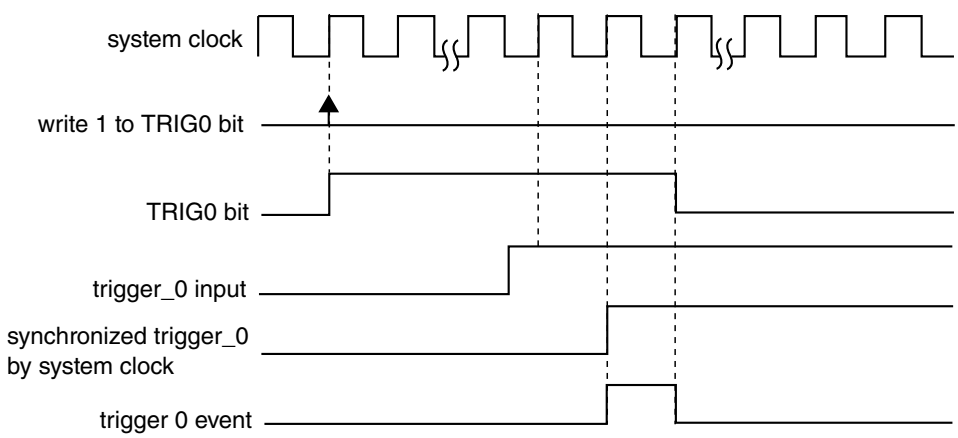
The legacy PWM synchronization (SYNCMODE = 0) is a subset of the enhanced PWM synchronization (SYNCMODE = 1). Thus, only the enhanced PWM synchronization must be used.

12.2.4.11.1 Hardware trigger

Three hardware trigger signal inputs of the FTM module are enabled when TRIGn = 1, where n = 0, 1 or 2 corresponding to each one of the input signals, respectively. The hardware trigger input n is synchronized by the system clock. The PWM synchronization with hardware trigger is initiated when a rising edge is detected at the enabled hardware trigger inputs.

If (HWTRIGMODE = 0) then the TRIGn bit is cleared when 0 is written to it or when the trigger n event is detected.

In this case, if two or more hardware triggers are enabled (for example, TRIG0 and TRIG1 = 1) and only trigger 1 event occurs, then only TRIG1 bit is cleared. If a trigger n event occurs together with a write setting TRIGn bit, then the synchronization is initiated, but TRIGn bit remains set due to the write operation.



Note
All hardware trigger inputs have the same behavior.

Figure 12-47. Hardware trigger event with HWTRIGMODE = 0

If HWTRIGMODE = 1, then the TRIGn bit is only cleared when 0 is written to it.

NOTE

The HWTRIGMODE bit must be 1 only with enhanced PWM synchronization (SYNCMODE = 1).

12.2.4.11.2 Software trigger

A software trigger event occurs when 1 is written to the SYNC[SWSYNC] bit. The SWSYNC bit is cleared when 0 is written to it or when the PWM synchronization, initiated by the software event, is completed.

If another software trigger event occurs (by writing another 1 to the SWSYNC bit) at the same time the PWM synchronization initiated by the previous software trigger event is ending, a new PWM synchronization is started and the SWSYNC bit remains equal to 1.

If SYNCMODE = 0 then the SWSYNC bit is also cleared by FTM according to PWMSYNC and REINIT bits. In this case if (PWMSYNC = 1) or (PWMSYNC = 0 and REINIT = 0) then SWSYNC bit is cleared at the next selected loading point after that the software trigger event occurred; see Boundary cycle and loading points and the following figure. If (PWMSYNC = 0) and (REINIT = 1) then SWSYNC bit is cleared when the software trigger event occurs.

If SYNCMODE = 1 then the SWSYNC bit is also cleared by FTM according to the SWRSTCNT bit. If SWRSTCNT = 0 then SWSYNC bit is cleared at the next selected loading point after that the software trigger event occurred; see the following figure. If SWRSTCNT = 1 then SWSYNC bit is cleared when the software trigger event occurs.

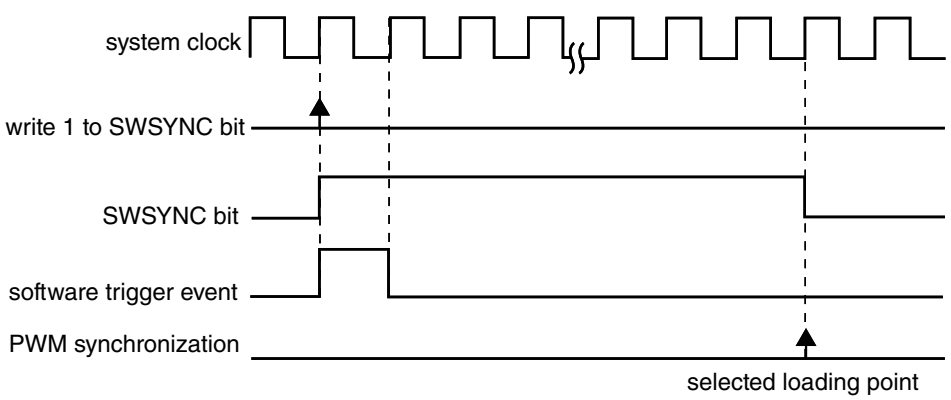


Figure 12-48. Software trigger event

12.2.4.11.3 Boundary cycle and loading points

The boundary cycle definition is important for the loading points for the registers MOD, CNTIN, and C(n)V.

In Up counting mode, the boundary cycle is defined as when the counter wraps to its initial value (CNTIN). If in Up-down counting mode, then the boundary cycle is defined as when the counter turns from down to up counting and when from up to down counting.

The following figure shows the boundary cycles and the loading points for the registers. In the Up Counting mode, the loading points are enabled if one of CNTMIN or CTMAX bits are 1. In the Up-Down Counting mode, the loading points are selected by CNTMIN and CNTMAX bits, as indicated in the figure. These loading points are safe places for register updates thus allowing a smooth transitions in PWM waveform generation.

For both counting modes, if neither CNTMIN nor CNTMAX are 1, then the boundary cycles are not used as loading points for registers updates. See the register synchronization descriptions in the following sections for details.

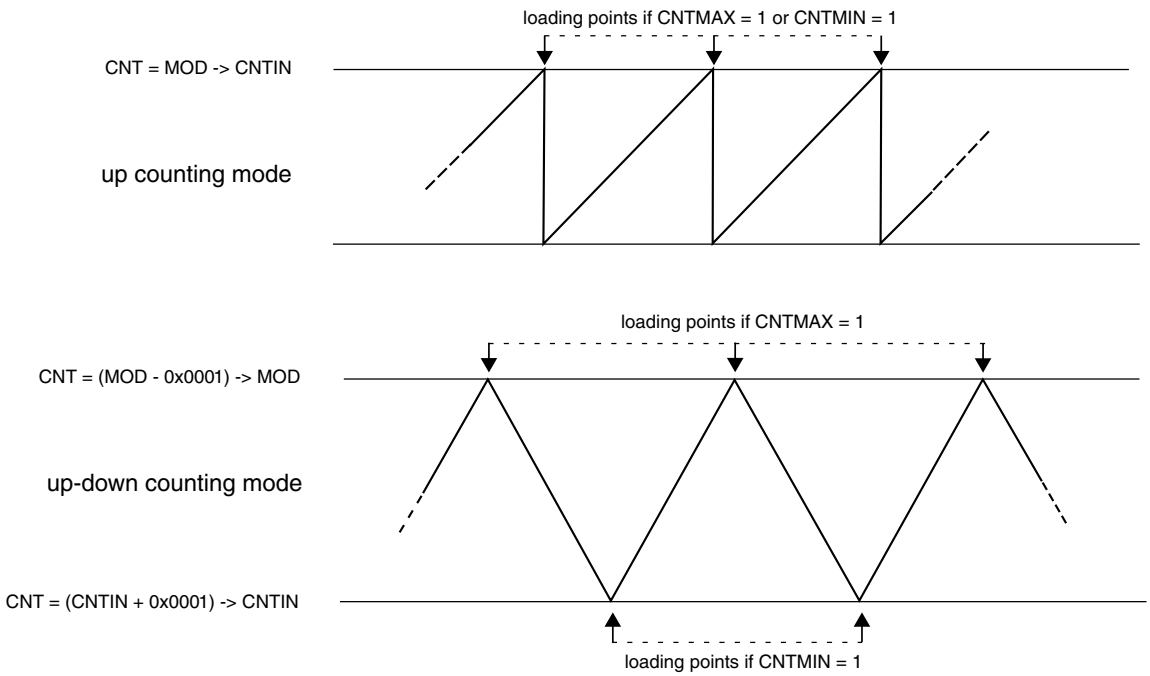


Figure 12-49. Boundary cycles and loading points

12.2.4.11.4 MOD register synchronization

The MOD register synchronization updates the MOD register with its buffer value. This synchronization is enabled if (FTMEN = 1).

The MOD register synchronization can be done by either the enhanced PWM synchronization (SYNCMODE = 1) or the legacy PWM synchronization (SYNCMODE = 0). However, it is expected that the MOD register be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the MOD register synchronization depends on SWWRBUF, SWRSTCNT, HWWRBUF, and HWRSTCNT bits according to this flowchart:

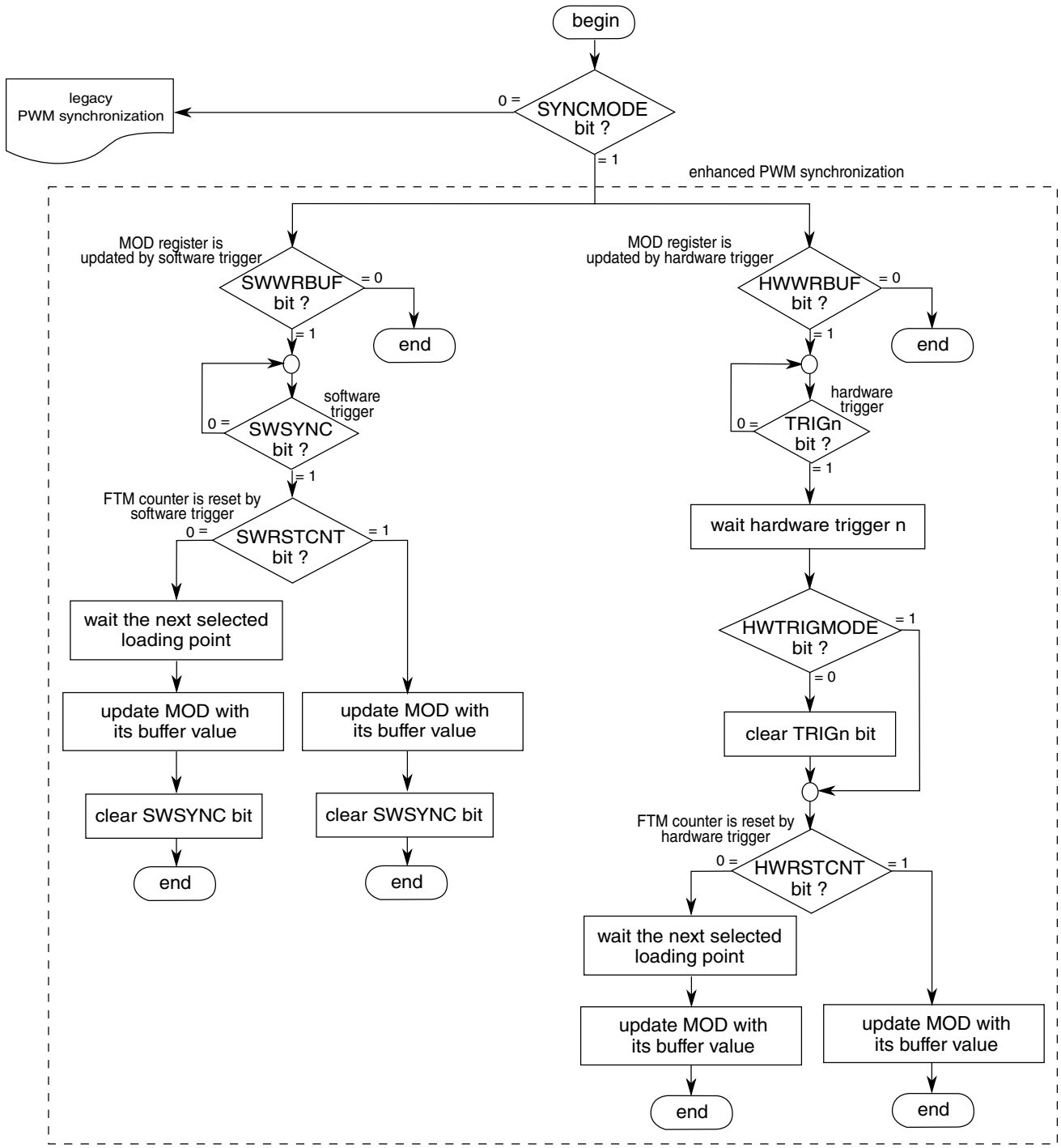


Figure 12-50. MOD register synchronization flowchart

In the case of legacy PWM synchronization, the MOD register synchronization depends on PWMSYNC and REINIT bits according to the following description.

If (SYNCMODE = 0), (PWMSYNC = 0), and (REINIT = 0), then this synchronization is made on the next selected loading point after an enabled trigger event takes place. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected loading point. If the trigger event was a hardware trigger, then the trigger enable bit (TRIGn) is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

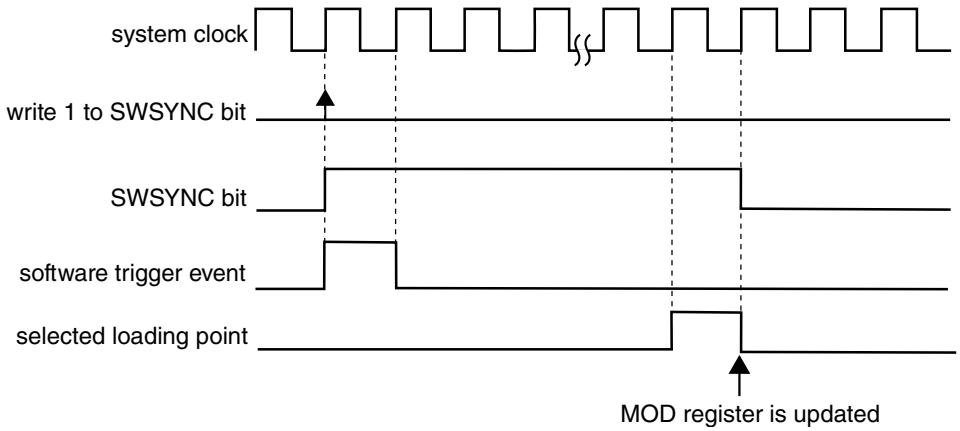


Figure 12-51. MOD synchronization with (SYNCMODE = 0), (PWMSYNC = 0), (REINIT = 0), and software trigger was used

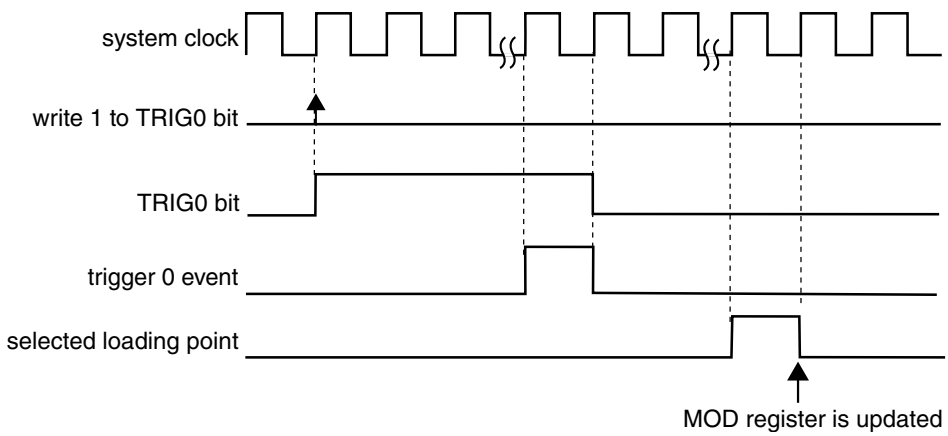


Figure 12-52. MOD synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (PWMSYNC = 0), (REINIT = 0), and a hardware trigger was used

If (SYNCMODE = 0), (PWMSYNC = 0), and (REINIT = 1), then this synchronization is made on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared according to the following example. If the trigger event was a hardware trigger, then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

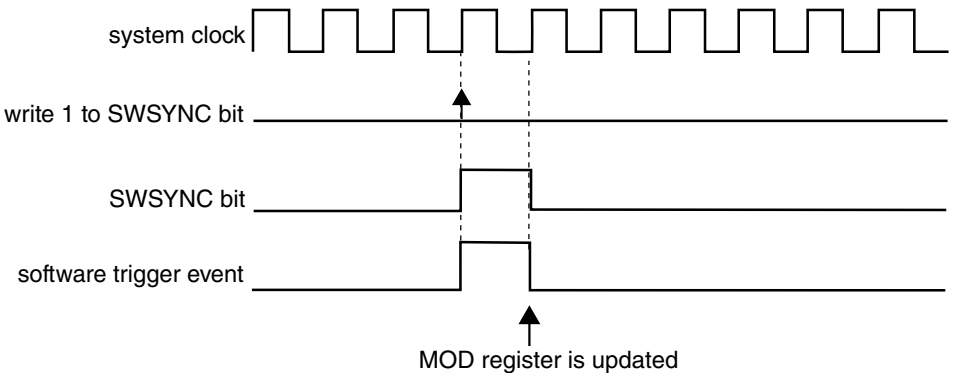


Figure 12-53. MOD synchronization with (SYNCMODE = 0), (PWMSYNC = 0), (REINIT = 1), and software trigger was used

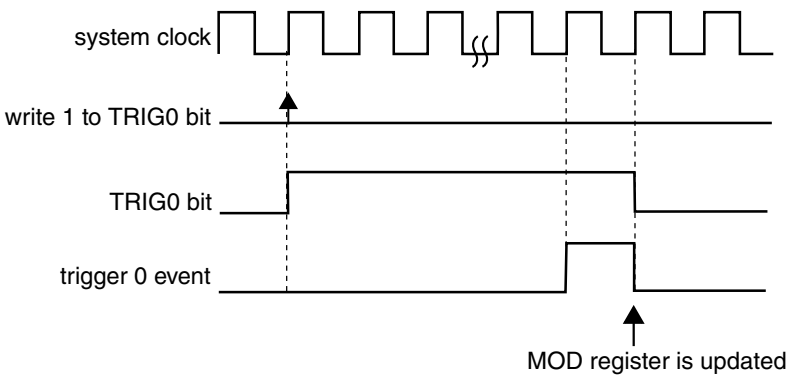


Figure 12-54. MOD synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (PWMSYNC = 0), (REINIT = 1), and a hardware trigger was used

If (SYNCMODE = 0) and (PWMSYNC = 1), then this synchronization is made on the next selected loading point after the software trigger event takes place. The SWSYNC bit is cleared on the next selected loading point:

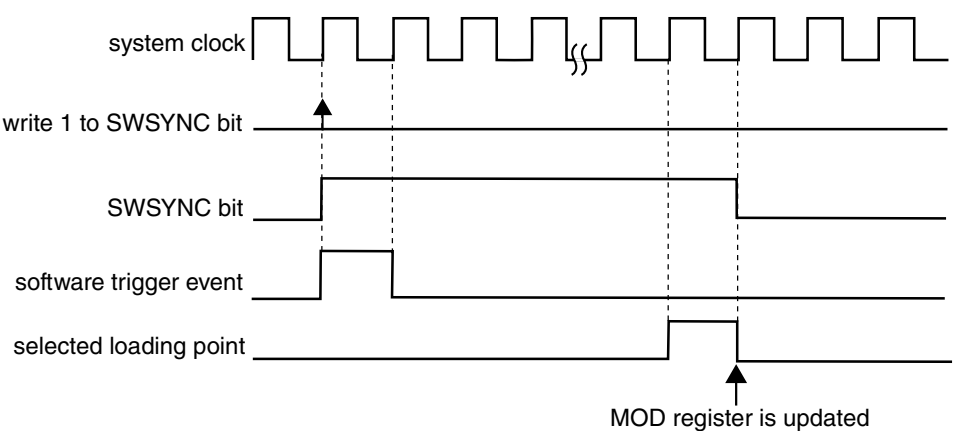


Figure 12-55. MOD synchronization with (SYNCMODE = 0) and (PWMSYNC = 1)

12.2.4.11.5 CNTIN register synchronization

The CNTIN register synchronization updates the CNTIN register with its buffer value.

This synchronization is enabled if (FTMEN = 1), (SYNCMODE = 1), and (CNTINC = 1). The CNTIN register synchronization can be done only by the enhanced PWM synchronization (SYNCMODE = 1). The synchronization mechanism is the same as the MOD register synchronization done by the enhanced PWM synchronization; see MOD register synchronization.

12.2.4.11.6 C(n)V and C(n+1)V register synchronization

The C(n)V and C(n+1)V registers synchronization updates the C(n)V and C(n+1)V registers with their buffer values.

This synchronization is enabled if (FTMEN = 1) and (SYNCEN = 1). The synchronization mechanism is the same as the MOD register synchronization. However, it is expected that the C(n)V and C(n+1)V registers be synchronized only by the enhanced PWM synchronization (SYNCMODE = 1).

12.2.4.11.7 OUTMASK register synchronization

The OUTMASK register synchronization updates the OUTMASK register with its buffer value.

The OUTMASK register can be updated at each rising edge of system clock (SYNCHOM = 0), by the enhanced PWM synchronization (SYNCHOM = 1 and SYNCMODE = 1) or by the legacy PWM synchronization (SYNCHOM = 1 and SYNCMODE = 0). However, it is expected that the OUTMASK register be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the OUTMASK register synchronization depends on SWOM and HWOM bits. See the following flowchart:

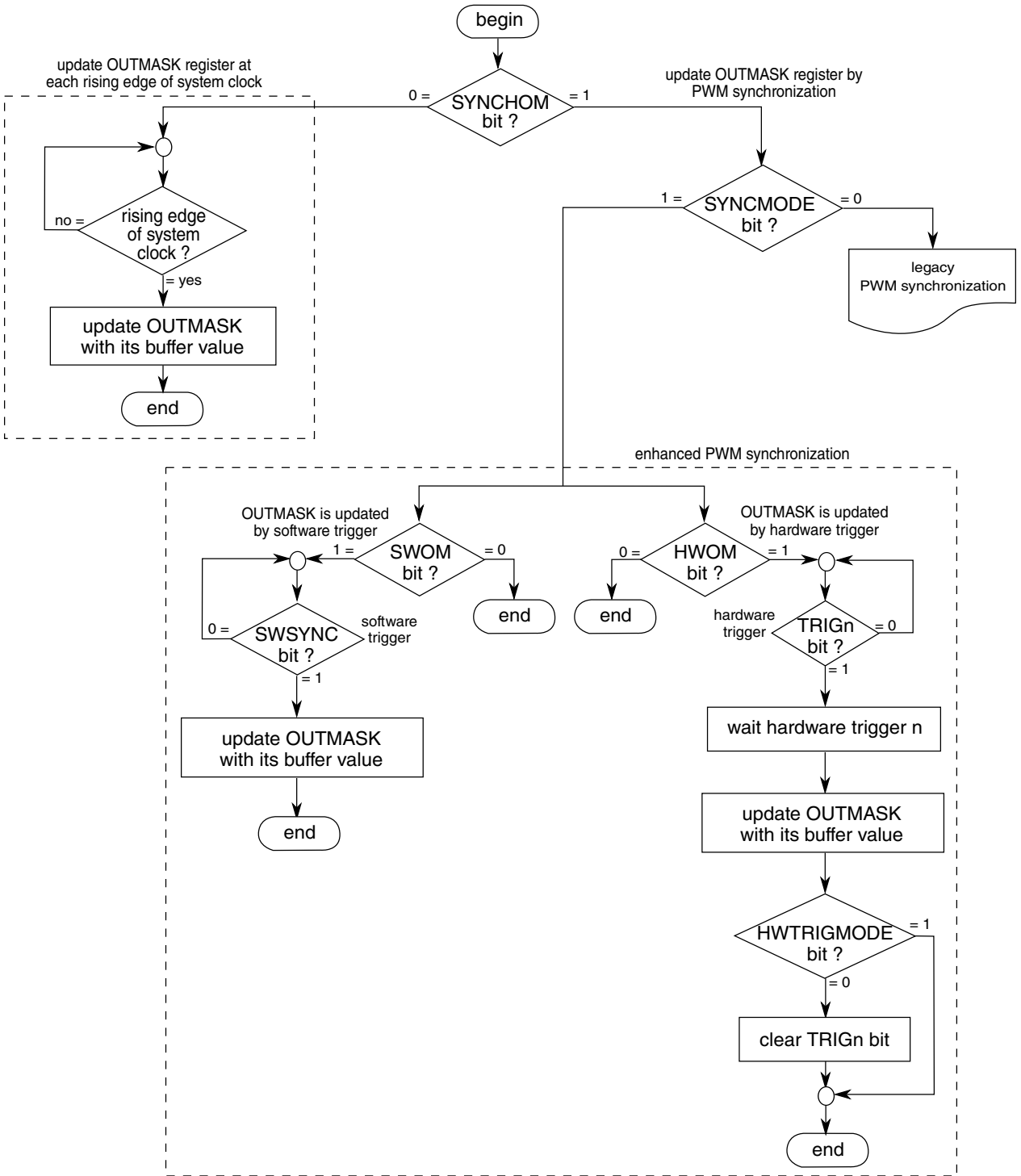


Figure 12-56. OUTMASK register synchronization flowchart

In the case of legacy PWM synchronization, the OUTMASK register synchronization depends on PWMSYNC bit according to the following description.

If (SYNCMODE = 0), (SYNCHOM = 1), and (PWMSYNC = 0), then this synchronization is done on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected loading point. If the trigger event was a hardware trigger, then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

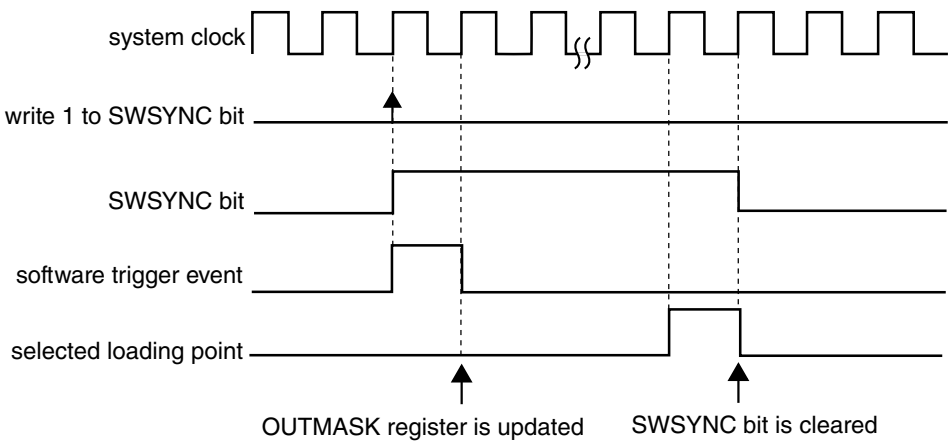


Figure 12-57. OUTMASK synchronization with (SYNCMODE = 0), (SYNCHOM = 1), (PWMSYNC = 0) and software trigger was used

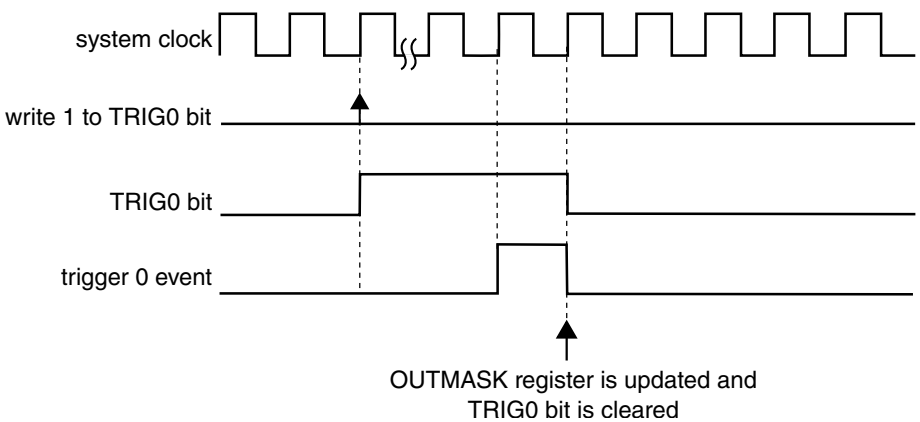


Figure 12-58. OUTMASK synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (SYNCHOM = 1), (PWMSYNC = 0), and a hardware trigger was used

If (SYNCMODE = 0), (SYNCHOM = 1), and (PWMSYNC = 1), then this synchronization is made on the next enabled hardware trigger. The TRIGn bit is cleared according to Hardware trigger. An example with a hardware trigger follows.

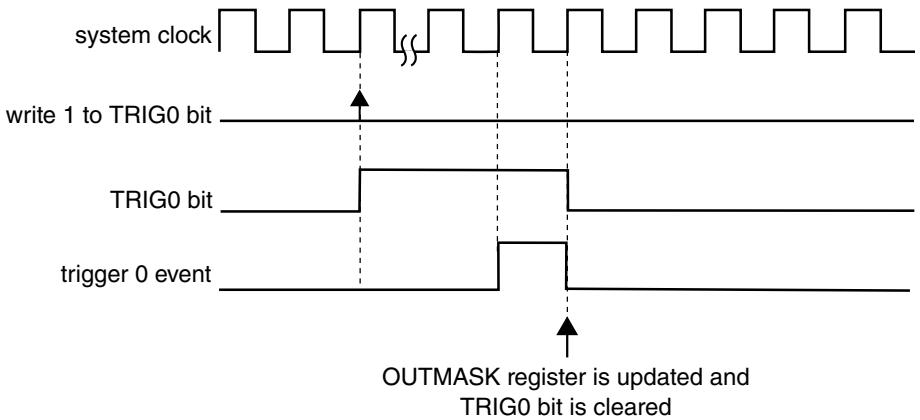


Figure 12-59. OUTMASK synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (SYNCHOM = 1), (PWMSYNC = 1), and a hardware trigger was used

12.2.4.11.8 INVCTRL register synchronization

The INVCTRL register synchronization updates the INVCTRL register with its buffer value.

The INVCTRL register can be updated at each rising edge of system clock (INVC = 0) or by the enhanced PWM synchronization (INVC = 1 and SYNCMODE = 1) according to the following flowchart.

In the case of enhanced PWM synchronization, the INVCTRL register synchronization depends on SWINVC and HWINVC bits.

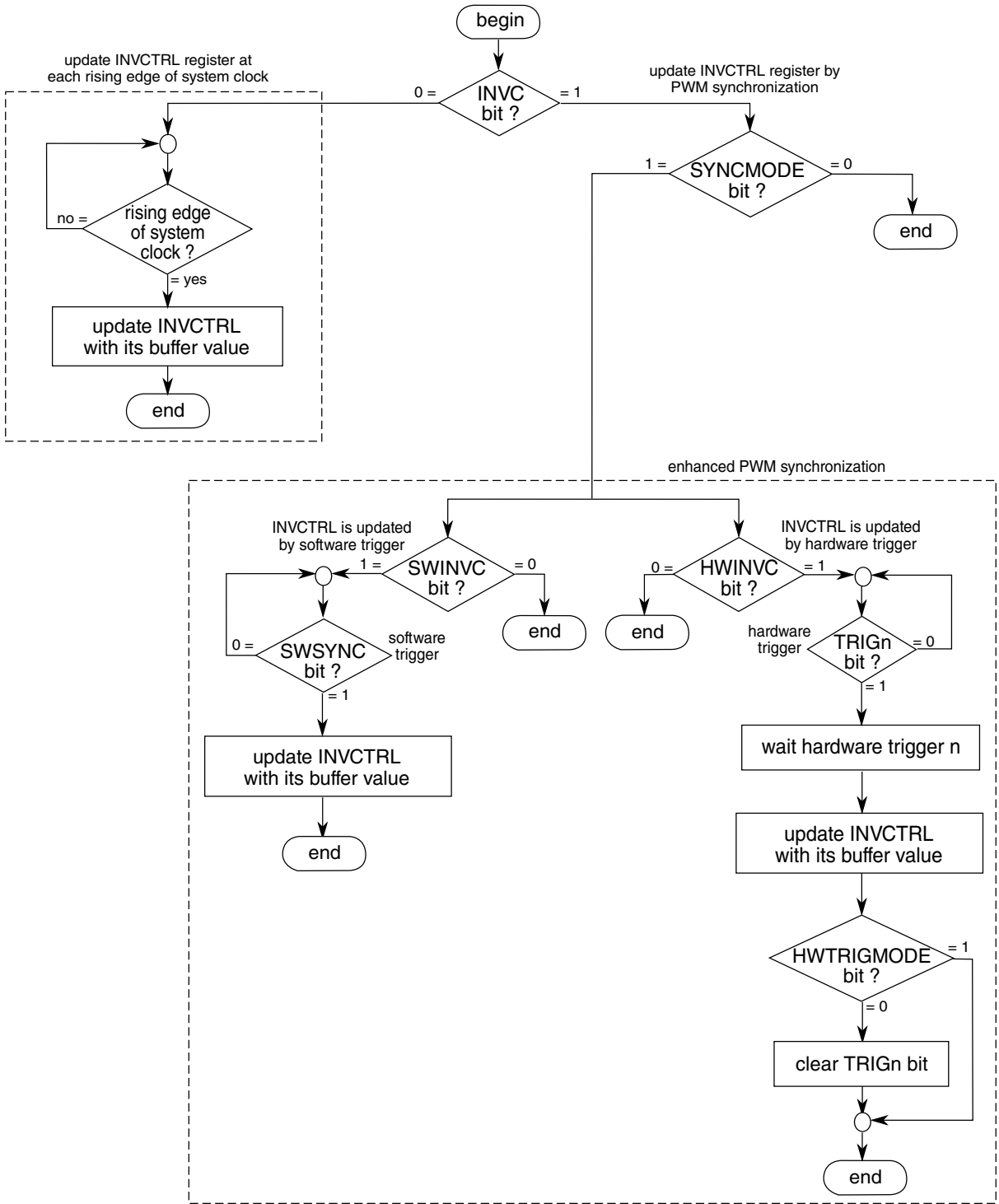


Figure 12-60. INVCTRL register synchronization flowchart

12.2.4.11.9 SWOCTRL register synchronization

The SWOCTRL register synchronization updates the SWOCTRL register with its buffer value.

The SWOCTRL register can be updated at each rising edge of system clock (SWOC = 0) or by the enhanced PWM synchronization (SWOC = 1 and SYNCMODE = 1) according to the following flowchart.

In the case of enhanced PWM synchronization, the SWOCTRL register synchronization depends on SWSOC and HWSOC bits.

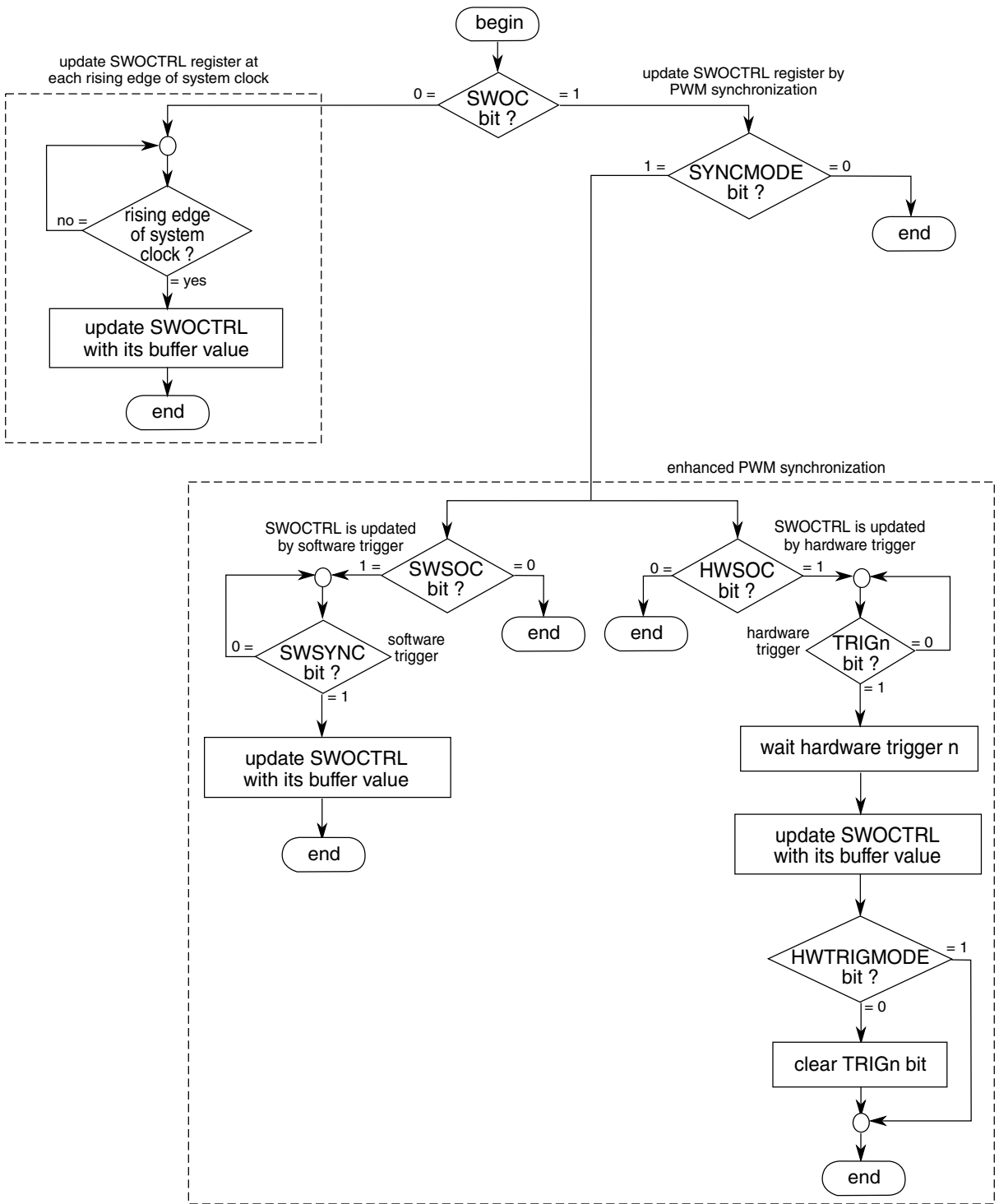


Figure 12-61. SWOCTRL register synchronization flowchart

12.2.4.11.10 FTM counter synchronization

The FTM counter synchronization is a mechanism that allows the FTM to restart the PWM generation at a certain point in the PWM period. The channels outputs are forced to their initial value, except for channels in Output Compare mode, and the FTM counter is forced to its initial counting value defined by CNTIN register.

The following figure shows the FTM counter synchronization. Note that after the synchronization event occurs, the channel (n) is set to its initial value and the channel (n +1) is not set to its initial value due to a specific timing of this figure in which the deadtime insertion prevents this channel output from transitioning to 1. If no deadtime insertion is selected, then the channel (n+1) transitions to logical value 1 immediately after the synchronization event occurs.

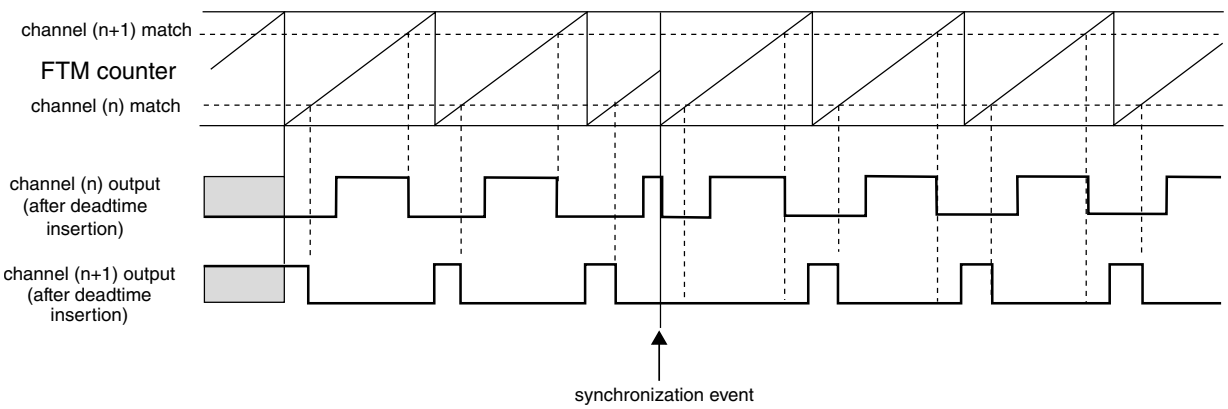


Figure 12-62. FTM counter synchronization

The FTM counter synchronization can be done by either the enhanced PWM synchronization (SYNCMODE = 1) or the legacy PWM synchronization (SYNCMODE = 0). However, the FTM counter must be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the FTM counter synchronization depends on SWRSTCNT and HWRSTCNT bits according to the following flowchart.

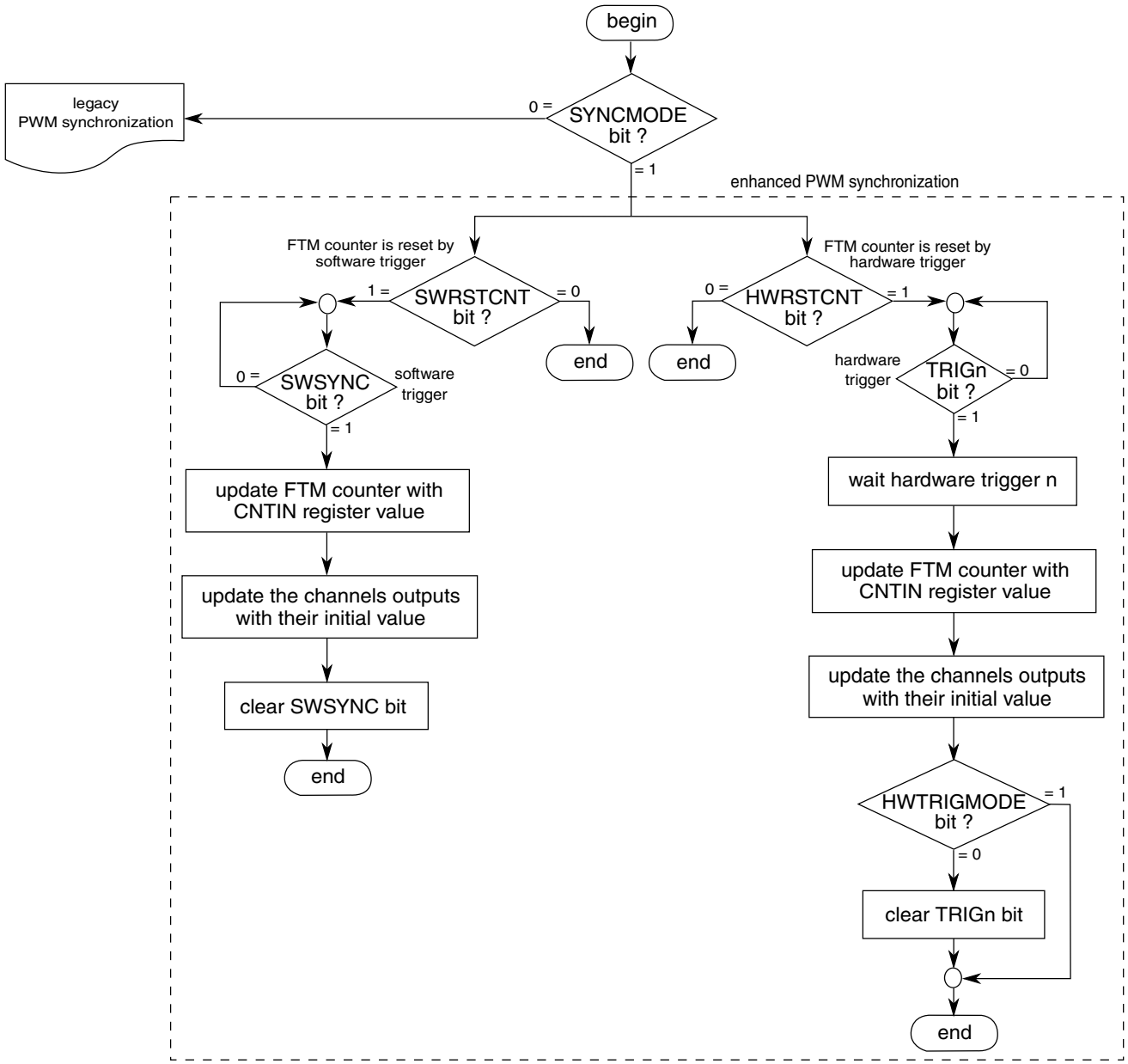


Figure 12-63. FTM counter synchronization flowchart

In the case of legacy PWM synchronization, the FTM counter synchronization depends on REINIT and PWMSYNC bits according to the following description.

If (SYNCMODE = 0), (REINIT = 1), and (PWMSYNC = 0) then this synchronization is made on the next enabled trigger event. If the trigger event was a software trigger then the SWSYNC bit is cleared according to the following example. If the trigger event was a hardware trigger then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

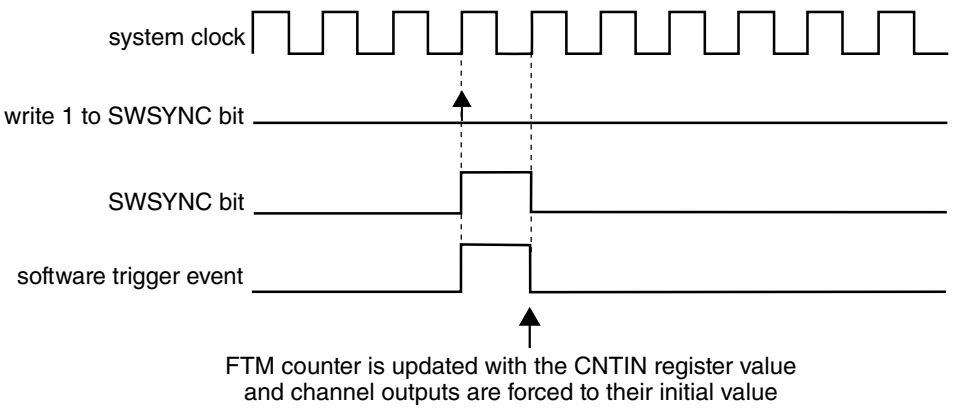


Figure 12-64. FTM counter synchronization with (SYNCMODE = 0), (REINIT = 1), (PWMSYNC = 0), and software trigger was used

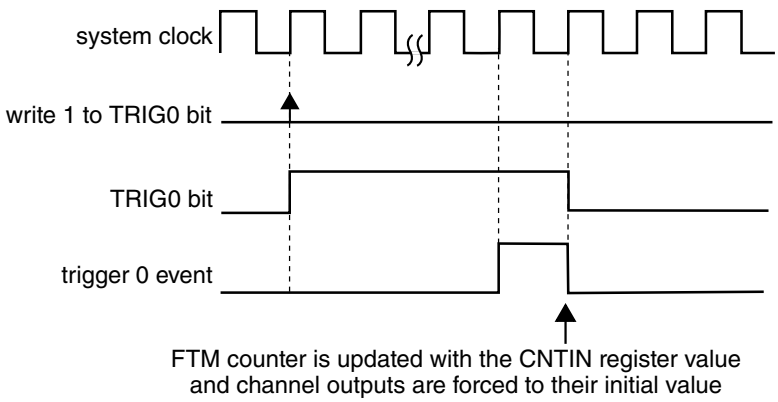


Figure 12-65. FTM counter synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (REINIT = 1), (PWMSYNC = 0), and a hardware trigger was used

If (SYNCMODE = 0), (REINIT = 1), and (PWMSYNC = 1) then this synchronization is made on the next enabled hardware trigger. The TRIGn bit is cleared according to Hardware trigger.

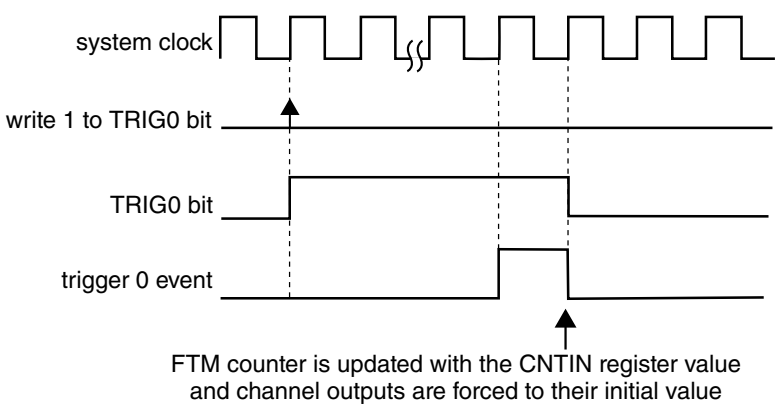


Figure 12-66. FTM counter synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (REINIT = 1), (PWMSYNC = 1), and a hardware trigger was used

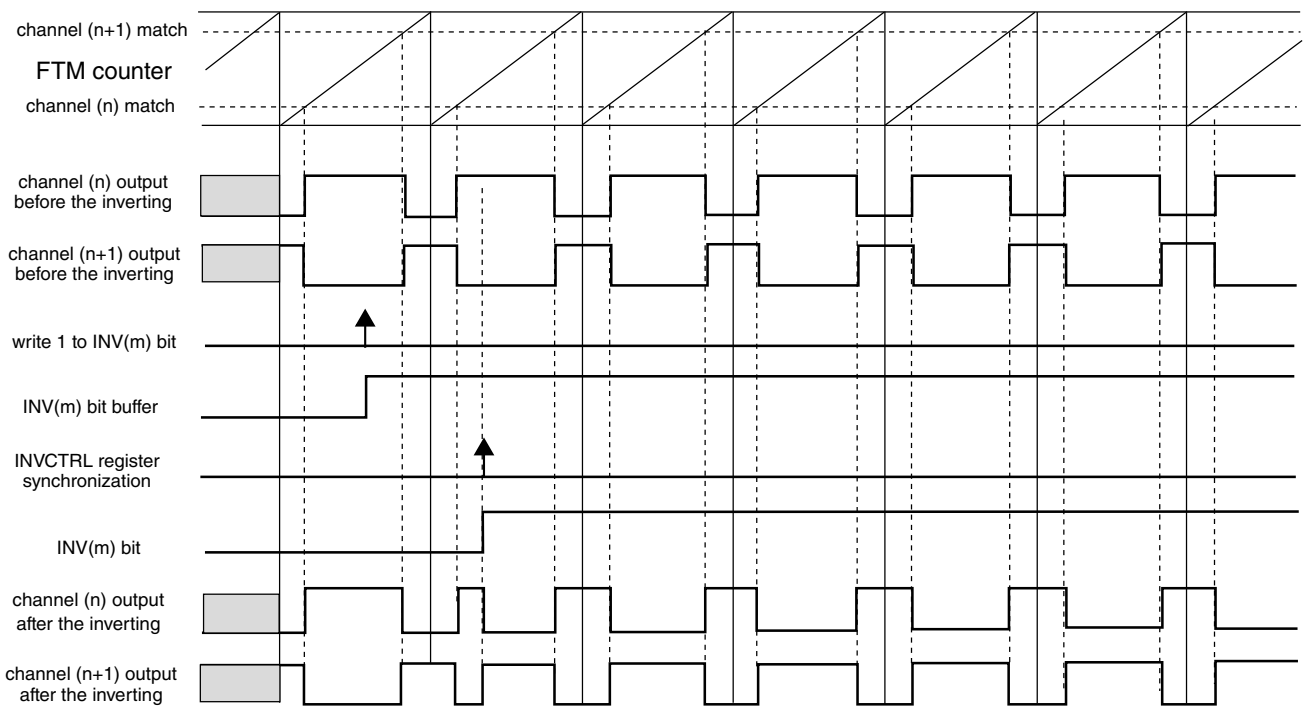
12.2.4.12 Inverting

The invert functionality swaps the signals between channel (n) and channel (n+1) outputs. The inverting operation is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMP = 1, and
- INVm = 1 (where m represents a channel pair)

The INVm bit in INVCTRL register is updated with its buffer value according to INVCTRL register synchronization

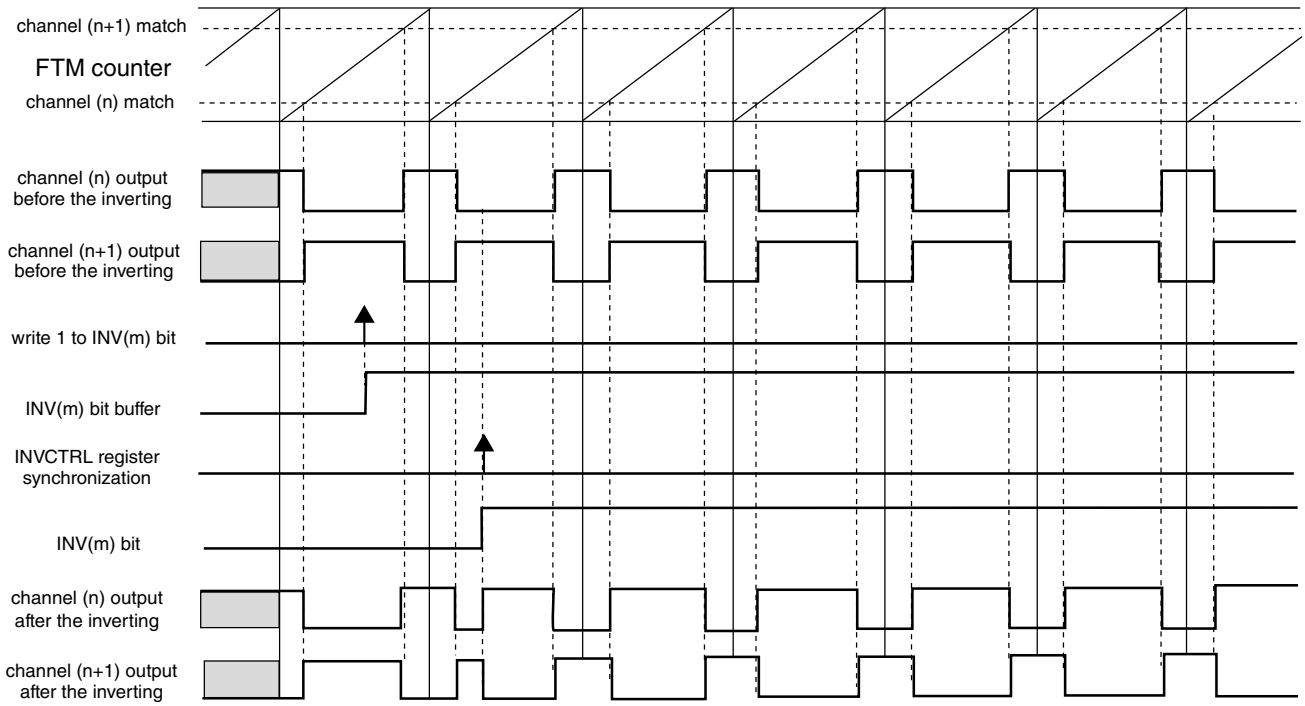
In High-True (ELSnB:ELSnA = 1:0) Combine mode, the channel (n) output is forced low at the beginning of the period (FTM counter = CNTIN), forced high at the channel (n) match and forced low at the channel (n+1) match. If the inverting is selected, the channel (n) output behavior is changed to force high at the beginning of the PWM period, force low at the channel (n) match and force high at the channel (n+1) match. See the following figure.



NOTE
INV(m) bit selects the inverting to the pair channels (n) and (n+1).

Figure 12-67. Channels (n) and (n+1) outputs after the inverting in High-True (ELSnB:ELSnA = 1:0) Combine mode

Note that the ELSnB:ELSnA bits value should be considered because they define the active state of the channels outputs. In Low-True (ELSnB:ELSnA = X:1) Combine mode, the channel (n) output is forced high at the beginning of the period, forced low at the channel (n) match and forced high at the channel (n+1) match. When inverting is selected, the channels (n) and (n+1) present waveforms as shown in the following figure.



NOTE
INV(m) bit selects the inverting to the pair channels (n) and (n+1).

Figure 12-68. Channels (n) and (n+1) outputs after the inverting in Low-True (ELSnB:ELSnA = X:1) Combine mode

Note

The inverting feature is not available in Output Compare mode.

12.2.4.13 Software output control

The software output control forces the channel output according to software defined values at a specific time in the PWM generation.

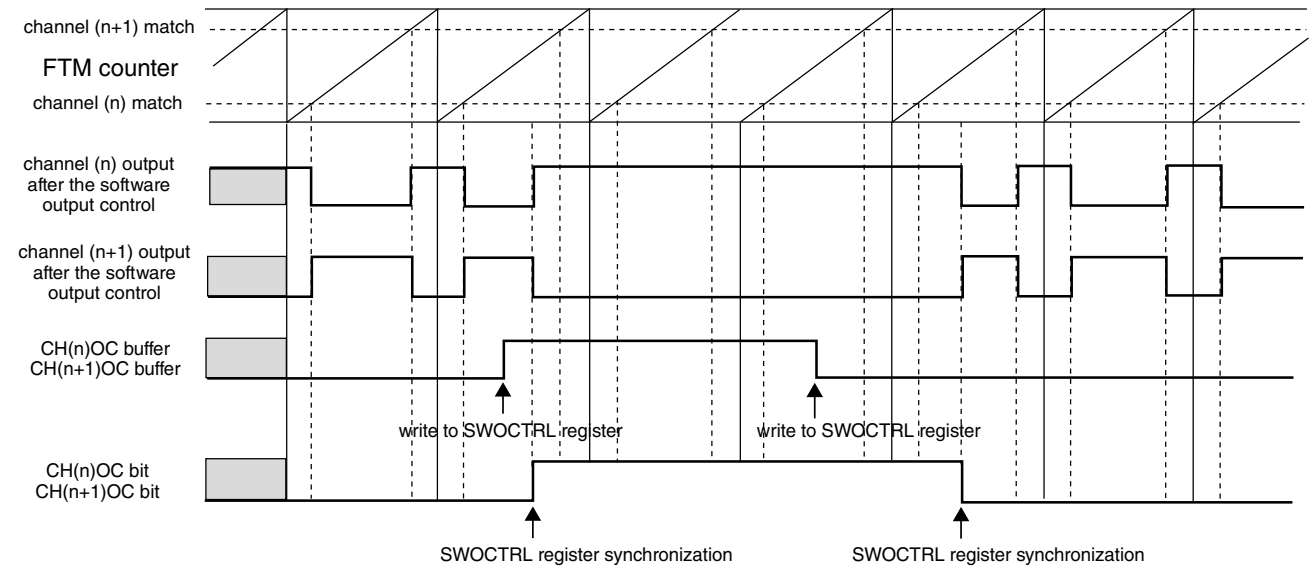
The software output control is selected when:

- QUADEN = 0
- DECAPEN = 0, and
- CHnOC = 1

The CHnOC bit enables the software output control for a specific channel output and the CHnOCV selects the value that is forced to this channel output.

Both CHnOC and CHnOCV bits in SWOCTRL register are buffered and updated with their buffer value according to SWOCTRL register synchronization.

The following figure shows the channels (n) and (n+1) outputs signals when the software output control is used. In this case the channels (n) and (n+1) are set to Combine and Complementary mode.



NOTE
CH(n)OCV = 1 and CH(n+1)OCV = 0.

Figure 12-69. Example of software output control in Combine and Complementary mode

Software output control forces the following values on channels (n) and (n+1) when the COMP bit is zero.

Table 12-10. Software ouput control behavior when (COMP = 0)

CH(n)OC	CH(n+1)OC	CH(n)OCV	CH(n+1)OCV	Channel (n) Output	Channel (n+1) Output
0	0	X	X	is not modified by SWOC	is not modified by SWOC
1	1	0	0	is forced to zero	is forced to zero
1	1	0	1	is forced to zero	is forced to one
1	1	1	0	is forced to one	is forced to zero
1	1	1	1	is forced to one	is forced to one

Software output control forces the following values on channels (n) and (n+1) when the COMP bit is one.

Table 12-11. Software ouput control behavior when (COMP = 1)

CH(n)OC	CH(n+1)OC	CH(n)OCV	CH(n+1)OCV	Channel (n) Output	Channel (n+1) Output
0	0	X	X	is not modified by SWOC	is not modified by SWOC
1	1	0	0	is forced to zero	is forced to zero
1	1	0	1	is forced to zero	is forced to one
1	1	1	0	is forced to one	is forced to zero
1	1	1	1	is forced to one	is forced to zero

Note

- The CH(n)OC and CH(n+1)OC bits should be equal.
- The COMP bit must not be modified when software output control is enabled, that is, CH(n)OC = 1 and/or CH(n +1)OC = 1.
- Software output control has the same behavior with disabled or enabled FTM counter (see the CLKS field description in the Status and Control register).

12.2.4.14 Deadtime insertion

The deadtime insertion is enabled when (DTEN = 1) and (DTVAL[5:0] is non- zero).

DEADTIME register defines the deadtime delay that can be used for all FTM channels. The DTPS[1:0] bits define the prescaler for the system clock and the DTVAL[5:0] bits define the deadtime modulo, that is, the number of the deadtime prescaler clocks.

The deadtime delay insertion ensures that no two complementary signals (channels (n) and (n+1)) drive the active state at the same time.

If POL(n) = 0, POL(n+1) = 0, and the deadtime is enabled, then when the channel (n) match (FTM counter = C(n)V) occurs, the channel (n) output remains at the low value until the end of the deadtime delay when the channel (n) output is set. Similarly, when the channel (n+1) match (FTM counter = C(n+1)V) occurs, the channel (n+1) output remains at the low value until the end of the deadtime delay when the channel (n+1) output is set. See the following figures.

If POL(n) = 1, POL(n+1) = 1, and the deadtime is enabled, then when the channel (n) match (FTM counter = C(n)V) occurs, the channel (n) output remains at the high value until the end of the deadtime delay when the channel (n) output is cleared. Similarly,

when the channel (n+1) match (FTM counter = C(n+1)V) occurs, the channel (n+1) output remains at the high value until the end of the deadtime delay when the channel (n +1) output is cleared.

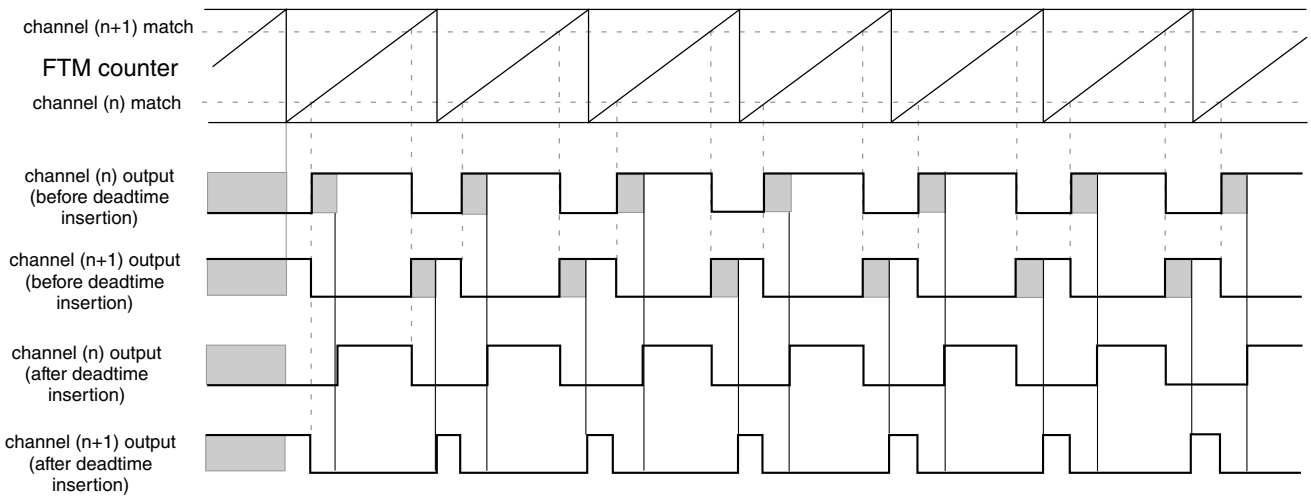


Figure 12-70. Deadtime insertion with ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0

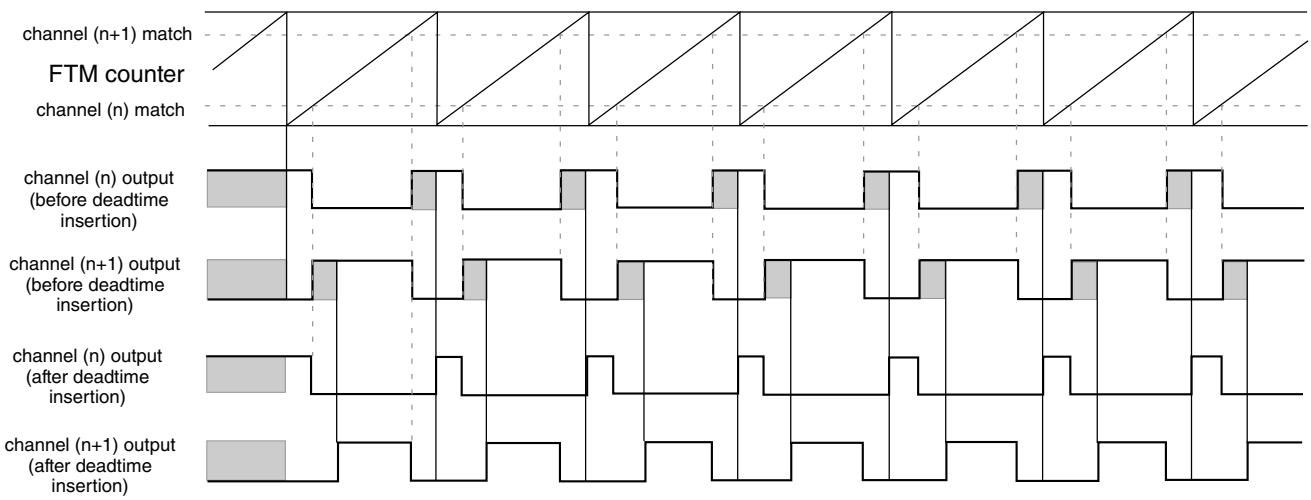


Figure 12-71. Deadtime insertion with ELSnB:ELSnA = X:1, POL(n) = 0, and POL(n+1) = 0

NOTE

- The deadtime feature must be used only in Complementary mode.
- The deadtime feature is not available in Output Compare mode.

12.2.4.14.1 Deadtime insertion corner cases

If (PS[2:0] is cleared), (DTPS[1:0] = 0:0 or DTPS[1:0] = 0:1):

- and the deadtime delay is greater than or equal to the channel (n) duty cycle ((C(n +1)V – C(n)V) × system clock), then the channel (n) output is always the inactive value (POL(n) bit value).
- and the deadtime delay is greater than or equal to the channel (n+1) duty cycle ((MOD – CNTIN + 1 – (C(n+1)V – C(n)V)) × system clock), then the channel (n+1) output is always the inactive value (POL(n+1) bit value).

Although, in most cases the deadtime delay is not comparable to channels (n) and (n+1) duty cycle, the following figures show examples where the deadtime delay is comparable to the duty cycle.

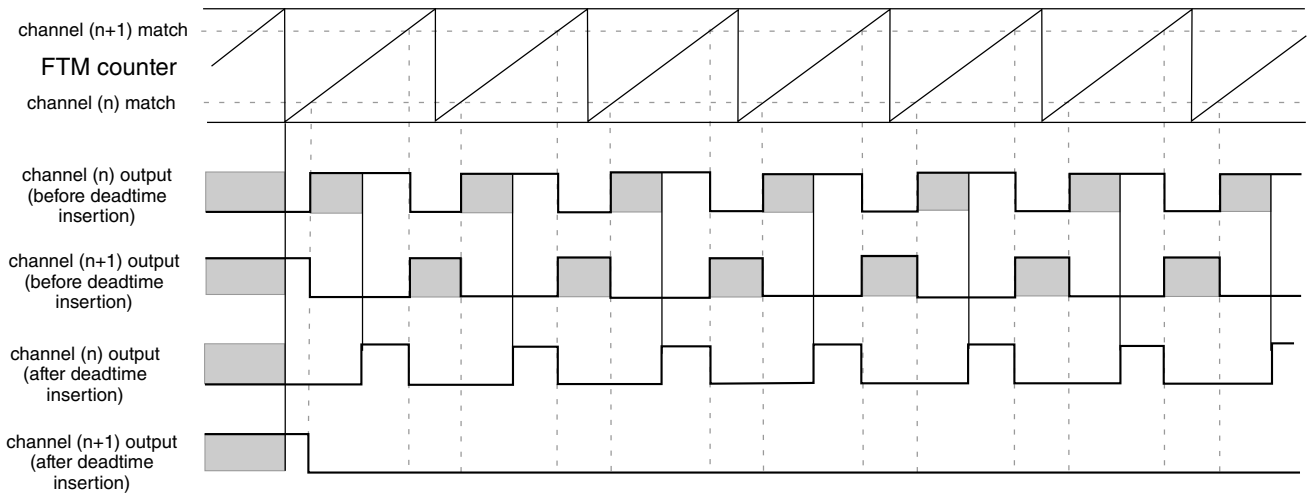


Figure 12-72. Example of the deadtime insertion (ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0) when the deadtime delay is comparable to channel (n+1) duty cycle

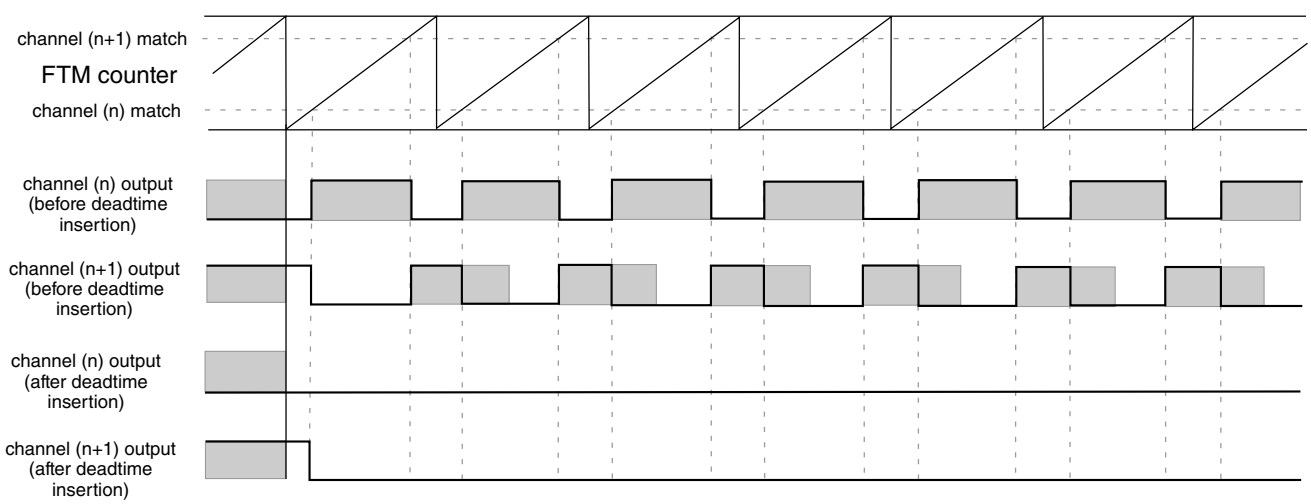


Figure 12-73. Example of the deadtime insertion (ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0) when the deadtime delay is comparable to channels (n) and (n+1) duty cycle

12.2.4.15 Output mask

The output mask can be used to force channels output to their inactive state through software. For example: to control a BLDC motor.

Any write to the OUTMASK register updates its write buffer. The OUTMASK register is updated with its buffer value by PWM synchronization; see OUTMASK register synchronization.

If CHnOM = 1, then the channel (n) output is forced to its inactive state (POLn bit value). If CHnOM = 0, then the channel (n) output is unaffected by the output mask. See the following figure.

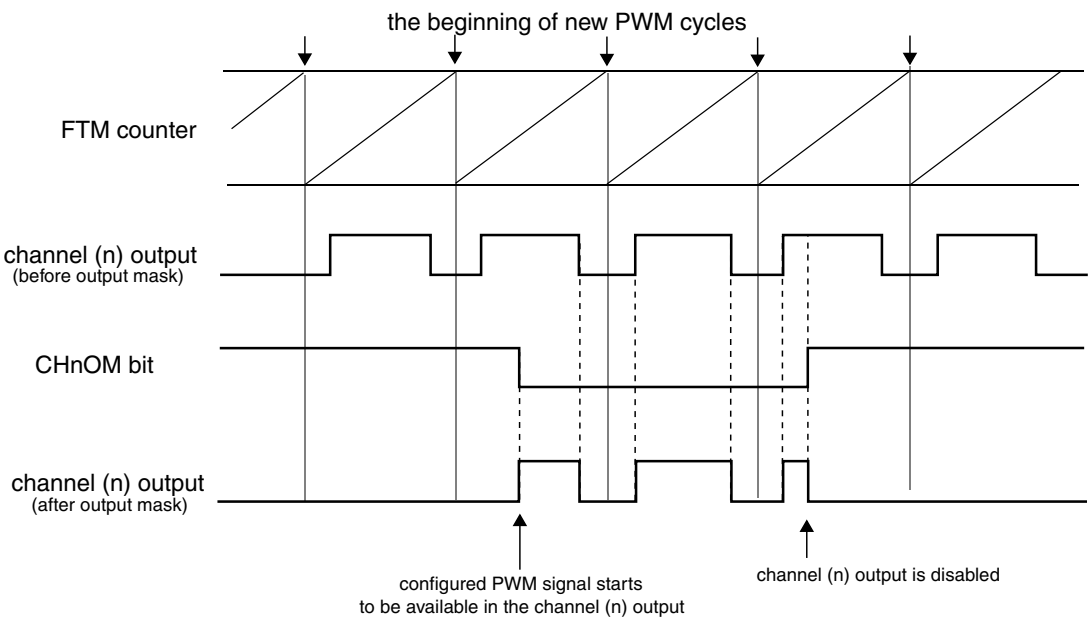


Figure 12-74. Output mask with POLn = 0

The following table shows the output mask result before the polarity control.

Table 12-12. Output mask result for channel (n) before the polarity control

CHnOM	Output Mask Input	Output Mask Result
0	inactive state	inactive state
	active state	active state
1	inactive state	inactive state
	active state	

12.2.4.16 Polarity control

The POLn bit selects the channel (n) output polarity:

- If POLn = 0, the channel (n) output polarity is high, so the logical one is the active state and the logical zero is the inactive state.
- If POLn = 1, the channel (n) output polarity is low, so the logical zero is the active state and the logical one is the inactive state.

12.2.4.17 Initialization

The initialization forces the CHnOI bit value to the channel (n) output when a one is written to the INIT bit.

The initialization depends on COMP and DTEN bits. The following table shows the values that channels (n) and (n+1) are forced by initialization when the COMP and DTEN bits are zero.

Table 12-13. Initialization behavior when (COMP = 0 and DTEN = 0)

CH(n)OI	CH(n+1)OI	Channel (n) Output	Channel (n+1) Output
0	0	is forced to zero	is forced to zero
0	1	is forced to zero	is forced to one
1	0	is forced to one	is forced to zero
1	1	is forced to one	is forced to one

The following table shows the values that channels (n) and (n+1) are forced by initialization when (COMP = 1) or (DTEN = 1).

Table 12-14. Initialization behavior when (COMP = 1 or DTEN = 1)

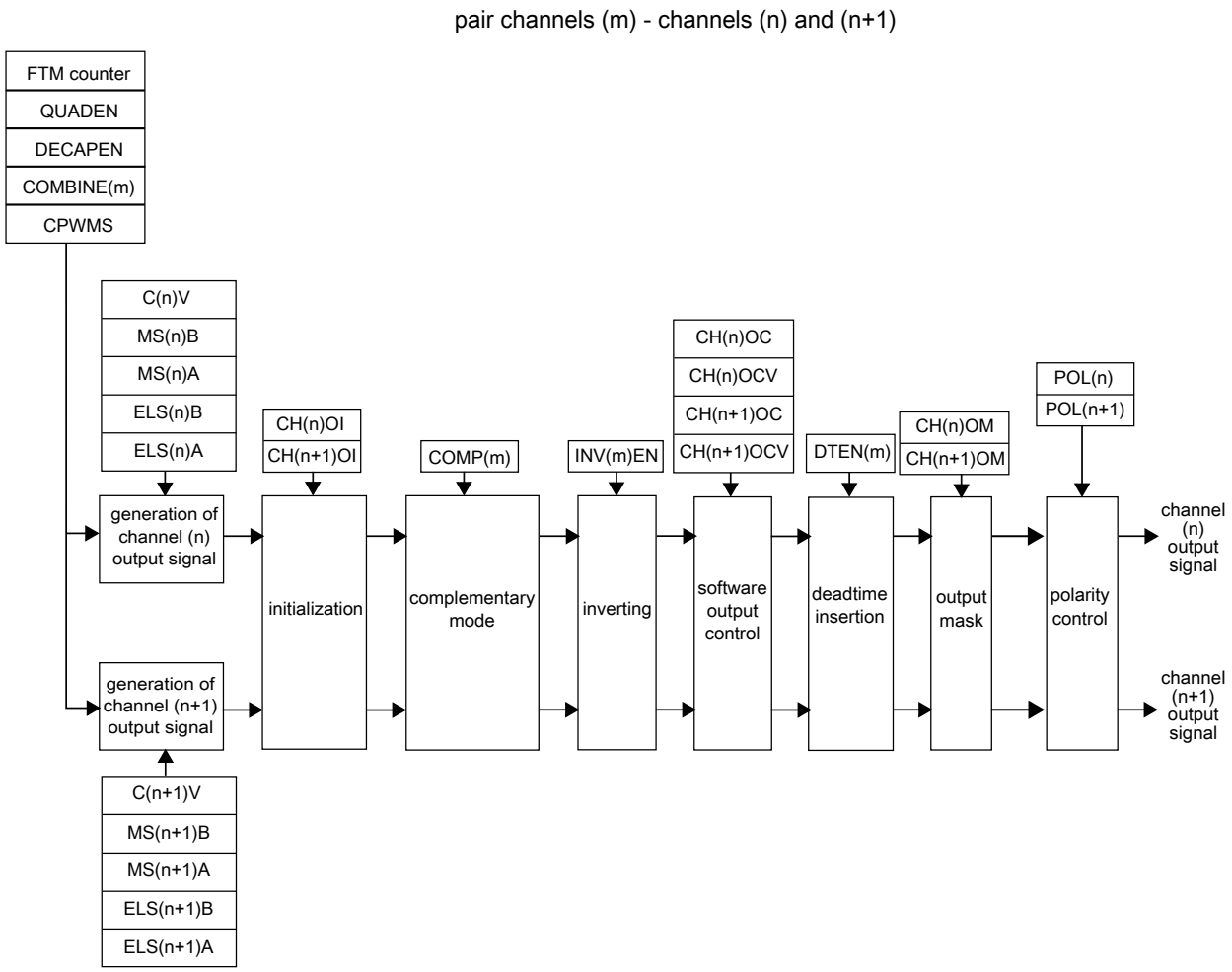
CH(n)OI	CH(n+1)OI	Channel (n) Output	Channel (n+1) Output
0	X	is forced to zero	is forced to one
1	X	is forced to one	is forced to zero

Note

The initialization feature must be used only with disabled FTM counter. See the description of the CLKS field in the Status and Control register.

12.2.4.18 Features priority

The following figure shows the priority of the features used at the generation of channels (n) and (n+1) outputs signals.



NOTE
The channels (n) and (n+1) are in output compare, EPWM, CPWM or combine modes.

Figure 12-75. Priority of the features used at the generation of channels (n) and (n+1) outputs signals

Note

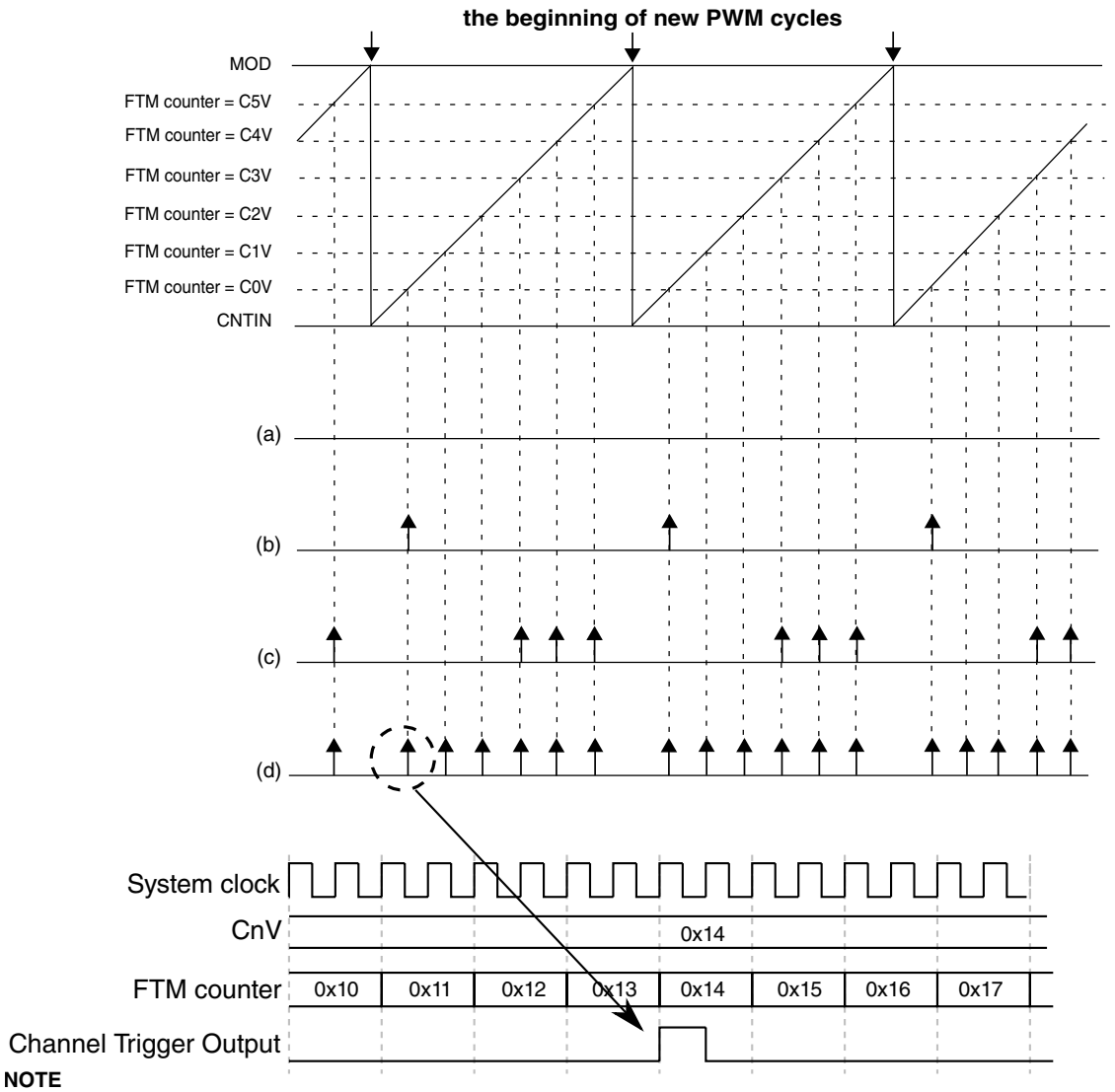
The Initialization feature must not be used with Inverting and Software output control features.

12.2.4.19 Channel trigger output

If CH(j)TRIG bit of the FTM External Trigger (FTM_EXTTRIG) register is set, where j = 0, 1, 2, 3, 4, or 5, then the FTM generates a trigger when the channel (j) match occurs (FTM counter = C(j)V).

The channel trigger output provides a trigger signal which has one FTM clock period width and is used for on-chip modules.

The FTM is able to generate multiple triggers in one PWM period. Because each trigger is generated for a specific channel, several channels are required to implement this functionality. This behavior is described in the following figure.



NOTE

(a) CH0TRIG = 0, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 0, CH4TRIG = 0, CH5TRIG = 0
(b) CH0TRIG = 1, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 0, CH4TRIG = 0, CH5TRIG = 0
(c) CH0TRIG = 0, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 1, CH4TRIG = 1, CH5TRIG = 1
(d) CH0TRIG = 1, CH1TRIG = 1, CH2TRIG = 1, CH3TRIG = 1, CH4TRIG = 1, CH5TRIG = 1

Figure 12-76. Channel match trigger

12.2.4.20 Initialization trigger

If INITTRIGEN = 1, then the FTM generates a trigger when the FTM counter is updated with the CNTIN register value in the following cases.

- The FTM counter is automatically updated with the CNTIN register value by the selected counting mode.
- When there is a write to CNT register.
- When there is the FTM counter synchronization.
- If (CNT = CNTIN), (CLKS[1:0] = 0:0), and a value different from zero is written to CLKS[1:0] bits.
- If the channel (n) is in Input Capture mode, (ICRST = 1) and the selected input capture event occurs in the channel (n) input.

The following figures show these cases.

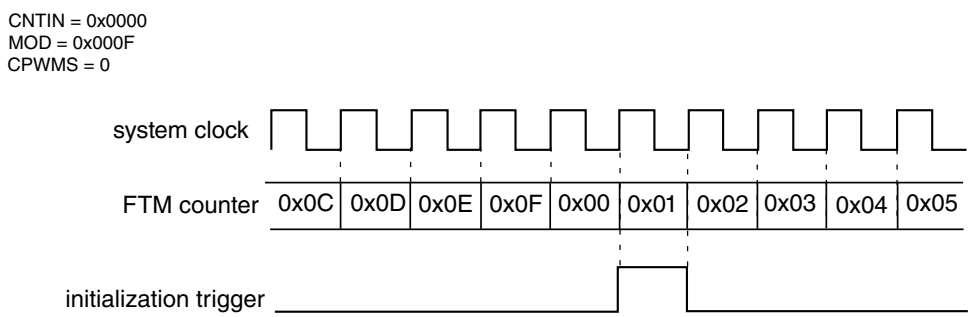


Figure 12-77. Initialization trigger is generated when the FTM counting achieves the CNTIN register value

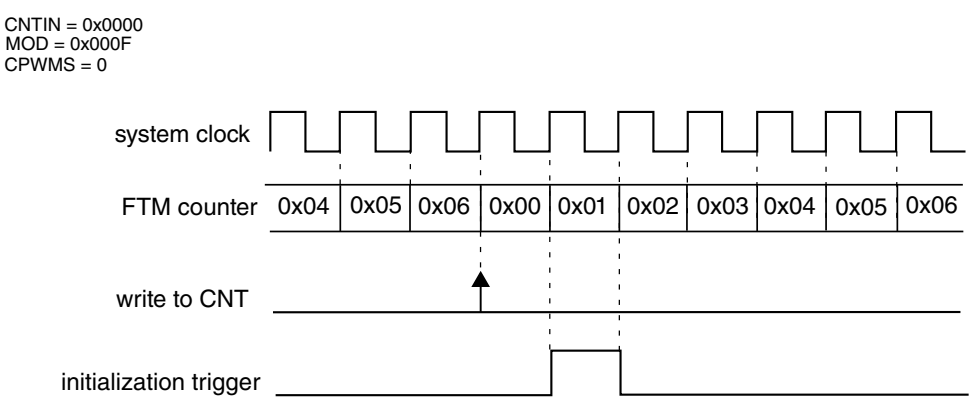


Figure 12-78. Initialization trigger is generated when there is a write to CNT register

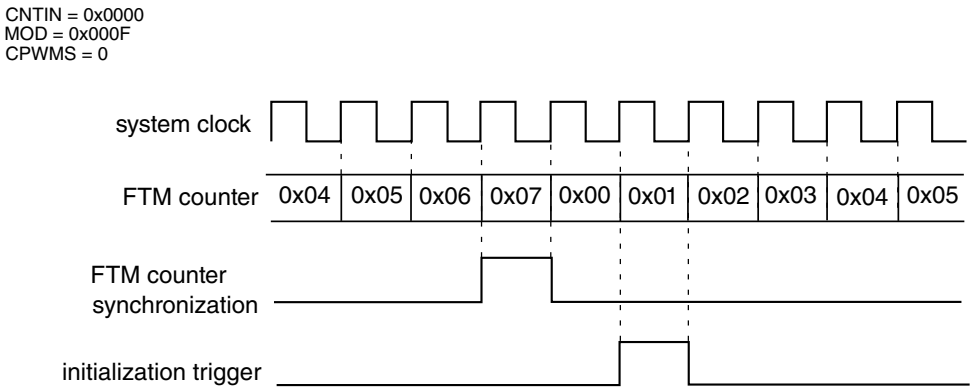


Figure 12-79. Initialization trigger is generated when there is the FTM counter synchronization

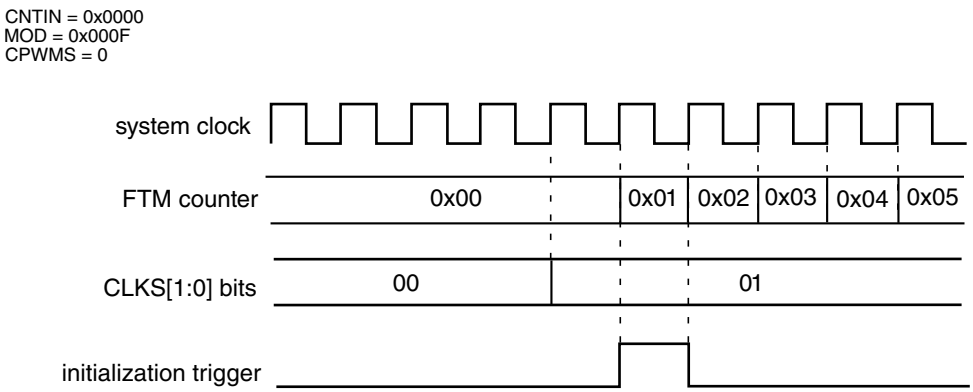
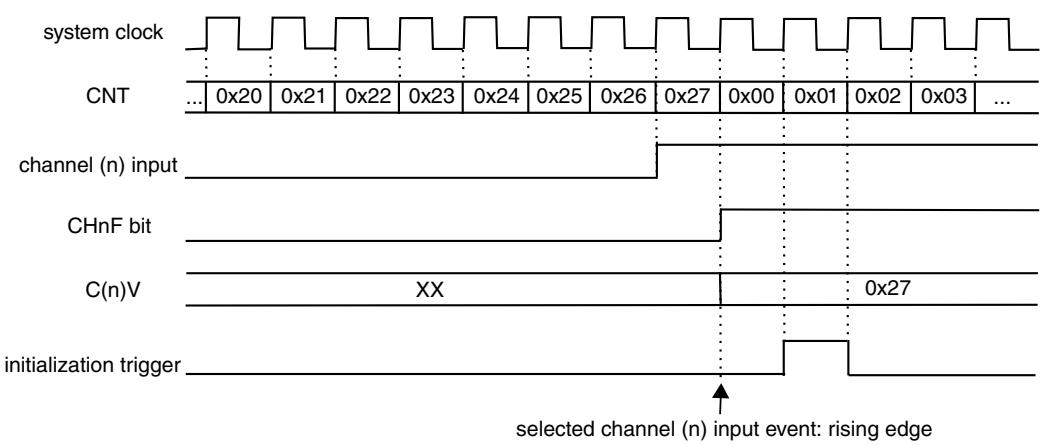


Figure 12-80. Initialization trigger is generated if (CNT = CNTIN), (CLKS[1:0] = 0:0), and a value different from zero is written to CLKS[1:0] bits



NOTE
Channel (n) input after its synchronizer and filter
MOD = 0xFFFF
CNTIN = 0x0000
PS[2:0] = 3'b000
ICRST = 1'b1

Figure 12-81. Initialization trigger is generated if the channel (n) is in Input Capture mode, ICRST = 1 and the selected input capture event occurs in the channel (n) input

The initialization trigger output provides a trigger signal that is used for on-chip modules.

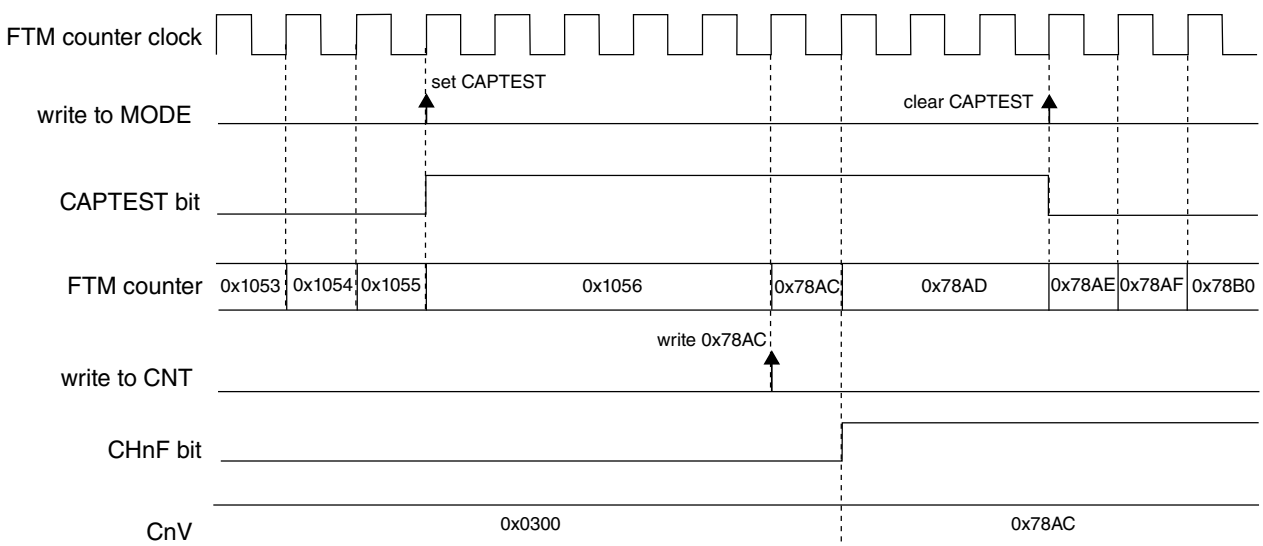
12.2.4.21 Capture Test mode

The Capture Test mode allows to test the CnV registers, the FTM counter and the interconnection logic between the FTM counter and CnV registers.

In this test mode, all channels must be configured for Input Capture mode and FTM counter must be configured to the Up counting.

When the Capture Test mode is enabled (CAPTEST = 1), the FTM counter is frozen and any write to CNT register updates directly the FTM counter; see the following figure. After it was written, all CnV registers are updated with the written value to CNT register and CHnF bits are set. Therefore, the FTM counter is updated with its next value according to its configuration. Its next value depends on CNTIN, MOD, and the written value to FTM counter.

The next reads of CnV registers return the written value to the FTM counter and the next reads of CNT register return FTM counter next value.



NOTE
- FTM counter configuration: (FTMEN = 1), (QUADEN = 0), (CAPTEST = 1), (CPWMS = 0), (CNTIN = 0x0000), and (MOD = 0xFFFF)
- FTM channel n configuration: input capture mode - (DECAPEN = 0), (COMBINE = 0), and (MSnB:MSnA = 0:0)

Figure 12-82. Capture Test mode

12.2.4.22 DMA

The channel generates a DMA transfer request according to DMA and CHnIE bits. See the following table.

Table 12-15. Channel DMA transfer request

DMA	CHnIE	Channel DMA Transfer Request	Channel Interrupt
0	0	The channel DMA transfer request is not generated.	The channel interrupt is not generated.
0	1	The channel DMA transfer request is not generated.	The channel interrupt is generated if (CHnF = 1).
1	0	The channel DMA transfer request is not generated.	The channel interrupt is not generated.
1	1	The channel DMA transfer request is generated if (CHnF = 1).	The channel interrupt is not generated.

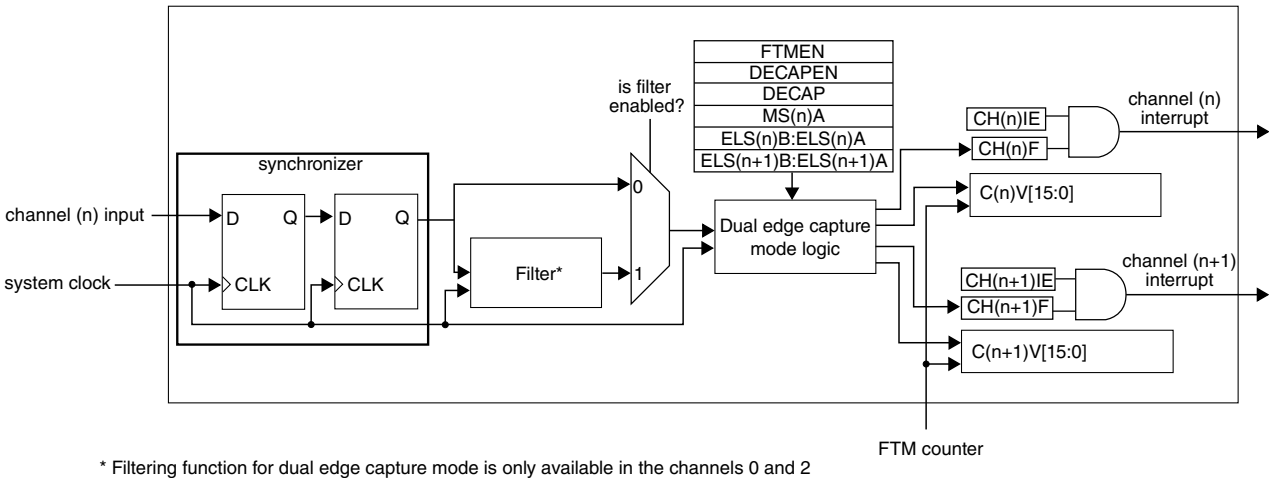
If DMA = 1, the CHnF bit is cleared either by channel DMA transfer done or reading CnSC while CHnF is set and then writing a zero to CHnF bit according to CHnIE bit. See the following table.

Table 12-16. Clear CHnF bit when DMA = 1

CHnIE	How CHnF Bit Can Be Cleared
0	CHnF bit is cleared either when the channel DMA transfer is done or by reading CnSC while CHnF is set and then writing a 0 to CHnF bit.
1	CHnF bit is cleared when the channel DMA transfer is done.

12.2.4.23 Dual Edge Capture mode

The Dual Edge Capture mode is selected if DECAPEN = 1. This mode allows to measure a pulse width or period of the signal on the input of channel (n) of a channel pair. The channel (n) filter can be active in this mode when n is 0 or 2.



* Filtering function for dual edge capture mode is only available in the channels 0 and 2

Figure 12-83. Dual Edge Capture mode block diagram

The MS(n)A bit defines if the Dual Edge Capture mode is one-shot or continuous.

The ELS(n)B:ELS(n)A bits select the edge that is captured by channel (n), and ELS(n +1)B:ELS(n+1)A bits select the edge that is captured by channel (n+1). If both ELS(n)B:ELS(n)A and ELS(n+1)B:ELS(n+1)A bits select the same edge, then it is the period measurement. If these bits select different edges, then it is a pulse width measurement.

In the Dual Edge Capture mode, only channel (n) input is used and channel (n+1) input is ignored.

If the selected edge by channel (n) bits is detected at channel (n) input, then CH(n)F bit is set and the channel (n) interrupt is generated (if CH(n)IE = 1). If the selected edge by channel (n+1) bits is detected at channel (n) input and (CH(n)F = 1), then CH(n+1)F bit is set and the channel (n+1) interrupt is generated (if CH(n+1)IE = 1).

The C(n)V register stores the value of FTM counter when the selected edge by channel (n) is detected at channel (n) input. The C(n+1)V register stores the value of FTM counter when the selected edge by channel (n+1) is detected at channel (n) input.

In this mode, a coherency mechanism ensures coherent data when the C(n)V and C(n +1)V registers are read. The only requirement is that C(n)V must be read before C(n +1)V.

Note

- The CH(n)F, CH(n)IE, MS(n)A, ELS(n)B, and ELS(n)A bits are channel (n) bits.

- The CH(n+1)F, CH(n+1)IE, MS(n+1)A, ELS(n+1)B, and ELS(n+1)A bits are channel (n+1) bits.
- The Dual Edge Capture mode must be used with ELS(n)B:ELS(n)A = 0:1 or 1:0, ELS(n+1)B:ELS(n+1)A = 0:1 or 1:0 and the FTM counter in Free running counter.

12.2.4.23.1 One-Shot Capture mode

The One-Shot Capture mode is selected when (DECAPEN = 1), and (MS(n)A = 0). In this capture mode, only one pair of edges at the channel (n) input is captured. The ELS(n)B:ELS(n)A bits select the first edge to be captured, and ELS(n+1)B:ELS(n+1)A bits select the second edge to be captured.

The edge captures are enabled while DECAP bit is set. For each new measurement in One-Shot Capture mode, first the CH(n)F and CH(n+1) bits must be cleared, and then the DECAP bit must be set.

In this mode, the DECAP bit is automatically cleared by FTM when the edge selected by channel (n+1) is captured. Therefore, while DECAP bit is set, the one-shot capture is in process. When this bit is cleared, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers.

Similarly, when the CH(n+1)F bit is set, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers.

12.2.4.23.2 Continuous Capture mode

The Continuous Capture mode is selected when (DECAPEN = 1), and (MS(n)A = 1). In this capture mode, the edges at the channel (n) input are captured continuously. The ELS(n)B:ELS(n)A bits select the initial edge to be captured, and ELS(n+1)B:ELS(n+1)A bits select the final edge to be captured.

The edge captures are enabled while DECAP bit is set. For the initial use, first the CH(n)F and CH(n+1)F bits must be cleared, and then DECAP bit must be set to start the continuous measurements.

When the CH(n+1)F bit is set, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers. The latest captured values are always available in these registers even after the DECAP bit is cleared.

In this mode, it is possible to clear only the CH(n+1)F bit. Therefore, when the CH(n+1)F bit is set again, the latest captured values are available in C(n)V and C(n+1)V registers.

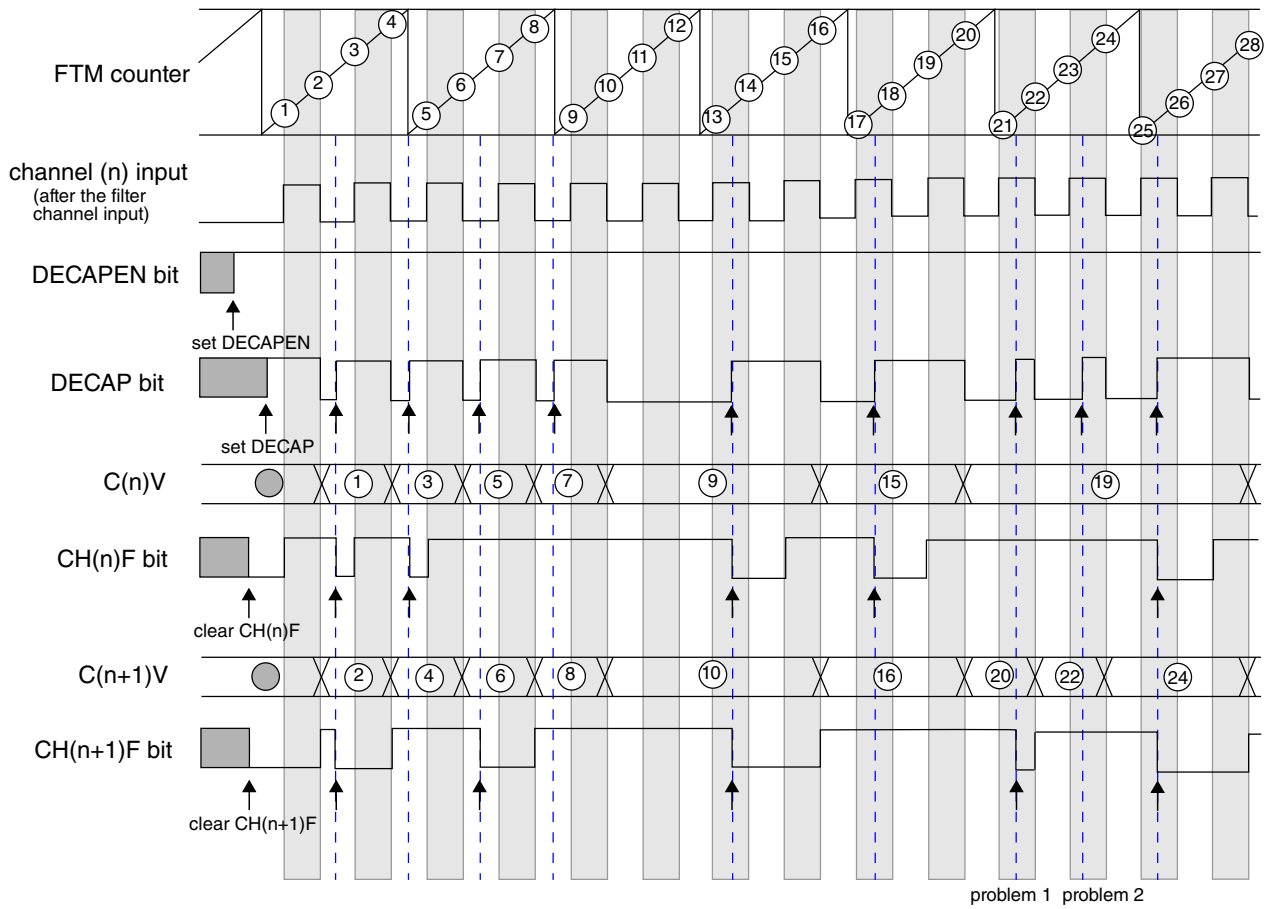
For a new sequence of the measurements in the Dual Edge Capture – Continuous mode, clear the CH(n)F and CH(n+1)F bits to start new measurements.

12.2.4.23.3 Pulse width measurement

If the channel (n) is configured to capture rising edges (ELS(n)B:ELS(n)A = 0:1) and the channel (n+1) to capture falling edges (ELS(n+1)B:ELS(n+1)A = 1:0), then the positive polarity pulse width is measured. If the channel (n) is configured to capture falling edges (ELS(n)B:ELS(n)A = 1:0) and the channel (n+1) to capture rising edges (ELS(n +1)B:ELS(n+1)A = 0:1), then the negative polarity pulse width is measured.

The pulse width measurement can be made in One-Shot Capture mode or Continuous Capture mode.

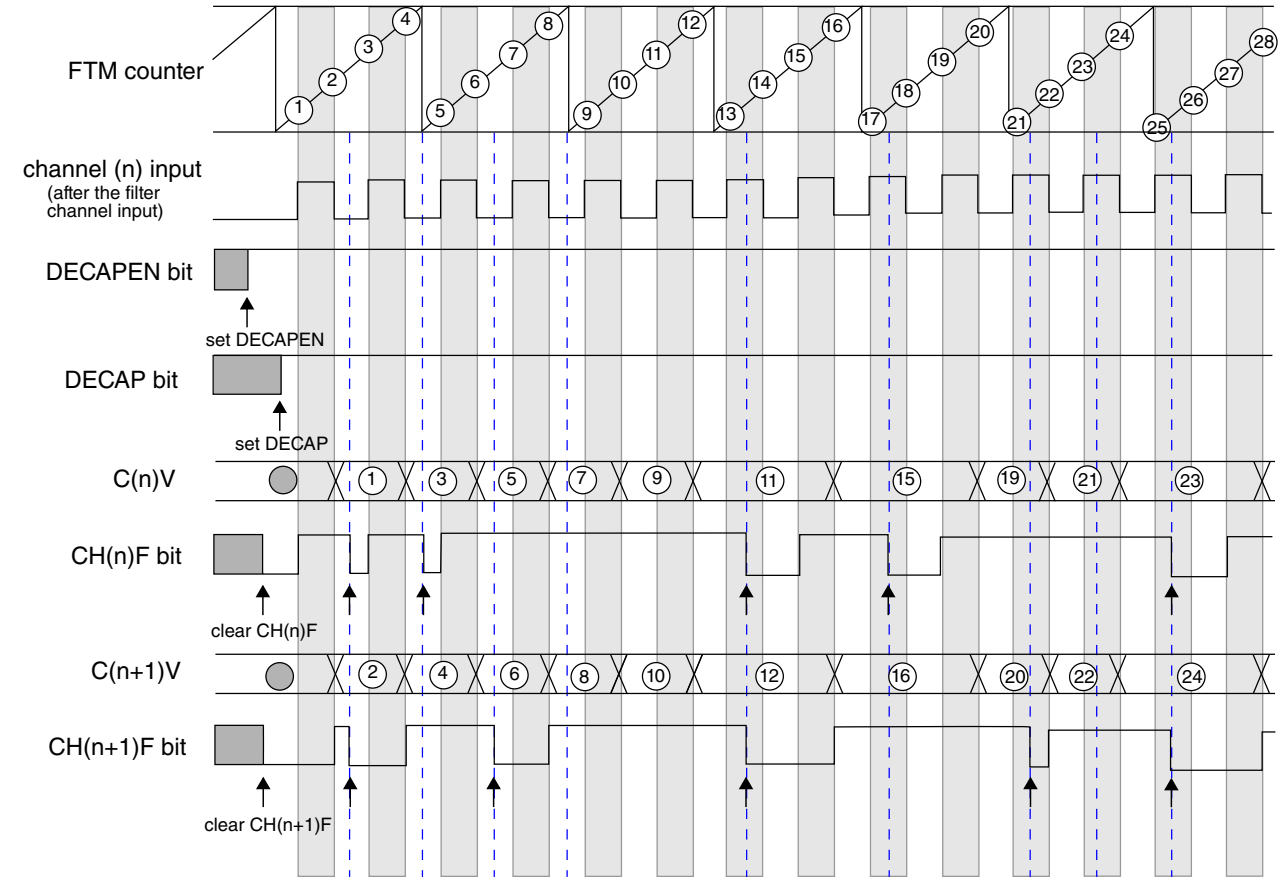
The following figure shows an example of the Dual Edge Capture – One-Shot mode used to measure the positive polarity pulse width. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. The DECAP bit is set to enable the measurement of next positive polarity pulse width. The CH(n)F bit is set when the first edge of this pulse is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set and DECAP bit is cleared when the second edge of this pulse is detected, that is, the edge selected by ELS(n+1)B:ELS(n+1)A bits. Both DECAP and CH(n+1)F bits indicate when two edges of the pulse were captured and the C(n)V and C(n+1)V registers are ready for reading.



Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.
- Problem 1: channel (n) input = 1, set DECAP, not clear CH(n)F, and clear CH(n+1)F.
- Problem 2: channel (n) input = 1, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.

Figure 12-84. Dual Edge Capture – One-Shot mode for positive polarity pulse width measurement

The following figure shows an example of the Dual Edge Capture – Continuous mode used to measure the positive polarity pulse width. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. While the DECAP bit is set the configured measurements are made. The CH(n)F bit is set when the first edge of the positive polarity pulse is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set when the second edge of this pulse is detected, that is, the edge selected by ELS(n +1)B:ELS(n+1)A bits. The CH(n+1)F bit indicates when two edges of the pulse were captured and the C(n)V and C(n+1)V registers are ready for reading.



Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.

Figure 12-85. Dual Edge Capture – Continuous mode for positive polarity pulse width measurement

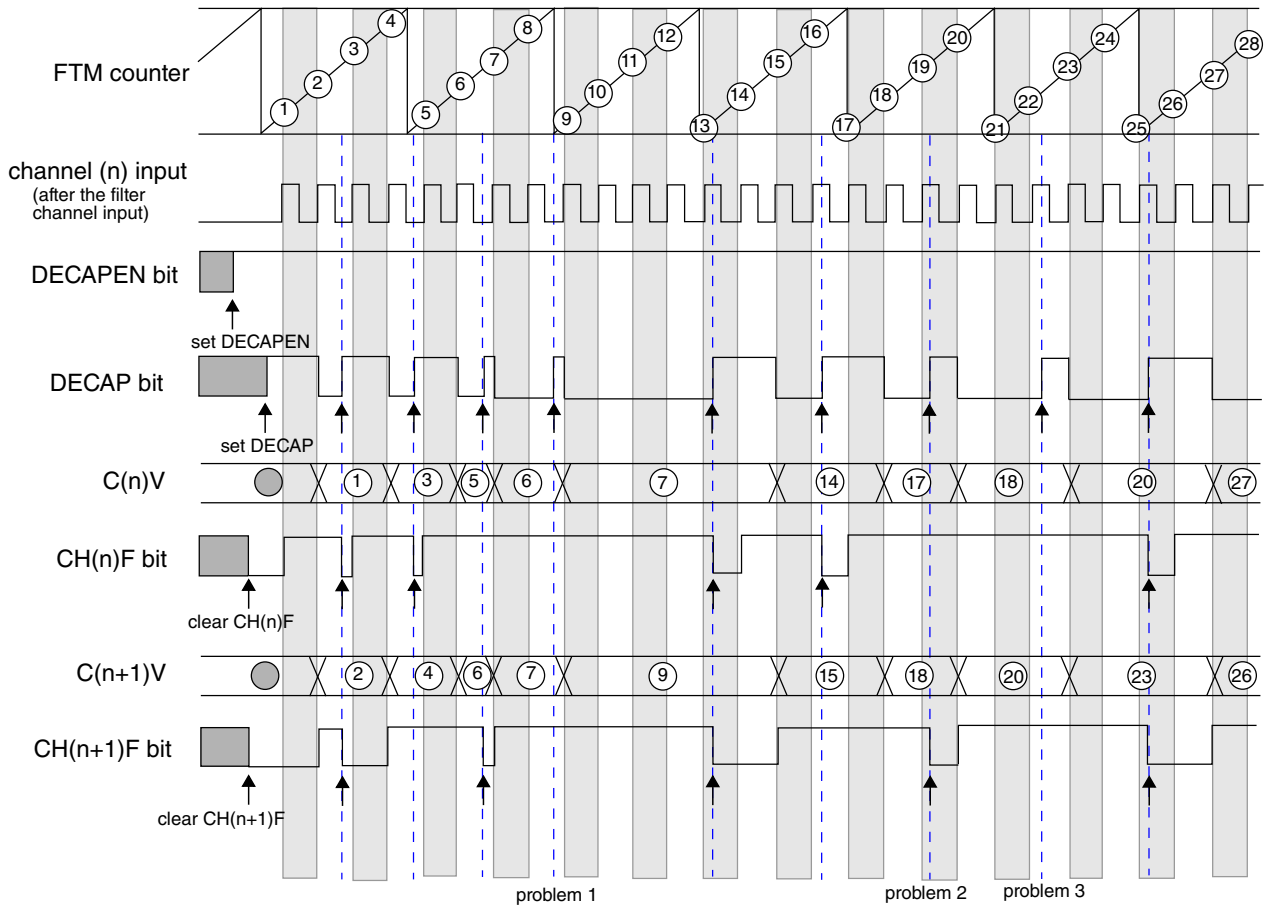
12.2.4.23.4 Period measurement

If the channels (n) and (n+1) are configured to capture consecutive edges of the same polarity, then the period of the channel (n) input signal is measured. If both channels (n) and (n+1) are configured to capture rising edges (ELS(n)B:ELS(n)A = 0:1 and ELS(n +1)B:ELS(n+1)A = 0:1), then the period between two consecutive rising edges is measured. If both channels (n) and (n+1) are configured to capture falling edges (ELS(n)B:ELS(n)A = 1:0 and ELS(n+1)B:ELS(n+1)A = 1:0), then the period between two consecutive falling edges is measured.

The period measurement can be made in One-Shot Capture mode or Continuous Capture mode.

The following figure shows an example of the Dual Edge Capture – One-Shot mode used to measure the period between two consecutive rising edges. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. The DECAP bit is set to enable the

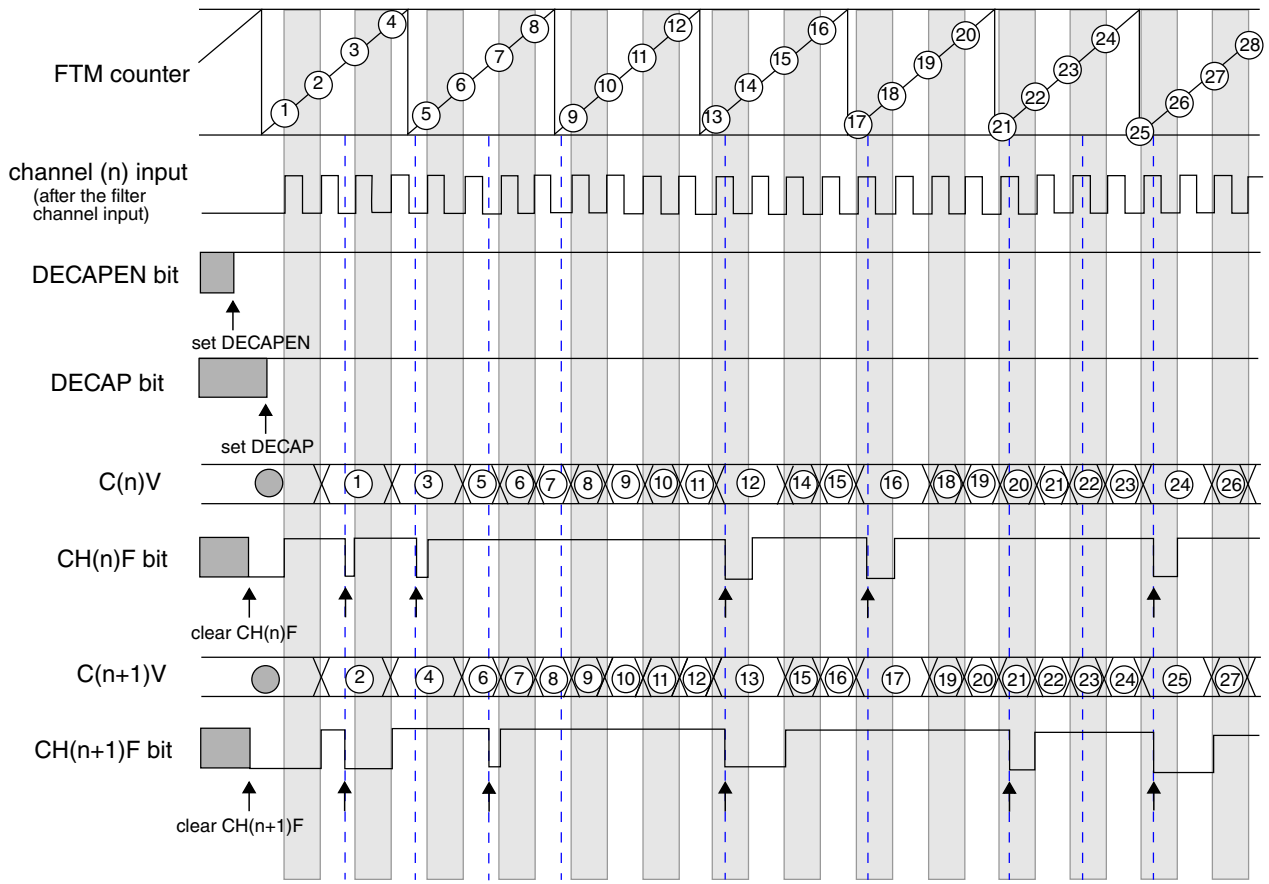
measurement of next period. The CH(n)F bit is set when the first rising edge is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set and DECAP bit is cleared when the second rising edge is detected, that is, the edge selected by ELS(n +1)B:ELS(n+1)A bits. Both DECAP and CH(n+1)F bits indicate when two selected edges were captured and the C(n)V and C(n+1)V registers are ready for reading.



Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.
- Problem 1: channel (n) input = 0, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.
- Problem 2: channel (n) input = 1, set DECAP, not clear CH(n)F, and clear CH(n+1)F.
- Problem 3: channel (n) input = 1, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.

Figure 12-86. Dual Edge Capture – One-Shot mode to measure of the period between two consecutive rising edges

The following figure shows an example of the Dual Edge Capture – Continuous mode used to measure the period between two consecutive rising edges. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. While the DECAP bit is set the configured measurements are made. The CH(n)F bit is set when the first rising edge is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set when the second rising edge is detected, that is, the edge selected by ELS(n+1)B:ELS(n +1)A bits. The CH(n+1)F bit indicates when two edges of the period were captured and the C(n)V and C(n+1)V registers are ready for reading.



Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.

Figure 12-87. Dual Edge Capture – Continuous mode to measure of the period between two consecutive rising edges

12.2.4.23.5 Read coherency mechanism

The Dual Edge Capture mode implements a read coherency mechanism between the FTM counter value captured in C(n)V and C(n+1)V registers. The read coherency mechanism is illustrated in the following figure. In this example, the channels (n) and (n +1) are in Dual Edge Capture – Continuous mode for positive polarity pulse width measurement. Thus, the channel (n) is configured to capture the FTM counter value when there is a rising edge at channel (n) input signal, and channel (n+1) to capture the FTM counter value when there is a falling edge at channel (n) input signal.

When a rising edge occurs in the channel (n) input signal, the FTM counter value is captured into channel (n) capture buffer. The channel (n) capture buffer value is transferred to C(n)V register when a falling edge occurs in the channel (n) input signal.

C(n)V register has the FTM counter value when the previous rising edge occurred, and the channel (n) capture buffer has the FTM counter value when the last rising edge occurred.

When a falling edge occurs in the channel (n) input signal, the FTM counter value is captured into channel (n+1) capture buffer. The channel (n+1) capture buffer value is transferred to C(n+1)V register when the C(n)V register is read.

In the following figure, the read of C(n)V returns the FTM counter value when the event 1 occurred and the read of C(n+1)V returns the FTM counter value when the event 2 occurred.

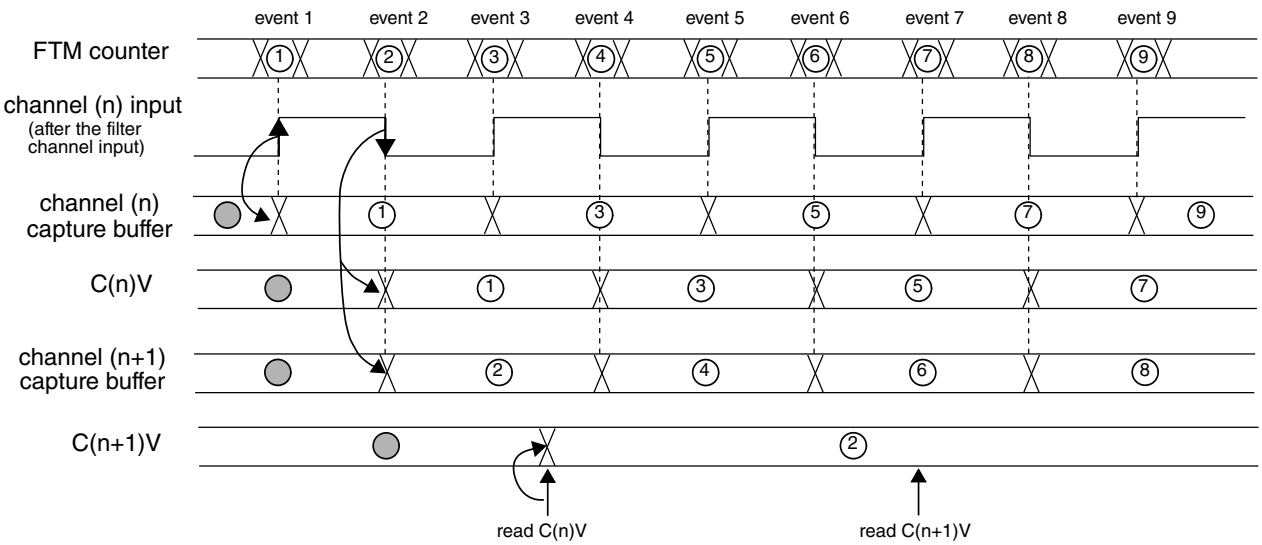


Figure 12-88. Dual Edge Capture mode read coherency mechanism

C(n)V register must be read prior to C(n+1)V register in dual edge capture one-shot and continuous modes for the read coherency mechanism works properly.

12.2.4.24 Quadrature Decoder mode

The Quadrature Decoder mode is selected if (QUADEN = 1). The Quadrature Decoder mode uses the input signals phase A and B to control the FTM counter increment and decrement. The following figure shows the quadrature decoder block diagram.

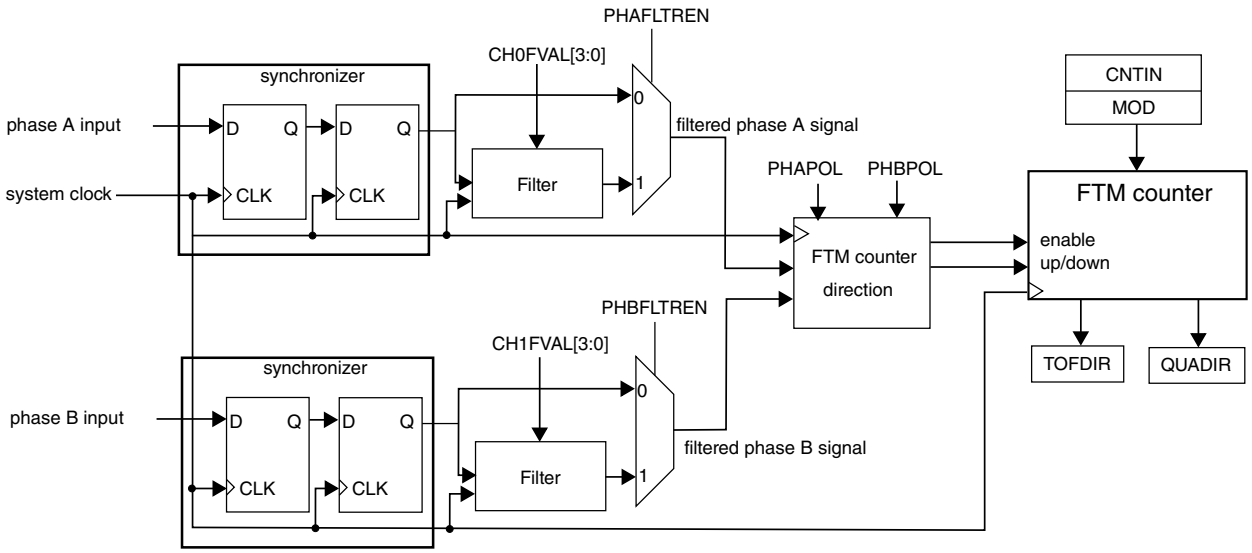


Figure 12-89. Quadrature Decoder block diagram

Each one of input signals phase A and B has a filter that is equivalent to the filter used in the channels input; Filter for Input Capture mode. The phase A input filter is enabled by PHAFLTREN bit and this filter's value is defined by CH0FVAL[3:0] bits (CH(n)FVAL[3:0] bits in FILTER0 register). The phase B input filter is enabled by PHBFLTREN bit and this filter's value is defined by CH1FVAL[3:0] bits (CH(n +1)FVAL[3:0] bits in FILTER0 register).

Except for CH0FVAL[3:0] and CH1FVAL[3:0] bits, no channel logic is used in Quadrature Decoder mode.

Note

Notice that the FTM counter is clocked by the phase A and B input signals when quadrature decoder mode is selected. Therefore it is expected that the Quadrature Decoder be used only with the FTM channels in input capture or output compare modes.

Note

An edge at phase A must not occur together an edge at phase B and vice-versa.

The PHAPOL bit selects the polarity of the phase A input, and the PHBPOL bit selects the polarity of the phase B input.

The QUADMODE selects the encoding mode used in the Quadrature Decoder mode. If QUADMODE = 1, then the count and direction encoding mode is enabled; see the following figure. In this mode, the phase B input value indicates the counting direction, and the phase A input defines the counting rate. The FTM counter is updated when there is a rising edge at phase A input signal.

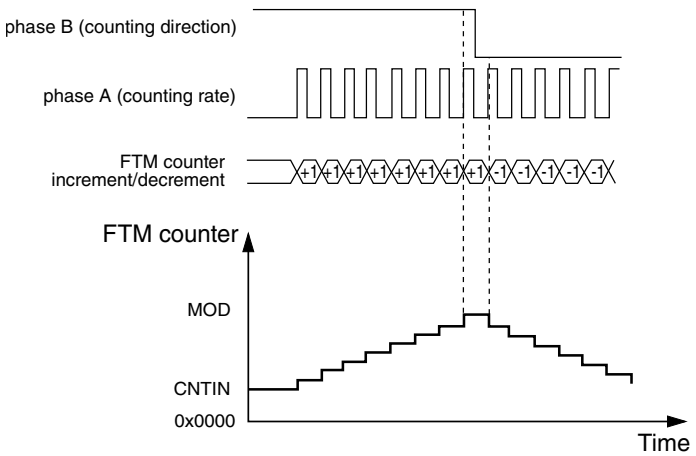


Figure 12-90. Quadrature Decoder – Count and Direction Encoding mode

If QUADMODE = 0, then the Phase A and Phase B Encoding mode is enabled; see the following figure. In this mode, the relationship between phase A and B signals indicates the counting direction, and phase A and B signals define the counting rate. The FTM counter is updated when there is an edge either at the phase A or phase B signals.

If PHAPOL = 0 and PHBPOL = 0, then the FTM counter increment happens when:

- there is a rising edge at phase A signal and phase B signal is at logic zero;
- there is a rising edge at phase B signal and phase A signal is at logic one;
- there is a falling edge at phase B signal and phase A signal is at logic zero;
- there is a falling edge at phase A signal and phase B signal is at logic one;

and the FTM counter decrement happens when:

- there is a falling edge at phase A signal and phase B signal is at logic zero;
- there is a falling edge at phase B signal and phase A signal is at logic one;
- there is a rising edge at phase B signal and phase A signal is at logic zero;
- there is a rising edge at phase A signal and phase B signal is at logic one.

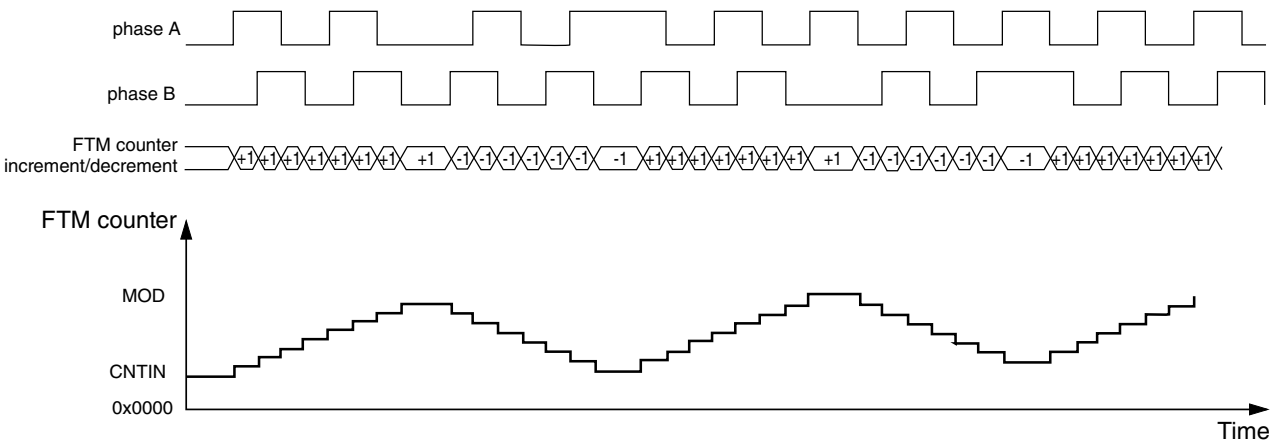


Figure 12-91. Quadrature Decoder – Phase A and Phase B Encoding mode

The following figure shows the FTM counter overflow in up counting. In this case, when the FTM counter changes from MOD to CNTIN, TOF and TOFDIR bits are set. TOF bit indicates the FTM counter overflow occurred. TOFDIR indicates the counting was up when the FTM counter overflow occurred.

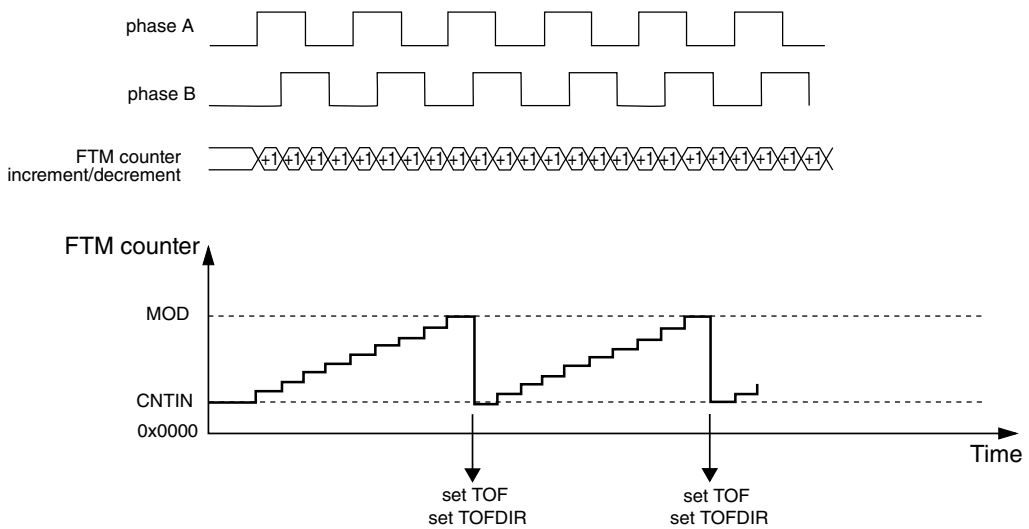


Figure 12-92. FTM Counter overflow in up counting for Quadrature Decoder mode

The following figure shows the FTM counter overflow in down counting. In this case, when the FTM counter changes from CNTIN to MOD, TOF bit is set and TOFDIR bit is cleared. TOF bit indicates the FTM counter overflow occurred. TOFDIR indicates the counting was down when the FTM counter overflow occurred.

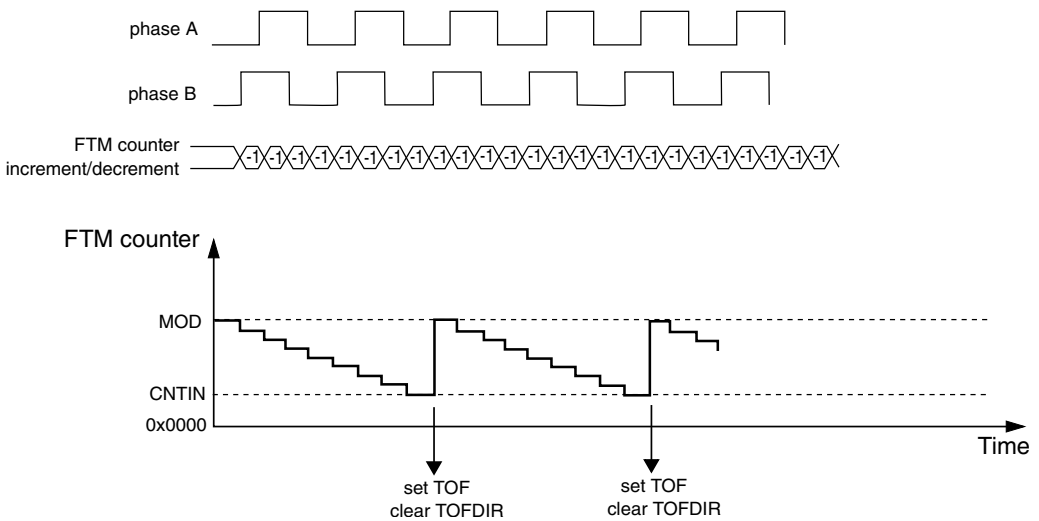


Figure 12-93. FTM counter overflow in down counting for Quadrature Decoder mode

12.2.4.24.1 Quadrature Decoder boundary conditions

The following figures show the FTM counter responding to motor jittering typical in motor position control applications.

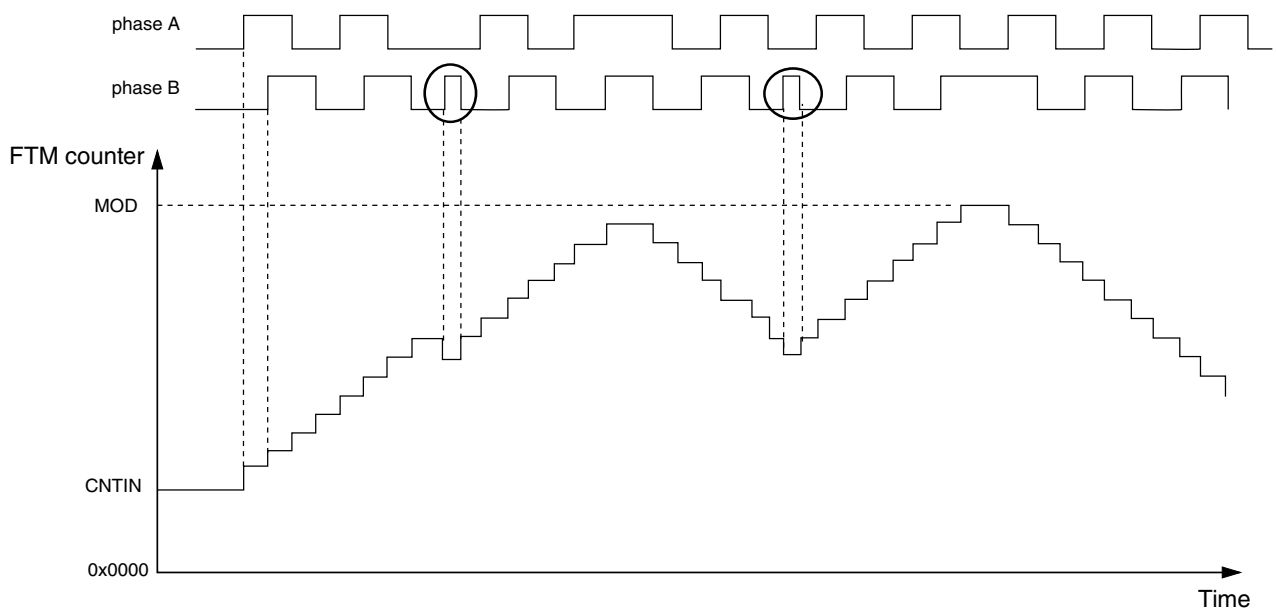


Figure 12-94. Motor position jittering in a mid count value

The following figure shows motor jittering produced by the phase B and A pulses respectively:

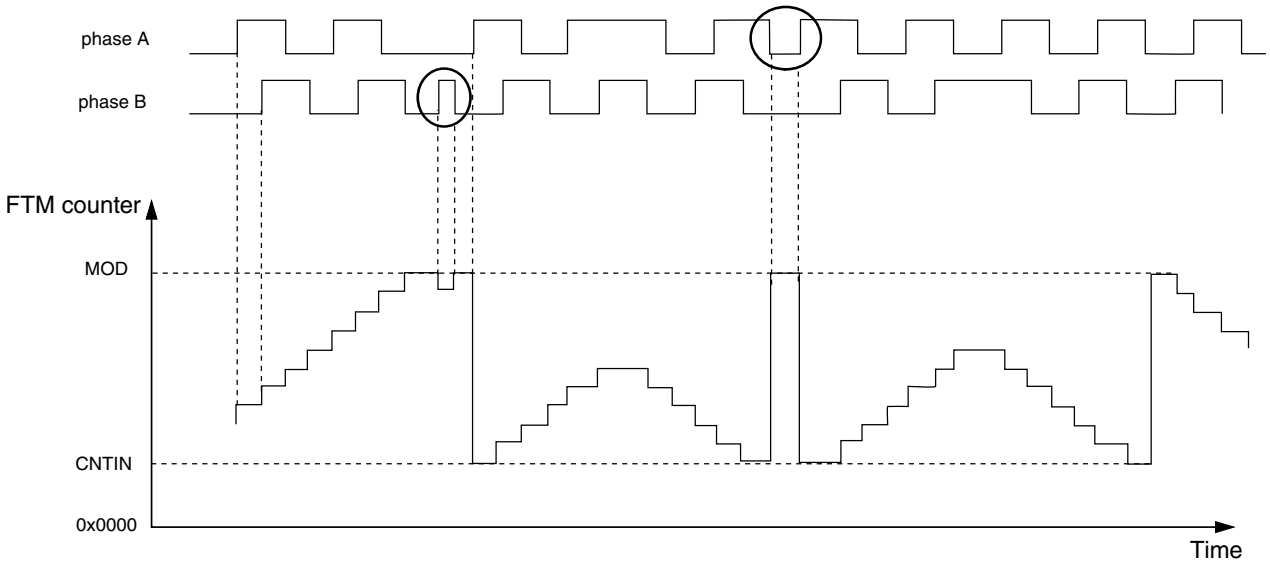


Figure 12-95. Motor position jittering near maximum and minimum count value

The first highlighted transition causes a jitter on the FTM counter value near the maximum count value (MOD). The second indicated transition occurs on phase A and causes the FTM counter transition between the maximum and minimum count values which are defined by MOD and CNTIN registers.

The appropriate settings of the phase A and phase B input filters are important to avoid glitches that may cause oscillation on the FTM counter value. The preceding figures show examples of oscillations that can be caused by poor input filter setup. Thus, it is important to guarantee a minimum pulse width to avoid these oscillations.

12.2.4.25 Intermediate load

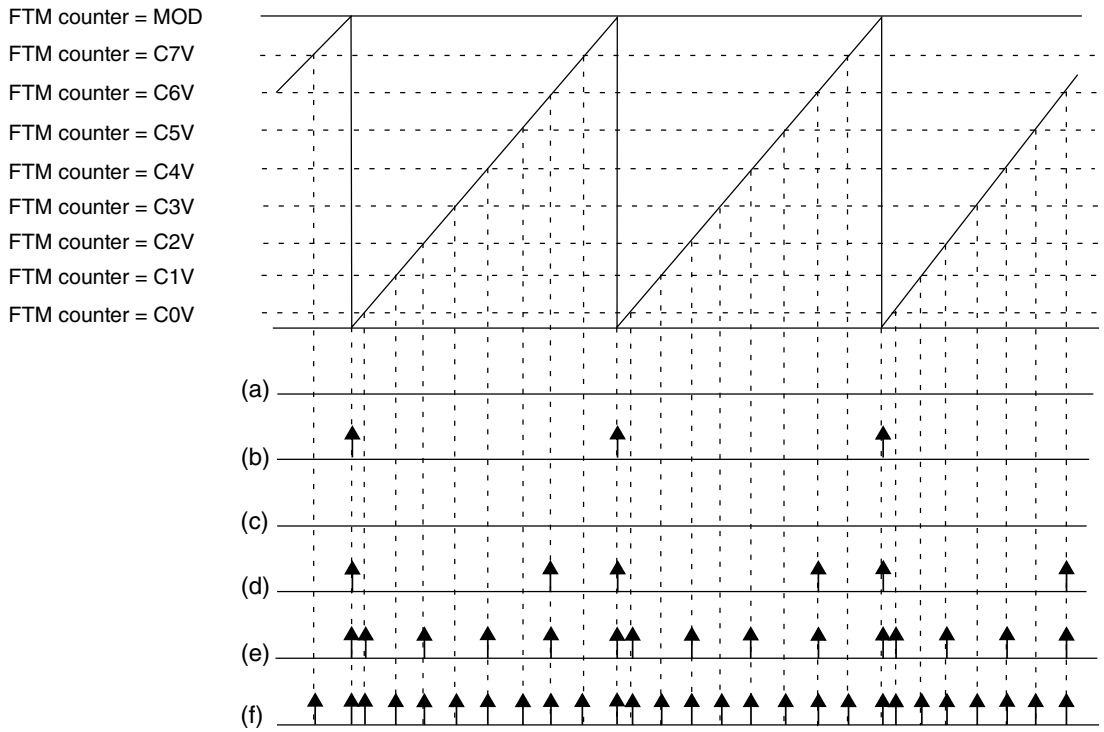
The PWMLOAD register allows to update the MOD, CNTIN, and C(n)V registers with the content of the register buffer at a defined load point. In this case, it is not required to use the PWM synchronization.

There are multiple possible loading points for intermediate load:

Table 12-17. When possible loading points are enabled

Loading point	Enabled
When the FTM counter wraps from MOD value to CNTIN value	Always
At the channel (j) match (FTM counter = C(j)V)	When CHjSEL = 1

The following figure shows some examples of enabled loading points.



NOTE

(a) LDOK = 0, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(b) LDOK = 1, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(c) LDOK = 0, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 1, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(d) LDOK = 1, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 1, CH7SEL = 0
(e) LDOK = 1, CH0SEL = 1, CH1SEL = 0, CH2SEL = 1, CH3SEL = 0, CH4SEL = 1, CH5SEL = 0, CH6SEL = 1, CH7SEL = 0
(f) LDOK = 1, CH0SEL = 1, CH1SEL = 1, CH2SEL = 1, CH3SEL = 1, CH4SEL = 1, CH5SEL = 1, CH6SEL = 1, CH7SEL = 1

Figure 12-96. Loading points for intermediate load

After enabling the loading points, the LDOK bit must be set for the load to occur. In this case, the load occurs at the next enabled loading point according to the following conditions:

Table 12-18. Conditions for loads occurring at the next enabled loading point

When a new value was written	Then
To the MOD register	The MOD register is updated with its write buffer value.
To the CNTIN register and CNTINC = 1	The CNTIN register is updated with its write buffer value.
To the C(n)V register and SYNCENm = 1 – where m indicates the pair channels (n) and (n+1)	The C(n)V register is updated with its write buffer value.
To the C(n+1)V register and SYNCENm = 1 – where m indicates the pair channels (n) and (n+1)	The C(n+1)V register is updated with its write buffer value.

NOTE

- If ELSjB and ELSjA bits are different from zero, then the channel (j) output signal is generated according to the configured output mode. If ELSjB and ELSjA bits are zero,

then the generated signal is not available on channel (j) output.
- If CHjIE = 1, then the channel (j) interrupt is generated when the channel (j) match occurs.
- At the intermediate load neither the channels outputs nor the FTM counter are changed. Software must set the intermediate load at a safe point in time.

12.2.4.26 Global time base (GTB)

The global time base (GTB) is a FTM function that allows the synchronization of multiple FTM modules on a chip. The following figure shows an example of the GTB feature used to synchronize two FTM modules. In this case, the FTM A and B channels can behave as if just one FTM module was used, that is, a global time base.

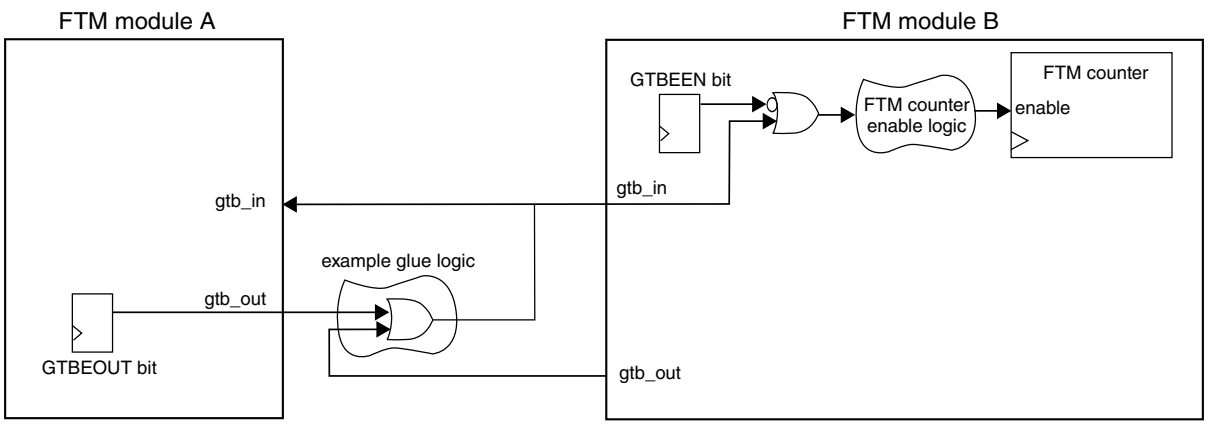


Figure 12-97. Global time base (GTB) block diagram

The GTB functionality is implemented by the GTBEEN and GTBEOUT bits in the CONF register, the input signal *gtb_in*, and the output signal *gtb_out*. The GTBEEN bit enables gtb_in to control the FTM counter enable signal:

- If GTBEEN = 0, each one of FTM modules works independently according to their configured mode.
- If GTBEEN = 1, the FTM counter update is enabled only when gtb_in is 1.

In the configuration described in the preceding figure, FTM modules A and B have their FTM counters enabled if at least one of the gtb_out signals from one of the FTM modules is 1. There are several possible configurations for the interconnection of the gtb_in and gtb_out signals, represented by the example glue logic shown in the figure. Note that these configurations are chip-dependent and implemented outside of the FTM modules. See the chip-specific FTM information for the chip's specific implementation.

NOTE

- In order to use the GTB signals to synchronize the FTM counter of different FTM modules, the configuration of each FTM module should guarantee that its FTM counter starts counting as soon as the gtb_in signal is 1.
- The GTB feature does not provide continuous synchronization of FTM counters, meaning that the FTM counters may lose synchronization during FTM operation. The GTB feature only allows the FTM counters to *start* their operation synchronously.

12.2.4.26.1 Enabling the global time base (GTB)

To enable the GTB feature, follow these steps for each participating FTM module:

1. Stop the FTM counter: Write 00b to SC[CLKS].
2. Program the FTM to the intended configuration. The FTM counter mode needs to be consistent across all participating modules.
3. Write 1 to CONF[GTBEEN] and write 0 to CONF[GTBEOUT] at the same time.
4. Select the intended FTM counter clock source in SC[CLKS]. The clock source needs to be consistent across all participating modules.
5. Reset the FTM counter: Write any value to the CNT register.

To initiate the GTB feature in the configuration described in the preceding figure, write 1 to CONF[GTBEOUT] in the FTM module used as the time base.

12.2.5 Reset overview

The FTM is reset whenever any chip reset occurs.

When the FTM exits from reset:

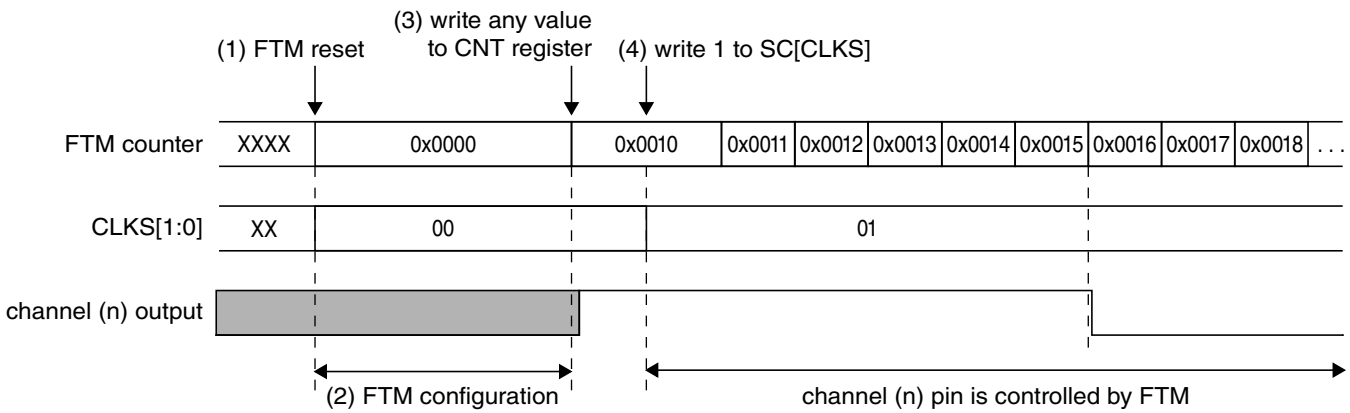
- the FTM counter and the prescaler counter are zero and are stopped (CLKS[1:0] = 00b);
- the timer overflow interrupt is zero, see Timer Overflow Interrupt;
- the channels interrupts are zero, see Channel (n) Interrupt;
- the channels are in input capture mode, see Input Capture mode;
- the channels outputs are zero;
- the channels pins are not controlled by FTM (ELS(n)B:ELS(n)A = 0:0) (See the table in the description of CnSC register).

The following figure shows the FTM behavior after the reset. At the reset (item 1), the FTM counter is disabled (see the description of the CLKS field in the Status and Control register), its value is updated to zero and the pins are not controlled by FTM (See the table in the description of CnSC register).

After the reset, the FTM should be configurated (item 2). It is necessary to define the FTM counter mode, the FTM counting limits (MOD and CNTIN registers value), the channels mode and CnV registers value according to the channels mode.

Thus, it is recommended to write any value to CNT register (item 3). This write updates the FTM counter with the CNTIN register value and the channels output with its initial value (except for channels in output compare mode) (Counter reset).

The next step is to select the FTM counter clock by the CLKS[1:0] bits (item 4). It is important to highlight that the pins are only controlled by FTM when CLKS[1:0] bits are different from zero (See the table in the description of CnSC register).



NOTES:
- CNTIN = 0x0010
- Channel (n) is in low-true combine mode with CNTIN < C(n)V < C(n+1)V < MOD
- C(n)V = 0x0015

Figure 12-98. FTM behavior after reset when the channel (n) is in Combine mode

The following figure shows an example when the channel (n) is in Output Compare mode and the channel (n) output is toggled when there is a match. In the Output Compare mode, the channel output is not updated to its initial value when there is a write to CNT register (item 3). In this case, use the software output control (Software output control) or the initialization (Initialization) to update the channel output to the selected value (item 4).

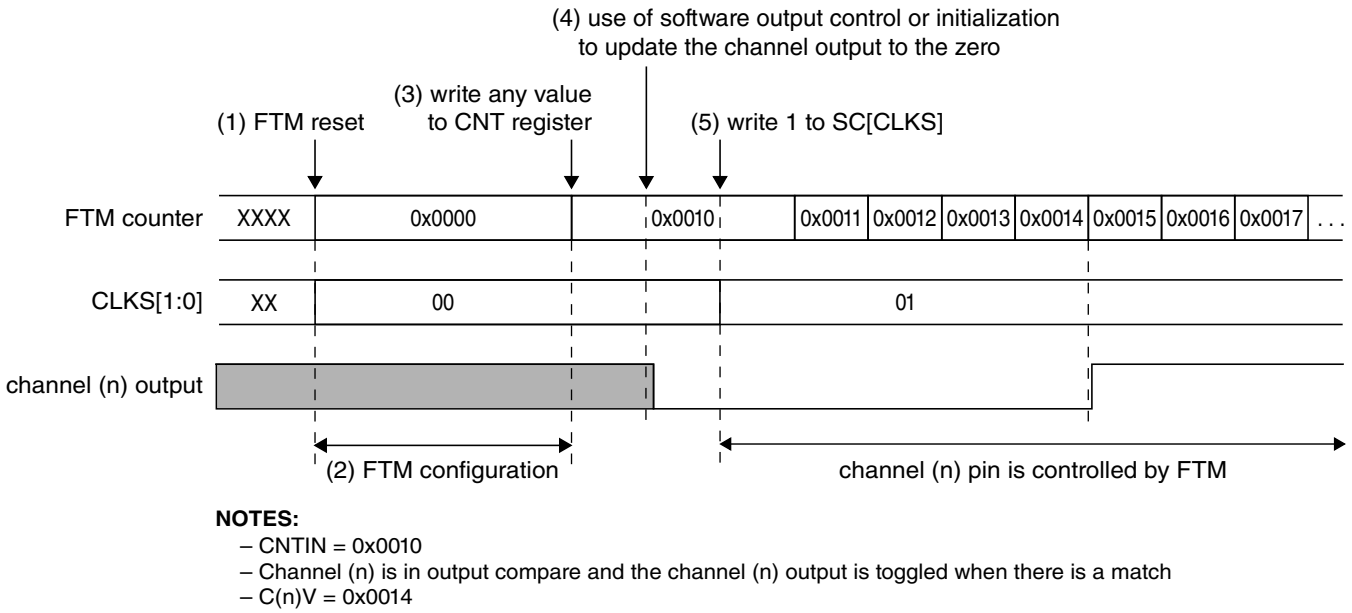


Figure 12-99. FTM behavior after reset when the channel (n) is in Output Compare mode

12.2.6 FTM Interrupts

12.2.6.1 Timer Overflow Interrupt

The timer overflow interrupt is generated when (TOIE = 1) and (TOF = 1).

12.2.6.2 Channel (n) Interrupt

The channel (n) interrupt is generated when (CHnIE = 1) and (CHnF = 1).

12.2.7 Initialization Procedure

The following initialization procedure is recommended to configure the FlexTimer operation. This procedure can also be used to do a new configuration of the FlexTimer operation.

- Define the POL bits.
- Mask the channels outputs using SYNCHOM = 0. Two clocks after the write to OUTMASK, the channels output are in the safe value.
- (Re)Configuration FTM counter and channels to generation of periodic signals - Disable the clock. If the selected mode is Quadrature Decoder, then disable this mode. Examples of the (re)configuration:

- Write to MOD.
- Write to CNTIN.
- Select OC, EPWM, CPWM, Combine, Complement modes for all channels that will be used
- Select the high-true and low-true channels modes.
- Write to CnV for all channels that will be used .
- (Re)Configure deadtime.
- Do not use the SWOC without SW synchronization (see item 6).
- Do not use the Inverting without SW synchronization (see item 6).
- Do not use the Initialization.
- Do not change the polarity control.
- Do not configure the HW synchronization

- Write any value to CNT. The FTM Counter is reset and the channels output are updated according to new configuration.
- Enable the clock. Write to CLKS[1:0] bits a value different from zero. If in the Quadrature Decoder mode, enable this mode.
- Configure the SW synchronization for SWOC (if it is necessary), Inverting (if it is necessary) and Output Mask (always)
  - Select synchronization for Output Mask Write to SYNC (SWSYNC = 0, TRIG2 = 0, TRIG1 = 0, TRIG0 = 0, SYNCHOM = 1, REINIT = 0, CNTMAX = 0, CNTMIN = 0)
  - Write to SYNCONF.
    - HW Synchronization can not be enabled (HWSOC = 0, HWINVC = 0, HWOM = 0, HWWRBUF = 0, HWRSTCNT = 0, HWTRIGMODE = 0).
    - SW Synchronization for SWOC (if it is necessary): SWSOC = [0/1] and SWOC = [0/1].
    - SW Synchronization for Inverting (if it is necessary): SWINVC = [0/1] and INVC = [0/1].
    - SW Synchronization for SWOM (always): SWOM = 1. No enable the SW Synchronization for write buffers (because the writes to registers with write buffer are done using CLKS[1:0] = 2’b00): SWWRBUF = 0 and CNTINC = 0 .
    - SW Synchronization for counter reset (always): SWRSTCNT = 1.
    - Enhanced synchronization (always): SYNCMODE = 1
  - If the SWOC is used (SWSOC = 1 and SWOC = 1), then write to SWOCTRL register.
  - If the Inverting is used (SWINVC = 1 and INVC = 1), then write to INVCTRL register.
  - Write to OUTMASK to enable the masked channels.
- Generate the Software Trigger Write to SYNC (SWSYNC = 1, TRIG2 = 0, TRIG1 = 0, TRIG0 = 0, SYNCHOM = 1, REINIT = 0, CNTMAX = 0, CNTMIN = 0)

12.3 Pulse Width Modulation (PWM)

12.3.1 Overview

The Pulse Width Modulation (PWM) has a 16-bit counter, and is optimized to generate sound from stored sample audio images and it can also generate tones. It uses 16-bit resolution and a 4 x 16 data FIFO.

This section presents an overview of the PWM. A block diagram of the PWM module is shown in the figure below.

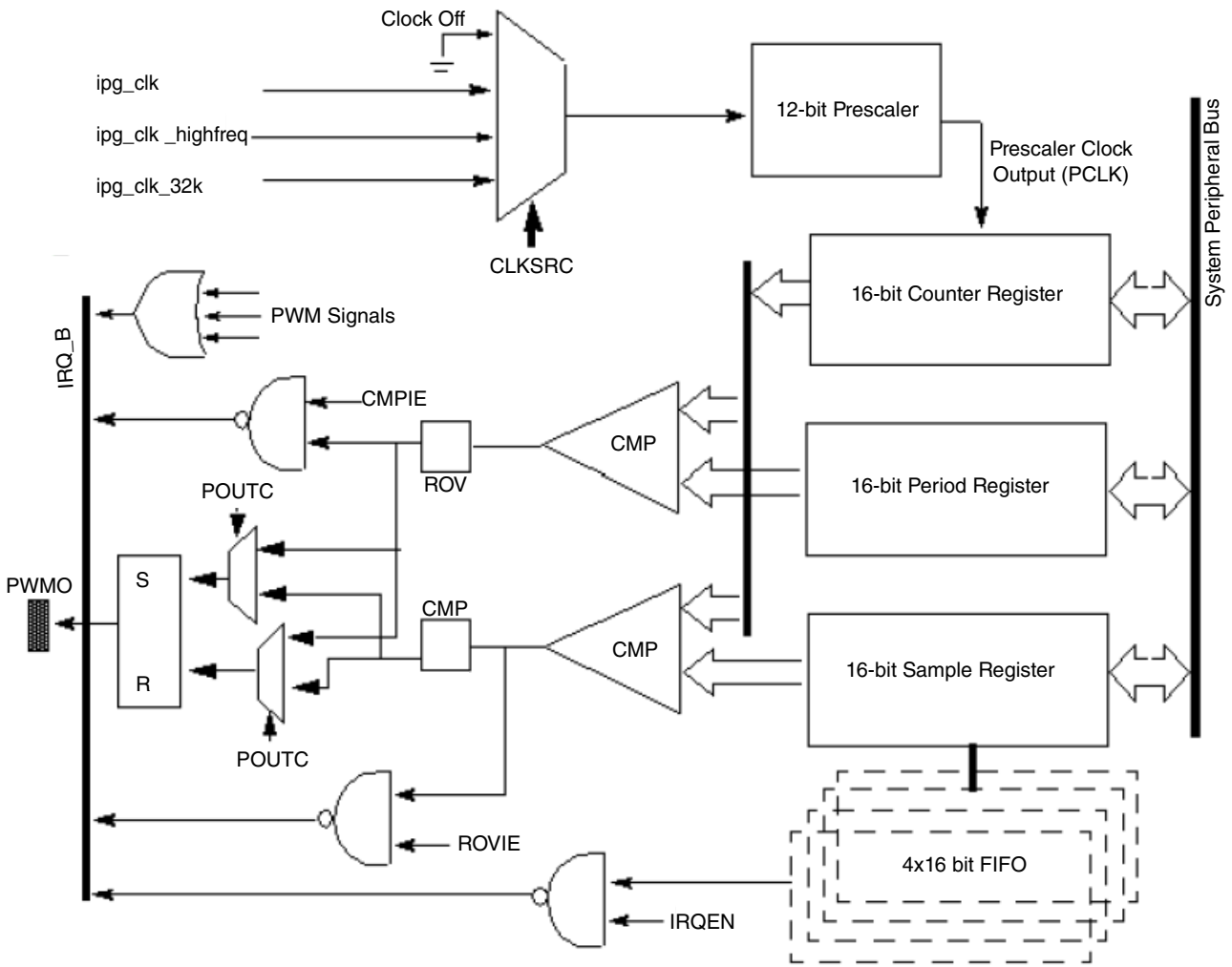


Figure 12-100. Pulse-Width Modulator Block Diagram

The following features characterize the PWM:

- 16-bit up-counter with clock source selection
- 4 x 16 FIFO to minimize interrupt overhead
- 12-bit prescaler for division of clock
- Sound and melody generation
- Active high or active low configured output
- Can be programmed to be active in low-power mode
- Can be programmed to be active in debug mode
- Interrupts at compare and rollover

12.3.2 External Signals

The PWM follows IP Bus protocol when interfacing with the processor core. PWM does not have any interface signals with any other block inside the chip except for clock and reset inputs from the Clock Control Module (CCM), System Reset Controller (SRC), and interrupt signals to the processor interrupt handler. There is a single output signal.

The following table outlines the external signals.

Table 12-19. PWM External Signals

Signal	Description	Pad	Mode	Direction
PWM1_OUT	This is the PWM1 functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	ENET1_RDATA0	ALT1	O
		GPIO1_IO01	ALT1	
		GPIO1_IO08	ALT7	
PWM2_OUT	This is the PWM2 functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	ENET1_RDATA1	ALT1	O
		GPIO1_IO02	ALT1	
		GPIO1_IO09	ALT7	
PWM3_OUT	This is the PWM3 functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	ENET1_TDATA0	ALT1	O
		GPIO1_IO03	ALT1	
		GPIO1_IO10	ALT7	
PWM4_OUT	This is the PWM4 functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	ENET1_TDATA1	ALT1	O
		GPIO1_IO00	ALT1	
		GPIO1_IO11	ALT7	

12.3.3 Clocks

The table found here describes the clock sources for PWM.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 12-20. PWM Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_32k	ckil_sync_clk_root	low-frequency reference clock (32 kHz)
ipg_clk_highfreq	perclk_clk_root	high-frequency reference clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

The clock that feeds the prescaler can be selected from:

- High-frequency reference clock (ipg_clk_highfreq) pat_ref or CKIH

  This is a high frequency clock, provided by the Clock Control Module (CCM). This clock should be on in the low power mode when the ipg_clk is turned off. Thus, the PWM can be run on this clock in the low power mode.

- Low-frequency reference clock (ipg_clk_32k, CKIL)

  This is the 32 KHz low reference clock which is provided by the CCM. This clock should be on in the low power mode when ipg_clk is turned off. Thus, PWM can be run on this clock in the low power mode.

- Peripheral clock (ipg_clk)

  This clock should be on in normal operations. In low power mode, it can be switched off.

- Peripheral access clock (ipg_clk_s)

  This clock is used for register read/write.

The clock input source is determined by the PWM control register field PWM_CR[CLKSRC]. The CLKSRC value should only be changed when the PWM is disabled.

A change in the value of the PRESCALER field of the control register is immediately reflected on its output clock frequency.

12.3.4 Functional Description

The following sections detail the PWM operation and function.

12.3.4.1 Operation

The output of the PWM is a toggling signal whose frequency and duty cycle can be modulated by programming the appropriate registers. It has a 16-bit up counter which counts from 0x0000 until the counter value equals the PWM_PR + 1. After this match occurs the counter is reset to 0x0000.

At the beginning of a count period cycle, the PWMO pin is set to one (default) and the counter begins counting up from 0x0000. The sample value in the sample FIFO is compared on each count of prescaler clock. When the sample and count values match, the PWMO signal is cleared to zero (default). The counter continues counting until the period match occurs and subsequently another period cycle begins.

When the PWM is enabled, the counter starts running and generates an output with the reset values in the period and sample registers. It is recommended that the programming of these registers be done before PWM is enabled.

A hardware reset results in all the PWM count and sample registers being cleared and the FIFO being flushed. The control register shows that FIFO is empty and it can be written into, and the PWM is disabled. A software reset has the same results, however the state of the STOPEN, DOZEN, WAITEN, and DBGEN bits in the control register are not affected. Software reset can be asserted even when the PWM is in disabled state.

12.3.4.1.1 FIFO

Digital sample values can be loaded into the pulse-width modulator as 16-bit words. The endianess can be changed using the BCTR and HCTR bits of the control register. A 4-word (16-bit) FIFO minimizes interrupt overhead. A maskable interrupt is generated when the number of data words fall below the water level set by the FWM field in the control register.

A write to the PWM_SAR sample register results in the value being stored into the FIFO if it is not full. A write when the FIFO is full sets FWE (FIFO write error) bit in the status register and the FIFO contents remain unchanged. The FIFO can be written at any time, but can be read only when the PWM is enabled. The PWM_SR[FIFOAV] field shows how many data words are currently contained in the FIFO and whether or not it can be written into.

A read on the sample register yields the current FIFO value that is being used, or will be used, by the PWM for generation on the output signal. Therefore, a write and a subsequent read on the sample register may result in different values being obtained.

12.3.4.1.2 Rollover and Compare Event

The counter is reset to 0x0000 after its value equals the PWM_PR[PERIOD] + 1 and resumes counting thereafter. This event is referred to as a rollover. For example, if PWM_PR[PERIOD] = 0x0000, the counter is reset when it equals 0x0001. When PWM_PR[PERIOD] = 0xFFFF or 0xFFFE, the counter is reset when it equals 0xFFFF. For more information, see the PWM Period Register (PWM_PR) description.

During a rollover event the output is either set (default), reset or has no effect according to the programming of the POUTC field in the control register. This event can also generate an interrupt if the respective interrupt enable bit is set in the control register.

When the counter value reaches the sample value, the output of the PWM is reset (default), set or has no effect according to the programming of the POUTC field of control register. This event is referred to as a compare event. This event can also generate an interrupt if the respective interrupt enable bit is set in the control register.

If the rollover event sets the PWM output signal, the compare event will reset it and vice versa for a particular programming configuration of POUTC field.

12.3.4.1.3 Low Power Mode Behavior

In low power mode, if the clock from the selected clock source is available, the PWM counter continues to run and an output is produced, depending on whether the control bit for that mode is set or not. In the absence of the clock itself, or if the corresponding low power bit in the control register is 0, the counter is reset and resumes counting when it exits the low power mode.

12.3.4.1.4 Debug Mode Behavior

In debug mode, PWM has the option of continuing to run or be halted. If the DBGEN bit is not set in the PWM_PWMCR, the PWM is halted. If the DBGEN bit is set, then the PWM will continue to run in the debug mode.

12.3.5 Enable Sequence for the PWM

The sequence found here should be used to enable the PWM.

1. Configure the desired settings for the PWM Control Register (PWMx_PWMCR) while keeping the PWM disabled (PWMx_PWMCR[0]=0).
2. Enable the desired interrupts in the PWM Interrupt Register (PWMx_PWMIR).
3. One to three initial samples may be written to the PWM Sample Register (PWMx_PWMSAR). The initial sample values will be loaded into the PWM FIFO even if the PWM is not yet enabled. Do not write a 4th sample because the FIFO will become full and trigger a FIFO Write Error (FWE). This error will prevent the PWM from starting once it is enabled.
4. Check the FIFO Write Error status bit (FWE), the Compare status bit (CMP) and the Roll-over status bit (ROV) in the PWM Status Register (PWMx_PWMSR) to make sure they are all zero. Any non-zero status bits should be cleared by writing a 1 to them.
5. Write the desired period to the PWM Period Register (PWMx_PWMPR).
6. Enable the PWM by writing a 1 to the PWM Enable bit, PWMx_PWMCR[0], while maintaining the other register bits in their previously configured state.

12.3.6 Disable Sequence for the PWM

The PWM can be disabled at any time by clearing the PWM enable bit, PWMx_PWMCR[0] to 0.

Any data remaining in the FIFO will not be produced at the PWM output after the PWM has been disabled and will remain in the FIFO until the PWM is enabled again. A software reset (setting PWMx_PWMCR[3] to 1) or a hardware reset will clear the FIFO and any remaining data will be lost.

12.3.7 PWM Memory Map/Register Definition

The PWM includes six user-accessible 32-bit registers.

PWM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3066_0000	PWM Control Register (PWM1_PWMCR)	32	R/W	0000_0000h	12.3.7.1/ 3530
3066_0004	PWM Status Register (PWM1_PWMSR)	32	w1c	0000_0008h	12.3.7.2/ 3532

Table continues on the next page...

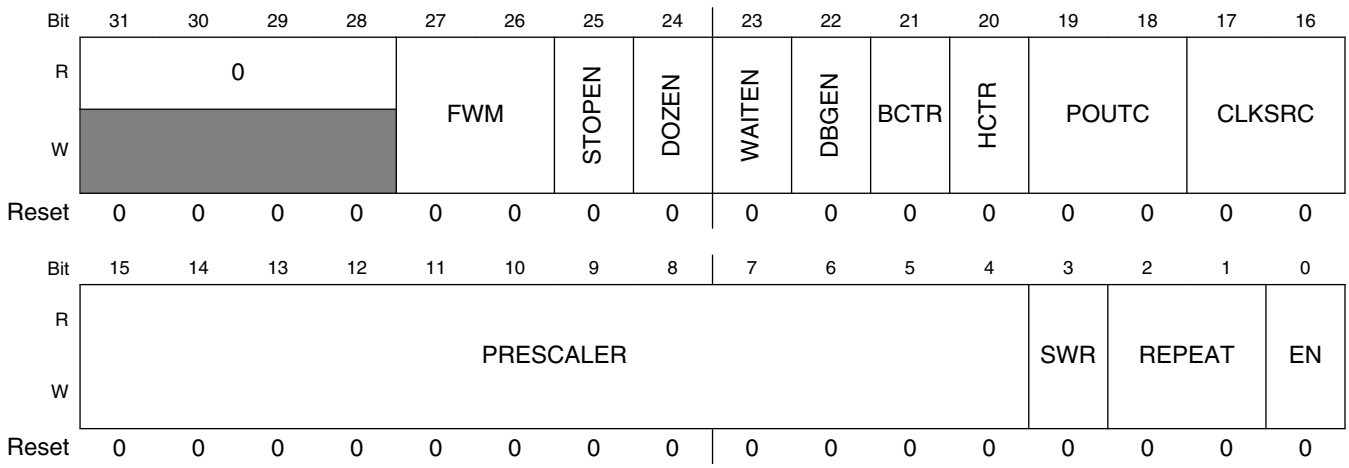
PWM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3066_0008	PWM Interrupt Register (PWM1_PWMIR)	32	R/W	0000_0000h	12.3.7.3/ 3533
3066_000C	PWM Sample Register (PWM1_PWMSAR)	32	R/W	0000_0000h	12.3.7.4/ 3534
3066_0010	PWM Period Register (PWM1_PWMPR)	32	R/W	0000_FFFEh	12.3.7.5/ 3535
3066_0014	PWM Counter Register (PWM1_PWMCNR)	32	R	0000_0000h	12.3.7.6/ 3535
3067_0000	PWM Control Register (PWM2_PWMCR)	32	R/W	0000_0000h	12.3.7.1/ 3530
3067_0004	PWM Status Register (PWM2_PWMSR)	32	w1c	0000_0008h	12.3.7.2/ 3532
3067_0008	PWM Interrupt Register (PWM2_PWMIR)	32	R/W	0000_0000h	12.3.7.3/ 3533
3067_000C	PWM Sample Register (PWM2_PWMSAR)	32	R/W	0000_0000h	12.3.7.4/ 3534
3067_0010	PWM Period Register (PWM2_PWMPR)	32	R/W	0000_FFFEh	12.3.7.5/ 3535
3067_0014	PWM Counter Register (PWM2_PWMCNR)	32	R	0000_0000h	12.3.7.6/ 3535
3068_0000	PWM Control Register (PWM3_PWMCR)	32	R/W	0000_0000h	12.3.7.1/ 3530
3068_0004	PWM Status Register (PWM3_PWMSR)	32	w1c	0000_0008h	12.3.7.2/ 3532
3068_0008	PWM Interrupt Register (PWM3_PWMIR)	32	R/W	0000_0000h	12.3.7.3/ 3533
3068_000C	PWM Sample Register (PWM3_PWMSAR)	32	R/W	0000_0000h	12.3.7.4/ 3534
3068_0010	PWM Period Register (PWM3_PWMPR)	32	R/W	0000_FFFEh	12.3.7.5/ 3535
3068_0014	PWM Counter Register (PWM3_PWMCNR)	32	R	0000_0000h	12.3.7.6/ 3535
3069_0000	PWM Control Register (PWM4_PWMCR)	32	R/W	0000_0000h	12.3.7.1/ 3530
3069_0004	PWM Status Register (PWM4_PWMSR)	32	w1c	0000_0008h	12.3.7.2/ 3532
3069_0008	PWM Interrupt Register (PWM4_PWMIR)	32	R/W	0000_0000h	12.3.7.3/ 3533
3069_000C	PWM Sample Register (PWM4_PWMSAR)	32	R/W	0000_0000h	12.3.7.4/ 3534
3069_0010	PWM Period Register (PWM4_PWMPR)	32	R/W	0000_FFFEh	12.3.7.5/ 3535
3069_0014	PWM Counter Register (PWM4_PWMCNR)	32	R	0000_0000h	12.3.7.6/ 3535

12.3.7.1 PWM Control Register (PWMx_PWMCR)

The PWM control register (PWM_PWMCR) is used to configure the operating settings of the PWM. It contains the prescaler for the clock division.

Address: Base address + 0h offset



PWMx_PWMCR field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–26 FWM	FIFO Water Mark. These bits are used to set the data level at which the FIFO empty flag will be set and the corresponding interrupt generated 00 FIFO empty flag is set when there are more than or equal to 1 empty slots in FIFO 01 FIFO empty flag is set when there are more than or equal to 2 empty slots in FIFO 10 FIFO empty flag is set when there are more than or equal to 3 empty slots in FIFO 11 FIFO empty flag is set when there are more than or equal to 4 empty slots in FIFO
25 STOPEN	Stop Mode Enable. This bit keeps the PWM functional while in stop mode. When this bit is cleared, the input clock is gated off in stop mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in stop mode 1 Active in stop mode
24 DOZEN	Doze Mode Enable. This bit keeps the PWM functional in doze mode. When this bit is cleared, the input clock is gated off in doze mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in doze mode 1 Active in doze mode
23 WAITEN	Wait Mode Enable. This bit keeps the PWM functional in wait mode. When this bit is cleared, the input clock is gated off in wait mode. This bit is not affected by software reset. It is cleared by hardware reset.

Table continues on the next page...

PWMx_PWMCR field descriptions (continued)

Field	Description
	0 Inactive in wait mode 1 Active in wait mode
22 DBGEN	Debug Mode Enable. This bit keeps the PWM functional in debug mode. When this bit is cleared, the input clock is gated off in debug mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in debug mode 1 Active in debug mode
21 BCTR	Byte Data Swap Control. This bit determines the byte ordering of the 16-bit data when it goes into the FIFO from the sample register. 0 byte ordering remains the same 1 byte ordering is reversed
20 HCTR	Half-word Data Swap Control. This bit determines which half word data from the 32-bit IP Bus interface is written into the lower 16 bits of the sample register. 0 Half word swapping does not take place 1 Half words from write data bus are swapped
19–18 POUTC	PWM Output Configuration. This bit field determines the mode of PWM output on the output pin. 00 Output pin is set at rollover and cleared at comparison 01 Output pin is cleared at rollover and set at comparison 10 PWM output is disconnected 11 PWM output is disconnected
17–16 CLKSRC	Select Clock Source. These bits determine which clock input will be selected for running the counter. After reset the system functional clock is selected. The input clock can also be turned off if these bits are set to 00. This field value should only be changed when the PWM is disabled 00 Clock is off 01 ipg_clk 10 ipg_clk_highfreq 11 ipg_clk_32k
15–4 PRESCALER	Counter Clock Prescaler Value. This bit field determines the value by which the clock will be divided before it goes to the counter. 0x000 Divide by 1 0x001 Divide by 2 0xfff Divide by 4096
3 SWR	Software Reset. PWM is reset when this bit is set to 1. It is a self clearing bit. A write 1 to this bit is a single wait state write cycle. When the block is in reset state this bit is set and is cleared when the reset procedure is over. Setting this bit resets all the registers to their reset values except for the STOPEN, DOZEN, WAITEN, and DBGEN bits in this control register. 0 PWM is out of reset 1 PWM is undergoing reset
2–1 REPEAT	Sample Repeat. This bit field determines the number of times each sample from the FIFO is to be used. 00 Use each sample once 01 Use each sample twice 10 Use each sample four times 11 Use each sample eight times

Table continues on the next page...

PWMx_PWMCR field descriptions (continued)

Field	Description
0 EN	PWM Enable. This bit enables the PWM. If this bit is not enabled, the clock prescaler and the counter is reset. When the PWM is enabled, it begins a new period, the output pin is set to start a new period while the prescaler and counter are released and counting begins. To make the PWM work with softreset and disable/enable, users can do software reset by seting the SWR bit, wait software reset done, configure the registers, and then enable the PWM by setting this bit to "1" Users can also disable/enable the PWM if PWM would like to be stopped and resumed with same registers configurations . 0 PWM disabled 1 PWM enabled

12.3.7.2 PWM Status Register (PWMx_PWMSR)

The PWM status register (PWM_PWMSR) contains seven bits which display the state of the FIFO and the occurrence of rollover and compare events. The FIFOAV bit is read-only but the other four bits can be cleared by writing 1 to them. The FE, ROV, and CMP bits are associated with FIFO-Empty, Roll-over, and Compare interrupts, respectively.

Address: Base address + 4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0									FWE	CMP	ROV	FE	FIFOAV		
W										w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

PWMx_PWMSR field descriptions

Field	Description
31–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 FWE	FIFO Write Error Status. This bit shows that an attempt has been made to write FIFO when it is full. 0 FIFO write error not occurred 1 FIFO write error occurred
5 CMP	Compare Status. This bit shows that a compare event has occurred. 0 Compare event not occurred 1 Compare event occurred

Table continues on the next page...

PWMx_PWMSR field descriptions (continued)

Field	Description
4 ROV	Roll-over Status. This bit shows that a roll-over event has occurred. 0 Roll-over event not occurred 1 Roll-over event occurred
3 FE	FIFO Empty Status Bit. This bit indicates the FIFO data level in comparison to the water level set by FWM field in the control register. 0 Data level is above water mark 1 When the data level falls below the mark set by FWM field
FIFOAV	FIFO Available. These read-only bits indicate the data level remaining in the FIFO. An attempted write to these bits will not affect their value and no transfer error is generated. 000 No data available 001 1 word of data in FIFO 010 2 words of data in FIFO 011 3 words of data in FIFO 100 4 words of data in FIFO 101 unused 110 unused 111 unused

12.3.7.3 PWM Interrupt Register (PWMx_PWMIR)

The PWM Interrupt register (PWM_PWMIR) contains three bits which control the generation of the compare, rollover and FIFO empty interrupts.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0													CIE	RIE	FIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMx_PWMIR field descriptions

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 CIE	Compare Interrupt Enable. This bit controls the generation of the Compare interrupt.

Table continues on the next page...

PWMx_PWMIR field descriptions (continued)

Field	Description
	0 Compare Interrupt not enabled 1 Compare Interrupt enabled
1 RIE	Roll-over Interrupt Enable. This bit controls the generation of the Rollover interrupt. 0 Roll-over interrupt not enabled 1 Roll-over Interrupt enabled
0 FIE	FIFO Empty Interrupt Enable. This bit controls the generation of the FIFO Empty interrupt. 0 FIFO Empty interrupt disabled 1 FIFO Empty interrupt enabled

12.3.7.4 PWM Sample Register (PWMx_PWMSAR)

The PWM sample register (PWM_PWMSAR) is the input to the FIFO. 16-bit words are loaded into the FIFO. The FIFO can be written at any time, but can be read only when the PWM is enabled. The PWM will run at the last set duty-cycle setting if all the values of the FIFO has been utilized, until the FIFO is reloaded or the PWM is disabled. When a new value is written, the duty cycle changes after the current period is over.

A value of zero in the sample register will result in the PWMO output signal always being low/high (POUTC = 00 it will be low and POUTC = 01 it will be high), and no output waveform will be produced. If the value in this register is higher than the PERIOD + 1, the output will never be set/reset depending on POUTC value.

Address: Base address + Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																SAMPLE															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMx_PWMSAR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SAMPLE	Sample Value. This is the input to the 4x16 FIFO. The value in this register denotes the value of the sample being currently used.

12.3.7.5 PWM Period Register (PWMx_PWMPR)

The PWM period register (PWM_PWMPR) determines the period of the PWM output signal. After the counter value matches PERIOD + 1, the counter is reset to start another period.

PWMO (Hz) = PCLK(Hz) / (period +2)

A value of zero in the PWM_PWMPR will result in a period of two clock cycles for the output signal. Writing 0xFFFF to this register will achieve the same result as writing 0xFFFE.

A change in the period value due to a write in PWM_PWMPR results in the counter being reset to zero and the start of a new count period.

NOTE

Settings PWM_PWMPR to 0xFFFF when PWMx_PWMCR REPEAT bits are set to non-zero values is not allowed.

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																PERIOD															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

PWMx_PWMPR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
PERIOD	Period Value. These bits determine the Period of the count cycle. The counter counts up to [Period Value] +1 and is then reset to 0x0000.

12.3.7.6 PWM Counter Register (PWMx_PWMCNR)

The read-only pulse-width modulator counter register (PWM_PWMCNR) contains the current count value and can be read at any time without disturbing the counter.

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMx_PWMCNR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Counter Value. These bits are the counter register value and denotes the current count state the counter register is in.

Chapter 13
Multimedia

13.1 Multimedia

13.1.1 Display and camera subsystem

The chip display and camera subsystem consists of the dedicated modules found here.

- LCD Interface (LCDIF): 24-bit parallel RGB LCD interface
- PXP pixel pipeline: pixel/image processing engine for LCD display
- CSI (camera sensor interface): up to 24-bit parallel interface for image sensor
- MIPI DSI Host controller with associated MIPI D-PHY Tx (one clock lane, two Data lanes)
- MIPI CSI-2 Host controller with associated MIPI D-PHY Rx (one clok lane, two Data lanes)

The following figure shows the high-level integration scheme of the chip display and camera system.

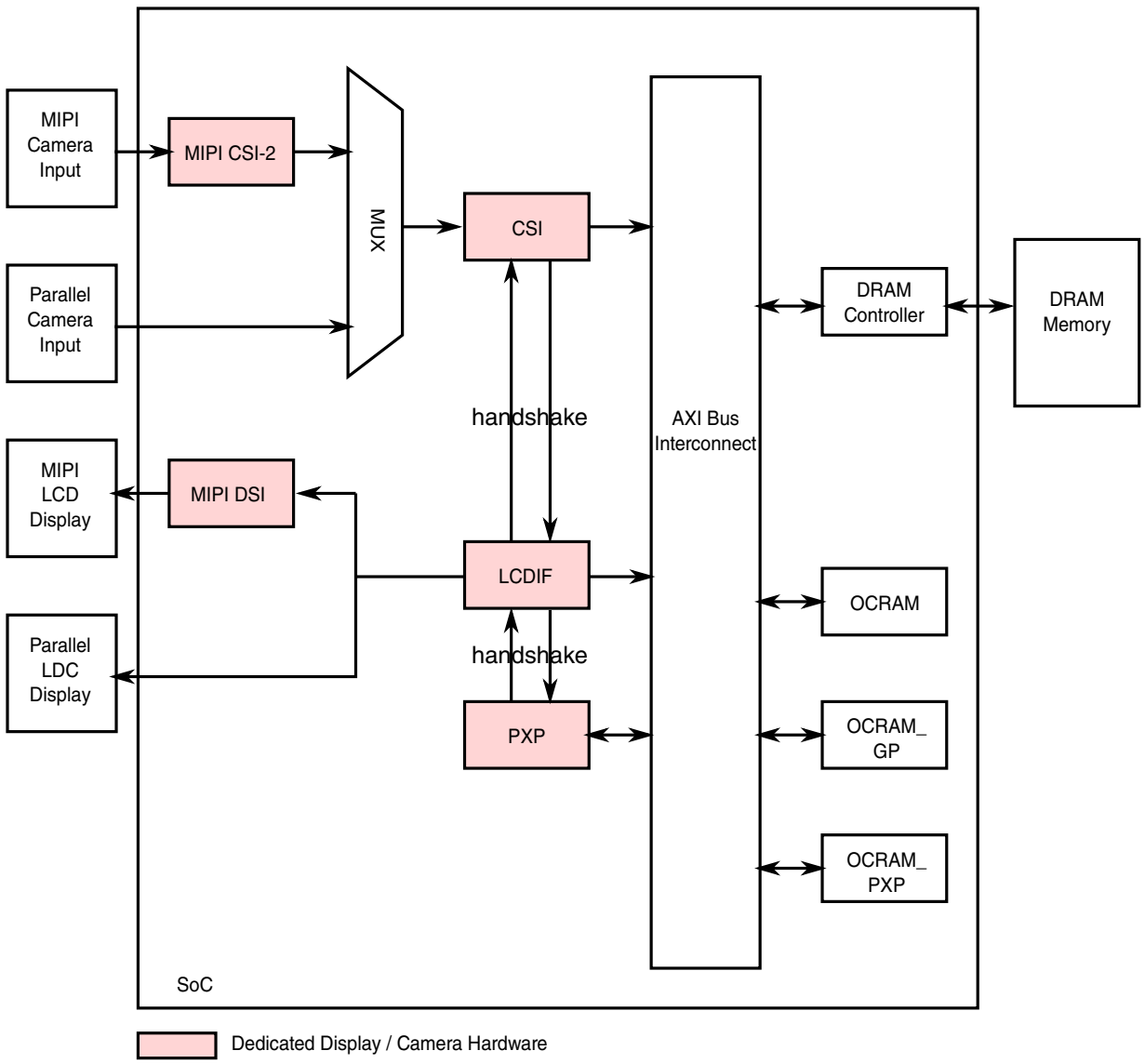


Figure 13-1. Chip display and camera subsystem

13.1.1.1 PiXel Processing Pipeline

The pixel processing pipeline is used to perform image processing on image/video buffers before sending to an LCD display.

The main features of PXP include:

- Multiple input/output format support, including YUV/RGB/Grayscale
- Supports both RGB/YUV scaling
- Supports overlay with Alpha blending
- Supports 2D Blit operation

- Supports HW dithering with configurable dithering algorithm
- Single-pass pixel processing for scaling, alpha blending, CSC, rotation and dithering without multiple access to DRAM
- Supports flexible programmable pixel processing engine

13.1.1.2 LCD Interface

The LCDIF is a general purpose display controller that is used to drive a wide range of display devices. These displays can vary in size and capability. Many of these displays have had an asynchronous parallel MPU interface for command and data transfer to an integrated frame buffer. There are other popular displays that support moving pictures and require the RGB interface mode (called DOTCLK interface in this document) or the VSYNC mode for high-speed data transfers. In addition to these displays, it is also common to provide support for digital video encoders that accept ITU-R BT.656 format 4:2:2 YCbCr digital component video and convert it to analog TV signals. The LCDIF block supports these different interfaces by providing fully programmable functionality.

The block has several major features:

- Bus master interface to source frame buffer data for display refresh and a DMA interface to manage input data transfers from the LCD requiring minimal CPU overhead.
- 8/16/18/24 bit LCD data bus support available depending on I/O mux options.
- Programmable timing and parameters for MPU, VSYNC and DOTCLK LCD interfaces to support a wide variety of displays.
- ITU-R BT.656 mode (called Digital Video Interface or DVI mode here) including progressive-to-interlace feature and RGB to YCbCr 4:2:2 color space conversion to support 525/60 and 625/50 operation.

13.1.1.3 Parallel CMOS Sensor Interface

The CSI enables the chip to connect directly to external CMOS image sensors. CMOS image sensors are separated into two classes, dumb and smart. Dumb sensors are those that support only traditional sensor timing (Vertical SYNC and Horizontal SYNC) and output only Bayer and statistics data, while smart sensors support CCIR656 video decoder formats and perform additional processing of the image (for example, image compression, image pre-filtering, and various data output formats).

The capabilities of the CSI include:

- Configurable interface logic to support most commonly available CMOS sensors.
- Support for CCIR656 video interface as well as traditional sensor interface.
- 8-bit data port for YCC, YUV, or RGB data input.
- 8-bit/10-bit/16-bit/24-bit data port for generic data input.
- Full control of 8-bit/pixel, 10-bit/pixel or 16-bit/pixel data format to 64-bit receive FIFO packing.
- 256 × 64 FIFO to store received image pixel data. Receive FIFO overrun protection mechanism.
- Embedded DMA controllers to transfer data from receive FIFO or statistic FIFO through AHB bus.
- Support for 2D DMA transfer from the receive FIFO to the frame buffers in the external memory.
- Support for double buffering two frames in the external memory.
- Single interrupt source to interrupt controller from maskable interrupt sources:
  - Start of Frame
  - End of Frame
  - Change of Field
  - FIFO full
  - FIFO overrun
  - DMA transfer done
  - CCIR error
  - AHB bus response error
- Configurable master clock frequency output to sensor.
- Statistic data generation for Auto Exposure (AE) and Auto White Balance (AWB) control of the camera (only for Bayer data and 8-bit/pixel format).

13.1.2 Display / Sensor MIPI interfaces

13.1.2.1 Introduction

i.MX Application Processor features MIPI DSI/CSI-2 interfaces that includes:

- MIPI DSI Host controller with associated MIPI D-PHY Tx (One clock lane, two Data lanes)
- MIPI CSI-2 Host controller with associated MIPI D-PHY Rx (one clock lane, Data Lanes)

13.1.2.2 MIPI DSI

MIPI DSI is a high performance serial interconnect bus for mobile applications connecting display system to the host system.

There is one instance of DSI port in the chip. This interface support from 80 Mbps up to 1.5 Gbps speed per data lane. The DSI Receiver core can mange one clock lane and up to 2 data lanes through the lane management, however two data lanes are implemented in the transmitter D-PHY, therefore the maximum throughput of display port is 3Gbps.

The DSI port supports the following standards:

- MIPI DSI Compliant
- DSI Version 1.01
- DPI Version 2.0
- DBI Version 2.0
- DCS Version 1.02
- PPI for D-PHY
- MIPI D-PHY Version 1.0

DSI host core is capable of supporting a variety of resolutions and formats:

- Resolution:
  - QQVGA
  - QCIF, QVGA
  - CIF
  - VGA
  - WVGA
  - SVGA
  - XVGA
  - WXGA
- Pixel format:
  - RGB565
  - LRGB565
  - RGB666
  - RGB888

The DSI can support both command and video modes and up to four virtual channels to accommodate multiple displays.

- Command and video mode support (type 1, 2, 3, and 4 display architecture)
- Mode switching: low power and ultra low power
- Burst mode: dual video channel
- Non-burst mode: single video channel

- Bus turnaround
- Fault error recovery scheme

DPI and DBI could coexists in the system but only one of them could be active in a certain time moment

13.1.2.3 MIPI CSI-2

MIPI CSI-2 is a high performance serial interconnect bus for mobile applications connecting camera sensors to the host system.

There is one instance of CSI-2 port in the chip. This interface support from 80 Mbps up to 1.5 Gbps speed per data lane. The CSI-2 Receiver core can mange one clock lane and up to two data lanes through the lane management and de-packetization, providing a maximum throughput of 3 Gbps transfer rate.

The CSI-2 port supports the following standards:
- MIPI CSI-2 Version 1.0
- MIPI PPI interface for D-PHY
- MIPI D-PHY Version 1.0

13.1.3 Audio subsystem

The audio subsystem consists of the following modules: SAI-1, SAI-2, SAI-3, and MQS. In addition, the IOMUX must be appropriately configured to get signals in and out of the chip.

Audio Subsystem Module Overview provides an overview of each of the audio subsystem component modules, followed by a module-specific section.

13.1.3.1 Audio Subsystem Module Overview

The following figure shows a high level block diagram of the audio subsystem.

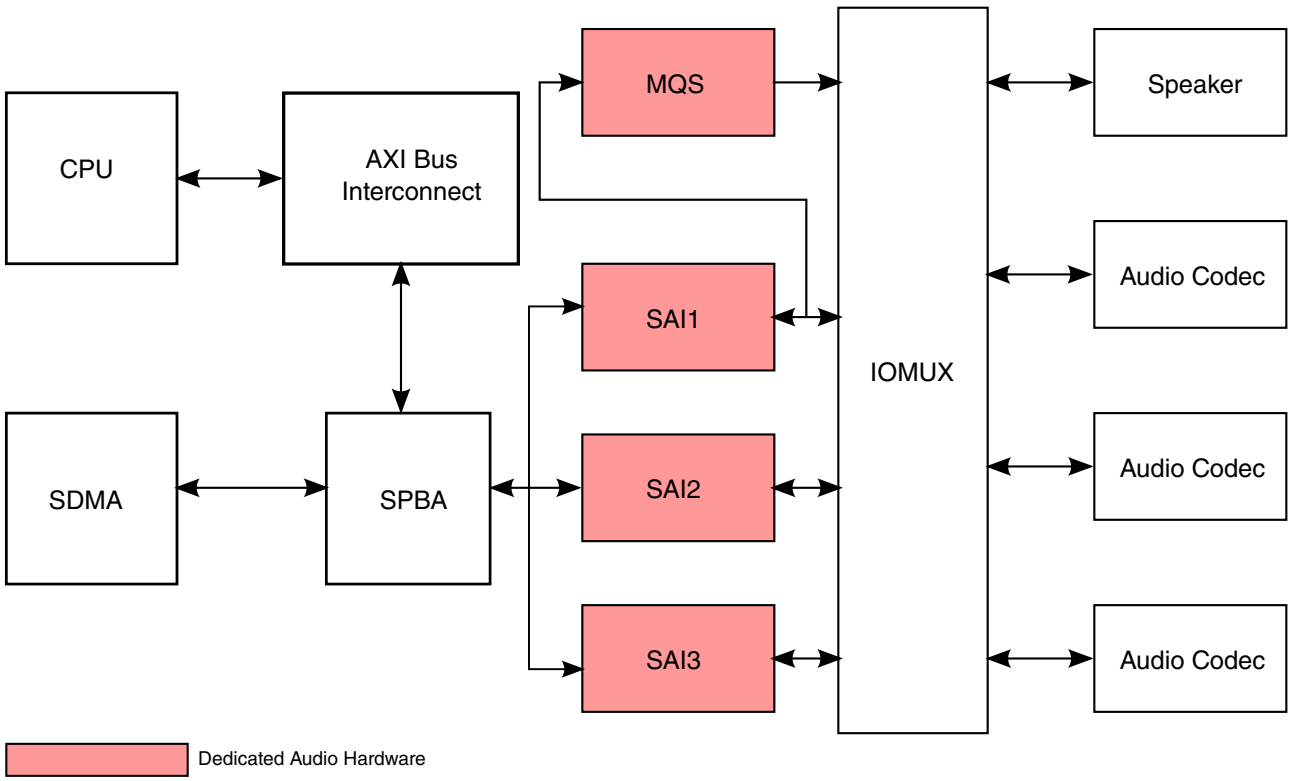


Figure 13-2. Audio subsystem block diagram

SAI1–3 are synchronous serial interfaces used to transfer audio data. SAI1–3 are on the shared peripheral bus. They can be accessed by both the SDMA and ARM CPUs. Their input/output are connected to the pads through IOMUX.

MQS (medium quality speaker) is used to convert the I2S audio data from SAI to PWM signals that can drive external speaker directly. Its audio source comes from SAI-1 and its output is connected to pads through IOMUX.

13.1.3.2 Synchronous Audio Interface (SAI)

- Transmitter with independent Bit Clock and Frame Sync supporting 1 data line
- Receiver with independent Bit Clock and Frame Sync supporting 1 data line
- Maximum Frame Size of 32 Words
- Word size programmable from 8-bits to 32-bits
- Word size configured separately for first word and remaining words in frame.
- Asynchronous FIFO for each Transmit and Receive data line
- Graceful restart after FIFO Error

13.2 Enhanced LCD Interface (eLCDIF)

13.2.1 Overview

The eLCDIF is a general purpose display controller used to drive a wide range of display devices varying in size and capability.

Many of these displays have had an asynchronous parallel MPU interface for command and data transfer to an integrated frame buffer. There are other popular displays that support moving pictures and require the RGB interface mode (called DOTCLK interface in this document) or the VSYNC mode for high-speed data transfers. In addition to these displays, it is also common to provide support for digital video encoders that accept ITU-R BT.656 format 4:2:2 YCbCr digital component video and convert it to analog TV signals. The eLCDIF block supports these interfaces by providing fully programmable functionality.

The block has several major features:

- Bus master interface to source frame buffer data for display refresh. This interface can also be used to drive data for "Smart" displays.
- PIO interface to manage data transfers between "Smart" displays and SoC.
- 8/16/18/24/32 bit LCD data bus support available depending on I/O mux options.
- Programmable timing and parameters for MPU, VSYNC and DOTCLK LCD interfaces to support a wide variety of displays.
- ITU-R BT.656 mode (called Digital Video Interface or DVI mode here) including progressive-to-interlace feature and RGB to YCbCr 4:2:2 color space conversion to support 525/60 and 625/50 operation.

13.2.2 External Signals

The following table describes the external signals of LCD:

Table 13-1. LCD External Signals

Signal	Description	Pad	Mode	Direction
LCD_BUSY	Busy Signal	EPDC1_DATA08	ALT7	I
		EPDC1_GDSP	ALT6	
LCD_CLK	Clock Signal	EPDC1_DATA00	ALT7	I
		EPDC1_SDCLK	ALT6	
		LCD1_CLK	ALT0	

Table continues on the next page...

Table 13-1. LCD External Signals (continued)

Signal	Description	Pad	Mode	Direction
LCD_CS	Chip Select	EPDC1_BDR0	ALT6	O
		EPDC1_DATA13	ALT7	
LCD_DATA00	Data Signal	EPDC1_DATA00	ALT6	IO
		EPDC1_DATA09	ALT7	
		LCD1_DATA00	ALT0	
LCD_DATA01	Data Signal	EPDC1_DATA01	ALT6	IO
		EPDC1_DATA11	ALT7	
		LCD1_DATA01	ALT0	
LCD_DATA02	Data Signal	EPDC1_DATA02	ALT6	IO
		EPDC1_SDCE3	ALT7	
		LCD1_DATA02	ALT0	
LCD_DATA03	Data Signal	EPDC1_DATA03	ALT6	IO
		EPDC1_SDCE2	ALT7	
		LCD1_DATA03	ALT0	
LCD_DATA04	Data Signal	EPDC1_DATA04	ALT6	IO
		EPDC1_SDCE1	ALT7	
		LCD1_DATA04	ALT0	
LCD_DATA05	Data Signal	EPDC1_DATA05	ALT6	IO
		EPDC1_SDCE0	ALT7	
		LCD1_DATA05	ALT0	
LCD_DATA06	Data Signal	EPDC1_BDR1	ALT7	IO
		EPDC1_DATA06	ALT6	
		LCD1_DATA06	ALT0	
LCD_DATA07	Data Signal	EPDC1_BDR0	ALT7	IO
		EPDC1_DATA07	ALT6	
		LCD1_DATA07	ALT0	
LCD_DATA08	Data Signal	EPDC1_DATA08	ALT6	IO
		EPDC1_SDLE	ALT7	
		LCD1_DATA08	ALT0	
LCD_DATA09	Data Signal	EPDC1_DATA09	ALT6	IO
		EPDC1_DATA10	ALT7	
		LCD1_DATA09	ALT0	
LCD_DATA10	Data Signal	EPDC1_DATA10	ALT6	IO
		EPDC1_SDSHR	ALT7	
		LCD1_DATA10	ALT0	
LCD_DATA11	Data Signal	EPDC1_DATA11	ALT6	IO
		EPDC1_PWRCOM	ALT7	
		LCD1_DATA11	ALT0	
LCD_DATA12	Data Signal	EPDC1_DATA12	ALT6	IO

Table continues on the next page...

Table 13-1. LCD External Signals (continued)

Signal	Description	Pad	Mode	Direction
		EPDC1_PWRSTAT	ALT7	
		LCD1_DATA12	ALT0	
LCD_DATA13	Data Signal	ECSPI2_SCLK	ALT4	IO
		EPDC1_DATA13	ALT6	
		LCD1_DATA13	ALT0	
LCD_DATA14	Data Signal	ECSPI2_MOSI	ALT4	IO
		EPDC1_DATA14	ALT6	
		LCD1_DATA14	ALT0	
LCD_DATA15	Data Signal	ECSPI2_MISO	ALT4	IO
		EPDC1_DATA15	ALT6	
		LCD1_DATA15	ALT0	
LCD_DATA16	Data Signal	EPDC1_GDCLK	ALT7	IO
		EPDC1_SDLE	ALT6	
		LCD1_DATA16	ALT0	
LCD_DATA17	Data Signal	EPDC1_GDSP	ALT7	IO
		EPDC1_SDOE	ALT6	
		LCD1_DATA17	ALT0	
LCD_DATA18	Data Signal	EPDC1_GDOE	ALT7	IO
		EPDC1_SDSHR	ALT6	
		LCD1_DATA18	ALT0	
LCD_DATA19	Data Signal	EPDC1_GDRL	ALT7	IO
		EPDC1_SDCE0	ALT6	
		LCD1_DATA19	ALT0	
LCD_DATA20	Data Signal	EPDC1_SDCE1	ALT6	IO
		EPDC1_SDCLK	ALT7	
		LCD1_DATA20	ALT0	
LCD_DATA21	Data Signal	EPDC1_DATA12	ALT7	IO
		EPDC1_SDCE2	ALT6	
		LCD1_DATA21	ALT0	
LCD_DATA22	Data Signal	EPDC1_DATA14	ALT7	IO
		EPDC1_SDCE3	ALT6	
		LCD1_DATA22	ALT0	
LCD_DATA23	Data Signal	EPDC1_GDCLK	ALT6	IO
		EPDC1_SDOE	ALT7	
		LCD1_DATA23	ALT0	
LCD_ENABLE	Enable Signal	EPDC1_BDR1	ALT6	IO
		EPDC1_DATA01	ALT7	
		LCD1_ENABLE	ALT0	
LCD_HSYNC	HSYNC enable	EPDC1_DATA03	ALT7	I

Table continues on the next page...

Table 13-1. LCD External Signals (continued)

Signal	Description	Pad	Mode	Direction
		EPDC1_PWRCOM	ALT6	
		LCD1_HSYNC	ALT0	
LCD_RD_E	RD_E Signal	EPDC1_GDRL	ALT6	IO
LCD_RESET	RS Signal	ECSPI2_SS0	ALT4	IO
		LCD1_RESET	ALT0	
LCD_VSYNC	VSYNC Signal	EPDC1_DATA02	ALT7	I
		EPDC1_PWRSTAT	ALT6	
		LCD1_VSYNC	ALT0	
LCD_WR_RWN	WR Signal	EPDC1_DATA15	ALT7	IO
		EPDC1_GDOE	ALT6	

13.2.3 Clocks

The following table describes the clock sources for eLCDIF. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 13-2. eLCDIF Clocks

Clock name	Clock Root	Description
apb_clk	MAIN_AXI_CLK_ROOT	AXI clock
pix_clk	LCDIF_PIXEL_CLK_ROOT	Pixel clock
ipg_clk_s	MAIN_AXI_CLK_ROOT	Peripheral access clock

13.2.4 Functional Description

Bus Interface Mechanisms through Initializing the eLCDIF, describe the internal pipeline for the MPU write/read interface, VSYNC, DOTCLK, and DVI interfaces. Differences for each mode are then described in separate sections, as follows:

- MPU Interface
- VSYNC Interface
- DOTCLK Interface
- ITU-R BT.656 Digital Video Interface (DVI)

eLCDIF pin usage by interface mode is described in eLCDIF Pin Usage by Interface Mode.

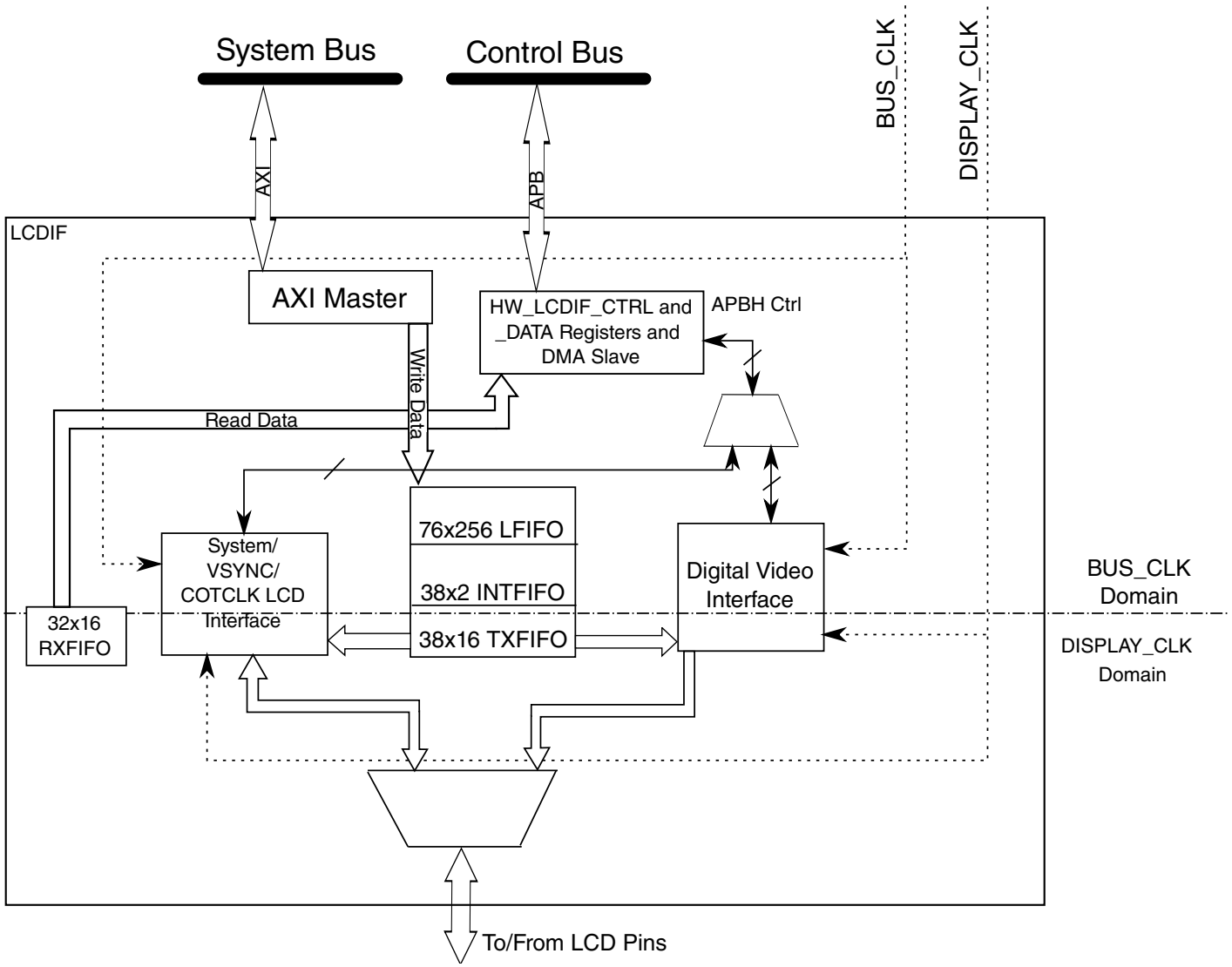


Figure 13-3. Top-Level Block Diagram of eLCDIF subsystem

13.2.4.1 Bus Interface Mechanisms

The LCDIF module has memory-mapped control, data and status registers. It provides several interfaces to transfer data between the display and SoC.

The bus master interface is used to initiate the requests to transfer data from external memory to the display. It is completely autonomous, or no CPU intervention is required, to manage the cyclical nature of refreshing standard display types. Bus mastering can also be used for MPU mode data writes.

The PIO interface is used to interface to "Smart" displays to transfer frame buffer data and control information to/from the external display. The host CPU executes display drivers to manage the display solution.

The following sections describe the system bus interface mechanisms.

13.2.4.1.1 Bus Master Operation in Write/Display Modes

The eLCDIF block has a bus master interface that initiates requests for data to drive the display. The LCDIF_MASTER bit must be set to 1 to enable the bus master interface. Software should program all control registers required to transfer the frame sequence.

In the MPU and VSYNC modes, single frames are transferred. When a complete frame is transferred, eLCDIF enters idle and clears the RUN bit in the CTRL register. For subsequent frame transmission, the eLCDIF setup sequence should be repeated.

The DOTCLK and DVI modes are used to refresh the display at the desired refresh rate and resolution. These modes are used to drive displays that do not integrate a display buffer memory. When the display is refreshed, the eLCDIF will automatically update the LCDIF_CUR_BUF_ADDR register with the value in LCDIF_NEXT_BUF_ADDR at the end of current frame and start fetching the next frame from the new address. If the LCDIF_NEXT_BUF_ADDR register was not updated within a frame refresh cycle, eLCDIF will keep transmitting the last frame until a new value is programmed into that register.

eLCDIF also provides the capability of interlacing a progressive frame by fetching odd lines in the first field and then fetching even lines in the second field. This feature can be used in the DVI mode and can be turned on by setting the INTERLACE_FIELDS bit in the LCDIF_CTRL1 register.

13.2.4.1.2 System Bus Master Performance

The performance of the eLCDIF block can be controlled by changing the burst length and the outstanding cycle issuing capability depending on the memory bandwidth requirements. Two fields in the LCDIF_CTRL2 register will throttle system memory requests. The LCDIF_CTRL2_OUTSTANDING_REQS field will control how many requests the eLCDIF can have in flight on any given clock cycle. This should be programmed based on the expected system bus latency for returned read data. Also, the LCDIF_CTRL2_BURST_LEN_8 bit will set the number of 64 bit words requested for each eLCDIF system bus request to either 8 or 16 QWORDS. Generally, 4 outstanding requests of length 16 will provide enough performance to drive any standard display resolution. These configuration bits are intended to change the access pattern of the eLCDIF to optimize system bus throughput when other system masters will contend for system memory resources.

The LCDIF_THRES register can also be used to optimize bus throughput and power consumption.

The LCDIF_THRES_PANIC value can be used to raise the priority of requests initiated by the eLCDIF to alter how the eLCDIF requests are arbitrated by the system bus infrastructure. The panic output control signal is raised when the number of 32bpp pixel equivalents in the LFIFO is less than this programmed value. Since the LFIFO is arranged as a 256x64bit quadword FIFO, it contains two 32bpp pixels per quadword, or 512 32bpp pixels total. To set the panic output when 3/4s of the LFIFO is empty, set the LCDIF_THRES_PANIC value to 3/4 * 512, or 128. The panic signal output is used to assess higher priority to eLCDIF system requests to avoid eLCDIF under run errors during periods of high system bandwidth utilization.

The features available with the LCDIF_THRES register require support from system clocking and dynamic priority control. Refer to the appropriate block documentation to assess the system support for these features.

13.2.4.2 Write Data Path

eLCDIF supports raster based frame buffers and there is no support for tiled buffers.

There are several options to accommodate endianness of display buffers in memory before the data is processed for the external display. The INPUT_DATA_SWIZZLE field in the LCDIF_CTRL register provides the following options for data word multiplexing:

```
00 (0): No swizzle (little-endian)
01 (1): Swap bytes 0 and 3, swap bytes 1 and 2 (big-endian)
10 (2): Swap half-words
11 (3): Swap bytes within each half-word
```

The WORD_LENGTH field of LCDIF_CTRL register indicates the input data/pixel format. LCDIF_TRANSFER_COUNT register denotes how much data is contained in each frame. The H_COUNT field of this register indicates the number of pixels per line and V_COUNT indicates the total number of lines per frame. A special bit field in the LCDIF_CTRL1 register, called the BYTE_PACKING_FORMAT, can be used to specify which bytes within the 32-bit word are going to be valid. For example, if the entire 32-bit word is valid, BYTE_PACKING_FORMAT should be set to 0xF, if only lower 3 bytes of each word in the frame buffer are valid, then BYTE_PACKING_FORMAT should be set to 0x7.

The LCD_DATABUS_WIDTH field in LCDIF_CTRL register suggests the width of the bus going to the external display controller. There is an option to source all 32 bits of the input word and transfer it to the output I/O display interface. Refer to the system I/O muxing options for support of this feature. If the LCD_DATABUS_WIDTH is not the same as WORD_LENGTH, eLCDIF will perform RGB to RGB color space conversion. For example, if the input frame has fewer bits per pixel than the display, as in a 16 bpp input frame going to 24 bpp LCD, eLCDIF will pad the MSBs of each color to the LSBs

of the same color for each pixel. If the input frame has more bits per pixel than the display, for example, 24 bpp input frame going to 16 bpp LCD, eLCDIF will drop the LSBs of each color channel to convert to the lower color depth. eLCDIF also has the capability to support delta pixel displays by swizzling the R, G and B colors of each pixel in the odd and even lines of the frame separately by programming the ODD_LINE_PATTERN and the EVEN_LINE_PATTERN bit fields. This operation occurs after the RGB-to-RGB color space conversion operation.

eLCDIF also supports RGB to YCbCr 4:2:2 color space conversion. This is useful in the DVI mode since the TV encoder requires input in YCbCr 4:2:2 format. The LCDIF_CSC* registers have complete programmability over the CSC coefficients and offsets. The values must be written into these registers in the signed two's complement format.

The following list shows how the different input/output combinations can be obtained:

- WORD_LENGTH=1 indicates that the input is 8-bit data. This is most likely going to be used for sending commands in MPU interface, or maybe a gray scale image. Any combination of BYTE_PACKING_FORMAT [3:0] is permissible.

  Limitation: H_COUNT must be a multiple of the sum of BYTE_PACKING_FORMAT [3], BYTE_PACKING_FORMAT [2], BYTE_PACKING_FORMAT [1] and BYTE_PACKING_FORMAT [0]. LCD_DATABUS_WIDTH must be 1, indicating an 8-bit data bus.

- WORD_LENGTH=0 implies the input frame buffer is RGB 16 bits per pixel. DATA_FORMAT_16_BIT field determines the pixels are RGB 555 or RGB 565.

  Limitation: BYTE_PACKING_FORMAT [3:0] should be 0x3 or 0xC if there is only one pixel per word. If there are two pixels per word, it should be 0xF and H_COUNT will be restricted to be a multiple of 2 pixels.

- WORD_LENGTH=2 indicates that input frame buffer is RGB 18 bits per pixel, that is, RGB 666. The valid RGB values can be left-aligned or right-aligned within a 32-bit word. The alignment of the valid 18 bits within a word is indicated by the DATA_FORMAT_18_BIT bit.

  Limitation: BYTE_PACKING_FORMAT can be 0x7, 0xE or 0xF. Packed pixels are not supported in this case. H_COUNT can be any number.

- WORD_LENGTH=3 indicates that the input frame-buffer is RGB 24 bits per pixel (RGB 888). If BYTE_PACKING_FORMAT [3:0] is 0x7, it indicates that there is only one pixel per 32-bit word and there is no restriction on H_COUNT. This is also the option that provides 32 bit output depending on the I/O muxing options available. The fourth byte, or bits [31:24], and connected to the I/Os if this muxing is available in the chip package.

Limitation: If BYTE_PACKING_FORMAT [3:0] is 0xF, it indicates that the pixels are packed, that is, there are 4 pixels in 3 words or 12 bytes and H_COUNT must be a multiple of 4 pixels.

- YCBCR422_INPUT=1 implies that the input frame is in YCbCr 4:2:2 format. BYTE_PACKING_FORMAT must be 0xF.

  Limitation: LCD_DATABUS_WIDTH must be 8-bit and H_COUNT must be a multiple of 2 pixels.

  ODD_LINE_PATTERN and EVEN_LINE_PATTERN must be 0 when any of RGB_TO_YCBCR422_CSC or INTERLACE_FIELDS or YCBCR422_INPUT bits is 1.

After the RGB to RGB or RGB to YCbCr 4:2:2 color space conversions, there is one more opportunity to swizzle the data before sending it out to the display or the encoder. This can be done with the CSC_DATA_SWIZZLE field in the LCDIF_CTRL register, and it provides the same options as the INPUT_DATA_SWIZZLE register.

Finally, there is an option to shift the output data before sending it out to the display. This is done based on the SHIFT_DIR and SHIFT_NUM_BITS fields in LCDIF_CTRL register.

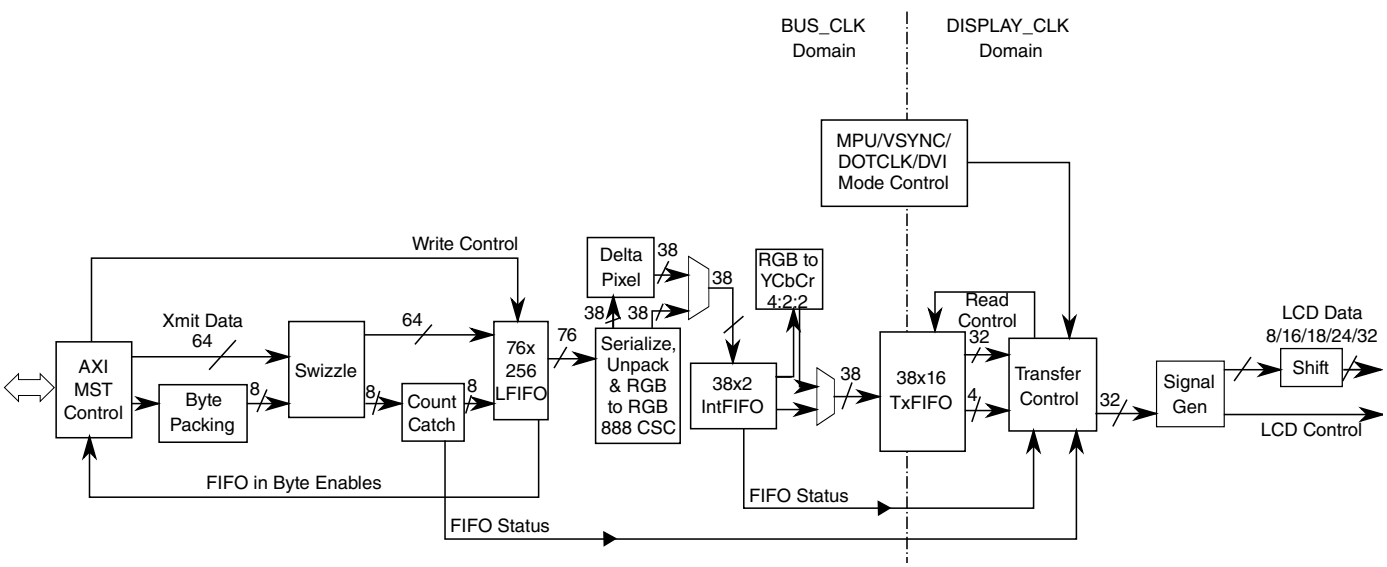


Figure 13-4. General Operations in Write Data Path

The examples in the following figures illustrate some different combinations of register programming for write mode. Assume that the data transferred over the system bus within a 32 bit word is organized as {A7-A0, B7-B0, C7-C0, D7-D0} in 8-bit mode and {A15-A0, B15-B0} in 16-bit mode.

In this example, all 32 bits of the input word are transferred out over an 8 bit display bus. Each byte within the 32 bit word is shifted to the right with zeros appended to I/O bits D[7:6]. The input data bits [7:2] are shifted to the right by 2 bits and presented on the D[5:0].

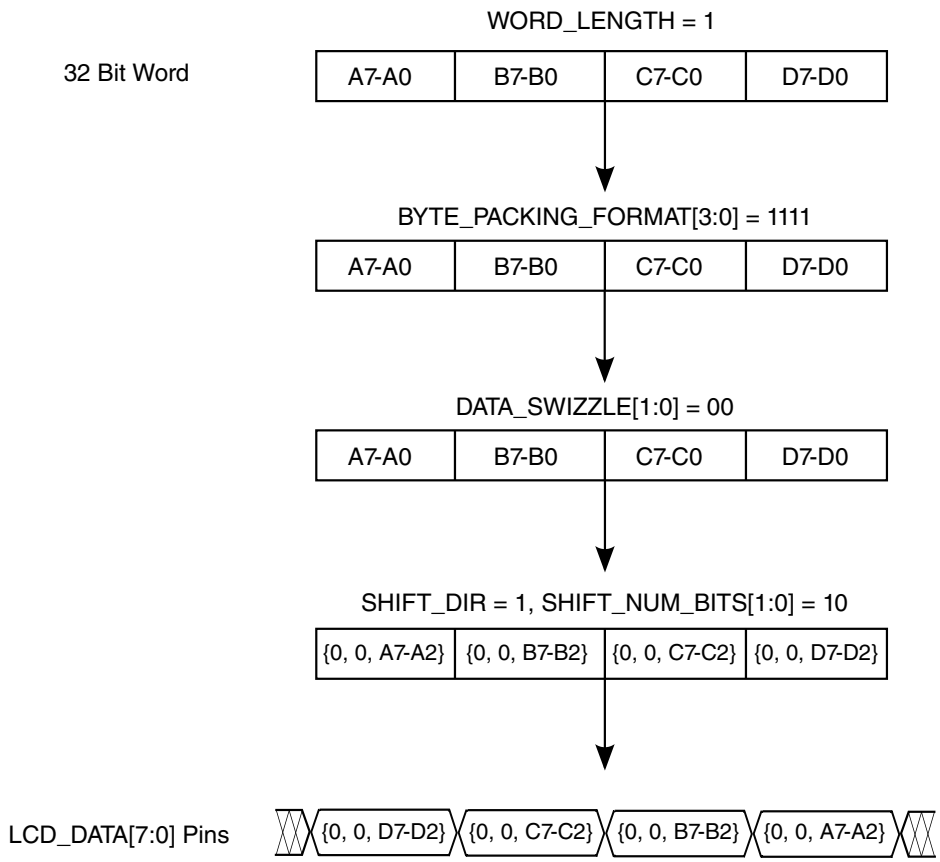


Figure 13-5. Register programming for write mode

In this 8 bit display interface example, one byte of the input word is deleted and not transferred over the external 8 bit display interface. This mode could be used to transfer 24bpp pixels over the 8 bit interface. In this case, the 4th unused byte is not transferred.

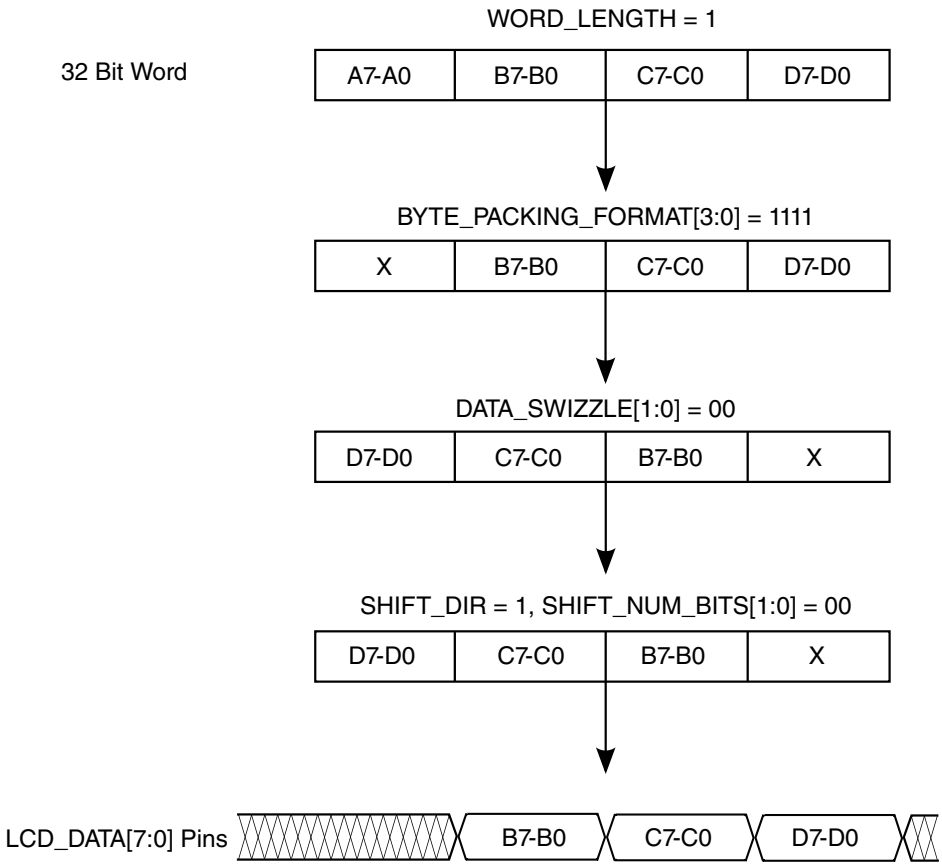


Figure 13-6. Register programming for write mode

The following example uses a 16 bit display interface. Each 16 bit half word is shifted to the right by two bits with zeros appended to the most significant two bits.

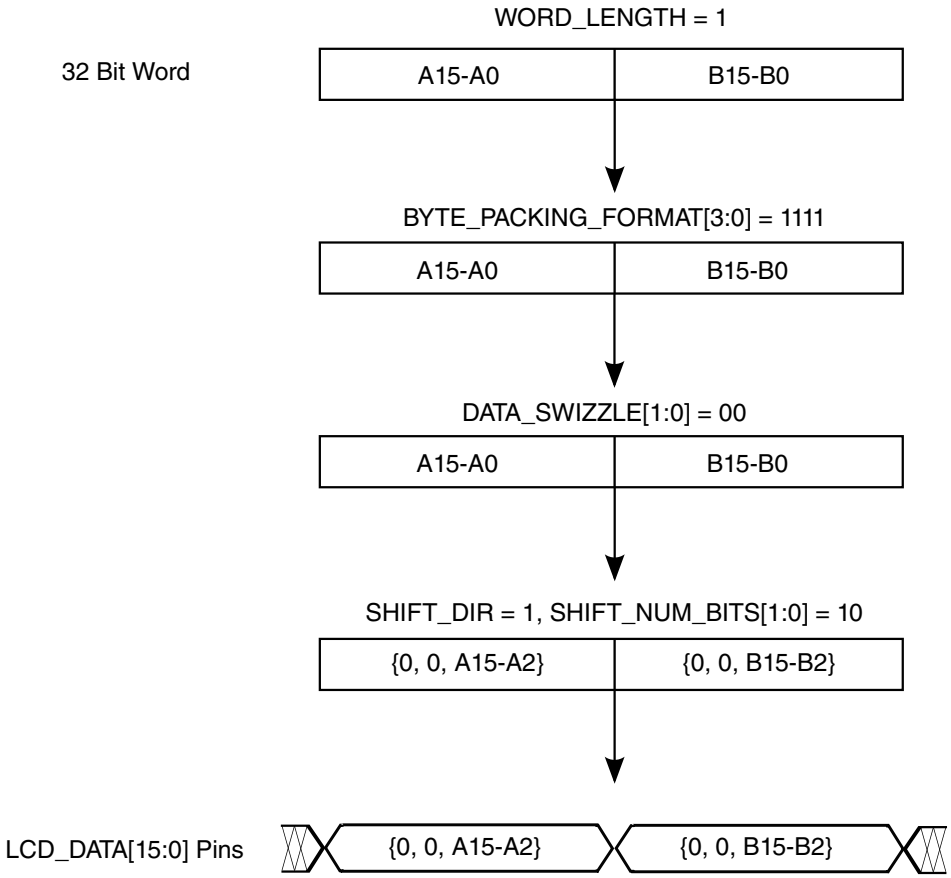


Figure 13-7. Register programming for write mode

This example indicates how an unpacked frame buffer can sourced for display. Only a single 16 bit half word within the 32 bit word is tranferred out via the 16 display bus.

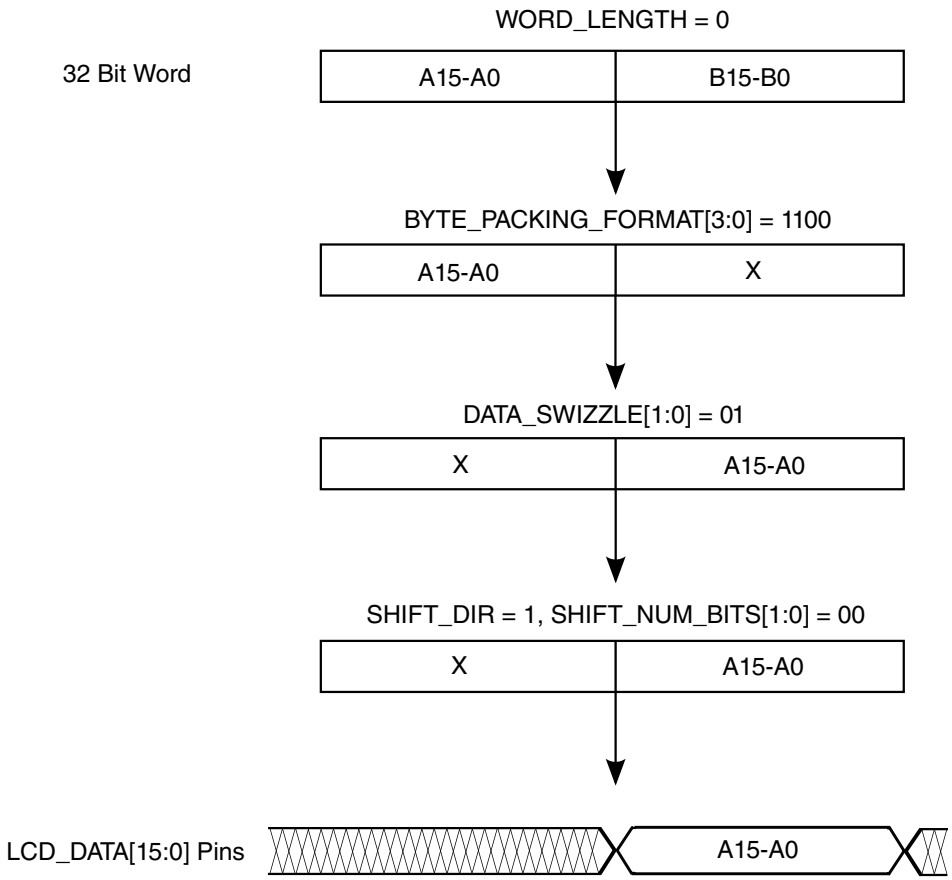


Figure 13-8. Register programming for write mode

13.2.4.3 Read Data Path

Figure 13-9 shows the MPU read data path in detail.

eLCDIF can read from an external display that follows the 6800/8080 MPU protocol.

The display bus width is determined by the LCD_DATABUS_WIDTH bit field. The data sampled at every read strobe is called a subword and the number of subwords that can be packed in a 32-bit word is given by the READ_MODE_NUM_PACKED_SUBWORDS bit field. The INITIAL_DUMMY_READ bit field directs the eLCDIF to skip the number of programmed subwords before starting to process read data. This feature is useful in the case of an LCD controller that returns the last written data the first time a read is issued, and then sends the correct data thereafter. SHIFT_DIR and SHIFT_NUM_BITS bit fields indicate whether the data needs to be shifted before getting stored in the internal registers. For example, a value of 2 in READ_MODE_NUM_PACKED_SUBWORDS if lcd databus width is 8 bits indicates two bytes should be packed in a 32-bit word, while if the lcd databus width is 16 bits, it indicates that two half words (or 4 bytes) should be packed.

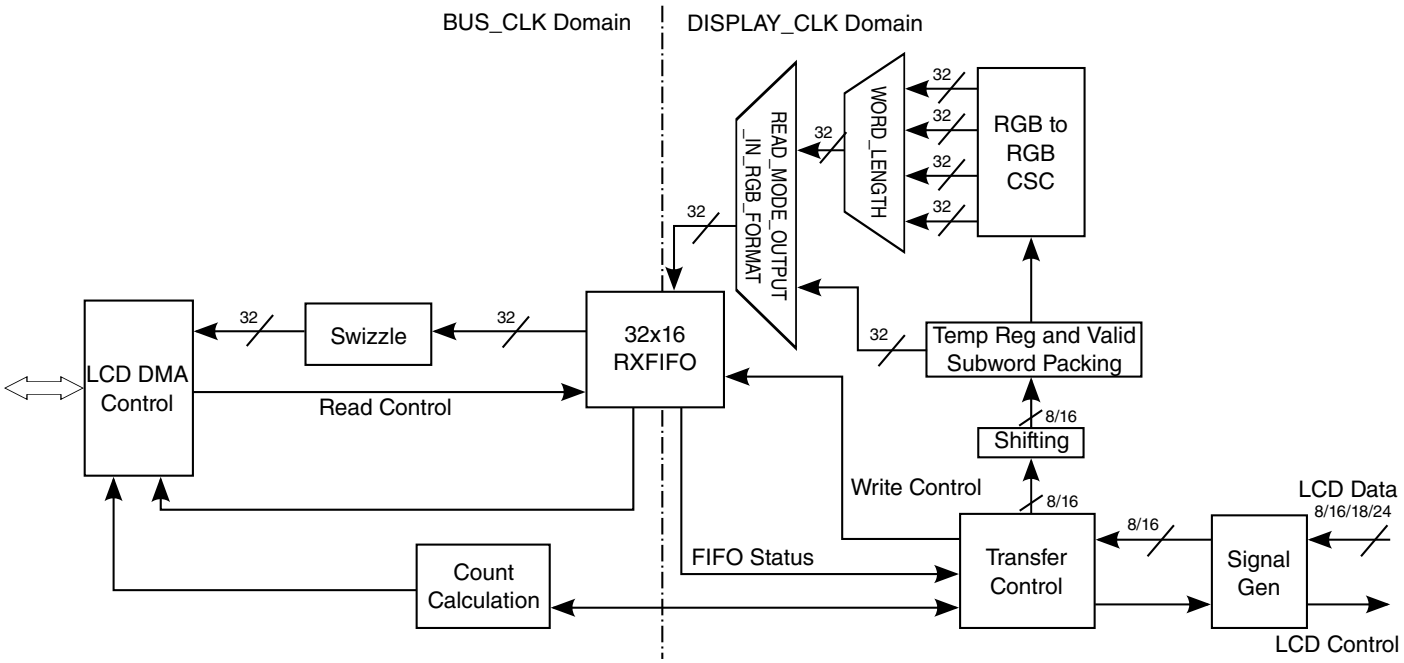


Figure 13-9. MPU Read Data Path

After the last subword within a word is reached, the block looks at the READ_PACK_DIR in the HW_LCDIF_CTRL2 register. If this bit is set, the block will swizzle the data, but only within the valid bytes, unlike in the write mode, where swizzle occurs across all 4 bytes. If the READ_MODE_OUTPUT_IN_RGB_FORMAT bit is set, eLCDIF will convert the data obtained from the READ_PACK_DIR operation into 24-bit unpacked RGB and then re-convert it into 16/18/24 bpp RGB depending on the WORD_LENGTH field. The DATA_FORMAT_16/18/24_BIT bit fields are also considered while converting to 24-bit unpacked RGB format. For example, if DATA_FORMAT_18_BIT is 1, the RGB666 data will be packed in the upper bits [31:4]of a 32-bit word, and that bit is 0, the data will be packed in the lower bits [17:0]. After all these operations, the data gets written into the RXFIFO.

The following figures show some examples of how data is handled in different MPU read modes.

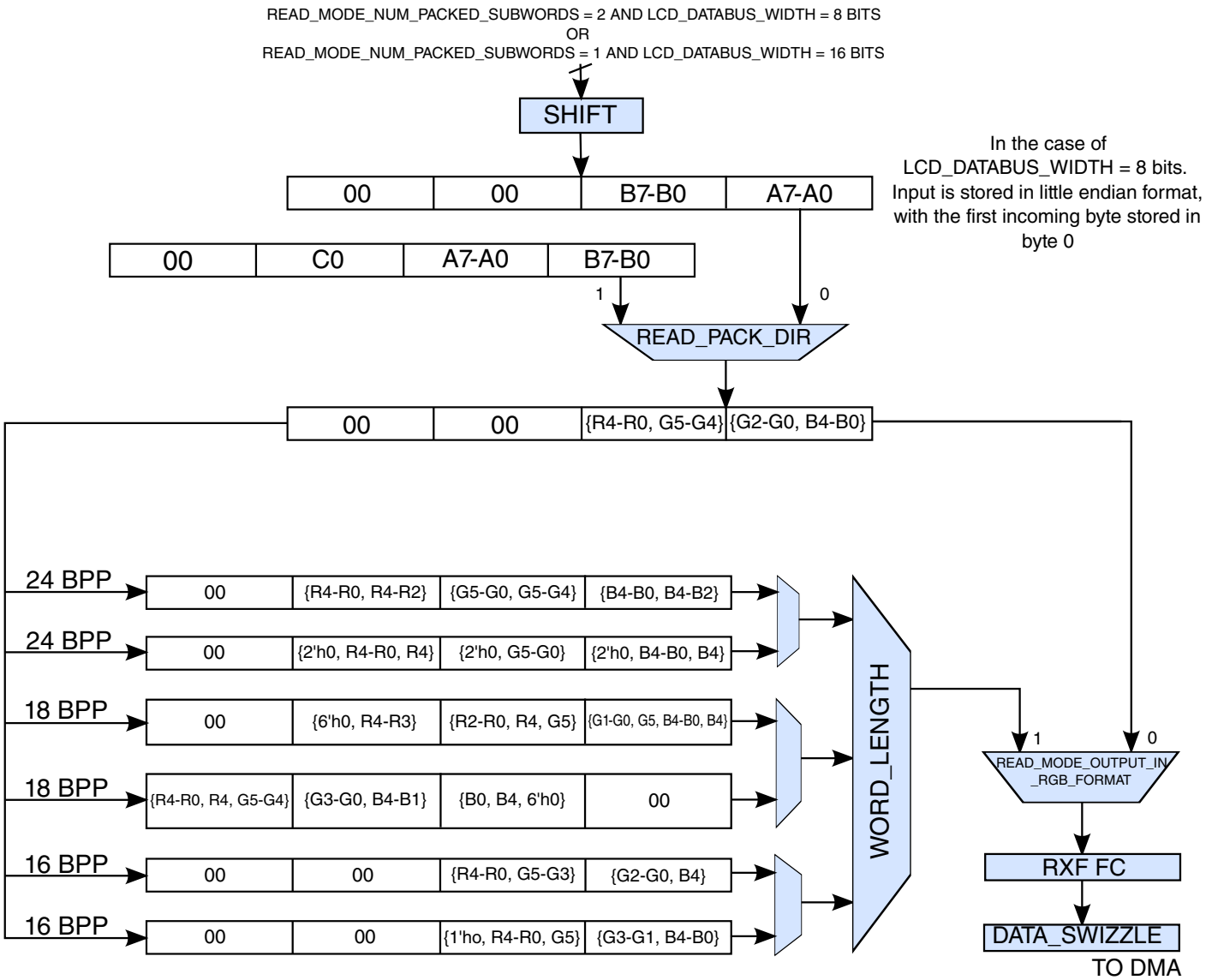


Figure 13-10. Data in MPU read mode

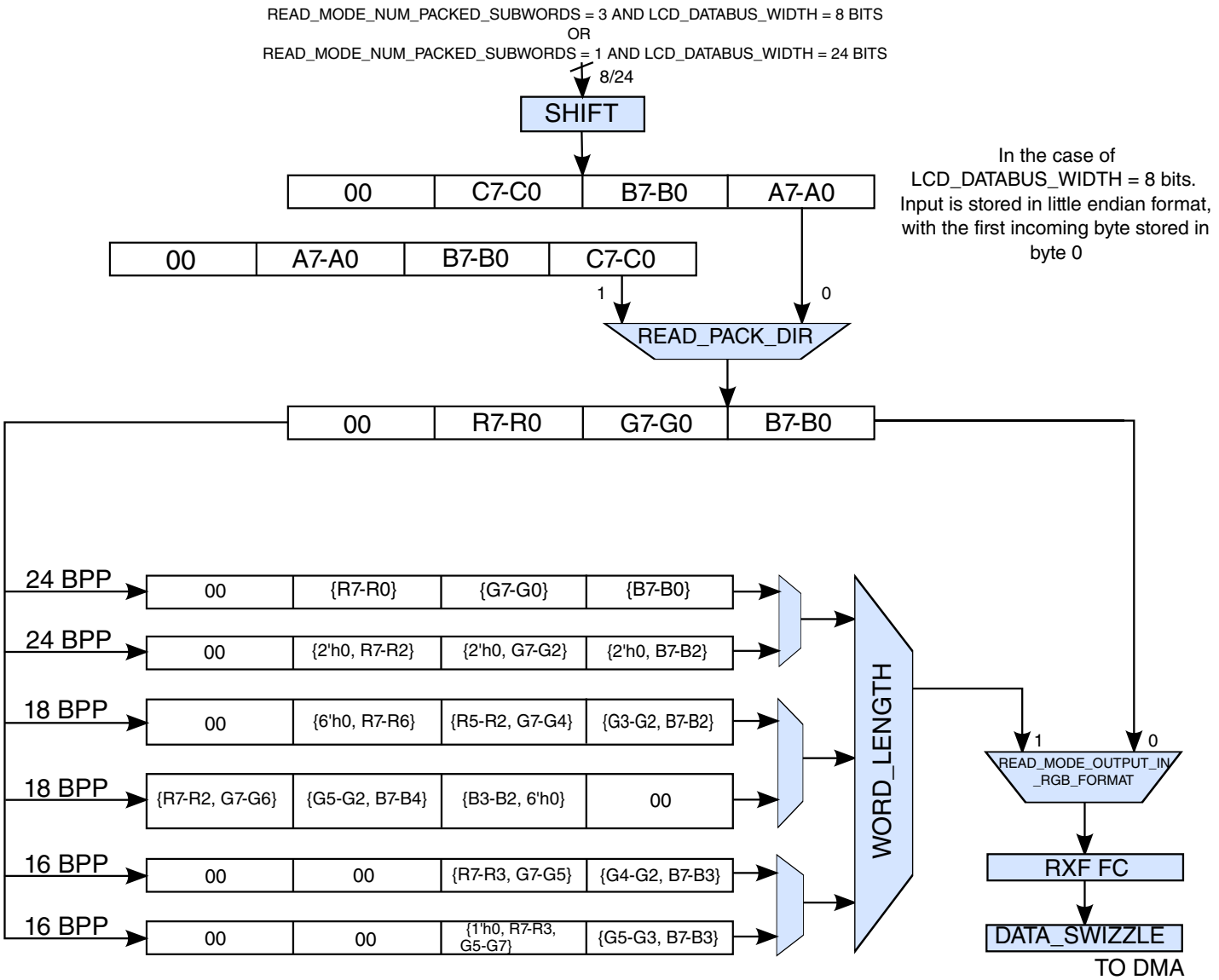


Figure 13-11. Data in MPU read mode

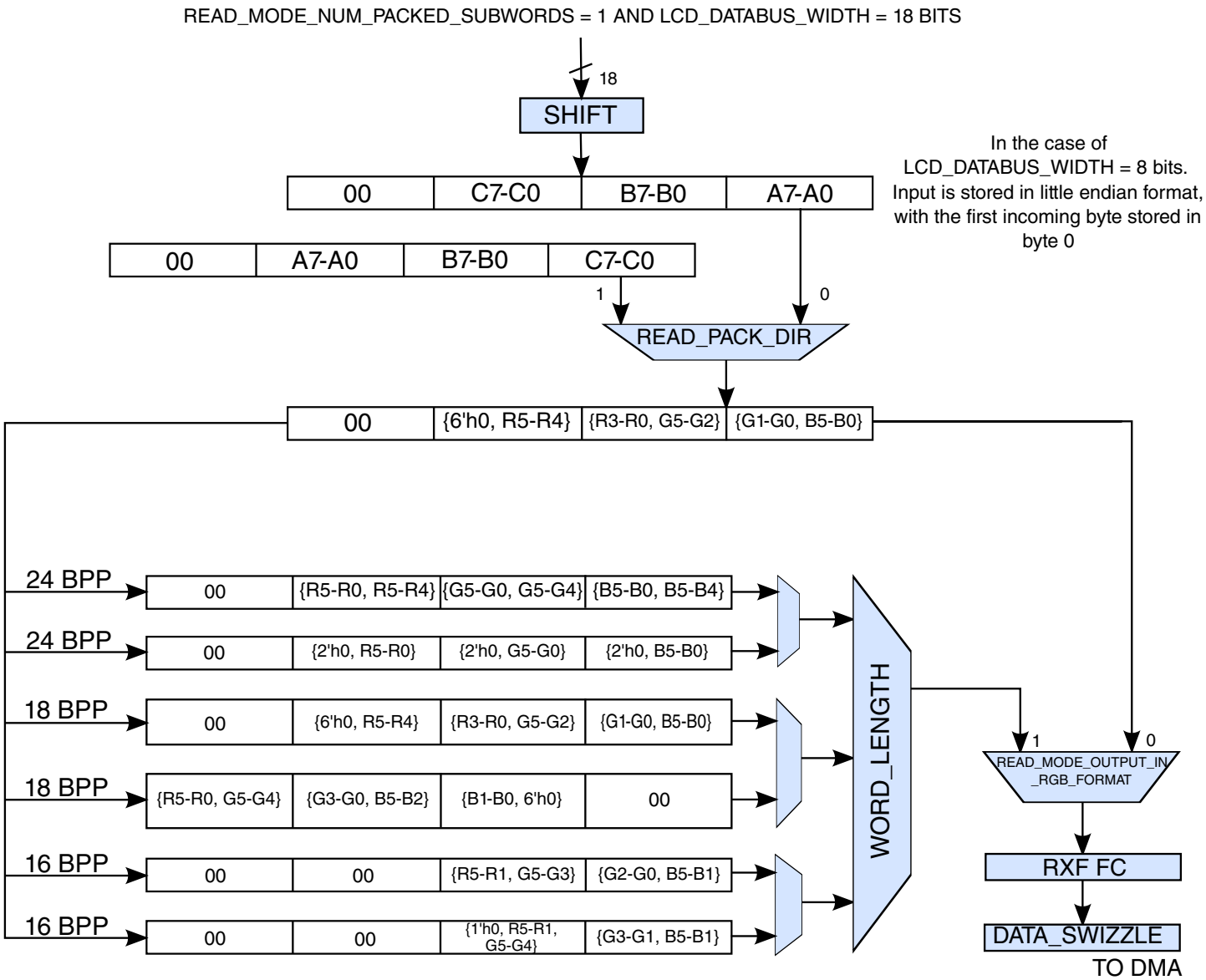


Figure 13-12. Data in MPU read mode

Restrictions:

READ_PACK_DIR should only be used if it is required to swizzle the subwords before doing RGB to RGB CSC, otherwise the DATA_SWIZZLE field should be used to swizzle across bytes.

READ_PACK_DIR must be 0 if LCD_DATABUS_WIDTH is 8 bits and READ_MODE_NUM_PACKED_SUBWORDS =1

If READ_MODE_OUTPUT_IN_RGB_FORMAT bit is set, the following restrictions should be followed:

- If LCD_DATABUS_WIDTH = 8 bits, then READ_MODE_NUM_PACKED_SUBWORDS <= 3.
- If LCD_DATABUS_WIDTH = 16/18/24 bits, then READ_MODE_NUM_PACKED_SUBWORDS = 1.

13.2.4.4 eLCDIF Interrupts

eLCDIF supports a number of interrupts to aid controlling and status reporting of the block.

All the interrupts have individual mask bits for enabling or disabling each of them. They all get funneled through a single interrupt line connected to the interrupt collector (ICOLL).

The following list describes the different interrupts supported by eLCDIF:

- Underflow interrupt is asserted when the clock domain crossing FIFO (TXFIFO) becomes empty but the block is in active display portion during that time. Software should take corrective action to make sure that this does not happen.
- In the bus master mode, the overflow interrupt will be asserted if the block has requested more data than it's FIFOs could hold. In the read mode, it will be asserted if the RxFIFO becomes full and the block reads more data.
- VSYNC edge interrupt will be asserted every time a leading VSYNC edge occurs.
- Cur_frame_done interrupt occurs at the end of every frame in all modes except DVI. In DVI mode, if IRQ_ON_ALTERNATE_FIELDS bit is set, it will occur at the end of every frame, otherwise it will occur at the end of every field.

13.2.4.5 Initializing the eLCDIF

This section describes write modes and MPU read mode.

13.2.4.5.1 Write Modes

The following initialization steps are common to all eLCDIF write modes of operation before entering any particular mode.

Initialization steps:

1. Configure the external I/Os to correctly interface the external display.
2. Start the DISPLAY_CLK clock and set the appropriate frequency by programming the registers in CCM.

3. Start the BUS_CLK and set the appropriate frequency by programming the registers in CCM.
4. Bring the eLCDIF out of soft reset and disable the clock gate bit.
5. Reset the LCD controller by setting LCDIF_CTRL1_RESET bit appropriately, being careful to observe the reset requirements of the controller. See Behavior During Reset for more information on Reset requirements.
6. Make sure READ_WRITEB bit in HW_LCDIF_CTRL register is 0.
7. Select the transfer mode of operation. The LCDIF_MASTER bit in HW_LCDIF_CTRL register determines the transfer mode selected. Bus master (LCDIF_MASTER =1) or PIO (LCDIF_MASTER =0) mode are the transfer modes to select.
8. Set the INPUT_DATA_SWIZZLE according to the endianness of the LCD controller. Also, set the DATA_SHIFT_DIR and SHIFT_NUM_BITS if it is required to shift the data left or right before it is output.
9. Set the WORD_LENGTH field appropriately: 0 = 16-bit input, 1 = 8-bit input, 2 = 18-bit input, 3 = 24/32-bit input. Also, select the correct 16/18/24 bit data format with the corresponding fields in HW_LCDIF_CTRL register.
10. Set the BYTE_PACKING_FORMAT field in HW_LCDIF_CTRL1 according to the input frame.
11. Set the LCD_DATABUS_WIDTH appropriately: 0 = 16-bit output, 1 = 8-bit output, 2 = 18-bit output, 3 = 24/32-bit output.
12. Enable the necessary IRQs.

13.2.4.5.2 MPU Read Mode

The following initialization steps should be done to enter the MPU read mode of operation:

Initialization steps:

1. Configure the external I/Os to correctly interface the external display.
2. Start the DISPLAY_CLK and set the appropriate frequency by programming the registers in CCM.
3. Start the BUS_CLK and set the appropriate frequency by programming the registers in CCM.
4. Bring the eLCDIF out of soft reset and clock gate.
5. Reset the LCD controller by setting LCDIF_CTRL1_RESET bit appropriately, being careful to observe the reset requirements of the controller.
6. Set the READ_WRITEB bit in LCDIF_CTRL register to 1.
7. Set the LCDIF_MASTER bit in LCDIF_CTRL register to 0. Bus master mode is not supported for reading data from the display.

8. Also, set the DATA_SHIFT_DIR and SHIFT_NUM_BITS if it is required to shift the data left or right before it is output.
9. Indicate if the read data needs to color-space-converted and stored in a different RGB format by setting the READ_MODE_OUTPUT_IN_RGB_FORMAT field accordingly.
10. Set the WORD_LENGTH field appropriately: 0 = 16-bit input, 1 = 8-bit input, 2 = 18-bit input, 3 = 24-bit input if READ_MODE_OUTPUT_IN_RGB_FORMAT is required. Also, select the correct 16/18/24 bit data format with the corresponding fields in LCDIF_CTRL register.
11. Set the READ_MODE_NUM_PACKED_SUBWORDS field in LCDIF_CTRL2 according to the number of subwords per word required to be packed.
12. Set the READ_PACK_DIR to 1 if it is required to store the data in big-endian format.
13. Set the LCD_DATABUS_WIDTH appropriately: 0 = 16-bit output, 1 = 8-bit output, 2 = 18-bit output, 3 = 24-bit output.
14. Enable the necessary IRQs.

13.2.4.6 MPU Interface

The MPU interface is used to transfer data and commands between the SoC via the eLCDIF and the external display at modest data rates.

Bus master or PIO transactions using the LCDIF_DATA register can be used for MPU mode write operations. For MPU mode read operations, only PIO can be used. eLCDIF can support the 6800 as well as the 8080 MPU protocol. If DOTCLK_MODE, DVI_MODE and VSYNC_MODE bits in LCDIF_CTRL registers are 0, it implies that the block is in MPU interface mode of operation. The LCDIF MPU mode has four basic timing parameters: Setup and Hold for the Command/Data register selection (TCS, TCH) and Setup and Hold for the Data bus (TDS, TDH). These parameters are expressed in DISPLAY_CLK cycles. The LCD_WR signal is used as the write strobe while LCD_RS signal is typically used to switch between command and data modes.

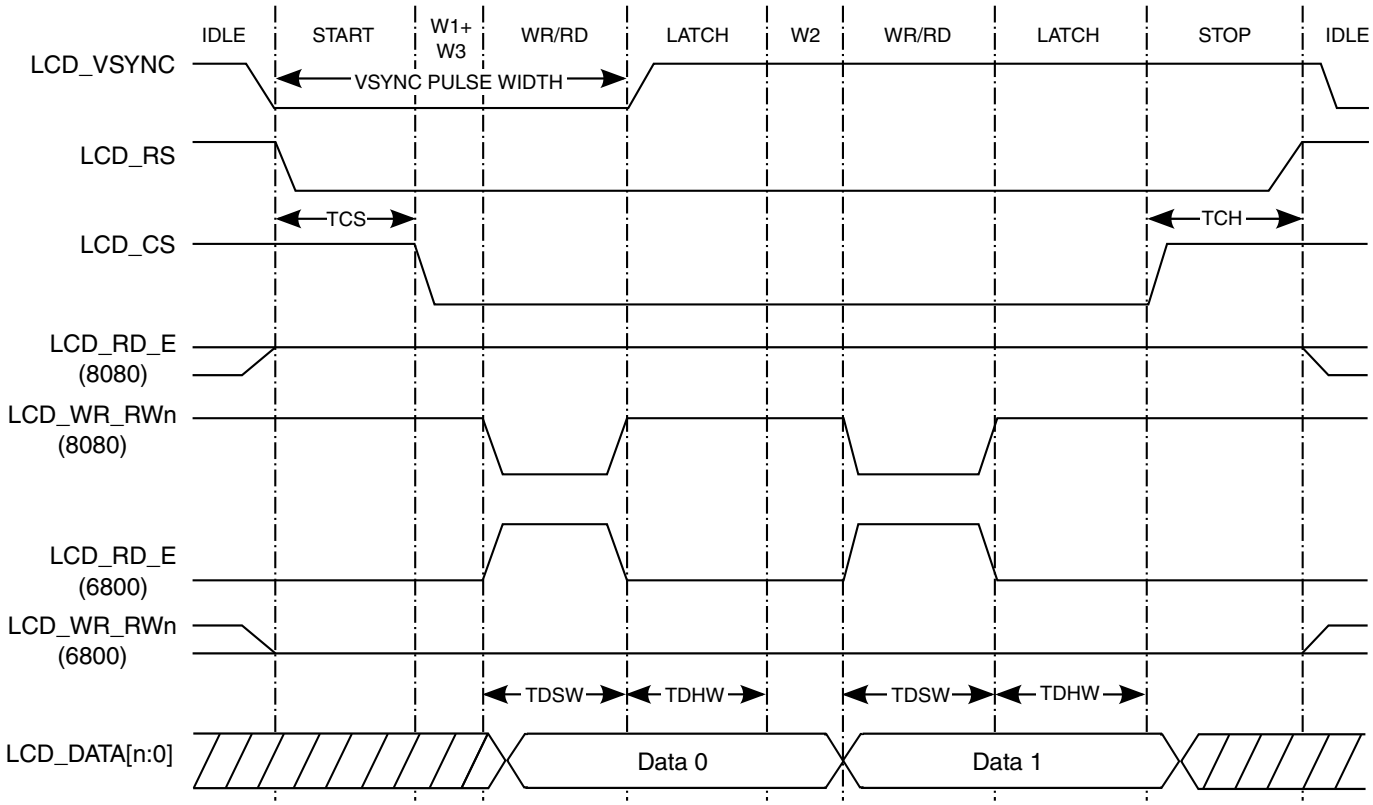


Figure 13-13. Timing in write mode of 6800 and 8080 protocols

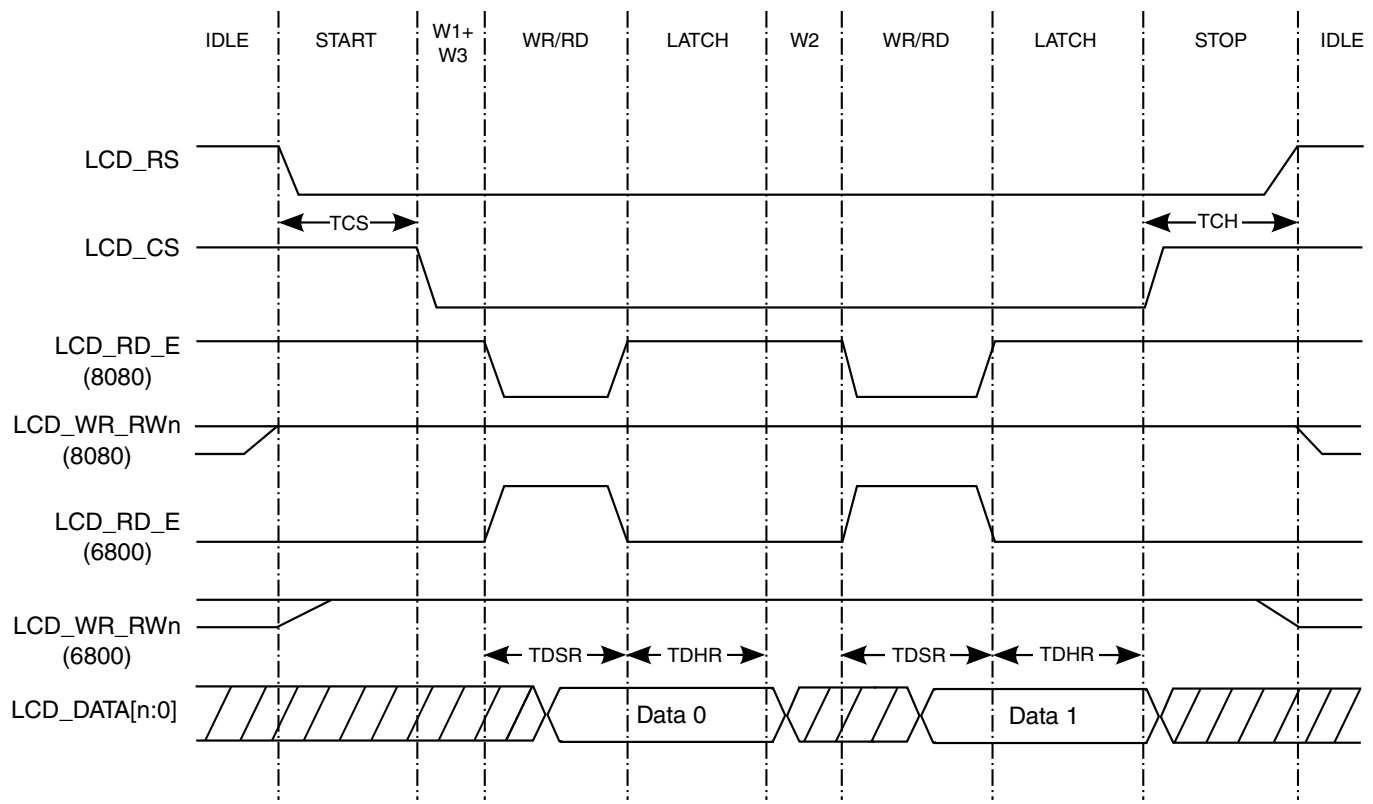


Figure 13-14. Read timing interface in 6800 and 8080 protocols

The eLCDIF has flexible pin and strobe timings which enable it to optimally support a wide range of LCDs. The minimum cycle time is two DISPLAY_CLK cycles (TDS=TDH=1). For example, this results in a maximum LCD data rate of 12 MB/s when DISPLAY_CLK is 24 MHz. TDS and TDH are 8-bit values, so the minimum eLCDIF period is 510 DISPLAY_CLK cycles (47 KHz with a 24 MHz DISPLAY_CLK). The timings are not automatically adjusted if the DISPLAY_CLK frequency changes, so it may be necessary to adjust the timings if DISPLAY_CLK changes.

In the MPU interface mode, the LCDIF_CTRL_ BYPASS_COUNT bit must be 0. The RUN bit is cleared automatically once the eLCDIF has received/transmitted all the data as per the LCDIF_TRANSFER_COUNT register and has completed the transfer to the panel. The current transfer can be cancelled/aborted if the RUN bit is manually made 0.

13.2.4.6.1 Code Example to Initialize the eLCDIF in MPU Write Mode

```
// Note: Common initialization steps in Initializing the eLCDIF must also be
// executed along with the following code
BF_CS1(LCDIF_CTRL, DATA_SELECT, 1); // 0 if sending command, 1 if sending data. Note that the
                                    // idle state for LCD_RS signal is high, regardless of the
                                     // programming of the DATA_SELECT register.
BF_CS1 (LCDIF_CTRL, MODE86, 8080_MODE);
BF_CS1 (LCDIF_CTRL, READ_WRITEB, 0);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 0); //Must be 0 in MPU mode
BF_CS1 (LCDIF_CTRL1, BUSY_ENABLE, 1);//Only if LCD controller implements a busy line
BF_CS4 (LCDIF_TIMING, CMD_HOLD, 2, CMD_SETUP, 2, DATA_HOLD, 2, DATA_SETUP, 2);  //Values
based
                              // on DISPLAY_CLK frequency and timing requirements of
controller.
                             // Note that these register must be non-zero for correct
operation.
BF_CS2 (LCDIF_TRANSFER_COUNT, H_COUNT, 320, V_COUNT, 240); //For a 320 RGB x 240 display
BF_CS1 (LCDIF_CTRL, RUN, 1);
```

The eLCDIF is now ready to receive data via bus master PIO write transactions using the LCDIF_DATA register. Note that when using the PIO write operations to the LCDIF_DATA register, the software will need to poll the FIFO STATUS bits to ensure that it does not overflow the eLCDIF data buffers. When eLCDIF is done transmitting H_COUNT x V_COUNT pixels, it will stop, turn off the RUN bit and assert the cur_frame_done interrupt.

13.2.4.7 VSYNC Interface

The VSYNC interface uses the same protocol as the MPU interface, with an additional signal VSYNC at the frame rate of the display, as shown in the figure given in MPU Interface section.

It is used in the moving picture display mode where data has to be written to the internal LCD buffer at a speed higher than the display rate and displayed in synchronization with the VSYNC signal. This mode is selected by setting the VSYNC_MODE bit in

LCDIF_CTRL register. The VSYNC signal is programmable for period, polarity and direction. Many other programmable parameters are shared with the MPU interface. The VSYNC_OEB bit in LCDIF_VDCTRL0 register indicates whether the display controller will send the VSYNC signal, or whether it should be generated by eLCDIF. The timing of the VSYNC signal is based on the DISPLAY_CLK (make sure VSYNC_PULSE_WIDTH_UNIT = VSYNC_PERIOD_UNIT = 0 and VSYNC_ONLY = 1) and it is determined by the VSYNC_PERIOD, VSYNC_PULSE_WIDTH and VSYNC_POL fields in LCDIF_VDCTRL0-4 registers. The SYNC_SIGNALS_ON bit in LCDIF_VDCTRL4 register must be set if the target requires the VSYNC signal to be generated by eLCDIF. If the WAIT_FOR_VSYNC_EDGE bit in LCDIF_CTRL register is set, it indicates that the hardware should wait until it sees the leading VSYNC edge before starting the data transfer. The VERITCAL_WAIT_CNT indicates the number of DISPLAY_CLK cycles from the leading VSYNC edge after which data transfer will be started on the interface.

In the VSYNC interface mode, the LCDIF_CTRL_ BYPASS_COUNT bit must be 0. The RUN bit is cleared automatically once the eLCDIF has received/transmitted all the data as per the LCDIF_TRANSFER_COUNT register and has completed the transfer to the panel. The current transfer can be cancelled/aborted if the RUN bit is manually made 0.

13.2.4.7.1 Code Example to Initialize eLCDIF in VSYNC Mode

```
// Note: Common initialization steps in Initializing the eLCDIF must also be
// executed along with the following code
BF_CS1 (LCDIF_CTRL, DATA_SELECT, 1); // 0 if sending command, 1 if sending data. Note that
//the idle state for LCD_RS signal is high, regardless of the programming of the DATA_SELECT
//register.

BF_CS1 (LCDIF_CTRL, MODE86, 8080_MODE);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 0); //Must be 0 in MPU mode
BF_CS1 (LCDIF_CTRL1, BUSY_ENABLE, 0);
BF_CS4 (LCDIF_TIMING, CMD_HOLD, 2, CMD_SETUP, 2, DATA_HOLD, 2, DATA_SETUP, 2);  //Values
//based on DISPLAY_CLK frequency and timing requirements of controller. Note that these
//register must be non-zero for the MPU and VSYNC modes.
BF_CS2 (LCDIF_TRANSFER_COUNT, H_COUNT, 320, V_COUNT, 240);//For a 320 RGB x 240 display
//The following section indicates setting up the VSYNC signal timing when VSYNC is an output
BF_CS1 (LCDIF_VDCTRL0, VSYNC_OEB, 0); //Making VSYNC signal an output
BF_CS1 (LCDIF_VDCTRL4, VSYNC_ONLY, 1); //Only need to generate VSYNC signal
BF_CS1 (VDCTRL0, VSYNC_POL, 0); //Setting the polarity of VSYNC signal to be low during
//VSYNC_PULSE_WIDTH time
BF_CS2 (LCDIF_VDCTRL0, VSYNC_PERIOD_UNIT, 0, VSYNC_PULSE_WIDTH_UNIT, 0);
BF_CS2 (LCDIF_VDCTRL1, VSYNC_PERIOD, 400000, VSYNC_PULSE_WIDTH, 100);//Frame display rate in
//terms of number of DISPLAY_CLKs.
BF_CS2 (LCDIF_VDCTRL2, HSYNC_PULSE_WIDTH, 0, HSYNC_PERIOD, 0);
BF_CS1 (LCDIF_VDCTRL3, VERTICAL_WAIT_CNT, 50);
BF_CS1 (LCDIF_VDCTRL4, SYNC_SIGNALS_ON, 1);
BF_CS2 (LCDIF_CTRL, VSYNC_MODE, 1, WAIT_FOR_VSYNC_EDGE, 1); //set WAIT_FOR_VSYNC_EDGE if
//software wishes to transfer the next frame after the VSYNC edge occurs.
BF_CS1 (LCDIF_CTRL, RUN, 1);
```

The eLCDIF is now ready to receive data via bus master requests or PIO writes to the LCDIF_DATA register. When eLCDIF is done transmitting H_COUNT x V_COUNT pixels, it will stop, turn off the RUN bit and assert the cur_frame_done interrupt.

13.2.4.8 DOTCLK Interface

The DOTCLK interface is another mode used in moving picture displays.

It includes the VSYNC, HSYNC, DOTCLK and (optional) ENABLE signals. The interface is popularly called the RGB interface if the ENABLE signal is present.

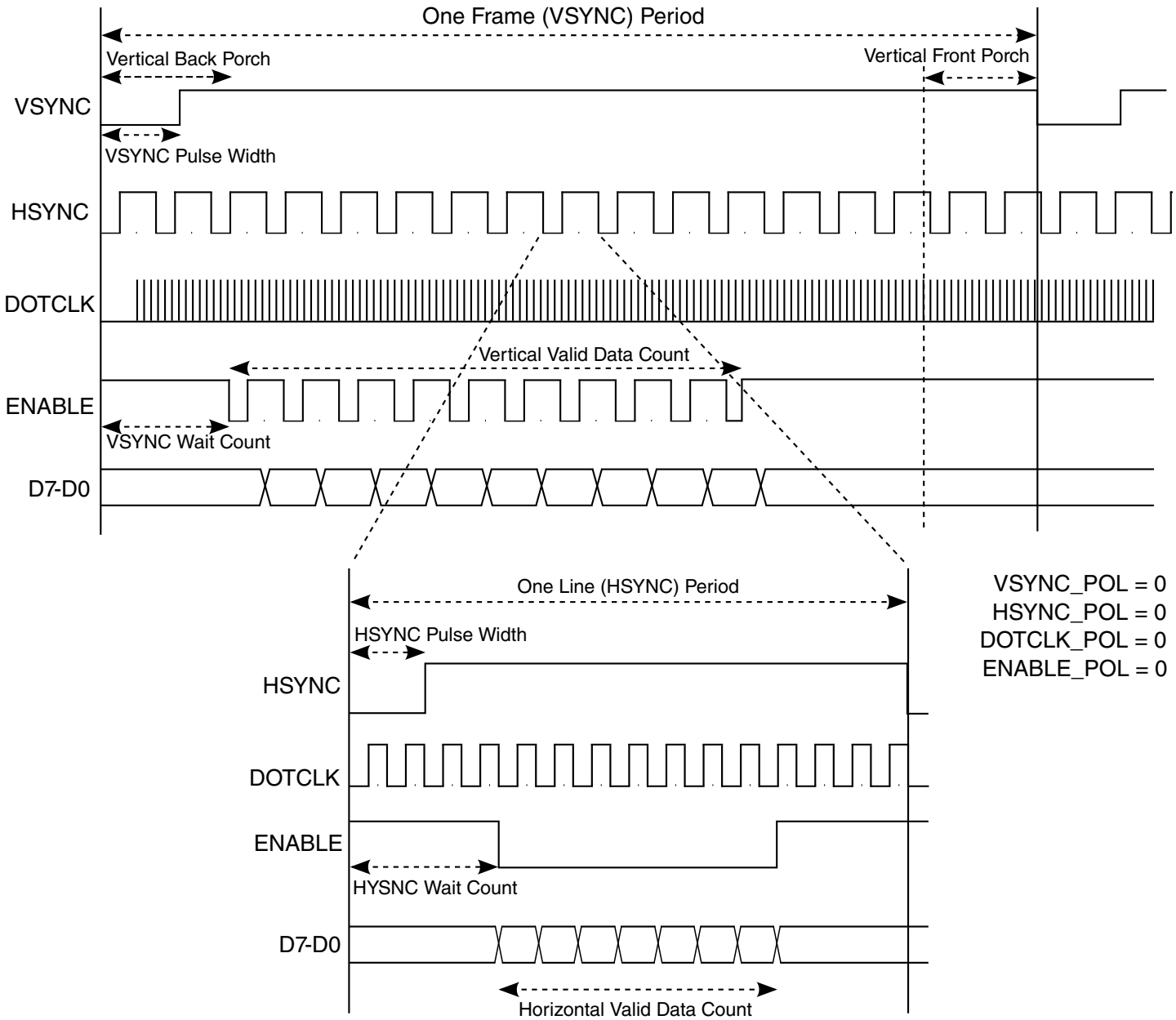


Figure 13-15. DOTCLK protocol with programmable parameters

The DOTCLK mode writes data at high speed to the LCD, and the display operation is synchronized with the VSYNC, HSYNC, ENABLE and DOTCLK signals. The polarities, periods and pulse-widths of the sync signals are programmable using the LCDIF_VDCTRL0-4 registers. The units for the VSYNC signal must be number of horizontal lines and can be selected using the VSYNC_PULSE_WIDTH_UNIT and VSYNC_PERIOD_UNIT bit fields. The VERTICAL_WAIT_CNT is by default given the same unit as the VSYNC_PERIOD. The DISPLAY_CLK frequency is managed by the CCM.

In DOTCLK mode, LCDIF_CTRL_BYPASS_COUNT bit must be set to 1. To end the current transfer, the software should make the DOTCLK_MODE bit 0, so that all data that is currently in the LCDIF LFIFO and TXFIFO is transmitted. Once that transfer is complete, the block will automatically clear the RUN bit and issue the cur_frame_done interrupt.

13.2.4.8.1 Code Example

The following code shows an example for programming a 320x240 display. Note that setting up the display must be done through the MPU mode or via SPI.

```
// Note: Common initialization steps in Initializing the eLCDIF must also be
// executed along with the following code
BF_CS1 (LCDIF_CTRL, DOTCLK_MODE, 1);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 1); //Always for DOTCLK mode
BF_CS1 (LCDIF_VDCTRL0, VSYNC_OEB, 0); //Vsync is always an output in the DOTCLK mode
BF_CS4 (LCDIF_VDCTRL0, VSYNC_POL, 0, HSYNC_POL, 0, DOTCLK_POL, 0, ENABLE_POL, 0);
BF_CS1 (LCDIF_VDCTRL0, ENABLE_PRESENT, 1);
BF_CS2 (LCDIF_VDCTRL0, VSYNC_PERIOD_UNIT, 1, VSYNC_PULSE_WIDTH_UNIT, 1);
BF_CS1 (LCDIF_VDCTRL0, VSYNC_PULSE_WIDTH, 2);
BF_CS1 (LCDIF_VDCTRL1, VSYNC_PERIOD, 280);
BF_CS2 (LCDIF_VDCTRL2, HSYNC_PULSE_WIDTH, 10, HSYNC_PERIOD, 360); //Assuming
                                                    //  LCD_DATABUS_WIDTH is 24bit
BF_CS2 (LCDIF_VDCTRL3, VSYNC_ONLY, 0);
BF_CS2 (LCDIF_VDCTRL3, HORIZONTAL_WAIT_CNT, 20, VERTICAL_WAIT_CNT, 20);
BF_CS1 (LCDIF_VDCTRL4, DOTCLK_H_VALID_DATA_CNT, 320);//Note that DOTCLK_V_VALID_DATA_CNT is
                              //implicitly assumed to be HW_LCDIF_TRANSFER_COUNT_V_COUNT
BF_CS1 (LCDIF_VDCTRL4, SYNC_SIGNALS_ON, 1);
BF_CS1 (LCDIF_CTRL, RUN, 1);
```

To stop the transfer completely, the ideal way is to make DOTCLK_MODE = 0. In that case, the block will transmit the contents in the FIFO and reset the RUN bit.

13.2.4.9 CSI HANDSHAKE INTERFACE

The LCDIF and CSI support a pipeline mode to use double buffers inside OCRAM for the video pass through from CSI to LCDIF, the pipeline buffer size can be 8 line or 16 line as configurable. The pipeline will be handle by hardware handshake signals between CSI and LCDIF. The LCDIF will have the capability to synchronize display with CSI based on the VSYNC signal from CSI. When LCDIF is enabled to start display,it can optionally wait for the VSYNC edge from CSI before it starts the display for next frame,The delay from VSYNC input to the start of next frame is programmable by LCDIF_SYNC_DELAY register.

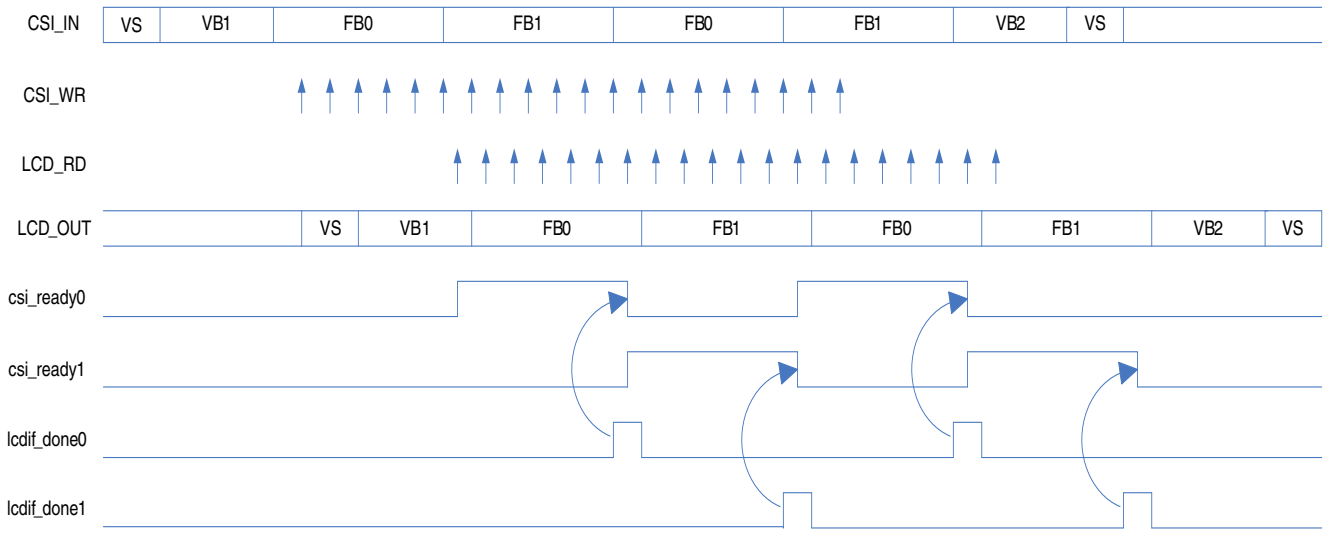


Figure 13-16. CSI HandShake Interface

When this mode is enabled ,The hardware handshake protocol will process. At the start of CSI with camera input,the CSI will put the input data to one frame buffer.When the frame buffer is ready for LCDIF display,CSI will assert csi_ready signal to indicate LCDIF to read the buffer and LCDIF will display data in this buffer.When one frame buffer reading finished ,LCDIF will set the lcdif_done signal to indicate CSI this buffer has been displayed and can be written again.With this double buffer handshake mode,the LCDIF can display the video input within very short delay and minimize DRAM bandwidth.

13.2.4.10 Alpha Blending Interface

The LCDIF have the capability to add an extra overlay on the normal display buffer,LCDIF can fetch data from two buffers and combine them before display,one buffer data can have the alpha value with the RGB pixels. With LCDIF_AS_CTRL[AS_ENABLE] is set, the LCDIF will start fetching alpha surface buffer data in bus master mode and combine it with another buffer.

The LCDIF_AS_CTRL[ALPHA_CTRL] bits determines how the alpha value is constructed for the alpha surface and alpha blending process is as same as in PXP block, for the alpha blend and color key process refer to the PXP block descriptions.

13.2.4.11 ITU-R BT.656 Digital Video Interface (DVI)

ITU-R BT.656 Digital Video Interface shown below transmits 4:2:2 YCbCr digital component video to a digital video encoder that can translate it into 525/60 or 625/50 analog TV signal.

Unique timing codes (timing reference signals) are embedded within the video stream to indicate the different timing events that would have been otherwise indicated by VSYNC, HSYNC and BLANK signals. The hardware supports 8-bit data transfers; the pins are shared with the lower 8 bits of LCD data bus. The LCD_RS pin is shared with the clock signal of the interface (called CCIRCLK here for uniqueness). CCIRCLK also can be obtained on the LCD_DOTCK pin. The mode shares the write FIFO with the LCD interface and the associated pipeline. The programmable parameters in registers LCDIF_DVICTRL0-3 allow setting the total number of horizontal lines per frame, vertical and horizontal blanking interval, odd and even field start and end positions, and so on. In short, these parameters are provided to ensure that the hardware has enough flexibility to generate the right 525/60 or 625/50 data streams. Most of the initialization steps in Initializing the eLCDIF such as data shifting, swizzle, and so on, are applicable to DVI mode also. The register descriptions in the programmable registers section at the end of this chapter include example code for programming the DVICTRL0-3 registers.

In DVI mode, LCDIF_CTRL_BYPASS_COUNT bit must be set to 1. To end the current transfer, the software should make the DVI_MODE bit the value 0, so that all data that is currently in the LCDIF LFIFO and TXFIFO is transmitted. Once that transfer is complete, the block will automatically clear the RUN bit and assert the cur_frame_done interrupt.

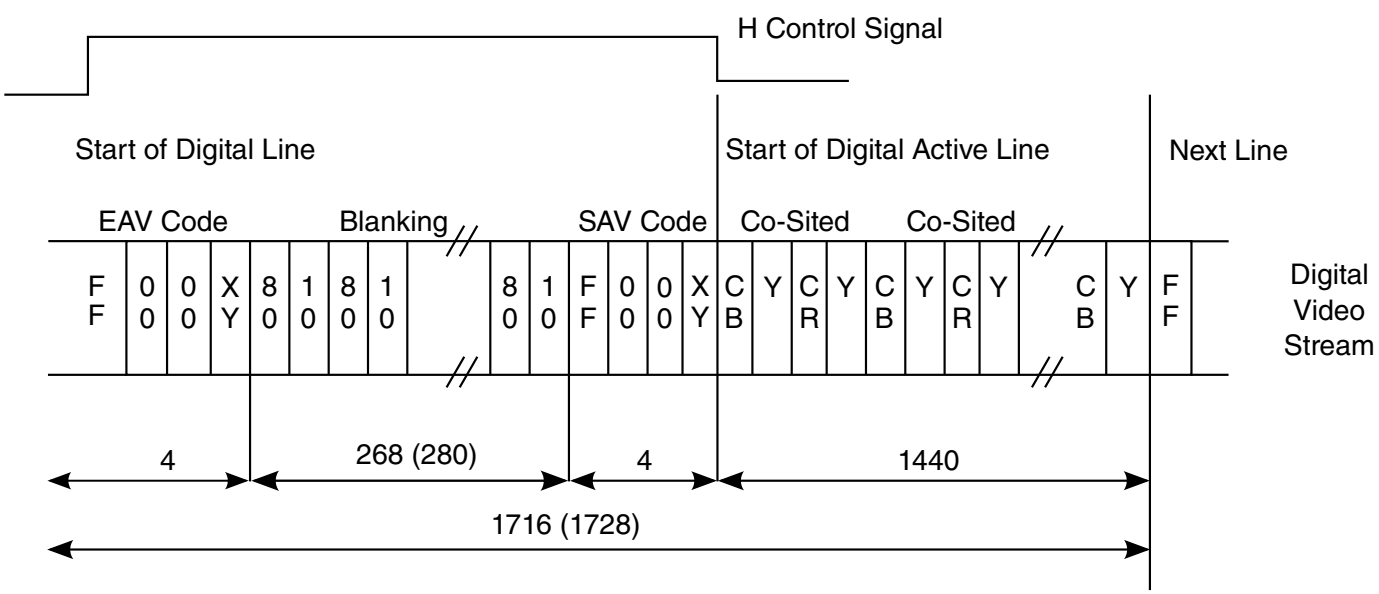


Figure 13-17. Digital Video Interface

13.2.4.12 eLCDIF Pin Usage by Interface Mode

Table 13-3 and Table 13-4 indicates how the eLCDIF level interface pins are used based on the desired mode of operation. The chip level I/Os should also be configured to be consistent with the desired eLCDIF operating mode.

The VSYNC signal has been mapped onto two pins, LCD_BUSY and LCD_VSYNC. The pin multiplexing can be programmed to select either of those pins to function as VSYN.

NOTE

There is an option to internally mux the HSYNC, DOTCLK and ENABLE signals in the DOTCLK mode by setting the MUX_SYNC_SIGNALS bit in the VDCTRL0 register. There is also an option to internally mux the LCD_WR_RWn and LCD_RD_E pins in the CTRL1 register for backward compatibility.

Table 13-3. Pin use in MPU Mode and VSYNC Mode

PIN NAME	8-bit MPU LCD IF	16-bit MPU LCD IF	18-bit MPU LCD IF	24-bit MPU LCD IF	8-bit VSYNC LCD IF	16-bit VSYNC LCD IF	18-bit VSYNC LCD IF	24-bit VSYNC LCD IF
LCD_RS	LCD_ RS	LCD_ RS	LCD_ RS	LCD_RS	LCD_RS	LCD_ RS	LCD_RS	LCD_RS
LCD_CS	LCD_CS	LCD_CS	LCD_ CS	LCD_CS	LCD_CS	LCD_CS	LCD_ CS	LCD_CS
LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn
LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E
LCD_VSYNC * (Two options)	X	X	X	X	LCD_ VSYNC	LCD_ VSYNC	LCD_ VSYNC	LCD_ VSYNC
LCD_HSYNC	X	X	X	X	X	X	X	X
LCD_DOTCLK	X	X	X	X	X	X	X	X
LCD_ENABLE	X	X	X	X	X	X	X	X
LCD_D23	X	X	X	LCD_D23	X	X	X	LCD_D23
LCD_D22	X	X	X	LCD_D22	X	X	X	LCD_D22
LCD_D21	X	X	X	LCD_D21	X	X	X	LCD_D21
LCD_D20	X	X	X	LCD_D20	X	X	X	LCD_D20
LCD_D19	X	X	X	LCD_D19	X	X	X	LCD_D19
LCD_D18	X	X	X	LCD_D18	X	X	X	LCD_D18
LCD_D17	X	X	LCD_D17	LCD_D17	X	X	LCD_D17	LCD_D17

Table continues on the next page...

Table 13-3. Pin use in MPU Mode and VSYNC Mode (continued)

PIN NAME	8-bit MPU LCD IF	16-bit MPU LCD IF	18-bit MPU LCD IF	24-bit MPU LCD IF	8-bit VSYNC LCD IF	16-bit VSYNC LCD IF	18-bit VSYNC LCD IF	24-bit VSYNC LCD IF
LCD_D16	X	X	LCD_D16	LCD_D16	X	X	LCD_D16	LCD_D16
LCD_D15 / VSYNC*	X	LCD_D15	LCD_D15	LCD_D15	VSYNC (optional)	LCD_D15	VSYNC (optional)	LCD_D15
LCD_D14 / HSYNC**	X	LCD_D14	LCD_D14	LCD_D14	X	LCD_D14	X	LCD_D14
LCD_D13 / LCD_DOTCLK**	X	LCD_D13	LCD_D13	LCD_D13	X	LCD_D13	X	LCD_D13
LCD_D12 / ENABLE**	X	LCD_D12	LCD_D12	LCD_D12	X	LCD_D12	X	LCD_D12
LCD_D11	X	LCD_D11	LCD_D11	LCD_D11	X	LCD_D11	X	LCD_D11
LCD_D10	X	LCD_D10	LCD_D10	LCD_D10	X	LCD_D10	X	LCD_D10
LCD_D9	X	LCD_D9	LCD_D9	LCD_D9	X	LCD_D9	X	LCD_D9
LCD_D8	X	LCD_D8	LCD_D8	LCD_D8	X	LCD_D8	X	LCD_D8
LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7
LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6
LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5
LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4
LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3
LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2
LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1
LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0
LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET
LCD_BUSY / LCD_VSYNC	LCD_BUSY	LCD_BUSY	LCD_BUSY	LCD_BUSY	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)

Table 13-4. Pin use in DOTCLK Mode and DVI Mode

PIN NAME	8-bit DOTCLK LCD IF	16-bit DOTCLK LCD IF	18-bit DOTCLK LCD IF	24-bit DOTCLK LCD IF	8-bit DVI LCD IF
LCD_RS	X	X	X	X	CCIR_CLK
LCD_CS	X	X	X	X	X
LCD_WR_RWn	X	X	X	X	X
LCD_RD_E	X	X	X	X	X
LCD_VSYNC*	LCD_VSYNC	LCD_VSYNC	LCD_VSYNC	LCD_VSYNC	X

Table continues on the next page...

Table 13-4. Pin use in DOTCLK Mode and DVI Mode (continued)

PIN NAME	8-bit DOTCLK LCD IF	16-bit DOTCLK LCD IF	18-bit DOTCLK LCD IF	24-bit DOTCLK LCD IF	8-bit DVI LCD IF
(Two options)					
LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	X
LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK	X
LCD_ENABLE	LCD_ENABLE	LCD_ENABLE	LCD_ENABLE	LCD_ENABLE	X
LCD_D23	X	X	X	LCD_D23	X
LCD_D22	X	X	X	LCD_D22	X
LCD_D21	X	X	X	LCD_D21	X
LCD_D20	X	X	X	LCD_D20	X
LCD_D19	X	X	X	LCD_D19	X
LCD_D18	X	X	X	LCD_D18	X
LCD_D17	X	X	LCD_D17	LCD_D17	X
LCD_D16	X	X	LCD_D16	LCD_D16	X
LCD_D15/ VSYNC*	X	LCD_D15	LCD_D15	LCD_D15	X
LCD_D14 / HSYNC**	X	LCD_D14	LCD_D14	LCD_D14	X
LCD_D13 / LCD_DOTCLK**	X	LCD_D13	LCD_D13	LCD_D13	X
LCD_D12 / ENABLE**	X	LCD_D12	LCD_D12	LCD_D12	X
LCD_D11	X	LCD_D11	LCD_D11	LCD_D11	X
LCD_D10	X	LCD_D10	LCD_D10	LCD_D10	X
LCD_D9	X	LCD_D9	LCD_D9	LCD_D9	X
LCD_D8	X	LCD_D8	LCD_D8	LCD_D8	X
LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7
LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6
LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5
LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4
LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3
LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2
LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1
LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0
LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	X
LCD_BUSY / LCD_VSYNC	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	X

13.2.5 Behavior During Reset

BUS_CLK and DISPLAY_CLK must be running before making any changes to SFTRST or CLKGATE bits.

A soft reset (SFTRST) can take multiple clock periods to complete, so do not set CLKGATE when setting SFTRST.

The reset process gates the clocks automatically.

13.2.6 ELCDIF Memory Map/Register Definition

eLCDIF Hardware Register Format Summary

LCDIF memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3073_0000	eLCDIF General Control Register (LCDIF1_RL)	32	R/W	C000_0000h	13.2.6.1/ 3581
3073_0004	eLCDIF General Control Register (LCDIF1_RL_SET)	32	R/W	C000_0000h	13.2.6.1/ 3581
3073_0008	eLCDIF General Control Register (LCDIF1_RL_CLR)	32	R/W	C000_0000h	13.2.6.1/ 3581
3073_000C	eLCDIF General Control Register (LCDIF1_RL_TOG)	32	R/W	C000_0000h	13.2.6.1/ 3581
3073_0010	eLCDIF General Control1 Register (LCDIF1_CTRL1)	32	R/W	000F_0000h	13.2.6.2/ 3584
3073_0014	eLCDIF General Control1 Register (LCDIF1_CTRL1_SET)	32	R/W	000F_0000h	13.2.6.2/ 3584
3073_0018	eLCDIF General Control1 Register (LCDIF1_CTRL1_CLR)	32	R/W	000F_0000h	13.2.6.2/ 3584
3073_001C	eLCDIF General Control1 Register (LCDIF1_CTRL1_TOG)	32	R/W	000F_0000h	13.2.6.2/ 3584
3073_0020	eLCDIF General Control2 Register (LCDIF1_CTRL2)	32	R/W	0020_0000h	13.2.6.3/ 3586
3073_0024	eLCDIF General Control2 Register (LCDIF1_CTRL2_SET)	32	R/W	0020_0000h	13.2.6.3/ 3586
3073_0028	eLCDIF General Control2 Register (LCDIF1_CTRL2_CLR)	32	R/W	0020_0000h	13.2.6.3/ 3586
3073_002C	eLCDIF General Control2 Register (LCDIF1_CTRL2_TOG)	32	R/W	0020_0000h	13.2.6.3/ 3586

Table continues on the next page...

LCDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3073_0030	eLCDIF Horizontal and Vertical Valid Data Count Register (LCDIF1_TRANSFER_COUNT)	32	R/W	0001_0000h	13.2.6.4/ 3589
3073_0040	LCD Interface Current Buffer Address Register (LCDIF1_CUR_BUF)	32	R/W	0000_0000h	13.2.6.5/ 3589
3073_0050	LCD Interface Next Buffer Address Register (LCDIF1_NEXT_BUF)	32	R/W	0000_0000h	13.2.6.6/ 3590
3073_0060	LCD Interface Timing Register (LCDIF1_TIMING)	32	R/W	0000_0000h	13.2.6.7/ 3590
3073_0070	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF1_VDCTRL0)	32	R/W	0000_0000h	13.2.6.8/ 3591
3073_0074	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF1_VDCTRL0_SET)	32	R/W	0000_0000h	13.2.6.8/ 3591
3073_0078	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF1_VDCTRL0_CLR)	32	R/W	0000_0000h	13.2.6.8/ 3591
3073_007C	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF1_VDCTRL0_TOG)	32	R/W	0000_0000h	13.2.6.8/ 3591
3073_0080	eLCDIF VSYNC Mode and Dotclk Mode Control Register1 (LCDIF1_VDCTRL1)	32	R/W	0000_0000h	13.2.6.9/ 3592
3073_0090	LCDIF VSYNC Mode and Dotclk Mode Control Register2 (LCDIF1_VDCTRL2)	32	R/W	0000_0000h	13.2.6.10/ 3593
3073_00A0	eLCDIF VSYNC Mode and Dotclk Mode Control Register3 (LCDIF1_VDCTRL3)	32	R/W	0000_0000h	13.2.6.11/ 3593
3073_00B0	eLCDIF VSYNC Mode and Dotclk Mode Control Register4 (LCDIF1_VDCTRL4)	32	R/W	0000_0000h	13.2.6.12/ 3594
3073_00C0	Digital Video Interface Control0 Register (LCDIF1_DVICTRL0)	32	R/W	0000_0000h	13.2.6.13/ 3595
3073_00D0	Digital Video Interface Control1 Register (LCDIF1_DVICTRL1)	32	R/W	0000_0000h	13.2.6.14/ 3596
3073_00E0	Digital Video Interface Control2 Register (LCDIF1_DVICTRL2)	32	R/W	0000_0000h	13.2.6.15/ 3597
3073_00F0	Digital Video Interface Control3 Register (LCDIF1_DVICTRL3)	32	R/W	0000_0000h	13.2.6.16/ 3598
3073_0100	Digital Video Interface Control4 Register (LCDIF1_DVICTRL4)	32	R/W	0000_0000h	13.2.6.17/ 3599
3073_0110	RGB to YCbCr 4:2:2 CSC Coefficient0 Register (LCDIF1_CSC_COEFF0)	32	R/W	0000_0000h	13.2.6.18/ 3600
3073_0120	RGB to YCbCr 4:2:2 CSC Coefficient1 Register (LCDIF1_CSC_COEFF1)	32	R/W	0000_0000h	13.2.6.19/ 3601
3073_0130	RGB to YCbCr 4:2:2 CSC Coefficent2 Register (LCDIF1_CSC_COEFF2)	32	R/W	0000_0000h	13.2.6.20/ 3601
3073_0140	RGB to YCbCr 4:2:2 CSC Coefficient3 Register (LCDIF1_CSC_COEFF3)	32	R/W	0000_0000h	13.2.6.21/ 3602
3073_0150	RGB to YCbCr 4:2:2 CSC Coefficient4 Register (LCDIF1_CSC_COEFF4)	32	R/W	0000_0000h	13.2.6.22/ 3603

Table continues on the next page...

LCDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3073_0160	RGB to YCbCr 4:2:2 CSC Offset Register (LCDIF1_CSC_OFFSET)	32	R/W	0080_0010h	13.2.6.23/ 3604
3073_0170	RGB to YCbCr 4:2:2 CSC Limit Register (LCDIF1_CSC_LIMIT)	32	R/W	00FF_00FFh	13.2.6.24/ 3604
3073_0180	LCD Interface Data Register (LCDIF1_DATA)	32	R/W	0000_0000h	13.2.6.25/ 3605
3073_0190	Bus Master Error Status Register (LCDIF1_BM_ERROR_STAT)	32	R/W	0000_0000h	13.2.6.26/ 3606
3073_01A0	CRC Status Register (LCDIF1_CRC_STAT)	32	R/W	0000_0000h	13.2.6.27/ 3606
3073_01B0	LCD Interface Status Register (LCDIF1_STAT)	32	R	9500_0000h	13.2.6.28/ 3607
3073_01C0	LCD Interface Version Register (LCDIF1_VERSION)	32	R	0400_0000h	13.2.6.29/ 3608
3073_01D0	LCD Interface Debug0 Register (LCDIF1_DEBUG0)	32	R	0E81_0000h	13.2.6.30/ 3609
3073_01E0	LCD Interface Debug1 Register (LCDIF1_DEBUG1)	32	R	0000_0000h	13.2.6.31/ 3612
3073_01F0	LCD Interface Debug2 Register (LCDIF1_DEBUG2)	32	R	0000_0000h	13.2.6.32/ 3613
3073_0200	eLCDIF Threshold Register (LCDIF1_THRES)	32	R/W	0100_000Fh	13.2.6.33/ 3613
3073_0210	eLCDIF AS Buffer Control Register (LCDIF1_AS_CTRL)	32	R/W	0000_0000h	13.2.6.34/ 3615
3073_0220	Alpha Surface Buffer Pointer (LCDIF1_AS_BUF)	32	R/W	0000_0000h	13.2.6.35/ 3617
3073_0230	LCDIF1_AS_NEXT_BUF	32	R/W	0000_0000h	13.2.6.36/ 3618
3073_0240	eLCDIF Overlay Color Key Low (LCDIF1_AS_CLRKEYLOW)	32	R/W	00FF_FFFFh	13.2.6.37/ 3618
3073_0250	eLCDIF Overlay Color Key High (LCDIF1_AS_CLRKEYHIGH)	32	R/W	0000_0000h	13.2.6.38/ 3619
3073_0260	LCD working insync mode with CSI for VSYNC delay (LCDIF1_SYNC_DELAY)	32	R/W	0000_0000h	13.2.6.39/ 3619
3073_0270	eLCDIF Interface Debug3 Register (LCDIF1_DEBUG3)	32	R/W	0000_0000h	13.2.6.40/ 3620
3073_0280	LCD Interface Debug4 (LCDIF1_DEBUG4)	32	R/W	0000_0000h	13.2.6.41/ 3621
3073_0290	LCD Interface Debug5 (LCDIF1_DEBUG5)	32	R/W	0000_0000h	13.2.6.42/ 3622
3073_4000	eLCDIF General Control Register (LCDIF2_RL)	32	R/W	C000_0000h	13.2.6.1/ 3581
3073_4004	eLCDIF General Control Register (LCDIF2_RL_SET)	32	R/W	C000_0000h	13.2.6.1/ 3581

Table continues on the next page...

LCDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3073_4008	eLCDIF General Control Register (LCDIF2_RL_CLR)	32	R/W	C000_0000h	13.2.6.1/ 3581
3073_400C	eLCDIF General Control Register (LCDIF2_RL_TOG)	32	R/W	C000_0000h	13.2.6.1/ 3581
3073_4010	eLCDIF General Control1 Register (LCDIF2_CTRL1)	32	R/W	000F_0000h	13.2.6.2/ 3584
3073_4014	eLCDIF General Control1 Register (LCDIF2_CTRL1_SET)	32	R/W	000F_0000h	13.2.6.2/ 3584
3073_4018	eLCDIF General Control1 Register (LCDIF2_CTRL1_CLR)	32	R/W	000F_0000h	13.2.6.2/ 3584
3073_401C	eLCDIF General Control1 Register (LCDIF2_CTRL1_TOG)	32	R/W	000F_0000h	13.2.6.2/ 3584
3073_4020	eLCDIF General Control2 Register (LCDIF2_CTRL2)	32	R/W	0020_0000h	13.2.6.3/ 3586
3073_4024	eLCDIF General Control2 Register (LCDIF2_CTRL2_SET)	32	R/W	0020_0000h	13.2.6.3/ 3586
3073_4028	eLCDIF General Control2 Register (LCDIF2_CTRL2_CLR)	32	R/W	0020_0000h	13.2.6.3/ 3586
3073_402C	eLCDIF General Control2 Register (LCDIF2_CTRL2_TOG)	32	R/W	0020_0000h	13.2.6.3/ 3586
3073_4030	eLCDIF Horizontal and Vertical Valid Data Count Register (LCDIF2_TRANSFER_COUNT)	32	R/W	0001_0000h	13.2.6.4/ 3589
3073_4040	LCD Interface Current Buffer Address Register (LCDIF2_CUR_BUF)	32	R/W	0000_0000h	13.2.6.5/ 3589
3073_4050	LCD Interface Next Buffer Address Register (LCDIF2_NEXT_BUF)	32	R/W	0000_0000h	13.2.6.6/ 3590
3073_4060	LCD Interface Timing Register (LCDIF2_TIMING)	32	R/W	0000_0000h	13.2.6.7/ 3590
3073_4070	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF2_VDCTRL0)	32	R/W	0000_0000h	13.2.6.8/ 3591
3073_4074	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF2_VDCTRL0_SET)	32	R/W	0000_0000h	13.2.6.8/ 3591
3073_4078	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF2_VDCTRL0_CLR)	32	R/W	0000_0000h	13.2.6.8/ 3591
3073_407C	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF2_VDCTRL0_TOG)	32	R/W	0000_0000h	13.2.6.8/ 3591
3073_4080	eLCDIF VSYNC Mode and Dotclk Mode Control Register1 (LCDIF2_VDCTRL1)	32	R/W	0000_0000h	13.2.6.9/ 3592
3073_4090	LCDIF VSYNC Mode and Dotclk Mode Control Register2 (LCDIF2_VDCTRL2)	32	R/W	0000_0000h	13.2.6.10/ 3593
3073_40A0	eLCDIF VSYNC Mode and Dotclk Mode Control Register3 (LCDIF2_VDCTRL3)	32	R/W	0000_0000h	13.2.6.11/ 3593
3073_40B0	eLCDIF VSYNC Mode and Dotclk Mode Control Register4 (LCDIF2_VDCTRL4)	32	R/W	0000_0000h	13.2.6.12/ 3594

Table continues on the next page...

LCDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3073_40C0	Digital Video Interface Control0 Register (LCDIF2_DVICTRL0)	32	R/W	0000_0000h	13.2.6.13/ 3595
3073_40D0	Digital Video Interface Control1 Register (LCDIF2_DVICTRL1)	32	R/W	0000_0000h	13.2.6.14/ 3596
3073_40E0	Digital Video Interface Control2 Register (LCDIF2_DVICTRL2)	32	R/W	0000_0000h	13.2.6.15/ 3597
3073_40F0	Digital Video Interface Control3 Register (LCDIF2_DVICTRL3)	32	R/W	0000_0000h	13.2.6.16/ 3598
3073_4100	Digital Video Interface Control4 Register (LCDIF2_DVICTRL4)	32	R/W	0000_0000h	13.2.6.17/ 3599
3073_4110	RGB to YCbCr 4:2:2 CSC Coefficient0 Register (LCDIF2_CSC_COEFF0)	32	R/W	0000_0000h	13.2.6.18/ 3600
3073_4120	RGB to YCbCr 4:2:2 CSC Coefficient1 Register (LCDIF2_CSC_COEFF1)	32	R/W	0000_0000h	13.2.6.19/ 3601
3073_4130	RGB to YCbCr 4:2:2 CSC Coefficent2 Register (LCDIF2_CSC_COEFF2)	32	R/W	0000_0000h	13.2.6.20/ 3601
3073_4140	RGB to YCbCr 4:2:2 CSC Coefficient3 Register (LCDIF2_CSC_COEFF3)	32	R/W	0000_0000h	13.2.6.21/ 3602
3073_4150	RGB to YCbCr 4:2:2 CSC Coefficient4 Register (LCDIF2_CSC_COEFF4)	32	R/W	0000_0000h	13.2.6.22/ 3603
3073_4160	RGB to YCbCr 4:2:2 CSC Offset Register (LCDIF2_CSC_OFFSET)	32	R/W	0080_0010h	13.2.6.23/ 3604
3073_4170	RGB to YCbCr 4:2:2 CSC Limit Register (LCDIF2_CSC_LIMIT)	32	R/W	00FF_00FFh	13.2.6.24/ 3604
3073_4180	LCD Interface Data Register (LCDIF2_DATA)	32	R/W	0000_0000h	13.2.6.25/ 3605
3073_4190	Bus Master Error Status Register (LCDIF2_BM_ERROR_STAT)	32	R/W	0000_0000h	13.2.6.26/ 3606
3073_41A0	CRC Status Register (LCDIF2_CRC_STAT)	32	R/W	0000_0000h	13.2.6.27/ 3606
3073_41B0	LCD Interface Status Register (LCDIF2_STAT)	32	R	9500_0000h	13.2.6.28/ 3607
3073_41C0	LCD Interface Version Register (LCDIF2_VERSION)	32	R	0400_0000h	13.2.6.29/ 3608
3073_41D0	LCD Interface Debug0 Register (LCDIF2_DEBUG0)	32	R	0E81_0000h	13.2.6.30/ 3609
3073_41E0	LCD Interface Debug1 Register (LCDIF2_DEBUG1)	32	R	0000_0000h	13.2.6.31/ 3612
3073_41F0	LCD Interface Debug2 Register (LCDIF2_DEBUG2)	32	R	0000_0000h	13.2.6.32/ 3613
3073_4200	eLCDIF Threshold Register (LCDIF2_THRES)	32	R/W	0100_000Fh	13.2.6.33/ 3613
3073_4210	eLCDIF AS Buffer Control Register (LCDIF2_AS_CTRL)	32	R/W	0000_0000h	13.2.6.34/ 3615

Table continues on the next page...

LCDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3073_4220	Alpha Surface Buffer Pointer (LCDIF2_AS_BUF)	32	R/W	0000_0000h	13.2.6.35/ 3617
3073_4230	LCDIF2_AS_NEXT_BUF	32	R/W	0000_0000h	13.2.6.36/ 3618
3073_4240	eLCDIF Overlay Color Key Low (LCDIF2_AS_CLRKEYLOW)	32	R/W	00FF_FFFFh	13.2.6.37/ 3618
3073_4250	eLCDIF Overlay Color Key High (LCDIF2_AS_CLRKEYHIGH)	32	R/W	0000_0000h	13.2.6.38/ 3619
3073_4260	LCD working insync mode with CSI for VSYNC delay (LCDIF2_SYNC_DELAY)	32	R/W	0000_0000h	13.2.6.39/ 3619
3073_4270	eLCDIF Interface Debug3 Register (LCDIF2_DEBUG3)	32	R/W	0000_0000h	13.2.6.40/ 3620
3073_4280	LCD Interface Debug4 (LCDIF2_DEBUG4)	32	R/W	0000_0000h	13.2.6.41/ 3621
3073_4290	LCD Interface Debug5 (LCDIF2_DEBUG5)	32	R/W	0000_0000h	13.2.6.42/ 3622

13.2.6.1 eLCDIF General Control Register (LCDIF*x*_RL*n*)

The LCD Interface Control Register provides overall control of the eLCDIF block. The eLCDIF Control Register provides a variety of control functions to the programmer. These functions allow the interface to be very flexible to work with a variety of LCD controllers, and to minimize overhead and increase performance of LCD programming. The register has been organized such that switching between the different LCD modes can be done with minimum PIO writes.

Address: Base address + 0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SFTRST	CLKGATE	YCBCR422_INPUT	READ_WRITEB	WAIT_FOR_VSYNC_EDGE	DATA_SHIFT_DIR	SHIFT_NUM_BITS					DVI_MODE	BYPASS_COUNT	VSYNC_MODE	DOTCLK_MODE	DATA_SELECT
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_DATA_SWIZZLE		CSC_DATA_SWIZZLE		LCD_DATABUS_WIDTH		WORD_LENGTH		RGB_TO_YCBCR422_CSC	ENABLE_PXP_HANDSHAKE	MASTER	Reserved	DATA_FORMAT_16_BIT	DATA_FORMAT_18_BIT	DATA_FORMAT_24_BIT	RUN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF*x*_RL*n* field descriptions

Field	Description
31 SFTRST	This bit must be set to zero to enable normal operation of the eLCDIF. When set to one, it forces a block level reset.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29 YCBCR422_INPUT	Zero implies input data is in RGB color space. One implies input data is in YCbCr 4:2:2 format, such that YCbYCr are packed in a 32-bit word. It also means that there are 2 pixels in 4 bytes. If this bit is set, software should program the H_COUNT field in the TRANSFER_COUNT register to the total number of pixels that will have to be fetched by the eLCDIF block per line and the BYTE_PACKING_FORMAT should be 0xF. The WORD_LENGTH does not matter in this case.

Table continues on the next page...

LCDIF*x*_RL*n* field descriptions (continued)

Field	Description
28 READ_WRITEB	By default, eLCDIF is in the write mode. Setting this bit to 1 will make the hardware go into 6800/8080 MPU read mode. The LCDIF_MASTER bit must be 0, since bus master mode can only be used for writing the display.
27 WAIT_FOR_ VSYNC_EDGE	Setting this bit to 1 will make the hardware wait for the triggering VSYNC edge before starting write transfers to the LCD. Used only in the VSYNC mode of operation.
26 DATA_SHIFT_ DIR	Use this bit to determine the direction of shift of transmit data. In the DVI mode, it works only on the active data, not on the timing codes and ancillary data. 0x0 **TXDATA_SHIFT_LEFT** — Data to be transmitted is shifted LEFT by SHIFT_NUM_BITS bits. 0x1 **TXDATA_SHIFT_RIGHT** — Data to be transmitted is shifted RIGHT by SHIFT_NUM_BITS bits.
25–21 SHIFT_NUM_ BITS	The data to be transmitted is shifted left or right by this number of bits.
20 DVI_MODE	Set this bit to 1 to get into the ITU-R BT.656 digital video interface mode. Toggle this bit from 1 to 0 to make the hardware go out of DVI mode after completing all data transfer and after the RUN bit has been deasserted.
19 BYPASS_ COUNT	When this bit is 0, it means that eLCDIF will stop the block operation and turn off the RUN bit after the amount of data indicated by the LCDIF_TRANSFER_COUNT register has been transferred out. When this bit is set to 1, the block will continue normal operation indefinitely until it is told to stop. This bit must be 0 in MPU and VSYNC modes, and must be 1 in DOTCLK and DVI modes of operation.
18 VSYNC_MODE	Setting this bit to 1 will make the eLCDIF hardware go into VSYNC mode. WAIT_FOR_VSYNC_EDGE can be used only if this bit is set. If VSYNC signal is required to be an output from the block, SYNC_SIGNALS_ON bit in LCDIF_VDCTRL4 register must be set.
17 DOTCLK_MODE	Set this bit to 1 to make the hardware go into the DOTCLK mode, i.e. VSYNC/HSYNC/DOTCLK/ENABLE interface mode. ENABLE is optional, selected by the ENABLE_PRESENT bit. Toggle this bit from 1 to 0 to make the hardware go out of DOTCLK mode after completing all data transfer and deasserting the RUN bit.
16 DATA_SELECT	Command Mode polarity bit. This bit should only be changed when RUN is 0. 0x0 **CMD_MODE** — Command Mode. DCn signal is Low. 0x1 **DATA_MODE** — Data Mode. DCn signal is High.
15–14 INPUT_DATA_ SWIZZLE	This field specifies how to swap the bytes fetched by the bus master interface. The swizzle function is independent of the WORD_LENGTH bit. The supported swizzle configurations are: 0x0 **NO_SWAP** — No byte swapping.(Little endian) 0x0 **LITTLE_ENDIAN** — Little Endian byte ordering (same as NO_SWAP). 0x1 **BIG_ENDIAN_SWAP** — Big Endian swap (swap bytes 0,3 and 1,2). 0x1 **SWAP_ALL_BYTES** — Swizzle all bytes, swap bytes 0,3 and 1,2 (aka Big Endian). 0x2 **HWD_SWAP** — Swap half-words. 0x3 **HWD_BYTE_SWAP** — Swap bytes within each half-word.
13–12 CSC_DATA_ SWIZZLE	This field specifies how to swap the bytes after the data has been converted into an internal representation of 24 bits per pixel and before it is transmitted over the LCD interface bus. The data is always transmitted with the least significant byte/hword (half word) first after the swizzle takes place. So, INPUT_DATA_SWIZZLE takes place first on the incoming data, and then CSC_DATA_SWIZZLE is applied. The swizzle function is independent of the WORD_LENGTH or the LCD_DATABUS_WIDTH fields. If RGB_TO_YCRCB422_CSC bit is set, the swizzle occurs on the Y, Cb, Cr values. The supported swizzle configurations are: 0x0 **NO_SWAP** — No byte swapping.(Little endian)

Table continues on the next page...

LCDIF*x*_RL*n* field descriptions (continued)

Field	Description
	0x0 **LITTLE_ENDIAN** — Little Endian byte ordering (same as NO_SWAP). 0x1 **BIG_ENDIAN_SWAP** — Big Endian swap (swap bytes 0,3 and 1,2). 0x1 **SWAP_ALL_BYTES** — Swizzle all bytes, swap bytes 0,3 and 1,2 (aka Big Endian). 0x2 **HWD_SWAP** — Swap half-words. 0x3 **HWD_BYTE_SWAP** — Swap bytes within each half-word.
11–10 LCD_DATABUS_WIDTH	LCD Data bus transfer width. 0x0 **16_BIT** — 16-bit data bus mode. 0x1 **8_BIT** — 8-bit data bus mode. 0x2 **18_BIT** — 18-bit data bus mode. 0x3 **24_BIT** — 24-bit data bus mode.
9–8 WORD_LENGTH	Input data format. 0x0 **16_BIT** — Input data is 16 bits per pixel. 0x1 **8_BIT** — Input data is 8 bits wide. 0x2 **18_BIT** — Input data is 18 bits per pixel. 0x3 **24_BIT** — Input data is 24 bits per pixel.
7 RGB_TO_YCBCR422_CSC	Set this bit to 1 to enable conversion from RGB to YCbCr colorspace. See the LCDIF_CSC_ registers for further details.
6 ENABLE_PXP_HANDSHAKE	If this bit is set and LCDIF_MASTER bit is set, the eLCDIF will act as bus master and the handshake mechanism between eLCDIF and PXP will be turned on. If LCDIF_MASTER bit is not set, this bit becomes a don't care.
5 MASTER	Set this bit to make the eLCDIF act as a bus master. If this bit is reset, the eLCDIF will support or PIO mode.
4 RSRVD0	This field is reserved. Reserved bits. Write as 0.
3 DATA_FORMAT_16_BIT	When this bit is 1 and WORD_LENGTH = 0, it implies that the 16-bit data is in ARGB555 format. When this bit is 0 and WORD_LENGTH = 0, it implies that the 16-bit data is in RGB565 format. When WORD_LENGTH is not 0, this bit does not care.
2 DATA_FORMAT_18_BIT	Used only when WORD_LENGTH = 2, i.e. 18-bit. 0x0 **LOWER_18_BITS_VALID** — Data input to the block is in 18 bpp format, such that lower 18 bits contain RGB 666 and upper 14 bits do not contain any useful data. 0x1 **UPPER_18_BITS_VALID** — Data input to the block is in 18 bpp format, such that upper 18 bits contain RGB 666 and lower 14 bits do not contain any useful data.
1 DATA_FORMAT_24_BIT	Used only when WORD_LENGTH = 3, i.e. 24-bit. Note that this applies to both packed and unpacked 24-bit data. 0x0 **ALL_24_BITS_VALID** — Data input to the block is in 24 bpp format, such that all RGB 888 data is contained in 24 bits. 0x1 **DROP_UPPER_2_BITS_PER_BYTE** — Data input to the block is actually RGB 18 bpp, but there is 1 color per byte, hence the upper 2 bits in each byte do not contain any useful data, and should be dropped.
0 RUN	When this bit is set by software, the eLCDIF will begin transferring data between the SoC and the display. This bit must remain set until the operation is complete.

13.2.6.2 eLCDIF General Control1 Register (LCDIF*x*_CTRL1*n*)

The eLCDIF Control Register provides overall control of the eLCDIF block.

The eLCDIF Control1 Register provides additional programming to the eLCDIF. It implements some bits which are unlikely to change often in a particular application. It also carries interrupt-related bits which are common across more than one mode of operation.

Address: Base address + 10h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				COMBINE_MPU_WR_STRB	BM_ERROR_IRQ_EN	BM_ERROR_IRQ	RECOVER_ON_UNDERFLOW	INTERLACE_FIELDS	START_INTERLACE_FROM_SECOND_FIELD	FIFO_CLEAR	IRQ_ON_ALTERNATE_FIELDS	BYTE_PACKING_FORMAT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OVERFLOW_IRQ_EN	UNDERFLOW_IRQ_EN	CUR_FRAME_DONE_IRQ_EN	VSYNC_EDGE_IRQ_EN	OVERFLOW_IRQ	UNDERFLOW_IRQ	CUR_FRAME_DONE_IRQ	VSYNC_EDGE_IRQ	Reserved					BUSY_ENABLE	MODE86	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF*x*_CTRL1*n* field descriptions

Field	Description
31–28 RSRVD1	This field is reserved. Reserved bits. Write as 0.
27 COMBINE_MPU_WR_STRB	If this bit is not set, the write strobe will be driven on LCD_WR_RWn pin in the 8080 mode and on the LCD_RD_E pin in the 6800 mode. If it is set, the write strobe of both the 6800 and 8080 modes will be driven only on the LCD_WR_RWn pin. Note that this does not work for read strobe.
26 BM_ERROR_IRQ_EN	This bit is set to enable bus master error interrupt in the eLCDIF master mode.
25 BM_ERROR_IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. This bit will be set when the eLCDIF is in master mode and an error response was returned by the slave.

Table continues on the next page...

LCDIF*x*_CTRL1*n* field descriptions (continued)

Field	Description
	0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
24 RECOVER_ON_ UNDERFLOW	Set this bit to enable the eLCDIF block to recover in the next field/frame if there was an underflow in the current field/frame.
23 INTERLACE_ FIELDS	Set this bit if it is required that the eLCDIF block fetches odd lines in one field and even lines in the other field. It will work only in LCDIF_MASTER is set to 1.
22 START_ INTERLACE_ FROM_ SECOND_FIELD	The default is to grab the odd lines first and then the even lines. Set this bit if it is required to grab the even lines first and then the odd lines. (Line numbers start from 1, so odd lines are 1,3,5,etc. and even lines are 2,4,6, etc.)
21 FIFO_CLEAR	Set this bit to clear all the data in the latency FIFO (LFIFO), TXFIFO and the RXFIFO.
20 IRQ_ON_ ALTERNATE_ FIELDS	If this bit is set, the eLCDIF block will assert the cur_frame_done interrupt only on alternate fields, otherwise it will issue the interrupt on both odd and even field. This bit is mostly relevant if INTERLACE_FIELDS is set. This feature is only available in DOTCLK and DVI modes.
19–16 BYTE_ PACKING_ FORMAT	This bitfield is used to show which data bytes in a 32-bit word are valid. Default value 0xf indicates that all bytes are valid. For 8-bit transfers, any combination in this bitfield will mean valid data is present in the corresponding bytes. In the 16-bit mode, a 16-bit half-word is valid only if adjacent bits [1:0] or [3:2] or both are 1. A value of 0x0 will mean that none of the bytes are valid and should not be used. For example, set the bit field value to 0x7 if the display data is arranged in the 24-bit unpacked format (A-R-G-B where A value does not have be transmitted). When input data is in YCbCr 4:2:2 format (YCBCR422_INPUT is 1), H_COUNT should be the number of pixels that should be fetched by the block and the BYTE_PACKING_FORMAT should be 0xF.(Note - YCBCR422_INPUT = 1 implies 2 pixels per 32 bits).
15 OVERFLOW_ IRQ_EN	This bit is set to enable an overflow interrupt in the TXFIFO in the write mode.
14 UNDERFLOW_ IRQ_EN	This bit is set to enable an underflow interrupt in the TXFIFO in the write mode.
13 CUR_FRAME_ DONE_IRQ_EN	This bit is set to 1 enable an interrupt every time the hardware enters in the vertical blanking state.
12 VSYNC_EDGE_ IRQ_EN	This bit is set to enable an interrupt every time the hardware encounters the leading VSYNC edge in the VSYNC and DOTCLK modes, or the beginning of every field in DVI mode.
11 OVERFLOW_ IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. A latency FIFO (LFIFO) overflow in the write mode (MPU/VSYNC/DOTCLK/DVI mode) was detected, data samples have been lost. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
10 UNDERFLOW_ IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. A TXFIFO underflow in the write mode (MPU/VSYNC/DOTCLK/DVI mode) was detected. Could produce an error in the DOTCLK / DVI modes.

Table continues on the next page...

LCDIF*x*_CTRL1*n* field descriptions (continued)

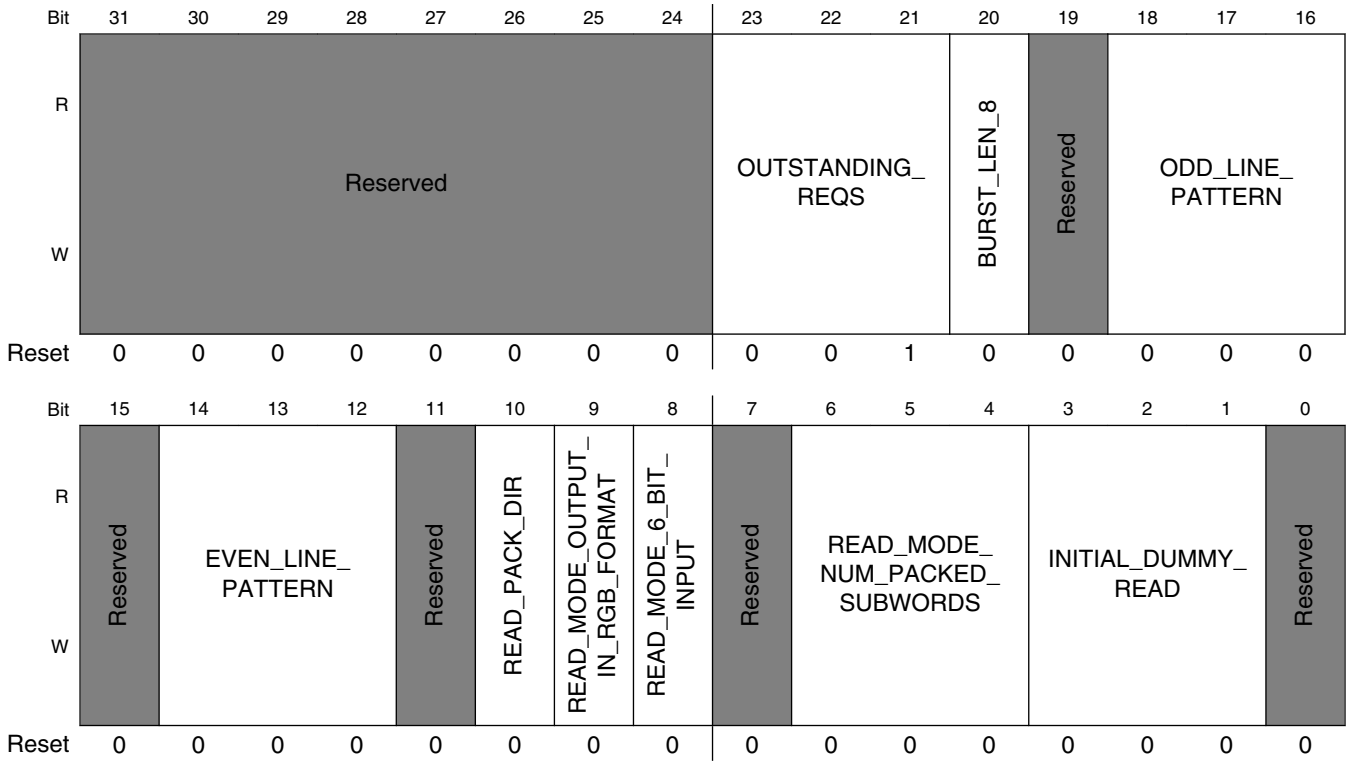
Field	Description
	0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
9 CUR_FRAME_DONE_IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. It indicates that the hardware has completed transmitting the current frame and is in the vertical blanking period in the DOTCLK/DVI modes. In the MPU and VSYNC modes, this IRQ is asserted at the end of the data transfer indicated by LCDIF_TRANSFER_COUNT register. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
8 VSYNC_EDGE_IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. It is set whenever the leading VSYNC edge is detected in the VSYNC and DOTCLK modes. In the DVI mode, it is asserted every time the block enters a new field. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
7–3 RSRVD0	This field is reserved. Reserved bits. Write as 0.
2 BUSY_ENABLE	This bit enables the use of the interface's busy signal input. This should be enabled for LCD controllers that implement a busy line (to stall the eLCDIF from sending more data until ready). Otherwise this bit should be cleared. 0x0 **BUSY_DISABLED** — The busy signal from the LCD controller will be ignored. 0x1 **BUSY_ENABLED** — Enable the use of the busy signal from the LCD controller.
1 MODE86	This bit is used to select between the 8080 and 6800 series of microprocessor modes. This bit should only be changed when RUN is 0. 0x0 **8080_MODE** — Pins LCD_WR_RWn and LCD_RD_E function as active low WR and active low RD signals respectively. 0x1 **6800_MODE** — Pins LCD_WR_RWn and LCD_RD_E function as Read/Write and active high Enable signals respectively.
0 RESET	Reset bit for the external LCD controller. This bit can be changed at any time. It CANNOT be reset by SFTRST. 0x0 **LCDRESET_LOW** — LCD_RESET output signal is low. 0x1 **LCDRESET_HIGH** — LCD_RESET output signal is high.

13.2.6.3 eLCDIF General Control2 Register (LCDIF*x*_CTRL2*n*)

The eLCDIF Control Register provides overall control of the eLCDIF block.

The eLCDIF Control2 Register provides additional programming to the eLCDIF. It implements some bits which are unlikely to change often in a particular application.

Address: Base address + 20h offset + (4d × i), where i=0d to 3d



LCDIFx_CTRL2*n* field descriptions

Field	Description
31–24 RSRVD5	This field is reserved. Reserved bits. Write as 0.
23–21 OUTSTANDING_ REQS	This bitfield indicates the maximum number of outstanding transactions that eLCDIF should request when it is acting as a bus master. Default is 2 outstanding transactions. 0x0 **REQ_1** — 0x1 **REQ_2** — 0x2 **REQ_4** — 0x3 **REQ_8** — 0x4 **REQ_16** —
20 BURST_LEN_8	By default, when the eLCDIF is in the bus master mode, it will issue AXI bursts of length 16 (except when in packed 24 bpp mode, it will issue bursts of length 15). When this bit is set to 1, the block will issue bursts of length 8 (except when in packed 24 bpp mode, it will issue bursts of length 9). Note that this bitfield is only applicable when LCDIF_MASTER is set to 1.
19 RSRVD4	This field is reserved. Reserved bits. Write as 0.
18–16 ODD_LINE_ PATTERN	This field determines the order of the RGB components of each pixel in ODD lines (line numbers 1,3,5,..). This bitfield must be 0 in DVI mode. 0x0 **RGB** — 0x1 **RBG** — 0x2 **GBR** — 0x3 **GRB** —

Table continues on the next page...

LCDIF*x*_CTRL2*n* field descriptions (continued)

Field	Description
	0x4 **BRG** — 0x5 **BGR** —
15 RSRVD3	This field is reserved. Reserved bits. Write as 0.
14–12 EVEN_LINE_ PATTERN	This field determines the order of the RGB components of each pixel in EVEN lines (line numbers 2,4,6,..). This bitfield must be 0 in DVI mode. 0x0 **RGB** — 0x1 **RBG** — 0x2 **GBR** — 0x3 **GRB** — 0x4 **BRG** — 0x5 **BGR** —
11 RSRVD2	This field is reserved. Reserved bits. Write as 0.
10 READ_PACK_DIR	The default value of 0 indicates data is stored in the little endian format. When LCD_DATABUS_WIDTH is 8-bit, this bit provides the option of rearranging the data byte-wise in the big endian format. For example, if READ_MODE_NUM_PACKED_SUBWORDS = 3 and the order of incoming data is 0x11, 0x22 and 0x33, then setting this bit to 1 will cause the data to be stored as 0x00112233 as opposed to the default 0x00332211. This operation occurs after the shifting operation done by SHIFT_NUM_BITS bitfield.
9 READ_MODE_ OUTPUT_IN_ RGB_FORMAT	Setting this bit will enable the eLCDIF to convert the incoming data to the RGB format given by WORD_LENGTH bitfield. This feature is not available when WORD_LENGTH is set to 8 bits. eLCDIF performs this operation of converting to RGB format after the endianness has been determined by the READ_PACK_DIR bitfield.
8 READ_MODE_6_ BIT_INPUT	Setting this bit to 1 indicates to eLCDIF that even though LCD_DATABUS_WIDTH is set to 8 bits, the input data is actually only 6 bits wide and exists on D5-D0.
7 RSRVD1	This field is reserved. Reserved bits. Write as 0.
6–4 READ_MODE_ NUM_PACKED_ SUBWORDS	Indicates the number of valid 8/16/18/24-bit subwords that will be packed into the 32-bit word in read mode. The subword size (8,16, 18 or 24 bits) is determined by the LCD_DATABUS_WIDTH field. The swizzle operation is performed after READ_MODE_NUM_PACKED_SUBWORDS number of subwords has been received and stored in little-endian format. For example, if LCD_DATABUS_WIDTH is set to 8-bit and data to be read back has to be stored in memory in 24-bit unpacked RGB format, set READ_MODE_NUM_PACKED_SUBWORDS to 0x3 so that each 32-bit word will contain only 3 valid bytes (RGB). Maximum value of READ_MODE_NUM_PACKED_SUBWORDS is 4 for 8-bit databus, 2 for 16-bit databus and 1 for 18/24-bit databus.
3–1 INITIAL_DUMMY_ READ	The value in this field determines the number of dummy 8/16/18/24-bit subwords that have to be read back from the LCD panel/controller. They will then not be stored in the read FIFO.
0 RSRVD0	This field is reserved. Reserved bits. Write as 0.

13.2.6.4 eLCDIF Horizontal and Vertical Valid Data Count Register (LCDIFx_TRANSFER_COUNT)

This register tells the eLCDIF how much data will be sent for this frame, or transaction. The total number of words is a product of the V_COUNT and H_COUNT fields. The word size is specified by the WORD_LENGTH field.

This register gives the dimensions of the input frame. For normal operation, but V_COUNT and H_COUNT should be non-zero.

Address: Base address + 30h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	V_COUNT																H_COUNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_TRANSFER_COUNT field descriptions

Field	Description
31–16 V_COUNT	Number of horizontal lines per frame which contain valid data. In DOTCLK mode, V_COUNT should be the same as the number of active horizontal lines in a progressive frame. In DVI mode, V_COUNT should be the number of active horizontal lines per frame, and not per field.
H_COUNT	Total valid data (pixels) in each horizontal line. The data size is given by the WORD_LENGTH. When input data is in YCbCr 4:2:2 format (YCBCR422_INPUT is 1), H_COUNT should be the number of 32-bit words that should be fetched by the block and the BYTE_PACKING_FORMAT should be 0xF. In 24-bit packed format (WORD_LENGTH=0x3, BYTE_PACKING_FORMAT=0xF), the H_COUNT must be a multiple of 4 pixels. In 16-bit packed format (WORD_LENGTH=0x0, BYTE_PACKING_FORMAT=0xF), the H_COUNT must be a multiple of 2 pixels.

13.2.6.5 LCD Interface Current Buffer Address Register (LCDIFx_CUR_BUF)

This register indicates the address of the current frame being transmitted by eLCDIF.

When the eLCDIF is behaving as a master, this address points to the address of the current frame of data being sent out via the LCDIF. When the current frame is done, the LCDIF block will assert the cur_frame_done interrupt for software to take action. The block will also copy the LCDIF_NEXT_BUF_ADDR into this bitfield so that the software can program the next frame address into the LCDIF_NEXT_BUF_ADDR bitfield. This address must always be double-word aligned.

Address: Base address + 40h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_CUR_BUF field descriptions

Field	Description
ADDR	Address of the current frame being transmitted by eLCDIF.

13.2.6.6 LCD Interface Next Buffer Address Register (LCDIFx_NEXT_BUF)

This register indicates the address of next frame that will be transmitted by eLCDIF.

When the eLCDIF is behaving as a master, this address points to the address of the next frame of data that will be sent out via the eLCDIF. It is up to the software to make sure that this register is programmed before the end of the current frame, otherwise it might result in old data going out the eLCDIF. This address must always be double-word aligned.

Address: Base address + 50h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_NEXT_BUF field descriptions

Field	Description
ADDR	Address of the next frame that will be transmitted by eLCDIF.

13.2.6.7 LCD Interface Timing Register (LCDIFx_TIMING)

The LCD interface timing register controls the various setup and hold times enforced by the LCD interface in the 6800/8080 MPU and VSYNC modes of operation.

The values used in this register are dependent on the particular LCD controller used, consult the users manual for the particular controller for required timings. Each field of the register must be non-zero, therefore the minimum value is: 0x01010101. NOTE: the timings are not automatically adjusted if the CLK_DIS_LCDIFn frequency changes—it may be necessary to adjust the timings if CLK_DIS_LCDIFn changes. NOTE: Each field in this register must be non-zero for the MPU and VSYNC modes to function. The settings in this register do not affect the DOTCLK and DVI modes.

Address: Base address + 60h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_HOLD								CMD_SETUP								DATA_HOLD								DATA_SETUP							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_TIMING field descriptions

Field	Description
31–24 CMD_HOLD	Number of CLK_DIS_LCDIFn cycles that the DCn signal is active after CEn is deasserted.
23–16 CMD_SETUP	Number of CLK_DIS_LCDIFn cycles that the DCn signal is active before CEn is asserted.
15–8 DATA_HOLD	Data bus hold time in CLK_DIS_LCDIFn cycles. Also the time that the data strobe is deasserted in a cycle
DATA_SETUP	Data bus setup time in CLK_DIS_LCDIFn cycles. Also the time that the data strobe is asserted in a cycle.

13.2.6.8 eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIFx_VDCTRL0*n*)

This register is used to control the VSYNC and DOTCLK modes of the LCDIF so as to work with different types of LCDs like moving picture displays and delta pixel displays.

This register gives general programmability to the VSYNC signal including polarity, direction, pulse width, etc.

Address: Base address + 70h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		VSYNC_OEB	ENABLE_PRESENT	VSYNC_POL	HSYNC_POL	DOTCLK_POL	ENABLE_POL	Reserved		VSYNC_PERIOD_UNIT	VSYNC_PULSE_WIDTH_UNIT	HALF_LINE	HALF_LINE_MODE	VSYNC_PULSE_WIDTH	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VSYNC_PULSE_WIDTH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_VDCTRL0*n* field descriptions

Field	Description
31–30 RSRVD2	This field is reserved. Reserved bits. Write as 0.

Table continues on the next page...

LCDIFx_VDCTRL0*n* field descriptions (continued)

Field	Description
29 VSYNC_OEB	0 means the VSYNC signal is an output, 1 means it is an input. Should be set to 0 in the DOTCLK mode. 0x0 **VSYNC_OUTPUT** — The VSYNC pin is in the output mode and the VSYNC signal has to be generated by the eLCDIF block. 0x1 **VSYNC_INPUT** — The VSYNC pin is in the input mode and the LCD controller sends the VSYNC signal to the block.
28 ENABLE_ PRESENT	Setting this bit to 1 will make the hardware generate the ENABLE signal in the DOTCLK mode, thereby making it the true RGB interface along with the remaining three signals VSYNC, HSYNC and DOTCLK.
27 VSYNC_POL	Default 0 active low during VSYNC_PULSE_WIDTH time and will be high during the rest of the VSYNC period. Set it to 1 to invert the polarity.
26 HSYNC_POL	Default 0 active low during HSYNC_PULSE_WIDTH time and will be high during the rest of the HSYNC period. Set it to 1 to invert the polarity.
25 DOTCLK_POL	Default is data launched at negative edge of DOTCLK and captured at positive edge. Set it to 1 to invert the polarity. Set it to 0 in DVI mode.
24 ENABLE_POL	Default 0 active low during valid data transfer on each horizontal line.
23–22 RSRVD1	This field is reserved. Reserved bits. Write as 0.
21 VSYNC_ PERIOD_UNIT	Default 0 for counting VSYNC_PERIOD in terms of CLK_DIS_LCDIFn cycles. Set it to 1 to count in terms of complete horizontal lines. CLK_DIS_LCDIFn cycles should be used in the VSYNC mode, while horizontal line should be used in the DOTCLK mode.
20 VSYNC_PULSE_ WIDTH_UNIT	Default 0 for counting VSYNC_PULSE_WIDTH in terms of CLK_DIS_LCDIFn cycles. Set it to 1 to count in terms of complete horizontal lines.
19 HALF_LINE	Setting this bit to 1 will make the total VSYNC period equal to the VSYNC_PERIOD field plus half the HORIZONTAL_PERIOD field (i.e. VSYNC_PERIOD field plus half horizontal line), otherwise it is just VSYNC_PERIOD. Should be only used in the DOTCLK mode, not in the VSYNC interface mode.
18 HALF_LINE_ MODE	When this bit is 0, the first field (VSYNC period) will end in half a horizontal line and the second field will begin with half a horizontal line. When this bit is 1, all fields will end with half a horizontal line, and none will begin with half a horizontal line.
VSYNC_PULSE_ WIDTH	Number of units for which VSYNC signal is active. For the DOTCLK mode, the unit is determined by the VSYNC_PULSE_WIDTH_UNIT. If the VSYNC_PULSE_WIDTH_UNIT is 0 for DOTCLK mode, VSYNC_PULSE_WIDTH must be less than HSYNC_PERIOD. For the VSYNC interface mode, it should be in terms of number of CLK_DIS_LCDIFn cycles only.

13.2.6.9 eLCDIF VSYNC Mode and Dotclk Mode Control Register1 (LCDIF*x*_VDCTRL1)

This register is used to control the VSYNC signal in the VSYNC and DOTCLK modes of the block.

This register determines the period and duty cycle of the VSYNC signal when it is generated in the block.

Address: Base address + 80h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VSYNC_PERIOD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_VDCTRL1 field descriptions

Field	Description
VSYNC_PERIOD	Total number of units between two positive or two negative edges of the VSYNC signal. If HALF_LINE is set, it is implicitly calculated to be VSYNC_PERIOD plus half HSYNC_PERIOD.

13.2.6.10 LCDIF VSYNC Mode and Dotclk Mode Control Register2 (LCDIFx_VDCTRL2)

This register is used to control the HSYNC signal in the DOTCLK mode of the block.

This register determines the period and duty cycle of the HSYNC signal when it is generated in the block.

Address: Base address + 90h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HSYNC_PULSE_WIDTH																HSYNC_PERIOD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_VDCTRL2 field descriptions

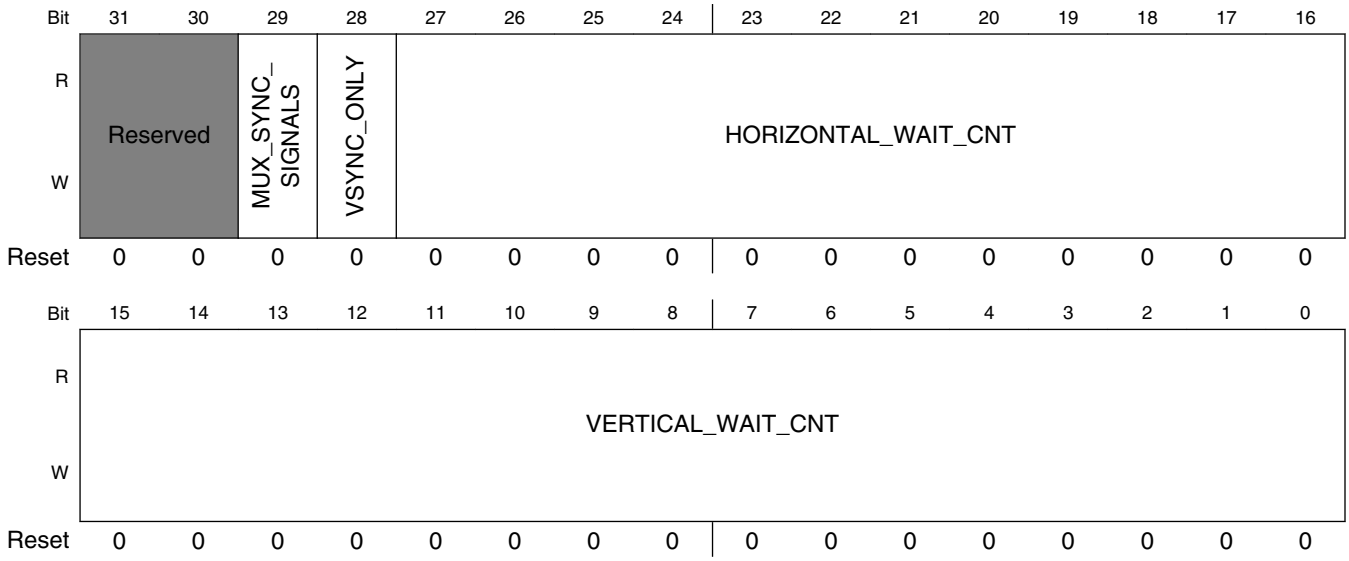
Field	Description
31–18 HSYNC_PULSE_WIDTH	Number of CLK_DIS_LCDIFn cycles for which HSYNC signal is active.
HSYNC_PERIOD	Total number of CLK_DIS_LCDIFn cycles between two positive or two negative edges of the HSYNC signal.

13.2.6.11 eLCDIF VSYNC Mode and Dotclk Mode Control Register3 (LCDIFx_VDCTRL3)

This register is used to determine the vertical and horizontal wait counts.

This register determines the back porches of HSYNC and VSYNC signals when they are generated by the block.

Address: Base address + A0h offset



LCDIFx_VDCTRL3 field descriptions

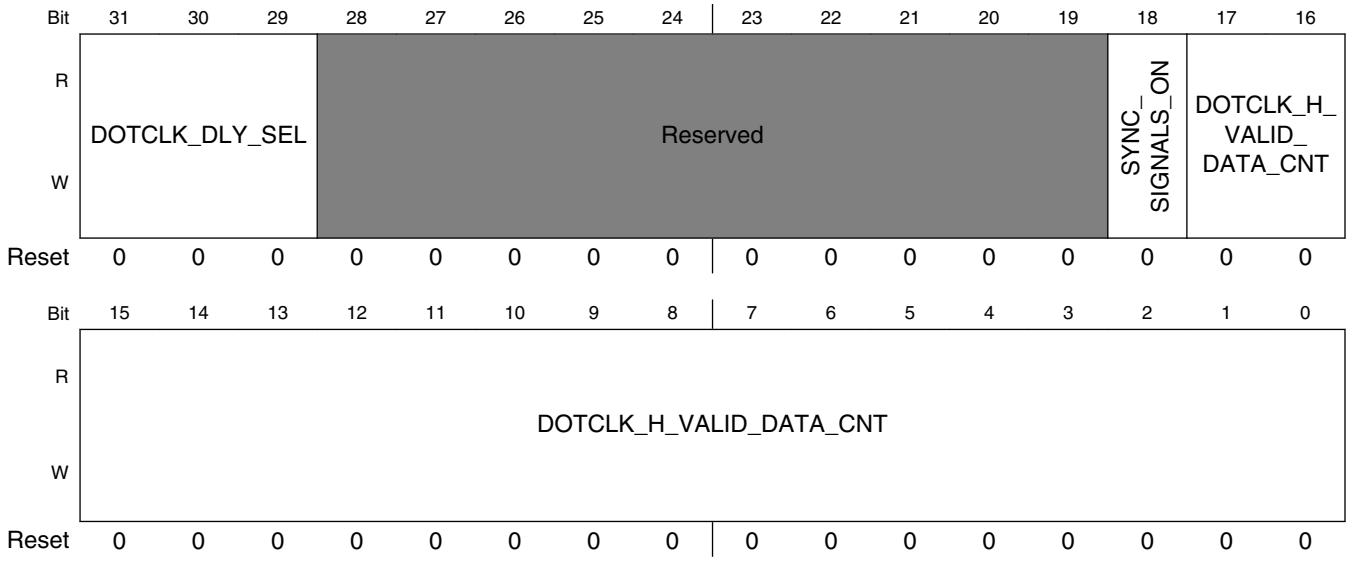
Field	Description
31–30 RSRVD0	This field is reserved. Reserved bits, write as 0.
29 MUX_SYNC_SIGNALS	When this bit is set, the eLCDIF block will internally mux HSYNC with LCD_D14, DOTCLK with LCD_D13 and ENABLE with LCD_D12, otherwise these signals will go out on separate pins. This feature can be used to maintain backward compatible with 37xx.
28 VSYNC_ONLY	This bit must be set to 1 in the VSYNC mode of operation, and 0 in the DOTCLK mode of operation.
27–16 HORIZONTAL_WAIT_CNT	In the DOTCLK mode, wait for this number of clocks from falling edge (or rising if HSYNC_POL is 1) of HSYNC signal to account for horizontal back porch plus the number of DOTCLKs before the moving picture information begins.
VERTICAL_WAIT_CNT	In the VSYNC interface mode, wait for this number of CLK_DIS_LCDIFn cycles from the falling VSYNC edge (or rising if VSYNC_POL is 1) before starting LCD transactions and is applicable only if WAIT_FOR_VSYNC_EDGE is set. Minimum is CMD_SETUP+5. In the DOTCLK mode, it accounts for the vertical back porch lines plus the number of horizontal lines before the moving picture begins. The unit for this parameter is inherently the same as the VSYNC_PERIOD_UNIT.

13.2.6.12 eLCDIF VSYNC Mode and Dotclk Mode Control Register4 (LCDIFx_VDCTRL4)

This register is used to control the DOTCLK mode of the block.

This register determines the active data in each horizontal line in the DOTCLK mode. Note that the total number of active horizontal lines in the DOTCLK mode is the same as the V_COUNT bitfield in the LCDIF_TRANSFER_COUNT register.

Address: Base address + B0h offset



LCDIFx_VDCTRL4 field descriptions

Field	Description
31–29 DOTCLK_DLY_ SEL	This bitfield selects the amount of time by which the DOTCLK signal should be delayed before coming out of the LCD_DOTCK pin. 0 = 2ns; 1=4ns;2=6ns;3=8ns. Remaining values are reserved.
28–19 RSRVD0	This field is reserved. Reserved bits, write as 0.
18 SYNC_ SIGNALS_ON	Set this field to 1 if the LCD controller requires that the VSYNC or VSYNC/HSYNC/DOTCLK control signals should be active at least one frame before the data transfers actually start and remain active at least one frame after the data transfers end. The hardware does not count the number of frames automatically. Rather, the VSYNC edge interrupt can be monitored by software to count the number of frames that have occurred after this bit is set and then the RUN bit can be set to start the data transactions. This bit must always be set in the DOTCLK mode of operation, and it must be set in the VSYNC mode of operation when VSYNC signal is an output.
DOTCLK_H_ VALID_DATA_ CNT	Total number of CLK_DIS_LCDIFn cycles on each horizontal line that carry valid data in DOTCLK mode.

13.2.6.13 Digital Video Interface Control0 Register (LCDIFx_DVICTRL0)

The Digital Video interface Control0 register provides the overall control of the Digital Video interface.

This register gives information about the horizontal active, horizontal blanking and total number of lines in the ITU-R BT.656 interface.

EXAMPLE

```
//525/60 video system
```

```
HW_LCDIF_DVICTRL0_H_ACTIVE_CNT_WR(0x5A0);//1440
HW_LCDIF_DVICTRL0_H_BLANKING_CNT_WR(0x106);//262
//625/50 video system
HW_LCDIF_DVICTRL0_H_ACTIVE_CNT_WR(0x5A0);//1440
HW_LCDIF_DVICTRL0_H_BLANKING_CNT_WR(0x112);//274
```

Address: Base address + C0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				H_ACTIVE_CNT												Reserved				H_BLANKING_CNT											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_DVICTRL0 field descriptions

Field	Description
31–28 RSRVD1	This field is reserved. Reserved bits, write as 0.
27–16 H_ACTIVE_CNT	Number of active video samples to be transmitted. (Mostly will be 1440 for both PAL and NTSC). Must always be a multiple of 4.
15–12 RSRVD0	This field is reserved. Reserved bits, write as 0.
H_BLANKING_ CNT	Number of blanking samples to be inserted between EAV and SAV during horizontal blanking interval.

13.2.6.14 Digital Video Interface Control1 Register (LCDIFx_DVICTRL1)

The Digital Video interface Control1 register provides the overall control of the Digital Video interface.

This register contains information about the Field1 start and end, and the Field2 start in the ITU-R BT.656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL1_F1_START_LINE_WR(0x4);//4
HW_LCDIF_DVICTRL1_F1_END_LINE_WR(0x109);//265
HW_LCDIF_DVICTRL1_F2_START_LINE_WR(0x10A);//266
//625/50 video system
HW_LCDIF_DVICTRL1_F1_START_LINE_WR(0x1);//1
HW_LCDIF_DVICTRL1_F1_END_LINE_WR(0x138);//312
HW_LCDIF_DVICTRL1_F2_START_LINE_WR(0x139);//313
```

Address: Base address + D0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		F1_START_LINE										F1_END_LINE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	F1_END_LINE								F2_START_LINE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_DVICTRL1 field descriptions

Field	Description
31–30 RSRVD0	This field is reserved. Reserved bits, write as 0.
29–20 F1_START_LINE	Vertical line number from which Field 1 begins.
19–10 F1_END_LINE	Vertical line number at which Field1 ends.
F2_START_LINE	Vertical line number from which Field 2 begins.

13.2.6.15 Digital Video Interface Control2 Register (LCDIFx_DVICTRL2)

The Digital Video interface Control2 register provides the overall control of the Digital Video interface.

This register contains information about the Field2 end, and the Vertical Blanking1 interval in the ITU-R BT.656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL2_F2_END_LINE_WR(0x3);//3
HW_LCDIF_DVICTRL2_V1_BLANK_START_LINE_WR(0x108);//264
HW_LCDIF_DVICTRL2_V1_BLANK_END_LINE_WR(0x11A);//282
//625/50 video system
HW_LCDIF_DVICTRL2_F2_END_LINE_WR(0x271);//625
HW_LCDIF_DVICTRL2_V1_BLANK_START_LINE_WR(0x137);//311
HW_LCDIF_DVICTRL2_V1_BLANK_END_LINE_WR(0x14F);//335
```

Address: Base address + E0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		F2_END_LINE										V1_BLANK_START_LINE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	V1_BLANK_START_LINE								V1_BLANK_END_LINE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_DVICTRL2 field descriptions

Field	Description
31–30 RSRVD0	This field is reserved. Reserved bits, write as 0.
29–20 F2_END_LINE	Vertical line number at which Field 2 ends.
19–10 V1_BLANK_ START_LINE	Vertical line number towards the end of Field1 where first Vertical Blanking interval starts.
V1_BLANK_ END_LINE	Vertical line number in the beginning part of Field2 where first Vertical Blanking interval ends.

13.2.6.16 Digital Video Interface Control3 Register (LCDIFx_DVICTRL3)

The Digital Video interface Control3 register provides the overall control of the Digital Video interface.

This register contains information about the Vertical Blanking2 interval in the ITU-R BT. 656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL3_V2_BLANK_START_LINE_WR(0x1);//1
HW_LCDIF_DVICTRL3_V2_BLANK_END_LINE_WR(0x13);//19
HW_LCDIF_DVICTRL0_V_LINES_CNT_WR(0x20D);//525
//625/50 video system
HW_LCDIF_DVICTRL3_V2_BLANK_START_LINE_WR(0x270);//624
HW_LCDIF_DVICTRL3_V2_BLANK_END_LINE_WR(0x16);//22
HW_LCDIF_DVICTRL0_V_LINES_CNT_WR(0x271);//625
```

Address: Base address + F0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		V2_BLANK_START_LINE										V2_BLANK_END_LINE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	V2_BLANK_END_LINE						V_LINES_CNT									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_DVICTRL3 field descriptions

Field	Description
31–30 RSRVD0	This field is reserved. Reserved bits, write as 0.

Table continues on the next page...

LCDIFx_DVICTRL3 field descriptions (continued)

Field	Description
29–20 V2_BLANK_ START_LINE	Vertical line number towards the end of Field2 where second Vertical Blanking interval starts.
19–10 V2_BLANK_ END_LINE	Vertical line number in the beginning part of Field1 where second Vertical Blanking interval ends.
V_LINES_CNT	Total number of vertical lines per frame (generally 525 or 625)

13.2.6.17 Digital Video Interface Control4 Register (LCDIFx_DVICTRL4)

The Digital Video interface Control4 register provides the overall control of the Digital Video interface.

This register is used to add side borders to the output if the input frame width is less than 720 pixels.

EXAMPLE

```
                //If input frame has only 640 pixels per line, but output is supposed to have
720 pixels per line.
                HW_LCDIF_DVICTRL4_H_FILL_CNT_WR(0x50);//80
                HW_LCDIF_DVICTRL4_Y_FILL_VALUE_WR(0x10);//16
                HW_LCDIF_DVICTRL4_CB_FILL_VALUE_WR(0x80);//128
                HW_LCDIF_DVICTRL4_CR_FILL_VALUE_WR(0x80);//128
```

Address: Base address + 100h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Y_FILL_VALUE								CB_FILL_VALUE								CR_FILL_VALUE								H_FILL_CNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_DVICTRL4 field descriptions

Field	Description
31–24 Y_FILL_VALUE	Value of Y component of filler data
23–16 CB_FILL_VALUE	Value of CB component of filler data
15–8 CR_FILL_VALUE	Value of CR component of filler data.
H_FILL_CNT	Number of active video samples that have to be filled with the filler data in the front and back portions of the active horizontal interval. Must be a multiple of 4. This field will have to be programmed if the input frame has less than 720 pixels per line.

13.2.6.18 RGB to YCbCr 4:2:2 CSC Coefficient0 Register (LCDIFx_CSC_COEFF0)

LCDIF_CSC_COEFF0 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF0_C0_WR(0x41);//0.257x256=65
HW_LCDIF_CSC_COEFF0_CSC_SUBSAMPLE_FILTER_WR(0x3);
```

Address: Base address + 110h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								C0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														CSC_ SUBSAMPL E_FILTER	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_CSC_COEFF0 field descriptions

Field	Description
31–26 RSRVD1	This field is reserved. Reserved bits, write as 0.
25–16 C0	Two's complement red multiplier coefficient for Y
15–2 RSRVD0	This field is reserved. Reserved bits, write as 0.
CSC_ SUBSAMPLE_ FILTER	This register describes the filtering and subsampling scheme to be performed on the chroma components in order to convert from YCbCr 4:4:4 to YCbCr 4:2:2 space. Note that the following descriptions apply individually to Cb and Cr. 0x0 **SAMPLE_AND_HOLD** — No filtering, simply keep every chroma value for samples numbered 2n and discard chroma values associated with all samples numbered 2n+1. 0x1 **RSRVD** — Reserved 0x2 **INTERSTITIAL** — Chroma samples numbered 2n and 2n+1 are averaged (weights 1/2, 1/2) and that chroma value replaces the two chroma values at 2n and 2n+1. This chroma now exists horizontally halfway between the two luma samples. 0x3 **COSITED** — Chroma samples numbered 2n-1, 2n, and 2n+1 are averaged (weights 1/4, 1/2, 1/4) and that chroma value exists at the same site as the luma sample numbered 2n and the chroma samples at 2n+1 are discarded.

13.2.6.19 RGB to YCbCr 4:2:2 CSC Coefficient1 Register (LCDIFx_CSC_COEFF1)

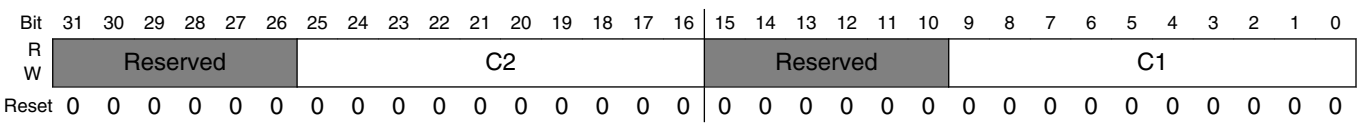
LCDIF_CSC_COEFF1 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF1_C1_WR(0x81);//0.504x256=129
HW_LCDIF_CSC_COEFF1_C2_WR(0x19);//0.098x256=25
```

Address: Base address + 120h offset



LCDIFx_CSC_COEFF1 field descriptions

Field	Description
31–26 RSRVD1	This field is reserved. Reserved bits, write as 0.
25–16 C2	Two's complement blue multiplier coefficient for Y
15–10 RSRVD0	This field is reserved. Reserved bits, write as 0.
C1	Two's complement green multiplier coefficient for Y

13.2.6.20 RGB to YCbCr 4:2:2 CSC Coefficent2 Register (LCDIFx_CSC_COEFF2)

LCDIF_CSC_COEFF2 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF2_C3_WR(0x3DB);//-0.148x256=-37
HW_LCDIF_CSC_COEFF2_C4_WR(0x3B6);//-0.291x256=-74
```

Address: Base address + 130h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						C4										Reserved						C3									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_CSC_COEFF2 field descriptions

Field	Description
31–26 RSRVD1	This field is reserved. Reserved bits, write as 0.
25–16 C4	Two's complement green multiplier coefficient for Cb
15–10 RSRVD0	This field is reserved. Reserved bits, write as 0.
C3	Two's complement red multiplier coefficient for Cb

13.2.6.21 RGB to YCbCr 4:2:2 CSC Coefficient3 Register (LCDIFx_CSC_COEFF3)

LCDIF_CSC_COEFF3 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF3_C5_WR(0x70);//0.439x256=112
HW_LCDIF_CSC_COEFF3_C6_WR(0x70);//0.439x256=112
```

Address: Base address + 140h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						C6										Reserved						C5									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_CSC_COEFF3 field descriptions

Field	Description
31–26 RSRVD1	This field is reserved. Reserved bits, write as 0.

Table continues on the next page...

LCDIFx_CSC_COEFF3 field descriptions (continued)

Field	Description
25–16 C6	Two's complement red multiplier coefficient for Cr
15–10 RSRVD0	This field is reserved. Reserved bits, write as 0.
C5	Two's complement blue multiplier coefficient for Cb

13.2.6.22 RGB to YCbCr 4:2:2 CSC Coefficient4 Register (LCDIFx_CSC_COEFF4)

LCDIF_CSC_COEFF4 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF4_C7_WR(0x3A2);//-0.368x256=-94
HW_LCDIF_CSC_COEFF4_C8_WR(0x3EE);//-0.071x256=-18
```

Address: Base address + 150h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						C8										Reserved						C7									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_CSC_COEFF4 field descriptions

Field	Description
31–26 RSRVD1	This field is reserved. Reserved bits, write as 0.
25–16 C8	Two's complement blue multiplier coefficient for Cr
15–10 RSRVD0	This field is reserved. Reserved bits, write as 0.
C7	Two's complement green multiplier coefficient for Cr

13.2.6.23 RGB to YCbCr 4:2:2 CSC Offset Register (LCDIFx_CSC_OFFSET)

LCDIF_CSC_ register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

Address: Base address + 160h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved							CBCR_OFFSET									Reserved							Y_OFFSET								
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

LCDIFx_CSC_OFFSET field descriptions

Field	Description
31–25 RSRVD1	This field is reserved. Reserved bits, write as 0.
24–16 CBCR_OFFSET	Two's complement offset for the Cb and Cr components
15–9 RSRVD0	This field is reserved. Reserved bits, write as 0.
Y_OFFSET	Two's complement offset for the Y component

13.2.6.24 RGB to YCbCr 4:2:2 CSC Limit Register (LCDIFx_CSC_LIMIT)

LCDIF_CSC_CTRL0 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC. Note that the values in this register are unsigned.

EXAMPLE

```
HW_LCDIF_CSC_LIMIT_CBCR_MIN_WR(0x10);//16
HW_LCDIF_CSC_LIMIT_CBCR_MAX_WR(0xF0);//240
HW_LCDIF_CSC_LIMIT_Y_MIN_WR(0x10);//16
HW_LCDIF_CSC_LIMIT_Y_MAX_WR(0xEB);//235
```

Address: Base address + 170h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CBCR_MIN								CBCR_MAX								Y_MIN								Y_MAX							
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

LCDIFx_CSC_LIMIT field descriptions

Field	Description
31–24 CBCR_MIN	Lower limit of Cb and Cr after RGB to 4:2:2 YCbCr conversion
23–16 CBCR_MAX	Upper limit of Cb and Cr after RGB to 4:2:2 YCbCr conversion
15–8 Y_MIN	Lower limit of Y after RGB to 4:2:2 YCbCr conversion
Y_MAX	Upper limit of Y after RGB to 4:2:2 YCbCr conversion

13.2.6.25 LCD Interface Data Register (LCDIFx_DATA)

This register is used to transfer data using the PIO interface mode of operation. In MPU mode, data written to this register will be transferred out to the display device. When receiving data from the display, data is read from this register using PIO operations. During write operations, data can be written to this register (from the processor's perspective) as bytes, half-words (16 bits), or words (32 bits) as desired.

This register holds the 32-bit word written by the ARM platform into LCDIF. This data then gets sent out by the block across the interface.

Address: Base address + 180h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA_THREE								DATA_TWO								DATA_ONE								DATA_ZERO							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_DATA field descriptions

Field	Description
31–24 DATA_THREE	Byte 3 (most significant byte) of data written to LCDIF.
23–16 DATA_TWO	Byte 2 of data written to eLCDIF.
15–8 DATA_ONE	Byte 1 of data written to eLCDIF.
DATA_ZERO	Byte 0 (least significant byte) of data written to eLCDIF.

13.2.6.26 Bus Master Error Status Register (LCDIFx_BM_ERROR_STAT)

This register reflects the virtual address at which the AXI master received an error response from the slave.

When the BM_ERROR_IRQ is asserted, the address of the bus error is updated in the register.

Address: Base address + 190h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_BM_ERROR_STAT field descriptions

Field	Description
ADDR	Virtual address at which bus master error occurred.

13.2.6.27 CRC Status Register (LCDIFx_CRC_STAT)

This register reflects the CRC value of each frame sent out by eLCDIF. The CRC is done on the final output bus, so the value will be dependent on the LCD_DATABUS_WIDTH bitfield even if the input data is the same.

This register will be updated when the CUR_FRAME_DONE_IRQ is asserted. In the case of DVI mode, the CRC is calculated for the entire frame, not separately for each field in the frame.

Address: Base address + 1A0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CRC_VALUE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_CRC_STAT field descriptions

Field	Description
CRC_VALUE	Calculated CRC value.

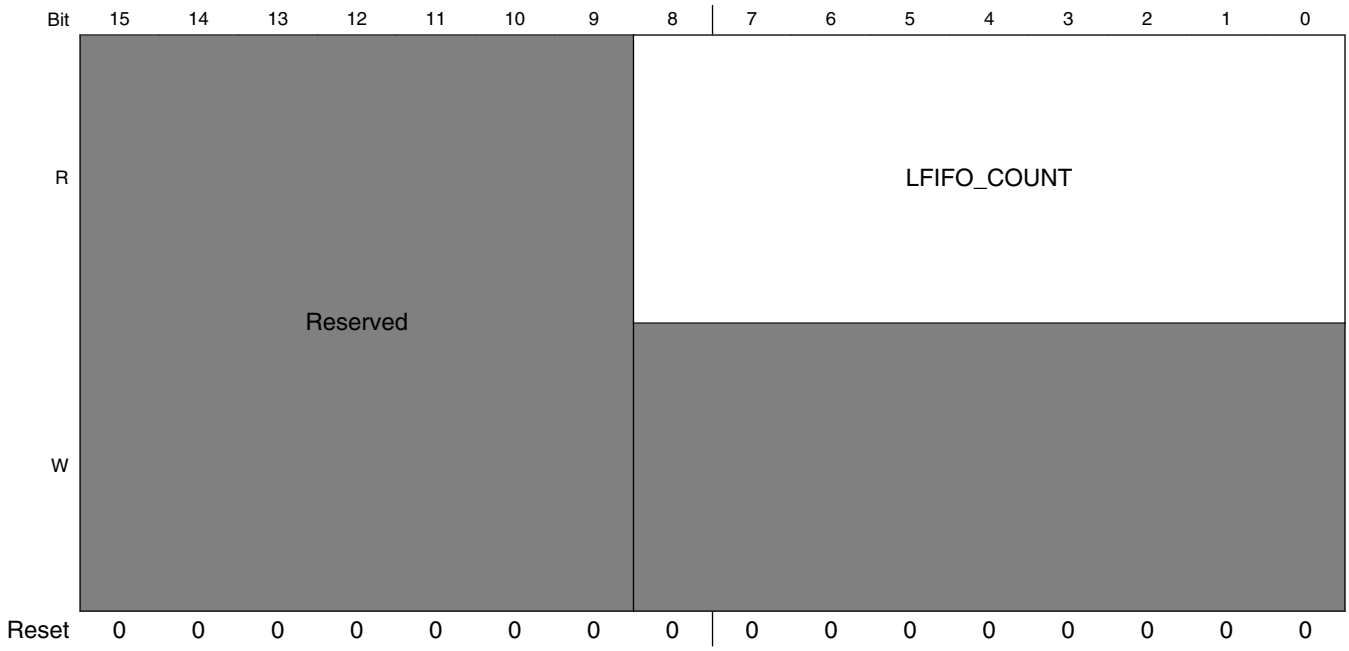
13.2.6.28 LCD Interface Status Register (LCDIFx_STAT)

The LCD interface status register can be used to check the current status of the eLCDIF block.

The LCD interface status register that contains read only views of some parameters or current state of the block.

Address: Base address + 1B0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRESENT	Reserved	LFIFO_FULL	LFIFO_EMPTY	TXFIFO_FULL	TXFIFO_EMPTY	BUSY	DVI_CURRENT_FIELD	Reserved							
W																
Reset	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0



LCDIFx_STAT field descriptions

Field	Description
31 PRESENT	0: eLCDIF not present on this product 1: eLCDIF is present.
30 -	This field is reserved. Reserved.
29 LFIFO_FULL	Read only view of the signal that indicates that LCD read datapath FIFO is full, will be generally used in the write mode of the LCD interface.
28 LFIFO_EMPTY	Read only view of the signal that indicates that LCD read dapatath FIFO is empty, will be generally used in the read mode of the LCD interface.
27 TXFIFO_FULL	Read only view of the signal that indicates that LCD write datapath FIFO is full, will be generally used in the write mode of the LCD interface.
26 TXFIFO_EMPTY	Read only view of the signal that indicates that LCD write dapatath FIFO is empty, will be generally used in the read mode of the LCD interface.
25 BUSY	Read only view of the input busy signal from the external LCD controller.
24 DVI_CURRENT_FIELD	Read only view of the current field being transmitted. DVI_CURRENT_FIELD = 0 means field 1. DVI_CURRENT_FIELD = 1 means field 2.
23–9 RSRVD0	This field is reserved. Reserved bits. Write as 0.
LFIFO_COUNT	Read only view of the current count in Latency buffer (LFIFO).

13.2.6.29 LCD Interface Version Register (LCDIFx_VERSION)

The LCD interface version register can be used to read the version of the eLCDIF IP being used in this SoC.

The LCD interface debug register is for diagnostic use only.

Address: Base address + 1C0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_VERSION field descriptions

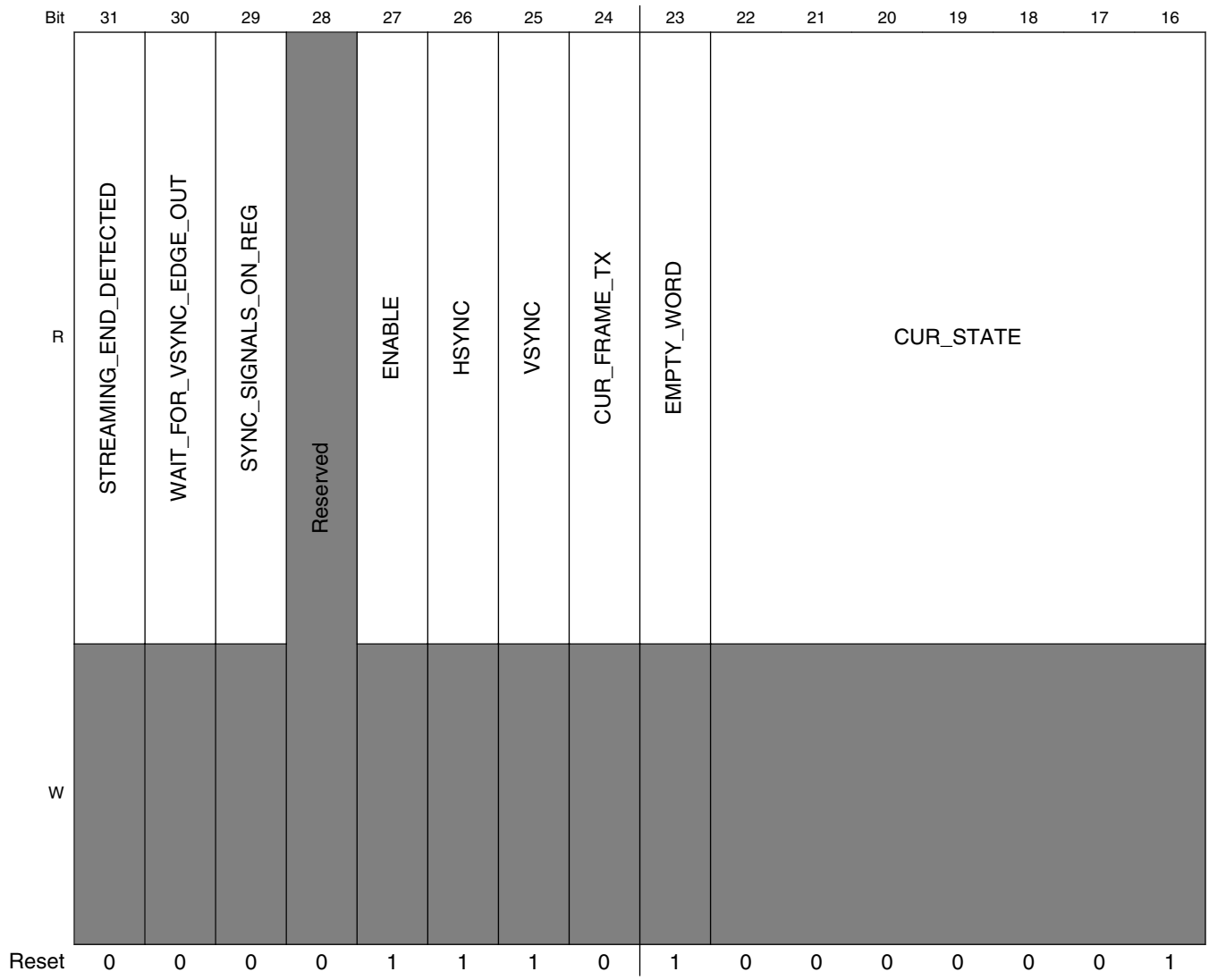
Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of RTL version.
STEP	Fixed read-only value reflecting the stepping of RTL version.

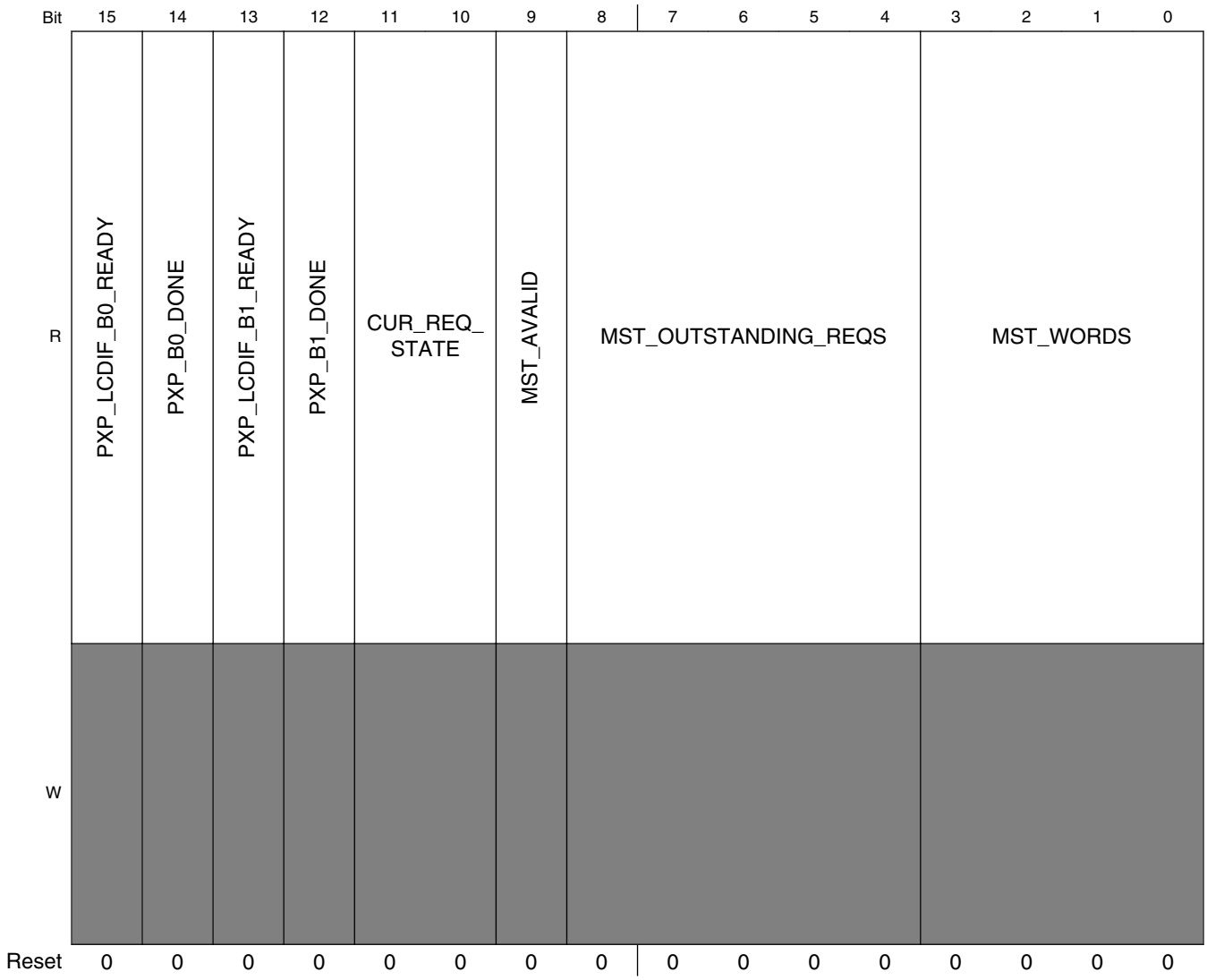
13.2.6.30 LCD Interface Debug0 Register (LCDIFx_DEBUG0)

The LCD interface debug0 register provides a diagnostic view of the state machine and other useful internal signals.

The LCD interface debug register is for diagnostic use only.

Address: Base address + 1D0h offset





LCDIFx_DEBUG0 field descriptions

Field	Description
31 STREAMING_ END_DETECTED	Read only view of the DOTCLK_MODE or DVI_MODE bit going from 1 to 0.
30 WAIT_FOR_ VSYNC_EDGE_ OUT	Read only view of WAIT_FOR_VSYNC_EDGE bit in the VSYNC mode after it comes out of the TXFIFO.
29 SYNC_SIGNALS_ ON_REG	Read only view of internal sync_signals_on_reg signal.
28 -	This field is reserved. Reserved.
27 ENABLE	Read only view of ENABLE signal.

Table continues on the next page...

LCDIFx_DEBUG0 field descriptions (continued)

Field	Description
26 HSYNC	Read only view of HSYNC signal.
25 VSYNC	Read only view of VSYNC signal.
24 CUR_FRAME_TX	This bit is 1 for the time the current frame is being transmitted in the VSYNC mode. Useful for VSYNC mode debug.
23 EMPTY_WORD	Indicates that the current word is empty.
22–16 CUR_STATE	Read only view of the current state machine state in the current mode of operation.
15 PXP_LCDIF_B0_READY	Buffer0 ready signal issued by PXP.
14 PXP_B0_DONE	Buffer0 done signal issued by eLCDIF.
13 PXP_LCDIF_B1_READY	Buffer1 ready signal issued by PXP.
12 PXP_B1_DONE	Buffer1 done signal issued by eLCDIF.
11–10 CUR_REQ_STATE	Read only view of the request state machine.
9 MST_AVALID	Read only view of the mst_avalid signal issued by the AXI bus master.
8–4 MST_OUTSTANDING_REQS	Read only view of the current outstanding requests issued by the AXI bus master.
MST_WORDS	Read only view of the current bursts issued by the AXI bus master.

13.2.6.31 LCD Interface Debug1 Register (LCDIFx_DEBUG1)

The LCD interface debug1 register provides a diagnostic view of the state machine and other useful internal signals.

The LCD interface debug register is for diagnostic use only.

Address: Base address + 1E0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	H_DATA_COUNT																V_DATA_COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_DEBUG1 field descriptions

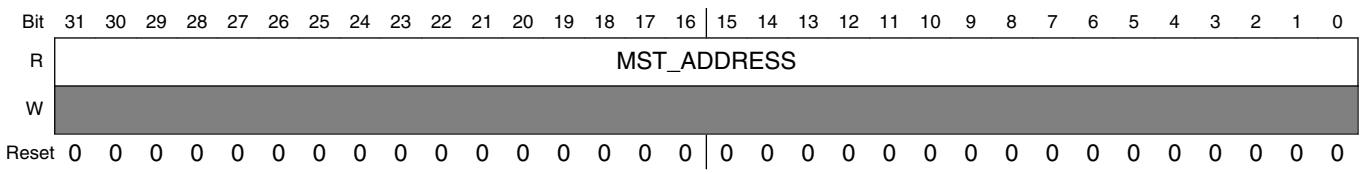
Field	Description
31–16 H_DATA_ COUNT	Read only view of the current state of the horizontal data counter.
V_DATA_ COUNT	Read only view of the current state of the vertical data counter.

13.2.6.32 LCD Interface Debug2 Register (LCDIFx_DEBUG2)

The LCD interface debug2 register provides a diagnostic view of the state machine and other useful internal signals.

The LCD interface debug register is for diagnostic use only.

Address: Base address + 1F0h offset



LCDIFx_DEBUG2 field descriptions

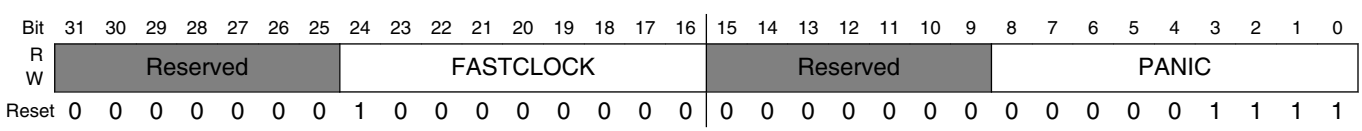
Field	Description
MST_ADDRESS	Read only view of the current address issued by the AXI bus master.

13.2.6.33 eLCDIF Threshold Register (LCDIFx_THRES)

This register is used to activate control signals when the number of pixels reaches the programmed threshold. These control signals, in turn, can be used to manipulate access priority or dynamically change the input clock frequency to meet the required pixel throughput.

Memory request priority threshold register.

Address: Base address + 200h offset



LCDIFx_THRES field descriptions

Field	Description
31–25 RSRVD2	This field is reserved. Reserved bits. Write as 0.
24–16 FASTCLOCK	This value should be set to a value of pixels, from 0 to 511. When the number of pixels in the input pixel FIFO is LESS than this value, the fast clock control output will be raised. This signal can be used to reduce the system bus clock frequency to save power during horizontal or vertical blanking intervals. This value should also be programmed to a value that is greater than the "PANIC" threshold value. This will allow a faster clock to recover the number of pixels in the FIFO before a "panic" level is encountered.
15–9 RSRVD1	This field is reserved. Reserved bits. Write as 0.
PANIC	This value should be set to a value of pixels from 0 to 511. When the number of pixels in the input pixel FIFO is less than this value, the internal panic control output will be raised. This signal can be used to raise the access eLCDIF's access priority.

13.2.6.34 eLCDIF AS Buffer Control Register (LCDIFx_AS_CTRL)

The Alpha Surface Parameter register provides additional controls for AS.

Address: Base address + 210h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CSI_VSYNC_ENABLE	CSI_VSYNC_POL	CSI_VSYNC_MODE	CSI_SYNC_ON_IRQ_EN	CSI_SYNC_ON_IRQ	RVDS1			PS_DISABLE	INPUT_DATA_SWIZZLE		ALPHA_INVERT	ROP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ALPHA								FORMAT				ENABLE_COLORKEY	ALPHA_CTRL		AS_ENABLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_AS_CTRL field descriptions

Field	Description
31 CSI_VSYNC_ENABLE	When this bit is set by software, the LCDIF work as sync mode with CSI input.
30 CSI_VSYNC_POL	Default 0 active low during VSYNC_PULSE_WIDTH time and will be high during the rest of the VSYNC period. Set it to 1 to invert the polarity.

Table continues on the next page...

LCDIFx_AS_CTRL field descriptions (continued)

Field	Description
29 CSI_VSYNC_MODE	this bit is set by software to decide which vsync generate mode. LCDIF vsync generate by internal counter when set to 0, LCDIF vsync delayed by each csi_vsync_in when set to 1; INT_SYNC_MODE = 0x0 LCDIF vsync generate by internal counter. EXT_SYNC_MODE = 0x1 LCDIF vsync delayed by each csi_vsync_in.
28 CSI_SYNC_ON_IRQ_EN	This bit is set to enable an interrupt when LCDIF lock with CSI vsync input.
27 CSI_SYNC_ON_IRQ	this bit is set by software to decide which vsync generate mode. LCDIF vsync generate by internal counter when set to 0, LCDIF vsync delayed by each csi_vsync_in when set to 1; INT_SYNC_MODE = 0x0 LCDIF vsync generate by internal counter. EXT_SYNC_MODE = 0x1 LCDIF vsync delayed by each csi_vsync_in.
26–24 RVDS1	Reserved, always set to zero.
23 PS_DISABLE	When this bit is set by software, the LCDIF will disable PS buffer data.
22–21 INPUT_DATA_SWIZZLE	This field specifies how to swap the bytes either in the HW_LCDIF_DATA register or those fetched by the AXI master part of LCDIF. The swizzle function is independent of the WORD_LENGTH bit. See the explanation of the HW_LCDIF_DATA below for names and definitions of data register fields. The supported swizzle configurations are: NO_SWAP = 0x0 No byte swapping.(Little endian) LITTLE_ENDIAN = 0x0 Little Endian byte ordering (same as NO_SWAP). BIG_ENDIAN_SWAP = 0x1 Big Endian swap (swap bytes 0, 3 and 1, 2). SWAP_ALL_BYTES = 0x1 Swizzle all bytes, swap bytes 0, 3 and 1, 2 (aka Big Endian). HWD_SWAP = 0x2 Swap half-words. HWD_BYTE_SWAP = 0x3 Swap bytes within each half-word.
20 ALPHA_INVERT	Setting this bit to logic 0 will not alter the alpha value. A logic 1 will invert the alpha value and apply (1-alpha) for image composition.
19–16 ROP	Indicates a raster operation to perform when enabled. Raster operations are enabled through the ALPHA_CTRL field. MASKAS = 0x0 AS AND PS MASKNOTAS = 0x1 nAS AND PS MASKASNOT = 0x2 AS AND nPS MERGEAS = 0x3 AS OR PS MERGENOTAS = 0x4 nAS OR PS MERGEASNOT = 0x5 AS OR nPS NOTCOPYAS = 0x6 nAS NOT = 0x7 nPS NOTMASKAS = 0x8 AS NAND PS NOTMERGEAS = 0x9 AS NOR PS XORAS = 0xA AS XOR PS NOTXORAS = 0xB AS XNOR PS

Table continues on the next page...

LCDIFx_AS_CTRL field descriptions (continued)

Field	Description
15–8 ALPHA	Alpha modifier used when the ALPHA_MULTIPLY or ALPHA_OVERRIDE values are programmed in REG_AS_CTRL[ALPHA_CTRL]. The output alpha value will either be replaced (ALPHA_OVERRIDE) or scaled (ALPHA_MULTIPLY) when selected.
7–4 FORMAT	Indicates the input buffer format for AS. ARGB8888 = 0x0 32-bit pixels with alpha RGB888 = 0x4 32-bit pixels without alpha (unpacked 24-bit format) ARGB1555 = 0x8 16-bit pixels with alpha ARGB4444 = 0x9 16-bit pixels with alpha RGB555 = 0xC 16-bit pixels without alpha RGB444 = 0xD 16-bit pixels without alpha RGB565 = 0xE 16-bit pixels without alpha
3 ENABLE_ COLORKEY	Indicates that colorkey functionality is enabled for this alpha surface. Pixels found in the alpha surface colorkey range will be displayed as transparent (the PS pixel will be used).
2–1 ALPHA_CTRL	Determines how the alpha value is constructed for this alpha surface. Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. Embedded = 0x0 Indicates that the AS pixel alpha value will be used to blend the AS with PS. The ALPHA field is ignored. Override = 0x1 Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. Multiply = 0x2 Indicates that the value in the ALPHA field should be used to scale all pixel alpha values. Each pixel alpha is multiplied by the value in the ALPHA field. ROPs = 0x3 Enable ROPs. The ROP field indicates an operation to be performed on the alpha surface and PS pixels.
0 AS_ENABLE	When this bit is set by software, the LCDIF will start fetching AS buffer data in bus master mode and combine it with another buffer.

13.2.6.35 Alpha Surface Buffer Pointer (LCDIFx_AS_BUF)

This register is used to indicate the base address of the AS buffer.

Address: Base address + 220h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_AS_BUF field descriptions

Field	Description
ADDR	Address pointer for the alpha surface 0 buffer.

13.2.6.36 LCDIFx_AS_NEXT_BUF

When the LCDIF is behaving as a master, this address points to the address of the next frame of data that will be sent out via the LCDIF. It is upto the software to make sure that this register is programmed before the end of the current frame, otherwise it might result in old data going out the LCDIF. This address must always be double-word aligned.

Address: Base address + 230h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_AS_NEXT_BUF field descriptions

Field	Description
ADDR	Address of the next frame that will be transmitted by eLCDIF.

13.2.6.37 eLCDIF Overlay Color Key Low (LCDIFx_AS_CLRKEYLOW)

If a pixel in the current overlay image with a color that falls in the range from the ASCOLORKEYLOW to ASCOLORKEYHIGH range, it will use the PS pixel value for that location. Colorkey operations are higher priority than alpha or ROP operations.

Address: Base address + 240h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD1								PIXEL																							
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

LCDIFx_AS_CLRKEYLOW field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	Low range of RGB color key applied to AS buffer

13.2.6.38 eLCDIF Overlay Color Key High (LCDIFx_AS_CLRKEYHIGH)

If a pixel in the current overlay image with a color that falls in the range from the ASCOLORKEYLOW to ASCOLORKEYHIGH range, it will use the PS pixel value for that location. Colorkey operations are higher priority than alpha or ROP operations.

Address: Base address + 250h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD1								PIXEL																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_AS_CLRKEYHIGH field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	High range of RGB color key applied to AS buffer

13.2.6.39 LCD working insync mode with CSI for VSYNC delay (LCDIFx_SYNC_DELAY)

The LCDIF DOTCLK mode VSYNC will delay from CSI_VSYNC as
(V_COUNT_DELAY * HSYNC_PERIOD + H_COUNT_DELAY) PIXCLK cycles

Address: Base address + 260h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	V_COUNT_DELAY																H_COUNT_DELAY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

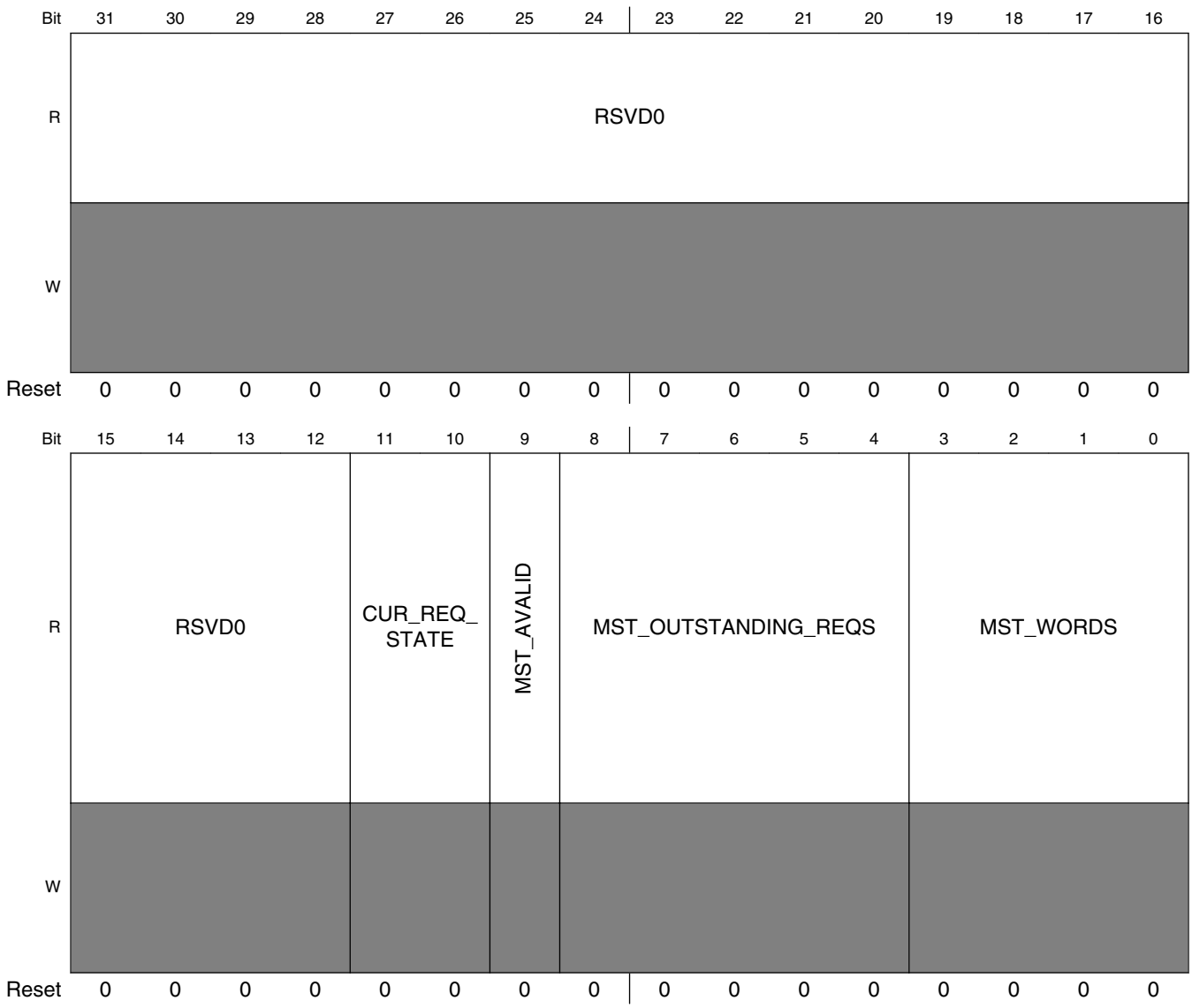
LCDIFx_SYNC_DELAY field descriptions

Field	Description
31–16 V_COUNT_ DELAY	LCDIF VSYNC delayed counter for CSI_VSYNC.
H_COUNT_ DELAY	LCDIF VSYNC delayed counter for CSI_VSYNC.

13.2.6.40 eLCDIF Interface Debug3 Register (LCDIFx_DEBUG3)

The LCD interface debug register is for diagnostic use only.

Address: Base address + 270h offset



LCDIFx_DEBUG3 field descriptions

Field	Description
31–12 RSVD0	Reserved bits, write as 0.

Table continues on the next page...

LCDIFx_DEBUG3 field descriptions (continued)

Field	Description
11–10 CUR_REQ_ STATE	Read only view of the request state machine
9 MST_AVALID	Read only view of the mst_avalid signal issued by the AXI bus master
8–4 MST_ OUTSTANDING_ REQS	Read only view of the current outstanding requests issued by the AXI bus master
MST_WORDS	Read only view of the current bursts issued by the AXI bus master.

13.2.6.41 LCD Interface Debug4 (LCDIFx_DEBUG4)

The LCD interface debug register is for diagnostic use only.

Address: Base address + 280h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	H_DATA_COUNT																V_DATA_COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_DEBUG4 field descriptions

Field	Description
31–16 H_DATA_ COUNT	Read only view of the current AS state of the horizontal data counter.
V_DATA_ COUNT	Read only view of the current AS state of the vertical data counter.

13.2.6.42 LCD Interface Debug5 (LCDIFx_DEBUG5)

The LCD interface debug register is for diagnostic use only.

Address: Base address + 290h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MST_ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIFx_DEBUG5 field descriptions

Field	Description
MST_ADDRESS	Read only view of the AS channel address issued by the AXI bus master.

13.3 CMOS Sensor Interface (CSI)

13.3.1 Overview

This chapter presents the CMOS Sensor Interface (CSI) architecture, operation principles, and programming model.

The CSI enables the chip to connect directly to external CMOS image sensors. CMOS image sensors are separated into two classes, dumb and smart. Dumb sensors are those that support only traditional sensor timing (Vertical SYNC and Horizontal SYNC) and output only Bayer and statistics data, while smart sensors support CCIR656 video decoder formats and perform additional processing of the image (for example, image compression, image pre-filtering, and various data output formats).

The capabilities of the CSI include:

- Configurable interface logic to support most commonly available CMOS sensors.
- Support for CCIR656 video interface as well as traditional sensor interface.
- 8-bit / 16-bit / 24-bit data port for YCbCr, YUV, or RGB data input.
- 8-bit / 10-bit / 16-bit data port for Bayer data input.
- Full control of 8-bit/pixel, 10-bit/pixel or 16-bit / pixel data format to 64-bitreceive FIFO packing.
- 256 x 64 FIFO to store received image pixel data.
- Reveive FIFO overrun protection mechanism.

- Embedded DMA controllers to transfer data from receive FIFO or statistic FIFO through AHB bus.
- Support 2D DMA transfer from the receive FIFO to the frame buffers in the external memory.
- Support double bufferring two frames in the external memory.
- Single interrupt source to interrupt controller from maskable interrupt sources: Start of Frame, End of Frame, Change of Field, FIFO full, FIFO overrun, DMA transfer done, CCIR error and AHB bus response error.
- Configurable master clock frequency output to sensor.
- Statistic data generation for Auto Exposure (AE) and Auto White Balance (AWB) control of the camera (only for Bayer data and 8-bit/pixel format).
- Supports simple deinterlacing of interlaced input.

13.3.2 External Signals

The table below describes the external signals for the CSI. The external signals are tied between the CSI module and an external CMOS sensor.

Table 13-5. CSI External Signals

Signal	Description	Pad	Mode	Direction
CSI_DATA00	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_DATA17	ALT3	I
CSI_DATA01	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_DATA16	ALT3	I
CSI_DATA02	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ECSPI1_SCLK	ALT3	I
		LCD1_DATA15	ALT3	
CSI_DATA03	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ECSPI1_MOSI	ALT3	I
		LCD1_DATA14	ALT3	
CSI_DATA04	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ECSPI1_MISO	ALT3	I
		LCD1_DATA13	ALT3	
CSI_DATA05	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ECSPI1_SS0	ALT3	I
		LCD1_DATA12	ALT3	
CSI_DATA06	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ECSPI2_SCLK	ALT3	I
		LCD1_DATA11	ALT3	
CSI_DATA07	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ECSPI2_MOSI	ALT3	I
		LCD1_DATA10	ALT3	

Table continues on the next page...

Table 13-5. CSI External Signals (continued)

Signal	Description	Pad	Mode	Direction
CSI_DATA08	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ECSPI2_MISO	ALT3	I
		LCD1_DATA09	ALT3	
CSI_DATA09	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	ECSPI2_SS0	ALT3	I
		LCD1_DATA08	ALT3	
CSI_DATA10	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_DATA23	ALT3	I
CSI_DATA11	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_DATA22	ALT3	I
CSI_DATA12	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_DATA21	ALT3	I
CSI_DATA13	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_DATA20	ALT3	I
CSI_DATA14	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_DATA19	ALT3	I
CSI_DATA15	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_DATA18	ALT3	I
CSI_DATA16	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_CLK	ALT3	I
CSI_DATA17	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_ENABLE	ALT3	I
CSI_DATA18	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_HSYNC	ALT3	I
CSI_DATA19	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_VSYNC	ALT3	I
CSI_DATA20	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_DATA00	ALT3	I
CSI_DATA21	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_DATA01	ALT3	I
CSI_DATA22	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_DATA02	ALT3	I
CSI_DATA23	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	LCD1_DATA03	ALT3	I
CSI_FIELD	CSI Field Signal	LCD1_RESET	ALT3	I

Table continues on the next page...

Table 13-5. CSI External Signals (continued)

Signal	Description	Pad	Mode	Direction
CSI_HSYNC	Horizontal Sync (Blank Signal)	I2C3_SDA	ALT3	I
		LCD1_DATA05	ALT3	
CSI_MCLK	"CMOS Sensor Master Clock *Note: MCLK is provided by the CCM module directly, not from the CSI module itself"	I2C4_SDA	ALT3	O
		LCD1_DATA07	ALT3	
CSI_PIXCLK	Pixel Clock	I2C4_SCL	ALT3	I
		LCD1_DATA06	ALT3	
CSI_VSYNC	Vertical Sync (Start Of Frame)	I2C3_SCL	ALT3	I
		LCD1_DATA04	ALT3	

13.3.3 Clocks

The following table describes the clock sources for CSI. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 13-6. CSI Clocks

Clock name	Clock Root	Description
csi_hclk	MAIN_AXI_CLK_ROOT	Module clock
ipg_clk	MAIN_AXI_CLK_ROOT	Peripheral clock
ipg_clk_s	MAIN_AXI_CLK_ROOT	Peripheral access clock
ipg_clk_s_raw	MAIN_AXI_CLK_ROOT	Peripheral raw data clock

13.3.4 Principles of Operation

The information found here describes the modes of operation of the sensor interface.

The CSI is designed to support generic sensor interface timing as well as CCIR656 video interface timing. Traditional CMOS sensors typically use VSYNC (SOF), HSYNC (BLANK), and PIXCLK signals to output Bayer or YUV data. Smart CMOS sensors, that come with on-chip imaging processing, usually support video mode transfer. They use an embedded timing codec to replace the VSYNC and HSYNC signal. The timing codec is defined by the CCIR656 standard.

The CSI can support connection with the sensor as follows.

- To connect with one 8-bit sensor, the sensor data interface should connect to CSI_DATA[9:2].

- To connect with one 10-bit sensor, the sensor data interface should connect to CSI_DATA[9:0].
- To connect with one 16-bit sensor, the sensor data interface should connect to CSI_DATA[15:0].
- To connect with two 8-bit sensors, the sensor data interfaces should connect to CSI_DATA[7:0] and CSI_DATA[15:8].

Table 13-7. CSI input data format

Signal Name	TVdecoder YCbCr 1 Cycle	RGB888 1 Cycle	RGB888/ YUV4444 3 Cycle	RGB666 1 Cycle	RGB565 1 Cycle	YCbCr422 1 Cycle	YCbCr422 2 Cycle	Generic 10 bit	CCIR656
ipp_csi_d[23]	Y[7]	R[7]		R[5]					
ipp_csi_d[22]	Y[6]	R[6]		R[4]					
ipp_csi_d[21]	Y[5]	R[5]		R[3]					
ipp_csi_d[20]	Y[4]	R[4]		R[2]					
ipp_csi_d[19]	Y[3]	R[3]		R[1]					
ipp_csi_d[18]	Y[2]	R[2]		R[0]					
ipp_csi_d[17]	Y[1]	R[1]		Y[5]					
ipp_csi_d[16]	Y[0]	R[0]		R[4]					
ipp_csi_d[15]	Cb[7]	G[7]		G[5]	R[4]	Y[7]			
ipp_csi_d[14]	Cb[6]	G[6]		G[4]	R[3]	Y[6]			
ipp_csi_d[13]	Cb[5]	G[5]		G[3]	R[2]	Y[5]			
ipp_csi_d[12]	Cb[4]	G[4]		G[2]	R[1]	Y[4]			
ipp_csi_d[11]	Cb[3]	G[3]		G[1]	R[0]	Y[3]			
ipp_csi_d[10]	Cb[2]	G[2]		G[0]	G[5]	Y[2]			
ipp_csi_d[9]	Cb[1]	G[1]	R/G/B[7]	G[5]	G[4]	Y[1]	Y/C[7]	Ge[9]	C/Y[7]
ipp_csi_d[8]	Cb[0]	G[0]	R/G/B[6]	G[4]	G[3]	Y[0]	Y/C[6]	Ge[8]	C/Y[6]
ipp_csi_d[7]	Cr[7]	B[7]	R/G/B[5]	B[5]	G[2]	C[7]	Y/C[5]	Ge[7]	C/Y[5]
ipp_csi_d[6]	Cr[6]	B[6]	R/G/B[4]	B[4]	G[1]	C[6]	Y/C[4]	Ge[6]	C/Y[4]

Table continues on the next page...

Table 13-7. CSI input data format (continued)

Signal Name	TVdecoder YCbCr 1 Cycle	RGB888 1 Cycle	RGB888/ YUV4444 3 Cycle	RGB666 1 Cycle	RGB565 1 Cycle	YCbCr422 1 Cycle	YCbCr422 2 Cycle	Generic 10 bit	CCIR656
ipp_csi_d[5]	Cr[5]	B[5]	R/G/B[3]	B[3]	G[0]	C[5]	Y/C[3]	Ge[5]	C/Y[3]
ipp_csi_d[4]	Cr[4]	B[4]	R/G/B[2]	B[2]	B[4]	C[4]	Y/C[2]	Ge[4]	C/Y[2]
ipp_csi_d[3]	Cr[3]	B[3]	R/G/B[1]	B[1]	B[3]	C[3]	Y/C[1]	Ge[3]	C/Y[1]
ipp_csi_d[2]	Cr[2]	B[2]	R/G/B[0]	B[0]	B[2]	C[2]	Y/C[0]	Ge[2]	C/Y[0]
ipp_csi_d[1]	Cr[1]	B[1]		B[5]	B[1]	C[1]		Ge[1]	
ipp_csi_d[0]	Cr[0]	B[0]		B[4]	B[0]	C[0]		Ge[0]	

13.3.4.1 Data Transfer with the Embedded DMA Controllers

The CSI has two embedded DMA controllers, one for the receive FIFO and the other for the statistic FIFO. It supports 2D DMA transfer from the receive FIFO to the frame buffers in the external memory and linear DMA transfer from the statistic FIFO.

To transfer data from the RxFIFO to the external memory, the user should set the start address in the frame buffer where the transferred data is stored, the parameters of the frame buffers, and the parameters of the image coming from the sensor. The user can have two frame buffers in the external memory. Each one will store a frame of image coming from the sensor. The embedded DMA controller will first write the frame buffer1 and then frame buffer2. These two frame buffers will be written by turns. The start address should be aligned in double words and set in the CSIDMASA-FB1 and CSIDMASA-FB2 registers. In the CSIFBUF_PARA register, the user should set the stride of the frame buffer to show how many double words to skip before starting to write the next row of the image. In the CSIIMAG_PARA register, the user should set the width and height of the image coming from the sensor. The RxFF_LEVEL and DMA_REQ_EN_RFF bits in CSICR3 registers also need to be set before the data transfer starts. When the number of the data in the RxFIFO reaches the trigger level, a DMA request will be sent to the embedded DMA controller and the data will be read out from the RxFIFO and written through AHB bus into the external frame buffers. The burst type of transfer can be INCR4, INCR8 and INCR16 by setting DMA_BURST_TYPE_RFF bits in CSICR2 register. After all data in an image frame are transferred, the DMA_TSF_DONE_FB1 or DMA_TSF_DONE_FB2 bit will be set in

CSISR register and the interrupt can be triggered if the corresponding enable bit is set in CSICR1 register. The DMA_REFLASH_RFF bit in CSICR3 can be used to activate or restart the embedded DMA controller.

The RxFIFO has the overrun protection mechanism in case the RxFIFO is overrun during data transfer. If the RxFIFO is full and more data needs to be received during the data transfer, the RxFIFO will be overwritten continuously and all 128 words of data in the RxFIFO before overrun occurred will be discarded; the corresponding 128 words memory space in the frame buffer will keep the previous values.

To transfer data from the statistic FIFO to the external memory, the user should set the start address of the externl memory where the transferred data is stored and the total transfer sizes. The start address and the transfer sizes are all aligned in double words and should be set in the CSIDMASA-STATFIFO and CSIDMATS-STATFIFO registers. The STATFF_LEVEL and DMA_REQ_EN_SFF bits in CSICR3 registers should also be set before the data transfer starts. When the number of the data in the STATFIFO reaches the trigger level, a dma request will be sent to the embedded DMA controller and the data will be read out from the STATFIFO and written through AHB bus into the external memory. The burst type of transfer can be INCR4, INCR8 and INCR16 by setting DMA_BURST_TYPE_SFF bits in CSICR2 register. After all expected data (defined by the total transfer sizes) are transferred, the DMA_TSF_DONE_SFF bit will be set in CSISR register and an interrupt can be triggered if the SFF_DMA_DONE_INTEN is enabled in CSICR1 register. The DMA_REFLASH_SFF bit in CSICR3 can be used to activate or re-start the embedded DMA controller.

13.3.4.2 Gated Clock Mode

VSYNC, HSYNC, and PIXCLK signals are used in gated clock mode.

A frame starts with an active edge on VSYNC, then HSYNC asserts and holds for the entire line. The Pixel clock is valid as long as HSYNC is asserted. Data is latched at the active edge of the valid pixel clocks. HSYNC deasserts at the end of line. Pixel clocks then become invalid and CSI stops receiving data from the stream. For the next line the HSYNC timing repeats. For the next frame the VSYNC timing repeats.

13.3.4.3 Non-Gated Clock Mode

In non-gated clock mode, only the VSYNC and PIXCLK signals are used; the HSYNC signal is ignored.

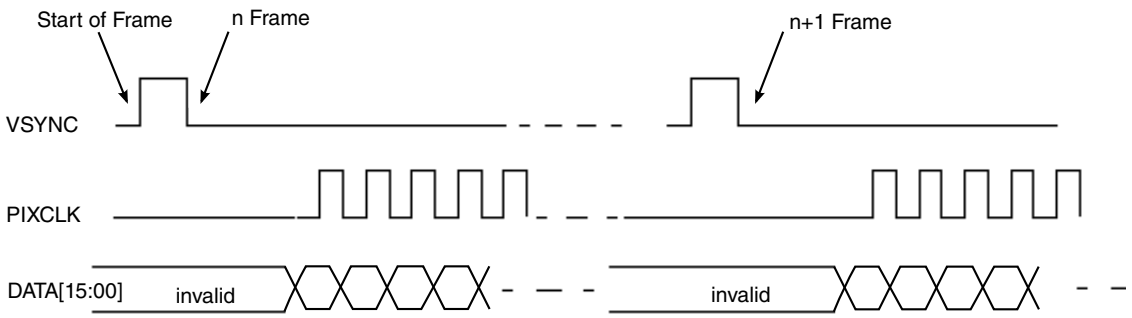


Figure 13-18. Non-Gated Clock Mode Timing Diagram

The overall timing of non-gated mode is the same as the gated-clock mode, except for the HSYNC signal. HSYNC signal is ignored by the CSI. All incoming pixel clocks are valid and cause data to be latched into RxFIFO. The PIXCLK signal is inactive (states low) until valid data is ready to be transmitted over the bus.

Figure 13-18 shows the timing of a typical sensor. Other sensors may have the slightly different timing from that shown. The CSI can be programed to support rising/falling-edge triggered VSYNC, active-high/low HSYNC, and rising/falling-edge triggered PIXCLK.

13.3.4.4 CCIR656 Interlace Mode

In CCIR656 mode, only the PIXCLK and CSI_DATA[13:6] signals are used. The start of frame and blank signals are replaced by a timing codec which is embedded in the data stream. Each active line starts with an Start of Active Video (SAV) code and ends with an End of Active Video (EAV) code. In some cases, digital blanking is inserted in between EAV and SAV code. The CSI decodes and filters out the timing-coding from the data stream, recovering VSYNC and HSYNC signals for internal use, such as statistical block control. Data is forwarded to the data receive and packing block in a sequential manner without reordering-that is, field 1 followed by field 2. The fields must be reordered in software to get back the original image.

Change of Field (COF) interrupt is triggered upon every field change. The interrupt service routine reads the status register to check for the current field.

According to the CCIR656 specification, the image must be in 625/50 PAL or 525/60 NTSC format. In addition, the image is interlaced into odd and even fields with vertical and horizontal blank data being filled into certain lines. Data must be in YCbCr422 format, each pixel contains 2 bytes, either Y + Cr or Y + Cb. These requirements are set for TV systems. The CSI module supports PAL and NTSC format only.

Figure below describes the frame structure in PAL system, showing vertical and horizontal blanking.

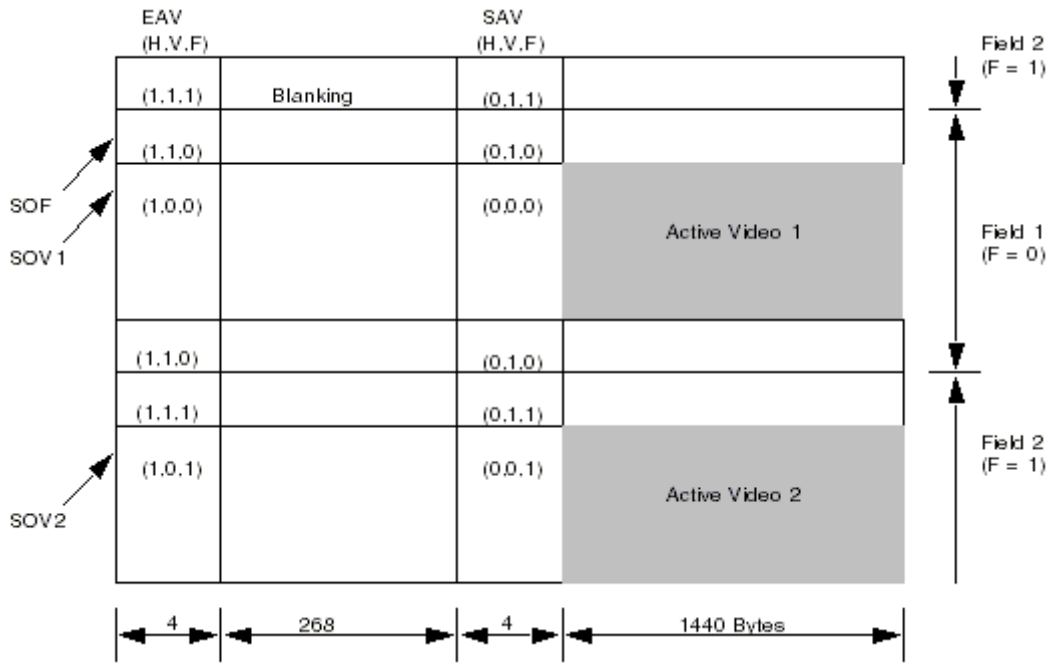


Figure 13-19. CCIR656 Interlace Mode (PAL)

Figure below describes the general timing for a single line, showing SAV and EAV.

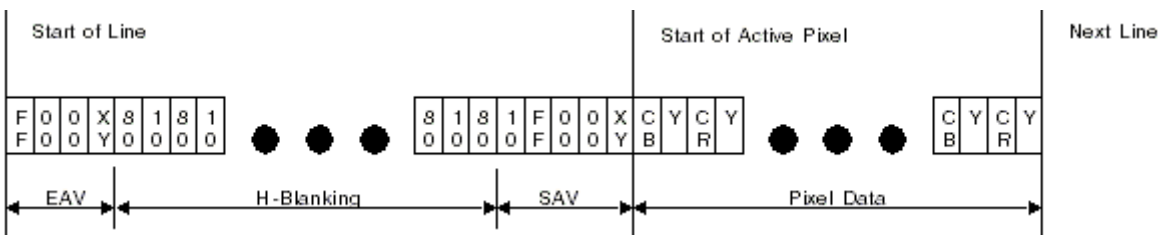


Figure 13-20. CCIR656 General Line Timing

The coding tables recommended by the CCIR656 specification are shown below. It is used in the CCIR656 mode to decode the video stream. An interrupt is generated for SOF, which is decoded from the embedded timing codec.

Table 13-8. Coding for SAV and EAV

Data Bit Number	1st Byte 0xFF	2nd Byte 0x00	3rd Byte 0x00	4th Byte 0xXY
7 (MSB)	1	0	0	1
6	1	0	0	F
5	1	0	0	V
4	1	0	0	H

Table continues on the next page...

Table 13-8. Coding for SAV and EAV (continued)

Data Bit Number	1st Byte 0xFF	2nd Byte 0x00	3rd Byte 0x00	4th Byte 0xXY
3	1	0	0	P3
2	1	0	0	P2
1	1	0	0	P1
0	1	0	0	P0

Table 13-9. Codes with Protection bits for Error Detection/Correction

F	V	H	P3	P2	P1	P0
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1

Table 13-10. Representations by F-Bit

F-Bit	Representations
0	ODD FIELD (FIELD 1)
1	EVEN FIELD (FIELD 2)

13.3.4.5 CCIR656 Progressive Mode

For a CMOS camera system of VGA or CIF resolution, strict adherence to the interlace requirements stated in the CIR standard is not required.

The image is considered to have only 1 active field which is scanned in a progressive manner. This active field is regarded as field 1 and the F-bit in the timing codec is ignored by the decoder. Most sensors support CCIR timing in this mode (progressive) by default.

Figure below shows the typical flow of progressive mode.

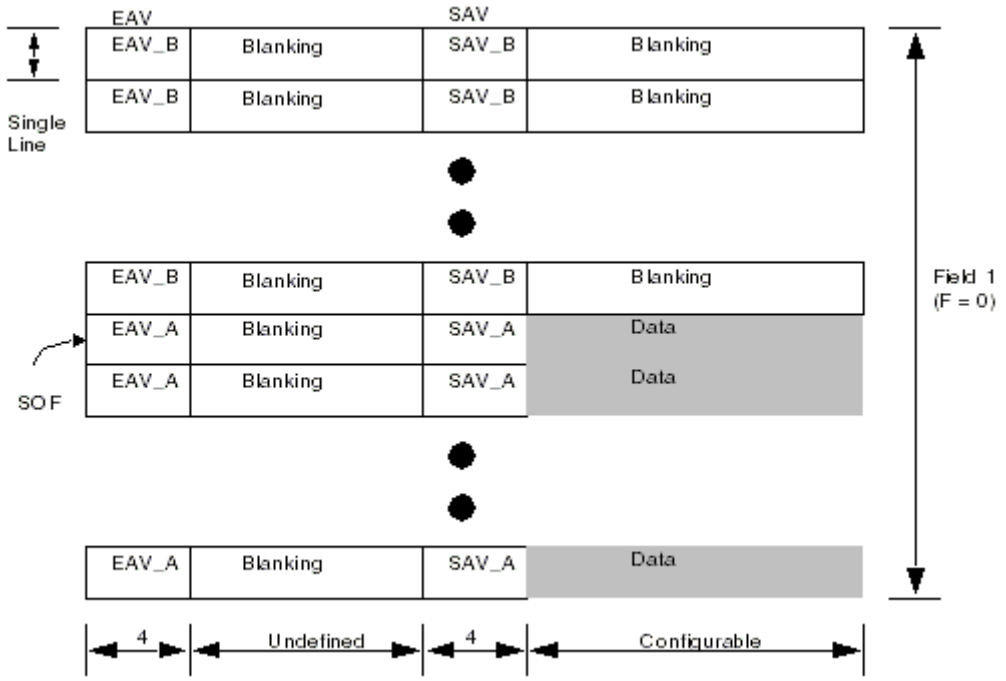


Figure 13-21. CCIR656 Progressive Mode (General Case)

An interrupt is generated for SOF but not for COF. In the general case, when SOF information is retrieved from the embedded coding, it is known as internal VSYNC mode. In other cases, when the VSYNC signal is provided by the sensor, it is known as external VSYNC mode. The CSI can be operated in internal or external VSYNC mode.

13.3.4.6 Error Correction for CCIR656 Coding

According to the algorithm for CCIR coding, protection bits in the SAV and EAV are encoded in the way that allows a 1-bit error to be corrected, or a 2-bit error to be detected by the decoder. This feature is supported by the interlace mode CCIR decoder in CSI.

For the 1-bit error case, users can select the error to be corrected automatically, or simply shown as a status flag instead. For the 2-bit error case, because the decoder is unable to make a correction, the error would be shown as a status flag only.

An interrupt can be generated upon the detection of an error. This signal can be enabled or disabled without affecting the operation of the status bit.

13.3.5 Interrupt Generation

The information found here describes CSI events that generate interrupts.

13.3.5.1 Start Of Frame Interrupt (SOF_INT)

The source of an SOF interrupt is dependent on the mode of operation.

In traditional mode, VSYNC signal is taken from sensor and SOF_INT is generated at the rising or falling edge (programmable) of VSYNC.

In CCIR interlace mode, the SOF interrupt information is retrieved from the embedded coding and SOF_INT is generated.

In CCIR progressive mode, there are two sources of an SOF interrupt:

- In *internal* VSYNC mode, SOF is retrieved from the embedded coding.
- In *external* VSYNC mode, VSYNC is taken from the sensor and SOF is generated at the rising edge of VSYNC.

13.3.5.2 End Of Frame Interrupt (EOF_INT)

An EOF interrupt is generated when the frame ends and the complete frame data in RXFIFO is read.

The EOF event triggering works with the RX count register (CSIRXCNT). Software sets the RX count register to the frame size (in words). The CSI RX logic then counts the number of pixel data being received and compares it with the RX count. If the preset value is reached, an EOF interrupt is generated and the data in the RXFIFO are read. If a SOF event is detected before this happens, the EOF interrupt is not generated.

13.3.5.3 Change Of Field Interrupt (COF_INT)

The Change of Field interrupt is only valid in CCIR Interlace mode. The COF interrupt is generated when the field toggles, either from field 1 to field 2, or field 2 to field 1.

Software should first check COF_INT bit in the CSI Status Register (CSISTAT) before checking that F1_INT or F2_INT is turned on.

In PAL systems, the field changes at the beginning of the frame and coincides with SOF. For the first field, a COF interrupt is not generated, only an SOF. The COF interrupt is generated for the second field.

13.3.5.4 CCIR Error Interrupt (ECC_INT)

The CCIR Error Interrupt is only valid for CCIR Interlace mode. An ECC interrupt is generated when an error is found on the SAV or EAV codes in the incoming stream. When this happens, the ECC_INT status bit is set.

13.3.5.5 RxFIFO Full Interrupt (RxFF_INT)

A RxFIFO full interrupt is generated when the number of data in RXFIFO reaches the water mark defined by RxFF_LEVEL in CSICR3.

13.3.5.6 Statistic FIFO Full Interrupt (STATFF_INT)

A StatFIFO full interrupt is generated when the number of data in STATFIFO reaches the water mark defined by STATFF_LEVEL in CSICR3.

13.3.5.7 RxFIFO Overrun Interrupt (RFF_OR_INT)

A RxFIFO Overrun interrupt is generated when the RxFIFO has 128 words data and more data is being written in.

13.3.5.8 Statistic FIFO Overrun Interrupt (SFF_OR_INT)

A StatFIFO Overrun interrupt is generated when the STATFIFO has 64 words data and more data is being written in.

13.3.5.9 Frame Buffer1 DMA Transfer Done Interrupt (DMA_TSF_DONE_FB1)

A DMA transfer done interrupt of frame buffer1 is generated when one frame of data are transferred from RxFIFO to the frame buffer1 in the external memory.

13.3.5.10 Frame Buffer2 DMA Transfer Done Interrupt (DMA_TSF_DONE_FB2)

A DMA transfer done interrupt of frame buffer2 is generated when one frame of data are transferred from RxFIFO to the frame buffer2 in the external memory.

13.3.5.11 Statistic FIFO DMA Transfer Done Interrupt (DMA_TSF_DONE_SFF)

A StatFIFO DMA transfer done interrupt is generated when all the data are transferred from StatFIFO to the external memory. The transfer size is defined in the STATFIFO DMA Transfer Size Register.

13.3.5.12 AHB Bus Response Error Interrupt (HRESP_ERR_INT)

An AHB Bus response error interrupt is generated when a bus error is detected.

13.3.6 Data Packing Style

Careful attention to endianess is needed given the different port sizes at different stages of the image capture path.

To enable flexible packing of image data before storage in the FIFOs, the CSI module can swap data fields by use of the PACK_DIR and the SWAP16_EN bit in CSI Control Register 1 (CSICR1).

The CSI module accepts 8-bit, 10-bit or 16-bit data from the sensor by configuring PIXEL_BIT bit in CSI Control Register 1 (CSICR1) and TWO_8BIT_SENSOR bit in CSI Control Register3 (CSICR3). The input data is packed according to the setting of PACK_DIR bit. The packed data is stored in the RX FIFO according to the setting of the SWAP16_EN bit.

For 10-bit per pixel data format, each pixel is expanded to 16 bits by appending 6 zeros bits to the most significant bit. For 16-bit data format, the data path can be a combination by two 8-bit sensors. One sensor is connected to the CSI_DATA[7:0]. The other sensor is connected to CSI_DATA[15:8].

13.3.6.1 RX FIFO Path

13.3.6.1.1 Bayer Data

Bayer data is a type of raw data from the image sensor. This byte-wide data must be converted to the RGB space or YUV space by software. The data path for Bayer data is from the CSI to memory. If the system is in little endian, then the PACK_DIR bit should be set to 0. 8-bit data format from a sensor is packed to 64 bits as

P7.P6.P5.P4.P3.P2.P1.P0, where P0 is the pixel coming in time slot 0 (first data) and P3 is the pixel coming in time slot 3 (the last data in the 64-bit word). When the data is addressed as bytes by software, P0 is transferred first, P1 is transferred next, and so on. 10-bit data format is packed to 64 bits as 000000.P3.000000.P2.000000.P1.000000.P0, where P0 is the 10-bit data coming in time slot 0 (first pixel) and P3 is the 10-bit data coming in time slot 3 (fourth pixel). 16-bit data, from two sensors, is packed to 64 bits as P7.P6.P5.P4.P3.P2.P1.P0, where P0 and P1 are the two 8-bit data coming in time slot 0 (P0 is the first pixel of the sensor connected with CSI_DATA[7:0] and P1 is the first pixel of the sensor connected with CSI_DATA[15:8]). P2 and P3 are the two 8-bit data coming in time slot 1 (second pixels of the two sensors).

13.3.6.1.2 RGB565 Data

RGB565 data is processed data from the image sensor, which can be put directly into the display buffer. The data is 16 bits wide. The data path is from CSI to memory to the display controller. On the sensor side, data must be transmitted as P0 first, followed by P1, and so on. For each pixel, whether the MSB or LSB is sent first depends on the endianness of the sensor. Data is 16 bits wide with the MSB labeled RG, and the LSB labeled GB. P0 is represented as RG0 and GB0.

CSI receives data in one of the following sequence:

- RG0, GB0, RG1, GB1, while RG0 comes out at time slot 0 (first data), and GB1 comes out at time slot 3 (last data)
- GB0, RG0, GB1, RG1

Using the first sequence as an example, and assuming the system is running in little endian, the data is presented as:

- 8-bit data from sensor: RG0, GB0, RG1, GB1, …
- 64-bit data before storage in the CSI RX FIFO (PACK_DIR bit = 1): RG0GB0RG1GB1RG2GB2RG3GB3
- 64-bitdata in CSI RX FIFO (SWAP16_EN bit enabled): RG3GB3RG2GB2RG1GB1RG0GB0
- 64-bit transfer to system memory: RG3GB3RG2GB2RG1GB1RG0GB0
- 16-bit read by display controller: RG0GB0, RG1GB1

13.3.6.1.3 RGB888 Data

This is another kind of processed data from image sensor, which can be used for further image processing directly. Each of the data consist of 8-bit Red, 8-bit Green, and 8-bit Blue data. An example of timing scheme is shown in the figure below.

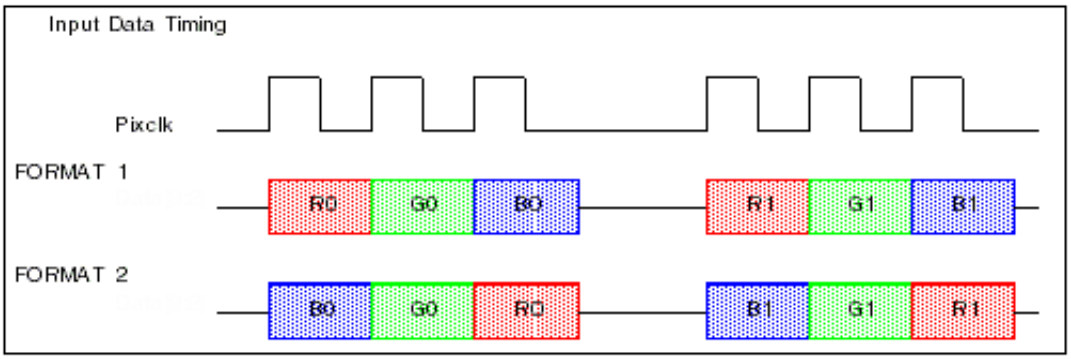
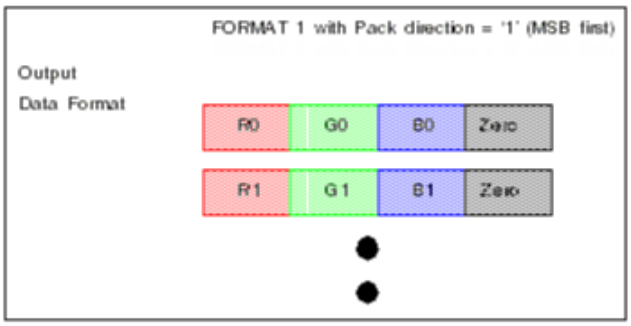


Figure 13-22. Sample Timing Diagram for RGB888 8 bits/cycle Data

An optional scheme to pack a dummy byte is provided. For every group of 3 bytes data, a dummy zero is packed to form a 32-bit word as shown in the figure below. The dummy zero can be packed at the LSB position or MSB position. Using RGB888A_FORMAT_SEL in CSI_CSICR18[18] to determine to put the dummy bytes packed at LSB or MSB position.


FORMAT 1 with Pack direction = '1' (MSB first)
Output
Data Format
R0
G0
B0
Zero
R1
G1
B1
Zero

13.3.6.2 STAT FIFO Path

Statistics only works for Bayer data in 8-bit per pixel format. It generates 16-bit statistical output from the 8-bit Bayer input (CSI_DATA[13:6]). The outputs are Sum of Green (G), Sum of Red (R), Sum of Blue (B), and Auto Focus (F). Each output is 16-bits wide.

The settings of PACK_DIR and SWAP16_EN bits in the CSICR1 register have no effect on the input path. The PACK_DIR only controls how the 16-bit stat output is packed into the 32-bit STAT FIFO.

When the PACK_DIR bit = 1, the stat data is packed as:

First 32-bit: RG

Second 32-bit: BF

…

When the PACK_DIR bit = 0, the stat data is packed as:

First 32-bit GR

Second 32-bit: FB

...

13.3.7 CSI Memory Map/Register Definition

All the 32-bit registers of the CSI module are summarized in the Memory Map below:

CSI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3071_0000	CSI Control Register 1 (CSI_CSICR1)	32	R/W	4000_0800h	13.3.7.1/ 3641
3071_0004	CSI Control Register 2 (CSI_CSICR2)	32	R/W	0000_0000h	13.3.7.2/ 3645
3071_0008	CSI Control Register 3 (CSI_CSICR3)	32	R/W	0000_0000h	13.3.7.3/ 3647
3071_000C	CSI Statistic FIFO Register (CSI_CSISTATFIFO)	32	R	0000_0000h	13.3.7.4/ 3649

Table continues on the next page...

CSI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3071_0010	CSI RX FIFO Register (CSI_CSIRFIFO)	32	R	0000_0000h	13.3.7.5/ 3649
3071_0014	CSI RX Count Register (CSI_CSIRXCNT)	32	R/W	0000_9600h	13.3.7.6/ 3650
3071_0018	CSI Status Register (CSI_CSISR)	32	R/W	0000_4000h	13.3.7.7/ 3651
3071_0020	CSI DMA Start Address Register - for STATFIFO (CSI_CSIDMASA_STATFIFO)	32	R/W	0000_0000h	13.3.7.8/ 3654
3071_0024	CSI DMA Transfer Size Register - for STATFIFO (CSI_CSIDMATS_STATFIFO)	32	R/W	0000_0000h	13.3.7.9/ 3654
3071_0028	CSI DMA Start Address Register - for Frame Buffer1 (CSI_CSIDMASA_FB1)	32	R/W	0000_0000h	13.3.7.10/ 3655
3071_002C	CSI DMA Transfer Size Register - for Frame Buffer2 (CSI_CSIDMASA_FB2)	32	R/W	0000_0000h	13.3.7.11/ 3656
3071_0030	CSI Frame Buffer Parameter Register (CSI_CSIFBUF_PARA)	32	R/W	0000_0000h	13.3.7.12/ 3656
3071_0034	CSI Image Parameter Register (CSI_CSIIMAG_PARA)	32	R/W	0000_0000h	13.3.7.13/ 3657
3071_0048	CSI Control Register 18 (CSI_CSICR18)	32	R/W	0002_D000h	13.3.7.14/ 3658

13.3.7.1 CSI Control Register 1 (CSI_CSICR1)

This register controls the sensor interface timing and interrupt generation. The interrupt enable bits in this register control the interrupt signals and the status bits. That means status bits will only function when the corresponding interrupt bits are enabled.

Address: 3071_0000h base + 0h offset = 3071_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SWAP16_EN	EXT_VSYNC	EOF_INT_EN	PrP_IF_EN	VIDEO_MODE	COF_INT_EN	SF_OR_INTEN	RF_OR_INTEN	Reserved	SFF_DMA_DONE_INTEN	STATFF_INTEN	FB2_DMA_DONE_INTEN	FB1_DMA_DONE_INTEN	RXFF_INTEN	SOF_POL	SOF_INTEN
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				HSYNC_POL	CCIR_EN	Reserved	FCC	PACK_DIR	CLR_STATFIFO	CLR_RXFIFO	GCLK_MODE	INV_DATA	INV_PCLK	REDGE	PIXEL_BIT
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

CSI_CSICR1 field descriptions

Field	Description
31 SWAP16_EN	SWAP 16-Bit Enable. This bit enables the swapping of 16-bit data. Data is packed from 8-bit or 10-bit to 32-bit first (according to the setting of PACK_DIR) and then swapped as 16-bit words before being put into the RX FIFO. The action of the bit only affects the RX FIFO and has no affect on the STAT FIFO. **NOTE:** Example of swapping enabled: Data input to FIFO = 0x11223344 Data in RX FIFO = 0x 33441122 **NOTE:** Example of swapping disabled: Data input to FIFO = 0x11223344 Data in RX FIFO = 0x11223344

Table continues on the next page...

CSI_CSICR1 field descriptions (continued)

Field	Description
	0 Disable swapping 1 Enable swapping
30 EXT_VSYNC	External VSYNC Enable. This bit controls the operational VSYNC mode. **NOTE:** This only works when the CSI is in CCIR progressive mode. 0 Internal VSYNC mode 1 External VSYNC mode
29 EOF_INT_EN	End-of-Frame Interrupt Enable. This bit enables and disables the EOF interrupt. 0 EOF interrupt is disabled. 1 EOF interrupt is generated when RX count value is reached.
28 PrP_IF_EN	CSI-PrP Interface Enable. This bit controls the CSI to PrP bus. When enabled the RxFIFO is detached from the AHB bus and connected to PrP. All CPU reads or DMA accesses to the RxFIFO register are ignored. All CSI interrupts are also masked. 0 CSI to PrP bus is disabled 1 CSI to PrP bus is enabled
27 VIDEO_MODE	Video mode select. This bit controls the video mode in CCIR mode and TV decoder input. 0 Progressive mode is selected 1 Interlace mode is selected
26 COF_INT_EN	Change Of Image Field (COF) Interrupt Enable. This bit enables the COF interrupt. This bit works only in CCIR interlace mode which is when CCIR_EN = 1 and CCIR_MODE = 1. 0 COF interrupt is disabled 1 COF interrupt is enabled
25 SF_OR_INTEN	STAT FIFO Overrun Interrupt Enable. This bit enables the STATFIFO overrun interrupt. 0 STATFIFO overrun interrupt is disabled 1 STATFIFO overrun interrupt is enabled
24 RF_OR_INTEN	RxFIFO Overrun Interrupt Enable. This bit enables the RX FIFO overrun interrupt. 0 RxFIFO overrun interrupt is disabled 1 RxFIFO overrun interrupt is enabled
23 -	This field is reserved. Reserved. This bit is reserved and should read 0.
22 SFF_DMA_ DONE_INTEN	STATFIFO DMA Transfer Done Interrupt Enable. This bit enables the interrupt of STATFIFO DMA transfer done. 0 STATFIFO DMA Transfer Done interrupt disable 1 STATFIFO DMA Transfer Done interrupt enable
21 STATFF_INTEN	STATFIFO Full Interrupt Enable. This bit enables the STAT FIFO interrupt. 0 STATFIFO full interrupt disable 1 STATFIFO full interrupt enable
20 FB2_DMA_ DONE_INTEN	Frame Buffer2 DMA Transfer Done Interrupt Enable. This bit enables the interrupt of Frame Buffer2 DMA transfer done.

Table continues on the next page...

CSI_CSICR1 field descriptions (continued)

Field	Description
	0 Frame Buffer2 DMA Transfer Done interrupt disable 1 Frame Buffer2 DMA Transfer Done interrupt enable
19 FB1_DMA_DONE_INTEN	Frame Buffer1 DMA Transfer Done Interrupt Enable. This bit enables the interrupt of Frame Buffer1 DMA transfer done. 0 Frame Buffer1 DMA Transfer Done interrupt disable 1 Frame Buffer1 DMA Transfer Done interrupt enable
18 RXFF_INTEN	RxFIFO Full Interrupt Enable. This bit enables the RxFIFO full interrupt. 0 RxFIFO full interrupt disable 1 RxFIFO full interrupt enable
17 SOF_POL	SOF Interrupt Polarity. This bit controls the condition that generates an SOF interrupt. 0 SOF interrupt is generated on SOF falling edge 1 SOF interrupt is generated on SOF rising edge
16 SOF_INTEN	Start Of Frame (SOF) Interrupt Enable. This bit enables the SOF interrupt. 0 SOF interrupt disable 1 SOF interrupt enable
15–12 Reserved	This field is reserved. Reserved. This field is reserved.
11 HSYNC_POL	HSYNC Polarity Select. This bit controls the polarity of HSYNC. This bit only works in gated-clock-that is, GCLK_MODE = 1 and CCIR_EN = 0. 0 HSYNC is active low 1 HSYNC is active high
10 CCIR_EN	CCIR656 Interface Enable. This bit selects the type of interface used. When the CCIR656 timing decoder is enabled, it replaces the function of timing interface logic. 0 Traditional interface is selected. Timing interface logic is used to latch data. 1 CCIR656 interface is selected.
9 Reserved	This field is reserved. This field is reserved.
8 FCC	FIFO Clear Control. This bit determines how the RXFIFO and STATFIFO are cleared. When Synchronous FIFO clear is selected the RXFIFO and STATFIFO are cleared, and STAT block is reset, on every SOF. FIFOs and STAT block restarts immediately after reset. For information on the operation when Asynchronous FIFO clear is selected, refer to the descriptions for the CLR_RXFIFO and CLR_STATFIFO bits. 0 Asynchronous FIFO clear is selected. 1 Synchronous FIFO clear is selected.
7 PACK_DIR	Data Packing Direction. This bit Controls how 8-bit/10-bit image data is packed into 32-bit RX FIFO, and how 16-bit statistical data is packed into 32-bit STAT FIFO. 0 Pack from LSB first. For image data, 0x11, 0x22, 0x33, 0x44, it will appear as 0x44332211 in RX FIFO. For stat data, 0xAAAA, 0xBBBB, it will appear as 0xBBBBAAAA in STAT FIFO. 1 Pack from MSB first. For image data, 0x11, 0x22, 0x33, 0x44, it will appear as 0x11223344 in RX FIFO. For stat data, 0xAAAA, 0xBBBB, it will appear as 0xAAAABBBB in STAT FIFO.

Table continues on the next page...

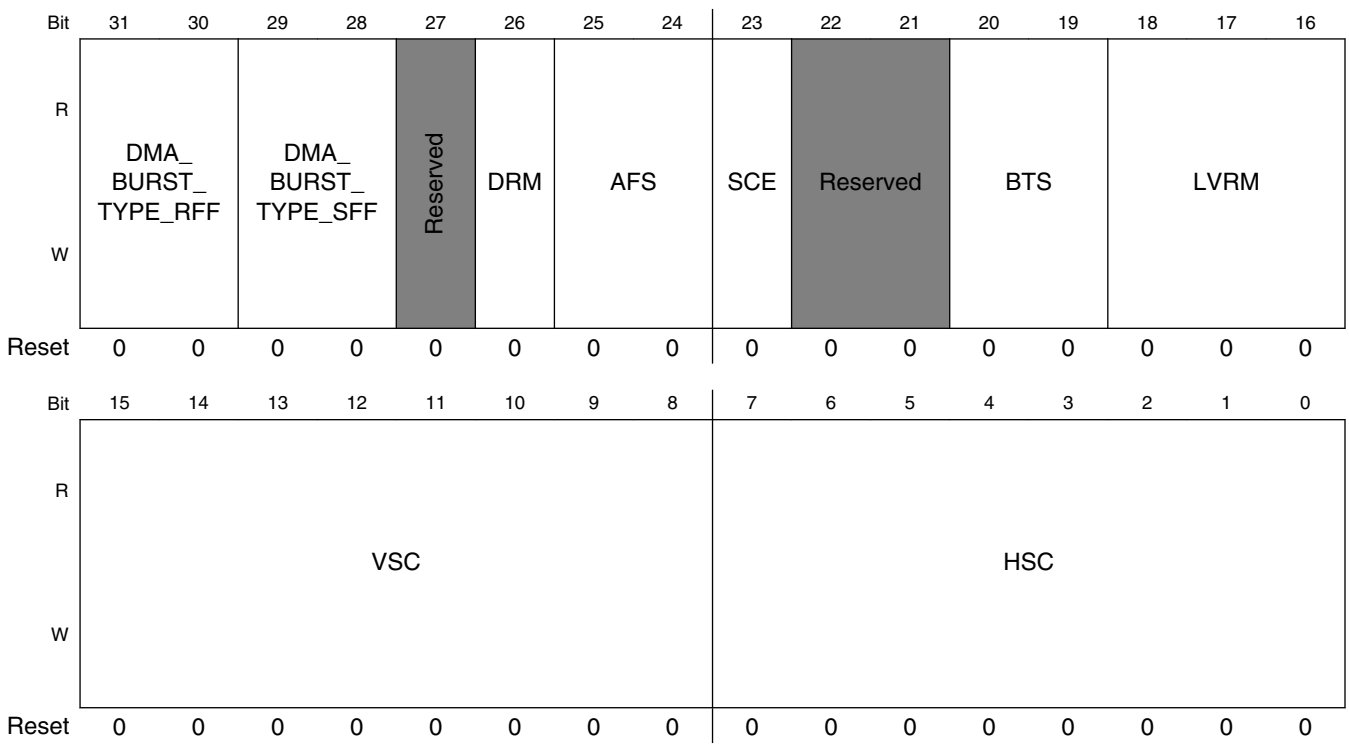
CSI_CSICR1 field descriptions (continued)

Field	Description
6 CLR_STATFIFO	Asynchronous STATFIFO Clear. This bit clears the STATFIFO and Reset STAT block. This bit works only in async FIFO clear mode-that is, FCC = 0. Otherwise this bit is ignored. Writing 1 will clear STATFIFO and reset STAT block immediately, STATFIFO and STAT block then wait and restart after the arrival of next SOF. The bit is restored to 0 automatically after finish. Normally reads 0.
5 CLR_RXFIFO	Asynchronous RXFIFO Clear. This bit clears the RXFIFO. This bit works only in async FIFO clear mode-that is, FCC = 0. Otherwise this bit is ignored. Writing 1 clears the RXFIFO immediately, RXFIFO restarts immediately after that. The bit is restored to 0 automatically after finish. Normally reads 0.
4 GCLK_MODE	Gated Clock Mode Enable. Controls if CSI is working in gated or non-gated mode. This bit works only in traditional mode-that is, CCIR_EN = 0. Otherwise this bit is ignored. 0 Non-gated clock mode. All incoming pixel clocks are valid. HSYNC is ignored. 1 Gated clock mode. Pixel clock signal is valid only when HSYNC is active.
3 INV_DATA	Invert Data Input. This bit enables or disables internal inverters on the data lines. 0 CSI_D[7:0] data lines are directly applied to internal circuitry 1 CSI_D[7:0] data lines are inverted before applied to internal circuitry
2 INV_PCLK	Invert Pixel Clock Input. This bit determines if the Pixel Clock (CSI_PIXCLK) is inverted before it is applied to the CSI module. 0 CSI_PIXCLK is directly applied to internal circuitry 1 CSI_PIXCLK is inverted before applied to internal circuitry
1 REDGE	Valid Pixel Clock Edge Select. Selects which edge of the CSI_PIXCLK is used to latch the pixel data. 0 Pixel data is latched at the falling edge of CSI_PIXCLK 1 Pixel data is latched at the rising edge of CSI_PIXCLK
0 PIXEL_BIT	Pixel Bit. This bit indicates the bayer data width for each pixel. This bit should be configured before activating or re-starting the embedded DMA controller. 0 8-bit data for each pixel 1 10-bit data for each pixel

13.3.7.2 CSI Control Register 2 (CSI_CSICR2)

This register provides the statistic block with data about which live view resolution is being used, and the starting sensor pixel of the Bayer pattern. It also contains the horizontal and vertical count used to determine the number of pixels to skip between the 64 x 64 blocks of statistics when generating statistics on live view image that are greater than 512 x 384.

Address: 3071_0000h base + 4h offset = 3071_0004h



CSI_CSICR2 field descriptions

Field	Description
31–30 DMA_BURST_ TYPE_RFF	Burst Type of DMA Transfer from RxFIFO. Selects the burst type of DMA transfer from RxFIFO. X0 INCR8 01 INCR4 11 INCR16
29–28 DMA_BURST_ TYPE_SFF	Burst Type of DMA Transfer from STATFIFO. Selects the burst type of DMA transfer from STATFIFO. X0 INCR8 01 INCR4 11 INCR16

Table continues on the next page...

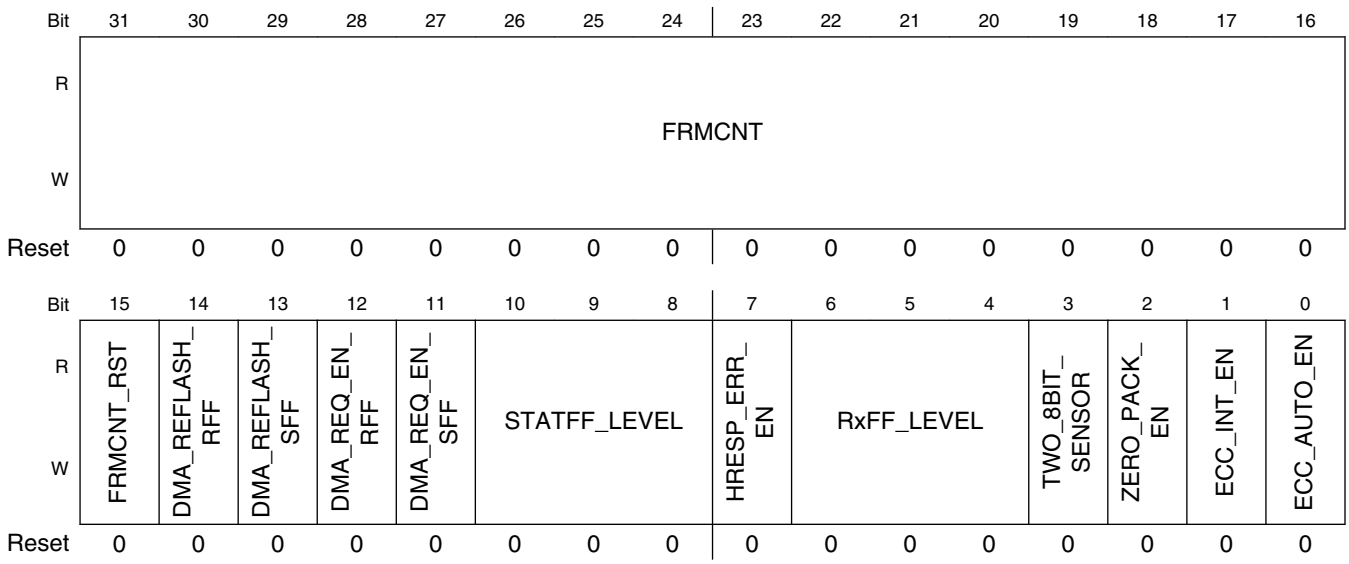
CSI_CSICR2 field descriptions (continued)

Field	Description
27 -	This field is reserved. Reserved. These bit is reserved and should read 0.
26 DRM	Double Resolution Mode. Controls size of statistics grid. 0 Stats grid of 8 x 6 1 Stats grid of 8 x 12
25–24 AFS	Auto Focus Spread. Selects which green pixels are used for auto-focus. 00 Abs Diff on consecutive green pixels 01 Abs Diff on every third green pixels 1x Abs Diff on every four green pixels
23 SCE	Skip Count Enable. Enables or disables the skip count feature. 0 Skip count disable 1 Skip count enable
22–21 -	This field is reserved. Reserved. These bits are reserved and should read 0.
20–19 BTS	Bayer Tile Start. Controls the Bayer pattern starting point. 00 GR 01 RG 10 BG 11 GB
18–16 LVRM	Live View Resolution Mode. Selects the grid size used for live view resolution. 0 512 x 384 1 448 x 336 2 384 x 288 3 384 x 256 4 320 x 240 5 288 x 216 6 400 x 300
15–8 VSC	Vertical Skip Count. Contains the number of rows to skip. SCE must be 1, otherwise VSC is ignored. 0-255 Number of rows to skip minus 1
HSC	Horizontal Skip Count. Contains the number of pixels to skip. SCE must be 1, otherwise HSC is ignored. 0-255 Number of pixels to skip minus 1

13.3.7.3 CSI Control Register 3 (CSI_CSICR3)

This read/write register acts as an extension of the functionality of the CSI Control register 1, adding additional control and features.

Address: 3071_0000h base + 8h offset = 3071_0008h



CSI_CSICR3 field descriptions

Field	Description
31–16 FRMCNT	Frame Counter. This is a 16-bit Frame Counter (Wraps around automatically after reaching the maximum)
15 FRMCNT_RST	Frame Count Reset. Resets the Frame Counter. (Cleared automatically after reset is done) 0 Do not reset 1 Reset frame counter immediately
14 DMA_REFLASH_RFF	Reflash DMA Controller for RxFIFO. This bit reflash the embedded DMA controller for RxFIFO. It should be reflashed before the embedded DMA controller starts to work. (Cleared automatically after reflashing is done) 0 No reflashing 1 Reflash the embedded DMA controller
13 DMA_REFLASH_SFF	Reflash DMA Controller for STATFIFO. This bit reflash the embedded DMA controller for STATFIFO. It should be reflashed before the embedded DMA controller starts to work. (Cleared automatically after reflashing is done) 0 No reflashing 1 Reflash the embedded DMA controller

Table continues on the next page...

CSI_CSICR3 field descriptions (continued)

Field	Description
12 DMA_REQ_EN_RFF	DMA Request Enable for RxFIFO. This bit enables the dma request from RxFIFO to the embedded DMA controller. 0 Disable the dma request 1 Enable the dma request
11 DMA_REQ_EN_SFF	DMA Request Enable for STATFIFO. This bit enables the dma request from STATFIFO to the embedded DMA controller. 0 Disable the dma request 1 Enable the dma request
10–8 STATFF_LEVEL	STATFIFO Full Level. When the number of data in STATFIFO reach this level, STATFIFO full interrupt is generated, or STATFIFO DMA request is sent. 000 4 Double words 001 8 Double words 010 12 Double words 011 16 Double words 100 24 Double words 101 32 Double words 110 48 Double words 111 64 Double words
7 HRESP_ERR_EN	Hresponse Error Enable. This bit enables the hresponse error interrupt. 0 Disable hresponse error interrupt 1 Enable hresponse error interrupt
6–4 RxFF_LEVEL	**RxFIFO Full Level**. When the number of data in RxFIFO reaches this level, a RxFIFO full interrupt is generated, or an RXFIFO DMA request is sent. 000 4 Double words 001 8 Double words 010 16 Double words 011 24 Double words 100 32 Double words 101 48 Double words 110 64 Double words 111 96 Double words
3 TWO_8BIT_SENSOR	Two 8-bit Sensor Mode. This bit indicates one 16-bit sensor or two 8-bit sensors are connected to the 16-bit data ports. This bit should be set if there is one 16-bit sensor or two 8-bit sensors are connected. This bit should be configured before activating or restarting the embedded DMA controller. 0 Only one sensor is connected. 1 Two 8-bit sensors are connected or one 16-bit sensor is connected.
2 ZERO_PACK_EN	Dummy Zero Packing Enable. This bit causes a dummy zero to be packed with every 3 incoming bytes, forming a 32-bit word. The dummy zero is always packed to the LSB position. This packing function is only available in 8-bit/pixel mode. 0 Zero packing disabled 1 Zero packing enabled

Table continues on the next page...

CSI_CSICR3 field descriptions (continued)

Field	Description
1 ECC_INT_EN	Error Detection Interrupt Enable. This bit enables and disables the error detection interrupt. This feature only works in CCIR interlace mode. 0 No interrupt is generated when error is detected. Only the status bit ECC_INT is set. 1 Interrupt is generated when error is detected.
0 ECC_AUTO_EN	Automatic Error Correction Enable. This bit enables and disables the automatic error correction. If an error occurs and error correction is disabled only the ECC_INT status bit is set. This feature only works in CCIR interlace mode. 0 Auto Error correction is disabled. 1 Auto Error correction is enabled.

13.3.7.4 CSI Statistic FIFO Register (CSI_CSISTATFIFO)

The StatFIFO is a read-only register containing statistic data from the sensor. Writing to this register has no effect.

Address: 3071_0000h base + Ch offset = 3071_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STAT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_CSISTATFIFO field descriptions

Field	Description
STAT	Static data from sensor

13.3.7.5 CSI RX FIFO Register (CSI_CSIRFIFO)

This read-only register contains received image data. Writing to this register has no effect.

Address: 3071_0000h base + 10h offset = 3071_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IMAGE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_CSIRFIFO field descriptions

Field	Description
IMAGE	Received image data

13.3.7.6 CSI RX Count Register (CSI_CSIRXCNT)

This register works for EOF interrupt generation. It should be set to the number of words to receive that would generate an EOF interrupt.

There is an internal counter that counts the number of words read from the RX FIFO. Whenever the RX FIFO is being read, by either the CPU or the embedded DMA controller, the counter value is updated and compared with this register. If the values match, then an EOF interrupt is triggered.

Address: 3071_0000h base + 14h offset = 3071_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										RXCNT																					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0

CSI_CSIRXCNT field descriptions

Field	Description
31–22 -	This field is reserved. Reserved. These bits are reserved and should read 0.
RXCNT	RxFIFO Count. This 22-bit counter for RXFIFO is updated each time the RXFIFO is read by CPU or DMA.This counter should be set to the expected number of words to receive that would generate an EOF interrupt.

13.3.7.7 CSI Status Register (CSI_CSISR)

This read/write register shows sensor interface status, and which kind of interrupt is being generated. The corresponding interrupt bits must be set for the status bit to function. Status bits should function normally even if the corresponding interrupt enable bits are not enabled.

Address: 3071_0000h base + 18h offset = 3071_0018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-			BASEADDR_CHHANGE_ERROR	DMA_FIELD0_DONE	DMA_FIELD1_DONE	SF_OR_INT	RF_OR_INT	Reserved	DMA_TSF_DONE_SFF	STATFF_INT	DMA_TSF_DONE_FB2	DMA_TSF_DONE_FB1	RxFF_INT	EOF_INT	SOF_INT
W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	F2_INT	F1_INT	COF_INT	Reserved					HRESP_ERR_INT	Reserved					ECC_INT	DRDY
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_CSISR field descriptions

Field	Description
31–29 -	Reserved. These bits are reserved and should read 0.
28 BASEADDR_CHHANGE_ERROR	When using base address switching enable, this bit will be 1 when switching occur before DMA complete. This bit will be clear by writing 1. When this interrupt happens, follow the steps listed below. 1. Unassert the CSI enable, CSIx_CSICR18 bit31, 2. Reflash the DMA, assert the CSIX_CSICR3 bit 14, 3. Assert the CSI enable, CSIx_CSICR18 bit31.
27 DMA_FIELD0_DONE	When DMA field 0 is complete, this bit will be set to 1(clear by writing 1).

Table continues on the next page...

CSI_CSISR field descriptions (continued)

Field	Description
26 DMA_FIELD1_DONE	When DMA field 0 is complete, this bit will be set to 1(clear by writing 1).
25 SF_OR_INT	STATFIFO Overrun Interrupt Status. Indicates the overflow status of the STATFIFO register. (Cleared by writing 1) 0 STATFIFO has not overflowed. 1 STATFIFO has overflowed.
24 RF_OR_INT	RxFIFO Overrun Interrupt Status. Indicates the overflow status of the RxFIFO register. (Cleared by writing 1) 0 RXFIFO has not overflowed. 1 RXFIFO has overflowed.
23 -	This field is reserved. Reserved. This bit is reserved and should read 0.
22 DMA_TSF_DONE_SFF	DMA Transfer Done from StatFIFO. Indicates that the dma transfer from StatFIFO is completed. It can trigger an interrupt if the corresponding enable bit is set in CSICR1. This bit can be cleared by writting 1 or reflashing the StatFIFO dma controller in CSICR3.(Cleared by writing 1) 0 DMA transfer is not completed. 1 DMA transfer is completed.
21 STATFF_INT	STATFIFO Full Interrupt Status. Indicates the number of data in the STATFIFO reaches the trigger level. (this bit is cleared automatically by reading the STATFIFO) 0 STATFIFO is not full. 1 STATFIFO is full.
20 DMA_TSF_DONE_FB2	DMA Transfer Done in Frame Buffer2. Indicates that the DMA transfer from RxFIFO to Frame Buffer2 is completed. It can trigger an interrupt if the corresponding enable bit is set in CSICR1. This bit can be cleared by by writting 1 or reflashing the RxFIFO dma controller in CSICR3. (Cleared by writing 1) 0 DMA transfer is not completed. 1 DMA transfer is completed.
19 DMA_TSF_DONE_FB1	DMA Transfer Done in Frame Buffer1. Indicates that the DMA transfer from RxFIFO to Frame Buffer1 is completed. It can trigger an interrupt if the corresponding enable bit is set in CSICR1. This bit can be cleared by by writting 1 or reflashing the RxFIFO dma controller in CSICR3. (Cleared by writing 1) 0 DMA transfer is not completed. 1 DMA transfer is completed.
18 RxFF_INT	RXFIFO Full Interrupt Status. Indicates the number of data in the RxFIFO reaches the trigger level. (this bit is cleared automatically by reading the RxFIFO) 0 RxFIFO is not full. 1 RxFIFO is full.
17 EOF_INT	End of Frame (EOF) Interrupt Status. Indicates when EOF is detected. (Cleared by writing 1) 0 EOF is not detected. 1 EOF is detected.
16 SOF_INT	Start of Frame Interrupt Status. Indicates when SOF is detected. (Cleared by writing 1)

Table continues on the next page...

CSI_CSISR field descriptions (continued)

Field	Description
	0 SOF is not detected. 1 SOF is detected.
15 F2_INT	CCIR Field 2 Interrupt Status. Indicates the presence of field 2 of video in CCIR mode. (Cleared automatically when current field does not match) **NOTE:** Only works in CCIR Interlace mode. 0 Field 2 of video is not detected 1 Field 2 of video is about to start
14 F1_INT	CCIR Field 1 Interrupt Status. Indicates the presence of field 1 of video in CCIR mode. (Cleared automatically when current field does not match) **NOTE:** Only works in CCIR Interlace mode. 0 Field 1 of video is not detected. 1 Field 1 of video is about to start.
13 COF_INT	Change Of Field Interrupt Status. Indicates that a change of the video field has been detected. Only works in CCIR Interlace mode. Software should read this bit first and then dispatch the new field from F1_INT and F2_INT. (Cleared by writing 1) 0 Video field has no change. 1 Change of video field is detected.
12–8 -	This field is reserved. Reserved. These bits are reserved and should read 0.
7 HRESP_ERR_INT	Hresponse Error Interrupt Status. Indicates that a hresponse error has been detected. (Cleared by writing 1) 0 No hresponse error. 1 Hresponse error is detected.
6–2 -	This field is reserved. Reserved. These bits are reserved and should read 0.
1 ECC_INT	CCIR Error Interrupt. This bit indicates an error has occurred. This only works in CCIR Interlace mode. (Cleared by writing 1) 0 No error detected 1 Error is detected in CCIR coding
0 DRDY	RXFIFO Data Ready. Indicates the presence of data that is ready for transfer in the RxFIFO. (Cleared automatically by reading FIFO) 0 No data (word) is ready 1 At least 1 datum (word) is ready in RXFIFO.

13.3.7.8 CSI DMA Start Address Register - for STATFIFO (CSI_CSIDMASA_STATFIFO)

This register provides the start address for the embedded DMA controller of STATFIFO. The embedded DMA controller will read data from STATFIFO and write it to the external memory from the start address. This register should be configured before activating or restarting the embedded DMA controller.

Address: 3071_0000h base + 20h offset = 3071_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DMA_START_ADDR_SFF															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMA_START_ADDR_SFF														Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_CSIDMASA_STATFIFO field descriptions

Field	Description
31–2 DMA_START_ ADDR_SFF	DMA Start Address for STATFIFO. Indicates the start address to write data. The embedded DMA controller will read data from STATFIFO and write it from this address through AHB bus. The address should be double words aligned.
-	This field is reserved. Reserved. These bits are reserved and should read 0.

13.3.7.9 CSI DMA Transfer Size Register - for STATFIFO (CSI_CSIDMATS_STATFIFO)

This register provides the total transfer size for the embedded DMA controller of STATFIFO. This register should be configured before activating or restarting the embedded DMA controller.

Address: 3071_0000h base + 24h offset = 3071_0024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMA_TSF_SIZE_SFF																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_CSIDMATS_STATFIFO field descriptions

Field	Description
DMA_TSF_SIZE_SFF	DMA Transfer Size for STATFIFO. Indicates how many words to be transfered by the embedded DMA controller. The size should be double words aligned.

13.3.7.10 CSI DMA Start Address Register - for Frame Buffer1 (CSI_CSIDMASA_FB1)

This register provides the start address in the frame buffer1 for the embedded DMA controller of RxFIFO. The embedded DMA controller will read data from RxFIFO and write it to the frame buffer1 from the start address. This register should be configured before activating or restarting the embedded DMA controller.

Address: 3071_0000h base + 28h offset = 3071_0028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DMA_START_ADDR_FB1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMA_START_ADDR_FB1														Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_CSIDMASA_FB1 field descriptions

Field	Description
31–2 DMA_START_ADDR_FB1	DMA Start Address in Frame Buffer1. Indicates the start address to write data. The embedded DMA controller will read data from RxFIFO and write it from this address through AHB bus. The address should be double words aligned.
-	This field is reserved. Reserved. These bits are reserved and should read 0.

13.3.7.11 CSI DMA Transfer Size Register - for Frame Buffer2 (CSI_CSIDMASA_FB2)

This register provides the start address in the frame buffer2 for the embedded DMA controller of RxFIFO. The embedded DMA controller will read data from RxFIFO and write it to the frame buffer2 from the start address. This register should be configured before activating or restarting the embedded DMA controller.

Address: 3071_0000h base + 2Ch offset = 3071_002Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DMA_START_ADDR_FB2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMA_START_ADDR_FB2														Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_CSIDMASA_FB2 field descriptions

Field	Description
31–2 DMA_START_ ADDR_FB2	DMA Start Address in Frame Buffer2. Indicates the start address to write data. The embedded DMA controller will read data from RxFIFO and write it from this address through AHB bus. The address should be double words aligned.
-	This field is reserved. Reserved. These bits are reserved and should read 0.

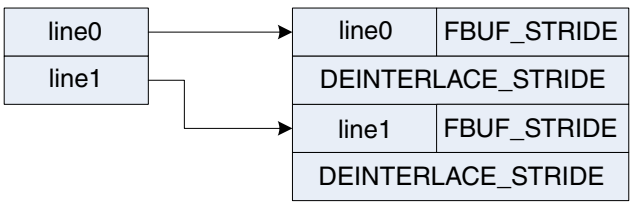
13.3.7.12 CSI Frame Buffer Parameter Register (CSI_CSIFBUF_PARA)

This register provides the stride of the frame buffer to show how many words to skip before starting to write the next row of the image. The width of the frame buffer minus the width of the image is the stride. This register should be configured before activating or restarting the embedded DMA controller.

Address: 3071_0000h base + 30h offset = 3071_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DEINTERLACE_STRIDE																FBUF_STRIDE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_CSIFBUF_PARA field descriptions

Field	Description
31–16 DEINTERLACE_ STRIDE	DEINTERLACE_STRIDE is only used in the deinterlace mode. If line stride feature is supported in deinterlace mode, FBUF_STRIDE and DEINTERLACE_STRIDE need to be configured at the same time. DEINTERLACE_STRIDE is configured the same as line width. In normal line stride feature, only FBUF_STRIDE needs to be configured. 
FBUF_STRIDE	Frame Buffer Parameter. Indicates the stride of the frame buffer. The width of the frame buffer(in double words) minus the width of the image(in double words) is the stride. The stride should be double words aligned. The embedded DMA controller will skip the stride before starting to write the next row of the image.

13.3.7.13 CSI Image Parameter Register (CSI_CSIIMAG_PARA)

This register provides the width and the height of the image from the sensor. The width and height should be aligned in pixel. The width of the image multiplied by the height is the total pixel size that will be transfered in a frame by the embedded DMA controller. This register should be configured before activating or restarting the embedded DMA controller.

Address: 3071_0000h base + 34h offset = 3071_0034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMAGE_WIDTH																IMAGE_HEIGHT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

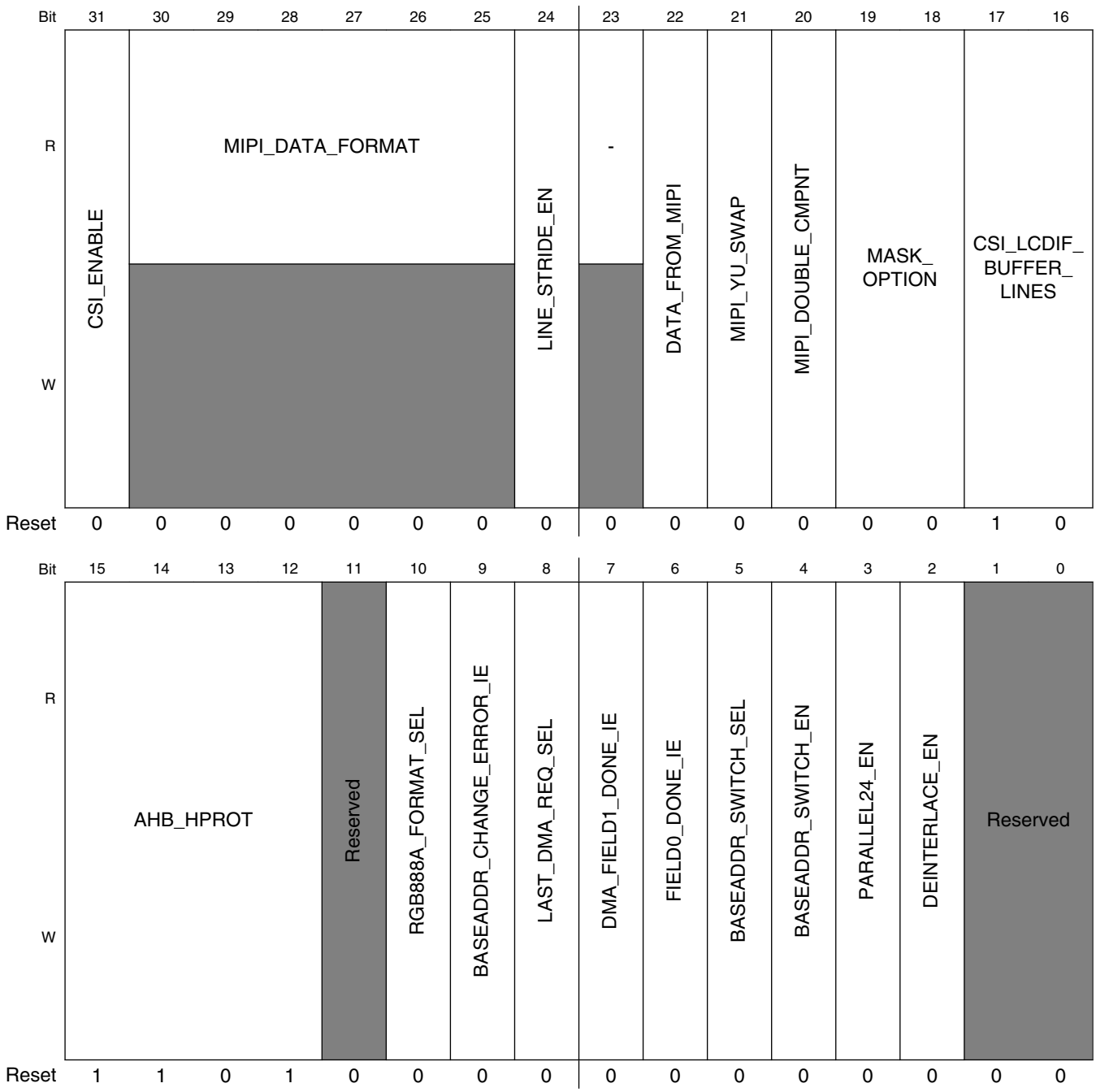
CSI_CSIIMAG_PARA field descriptions

Field	Description
31–16 IMAGE_WIDTH	Image Width. Indicates how many pixels in a line of the image from the sensor. If the input data from the sensor is 8-bit/pixel format, the IMAGE_WIDTH should be a multiple of 8 pixels. If the input data from the sensor is 10-bit/pixel or 16-bit/pixel format, the IMAGE_WIDTH should be a multiple of 2 pixels.
IMAGE_HEIGHT	Image Height. Indicates how many pixels in a column of the image from the sensor.

13.3.7.14 CSI Control Register 18 (CSI_CSICR18)

This read/write register acts as an extension of the functionality of the CSI Control register 1

Address: 3071_0000h base + 48h offset = 3071_0048h



CSI_CSICR18 field descriptions

Field	Description
31 CSI_ENABLE	CSI global enable signal. Only when this bit is 1, CSI can start to receive the data and store to memory.
30–25 MIPI_DATA_FORMAT	Image Data Format Generic Short Packet: 0x08 ~ 0x0F Embedded 8-bit non Image: 0x12 YUV420 (8-bit) 0x18 YUV420 (10-bit) 0x19 YUV420 (8-bit legacy) 0x1A YUV420 (8-bit CSPS) 0x1C YUV420 (10-bit CSPS) 0x1D YUV422 (8-bit) 0x1E YUV422 (10-bit) 0x1F RGB444 0x20 (Not support) RGB555 0x21 (Not support) RGB565 0x22 RGB666 0x23 RGB888 0x24 RAW6 0x28 RAW7 0x29 RAW8 0x2A RAW10 0x2B RAW12 0x2C RAW14 0x2D User defined 1 0x30 User defined 2 0x31 User defined 3 0x32 User defined 4 0x33 User defined 5 0x34 User defined 6 0x35 User defined 7 0x36 User defined 8 0x37
24 LINE_STRIDE_EN	When the line width are not the multiple of the burst length, assert this bit.
23 -	Reserved
22 DATA_FROM_MIPI	0 Data from parallel sensor 1 Data from MIPI
21 MIPI_YU_SWAP	It only works in MIPI CSI YUV422 double component mode.
20 MIPI_DOUBLE_CMPNT	Double component per clock cycle in YUV422 formats. 0 Single component per clock cycle

Table continues on the next page...

CSI_CSICR18 field descriptions (continued)

Field	Description
	(half pixel per clock cycle) 1 Double component per clock cycle (a pixel per clock cycle)
19–18 MASK_OPTION	These bits used to choose the method to mask the CSI input. 00 Writing to memory from first completely frame, when using this option, the CSI_ENABLE should be 1. 01 Writing to memory when CSI_ENABLE is 1. 02 Writing to memory from second completely frame, when using this option, the CSI_ENABLE should be 1. 03 Writing to memory when data comes in, not matter the CSI_ENABLE is 1 or 0.
17–16 CSI_LCDIF_ BUFFER_LINES	The number of lines are used in handshake mode with LCDIF. 00 4 lines 01 8 lines 02 16 lines 03 16 lines
15–12 AHB_HPROT	Hprot value in AHB bus protocol.
11 -	This field is reserved.
10 RGB888A_ FORMAT_SEL	Output is 32-bit format. 0 {8'h0, data[23:0]} 1 {data[23:0], 8'h0}
9 BASEADDR_ CHANGE_ ERROR_IE	Base address change error interrupt enable signal.
8 LAST_DMA_ REQ_SEL	Choosing the last DMA request condition. 0 fifo_full_level 1 hburst_length
7 DMA_FIELD1_ DONE_IE	When in interlace mode, field 1 done interrupt enable. 0 Interrupt disabled 1 Interrupt enabled
6 FIELD0_DONE_ IE	In interlace mode, fileld 0 means interrupt enabled. 0 Interrupt disabled 1 Interrupt enabled
5 BASEADDR_ SWITCH_SEL	CSI 2 base addresses switching method. When using this bit, BASEADDR_SWITCH_EN is 1. 0 Switching base address at the edge of the vsync 1 Switching base address at the edge of the first data of each frame
4 BASEADDR_ SWITCH_EN	When this bit is enabled, CSI DMA will switch the base address according to BASEADDR_SWITCH_SEL rather than atomically by DMA completed.

Table continues on the next page...

CSI_CSICR18 field descriptions (continued)

Field	Description
3 PARALLEL24_ EN	When input is parallel rgb888/yuv444 24bit, this bit can be enabled.
2 DEINTERLACE_ EN	This bit is used to select the output method When input is standard CCIR656 video. 0 Deinterlace disabled 1 Deinterlace enabled
-	This field is reserved. Reserved.

13.4 MIPI DSI Host Controller (MIPI_DSI)

13.4.1 Overview

13.4.1.1 Key features

The following list summarizes the key features of the MIPI DSI:

- Complies to MIPI DSI Standard Specification V1.01r11
  - Maximum resolution ranges up to SXGA+(1400 x 11050 @ 60 Hz, 24 bpp)
    - It should be decided on bandwidth between input clock (video clock) and output clock (D-PHY HS clock).
  - Supports 1, 2 data lanes
  - Supports pixel format: 16 bpp, 18 bpp packed, 18 bpp loosely packed (3 byte format), and 24bpp
- Interfaces
  - Complies with Protocol-to-PHY Interface (PPI) in 1.0 Gbps / 1.5 Gbps MIPI D-PHY
  - Supports RGB Interface for Video Image Input from general display controller
  - Supports S-i80 (Synchronous i80) Interface for Command Mode Image input from display controller
  - Supports PMS control interface for PLL to configure byte clock frequency
  - Supports Prescaler to generate escape clock from byte clock

13.4.1.2 Block diagram

13.4.2 External Signals

The following table describes the external signals of MIPI DSI:

Table 13-11. MIPI DSI External Signals

Signal	Description	Pad	Mode	Direction
MIPI_DSI_CLK_N	D-PHY Negative D-Phy differential clock line Receiver input	MIPI_DSI_CLK_N	No muxing	I
MIPI_DSI_CLK_P	D-PHY Positive D-Phy differential clock line Receiver input	MIPI_DSI_CLK_P	No muxing	I
MIPI_DSI_D0_N	D-PHY Negative D-Phy differential data line Receiver input	MIPI_DSI_D0_N	No muxing	I
MIPI_DSI_D0_P	D-PHY Positive D-Phy differential data line Receiver input	MIPI_DSI_D0_P	No muxing	I
MIPI_DSI_D1_N	D-PHY Negative D-Phy differential data line Receiver input	MIPI_DSI_D1_N	No muxing	I
MIPI_DSI_D1_P	D-PHY Positive D-Phy differential data line Receiver input	MIPI_DSI_D1_P	No muxing	I

13.4.3 Functional Description

13.4.3.1 Total system block diagram

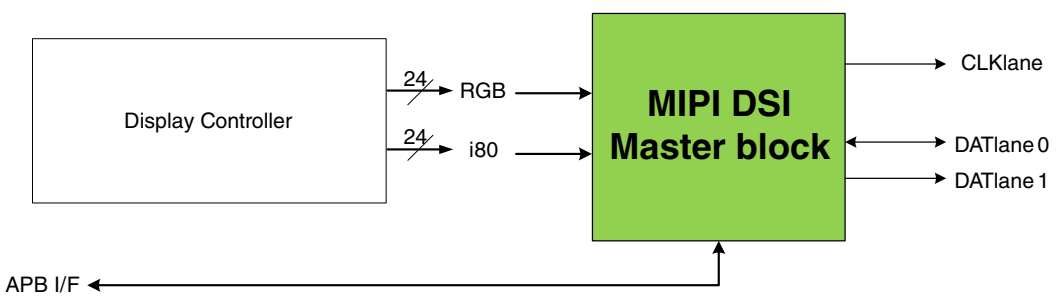


Figure 13-24. MIPI DSI Master System Block Diagram

13.4.3.2 MIPI DSI master and D-PHY I/F block diagram

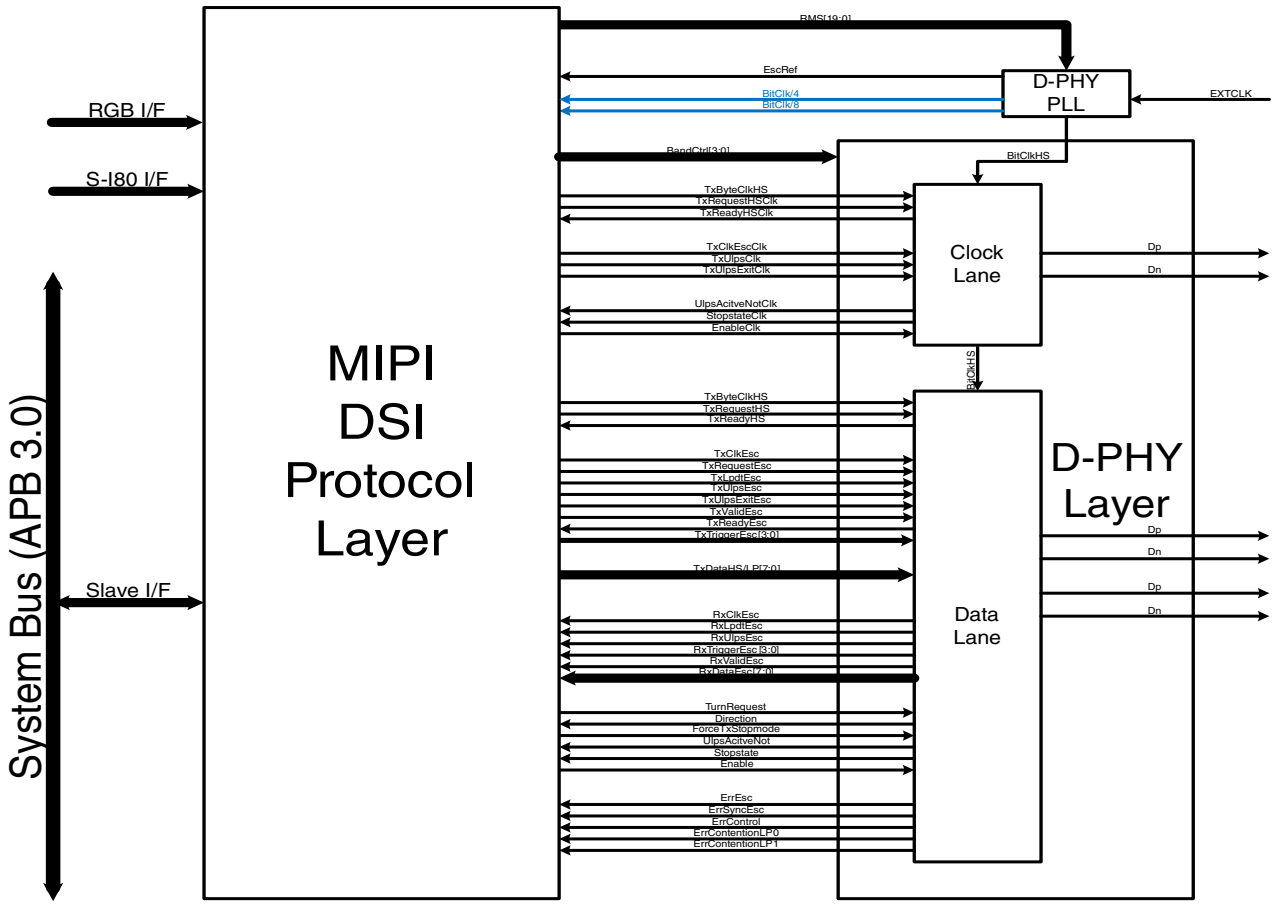


Figure 13-25. PPI I/F block diagram

13.4.3.3 MIPI DSI master block diagram

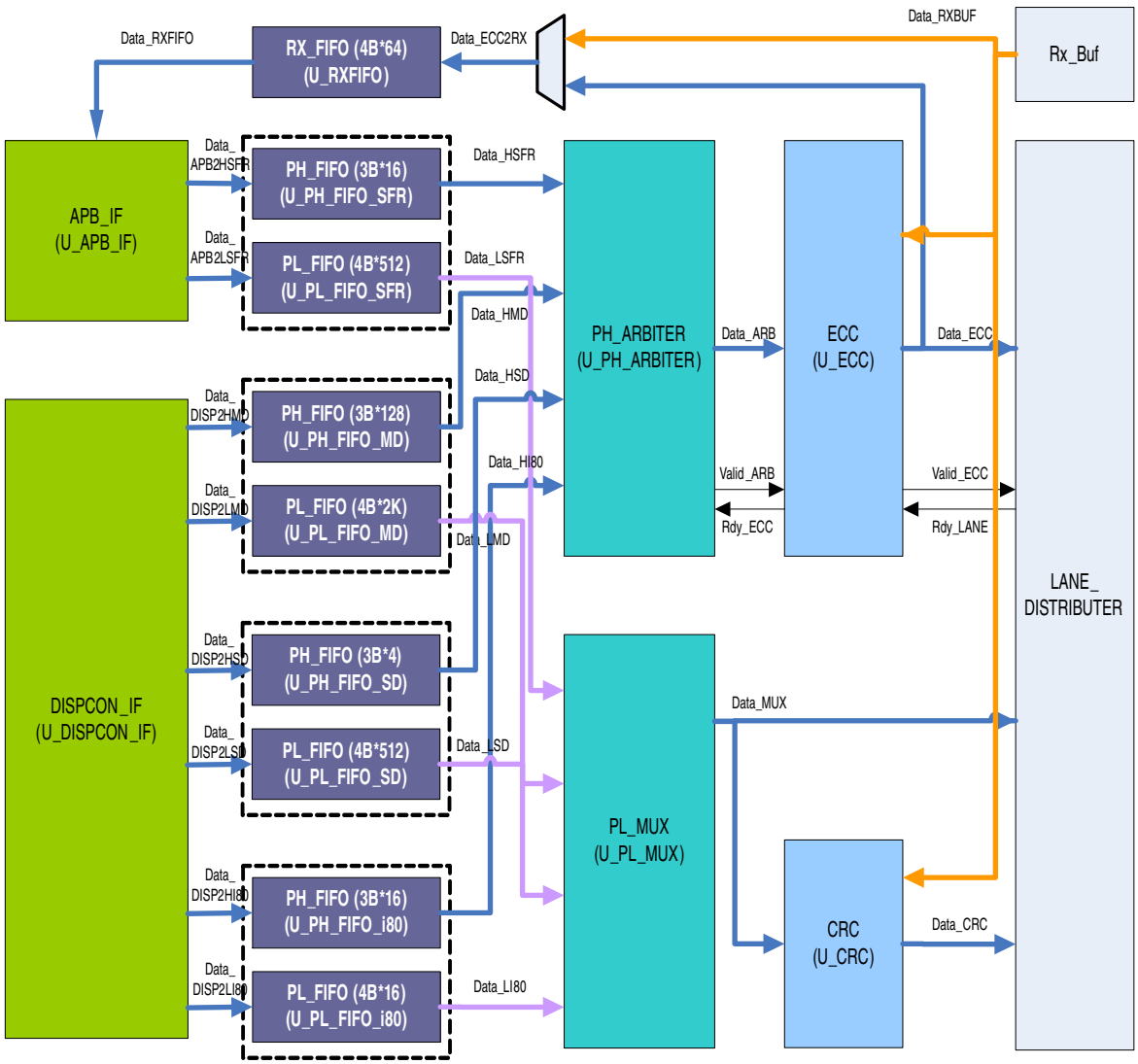


Figure 13-26. MIPI DSI Master Module Block Diagram

13.4.3.4 Internal primary FIFOs

The following table describes configurable-sized primary FIFOs.

Table 13-12. Internal Primary FIFO List

Port	FIFO Type	Size	Description
Main display for RGB I/F and S-i80 I/F	Packet Header FIFO	3 Bytes X 128 depth	Packet header FIFO for main display
	Payload FIFO	4 Bytes X 2048 depth	Payload FIFO for main display image

Table continues on the next page...

Table 13-12. Internal Primary FIFO List (continued)

Port	FIFO Type	Size	Description
			(SRAM used)
Sub display for S-i80 I/F image data	Packet Header FIFO	3 Bytes X 4 depth	Packet header FIFO for S-i80 I/F sub display
	Payload FIFO	4 Bytes X 512 depth	Payload FIFO for S-i80 I/F sub display image (SRAM used)
Command for S-i80 I/F command	Packet Header FIFO	3 Bytes X 16 depth	Packet header FIFO for S-i80 I/F command packet
	Payload FIFO	4 Bytes X 16 depth	Payload FIFO for S-i80 I/F command long packet payload

13.4.3.5 Packet header arbitration

There are four-packet headers FIFOs for Tx, namely, main display, sub display, S-i80 INTERFACE command, and SFR FIFO. The main and sub display FIFO packet headers contain the image data, while the S-i80 INTERFACE command FIFO packet header contains the command packets. On the other hand, the SFR FIFO packet header contains command packets, sub display image data (in Video mode), and so on.

The packet header arbiter has a “Fixed priority” algorithm. Priority order is fixed as main display, sub display, S-i80 INTERFACE command, and SFR FIFO packet header.

In the Video mode, sub display and S-i80 INTERFACE command FIFO are not used. The SFR FIFO packet header checks if the main display FIFO is empty (no request) in not-active image region and then sends its request.

13.4.3.6 RxFIFO structure

To read the packets received via low power data receiving mode, RxFIFO acts like an SFR. RxFIFO is an asynchronous FIFO with ByteClk and PCLK domains as input clock and output clock domains respectively. The Rx data is synchronized to RxClk. RXBUF has four Rx Byte buffers for aligning byte to word.

The packet headers of all the packets stored in RXFIFO are word-aligned, that is, the first byte of a packet is always stored in LSB. For example, if a long packet has 7-byte payload, the last byte is filled with dummy byte and the next packet is stored in the next word, as shown in the following figure.

NOTE

CRC data is not stored in RXFIFO.

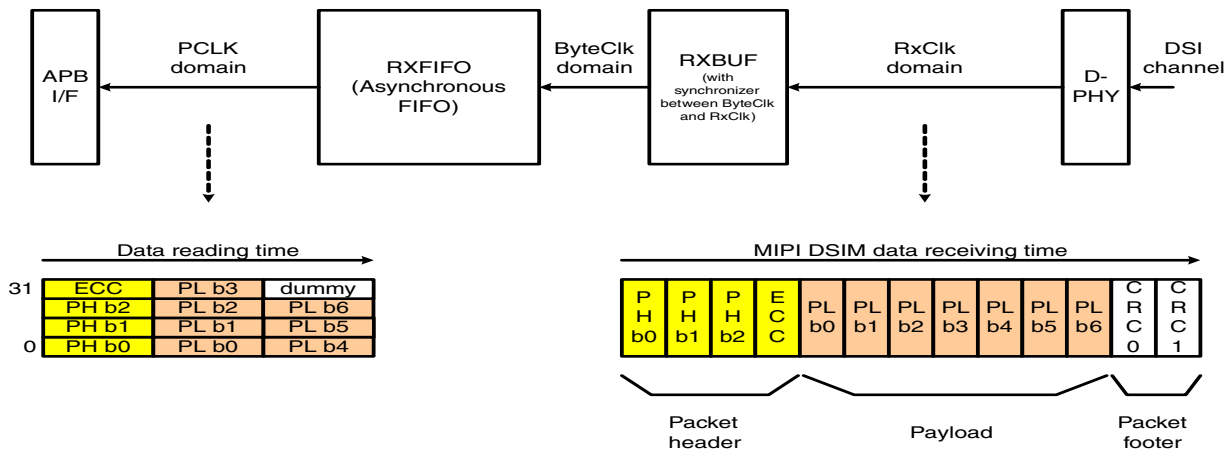


Figure 13-27. Rx Data Word Alignment

13.4.3.7 Interfaces and protocol

Display Controller-to-DSI Conceptual Signal Converting Diagram in Video Mode.

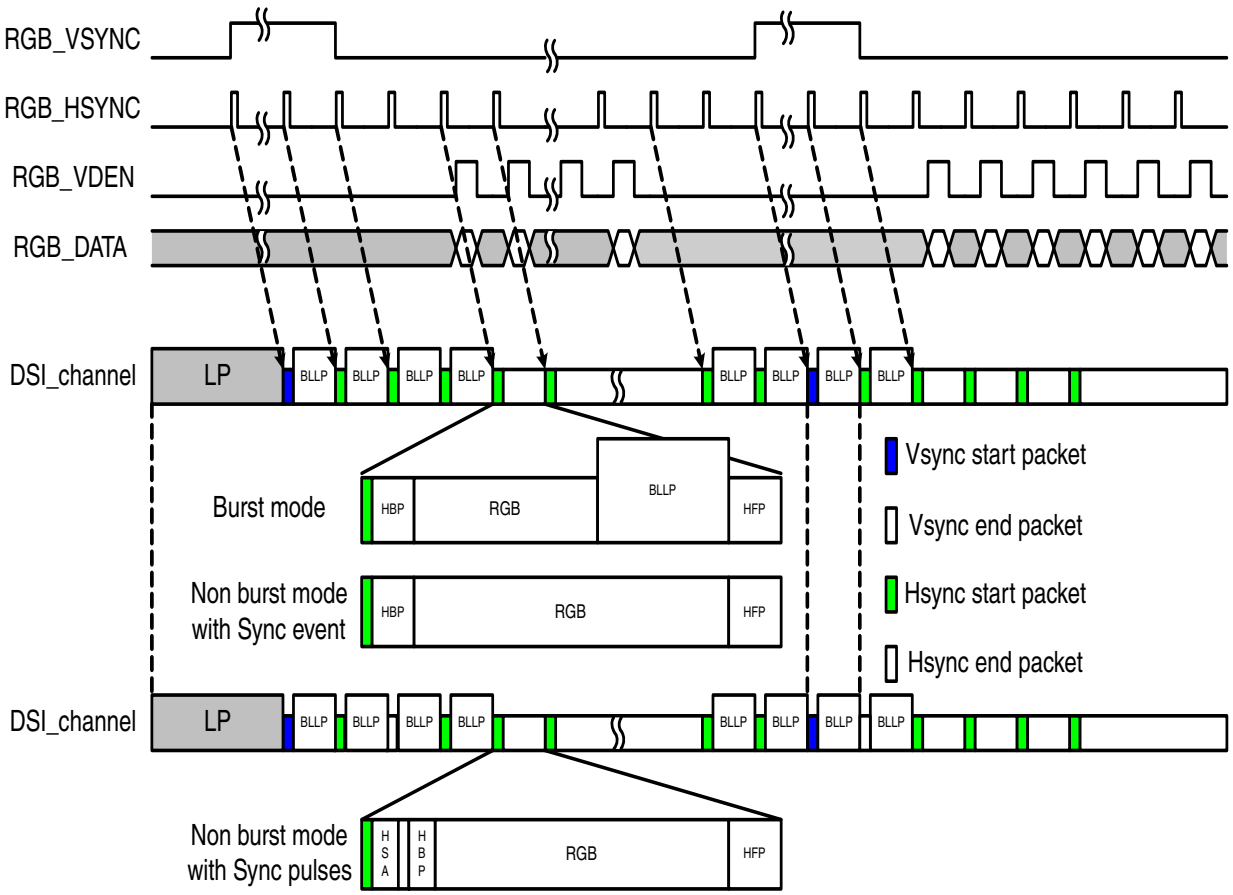


Figure 13-28. Signal Converting Diagram in Video Mode

13.4.3.8 Interface timing and protocol

13.4.3.8.1 Display controller interface

MIPI DSI Master has two-display controller interfaces, namely, RGB INTERFACE for main display and CPU INTERFACE (S-i80 INTERFACE) for main/ sub display. The Video mode uses RGB INTERFACE while the Command mode uses CPU INTERFACE.

The RGB image data is loaded on the data bus of RGB INTERFACE and S-i80 INTERFACE with the same order: RGB_VD[23:0] or SYS_VDOUT[23:0] is {R[7:0],G[7:0],B[7:0]}. Each byte aligns to the most significant bit. For instance, in the 12-bit mode, only three 4-bit values are valid as R, G, and B each, that is, data[23:20], data[15:12], and data[7:4]. The DSIM ignores rest of the bits.

13.4.3.8.2 RGB interface

Vsync, Hsync, and VDEN are active high signals. Among the three signals, Vsync and Hsync are pulse types that spend several video clocks. RGB_VD[23:0] is {R[7:0], G[7:0], B[7:0]}. All sync signals are synchronized to the rising edge of RGB_VCLK. The display controller sends minimum one horizontal line length of Vsync pulse, V back porch, and V front porch. Hsync pulse width should be longer than 1-byte clock cycle.

13.4.3.8.2.1 HSA disable mode

HSA mode specifies the Horizontal Sync Pulse area disable mode.

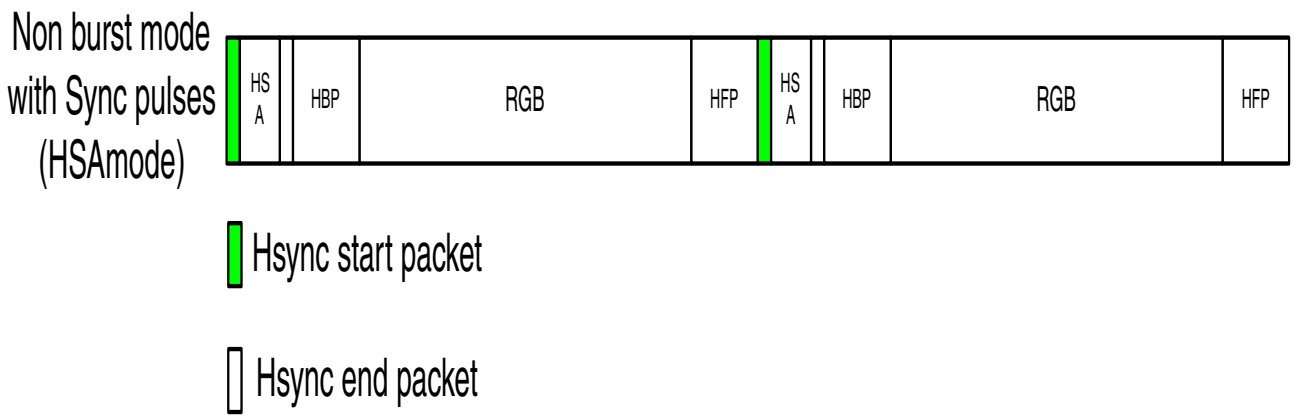


Figure 13-29. Block Timing Diagram of HSA Disable Mode (HSA Mode Reset : DSIM_CONFIG[20] = 0)

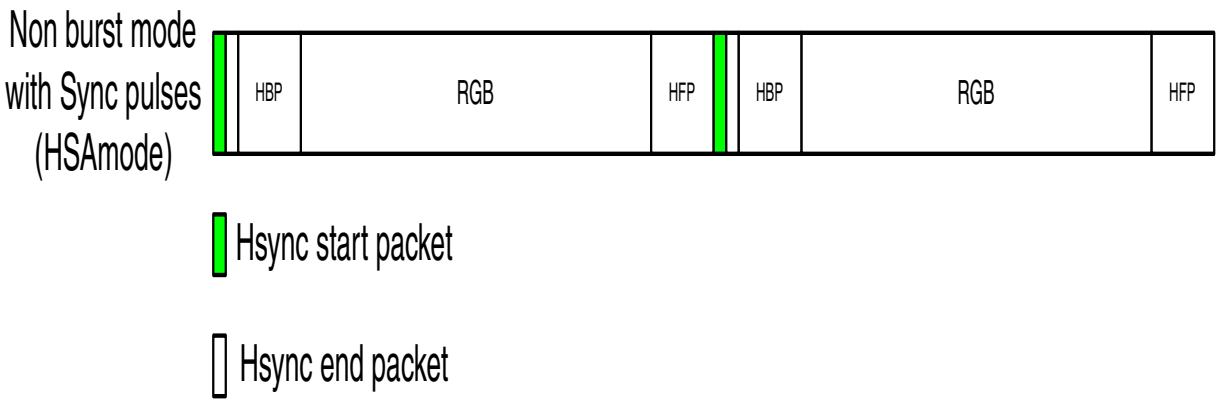


Figure 13-30. Block Timing Diagram of HSA Disable Mode (HSA Mode Reset : DSIM_CONFIG[20] = 1)

13.4.3.8.2.2 HBP disable mode

This is optional spec what is used only burst mode. HBP mode is Horizontal Back Porch disable mode.

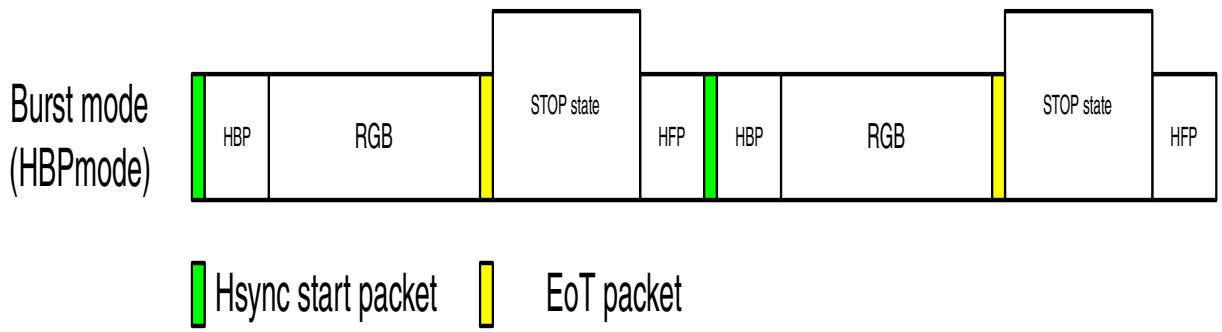
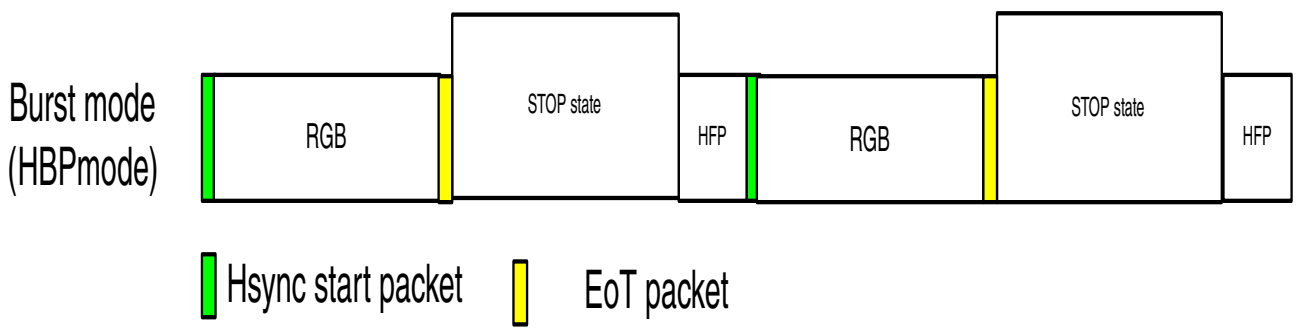


Figure 13-31. Block Timing Diagram of HBP Disable Mode (HBP Mode Reset : DSIM_CONFIG[21] = 0)

Figure 13-32. Block Timing Diagram of HBP Disable Mode (HBP Mode Set : DSIM_CONFIG[21] = 1)



13.4.3.8.2.3 HFP disable mode

This is optional spec what is used only burst mode. HFP mode is Horizontal Front Porch disable mode.

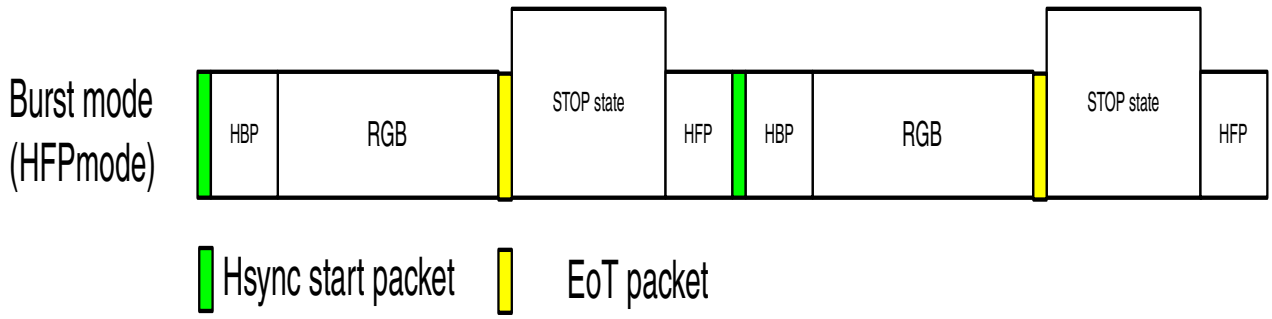


Figure 13-33. Block Timing Diagram of HFP Disable Mode (HFP Mode Set : DSIM_CONFIG[22] = 1)

13.4.3.8.2.4 HSE disable mode

HSE mode is Horizontal Sync End packet enable mode in Vsync pulse or Vporch area.

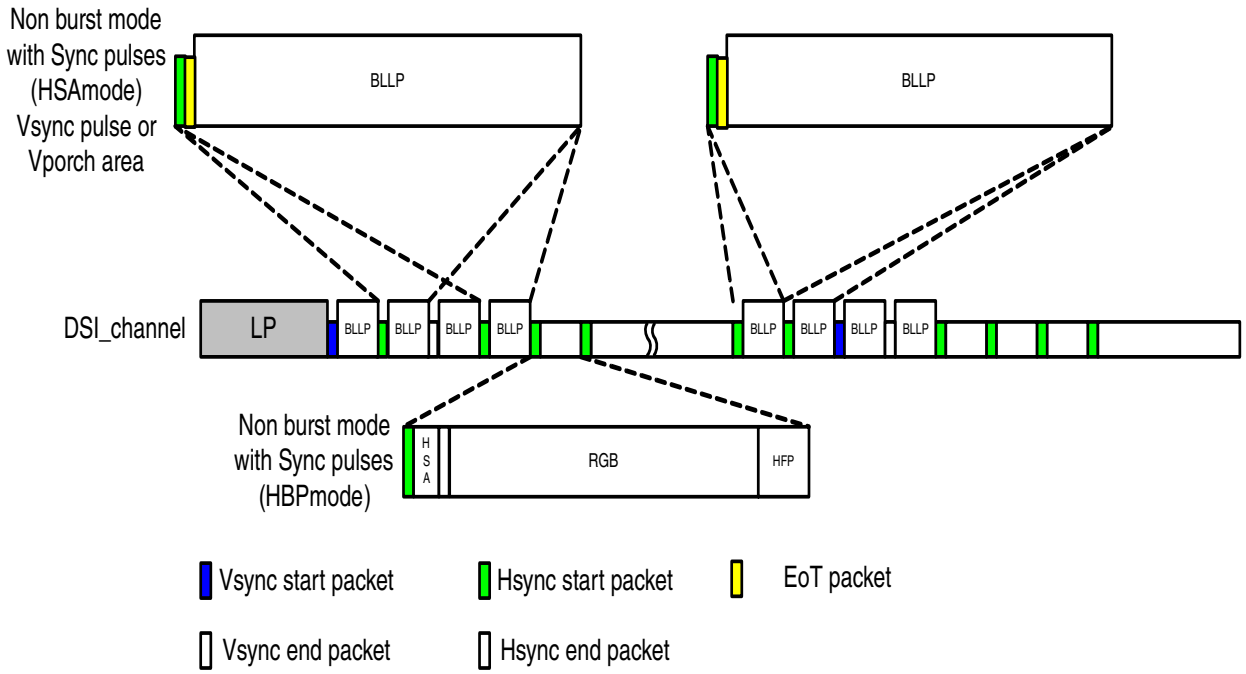


Figure 13-34. Block Timing Diagram of HSE Disable Mode (HSE Mode Reset : DSIM_CONFIG[23] = 0)

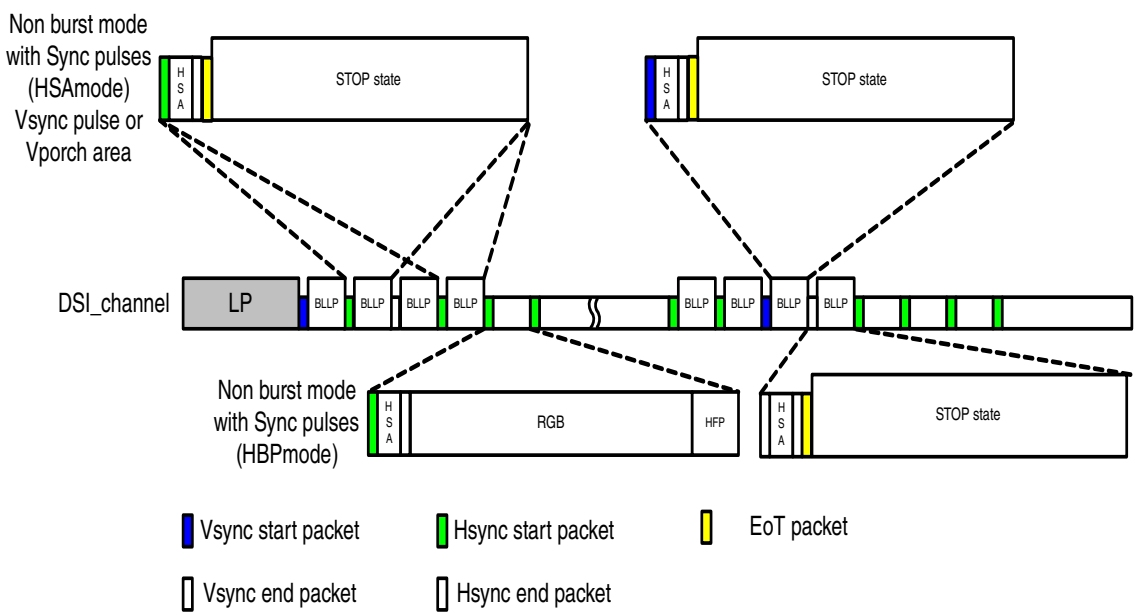


Figure 13-35. Block Timing Diagram of HSE Disable Mode (HSE Mode Reset : DSIM_CONFIG[23] = 1)

13.4.3.8.2.5 Transfer general data in video mode

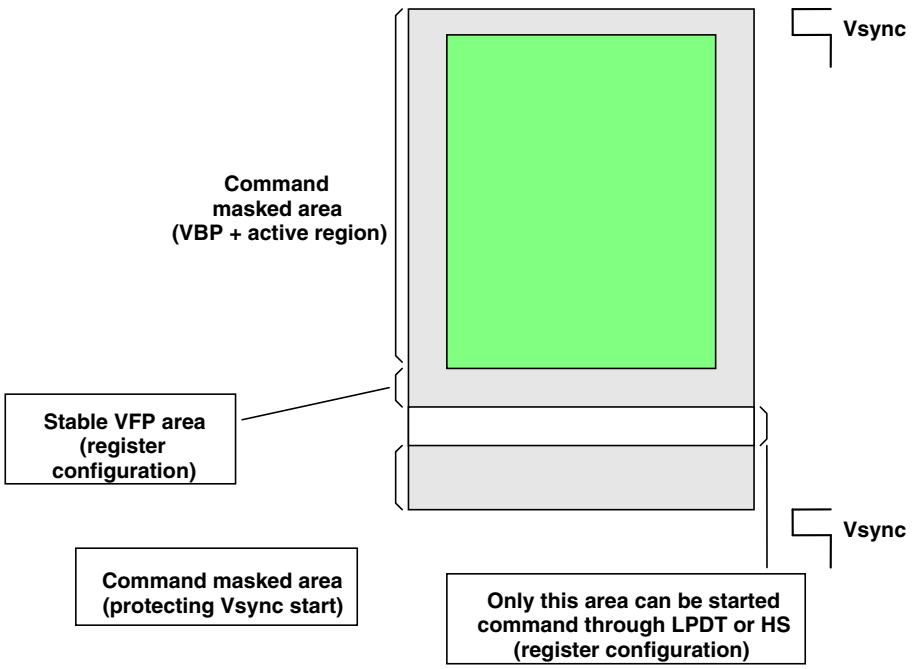


Figure 13-36. Stable VFP Area Before Command Transfer Allowing Area

MIPI DSIM Converts RGB Interface to Video Mode.

Vsync and Hsync packets are extremely important to protect image in Video mode. MIPI DSIM allows several lines in VFP area to transfer general data transfer. As shown in the figure above, the vertical front porch is divided into three areas, namely stable VFP area, command allowed area, and command masked area.

The register configures stable VFP area. Configuration boundary is 11'h000 ~ 11'h7FFF in DSIM_MVPORCH.

The register also configures the command allowed area. Configuration boundary is 4'h0 ~ 4'hF in DSIM_MVPORCH. Only this area is allowed to start "command transfer" through HS mode or LPDT. In LPDT, data transferring takes a long time to complete (approximately hundreds of microseconds or more). In this time, Hsync packet does not arrive due to LPDT long packet. MIPI DSIM comprises of big size FIFO for lost Hsync packet. After LPDT, MIPI DSIM transfers these Hsync packets immediately through HS mode.

To protect Vsync, command masked area is masked area. This area is calculated using LPDT bandwidth. For example, if EscClk is 10 MHz, the maximum long packet payload size is 1 KB and LPDT, LPDT transferring time is 824 µs (packet size: 1030 byte, LPDT maximum bandwidth: 10Mbps). If one line time is 20 µs, the line timing violation occurs in 42 lines. Therefore, command masked area is larger than 42 + α. This 'α' is transferring time of the violated Hsync packets.

Display controller should be configured in such a way that VFP lines are sum of stable VFP, command allowed area, and command masked area.

13.4.3.8.3 S-i80 interface

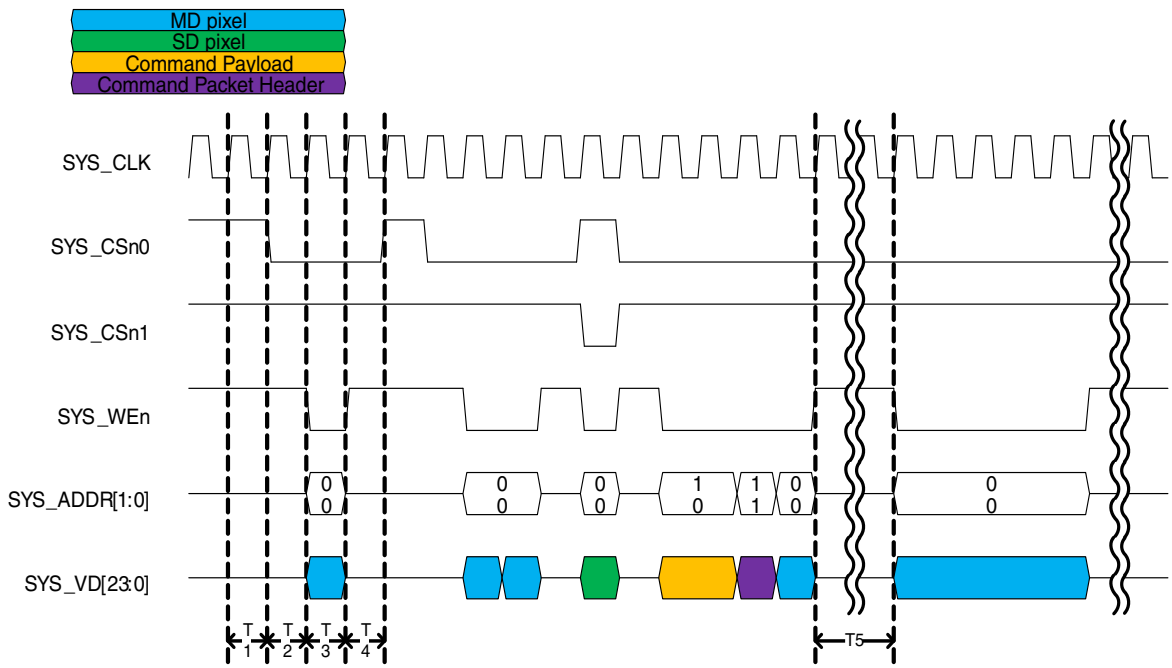


Figure 13-37. S-i80 I/F Timing Diagram

- T1 ≥ 1 clock cycle.
- T2 ≥ 0 clock cycle.
- T3 ≥ 1 clock cycle.
- T4 ≥ 0 clock cycle.
- T2 + T3 + T4 > 1 cycle of Byte clock
- T5 ≥ 10 clock cycles. (every horizontal blank period)

A display controller generates these signals with its internal clock: SYS_CS0 / CS1, SYS_WE, and SYS_VD. MIPI DSI master decodes the SYS_ADDR. The follwoing table describes the Command mode Interface address map.

Table 13-13. Command Mode Interface Address Map

SYS_ADDR[1:0]	Description
2'b00	Image data
2'b01	Reserved
2'b10	Payload data
2'b11	Packet Header

The following figure shows how MIPI DSI Master makes packet from the image data stream via S-i80 INTERFACE in Command mode. MIPI DSI master makes packet from the first line with DCS command "write_memory_start" and the other lines with DCS command "write_memory_continue".

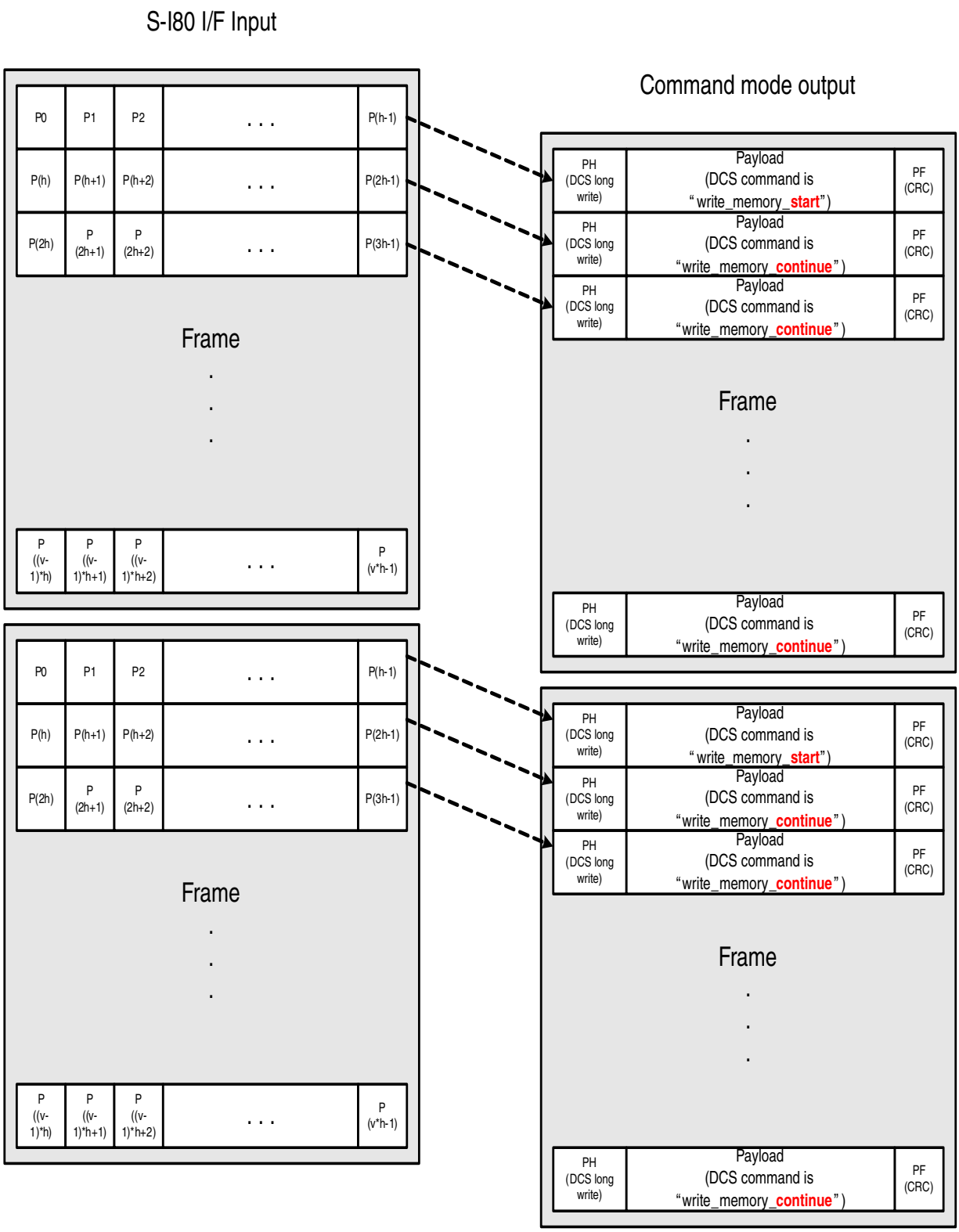


Figure 13-38. Packetizing for MIPI DSI Command Mode from S-i80 I/F

13.4.3.8.4 Relation between input transactions and DSI transactions

Table 13-14. Relation between Input Transactions and DSI Transactions

Input Interface	Input Transaction	DSI Transaction
RGB	RGB transaction	Specifies the RGB Packet
		888, 666, 666 (loosely packed), and 565 should be specified via register configuration.
RGB / S-i80	RGB / S-i80 Image Transaction	Specifies the Data type, that is, “DCS Long Write packet”. (DCS command is “memory write start/continue”.)
S-i80	S-i80 Command Transaction	Specifies any DSI packet. Bytes in S-i80 transaction should be the same bytes in DSI packets.
SFR	Header and Payload FIFO access	Specifies any DSI Packets. Bytes in APB transaction should be the same bytes in DSI packets.

13.4.3.8.5 PPI interface timing and protocol

13.4.3.8.5.1 Initialization after power-on / reset

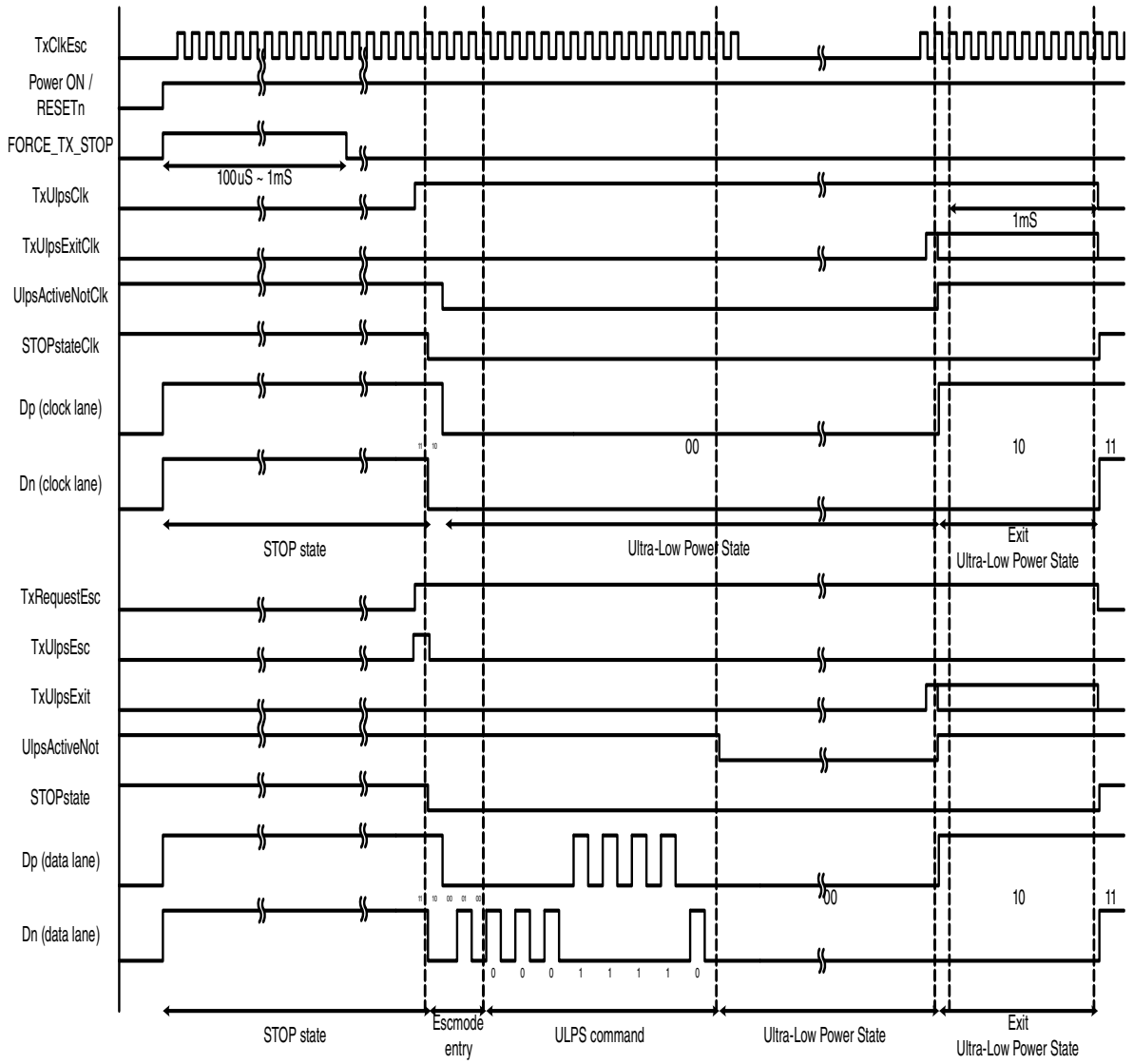


Figure 13-39. Timing Diagram of Initialization

13.4.3.8.5.2 High speed data transfer

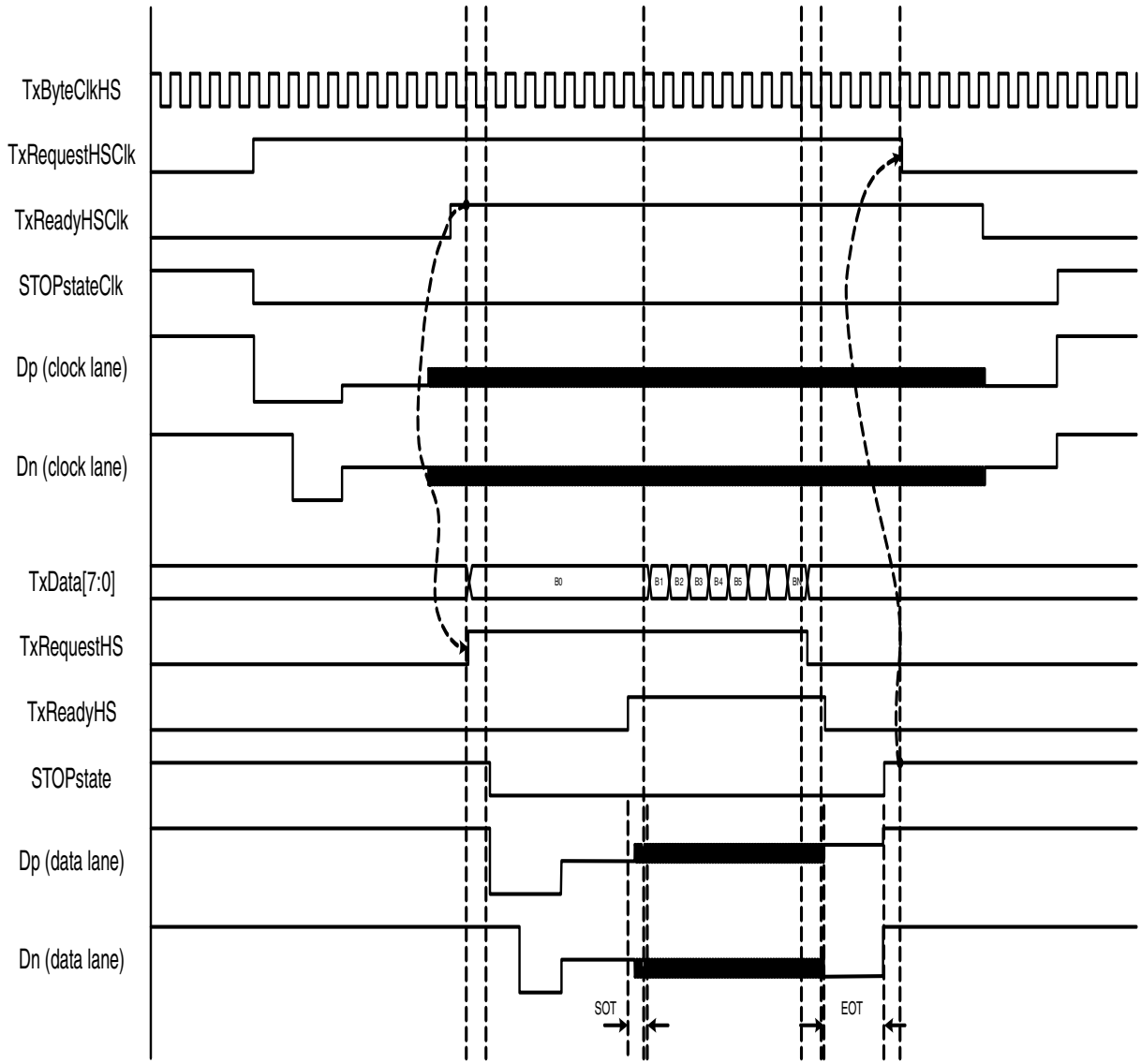


Figure 13-40. Timing Diagram of High Speed Data Transfer

13.4.3.8.5.3 Low power data transfer

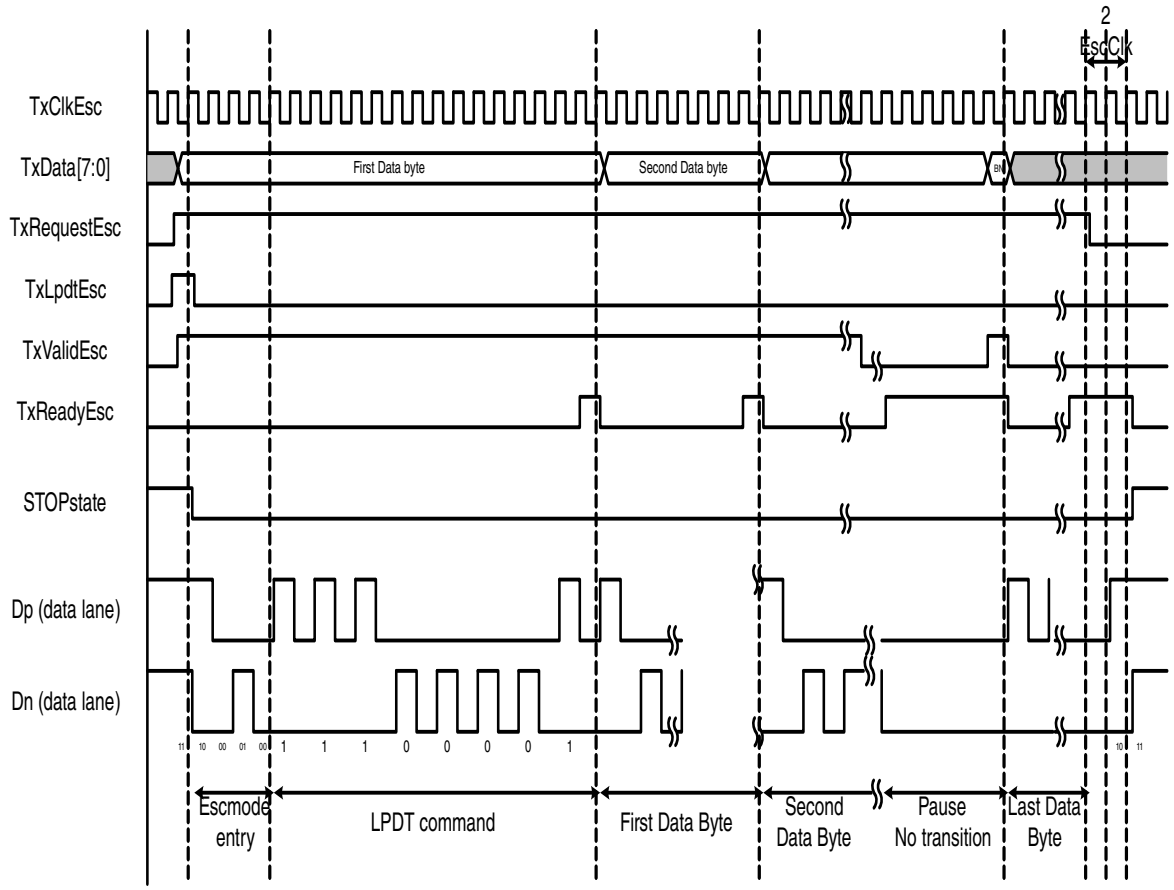


Figure 13-41. Timing Diagram of Low Power Data Transfer

13.4.3.8.5.4 Ultra-Low power state

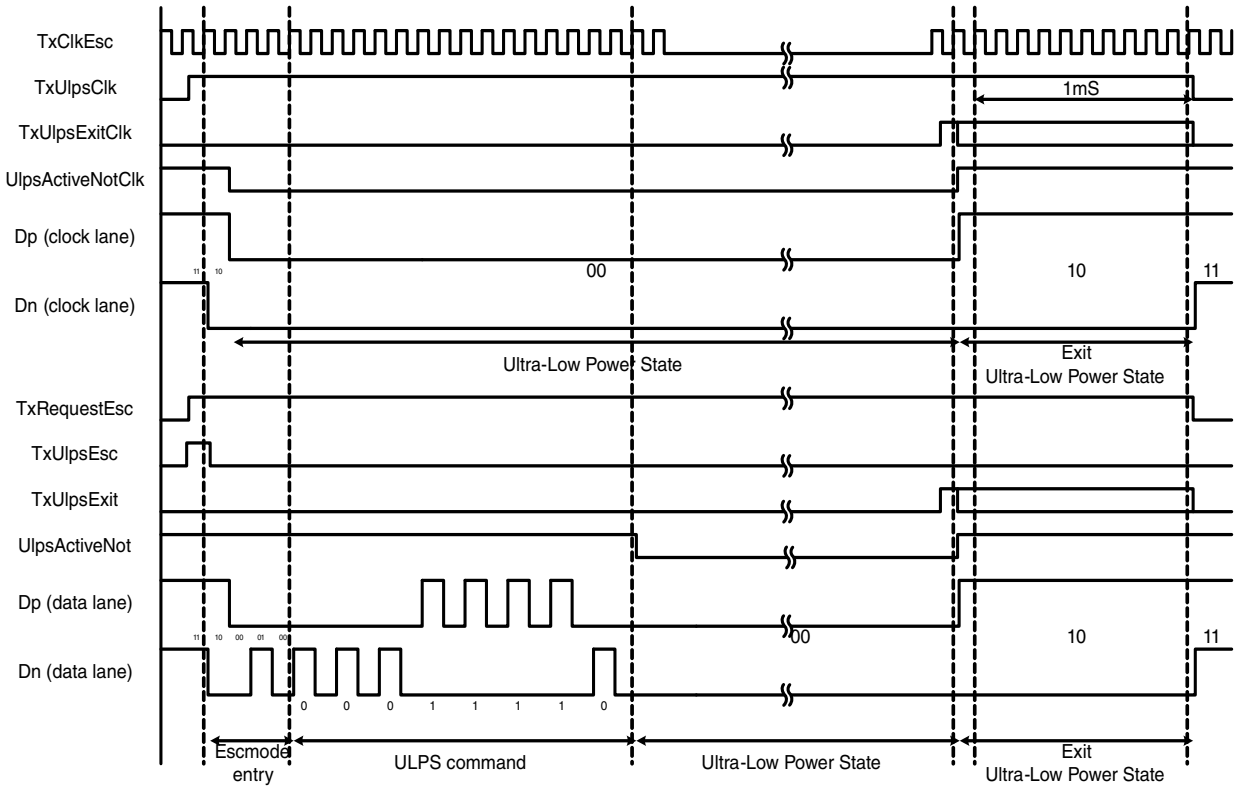


Figure 13-42. Timing Diagram of Ultra Low Power State

13.4.3.8.5.5 Trigger function

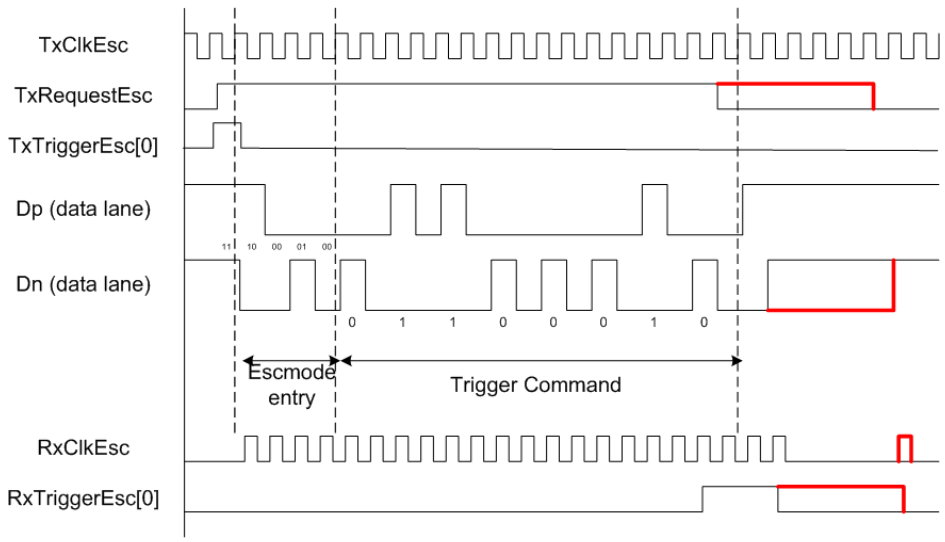


Figure 13-43. Diagram of Trigger Function “Remote Reset”

There are four types of trigger commands:

- 01100010 : remote reset – forward direction
- 01011101 : Tearing Effect – reverse direction
- 00100001 : Acknowledge – reverse direction
- 10100000 : Unknown

If DSI D-PHY received these signals, D-PHY indicates trigger function to protocol layer.

13.4.3.8.5.6 Turn-around and low power data receiving

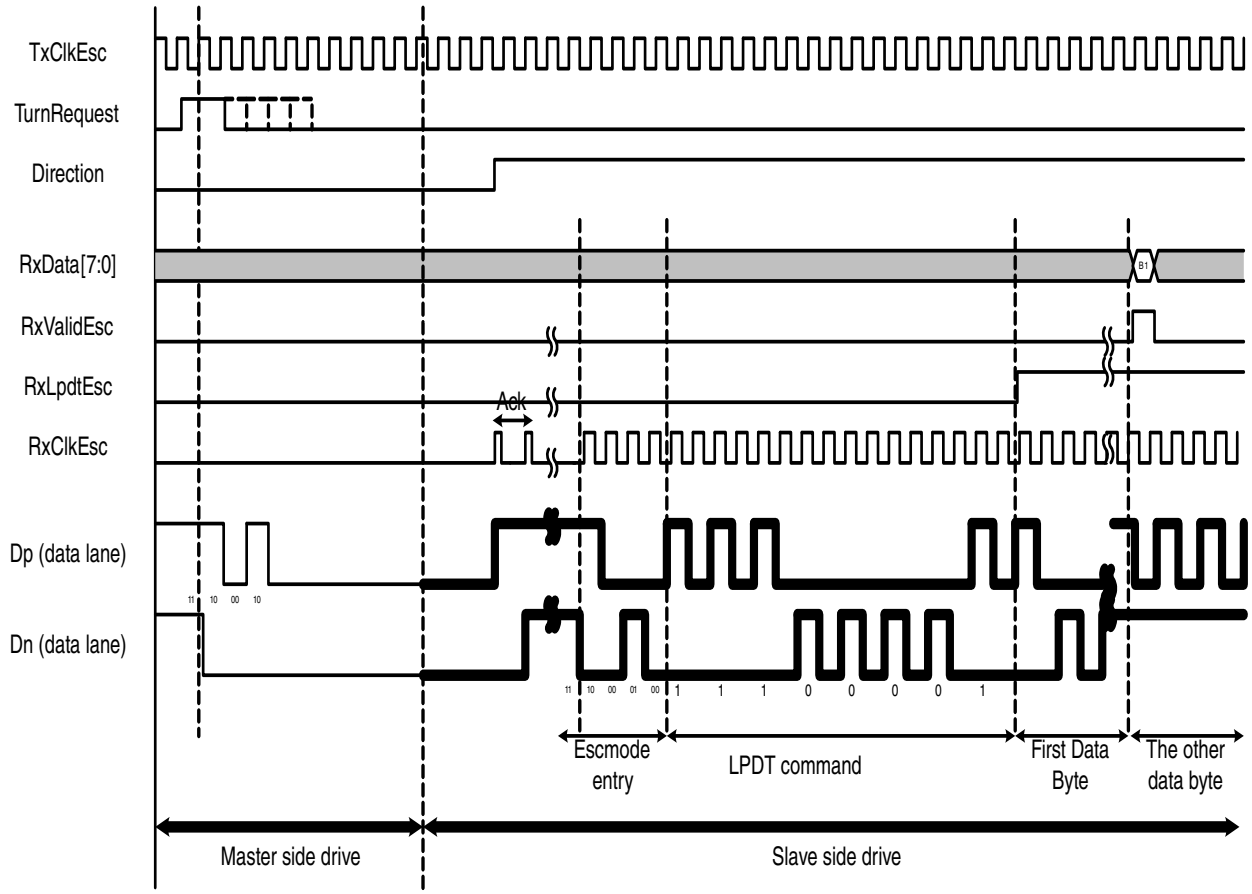


Figure 13-44. Timing Diagram of Bus-Turn-Around

13.4.3.8.5.7 TE signaling

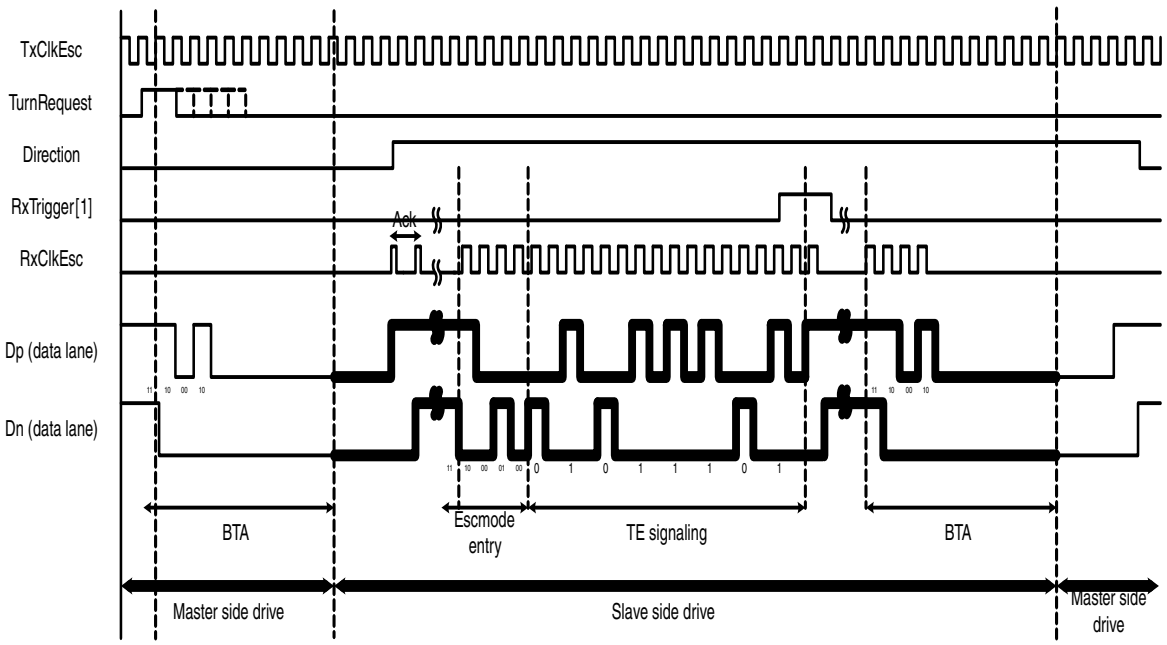


Figure 13-45. TE Signaling

13.4.3.9 Configuration

13.4.3.9.1 Video mode VS Command mode

MIPI DSI Master Block supports two modes, namely, Video mode and Command mode.

Command mode uses RGB interface and S-i80 interface. Mode can change via SFRs.

13.4.3.9.2 Dual display VS Single display

13.4.3.9.2.1 Dual display

MIPI DSI Master supports dual display configuration in Command mode only, that is, both main and sub display image should be transmitted via S-i80 INTERFACE.

13.4.3.9.2.2 Single display

For single display configuration, use video mode or command mode.

13.4.3.9.3 PLL

To transmit Image data, MIPI DSI Master Block needs high frequency clock (80 MHz ~ 1.5 GHz) generated by PLL.

To configure PLL, MIPI DSI Master comprises of SFRs and corresponding interface signals. PLL is embedded in PHY module. You should use other PLL in SoC if it meets the timing specification.

13.4.3.9.4 Buffer

In MIPI DSI standard specification, DSI Master sends image stream in burst mode. The image stream transmits in high-speed and bit-clock frequency. This mode allows the device to stay in stop state longer to reduce power consumption. For this mode, MIPI DSI Master has a line buffer to store one complete line and send it faster at the next line time.

13.4.3.10 Application Scenario

13.4.3.10.1 Display mode

13.4.3.10.1.1 Video mode

Video mode starting sequence:

1. Set RGB I/F in Display Controller (RGB888), also set Video mode at MIPI DSI Master
2. Configure all parameters for displaying image (define resolution, back porch, front porch, data lane number, HS clock frequency (byte clock), etc).
3. Start image transmission.

13.4.3.10.1.2 Command mode

Command mode starting sequence:

1. Set S-i80 I/F in Display Controller (RGB888), also set Command mode at MIPI DSI Master
2. Configure all parameters for displaying image.
3. Set the command at MIPI DSI registers using MIPI DCS.

4. S-i80 I/F is just used to transmit image data. The users have to read display peripheral status and write command through MIPI DSI registers.

13.4.3.10.2 Programming model

Table 13-15. Behavior Programming Each Case

No.	Mode	Case	Pseudo Code

1	Video	Initialize	1. reset or power on
			2. SFR_Write(Display Controller_registers, RGB I/F_888);
			3. SFR_Write(DSIM_CLKCNT,{16'h1000,Esc_prescaler}); //EscClk = ByteClk/Esc_prescaler;
			4. SFR_Write(DSIM_MDRESOL,{M_vert,M_horizon});
			5. SFR_Write(DSIM_MVPORCH,M_Vback);
			6. SFR_Write(DSIM_MHPORCH,{M_Hfront,M_Hback});
			7. SFR_Write(DSIM_MBLANK,{M_SBLLP,M_LBLLP});
			8. SFR_Write(DSIM_MSYNC,{M_VSA,M_HSA}); // In SyncEvent mode, Ignore it
			9. SFR_Write(DSIM_CONFIG,{HFPdismode, MainVC,MPix, SubVC,SPix,Sync_inform,BLLP,Burst, Videomode(1),NumofDatlane,Lane_EN); // set the value of each bit
			10. SFR_Write(DSIM_PLLCTRL, {4'h0, Freq_band,4'h8,PMS}); // PLL enable
			11. Wait for 100µS ~ 1mS using system timer after reset or power on. In this time, ForceSTOPstate at DSIM_CONFIG is assigned to D-PHY by default. 100µS ~ 1mS is not mendatory. Please see DSI slave spec
			12. SFR_Write(DSIM_CONFIG, ForceSTOPstate(0)); // disable
			13. While(~SFR_Read(DSIM_STATUS, StopstateClk, StopstateDat)); D-PHY & protocol layer(MIPI DSI Master block) in STOP state
			14. SFR_Write(DSIM_ESCMODE, TxUlps(1),TxUlpsClk(1)); D-PHY & protocol layer(MIPI DSI Master block) in ULPS In ULPS, User can turn off Byte/Esc clock for Low power estimation SFR_Write(DSIM_CLKCNT, ESCClk_EN(0), BytClk_En(0)); Or SFR_Write(DSIM_CLKCNT,ByteClk_EN(0),Lane_EscClkEn[4 :0](0));
			If user wants to turn on display, Turn on EscClk and
			15. SFR_Write(DSIM_ESCMODE, TxUlpsExit(1),TxUlpsClkExit(1));
			16. While(~SFR_Read(DSIM_STATUS, UlpsClk, UlpsDat));
			17. SFR_Write (DSIM_ESCMODE, TxUlpsExit (0), TxUlpsClkExit (0));
			18. Wait for about 1mS using system timer. 1mS is not mendatory, Just example. Please see DSI slave spec.
			19. SFR_Write (DSIM_ESCMODE, TxUlps (0),TxUlpsClk (0));
			20. While (~SFR_Read(DSIM_STATUS, StopstateClk, StopstateDat)); D-PHY & protocol layer (MIPI DSI Master block) in STOP state
			21. MIPI DSI Master is ready for HS/LP operation.

2	Video	HS Data Transfer	In Video mode, there is no pseudo code. After initializing, SFR_Write (Display Controller_register, START); If line image is full in line buffer, MIPI DSI master transmit data through long data packet. If user wants to send general command to display or transmit image to sub display from memory, user will have to set following sequence. 1. SFR_Write (DSIM_PAYLOAD,PAYLOAD_DATA);

Table continues on the next page...

Table 13-15. Behavior Programming Each Case (continued)

No.	Mode	Case	Pseudo Code
			2. SFR_Write (DSIM_PKTHDR,PKTHDR_DATA); If user wants to send command to Sub display, User shall set “Sub display Virtual Channel field” in Packet Header (PKTHDR[7:6]).
3	Video	LP Data Transfer	If user wants to send general command to display or transmit image to sub display from memory via LPDT, 1. SFR_Write (DSIM_ESCMODE, TxLpdt); 2. If DSIM transfer data in HS mode, DSIM will wait STOPstate and transfer mode change HS to LP. 3. Same way previous test scenario cases. If user wants to send command to Sub display, User shall set “Sub display Virtual Channel field” in Packet Header (PKTHDR[7:6]).
4	Video	ULPS	ULPS sequence is same the initialization process(test scenario #1) 13 ~20
5	Video	Trigger function	Follow the process : 1. SFR_Write (DSIM_ESCMODE, TxTriggerRst); 2. If DSI master state is not STOP state, DSIM will wait for LP stop mode and transmit trigger automatically
6	Video	BTA	There are two cases of BTA. One case is after transferring general read data packet or DCS command write(set_tear_on) and the other on is BTA_req bit setting by software. Previous case is automatically setting BTA. Next case is following sequence. If user reads the state or data of display panel, user will use BTA. 1. SFR_Write(DSIM_ESCMODE, BTA); 2. If DSIM transfer data, DSIM will wait STOPstate. 3. After BTA assert, DSIM release DSI channel. In reverse direction (DSI slave has DSI channel grant), all DSIM FIFO is flush and masked Video / SFR inputs. Only LPRX data input is unmasked. 4. If DSIS release DSI channel to DSIM, Int_BusTurnOver is generated. After ISR, DSIM can transfer data.

7	Command	Initialize	1. Reset or power on
			2. SFR_Write(Display Controller_registers, S-i80 I/F_888);
			3. SFR_Write(DSIM_CLKCNT,{16’h1000,Esc_prescaler});
			4. SFR_Write(DSIM_MDRESOL,{M_vert,M_horizon});
			5. SFR_Write(DSIM_SDRESOL,{S_vert,S_horizon});
			6. SFR_Write(DSIM_CONFIG,{ MainVC,MPix,SubVC,SPix, Sync_inform,NumofDatlane,Videomode(0), Lane_EN); // set the value of each bit
			7. SFR_Write(DSIM_PLLCTRL, {4’h0, Freq_band,4’h8,PMS}); // PLL enable
			8. Wait for 100μS ~ 1mS using system timer after reset or power on. In this time, ForceSTOPstate at DSIM_CONFIG is assigned to D-PHY by default. 100μS ~ 1mS is not mendatory. Please see DSI slave spec.
			9. SFR_Write(DSIM_CONFIG, ForceSTOPstate(0)); // disable
			10. While(~SFR_Read(DSIM_STATUS, StopstateClk, StopstateDat)); D-PHY & protocol layer(MIPI DSI Master block) in STOP state
			11. SFR_Write(DSIM_ESCMODE, TxUlps(1),TxUlpsClk(1)); D-PHY & protocol layer(MIPI DSI Master block) in ULPS In ULPS, User can turn off Byte/Esc clock for Low power estimation SFR_Write(DSIM_CLKCNT,

Table continues on the next page...

Table 13-15. Behavior Programming Each Case (continued)

<table>
<tr><th>No.</th><th>Mode</th><th>Case</th><th>Pseudo Code</th></tr>
<tr><td></td><td></td><td></td><td>ESCClk_EN(0), BytClk_En(0)); Or
SFR_Write(DSIM_CLKCNT,ByteClk_EN(0),Lane_EscClkEn[4 :0](0));

If user wants to turn on display, Turn on EscClk and

12. SFR_Write(DSIM_ESCMODE, TxUlpsExit(1),TxUlpsClkExit(1));
13. While(~SFR_Read(DSIM_STATUS, UlpsClk, UlpsDat));
14. SFR_Write(DSIM_ESCMODE, TxUlpsExit(0), TxUlpsClkExit(0));
15. Wait for about 1mS using system timer. 1mS is not mendatory, Just example. Please see DSI slave spec.
16. SFR_Write(DSIM_ESCMODE, TxUlps(0),TxUlpsClk(0));
17. While(~SFR_Read(DSIM_STATUS, StopstateClk, StopstateDat)); D-PHY & protocol layer(MIPI DSI Master block) in STOP state MIPI DSI Master is ready for HS / LP operation.</td></tr>
<tr><td>8</td><td>Command</td><td>HS Data Transfer</td><td>In Command mode, there is no pseudo code.

After initializing, SFR_Write(Display Controller_register, START);

If line image is full in line buffer, MIPI DSI master transmit data through long data packet.

<table>
<tr><th>SYS_ ADD[1]</th><th>SYS_ ADD[0]</th><th>SYS_ CS0</th><th>SYS_ CS1</th><th>Contents</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>Main display Image</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>Sub display Image</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>Payload data</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>Packet Header data User can control main display and sub display via virtual channel field in Packet Header. (Virtual channel field : PKTHDR[7:6])</td></tr>
<tr><td colspan="4">The other cases</td><td>Reserved</td></tr>
</table>

If user wants to send general command to display or transmit image to sub display from memory, user will have to set following sequence.

SFR_Write(DSIM_PAYLOAD,PAYLOAD_DATA);
SFR_Write(DSIM_PKTHDR,PKTHDR_DATA);

If user wants to update partial image, wait after Display Controller transfer last line. wait(DSIM_INTSRC, Frame_done); or check STOPstate.
SFR_Write(DSIM_MDRESOL,{M_vert,M_horizon}); or
SFR_Write(DSIM_SDRESOL,{S_vert,S_horizon}); Turn on Display Controller for Image transfer</td></tr>
</table>

Table continues on the next page...

Table 13-15. Behavior Programming Each Case (continued)

No.	Mode	Case	Pseudo Code
			If user wants to send command to Sub display, User shall set “Sub display Virtual Channel field” in Packet Header(PKTHDR[7:6]).
9	Command	LP Data Transfer	If user wants to send general command to display or transmit image to sub display from memory via LPDT, SFR_Write(DSIM_ESCMODE, TxLpdt); If DSIM transfer data in HS mode, DSIM will wait STOPstate and transfer mode change HS to LP. Same way previous test scenario cases. If user wants to send command to Sub display, User shall set “Sub display Virtual Channel field” in Packet Header(PKTHDR[7:6]).
10	Command	ULPS	ULPS sequence is same the initialization process(test scenario #1) 13 ~20
11	Command	Trigger function	Follow the process : SFR_Write(DSIM_ESCMODE, TxTriggerRst); If DSI master state is not STOP state, DSIM will wait for LP stop mode and transmit trigger automatically.
12	Command	BTA	There are two cases of BTA. One case is after transferring general read data packet or DCS command write(set_tear_on) and the other on is BTA_req bit setting by software. Previous case is automatically setting BTA. Next case is following sequence. If user read the state or data of display panel, user will use BTA. SFR_Write(DSIM_ESCMODE, BTA); If DSIM transfer data, DSIM will wait STOPstate. After BTA assert, DSIM release DSI channel. In reverse direction (DSI slave has DSI channel grant), all DSIM FIFO is flush and masked Video/SFR inputs. Only LPRX data input is unmasked. If DSIS release DSI channel to DSIM, “Int_BusTurnOver” is generated. After ISR, DSIM can transfer data.

13	ISR (Error)		SFR_Write(Global_INTC_MASK, INTMASK(DSIM));
			SFR_Read(Global_INTC_SRC);
			SFR_Write(DSIM_INTMSK, all_int_mask);
			SFR_Read(DSIM_INTSRC);
			Error reporting to application layer SFR_Write(DSIM_INTSRC, intsrc); // clear interrupt source
			SFR_Write(DSIM_INTMSK, all_int_unmask);
			SFR_Write(Global_INT_SRC, intsrc); // clear interrupt source
			SFR_Write(Global_INT_MASK, INTUNMASK(DSIM));
			SFR_Write(DSIM_CONFIG, ForceSTOPstate(1)); enough time later
			SFR_Write(DSIM_CONFIG, ForceSTOPstate(0));

14	ISR (Rx Data)		After BTA, wait interrupt “Int_BusTurnOver”. And following SFR_Write(Global_INTC_MASK, INTMASK(DSIM)); SFR_Read(Global_INTC_SRC); SFR_Write(DSIM_INTMSK, all_int_mask);

Table continues on the next page...

Table 13-15. Behavior Programming Each Case (continued)

No.	Mode	Case	Pseudo Code

			SFR_Read(DSIM_INTSRC); SFR_Write(DSIM_INTSRC, {BusTurnOver,RxDone}); // clear interrupt source
			SFR_Write(DSIM_INTMSK, all_int_unmask);
			SFR_Write(Global_INT_SRC, intsrc); // clear interrupt source
			SFR_Write(Global_INT_MASK, INTUNMASK(DSIM));
			SFR_Read(DSIM_RXFIFO);

15	ISR (trigger flag)		After BTA, wait interrupt “Int_BusTurnOver”. And following
			SFR_Write(Global_INTC_MASK, INTMASK(DSIM));
			SFR_Read(Global_INTC_SRC);
			SFR_Write(DSIM_INTMSK, all_int_mask);
			SFR_Read(DSIM_INTSRC); flag reporting to application layer
			SFR_Write(DSIM_INTSRC, {BusTurnOver, trigflags}); // clear interrupt source
			SFR_Write(DSIM_INTMSK, all_int_unmask);
			SFR_Write(Global_INT_SRC, intsrc); // clear interrupt source
			SFR_Write(Global_INT_MASK, INTUNMASK(DSIM));

16	Software reset	Function reset	If you want to assert Software reset, all clock must be turned on because software reset is synchronous reset. Turn on RGB_VCLK only from Display controller SFR_Write(DSIM_PLLTMR, PllTimer); SFR_Write(DSIM_CLKCTRL, {EscClkEn, EscPrescaler}); SFR_Write(DSIM_PLLCTRL, {PllEn,PMS}); wait Pll stable (polling status register or ISR) SFR_Write(DSIM_SWRST, FuncRst); wait software reset release (polling status register or ISR) ready for operation
		Software reset	If you want to assert Software reset, all clock must be turned on because software reset is synchronous reset. Turn on RGB_VCLK only from Display controller SFR_Write(DSIM_PLLTMR, PllTimer); SFR_Write(DSIM_CLKCTRL, {EscClkEn, EscPrescaler}); SFR_Write(DSIM_PLLCTRL, {PllEn,PMS}); wait Pll stable (polling status register or ISR) SFR_Write(DSIM_SWRST, SwRst); wait software reset release (polling status register or ISR) re-configure all registers ready for operation
17	Clock source changing from PLL to external clock source		Turn on RGB_VCLK (only clock) from Display controller and STOP data inputing to MIPI DSIM SFR_Write(DSIM_SWRST, FuncRst); or SFR_Write(DSIM_SWRST, SwRst);

Table continues on the next page...

Table 13-15. Behavior Programming Each Case (continued)

No.	Mode	Case	Pseudo Code
			Turn off any inputs in MIPI DSIM. SFR_Write(DSIM_PLLCTRL, PLL disable); SRF_Write(DSIM_CLKCNTRL, ByteClkSel); SFR_Write(DSIM_SWRST, FuncRst); or SFR_Write(DSIM_SWRST, SwRst); reconfiguration Turn on Display controller or input data to MIPI DSIM ready for operation
18	Usage of EoT packet when user want to transfer command packet through HS mode.		Video mode after turning on display controller FSM of Display controller I/F of MIPI DSIM generates EoT packet automatically Just write command or general packet in SFR FIFO Video mode after turning off/before turning on display controller Write command or general packet in SFR FIFO If user doesn't want to transfer general or command packet, user should write EoT packet in SFR FIFO. If general or command packets are transferred through LP mode, previous programming step is "don't care".

13.4.4 Register Description

MIPI_DSI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3076_0000	Version Register (MIPI_DSI_VERSION)	32	R/W	0106_0100h	13.4.4.1/ 3693
3076_0004	MIPI_DSI_STATUS	32	R/W	0010_010Fh	13.4.4.2/ 3694
3076_0008	RGB Status Register (MIPI_DSI_RGB_STATUS)	32	R/W	0000_0001h	13.4.4.3/ 3697
3076_000C	MIPI_DSI_SWRST	32	R/W	0000_0000h	13.4.4.4/ 3698
3076_0010	Clock Control Register (MIPI_DSI_CLKCTRL)	32	R/W	0000_FFFFh	13.4.4.5/ 3700
3076_0014	MIPI_DSI_TIMEOUT	32	R/W	00FF_FFFFh	13.4.4.6/ 3701
3076_0018	MIPI_DSI_CONFIG	32	R/W	0020_0000h	13.4.4.7/ 3703

Table continues on the next page...

MIPI_DSI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3076_001C	Escape Mode Register (MIPI_DSI_ESCMODE)	32	R/W	0300_0400h	13.4.4.8/ 3708
3076_0020	Main Display Image Resolution Register (MIPI_DSI_MDRESOL)	32	R/W	0000_0000h	13.4.4.9/ 3710
3076_0024	Main Display VPORCH Register (MIPI_DSI_MVPORCH)	32	R/W	F000_0000h	13.4.4.10/ 3711
3076_0028	MIPI_DSI_MHPORCH	32	R/W	0000_0000h	13.4.4.11/ 3712
3076_002C	MIPI_DSI_MSYNC	32	R/W	0000_0000h	13.4.4.12/ 3712
3076_0030	Sub Display Image Resolution Register (MIPI_DSI_SDRESOL)	32	R/W	0300_0400h	13.4.4.13/ 3713
3076_0034	Interrupt Source Register (MIPI_DSI_INTSRC)	32	R/W	0000_0000h	13.4.4.14/ 3714
3076_0038	Interrupt Mask Register (MIPI_DSI_INTMSK)	32	R/W	BB3F_7FFFh	13.4.4.15/ 3717
3076_003C	Packet Header FIFO Register (MIPI_DSI_PKTHDR)	32	R/W	0000_0000h	13.4.4.16/ 3720
3076_0040	Payload FIFO Register (MIPI_DSI_PAYLOAD)	32	R/W	0000_0000h	13.4.4.17/ 3720
3076_0044	Payload FIFO Register (MIPI_DSI_RXFIFO)	32	R/W	0000_0000h	13.4.4.18/ 3721
3076_0048	FIFO Threshold Level Register (MIPI_DSI_FIFOTHLD)	32	R/W	0000_01FFh	13.4.4.19/ 3721
3076_004C	FIFO Status and Control Register (MIPI_DSI_FIFOCTRL)	32	R/W	0155_551Fh	13.4.4.20/ 3722
3076_0050	FIFO Memory AC Characteristic Register (MIPI_DSI_MEMACCHR)	32	R/W	0000_4040h	13.4.4.21/ 3724
3076_0078	MIPI_DSI_MULTI_PKT	32	R/W	0001_0002h	13.4.4.22/ 3726
3076_0090	1 Gbps D-PHY PLL Control Register (MIPI_DSI_PLLCTRL_1G)	32	R/W	0000_0000h	13.4.4.23/ 3728
3076_0094	PLL Control register (MIPI_DSI_PLLCTRL)	32	R/W	0000_0000h	13.4.4.24/ 3730
3076_0098	PLL Control Register 1 (MIPI_DSI_PLLCTRL1)	32	R/W	0000_0000h	13.4.4.25/ 3731
3076_009C	PLL control register 2 (MIPI_DSI_PLLCTRL2)	32	R/W	0000_0000h	13.4.4.26/ 3731
3076_00A0	PLL Timer Register (MIPI_DSI_PLLTMR)	32	R/W	FFFF_FFFFh	13.4.4.27/ 3732
3076_00A4	D-PHY Master and Slave Analog Block Control Register 1 (MIPI_DSI_PHYCTRL_B1)	32	R/W	0000_0000h	13.4.4.28/ 3732
3076_00A8	D-PHY Master and Slave Analog Block Control Register 2 (MIPI_DSI_PHYCTRL_B2)	32	R/W	0000_0000h	13.4.4.29/ 3733

Table continues on the next page...

MIPI_DSI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3076_00AC	D-PHY Master Analog Block Control Register 1 (MIPI_DSI_PHYCTRL_M1)	32	R/W	0000_0000h	13.4.4.30/ 3733
3076_00B0	D-PHY Master Analog Block Control Register 1 (MIPI_DSI_PHYCTRL_M2)	32	R/W	0000_0000h	13.4.4.31/ 3734
3076_00B4	D-PHY Timing register (MIPI_DSI_PHYTIMING)	32	R/W	0000_0000h	13.4.4.32/ 3734
3076_00B8	MIPI_DSI_PHYTIMING1	32	R/W	0000_0000h	13.4.4.33/ 3735
3076_00BC	D-PHY Timing Register 2 (MIPI_DSI_PHYTIMING2)	32	R/W	0000_0000h	13.4.4.34/ 3735

13.4.4.1 Version Register (MIPI_DSI_VERSION)

Address: 3076_0000h base + 0h offset = 3076_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VERSION																															
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

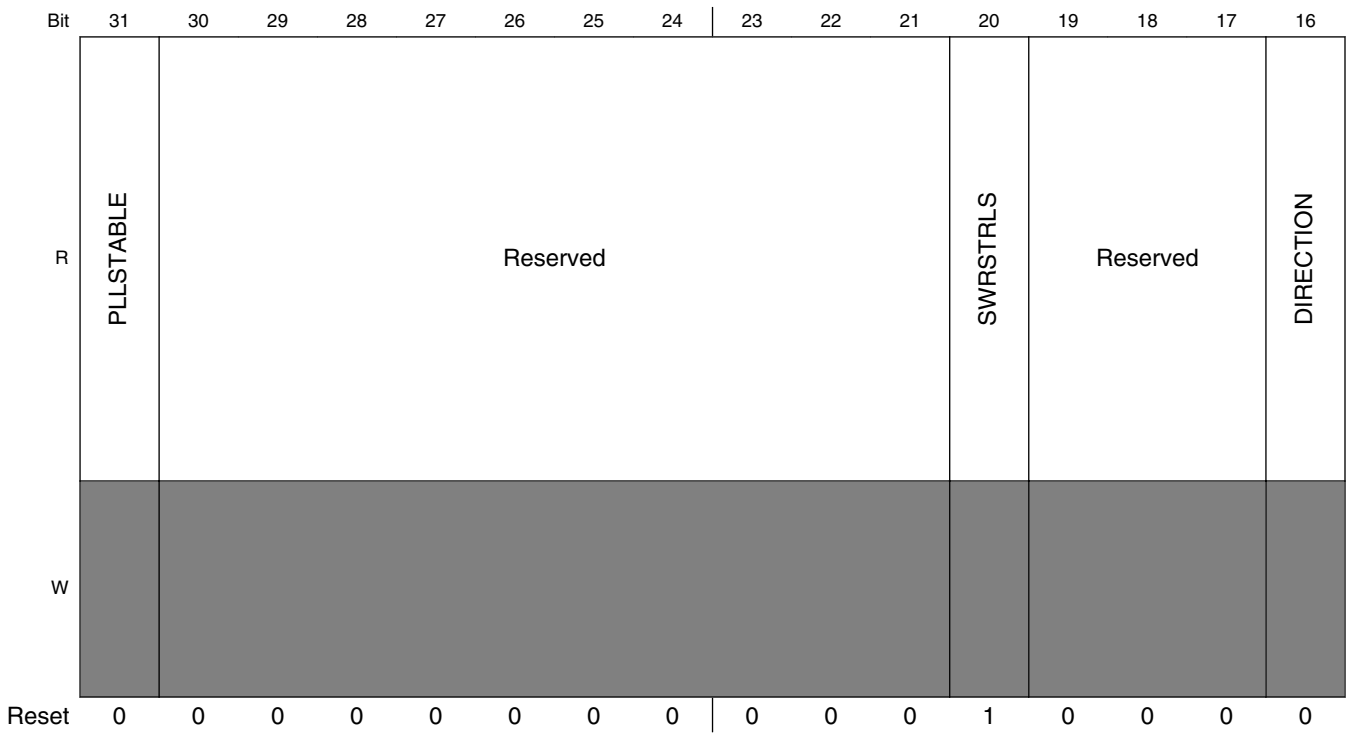
MIPI_DSI_VERSION field descriptions

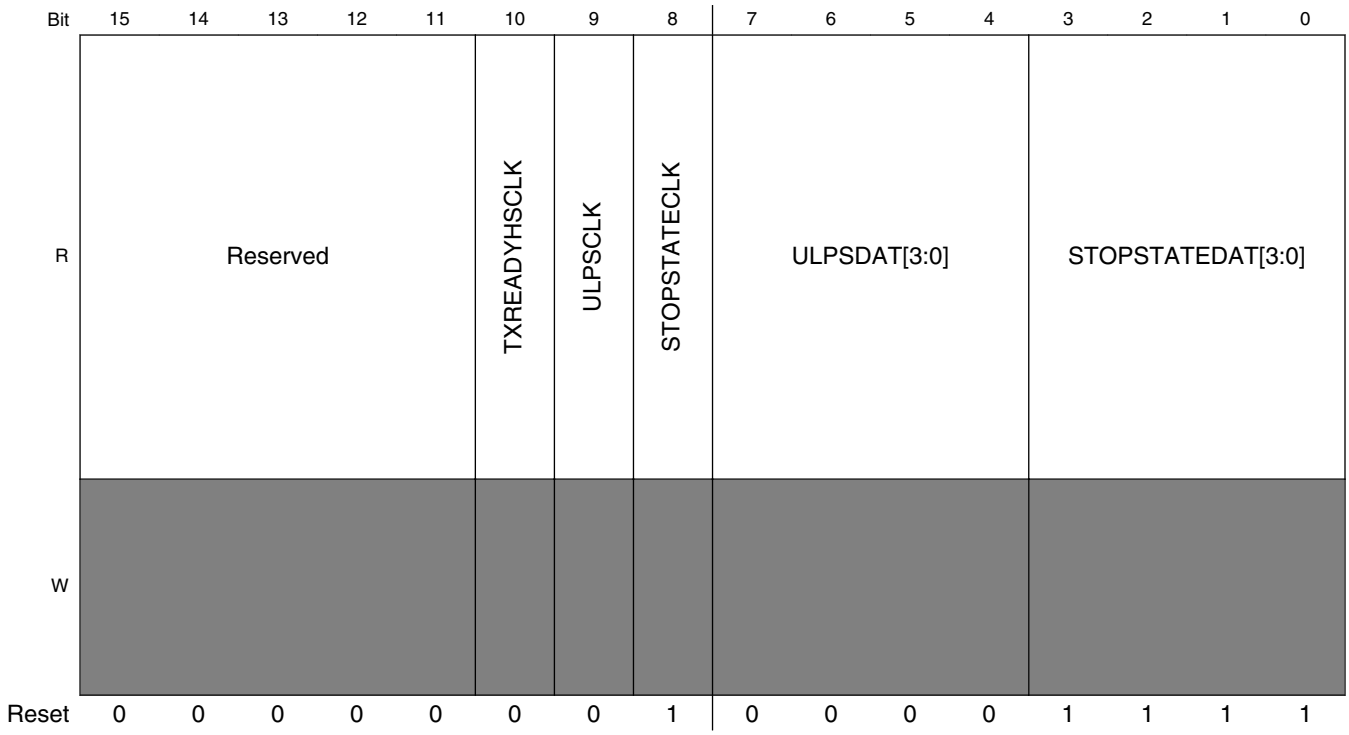
Field	Description
VERSION	Specifies the DSIM version information

13.4.4.2 MIPI_DSI_STATUS

This register reads and checks internal and interface status. It also checks FSM status, Line buffer status, current image line number, and so on.

Address: 3076_0000h base + 4h offset = 3076_0004h





MIPI_DSI_STATUS field descriptions

Field	Description
31 PLLSTABLE	D-PHY PLL generates stable byteclk.
30–21 Reserved	This field is reserved.
20 SWRSTRLS	Specifies the software reset status 0 Reset state 1 Release state
19–17 Reserved	This field is reserved.
16 DIRECTION	Specifies the data direction indicator 0 Forward direction 1 Backward direction
15–11 Reserved	This field is reserved.
10 TXREADYHSCLK	Specifies the HS clock ready at clock lane 0 Not ready for transmitting HS data at clock lane 1 Ready for transmitting HS data at clock lane
9 ULPSCLK	Specifies the ULPS indicator at clock lane 0 No ULPS in clock lane 1 ULPS in clock lane

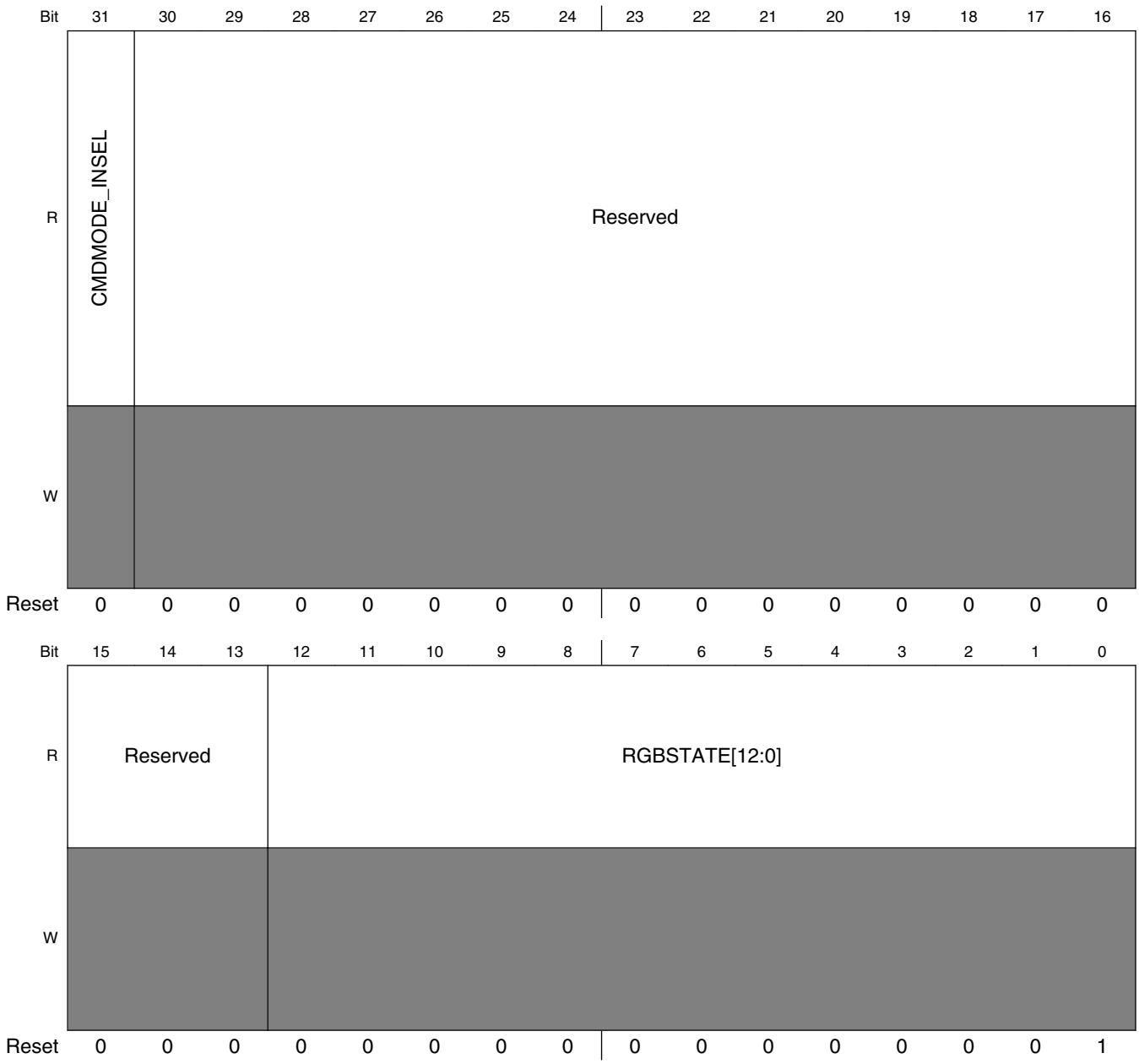
Table continues on the next page...

MIPI_DSI_STATUS field descriptions (continued)

Field	Description
8 STOPSTATECLK	Specifies the stop state indicator at clock lane 0 No stop state in clock lane 1 Stop state in clock lane
7–4 ULPSDAT[3:0]	Specifies the ULPS indicator at data lanes • UlpsDat[0]: Data lane 0 • UlpsDat[1]: Data lane 1 • UlpsDat[2]: Reserved • UlpsDat[3]: Reserved 0 No ULPS in each data lane 1 ULPS in each data lane
STOPSTATEDAT[3:0]	Specifies the stop state indicator at data lane • StopstateDat[0]: Data lane 0 • StopstateDat[1]: Data lane 1 • StopstateDat[2]: Reserved • StopstateDat[3]: Reserved

13.4.4.3 RGB Status Register (MIPI_DSI_RGB_STATUS)

Address: 3076_0000h base + 8h offset = 3076_0008h



MIPI_DSI_RGB_STATUS field descriptions

Field	Description
31 CMDMODE_ INSEL	Specifies the command mode input selection

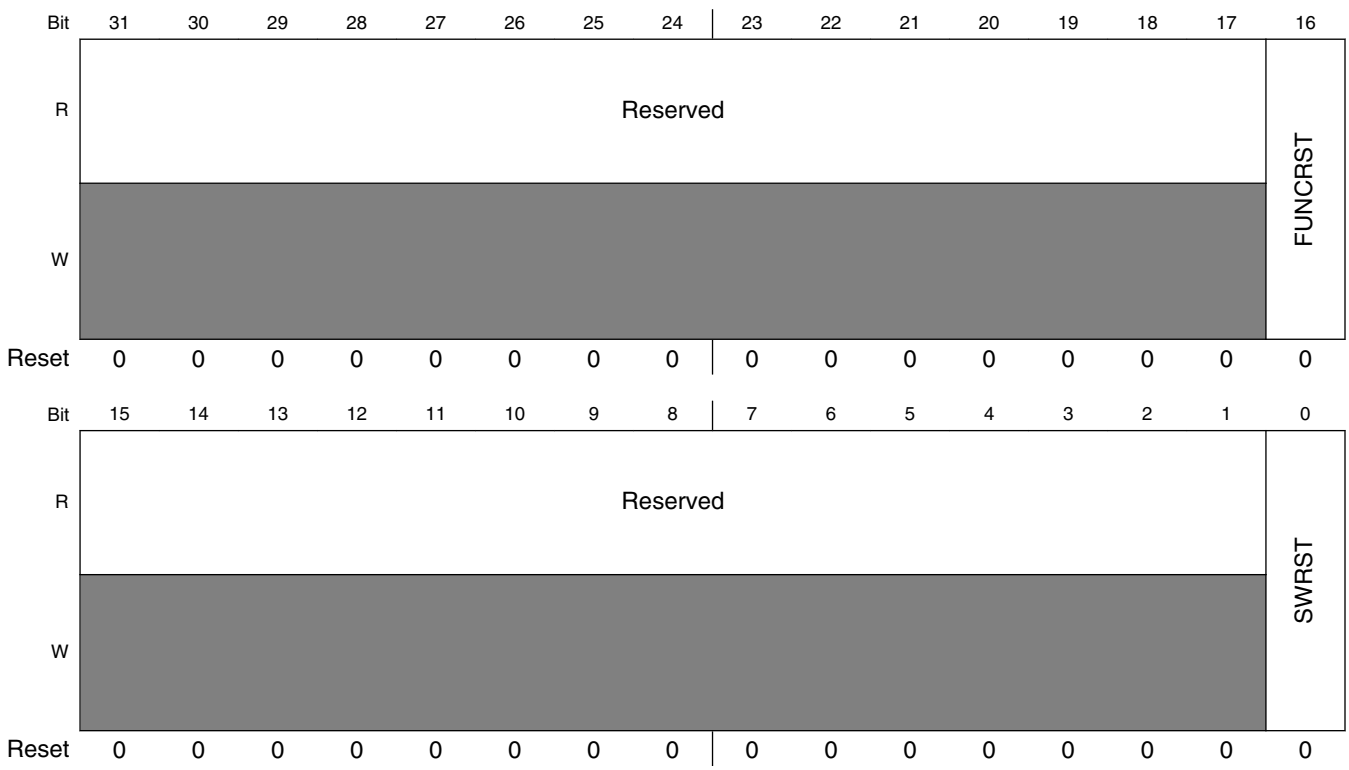
Table continues on the next page...

MIPI_DSI_RGB_STATUS field descriptions (continued)

Field	Description
	0 Using RGB video interface 1 Using S-i80 interface
30–13 Reserved	This field is reserved.
RGBSTATE[12:0]	Specifies the RGB packetize FSM status 0x0001 IDLE 0x0002 STOP 0x0004 VSYS 0x0008 VSE 0x0010 HSS 0x0020 HSA 0x0040 HSE 0x0080 HBP 0x0100 RGB 0x0200 NULL 0x0400 HFP 0x0800 EOT 0x1000 NHOLD

13.4.4.4 MIPI_DSI_SWRST

Address: 3076_0000h base + Ch offset = 3076_000Ch



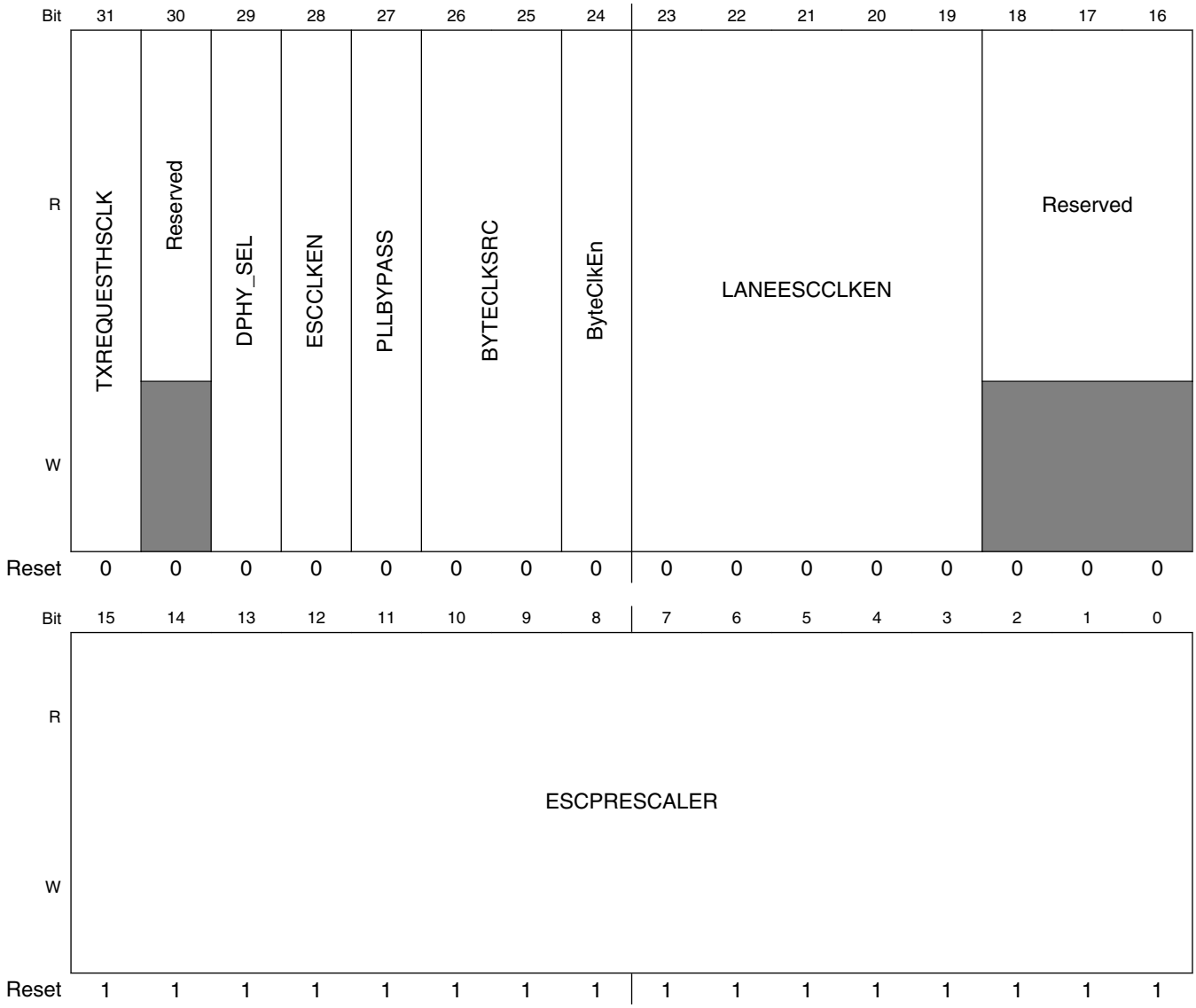
MIPI_DSI_SWRST field descriptions

Field	Description
31–17 Reserved	This field is reserved.
16 FUNCRST	Specifies the software reset (High active). "Function reset" resets all FF in MIPI DSIM (except SFRs: STATUS, RGB_STATUS, SWRST, CLKCTRL, TIMEOUT, CONFIG, ESCMODE[1], MDRESOL, MVPORCH, MHPORCH, MSYNC, INTMSK, SDRESOL, S3D_CTL, P3D_CTL, MIC_CTL, P3D_ON_MIC_OFF_HSA, P3D_OFF_MIC_ON_HSA, P3D_ON_MIC_ON_HSA, P3D_ON_MIC_OFF_HFP, P3D_OFF_MIC_ON_HFP, P3D_ON_MIC_ON_HFP, MULTI_PKT, FIFOTHLD, FIFOCTRL, [2], MEMACCHR, PLLCTRL, PLLTMR, PHYACCHR, and VERINFORM). 0 Standy 1 Reset
15–1 Reserved	This field is reserved.
0 SWRST	Specifies the software reset (High active). "Software reset" resets all FF in MIPI DSIM (except some SFRs: STATUS, SWRST, CLKCTRL, PLLCTRL, PLLTMR, PHYTUNE, and VERINFORM). 0 Standy 1 Reset

1. ForceStopstate, CmdLpdt, TxLpdt
2. nInitRx, nInitSfr, nInitI80, nInitSub, nInitMD

13.4.4.5 Clock Control Register (MIPI_DSI_CLKCTRL)

Address: 3076_0000h base + 10h offset = 3076_0010h



MIPI_DSI_CLKCTRL field descriptions

Field	Description
31 TXREQUESTHSCLK	Specifies the HS clock request for HS transfer at clock lane (Turn on HS clock)
30 Reserved	This field is reserved.
29 DPHY_SEL	D-PHY Select 0 1.5 Gbps D-PHY (default) 1 1 Gbps D-PHY

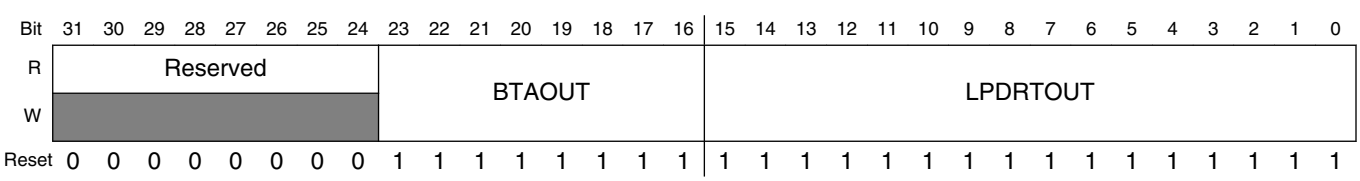
Table continues on the next page...

MIPI_DSI_CLKCTRL field descriptions (continued)

Field	Description
28 ESCCLKEN	Enables the escape clock generating prescaler 0 Disables 1 Enables
27 PLLBYPASS	Sets the PLLBypass signal connected to D-PHY module input for selecting clock source bit. This bit must be set to 0. 0 PLL output 1 External Serial clock
26–25 BYTECLKSRC	Selects byte clock source. (It must be 2'b00.) 00 = D-PHY PLL (default). PLL_out clock is used to generate ByteClk by dividing 8.
24 ByteClkEn	Enables byte clock 0 Disables 1 Enables
23–19 LANEESCCLKEN	Enables escape clock for D-phy lane • LaneEscClkEn[0] = Clock lane • LaneEscClkEn[1] = Data lane 0 • LaneEscClkEn[2] = Data lane 1 • LaneEscClkEn[3] = Reserved • LaneEscClkEn[4] = Reserved 0 Disables 1 Enables
18–16 Reserved	This field is reserved.
ESCPRESCALER	Specifies the escape clock prescaler value. The escape clock frequency range varies up to 20 MHz. **NOTE:** The requirement for BTA is that the Host Escclk frequency should range between 66.7 ~ 150% of the peripheral escape clock frequency. EscClk = ByteClk / (EscPrescaler)

13.4.4.6 MIPI_DSI_TIMEOUT

Address: 3076_0000h base + 14h offset = 3076_0014h



MIPI_DSI_TIMEOUT field descriptions

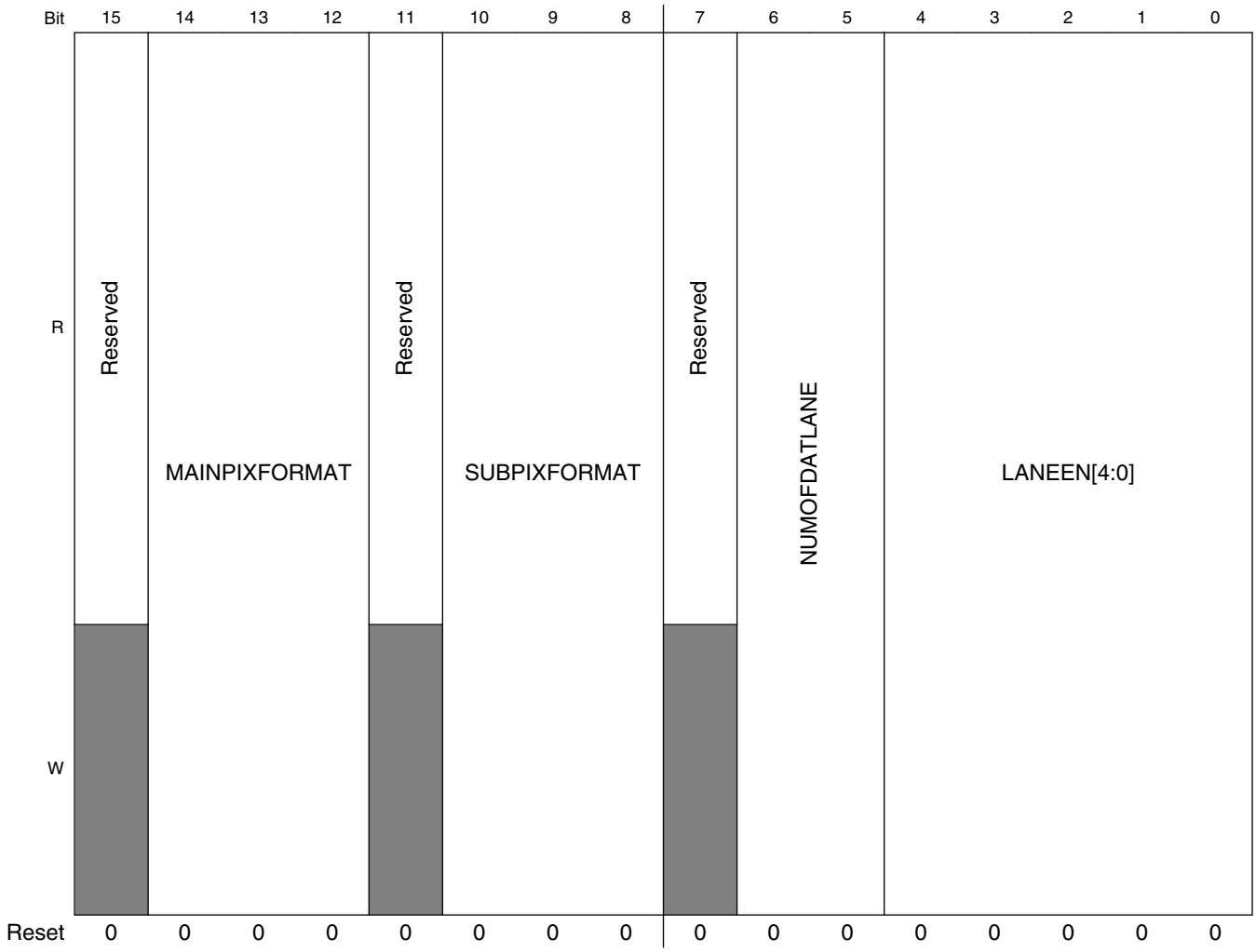
Field	Description
31–24 Reserved	This field is reserved.
23–16 BTAOUT	Specifies the timer for BTA. This register specifies time out from BTA request to change the direction with respect to Tx escape clock.
LPDRTOUT	Specifies the timer for LP Rx mode timeout. This register specifies time out on how long RxValid deasserts, after RxLpdt asserts with respect to Tx escape clock. RxValid specifies Rx data valid indicator. RxLpdt specifies an indicator that D-phy is under RxLpdt mode. RxValid and RxLpdt specifies signal from D-phy.

13.4.4.7 MIPI_DSI_CONFIG

This register configures MIPI DSI master such as data lane number, input interface, porch area, frame rate, BTA, LPDT, ULPS, and so on.

Address: 3076_0000h base + 18h offset = 3076_0018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NON_CONTINUOUS_CLOCK_LANE	CLKLANE_STOP_START	MFLUSH_VS	EOT_ R03	SYNCINFORM	BURSTMODE	VIDEOMODE	AUTOMODE	HSEDISABLEMODE	HFPDISABLEMODE	HBPDISABLEMODE	HSADISABLEMODE	MAINVC		SUBVC	
W						BURSTMODE										
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0



MIPI_DSI_CONFIG field descriptions

Field	Description
31 NON_ CONTINUOUS_ CLOCK_LANE	Non-continuous clock mode. This feature is D-PHY clock lane control DP / DN. The follwoing figure shows timing diagram of clock lane. Multi Packet register shows configuration of count values.

Table continues on the next page...

MIPI_DSI_CONFIG field descriptions (continued)

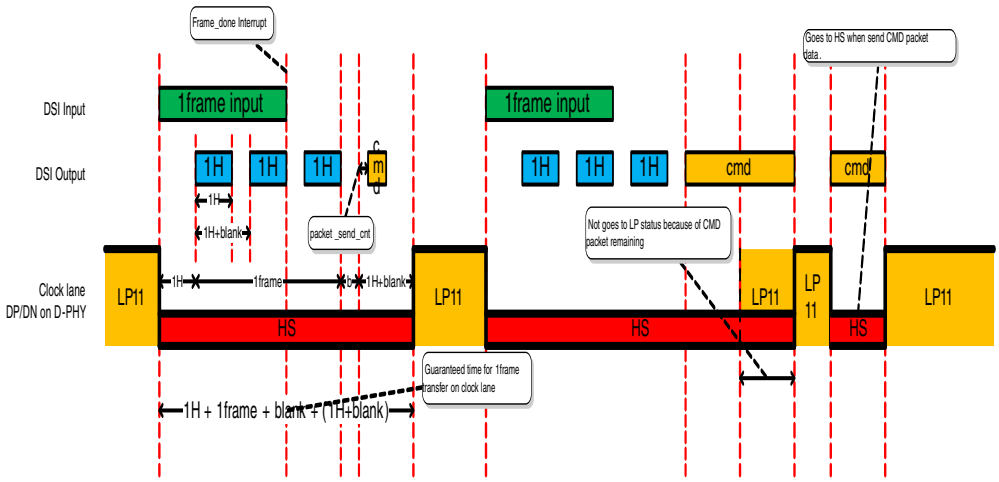
Field	Description
	 **Figure 13-46. Non-continuous Clock Lane Timing Diagram** 0 Disable (default) 1 Enable
30 CLKLANE_STOP_START	PHY clock lane On / Off for ESD. It is operated when first HBP to second HFP in every VSYNC. This feature not guarantees on Command mode. 0 Disable (default) 1 Enable
29 MFLUSH_VS	Auto flush of MD FIFO using Vsync pulse. It needs that Main display FIFO should be flushed for deleting garbage data. 0 Disable (default) 1 Enable
28 EOT_R03	Disables EoT packet in HS mode. 0 Enables EoT packet generation for V1.01r11 1 Disables EoT packet generation for V1.01r03
27 SYNCINFORM	Selects Sync Pulse or Event mode in Video mode In command mode, this bit is ignored. 0 Event mode (non burst, burst) 1 Pulse mode (non burst only)
26 BURSTMODE	Selects Burst mode in Video mode In Non-burst mode, RGB data area is filled with RGB data and Null packets, according to input bandwidth of RGB interface. In Burst mode, RGB data area is filled with RGB data only.

Table continues on the next page...

MIPI_DSI_CONFIG field descriptions (continued)

Field	Description
	In command mode, this bit is ignored. 0 Non-burst mode 1 Burst mode
25 VIDEOMODE	Specifies display configuration
24 AUTOMODE	Specifies auto vertical count mode In Video mode, the vertical line transition uses line counter configured by VSA, VBP, and Vertical resolution. If this bit is set to '1', the line counter does not use VSA and VBP registers. In command mode, this bit is ignored. 0 Configuration mode 1 Auto mode
23 HSEDISABLEMODE	In Vsync pulse and Vporch area, MIPI DSI master transfers only Hsync start packet to MIPI DSI slave at MIPI DSI spec 1.1r02. This bit transfers Hsync end packet in Vsync pulse and Vporch area (optional). In command mode, this bit is ignored. 0 Disables transfer 1 Enables transfer
22 HFPDISABLEMODE	Specifies HFP disable mode. If this bit set, DSI master ignores HFP area in Video mode. In command mode, this bit is ignored. 0 Enables 1 Disables
21 HBPDISABLEMODE	Specifies HBP disable mode. If this bit set, DSI master ignores HBP area in Video mode. In command mode, this bit is ignored. 0 Enables 1 Disables
20 HSADISABLEMODE	Specifies HSA disable mode. If this bit set, DSI master ignores HSA area in Video mode. In command mode, this bit is ignored. 0 Enables 1 Disables
19–18 MAINVC	Specifies virtual channel number for main display
17–16 SUBVC	Specifies virtual channel number for sub display
15 Reserved	This field is reserved.
14–12 MAINPIXFORMAT	Specifies pixel stream format for main display 000 3 bpp (for Command mode only) 001 8 bpp (for Command mode only) 010 12 bpp (for Command mode only)

Table continues on the next page...

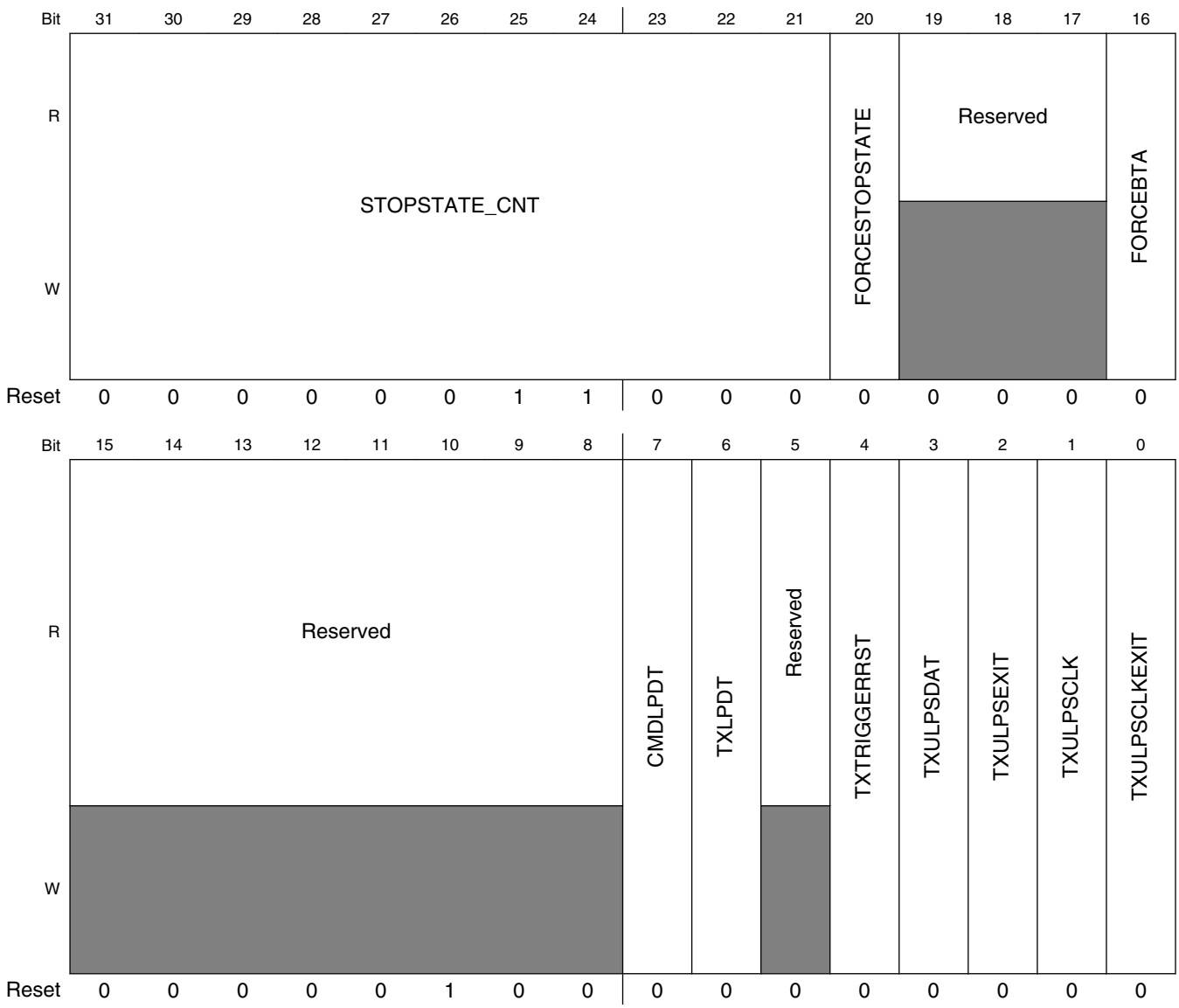
MIPI_DSI_CONFIG field descriptions (continued)

Field	Description
	011 16 bpp (for Command mode only) 100 16-bit RGB (565) (for Video mode only) 101 18-bit RGB (666: packed pixel stream) (for Video mode only) 110 18-bit RGB (666: loosely packed pixel stream) for Common 111 24-bit RGB (888) for common
11 Reserved	This field is reserved.
10–8 SUBPIXFORMAT	Specifies pixel stream format for main display 000 3 bpp (for Command mode only) 001 8 bpp (for Command mode only) 010 12 bpp (for Command mode only) 011 16 bpp (for Command mode only) 100 16-bit RGB (565) (for Video mode only) 101 18-bit RGB (666: packed pixel stream) (for Video mode only) 110 18-bit RGB (666: loosely packed pixel stream) for Common 111 24-bit RGB (888) for common
7 Reserved	This field is reserved.
6–5 NUMOFDATLANE	Sets the data lane number 00 Data lane 0 (1 data lane) 01 Data lane 0 ~ 1 (2 data lanes) 10 Reserved. 11 Reserved.
LANEEN[4:0]	Enables the lane. If Lane_EN is disabled, the lane ignores input and drives initial value through output port. • LaneEn[0] = Clock lane enabler • LaneEn[1] = Data lane 0 enabler • LaneEn[2] = Data lane 1 enabler • LaneEn[3] = Reserved • LaneEn[4] = Reserved

13.4.4.8 Escape Mode Register (MIPI_DSI_ESCMODE)

This register configures MIPI DSI master.

Address: 3076_0000h base + 1Ch offset = 3076_001Ch



MIPI_DSI_ESCMODE field descriptions

Field	Description
31–21 STOPSTATE_CNT	After transmitting read packet or write “set_tear_on” command, BTA requests to D-phy automatically. This counter value specifies the interval value between transmitting read packet (or write “set_tear_on” command) and BTA request. 11'h000 = 2 EscClk

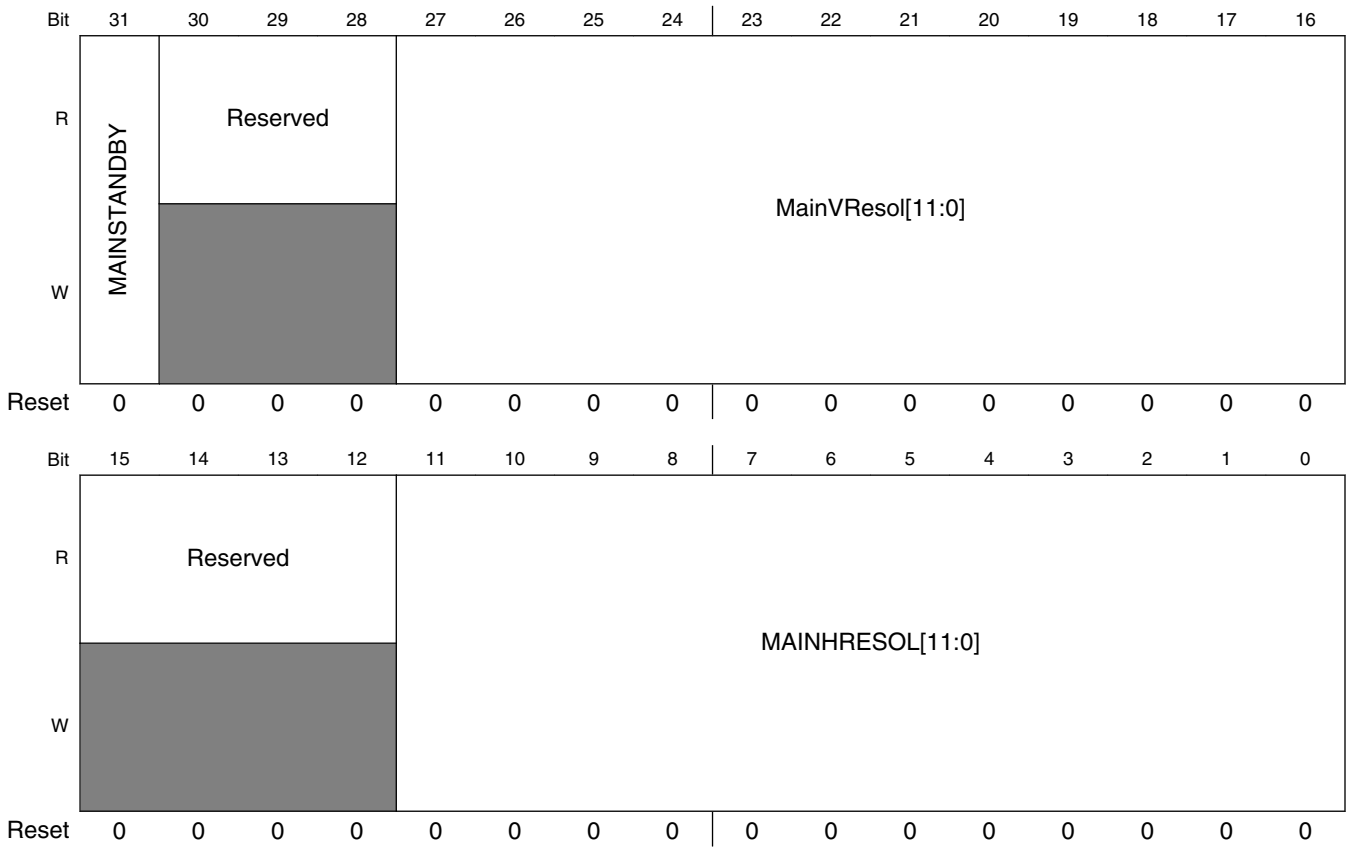
Table continues on the next page...

MIPI_DSI_ESCMODE field descriptions (continued)

Field	Description
	11'h001 = 2 EscClk + 1 EscClk ~ 11'h3FF = 2 EscClk + 1023 EscClk
20 FORCESTOPSTATE	Forces Stopstate for D-PHY
19–17 Reserved	This field is reserved.
16 FORCEBTA	Forces Bus Turn Around 1 = Sends the protocol layer request to D-PHY. MIPI DSI peripheral becomes master after BTA sequence. This bit clears automatically after receiving BTA acknowledge from MIPI DSI peripheral.
15–8 Reserved	This field is reserved.
7 CMDLPDT	Specifies LPDT transfers command in SFR FIFO 0 HS Mode 1 LP Mode
6 TXLPDT	Specifies data transmission in LP mode (all data transfer in LPDT) 0 HS Mode 1 LP Mode
5 Reserved	This field is reserved.
4 TXTRIGGERRST	Specifies remote reset trigger function. After trigger operation, these bits will be cleared automatically.
3 TXULPSDAT	Specifies ULPS request for data lane. Manually clears after ULPS exit.
2 TXULPSEXIT	Specifies ULPS exit request for data lane. Manually clears after ULPS exit.
1 TXULPSCLK	Specifies ULPS request for clock lane. Manually clears after ULPS exit.
0 TXULPSCLKEXIT	Specifies ULPS exit request for clock lane. Manually clears after ULPS exit.

13.4.4.9 Main Display Image Resolution Register (MIPI_DSI_MDRESOL)

Address: 3076_0000h base + 20h offset = 3076_0020h



MIPI_DSI_MDRESOL field descriptions

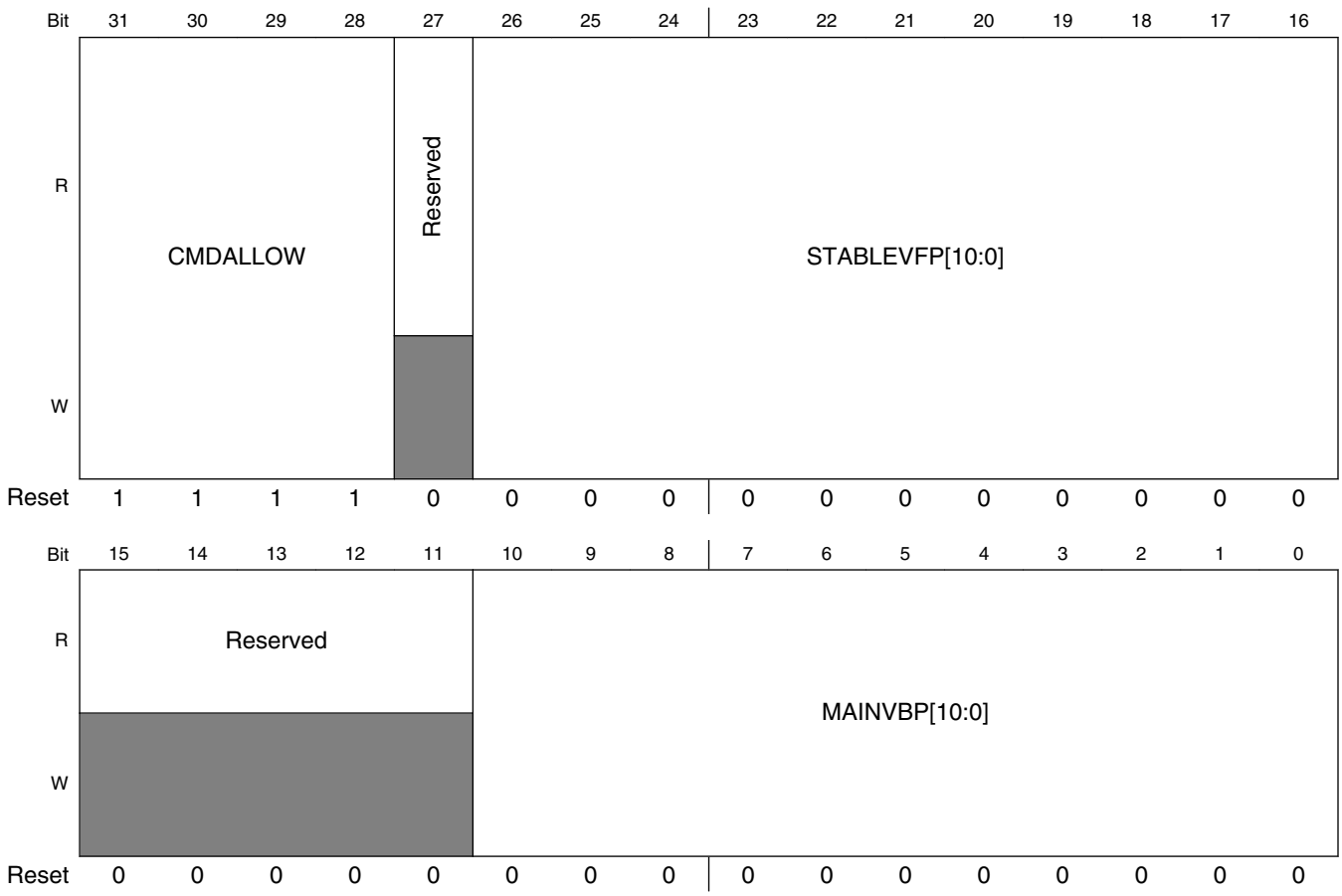
Field	Description
31 MAINSTANDBY	Specifies standby for receiving DISPCON output in Command mode after setting all configuration. Standby should be set after configuration (resolution, reqtype, pixelform, and so on) is set for command mode. In Video mode, if this bit value is 0, data is not transferred. 0 Not ready 1 Standby
30–28 Reserved	This field is reserved.
27–16 MainVResol[11:0]	Specifies Vertical resolution (1 ~ 2047)
15–12 Reserved	This field is reserved.
MAINHRESOL[11:0]	Specifies Horizontal resolution (1 ~ 2047)

13.4.4.10 Main Display VPORCH Register (MIPI_DSI_MVPORCH)

NOTE

Transfers command packets after Stable VFP area. Display controller VFP lines should be set based on sum of these values: Stable VFP, command allowing area and command masked area. See the section for transferring general data in Video mode.

Address: 3076_0000h base + 24h offset = 3076_0024h



MIPI_DSI_MVPORCH field descriptions

Field	Description
31–28 CMDALLOW	Specifies the number of horizontal lines, where command packet transmission is allowed after Stable VFP period. For more information, see Figure 13-36.
27 Reserved	This field is reserved.

Table continues on the next page...

MIPI_DSI_MVPORCH field descriptions (continued)

Field	Description
26–16 STABLEVFP[10:0]	Specifies the number of horizontal lines, where command packet transmission is not allowed after end of active region. For more information, see Figure 13-36.. **NOTE:** In Command mode, these bits are ignored.
15–11 Reserved	This field is reserved.
MAINVBP[10:0]	Specifies vertical back porch width for Video mode (line count). In Command mode, these bits are ignored.

13.4.4.11 MIPI_DSI_MHPORCH

Address: 3076_0000h base + 28h offset = 3076_0028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MAINHFP[15:0]																MAINHBP[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_DSI_MHPORCH field descriptions

Field	Description
31–16 MAINHFP[15:0]	Specifies the horizontal front porch width for Video mode. HFP is specified using blank packet. These bits specify the word counts for blank packet in HFP. In Command mode, these bits are ignored.
MAINHBP[15:0]	Specifies the horizontal back porch width for Video mode. HBP is specified using blank packet. These bits specify the word counts for blank packet in HBP. In Command mode, these bits are ignored.

13.4.4.12 MIPI_DSI_MSYNC

Address: 3076_0000h base + 2Ch offset = 3076_002Ch

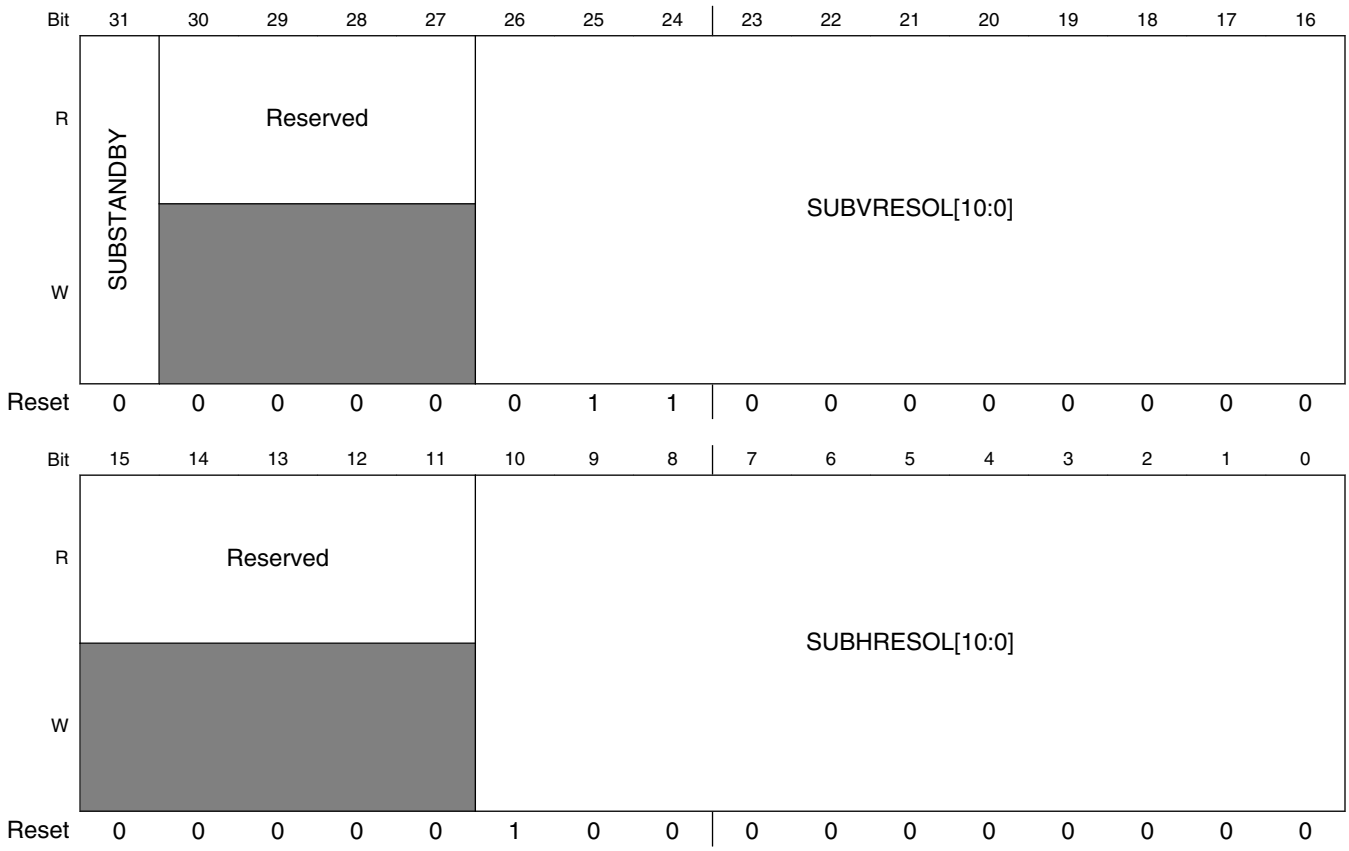
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MAINVSA[9:0]										Reserved						MAINHSA[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_DSI_MSYNC field descriptions

Field	Description
31–22 MAINVSA[9:0]	Specifies the vertical sync pulse width for Video mode (Line count). In command mode, these bits are ignored.
21–16 Reserved	This field is reserved.
MAINHSA[15:0]	Specifies the horizontal sync pulse width for Video mode. HSA is specified using blank packet. These bits specify word counts for blank packet in HSA. In command mode, these bits are ignored.

13.4.4.13 Sub Display Image Resolution Register (MIPI_DSI_SDRESOL)

Address: 3076_0000h base + 30h offset = 3076_0030h



MIPI_DSI_SDRESOL field descriptions

Field	Description
31 SUBSTANDBY	Specifies standby for receiving DISPCON output in Command mode after setting all configuration. Standby should be set after configuration (resolution, reqtype, pixelform, and so on) is set for command mode. In Video mode, this bit is ignored. 0 Not ready 1 Standby
30–27 Reserved	This field is reserved.
26–16 SUBVRESOL[10:0]	Specifies the Vertical resolution (1 ~ 1024)
15–11 Reserved	This field is reserved.
SUBHRESOL[10:0]	Specifies the Horizontal resolution (1 ~ 1024)

13.4.4.14 Interrupt Source Register (MIPI_DSI_INTSRC)

This register identifies interrupt sources.

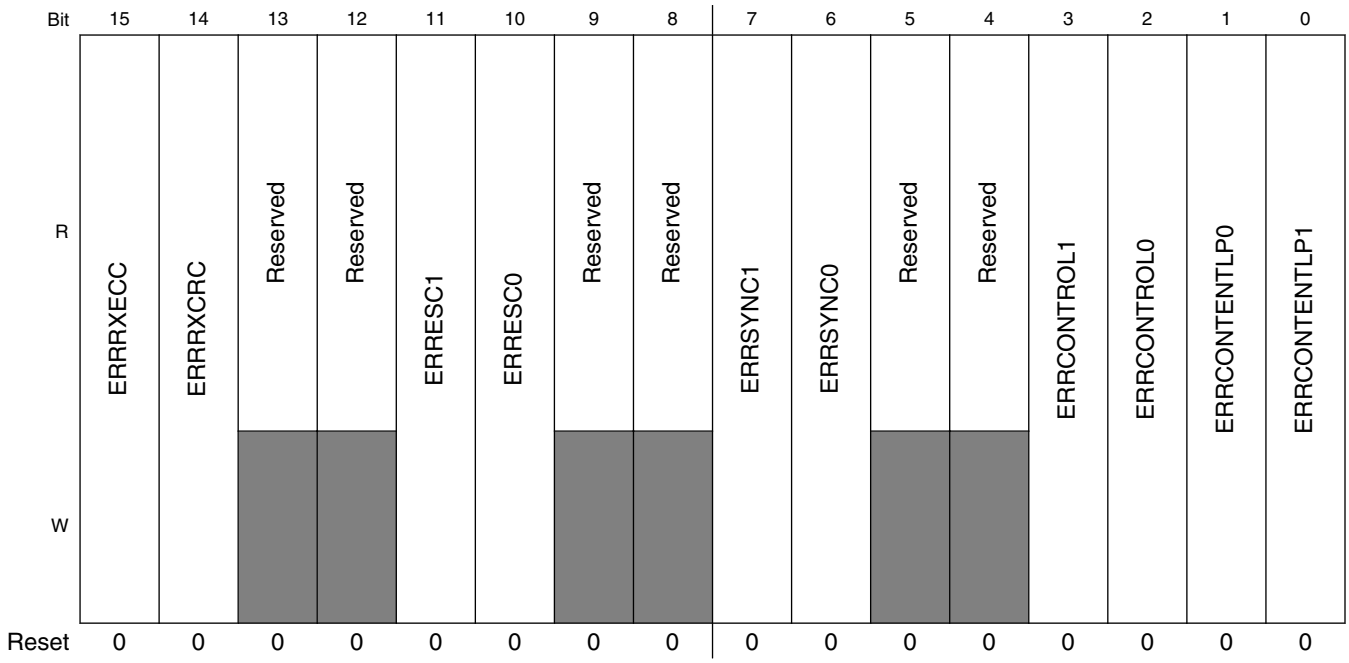
Internal block error, data transmit interrupt, inter-layer (D_PHY) error, etc.

The bits are set even if they are masked off by DSIM_INTMSK_REG.

Write ‘1’ to clear the Interrupt.

Address: 3076_0000h base + 34h offset = 3076_0034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PLLSTABLE	SWRSTRELEASE	SFRPLFIFOEMPTY	SFRPHFIFOEMPTY	SYNCOVERRIDE	Reserved	BUSTURNOVER	FRAMEDONE	Reserved		LPDRTOUT	TATOUT	Reserved	RXDATDONE	RXTE	RXACK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MIPI_DSI_INTSRC field descriptions

Field	Description
31 PLLSTABLE	Indicates that D-phy PLL is stable.
30 SWRSTRELEASE	Releases the software reset.
29 SFRPLFIFOEMPTY	Specifies the SFR payload FIFO empty.
28 SFRPHFIFOEMPTY	Specifies the SFR Packet Header FIFO emtpy
27 SYNCOVERRIDE	Indicates that other DSI command transfer have overridden sync timing.
26 Reserved	This field is reserved.
25 BUSTURNOVER	Indicates when bus grant turns over from DSI slave to DSI master.
24 FRAMEDONE	Indicates when MIPI DSIM transfers the whole image frame. **NOTE:** If Hsync is not received during two line times, internal timer is timed out and this bit is flagged.
23–22 Reserved	This field is reserved.
21 LPDRTOUT	Specifies the LP Rx timeout. See time out register (0x10).
20 TATOUT	Turns around Acknowledge Timeout. See time out register (0x10).
19 Reserved	This field is reserved.

Table continues on the next page...

MIPI_DSI_INTSRC field descriptions (continued)

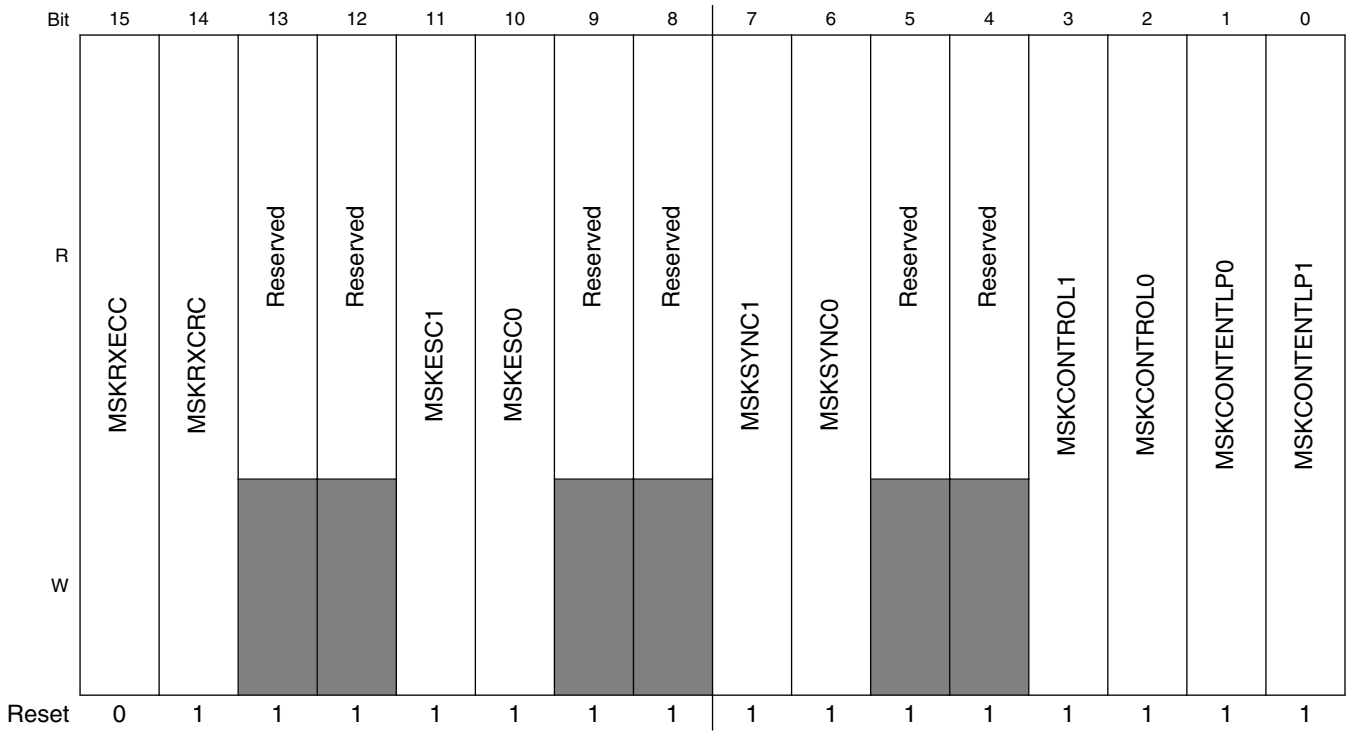
Field	Description
18 RXDATDONE	Completes receiving data
17 RXTE	Receives TE Rx trigger
16 RXACK	Receives ACK Rx trigger
15 ERRRXECC	Specifies the ECC multi bit error in LPDR
14 ERRRXCRC	Specifies the CRC error in LPDR
13 Reserved	This field is reserved.
12 Reserved	This field is reserved.
11 ERRESC1	Specifies the escape mode entry error lane 1. For more information, refer to standard D-PHY specification.
10 ERRESC0	Specifies the escape mode entry error lane 0. For more information, refer to standard D-PHY specification.
9 Reserved	This field is reserved.
8 Reserved	This field is reserved.
7 ERRSYNC1	Specifies the LPDT Sync Error lane1. For more information, refer to standard D-PHY specification.
6 ERRSYNC0	Specifies the LPDT Sync Error lane0. For more information, refer to standard D-PHY specification.
5 Reserved	This field is reserved.
4 Reserved	This field is reserved.
3 ERRCONTROL1	Controls Error lane1. For more information, refer to standard D-PHY specification.
2 ERRCONTROL0	Controls Error lane0. For more information, refer to standard D-PHY specification.
1 ERRCONTENTLP0	Specifies the LP0 Contention Error (only lane0, because BTA occurs at lane0 only). For more information, refer to standard D-PHY specification.
0 ERRCONTENTLP1	Specifies the LP1 Contention Error (only lane0, because BTA occurs at lane0 only). For more information, refer to standard D-PHY specification.

13.4.4.15 Interrupt Mask Register (MIPI_DSI_INTMSK)

This register masks interrupt sources.

Address: 3076_0000h base + 38h offset = 3076_0038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MSKPLLSTABLE	MSKSWRSTRELEASE	MSKSFRPLFIFOEMPTY	MSKSFRPHFIFOEMPTY	MSKSYNCOVERRIDE	Reserved	MSKBUSTURNOVER	MSKFRAMEDONE	Reserved		MSKLPDRTOUT	MSKTATOUT	Reserved	MSKRXDATDONE	MSKRXTE	MSKRXACK
W																
Reset	1	0	1	1	1	0	1	1	0	0	1	1	1	1	1	1



MIPI_DSI_INTMSK field descriptions

Field	Description
31 MSKPLLSTABLE	Indicates that D-PHY PLL is stable.
30 MSKSWRSTRELEASE	Releases software reset.
29 MSKSFRPLFIFOEMPTY	Empties SFR payload FIFO.
28 MSKSFRPHFIFOEMPTY	Interrupt Mask for SFR packet header FIFO empty
27 MSKSYNCOVERRIDE	Indicates that other DSI command transfer have overridden sync timing.
26 Reserved	This field is reserved.
25 MSKBUSTURNOVER	Indicates when bus grant turns over from DSI slave to DSI master.
24 MSKFRAMEDONE	Indicates when MIPI DSIM transfers whole image frame.
23–22 Reserved	This field is reserved.
21 MSKLPDRTOUT	Specifies LP Rx timeout. See time out register (0x10).
20 MSKTATOUT	Specifies turnaround acknowledge timeout. See time out register (0x10)
19 Reserved	This field is reserved.

Table continues on the next page...

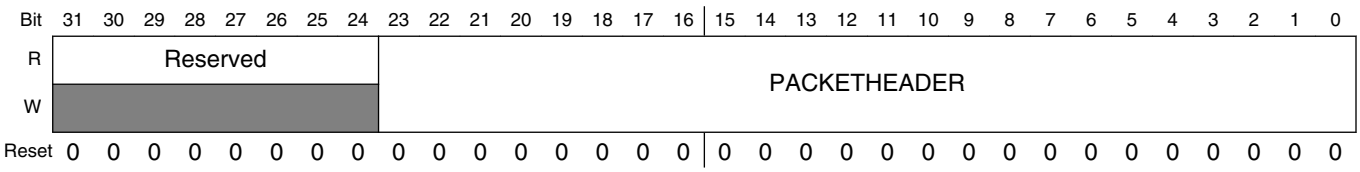
MIPI_DSI_INTMSK field descriptions (continued)

Field	Description
18 MSKRXDATDONE	Specifies completion of data receiving
17 MSKRXTE	Specifies receipt of TE Rx trigger
16 MSKRXACK	Specifies receipt of ACK Rx trigger
15 MSKRXECC	Specifies ECC multibit error in LPDR
14 MSKRXCRC	Specifies CRC error in LPDR
13 Reserved	This field is reserved.
12 Reserved	This field is reserved.
11 MSKESC1	Specifies escape mode entry error in lane1. For more information, refer to standard D-PHY specification.
10 MSKESC0	Specifies escape mode entry error in lane0. For more information, refer to standard D-PHY specification.
9 Reserved	This field is reserved.
8 Reserved	This field is reserved.
7 MSKSYNC1	Specifies LPDT sync error in lane1. For more information, refer to standard D-PHY specification.
6 MSKSYNC0	Specifies LPDT sync error in lane0. For more information, refer to standard D-PHY specification.
5 Reserved	This field is reserved.
4 Reserved	This field is reserved.
3 MSKCONTROL1	Controls error in lane1. For more information, refer to standard D-PHY specification.
2 MSKCONTROL0	Controls error in lane0. For more information, refer to standard D-PHY specification.
1 MSKCONTENTLP0	Specifies LP0 contention error. For more information, refer to standard D-PHY specification.
0 MSKCONTENTLP1	Specifies LP1 contention error. For more information, refer to standard D-PHY specification.

13.4.4.16 Packet Header FIFO Register (MIPI_DSI_PKTHDR)

This register is the FIFO for packet header to send DSI packets.

Address: 3076_0000h base + 3Ch offset = 3076_003Ch



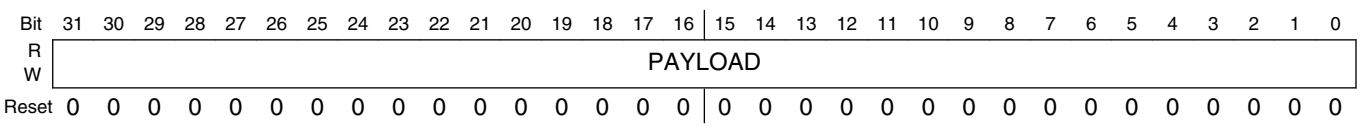
MIPI_DSI_PKTHDR field descriptions

Field	Description
31–24 Reserved	This field is reserved.
PACKETHEADER	Writes the packet header of Tx packet. • [7:0] = DI • [15:8] = Dat0 (Word Count lower byte for long packet) • [23:16] = Dat1 (Word Count upper byte for long packet)

13.4.4.17 Payload FIFO Register (MIPI_DSI_PAYLOAD)

This register is the FIFO for packet header to send DSI packets.

Address: 3076_0000h base + 40h offset = 3076_0040h



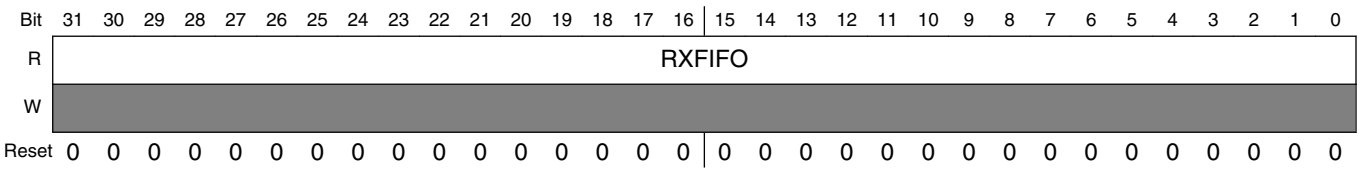
MIPI_DSI_PAYLOAD field descriptions

Field	Description
PAYLOAD	Writes the Payload of Tx packet

13.4.4.18 Payload FIFO Register (MIPI_DSI_RXFIFO)

This register is the gate of FIFO read.

Address: 3076_0000h base + 44h offset = 3076_0044h

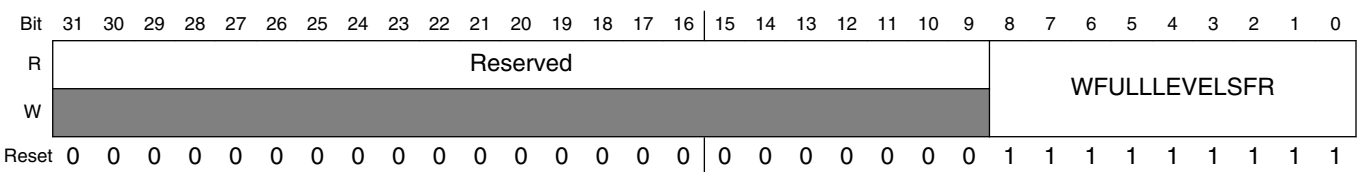


MIPI_DSI_RXFIFO field descriptions

Field	Description
RXFIFO	In the Rx mode, you can read Rx data through this register. **NOTE:** The CRC in packet is not stored in RxFIFO.

13.4.4.19 FIFO Threshold Level Register (MIPI_DSI_FIFOTHLD)

Address: 3076_0000h base + 48h offset = 3076_0048h



MIPI_DSI_FIFOTHLD field descriptions

Field	Description
31–9 Reserved	This field is reserved.
WFULLLEVELSFR	Almost full level of SFR payload FIFO

13.4.4.20 FIFO Status and Control Register (MIPI_DSI_FIFOCTRL)

Address: 3076_0000h base + 4Ch offset = 3076_004Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved						FULLRX	EMPTYRX	FullHSfr	EMPTYHSFR	FULLLSFR	EMPTYLSFR	FULLHI80	EMPTYHI80	FULLLI80	EMPTYLI80
W																
Reset	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FULLHSUB	EMPTYHSUB	FULLLSUB	EMPTYLSUB	FULLHMAIN	EMPTYHMAIN	FULLLMAIN	EMPTYLMAIN	Reserved			NINITRX	NINITSFR	NLNITL80	NINITSUB	NINITMAIN
W																
Reset	0	1	0	1	0	1	0	1	0	0	0	1	1	1	1	1

MIPI_DSI_FIFOCTRL field descriptions

Field	Description
31–26 Reserved	This field is reserved.
25 FULLRX	Rx FIFO full
24 EMPTYRX	Rx FIFO empty
23 FullHSfr	SFR packet header FIFO full
22 EMPTYHSFR	SFR packet header FIFO empty
21 FULLLSFR	SFR payload FIFO full
20 EMPTYLSFR	SFR payload FIFO empty
19 FULLHI80	S-i80 packet header FIFO full
18 EMPTYHI80	S-i80 packet header FIFO empty
17 FULLLI80	S-i80 payload FIFO full
16 EMPTYLI80	S-i80 payload FIFO empty
15 FULLHSUB	Sub display packet header FIFO full
14 EMPTYHSUB	Sub display packet header FIFO empty
13 FULLLSUB	Sub display payload FIFO full
12 EMPTYLSUB	Sub display payload FIFO empty
11 FULLHMAIN	Main display packet header FIFO full
10 EMPTYHMAIN	Main display packet header FIFO empty
9 FULLLMAIN	Main display payload FIFO full
8 EMPTYLMAIN	Main display payload FIFO empty
7–5 Reserved	This field is reserved.
4 NINITRX	MD FIFO read point initialize
3 NINITSFR	SFR FIFO write point initialize
2 NLNITL80	S-i80 FIFO write point initialize

Table continues on the next page...

MIPI_DSI_FIFOCTRL field descriptions (continued)

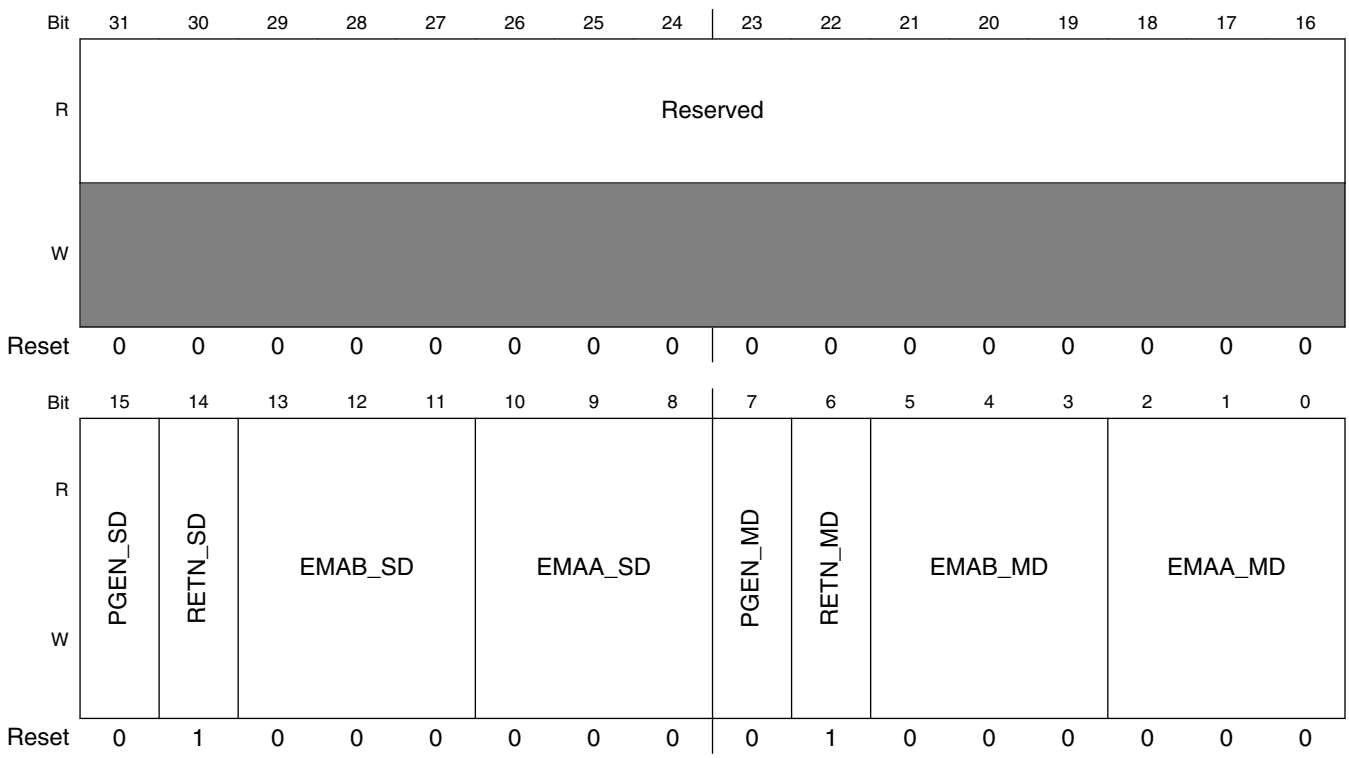
Field	Description
1 NINITSUB	SD FIFO write point initialize
0 NINITMAIN	MD FIFO write point initialize

13.4.4.21 FIFO Memory AC Characteristic Register (MIPI_DSI_MEMACCHR)

NOTE

In current design, these memory port controls were disabled. User can ignore the function of this register.

Address: 3076_0000h base + 50h offset = 3076_0050h



MIPI_DSI_MEMACCHR field descriptions

Field	Description
31–16 Reserved	This field is reserved.

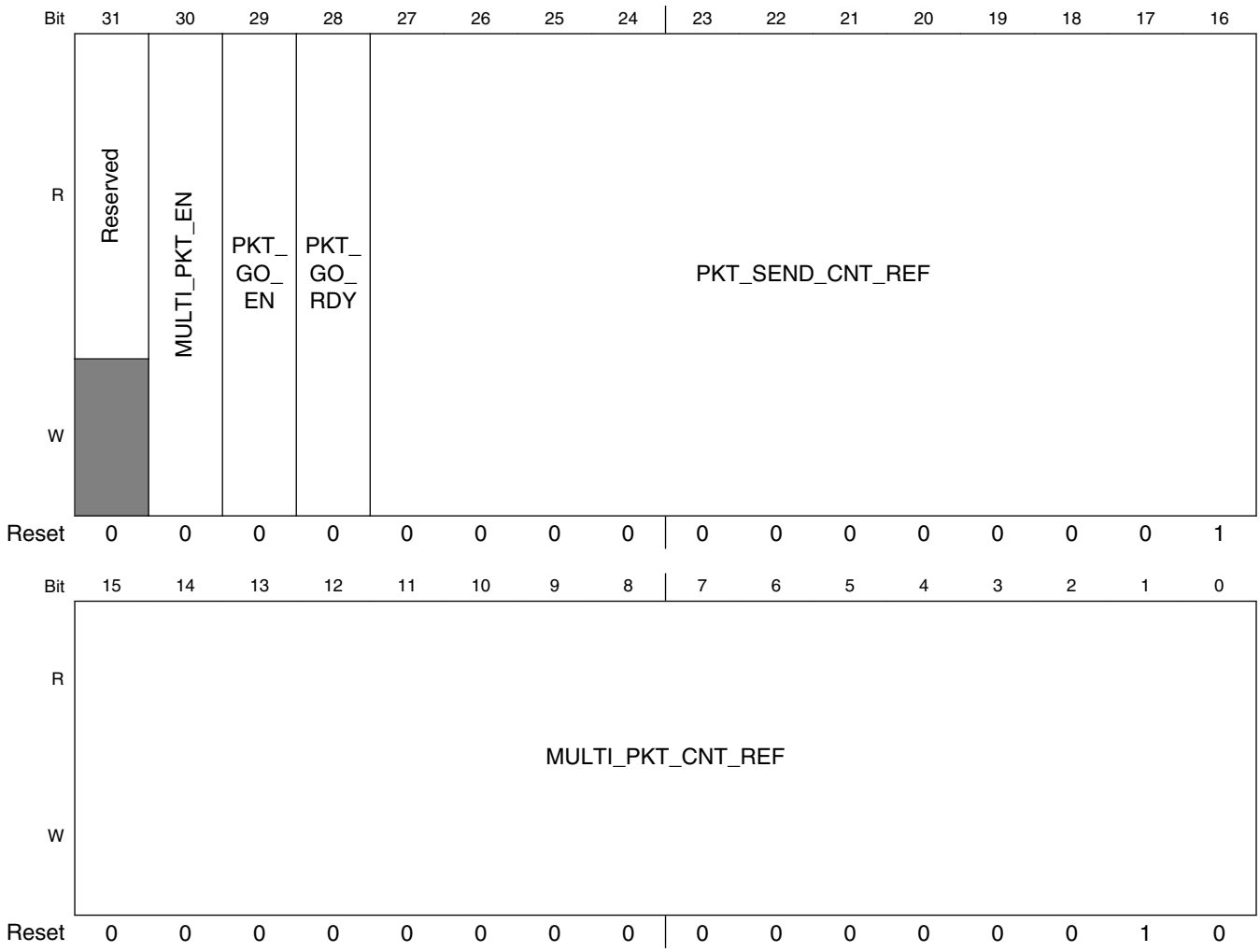
Table continues on the next page...

MIPI_DSI_MEMACCHR field descriptions (continued)

Field	Description
15 PGEN_SD	Sub display FIFO memory power gating
14 RETN_SD	Sub display FIFO memory Retention
13–11 EMAB_SD	Sub display FIFO memory B port margin adjustment
10–8 EMAA_SD	Sub display FIFO memory A port margin adjustment
7 PGEN_MD	Main display FIFO memory power gating
6 RETN_MD	Main display FIFO memory Retention
5–3 EMAB_MD	Main display FIFO memory B port margin adjustment
EMAA_MD	Main display FIFO memory A port margin adjustment

13.4.4.22 MIPI_DSI_MULTI_PKT

Address: 3076_0000h base + 78h offset = 3076_0078h



MIPI_DSI_MULTI_PKT field descriptions

Field	Description
31 Reserved	This field is reserved.
30 MULTI_PKT_EN	Specifies the send multi command packets on single transmission. Multi command packet can transfer by only HS mode. 0 Send single command packet 1 Send multi command packets
29 PKT_GO_EN	Specifies the send command packet(s) per frame enable . Packet go can transfer by only HS mode.

Table continues on the next page...

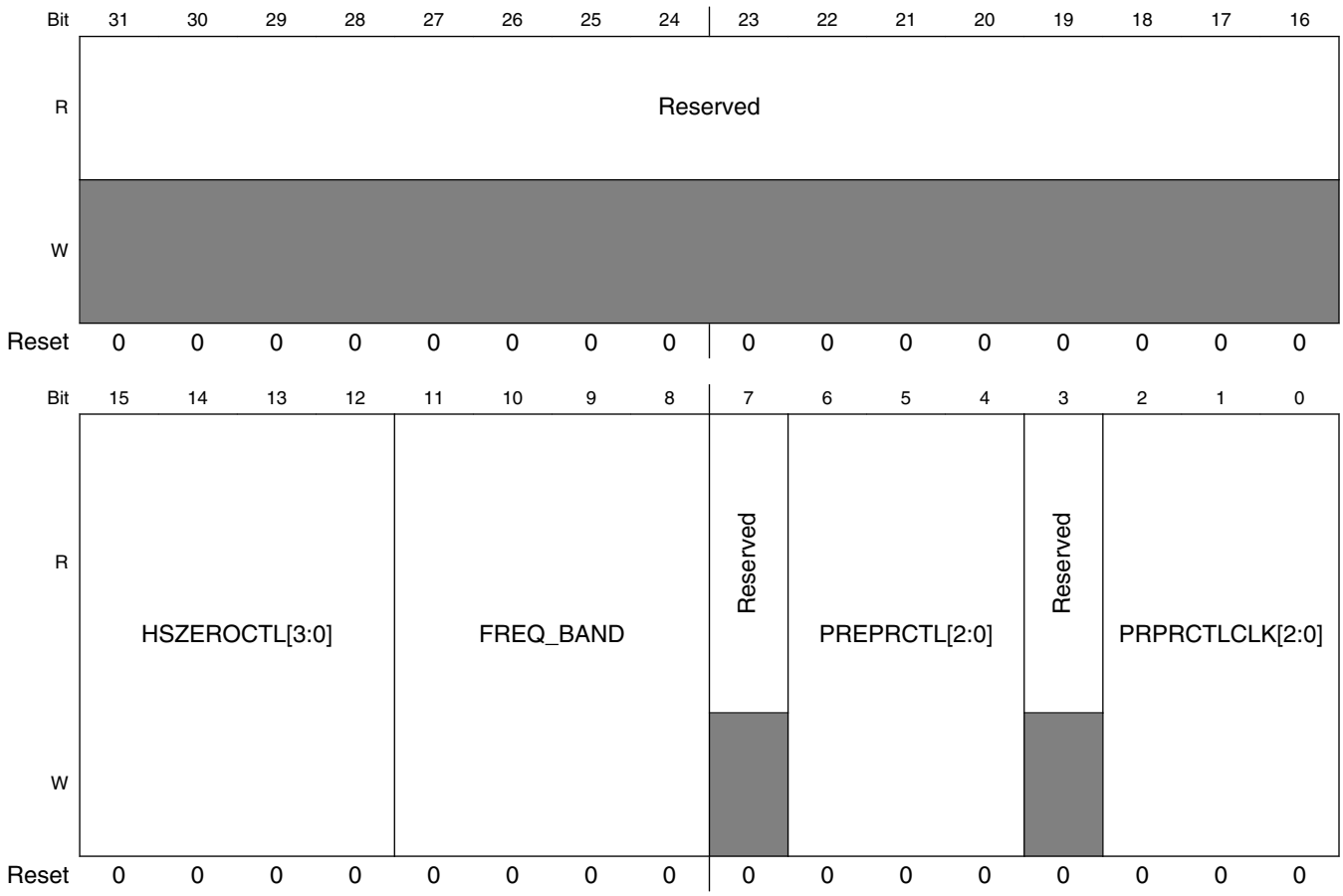
MIPI_DSI_MULTI_PKT field descriptions (continued)

Field	Description
	0 Send command packet(s) during every VFP 1 Send command packet(s) during 1frame VFP
28 PKT_GO_RDY	Specifies the send command packet(s) on this frame VFP. If packet transferred or PKT_Go_EN is disabled, this bit goes to 1'b0.
27–16 PKT_SEND_ CNT_REF	Specifies the command packet(s) send point indicator. That value is 1H time + D-PHY LP-HS-LP latency by ByteCLK. Do not use '0'.
MULTI_PKT_ CNT_REF	Specifies the number of packets on single transmission. Do not use '0' and '1'. 0 and 1 are same as '2'

13.4.4.23 1 Gbps D-PHY PLL Control Register (MIPI_DSI_PLLCTRL_1G)

This register configures 1Gbps D-PHY PLL control, D-PHY, clock range indication, and so on.

Address: 3076_0000h base + 90h offset = 3076_0090h



MIPI_DSI_PLLCTRL_1G field descriptions

Field	Description
31–16 Reserved	This field is reserved.
15–12 HSZEROCTL[3:0]	1 Gbps D-PHY HS-Zero driving timing control.
11–8 FREQ_BAND	1 Gbps D-PHY Timing control for D-PHY global operation timing.
7 Reserved	This field is reserved.

Table continues on the next page...

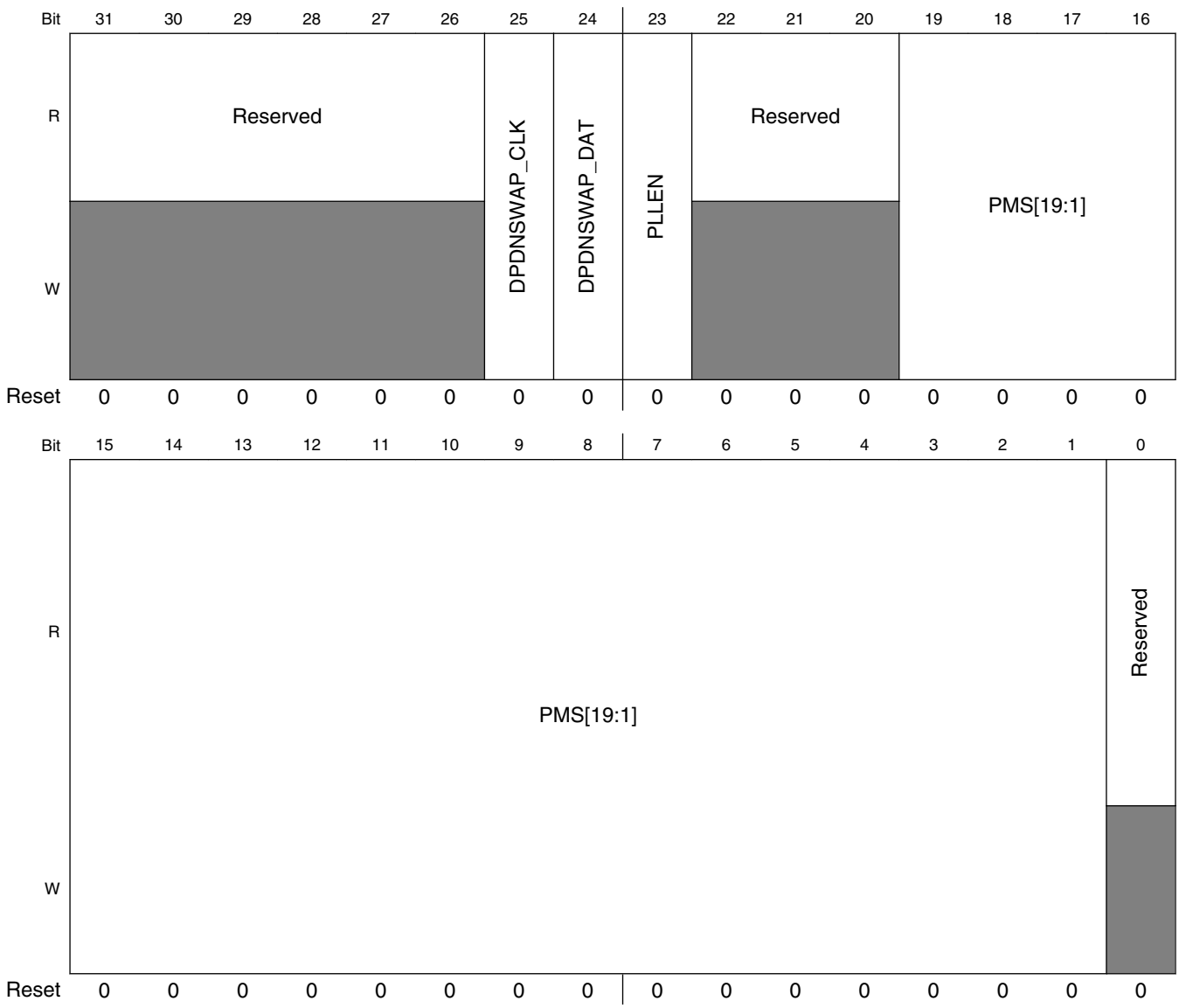
MIPI_DSI_PLLCTRL_1G field descriptions (continued)

Field	Description
6–4 PREPRCTL[2:0]	1 Gbps D-PHY PLL Tclk-prepare and Ths-prepare driving control.
3 Reserved	This field is reserved.
PRPRCTLCLK[2:0]	1 Gbps D-PHY PLL Ths-prepare driving time control

13.4.4.24 PLL Control register (MIPI_DSI_PLLCTRL)

This register configures PLL control, D-PHY, clock range indication, and so on.

Address: 3076_0000h base + 94h offset = 3076_0094h



MIPI_DSI_PLLCTRL field descriptions

Field	Description
31–26 Reserved	This field is reserved.

Table continues on the next page...

MIPI_DSI_PLLCTRL field descriptions (continued)

Field	Description
25 DPDNSWAP_CLK	Swaps Dp / Dn channel of clock lane. If this bit is set, Dp and Dn channel swap each other.
24 DPDNSWAP_DAT	Swaps Dp / Dn channel of Data lanes. If this bit is set, Dp and Dn channel swap each other.
23 PLLEN	Enables PLL
22–20 Reserved	This field is reserved.
19–1 PMS[19:1]	Specifies the PLL PMS value
0 Reserved	This field is reserved.

13.4.4.25 PLL Control Register 1 (MIPI_DSI_PLLCTRL1)

This register configures D-PHY PLL control (M_PLLCTL[31:0]).

Address: 3076_0000h base + 98h offset = 3076_0098h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	M_PLLCTL0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_DSI_PLLCTRL1 field descriptions

Field	Description
M_PLLCTL0	M_PLLCTL[31:0] to D-PHY

13.4.4.26 PLL control register 2 (MIPI_DSI_PLLCTRL2)

This register configures D-PHY PLL control (M_PLLCTL[39:32]).

Address: 3076_0000h base + 9Ch offset = 3076_009Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																								M_PLLCTL1							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_DSI_PLLCTRL2 field descriptions

Field	Description
31–8 Reserved	This field is reserved.
M_PLLCTL1	M_PLLCTL[39:32] to D-PHY

13.4.4.27 PLL Timer Register (MIPI_DSI_PLLTMR)

Address: 3076_0000h base + A0h offset = 3076_00A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PLLTIMER																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

MIPI_DSI_PLLTMR field descriptions

Field	Description
PLLTIMER	Specifies the PLL Timer for stability of the generated clock (System clock cycle base). If the timer value goes to 0x00000000, the clock stable bit of status and interrupt register is set.

13.4.4.28 D-PHY Master and Slave Analog Block Control Register 1 (MIPI_DSI_PHYCTRL_B1)

D-PHY Master and Slave Analog block characteristics control registers (B_DPHYCTL).

Address: 3076_0000h base + A4h offset = 3076_00A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	B_DPHYCTL0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_DSI_PHYCTRL_B1 field descriptions

Field	Description
B_DPHYCTL0	B_DPHYCTL[31:0] to D-PHY

13.4.4.29 D-PHY Master and Slave Analog Block Control Register 2 (MIPI_DSI_PHYCTRL_B2)

D-PHY Master and Slave Analog block characteristics control registers (B_DPHYCTL).

Address: 3076_0000h base + A8h offset = 3076_00A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	B_DPHYCTL1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_DSI_PHYCTRL_B2 field descriptions

Field	Description
B_DPHYCTL1	B_DPHYCTL[63:32] to D-PHY

13.4.4.30 D-PHY Master Analog Block Control Register 1 (MIPI_DSI_PHYCTRL_M1)

D-PHY Master Analog block characteristics control registers (M_DPHYCTL).

Address: 3076_0000h base + ACh offset = 3076_00ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	M_DPHYCTL0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_DSI_PHYCTRL_M1 field descriptions

Field	Description
M_DPHYCTL0	M_DPHYCTL[31:0] to D-PHY

13.4.4.31 D-PHY Master Analog Block Control Register 1 (MIPI_DSI_PHYCTRL_M2)

D-PHY Master Analog block characteristics control registers (M_DPHYCTL).

Address: 3076_0000h base + B0h offset = 3076_00B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	M_DPHYCTL1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_DSI_PHYCTRL_M2 field descriptions

Field	Description
M_DPHYCTL1	M_DPHYCTL[63:32] to D-PHY

13.4.4.32 D-PHY Timing register (MIPI_DSI_PHYTIMING)

D-PHY Master global operating timing registers

Address: 3076_0000h base + B4h offset = 3076_00B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																M_TLPXCTL								M_THSEXITCTL							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_DSI_PHYTIMING field descriptions

Field	Description
31–16 Reserved	This field is reserved.
15–8 M_TLPXCTL	M_TLPXCTL[7:0] to D-PHY
M_THSEXITCTL	M_THSEXITCTL[7:0] to D-PHY

13.4.4.33 MIPI_DSI_PHYTIMING1

D-PHY Master global operating timing registers.

Address: 3076_0000h base + B8h offset = 3076_00B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	M_TCLKPRPRCTL								M_TCLKZEROCTL								M_TCLKPOSTCTL								M_TCLKTRAILCTL							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_DSI_PHYTIMING1 field descriptions

Field	Description
31–24 M_ TCLKPRPRCTL	M_TCLKPRPRCTL[7:0] to D-PHY
23–16 M_ TCLKZEROCTL	M_TCLKZEROCTL[7:0] to D-PHY
15–8 M_ TCLKPOSTCTL	M_TCLKPOSTCTL[7:0] to D-PHY
M_ TCLKTRAILCTL	M_TCLKTRAILCTL[7:0] to D-PHY

13.4.4.34 D-PHY Timing Register 2 (MIPI_DSI_PHYTIMING2)

D-PHY Master global operating timing registers

Address: 3076_0000h base + BCh offset = 3076_00BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								M_THSPRPRCTL								M_THSZEROCTL								M_THSTRAILCTL							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_DSI_PHYTIMING2 field descriptions

Field	Description
31–24 Reserved	This field is reserved.

Table continues on the next page...

MIPI_DSI_PHYTIMING2 field descriptions (continued)

Field	Description
23–16 M_ THSPRPRCTL	M_THSPRPRCTL[7:0] to D-PHY
15–8 M_ THSZEROCTL	M_THSZEROCTL[7:0] to D-PHY
M_ THSTRAILCTL	M_THSTRAILCTL[7:0] to D-PHY

13.5 MIPI CSI2 Host Controller (MIPI_CSI2)

13.5.1 Overview

The CSI-2 Host Controller is a digital core that implements all protocol functions defined in the MIPI CSI-2 Specification, providing an interface between the System and the MIPI D-PHY, allowing the communication with a MIPI CSI-2 compliant Camera Sensor.

13.5.1.1 Features

The following list summarizes the key features of the MIPI CSI2:

- Compliant to MIPI D-phy standard specification V1.1
  - Compliant to previous version of Samsung D-phy
- Compliant to MIPI CSI2 Standard Specification V1.01r06
- Support primary and secondary Image format
  - YUV420, YUV420 (Legacy), YUV420 (CSPS), YUV422 of 8-bits and 10-bits
  - RGB565, RGB666, RGB888
  - RAW6, RAW7, RAW8, RAW10, RAW12, RAW14
  - Compressed format: 10-6-10, 10-7-10, 10-8-10, 14-10-14
  - All of user defined byte-based Data packet
- Support embedded byte-based non-Image data packet and generic short packets
- Support interleave mode using Virtual channel
- Support up to two D-PHY Rx Data Lanes
- Interfaces
  - Compatible to PPI (Protocol-to-PHY Interface) in MIPI D-PHY Specification
  - Image output data bus width: 32 bit
  - Support four channel virtual channels or data interleave
- Memory

Non-image memory

- 8 KB SRAM for asynchronous clock domain
- This memory can store maximum 4 KB of non-image data per frame.

Image Memory

- 1KB SRAM for image memory
- This memory works as buffering for the difference of input / output bandwidth

- Pixel clock can be gated when no ppi data is coming.

13.5.1.2 Block diagram

13.5.1.2.1 Block diagram of camera system with CSIS V3.3

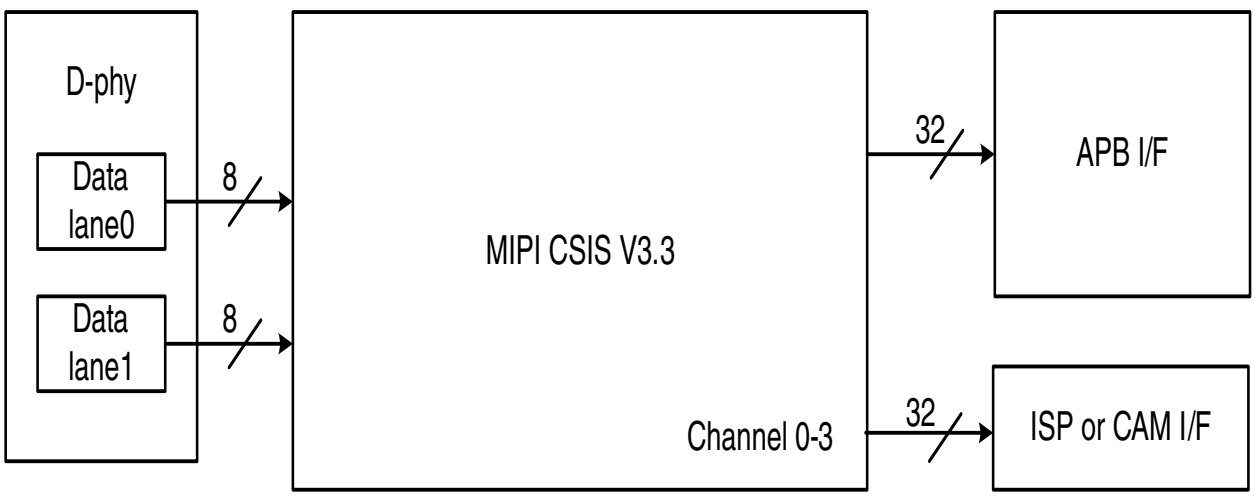


Figure 13-47. Block Diagram of Camera System with CSIS V3.3

13.5.1.2.2 MIPI CSI Slave and D-PHY I/F block diagram

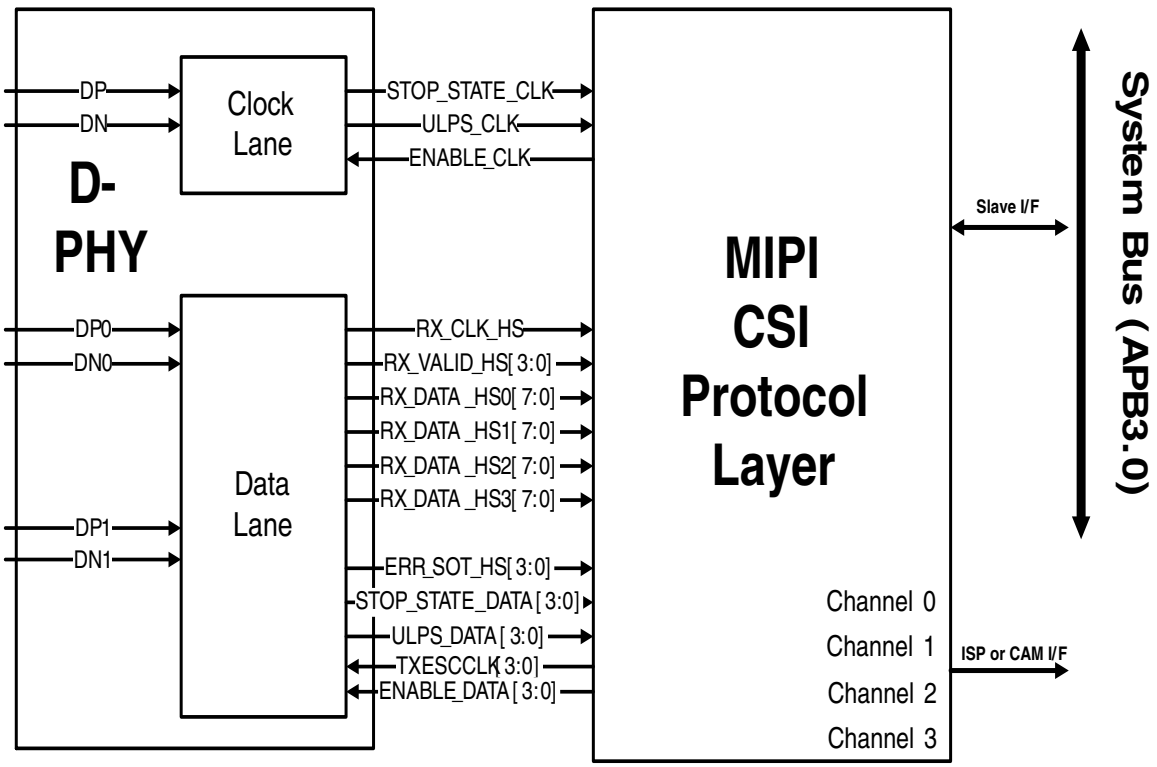


Figure 13-48. PPI I/F Block Diagram

13.5.1.2.3 MIPI CSI Slave block diagram

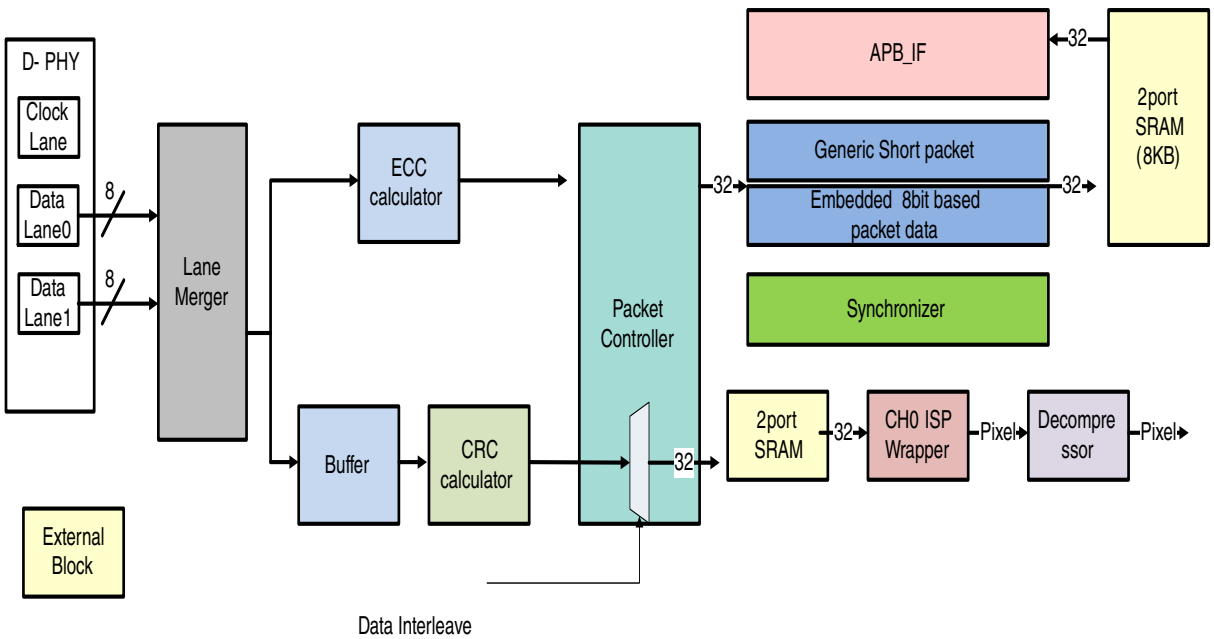


Figure 13-49. Block Diagram of CSIS V3.3 Slave Module

13.5.1.3 Internal memories

There are some configurable-sized Memories. The following table describes these Memories.

Table 13-16. Internal Memory List

Usage	Memory Type	Size	Description
Generic data	2 port SRAM	8 KB (2048 X 32 bits)	Store the Embedded data or Generic data (This memory can be split several memories and then need mux for those memories)
Image data	2 port SRAM	1 KB (256 x 32 bits)	Store the Image data of Virtual Channel

There is the timing diagram for read memory in the following figure. If the mux for the split memories is needed, register for storing the address is necessary, because the RDATA of memory is prepared at one clock later when the REN and RADDR are accepted.

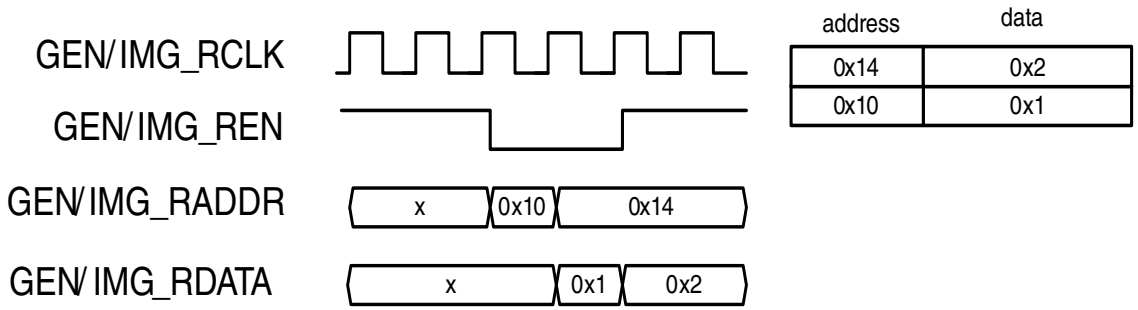


Figure 13-50. Timing Diagram of Read Memory

13.5.2 External Signals

The following table describes the external signals of MIPI CSI:

Table 13-17. MIPI CSI External Signals

Signal	Description	Pad	Mode	Direction
MIPI_CSI_CLK_N	D-PHY Negative D-Phy differential clock line Receiver input	MIPI_CSI_CLK_N	No muxing	I
MIPI_CSI_CLK_P	D-PHY Positive D-Phy differential clock line Receiver input	MIPI_CSI_CLK_P	No muxing	I
MIPI_CSI_D0_N	D-PHY Negative D-Phy differential data line Receiver input	MIPI_CSI_D0_N	No muxing	I
MIPI_CSI_D0_P	D-PHY Positive D-Phy differential data line Receiver input	MIPI_CSI_D0_P	No muxing	I
MIPI_CSI_D1_N	D-PHY Negative D-Phy differential data line Receiver input	MIPI_CSI_D1_N	No muxing	I
MIPI_CSI_D1_P	D-PHY Positive D-Phy differential data line Receiver input	MIPI_CSI_D1_P	No muxing	I

13.5.3 Functional Description

13.5.3.1 Interface and protocol

13.5.3.1.1 D-PHY layer FSM

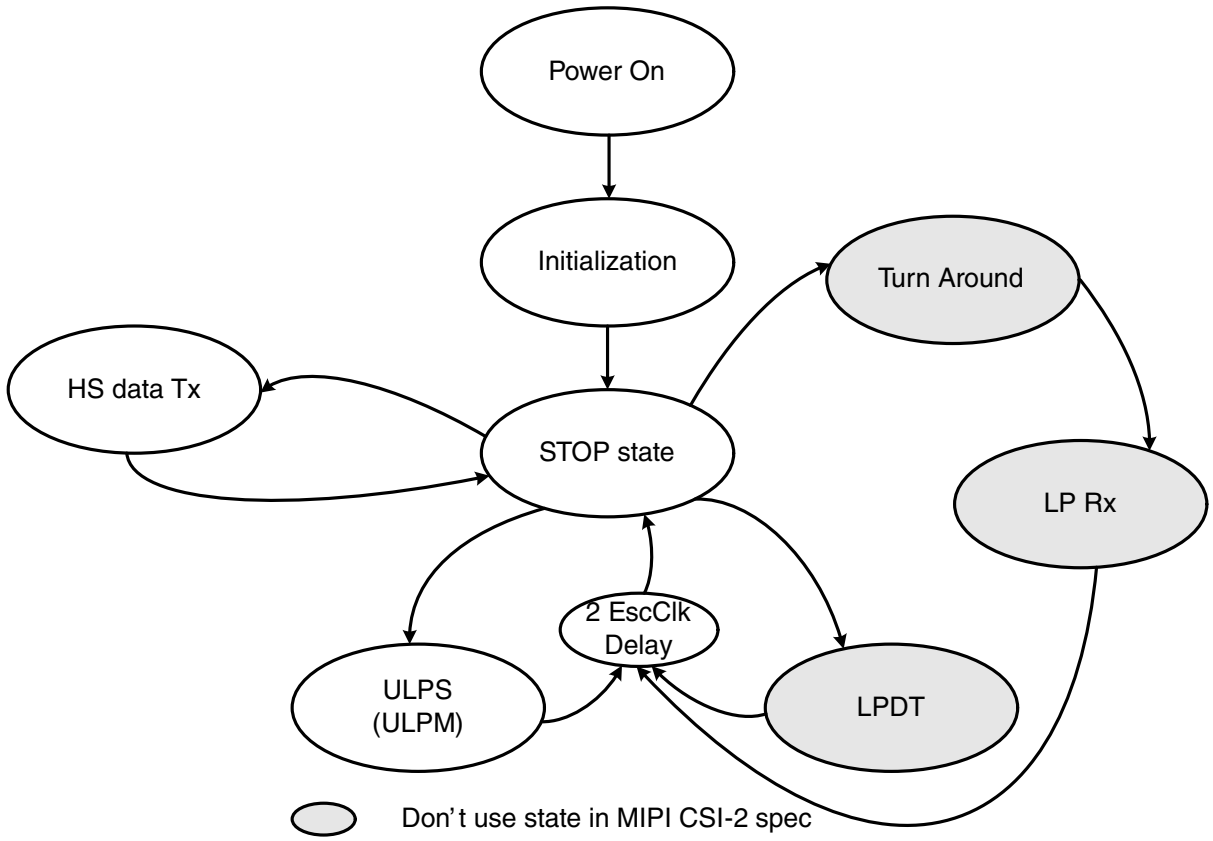


Figure 13-51. D-PHY Finite State Machine

CSIS V3.3 system supports HSDT (High Speed Data Transfer) and ULPS (ULPM–Utra-Low Power State or Mode) only. There is no trigger function, LPDT, and BTA.

13.5.3.1.2 Interface timing and protocol

13.5.3.1.2.1 PPI interface timing and protocol

CSIS V3.3 supports HSDT and ULPM.

13.5.3.1.2.1.1 High speed data transfer

In the following figure, the upper five signals are related with clock lane and the lower six signals are related with data lane. Dp and Dn of upper signals are D-phy channel of clock lane. BYTE_CLK is generated clock in the D-PHY. Dp and Dn of lower signals are

Data lane. Another signals of lower are Phy Status signals. STOPstate (and STOPstateClk) indicates that differential channel state of D-phy is LP11 (the voltage level of Dp and Dn is 1.2 V)

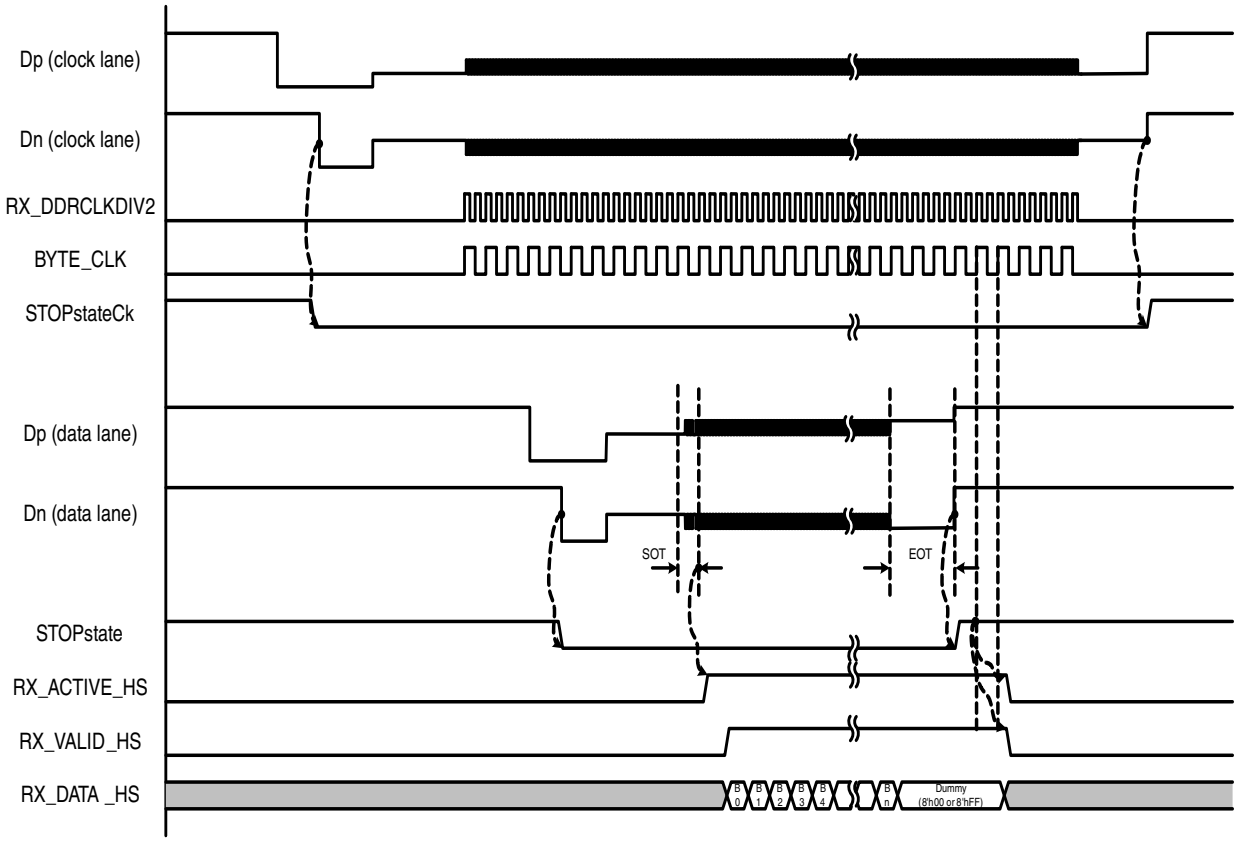


Figure 13-52. Timing Diagram of High Speed Data Transfer

13.5.3.1.2.1.2 Ultra-low power mode

In the following figure, UlpsActiveNot* and STOPstate* is in PPI. ULPS command generated by D- PHY Tx is only for D-PHY Rx, it is decoded by D-PHY Rx and transferred to Link with Ulps data.

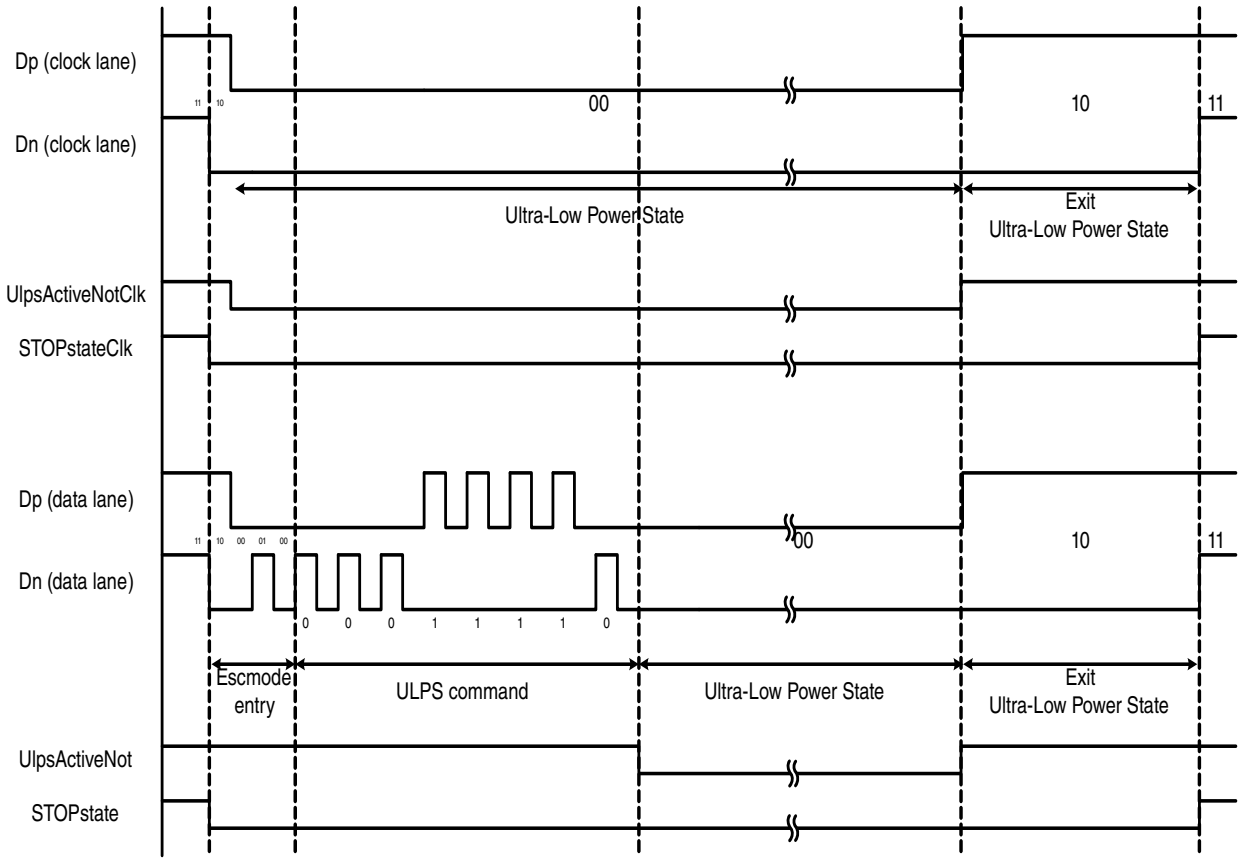


Figure 13-53. Timing Diagram of Ultra Low Power Mode

13.5.3.1.2.2 ISP (CAM I/F) interface

CSIS V3.3 has output signals, which are PIX_CLK, VVALID, HVALID, BVALID, DATA , StopReqEn, and input signal which is StopReq. PIX_CLK is output pixel clock what is generated from PCLK or EXTCLK (WRAP_CLK). VVALID is vertical sync signal. HVALID is horizontal sync signal. BVALID indicates the valid number of bytes, which is only used in User Defined type. DATA is image data bus. DATA bus width is dependent on Image format. Maximum bus width is 24 bits, which is RGB888. The following figure describes the output protocol of MIPI CSIS. All signal s are synchronized with the rising edge of PIX_CLK.

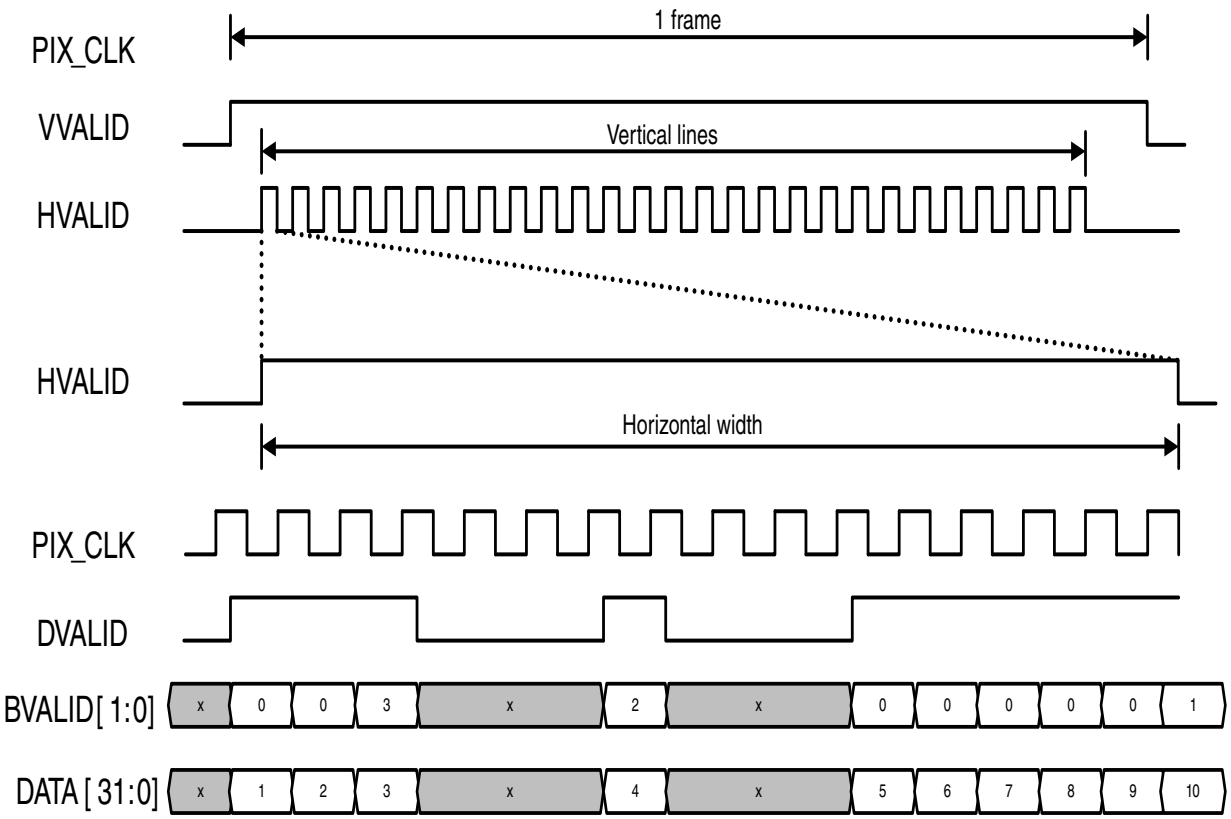


Figure 13-54. Output Protocol of ISP Wrapper of CSIS V3.3

The following figure describes timing diagram of ISP wrapper output protocol.

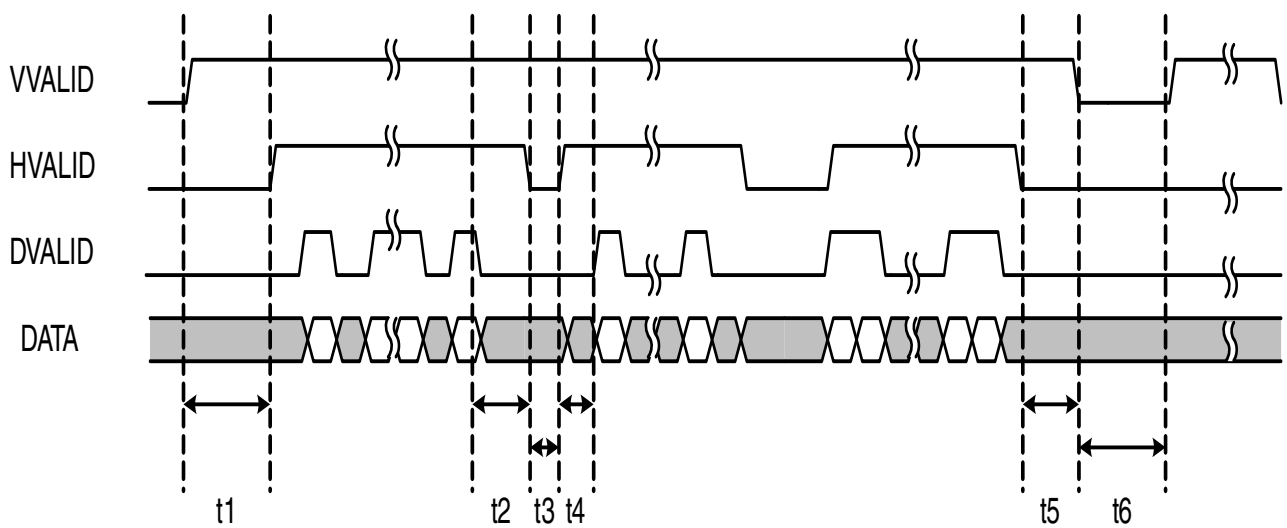


Figure 13-55. Timing Diagram of ISP Wrapper Output Protocol

The following table is a timing table of output protocol.

Table 13-18. Description of Output Protocol

	Description	Minimum Cycle of Pixel Clock	Maximum Cycle of Pixel clock
t1	Interval between rising of VVALID and first rising of HVALID	Vsync_SIntv + 1 (1 ~ 64)	—
t2	Interval between last falling of DVALID and falling of HVALID	Hsync_LIntv + 1 (1 ~ 6 4)	—
t3	Interval between falling of HVALID and rising of next HVALID	1	—
t4	Interval between rising of HVALID and first rising of DVALID	0	—
t5	Interval between last falling of HVALID and falling of VVALID	Vsync_EIntv (0 ~ 4095)	—
t6	Interval between falling of VVALID and rising of next VVALID	1	—

13.5.3.1.2.2.1 Handshaking with back pressure

In the following figure, there are handshaking signals which are StopReqEn and StopReq to stall data flow when the Post-IP needs more clock cycles to process its computation. The handshaking is called as back pressure interface.

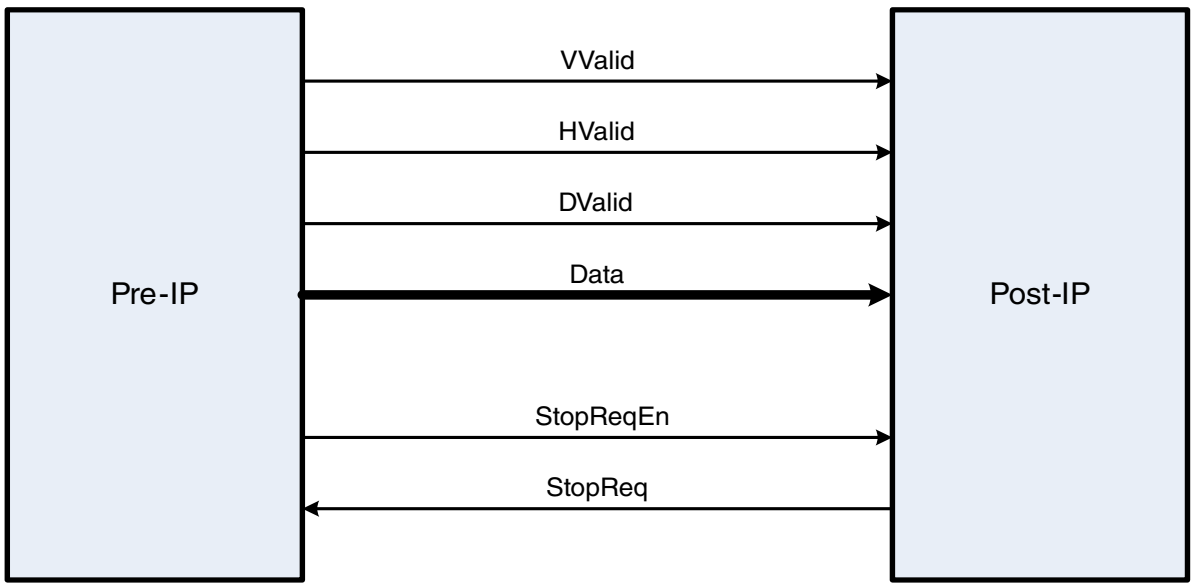


Figure 13-56. Interface between Two IPs

Under the stoppable condition (StopReqEn and StopReq = 1), sync signals (VVALID, HVALID, DVALID, and BVALID) are not changed. Only after releasing of the stoppable condition, sync signals can be changed. In the following figure, (b) shows that StopReq is ignored when StopReqEn is not active. StopReqEn is high when the room of memory is greater than IMG_MEM_FULL_GAP, because of the delay of full signal propagation, IMG_MEM_FULL_GAP must not be set under 4 to avoid memory overflow.

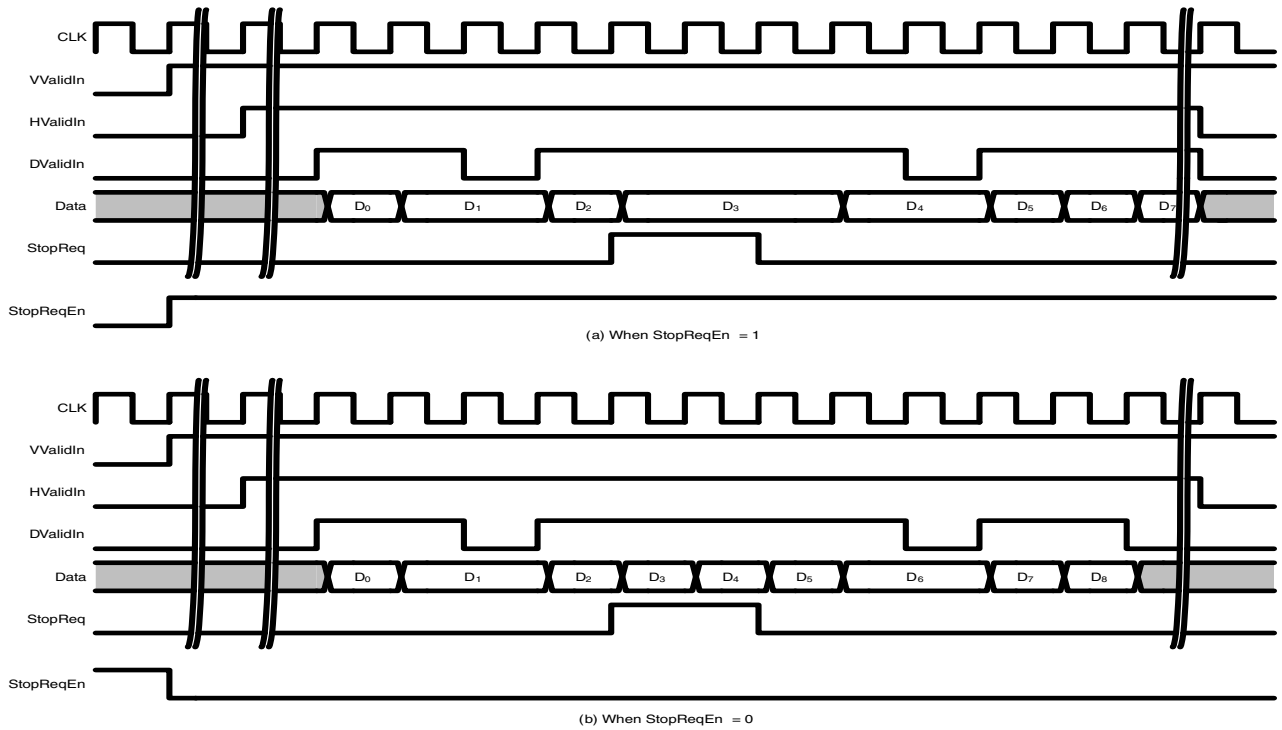


Figure 13-57. Timing Diagram According to StopReqEn and StopReq

13.5.3.1.2.2.2 Normal Data alignment in output Data bus

The following figure describes data alignment of RGB formats.

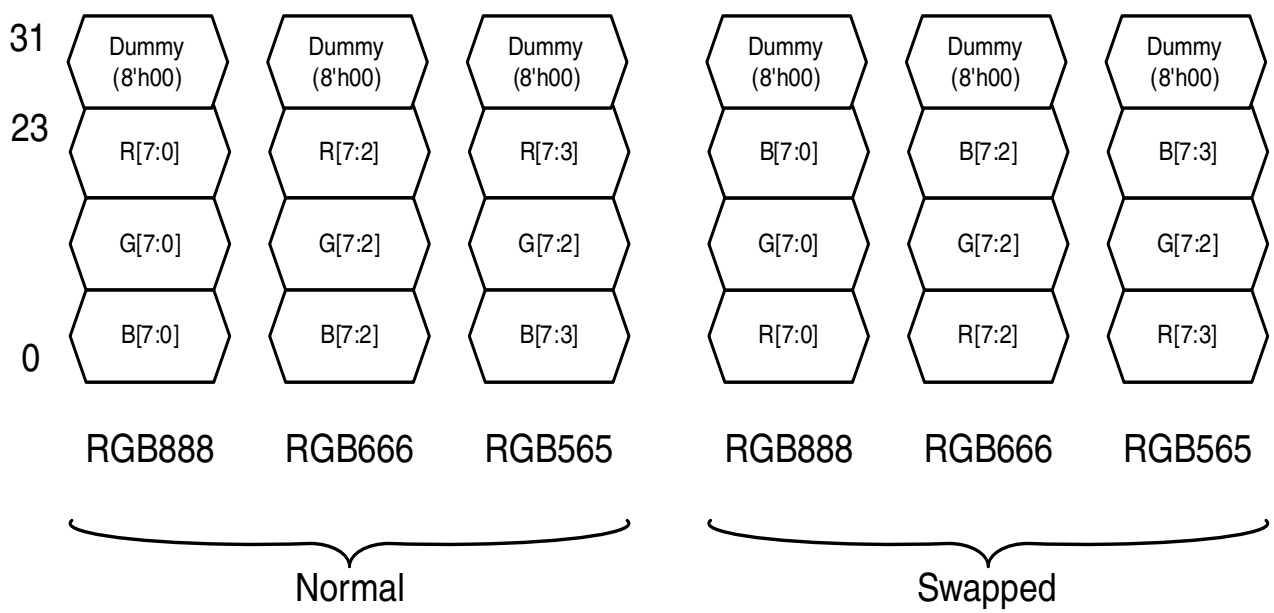


Figure 13-58. RGB Data Alignment

The following figure describes data alignment of RAW and YUV formats.

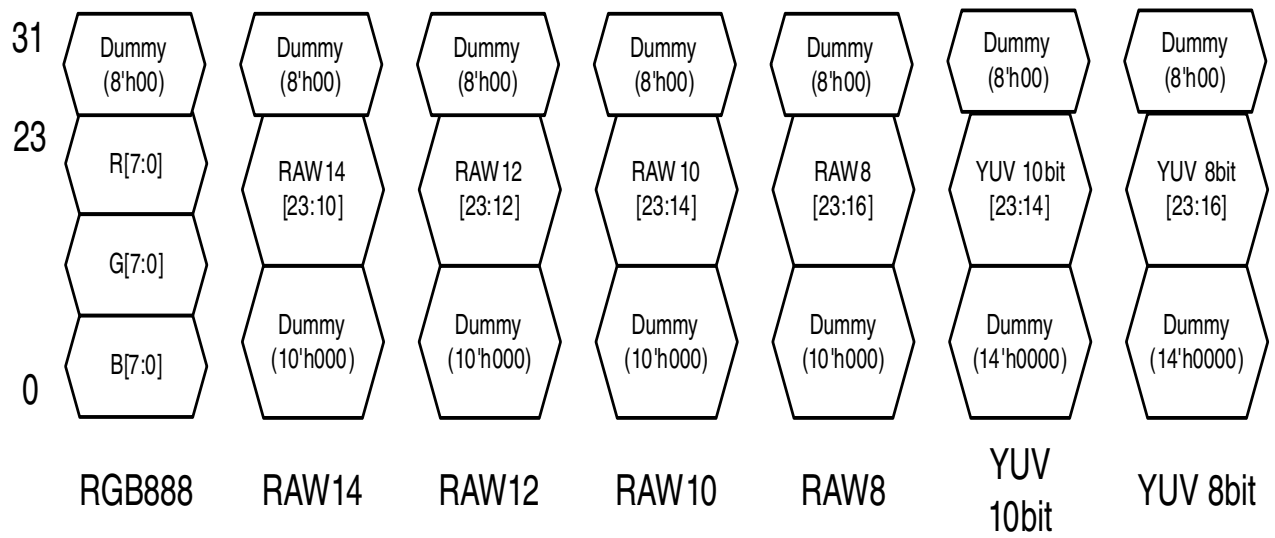


Figure 13-59. RAW and YUV Data Alignment

The following table describes present of each bit in image data bus.

Table 13-19. MIPI CSIS Output Format in Bus

Bit #	RGB Format			RAW Format				YUV Format		User Defined Packet
888	666	565	14	12	10	8	10	8		
31	These bits are used in parallel data out mode.									B3[7]
30	See bit20 of CSIS_CTRL register.									B3[6]
29	(When data format is User defined format,									B3[5]
28	these fields are always filled by valid data.)									B3[4]
27										B3[3]
26										B3[2]
25										B3[1]
24										B3[0]
23	R[7]	R[5]	R[4]	P[13]	P[12]	P[9]	P[7]	P[9]	P[7]	B2[7]
22	R[6]	R[4]	R[3]	P[12]	P[11]	P[8]	P[6]	P[8]	P[6]	B2[6]
21	R[5]	R[3]	R[2]	P[11]	P[10]	P[7]	P[5]	P[7]	P[5]	B2[5]
20	R[4]	R[2]	R[1]	P[10]	P[9]	P[6]	P[4]	P[6]	P[4]	B2[4]
19	R[3]	R[1]	R[0]	P[9]	P[8]	P[5]	P[3]	P[5]	P[3]	B2[3]
18	R[2]	R[0]		P[8]	P[7]	P[4]	P[2]	P[4]	P[2]	B2[2]
17	R[1]			P[7]	P[6]	P[3]	P[1]	P[3]	P[1]	B2[1]
16	R[0]			p[6]	P[5]	P[2]	P[0]	P[2]	P[0]	B2[0]
15	G[7]	G[5]	G[5]	P[5]	P[4]	P[1]		P[1]		B1[7]

Table continues on the next page...

Table 13-19. MIPI CSIS Output Format in Bus (continued)

Bit #	RGB Format			RAW Format				YUV Format		User Defined Packet
14	G[6]	G[4]	G[4]	P[4]	P[3]	P[0]		P[0]		B1[6]
13	G[5]	G[3]	G[3]	P[3]	P[2]					B1[5]
12	G[4]	G[2]	G[2]	P[2]	P[1]					B1[4]
11	G[3]	G[1]	G[1]	P[1]	P[0]					B1[3]
10	G[2]	G[0]	G[0]	P[0]						B1[2]
9	G[1]									B1[1]
8	G[0]									B1[0]
7	B[7]	B[5]	B[4]							B0[7]
6	B[6]	B[4]	B[3]							B0[6]
5	B[5]	B[3]	B[2]							B0[5]
4	B[4]	B[2]	B[1]							B0[4]
3	B[3]	B[1]	B[0]							B0[3]
2	B[2]	B[0]								B0[2]
1	B[1]									B0[1]
0	B[0]									B0[0]

Table 13-20. Output Image Stream of YUV Formats

YUV Format	Line	Image Stream of Content
YUV420 legacy	Odd	U1 Y1 Y2 U3 Y3 Y4 U5 Y5 ...
	Even	V1 Y1 Y2 V3 Y3 Y4 V5 Y5 ...
YUV420	Odd	Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8 ...
	Even	U1 Y1 V1 Y2 U3 Y3 V3 Y4...
YUV422	Odd / Even	U1 Y1 V1 Y2 U3 Y3 V3 Y4...

13.5.3.1.2.2.3 Parallel-memory-storing data alignment in output data bus

When the “Parallel” bit of CSIS_CTRL register is set, the outer bus width is 32 bits.

13.5.3.1.2.2.4 Double component per clock cycle in YUV422 8-bit / 10-bit format

When the DOUBLE_CMPNT bit of CSIS_CTRL register is set, the outer bus width is 16 or 20 bits. User can refer to the following table and figure.

Table 13-21. Double Component Mode in YUV422

Bit #	YUV format	
10	8	

Table continues on the next page...

Table 13-21. Double Component Mode in YUV422 (continued)

Bit #	YUV format	
31	These bits are used in parallel data out mode.	
30	See bit20 of CSIS_CTRL register.	
29	(When data format is User defined format,	
28	these fields are always filled by valid data.)	
27		
26		
25		
24		
23	PIXEL_Y[9]	PIXEL_Y[7]
22	PIXEL_Y[8]	PIXEL_Y[6]
21	PIXEL_Y[7]	PIXEL_Y[5]
20	PIXEL_Y[6]	PIXEL_Y[4]
19	PIXEL_Y[5]	PIXEL_Y[3]
18	PIXEL_Y[4]	PIXEL_Y[2]
17	PIXEL_Y[3]	PIXEL_Y[1]
16	PIXEL_Y[2]	PIXEL_Y[0]
15	PIXEL_U[9]/PIXEL_V[9]	PIXEL_U[7]/PIXEL_V[7]
14	PIXEL_U[8]/PIXEL_V[8]	PIXEL_U[6]/PIXEL_V[6]
13	PIXEL_U[7]/PIXEL_V[7]	PIXEL_U[5]/PIXEL_V[5]
12	PIXEL_U[6]/PIXEL_V[6]	PIXEL_U[4]/PIXEL_V[4]
11	PIXEL_U[5]/PIXEL_V[5]	PIXEL_U[3]/PIXEL_V[3]
10	PIXEL_U[4]/PIXEL_V[4]	PIXEL_U[2]/PIXEL_V[2]
9	PIXEL_U[3]/PIXEL_V[3]	PIXEL_U[1]/PIXEL_V[1]
8	PIXEL_U[2]/PIXEL_V[2]	PIXEL_U[0]/PIXEL_V[0]
7	PIXEL_Y[1]	
6	PIXEL_Y[0]	
5	PIXEL_U[1]/PIXEL_V[1]	
4	PIXEL_U[0]/PIXEL_V[0]	
3		
2		
1		
0		

The following figure describes the waveform of double component mode.

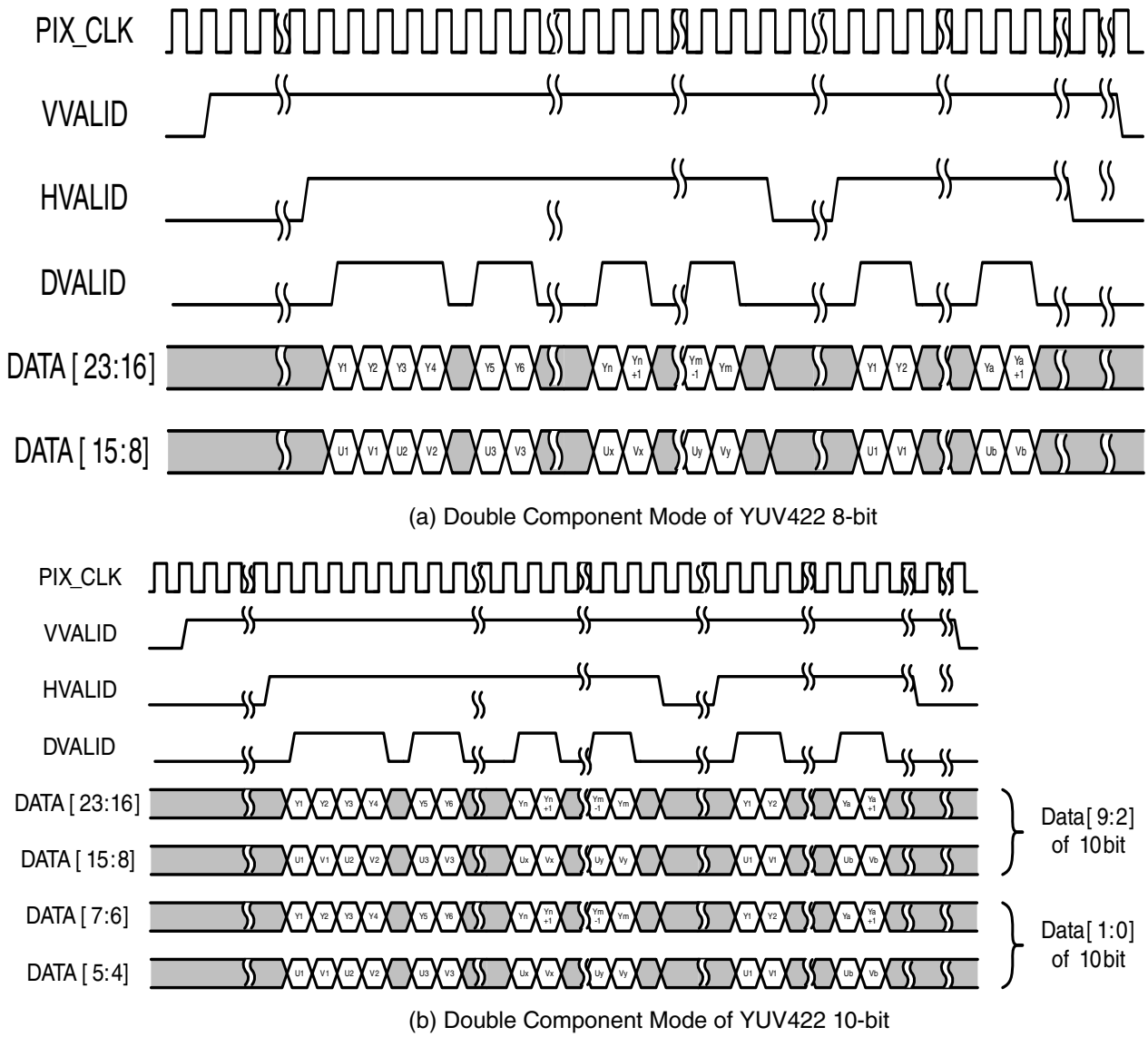


Figure 13-60. Waveform of Double Component Mode

13.5.3.2 Configuration

13.5.3.2.1 Image resolution

MIPI CSI Slave block needs the configuration of Image resolution to measure Hsync pulse length exactly and detect frame end.

Vertical resolution register has 16 bits (16'h0001 ~ 16'hFFFF).

Horizontal resolution register has 16 bits (16'h0001 ~ 16'hFFFF).

13.5.3.2.2 Image data format

In CSIS V3.3, Image data format can be configured for transferring image data to ISP or CAM I/F.

13.5.3.2.2.1 YUV format

MIPI CSI2 supports YUV formats: YUV420 8-bit, YUV420 10-bit, YUV420 Legacy, YUV420 8-bit Chroma Shifted Pixel Sampling, YUV420 10-bit Chroma Shifted Pixel Sampling, YUV422 8-bit, and YUV422 10-bit.

The following table describes Data type of YUV packets.

Table 13-22. Data Type of YUV Packets

Data Type	Description
0x18	YUV420 8-bit
0x19	YUV420 10-bit
0x1A	Legacy YUV420 8-bit
0x1B	Reserved
0x1C	YUV420 8-bit CSPS (Chroma Shifted Pixel Sampling)
0x1D	YUV420 10-bit CSPS (Chroma Shifted Pixel Sampling)
0x1E	YUV422 8-bit
0x1F	YUV422 10-bit

13.5.3.2.2.2 RGB format

MIPI CSI2 supports RGB formats: RGB444, RGB555, RGB565, RGB666, and RGB888.

The following table describes data type of RGB packets.

Table 13-23. Data Type of RGB Packets

Data Type	Description
0x20	RGB444 (not support)
0x21	RGB555 (not support)
0x22	RGB565
0x23	RGB666
0x24	RGB888
0x25	Reserved
0x26	
0x27	

13.5.3.2.2.3 RAW format (Bayer RGB)

MIPI CSI2 supports RAW formats: RAW6, RAW7, RAW8, RAW10, RAW12, and RAW14.

The following table describes data type of RAW packets.

Table 13-24. Data Type of RAW Packets

Data Type	Description
0x28	RAW6
0x29	RAW7
0x2A	RAW8
0x2B	RAW10
0x2C	RAW12
0x2D	RAW14
0x2E	Reserved
0x2F	

13.5.3.2.2.4 User defined Byte-based format

CSIS V3.3 supports eight formats of user defined Byte-based data packet.

The following table describes data type of user defined packets.

Table 13-25. Data Type of RAW Packets

Data Type	Description
0x30	User defined Byte-based packet 1
0x31	User defined Byte-based packet 2
0x32	User defined Byte-based packet 3
0x33	User defined Byte-based packet 4
0x34	User defined Byte-based packet 5
0x35	User defined Byte-based packet 6
0x36	User defined Byte-based packet 7
0x37	User defined Byte-based packet 8

13.5.3.2.2.5 Generic format

CSIS V3.3 supports Generic Short and Long data type packet through only Virtual Channel 0. For the lower size and the Generic type data is not used frequently, it is limited that only the VC 0 supports the Generic format packet.

The intention of the Generic Short Packet data is to provide timing information for the opening / closing of shutters, triggering of flashes, etc. And the Embedded 8-bit contains the additional information of the picture frame.

All the Generic type data in a frame is stored at the SRAM and system is required to pop the data. For the double buffering, the embedded data can be used up to the half of SRAM size (EX: If SRAM is 8 KB allocated for the generic data type then each frame can use up to 4 KB generic data).

Table 13-26. Generic Short / Long Data Type

Data Type	Description
0x08	Generic Short Packet Code 1
0x09	Generic Short Packet Code 2
0x0A	Generic Short Packet Code 3
0x0B	Generic Short Packet Code 4
0x0C	Generic Short Packet Code 5
0x0D	Generic Short Packet Code 6
0x0E	Generic Short Packet Code 7
0x0F	Generic Short Packet Code 8
0x12	Embedded 8-bit non Image data (Generic long type)

13.5.3.2.2.6 Null and blanking data

For both the null and blanking data types CSIS V3.3 ignore the content of the packet payload data.

Table 13-27. Null and Blanking Data Type

Data Type	Description
0x10	Null
0x11	Blanking Data

13.5.3.3 User defined packet format

The User Defined Data Type values can be used to transmit arbitrary byte-based data, such as JPEG and 1226 MPEG4 data over the CSI-2 bus.

The packet data size in bits can be divisible by 8. For example, whole number of bytes can be transmitted.

For User Defined data:

- The frame is transmitted as a sequence of arbitrary sized packets.
- The packet size is vary from packet to packet.
- The packet spacing is vary between packets.

13.5.3.3.1 ISP / CAM I/F for user defined packet

For supporting user defined packet, MIPI CSIS puts out more signals: 2 bits of byte validation.

Byte validation signal indicates what is valid byte in data of word size. In the following figure, n is 0, 1, 2, or 3.

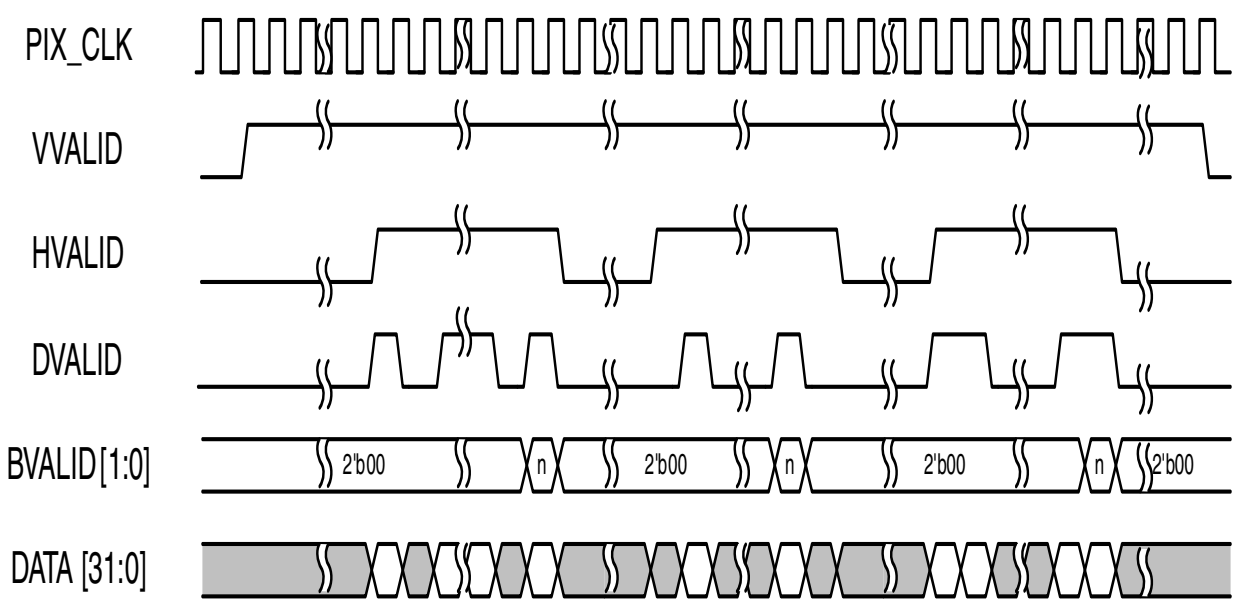


Figure 13-61. Output Protocol of CSIS V3.3 in User Defined Format

The following table describes BVALID in detail.

Table 13-28. Description Table of BVALID[1:0]

BVALID[1:0]	Description
2'b00	All bytes in data bus are valid.
2'b01	Only byte 0 is valid.
2'b10	2 bytes from LSB are valid.
2'b11	3 bytes from LSB are valid.

13.5.3.3.2 Pixel clock of ISP wrapper for user defined packet

Since CSIS V3.3, Byteclk is not used for Pixel clock even in the user defined format.

13.5.3.4 Decompressor for Bayer RGB

In Annex E of MIPI CSI-2 standard spec, compress formula is described. CSIS V3.3 can support 6 / 7 / 8-to-10 formats and 10-to-14 formats.

There are two types of determining predictor to decompress which are Simple and Advanced method (Refer 5.2 and 5.8's decomp_predict field). The Simple method is intended to minimize processing power and it is typically used with 8-10, 7-10 schemes. The Advanced method is more complex and provide slightly better quality, so it is usually used with 6-10 and 10-14 schemes. CSIS V3.3 can support both method at 6 / 7-10, 10-14 schemes, and only simple method at 8-10 scheme.

The following figure describes the block diagram of Decompressor logic in CSIS V3.3.

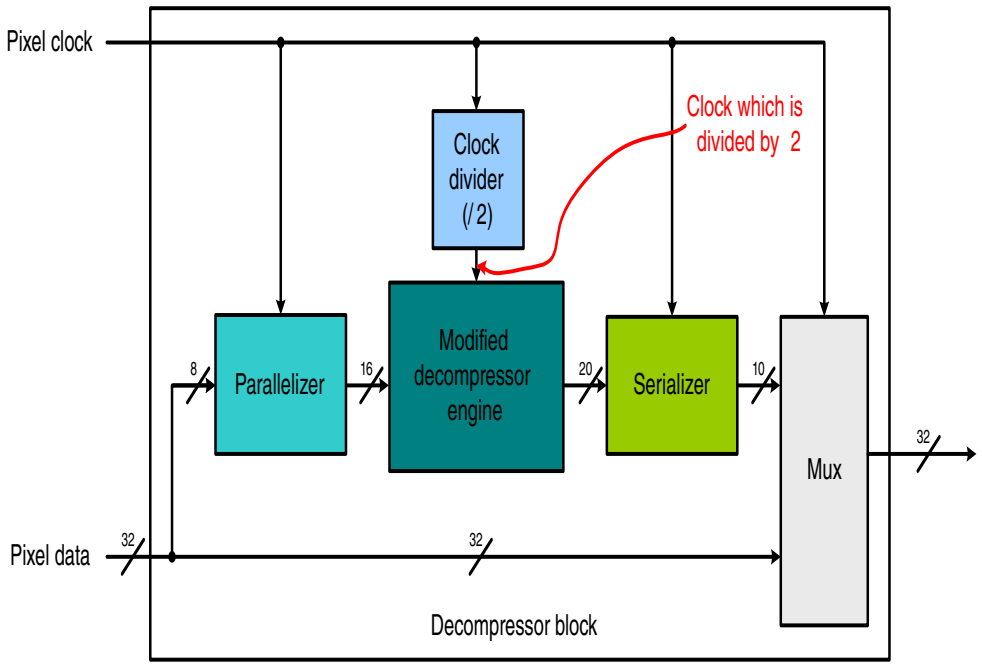


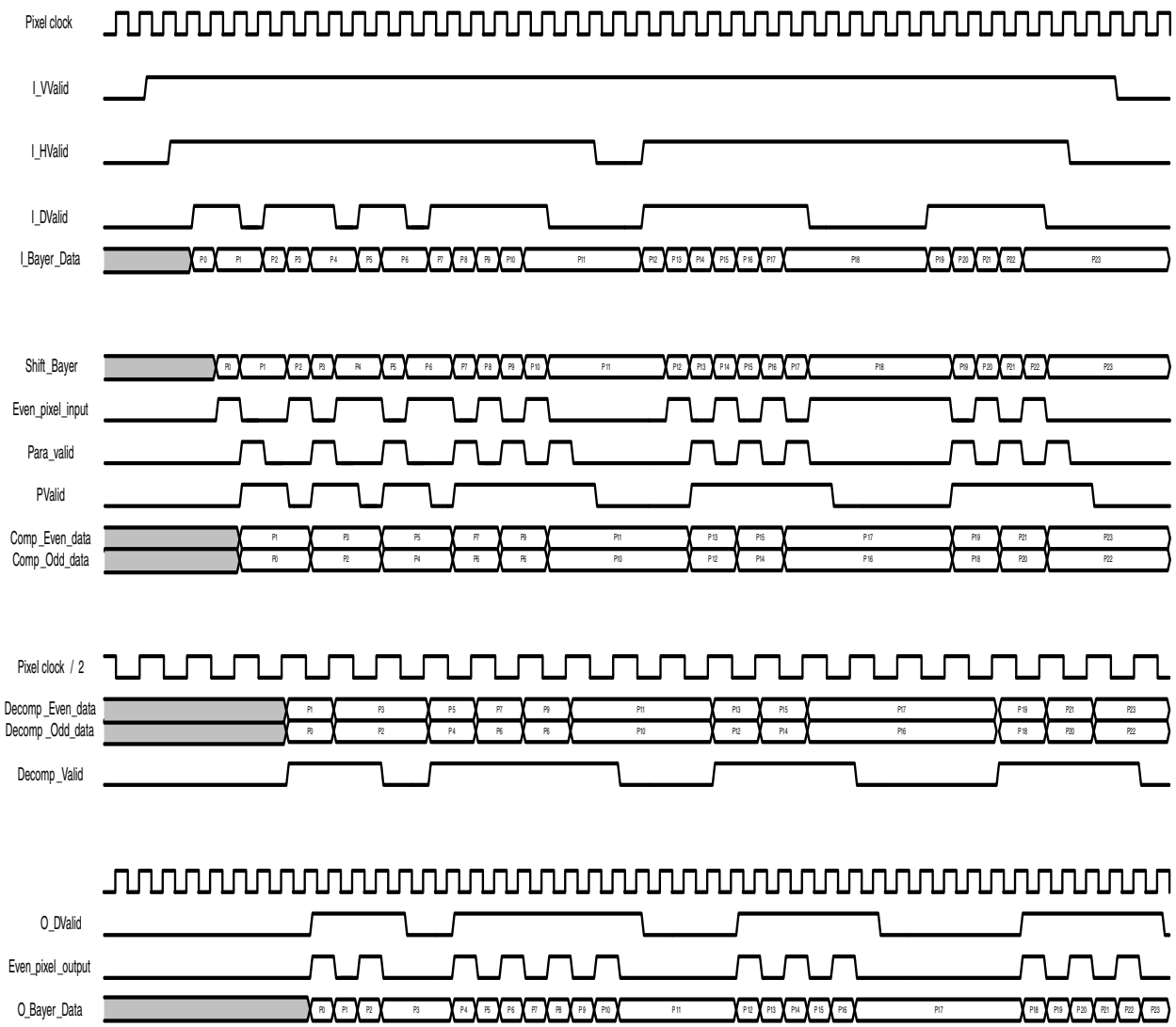
Figure 13-62. Block Diagram of Decompressor

Decompressor engine was made by SysLSI Image team. It was modified for using Dvalid. Another logic of Decompressor was designed for processing two pixel per clock because of decompressor engine.

When camera module can send compressed Bayer RGB, user can be set CSIS_CTRL and CSIS_CONFIG registers. Turn the Decompress enable and format in CSIS_CTRL register on. And then set the compressed format (RAW6, 7, 8) before turning camera module on.

The following table is an example of wave of decompressor.

Figure 13-63. Waveform of Decompressor



1st group Figure 13-62 of is input signals of Decompressor. 2nd group of Figure 13-63 is signals of parallelizer of Decompressor. 3rd group of Figure 13-63 is output of Decompressor engine. The other group is output of Decompressor.

13.5.3.5 Interrupt

CSIS V3.3 has many interrupts for checking status, indicating error case and receiving generic data.

13.5.3.5.1 Odd_Before / Odd_After / Even_Before / Even_After

These interrupts are related with generic and embedded data.

Odd_Before and Odd_After interrupts are generated in odd frame. Even_Before and Even_After interrupts are generated in even frame. Odd_Before and Even_Before interrupts are generated when Generic Short packet or Embedded 8-bit based packet is received before image data (Vertical Back porch area).

13.5.3.6 Synchronization short packet data type

Image Frame begins with a Frame Start and finishes with a Frame End packet. With these Frame synch packet, CSIS V3.3 generate Vertical Sync signal (VVALID). The Line synchronization packet (Line Start and Line End) generate Horizontal Sync signal (HVALID). But it can be generated by image data packet so that CSIS V3.3 does not support the Line Synchronization packets (0x02 ~ 0x03) and ignore it if it is received.

Table 13-29. Synchronization Short Packet Data Type

Data Type	Description
0x00	Frame Start Code
0x01	Frame End Code
0x02	Line Start Code (Not support)
0x03	Line End Code (Not support)
0x04 to 0x07	Reserved

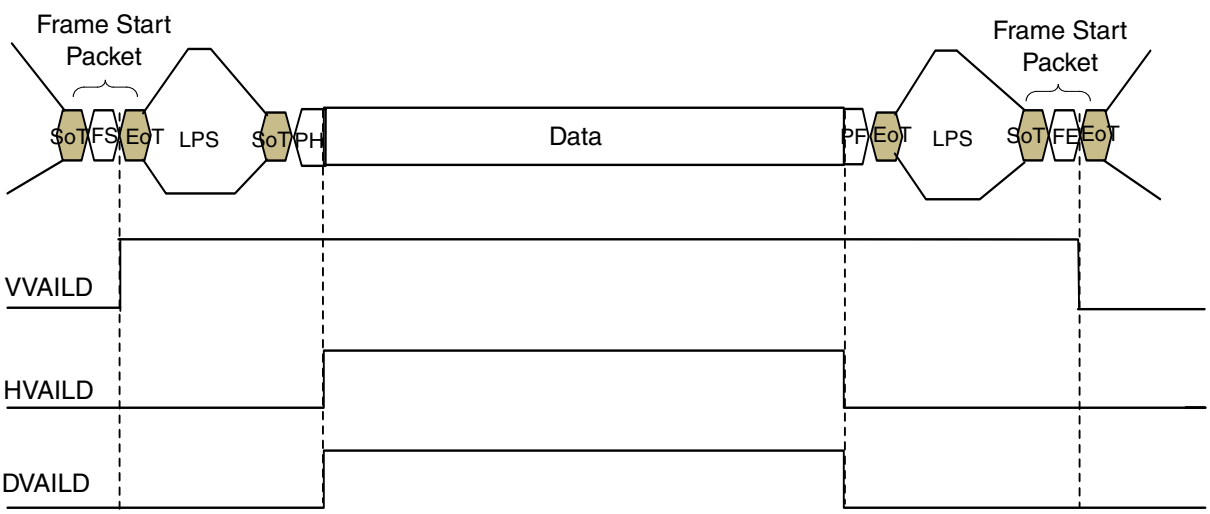


Figure 13-64. Synchronization Example

13.5.3.7 Clock gating for Pixel clock in the idle state

Since CSIS V3.3, it gates the Pixel clock when the target Channel is in idle state. In the following figure , the CLK_GATE signal is triggered at negative edge of pixel clock. The CLOCK_OPEN signal is from VVALID in the Packet Controller. So, Gated pixel clock is alive during the whole frame. The CLOCK_CLOSE signal is set when the O_VVALID of ISP wrapper set to zero, then GATE_FSM generate trail clock for flushing of the F/F in the ISP or CAM. After generating all the trail clocks, the GATE_PIXEL_CLOCK is closed. If the CLOCK_OPEN is set during the TRAIL_CLK state then it quit the state and enters the open state immediately.

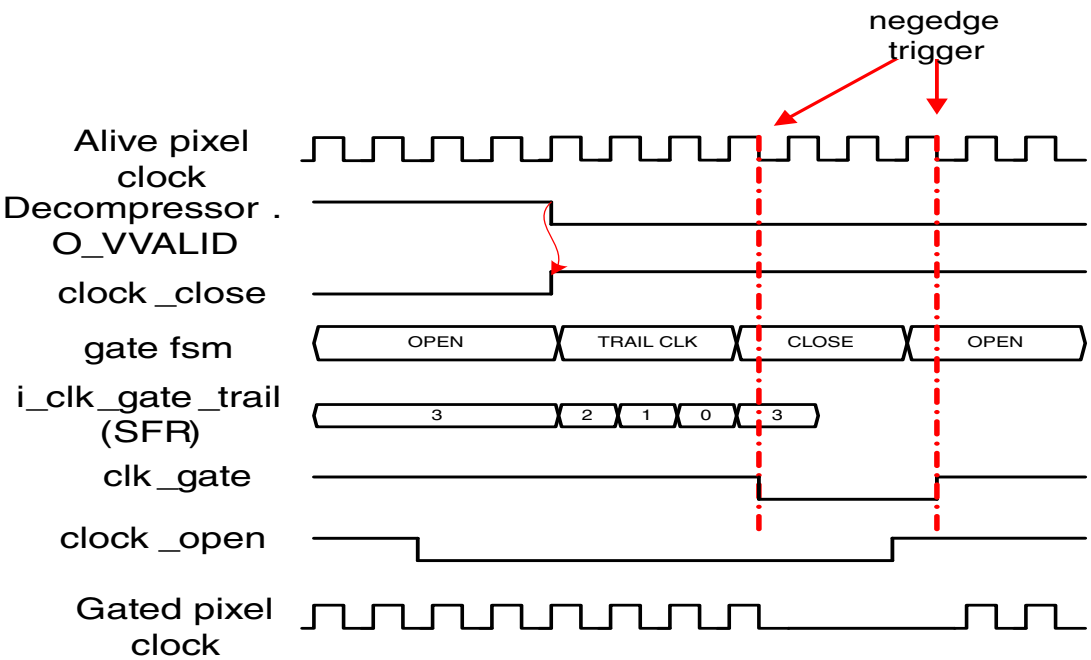


Figure 13-65. Pixel Clock Gating Timing

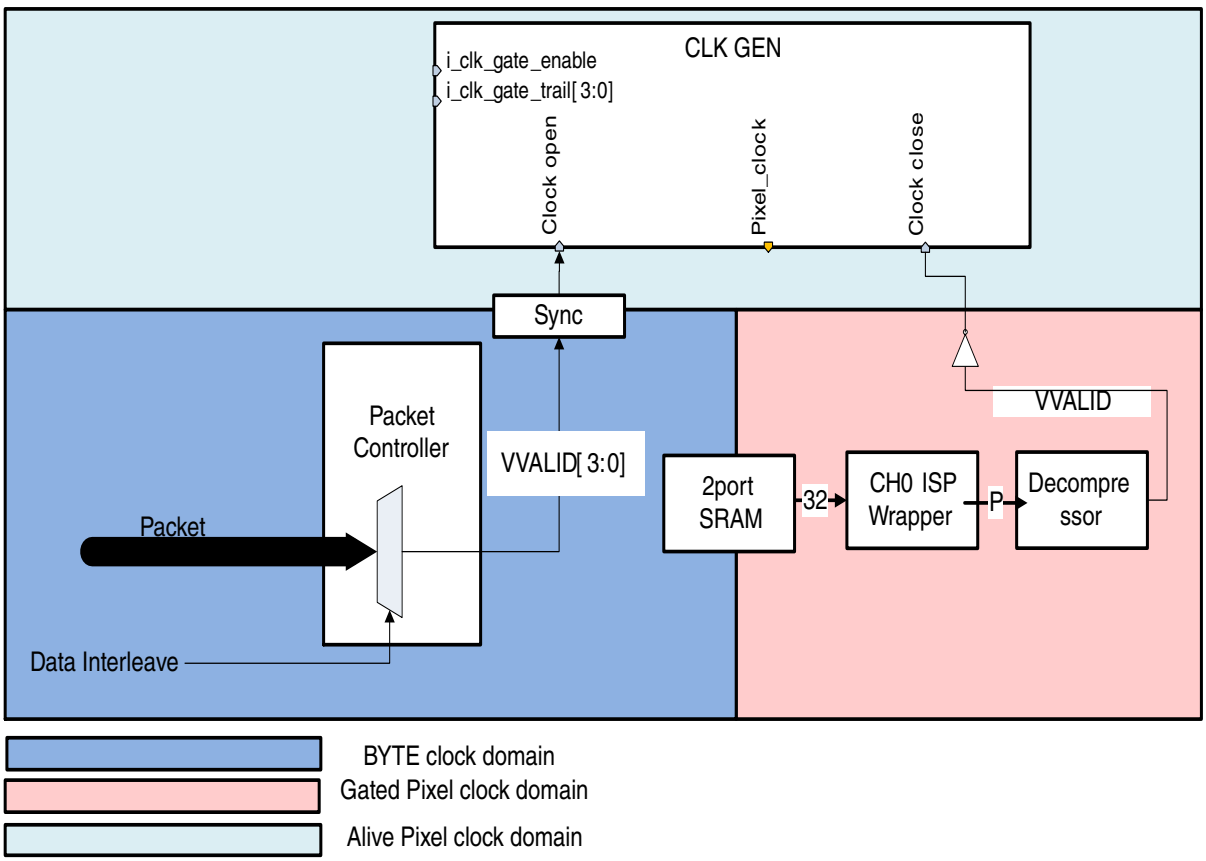


Figure 13-66. Clock Gating Block

13.5.3.8 Shadow update

The configurations related to ISP registers (which are dataformat, vci, resolution and so on.) cannot be varied during the run-time. So they are updated by programming 'UPDATE_SHADOW' (in CSIS_CMN_CTRL register) in idle state. The scheme describes in the following figure.

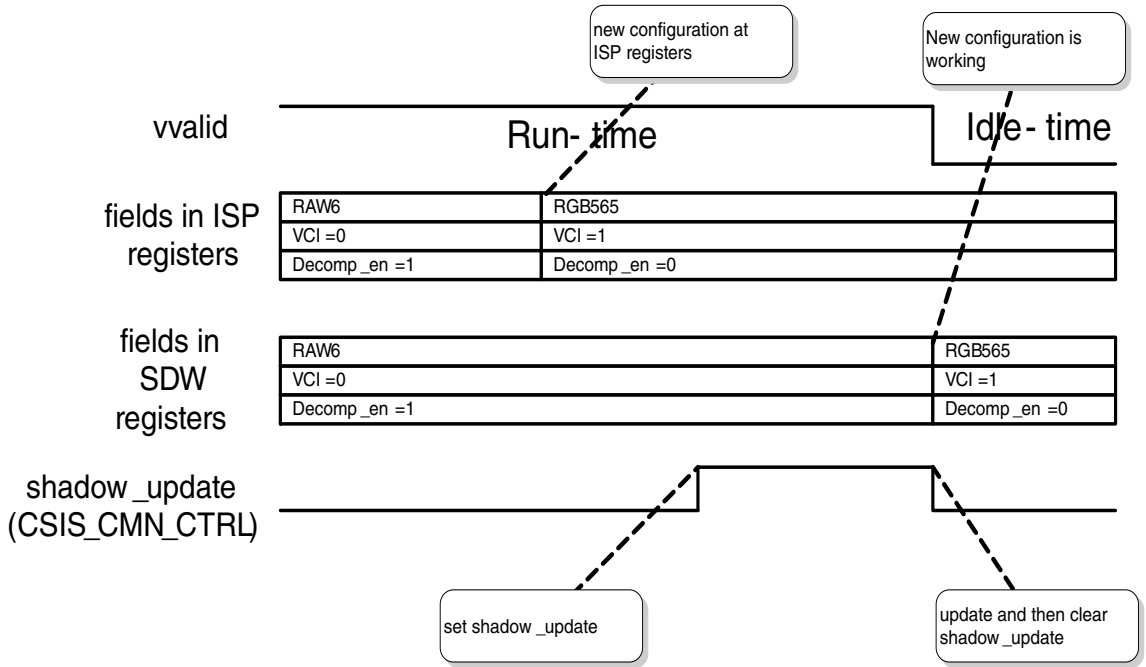


Figure 13-67. Operation of SHADOW_UPDATE

If the frame interleaving scheme(interleave_mode in CSIS_CMN_CTRL = 2 or 3) is used then the instance of entering IDLE_STATE at each ISP channels may be different. In this case, updating shadow registers can be delayed until that all ISP channels are in idle state by setting UPDATE_SHADOW_CTRL.

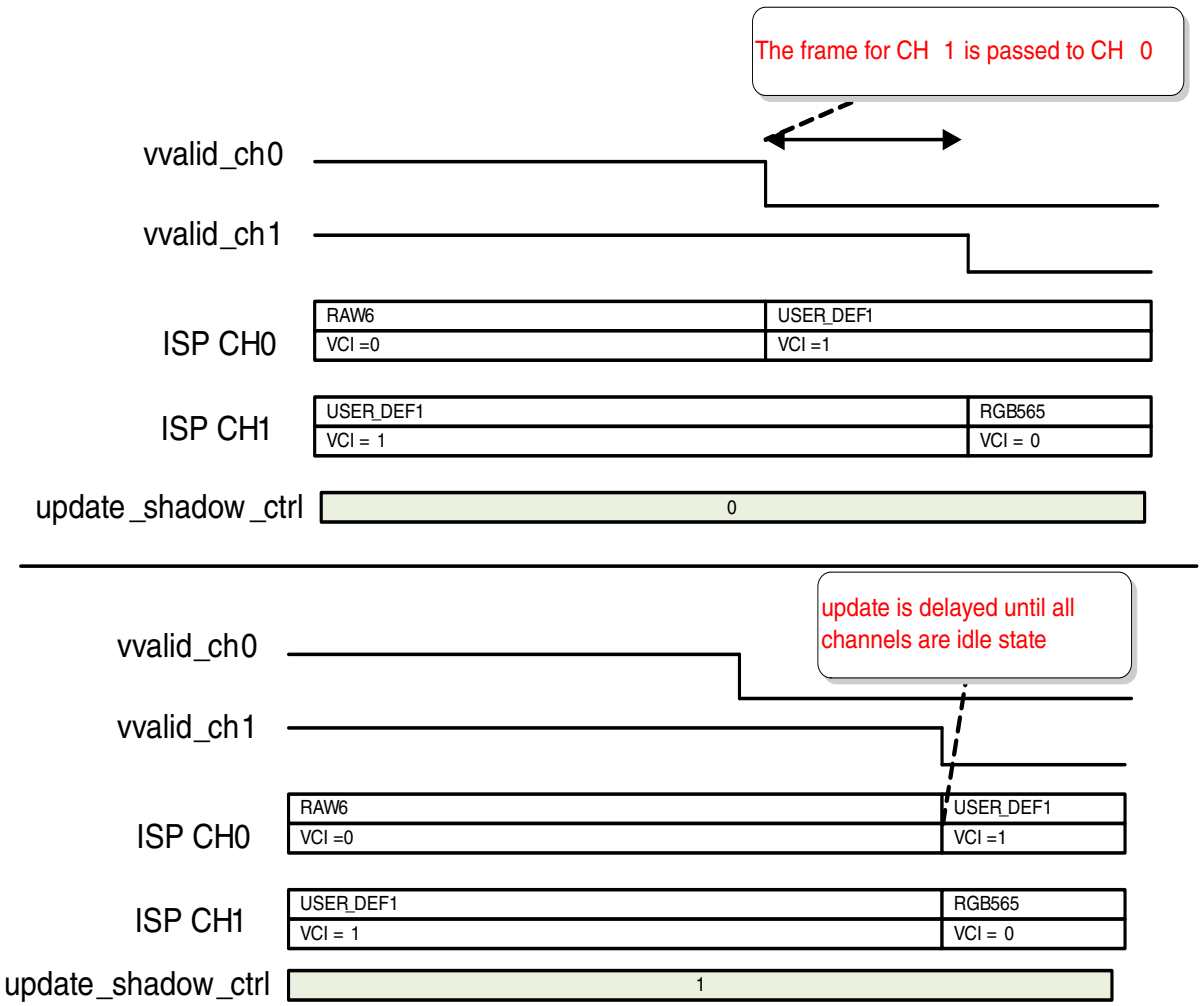


Figure 13-68. Operation of SHADOW_UPDATE_CTRL

13.5.3.9 Guide of MEM_FULL_GAP

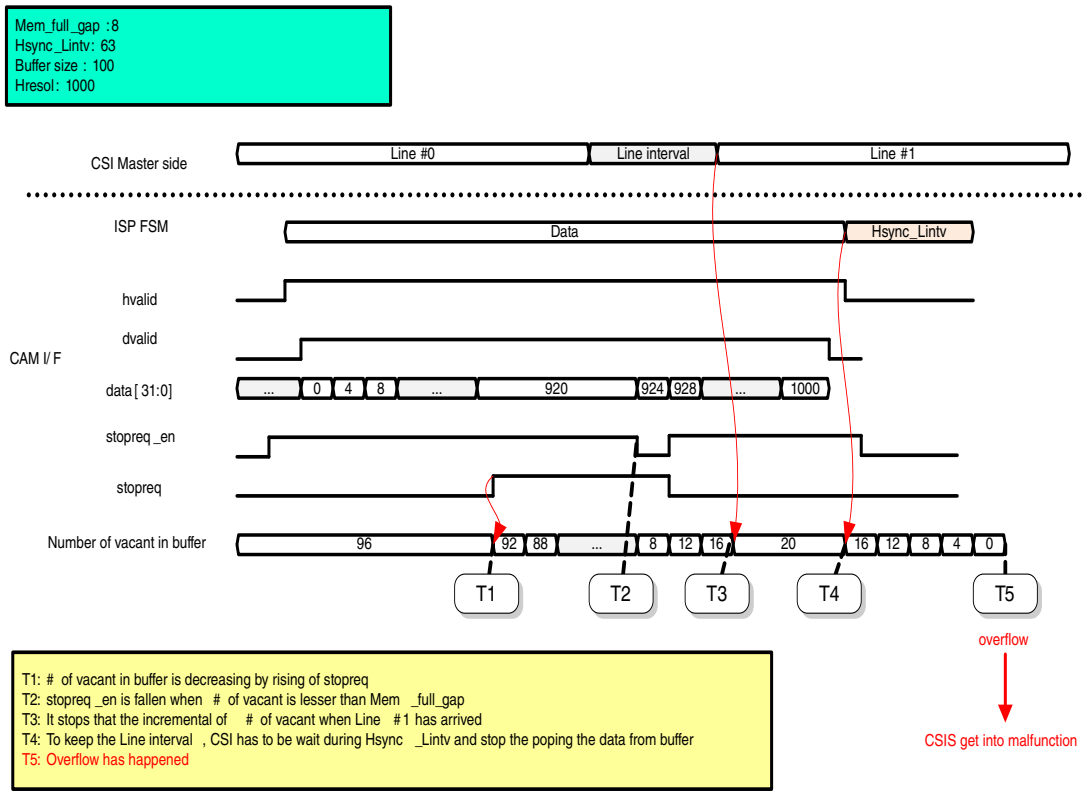


Figure 13-69. Relation Between MEM_FULL_GAP and Sync Interval

Related with Handshaking with back pressure, MEM_FULL_GAP in ISP configuration register should be set carefully because CSIS will be in malfunction if overflow has happened. MEM_FULL_GAP plays a role of preventing buffer overflow by poping out data from the buffer when the empty room of the buffer is lesser than the number of MEM_FULL_GAP. In the HSYNC_LINTV state, it is stopped to pop out data from the buffer. So, the minimum value of the MEM_FULL_GAP should be greater than the amount of data stored during the HSYNC_LINTV state. The related formula is following:

MEM_FULL_GAP ≥ (HSYNC_LINTV + 1) * Pixel clock period / Byte clock period + 8

NOTE

Propagation delay factor

For example:

Condition: HSYNC_LINTV = 31, Pixel clock = 666 MHz, Byte clock = 250 MHz MEM FULL_GAP: (31 + 1) * (1 / 666) / (1 / 250) = 12

'Don't stop last line' plays a role of preventing abnormal operation by ignoring stop request when the last line is transferred. So, do not change the 'Do not stop last line' values (default = 4'b1111) except for debugging.

13.5.3.10 Clock Specification

MIPI CSI may have three clock sources: RX_BYTE_CLK_HS0, I_PCLK, and I_WRAP_CLK.

I_PCLK	PCLK is system clock generated by general processor PLL.
RX_BYTE_CLK_HS0	This signal comes from data lane 0 of D-phy.
I_WRAP_CLK	I_WRAP_CLK is external clock for pixel clock what is for using ISP or CAM I/F.
I_DTLPCLK	D-phy Scan Test clock which is needed at 45 nm process (since 32 nm D-PHY has its own Test clock).

13.5.3.10.1 Clock domain

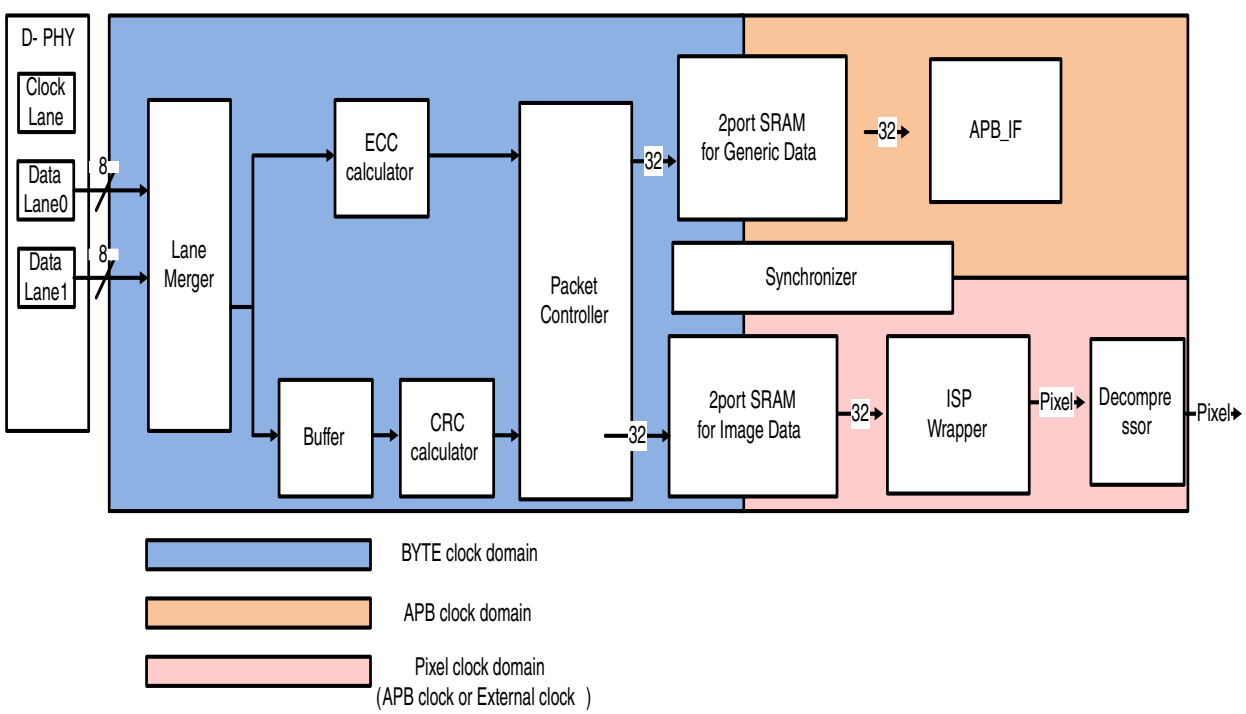


Figure 13-70. Bock Diagram of Clock Domain

Three number of clock domain of CSIS V3.3 is:

- BYTE clock (RX_BYTE_CLK_HS0)
- External clock (I_WRAP_CLK)
- APB clock (I_PCLK)

Pixel clock uses one of those clocks (Wrapper, APB), since CSIS V3.3 does not use BYTE clock for Pixel clock in the User defined data type.

The relationship between input and output bandwidth is that the output bandwidth should be faster than input bandwidth. There is an equation of previous relationship.

(freq. of RX_BYTE_CLK_HS) * (number of data lane) * 8 bits ≤ (freq. of Pixel clock) * (bitwidth of image format)

13.5.3.11 Interface Signals

13.5.3.11.1 Reset and clock signal

Table 13-30. Variation of Clock and Reset

Clock and Reset	Frequency	Clock Generator	Usage
I_PRESETn	—	—	Global Asynchronous reset
I_PCLK	—	System controller	Main control clock (system dependent)
RX_BYTE_CLK_HS0	up to 187.5 MHz (1.5 G) 125 MHz (1 G)	Data lane 0 of D-phy	MIPI CSI processes data with ByteCLK
I_WRAP_CLK	up to 200 MHz	External clock source	Output pixel clock
I_DTLPCLK	up to 200 MHz	External test device	For D-phy scan test mode

13.5.3.11.2 PPI (PHY-Protocol Interface) signals

BandCtrl and HSzeroCtl signals of following table are additional signals for D-phy timing configuration.The other PPI signals in the following table are compatible with MIPI D-phy spec.

Table 13-31. Protocol-to-Phy Interface Signal List

Signal	I/O	Description
Clock lane		
STOP_STATE_CLK	I	Stop state indicator (active high)
ULPS_CLK	I	ULPS state indicator (active low)
ENABLE_CLK	O	Clock lane enabler
Data lane (lane0 ~ 1 common, % = 0 and 1)		

Table continues on the next page...

Table 13-31. Protocol-to-Phy Interface Signal List (continued)

Signal	I/O	Description
RX_BYTE_CLK_HS0	I	Byte Clock
RX_VALID_HS%	I	Received Data is valid.
RX_DATA_HS%[7:0]	I	Data bus for HS transfer
STOP_STATE %	I	Stop state indicator (active high)
ULPS_DAT%	I	ULPS state indicator (active low)
ERR_SOT_HS%	I	Error flag of Start of transmission in HS mode.
ENABLE_D%	O	Data lane enabler
D-Sphy test		
I_DPHY_TESTMODE	I	D-phy test mode
I_DTLPCLK	I	D-phy Test clock
TXESCCLK%	O	Escape mode clock for D-phy test
D-phy config		
O_HS_SETTLE[7:0]	O	Configuration for tuning HS settle time
O_S_CLKSETTLECTRL[1:0]	O	Configuration for D-phy control
O_B_DPHYCTRL[63:0]	O	Configuration for D-phy ac characteristics
O_S_DPHYCTRL[63:0]	O	Configuration for D-phy ac characteristics
O_S_DPDN_SWAP_CLK	O	Swapping Dp and Dn channel of clock lane in D-phy.
O_S_DPDN_SWAP_DAT	O	Swapping Dp and Dn channel of data lanes in D-phy.

13.5.3.11.3 AMBA signal

Table 13-32. APB slave Interface signal list

Signal	I/O	Description
APB Slave		
I_PCLK	I	Main system clock
I_PRESETn	I	Main system reset
I_PADDR[31:0]	I	System address
I_PSEL	I	Module select strobe
I_PWRITE	I	Write strobe
I_PENABLE	I	Enable strobe
I_PWDATA[31:0]	I	Write data to slave module
O_PREADY	O	Slave module is ready (but tied high: always ready).
O_PSLVERR	O	Slave module with error
O_PRDATA[31:0]	O	Read data from slave module

13.5.3.11.4 Image signal

Image output signals are duplicated for output channel 1 to 3.

Table 13-33. Image Output Interface Signal List

Signal	I/O	Description
ISP or CAM I/F		
O_P IX C LK[0]	O	Pixel clock for ISP or CAM I/F This is pixel clock for transferring image date to ISP or CAM I/F. The default clock of this is PCLK. If user wants to assign another clock, Set the WCLK_SRC of CSIS_CNFG register. The relationship of bandwidth between input and output is that output bandwidth should be faster than input bandwidth. The frequency of this clock is determined as following equation. (# of data lane) * (frequency of RX_BYTE_CLK_HS) ≤ (bitwidths of image format) * (frequency of PIXCLK). O_PIX_CLK[0] are worked when the corresponding channels are used
O_VVALID [0]	O	Vertical sync (level) O_VVALID[0] is worked when the corresponding channels are used.
O_HVALID [0]	O	Horizontal sync (level) O_HVALID[0] is worked when the corresponding channels are used.
O_DVALID [0]	O	Data valid O_DVALID[0] is worked when the corresponding channels are used.
O_BVALID[1:0]	O	Byte valid O_BVALID[1:0] is worked when the corresponding channels are used.
O_DATA[31:0]	O	Pixel data O_DATA[31:0] is worked when the corresponding channels are used.

13.5.3.11.5 Sideband signal

Table 13-34. Image Sideband signal list

Signal	I/O	Description
Interrupt		
O_ INT_CSI S	O	Interrupt request signal
I_ INTGEN_SEL	I	Interrupt type selection signal: • 0: Pulse interrupt generation • 1: Level interrupt generation
External clock source		
I_W RAP_ CLK	I	External clock input for wrapper

13.5.3.11.6 Memory I/F

Table 13-35. Memory I/F

Signal	I/O	Description
Memory for Generic data		

Table continues on the next page...

Table 13-35. Memory I/F (continued)

Signal	I/O	Description
O_GEN_MEM_WCLK	O	Write clock (ByteClk)
O_GEN_MEM_WEN	O	Generic data Write enable strobe
O_GEN_MEM_WADDR[10:0]	O	Generic data Write Address
O_GEN_MEM_WDATA[31:0]	O	Generic data Write Data bus
O_GEN_MEM_RCLK	O	Read clock (PCLK)
O_GEN_MEM_RADDR[10:0]	O	Generic data Read Address
O_GEN_MEM_REN	O	Generic data Read enable strobe
I_GEN_MEM_RDATA[31:0]	I	Generic data Read Data bus
Memory for Image data		
O_IMG_MEM_WCLK	O	Write clock (ByteClk)
O_IMG_MEM_WEN	O	Image data Write enable strobe
O_IMG_MEM_WADDR[7:0]	O	Image data Write Address
O_IMG_MEM_WDATA[32:0]	O	Image data Write Data bus
O_IMG_MEM_RCLK	O	Read clock (PIXCLK)
O_IMG_MEM_RADDR[7:0]	O	Image data Read Address
O_IMG_MEM_REN	O	Image data Read enable strobe
I_IMG_MEM_RDATA[32:0]	I	Image data Read Data bus

13.5.3.11.7 Test signal

Table 13-36. Test Related Signal List

Signal	I/O	Description
I_ STMODE	I	Scan Test mode
I_ TCLK_WRAP	I	Scan Test Clock for WRAP clock domain This port is used for testing SCAN at the speed of WRAP clock. If user uses only 1 SCAN test net, this port should be connected to TCLK which is SCAN test clock.
I_ TCLK_WRAP_HALF	I	Scan Test Clock for WRAP / 2 clock domain This port is used for testing SCAN at the half speed of WRAP clock. If user uses only 1 SCAN test net, this port should be connected to TCLK which is SCAN test clock.
I_ TCLK_BYTE	I	Scan Test Clock for BYTE clock domain This port is used for testing SCAN at the speed of BYTE clock. If user uses only 1 SCAN test net, this port should be connected to TCLK which is SCAN test clock.

13.5.3.12 Application Scenario

13.5.3.12.1 Programming model

Table 13-37. Behavior Programming in Each Case

No.	Pseudo Code	Description
Initialize		
1	APB_WRITE(CSIS_INT R_ MSK, -)	Set Interrupt Mask
2	APB_WRITE(DPHY _B CTRL_ L , -)	Set DPHYBCTRL LSB
3	APB_WRITE(DPHY _B CTRL_ H , -)	Set DPHYBCTRL MSB
4	APB_WRITE(DPHY _CMN_ CTRL, 0x11)	Set DPHY HSSETTLE
5	APB_WRITE(ISP _ CONFIG_CH0 , -)	Set ISP_CH0 configuration
6	APB_WRITE(ISP_RESOL_CH0 , -)	Set ISP_CH0 resolution
7	APB_WRITE(ISP _ SYNC_CH0 , -)	Set ISP_CH0 sync
8	APB_WRITE(ISP_CONFIG _CH1, -)	Set ISP_CH1 configuration
9	APB_WRITE(ISP_RESOL _CH1, -)	Set ISP_CH1 resolution
10	APB_WRITE(ISP_SYNC _CH 1 , -)	Set ISP_CH1 sync
11	APB_WRITE(CSIS_CLK_CTRL, -)	Set CSIS clock control
12	APB_WRITE(CSIS_CMN_CTRL, -)	Set CSIS common control (enable CSI)
Run in Data transfer		

13.5.3.12.2 Enable decompressor

Table 13-38. Config CH0 with RAW10 Decomp-en (10 - 14 Decompressor)

No.	Pseudo Code	Description
Initialize		
1	APB_WRITE (ISP_RESOL_CH0, 0x0280_01E0)	Set CH0 Resol (640x480)
2	APB_WRITE (ISP_CONFIG_CH0, 0x0800_03AC)	Set CH0 Config (RAW10, VC:0, Fullgap:8, advanced predict decomp enabled)
3	APB_WRITE (CSIS_CMN_CTRL, 0x0001_0F01)	Set CSIS CMN CTRL (enable system, 4lane, VCI and DT interleave, ISP0 update)
Run in Data transfer		

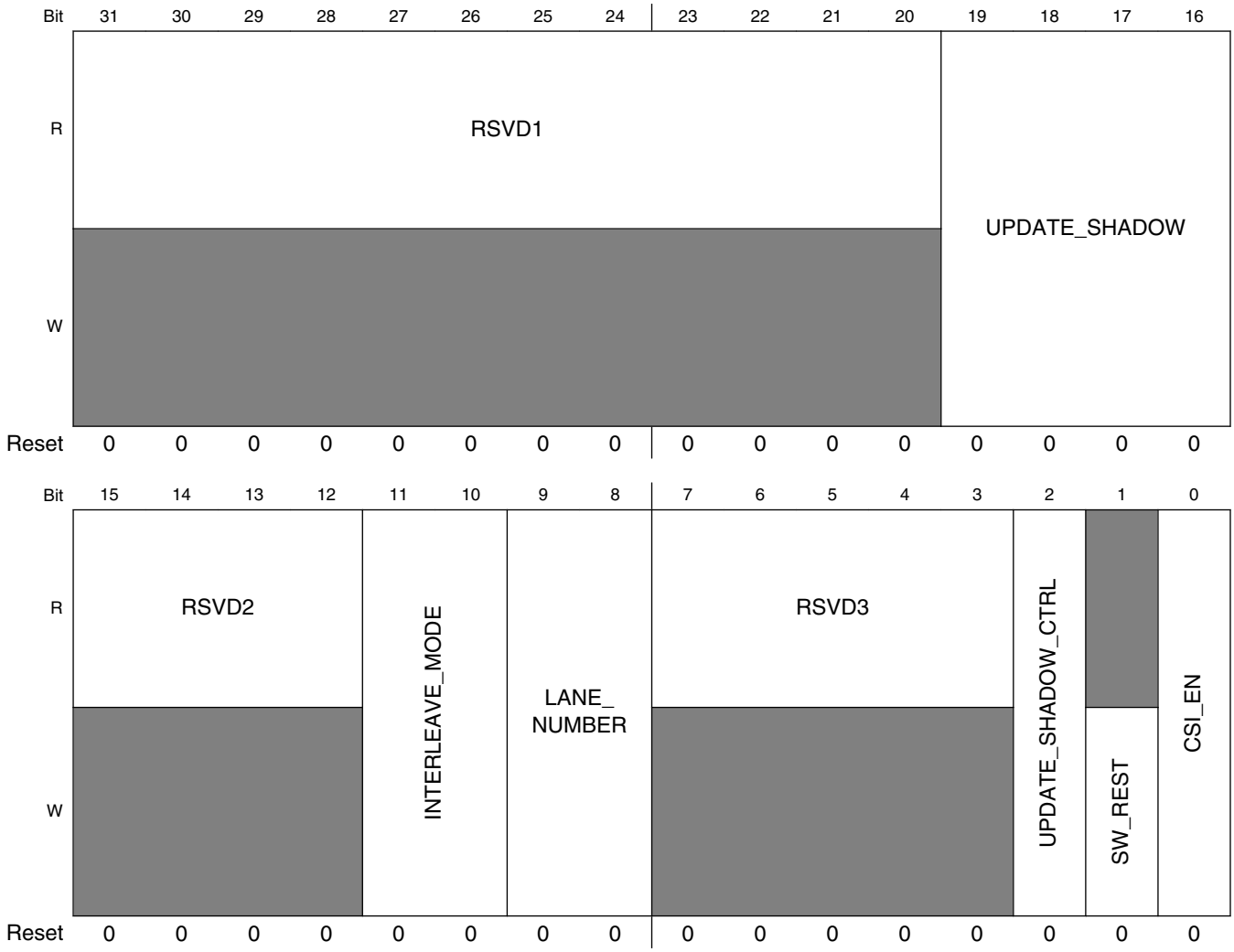
13.5.4 Register

MIPI_CSI2 memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3075_0004	CSIS Common Control (MIPI_CSI2_CSIS_CMN_CTRL)	32	R/W	0000_0000h	13.5.4.1/ 3772
3075_0008	CSIS Clock gate Control (MIPI_CSI2_CSIS_CLK_CTRL)	32	R/W	0000_00F0h	13.5.4.2/ 3774
3075_0010	CSIS Interrupt Mask (MIPI_CSI2_CSIS_INT_MSK)	32	R/W	0000_0000h	13.5.4.3/ 3775
3075_0014	CSIS Interrupt Source (MIPI_CSI2_CSIS_INT_SRC)	32	R/W	0000_0000h	13.5.4.4/ 3777
3075_0020	D-PHY Status (MIPI_CSI2_DPHY_STATUS)	32	R/W	0000_00F1h	13.5.4.5/ 3779
3075_0024	D-PHY Common Control (MIPI_CSI2_DPHY_CMN_CTRL)	32	R/W	0000_0000h	13.5.4.6/ 3781
3075_0030	D-PHY Master and Slave Control register low (MIPI_CSI2_DPHY_BCTRL_L)	32	R/W	0000_0000h	13.5.4.7/ 3782
3075_0034	D-PHY Master and Slave Control register high (MIPI_CSI2_DPHY_BCTRL_H)	32	R/W	0000_0000h	13.5.4.8/ 3782
3075_0038	D-PHY Slave Control register low (MIPI_CSI2_DPHY_SCTRL_L)	32	R/W	0000_0000h	13.5.4.9/ 3783
3075_003C	D-PHY Slave Control register high (MIPI_CSI2_DPHY_SCTRL_H)	32	R/W	0000_0000h	13.5.4.10/ 3783
3075_0040	ISP Configuration register of CH0 (MIPI_CSI2_ISP_CONFIG_CH0)	32	R/W	0800_08FCh	13.5.4.11/ 3784
3075_0044	ISP Image Resolution register of CH0 (MIPI_CSI2_ISP_RESOL_CH0)	32	R/W	8000_8000h	13.5.4.12/ 3786
3075_0048	ISP SYNC register of CH0 (MIPI_CSI2_ISP_SYNC_CH0)	32	R/W	0000_0000h	13.5.4.13/ 3787
3075_0080	Shadow Configuration register of CH0 (MIPI_CSI2_SDW_CONFIG_CH0)	32	R	0800_08FCh	13.5.4.14/ 3788
3075_0084	Shadow Resolution register of CH0 (MIPI_CSI2_SDW_RESOL_CH0)	32	R	0800_0800h	13.5.4.15/ 3790
3075_0088	Shadow SYNC register of CH0 (MIPI_CSI2_SDW_SYNC_CH0)	32	R/W	0000_0000h	13.5.4.16/ 3790
3075_00C0	Debug Control register (MIPI_CSI2_DBG_CTRL)	32	R/W	0000_0F00h	13.5.4.17/ 3791
3075_00C4	Debug Interrupt Mask (MIPI_CSI2_DBG_INTR_MSK)	32	R/W	0000_0000h	13.5.4.18/ 3792
3075_00C8	Debug Interrupt Mask (MIPI_CSI2_DBG_INTR_SRC)	32	R/W	0000_0000h	13.5.4.19/ 3794
3075_2000	Non Image Data (MIPI_CSI2_NON_IMG_DATA)	32	R/W	Undefined	13.5.4.20/ 3795

13.5.4.1 CSIS Common Control (MIPI_CSI2_CSIS_CMN_CTRL)

Address: 3075_0000h base + 4h offset = 3075_0004h



MIPI_CSI2_CSIS_CMN_CTRL field descriptions

Field	Description
31–20 RSVD1	Reserved Read as zero, do not modify.
19–16 UPDATE_ SHADOW	Strobe of updating shadow registers After configuration, user has to set this bit for updating shadow registers. This bit is cleared automatically after updating shadow registers. Bit 19, 18, 17 is reserved. Bit 16 set for CH0. 0 Default 1 Update the shadow registers

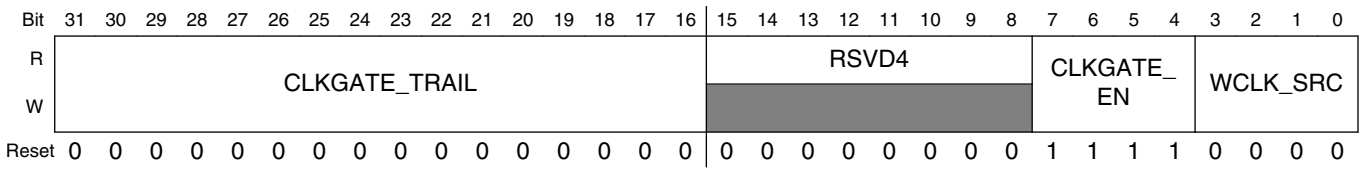
Table continues on the next page...

MIPI_CSI2_CSIS_CMN_CTRL field descriptions (continued)

Field	Description
15–12 RSVD2	Reserved Read as zero, do not modify.
11–10 INTERLEAVE_ MODE	Select Interleave mode, VC (Virtual channel) and DT (Data type) 11 VC and DT 10 VC only 01 DT only 00 CH0 only, no data interleave
9–8 LANE_NUMBER	Number of data lane 00 1 data lane 01 2 data lane 10 Reserved 11 Reserved
7–3 RSVD3	Reserved Read as zero, do not modify.
2 UPDATE_ SHADOW_CTRL	Update Shadow Control Enable this reg is recommended when interleave_mode is 2 or 3. 0 Disable (default) 1 Enable
1 SW_REST	Software Reset All writable registers in CSI2 go back to initial state. After this bit is active for three cycles, this bit will be deasserted automatically. **NOTE:** Almost MIPI CSI2 block uses “ByteClk” from D-phy. This “ByteClk” is not continuous clock. User has to assert software reset when Camera module is turned off. 0 Ready 1 Reset
0 CSI_EN	MIPI CSI2 system enable 0 Disable 1 Enable

13.5.4.2 CSIS Clock gate Control (MIPI_CSI2_CSIS_CLK_CTRL)

Address: 3075_0000h base + 8h offset = 3075_0008h

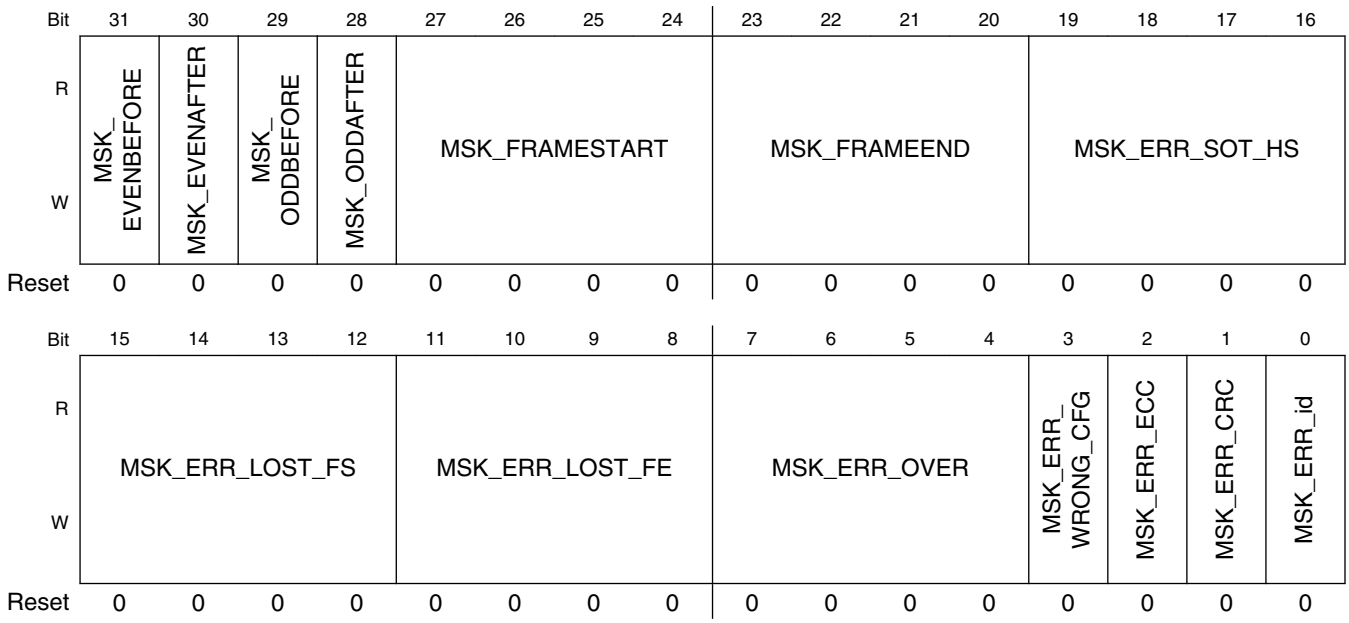


MIPI_CSI2_CSIS_CLK_CTRL field descriptions

Field	Description
31–16 CLKGATE_TRAIL	0 ~ 15 (1~16 Trailing clocks) [31:28], [27:24], [23:20] Reserved; [19:16] CH0. Trailing clocks are used for poping the F/F of ISP or CAM.
15–8 RSVD4	Reserved Read as zero, do not modify.
7–4 CLKGATE_EN	Bit 7, 6, 5 is reserved, bit 4 set for CH0. 0 Pixel clock is always alive. 1 Pixel clock is alive during the interval of frame.
WCLK_SRC	Pixel Clock Source This bit determines source of Pixel clock, which is clock of transferring image data to ISP or CAM I/F. Bit3, 2, 1 is reserved, bit0 set for CH0. 0 PCLK 1 EXTCLK (WRAP_CLK)

13.5.4.3 CSIS Interrupt Mask (MIPI_CSI2_CSIS_INT_MSK)

Address: 3075_0000h base + 10h offset = 3075_0010h



MIPI_CSI2_CSIS_INT_MSK field descriptions

Field	Description
31 MSK_ EVENBEFORE	Non Image data are received at Even frame and Before Image. 0 Disable (masked) 1 Enable (unmasked)
30 MSK_ EVENAFTER	Non Image data are received at Even frame and After Image. 0 Disable (masked) 1 Enable (unmasked)
29 MSK_ ODDBEFORE	Non Image data are received at Odd frame and Before Image. 0 Disable (masked) 1 Enable (unmasked)
28 MSK_ ODDAFTER	Non Image data are received at Odd frame and After Image. 0 Disable (masked) 1 Enable (unmasked)
27–24 MSK_ FRAMESTART	FS packet is received, [CH3,CH2,CH1,CH0] 0 Disable (masked) 1 Enable (unmasked)

Table continues on the next page...

MIPI_CSI2_CSIS_INT_MSK field descriptions (continued)

Field	Description
23–20 MSK_ FRAMEEND	FE packet is received, [CH3,CH2,CH1,CH0] 0 Disable (masked) 1 Enable (unmasked)
19–16 MSK_ERR_ SOT_HS	Start of transmission error [reserved, reserved, Lane1, Lane0] 0 Disable (masked) 1 Enable (unmasked)
15–12 MSK_ERR_ LOST_FS	Lost of Frame Start packet, [CH3,CH2,CH1,CH0] 0 Disable (masked) 1 Enable (unmasked)
11–8 MSK_ERR_ LOST_FE	Lost of Frame End packet, [CH3,CH2,CH1,CH0] 0 Disable (masked) 1 Enable (unmasked)
7–4 MSK_ERR_ OVER	Image FIFO overflow interrupt, [CH3,CH2,CH1,CH0] 0 Disable (masked) 1 Enable (unmasked)
3 MSK_ERR_ WRONG_CFG	Wrong configuration 0 Disable (masked) 1 Enable (unmasked)
2 MSK_ERR_ECC	ECC Error 0 Disable (masked) 1 Enable (unmasked)
1 MSK_ERR_CRC	CRC Error 0 Disable (masked) 1 Enable (unmasked)
0 MSK_ERR_id	Unknown ID Error 0 Disable (masked) 1 Enable (unmasked)

13.5.4.4 CSIS Interrupt Source (MIPI_CSI2_CSIS_INT_SRC)

Address: 3075_0000h base + 14h offset = 3075_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EVENBEFORE	EVENAFTER	ODDBEFORE	ODDAFTER	FRAMESTART				FRAMEEND				ERR_SOT_HS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ERR_LOST_FS				ERR_LOST_FE				ERR_OVER				ERR_WRONG_CFG	ERR_ECC	ERR_CRC	ERR_ID
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_CSI2_CSIS_INT_SRC field descriptions

Field	Description
31 EVENBEFORE	Non Image data are received at Even frame and Before Image.
30 EVENAFTER	Non Image data are received at Even frame and After Image.
29 ODDBEFORE	Non Image data are received at Odd frame and Before Image.
28 ODDAFTER	Non Image data are received at Odd frame and After Image.
27–24 FRAMESTART	FS packet is received, [CH3,CH2,CH1,CH0]
23–20 FRAMEEND	FE packet is received, [CH3,CH2,CH1,CH0]
19–16 ERR_SOT_HS	Start of transmission error, [reserved, reserved, Lane1 , Lane0]
15–12 ERR_LOST_FS	Indication of lost of Frame Start packet, [CH3,CH2,CH1,CH0]
11–8 ERR_LOST_FE	Indication of lost of Frame End packet, [CH3,CH2,CH1,CH0]
7–4 ERR_OVER	Overflow is caused in image FIFO, [CH3,CH2,CH1,CH0] Outer bandwidth has to be faster than imputer bandwidth. But image FIFO can be overflow because of user's fault. There are two ways for preventing overflow. 1. Tune output pixel clock faster than current

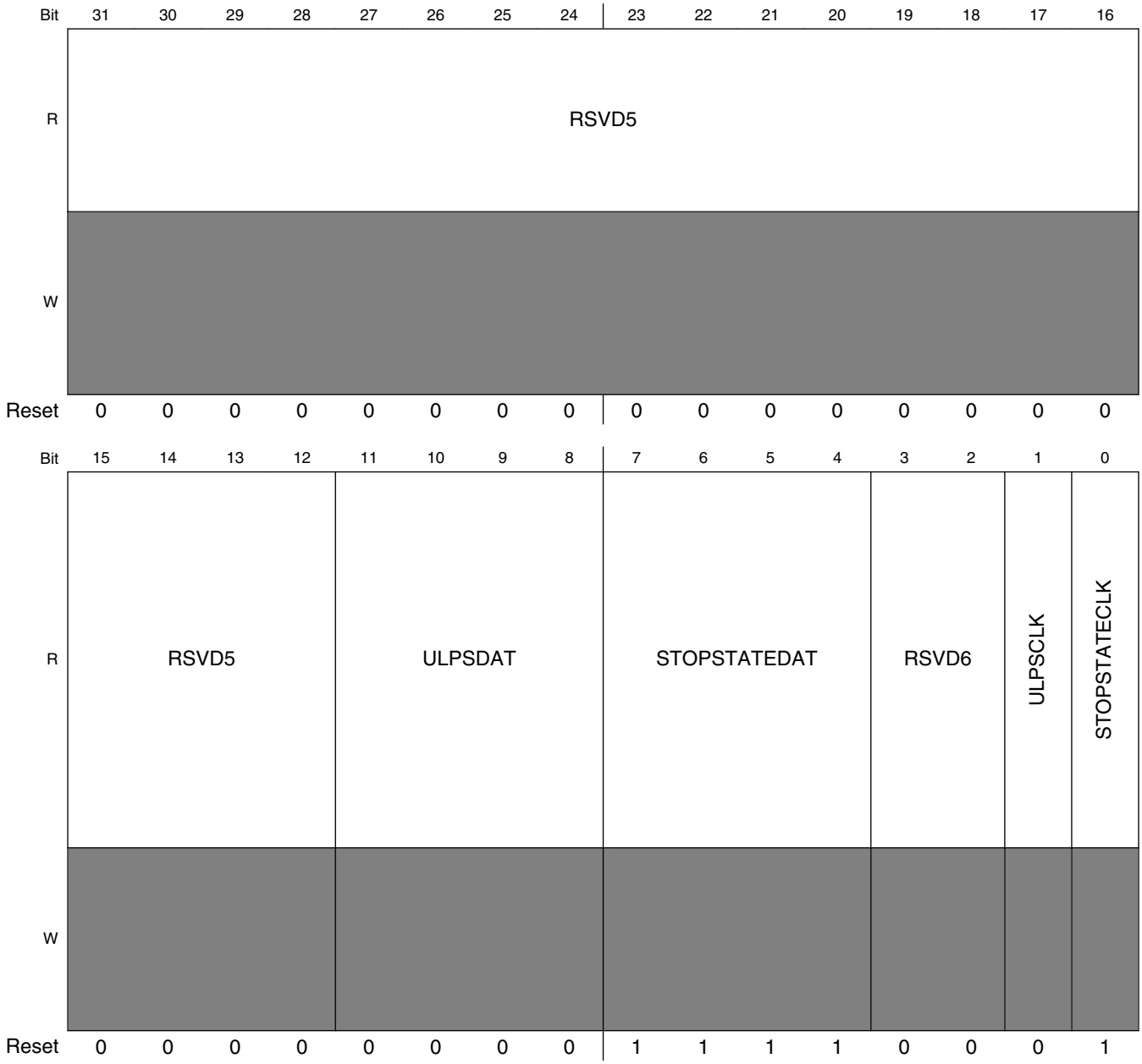
Table continues on the next page...

MIPI_CSI2_CSIS_INT_SRC field descriptions (continued)

Field	Description
	2. Tune input byte clock slowr than current First case: WCLK_SRC in CSIS_CTRL register can be set 1, and then assign faster clock Second case: user can set register in camera module through I^2C channel. When this interrupt is generated, 1. Turn camera off 2. Assert software reset, if you didn't assert software reset, MIPI CSIS could not receive any more data. 3. Tune the clock frequency and re-configure all related registers MIPI CSIS module is ready for operating.
3 ERR_WRONG_CFG	Wrong Configuration The received packet is not decoded at any channels of ISP (due to wrong data type or virtual channel ID)
2 ERR_ECC	ECC Error
1 ERR_CRC	CRC Error
0 ERR_ID	Unknown ID Error

13.5.4.5 D-PHY Status (MIPI_CSI2_DPHY_STATUS)

Address: 3075_0000h base + 20h offset = 3075_0020h



MIPI_CSI2_DPHY_STATUS field descriptions

Field	Description
31–12 RSVD5	Reserved Read as zero, do not modify.
11–8 ULPSDAT	Data lane [3:0] is in ULPS.

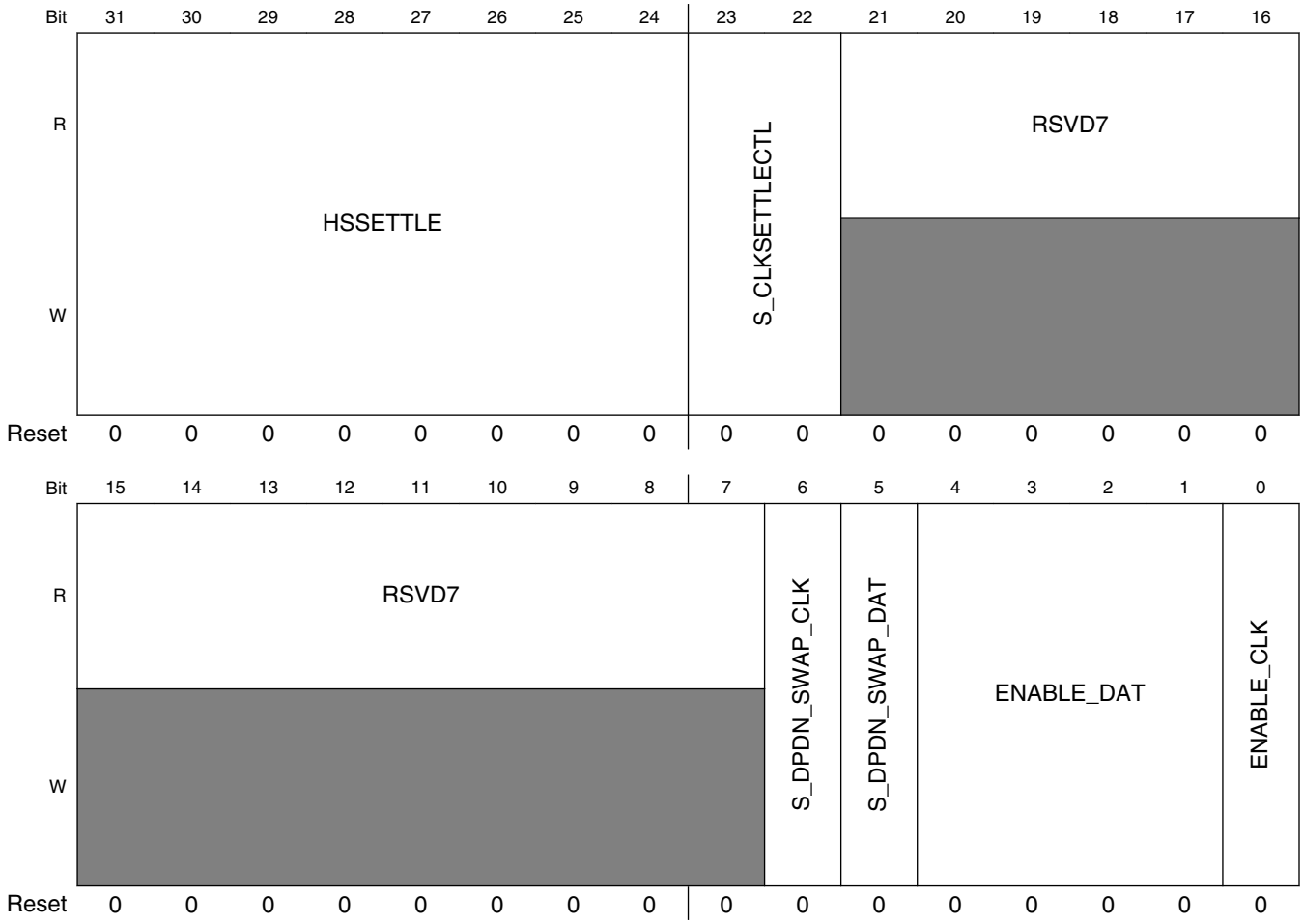
Table continues on the next page...

MIPI_CSI2_DPHY_STATUS field descriptions (continued)

Field	Description
	[11]: Reserved [10]: Reserved [9]: Data lane 1 [8]: Dsata lane 0 0 Not ULPS 1 ULPS —
7–4 STOPSTATEDAT	Data lane [3:0] is in Stop State. [7]: Reserved [6]: Reserved [5]: Data lane 1 [4]: Data lane 0 0 Not Stop state 1 Stop state
3–2 RSVD6	Reserved Read as zero, do not modify.
1 ULPSCLK	Clock lane is in ULPS. 0 Not ULPS 1 ULPS
0 STOPSTATECLK	Clock lane is in Stop state. 0 Not Stop state 1 Stop state

13.5.4.6 D-PHY Common Control (MIPI_CSI2_DPHY_CMN_CTRL)

Address: 3075_0000h base + 24h offset = 3075_0024h



MIPI_CSI2_DPHY_CMN_CTRL field descriptions

Field	Description
31–24 HSSETTLE	HS-RX settle time control register
23–22 S_ CLKSETTLECTL	D-PHY control register is for standard spec v0.9 of MIPI CSI2.
21–7 RSVD7	Reserved Read as zero, do not modify.
6 S_DPDN_ SWAP_CLK	Swapping Dp and Dn channel of clock lane. 0 Default 1 Swapped

Table continues on the next page...

MIPI_CSI2_DPHY_CMN_CTRL field descriptions (continued)

Field	Description
5 S_DPDN_ SWAP_DAT	Swapping Dp and Dn channel of data lanes. 0 Default 1 Swapped
4–1 ENABLE_DAT	D-PHY enable [3]: Reserved [2]: Reserved [1]: Data lane 1 [0]: Data lane 0 0 Disable 1 Enable
0 ENABLE_CLK	D-PHY clock lane enable 0 Disable 1 Enable

13.5.4.7 D-PHY Master and Slave Control register low (MIPI_CSI2_DPHY_BCTRL_L)

Address: 3075_0000h base + 30h offset = 3075_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	B_DPHYCTRL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_CSI2_DPHY_BCTRL_L field descriptions

Field	Description
B_DPHYCTRL	D-PHY Master and Slave control register Low part

13.5.4.8 D-PHY Master and Slave Control register high (MIPI_CSI2_DPHY_BCTRL_H)

Address: 3075_0000h base + 34h offset = 3075_0034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	B_DPHYCTRL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_CSI2_DPHY_BCTRL_H field descriptions

Field	Description
B_DPHYCTRL	D-PHY Master and Slave control register High part

13.5.4.9 D-PHY Slave Control register low (MIPI_CSI2_DPHY_SCTRL_L)

Address: 3075_0000h base + 38h offset = 3075_0038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	S_DPHYCTRL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_CSI2_DPHY_SCTRL_L field descriptions

Field	Description
S_DPHYCTRL	D-PHY Slave control register Low part

13.5.4.10 D-PHY Slave Control register high (MIPI_CSI2_DPHY_SCTRL_H)

Address: 3075_0000h base + 3Ch offset = 3075_003Ch

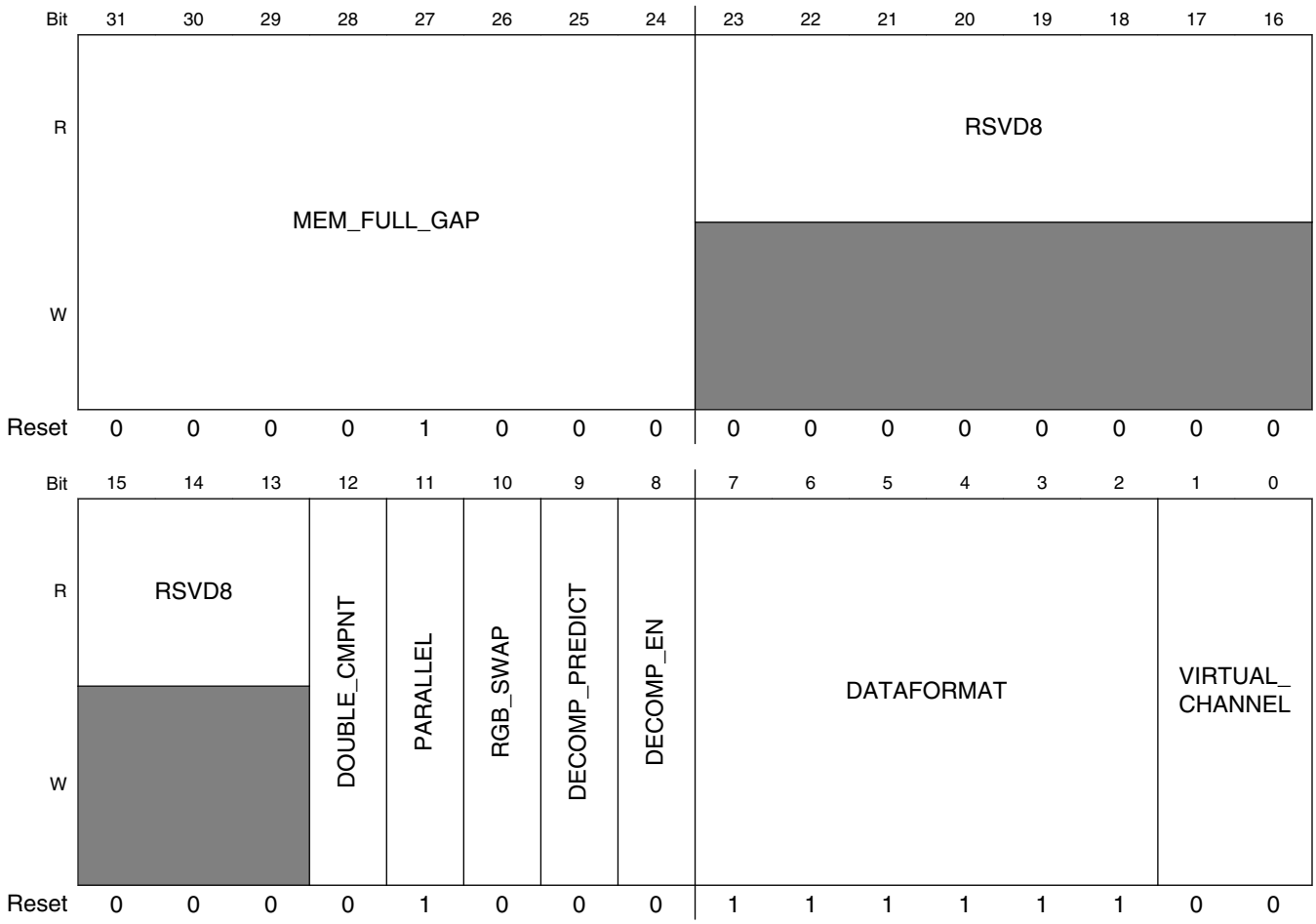
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	S_DPHYCTRL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_CSI2_DPHY_SCTRL_H field descriptions

Field	Description
S_DPHYCTRL	D-PHY Slave control register High part

13.5.4.11 ISP Configuration register of CH0 (MIPI_CSI2_ISP_CONFIG_CH0)

Address: 3075_0000h base + 40h offset = 3075_0040h



MIPI_CSI2_ISP_CONFIG_CH0 field descriptions

Field	Description
31–24 MEM_FULL_ GAP	When the room of Image memory is lower than this, STOPREQ_EN is 0, then the image data cannot be blocked (Do not set this register lower than 4).
23–13 RSVD8	Reserved Read as zero, do not modify.
12 DOUBLE_ CMPNT	Double component per clock cycle in YUV422 formats 0 Single component per clock cycle (half pixel per clock cycle) 1 Double component per clock cycle (a pixel per clock cycle)

Table continues on the next page...

MIPI_CSI2_ISP_CONFIG_CH0 field descriptions (continued)

Field	Description
11 PARALLEL	Output bus width of CH0 is 32 bits. When this bit is set, the outer bus width of CSIS V3.3 is 32. 0 Normal output 1 32-bit data alignment
10 RGB_SWAP	Swapping RGB sequence 0 MSB is R and LSB is B. 1 MSB is B and LSB is R (swapped).
9 DECOMP_ PREDICT	Decompress prediction mode of CH0 **NOTE:** Cannot use advanced prediction for RAW8 type 0 Simple prediction 1 Advanced prediction
8 DECOMP_EN	Decompress Enable **NOTE** Do not enable this when data format is not RAW6 / 7 / 8 / 10. 0 Disable (default) 1 Enable
7–2 DATAFORMAT	Image Data Format Generic Short Packet: 0x08 ~ 0x0F Embedded 8-bit non Image: 0x12 **NOTE:** • Not described types are ignored. • Only VC0 can use the Generic type packets. 0x18 YUV420 (8-bit) 0x19 YUV420 (10-bit) 0x1A YUV420 (8-bit legacy) 0x1C YUV420 (8-bit CSPS) 0x1D YUV420 (10-bit CSPS) 0x1E YUV422 (8-bit) 0x1F YUV422 (10-bit) 0x20 (NOT SUPPORT) RGB444 0x21 (NOT SUPPORT) RGB555 0x22 RGB565 0x23 RGB666 0x24 RGB888 0x28 RAW6

Table continues on the next page...

MIPI_CSI2_ISP_CONFIG_CH0 field descriptions (continued)

Field	Description
	0x29 RAW7 0x2A RAW8 0x2B RAW10 0x2C RAW12 0x2D RAW14 0x30 User defined 1 0x31 User defined 2 0x32 User defined 3 0x33 User defined 4 0x34 User defined 5 0x35 User defined 6 0x36 User defined 7 0x37 User defined 8
VIRTUAL_CHANNEL	Set Virtual channel for data interleave 00 VC = 0 01 VC = 1 10 VC = 2 11 VC = 3

13.5.4.12 ISP Image Resolution register of CH0 (MIPI_CSI2_ISP_RESOL_CH0)

When Data format is User defined packet type, this register is ignored.

Address: 3075_0000h base + 44h offset = 3075_0044h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VRESOL																HRESOL															
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_CSI2_ISP_RESOL_CH0 field descriptions

Field	Description
31–16 VRESOL	Vertical Image resolution Input boundary: 0x0001 ~ 0xFFFF
HRESOL	Horizontal Image resolution

Table continues on the next page...

MIPI_CSI2_ISP_RESOL_CH0 field descriptions (continued)

Field	Description
	Input boundary of each image format YUV420 (8-bit) 0x0001 - 0xFFFF YUV420 (10-bit) 4n (n is 1,2,3, …) YUV420 (8-bit legacy) 0x0001 - 0xFFFF YUV420 (8-bit CSPS) 0x0001 - 0xFFFF YUV420 (10-bit CSPS) 4n (n is 1,2,3, …) YUV422 (8-bit) 0x0001 - 0xFFFF YUV422 (10-bit) 4n (n is 1,2,3,…) RGB565 0x0001 - 0xFFFF RGB666 4n (n is 1,2,3,…) RGB888 0x0001 - 0xFFFF RAW8 0x0001 - 0xFFFF RAW10 4n (n is 1,2,3,…) RAW12 2n (n is 1,2,3,…) RAW14 4n (n is 1,2,3,…)

13.5.4.13 ISP SYNC register of CH0 (MIPI_CSI2_ISP_SYNC_CH0)

Address: 3075_0000h base + 48h offset = 3075_0048h

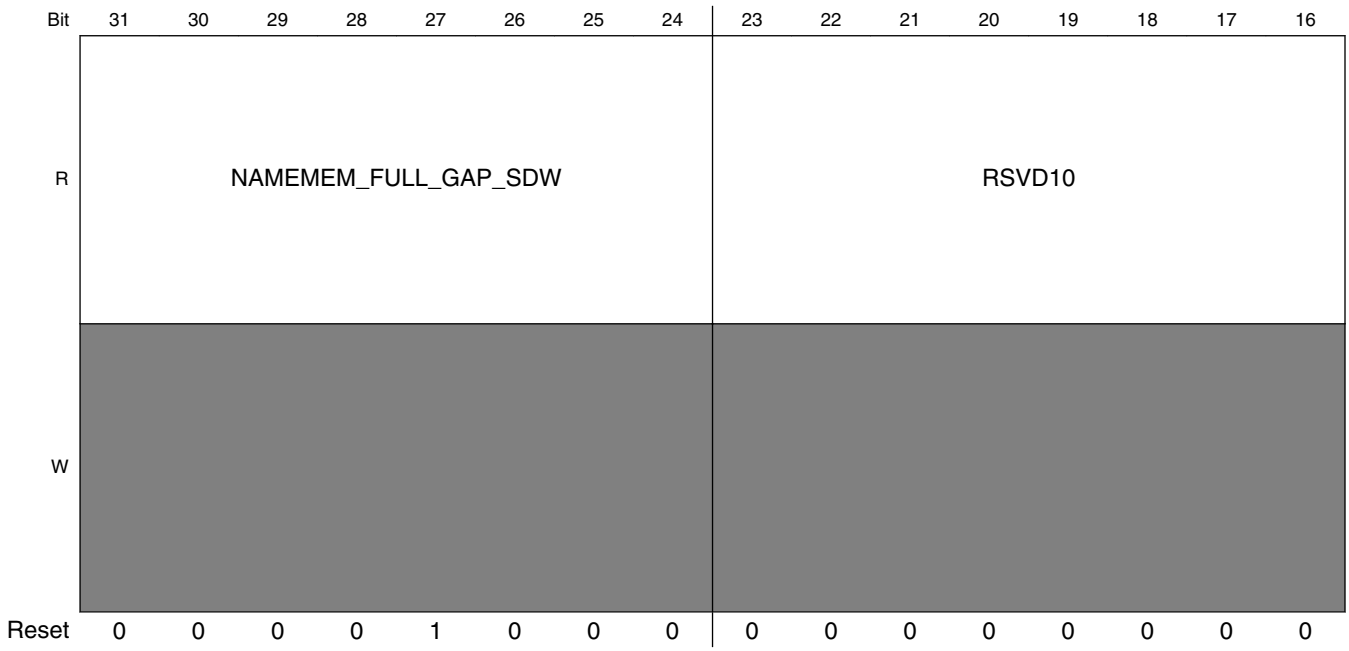
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD9								HSYNC_LINTV						VSYNC_SINTV						VSYNC_EINTV											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

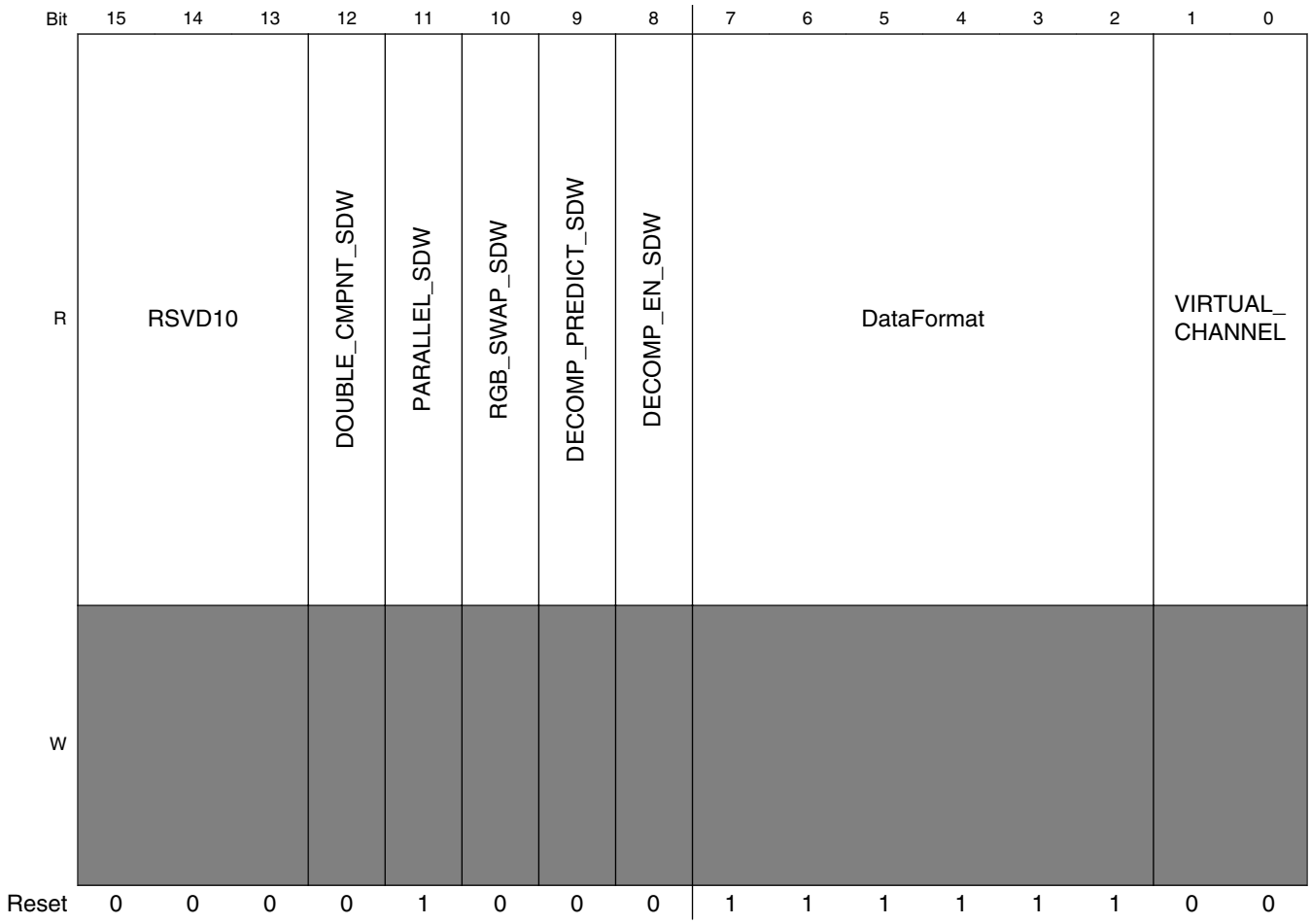
MIPI_CSI2_ISP_SYNC_CH0 field descriptions

Field	Description
31–24 RSVD9	Reserved Read as zero, do not modify.
23–18 HSYNC_LINTV	Interval between Hsync falling and Hsync rising (Line interval) 6'h00 ~ 6'h3F cycle of Pixel clock
17–12 VSYNC_SINTV	Interval between Vsync rising and first Hsync rising 6'h00 ~ 6'h3F cycle of Pixel clock
VSYNC_EINTV	Interval between last Hsync falling and Vsync falling 12'h000 ~ 12'hFFF cycle of Pixel clock

13.5.4.14 Shadow Configuration register of CH0 (MIPI_CSI2_SDW_CONFIG_CH0)

Address: 3075_0000h base + 80h offset = 3075_0080h





MIPI_CSI2_SDW_CONFIG_CH0 field descriptions

Field	Description
31–24 NAMEMEM_ FULL_GAP_ SDW	Current MEM_FULL_GAP
23–13 RSVD10	Reserved Read as zero, do not modify.
12 DOUBLE_ CMPNT_SDW	Current Double component
11 PARALLEL_ SDW	Current Parallel
10 RGB_SWAP_ SDW	Current RGB_SWAP
9 DECOMP_ PREDICT_SDW	Current Decompress prediction mode

Table continues on the next page...

MIPI_CSI2_SDW_CONFIG_CH0 field descriptions (continued)

Field	Description
8 DECOMP_EN_SDW	Current Decompress enable
7–2 DataFormat	Current Image Data Format
VIRTUAL_CHANNEL	Current Virtual channel

13.5.4.15 Shadow Resolution register of CH0 (MIPI_CSI2_SDW_RESOL_CH0)

Address: 3075_0000h base + 84h offset = 3075_0084h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VRESOL_SDW																HRESOL_SDW															
W																																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

MIPI_CSI2_SDW_RESOL_CH0 field descriptions

Field	Description
31–16 VRESOL_SDW	Current Vertical Image resolution
HRESOL_SDW	Current Horizontal Image resolution

13.5.4.16 Shadow SYNC register of CH0 (MIPI_CSI2_SDW_SYNC_CH0)

Address: 3075_0000h base + 88h offset = 3075_0088h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD11								HSYNC_LINTV_SDW						VSYNC_SINTV_SDW		VSYNC_EINTV_SDW															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIPI_CSI2_SDW_SYNC_CH0 field descriptions

Field	Description
31–24 RSVD11	Reserved Read as zero, do not modify.

Table continues on the next page...

MIPI_CSI2_SDW_SYNC_CH0 field descriptions (continued)

Field	Description
23–18 HSYNC_LINTV_SDW	Current interval between Hsync falling and Hsync rising (Line interval)
17–12 VSYNC_SINTV_SDW	Current interval between Vsync rising and first Hsync rising
VSYNC_EINTV_SDW	Current interval between last Hsync falling and Vsync falling

13.5.4.17 Debug Control register (MIPI_CSI2_DBG_CTRL)

In the case of FS or FE lost during continous frame transfers, CSIS transfer the data of second frame as it belongs to the first frame when DBG_BLK_EXC_FRAME is low. DBG_CH_OUTPUT forces the result of demux for the incoming packet.

Address: 3075_0000h base + C0h offset = 3075_00C0h

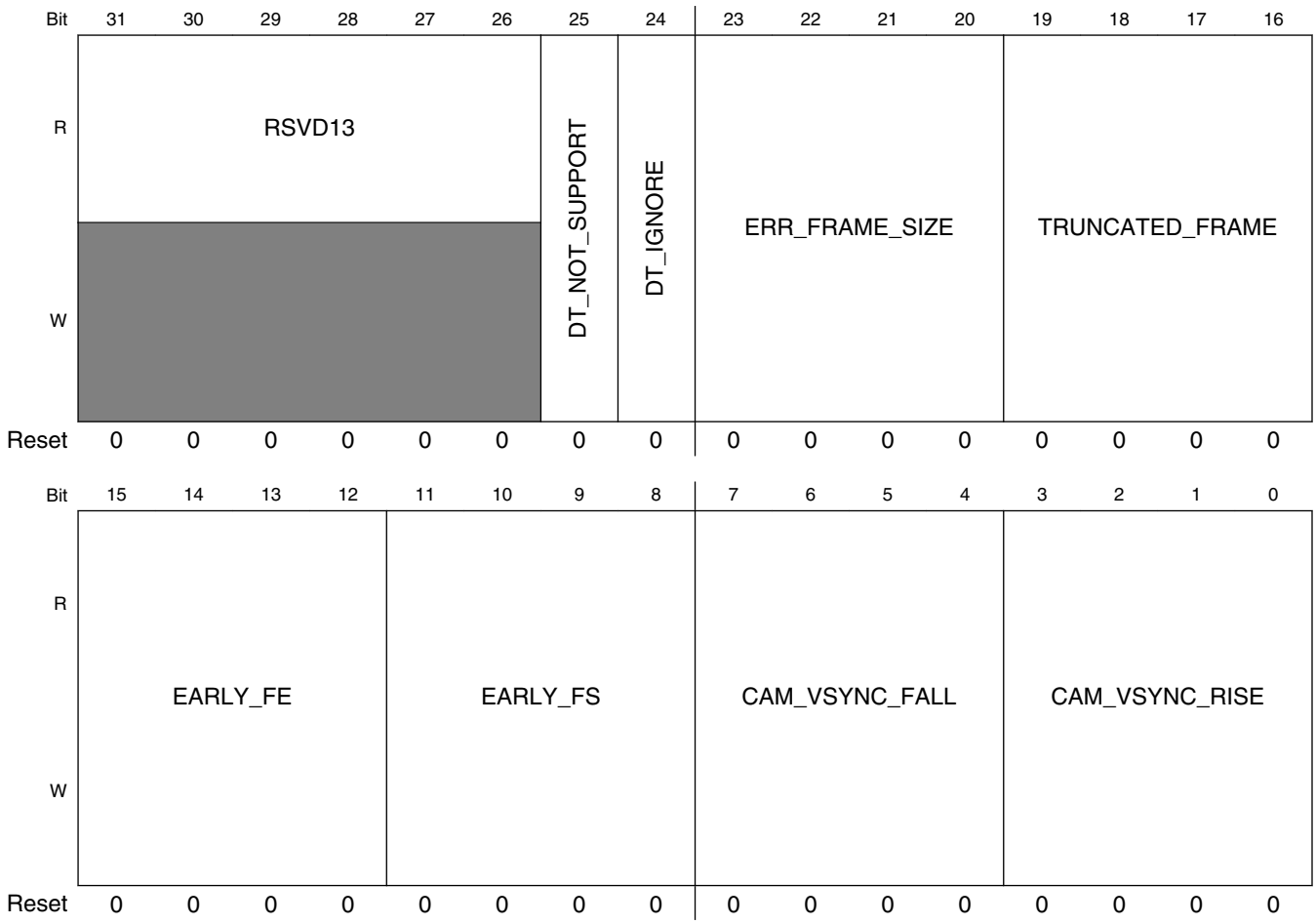
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD12																DBG_FORCE_UPDATE				DBG_DONT_STOP_LAST_LINE				DBG_BLK_EXC_FRAME				DBG_CH_OUTPUT			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0

MIPI_CSI2_DBG_CTRL field descriptions

Field	Description
31–16 RSVD12	Reserved Read as zero, do not modify.
15–12 DBG_FORCE_UPDATE	[15] CH3 [14] CH2 [13] CH1 [12] CH0 0 Update shadow reg normally 1 Force to update shadow reg without waiting end of frame
11–8 DBG_DONT_STOP_LAST_LINE	[11] CH3 [10] CH2 [9] CH1 [8] CH0 0 Can receive STOP_REQ from ISP any time 1 Ignore STOP_REQ from ISP at the LAST_LINE
7–4 DBG_BLK_EXC_FRAME	[7] CH3 [6] CH2 [5] CH1 [4] CH0 0 Do not block the exceeded frame 1 Block the exceeded frame
DBG_CH_OUTPUT	[3] CH3 [2] CH2 [1] CH1 [0] CH0 1 Force the channel output to 1

13.5.4.18 Debug Interrupt Mask (MIPI_CSI2_DBG_INTR_MSK)

Address: 3075_0000h base + C4h offset = 3075_00C4h



MIPI_CSI2_DBG_INTR_MSK field descriptions

Field	Description
31–26 RSVD13	Reserved Read as zero, do not modify.
25 DT_NOT_ SUPPORT	DT_NOT_SUPPORT 0 Disable (masked) 1 Enable (unmasked)
24 DT_IGNORE	DT_IGNORE 0 Disable (masked) 1 Enable (unmasked)
23–20 ERR_FRAME_ SIZE	Error Frame Size [23] CH3 [22] CH2 [21] CH1 [20] CH0

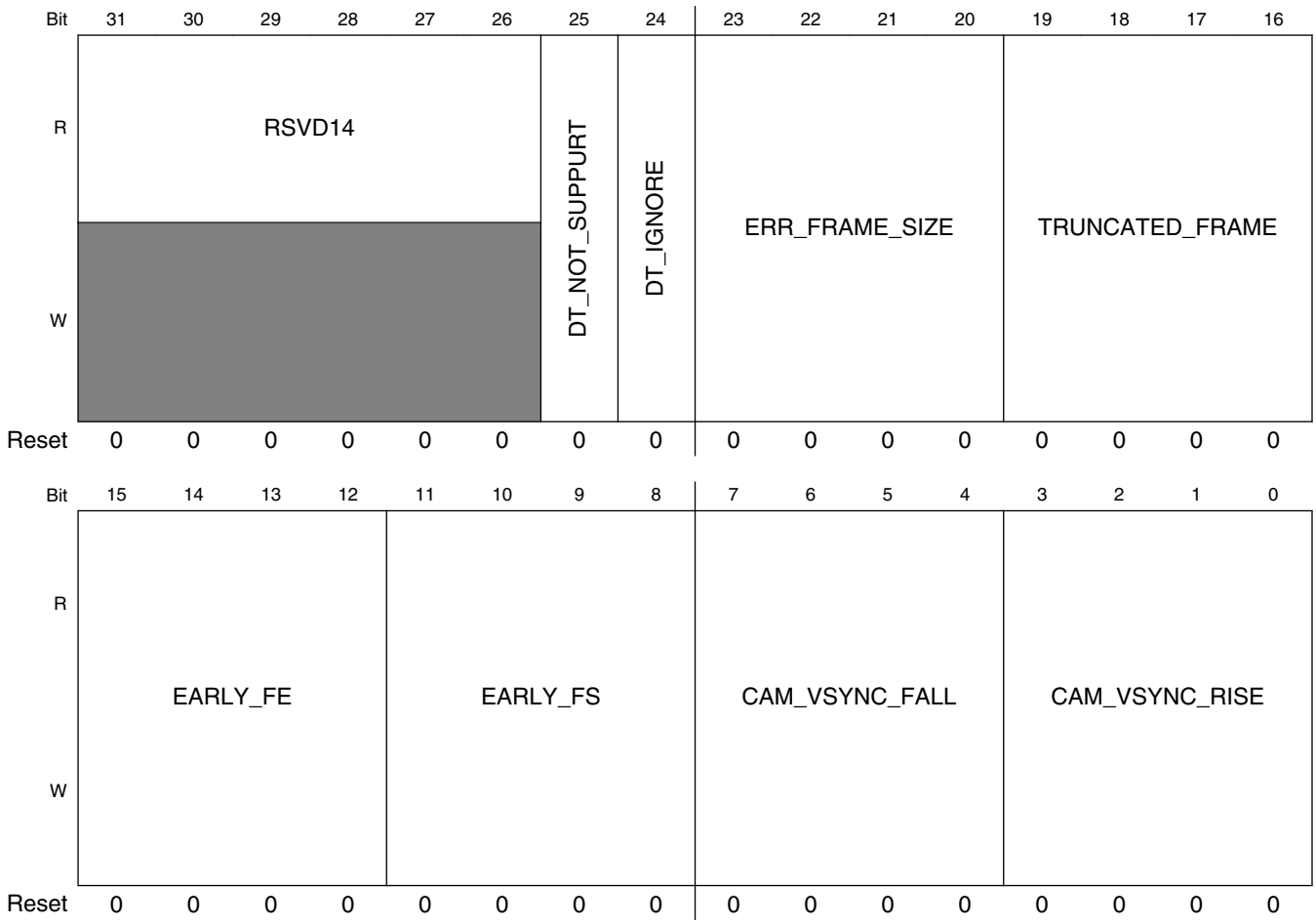
Table continues on the next page...

MIPI_CSI2_DBG_INTR_MSK field descriptions (continued)

Field	Description
	0 Disable (masked) 1 Enable (unmasked)
19–16 TRUNCATED_ FRAME	Truncated Frame [19] CH3 [18] CH2 [17] CH1 [16] CH0 0 Disable (masked) 1 Enable (unmasked)
15–12 EARLY_FE	Early FE [15] CH3 [14] CH2 [13] CH1 [12] CH0 0 Disable (masked) 1 Enable (unmasked)
11–8 EARLY_FS	Early FS [11] CH3 [10] CH2 [9] CH1 [8] CH0 0 Disable (masked) 1 Enable (unmasked)
7–4 CAM_VSYNC_ FALL	[7] CH3 [6] CH2 [5] CH1 [4] CH0 0 Disable (masked) 1 Enable (unmasked)
CAM_VSYNC_ RISE	[3] CH3 [2] CH2 [1] CH1 [0] CH0 0 Disable (masked) 1 Enable (unmasked)

13.5.4.19 Debug Interrupt Mask (MIPI_CSI2_DBG_INTR_SRC)

Address: 3075_0000h base + C8h offset = 3075_00C8h



MIPI_CSI2_DBG_INTR_SRC field descriptions

Field	Description
31–26 RSVD14	Reserved Read as zero, do not modify.
25 DT_NOT_ SUPPURT	The data type of the received packet is not supported (RGB444 or RGB555)
24 DT_IGNORE	The data type of the received packet is ignored (NULL or BLANKING)
23–20 ERR_FRAME_ SIZE	The received frame is not matched with the configured. [23] CH3 [22] CH2 [21] CH1 [20] CH0
19–16 TRUNCATED_ FRAME	Truncated frame is received. [19] CH3 [18] CH2 [17] CH1 [16] CH0

Table continues on the next page...

MIPI_CSI2_DBG_INTR_SRC field descriptions (continued)

Field	Description
15–12 EARLY_FE	Frame End packet is received during transfer of image. [15] CH3 [14] CH2 [13] CH1 [12] CH0
11–8 EARLY_FS	Frame Start packet is received during transfer of image. [11] CH3 [10] CH2 [9] CH1 [8] CH0
7–4 CAM_VSYNC_FALL	The falling of vsync in the CAM I/F [7] CH3 [6] CH2 [5] CH1 [4] CH0
CAM_VSYNC_RISE	The rising of vsync in the CAM I/F [3] CH3 [2] CH2 [1] CH1 [0] CH0

13.5.4.20 Non Image Data (MIPI_CSI2_NON_IMG_DATA)

For the double buffering, Non-image data region is divided as following address.

- 0x2000 - 0x2FFC: Odd frame / 0x3000 - 0x3FFC: Even frame (8 KB memory)
- 0x2000 - 0x27FC: Odd frame / 0x2800 - 0x2FFC: Even frame (4 KB memory)
- 0x2000 - 0x23FC: Odd frame / 0x2400 – 0x27FC: Even frame (2 KB memory)

Address: 3075_0000h base + 2000h offset = 3075_2000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NONIMGDATA																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

MIPI_CSI2_NON_IMG_DATA field descriptions

Field	Description
NONIMGDATA	Non Image data memory

13.6 Pixel Pipeline (PXP)

13.6.1 Overview

This document describes the micro-architecture for the Pixel Processing Pipeline used to process graphics buffers or composite video and graphics data before sending to an LCD display or TV encoder.

It is used to minimize the memory footprint required for the display pipeline and provide an area and performance optimized to both SDRAM-less and SRAM-based systems.

The PXP integrates of several independent processing stages into a cohesive strategy to create flexible pixel pipeline.

The PXP combines scaling, color space conversion (or CSC), alpha-blending, secondary color space conversion (or CSC2), pixel conversion lookup memory table (or LUT) ,rotation , dithering and waveform processing into a single processing engine, as shown in Figure 13-71. By integrating multiple blocks, a few intermediate buffer operations to external memory are removed, reducing external memory bandwidth, power, and software control complexity.

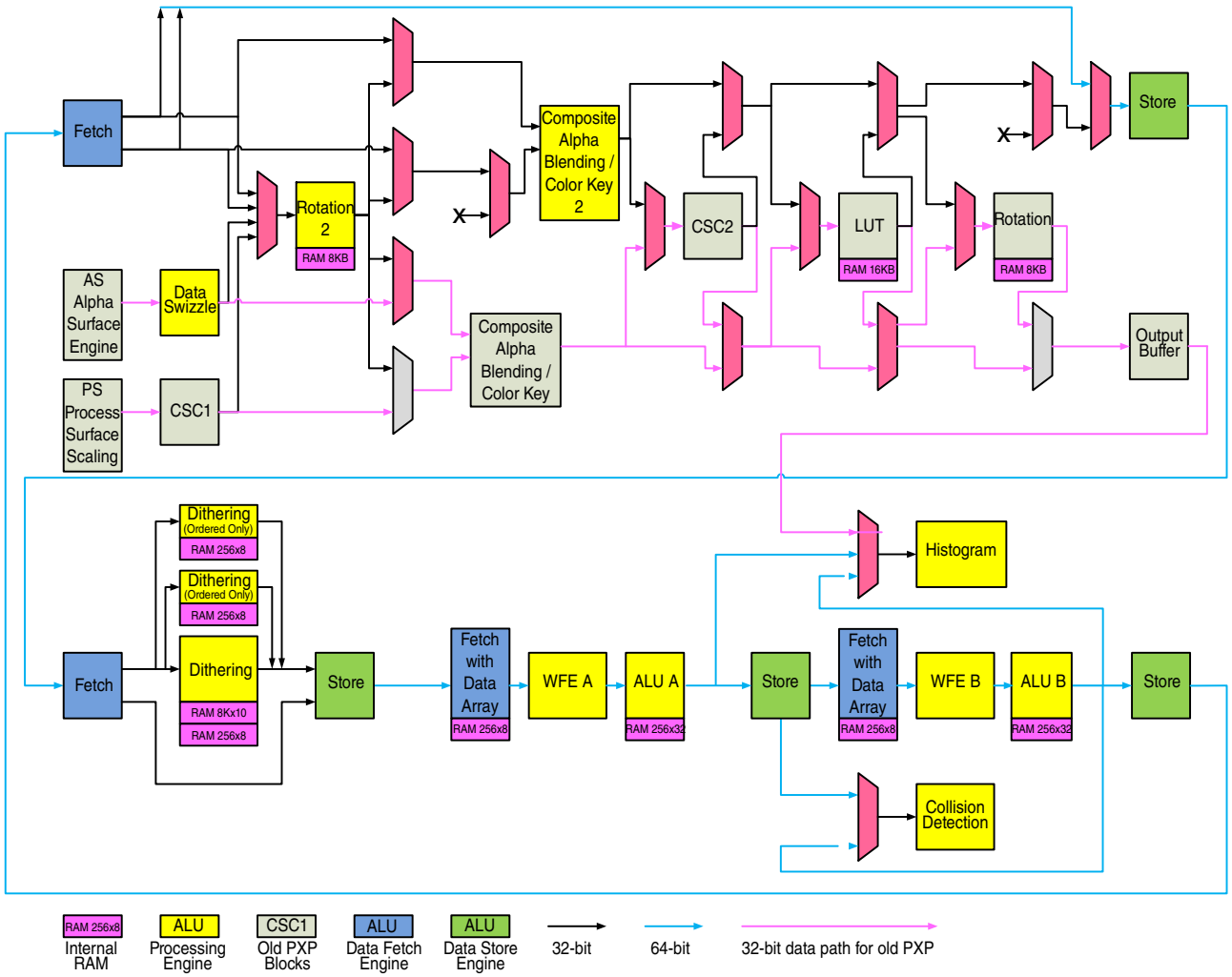


Figure 13-71. PXP Architecture

13.6.2 Clocks

The following table describes the clock sources for PXP. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 13-39. PXP Clocks

Clock name	Clock Root	Description
clk	pxp_axi_clk_root	PXP clock

13.6.3 Top-level architecture

The PXP consist of several pipelined blocks that perform the video source frame scaling, color space conversion, alpha-blending/color key algorithm, secondary CSC, pixel correction , input and/or output rotation, dithering and waveform processing. It is also capable of fetching data from and storing data to memory.

The legacy blocks operate within the requirements of the legacy PXP architecture, and perform operations on either 8x8 or 16x16 pixel blocks in the representative source buffers. The legacy pipeline operate within the context of two iteration counters that iterate through the appropriate grid of input blocks to produce the rotated output grid blocks in scan-line order. The dither blocks and the waveform processing engines (WFE) will operate in a scan line format based on the active size. The input fetch and store engines can work on either on a block by block basis or in the scan line format.

Figure 13-71 shows the high-level architecture of the scaling, color space conversion, blending, pixel correction , rotation engines, dithering, waveform processing, histogram along with four sets of fetch and store engines. The Alpha Surface Engine fetches one RGB graphics plane alpha surface (AS). The scaling engine fetches a single processed surface (PS), which can be blended with the AS surface. The scaling engine also supports an alpha channel for the PS image. Although the legacy PXP processes NxN pixel macro blocks, each of the AS or PS surfaces can have any pixel alignment within the output buffer. There are no restrictions and any pixel coordinates within the output buffer are valid. The upper left origin of the output buffer is defined as pixel 0,0. The upper left and lower right coordinates for each of the AS and PS are inclusive within the output buffer.

Figure 13-72 represents a sample output buffer configuration with both an AS and PS included. The alignment of each AS and PS within the output buffer can be at any arbitrary pixel locations. For example, the PS has an upper left coordinate (ULC) of 2,2 and a lower right coordinate (LRC) at pixel 13,13. The maximum value for the ULC and LRC for each of the AS and PS is bounded by the LRC of the output buffer, 15,15 for this example.

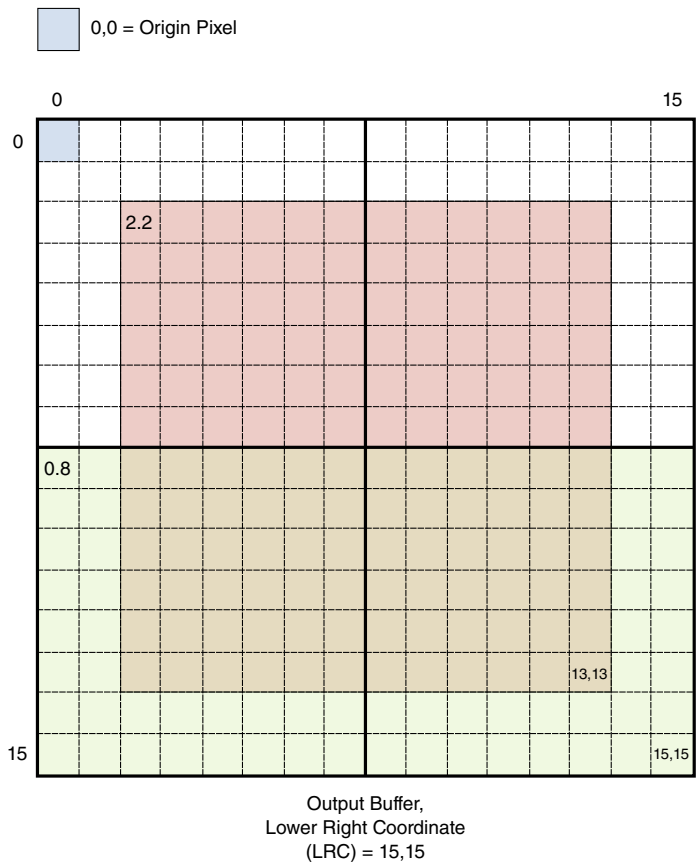


Figure 13-72. Sample output buffer configuration

The AS engine supports RGB pixel formats, and the PS engine supports ARGB, RGB, YUV, and YCbCr pixel formats. The CSC1 can be used to convert to RGB pixel formats so that the PS surface can be blended with the AS surfaces in the compositing engine in the RGB color space. There are two rotate engines in the PXP pipeline. Rotation can occur after image composition, at the output of the PS engine, or both. In the first scenario, all the data produced by the AS and PS engines is rotated. When the rotation module is programmed to rotate only PS images, the AS is not rotated, and AS pixels are combined with rotated PS surfaces. The Rotate engine at the input can also be used with the input fetch engine. For this, the fetch engine needs to be programmed in the blok mode to fetch a block of size 8x8. The CSC2 unit can convert to any RGB, YUV, or YCbCr color spaces for final output. Pixels can be corrected using a programmable LUT resource to achieve any desired pixels effects. For detailed information, please refer to Output Modes to RGBW4444CFA. The dither engine supports 3 dithering modes in addition to the standard quantization and pass-through modes. The dithering algorithms supported are: Ordered, Floyd-Steinberg and Atkinson. The WFE is a programmable engine that can be programed as required to process each pixel and pixel meta data. The Histogram sub-block collects statistics about the pixel data passing through it that is

useful to the EPDC in waveform selection and displaying the frame. The Histogram also contains mask and comparison functionality that can be used for collision detection and reporting of the collided area within an update area.

The PXP also supports two parallel image processing paths. The legacy flow can operate in parallel with the input fetch engine or the dithering or waveform processing engines. Please refer to HW_PXP_CTRL and HW_PXP_CTRL2 for details on how to enable each of the individual data flows. In addition to this, the CSC, Rotation 1 and LUT engines can be used as either part of the legacy flow or as a part of the second parallel data flow. These blocks can work in the scan line format or in block format. There are two separate blending engines for each flow.

13.6.3.1 Processing Details

The legacy PXP architecture has been driven primarily by the requirement that the output buffer must be processed and rotated without intermediate frame buffer stored in external memory.

This reduces the use of external memory bandwidth requirements thus reducing overall system power consumed. The new architecture has four sets of fetch and store engines to enable parallel image processing and the ability to use only parts of the block like the dithering, or WFE or parts of the legacy flow.

Since the output of the rotation block must be NxN pixel blocks in scan order, the entire legacy pipeline will operate on NxN pixel blocks. In essence, the pipeline will be able to operate on blocks in a random access fashion, but the entire pipeline will operate within the context of two iteration counters that will iterate through the horizontal and vertical input blocks to generate the required output block.

Legacy Processing Pipeline

The control block will coordinate the processing of the pixel blocks within the source and destination image buffers. It begins by issuing a command to each stage of the pipeline requesting that operations be done for the block at offset x, y. When the block accepts the command, it asserts its acknowledge signal for a single cycle to indicate the acceptance and allow the control unit to move to the next block.

When the PS and AS fetch engines have received a command, they will fetch the required data and place it into their fetch buffers. If compositing the RGB AS surface with the PS surface, then the output of the PS engine needs to be converted to the RGB color space using CSC1, since all compositing occurs in the RGB color space. For YUV output pixel formats, the CSC1 unit can be enabled to convert pixels into the RGB space for subsequent compositing with AS pixels. Then, the CSC2 module can convert the

resulting pixels back into the YUV output color space. If the final output color space is YUV and there is no compositing required (AS not present, for example), then both the CSC units can be bypassed and the pixel data path will pass the YUV pixels to the rotation engine. For YUV output formats, scaling operations, LUT, and rotation operations are still valid, but blending RGB AS surfaces with YUV PS surfaces is NOT supported. The two CSC units in the overall pixel data path must be used to achieve the desired source frame compositing and output pixel formatting.

The alpha blender/color key module will process a pixel any time that both inputs present valid data.

A handshake will be created between each stage and a pipeline controller to handle the advance of the pipeline and generation of the iteration counters. The pipeline controller will also maintain the interlocks with the LCD interface for the case where the LCD display and pixel processing pipeline use the SRAM to maintain the double buffer block intermediate buffer.

The New Processing Pipelines

The input fetch and store engines can work in either block mode or in scan line format. As they can work with the rotate1 engine, they were designed to support blocks of size 8x8. The dither and WFE fetch and store eingines work in scan line format. The WFE fetch engines have internal memory of 32 words where each word is 64 bits. The store engines can co-ordinate data flow through the pipeline to the next fetch engine by either using the handshake mode or the bypass mode. In the handshake mode, the store engine of the previous stage writes to the memory and signals the next stage fetch engine that the programmed number of lines are ready to be fetched. In the bypass mode, the store engine sends data to the next stage fetch engine through standard handshake protocol followed by the legacy modules.

The input fetch engine has another alpha blender/color key module connected to it. Along with that, the muxes can be programed to use the rotate1 engine, the LUT and the CSC2 engine.

13.6.3.2 Scaling Operation

The scaling engine operates on YUV (or YCbCr) 422 or 420 and any RGB formatted pixels. Each color plane is sourced from color planes indicated by different base address registers.

The scaling source data can be stored as 3 individual planes for each Y, U, and V data, stored as two planes as a single Y and interleaved UV plane, or stored as a single plane with YUV/RGB interleaved on a per byte basis.

The scaled output image is presented to the CSC module as YUV444 or RGB888 pixels with a single byte for each color channel. The scaler can reduce an input image by a maximum factor of 16. In this case, the output image will be 1/16 the dimension of the input image in each of the X and Y axis. There are no limits, essentially, on increasing the source image size. The theoretical maximum increase is 4096 since a 12 bit fractional step function is used when scaling an input image. Scaling in either axis, X or Y is independent, so a source image can appear stretched in either direction.

All source images pass through the scale engine. The PXP alpha blend module and AS pixel streams are in the RGB888 format, so PS pixel buffers must be converted to the RGB888 format for alpha blending. The scaling engine works with the CSC1 module to translate YUV/YCbCr pixel formats to RGB888 for output frame buffer compositing using the alpha blender. The CSC2 module can by bypassed or enabled to convert pixels to any output color space. In the case of processing RGB pixels in the PS engine, the CSC1 unit can be bypassed so compositing can occur in the alpha engine.

The scaling operation is divided into two scaling steps. The first step is a decimation scaler, and the second step is a bilinear filter. The decimation filter provide a maximum down scaling factor of 8, and the subsequent bilinear filter provides a maximum scaling factor of 2. Combined, the maximum scaling factor can be up to 16. The decimation and bilinear scaling engines are independently programmable. There is also an initial offset that is programmable to allow more source data to be considered in the bilinear scaling engine.

13.6.3.3 Decimation Image Scaling

The first of two scaling engines is the decimation filter.

The intent of the decimation filter is to use as much source data as is possible to create the output image frame buffer. The decimation filter simply discards certain pixels from the source PS image depending on the reduction selected.

For RGB pixel formats, each color channel is treated equally since there is the same amount of pixel data within each color plane. For YUV422/420 formats, the chroma samples are already subsampled by 2. In these decimation scenarios, the chroma decimation factor is adjusted to account for the pre-decimation of the chroma samples. For example, since YUV422 is already sub-sampled by 2 horizontally, an X decimation factor of 2 does not apply to the YUV422 pixels in the X direction. All the chroma samples are passed on to the bilinear filter in this case. As another example, an X decimation factor of 4 will decimate the chroma samples by 2, since this factor combined with the pre-decimation factor of 2 in the pixel source buffers totals an overall decimation factor of 4.

The following example will show which pixels (in green) in a source RGB buffer that are passed to the bilinear filter for an X decimation factor of 2 and a Y decimation factor of 4. All pixels coincident with dashed lines are discarded.

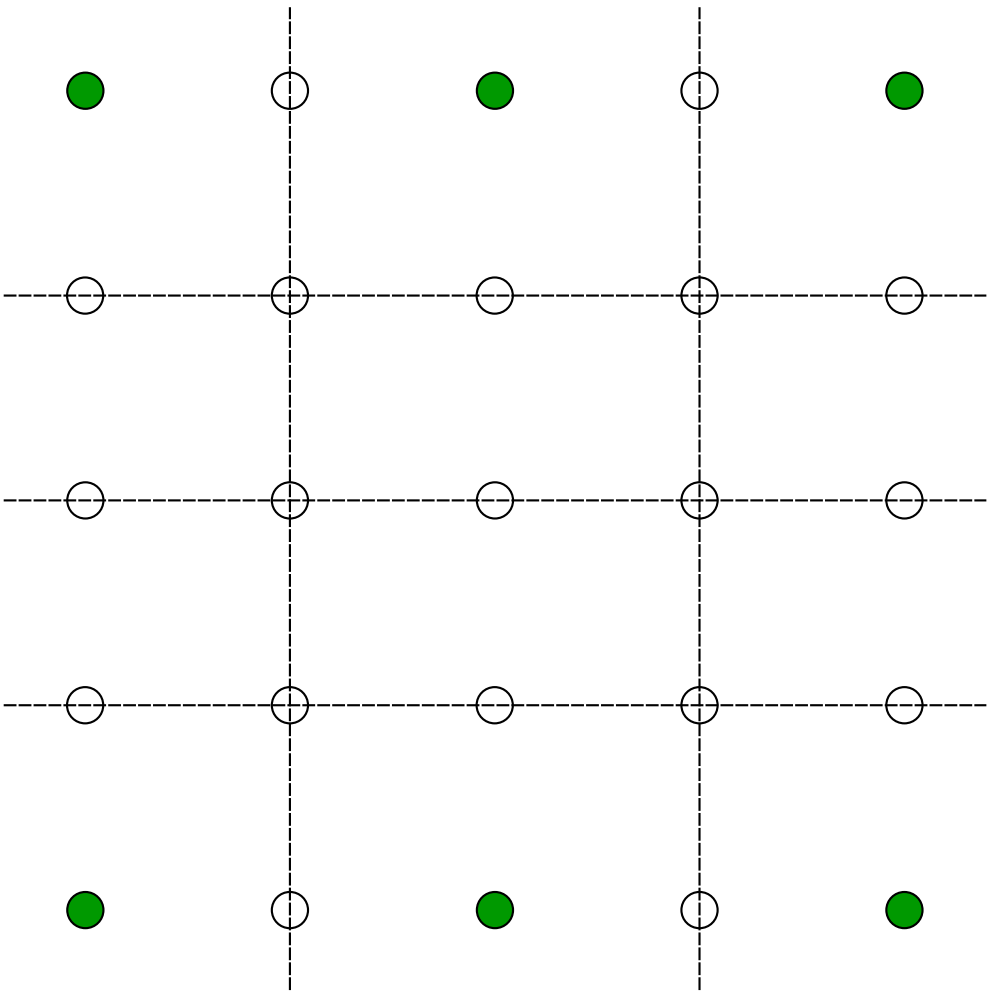


Figure 13-73. RGB decimation X /2, Y /4

Using the same decimation factor as the above scenario for RGB pixels, but using YUV420 source buffers, it can be shown that the decimation factor for the Y and UV components of data are decimated differently. This is due to the pre-decimation of the chroma samples in the source frame buffers. Figure 4: YUV420 decimation X /2, Y /4 indicates that the U/V samples in the X direction are not decimated, but the Y samples in the X direction are decimated by the factor of 2.

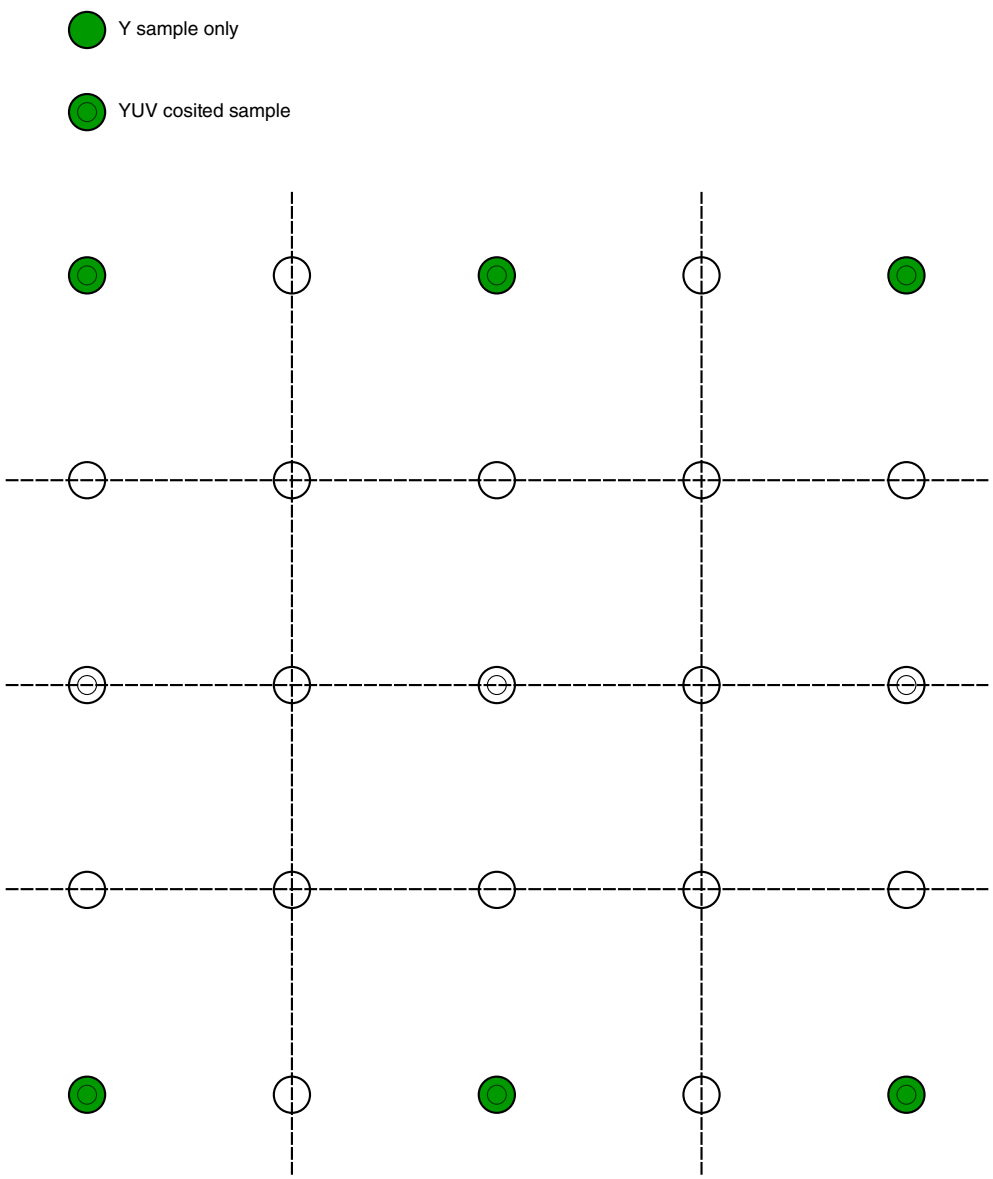


Figure 13-74. YUV420 decimation X /2, Y /4

13.6.3.4 Bilinear Image Scaling Filter

The PXP implements a bilinear scaling filter to resize an input image to a different resolution for display output.

The bilinear filter is a weighted average of the four nearest pixels that can be sourced to approximate the pixel in the output frame buffer.

When scaling YUV data, the UV values are offset by 0x80 (top bit inverted) to shift the signed UV bits into an unsigned equivalent with a range of 0 to 255. YCbCr data does not have to be shifted since it is defined as an unsigned byte. The REG_CSC1_COEF0[YCBCR_MODE] bit controls whether this operation is applied to the input UV bytes.

After scaling, the offset is removed so that the range for UV data is signed from -128 to 127.

The reason for this adjustment is based on the implementation of an unsigned scaling engine, and therefore, is to ensure that the scaled values are handled properly. Consider the following table:

Format	pixel0	pixel1	average	Result
decimal	-2	+2	0	Correct
CbCr	0x7E	0x82	0x80	Correct (0x80 is 0 in CbCr)
UV	0xFE	0x02	0x80	Incorrect (0x80 is -128 in UV)
decimal	-32	+16	-8	Correct
CbCr	0x60	0x90	0x78	Correct (0x78 is -8 in CbCr)
UV	0xE0	0x10	0x78	Incorrect (0x78 is +120 in UV)

To compute the output pixel value at position as indicated by P, consider the diagram below.

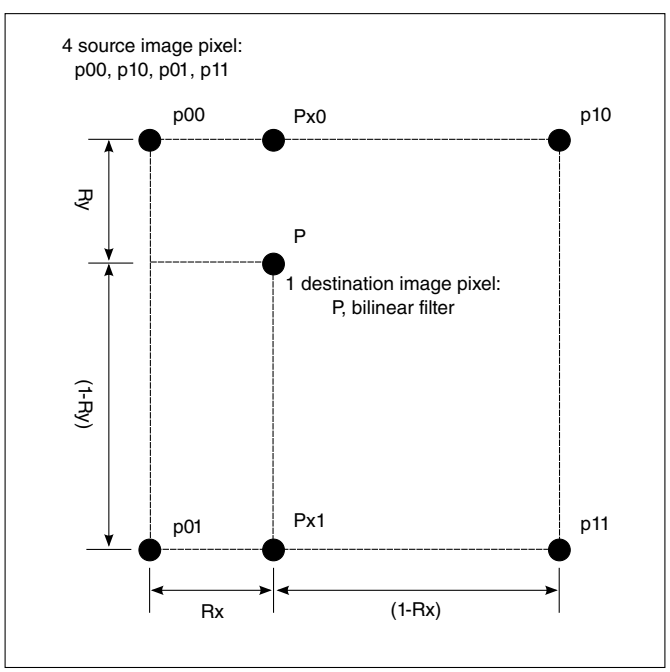


Figure 13-75. Output Pixel Value

A step function is used to indicate the position of the pixel "P" in the output frame. This position may not coincide with a single pixel position in the input frame buffer. In this case, the four closest pixels in the input frame are used to approximate the value of the pixel in the output frame.

The PXP scaler first computes a linear filter in the X axis to create the two intermediate pixel values Px0 and Px1. The step function's X fractional component is used to provide the weighting factor for blending p00 with p10 to provide Px0. Likewise, Px1 is also derived from a linear filter using p01 and p11.

The equations for Px0 and Px1 are as follows:

Px0 = p00*(1-Rx) + p10*Rx

Px1 = p01*(1-Rx) + p11*Rx

The PXP scaler uses the intermediate X pixels Px0 and Px1 and implements a bilinear filter on these two pixel values to produce the final pixel value at position P. The remainder of the step function for the Y axis is used to compute the weighted average pixel result. The equation for final filtered pixel is:

P = Px0*(1-Ry) + Px1*Ry

13.6.3.5 YUV 4:2:2 Image Scaling

The following figure illustrates the positioning of YUV samples for the 4:2:2 formats. There are twice as many Y luma samples then U and V chroma samples horizontally.

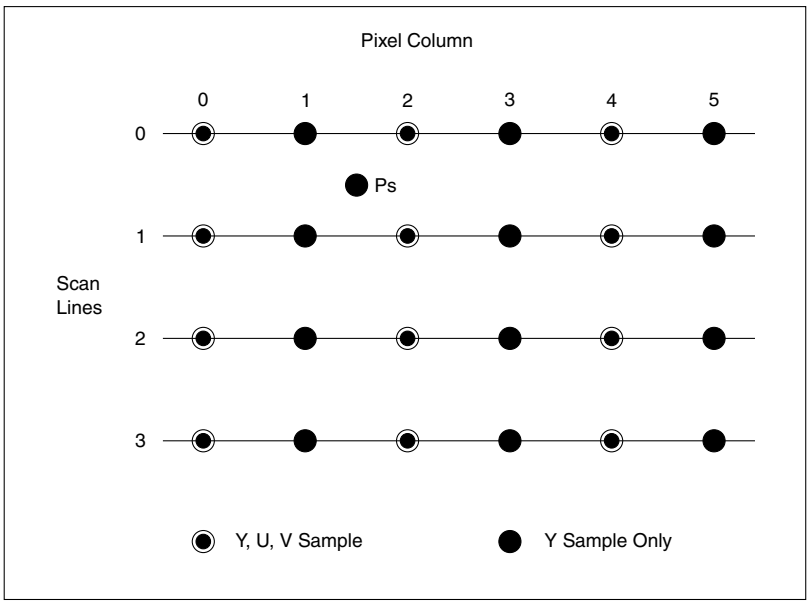


Figure 13-76. YUV Sample Positioning, 4:2:2

Consider the scaled output pixel Ps (pixel scaled) which has an accumulated step function of X=1.5 and Y=0.5. The remainder for the step function is Rx = 0.5 and Ry = 0.5. Or, the sub pixel position of output pixel Ps is half way between line 0 and 1 and half way between column 1 and 2.

The Y output component of Ps is simply the bilinear function of the four nearest Y samples from the input image. Specifically, the Y values at [1,0], [2,0], [1,1], and [2,1] are used to compute the Y for Ps.

For the U and V components of Ps, there are no samples present in the column position 1. The bilinear filter uses chroma components located at [0,0], [2,0], [0,1] and [2,1]. Since the chroma components are not sub sampled vertically, the remainder used to combine pixels vertically is Ry=0.5 (the same as for Y). However, horizontally, the scaling engine shifts the remainder by a factor of 2. So an X axis step function value of X=1.5 has a remainder Rx=0.75. Source chroma values are not replicated, they are completely interpolated using the four nearest chroma samples to approximate U and V at Ps.

13.6.3.6 YUV 4:2:0 Image Scaling

The following figure illustrates the positioning of YUV samples for the 4:2:0 formats. Chroma is sub sampled both horizontally and vertically. In this format, the chroma frame buffers contain ¼ the data that the luma frame buffers store.

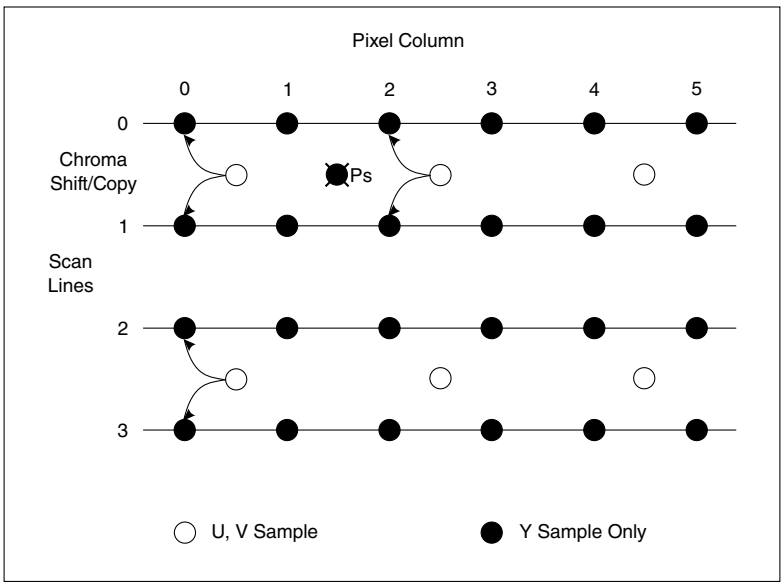


Figure 13-77. YUV Sample Positioning, 4:2:0

The Y output component for all scaled pixels in 4:2:0 formats are the same as for the 4:2:2 pixel formats.

The U and V output components have two considerations when computing the output pixel Ps.

1. All chroma samples from the input source image are shifted left and up by ½ a sample position of the input pixel matrix.
2. Odd scan lines are replicated using the previous even chroma scan line values. So, output image chroma values that map between even to odd scan lines are replicated in the vertical axis. In contrast, output image chroma values between odd to even scan lines are interpolated vertically.

The chroma values are interpolated horizontally as in the 4:2:2 pixel format.

As an example, consider the interpolated pixel Ps in the 4:2:0 diagram above. For the Y component, the interpolated output luma is a function of the Y values in the source frame buffer at position [1,0], [2,0], [1,1], [2,1].

For the U and V interpolated samples, the chroma values on scan line position 0.5 are shifted so that they coincide with the even luma sample points. They are also replicated so that a single chroma scan line is used twice. The chroma scan line at 0.5 is replicated to represent the 4:2:2 sample points for scan line 0 and 1. The chroma scan line at 2.5 is replicated to represent the 4:2:2 sample points for scan line 2 and 3. This pattern of chroma replication occurs for the entire source frame buffer during the scaling operation.

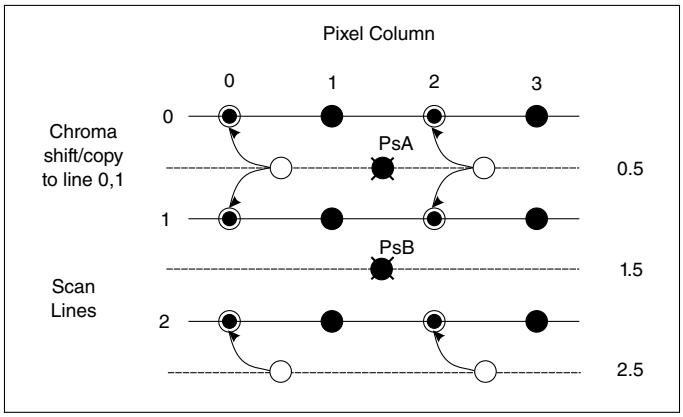


Figure 13-78. Scaled Chroma Computation Examples

The preceding diagram has two examples for the computation of the scaled chroma output pixel. For chroma at output position PsA (vertical position 0.5), interpolation occurs in the X axis using chroma values at column 0 and column 2. However, since line 0 and line 1 have equal chroma values due to chroma line replication, scaling in the Y axis results in replication of chroma values.

For chroma at output position PsB (vertical position 1.5), interpolation occurs in both the X and Y axis. The Y axis is an interpolation since the chroma values copied to scan line 1 and 2 and not the same.

In summary, any output image pixels that map to an odd scan line above and an even scan line below are interpolated vertically. Output image pixels that map to an even scan line above and an odd scan line below are replicated vertically.

13.6.3.7 RGB/YUV444 Image Scaling

For all RGB formats, the RGB pixels are converted up to RGB888 with 8 bits per each color component.

Then each color component is passed to the scaling engine and each component is treated in the same manor. The RGB scaling operation is the same as for the Y scaling operation described in the preceding sections. Also, YUV444 contains a byte for each color plane at each pixel location, so all three color components are scaled in the same manor.

13.6.3.8 Color Space Conversion (CSC)

There are two modules in the PXP to convert pixels between color spaces. They are referred to as CSC1 and CSC2 (for lack of a better naming convention).

CSC1 exists after the scaling unit and is dedicated to converting from YUV to RGB. CSC2 is a full duplex color space converter in that is can convert into either RGB or YUV (or YCbCr) color spaces depending on the desired output pixel format. All coefficients are programmed as two's compliment numbers and both CSC units can be bypassed if CSC is not desired at either position of these CSC units in the pixel data path.

13.6.3.9 CSC1 Operation

The CSC1 module receives scaled YUV/YCbCr444 pixels from the scale engine and converts the pixels to the RGB888 color space only if CSC1 is enabled.

The CSC1 module will convert only to the RGB color space and it can be bypassed to allow YUV pixels through the data path. These pixels are loaded into the pixel FIFO for processing by subsequent modules in the pixel data path.

The following equations are used to perform YUV/YCbCr -> RGB conversion. The constants will be stored in the PXP control registers as two's compliment values to allow flexibility in the implementation and to allow for differences in the video encode and decode operations. In addition, this provides a software mechanism to manipulate brightness or contrast.

R = C0(Y+Yoffset) + C1(V+UVoffset)

G = C0(Y+Yoffset) + C3(U+UVoffset) + C2(V+UVoffset)

B = C0(Y+Yoffset) + C4(U+UVoffset)

Note: In the equations above, U and V are synonymous with Cb and Cr in regards to the color space format of the source frame buffer.

Saturation of each color channel is checked and corrected for excursions outside the nominal YUV/YCbCr color spaces. Overflow for the three channels are saturated at 0x255 and underflow is saturated at 0x00.

The table below indicates the expected coefficients for YUV and YCbCr modes of operation:

Coefficient	YUV	YCbCr
Yoffset	0x000	0x1F0 (-16)
UVoffset	0x000	0x180 (-128)
C0	0x100 (1.00)	0x12A (1.164)
C1	0x123 (1.140)	0x198 (1.596)
C2	0x76B (-0.581)	0x730 (-0.813)
C3	0x79B (-0.394)	0x79C (-0.392)
C4	0x208 (2.032)	0x204 (2.017)

13.6.3.10 YUV versus YCbCr Support

By default, the PXP color space coefficients are set to support the conversion of YUV data to RGB data.

If YCbCr input is present, software must change the coefficient registers appropriately (see the register definitions for values). Software must also set the YCBCR_MODE bit in the COEFF0 register to ensure proper conversion of YUV versus YCBCR data.

13.6.3.11 CSC2 operation

The CSC2 module receives pixels in any color space and can convert the pixels into any of RGB, YUV, or YCbCr color spaces.

All coefficients are programmable and in the two's compliment notation. The output pixels are passed onto the LUT and rotation engine for further processing.

The following equations indicate the CSC2 modules ALU architecture.

Selecting RGB output in REG_CSC2_CTRL[CSC_MODE] configures the ALU in the following manor:

R = A1(Y-D1) + A2(U-D2) + A3(V-D3)

G = B1(Y-D1) + B2(U-D2) + B3(V-D3)

B = C1(Y-D1) + C2(U-D2) + C3(V-D3)

Selecting YUV output configures the ALU in the alternate manor:

Y = A1*R + A2*G + A3*B + D1

U = B1*R + B2*G + B3*B + D2

V = C1*R + C2*G + C3*B + D3

Saturation of each color channel is checked and corrected for excursions outside the nominal color space. Overflow for the three channels are saturated at 0x255 and underflow is saturated at 0x00.

NOTE

CSC2 may not be able to perform RGB to YCbCr or RGB to YUV conversion correctly. Only the Y conversion may be correct. In that case, conversion for CbCr and UV components

may require alternate chip resources either from GPU, IPU, or CPU if applicable.

13.6.3.12 Alpha Blending/Color Key

There are two alpha blending/color key engines in the new PXP. Regardless of pixel input format, the PS and AS pixels are normalized to 32-bits, organized as one alpha and three data bytes. For the second alpha blending engine, the fetch engine should be programmed to fetch data in the RGB format. You can either use the 2 fetch channels or use either fetch channel with the PS.

Alpha blending occurs in the RGB space, if blending is required, PS pixels should be converted to RGB space. If no alpha blending is required, then YUV pixels can bypass the alpha blending ALU without color space conversion.

All pixels are processed by the pixel ALU, but the ALU operations can be disabled to achieve pixel pass through for either PS or AS or fetch data channels source pixels.

13.6.3.13 Alpha Blend

The alpha value for an individual pixel represents a mathematical weighting factor applied to the AS pixel.

An alpha value of 0x00 corresponds to a transparent pixel and a value of 0xFF corresponds to an opaque pixel.

The effective alpha value for an AS pixel is determined by the REG_AS_CTRL[ALPHA] and REG_AS_CTRL[ALPHA_CTRL] register fields. If REG_AS_CTRL[ALPHA_CTRL] = ALPHA_OVERRIDE , the alpha value for the pixel is taken from the REG_AS_CTRL[ALPHA] . This can be useful for applying a constant alpha to an entire image or for image formats that don't include an alpha value. If REG_AS_CTRL[ALPHA_CTRL] = ALPHA_MULTIPLY, the pixel's alpha value will be multiplied by the pixel's ALPHA value in order to allow scaling of the pixel's alpha or to provide better control for pixel formats such as RGB1555, which only contains a single bit of alpha.

For each color channel, the equation used to blend two source pixels is defined below:

Gá = PIO programmed global alpha (8-bit value).

Eá = Embedded alpha associated with AS pixel.

á = Gá * Eá + 0x80

The result for the red channel as an example:

R[7:0] = (á * PS.r) + ((1 - á) * AS.r)

When á is 0xff, the PS pixel will not be blended with the AS pixel, but PS will be passed as the output pixel and will not be blended with AS. In this case, AS will be discarded. Likewise, if á is 0x00 for a given pixel, PS will be loaded as the output pixel.

REG_AS_CTRL[ALPHA_INVERT] provides the option to invert the final alpha value. This essentially inverts the effect the alpha value has on the AS and PS blending operation.

13.6.3.14 Color Key

The color key function is provided to create transparent effects on the output pixel.

Color keying is applied on the input pixels after they are converted to 8-bits for each red, green, and blue color channels (color keys are not applied directly to 16-bit pixel formats but to their corresponding 24-bit representation). A color key range is programmable for both PS and AS pixels. If the PS 24-bit pixel is within the PS color key range, then AS is passed through the pixel pipeline. In this case, alpha blending does NOT occur. Conversely, if PS is within the AS color key range, then PS is passed via the PXP data pipeline. If both PS and AS color key tests pass, then the back ground color register is passed onto following PXP processing components in the pipeline.

The condition for color keying to be satisfied is:

CK0.r.low <= PS.r <= CK0.r.high

CK0.g.low <= PS.g <= CK0.g.high

CK0.b.low <= PS.b <= CK0.b.high

For example, if the "red" 8-bit value for the PS pixel (or PS.r) is between the color key low and high values (CK0.r.l and CK0.r.h), the condition is true for the red color plane. When ALL three color planes meet this condition, then only the PS pixel is loaded into the output register.

To disable color keying, program the low color key register value to 0xff and the high value to 0x00. This will guarantee that the color key range test will never be true.

13.6.3.15 LUT

The lookup table (LUT) is used to modify pixels in a manner that is not linear and that cannot be achieved by the color space conversion modules.

Nonlinear response to the input pixels can be achieved based on how the lookup table is programmed.

Programming of the direct access LUT table can be facilitated by single PIO register writes or DMA access. For efficient loading of the LUT, DMA access should be used.

13.6.3.16 Lookup Modes

The LUT has four lookup modes. The lookup modes determine how the src_pixel is used to address the LUT memory.

The four lookup modes are:

1. DIRECT_Y8
2. DIRECT_RGB444
3. DIRECT_RGB454
4. CACHE_RGB565

The DIRECT modes access the LUT memory as a monolithic SRAM. The CACHE_RGB565 will access the memory as a 2-way set associative cache.

13.6.3.17 DIRECT_Y8

DIRECT_Y8 is used for a 256-byte lookup. In DIRECT_Y8, the most significant byte of the pixel is used to address the LUT entry.

This byte reflects the Y/R channel of the pixel data path. Luma, or monochrome, transformations are possible with this lookup mode. The address is generated as: src_pixel[23:16]. In DIRECT_Y8 operation, the memory is byte addressable.

13.6.3.18 DIRECT_RGB444

DIRECT_RGB444 is used for a 8KB (4K pixel) RGB444 to RGB565 lookup.

Pixel formats that are in the YUV color space at the position of the LUT in the PXP data path can also be converted. To take advantage of the full 16KBmemory, the REG_LUT_CTRL[SEL_8KB] bit can be used to select the upper or lower 8KB memory, thus facilitating the use of 2 separate 444 LUT tables. In DIRECT_RGB444, the src_pixel is RGB/YUV[23;0] data is used to generate the lookup address. The address is generated as: {R/Y[23:20],G/U[15:12],B/V[7:4]}. In DIRECT_RGB444, the memory is pixel (2 byte) addressable.

13.6.3.19 DIRECT_RGB454

DIRECT_RGB454 is used for a 16KB (8K pixel) RGB454 to RGB565 lookup.

Pixel formats that are in the YUV color space at the position of the LUT in the PXP data path can also be converted. In DIRECT_RGB454, the src_pixel is RGB/YUV[23;0] data is used to generate the lookup address. The address is generated as: {R/Y[23:20],G/U[15:11],B/V[7:4]}. In DIRECT_RGB454, the memory is pixel (2 byte) addressable.

13.6.3.20 CACHE_RGB565

The CACHE_RGB565 lookup is used for a 128KB (65K pixel) RGB/YUV565 to RGB/YUV565 lookup.

The 128KB memory requirement is too costly from an area perspective to implement a complete lookup table in the LUT's on chip memory. For this reason, a LRU (least recently used) 16KB 2-way set associative cache has been implemented to reference the full 128KB lookup table stored in external memory.

The 2-way set associative cache is organized in the following way:

- 16KB total data storage
- 512 entries split between 256 ways
- 6 pixels/entry cache line

Cache efficiency is very critical. For a cache miss, the PXP will be stalled until the cache line can be filled. For a 32 bit DDR memory interface, the latency can be calculated as follows:

cycles latency for read command through DDR controller + CAS Latency + # burst cycles for read data to be returned + # cycles latency for data through DDR controller to LUT + 1 cycle for cache access of new data

The src_pixel is RGB[23:0] data used to generate the lookup address. To improve the cache efficiency the RGB/YUV565 address and the lookup table must be formatted in the following way:

Address: A[15:0] = $R_7G_7G_6B_7$ $R_6G_5B_6$ $R_5G_4B_5$ $R_4G_3B_4$ $R_3G_2B_3$

The address is organized as follows:

- A[15:12] tag address, used to compare a hit between cache ways
- A[11:4] index address, used to select cache set (i.e. row of memory)
- A[3:0] block address, used to select Pixel in the cache line

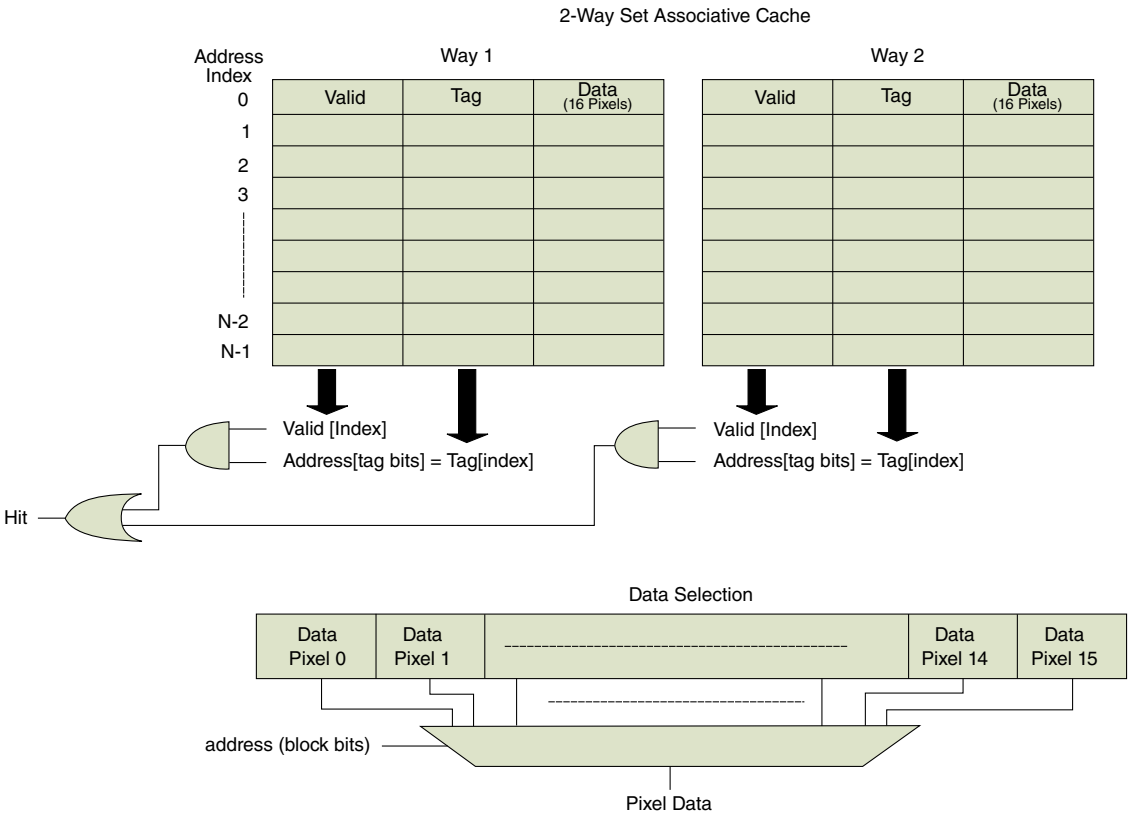


Figure 13-79. 2-Way Set Associative Cache and Data Selection

13.6.3.21 Output Modes

The LUT has three output modes for color space conversion.

1. Y8
2. RGBW4444CFA
3. RGB888

13.6.3.22 Y8

With out_mode set to Y8, in conjunction with DIRECT_Y8 lookup mode, the intended operation is Gama Correction.

Only the third byte is processed by the lookup table. The third byte is represented by the Y value or the R value in the data path since pixel data is either YUV[23:0] or RGB[23:0] where the Y or R byte encompass bits [23:16] respectfully. So, bits 15:0 are

always bypassed and left unchanged. Currently, the LUT is intended to process Y data when any YUV, Y8, or Y4 output pixel formats are selected. However, this resource can be enabled and used for any conceivable purpose.

Note: When the DIRECT_RGB444, DIRECT_RGB454 or CACHE_RGB565 lookup mode is selected in conjunction with the Y8 output mode the low order byte of the two bytes read from the LUT memory will be used as the Gama Correction value.

13.6.3.23 RGBW4444CFA

With REG_LUT_CTRL[OUT_MODE] set to RGBW4444CFA, the REG_CFA[DATA] is used to select one nibble from the LUT 16 bit output value.

The LUT memory lookup will contain a RGBW4444 value. The REG_CFA[DATA] will select the R,G,B or W nibble as the pixel value to present to the PXP data path based on the matrix defined by the REG_CFA[DATA] register. The 4 bit value is presented in the Y, or third byte lane, of the PXP data path. The final pixel transferred to the next PXP stage is {CFA[3:0],CFA[3:0],LUT[15:0]}

13.6.3.23.1 CFA Correction

The 32-bit REG__CFA[DATA] register is used to encode 16 CFA correction values.

The CFA correction values are encoded as follows:

- 00 selects R
- 01 selects G
- 10 selects B
- 11 selects W

The CFA correction uses 4x4 block processing. Figure 5; CFA mapping translation shows how CFA 4x4 blocks will iterate over the PXP's 8x8 or 16x16 pixel block being processed:

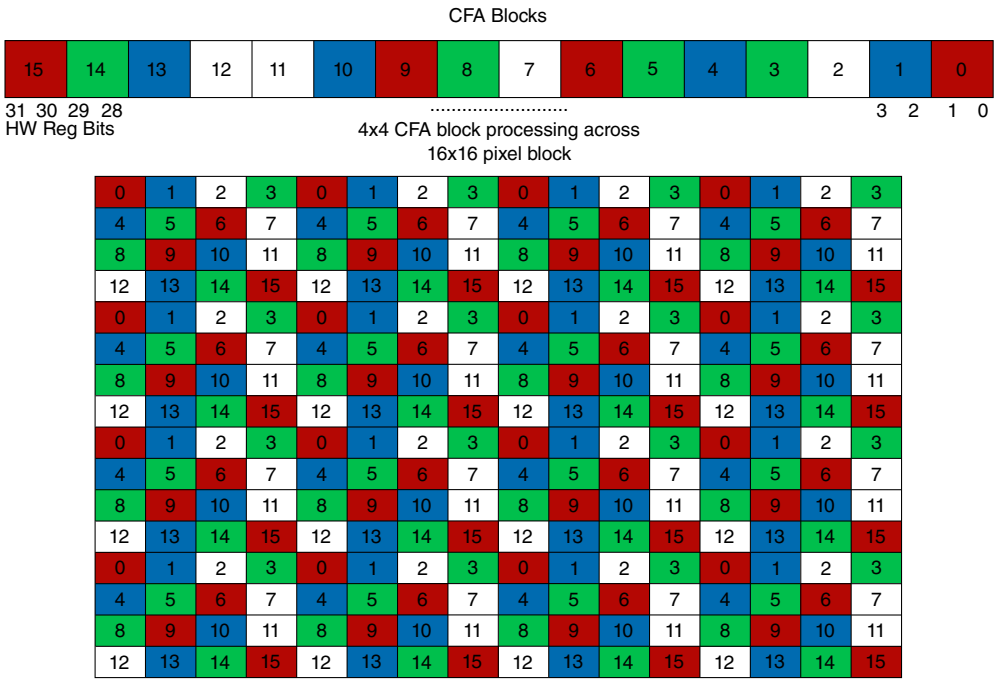


Figure 13-80. CFA mapping translation

13.6.3.24 RGB888

With out_mode set to RGB888, the memory output data is interpolated from RGB565 to RGB888.

The RGB[23:0] data is formatted as follows: R[7:3]R[7:5],G[7:2]G[7:6],B[7:3]B[7:5].

13.6.3.25 Rotation

There are two rotation engines integrated into the PXP. Rotation can occur after compositing the AS and PS buffers in the output stage.

The PS buffer can be rotated and later composited with the AS surface that is not rotated. The rotation1 engine can also be used with the Input Fetch engine.

To rotate graphics, the hardware must read pixels in one direction across a frame buffer and write them in a alternate orientation. For the 90 and 270 degree cases, this means that lines of pixels must either be read or written vertically in a frame buffer.

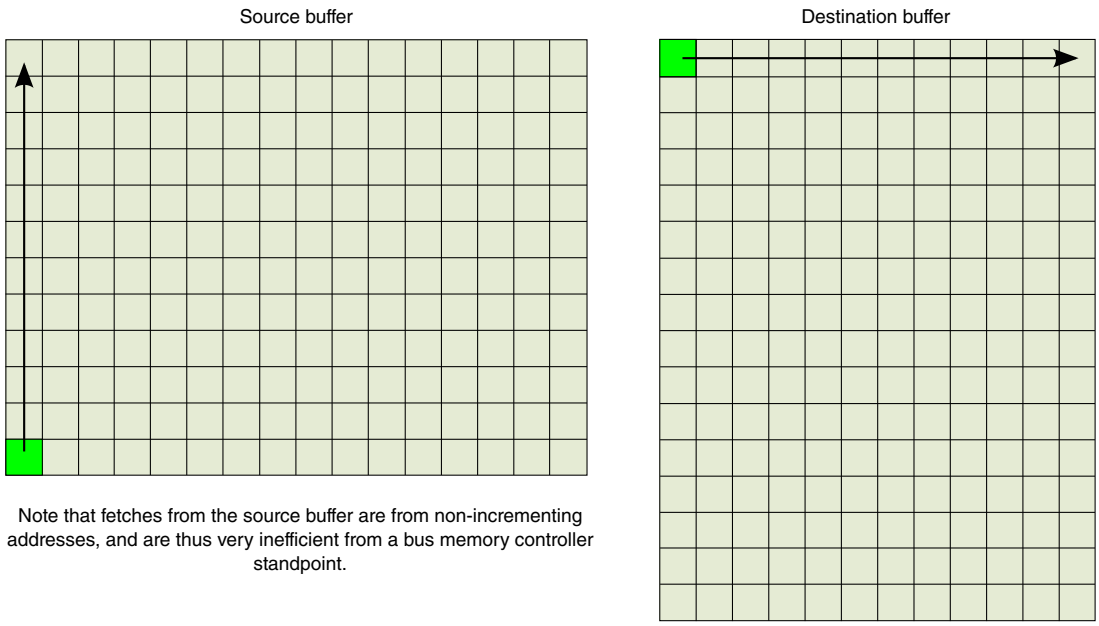


Figure 13-81. Rotation Read and Write

In order to rotate efficiently, multiple columns must be rotated to enable the engine to both fetch and store bursts of pixels, thus improving memory performance. The simplest method of doing this is to operate on square blocks of pixels. To rotate the image, each sub-block of pixels must be rotated by the required rotation angle.

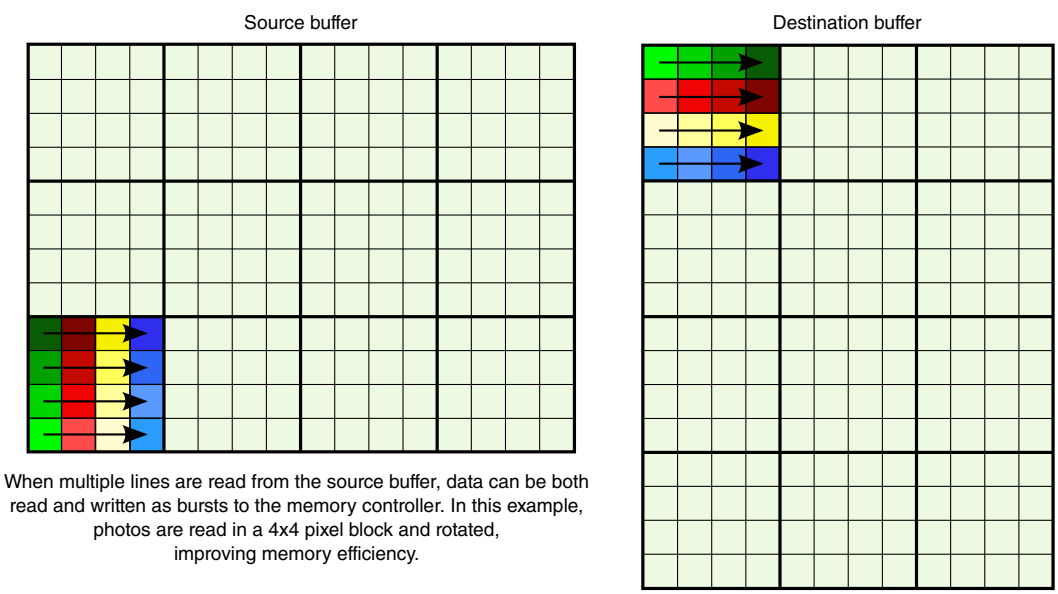


Figure 13-82. Rotated Sub-blocks

To manage the rotation process, the source image can be broken into a grid of sub-blocks that have coordinates as shown in the diagram below. In addition to rotating the sub-block, each block must be translated to a new coordinate location. For each of the rotation angles (0, 90, 180, 270), it is possible to define a simple algorithm for computing the new translated grid address. The hardware must then simply compute the memory address from the base grid address for both load and store operations.

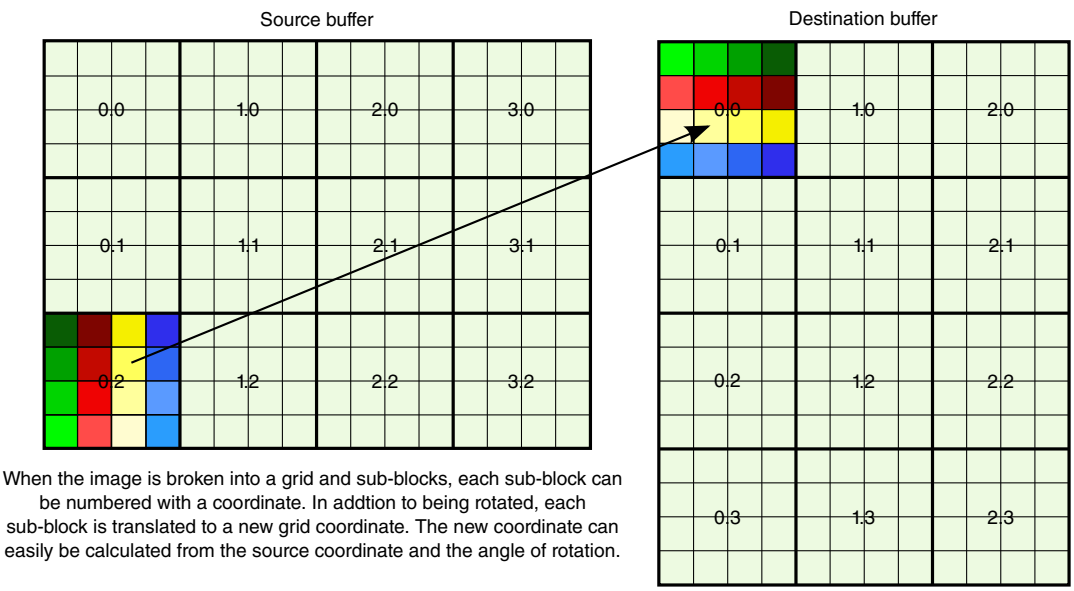


Figure 13-83. Grid of Sub-Blocks with Coordinates

In order to balance the requirements of reasonable burst sizes to the memory controller as well as keep the hardware storage requirements to a minimum, the blending/rotation engine will operate on either 8x8 or 16x16 pixel blocks. When using the Rotate engine with the input fetch engine, you need to program the input fetch engine to work in 8x8 block mode.

NOTE

An important artifact of the PXP is when rotating a source image and the output is NOT divisible by the block size selected. The output engine essentially truncates any output pixels after the desired number of pixels has been written. Since the output buffer is written as a horizontal row of blocks, the incorrect pixels could be truncated and the final output image can look shifted. In the case where the block size is programmed to 8x8, and the output size that is programmed is 12x12, then there is a remainder of 4 pixels that will be truncated in either the X and/or Y axis when the PXP operation

is complete. The output will be shifted by 4 pixels in this example. To compensate for this, the source base address needs to be adjusted so the correct pixels get truncated and the image does not look shifted. In this example, with 90 degrees of rotation, the PS base address should be adjusted by 4 times the actual PS base address -(4*pitch).

NOTE

First position Rotation occurs after compositing the AS and PS buffers in the output stage. When processing PS and AS buffers that are unaligned (buffers are not aligned to block boundaries), a rotation operation that is combined with a flip, decimation, or scaling operation will not execute correctly. Rotation operation must be done in separate passes. Combination operations execute correctly for aligned buffers.

NOTE

Second position Rotation only supports simple aligned rotation. Unaligned buffer rotation will not execute correctly. Rotation operations combined with flip, scaling, or decimation will not execute correctly.

13.6.3.26 Output Buffer

The output buffer engine accepts data from the PXP pixel pipeline and issues requests to transfer the output pixels to external DRAM or the internal SRAM double buffer row of blocks. It can also connect to the dither engine using the handshake mode. This works for the output engine programmed in the scan line mode as well as block mode. For more details, please refer to the Dither fetch engine.

13.6.3.27 Address calculator

Each of the blocks will manage its own fetch address using a common address calculator block that computes real addresses from a base address and relative block offset from the base.

Each block will then perform the multiple line fetches (or stores) required to perform the operation. This hides all the address buffer computations from the processing blocks and allows each block to simply track the coordinate of the block it is working on.

13.6.3.28 Block size selection

The PXP can be configured to process blocks that are either 8x8 pixels or 16x16 pixels with the REG_CTRL[BLOCK_SIZE] control bit.

When selecting a 16x16 pixel block size, the accesses to fetch AS and PS images and write the final frame buffer are more efficient since twice as much data is requested and processed per memory request.

When optimizing the system for memory bandwidth and image processing time, configure the PXP to process 16x16 pixel blocks.

13.6.3.29 Interlaced Video Support

The PXP has some minimal ability to generate interlaced video content from a progressive source. There two available options, based on the bandwidth requirements and how software is managing video frames.

The PXP can either interlace on the input side (by reading every other line of input data) or on the output side (by writing the individual lines of video into two separate fields). Generally, output interleaving should be used since it is the most flexible mode (it allows scaling and full overlay support) and it only requires a single pass of the PXP to generate two separate output fields.

Input interleaving can be beneficial in cases where the PXP is running at 60fps, since it requires fewer fetches to produce the output data. There is no direct hardware support for input interleaving, in that, there is no configuration bit that can be set to alter how the PXP processes a frame for input interleaving. Input interleaving is achieved by simply setting the source frame buffer pitch value to twice the value it would normally be set to for the equivalent progressive frame. The output parameters also need to be consistent with the desired processing effect. For example, the vertical resolution would be set to account for the reduced resolution to process the interlaced input buffers.

13.6.3.30 LCDIF Handshake

The PXP and LCDIF support a mode where the internal SRAM can be used for the frame buffer to minimize external memory bandwidth required.

This is accomplished by creating two buffers in SRAM, a double buffer row of blocks, where each correspond to 8/16-lines of the frame buffer. The buffers must be consecutive and allocated as a single block of data.

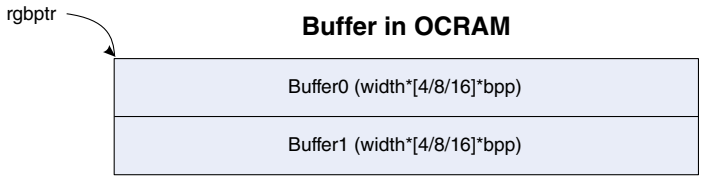


Figure 13-84. Buffer in OCRAM

The storage required can be calculated for an 8x8 block size as

- storage = 16 (lines) * rotated_row_length * pixel_size

and for 16x16 block size as

- storage = 32 (lines) * rotated_row_length * pixel_size

where pixel_size = 4 for 32bpp or 2 for 16bpp modes. The following table lists the storage requirements for common image sizes using 8x8 block size:

Image Size	Storage (16bpp)	Storage (24bpp)	Storage (32bpp)
320x240 (QVGA) - 0/180 rotation	10KB	15KB	20KB
320x240 (QVGA) - 90/270 rotation	7.5KB	11.5KB	15KB
640x480 (VGA) - 0/180 rotation	20KB	30KB	40KB
640x480 (VGA) - 90/270 rotation	15KB	22.5KB	30KB

The following diagram shows how the minimal rotation buffer would be organized. As the engine and LCD progress down the image, they continually swap roles of filling and emptying each eight-line buffer.

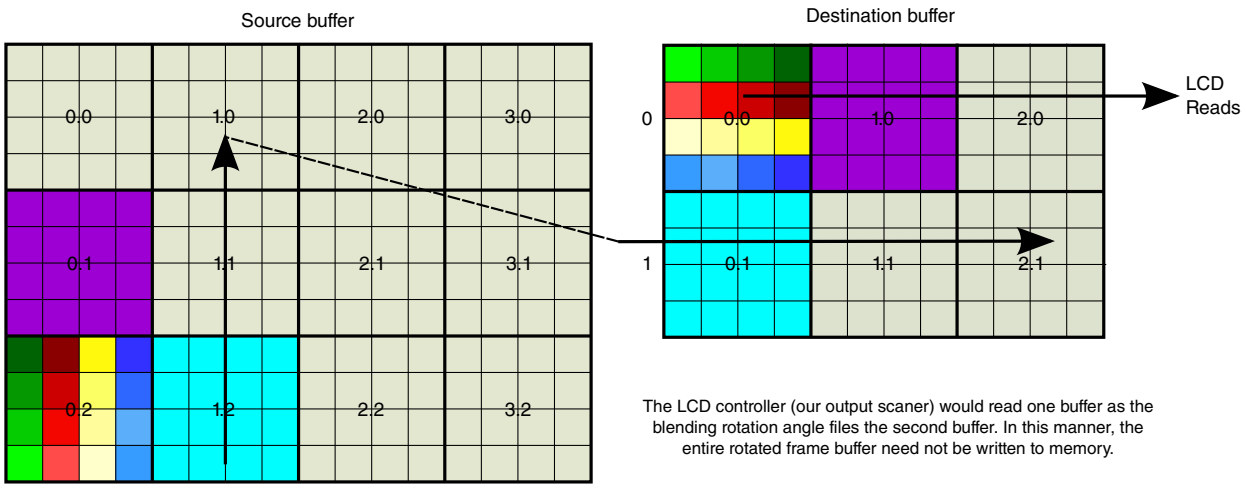


Figure 13-85. Minimal Rotation Buffer Organization

When this mode is enabled, the PXP will process one row of pixel blocks and write the results to the first SRAM buffer (buffer 0). The PXP will then alternate between writing subsequent rows to buffer 0 and buffer 1. After the PXP generates the data for one buffer, the LCDIF will begin reading that buffer and send the contents to the display device. Once the LCDIF finishes reading a buffer, it will start displaying from the other buffer while the PXP continues filling the previously processed buffer.

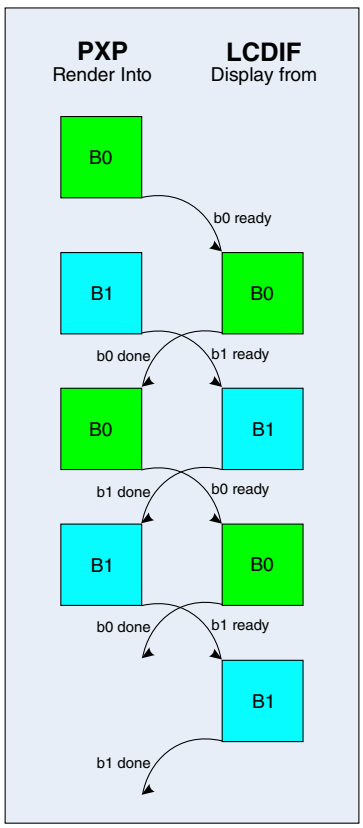


Figure 13-86. PXP and LCDIF Buffer Sharing

To accomplish the buffer sharing, the PXP and LCDIF will maintain buffer status using a pair of handshake signals. When a buffer is filled by the PXP, it will assert the pxp_lcdif_bx_ready (where x is 0 or 1) signal to indicate to the LCDIF that the buffer has valid data. The LCDIF will then release the buffer by asserting the lcdif_pxp_bx_done signal.

The basic protocol is shown in the diagram below:

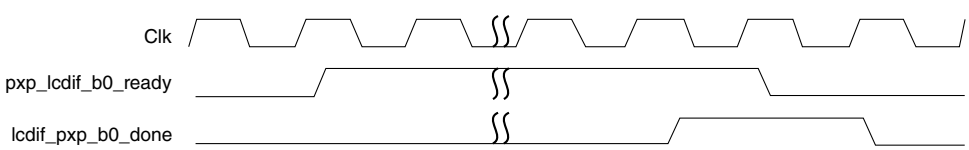


Figure 13-87. Buffer Sharing Protocol

The PXP will continue to assert the bx_ready signal until the corresponding bx_done signal is sampled high for one clock cycle. It will then deassert the bn_ready until the next time the buffer has been filled. After the PXP samples the bx_done signal asserted, it is free to begin filling the buffer with the next block size lines of display data. If a buffer has not been released when the PXP is ready to process data for that buffer, it will suspend rendering operations until the buffer has been released by the LCDIF.

13.6.3.31 LCDIF Abort

When the memory subsystem is not loaded, the PXP should be able to render the buffers faster than the LCDIF can drain the buffers.

It is possible under some scenarios (high LCDIF output rates with high memory latency) that the PXP may not be able to keep up with the LCDIF, even in the SRAM mode of operation. When this happens, the LCDIF will signal that it has completed one of the buffers before the other has been rendered by the PXP. This condition will be detected by the PXP's control logic as an "LCDIF Abort", which will cause the PXP to abort processing in the current row and proceed to the following row. It will acknowledge the abort to the LCDIF by raising the buffer_ready signal for the current buffer to enable the LCDIF to begin displaying the partially-filled buffer. While an abort will create artifacts in the video display, it does minimize the artifacts by limiting them to the remaining pixels blocks in the current row versus ruining the entire frame buffer.

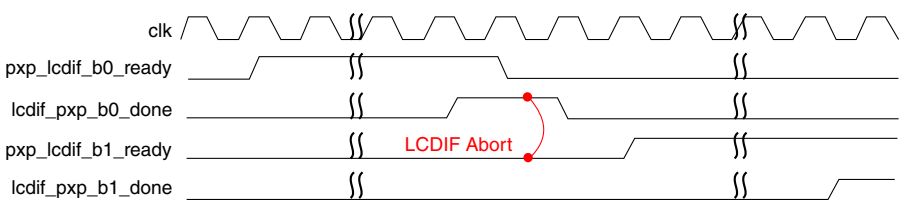


Figure 13-88. LCDIF Abort

13.6.3.32 Theory of Operation

The PXP can be used to accelerate graphics operations by offloading graphics processing from the processor. The block can perform alpha blending and color key substitution on two RGB graphics buffers.

The PXP is organized as having a processed surface (PS) and an alpha surface (AS) that can be blended with the processed surface. There are no restrictions on the location of the AS or PS within the output surface (OS). As the PXP processes NxN blocks, operations

are performed on a pixel by pixel basis. The AS and PS pixels are alpha blended, color keyed, process by CSC and LUT resources as individual pixel components. This allows efficient block processing with supporting arbitrary alignment for both the AS and PS surfaces. The resulting pixel block is then written to the corresponding block in the output buffer.

13.6.3.33 Pixel Handling

All pixels are internally represented as 24-bit values regardless of input or output pixel formats.

The pixels get converted in the AS and PS buffer engines to 24-bit pixel values.. There is also an 8-bit alpha value at stages up to the alpha blender within the PXP for blending within the RGB color space. Compositing of AS and PS images can only occur in the RGB color space. If composting is not required, then YUV pixels can be transferred and processed at all PXP pixel resource components.. The color orientation of pixels within the PXP can be controlled by the CSC1, CSC2, and LUT resources.

For RGB, input pixels are converted into 24-bit pixel values using the following rules:

1. 32-bit ARGB8888 pixels are read directly with no conversion for both the AS and PS.
2. 32-bit RGB888 pixels are assumed to have an alpha value of 0xFF (full opaque).
3. 6-bit RGB565 and RGB555 values are expanded into the corresponding 24-bit color space and assigned an alpha value of 0xFF (opaque). The expansion process replicates the upper pixel bits into the lower pixel bits (for instance a 16-bit RGB555 triplet of 0x1F/0x10/0x07 would be expanded to 0xFF/0x84/0x39).
4. 16-bit RGB1555 values are expanded into the corresponding 24-bit color space and assigned an alpha value of either 0x00 or 0xFF, based on the 1-bit alpha value in the pixel. The ALPHA_MULTIPLY function is useful in this scenario to allow scaling of the opaque pixels to a semi-transparent value.

Alpha values can be passed through the entire PXP data path and output in ARGB888 and ARGB555 pixel modes. Also, output pixels can be assigned an alpha value using the REG_OUT_CTRL[ALPHA] register. 16-bit pixels values are formed from the most significant bits of the 24-bit pixel values.

When YUV/YCbCr output formats are selected, all pixels are internally represented as either RGB or YUV pixels values. The CSC2 or LUT can convert internal RGB/YUV pixels into the correct output format. In this way, any PS color space can be blended with AS RGB pixels and output in any color space.

13.6.3.34 Output Buffer Composition

The output buffer will be rendered by composing each pixel block from the associated PS and AS buffers.

The AS pixel buffer can be blended or color-keyed with the associated data from the PS buffer (either the PS image pixels or REG_PS_BACKGROUND register based on PS programmed coordinates).

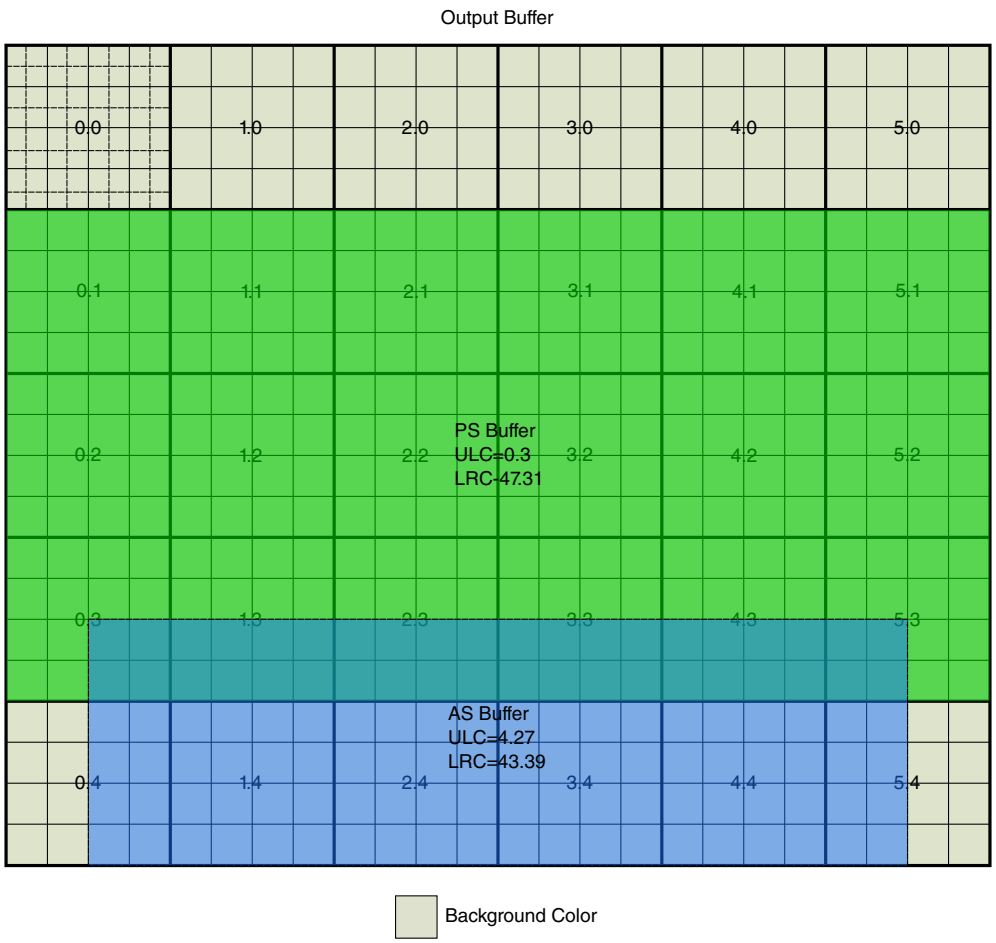


Figure 13-89. Output Buffer Composition

13.6.3.35 PS Image Processing

As the PXP processes image buffers, it iterates over the output buffer by fetching the corresponding input buffer blocks and processing the pixels embedded in these.

13.6.3.36 Letterboxing

At each pixel coordinate, the control logic determines if the PS pixel (argument also applies to AS pixels) will be used in rendering the output pixel.

This is determined by checking the output pixel's coordinates against the REG_OUT_PS_ULC and REG_OUT_PS_LRC (ULC and LRC in short) register contents. For pixels outside this region, the PS pixel will be loaded with the pixel value from REG_PS_BACKGROUND, which can be used to effectively control the letterboxing color. There are no block size or block boundary restrictions when setting the ULC or LRC for either the AS or PS. The only restriction is that the ULC and LRC are within the OUT LRC extents.

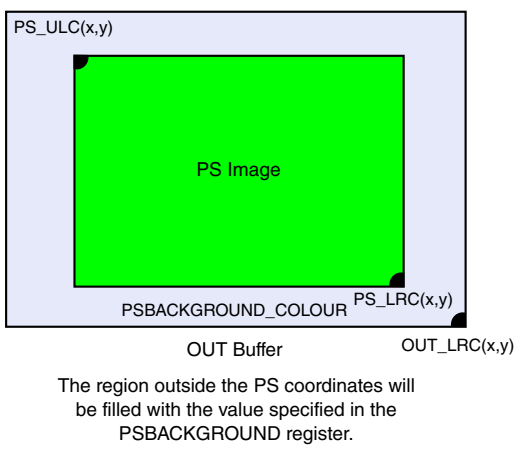


Figure 13-90. OUT Buffer

13.6.3.37 Clipping source images

A subset of the PS buffer can be used in rendering the output buffer. The PXP_PS_BUF register can indicate an offset into the PS buffer that will be used for display within the OUTPUT buffer.

The pixel at the address defined in the PXP_PS_BUF register will be the pixel that is displayed at the pixel coordinate indicated by PXP_OUT_PS_ULC within the output buffer. Essentially, the PXP_PS_BUF register can be used to establish an offset into the PS buffer thus clipping all PS buffer pixels that are at a lower address. The PXP_PS_PITCH will always indicate the number of bytes that are vertically adjacent in

the PS buffer. The settings in the PXP_PS_BUF, PXP_OUT_PS_ULC, and PXP_OUT_PS_LRC will determine the subset of the PS buffer, or clipped PS source buffer, that will be used in the output buffer.

It is important to note that when scaling the PS buffer, the coordinates of the PS buffer within the output buffer need to be consistent with the scaling factors and original PS buffer size.

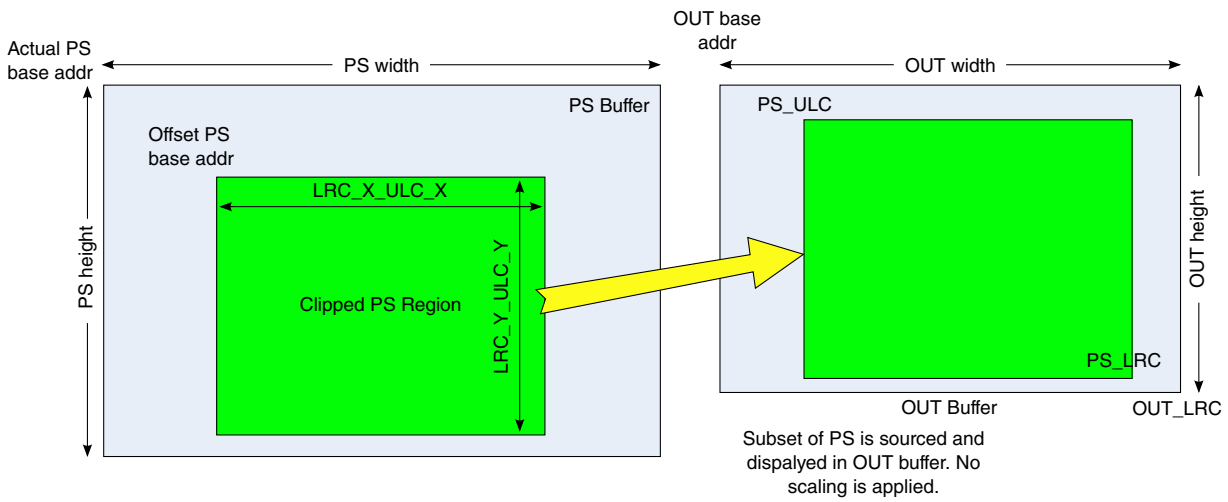


Figure 13-91. PS Buffer Scaling

When sourcing a subset of the PS image, it should fall completely within the PS buffer to avoid displaying incorrect data. The following conditions should be met:

x_base_addr_offset + x_scale * (LRC_X-ULC_X) <= PS_pitch

y_base_addr_offset + y_scale * (LRC_Y-ULC_Y) <= PS_size

The PXP hardware does not check for these conditions and will render the image as programmed. The following case could indicate invalid programming parameters for the PXP:

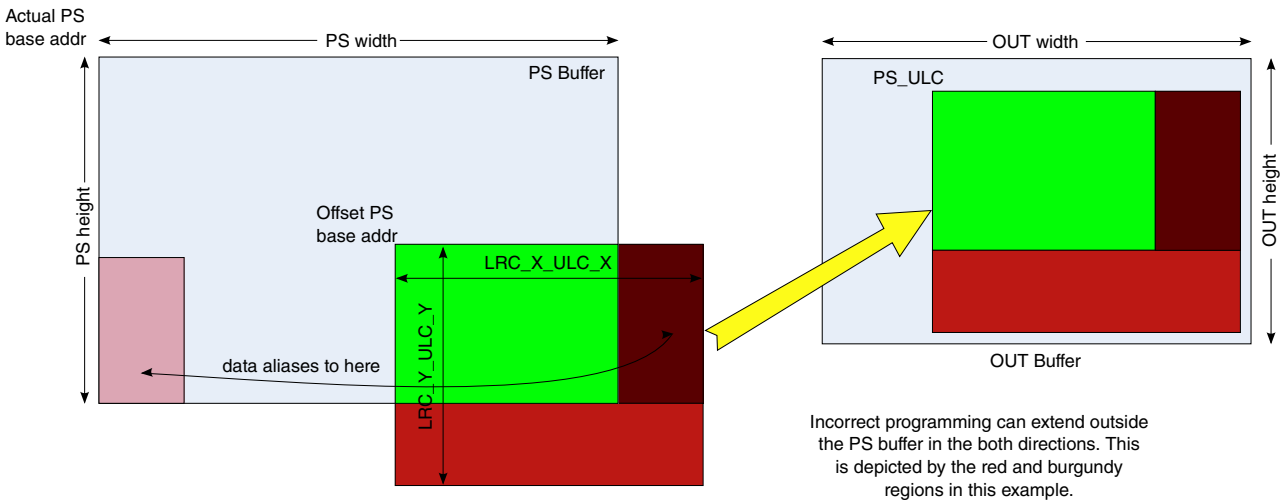


Figure 13-92. Example with Invalid Parameters

13.6.3.38 Color Key Processing

Pixels may be made transparent to the corresponding AS by using the PS color key registers.

If a PS pixel matches the range specified by the REG_PS_COLORKEYLOW and REG_PS_COLORKEYHIGH registers, the pixel from the associated AS will be displayed. If no AS is present for the pixel, a black pixel will be generated since the default AS pixel is 0x00000000 (transparent black pixel).

The most common use for this is when a bitmap does not support an alpha-field or for applications such as "green screen" where an image is substituted for a solid background color .

Figure 13-93. The PS image (player) and AS image (stadium)

The green portion of the background image can be color keyed to display the contents of the AS buffer for locations that match the color range. For this example, the color range is:

PS Colorkey: 00<R<80 70<G<ff 00<B<80

The resulting image becomes:

Figure 13-94. Resulting Image

13.6.3.39 In Place Processing (PS buffer is destination buffer)

The PXP also has the ability to process an image and write the resulting buffer back to the original PS buffer. This is referred to as "in place" rendering.

This could be useful for basic blit operations into the PS buffer. IN_PLACE operations are achieved by programming the OUT base address to the pixel location in the PS buffer that marks the upper left pixel of the update region. The actual region that is updated should be indicated by programming the ULC = (0,0) and the LRC = (X,Y). The region bounded by the coordinates will be updated, and the rest of the PS buffer will not be modified.

13.6.3.40 Alpha Surface (AS) Processing

The AS surface has a complete set of registers that determines how the AS effects the final OUT surface.

Most of the registers that exist for the PS surface also are defined for the AS surface where applicable. This is provided to replicate the SW interface for each PS and AS processes.

13.6.3.41 Alpha Handling

Alpha values in the AS are embedded in the source image pixels. For AS pixel formats that do not support an alpha value, the pixel is assigned an alpha value of 0xFF (opaque).

This can be modified by the AS control by setting either the ALPHA_MULTIPLY or ALPHA_OVERRIDE bit in the associated AS_CTRL register. If ALPHA_MULTIPLY is enabled, the 8-bit ALPHA value from the AS_CTRL register is multiplied by the source alpha before blending with the PS image. If the ALPHA_OVERRIDE bit is set, the 8-bit ALPHA value is simply substituted for the pixel.

13.6.3.42 Color Key Processing (AS_CTRL)

The AS_CTRL register also contains an ENABLE_COLORKEY bit that can be used to enable or disable color key substitution for the AS.

When enabled, the pixel values are compared to the ASCOLORKEYLOW and ASCOLORKEYHIGH registers to determine if a match has occurred. When an AS pixel matches the color key range, the pixel from the AS image is considered transparent and the corresponding PS pixel is rendered. If both the PS and AS pixels match their corresponding color key ranges, the AS pixel is displayed unmodified.

AS color keys are handled in a manner similar to PS color keys. The same images used in the PS color key example could be used with the images swapped. In this case, matches on the AS image to the ASCOLORKEY register would display the PS pixels.

13.6.4 Output Image Processing

Several PXP options affect the resulting output image.

13.6.4.1 Output Image Size

The PXP generates an output image in the resolution programmed by the REG_OUT_LRC. As the PXP processes pixels, it iterates over the NxN blocks (in output scan-block order) based on the final image resolution.

13.6.4.2 Output Format

The result of PXP operations are written to the buffer pointed to by the REG_OUT_BUF/ REG_OUT_BUF2 registers. The pixel format is controlled by the REG_OUT_CTRL[FORMAT] bit-field.

32-bit pixels are formed directly from the internal 24-bit representations and 16-bit pixel formats are generated by truncating the internal 24-bit values to the appropriate number of bits. For formats supporting an alpha value, the PXP assigns the alpha using the 8-bit value in the REG_OUT_CTRL[ALPHA] field. For ARGB1555, the most significant alpha bit is appended to the output pixel. Also, for ARGB4444, the most significant nibble is appended to the output pixel. Single and dual buffer YUV output formats are also available. Since each pixel in the data path is represented by a full YUV444 24bpp value, decimation reduces the output in cases of YUV422/420 output formats.

13.6.4.3 Rotation/Flip operations

The PXP supports four rotation angles in conjunction with vertical and horizontal flip options. The flip operations effectively take place before the rotation.

Rotations of 0, 90, 180, and 270 degrees are supported and any combination of rotation and flip are supported. There is no performance difference between any of these modes of operation.

13.6.5 Queuing PXP transactions

The PXP supports a primitive ability to queue up one operation while the current operation is running. This is enabled through the use of the REG_NEXT register.

When this register is written, it enables the PXP to reload its current register contents with the data found at the location pointed to by this address when it completes processing of the current frame. This feature may be useful in helping to reduce the interrupt latency in servicing the PXP, especially in cases where the PXP and LCDIF are using the on-chip SRAM buffer handshake (since the PXP must begin generating next frame data immediately).

If the PXP is idle when the REG_NEXT register is written, the PXP treats this as an indication that it should immediately load the values at the pointer and begin processing the frame. This ability should allow software to use the same routines when programming the PXP (so that the first frame doesn't differ from subsequent frames).

When loading values from the NEXT register, all registers in the PXP are reloaded. Some register loads have no effect

After writing the REG_NEXT register, the PXP will set the REG_NEXT[ENABLED] bit of the REG_NEXT register to indicate that the next command has been queued. Software should first check the status of this bit to ensure that a previous command has not been enabled. Likewise, after programming the first frame in a sequence of frames, software should poll this bit until it is sampled logic 1'b0 before queuing the next operation.

The PXP will issue interrupts from frames as they complete, regardless of whether they were started by writing the control registers directly or using the REG_NEXT register. When software receives an interrupt, it should check/clear the PXP's status register as normal, poll the REG_PXP[ENABLED] bit, and then issue the next operation. A queued operation may be cancelled by issuing a CLEAR operation to the REG_PXP[ENABLED] register bit. The SET and TOGGLE operations should never be used with this register.

13.6.6 Error Handling

The PXP does minimal checking on the control registers, so it is important that these are correctly specified. The PXP does monitor the bus transactions for errors and will report errors in the status register.

Upon receipt of a bus error, the PXP will set the ERROR interrupt and abort any further operations. Bus errors can be generated from any system access that results in an error response returned from the internal SIM Bus errors in the PXP are signaled as either a read or a write error, but do not indicate the failing address. Software may deduce the failing address from the current block status indicators.

13.6.6.1 Known PXP Limitations/Issues

The PXP has the following known limitations:

1. When using the NEXT register, the interrupt enable setting should remain the same for all frames. If not, the PXP will change the interrupt enable register value and possible cause the loss of an interrupt.
2. Rotations of 180/270 are not supported when performing LCD handshakes
3. Rotation 1 engine doesn't support rotation of data in YUV420 format.
4. CSC2 cannot perform RGB to YCbCr and RGB to YUV conversion correctly.

13.6.7 Dither Engine Block

The dither engine takes in a buffer of Y8 grayscale pixels (8 bpp) one at a time in raster order from the Fetch engine that is connected to its input. It is designed to process ~1 pixel/clk. It can process the pixels in a number of ways. It can write out the pixels unmodified (pass through), it can quantize the pixels from 8 bits to any smaller number of bits down to 1 bpp. In addition to quantization, it can perform 1 of three different dithering algorithms on the pixel stream. The dithering algorithms supported are Floyd-Steinberg, Atkinson and Ordered dithering. The block contains internal storage for intermediate error data and LUT data which is completely managed within the block.

The pixels passed through the output interface to the connected Store Engine on the output side. The Store Engine is responsible for storing the dithered pixels out to memory or passes them to the WFE-A Fetch engine through bypass mode. The Fetch, Dither and Store engines work together as a unit and they must all three be programmed appropriately and started for the pixels to be read, processed and written correctly. The Store Engine interrupt signal is the indication to software that the operation has completed.

A LUT (Look Up Table) transform can be configured to be done on the pixel either before or after dither dithering but not both. In addition there is a “final” independent register based LUT that can be applied at the end of the dither process just before the pixel is written out to the Store Engine. The LUT values for this final LUT are programmed by software into register.

There are 3 instances of the dither processing block (Dither0, Dither1 and Dither2) instantiated in the system and each has separate controls.

There are 3 instances of the dither processing block (Dither0, Dither1 and Dither2) instantiated in the system and each has separate controls.

13.6.7.1 Top Level Connections

The dithering engine is connected to the Fetch Engine on the input side and the Store Engine on the output side. See the figure below:

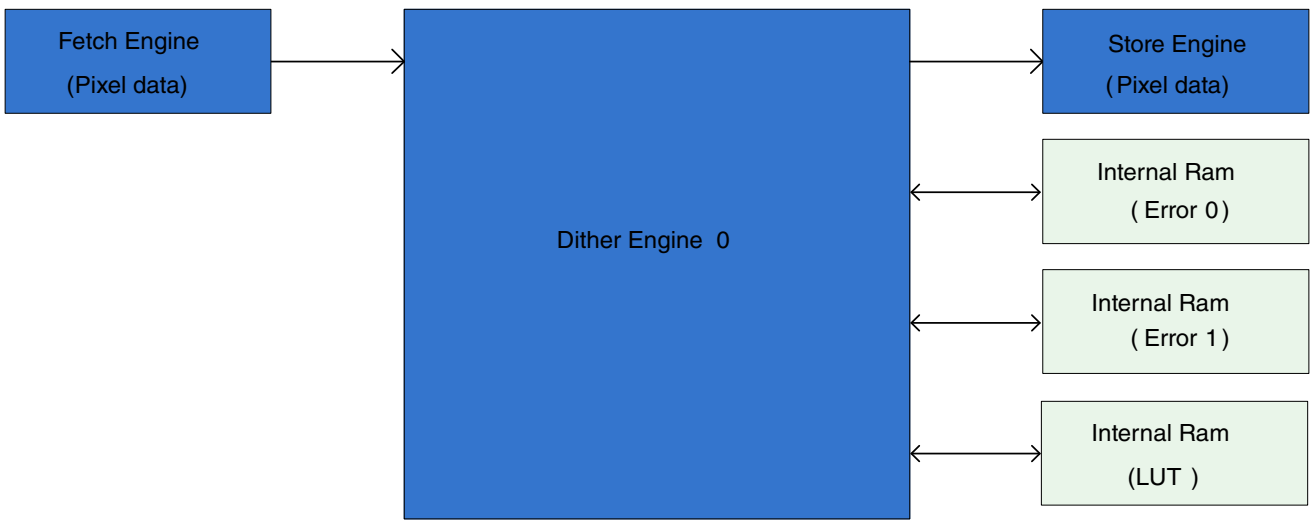


Figure 13-95. Dither Engine External Connections

The Dither Engine must be configured and enabled through the HW_DITHER_CTRL control register. The data flow is controlled through the valid, ready and pixel handshake signals with the Fetch and Store blocks. The Dither Engine must be enabled and given all required information such as which dithering algorithm to use, LUT configuration, etc. After starting, the Fetch Engine will signal to the dither engine when it has data for processing. When error information is needed, the block will only take in the pixel and error information when all handshakes are ready. In this case the pixel and error data must be kept synchronized at the processing step. After that however, they can be written out independently to the store engine (They don't have to be written out on the same cycle).

The dimension information of the operation is not programmed in the Dither engine. It comes from the higher level pxp control register fields.

13.6.7.2 Dither Engine Design

The following diagram shows the detailed internal structure of the Dither Engine.

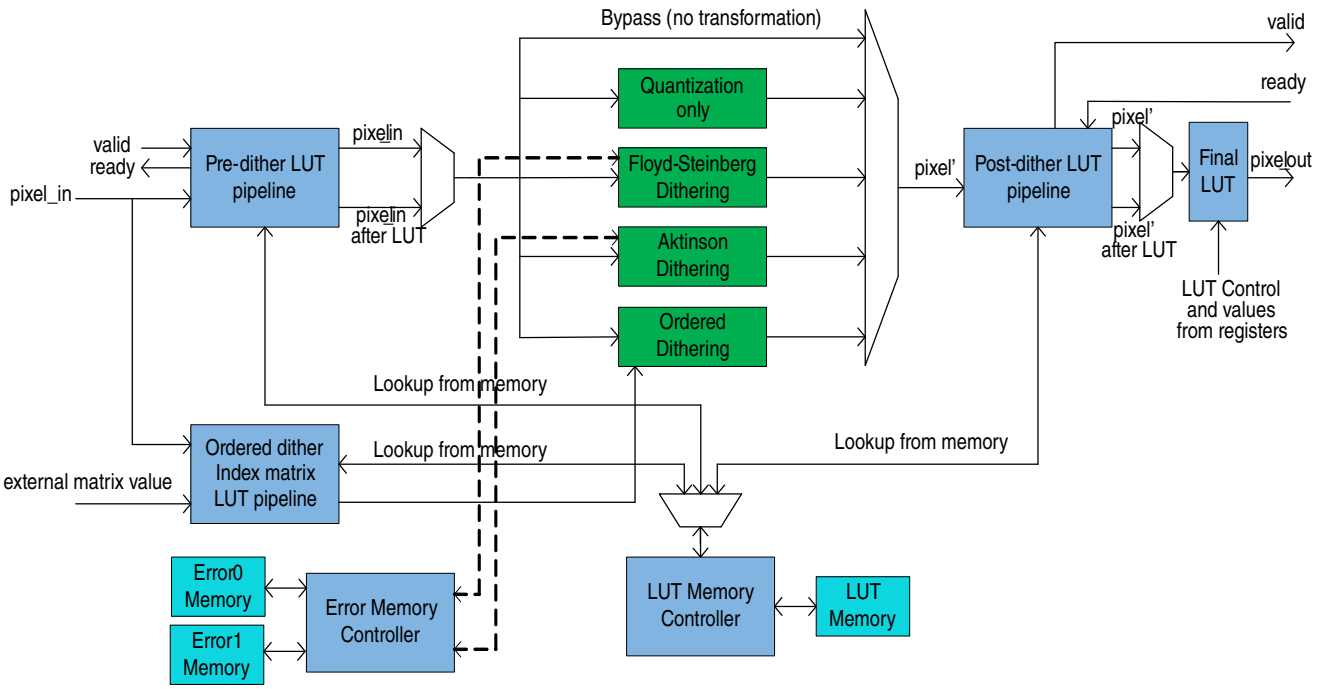


Figure 13-96. Detailed Block Diagram

Before a detailed discussion of block functionality, it is important to note that there are 3 instances of the dither block within the PXP. All three, Dither0, Dither1 and Dither2 are connected to the same Fetch and Store engines. The system works in two configurations

1. Y8 configuration:- In this configuration only Dither0 is enabled. The Fetch Engine will fetch Y8 data and feet it to Dither0. Dither0 then writes resultant data, Y4 for instance, to the Store engine.
2. RGB configuration:- In this mode all three dither engine instances are enabled. The Fetch Engine reads in RGB data from memory and gives 8 bits of the triplet to each dither engine. The resulting quantized or dithered data from each dither engine is passed to the same Store engine as in Y8 configuration and the Store Engine synchronizes, packs the data back together and writes it memory.

The Dither0 instance can operate in any of the 5 dither modes mentioned below. The Dither1 and Dither2 instances can only operate in Pass Through, Ordered Dither and Quantization Only modes. It also has the functionality to do a regsiter based final LUT at the end of dither processing. This LUT is controlled by the FINAL_LUT_ENABLE bit. Notice that in the HW_PXP_DITHER_CTRL register there are separate ENABLEx,

DITHER_MODEx and IDX_MATRIXx_SIZE controls for each engine. The NUM_QUANT_BIT and LUT_MODE control fields are common and apply to all dither engine instances.

The Dither0 instance is connected to three dedicated instances of internal memory. The error memories are only used when Floyd-Steinberg or Atkinson dithering is selected and are only connected and used by the Dither0 instance. Initialization of these memories is not required.

The Dither1 and Dither2 instances only have dedicated LUT memories connected, one RAM for each of the two instances. There are no error RAM's associated with Dither1 or Dither2.

To be included in a pxp processing flow operation, the HW_PXP_CTRL_ENABLE_DITHER or HW_PXP_CTRL2_ENABLE_DITHER bit must be set when the operation is executed. This is in addition to the HW_PXP_DITHER_CTRL.ENABLEx bits.

The process flow through the Dither Engine is detailed in the following sections.

Step1: Pre-Dither

The pixels are read into the pre-dither pipeline sub-block through the handshake with the Fetch Engine/ If there is error data or Ordered dither LUT matrix data that corresponds with the current pixel (this is mode dependent) then all these elements are fetched, synchronized and presented to the dither stage (green blocks in the above figure) simultaneously. The pre-dither, post_dither and Ordered dither matrix LUT pipeline module instances keep the data synchronized and flowing through the block. The pixel can go through a lookup table before the dither stage in the Pre-dither LUT pipeline or after in the Post-dither LU T pipeline. These blocks and the Ordered dither index pipeline block share the LUT memory through the LUT Memory Controller block. There is only one memory that is shared between these three functions and only one use can be selected for a given operation. In other words, the block can be configured to use pre-dither lookup, post-dither lookup or ordered dither mode but only one at a time. The pre or post lookup function is enabled through the HW_PXP_DITHER_CTRL.LUT_MODE field. Ordered dithering is enabled through the HW_PXP_DITHER_CTRL.DITHER_MODEx field (Again, the memory can be used for only one purpose during any given operation). The LUT function uses an 8 bit address (the input data) to lookup 8 bits of data. The LUT memory must be pre-configured with the desired lookup table data.

Step2: Dither

After the pre-dither stage, the pixel and colateral data passes through the dither stage. There are 5 programmable modes in this stage that are specified with the HW_PXP_DITHER_CTRL.DITHER_MODEx field.

1. Pass Through:- In this mode the pixel data passes through unmodified
2. Floyd-Steinberg Dithering:- This mode quantizes and dithers the pixels according to the standard Floyd-Steinberg algorithm that can be found in the literature. This algorithm employs an error dispersion technique that saves a fraction of the quantization error from each pixel and adds those partial errors to neighboring pixels. Therefore, the error memory buffers shown in the diagram are used in this mode to save the partial errors and retreives them as they are needed when processing corresponding pixels in the next row below the current row.
3. Atkinson Dithering:- This mode employs the standard Atkinson dithering algorithm to quantize and dither the pixels. The standard algorithm in the literature is implemented. Atkinson is an error dispersion algorithm similar to Floyd-Steinberg except that the partial error coeficients are different and the error is dispersed two lines below the current row instead of one line below.
4. Ordered Dithering:- This is a standard algorithm that applies an index, or bayer matrix to the pixels. Applying this matrix has the effect of changing the likelihood of a pixel being rounded up or down when being quatized. If this mode is specified, it requires use of the LUT memory so the HW_PXP_DITHER_CTRL.LUT_MODE field must be set to 0, or off. The IDX_MATRIXx_SIZE field specifies the dimensions of the index, or bayer matrix. This is used to properly index into the LUT memory to retreive the correct corresponding value to apply to the current pixel.
5. Quantization only:- This mode simply quantizes the pixel from 8 down to whatever bits per pixel are specifed in the NUM_QUANT_BIT field. No dithering algorithm is used.

The output of this step is the quantized/dithered pixel packed into the most significant bits of an 8 bit byte. The unused bits in the lower part of the byte will be set to zero. Also, any partial error information that was generated is sent to the Error Memory Controller to be stored for future processing.

Step3: Post-Dither

The post-dither block is similar to its pre-dither counterpart. The processed pixel is queued up in preparation for write to the Store Engine. Also any relevant error information is stored out to memory for the next lines of the image. If DITHER_MODEx is not equal to Ordered mode, and a lookup operation is not selected in the pre-dither step, the LUT_MODE can be set to select a lookup operation in this step using the LUT memory.

Step4: Final Lookup

This combinatorial LUT step takes bits [7:4] of the output pixel and looks up an 8 bit value from the 16 value LUT to generate the final output pixel to the Store Engine. The LUT values are programmed into the LUT through the

HW_PXP_DITHER_FINAL_LUT_DATA0 - 3 registers. This operation is enabled through the HW_PXP_DITHER_CTRL.FINAL_LUT_ENABLE register field. This LUT is only available on the Dither0 instance.

1. Y8 configuration:- In this configuration only Dither0 is enabled. The Fetch Engine will fetch Y8 data and feet it to Dither0. Dither0 then writes resultant data, Y4 for instance, to the Store engine.
2. RGB configuration:- In this mode all three dither engine instances are enabled. The Fetch Engine reads in RGB data from memory and gives 8 bits of the triplet to each dither engine. The resulting quantized or dithered data from each dither engine is packed to 24 bit RGB and be written to memory by Store Engine.

1. Pass Through:- In this mode the pixel data passes through unmodified
2. Ordered Dithering:- This is a standard algorithm that applies an index, or bayer matrix to the pixels. Applying this matrix has the effect of changing the likelihood of a pixel being rounded up or down when being quatized. If this mode is specified, it requires use of the LUT memory so the HW_PXP_DITHER_CTRL.LUT_MODE field must be set to 0, or off. The IDX_MATRIXx_SIZE field specifies the dimensions of the index, or bayer matrix. This is used to properly index into the LUT memory to retreive the correct corresponding value to apply to the current pixel.
3. Quantization only:- This mode simply quantizes the pixel from 8 down to whatever bits per pixel are specifed in the NUM_QUANT_BIT field. No dithering algorithm is used.

13.6.7.3 Pipelined Data Flow

It is possible that the fetch engine could stall when there are valid pixels in various stages of the dither block. In this case the dither pipelines will not stall but will continue to drain and move data out to the store engine. Also, if Floyd-Steinberg or Atkinson dithering is selected, at the end of each row after the last pixel has been written out to the store engine there will be one extra error that will be written out to the error memories because the pixels and the corresponding errors are offset by one. This is a natural result of the error dispersion algorithms. The dither block handles flushing out this last error automatically before processing the next row. Also, if the Store Engine stalls for a period of time and can not take any more data, flow control is implemented within the Dither block back to the Fetch Engine so that no pixel or error data is lost or becomes unsynchonized.

There are three BUSY bits in the HW_PXP_DITHER_CTRL register, one for each Dither Engine instance that software can poll to discover if the engine is busy or idle.

13.6.7.4 Initialization of Dedicated Memories

The internal memories for the Dither Engine instances, including 2 error memories and 3 LUT memories can all be initialized through the register interface. This is done by configuring the HW_PXP_MEM_CTRL register and then writing the initialization data into the HW_PXP_INIT_MEM_DATA register. This simple interface is used as follows. Write the HW_PXP_MEM_CTRL.SELECT field with a value between 0-4 corresponding to the memory to be initialized. Write the HW_PXP_MEM_CTRL.ADDR field with the starting memory address to be written. Set the HW_PXP_MEM_CTRL.START field to begin the initialization operation. Note at all these fields can be set with a single write to the register. Then write the HW_PXP_MEM_DATA register with the data to be written to the location at HW_PXP_MEM_CTRL.ADDR. Each write to the data register increments and value of address so repeated writes to this register will put the data into sequential memory locations beginning with the offset in the ADDR field. The LUT memories are 8 bits wide. When 32 bit DATA writes occur with one of those memories selected, the lower 8 bits are written. To fill these memories 256 writes to the HW_PXP_MEM_DATA register are required. The error memories are 32 bits wide so all 32 bits are data are used to fill each location and they are 2048 locations deep. Note that initialization for these memories are not necessary.

13.6.7.5 Register Configuration Interface

The HW_PXP_DITHER_CTRL register control fields to be set for each operation are included in the following table.

Signal or Group	Description
ENABLE	The hardware signal enables the dither block to function.
DITHER_MODE	Dither mode 0: passthrough, 1: Floyd-Steinberg, 2: Atkinson, 3: Ordered, 4: no dithering quantization only, 5,6,7: Reserved. There is on field for each Dither Engine instance.
NUM_QUANT_BIT	How many bits to quantize to. Valid values are 7 down to 1.
LUT_MODE	Configures a LUT operation from local memory. The options are either pre-dither, post-dither or off. This setting affects all three Dither Engine instances.
IDX_MATRIXx_SIZE	These fields, one per instance, specifies the dimension (the matrix is square) of the Index Matrix used in Ordered dither mode.
FINAL_LUT_ENABLE	This field is only applicble to instance 0 of the three Dither Engines and enables/disables the final register based LUT of the output pixel just before outputing it to the Store Engine.

13.6.8 Waveform Engines

13.6.8.1 Overview

The WFE (Wavefrom Engine) is a very flexible, programable engine that can be used to implement various algorithms to transform the incoming data to the output. It is always used in connection with the Fetch Engine on the input side and a Store Engine on the output. These three sub-blocks work as a unit. There is an optional ALU block between the WFE and Store engines that can be used to make further modifications to the WFE output before storing the output. See the figure below.

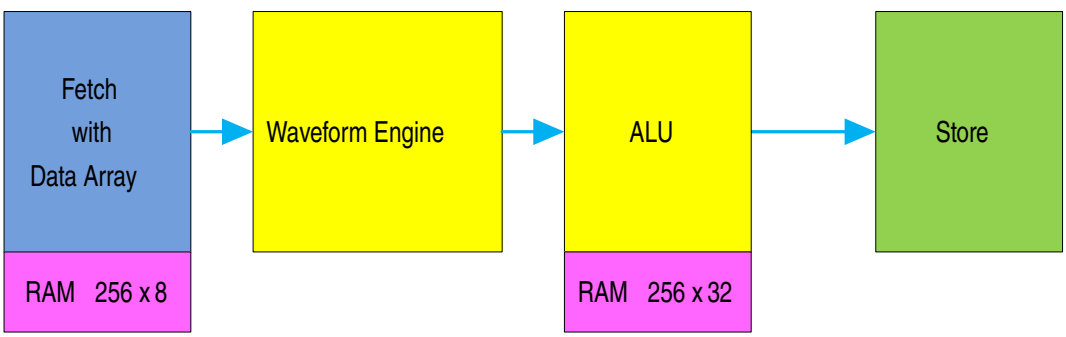


Figure 13-97. Block Diagram

The WFE can process one pixel and associated data per clock.

The Fetch Engine fetches and formats the input pixel data streams according to the requirements and programming of the WFE. The WFE takes in the data from the Fetch Engine through a standard valid/ready signalling interface.

Inside the WFE the data passes through a 3 stage pipeline. Each stage consists of various programmable logic elements including muxes, lookup tables, comparitors, ALU's and registers. The output information calculated in the WFE is output to the Store Engine also using a standard valid/ready signalling interface.

The Store Engine reads the data from the WFE in the programmed format reformats the data according to working buffer pixel formats and stores the frame and associated data to memory.

13.6.8.2 Functionality

There are two WFE instances arranged serially in the PXP standard flow, WFE-A and WFE-B.

In the standard usecase WFE-A is used for pixel collision detection and also to compute grayscale color transition information used later in the REGL/D algorithm.

In the standard usecase the WFE-B engine takes in pixel information and the results of WFE-A processing and implements the REAGL/D algorithm on the data. The output results in all the information necessary for the EPDC block to update the E-INK panel.

The definitions, functionality and programming of the logical elements within the WFE is Freescale, Inc. prorietary information. The register programming necessary for any given function required will be supplied by Freescale to the customer.

The WFE block can be independently held in reset by setting the HW_PXP_WFE_x_CTRL.SW_RESET bit to 1. This bit should only be set when the HW_PXP_WFE_x_CTRL.ENABLE bit is 0.

13.6.9 PXP Store Engine Block Description

13.6.9.1 Overview

The essential function of the Store Engine is to store input data from the input sub-block to memory or directly output to next sub-block. The input source is 32 bits per channel per valid clock. There are 4 instances of the Store Engine in the PXP. Please refer to the PXP block diagram for details. The main supported features are:

- Two DMA controllers thus it can work on two channel data storage at the output at the same time. If there is only one stream data input, store engine should use channel 0 but not channel1.
- Accepts as input up to 8 bytes data (two channels, 32 bits each) and 8 bits flag input. After shift operation it can store up to 64bpp per valid clock to memory.
- Data fill function for initialization. It can fill a fixed 32 bit data value to a specified memory buffer.
- Handshake mode with a downstream connected data Fetch Engine. It supports 1 scanline handshake with and 1 line, 3 lines and 5 lines array handshake modes. In handshake mode, the Store Engine signals to the Fetch Engine when data is avalible for processing thus eliminating the need for software intervention through interrupt service routines.

- Bypass mode which directly outputs the data to the downstream connected Fetch Engine after shift operation. This is done through a valid/ready data interface.
- Support for 8x8 and 16x16 block mode and scanline mode.
- Support for 8, 16 and 32 bpp input data format.

13.6.9.2 Top-level architecture

The PXP consist of several pipelined blocks that perform the video source frame scaling, color space conversion, alpha-blending/color key algorithm, secondary CSC, pixel correction , input and/or output rotation, dithering and waveform processing. It is also capable of fetching data from and storing data to memory.

The legacy blocks operate within the requirements of the legacy PXP architecture, and perform operations on either 8x8 or 16x16 pixel blocks in the representative source buffers. The legacy pipeline operate within the context of two iteration counters that iterate through the appropriate grid of input blocks to produce the rotated output grid blocks in scan-line order. The dither blocks and the waveform processing engines (WFE) will operate in a scan line format based on the active size. The input fetch and store engines can work on either on a block by block basis or in the scan line format.

Figure 13-71 shows the high-level architecture of the scaling, color space conversion, blending, pixel correction , rotation engines, dithering, waveform processing, histogram along with four sets of fetch and store engines. The Alpha Surface Engine fetches one RGB graphics plane alpha surface (AS). The scaling engine fetches a single processed surface (PS), which can be blended with the AS surface. The scaling engine also supports an alpha channel for the PS image. Although the legacy PXP processes NxN pixel macro blocks, each of the AS or PS surfaces can have any pixel alignment within the output buffer. There are no restrictions and any pixel coordinates within the output buffer are valid. The upper left origin of the output buffer is defined as pixel 0,0. The upper left and lower right coordinates for each of the AS and PS are inclusive within the output buffer.

Figure 13-72 represents a sample output buffer configuration with both an AS and PS included. The alignment of each AS and PS within the output buffer can be at any arbitrary pixel locations. For example, the PS has an upper left coordinate (ULC) of 2,2 and a lower right coordinate (LRC) at pixel 13,13. The maximum value for the ULC and LRC for each of the AS and PS is bounded by the LRC of the output buffer, 15,15 for this example.

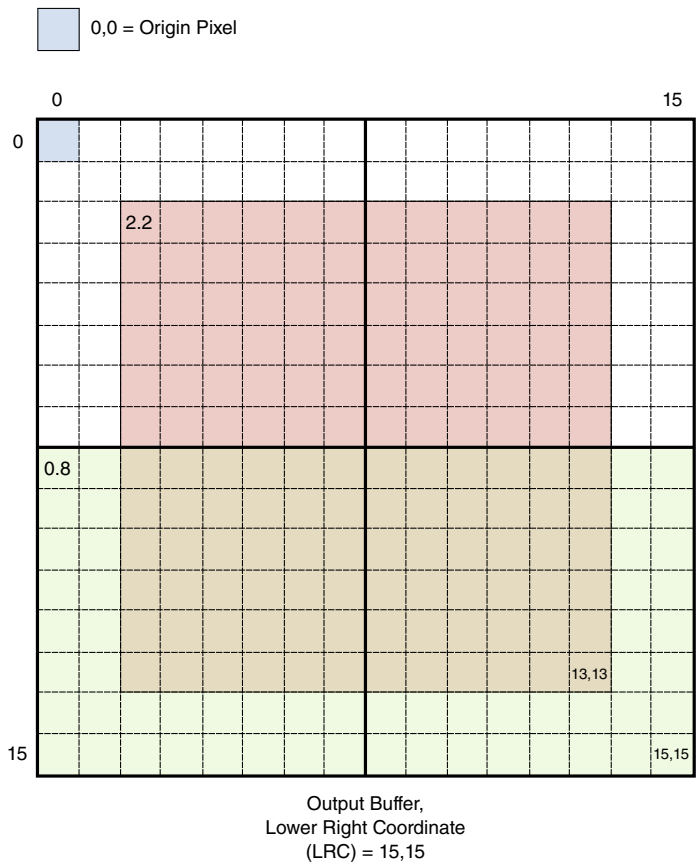


Figure 13-98. Sample output buffer configuration

The AS engine supports RGB pixel formats, and the PS engine supports ARGB, RGB, YUV, and YCbCr pixel formats. The CSC1 can be used to convert to RGB pixel formats so that the PS surface can be blended with the AS surfaces in the compositing engine in the RGB color space. There are two rotate engines in the PXP pipeline. Rotation can occur after image composition, at the output of the PS engine, or both. In the first scenario, all the data produced by the AS and PS engines is rotated. When the rotation module is programmed to rotate only PS images, the AS is not rotated, and AS pixels are combined with rotated PS surfaces. The Rotate engine at the input can also be used with the input fetch engine. For this, the fetch engine needs to be programmed in the blok mode to fetch a block of size 8x8. The CSC2 unit can convert to any RGB, YUV, or YCbCr color spaces for final output. Pixels can be corrected using a programmable LUT resource to achieve any desired pixels effects. For detailed information, please refer to Output Modes to RGBW4444CFA. The dither engine supports 3 dithering modes in addition to the standard quantization and pass-through modes. The dithering algorithms supported are: Ordered, Floyd-Steinberg and Atkinson. The WFE is a programmable engine that can be programed as required to process each pixel and pixel meta data. The Histogram sub-block collects statistics about the pixel data passing through it that is

useful to the EPDC in waveform selection and displaying the frame. The Histogram also contains mask and comparison functionality that can be used for collision detection and reporting of the collided area within an update area.

The PXP also supports two parallel image processing paths. The legacy flow can operate in parallel with the input fetch engine or the dithering or waveform processing engines. Please refer to HW_PXP_CTRL and HW_PXP_CTRL2 for details on how to enable each of the individual data flows. In addition to this, the CSC, Rotation 1 and LUT engines can be used as either part of the legacy flow or as a part of the second parallel data flow. These blocks can work in the scan line format or in block format. There are two separate blending engines for each flow.

13.6.9.3 Store Engine Design

13.6.9.3.1 Input Data Source

1. **Fixed Data Input**

   The Store Engine supports a fill function which writes a fixed value to a specific set of memory locations. The data received from the input is ignored. This function is usually used to initialize a memory buffer. The address of the buffer to fill is set in the HW_PXP_x_STORE_FILL_DATA_CH0 register. Also, this mode can only work through channel 0 and not channel 1.

   In this mode HW_PXP_INPUT_STORE_CTRL_CH0.FILL_DATA_EN=1.The fixed value is set in FILL_DATA_CH0 register. How many bits are valid from HW_PXP_INPUT_STORE_FILL_DATA_CH0 register is decided by HW_PXP_x_STORE_SHIFT_CTRL_CH0.OUTPUT_ACTIVE_BPP.

2. **Normal Data Input**

   Store Engine supports two 32 bits wide data channels and 8 bits flag input. If there is only one channel of data input, the channel 0 must be used not channel 1. In this case, HW_PXP_INPUT_STORE_CTRL_CH0.COMBINE_2CHANNEL=0.

   If both channels and 8 bits flag input contain valid data, then the COMBINE_2CHANNEL bit should be set to 1. The Store Engine will then combine the two channels of data into 64 bits of data send them with the 8 bits of flags to the shift module. After shift, the Store Engine will choose the active bits from the 64 bit

output according to HW_PXP_x_STORE_SHIFT_CTRL_CHx.OUTPUT_ACTIVE_BPP and then pack them in preparation for output.

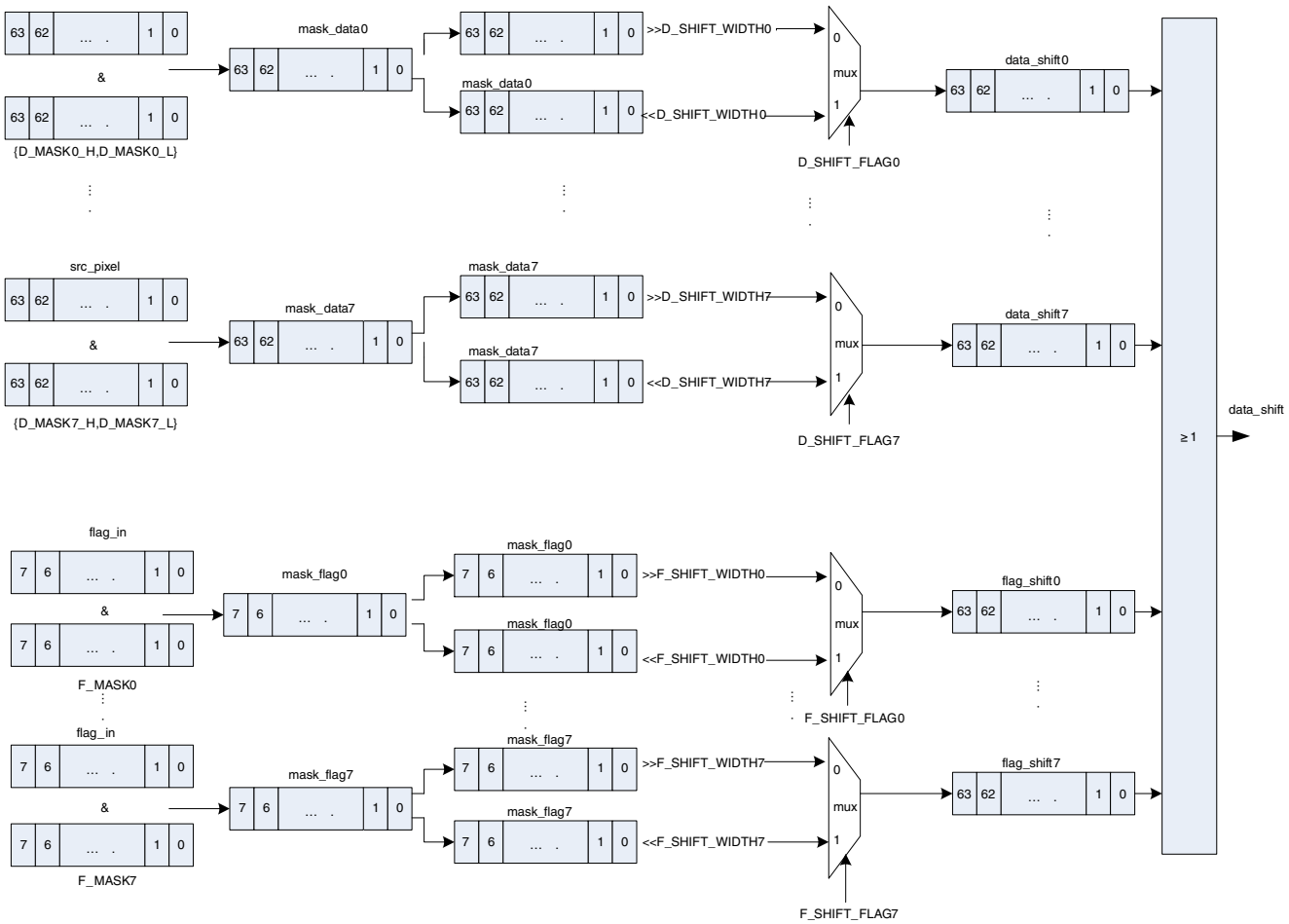
13.6.9.3.2 Store data shift operation

The data shift operation supports up to 8 data fields and 8 flag components to be shifted to any position within the 64 bit word. The data fields can be any width. This function is a way to reorder or reformat the component parts of the 64 bit word output. As data flows through the Store Engine each data word is processed the same way according to register configuration.

The following registers are used in shift operation. The pairs of {HW_PXP_x_STORE_D_MASKx_H_CHx, HW_PXP_x_STORE_D_MASKx_L_CHx} make up the 64 bits of mask for each MASKx component. There are a set of masks for each channel in each instance. The D_SHIFT_FLAG0 ~ D_SHIFT_FLAG7 fields in the HW_PXP_x_STORE_D/F_SHIFT_L/D_CHx registers specify which direction to shift (0: right shift 1: left shift) each data component0 ~ component7 after the mask is applied. The D_SHIFT_WIDTH0 through D_SHIFT_WIDTH7 fields specify how many bits to right or left shift each data component 0 through7 after the mask is applied.

The shift function works the same for the 8 bits of flag data. There are analogous “F_SHIFT” regsiters that contain the F_MASKx, F_SHIFT_FLAGx and F_SHIFT_WIDTHx register fields for swizzling the input flags.

The figure below shows how the shift function works.



13.6.9.3.3 Data Packing

After data shift, the 64 bits of data can be divided to two 32bit words depending on the value of the HW_PXP_x_STORE_CTRL_CHx.PACK_IN_SEL bits. If HW_PXP_x_STORE_CTRL_CH0.PACK_IN_SEL = 0, it will select active bits from all 64 bits of data. If equal to 1, it will select active bits from the lower 32 bits data.

For channel 1, if HW_PXP_x_STORE_CTRL_CH1.PACK_IN_SEL = 0, it will also select active bits from all 64 bits data and if equal to 1, it will select active bits from the higher 32 bits data.

Store Engine supports 8bpp, 16bpp, 32bpp and 64bpp active bits per pixel according to OUTPUT_ACTIVE_BPP register setting. When either channel's PACK_IN_SEL = 1 only 8bpp, 16bpp, 32bpp can be selected.

After shift and data output selection, the packing module will pack the data into a 64 bit string and store to a FIFO before writing out to memory.

Packing style according to active bpp is showed below.

OUTPUT_ACTIVE_BPP	Active bpp from 64 shift output data	Packed data
0	8 bits	{pix7, pix6, pix5, pix4, pix3, pix2, pix1, pix0}
1	16 bits	{pix3, pix2, pix1, pix0}
2	32 bits	{pix1, pix0}
3	64 bits	{pix0}

13.6.9.3.4 Data Store Format

- **Scanline mode**

  When HW_PXP_x_STORE_CTRL_CHx.BLOCK_EN=0, the Store Engine will store data the line by line in scanline mode. The AXI burst length is set by the HW_PXP_x_STORE_CTRL_CHx.WR_NUM_BYTES register field. In Scanline mode, it can support 8, 16, 32 and 64 bytes in a burst.

- **Block mode**

  When HW_PXP_x_STORE_CTRL_CHx.BLOCK_EN=1, the Store Engine support 8x8 and 16x16 block mode according to HW_PXP_x_STORE_CTRL_CHx.BLOCK_16 register bit. When BLOCK_16=1, the block size is 16*16 while if BLOCK_16=0 the size is 8*8 blocks. When BLOCK_EN=1, then HW_PXP_x_STORE_SHIFT_CTRL_CHx.OUTPUT_ACTIVE_BPP cannot be 3, that is, block mode can not work on a 64bpp data stream.

  When the Store Engine is configured block mode, the upstream Fetch Engine should be configured in block mode and with the same block size as well.

  When the total width or total height of image is no divisible by 8 or 16, the upstream Fetch Engine will fetch extra pixels or lines to fill out the final partial blocks. For example, if the size is 23*23, and block mode is 8*8, Fetch Engine will fetch 24*24 data pixels and output to store engine. The Store Engine has a crop function for this. That is, the extra pixels or lines will not be stored to memory by the Store Engine. This is accomplished by the hardware using the AXI bus byte enables.

  In block mode, the number of AXI bus write bytes is internally calculated from the OUTPUT_ACTIVE_BPP and BLOCK_16 (block size) register fields rather than the WR_NUM_BYTES field.

One limitation exists with block mode configuration. No YUV422 output format is not supported. So HW_PXP_x_STORE_SHIFT_CTRL_CHx.OUT_YUV422_1P_EN and OUT_YUV422_2P_EN must be disabled.

13.6.9.3.5 Output Format Modes

- **Normal mode**

  If HW_PXP_x_STORE_CTRL_CHx.STORE_BYPASS_EN = 0,

  HW_PXP_x_STORE_CTRL_CHx.STORE_MEMORY_EN=1 and

  HW_PXP_x_STORE_CTRL_CHx.HANDSHAKE_EN=0, the Store Engine is working in normal mode.

  In normal mode, the Store Engine will store the input data to memory frame by frame. The store frame start address is set in HW_PXP_x_STORE_ADDR_x_CHx. Two address registers exist for each channel. The second one is used for YUV422 2 plane mode.

  The Store Engine will stop storing data when the frame of data is finished. Another pxp_start signal will trigger the next frame operation. The frame size is specified by the HW_PXP_x_STORE_SIZE_CHx registers. The values should be set up for each channel in use. If both channels are being used, the height and width values in the HW_PXP_x_STORE_SIZE_CH0 and HW_PXP_x_STORE_SIZE_CH1 registers may be different but the total number of pixels in the frame must be the same. The only exception to this is when both channels are being used and the Store Engine is outputing data in handshake mode. In handshake mode the width and height must be set the same in both channel size registers.

  If the data shift mode is set to YUV422 single plane or 2 plane, two pixels, 64 bits, are read in and 32 bits are written. If the YUV data is coming from the upstream Fetch Engine it will be in YUV444 format as {8'h0, Y0, U0, V0, 8'h0, Y1, U1, V1}. Even though the YUV422 mode is set, the shift function must be set to shift the components into their proper position. For instance, for 1p mode, the shift MASK, FLAG and WIDTH parameters need to be set up on the 64 bits of pixel data to format each 32 bits (2 pixels) to be written out as {Y0, U0, Y1, V0}. Likewise in 2p mode, the ch0 data would be shifted to yield {Y0, Y1} with ch1 as {U0, Y0} for the first two pixels.

- **Bypass mode**

If STORE_BYPASS_EN=1, STORE_MEMORY_EN=0, The Store Engine will output the input data, after the shift function, directly to the downstream Fetch Engine instead of storing to memory. The AXI bus is inactive in this case. The data is output using the valid/ready external interface to the downstream connected Fetch Engine.

In bypass mode, if HW_PXP_x_STORE_CTRL_CH0.CH_EN=1, it will output the lower 32 bits of data. If HW_PXP_x_STORE_CTRL_CH1.CH_EN=1, it will output the high 32 bits of data.

- **Dual mode**

  If STORE_BYPASS_EN = 1 and HW_PXP_x_STORE_CTRL_CHx.STORE_MEMORY_EN = 1, the Store Engine is configured in Dual mode. That is, the Store Engine supports bypassing the data and also storing it to memory at the same time.

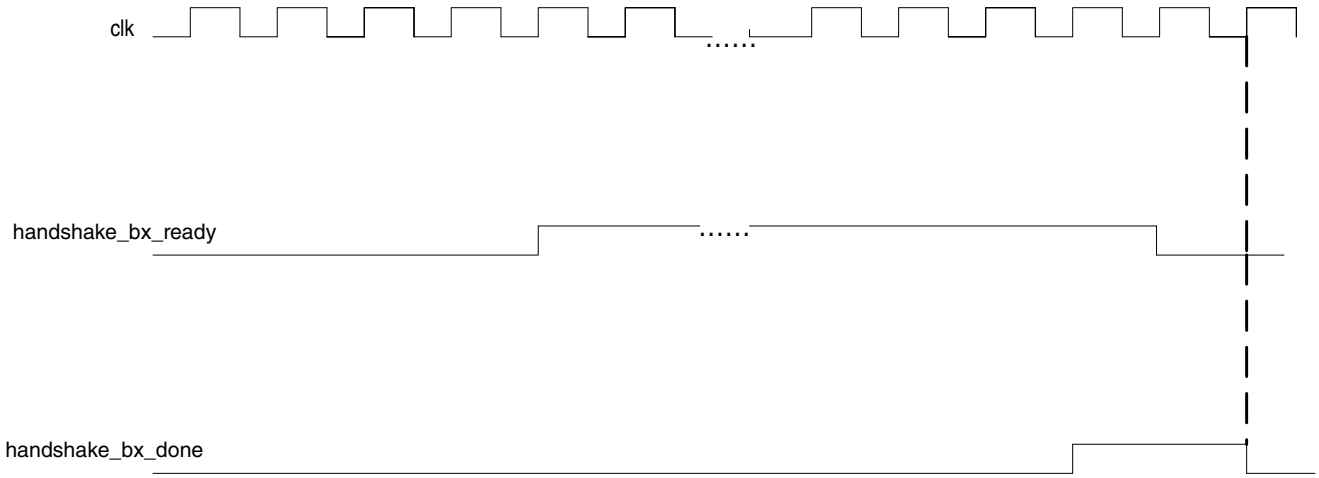
- **Handshake mode**

  The Store Engine supports a 1 scanline handshake configuration and 1 line, 3 lines and 5 lines array handshake configurations with next Fetch Engine. The 1 scanline mode is used for Store Engines connected to the Input_Fetch and Dither_Fetch instances. The array handshake modes are used in cases where the Store Engine is connected to the WFE_A_Fetch or WFE_B_Fetch instances. When HW_PXP_x_STORE_CTRL_CHx.HANDSHAKE_EN = 1, ARRAY_EN =1 and STORE_MEMORY_EN = 1, the Store Engine is in handshake mode(1 scanline handshake configuration is identical to 1 line array handshake configuration so the following sections will only discuss array handshake modes).

  The array handshake is used in cases where the Fetch Engine is in array mode. One line handshake config indicates that the Fetch Engine is working on 1x1 array mode. The 3 line handshake indicates the Fetch Engine work is in 3x3 array while 5 line handshake indicates 5x5 array mode.

  To accomplish the buffer sharing, Store Engine and the attached Fetch Engine will maintain buffer status using a pair of handshake signals. When a buffer is filled by the Store Engine, it will assert the handshake_bx_ready (where x is 0 or 1) signal to indicate to the fetch engine that the buffer has valid data. The prefetch engine will then release the buffer by asserting the handshake_bx_done signal.

  The basic protocol and timing is shown in the diagram below:



Unlike scanline handshake, there is only 1 buffer in memory. The buffer size is (array_line + 1).

- 1 line array handshake

  When HW_PXP_x_STORE_CTRL_CHx.ARRAY_LINE_NUM = 0, the configuration is 1 line handshake mode. The buffer size is 2 * OUT_PITCH (2 lines). The first ready occurs when the Store Engine finishes writing 1 line.

- 3 line array handshake

  When ARRAY_LINE_NUM = 1, the configuration is 3 line handshake mode. The buffer size is 4 * OUT_PITCH. The first ready occurs when the Store Engine finishes writing 2 lines.

- 5 line array handshake

  When ARRAY_LINE_NUM=2, the configuration is 5 line handshake mode. The buffer size is 6*OUT_PITCH. The first ready occurs when store engine finishes writing 3 lines.

13.6.9.3.6 Limitations

When both channels are active, they should be configured to work in the same mode. For example, if channel 0 is configured for bypass mode then channel 1 must also be configured the same way. In the case of the input Fetch and Store instances, see overall block diagram, if the path between them includes the LUT processing block then the Fetch and Store Engines must be in block mode. The LUT will not function correctly in scanline mode.

13.6.10 PXP Fetch Engine Block Description

13.6.10.1 Overview

The main function of Fetch Engine is to fetch data from memory or directly from the input bypass channel and output to another downstream module as 32 bits per pixel per valid clock.

The Fetch Engine has two channel DMA controllers and thus can fetch two memory data buffers at the same time. The Fetch Engine supports an expansion and shift function on the fetched data and then output as 32 bits per pixel according to different data format. It supports 8x8 or 16x16 block mode and scanline mode. The block also supports rotation function with rotation module when in block mode. It can support vertical flip, horizontal flip, 90/180/270 degree rotation angle.Using the expansion function, the user can read in rgb555, rgb565, rgb444, argb1555, argb4444, yuv422_2p or yuv422_1p input data format and convert to argb8888, rgb888 or yuv444 format at the output.The Fetch Engine can interface with a connected Store Engine block directly in one of two ways, handshake mode and bypass mode as described further below. In handshake mode 1 line, 8 line and 16 line configurations are possible. In bypass mode memory accesses are bypassed altogether and data passes directly from a Store Engine to its connected Fetch.Input formats of 8bpp, 16bpp, 32bpp, 64bpp are supported.

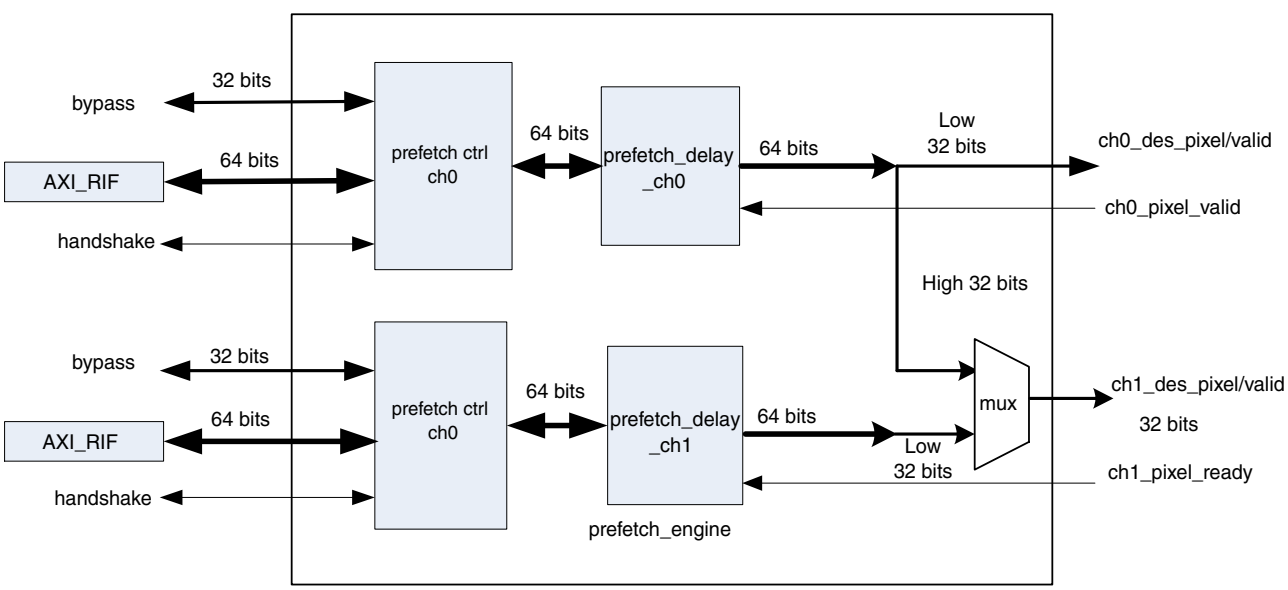


Figure 13-99. High level block diagram

The following diagram shows the internal structure of the Fetch_ctrl engine.

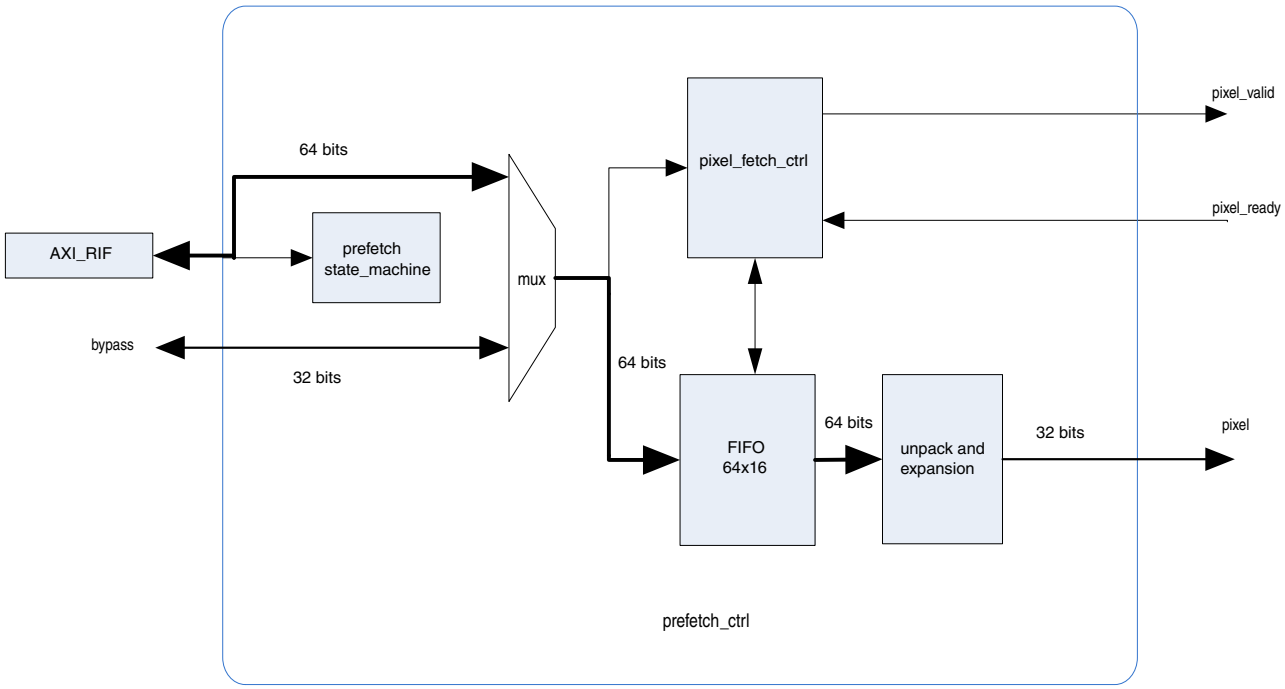


Figure 13-100. Fetch Control block diagram

13.6.10.2 Fetch Data formats

The Fetch Engine supports input 8bpp, 16bpp, 32bpp, 64bpp data format according to INPUT_ACTIVE_BPP register setting in the HW_PXP_x_FETCH_SHIFT_CTRL_CHx registers. (“x” is a wildcard that is replaced with the fetch module instance or the channel number. This notation is used because there are several instances with unique names and channels 0 and 1.)

When input data is 64bpp, since output data bus is 32 bits per channel, the Fetch Engine will need to be configured to use both output channels (32 bits each) to transmit the 64 bits of data at the output. Thus, when input data is 64bpp, the Fetch Engine supports only channel 0 data input but not channel 1 at the same time.

The Fetch Engine supports an expansion function. When you set EXPAND_EN=1, the engine will change the input format to a unified YUV444 or ARGB888/RGB888 format. Otherwise it will remain in the same format and output to an external module. Note the expand funtion is not available if the input data is YUV422 format, 1 plane or 2 plane data fetching is supported. When EXPAND_FORMAT=7, pefetch_engine will fetch YUV422 format data from two plane. One is for Y, the other is for UV. The start address of these two buffer planes are set by INPUT_BASE_ADDR_0_CHx (Y buffer) and

INPUT_BASE_ADDR_1_CHx (UV buffer) fields in the HW_PXP_x_FETCH_ADDR_x_CHx registers for that Fetch instance. Also, if input format is YUV422_2P, the EXPAND_EN must always be asserted. This combination is only used to transform YUV422_2P to YUV444 format.

The data format in memory and output is shown in table below. For input formats that read multiple pixels within the 64 bit input word, there will be multiple output cycles for each input read to expand and output all the pixels in the input word.

X means this parameter is a “don’t care”.

INPUT_ACTIVE_BPP	Input data valid bits(bpp)	Data store in memory	EXPAND_EN	EXPAND_FORMAT	After expanding(32 bit word)	Data output format in 32 bit bus
0	8 bits	{pix7, pix6, pix5, pix4, pix3, pix2, pix1, pix0}	0	Any value except 7	NA	{24’h0, pix0}
1	16bits	{pix3, pix2, pix1, pix0}	0	Any value except 7	NA	{16’h0,pix0}
2	32bits	{pix1, pix0}	0	Any value except 7	NA	{pix0}
3	64bits	{ pix0}	X	X	NA	Channel 0 bus output { pix0[31:0]} , Channel 1 bus output { pix0[63:32]}
1	16bits	{pix3, pix2, pix1, pix0}	1	0 (rgb565)	{8’h0,rgb888}	{pix0}
1	16bits	{pix3, pix2, pix1, pix0}	1	1 (rgb555)	{8’h0,rgb888}	{pix0}
1	16bits	{pix3, pix2, pix1, pix0}	1	2 (argb1555)	{8{a}},rgb888 }	{pix0}
1	16bits	{pix3, pix2, pix1, pix0}	1	3 (rgb444)	{8’h0,rgb888}	{pix0}
1	16bits	{pix3, pix2, pix1, pix0}	1	4 (argb4444)	{ {2{a}}, rgb888 }	{pix0}
1	16bits	{pix3, pix2, pix1, pix0}	1	5 (yuyv/yvyu)	{8’h0,yuv444} /{8’h0,yvu444}	{pix0}
1	16bits	{pix3, pix2, pix1, pix0}	1	6 (uyvy/vyuy)	{8’h0,yuv444} /{8’h0,yvu444}	{pix0}
x	x	Address0 {Y7,Y6,Y5,Y4,Y3,Y2,Y1,Y0} Address1 {U6V6U4V4U2V2U0V0}	1	7 (yuv422_2p)	{8’h0,yuv444}	{pix0}

Note that in 64 bpp mode, the expand and shift functions are not supported.

13.6.10.3 Data Fetching Format

- **Scanline mode**

  When HW_PXP_x_FETCH_CTRL_CHx.BLOCK_EN=0, the Fetch Engine will fetch data line by line using scanline mode. The burst length can be set by RD_NUM_BYTES register bits. In Scanline mode, it can support 8, 16, 32 and 64 bytes for a burst. The values RD_NUM_BYTES = 0,1,2,3 indicates 8,16,32,64 bytes per burst.

- **Block mode**

  When BLOCK_EN=1, Fetch Engine support 8*8 and 16*16 block mode according to BLOCK_16 register bit. When BLOCK_16=1 the block size is 16*16. When BLOCK_16=0, it is 8*8 block mode. When BLOCK_EN=1, INPUT_ACTIVE_BPP cannot be 3, that is, block mode cannot work on a 64bpp data stream.

  When the total width or total height of image is not divisible by 8 or 16 (according to block size) it will fetch extra pixels or lines to compensate to a complete block. For example, if the size is 23*23, and block mode is 8*8, Fetch Engine will fetch 24*24 data pixels and output to store engine. Store engine has a crop function for this.

  In block mode, AXI bus read number bytes is decided by input format and block size and overrides the RD_NUM_BYTES register field.

13.6.10.4 Fetch Data Shift Function

Fetch Engine supports a shift operation that will swizzle register defined data fields in the incoming data words. The post-shifted data is then output to the proper channel. Up to 4 component shift operations are supported.

The shift function can be disabled by asserting the HW_PXP_x_FETCH_SHIFT_CTRL_CHx.SHIFT_BYPASS bit. This function is bypassed by default.

The shift is configured with the HW_PXP_x_FETCH_SHIFT_OFFSET_CHx.OFFSETx and HW_PXP_x_FETCH_SHIFT_WIDTH_CHx.WIDTHx register fields. Each channel is controlled separately. The OFFSETx fields specify the starting position of the component fields in the incoming data word. The WIDTHx register fields specify the width of that component. There is register support for 4 component fields for each

channel. The specified fields are shifted in order and packed to the least significant part of the data word. The unspecified bits in the input word are discarded. If there are any unspecified bits in the most significant part of the output word they are zeroed out.

Not all fields must be specified. For example if field number three is not used then the WIDTH3 field should be left with a value of 0.

A few examples are shown in the figure below. In the example in figure, left, we are swizzling the data format of YVU444 to YUV444. The register fields should be set as follows:

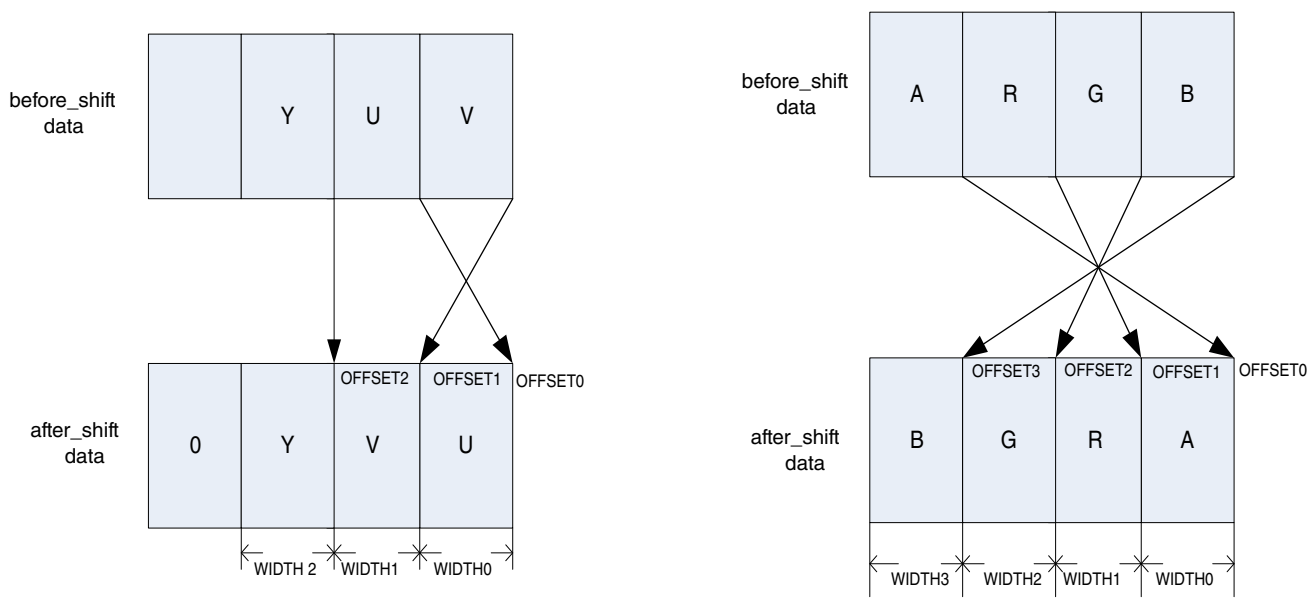
OFFSET0=8, WIDTH0=8, OFFSET1=0, WIDTH1=8, OFFSET2=16, WIDTH2=8, OFFSET3=0, WIDTH3=0

The result switches the position of the 8 but U and V components. The Y component remains in the same place and the high byte remains unchanged.

In the example in figure, right, we are swizzling the data format of ARGB888 to BGRA888. The register settings are:

OFFSET0=24, WIDTH0=8, OFFSET1=16, WIDTH1=8, OFFSET2=8, WIDTH2=8, OFFSET3=0, WIDTH3=8

All the components are swizzled and packed to the right in order from component 0 to 3.



13.6.10.5 Fetch Interface Modes

- **Normal mode**

If HW_PXP_x_FETCH_CTRL_CHx.BYPASS_PIXEL_EN=0 and HW_PXP_x_FETCH_CTRL_CHx.HANDSHAKE_EN=0, the Fetch Engine is working in the “normal” or default mode.

In normal mode, the Fetch engine will fetch data from output memory usually DDR frame by frame. The fetch frame start address is set in registers HW_PXP_DITHER_FETCH_ADDR_x_CH0.INPUT_BASE_ADDRx depending on which channel and plane format is used.

The Fetch Engine will stop to fetch data when one frame data is finished. Another pxp_start signal will trigger another frame fetching work.

- **Bypass mode**

  In the Fetch Engine, if HW_PXP_x_FETCH_CTRL_CHx.BYPASS_PIXEL_EN=1 it will fetch data directly from the input pixel of the upstream Store Engine and not from memory. In this case the AXI bus is inactive.

  In bypass mode, the pixel input will be 32 bits per pixel per valid cycle. The bypassed data will be stored to an internal FIFO first and then output to the next downstream block after the shift operation takes place.

  When BYPASS_EN=1, HANDSHAKE_EN should be equal to 0. Bypass mode and Handshake mode can not be set at the same time.

- **Handshake mode**

  If HW_PXP_x_FETCH_CTRL_CHx.HANDSHAKE_EN=1, the Fetch Engine is set to function in handshake mode.

  The Fetch Engine supports 1 line/ 8line/16lines handshake with the Store Engine module according to HW_PXP_x_FETCH_CTRL_CHx.HANDSHAKE_SCAN_LINE_NUM register bits.

  HANDSHAKE_SCAN_LINE_NUM=0, 1, 2 indicates 1 line, 8 lines, 16 lines handshake mode separately.

  One line handshake can only work in scanline mode, that is, under the condition of BLOCK_EN = 0. Eight lines handshake can only work on block 8x8 mode -- BLOCK_EN=1, BLOCK_16=0.

  Sixteen lines handshake configuration only works in block16x16 mode -- BLOCK_EN=1, BLOCK_16=1.

This is accomplished by creating two buffers in memory, where each correspond to 1/8/16-lines of the frame buffer. The buffers must be consecutive and allocated as a single block of data.

The storage required can be calculated for 1 line handshake as:

Storage = 1(line) * 2(buffer) * width * bytes/pixel;

For 8 lines handshake config the requirements are:

Storage = 8(line) * 2(buffer) * width * bytes/pixel;

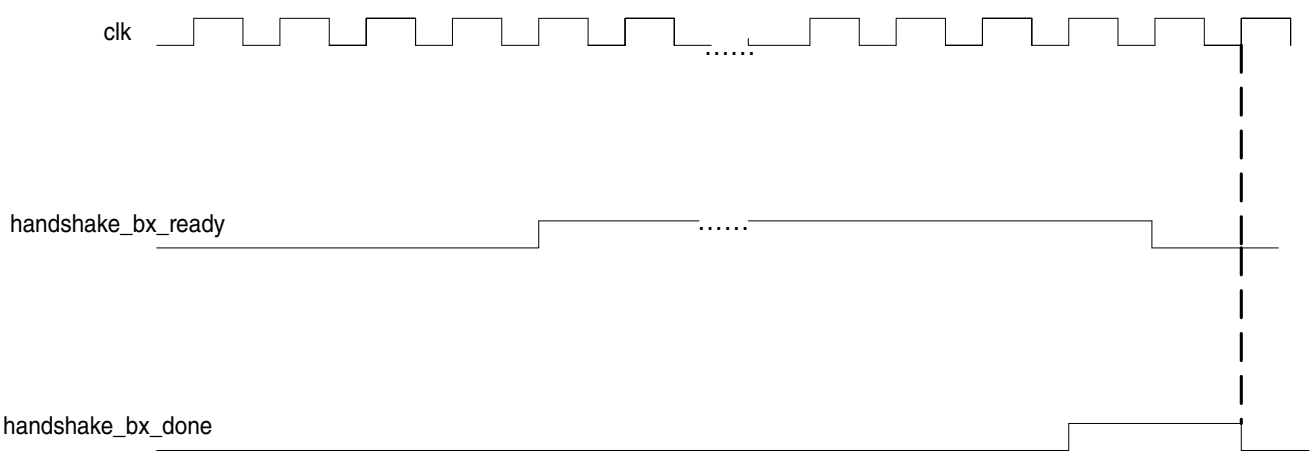
For 16 lines handshake config:

Storage =16(line) * 2(buffer) * width * bytes/pixel;

To accomplish the buffer sharing, the Fetch Engine and store engine will maintain buffer status using a pair of handshake signals. When a buffer is filled by store engine, it will assert the handshake_bx_ready (where x is 0 or 1) signal to indicate to the Fetch Engine that the buffer has valid data. After reading it the Fetch Engine will then release the buffer by asserting the handshake_bx_done signal. The Store Engine is then free to refill it and assert the handshake_bx_ready again.

The 1 scanline mode is used when a Fetch Engine is connected to a Store Engine. An example would be the Input_Store communicating with the Dither_Fetch. The 8 and 16 scanline modes are used when the Output Buffer is communicating with the Dither_Fetch.

The basic protocol is shown in the diagram below:



13.6.11 Histogram

The histogram block collects statistics about the pixel -5data passing through it. Its primary function is to compare the value against programmed values representing the allowed values in different bpp (bits per pixel) modes or bins. A flag value of 1 is recorded if all the pixel values were a match for one of the values programmed for that bit per pixel mode or bin. If there is a mismatch for at least one pixel then the update is not completely contained within that bin and the output flag is 0. There are 5 bits reported. One for each of 1-5 bpp. The results for each bpp bin are always reported. These results are a necessary input to the EPDC for selecting the proper waveform for update to the panel. The pixel value and the compare values are always 6 bits wide although the max effective bpp is 5.

There is also a mask function that can be used to filter out update pixels and exclude them from histogram calculations. It's primary use-case is to collect and report collisions of the current update being processed through the WFE-A and report to software how many pixels collided with current live updates, if any, and the coordinates of the minimum rectangle within the update that contains all collided pixels.

There are two identical instances of the histogram block in the system. The first one can be programmed through a system mux to take data from the outbuf sub-block from the legacy flow, WFE-A or WFE-B. The second instance can take data only from WFE-A or WFE-B.

13.6.11.1 Basic Operation

The basic controls for each histogram engine are contained within their HW_PXP_HIST_x_CTRL and HW_PXP_HIST_x_MASK registers. We will use histogram A registers for explanation purposes in the description below.

The HW_PXP_HIST_A_CTRL.ENABLE bit turns on the histogram block to process pixels. It must be turned on and off manually. The HW_PXP_HIST_A_CTRL.CLEAR bit must be set to clear the histogram results. The bit is self-clearing. The input data (pixel) to the block is 72 bits wide. The comparison is done on 6 bits of pixel data. Any contiguous 6 bits in the 72 bit string can be chosen for comparison using the HW_PXP_HIST_A_CTRL. PIXEL_OFFSET and HW_PXP_HIST_A_CTRL.PIXEL_WIDTH fields. The offset specifies the starting bit position in the string and the width field specifies the width which should not be wider than 6 bits in the current implementation. The mode values for comparison are contained

in the HW_PXP_HISTx_PARAMx register fields. Even though these fields are 6 bits wide, 5 bits is the effective width as the histogram can check for 32 unique pixel values in the largest mode, HIST32.

The result of which bpp modes were a match for all pixels since the output was cleared are continuously updated in the HW_PXP_HIST_A_CTRL.STATUS field. There is one bit for each bpp mode. All histogram bpp modes are checked simultaneously. The HW_PXP_HIST_A_BUF_SIZE fields specify the height and width of the update area. This allows the histogram engine to keep track of the relative coordinates of each new pixel it receives. It assumes the pixels are fed to it in raster order, left to right and top to bottom.

13.6.11.2 Mask Functionality

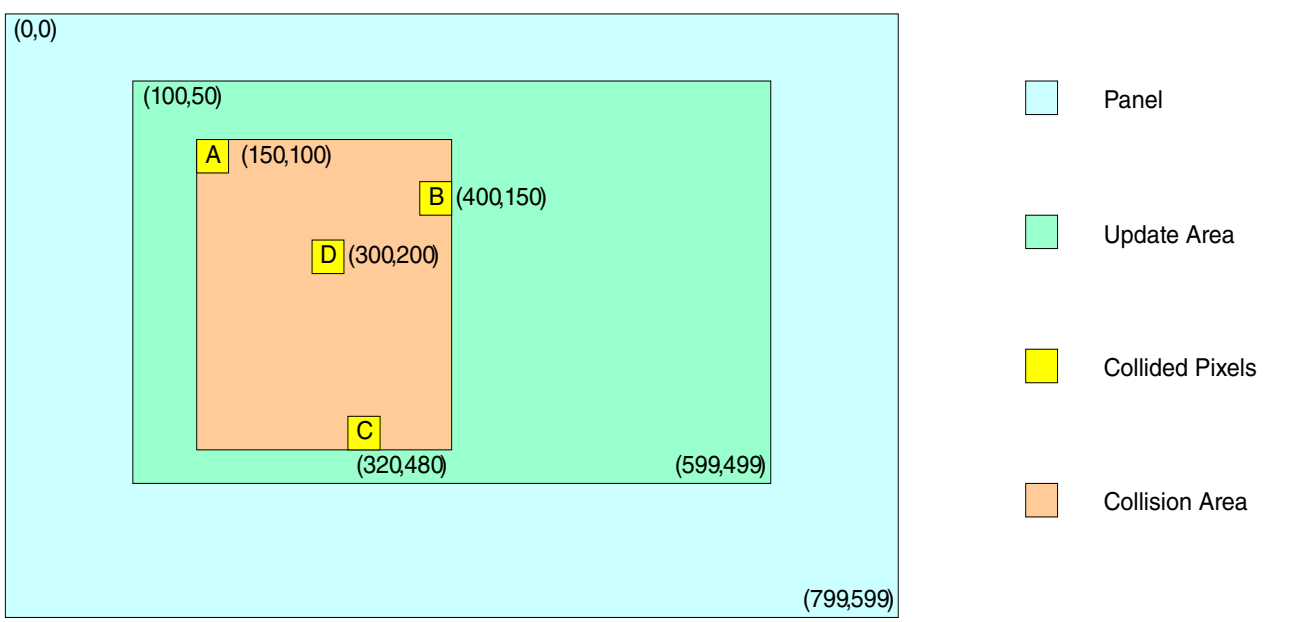
There is also a mask function within the histogram engine that tests a specified string of bits from the input against values programmed into the mask register. The use-case for this is to report collision detection back to controlling software. If the mask test passes then the pixel is active and is processed in the engine and statistics collected. If the pixel is not active then it is discarded and not processed – does not contribute to the output results. Based on the outcome of this mask test, there are some additional results that are output to register fields. They are the number of active pixels, the coordinates of the smallest bounding box including all active pixels and an array of 64 bits where the offset of each bit corresponds to the pixel value of processed pixel. The bit is set for each valid pixel value processed.

The configuration of the mask function is contained in the HW_PXP_HIST_x_MASK register. The HW_PXP_HIST_A_MASK.MASK_EN bit turns on the function. . The HW_PXP_HIST_A_CTRL.CLEAR bit clears the results. The HW_PXP_HIST_A_MASK.MASK_MODE field specifies the logical operation for the test. The options are “equal”, “not equal”, “inside” or “outside.” The HW_PXP_HIST_A_MASK.MASK_VALUE0 and HW_PXP_HIST_A_MASK.MASK_VALUE1 are used for comparison with the incoming value depending on the mode chosen. The starting bit of mask field in the 72 bit input data is specified by the HW_PXP_HIST_A_MASK.MASK_OFFSET register field and the width is given by HW_PXP_HIST_A_MASK.MASK_WIDTH. The width of the mask is a maximum of 8 bits. If the mask mode test passes for a given input then the HW_PXP_HIST_A_TOTAL_PIXEL read only value is incremented. The x and y coordinates of the pixel are used to calculate the minimum rectangle the will enclose all valid pixels. This information can be read from the HW_PXP_HIST_A_ACTIVE_AREA_X and HW_PXP_HIST_A_ACTIVE_AREA_Y registers. Also, the values of the valid pixels are recorded in the HW_PXP_HIST_A_RAW_STAT0 and HW_PXP_HIST_A_RAW_STAT0 registers.

These registers make up 64 one bit fields where the offsets of the bit that are set are the pixel values of the active pixels. (The values as specifed by the HW_PXP_HIST_A_CTRL. PIXEL_OFFSET and HW_PXP_HIST_A_CTRL.PIXEL_WIDTH fields.)

13.6.11.3 Collision use-case Example

In this example, the histogram engine is used to test for and report collision information within the update. Refer to the figure below. With normal WFE-A processing, within the 72 bit output there will be a one bit collision flag that is assert to 1 if there was a collision on that update pixel. There will also be a 6 bit field that specifies the EPDC LUT number for that pixel in the working buffer. If there was a collsion this LUT number identifies the unique colliding live update. To covey all this information to software using the histogram engine, the HW_PXP_HIST_A_CTRL. PIXEL_OFFSET and HW_PXP_HIST_A_CTRL.PIXEL_WIDTH should be set to point to the current LUT in the input bus. The HW_PXP_HIST_A_MASK.MASK_OFFSET and HW_PXP_HIST_A_MASK.MASK_WIDTH fields should be set to point to the collsion bit. The HW_PXP_HIST_A_MASK.MASK_MODE should be 0 or “equal” and the HW_PXP_HIST_A_MASK.MASK_VALUE0 should be set to 1.



The figure above shows the processed results of our example. There is an update region, 499x449, offset within the800x600 panel display. After enabling the histogram and processing the update, there is a collided region within the update region and 4 collided pixels were detected. The results are as follows.

The HW_PXP_HIST_A_TOTAL_PIXEL register would equal to 4 showing that 4 pixels were found by the mask function to have their collision bits set.

The relative collision area are would be given in the HW_PXP_HIST_A_ACTIVE_AREA_X and Y register as MIN_X_OFFSET = 50, MAX_X_OFFSET = 300, MIN_Y_OFFSET = 50, MAX_Y_OFFSET = 380.

If pixel A is being updated by LUT 1, pixel B and C are being updated LUT 5, and pixel D is being updated by LUT 12, then the LUT Collision Status = 0x00001022 (bit 1, 2, 12 are set). That is HW_PXP_HIST_A_RAW_STAT0 would be equal to 0x00001022 and HW_PXP_HIST_A_RAW_STAT1 would be 0x00000000.

13.6.12 PXP Memory Map/Register Definition

PXP Hardware Register Format Summary

PXP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3070_0000	Control Register 0 (PXP_HW_PXP_CTRL)	32	R/W	C700_8000h	13.6.12.1/ 3875
3070_0010	Status Register (PXP_HW_PXP_STAT)	32	R/W	0000_0000h	13.6.12.2/ 3878
3070_0020	Output Buffer Control Register (PXP_HW_PXP_OUT_CTRL)	32	R/W	0000_0000h	13.6.12.3/ 3880
3070_0030	Output Frame Buffer Pointer (PXP_HW_PXP_OUT_BUF)	32	R/W	0000_0000h	13.6.12.4/ 3882
3070_0040	Output Frame Buffer Pointer #2 (PXP_HW_PXP_OUT_BUF2)	32	R/W	0000_0000h	13.6.12.5/ 3883
3070_0050	Output Buffer Pitch (PXP_HW_PXP_OUT_PITCH)	32	R/W	0000_0000h	13.6.12.6/ 3883
3070_0060	Output Surface Lower Right Coordinate (PXP_HW_PXP_OUT_LRC)	32	R/W	0000_0000h	13.6.12.7/ 3884
3070_0070	Processed Surface Upper Left Coordinate (PXP_HW_PXP_OUT_PS_ULC)	32	R/W	0000_0000h	13.6.12.8/ 3885
3070_0080	Processed Surface Lower Right Coordinate (PXP_HW_PXP_OUT_PS_LRC)	32	R/W	0000_0000h	13.6.12.9/ 3886
3070_0090	Alpha Surface Upper Left Coordinate (PXP_HW_PXP_OUT_AS_ULC)	32	R/W	0000_0000h	13.6.12.10/ 3887
3070_00A0	Alpha Surface Lower Right Coordinate (PXP_HW_PXP_OUT_AS_LRC)	32	R/W	0000_0000h	13.6.12.11/ 3888
3070_00B0	Processed Surface (PS) Control Register (PXP_HW_PXP_PS_CTRL)	32	R/W	0000_0000h	13.6.12.12/ 3889

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3070_00C0	PS Input Buffer Address (PXP_HW_PXP_PS_BUF)	32	R/W	0000_0000h	13.6.12.13/ 3891
3070_00D0	PS U/Cb or 2 Plane UV Input Buffer Address (PXP_HW_PXP_PS_UBUF)	32	R/W	0000_0000h	13.6.12.14/ 3891
3070_00E0	PS V/Cr Input Buffer Address (PXP_HW_PXP_PS_VBUF)	32	R/W	0000_0000h	13.6.12.15/ 3892
3070_00F0	Processed Surface Pitch (PXP_HW_PXP_PS_PITCH)	32	R/W	0000_0000h	13.6.12.16/ 3893
3070_0100	PS Background Color (PXP_HW_PXP_PS_BACKGROUND_0)	32	R/W	0000_0000h	13.6.12.17/ 3894
3070_0110	PS Scale Factor Register (PXP_HW_PXP_PS_SCALE)	32	R/W	1000_1000h	13.6.12.18/ 3894
3070_0120	PS Scale Offset Register (PXP_HW_PXP_PS_OFFSET)	32	R/W	0000_0000h	13.6.12.19/ 3896
3070_0130	PS Color Key Low (PXP_HW_PXP_PS_CLRKEYLOW_0)	32	R/W	00FF_FFFFh	13.6.12.20/ 3897
3070_0140	PS Color Key High (PXP_HW_PXP_PS_CLRKEYHIGH_0)	32	R/W	0000_0000h	13.6.12.21/ 3897
3070_0150	Alpha Surface Control (PXP_HW_PXP_AS_CTRL)	32	R/W	0000_0000h	13.6.12.22/ 3898
3070_0160	Alpha Surface Buffer Pointer (PXP_HW_PXP_AS_BUF)	32	R/W	0000_0000h	13.6.12.23/ 3900
3070_0170	Alpha Surface Pitch (PXP_HW_PXP_AS_PITCH)	32	R/W	0000_0000h	13.6.12.24/ 3901
3070_0180	Overlay Color Key Low (PXP_HW_PXP_AS_CLRKEYLOW_0)	32	R/W	00FF_FFFFh	13.6.12.25/ 3902
3070_0190	Overlay Color Key High (PXP_HW_PXP_AS_CLRKEYHIGH_0)	32	R/W	0000_0000h	13.6.12.26/ 3902
3070_01A0	Color Space Conversion Coefficient Register 0 (PXP_HW_PXP_CSC1_COEF0)	32	R/W	0400_0000h	13.6.12.27/ 3903
3070_01B0	Color Space Conversion Coefficient Register 1 (PXP_HW_PXP_CSC1_COEF1)	32	R/W	0123_0208h	13.6.12.28/ 3905
3070_01C0	Color Space Conversion Coefficient Register 2 (PXP_HW_PXP_CSC1_COEF2)	32	R/W	079B_076Ch	13.6.12.29/ 3905
3070_01D0	Color Space Conversion Control Register. (PXP_HW_PXP_CSC2_CTRL)	32	R/W	0000_0001h	13.6.12.30/ 3906
3070_01E0	Color Space Conversion Coefficient Register 0 (PXP_HW_PXP_CSC2_COEF0)	32	R/W	0000_0000h	13.6.12.31/ 3908
3070_01F0	Color Space Conversion Coefficient Register 1 (PXP_HW_PXP_CSC2_COEF1)	32	R/W	0000_0000h	13.6.12.32/ 3908
3070_0200	Color Space Conversion Coefficient Register 2 (PXP_HW_PXP_CSC2_COEF2)	32	R/W	0000_0000h	13.6.12.33/ 3909
3070_0210	Color Space Conversion Coefficient Register 3 (PXP_HW_PXP_CSC2_COEF3)	32	R/W	0000_0000h	13.6.12.34/ 3909

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3070_0220	Color Space Conversion Coefficient Register 4 (PXP_HW_PXP_CSC2_COEF4)	32	R/W	0000_0000h	13.6.12.35/ 3910
3070_0230	Color Space Conversion Coefficient Register 5 (PXP_HW_PXP_CSC2_COEF5)	32	R/W	0000_0000h	13.6.12.36/ 3910
3070_0240	Lookup Table Control Register. (PXP_HW_PXP_LUT_CTRL)	32	R/W	8001_0000h	13.6.12.37/ 3911
3070_0250	Lookup Table Control Register. (PXP_HW_PXP_LUT_ADDR)	32	R/W	0000_0000h	13.6.12.38/ 3913
3070_0260	Lookup Table Data Register. (PXP_HW_PXP_LUT_DATA)	32	R/W	0000_0000h	13.6.12.39/ 3915
3070_0270	Lookup Table External Memory Address Register. (PXP_HW_PXP_LUT_EXTMEM)	32	R/W	0000_0000h	13.6.12.40/ 3915
3070_0280	Color Filter Array Register. (PXP_HW_PXP_CFA)	32	R/W	0000_0000h	13.6.12.41/ 3915
3070_0290	PXP Alpha Engine A Control Register. (PXP_HW_PXP_ALPHA_A_CTRL)	32	R/W	0000_0000h	13.6.12.42/ 3916
3070_02A0	PXP Alpha Engine B Control Register. (PXP_HW_PXP_ALPHA_B_CTRL)	32	R/W	0000_0000h	13.6.12.43/ 3918
3070_02B0	PXP_HW_PXP_ALPHA_B_CTRL_1	32	R/W	0000_0000h	13.6.12.44/ 3921
3070_02C0	PS Background Color 1 (PXP_HW_PXP_PS_BACKGROUND_1)	32	R/W	0000_0000h	13.6.12.45/ 3922
3070_02D0	PS Color Key Low 1 (PXP_HW_PXP_PS_CLRKEYLOW_1)	32	R/W	00FF_FFFFh	13.6.12.46/ 3923
3070_02E0	PS Color Key High 1 (PXP_HW_PXP_PS_CLRKEYHIGH_1)	32	R/W	0000_0000h	13.6.12.47/ 3923
3070_02F0	Overlay Color Key Low (PXP_HW_PXP_AS_CLRKEYLOW_1)	32	R/W	00FF_FFFFh	13.6.12.48/ 3924
3070_0300	Overlay Color Key High (PXP_HW_PXP_AS_CLRKEYHIGH_1)	32	R/W	0000_0000h	13.6.12.49/ 3925
3070_0310	Control Register 2 (PXP_HW_PXP_CTRL2)	32	R/W	0000_0000h	13.6.12.50/ 3926
3070_0320	PXP Power Control Register. (PXP_HW_PXP_POWER_REG0)	32	R/W	0000_0000h	13.6.12.51/ 3928
3070_0330	PXP Power Control Register 1. (PXP_HW_PXP_POWER_REG1)	32	R/W	0000_0000h	13.6.12.52/ 3929
3070_0350	PXP_HW_PXP_DATA_PATH_CTRL1	32	R/W	0000_0000h	13.6.12.53/ 3931
3070_0360	Initialize memory buffer control Register (PXP_HW_PXP_INIT_MEM_CTRL)	32	R/W	0000_0000h	13.6.12.54/ 3932
3070_0370	Write data Register (PXP_HW_PXP_INIT_MEM_DATA)	32	R/W	0000_0000h	13.6.12.55/ 3933
3070_0380	Write data Register (PXP_HW_PXP_INIT_MEM_DATA_HIGH)	32	R/W	0000_0000h	13.6.12.56/ 3934

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3070_0390	PXP IRQ Mask Register (PXP_HW_PXP_IRQ_MASK)	32	R/W	0000_0000h	13.6.12.57/ 3934
3070_03A0	PXP Interrupt Register (PXP_HW_PXP_IRQ)	32	R/W	0000_0000h	13.6.12.58/ 3937
3070_0400	Next Frame Pointer (PXP_HW_PXP_NEXT)	32	R/W	0000_0000h	13.6.12.59/ 3939
3070_0450	Pre-fetch engine Control Channel 0 Register (PXP_HW_PXP_INPUT_FETCH_CTRL_CH0)	32	R/W	0002_0000h	13.6.12.60/ 3941
3070_0460	Pre-fetch engine Control Channel 1 Register (PXP_HW_PXP_INPUT_FETCH_CTRL_CH1)	32	R/W	0002_0000h	13.6.12.61/ 3943
3070_0470	Pre-fetch engine status Channel 0 Register (PXP_HW_PXP_INPUT_FETCH_STATUS_CH0)	32	R/W	0000_0000h	13.6.12.62/ 3946
3070_0480	Store engine status Channel 1 Register (PXP_HW_PXP_INPUT_FETCH_STATUS_CH1)	32	R/W	0000_0000h	13.6.12.63/ 3947
3070_0490	PXP_HW_PXP_INPUT_FETCH_ACTIVE_SIZE_ULC_CH0	32	R/W	0000_0000h	13.6.12.64/ 3947
3070_04A0	PXP_HW_PXP_INPUT_FETCH_ACTIVE_SIZE_LRC_CH0	32	R/W	0000_0000h	13.6.12.65/ 3948
3070_04B0	PXP_HW_PXP_INPUT_FETCH_ACTIVE_SIZE_ULC_CH1	32	R/W	0000_0000h	13.6.12.66/ 3948
3070_04C0	PXP_HW_PXP_INPUT_FETCH_ACTIVE_SIZE_LRC_CH1	32	R/W	0000_0000h	13.6.12.67/ 3949
3070_04D0	PXP_HW_PXP_INPUT_FETCH_SIZE_CH0	32	R/W	0000_0000h	13.6.12.68/ 3949
3070_04E0	PXP_HW_PXP_INPUT_FETCH_SIZE_CH1	32	R/W	0000_0000h	13.6.12.69/ 3950
3070_04F0	PXP_HW_PXP_INPUT_FETCH_BACKGROUND_COLOR_CH0	32	R/W	0000_0000h	13.6.12.70/ 3950
3070_0500	PXP_HW_PXP_INPUT_FETCH_BACKGROUND_COLOR_CH1	32	R/W	0000_0000h	13.6.12.71/ 3951
3070_0510	PXP_HW_PXP_INPUT_FETCH_PITCH	32	R/W	0000_0000h	13.6.12.72/ 3951
3070_0520	PXP_HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH0	32	R/W	0000_1000h	13.6.12.73/ 3952
3070_0530	PXP_HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH1	32	R/W	0000_1000h	13.6.12.74/ 3953
3070_0540	PXP_HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH0	32	R/W	0000_0000h	13.6.12.75/ 3955
3070_0550	PXP_HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH1	32	R/W	0000_0000h	13.6.12.76/ 3956
3070_0560	PXP_HW_PXP_INPUT_FETCH_SHIFT_WIDTH_CH0	32	R/W	0000_8888h	13.6.12.77/ 3957
3070_0570	PXP_HW_PXP_INPUT_FETCH_SHIFT_WIDTH_CH1	32	R/W	0000_8888h	13.6.12.78/ 3958

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3070_0580	PXP_HW_PXP_INPUT_FETCH_ADDR_0_CH0	32	R/W	0000_0000h	13.6.12.79/ 3958
3070_0590	PXP_HW_PXP_INPUT_FETCH_ADDR_1_CH0	32	R/W	0000_0000h	13.6.12.80/ 3959
3070_05A0	PXP_HW_PXP_INPUT_FETCH_ADDR_0_CH1	32	R/W	0000_0000h	13.6.12.81/ 3959
3070_05B0	PXP_HW_PXP_INPUT_FETCH_ADDR_1_CH1	32	R/W	0000_0000h	13.6.12.82/ 3960
3070_05C0	Store engine Control Channel 0 Register (PXP_HW_PXP_INPUT_STORE_CTRL_CH0)	32	R/W	0002_0200h	13.6.12.83/ 3960
3070_05D0	Store engine Control Channel 1 Register (PXP_HW_PXP_INPUT_STORE_CTRL_CH1)	32	R/W	0002_0200h	13.6.12.84/ 3963
3070_05E0	Store engine status Channel 0 Register (PXP_HW_PXP_INPUT_STORE_STATUS_CH0)	32	R/W	0000_0000h	13.6.12.85/ 3965
3070_05F0	Store engine status Channel 1 Register (PXP_HW_PXP_INPUT_STORE_STATUS_CH1)	32	R/W	0000_0000h	13.6.12.86/ 3966
3070_0600	PXP_HW_PXP_INPUT_STORE_SIZE_CH0	32	R/W	0000_0000h	13.6.12.87/ 3966
3070_0610	PXP_HW_PXP_INPUT_STORE_SIZE_CH1	32	R/W	0000_0000h	13.6.12.88/ 3967
3070_0620	PXP_HW_PXP_INPUT_STORE_PITCH	32	R/W	0000_0000h	13.6.12.89/ 3967
3070_0630	PXP_HW_PXP_INPUT_STORE_SHIFT_CTRL_CH0	32	R/W	0000_0000h	13.6.12.90/ 3968
3070_0640	PXP_HW_PXP_INPUT_STORE_SHIFT_CTRL_CH1	32	R/W	0000_0000h	13.6.12.91/ 3969
3070_0690	PXP_HW_PXP_INPUT_STORE_ADDR_0_CH0	32	R/W	0000_0000h	13.6.12.92/ 3971
3070_06A0	PXP_HW_PXP_INPUT_STORE_ADDR_1_CH0	32	R/W	0000_0000h	13.6.12.93/ 3971
3070_06B0	PXP_HW_PXP_INPUT_STORE_FILL_DATA_CH0	32	R/W	0000_0000h	13.6.12.94/ 3972
3070_06C0	PXP_HW_PXP_INPUT_STORE_ADDR_0_CH1	32	R/W	0000_0000h	13.6.12.95/ 3972
3070_06D0	PXP_HW_PXP_INPUT_STORE_ADDR_1_CH1	32	R/W	0000_0000h	13.6.12.96/ 3973
3070_06E0	PXP_HW_PXP_INPUT_STORE_D_MASK0_H_CH0	32	R/W	0000_0000h	13.6.12.97/ 3973
3070_06F0	PXP_HW_PXP_INPUT_STORE_D_MASK0_L_CH0	32	R/W	0000_0000h	13.6.12.98/ 3974
3070_0700	PXP_HW_PXP_INPUT_STORE_D_MASK1_H_CH0	32	R/W	0000_0000h	13.6.12.99/ 3974
3070_0710	PXP_HW_PXP_INPUT_STORE_D_MASK1_L_CH0	32	R/W	0000_0000h	13.6.12.100/ 3974

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3070_0720	PXP_HW_PXP_INPUT_STORE_D_MASK2_H_CH0	32	R/W	0000_0000h	13.6.12.101/ 3975
3070_0730	PXP_HW_PXP_INPUT_STORE_D_MASK2_L_CH0	32	R/W	0000_0000h	13.6.12.102/ 3975
3070_0740	PXP_HW_PXP_INPUT_STORE_D_MASK3_H_CH0	32	R/W	0000_0000h	13.6.12.103/ 3976
3070_0750	PXP_HW_PXP_INPUT_STORE_D_MASK3_L_CH0	32	R/W	0000_0000h	13.6.12.104/ 3976
3070_0760	PXP_HW_PXP_INPUT_STORE_D_MASK4_H_CH0	32	R/W	0000_0000h	13.6.12.105/ 3977
3070_0770	PXP_HW_PXP_INPUT_STORE_D_MASK4_L_CH0	32	R/W	0000_0000h	13.6.12.106/ 3977
3070_0780	PXP_HW_PXP_INPUT_STORE_D_MASK5_H_CH0	32	R/W	0000_0000h	13.6.12.107/ 3978
3070_0790	PXP_HW_PXP_INPUT_STORE_D_MASK5_L_CH0	32	R/W	0000_0000h	13.6.12.108/ 3978
3070_07A0	PXP_HW_PXP_INPUT_STORE_D_MASK6_H_CH0	32	R/W	0000_0000h	13.6.12.109/ 3978
3070_07B0	PXP_HW_PXP_INPUT_STORE_D_MASK6_L_CH0	32	R/W	0000_0000h	13.6.12.110/ 3979
3070_07C0	PXP_HW_PXP_INPUT_STORE_D_MASK7_H_CH0	32	R/W	0000_0000h	13.6.12.111/ 3979
3070_07E0	PXP_HW_PXP_INPUT_STORE_D_MASK7_L_CH0	32	R/W	0000_0000h	13.6.12.112/ 3980
3070_07F0	PXP_HW_PXP_INPUT_STORE_D_SHIFT_L_CH0	32	R/W	0000_0000h	13.6.12.113/ 3980
3070_0800	PXP_HW_PXP_INPUT_STORE_D_SHIFT_H_CH0	32	R/W	0000_0000h	13.6.12.114/ 3982
3070_0810	PXP_HW_PXP_INPUT_STORE_F_SHIFT_L_CH0	32	R/W	0000_0000h	13.6.12.115/ 3984
3070_0820	PXP_HW_PXP_INPUT_STORE_F_SHIFT_H_CH0	32	R/W	0000_0000h	13.6.12.116/ 3986
3070_0830	PXP_HW_PXP_INPUT_STORE_F_MASK_L_CH0	32	R/W	0000_0000h	13.6.12.117/ 3988
3070_0840	PXP_HW_PXP_INPUT_STORE_F_MASK_H_CH0	32	R/W	0000_0000h	13.6.12.118/ 3988
3070_0850	Pre-fetch engine Control Channel 0 Register (PXP_HW_PXP_DITHER_FETCH_CTRL_CH0)	32	R/W	0002_0000h	13.6.12.119/ 3989
3070_0860	Pre-fetch engine Control Channel 1 Register (PXP_HW_PXP_DITHER_FETCH_CTRL_CH1)	32	R/W	0002_0000h	13.6.12.120/ 3991
3070_0870	Pre-fetch engine status Channel 0 Register (PXP_HW_PXP_DITHER_FETCH_STATUS_CH0)	32	R/W	0000_0000h	13.6.12.121/ 3994
3070_0880	Store engine status Channel 1 Register (PXP_HW_PXP_DITHER_FETCH_STATUS_CH1)	32	R/W	0000_0000h	13.6.12.122/ 3995

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3070_0890	PXP_HW_PXP_DITHER_FETCH_ACTIVE_SIZE_ULC_CH 0	32	R/W	0000_0000h	13.6.12.123/ 3995
3070_08A0	PXP_HW_PXP_DITHER_FETCH_ACTIVE_SIZE_LRC_CH 0	32	R/W	0000_0000h	13.6.12.124/ 3996
3070_08B0	PXP_HW_PXP_DITHER_FETCH_ACTIVE_SIZE_ULC_CH 1	32	R/W	0000_0000h	13.6.12.125/ 3996
3070_08C0	PXP_HW_PXP_DITHER_FETCH_ACTIVE_SIZE_LRC_CH 1	32	R/W	0000_0000h	13.6.12.126/ 3997
3070_08D0	PXP_HW_PXP_DITHER_FETCH_SIZE_CH0	32	R/W	0000_0000h	13.6.12.127/ 3997
3070_08E0	PXP_HW_PXP_DITHER_FETCH_SIZE_CH1	32	R/W	0000_0000h	13.6.12.128/ 3998
3070_08F0	PXP_HW_PXP_DITHER_FETCH_BACKGROUND_COLOR _CH0	32	R/W	0000_0000h	13.6.12.129/ 3998
3070_0900	PXP_HW_PXP_DITHER_FETCH_BACKGROUND_COLOR _CH1	32	R/W	0000_0000h	13.6.12.130/ 3999
3070_0910	PXP_HW_PXP_DITHER_FETCH_PITCH	32	R/W	0000_0000h	13.6.12.131/ 3999
3070_0920	PXP_HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH0	32	R/W	0000_1000h	13.6.12.132/ 4000
3070_0930	PXP_HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH1	32	R/W	0000_1000h	13.6.12.133/ 4001
3070_0940	PXP_HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH0	32	R/W	0000_0000h	13.6.12.134/ 4003
3070_0950	PXP_HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH1	32	R/W	0000_0000h	13.6.12.135/ 4004
3070_0960	PXP_HW_PXP_DITHER_FETCH_SHIFT_WIDTH_CH0	32	R/W	0000_8888h	13.6.12.136/ 4005
3070_0970	PXP_HW_PXP_DITHER_FETCH_SHIFT_WIDTH_CH1	32	R/W	0000_8888h	13.6.12.137/ 4006
3070_0980	PXP_HW_PXP_DITHER_FETCH_ADDR_0_CH0	32	R/W	0000_0000h	13.6.12.138/ 4006
3070_0990	PXP_HW_PXP_DITHER_FETCH_ADDR_1_CH0	32	R/W	0000_0000h	13.6.12.139/ 4007
3070_09A0	PXP_HW_PXP_DITHER_FETCH_ADDR_0_CH1	32	R/W	0000_0000h	13.6.12.140/ 4007
3070_09B0	PXP_HW_PXP_DITHER_FETCH_ADDR_1_CH1	32	R/W	0000_0000h	13.6.12.141/ 4008
3070_09C0	Store engine Control Channel 0 Register (PXP_HW_PXP_DITHER_STORE_CTRL_CH0)	32	R/W	0002_0200h	13.6.12.142/ 4008
3070_09D0	Store engine Control Channel 1 Register (PXP_HW_PXP_DITHER_STORE_CTRL_CH1)	32	R/W	0002_0200h	13.6.12.143/ 4011
3070_09E0	Store engine status Channel 0 Register (PXP_HW_PXP_DITHER_STORE_STATUS_CH0)	32	R/W	0000_0000h	13.6.12.144/ 4013

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3070_09F0	Store engine status Channel 1 Register (PXP_HW_PXP_DITHER_STORE_STATUS_CH1)	32	R/W	0000_0000h	13.6.12.145/ 4014
3070_0A00	PXP_HW_PXP_DITHER_STORE_SIZE_CH0	32	R/W	0000_0000h	13.6.12.146/ 4014
3070_0A10	PXP_HW_PXP_DITHER_STORE_SIZE_CH1	32	R/W	0000_0000h	13.6.12.147/ 4015
3070_0A20	PXP_HW_PXP_DITHER_STORE_PITCH	32	R/W	0000_0000h	13.6.12.148/ 4015
3070_0A30	PXP_HW_PXP_DITHER_STORE_SHIFT_CTRL_CH0	32	R/W	0000_0000h	13.6.12.149/ 4016
3070_0A40	PXP_HW_PXP_DITHER_STORE_SHIFT_CTRL_CH1	32	R/W	0000_0000h	13.6.12.150/ 4017
3070_0A90	PXP_HW_PXP_DITHER_STORE_ADDR_0_CH0	32	R/W	0000_0000h	13.6.12.151/ 4019
3070_0AA0	PXP_HW_PXP_DITHER_STORE_ADDR_1_CH0	32	R/W	0000_0000h	13.6.12.152/ 4019
3070_0AB0	PXP_HW_PXP_DITHER_STORE_FILL_DATA_CH0	32	R/W	0000_0000h	13.6.12.153/ 4020
3070_0AC0	PXP_HW_PXP_DITHER_STORE_ADDR_0_CH1	32	R/W	0000_0000h	13.6.12.154/ 4020
3070_0AD0	PXP_HW_PXP_DITHER_STORE_ADDR_1_CH1	32	R/W	0000_0000h	13.6.12.155/ 4021
3070_0AE0	PXP_HW_PXP_DITHER_STORE_D_MASK0_H_CH0	32	R/W	0000_0000h	13.6.12.156/ 4021
3070_0AF0	PXP_HW_PXP_DITHER_STORE_D_MASK0_L_CH0	32	R/W	0000_0000h	13.6.12.157/ 4022
3070_0B00	PXP_HW_PXP_DITHER_STORE_D_MASK1_H_CH0	32	R/W	0000_0000h	13.6.12.158/ 4022
3070_0B10	PXP_HW_PXP_DITHER_STORE_D_MASK1_L_CH0	32	R/W	0000_0000h	13.6.12.159/ 4022
3070_0B20	PXP_HW_PXP_DITHER_STORE_D_MASK2_H_CH0	32	R/W	0000_0000h	13.6.12.160/ 4023
3070_0B30	PXP_HW_PXP_DITHER_STORE_D_MASK2_L_CH0	32	R/W	0000_0000h	13.6.12.161/ 4023
3070_0B40	PXP_HW_PXP_DITHER_STORE_D_MASK3_H_CH0	32	R/W	0000_0000h	13.6.12.162/ 4024
3070_0B50	PXP_HW_PXP_DITHER_STORE_D_MASK3_L_CH0	32	R/W	0000_0000h	13.6.12.163/ 4024
3070_0B60	PXP_HW_PXP_DITHER_STORE_D_MASK4_H_CH0	32	R/W	0000_0000h	13.6.12.164/ 4025
3070_0B70	PXP_HW_PXP_DITHER_STORE_D_MASK4_L_CH0	32	R/W	0000_0000h	13.6.12.165/ 4025
3070_0B80	PXP_HW_PXP_DITHER_STORE_D_MASK5_H_CH0	32	R/W	0000_0000h	13.6.12.166/ 4026

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3070_0B90	PXP_HW_PXP_DITHER_STORE_D_MASK5_L_CH0	32	R/W	0000_0000h	13.6.12.167/ 4026
3070_0BA0	PXP_HW_PXP_DITHER_STORE_D_MASK6_H_CH0	32	R/W	0000_0000h	13.6.12.168/ 4026
3070_0BB0	PXP_HW_PXP_DITHER_STORE_D_MASK6_L_CH0	32	R/W	0000_0000h	13.6.12.169/ 4027
3070_0BC0	PXP_HW_PXP_DITHER_STORE_D_MASK7_H_CH0	32	R/W	0000_0000h	13.6.12.170/ 4027
3070_0BD0	PXP_HW_PXP_DITHER_STORE_D_MASK7_L_CH0	32	R/W	0000_0000h	13.6.12.171/ 4028
3070_0BE0	PXP_HW_PXP_DITHER_STORE_D_SHIFT_L_CH0	32	R/W	0000_0000h	13.6.12.172/ 4028
3070_0BF0	PXP_HW_PXP_DITHER_STORE_D_SHIFT_H_CH0	32	R/W	0000_0000h	13.6.12.173/ 4030
3070_0C00	PXP_HW_PXP_DITHER_STORE_F_SHIFT_L_CH0	32	R/W	0000_0000h	13.6.12.174/ 4032
3070_0C10	PXP_HW_PXP_DITHER_STORE_F_SHIFT_H_CH0	32	R/W	0000_0000h	13.6.12.175/ 4034
3070_0C20	PXP_HW_PXP_DITHER_STORE_F_MASK_L_CH0	32	R/W	0000_0000h	13.6.12.176/ 4036
3070_0C30	PXP_HW_PXP_DITHER_STORE_F_MASK_H_CH0	32	R/W	0000_0000h	13.6.12.177/ 4036
3070_1670	Dither Control Register 0 (PXP_HW_PXP_DITHER_CTRL)	32	R/W	0054_4000h	13.6.12.178/ 4037
3070_1680	Final stage lookup value Register (PXP_HW_PXP_DITHER_FINAL_LUT_DATA0)	32	R/W	0000_0000h	13.6.12.179/ 4040
3070_1690	Final stage lookup value Register (PXP_HW_PXP_DITHER_FINAL_LUT_DATA1)	32	R/W	0000_0000h	13.6.12.180/ 4041
3070_16A0	Final stage lookup value Register (PXP_HW_PXP_DITHER_FINAL_LUT_DATA2)	32	R/W	0000_0000h	13.6.12.181/ 4042
3070_16B0	Final stage lookup value Register (PXP_HW_PXP_DITHER_FINAL_LUT_DATA3)	32	R/W	0000_0000h	13.6.12.182/ 4042
3070_2A00	Histogram Control Register. (PXP_HW_PXP_HIST_A_CTRL)	32	R/W	0500_1F00h	13.6.12.183/ 4043
3070_2A10	Histogram Pixel Mask Register. (PXP_HW_PXP_HIST_A_MASK)	32	R/W	0000_0000h	13.6.12.184/ 4044
3070_2A20	Histogram Pixel Buffer Size Register. (PXP_HW_PXP_HIST_A_BUF_SIZE)	32	R/W	0000_0000h	13.6.12.185/ 4046
3070_2A30	Total Number of Pixels Used by Histogram Engine. (PXP_HW_PXP_HIST_A_TOTAL_PIXEL)	32	R/W	0000_0000h	13.6.12.186/ 4046
3070_2A40	The X Coordinate Offset for Active Area. (PXP_HW_PXP_HIST_A_ACTIVE_AREA_X)	32	R/W	0000_0000h	13.6.12.187/ 4047
3070_2A50	The Y Coordinate Offset for Active Area. (PXP_HW_PXP_HIST_A_ACTIVE_AREA_Y)	32	R/W	0000_0000h	13.6.12.188/ 4047

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3070_2A60	Histogram Result Based on RAW Pixel Value. (PXP_HW_PXP_HIST_A_RAW_STAT0)	32	R/W	0000_0000h	13.6.12.189/ 4048
3070_2A70	Histogram Result Based on RAW Pixel Value. (PXP_HW_PXP_HIST_A_RAW_STAT1)	32	R/W	0000_0000h	13.6.12.190/ 4048
3070_2A80	Histogram Control Register. (PXP_HW_PXP_HIST_B_CTRL)	32	R/W	0500_1F00h	13.6.12.191/ 4049
3070_2A90	Histogram Pixel Mask Register. (PXP_HW_PXP_HIST_B_MASK)	32	R/W	0000_0000h	13.6.12.192/ 4050
3070_2AA0	Histogram Pixel Buffer Size Register. (PXP_HW_PXP_HIST_B_BUF_SIZE)	32	R/W	0000_0000h	13.6.12.193/ 4051
3070_2AB0	Total Number of Pixels Used by Histogram Engine. (PXP_HW_PXP_HIST_B_TOTAL_PIXEL)	32	R/W	0000_0000h	13.6.12.194/ 4052
3070_2AC0	The X Coordinate Offset for Active Area. (PXP_HW_PXP_HIST_B_ACTIVE_AREA_X)	32	R/W	0000_0000h	13.6.12.195/ 4052
3070_2AD0	The Y Coordinate Offset for Active Area. (PXP_HW_PXP_HIST_B_ACTIVE_AREA_Y)	32	R/W	0000_0000h	13.6.12.196/ 4053
3070_2AE0	Histogram Result Based on RAW Pixel Value. (PXP_HW_PXP_HIST_B_RAW_STAT0)	32	R/W	0000_0000h	13.6.12.197/ 4053
3070_2AF0	Histogram Result Based on RAW Pixel Value. (PXP_HW_PXP_HIST_B_RAW_STAT1)	32	R/W	0000_0000h	13.6.12.198/ 4054
3070_2B00	2-level Histogram Parameter Register. (PXP_HW_PXP_HIST2_PARAM)	32	R/W	0000_0F00h	13.6.12.199/ 4054
3070_2B10	4-level Histogram Parameter Register. (PXP_HW_PXP_HIST4_PARAM)	32	R/W	0F0A_0500h	13.6.12.200/ 4055
3070_2B20	8-level Histogram Parameter 0 Register. (PXP_HW_PXP_HIST8_PARAM0)	32	R/W	0604_0200h	13.6.12.201/ 4056
3070_2B30	8-level Histogram Parameter 1 Register. (PXP_HW_PXP_HIST8_PARAM1)	32	R/W	0F0D_0B09h	13.6.12.202/ 4057
3070_2B40	16-level Histogram Parameter 0 Register. (PXP_HW_PXP_HIST16_PARAM0)	32	R/W	0302_0100h	13.6.12.203/ 4058
3070_2B50	16-level Histogram Parameter 1 Register. (PXP_HW_PXP_HIST16_PARAM1)	32	R/W	0706_0504h	13.6.12.204/ 4059
3070_2B60	16-level Histogram Parameter 2 Register. (PXP_HW_PXP_HIST16_PARAM2)	32	R/W	0B0A_0908h	13.6.12.205/ 4060
3070_2B70	16-level Histogram Parameter 3 Register. (PXP_HW_PXP_HIST16_PARAM3)	32	R/W	0F0E_0D0Ch	13.6.12.206/ 4061
3070_2B80	32-level Histogram Parameter 0 Register. (PXP_HW_PXP_HIST32_PARAM0)	32	R/W	0302_0100h	13.6.12.207/ 4062
3070_2B90	32-level Histogram Parameter 1 Register. (PXP_HW_PXP_HIST32_PARAM1)	32	R/W	0706_0504h	13.6.12.208/ 4063
3070_2BA0	32-level Histogram Parameter 2 Register. (PXP_HW_PXP_HIST32_PARAM2)	32	R/W	0B0A_0908h	13.6.12.209/ 4064
3070_2BB0	32-level Histogram Parameter 3 Register. (PXP_HW_PXP_HIST32_PARAM3)	32	R/W	0F0E_0D0Ch	13.6.12.210/ 4065

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3070_2BC0	32-level Histogram Parameter 0 Register. (PXP_HW_PXP_HIST32_PARAM4)	32	R/W	0302_0100h	13.6.12.211/ 4066
3070_2BD0	32-level Histogram Parameter 1 Register. (PXP_HW_PXP_HIST32_PARAM5)	32	R/W	0706_0504h	13.6.12.212/ 4067
3070_2BE0	32-level Histogram Parameter 2 Register. (PXP_HW_PXP_HIST32_PARAM6)	32	R/W	0B0A_0908h	13.6.12.213/ 4068
3070_2BF0	32-level Histogram Parameter 3 Register. (PXP_HW_PXP_HIST32_PARAM7)	32	R/W	0F0E_0D0Ch	13.6.12.214/ 4069
3070_2C00	PXP_HW_PXP_COMP_CTRL	32	R/W	0000_0000h	13.6.12.215/ 4069
3070_2C10	PXP_HW_PXP_COMP_FORMAT0	32	R/W	0000_0000h	13.6.12.216/ 4070
3070_2C20	PXP_HW_PXP_COMP_FORMAT1	32	R/W	0000_0000h	13.6.12.217/ 4072
3070_2C30	PXP_HW_PXP_COMP_FORMAT2	32	R/W	0000_0000h	13.6.12.218/ 4073
3070_2C40	PXP_HW_PXP_COMP_MASK0	32	R/W	0000_0000h	13.6.12.219/ 4073
3070_2C50	PXP_HW_PXP_COMP_MASK1	32	R/W	0000_0000h	13.6.12.220/ 4074
3070_2C60	PXP_HW_PXP_COMP_BUFFER_SIZE	32	R/W	0000_0000h	13.6.12.221/ 4074
3070_2C70	PXP_HW_PXP_COMP_SOURCE	32	R/W	0000_0000h	13.6.12.222/ 4075
3070_2C80	PXP_HW_PXP_COMP_TARGET	32	R/W	0000_0000h	13.6.12.223/ 4075
3070_2C90	PXP_HW_PXP_COMP_BUFFER_A	32	R/W	0000_0000h	13.6.12.224/ 4076
3070_2CA0	PXP_HW_PXP_COMP_BUFFER_B	32	R/W	0000_0000h	13.6.12.225/ 4076
3070_2CB0	PXP_HW_PXP_COMP_BUFFER_C	32	R/W	0000_0000h	13.6.12.226/ 4077
3070_2CC0	PXP_HW_PXP_COMP_BUFFER_D	32	R/W	0000_0000h	13.6.12.227/ 4077
3070_2CD0	PXP_HW_PXP_COMP_DEBUG	32	R/W	0000_0000h	13.6.12.228/ 4077
3070_2CE0	PXP_HW_PXP_BUS_MUX	32	R/W	0000_0000h	13.6.12.229/ 4078
3070_2CF0	PXP_HW_PXP_HANDSHAKE_READY_MUX0	32	R/W	7654_3210h	13.6.12.230/ 4078
3070_2D00	PXP_HW_PXP_HANDSHAKE_READY_MUX1	32	R/W	FEDC _BA98h	13.6.12.231/ 4079
3070_2D10	PXP_HW_PXP_HANDSHAKE_DONE_MUX0	32	R/W	7654_3210h	13.6.12.232/ 4080

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3070_2D20	PXP_HW_PXP_HANDSHAKE_DONE_MUX1	32	R/W	FEDC _BA98h	13.6.12.233/ 4081
3070_2D30	PXP_HW_PXP_HANDSHAKE_CPU_FETCH	32	R/W	0010_0010h	13.6.12.234/ 4082
3070_2D40	PXP_HW_PXP_HANDSHAKE_CPU_STORE	32	R/W	0010_0010h	13.6.12.235/ 4084

13.6.12.1 Control Register 0 (PXP_HW_PXP_CTRL)

The CTRL register contains controls for the PXP module.

HW_PXP_CTRL: 0x000

HW_PXP_CTRL_SET: 0x004

HW_PXP_CTRL_CLR: 0x008

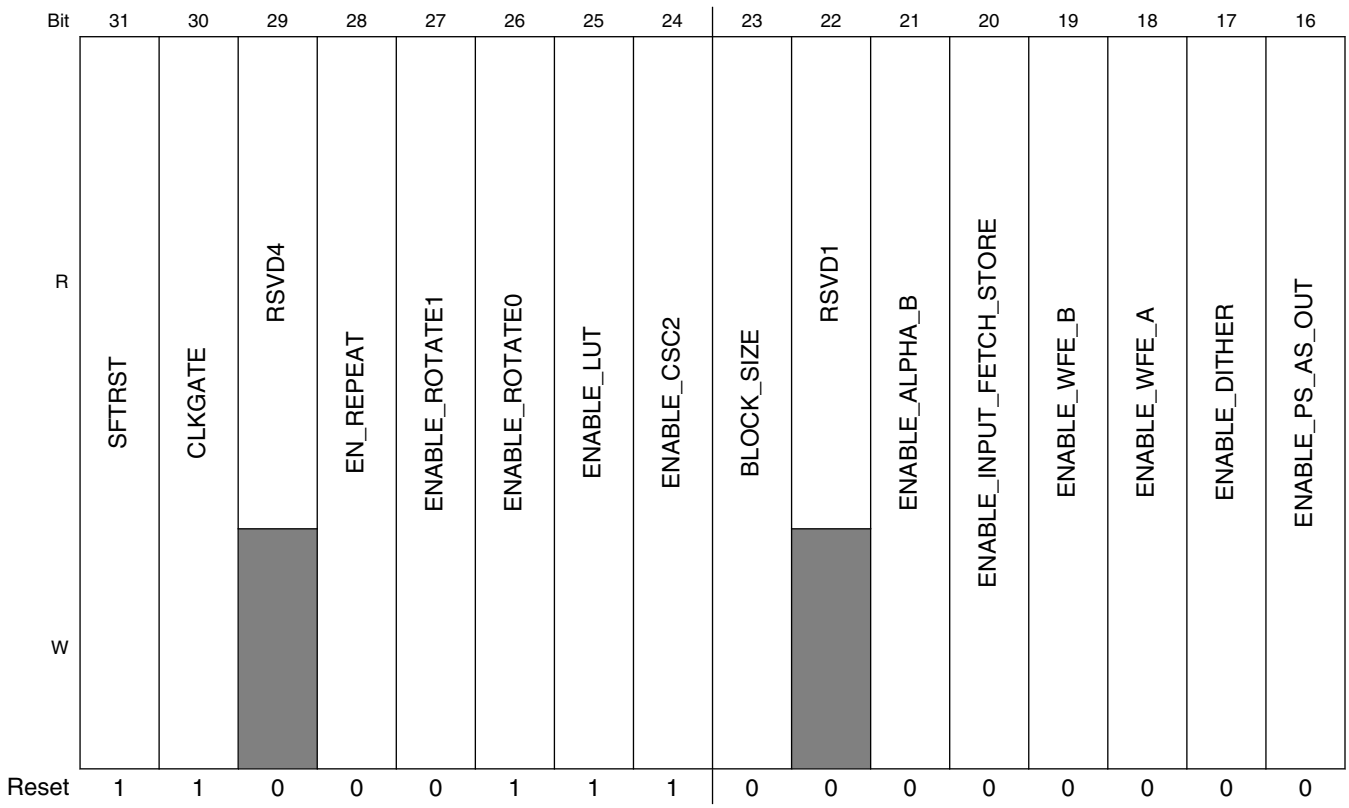
HW_PXP_CTRL_TOG: 0x00C

The Control register contains the primary controls for the PXP block.

EXAMPLE

```
REG_CTRL_SET(BM_PXP_CTRL_SFTRST);
REG_CTRL_CLR(BM_PXP_CTRL_SFTRST | BM_PXP_CTRL_CLKGATE);
```

Address: 3070_0000h base + 0h offset = 3070_0000h



PXP_HW_PXP_CTRL field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal PXP operation. Set this bit to one (default) to disable clocking with the PXP and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the PXP block to its default state.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.

Table continues on the next page...

PXP_HW_PXP_CTRL field descriptions (continued)

Field	Description
29 RSVD4	Reserved, always set to zero.
28 EN_REPEAT	Enable the PXP to run continuously. When this bit is set, the PXP will repeat based on the current configuration register settings. If this bit is not set, the PXP will complete the process and enter the idle state ready to accept the next frame to be processed. This bit should be set when the LCDIF handshake mode is enabled so that the next frame is automatically generated for the next screen refresh cycle. If it not set and the handshake mode is enabled, the CPU will have to initiate the PXP for the next refresh cycle. When the PXP NEXT feature is used, it has priority over the REPEAT mode, in that the new register settings are fetched first, and then the next PXP operation will continue.
27 ENABLE_ ROTATE1	Enable the ROTATE1 engine in the PXP primary processing flow.
26 ENABLE_ ROTATE0	Enable the ROTATE0 engine in the PXP primary processing flow.
25 ENABLE_LUT	Enable the LUT engine in the PXP primary processing flow.
24 ENABLE_CSC2	Enable the CSC2 engine in the PXP primary processing flow.
23 BLOCK_SIZE	Select the block size to process through the Rotate block. 0x0 **8X8** — Process 8x8 pixel blocks. 0x1 **16X16** — Process 16x16 pixel blocks.
22 RSVD1	Reserved, always set to zero.
21 ENABLE_ ALPHA_B	Enable the Alpha-B engine in the PXP primary processing flow.
20 ENABLE_ INPUT_FETCH_ STORE	Enable the Input Fetch and Store engine in the PXP primary processing flow.
19 ENABLE_WFE_B	Enable the WFE-B engine in the PXP primary processing flow.
18 ENABLE_WFE_A	Enable the WFE-A engine in the PXP primary processing flow.
17 ENABLE_ DITHER	Enable the Dithering engine in the PXP primary processing flow.
16 ENABLE_PS_ AS_OUT	Enable the PS engine, AS engine, OUTBUF in the PXP primary processing flow.
15 VFLIP1	Indicates that the input should be flipped vertically (effect applied before rotation).
14 HFLIP1	Indicates that the input should be flipped horizontally (effect applied before rotation).
13–12 ROTATE1	Indicates the clockwise rotation to be applied at the input buffer. The rotation effect is defined as occurring after the FLIP_X and FLIP_Y permutation.

Table continues on the next page...

PXP_HW_PXP_CTRL field descriptions (continued)

Field	Description
	0x0 **ROT_0** — 0x1 **ROT_90** — 0x2 **ROT_180** — 0x3 **ROT_270** —
11 VFLIP0	Indicates that the output buffer should be flipped vertically (effect applied before rotation).
10 HFLIP0	Indicates that the output buffer should be flipped horizontally (effect applied before rotation).
9–8 ROTATE0	Indicates the clockwise rotation to be applied at the output buffer. The rotation effect is defined as occurring after the FLIP_X and FLIP_Y permutation. 0x0 **ROT_0** — 0x1 **ROT_90** — 0x2 **ROT_180** — 0x3 **ROT_270** —
7–6 RSVD0	Reserved, always set to zero.
5 HANDSHAKE_ ABORT_SKIP	When skip is enable, even the abort asserted, pxp will not assert the ready directly but wait for whole block line complete.
4 ENABLE_LCD0_ HANDSHAKE	Enable handshake with LCD0 controller. When this is set, the PXP will not process an entire framebuffer, but will instead process rows of NxN blocks in a double-buffer handshake with the LCDIF. This enables the use of the onboard SRAM for a partial frame buffer.
3 LUT_DMA_IRQ_ ENABLE	LUT DMA interrupt enable. When set, the PXP will issue an interrupt when the LUT DMA has finished transferring data.
2 NEXT_IRQ_ ENABLE	Next command interrupt enable. When set, the PXP will issue an interrupt when a queued command initiated by a write to the PXP_NEXT register has been loaded into the PXP's registers. This interrupt also indicates that a new command may now be queued.
1 IRQ_ENABLE	Interrupt enable. NOTE: When using the HW_PXP_NEXT functionality to reprogram the PXP, the new value of this bit will be used and may therefore enable or disable an interrupt unintentionally.
0 ENABLE	Enables PXP operation with specified parameters. The ENABLE bit will remain set while the PXP is active and will be cleared once the current operation completes. Software should use the IRQ bit in the HW_PXP_STAT when polling for PXP completion.

13.6.12.2 Status Register (PXP_HW_PXP_STAT)

The PXP Interrupt Status register provides interrupt status information.

HW_PXP_STAT: 0x010

HW_PXP_STAT_SET: 0x014

HW_PXP_STAT_CLR: 0x018

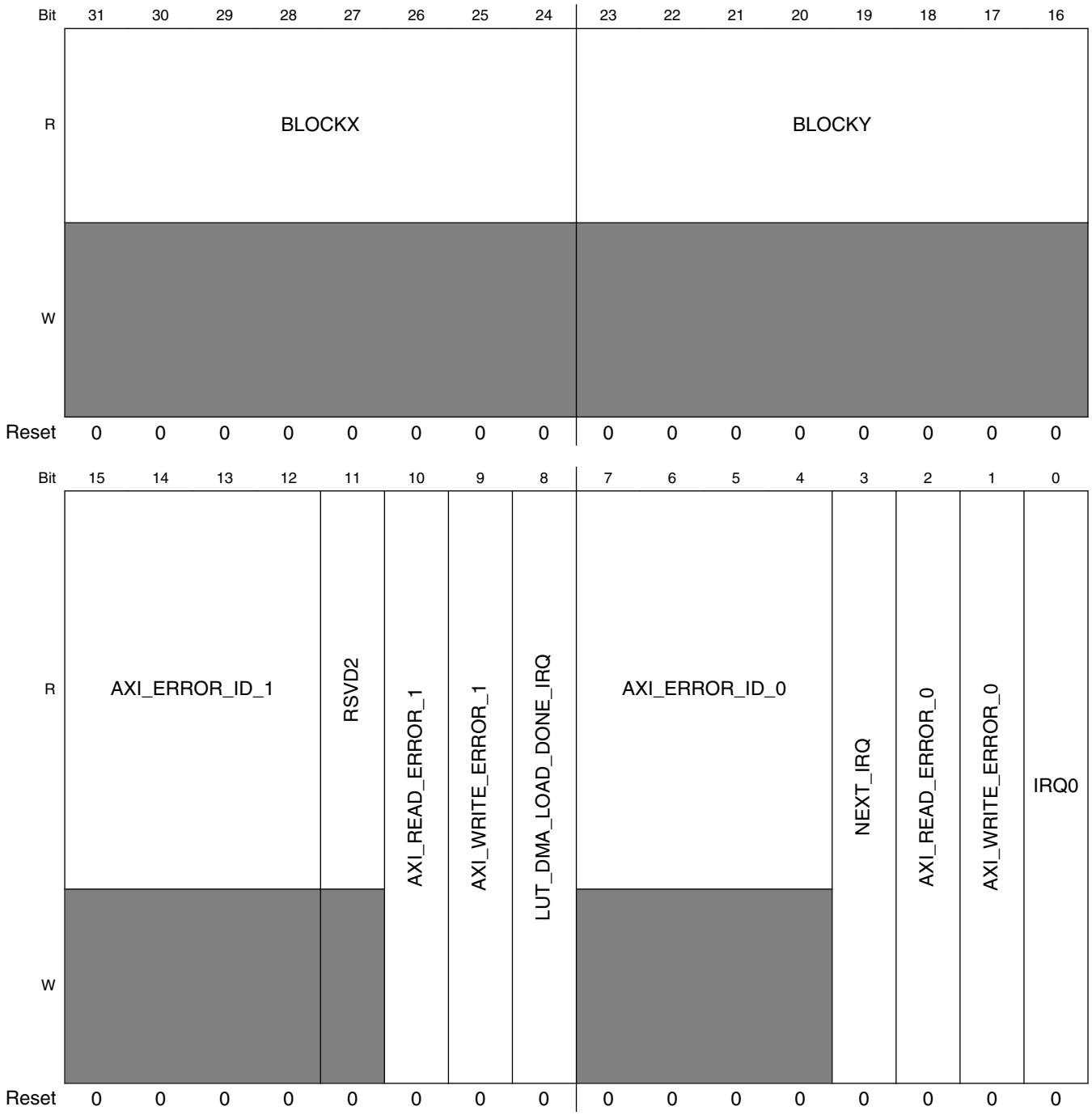
HW_PXP_STAT_TOG: 0x01C

This register provides PXP interrupt status and the current X/Y block coordinate that is being processed.

EXAMPLE

```
REG_STAT_CLR(BM_PXP_STAT_IRQ);  // clear CSC interrupt
```

Address: 3070_0000h base + 10h offset = 3070_0010h



PXP_HW_PXP_STAT field descriptions

Field	Description
31–24 BLOCKX	Indicates the X coordinate of the block currently being rendered.
23–16 BLOCKY	Indicates the X coordinate of the block currently being rendered.
15–12 AXI_ERROR_ID_1	Indicates the AXI1 ID of the failing bus operation.
11 RSVD2	Reserved, always set to zero.
10 AXI_READ_ERROR_1	Indicates PXP encountered an AXI read error and processing has been terminated.
9 AXI_WRITE_ERROR_1	Indicates PXP encountered an AXI write error and processing has been terminated.
8 LUT_DMA_LOAD_DONE_IRQ	Indicates that the LUT DMA transfer has completed.
7–4 AXI_ERROR_ID_0	Indicates the AXI0 ID of the failing bus operation.
3 NEXT_IRQ	Indicates that a command issued with the "Next Command" functionality has been issued and that a new command may be initiated with a write to the PXP_NEXT register.
2 AXI_READ_ERROR_0	Indicates PXP encountered an AXI read error and processing has been terminated.
1 AXI_WRITE_ERROR_0	Indicates PXP encountered an AXI write error and processing has been terminated.
0 IRQ0	Indicates current PXP interrupt status. The IRQ is routed through the pxp_irq when the IRQ_ENABLE bit in the control register is set.

13.6.12.3 Output Buffer Control Register (PXP_HW_PXP_OUT_CTRL)

The OUT_CTRL register contains controls for the Output Buffer.

HW_PXP_OUT_CTRL: 0x020

HW_PXP_OUT_CTRL_SET: 0x024

HW_PXP_OUT_CTRL_CLR: 0x028

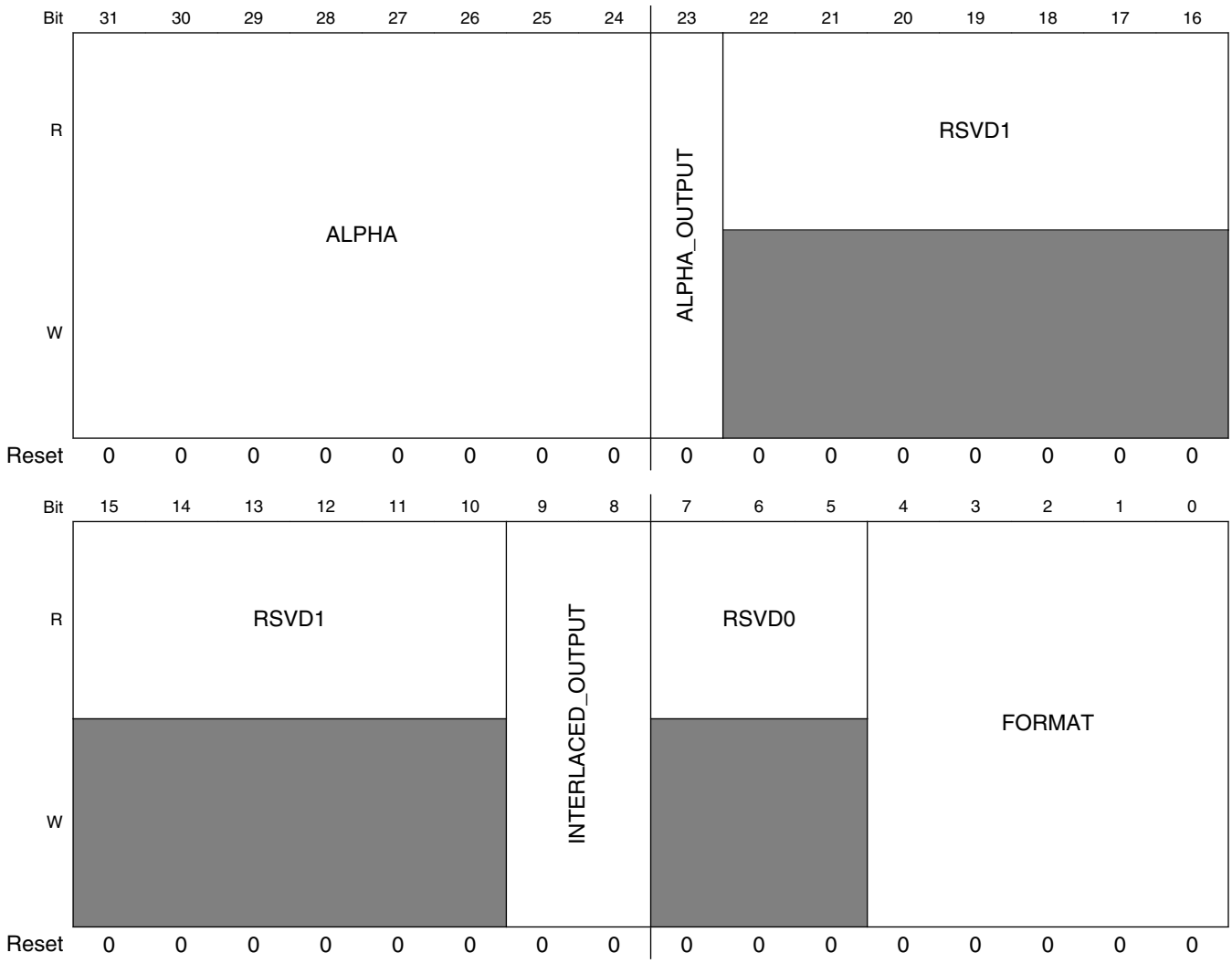
HW_PXP_OUT_CTRL_TOG: 0x02C

Control register to determine OUT buffer processing.

EXAMPLE

```
REG_CTRL_SET(BM_PXP_CTRL_SFTRST);
REG_CTRL_CLR(BM_PXP_CTRL_SFTRST | BM_PXP_CTRL_CLKGATE);
```

Address: 3070_0000h base + 20h offset = 3070_0020h



PXP_HW_PXP_OUT_CTRL field descriptions

Field	Description
31–24 ALPHA	When generating an output buffer with an alpha component, the value in this field will be used when enabled to override the alpha passed through the pixel data pipeline.
23 ALPHA_ OUTPUT	Indicates that alpha component in output buffer pixels should be overwritten by REG_OUT_CTRL[ALPHA] register. If 0, retain their alpha value from the computed alpha for that pixel.
22–10 RSVD1	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_OUT_CTRL field descriptions (continued)

Field	Description
9–8 INTERLACED_ OUTPUT	Determines how the PXP writes it's output data. Output interlacing should not be used in conjunction with input interlacing. Splitting frames into fields is most efficient using output interlacing. 2-plane output formats AND interlaced output is NOT supported. 0x0 **PROGRESSIVE** — All data written in progressive format to the OUTBUF Pointer. 0x1 **FIELD0** — Interlaced output: only data for field 0 is written to the OUTBUF Pointer. 0x2 **FIELD1** — Interlaced output: only data for field 1 is written to the OUTBUF2 Pointer. 0x3 **INTERLACED** — Interlaced output: data for field 0 is written to OUTBUF and data for field 1 is written to OUTBUF2.
7–5 RSVD0	Reserved, always set to zero.
FORMAT	Output framebuffer format. The UV byte lanes are synonymous with CbCr byte lanes for YUV output pixel formats. For example, the YUV2P420 format should be selected when the output is YCbCr 2-plane 420 output format. 0x0 **ARGB8888** — 32-bit pixels 0x4 **RGB888** — 32-bit pixels (unpacked 24-bit pixel in 32 bit DWORD.) 0x5 **RGB888P** — 24-bit pixels (packed 24-bit format) 0x8 **ARGB1555** — 16-bit pixels 0x9 **ARGB4444** — 16-bit pixels 0xC **RGB555** — 16-bit pixels 0xD **RGB444** — 16-bit pixels 0xE **RGB565** — 16-bit pixels 0x10 **YUV1P444** — 32-bit pixels (1-plane XYUV unpacked) 0x12 **UYVY1P422** — 16-bit pixels (1-plane U0,Y0,V0,Y1 interleaved bytes) 0x13 **VYUY1P422** — 16-bit pixels (1-plane V0,Y0,U0,Y1 interleaved bytes) 0x14 **Y8** — 8-bit monochrome pixels (1-plane Y luma output) 0x15 **Y4** — 4-bit monochrome pixels (1-plane Y luma, 4 bit truncation) 0x18 **YUV2P422** — 16-bit pixels (2-plane UV interleaved bytes) 0x19 **YUV2P420** — 16-bit pixels (2-plane UV) 0x1A **YVU2P422** — 16-bit pixels (2-plane VU interleaved bytes) 0x1B **YVU2P420** — 16-bit pixels (2-plane VU)

13.6.12.4 Output Frame Buffer Pointer (PXP_HW_PXP_OUT_BUF)

Output Framebuffer Pointer. This register points to the beginning of the output frame buffer. This pointer is used for progressive format and field 0 when generating interlaced output.

This register is used by the logic to point to the current output location for the output frame buffer.

EXAMPLE

```
REG_OUT_BUF_WR( buffer );
```

Address: 3070_0000h base + 30h offset = 3070_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_OUT_BUF field descriptions

Field	Description
ADDR	Current address pointer for the output frame buffer. The address can have any byte alignment. 64B alignment is recommended for optimal performance.

13.6.12.5 Output Frame Buffer Pointer #2 (PXP_HW_PXP_OUT_BUF2)

Output Framebuffer Pointer #2. This register points to the beginning of the output frame buffer for either field 1 when generating interlaced output or for the UV buffer when in YUV 2-plane output modes. Both interlaced output AND 2-plane output modes are not supported in a single PXP operation. This register is NOT used as the pointer to the 2nd buffer when in LCDIF_HANDSHAKE mode.

This register is used by the logic to point to the current output location for the field 1 or UV output frame buffer.

EXAMPLE

```
REG_OUT_BUF_WR( field0 );  // buffer for interlaced field 0
REG_OUT_BUF2_WR( field1 ); // buffer for interlaced field 1
```

Address: 3070_0000h base + 40h offset = 3070_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_OUT_BUF2 field descriptions

Field	Description
ADDR	Current address pointer for the output frame buffer. The address can have any byte alignment. 64B alignment is recommended for optimal performance.

13.6.12.6 Output Buffer Pitch (PXP_HW_PXP_OUT_PITCH)

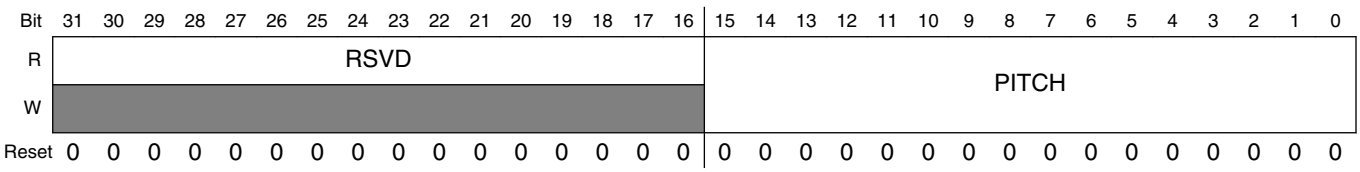
This register contains the output buffer pitch in bytes.

Any byte value will indicate the vertical pitch. This value will be used in output pixel address calculations.

EXAMPLE

```
REG_OUT_PITCH_WR( 68 * 4 );  // The output buffer pitch is 68 pixels times 32 bits per pixel
```

Address: 3070_0000h base + 50h offset = 3070_0050h



PXP_HW_PXP_OUT_PITCH field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
PITCH	Indicates the number of bytes in memory between two vertically adjacent pixels.

13.6.12.7 Output Surface Lower Right Coordinate (PXP_HW_PXP_OUT_LRC)

This register contains the size, or lower right coordinate, of the output buffer NOT rotated. It is implied that the upper left coordinate of the output surface is always [0,0]. When rotating the framebuffer, the PXP will automatically swap the X/Y, or WIDTH/HEIGHT, to accomodate the rotated size.

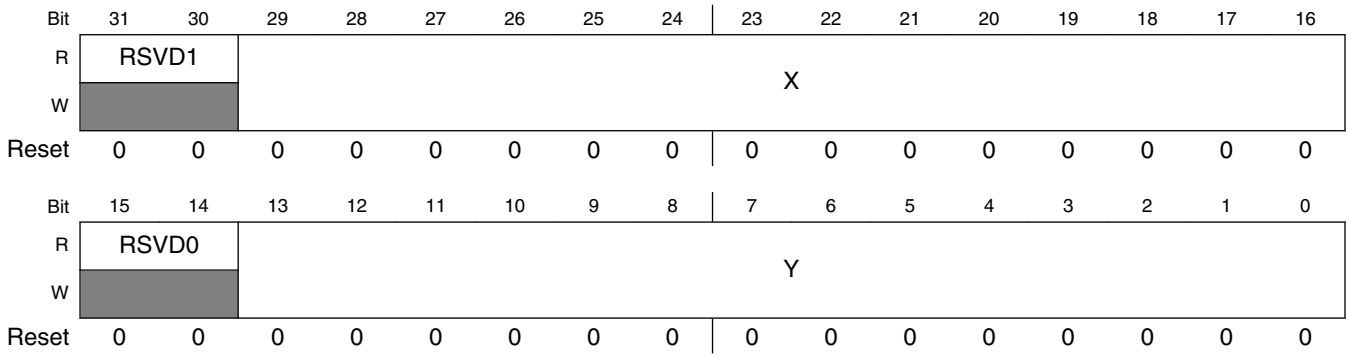
This register sets the size of the output frame buffer in pixels, not blocks. The frame buffer need not be a multiple of NxN pixels. Partial blocks will be written for output frame buffer sizes that are not divisable by N pixels in either dimension.

EXAMPLE

```
REG_OUT_LRC[X]=319;  // set width of output frame buffer to 320 pixels
REG_OUT_LRC[Y]=243;  // set height of output frame buffer to 244 pixels which is not
divisable by block size N

REG_OUT_LRC_WR( BF_PXP_OUT_LRC_X(319)  |  BF_PXP_OUT_LRC_Y(243) );
```

Address: 3070_0000h base + 60h offset = 3070_0060h



PXP_HW_PXP_OUT_LRC field descriptions

Field	Description
31–30 RSVD1	Reserved, always set to zero.
29–16 X	Indicates number of horizontal PIXELS in the output surface (non-rotated). The output buffer pixel width minus 1 should be programmed. The image size is not required to be a multiple of 8 pixels. The PXP will clip the pixel output at this boundary.
15–14 RSVD0	Reserved, always set to zero.
Y	Indicates the number of vertical PIXELS in the output surface (non-rotated). The output buffer pixel height minus 1 should be programmed. The image size is not required to be a multiple of 8 pixels. The PXP will clip the pixel output at this boundary.

13.6.12.8 Processed Surface Upper Left Coordinate (PXP_HW_PXP_OUT_PS_ULC)

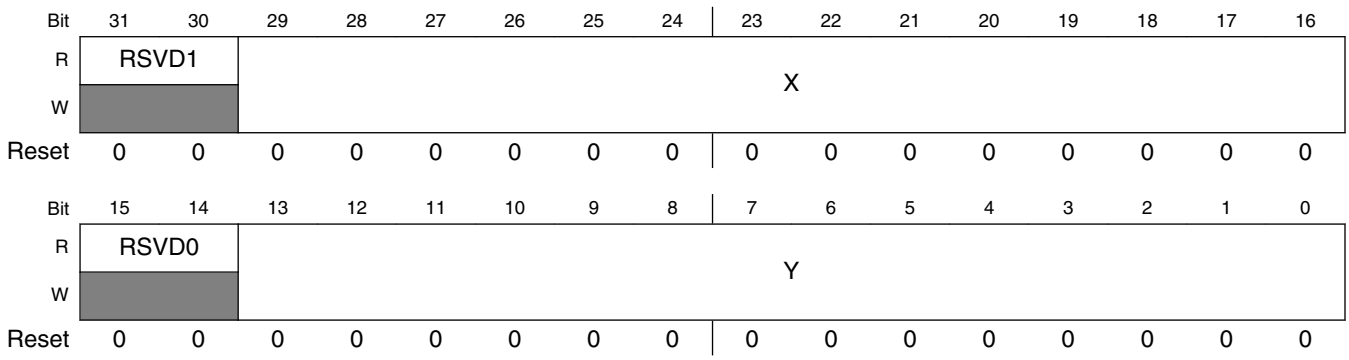
This register contains the upper left pixel coordinate for the Processed Surface in the OUTPUT buffer.

This register contains the upper left coordinate of the Processed Surface in the output frame buffer (in pixels). Values that are within the REG_OUT_LRC[X,Y] extents are valid. The lowest valid value for these fields is 0,0. If the value of the REG_OUT_PS_ULC is greater than the REG_OUT_LRC, then no PS pixels will be fetched from memory, but only REG_PS_BACKGROUND pixels will be processed by the PS engine. Pixel locations that are greater than or equal to the PS upper left coordinates, less than or equal to the PS lower right coordinates, and within the REG_OUT_LRC extents will use the PS to render pixels into the output buffer.

EXAMPLE

```
REG_OUT_PS_ULC_WR(0,0x0002_0002); // Processed Surface upper left coordinate at (X,Y) =
2,2.  The PS surface will not effect pixels in the first and second row and column of the
output buffer.
```

Address: 3070_0000h base + 70h offset = 3070_0070h



PXP_HW_PXP_OUT_PS_ULC field descriptions

Field	Description
31–30 RSVD1	Reserved, always set to zero.
29–16 X	This field indicates the upper left X-coordinate (in pixels) of the processed surface (PS) in the output buffer.
15–14 RSVD0	Reserved, always set to zero.
Y	This field indicates the upper left Y-coordinate (in pixels) of the processed surface in the output buffer.

13.6.12.9 Processed Surface Lower Right Coordinate (PXP_HW_PXP_OUT_PS_LRC)

This register contains the lower right extent for the Processed Surface in the OUTPUT buffer.

This register contains the lower right coordinate of the Processed Surface in the output frame buffer (in pixels). Values that are within the REG_OUT_LRC[X,Y] extents are valid. The lowest valid value for these fields is 0,0. Pixel locations that are greater than or equal to the PS upper left coordinates, less than or equal to the PS lower right coordinates, and within the REG_OUT_LRC extents will use the PS to render pixels into the output buffer.

EXAMPLE

```
REG_OUT_PS_ULC_WR(0,0x03FF_03FF); // With this UL/LR pair of pixel coordinates, only one
pixel at OUT[X,Y]=1023,1023 will use the PS to contribute to its value.
REG_OUT_PS_LRC_WR(0,0x03FF_03FF);
```

Address: 3070_0000h base + 80h offset = 3070_0080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1								X							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0								Y							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_OUT_PS_LRC field descriptions

Field	Description
31–30 RSVD1	Reserved, always set to zero.
29–16 X	This field indicates the lower right X-coordinate (in pixels) of the processed surface (PS) in the output frame buffer.
15–14 RSVD0	Reserved, always set to zero.
Y	This field indicates the lower right Y-coordinate (in pixels) of the processed surface in the output frame buffer.

13.6.12.10 Alpha Surface Upper Left Coordinate (PXP_HW_PXP_OUT_AS_ULC)

This register contains the upper left location for the Alpha Surface in the output buffer.

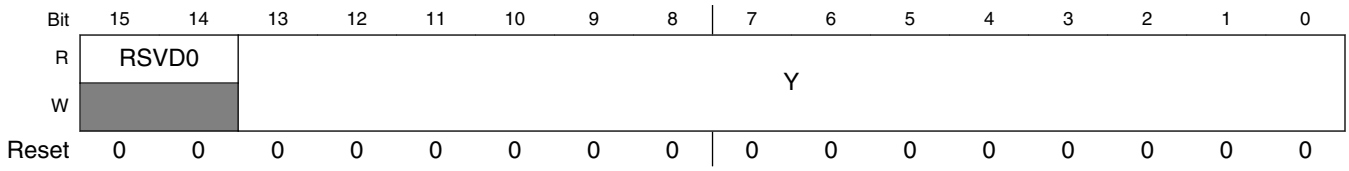
This register contains the upper left coordinate of AS in the output frame buffer (in pixels). Values that are within the REG_OUT_LRC[X,Y] extents are valid. The lowest valid value for these fields is 0,0. Pixel locations that are greater than or equal to the upper left coordinates will use the AS to render pixels in the output buffer.

EXAMPLE

```
REG_OUT_AS_ULC_WR(0,0x0001_0001); // Alpha Surface upper left coordinate at (X,Y) = 1,1.
The AS surface will not effect pixels in the first row or first column of the output buffer.
```

Address: 3070_0000h base + 90h offset = 3070_0090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1								X							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



PXP_HW_PXP_OUT_AS_ULC field descriptions

Field	Description
31–30 RSVD1	Reserved, always set to zero.
29–16 X	This field indicates the upper left X-coordinate (in pixels) of the alpha surface (AS) in the output frame buffer.
15–14 RSVD0	Reserved, always set to zero.
Y	This field indicates the upper left Y-coordinate (in pixels) of the alpha surface in the output frame buffer.

13.6.12.11 Alpha Surface Lower Right Coordinate (PXP_HW_PXP_OUT_AS_LRC)

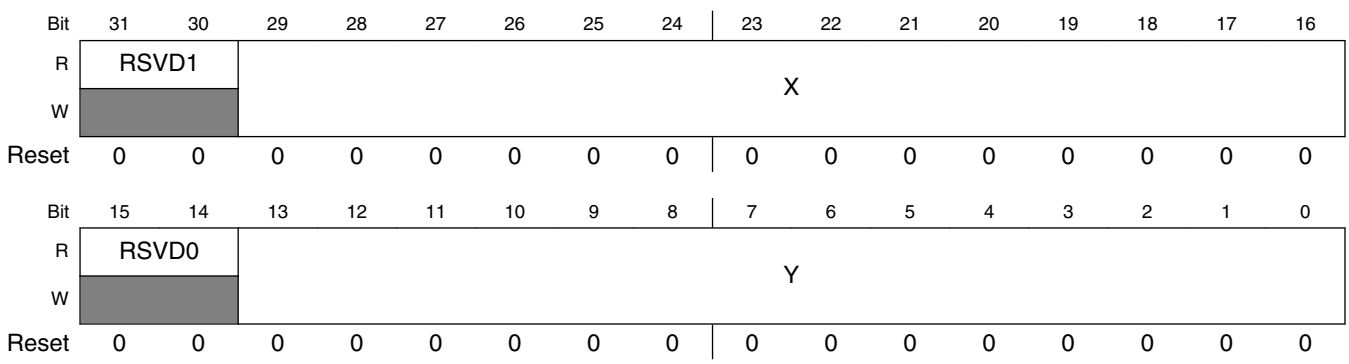
This register contains the lower right extent for Alpha Surface in the output buffer.

This register contains the lower right coordinate of AS in the output frame buffer (in pixels). Values that are within the REG_OUT_LRC[X,Y] extents are valid. The lowest valid value for these fields is 0,0. Pixel locations that are less than or equal to the lower right coordinates will use the AS to render pixels in the output buffer.

EXAMPLE

```
REG_AS_LRC_WR(0,0x03FF_03FF); // Alpha Surface lower right coordinate at (X,Y) = 1023,1023.
```

Address: 3070_0000h base + A0h offset = 3070_00A0h



PXP_HW_PXP_OUT_AS_LRC field descriptions

Field	Description
31–30 RSVD1	Reserved, always set to zero.
29–16 X	This field indicates the lower right X-coordinate (in pixels) of the alpha surface (AS) in the output frame buffer.
15–14 RSVD0	Reserved, always set to zero.
Y	This field indicates the lower right Y-coordinate (in pixels) of the alpha surface in the output frame buffer.

13.6.12.12 Processed Surface (PS) Control Register (PXP_HW_PXP_PS_CTRL)

The PS_CTRL register contains controls for the Processed Surface Buffer.

HW_PXP_PS_CTRL: 0x0B0

HW_PXP_PS_CTRL_SET: 0x0b4

HW_PXP_PS_CTRL_CLR: 0x0B8

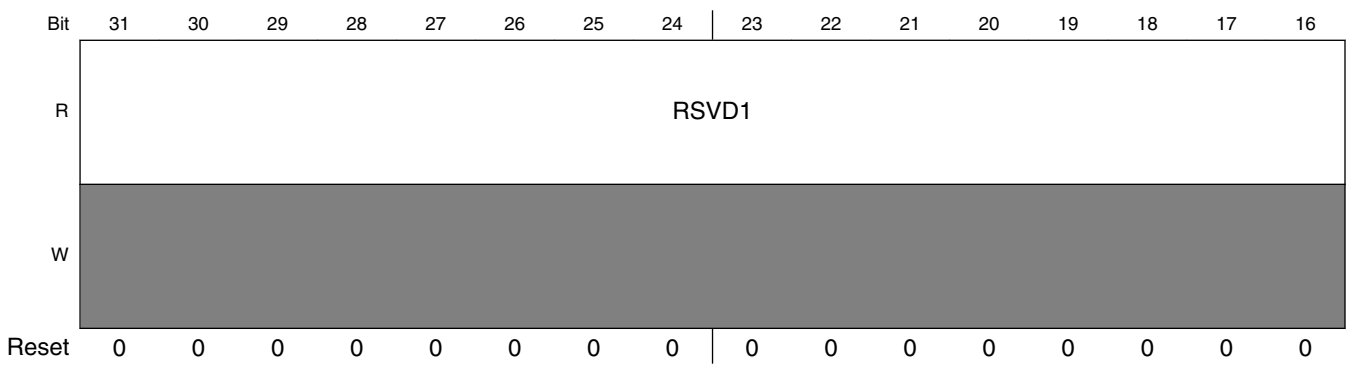
HW_PXP_PS_CTRL_TOG: 0x0BC

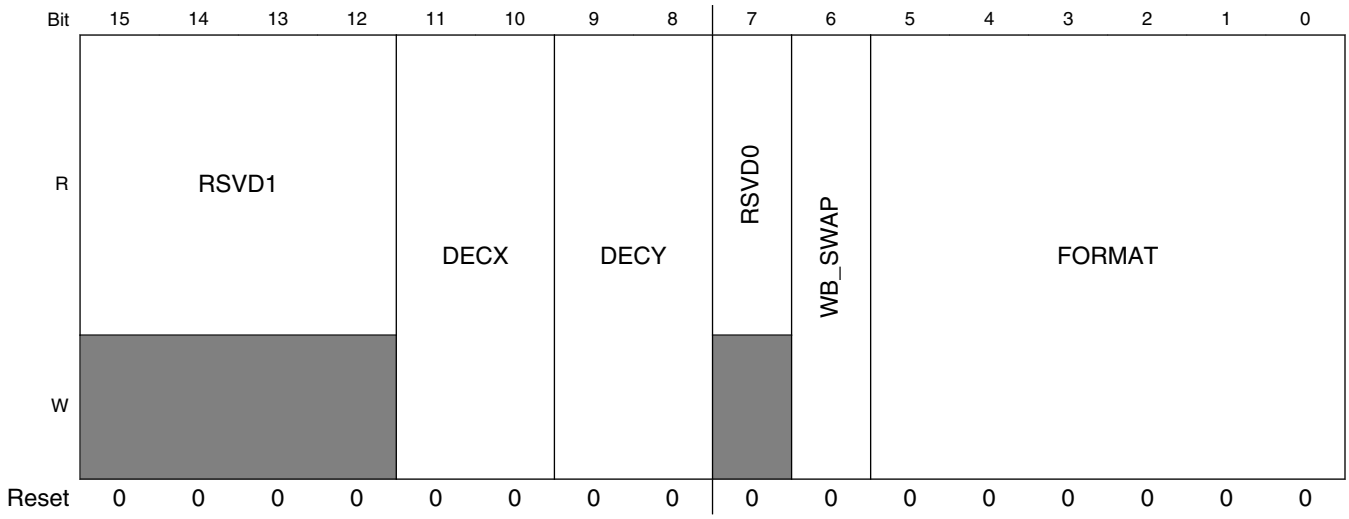
Control register to determine PS buffer processing.

EXAMPLE

```
REG_CTRL_SET(BM_PXP_CTRL_SFTRST);
REG_CTRL_CLR(BM_PXP_CTRL_SFTRST | BM_PXP_CTRL_CLKGATE);
```

Address: 3070_0000h base + B0h offset = 3070_00B0h





PXP_HW_PXP_PS_CTRL field descriptions

Field	Description
31–12 RSVD1	Reserved, always set to zero.
11–10 DECX	Horizontal pre decimation filter control. 0x0 **DISABLE** — Disable pre-decimation filter. 0x1 **DECX2** — Decimate PS by 2. 0x2 **DECX4** — Decimate PS by 4. 0x3 **DECX8** — Decimate PS by 8.
9–8 DECY	Verticle pre decimation filter control. 0x0 **DISABLE** — Disable pre-decimation filter. 0x1 **DECY2** — Decimate PS by 2. 0x2 **DECY4** — Decimate PS by 4. 0x3 **DECY8** — Decimate PS by 8.
7 RSVD0	Reserved, always set to zero.
6 WB_SWAP	Swap bytes in words. For each 16 bit word, the two bytes will be swapped.
FORMAT	PS buffer format. To select between YUV and YCbCr formats, see bit 31 of the CSC1_COEF0 register. 0x4 **RGB888** — 32-bit pixels (unpacked 24-bit format) 0xC **RGB555** — 16-bit pixels 0xD **RGB444** — 16-bit pixels 0xE **RGB565** — 16-bit pixels 0x10 **YUV1P444** — 32-bit pixels (1-plane XYUV unpacked) 0x12 **UYVY1P422** — 16-bit pixels (1-plane U0,Y0,V0,Y1 interleaved bytes) 0x13 **VYUY1P422** — 16-bit pixels (1-plane V0,Y0,U0,Y1 interleaved bytes) 0x14 **Y8** — 8-bit monochrome pixels (1-plane Y luma output) 0x15 **Y4** — 4-bit monochrome pixels (1-plane Y luma, 4 bit truncation) 0x18 **YUV2P422** — 16-bit pixels (2-plane UV interleaved bytes) 0x19 **YUV2P420** — 16-bit pixels (2-plane UV)

Table continues on the next page...

PXP_HW_PXP_PS_CTRL field descriptions (continued)

Field	Description
	0x1A **YVU2P422** — 16-bit pixels (2-plane VU interleaved bytes) 0x1B **YVU2P420** — 16-bit pixels (2-plane VU) 0x1E **YUV422** — 16-bit pixels (3-plane format) 0x1F **YUV420** — 16-bit pixels (3-plane format)

13.6.12.13 PS Input Buffer Address (PXP_HW_PXP_PS_BUF)

PS Input Buffer Address. This should be programmed to the starting address of the RGB data or Y (luma) data for the PS plane.

This register contains the pointer to the Luma/RGB buffer. If the application requires an offset into the PS buffer, then this address can be set so that the desired offset is achieved. Any byte address is valid. For best performance, 64B alignment is recommended.

EXAMPLE

```
REG_PS_BUF_WR(image_rgb); // RGB image
REG_PS_BUF_WR(image_y);   // Y (luma) image data
REG_PS_UBUF_WR(image_u);  // U (Cb) image data
REG_PS_VBUF_WR(image_v);  // V (Cr) image data
```

Address: 3070_0000h base + C0h offset = 3070_00C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_PS_BUF field descriptions

Field	Description
ADDR	Address pointer for the PS RGB or Y (luma) input buffer.

13.6.12.14 PS U/Cb or 2 Plane UV Input Buffer Address (PXP_HW_PXP_PS_UBUF)

PS Chroma (U/Cb/UV) Input Buffer Address. This register points to the beginning of the PS U/Cb input buffer. In two plane operation, this register points to the beginning of the PS UV chroma input buffer.

This register contains the pointer to the Chroma U/Cb or 2 plane UV buffer. This register is unused when processing 1-plane buffer formats. If the application requires an offset into the PS buffer, then this address can be set so that the desired offset is achieved. Any byte address is valid. For best performance, 64B alignment is recommended.

EXAMPLE

```
REG_PS_BUF_WR(image_y);   // Y (luma) image data
REG_PS_UBUF_WR(image_u);  // U (Cb) image data
REG_PS_VBUF_WR(image_v);  // V (Cr) image data
```

Address: 3070_0000h base + D0h offset = 3070_00D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_PS_UBUF field descriptions

Field	Description
ADDR	Address pointer for the PS U/Cb or 2 plane UV Chroma input buffer.

13.6.12.15 PS V/Cr Input Buffer Address (PXP_HW_PXP_PS_VBUF)

PS Chroma (V/Cr) Input Buffer Address. This register points to the beginning of the PS V/Cr input buffer. In one or two plane operation, this register is not used. In monochrome modes Y8 and Y4, the low 16 bits are used as the U/V data in the datapath instead of sourcing U/V data from external buffers. In this case, it represents a fixed value for U/V data.

This register contains the pointer to the Chroma V/Cr buffer. For Y8/Y4 modes, the low 16 bits are used as the monochrome U and V values in the data path. Bits [15:8] represent the U data byte, and bits]7:0] represent the V data byte. Other than with Y8/Y4 input buffer formats, this register is unused when processing 1 or 2-plane buffer formats. If the application requires an offset into the PS buffer, then this address can be set so that the desired offset is achieved. Any byte address is valid. For best performance, 64B alignment is recommended.

EXAMPLE

```
REG_PS_BUF_WR(image_y);   // Y (luma) image data
REG_PS_UBUF_WR(image_u);  // U (Cb) image data
REG_PS_VBUF_WR(image_v);  // V (Cr) image data
```

Address: 3070_0000h base + E0h offset = 3070_00E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_PS_VBUF field descriptions

Field	Description
ADDR	Address pointer for the PS V/Cr Chroma input buffer.

13.6.12.16 Processed Surface Pitch (PXP_HW_PXP_PS_PITCH)

This register contains the processed surface pitch in bytes.

Any byte value will indicate the vertical pitch of the PS source frame buffer. This value will be used in PS pixel address calculations. This value has no relation to the UL and LR registers. It specifies how many bytes are between two vertically adjacent pixels in the input PS surface. For multi-plane formats, the Y buffer pitch should be programmed. For 2-plane YUV422, the UV pitch is the same as the Y pitch. For 3-plane YUV422, the U and V pitch is 1/2 the Y pitch. For 2-plane YUV420, the UV pitch is 1/2 the Y pitch. For 3-plane YUV420, the U and V pitch is 1/4 the Y pitch. All source buffers should comply with these U and V resolution reductions with respect to their Y source buffers.

EXAMPLE

```
REG_PS_PITCH_WR( 64 * 4 );  // The output buffer pitch is 64 pixels times 32 bits per pixel
```

Address: 3070_0000h base + F0h offset = 3070_00F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																PITCH															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_PS_PITCH field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
PITCH	Indicates the number of bytes in memory between two vertically adjacent pixels.

13.6.12.17 PS Background Color (PXP_HW_PXP_PS_BACKGROUND_0)

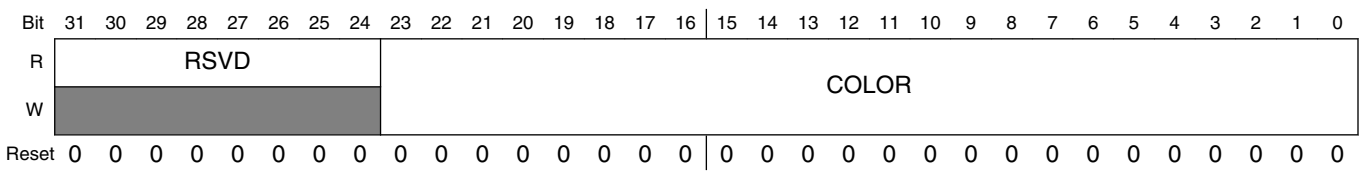
PS Background Pixel Color. This register provides a pixel value used when processing pixels outside of the region specified by the PS Coordinate registers. This value can effectively be used to set the color of the letterboxing region around the PS image. This value is used by the Alpha A block.

This register contains a pixel value to be used for any PS pixels that fall outside the PS extents. This is effectively a background or letterbox color. The CSC1 control and datapath pixel format should be considered when selecting the background color.

EXAMPLE

```
REG_PS_BACKGROUND_WR(0x00000000);  // letterbox is black
REG_PS_BACKGROUND_WR(0x00800000);  // letterbox is dark red
REG_PS_BACKGROUND_WR(0x00008000);  // letterbox is dark green
REG_PS_BACKGROUND_WR(0x00000080);  // letterbox is dark blue
```

Address: 3070_0000h base + 100h offset = 3070_0100h



PXP_HW_PXP_PS_BACKGROUND_0 field descriptions

Field	Description
31–24 RSVD	Reserved, always set to zero.
COLOR	Background color (in 24bpp format) for any pixels not within the buffer range specified by the PS ULC/ LRC.

13.6.12.18 PS Scale Factor Register (PXP_HW_PXP_PS_SCALE)

PS Scale Factor. This register provides the scale factor for the PS buffer.

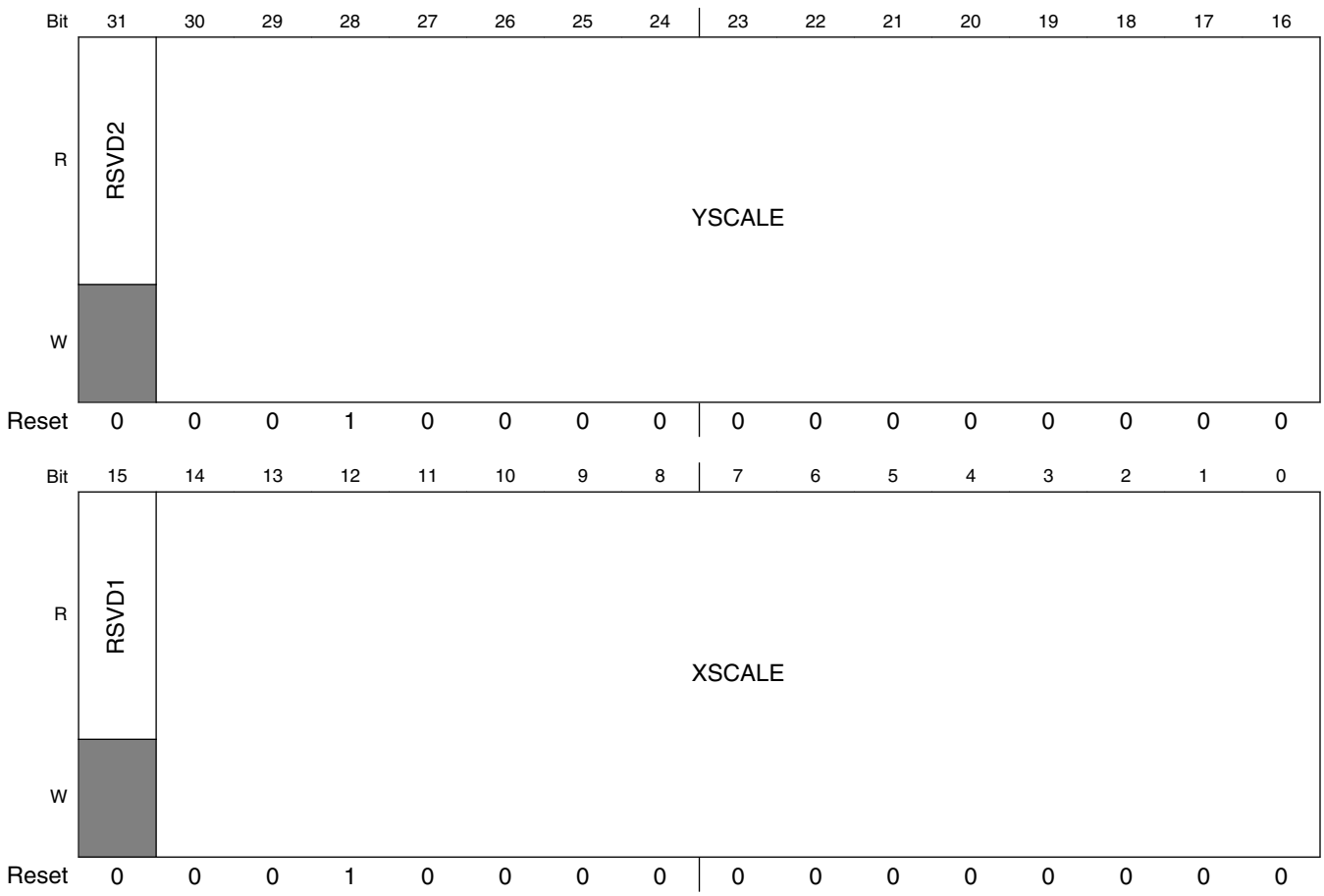
The maximum down scaling factor is 1/2 such that the output image in either axis is 1/2 the size of the source. The maximum up scaling factor is 2^12 for either axis. The reciprocal of the scale factor should be loaded into this register. To reduce the PS buffer by a factor of two in the output frame buffer, a value of 10.0000_0000_0000 should be

loaded into this register. To scale up by a factor of 4, the value of 1/4, or 00.0100_0000_0000, should be loaded into this register. To scale up by 8/5, the value of 00.1010_0000_0000 should be loaded.

EXAMPLE

```
REG_PS_SCALE_WR(0x10001000);  // 1:1  scaling (0x1.000)
REG_PS_SCALE_WR(0x08000800);  // 2x   scaling (0x0.800)
REG_PS_SCALE_WR(0x20002000);  // 1/2x scaling (0x2.000)
```

Address: 3070_0000h base + 110h offset = 3070_0110h



PXP_HW_PXP_PS_SCALE field descriptions

Field	Description
31 RSVD2	Reserved, always set to zero.
30–16 YSCALE	This is a two bit integer and 12 bit fractional representation (##.####_####_####) of the Y scaling factor for the PS source buffer. The maximum value programmed should be 2 since scaling down by a factor greater than 2 is not supported with the bilinear filter. Decimation and the bilinear filter should be used together to achieve scaling by more than a factor of 2.
15 RSVD1	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_PS_SCALE field descriptions (continued)

Field	Description
XSCALE	This is a two bit integer and 12 bit fractional representation (##.####_####_####) of the X scaling factor for the PS source buffer. The maximum value programmed should be 2 since scaling down by a factor greater than 2 is not supported with the bilinear filter. Decimation and the bilinear filter should be used together to achieve scaling by more than a factor of 2.

13.6.12.19 PS Scale Offset Register (PXP_HW_PXP_PS_OFFSET)

PS Scale Offset. This register provides the initial scale offset for the PS buffer.

The X and Y offset provides the ability to access the source image with a per sub-pixel granularity. This provides the capability to use all source pixels to effect the output PS image. The fixed offset values can be used for sub-pixel adjustments in the bilinear scaling filter. For example, when scaling an image down by a factor of 2, an initial offset of 0x0 would result in sub-sampling every other pixel. If a fixed offset of 0x800 (1/2), all pixels are used in scaling the final output pixel value. In this case, the first output pixel would be the sum of (1/2*P0) + (1/2*P1). This fixed offset is applied after the decimation filter stage, and before the bilinear filter stage.

EXAMPLE

```
REG_PS_SCALE_WR(0x2000_2000);   // 1/2x scaling (0x2.000)
REG_PS_OFFSET_WR(0x0800_0800);  // half-pixel offset in both X and Y to ensure averaging
versus pixel decimation
```

Address: 3070_0000h base + 120h offset = 3070_0120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2				YOFFSET												RSVD1				XOFFSET											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_PS_OFFSET field descriptions

Field	Description
31–28 RSVD2	Reserved, always set to zero.
27–16 YOFFSET	This is a 12 bit fractional representation (0.####_####_####) of the Y scaling offset. This represents a fixed pixel offset which gets added to the scaled address to determine source data for the scaling engine.
15–12 RSVD1	Reserved, always set to zero.
XOFFSET	This is a 12 bit fractional representation (0.####_####_####) of the X scaling offset. This represents a fixed pixel offset which gets added to the scaled address to determine source data for the scaling engine.

13.6.12.20 PS Color Key Low (PXP_HW_PXP_PS_CLRKEYLOW_0)

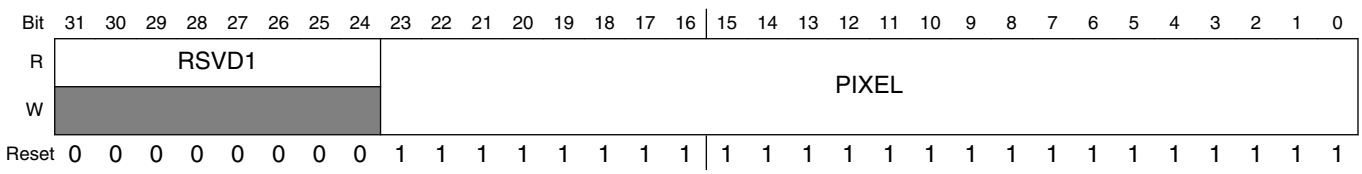
This register contains the color key low value for the PS buffer. This value is used by the Alpha A block.

When processing an image, if the PXP finds a pixel in the PS buffer with a color that falls in the range between REG_PS_CLRKEYLOW and REG_PS_CLRKEYHIGH, it will insert the pixel from the AS channel. If the current AS pixel is letterboxed or if the AS also matches its colorkey range, the REG_PS_BACKGROUND color is passed down the pixel pipeline.

EXAMPLE

```
                                    // colorkey values between
REG_PS_CLRKEYLOW_WR (0x008000); // medium green and
REG_PS_CLRKEYHIGH_WR(0x00FF00); // light green
```

Address: 3070_0000h base + 130h offset = 3070_0130h



PXP_HW_PXP_PS_CLRKEYLOW_0 field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	Low range of color key applied to PS buffer. To disable PS colorkeying, set the low colorkey to 0xFFFFFF and the high colorkey to 0x000000.

13.6.12.21 PS Color Key High (PXP_HW_PXP_PS_CLRKEYHIGH_0)

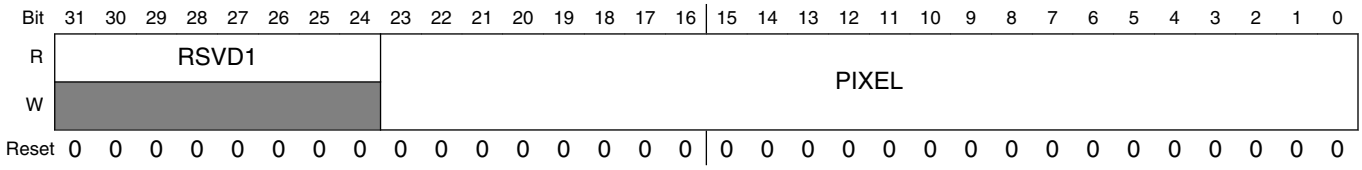
This register contains the color key high value for the PS buffer. This value is used by the Alpha A block.

When processing an image, if the PXP finds a pixel in the PS buffer with a color that falls in the range between REG_PS_CLRKEYLOW and REG_PS_CLRKEYHIGH, it will insert the pixel from the AS channel. If the current AS pixel is letterboxed or if the AS also matches its colorkey range, the REG_PS_BACKGROUND color is passed down the pixel pipeline.

EXAMPLE

```
                                        // colorkey values between
REG_PS_CLRKEYLOW_WR (0x008000); // medium green and
REG_PS_CLRKEYHIGH_WR(0x00FF00); // light green
```

Address: 3070_0000h base + 140h offset = 3070_0140h



PXP_HW_PXP_PS_CLRKEYHIGH_0 field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	High range of color key applied to PS buffer. To disable PS colorkeying, set the low colorkey to 0xFFFFFF and the high colorkey to 0x000000.

13.6.12.22 Alpha Surface Control (PXP_HW_PXP_AS_CTRL)

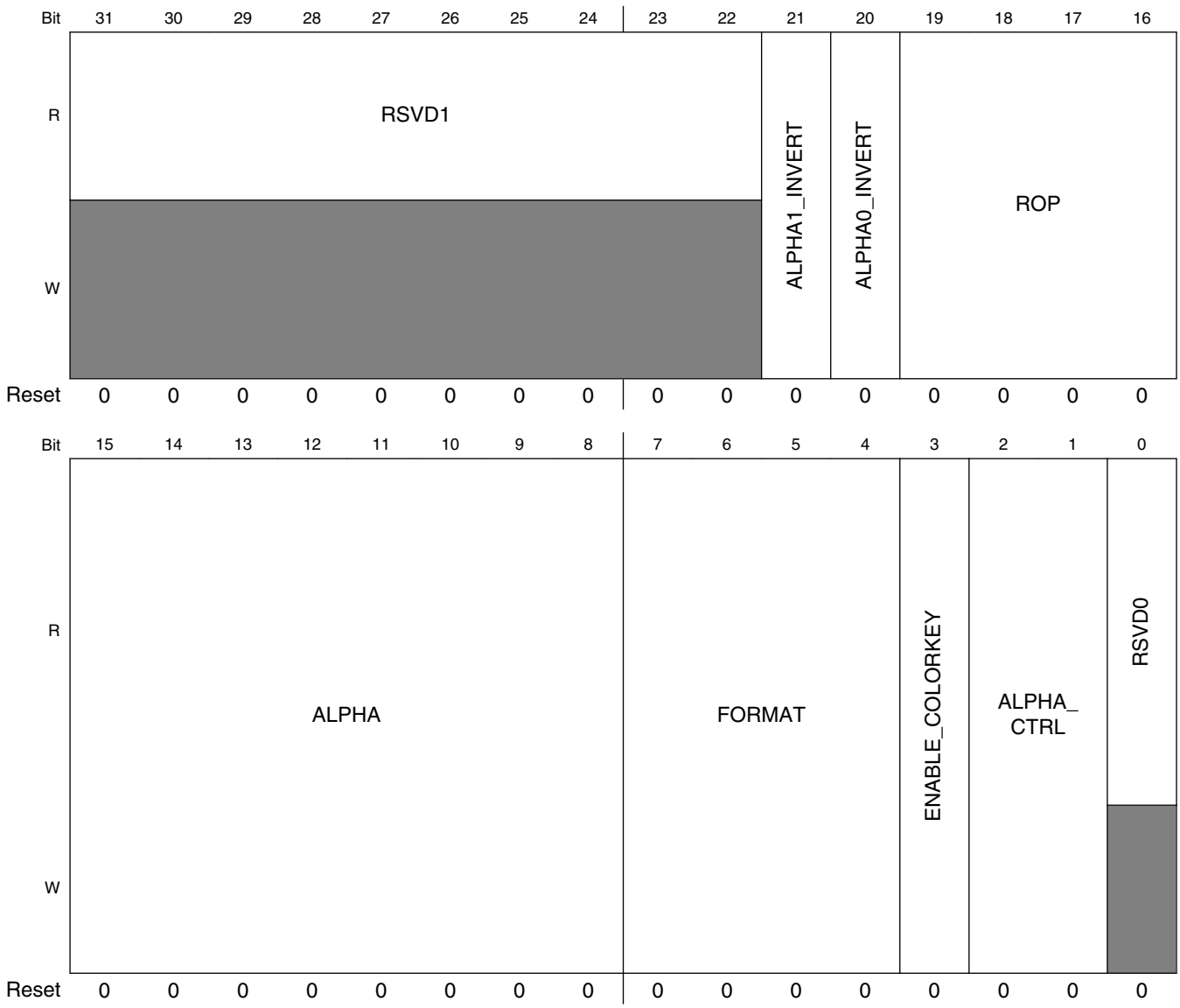
This register contains buffer control for the Alpha Surface 0 input buffer.

Control register to determine AS buffer processing.

EXAMPLE

```
u32 asparam;
      asparam  = BF_PXP_ASPARAM_ENABLE    (1);
      asparam |= BF_PXP_ASPARAM_ALPHA_CTRL(BV_PXP_ASPARAM_ALPHA_CTRL__ROPs);
      asparam |= BF_PXP_ASPARAM_FORMAT    (BV_PXP_ASPARAM_FORMAT__ARGB8888);
      asparam |= BF_PXP_ASPARAM_ROP       (BV_PXP_ASPARAM_ROP__XORAS);
REG_ASPARAM_WR(0,asparam);  // enable alpha surface to perform XOR ROP using RGB8888 AS
pixel format
```

Address: 3070_0000h base + 150h offset = 3070_0150h



PXP_HW_PXP_AS_CTRL field descriptions

Field	Description
31–22 RSVD1	Reserved, always set to zero.
21 ALPHA1_ INVERT	Setting this bit to logic 0 will not alter the alpha1 value. A logic 1 will invert the alpha1 value and apply (1-alpha) for image composition.
20 ALPHA0_ INVERT	Setting this bit to logic 0 will not alter the alpha0 value. A logic 1 will invert the alpha0 value and apply (1-alpha) for image composition.
19–16 ROP	Indicates a raster operation to perform when enabled. Raster operations are enabled through the ALPHA_CTRL field. 0x0 **MASKAS** — AS AND PS

Table continues on the next page...

PXP_HW_PXP_AS_CTRL field descriptions (continued)

Field	Description
	0x1 **MASKNOTAS** — nAS AND PS 0x2 **MASKASNOT** — AS AND nPS 0x3 **MERGEAS** — AS OR PS 0x4 **MERGENOTAS** — nAS OR PS 0x5 **MERGEASNOT** — AS OR nPS 0x6 **NOTCOPYAS** — nAS 0x7 **NOT** — nPS 0x8 **NOTMASKAS** — AS NAND PS 0x9 **NOTMERGEAS** — AS NOR PS 0xA **XORAS** — AS XOR PS 0xB **NOTXORAS** — AS XNOR PS
15–8 ALPHA	Alpha modifier used when the ALPHA_MULTIPLY or ALPHA_OVERRIDE values are programmed in REG_AS_CTRL[ALPHA_CTRL]. The output alpha value will either be replaced (ALPHA_OVERRIDE) or scaled (ALPHA_MULTIPLY) when selected.
7–4 FORMAT	Indicates the input buffer format for AS. 0x0 **ARGB8888** — 32-bit pixels with alpha 0x1 **RGBA8888** — 32-bit pixels with alpha 0x4 **RGB888** — 32-bit pixels without alpha (unpacked 24-bit format) 0x8 **ARGB1555** — 16-bit pixels with alpha 0x9 **ARGB4444** — 16-bit pixels with alpha 0xC **RGB555** — 16-bit pixels without alpha 0xD **RGB444** — 16-bit pixels without alpha 0xE **RGB565** — 16-bit pixels without alpha
3 ENABLE_ COLORKEY	Indicates that colorkey functionality is enabled for this alpha surface. Pixels found in the alpha surface colorkey range will be displayed as transparent (the PS pixel will be used).
2–1 ALPHA_CTRL	Determines how the alpha value is constructed for this alpha surface. Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x0 **Embedded** — Indicates that the AS pixel alpha value will be used to blend the AS with PS. The ALPHA field is ignored. 0x1 **Override** — Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x2 **Multiply** — Indicates that the value in the ALPHA field should be used to scale all pixel alpha values. Each pixel alpha is multiplied by the value in the ALPHA field. 0x3 **ROPs** — Enable ROPs. The ROP field indicates an operation to be performed on the alpha surface and PS pixels.
0 RSVD0	Reserved, always set to zero.

13.6.12.23 Alpha Surface Buffer Pointer (PXP_HW_PXP_AS_BUF)

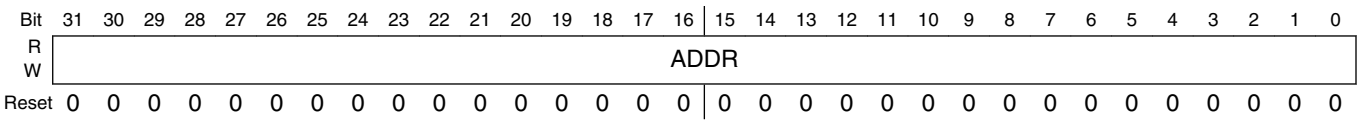
Alpha Surface 0 Buffer Address Pointer. This register points to the beginning of the Alpha Surface 0 input buffer.

This register is used to indicate the base address of the AS buffer.

EXAMPLE

```
u32* alpha_ptr;
REG_ASn_WR(0,alpha_ptr);
```

Address: 3070_0000h base + 160h offset = 3070_0160h



PXP_HW_PXP_AS_BUF field descriptions

Field	Description
ADDR	Address pointer for the alpha surface 0 buffer.

13.6.12.24 Alpha Surface Pitch (PXP_HW_PXP_AS_PITCH)

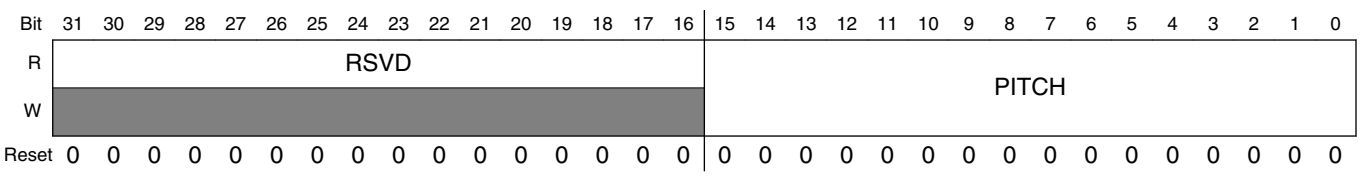
This register contains the alpha surface pitch in bytes.

Any byte value will indicate the vertical pitch. This value will be used in AS pixel address calculations. This value has no relation to the UL and LR registers. It specifies how many bytes are between two vertically adjacent pixels in the input AS surface.

EXAMPLE

```
REG_AS_PITCH_WR( 1920 * 4 );  // The output buffer pitch is HD resolution at 32 bits per
pixel
```

Address: 3070_0000h base + 170h offset = 3070_0170h



PXP_HW_PXP_AS_PITCH field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
PITCH	Indicates the number of bytes in memory between two vertically adjacent pixels.

13.6.12.25 Overlay Color Key Low (PXP_HW_PXP_AS_CLRKEYLOW_0)

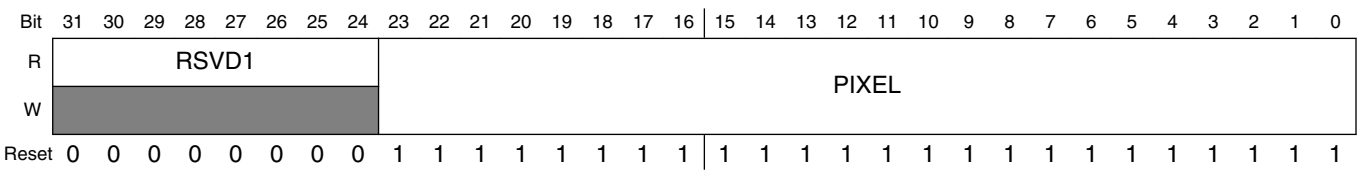
This register contains the color key low value for the AS buffer. This value is used by the Alpha A block.

When processing an image, the if the PXP finds a pixel in the current overlay image with a color that falls in the range from the ASCOLORKEYLOW to ASCOLORKEYHIGH range, it will use the PS pixel value for that location. If no PS image is present or if the PS image also matches its colorkey range, the PS background color is used. Colorkey operations are higher priority than alpha or ROP operations.

EXAMPLE

```
                                    // colorkey values between
REG_AS_CLRKEYLOW_WR (0x000000); // black and
REG_AS_CLRKEYHIGH_WR(0x800000); // medium red
```

Address: 3070_0000h base + 180h offset = 3070_0180h



PXP_HW_PXP_AS_CLRKEYLOW_0 field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	Low range of RGB color key applied to AS buffer. Each overlay has an independent colorkey enable.

13.6.12.26 Overlay Color Key High (PXP_HW_PXP_AS_CLRKEYHIGH_0)

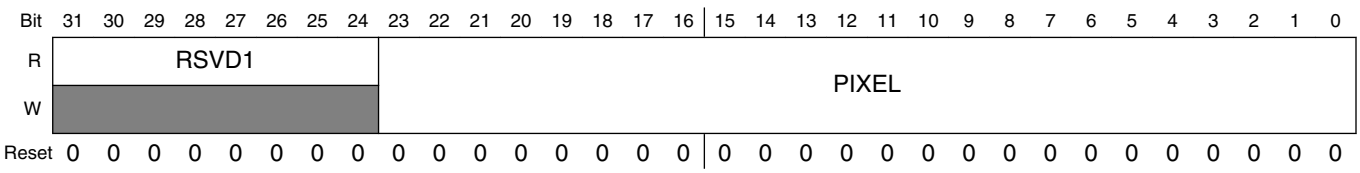
This register contains the color key high value for the AS buffer. This value is used by the Alpha A block.

When processing an image, the if the PXP finds a pixel in the current overlay image with a color that falls in the range from the ASCOLORKEYLOW to ASCOLORKEYHIGH range, it will use the PS pixel value for that location. If no PS image is present or if the PS image also matches its colorkey range, the PS background color is used. Colorkey operations are higher priority than alpha or ROP operations.

EXAMPLE

```
                                    // colorkey values between
REG_AS_CLRKEYLOW_WR (0x000000); // black and
REG_AS_CLRKEYHIGH_WR(0x800000); // medium red
```

Address: 3070_0000h base + 190h offset = 3070_0190h



PXP_HW_PXP_AS_CLRKEYHIGH_0 field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	High range of RGB color key applied to AS buffer. Each overlay has an independent colorkey enable.

13.6.12.27 Color Space Conversion Coefficient Register 0 (PXP_HW_PXP_CSC1_COEF0)

This register contains color space conversion coefficients in two's compliment notation.

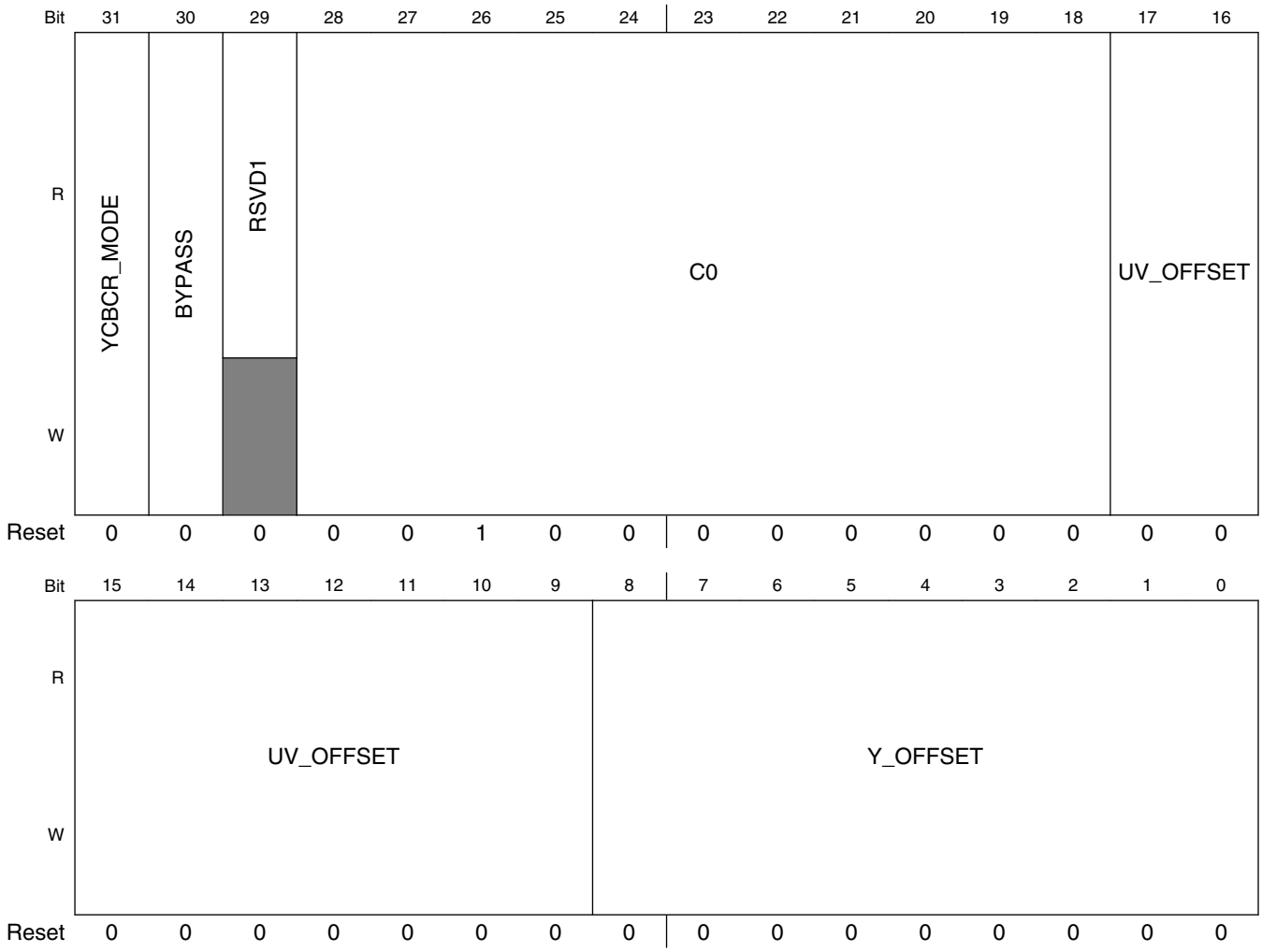
The Coefficient 0 register contains coefficients used in the color space conversion algorithm. The Y and UV offsets are added to the source buffer to normalize them before the conversion. C0 is the coefficient that is used to multiply the luma component of the data for all three RGB components.

EXAMPLE

```
//  The equations used for Colorspace conversion are:
//     R = C0*(Y+YOFFSET)                    + C1(V+UV_OFFSET)
//     G = C0*(Y+YOFFSET) + C3(U+UV_OFFSET) + C2(V+UV_OFFSET)
//     R = C0*(Y+YOFFSET) + C4(U+UV_OFFSET)

   REG_CSCCOEF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
   REG_CSCCOEF1_WR(0x01230208); // YUV coefficients: C1, C4
   REG_CSCCOEF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: 3070_0000h base + 1A0h offset = 3070_01A0h



PXP_HW_PXP_CSC1_COEF0 field descriptions

Field	Description
31 YCBCR_MODE	Set to 1 when performing YCbCr conversion to RGB. Set to 0 when converting YUV to RGB data. This bit changes the behavior of the scaler when performing U/V scaling.
30 BYPASS	Bypass the CSC unit in the scaling engine. When set to logic 1, bypass is enabled and the output pixels will be in the YUV/YCbCr color space. When set to logic 0, the CSC unit is enabled and the pixels will be converted based on the programmed coefficients.
29 RSVD1	Reserved, always set to zero.
28–18 C0	Two's compliment Y multiplier coefficient. YUV=0x100 (1.000) YCbCr=0x12A (1.164)
17–9 UV_OFFSET	Two's compliment phase offset implicit for CbCr data. Generally used for YCbCr to RGB conversion. YCbCr=0x180, YUV=0x000 (typically -128 or 0x180 to indicate normalized -0.5 to 0.5 range)
Y_OFFSET	Two's compliment amplitude offset implicit in the Y data. For YUV, this is typically 0 and for YCbCr, this is typically -16 (0x1F0)

13.6.12.28 Color Space Conversion Coefficient Register 1 (PXP_HW_PXP_CSC1_COEF1)

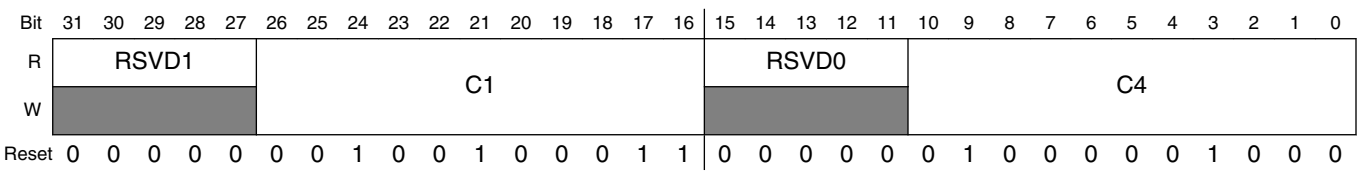
This register contains color space conversion coefficients in two's compliment notation.

The Coefficient 1 register contains coefficients used in the color space conversion algorithm. C1 is the coefficient that is used to multiply the chroma (Cr/V) component of the data for the red component. C4 is the coefficient that is used to multiply the chroma (Cb/U) component of the data for the blue component. Both values should be coded as a two's compliment fixed point number with 8 bits right of the decimal.

EXAMPLE

```
REG_CSCCOEF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
REG_CSCCOEF1_WR(0x01230208); // YUV coefficients: C1, C4
REG_CSCCOEF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: 3070_0000h base + 1B0h offset = 3070_01B0h



PXP_HW_PXP_CSC1_COEF1 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 C1	Two's compliment Red V/Cr multiplier coefficient. YUV=0x123 (1.140) YCbCr=0x198 (1.596)
15–11 RSVD0	Reserved, always set to zero.
C4	Two's compliment Blue U/Cb multiplier coefficient. YUV=0x208 (2.032) YCbCr=0x204 (2.017)

13.6.12.29 Color Space Conversion Coefficient Register 2 (PXP_HW_PXP_CSC1_COEF2)

This register contains color space conversion coefficients in two's compliment notation.

The Coefficient 2 register contains coefficients used in the color space conversion algorithm. C2 is the coefficient that is used to multiply the chroma (Cr/V) component of the data for the green component. C3 is the coefficient that is used to multiply the chroma (Cb/U) component of the data for the green component. Both values should be coded as a two's compliment fixed point number with 8 bits right of the decimal.

EXAMPLE

```
// NOTE: The default values for the CSCCOEF2 register are incorrect.  C2 should be 0x76B and
C3 should be 0x79C for proper operation.

   REG_CSCCOEF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
   REG_CSCCOEF1_WR(0x01230208); // YUV coefficients: C1, C4
   REG_CSCCOEF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: 3070_0000h base + 1C0h offset = 3070_01C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1					C2											RSVD0					C3										
W																																
Reset	0	0	0	0	0	1	1	1	1	0	0	1	1	0	1	1	0	0	0	0	0	1	1	1	0	1	1	0	1	1	0	0

PXP_HW_PXP_CSC1_COEF2 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 C2	Two's compliment Green V/Cr multiplier coefficient. YUV=0x76B (-0.581) YCbCr=0x730 (-0.813)
15–11 RSVD0	Reserved, always set to zero.
C3	Two's compliment Green U/Cb multiplier coefficient. YUV=0x79C (-0.394) YCbCr=0x79C (-0.392)

13.6.12.30 Color Space Conversion Control Register. (PXP_HW_PXP_CSC2_CTRL)

This register contains the control registers to configure the CSC module.

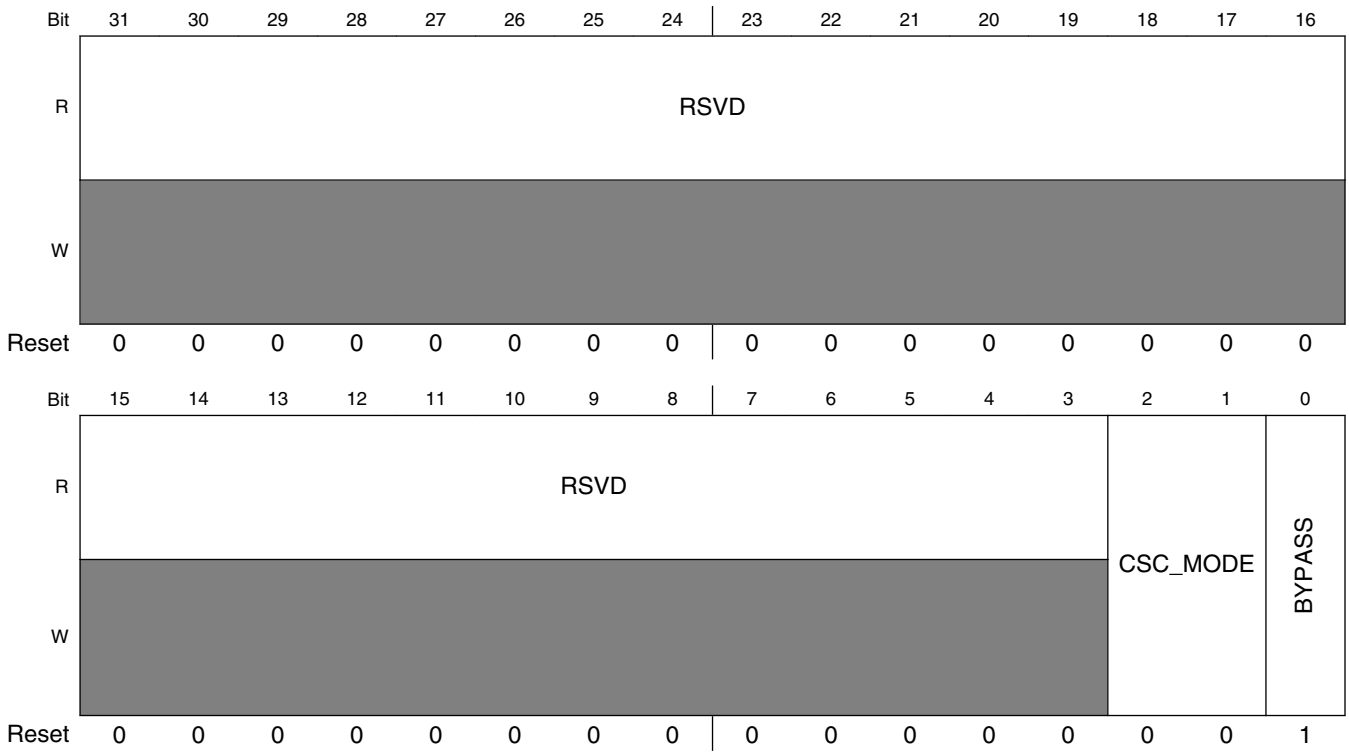
The CSC control register will configure the CSC module to perform color space conversion between the RGB/YUV/YCbCr color spaces.

EXAMPLE

```
//Converting from YUV/YCbCr color spaces to the RGB color space uses the
//following equation structure:
//
// R = A1(Y-D1) + A2(U-D2) + A3(V-D3)
// G = B1(Y-D1) + B2(U-D2) + B3(V-D3)
```

```
// B = C1(Y-D1) + C2(U-D2) + C3(V-D3)
//
//Converting from the RGB color space to YUV/YCbCr color spaces uses the
//following equation structure:
//
// Y = A1*R + A2*G + A3*B + D1
// U = B1*R + B2*G + B3*B + D2
// V = C1*R + C2*G + C3*B + D3
//
//All math is signed, so all coefficients come in as two's comp numbers
//
```

Address: 3070_0000h base + 1D0h offset = 3070_01D0h



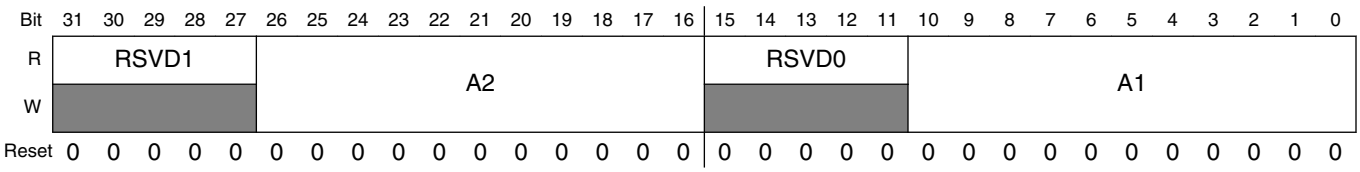
PXP_HW_PXP_CSC2_CTRL field descriptions

Field	Description
31–3 RSVD	Reserved, always set to zero.
2–1 CSC_MODE	This field controls how the CSC unit operates on pixels when the CSC is not bypassed. 0x0 **YUV2RGB** — Convert from YUV to RGB. 0x1 **YCbCr2RGB** — Convert from YCbCr to RGB. 0x2 **RGB2YUV** — Convert from RGB to YUV. 0x3 **RGB2YCbCr** — Convert from RGB to YCbCr.
0 BYPASS	This bit controls whether the pixels entering the CSC2 unit get converted or not. When BYPASS is set, no operations occur on the pixels. When BYPASS is cleared, the selected CSC operation takes place.

13.6.12.31 Color Space Conversion Coefficient Register 0 (PXP_HW_PXP_CSC2_COEF0)

This register contains color space conversion coefficients in two's compliment notation.

Address: 3070_0000h base + 1E0h offset = 3070_01E0h



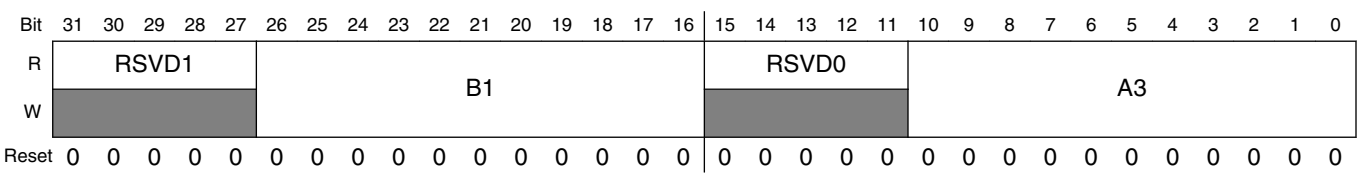
PXP_HW_PXP_CSC2_COEF0 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 A2	Two's compliment coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.
15–11 RSVD0	Reserved, always set to zero.
A1	Two's compliment coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

13.6.12.32 Color Space Conversion Coefficient Register 1 (PXP_HW_PXP_CSC2_COEF1)

This register contains color space conversion coefficients in two's compliment notation.

Address: 3070_0000h base + 1F0h offset = 3070_01F0h



PXP_HW_PXP_CSC2_COEF1 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 B1	Two's compliment coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

Table continues on the next page...

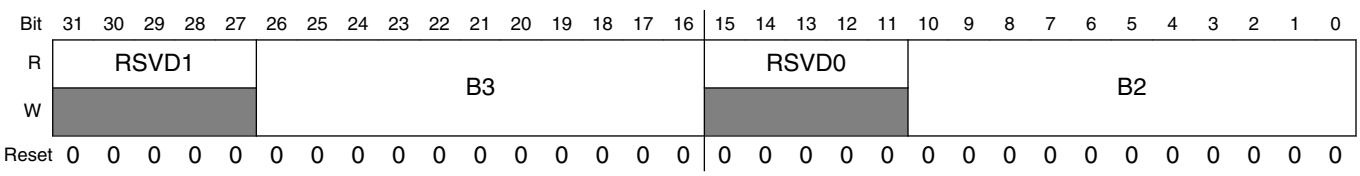
PXP_HW_PXP_CSC2_COEF1 field descriptions (continued)

Field	Description
15–11 RSVD0	Reserved, always set to zero.
A3	Two's compliment coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

13.6.12.33 Color Space Conversion Coefficient Register 2 (PXP_HW_PXP_CSC2_COEF2)

This register contains color space conversion coefficients in two's compliment notation.

Address: 3070_0000h base + 200h offset = 3070_0200h



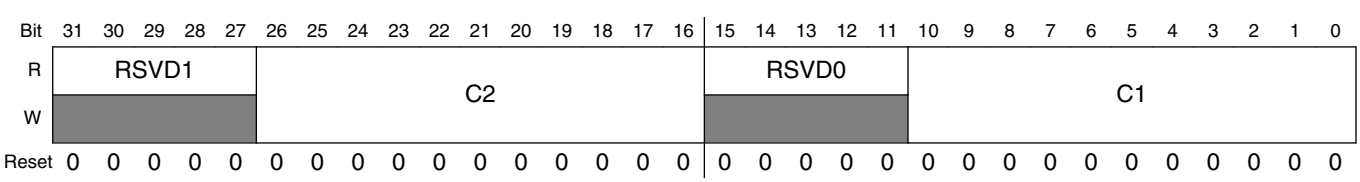
PXP_HW_PXP_CSC2_COEF2 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 B3	Two's compliment coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.
15–11 RSVD0	Reserved, always set to zero.
B2	Two's compliment coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

13.6.12.34 Color Space Conversion Coefficient Register 3 (PXP_HW_PXP_CSC2_COEF3)

This register contains color space conversion coefficients in two's compliment notation.

Address: 3070_0000h base + 210h offset = 3070_0210h



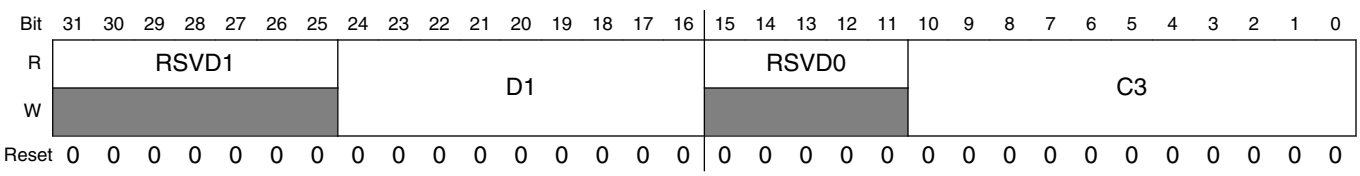
PXP_HW_PXP_CSC2_COEF3 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 C2	Two's compliment coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.
15–11 RSVD0	Reserved, always set to zero.
C1	Two's compliment coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

13.6.12.35 Color Space Conversion Coefficient Register 4 (PXP_HW_PXP_CSC2_COEF4)

This register contains color space conversion coefficients in two's compliment notation.

Address: 3070_0000h base + 220h offset = 3070_0220h



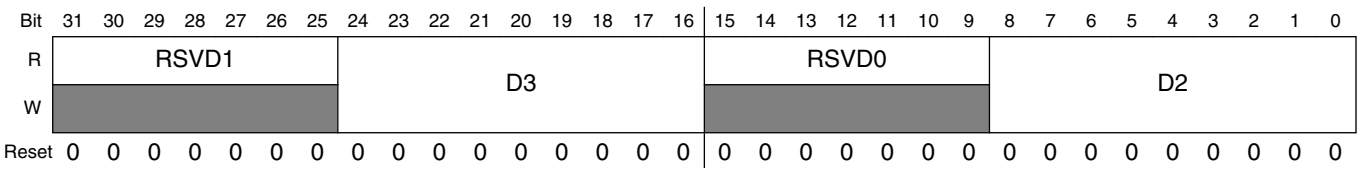
PXP_HW_PXP_CSC2_COEF4 field descriptions

Field	Description
31–25 RSVD1	Reserved, always set to zero.
24–16 D1	Two's compliment coefficient integer offset to be added.
15–11 RSVD0	Reserved, always set to zero.
C3	Two's compliment coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

13.6.12.36 Color Space Conversion Coefficient Register 5 (PXP_HW_PXP_CSC2_COEF5)

This register contains color space conversion coefficients in two's compliment notation.

Address: 3070_0000h base + 230h offset = 3070_0230h



PXP_HW_PXP_CSC2_COEF5 field descriptions

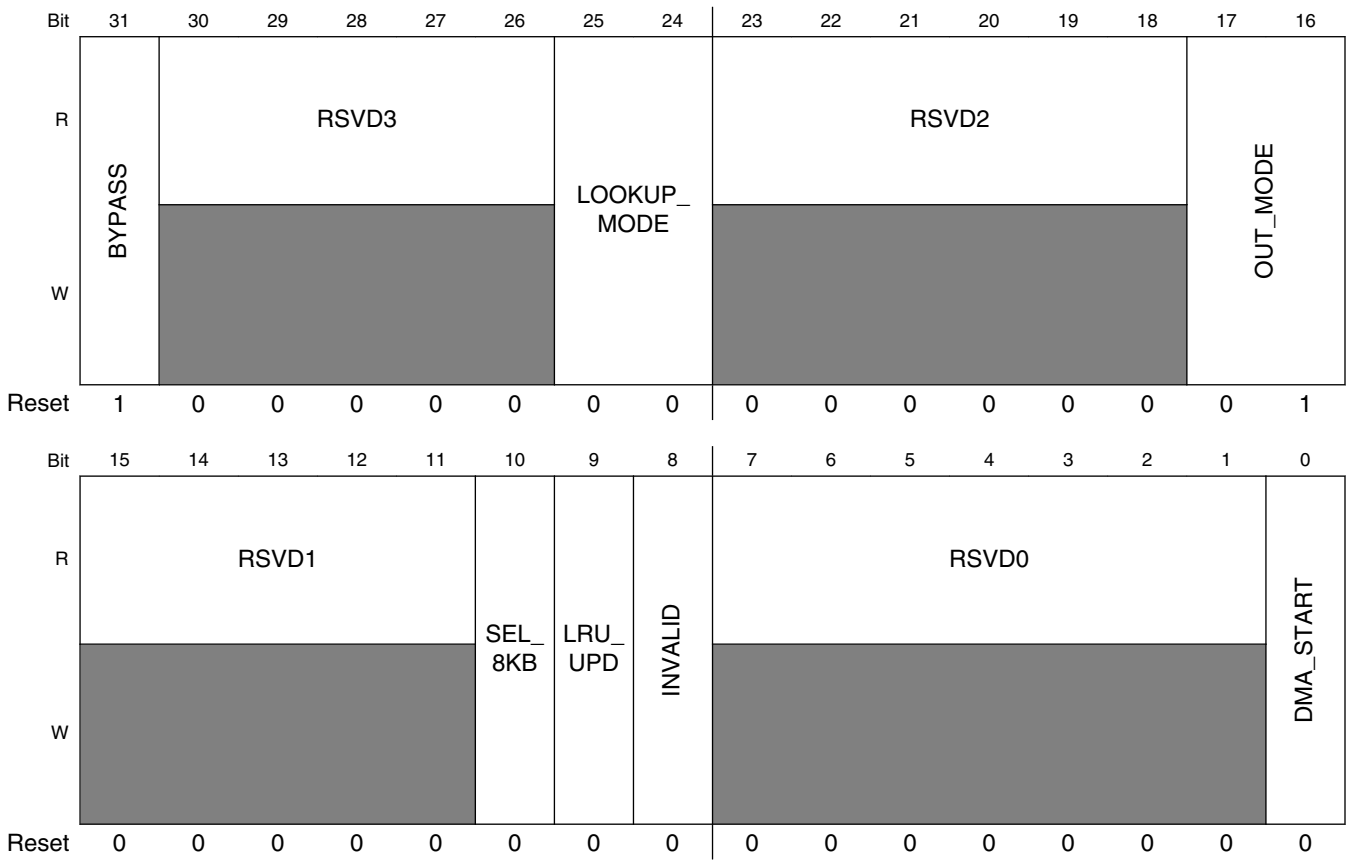
Field	Description
31–25 RSVD1	Reserved, always set to zero.
24–16 D3	Two's compliment coefficient integer offset to be added.
15–9 RSVD0	Reserved, always set to zero.
D2	Two's compliment D1 coefficient integer offset to be added.

13.6.12.37 Lookup Table Control Register. (PXP_HW_PXP_LUT_CTRL)

This register is used to access/control the Monochrome Lookup table.

The Y8 LUT input mode will take the high order data path byte and transform it using the LUT memory. This is an 8-bit to 8-bit transformation. The two low order bytes bypass the LUT and are not transformed, but bypassed without modification. This option can be used for monochrome gamma correction. The Direct Lookup mode will use the high nibble of each data byte and truncate the low nibble to generate the lookup address, i.e. R[7:0]G[7:0]B[7:0] -> R[7:4]G[7:4]B[7:4]. 4K pixels (12-bit address) with 2 bytes per pixel is supported in this mode. Cached Lookup mode will use the high order bits, R[7:3],G[7:2],B[7:3] or RGB565, to address the cached LUT memory. 64KB LUT tables, using 16KB of internal LUT memory, can be indirectly transformed to 16-bit output pixels (as in RGBW4444/RGB565). This is used for 16bpp gamma correction or EPD color panel support. Cache misses are internally managed by the PXP LUT Cache controller.

Address: 3070_0000h base + 240h offset = 3070_0240h



PXP_HW_PXP_LUT_CTRL field descriptions

Field	Description
31 BYPASS	Setting this bit will bypass the LUT memory resource completely. No pixel transfermations will occur at this stage of the PXP pixel rpocessing pipeline.
30–26 RSVD3	Reserved, always set to zero.
25–24 LOOKUP_MODE	Configure the input address for the 16KB LUT memory. The address into the LUT uses different parts of the pixel data path bytes. The data path is defined as three bytes, conceptually as RGB/YUV/YCbCr[23:0]. Also referred to as R/Y[7:0],G/U[7:0],B/V[7:0] 0x0 **CACHE_RGB565** — LUT ADDR = R[7:3],G[7:2],B[7:3]. Use all 16KB of LUT for indirect cached 128KB lookup. 0x1 **DIRECT_Y8** — LUT ADDR = 16'b0,Y[7:0]. Use only the first 256 bytes of LUT. Only the Y, or third data path byte, is tranformed. 0x2 **DIRECT_RGB444** — LUT ADDR = R[7:4],G[7:4],B[7:4]. Use one 8KB bank of LUT selected by SEL_8KB. 0x3 **DIRECT_RGB454** — LUT ADDR = R[7:4],G[7:3],B[7:4]. Use all 16KB of LUT.
23–18 RSVD2	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_LUT_CTRL field descriptions (continued)

Field	Description
17–16 OUT_MODE	Select the output mode of operation for the LUT resource. There are four bytes [3-0] in the data path at the output of the LUT resource. Byte lane 3 is always bypassed and usually contains an alpha value. The LUT can be programmed to transform bytes 2,1,0 according to the options available in this field. 0x0 **RESERVED** — Reserved, not valid when using the LUT to transform pixels. 0x1 **Y8** — R/Y byte lane 2 lookup, bytes 1,0 bypassed. 0x2 **RGBW4444CFA** — Byte lane 2 = CFA_Y8, byte lane 1,0 = RGBW4444. 0x3 **RGB888** — RGB565->RGB888 conversion for Gamma correction.
15–11 RSVD1	Reserved, always set to zero.
10 SEL_8KB	Selects which 8KB bank of memory to use for direct 12bpp lookup modes. Logic 0 indicates first 8KB, logic 1 indicates second 8KB. Two direct LUT arrays can be stored and one can be selected for a given PXP operation.
9 LRU_UPD	Least Recently Used Policy Update Control: 1=> block LRU update for hit after miss. 0=> update LRU for all hits including hit after miss.
8 INVALID	Invalidate the cache LRU and valid bits. This bit will automatically reset when set to a logic 1.
7–1 RSVD0	Reserved, always set to zero.
0 DMA_START	Setting this bit will result in the DMA operation to load the PXP LUT memory based on REG_LUT_ADDR_NUM_BYTES, REG_LUT_ADDR_ADDR, and REG_LUT_MEM_ADDR. This bit will automatically reset when set to a logic 1. Note: The LOOKUP_MODE must not be set to CACHE_RGB565 when starting and performing DMA transfers.

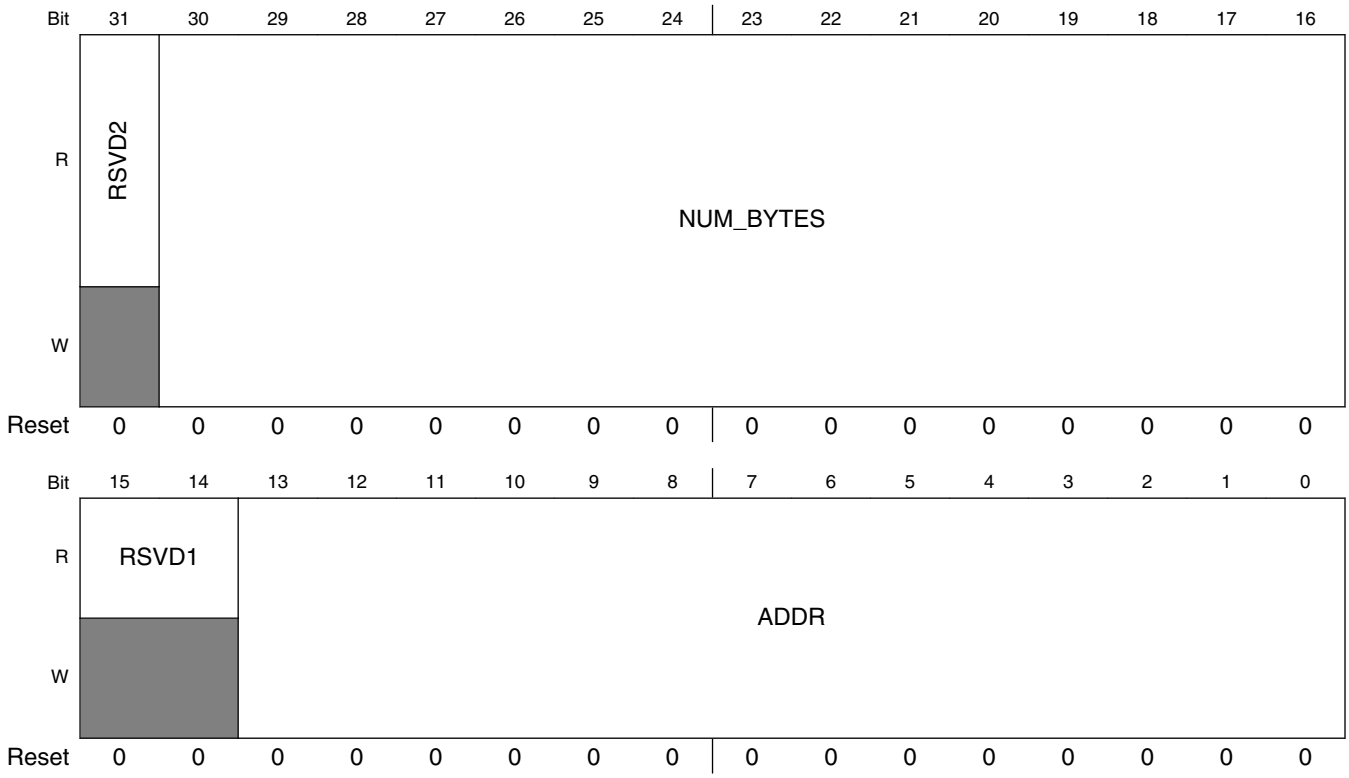
13.6.12.38 Lookup Table Control Register. (PXP_HW_PXP_LUT_ADDR)

This register is used to access/control the Monochrome Lookup table.

The Y8 LUT input mode will take the high order data path byte and transform it using the LUT memory. This is an 8-bit to 8-bit transformation. The two low order bytes bypass the LUT and are not transformed, but bypassed without modification. This option can be used for monochrome gamma correction. The Direct Lookup mode will use the high nibble of each data byte and truncate the low nibble to generate the lookup address, i.e. R[7:0]G[7:0]B[7:0] -> R[7:4]G[7:4]B[7:4]. 4K pixels (12-bit address) with 2 bytes per pixel is supported in this mode. Cached Lookup mode will use the high order bits, R[7:3],G[7:2],B[7:3] or RGB565, to address the cached LUT memory. 64KB LUT tables, using 16KB of internal LUT memory, can be indirectly transformed to 16-bit output

pixels (as in RGBW4444/RGB565). This is used for 16bpp gamma correction or EPD color panel support. Cache misses are internally managed by the PXP LUT Cache controller.

Address: 3070_0000h base + 250h offset = 3070_0250h



PXP_HW_PXP_LUT_ADDR field descriptions

Field	Description
31 RSVD2	Reserved, always set to zero.
30–16 NUM_BYTES	Indicates the number of bytes to load via a DMA operation. This field must be divisable by 8 and the least significant 3 bits must be 0. The value 8 indicates load 8 bytes from the external address indicated by REG_LUT_MEM_ADDR to the LUT memory location indicated by REG_LUT_CTRL_ADDR.
15–14 RSVD1	Reserved, always set to zero.
ADDR	LUT indexed address pointer. This address into the LUT memory is always four byte aligned for PIO access, and eight byte aligned for DMA access. The least two significant bits are not used to drive the LUT memory array. For PIO LUT access, when the LUT data register is written, the contents of the LUT at the address specified by this address field will be loaded with a 32-bit DWORD. This address pointer will be incremented after the LUT data is written. This will provide recursive writes to the LUT data register to initialize the entire LUT array with recursive writes to the LUT data register. For DMA access, this register indicates the LUT memory address of the 8 byte QWORD to be loaded. When using the NUM_BYTES field to load more than 8 bytes, the register should be programmed with the first LUT memory location to be filled and each load of the LUT memory will increment this address field until NUM_BYTES has been loaded.

13.6.12.39 Lookup Table Data Register. (PXP_HW_PXP_LUT_DATA)

This register is used to load data into the lookup table.

Address: 3070_0000h base + 260h offset = 3070_0260h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_LUT_DATA field descriptions

Field	Description
DATA	Writing this field will load 4 bytes, aligned to four byte boundaries, of data indexed by the ADDR field of the REG_LUT_CTRL register.

13.6.12.40 Lookup Table External Memory Address Register. (PXP_HW_PXP_LUT_EXTMEM)

This register is used as the external memory base address for LUT data transfers.

For DMA LUT memory loads, this is the base address from which data will be sourced to store into the LUT memory array. For Cached LUT memory pixel transformations, this register will store the base address of the full 64K pixel LUT translation table.

Address: 3070_0000h base + 270h offset = 3070_0270h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_LUT_EXTMEM field descriptions

Field	Description
ADDR	This register contains the external memory address used for LUT memory operation.

13.6.12.41 Color Filter Array Register. (PXP_HW_PXP_CFA)

This register contains the pattern of the CFA mapping within PXP macro blocks.

There are sixteen 2 bit values in this register each mapping a selected component to the output pixel. The two bit values are defined as 0=>R, 1=>G, 2=>B, and 3=>W. The first byte represents the repetitive pattern of RGBW pixels in the CFA for the first line

segment of each processed PXP block. The second byte represents the pattern in the second line segment of the block, and so on. The first byte repeats two times for 8x8 macro block mode, and repeats four times for 16x16 block mode.

Address: 3070_0000h base + 280h offset = 3070_0280h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_CFA field descriptions

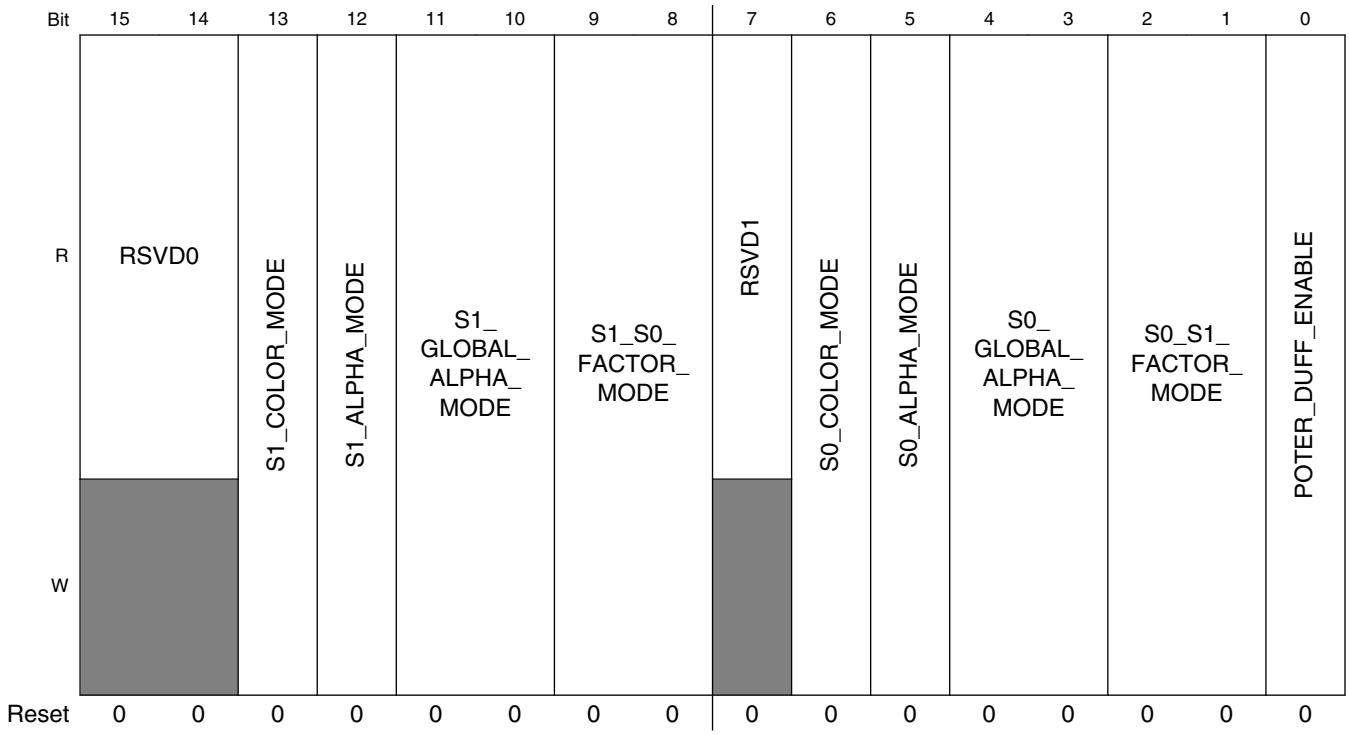
Field	Description
DATA	This register contains the Color Filter Array pattern for decimation of RGBW4444 16 bit pixels to individual R, G, B, W values. The pattern represents a replicated 4x4 color filter array for the entire output frame buffer.

13.6.12.42 PXP Alpha Engine A Control Register. (PXP_HW_PXP_ALPHA_A_CTRL)

This register controls alpha blending function in PXP.

Address: 3070_0000h base + 290h offset = 3070_0290h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	S1_GLOBAL_ALPHA								S0_GLOBAL_ALPHA							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



PXP_HW_PXP_ALPHA_A_CTRL field descriptions

Field	Description
31–24 S1_GLOBAL_ ALPHA	s1 global alpha
23–16 S0_GLOBAL_ ALPHA	s0 global alpha
15–14 RSVD0	Reserved, always set to zero.
13 S1_COLOR_ MODE	s1 color mode 0x0 **0** — straight mode for s1 color 0x1 **1** — multiply mode for s1 color
12 S1_ALPHA_ MODE	s1 alpha mode 0x0 **0** — straight mode for s1 alpha 0x1 **1** — inversed mode for s1 alpha
11–10 S1_GLOBAL_ ALPHA_MODE	s1 global alpha mode 0x0 **0** — using global alpha. 0x0 **1** — using local alpha. 0x0 **2** — using scaled alpha. 0x0 **3** — using scaled alpha.
9–8 S1_S0_ FACTOR_MODE	s1 to s0 factor mode 0x0 **0** — using 1.

Table continues on the next page...

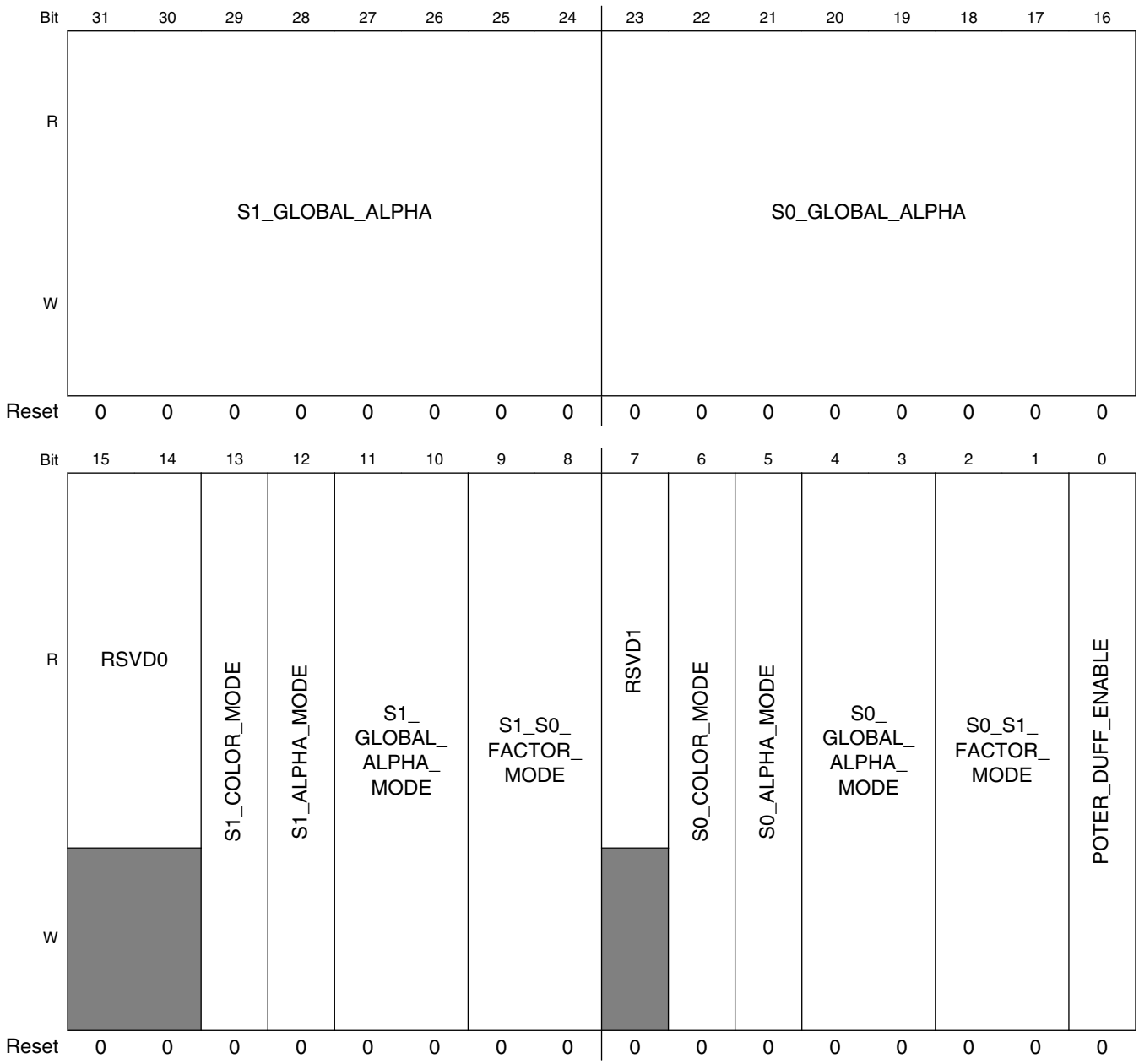
PXP_HW_PXP_ALPHA_A_CTRL field descriptions (continued)

Field	Description
	0x1 **1** — using 0. 0x2 **2** — using straight alpha. 0x3 **3** — using inverse alpha.
7 RSVD1	Reserved, always set to zero.
6 S0_COLOR_ MODE	s0 color mode 0x0 **0** — straight mode for s0 color 0x1 **1** — multiply mode for s0 color
5 S0_ALPHA_ MODE	s0 alpha mode 0x0 **0** — straight mode for s0 alpha 0x1 **1** — inversed mode for s0 alpha
4–3 S0_GLOBAL_ ALPHA_MODE	s0 global alpha mode 0x0 **0** — using global alpha. 0x1 **1** — using local alpha. 0x2 **2** — using scaled alpha. 0x3 **3** — using scaled alpha.
2–1 S0_S1_ FACTOR_MODE	s0 to s1 factor mode 0x0 **0** — using 1. 0x1 **1** — using 0. 0x2 **2** — using straight alpha. 0x3 **3** — using inverse alpha.
0 POTER_DUFF_ ENABLE	poter_duff enable 0x0 **0** — porter duff disable. 0x1 **1** — porter duff enable.

13.6.12.43 PXP Alpha Engine B Control Register. (PXP_HW_PXP_ALPHA_B_CTRL)

This register controls alpha blending function in PXP.

Address: 3070_0000h base + 2A0h offset = 3070_02A0h



PXP_HW_PXP_ALPHA_B_CTRL field descriptions

Field	Description
31–24 S1_GLOBAL_ ALPHA	s1 global alpha
23–16 S0_GLOBAL_ ALPHA	s0 global alpha
15–14 RSVD0	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_ALPHA_B_CTRL field descriptions (continued)

Field	Description
13 S1_COLOR_ MODE	s1 color mode 0x0 **0** — straight mode for s1 color 0x1 **1** — multiply mode for s1 color
12 S1_ALPHA_ MODE	s1 alpha mode 0x0 **0** — straight mode for s1 alpha 0x1 **1** — inversed mode for s1 alpha
11–10 S1_GLOBAL_ ALPHA_MODE	s1 global alpha mode 0x0 **0** — using global alpha. 0x1 **1** — using local alpha. 0x2 **2** — using scaled alpha. 0x3 **3** — using scaled alpha.
9–8 S1_S0_ FACTOR_MODE	s1 to s0 factor mode 0x0 **0** — using 1. 0x1 **1** — using 0. 0x2 **2** — using straight alpha. 0x3 **3** — using inverse alpha.
7 RSVD1	Reserved, always set to zero.
6 S0_COLOR_ MODE	s0 color mode 0x0 **0** — straight mode for s0 color 0x1 **1** — multiply mode for s0 color
5 S0_ALPHA_ MODE	s0 alpha mode 0x0 **0** — straight mode for s0 alpha 0x1 **1** — inversed mode for s0 alpha
4–3 S0_GLOBAL_ ALPHA_MODE	s0 global alpha mode 0x0 **0** — using global alpha. 0x1 **1** — using local alpha. 0x2 **2** — using scaled alpha. 0x3 **3** — using scaled alpha.
2–1 S0_S1_ FACTOR_MODE	s0 to s1 factor mode 0x0 **0** — using 1. 0x1 **1** — using 0. 0x2 **2** — using straight alpha. 0x3 **3** — using inverse alpha.
0 POTER_DUFF_ ENABLE	poter_duff enable 0x0 **0** — porter duff disable. 0x1 **1** — porter duff enable.

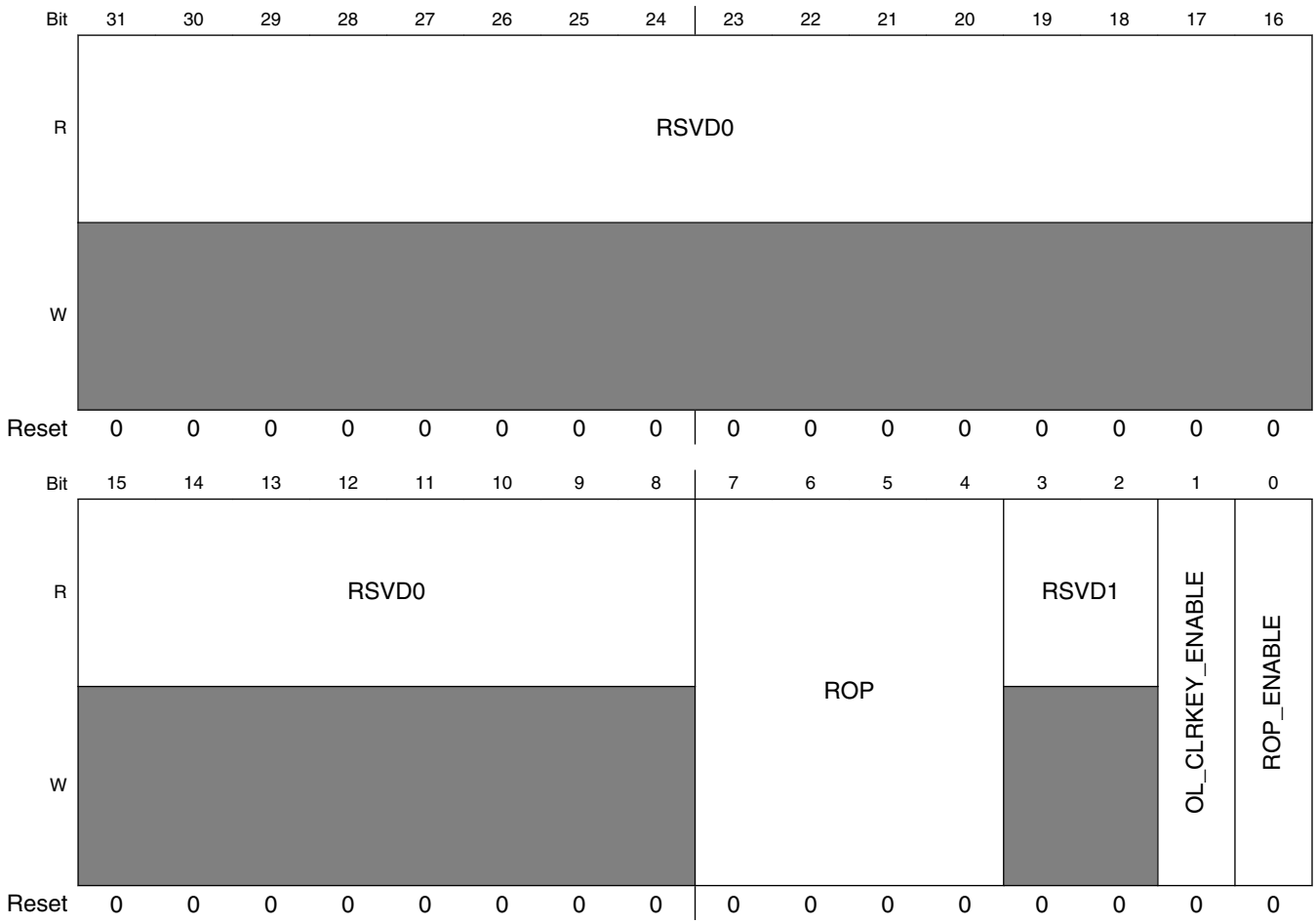
13.6.12.44 PXP_HW_PXP_ALPHA_B_CTRL_1

This register defines software define pixels for alpha blending.

This register defines alpha1 legacy blending parameters.

EXAMPLE

Address: 3070_0000h base + 2B0h offset = 3070_02B0h



PXP_HW_PXP_ALPHA_B_CTRL_1 field descriptions

Field	Description
31–8 RSVD0	Reserved, always set to zero.
7–4 ROP	Indicates a raster operation to perform when enabled. 0x0 **MASKAS** — AS AND PS 0x1 **MASKNOTAS** — nAS AND PS 0x2 **MASKASNOT** — AS AND nPS

Table continues on the next page...

PXP_HW_PXP_ALPHA_B_CTRL_1 field descriptions (continued)

Field	Description
	0x3 **MERGEAS** — AS OR PS 0x4 **MERGENOTAS** — nAS OR PS 0x5 **MERGEASNOT** — AS OR nPS 0x6 **NOTCOPYAS** — nAS 0x7 **NOT** — nPS 0x8 **NOTMASKAS** — AS NAND PS 0x9 **NOTMERGEAS** — AS NOR PS 0xA **XORAS** — AS XOR PS 0xB **NOTXORAS** — AS XNOR PS
3–2 RSVD1	Reserved, always set to zero.
1 OL_CLRKEY_ENABLE	Indicates that colorkey functionality is enabled for this alpha surface. Pixels found in the alpha surface colorkey range will be displayed as transparent (the PS pixel will be used).
0 ROP_ENABLE	ROP ENABLE

13.6.12.45 PS Background Color 1 (PXP_HW_PXP_PS_BACKGROUND_1)

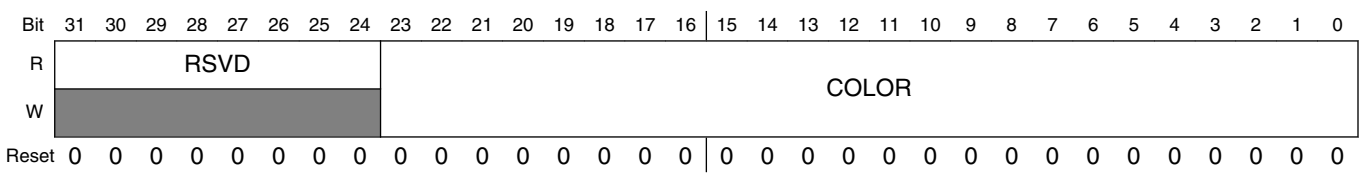
PS Background Pixel Color. This register provides a pixel value used when processing pixels outside of the region specified by the PS Coordinate registers. This value can effectively be used to set the color of the letterboxing region around the PS image. This value is used by the Alpha B block.

This register contains a pixel value to be used for any PS pixels that fall outside the PS extents. This is effectively a background or letterbox color. The CSC1 control and datapath pixel format should be considered when selecting the background color.

EXAMPLE

```
REG_PS_BACKGROUND_WR(0x00000000);  // letterbox is black
REG_PS_BACKGROUND_WR(0x00800000);  // letterbox is dark red
REG_PS_BACKGROUND_WR(0x00008000);  // letterbox is dark green
REG_PS_BACKGROUND_WR(0x00000080);  // letterbox is dark blue
```

Address: 3070_0000h base + 2C0h offset = 3070_02C0h



PXP_HW_PXP_PS_BACKGROUND_1 field descriptions

Field	Description
31–24 RSVD	Reserved, always set to zero.
COLOR	Background color (in 24bpp format) for any pixels not within the buffer range specified by the PS ULC/ LRC.

13.6.12.46 PS Color Key Low 1 (PXP_HW_PXP_PS_CLRKEYLOW_1)

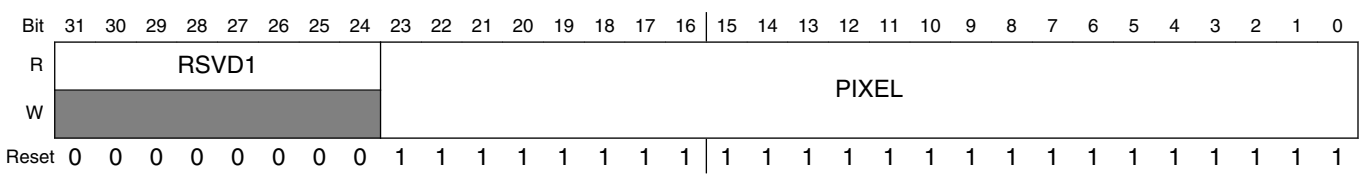
This register contains the color key low value for the PS buffer 1. This value is used by the Alpha B block.

When processing an image, if the PXP finds a pixel in the PS buffer with a color that falls in the range between REG_PS_CLRKEYLOW and REG_PS_CLRKEYHIGH, it will insert the pixel from the AS channel. If the current AS pixel is letterboxed or if the AS also matches its colorkey range, the REG_PS_BACKGROUND color is passed down the pixel pipeline.

EXAMPLE

```
                                    // colorkey values between
REG_PS_CLRKEYLOW_WR (0x008000); // medium green and
REG_PS_CLRKEYHIGH_WR(0x00FF00); // light green
```

Address: 3070_0000h base + 2D0h offset = 3070_02D0h



PXP_HW_PXP_PS_CLRKEYLOW_1 field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	Low range of color key applied to PS buffer. To disable PS colorkeying, set the low colorkey to 0xFFFFFF and the high colorkey to 0x000000.

13.6.12.47 PS Color Key High 1 (PXP_HW_PXP_PS_CLRKEYHIGH_1)

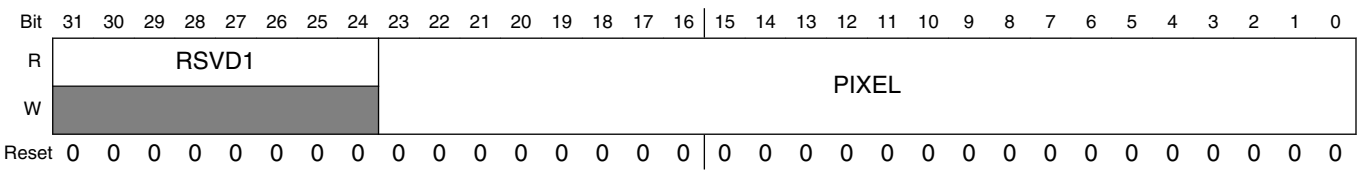
This register contains the color key high value for the PS buffer. This value is used by the Alpha B block.

When processing an image, if the PXP finds a pixel in the PS buffer with a color that falls in the range between REG_PS_CLRKEYLOW and REG_PS_CLRKEYHIGH, it will insert the pixel from the AS channel. If the current AS pixel is letterboxed or if the AS also matches its colorkey range, the REG_PS_BACKGROUND color is passed down the pixel pipeline.

EXAMPLE

```
                                    // colorkey values between
REG_PS_CLRKEYLOW_WR (0x008000); // medium green and
REG_PS_CLRKEYHIGH_WR(0x00FF00); // light green
```

Address: 3070_0000h base + 2E0h offset = 3070_02E0h



PXP_HW_PXP_PS_CLRKEYHIGH_1 field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	High range of color key applied to PS buffer. To disable PS colorkeying, set the low colorkey to 0xFFFFFF and the high colorkey to 0x000000.

13.6.12.48 Overlay Color Key Low (PXP_HW_PXP_AS_CLRKEYLOW_1)

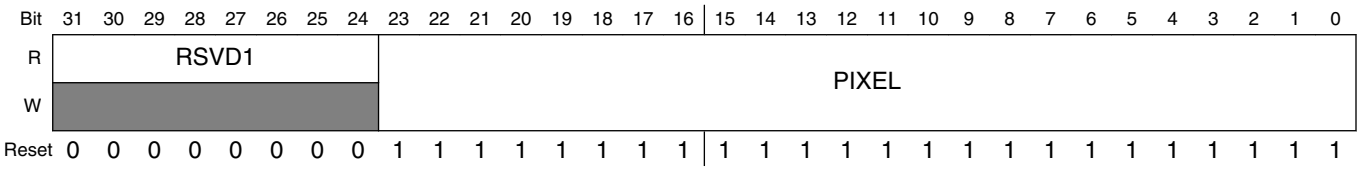
This register contains the color key low value for the AS buffer. This value is used by the Alpha B block.

When processing an image, the if the PXP finds a pixel in the current overlay image with a color that falls in the range from the ASCOLORKEYLOW to ASCOLORKEYHIGH range, it will use the PS pixel value for that location. If no PS image is present or if the PS image also matches its colorkey range, the PS background color is used. Colorkey operations are higher priority than alpha or ROP operations.

EXAMPLE

```
                                    // colorkey values between
REG_AS_CLRKEYLOW_WR (0x000000); // black and
REG_AS_CLRKEYHIGH_WR(0x800000); // medium red
```

Address: 3070_0000h base + 2F0h offset = 3070_02F0h



PXP_HW_PXP_AS_CLRKEYLOW_1 field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	Low range of RGB color key applied to AS buffer. Each overlay has an independent colorkey enable.

13.6.12.49 Overlay Color Key High (PXP_HW_PXP_AS_CLRKEYHIGH_1)

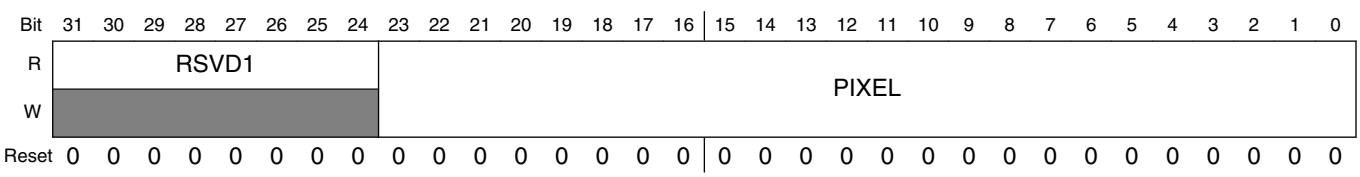
This register contains the color key high value for the AS buffer. This value is used by the Alpha B block.

When processing an image, the if the PXP finds a pixel in the current overlay image with a color that falls in the range from the ASCOLORKEYLOW to ASCOLORKEYHIGH range, it will use the PS pixel value for that location. If no PS image is present or if the PS image also matches its colorkey range, the PS background color is used. Colorkey operations are higher priority than alpha or ROP operations.

EXAMPLE

```
                                    // colorkey values between
REG_AS_CLRKEYLOW_WR (0x000000); // black and
REG_AS_CLRKEYHIGH_WR(0x800000); // medium red
```

Address: 3070_0000h base + 300h offset = 3070_0300h



PXP_HW_PXP_AS_CLRKEYHIGH_1 field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
PIXEL	High range of RGB color key applied to AS buffer. Each overlay has an independent colorkey enable.

13.6.12.50 Control Register 2 (PXP_HW_PXP_CTRL2)

The CTRL register contains controls for the secondary processing flow in PXP module.

HW_PXP_CTRL2: 0x310

HW_PXP_CTRL2_SET: 0x314

HW_PXP_CTRL2_CLR: 0x318

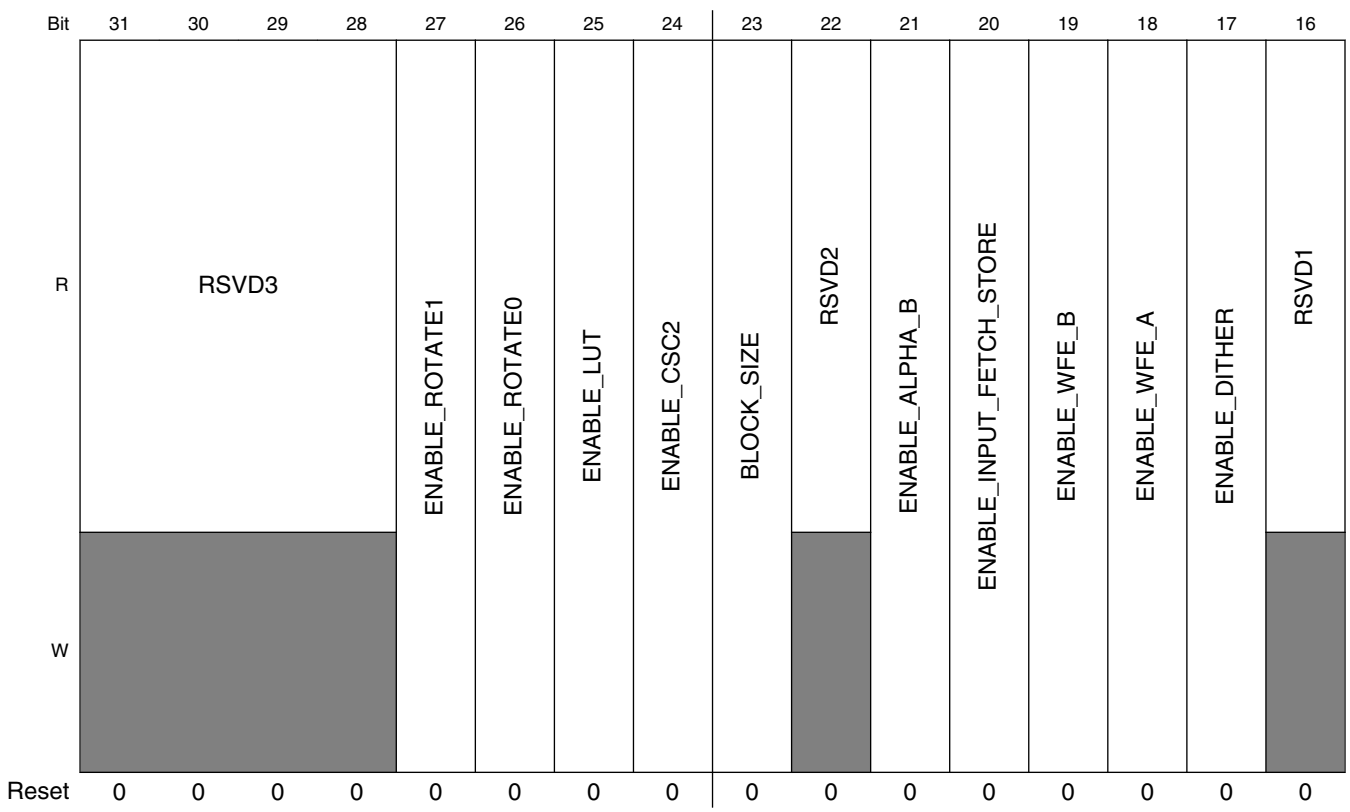
HW_PXP_CTRL2_TOG: 0x31C

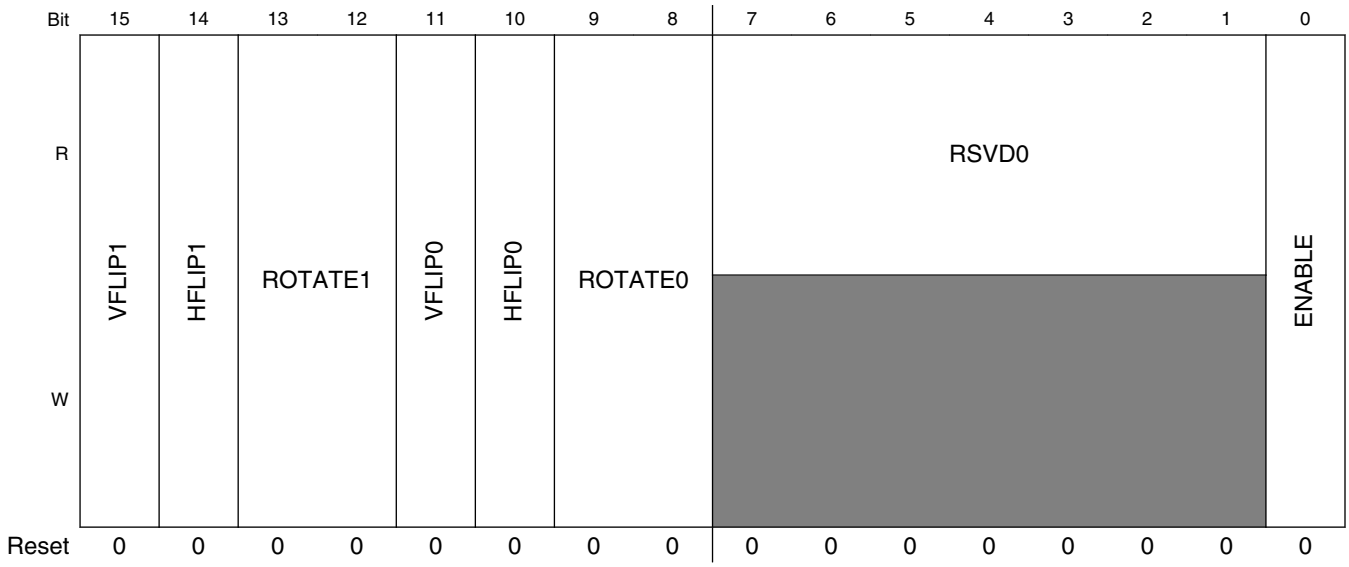
The Control register contains the controls for the secondary data flow in PXP block.

EXAMPLE

```
REG_CTRL_SET(BM_PXP_CTRL_SFTRST);
REG_CTRL_CLR(BM_PXP_CTRL_SFTRST | BM_PXP_CTRL_CLKGATE);
```

Address: 3070_0000h base + 310h offset = 3070_0310h





PXP_HW_PXP_CTRL2 field descriptions

Field	Description
31–28 RSVD3	Reserved, always set to zero.
27 ENABLE_ ROTATE1	Enable the ROTATE1 engine in the PXP secondary processing flow.
26 ENABLE_ ROTATE0	Enable the ROTATE0 engine in the PXP secondary processing flow.
25 ENABLE_LUT	Enable the LUT engine in the PXP secondary processing flow.
24 ENABLE_CSC2	Enable the CSC2 engine in the PXP secondary processing flow.
23 BLOCK_SIZE	Select the block size to process through the Rotate block. 0x0 **8X8** — Process 8x8 pixel blocks. 0x1 **16X16** — Process 16x16 pixel blocks.
22 RSVD2	Reserved, always set to zero.
21 ENABLE_ ALPHA_B	Enable the Alpha-B engine in the PXP secondary processing flow.
20 ENABLE_ INPUT_FETCH_ STORE	Enable the Input Fetch and Store engine in the PXP secondary processing flow.
19 ENABLE_WFE_B	Enable the WFE-B engine in the PXP secondary processing flow.
18 ENABLE_WFE_A	Enable the WFE-A engine in the PXP secondary processing flow.

Table continues on the next page...

PXP_HW_PXP_CTRL2 field descriptions (continued)

Field	Description
17 ENABLE_ DITHER	Enable the Dithering engine in the PXP secondary processing flow.
16 RSVD1	Reserved, always set to zero.
15 VFLIP1	Indicates that the input should be flipped vertically (effect applied before rotation).
14 HFLIP1	Indicates that the input should be flipped horizontally (effect applied before rotation).
13–12 ROTATE1	Indicates the clockwise rotation to be applied at the input buffer. The rotation effect is defined as occurring after the FLIP_X and FLIP_Y permutation. 0x0 **ROT_0** — 0x1 **ROT_90** — 0x2 **ROT_180** — 0x3 **ROT_270** —
11 VFLIP0	Indicates that the output buffer should be flipped vertically (effect applied before rotation).
10 HFLIP0	Indicates that the output buffer should be flipped horizontally (effect applied before rotation).
9–8 ROTATE0	Indicates the clockwise rotation to be applied at the output buffer. The rotation effect is defined as occurring after the FLIP_X and FLIP_Y permutation. 0x0 **ROT_0** — 0x1 **ROT_90** — 0x2 **ROT_180** — 0x3 **ROT_270** —
7–1 RSVD0	Reserved, always set to zero.
0 ENABLE	Enables PXP secondary data processing flow with specified parameters. The ENABLE bit will remain set while the PXP is active and will be cleared once the current operation completes. Software should use the IRQ bit in the HW_PXP_STAT when polling for PXP completion.

13.6.12.51 PXP Power Control Register. (PXP_HW_PXP_POWER_REG0)

This register controls power states for PXP memories.

Address: 3070_0000h base + 320h offset = 3070_0320h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTRL																				ROT0_ MEM_ LP_ STATE			LUT_LP_ STATE_ WAY1_ BANKN			LUT_LP_ STATE_ WAY0_ BANKN			LUT_LP_ STATE_ WAY0_ BANK0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_POWER_REG0 field descriptions

Field	Description
31–12 CTRL	This register contains power control for the PXP.
11–9 ROT0_MEM_LP_ STATE	Select the low power state of the ROT 0 memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.
8–6 LUT_LP_ STATE_WAY1_ BANKN	Select the low power state of the LUT's WAY0-BANK0,1,2,3 memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.
5–3 LUT_LP_ STATE_WAY0_ BANKN	Select the low power state of the LUT's WAY0-BANK1,2,3 memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.
LUT_LP_ STATE_WAY0_ BANK0	Select the low power state of the LUT's WAY0-BANK0 memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.

13.6.12.52 PXP Power Control Register 1. (PXP_HW_PXP_POWER_REG1)

This register controls power states for PXP memories.

Address: 3070_0000h base + 330h offset = 3070_0330h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0								ALU_B_MEM_LP_STATE			ALU_A_MEM_LP_STATE			DITH2_LUT_MEM_LP_STATE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DITH2_LUT_MEM_LP_STATE	DITH1_LUT_MEM_LP_STATE			DITH0_ERR1_MEM_LP_STATE			DITH0_ERR0_MEM_LP_STATE			DITH0_LUT_MEM_LP_STATE			ROT1_MEM_LP_STATE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_POWER_REG1 field descriptions

Field	Description
31–24 RSVD0	This register contains power control for the PXP.
23–21 ALU_B_MEM_LP_STATE	Select the low power state of the ALU B memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.
20–18 ALU_A_MEM_LP_STATE	Select the low power state of the ALU A memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.
17–15 DITH2_LUT_MEM_LP_STATE	Select the low power state of the dither2 LUT memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.
14–12 DITH1_LUT_MEM_LP_STATE	Select the low power state of the dither1 LUT memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.
11–9 DITH0_ERR1_MEM_LP_STATE	Select the low power state of the dither0 ERR1 memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.
8–6 DITH0_ERR0_MEM_LP_STATE	Select the low power state of the dither0 ERR0 memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.

Table continues on the next page...

PXP_HW_PXP_POWER_REG1 field descriptions (continued)

Field	Description
5–3 DITH0_LUT_ MEM_LP_STATE	Select the low power state of the dither0 LUT memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.
ROT1_MEM_LP_ STATE	Select the low power state of the ROT 1 memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.

13.6.12.53 PXP_HW_PXP_DATA_PATH_CTRL1

This register helps decide the data path gthrough the PXP.

HW_PXP_DATA_PATH_CTRL1: 0x350

HW_PXP_DATA_PATH_CTRL1_SET: 0x354

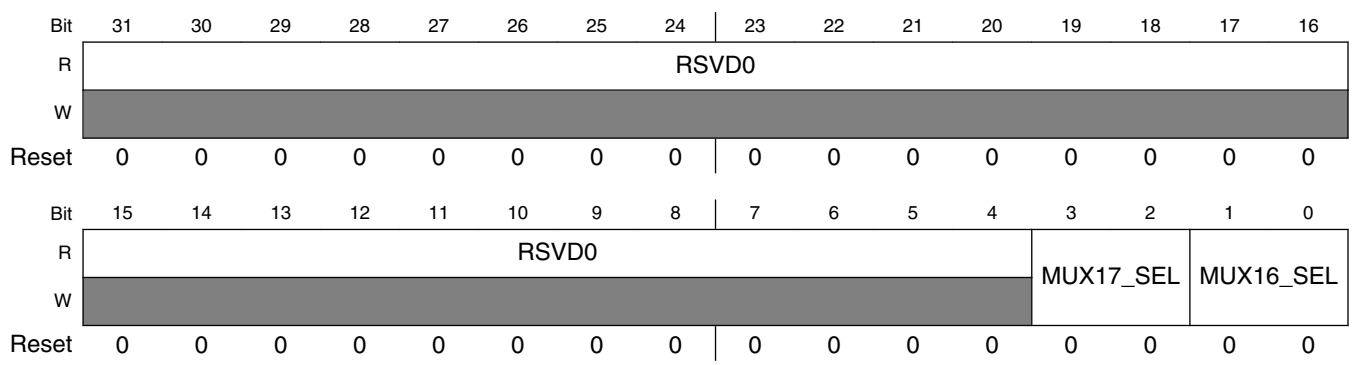
HW_PXP_DATA_PATH_CTRL1_CLR: 0x358

HW_PXP_DATA_PATH_CTRL1_TOG: 0x35C

The Control register contains the control bits for the data path through the PXP.

EXAMPLE

Address: 3070_0000h base + 350h offset = 3070_0350h



PXP_HW_PXP_DATA_PATH_CTRL1 field descriptions

Field	Description
31–4 RSVD0	Reserved. This field always reads 0.
3–2 MUX17_SEL	This field chooses the data path through MUX 17. 0x0 **0** — Output of ALU A 0x1 **1** — Output of ALU B 0x2 **2** — No output 0x3 **3** — No Output
MUX16_SEL	This mux chooses the data path through MUX 16. 0x0 **0** — Output of ALU A Engine 0x1 **1** — histogram_pixel output from output 0x2 **2** — Output of ALU B Engine 0x3 **3** — No output

13.6.12.54 Initialize memory buffer control Register (PXP_HW_PXP_INIT_MEM_CTRL)

Controls initializing/writing to dither and ALU memory buffers.

HW_PXP_INIT_MEM_CTRL: 0x360

HW_PXP_INIT_MEM_CTRL_SET: 0x364

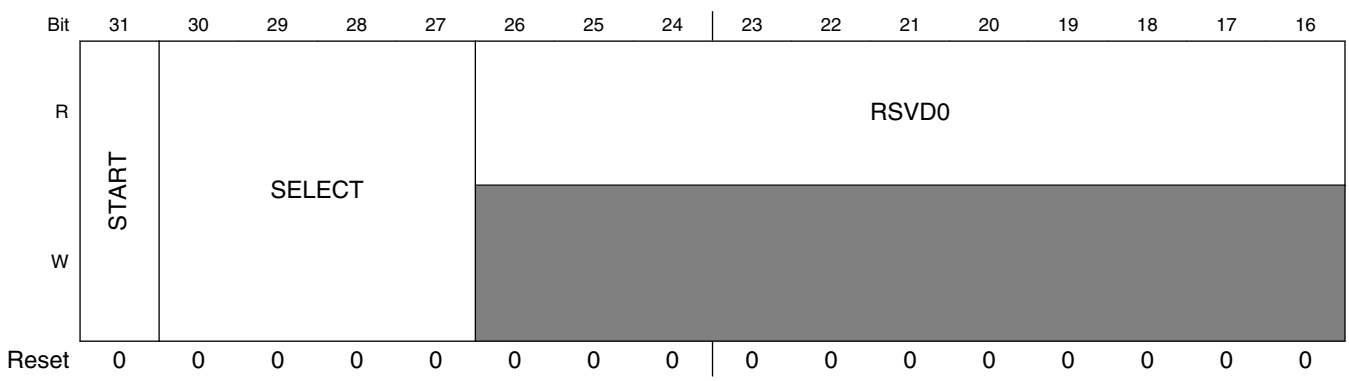
HW_PXP_INIT_MEM_CTRL_CLR: 0x368

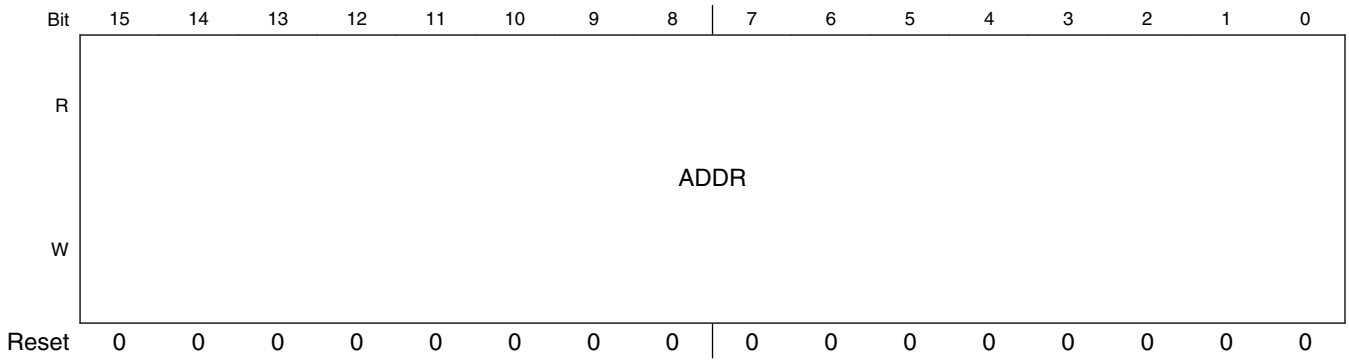
HW_PXP_INIT_MEM_CTRL_TOG: 0x36C

This register controls IRQ the intializing of internal pxp rams.

EXAMPLE

Address: 3070_0000h base + 360h offset = 3070_0360h





PXP_HW_PXP_INIT_MEM_CTRL field descriptions

Field	Description
31 START	Enable writing to the memory.
30–27 SELECT	Select which memory to write. 0x0 **DITHER0_LUT** — Select the LUT memory for access 0x1 **DITHER0_ERR0** — Select the ERR0 memory for access 0x2 **DITHER0_ERR1** — Select the ERR1 memory for access 0x3 **DITHER1_LUT** — Select the LUT memory for access 0x4 **DITHER2_LUT** — Select the LUT memory for access 0x5 **ALU_A** — Select the ALU instr memory for access 0x6 **ALU_B** — Select the ALU instr memory for access 0x7 **WFE_A_FETCH** — Select the WFE-A fetch memory for access 0x8 **WFE_B_FETCH** — Select the WFE-B fetch memory for access
26–16 RSVD0	Reserved.
ADDR	Base address to start writing. The control logic will increment the address value internally each time the data register is written.

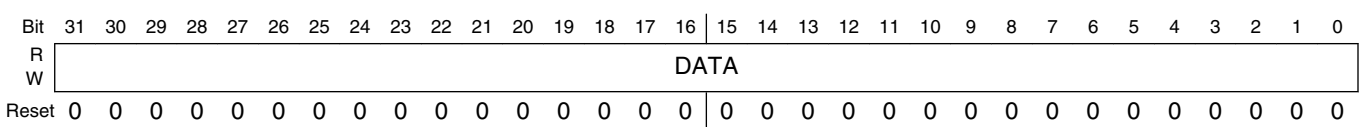
13.6.12.55 Write data Register (PXP_HW_PXP_INIT_MEM_DATA)

Write data for initializing/writing to dither, ALU and fetch memory buffers.

This register holds the data word to intialize internal pxp rams.

EXAMPLE

Address: 3070_0000h base + 370h offset = 3070_0370h



PXP_HW_PXP_INIT_MEM_DATA field descriptions

Field	Description
DATA	Data value to be written to the memory. A write to this register kicks off a write cycle to the memory selected.

13.6.12.56 Write data Register (PXP_HW_PXP_INIT_MEM_DATA_HIGH)

Write data for initializing/writing to the most significant 32 bits of the fetch memory buffers.

This register holds the upper data word to intialize internal pxp fetch rams.

EXAMPLE

Address: 3070_0000h base + 380h offset = 3070_0380h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INIT_MEM_DATA_HIGH field descriptions

Field	Description
DATA	Data value to be written to the most significant 32 bits of the fetch memories. this register must be written before the HW_PXP_INIT_MEM_DATA as that register triggers the write cycle to memory.

13.6.12.57 PXP IRQ Mask Register (PXP_HW_PXP_IRQ_MASK)

Controls masking for all PXP interrupts

HW_PXP_IRQ_MASK: 0x390

HW_PXP_IRQ_MASK_SET: 0x394

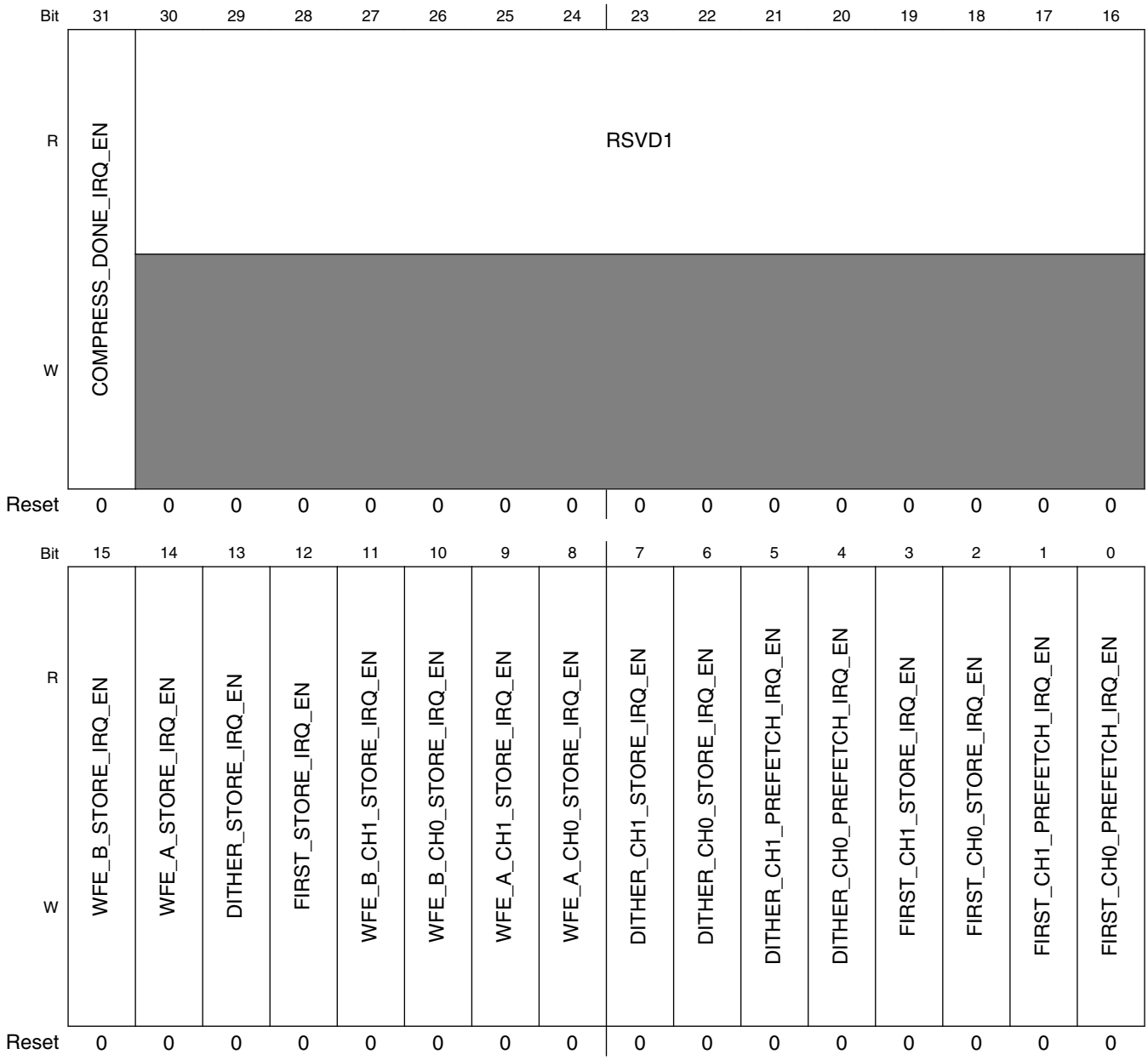
HW_PXP_IRQ_MASK_CLR: 0x398

HW_PXP_IRQ_MASK_TOG: 0x39C

This register controls IRQ masks for all PXP interrupts

EXAMPLE

Address: 3070_0000h base + 390h offset = 3070_0390h



PXP_HW_PXP_IRQ_MASK field descriptions

Field	Description
31 COMPRESS_ DONE_IRQ_EN	Enable compression done interrupt detection.
30–16 RSVD1	Reserved.
15 WFE_B_ STORE_IRQ_EN	Enable WFE B store engine interrupt detection.

Table continues on the next page...

PXP_HW_PXP_IRQ_MASK field descriptions (continued)

Field	Description
14 WFE_A_ STORE_IRQ_EN	Enable WFE A store engine interrupt detection.
13 DITHER_ STORE_IRQ_EN	Enable dither store engine interrupt detection.
12 FIRST_STORE_ IRQ_EN	Enable First store engine interrupt detection
11 WFE_B_CH1_ STORE_IRQ_EN	Enable WFE B ch1 store engine interrupt detection.
10 WFE_B_CH0_ STORE_IRQ_EN	Enable WFE B ch0 store engine interrupt detection.
9 WFE_A_CH1_ STORE_IRQ_EN	Enable WFE A ch1 store engine interrupt detection.
8 WFE_A_CH0_ STORE_IRQ_EN	Enable WFE A ch0 store engine interrupt detection.
7 DITHER_CH1_ STORE_IRQ_EN	Enable dither ch1 store engine interrupt detection.
6 DITHER_CH0_ STORE_IRQ_EN	Enable dither ch0 store engine interrupt detection.
5 DITHER_CH1_ PREFETCH_ IRQ_EN	Enable Dither ch1 prefetch engine interrupt detection
4 DITHER_CH0_ PREFETCH_ IRQ_EN	Enable Dither ch0 prefetch engine interrupt detection
3 FIRST_CH1_ STORE_IRQ_EN	Enable First ch1 store engine interrupt detection
2 FIRST_CH0_ STORE_IRQ_EN	Enable First ch0 store engine interrupt detection
1 FIRST_CH1_ PREFETCH_ IRQ_EN	Enable First ch1 prefetch engine interrupt detection
0 FIRST_CH0_ PREFETCH_ IRQ_EN	Enable First ch0 prefetch engine interrupt detection

13.6.12.58 PXP Interrupt Register (PXP_HW_PXP_IRQ)

PXP Fetch and Store engine IRQs. Each interrupt has a corresponding mask register in HW_EPDC_IRQ_MASK

HW_PXP_IRQ: 0x3A0

HW_PXP_IRQ_SET: 0x3A4

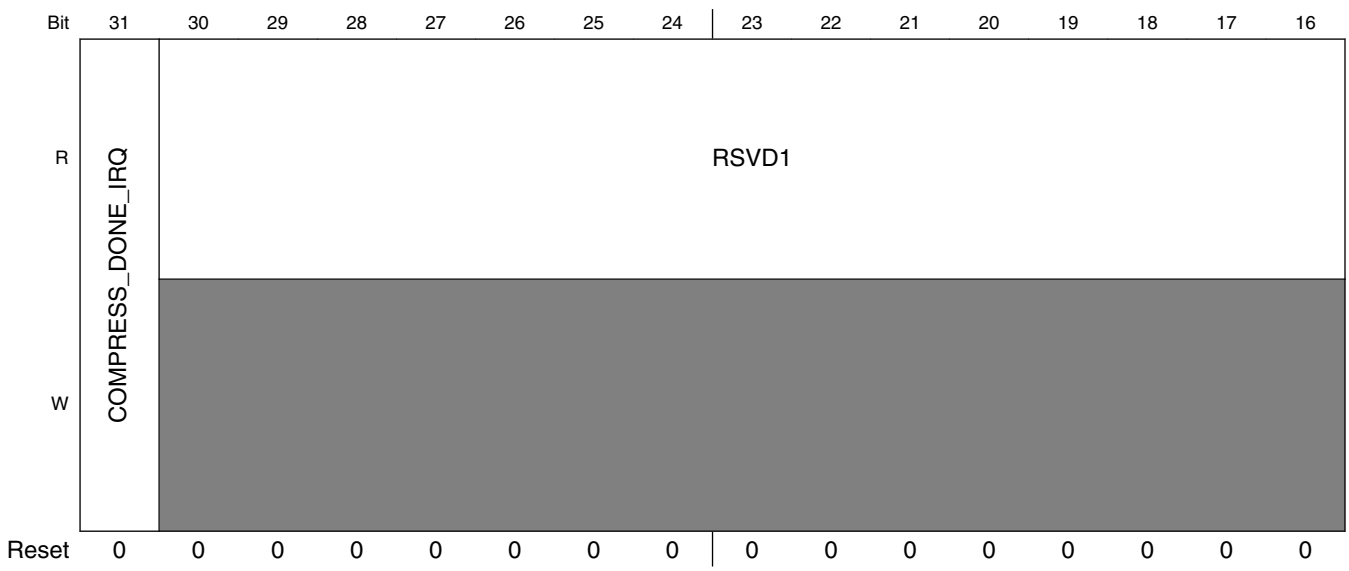
HW_PXP_IRQ_CLR: 0x3a8

HW_PXP_IRQ_TOG: 0x3AC

This register houses the interrupt bits for the Prefetch and Store Engines

EXAMPLE

Address: 3070_0000h base + 3A0h offset = 3070_03A0h



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WFE_B_STORE_IRQ	WFE_A_STORE_IRQ	DITHER_STORE_IRQ	FIRST_STORE_IRQ	WFE_B_CH1_STORE_IRQ	WFE_B_CH0_STORE_IRQ	WFE_A_CH1_STORE_IRQ	WFE_A_CH0_STORE_IRQ	DITHER_CH1_STORE_IRQ	DITHER_CH0_STORE_IRQ	DITHER_CH1_PREFETCH_IRQ	DITHER_CH0_PREFETCH_IRQ	FIRST_CH1_STORE_IRQ	FIRST_CH0_STORE_IRQ	FIRST_CH1_PREFETCH_IRQ	FIRST_CH0_PREFETCH_IRQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_IRQ field descriptions

Field	Description
31 COMPRESS_ DONE_IRQ	compression done Interrupt
30–16 RSVD1	Reserved.
15 WFE_B_ STORE_IRQ	WFE B store engine Interrupt
14 WFE_A_ STORE_IRQ	WFE A store engine Interrupt.
13 DITHER_ STORE_IRQ	Dither store engine Interrupt
12 FIRST_STORE_ IRQ	Initial store engine interrupt
11 WFE_B_CH1_ STORE_IRQ	WFE B ch1 store engine Interrupt
10 WFE_B_CH0_ STORE_IRQ	WFE B ch0 store engine Interrupt
9 WFE_A_CH1_ STORE_IRQ	WFE A ch1 store engine Interrupt.
8 WFE_A_CH0_ STORE_IRQ	WFE A ch0 store engine Interrupt.

Table continues on the next page...

PXP_HW_PXP_IRQ field descriptions (continued)

Field	Description
7 DITHER_CH1_ STORE_IRQ	Dither ch1 store engine Interrupt
6 DITHER_CH0_ STORE_IRQ	Dither ch0 store engine Interrupt
5 DITHER_CH1_ PREFETCH_IRQ	Dither ch1 prefetch engine interrupt
4 DITHER_CH0_ PREFETCH_IRQ	Dither ch0 prefetch engine interrupt
3 FIRST_CH1_ STORE_IRQ	Initial ch1 store engine interrupt
2 FIRST_CH0_ STORE_IRQ	Initial ch0 store engine interrupt
1 FIRST_CH1_ PREFETCH_IRQ	Initial ch1 prefetch engine interrupt
0 FIRST_CH0_ PREFETCH_IRQ	Initial ch0 prefetch engine interrupt

13.6.12.59 Next Frame Pointer (PXP_HW_PXP_NEXT)

This register contains a pointer to a data structure used to reload the PXP registers at the end of the current frame.

To enable this functionality, software must write this register while the PXP is processing the current data frame (if the PXP is currently idle, this will also initiate an immediate load of registers from the pointer). The process of writing this register (WRITE operation) will set a semaphore in hardware to notify the control logic that a register reload operation must be performed when the current frame processing is complete. At the end of a frame, the PXP will fetch the register settings from this location, signal an interrupt to software, then proceed with rendering the next frame of data. Software may cancel the reload operation by issuing a CLEAR operation to this register. SET and TOGGLE operations should not be used when addressing this register. All registers will be reloaded with the exception of the following: STAT, CSCCOEFn, NEXT, VERSION. All other registers will be loaded in the order they appear in the register map. Once the pointer's contents have been loaded into the PXP's registers, the NEXT_IRQ interrupt will be issued (see the PXP_STATUS register).

EXAMPLE

```
// create register command structure in memory
  u32* pxp_commands0[48], pxp_commands1;
  u32  rc;

// initialize control structure for frame 0
  pxp_commands0[0] = ...; // CTRL
  pxp_commands0[1] = ...; // OUT Buffer
  ...
  pxp_commands0[47] = ..; // Overlay7 param2

// initialize control structure for frame 1
  pxp_commands1[0] = ...; // CTRL
  pxp_commands1[1] = ...; // OUT Buffer
  ...
  pxp_commands1[47] = ..; // Overlay7 param2

// poll until a command isn't queued
  while (rc=REG_NEXT_RD() & BM_PXP_NEXT_ENABLED );
  REG_NEXT_WR(pxp_commands0);  // enable PXP operation 0 via command pointer

// poll until first command clears
  while (rc=REG_NEXT_RD() & BM_PXP_NEXT_ENABLED );
  REG_NEXT_WR(pxp_commands1);  // enable PXP operation 1 via command pointer
```

Address: 3070_0000h base + 400h offset = 3070_0400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	POINTER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	POINTER														RSVD	ENABLED
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_NEXT field descriptions

Field	Description
31–2 POINTER	A pointer to a data structure containing register values to be used when processing the next frame. The pointer must be 32-bit aligned and should reside in on-chip or off-chip memory.
1 RSVD	Reserved, always set to zero.
0 ENABLED	Indicates that the "next frame" functionality has been enabled. This bit reflects the status of the hardware semaphore indicating that a reload operation is pending at the end of the current frame.

13.6.12.60 Pre-fetch engine Control Channel 0 Register (PXP_HW_PXP_INPUT_FETCH_CTRL_CH0)

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_INPUT_FETCH_CTRL_CH0: 0x450

HW_PXP_INPUT_FETCH_CTRL_CH0_SET: 0x454

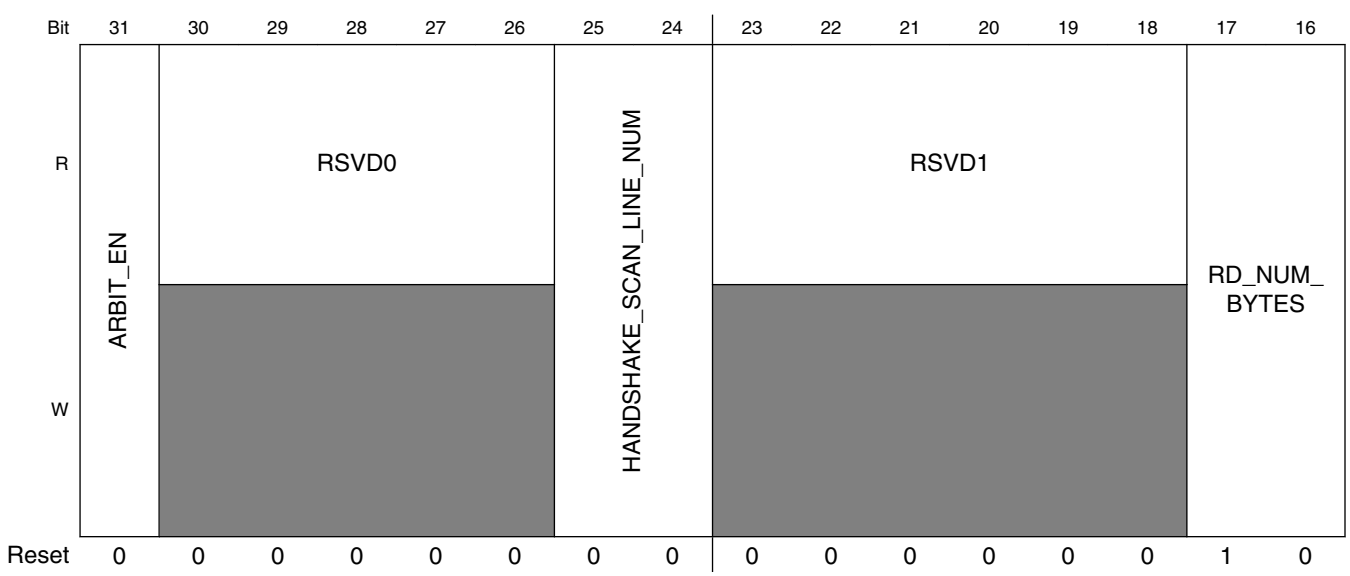
HW_PXP_INPUT_FETCH_CTRL_CH0_CLR: 0x458

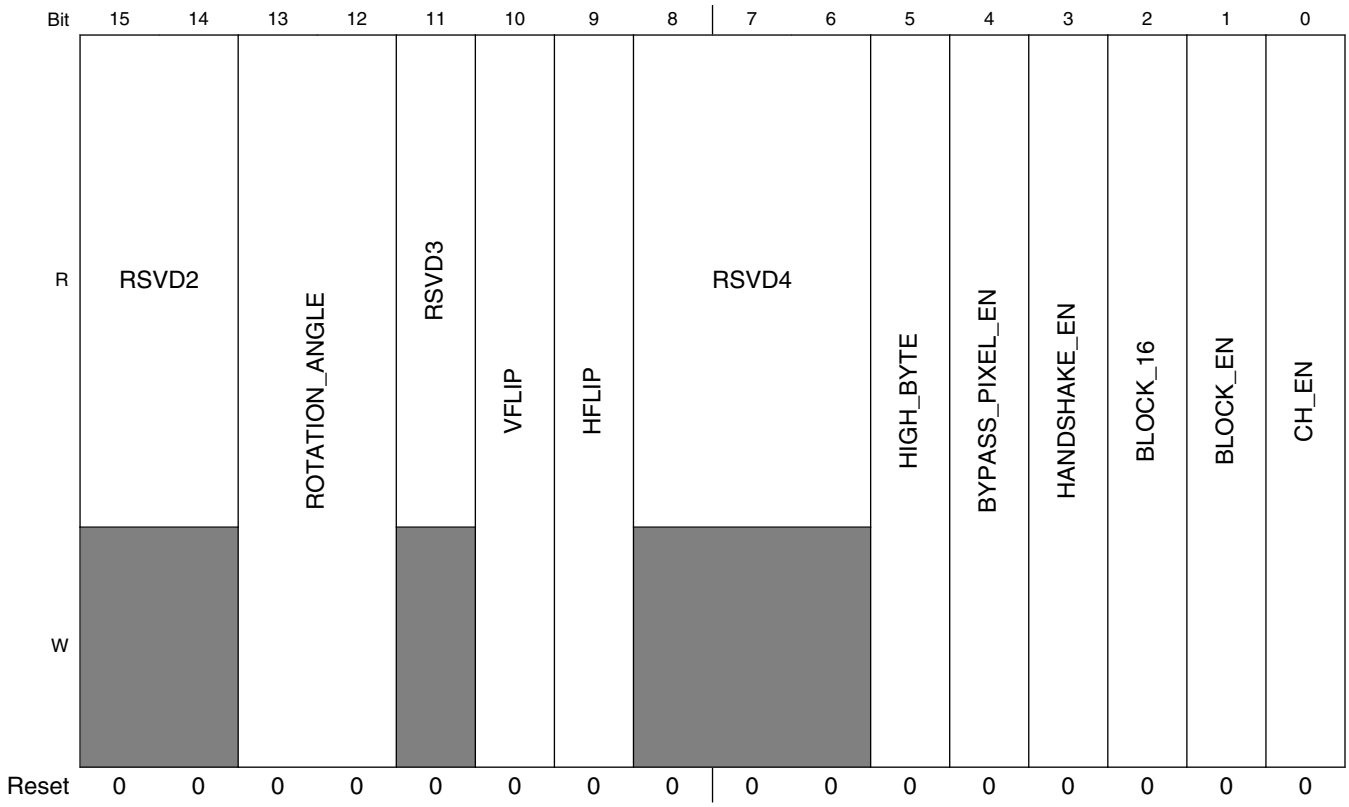
HW_PXP_INPUT_FETCH_CTRL_CH0_TOG: 0x45C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 450h offset = 3070_0450h





PXP_HW_PXP_INPUT_FETCH_CTRL_CH0 field descriptions

Field	Description
31 ARBIT_EN	Enables Arbitration 0x0 **0** — Arbitration disable. If using 2 channels, will output 2 axi bus sets. 0x1 **1** — Arbitration enable. If using 2 channel, will only output 1 axi bus sets
30–26 RSVD0	Reserved, always set to zero.
25–24 HANDSHAKE_ SCAN_LINE_ NUM	scan handshake line number 0x0 **0** — 1 line. 0x1 **1** — 8 lines 0x2 **2** — 16 lines 0x3 **3** — 16 lines
23–18 RSVD1	Reserved, always set to zero.
17–16 RD_NUM_ BYTES	Bytes in a read burst 0x0 **8_bytes** — 8 bytes. 0x1 **16_bytes** — 16 bytes. 0x2 **32_bytes** — 32 bytes. 0x3 **64_bytes** — 64 bytes.
15–14 RSVD2	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_INPUT_FETCH_CTRL_CH0 field descriptions (continued)

Field	Description
13–12 ROTATION_ ANGLE	0x0 **ROT_0** — Rotate image by 0 degrees. 0x1 **ROT_90** — Rotate image by 90 degrees. 0x2 **ROT_180** — Rotate image by 180 degrees. 0x3 **ROT_270** — Rotate image by 270 degrees.
11 RSVD3	Reserved, always set to zero.
10 VFLIP	Enables VFLIP 0x0 **0** — VFLIP disable 0x1 **1** — VFLIP enable
9 HFLIP	Enables HFLIP. 0x0 **0** — HFLIP disable 0x1 **1** — VFLIP enable
8–6 RSVD4	Reserved, always set to zero.
5 HIGH_BYTE	channel 0 high byte selection 0x0 **0** — In 64 bit mode, the output high byte will use channel1. 0x1 **1** — In 64 bit mode, the output high byte will use channel0
4 BYPASS_ PIXEL_EN	Selects Channel 0 pixel source 0x0 **0** — Channel 0 is from memory 0x1 **1** — Channel 0 is from previous process engine
3 HANDSHAKE_ EN	Enable bit for handshake with the store engine. 0x0 **0** — Handshake with the store engine is disabled 0x1 **1** — Handshake with the store engine is enabled
2 BLOCK_16	Determines the block sixe. 0x0 **8x8** — Block size is 8x8 0x1 **16x16** — Block size is 16x16
1 BLOCK_EN	Choses the prefetch mode. 0x0 **0** — Prefetch in scan mode 0x1 **1** — Prefetch in block mode
0 CH_EN	Channel enable. 0x0 **0** — Prefetch function is disable 0x1 **1** — Prefetch function is enable

13.6.12.61 Pre-fetch engine Control Channel 1 Register (PXP_HW_PXP_INPUT_FETCH_CTRL_CH1)

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_INPUT_FETCH_CTRL_CH1: 0x460

HW_PXP_INPUT_FETCH_CTRL_CH1_SET: 0x464

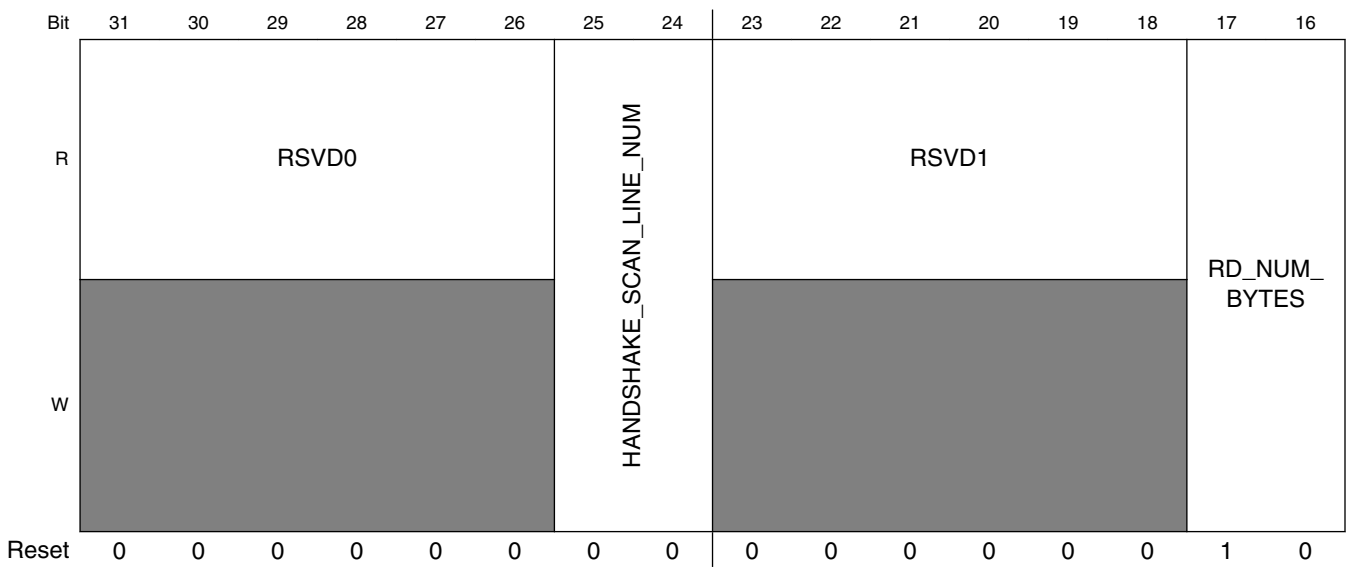
HW_PXP_INPUT_FETCH_CTRL_CH1_CLR: 0x468

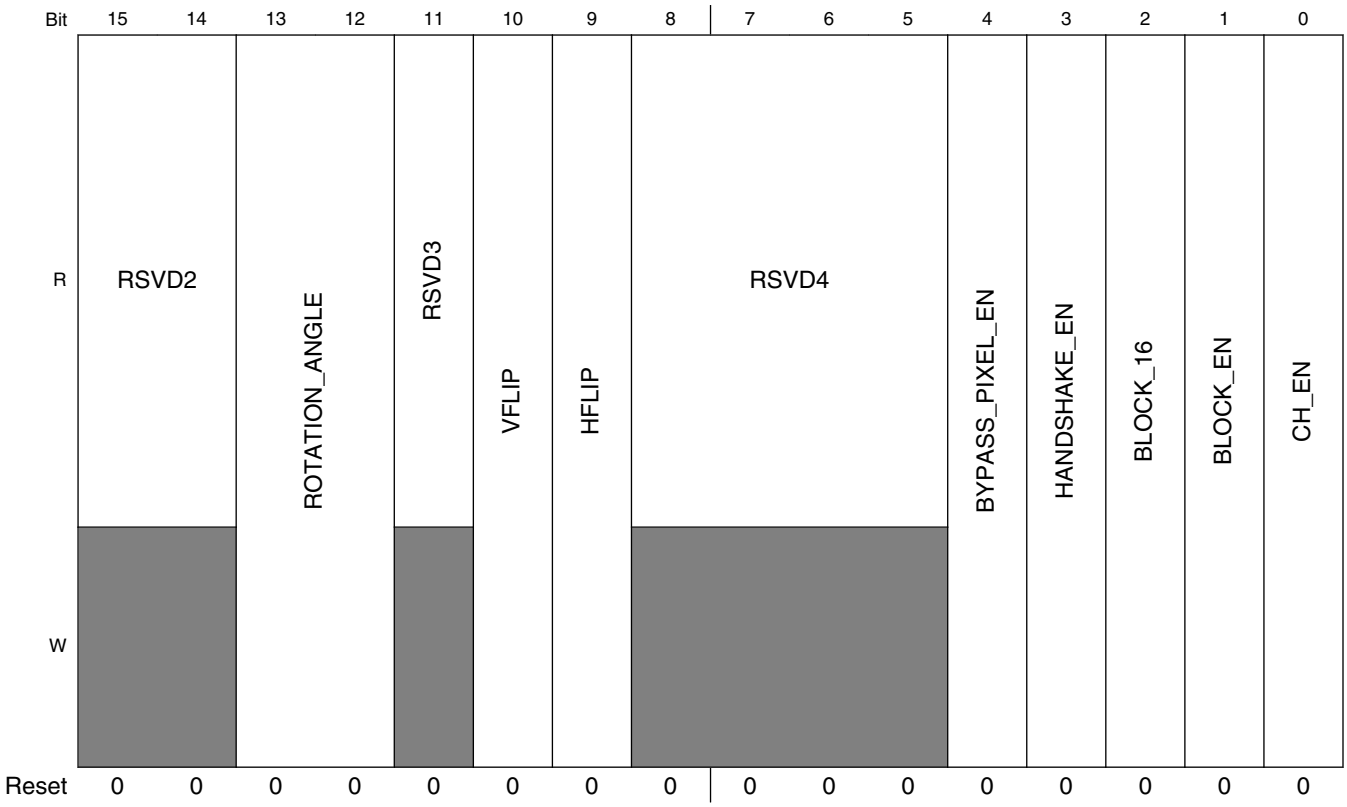
HW_PXP_INPUT_FETCH_CTRL_CH1_TOG: 0x46C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 460h offset = 3070_0460h





PXP_HW_PXP_INPUT_FETCH_CTRL_CH1 field descriptions

Field	Description
31–26 RSVD0	Reserved, always set to zero.
25–24 HANDSHAKE_ SCAN_LINE_ NUM	scan handshake line number 0x0 **0** — 1 line. 0x1 **1** — 8 lines 0x2 **2** — 16 lines 0x3 **3** — 16 lines
23–18 RSVD1	Reserved, always set to zero.
17–16 RD_NUM_ BYTES	Bytes in a read burst 0x0 **8_bytes** — 8 bytes. 0x1 **16_bytes** — 16 bytes. 0x2 **32_bytes** — 32 bytes. 0x3 **64_bytes** — 64 bytes.
15–14 RSVD2	Reserved, always set to zero.
13–12 ROTATION_ ANGLE	0x0 **ROT_0** — Rotate image by 0 degrees. 0x1 **ROT_90** — Rotate image by 90 degrees. 0x2 **ROT_180** — Rotate image by 180 degrees. 0x3 **ROT_270** — Rotate image by 270 degrees.

Table continues on the next page...

PXP_HW_PXP_INPUT_FETCH_CTRL_CH1 field descriptions (continued)

Field	Description
11 RSVD3	Reserved, always set to zero.
10 VFLIP	Enables VFLIP 0x0 **0** — VFLIP disable 0x1 **1** — VFLIP enable
9 HFLIP	Enables HFLIP. 0x0 **0** — HFLIP disable 0x1 **1** — VFLIP enable
8–5 RSVD4	Reserved, always set to zero.
4 BYPASS_ PIXEL_EN	Selects Channel 1 pixel source 0x0 **0** — Channel 1 is from memory 0x1 **1** — Channel 1 is from previous process engine
3 HANDSHAKE_ EN	Enable bit for handshake with the store engine. 0x0 **0** — Handshake with the store engine is disabled 0x1 **1** — Handshake with the store engine is enabled
2 BLOCK_16	Determines the block sixe. 0x0 **8x8** — Block size is 8x8 0x1 **16x16** — Block size is 16x16
1 BLOCK_EN	Choses the prefetch mode. 0x0 **0** — Prefetch in scan mode 0x1 **1** — Prefetch in block mode
0 CH_EN	Channel enable. 0x0 **0** — prefetch function is disable 0x1 **1** — prefetch function is enable

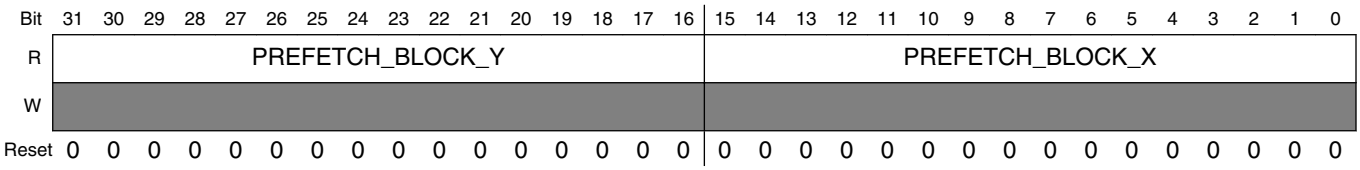
13.6.12.62 Pre-fetch engine status Channel 0 Register (PXP_HW_PXP_INPUT_FETCH_STATUS_CH0)

This register defines the status bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 470h offset = 3070_0470h



PXP_HW_PXP_INPUT_FETCH_STATUS_CH0 field descriptions

Field	Description
31–16 PREFETCH_ BLOCK_Y	When in scan mode, this field indicates the current Y coordinate of the frame. In block mode, it indicates the Y coordinate of the block currently being rendered.
PREFETCH_ BLOCK_X	When in scan mode, this field is always 0. In block mode, it indicates the X coordinate of the block currently being rendered.

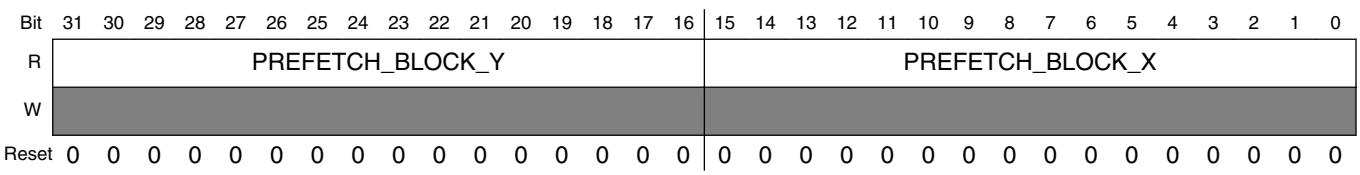
13.6.12.63 Store engine status Channel 1 Register (PXP_HW_PXP_INPUT_FETCH_STATUS_CH1)

This register defines the status bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 480h offset = 3070_0480h



PXP_HW_PXP_INPUT_FETCH_STATUS_CH1 field descriptions

Field	Description
31–16 PREFETCH_ BLOCK_Y	When in scan mode, this field indicates the current Y coordinate of the frame. In block mode, it indicates the Y coordinate of the block currently being rendered.
PREFETCH_ BLOCK_X	When in scan mode, this field is always 0. In block mode, it indicates the X coordinate of the block currently being rendered.

13.6.12.64 PXP_HW_PXP_INPUT_FETCH_ACTIVE_SIZE_ULC_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 490h offset = 3070_0490h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ACTIVE_SIZE_ULC_Y																ACTIVE_SIZE_ULC_X															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_FETCH_ACTIVE_SIZE_ULC_CH0 field descriptions

Field	Description
31–16 ACTIVE_SIZE_ ULC_Y	This field indicates the upper left Y-coordinate(in pixels) of the active surface of the total input memory
ACTIVE_SIZE_ ULC_X	This field indicates the upper left X-coordinate(in pixels) of the active surface of the total input memory

13.6.12.65 PXP_HW_PXP_INPUT_FETCH_ACTIVE_SIZE_LRC_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 4A0h offset = 3070_04A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ACTIVE_SIZE_LRC_Y																ACTIVE_SIZE_LRC_X															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_FETCH_ACTIVE_SIZE_LRC_CH0 field descriptions

Field	Description
31–16 ACTIVE_SIZE_ LRC_Y	This field indicates the upper left Y-coordinate(in pixels) of the active surface of the total input memory
ACTIVE_SIZE_ LRC_X	This field indicates the upper left X-coordinate(in pixels) of the active surface of the total input memory

13.6.12.66 PXP_HW_PXP_INPUT_FETCH_ACTIVE_SIZE_ULC_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 4B0h offset = 3070_04B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ACTIVE_SIZE_ULC_Y																ACTIVE_SIZE_ULC_X															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_FETCH_ACTIVE_SIZE_ULC_CH1 field descriptions

Field	Description
31–16 ACTIVE_SIZE_ ULC_Y	This field indicates the upper left Y-coordinate(in pixels) of the active surface of the total input memory
ACTIVE_SIZE_ ULC_X	This field indicates the upper left X-coordinate(in pixels) of the active surface of the total input memory

13.6.12.67 PXP_HW_PXP_INPUT_FETCH_ACTIVE_SIZE_LRC_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 4C0h offset = 3070_04C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ACTIVE_SIZE_LRC_Y																ACTIVE_SIZE_LRC_X															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_FETCH_ACTIVE_SIZE_LRC_CH1 field descriptions

Field	Description
31–16 ACTIVE_SIZE_ LRC_Y	This field indicates the upper left Y-coordinate(in pixels) of the active surface of the total input memory
ACTIVE_SIZE_ LRC_X	This field indicates the upper left X-coordinate(in pixels) of the active surface of the total input memory

13.6.12.68 PXP_HW_PXP_INPUT_FETCH_SIZE_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 4D0h offset = 3070_04D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_TOTAL_HEIGHT																INPUT_TOTAL_WIDTH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_FETCH_SIZE_CH0 field descriptions

Field	Description
31–16 INPUT_TOTAL_ HEIGHT	actual total height - 1
INPUT_TOTAL_ WIDTH	actual total width -1

13.6.12.69 PXP_HW_PXP_INPUT_FETCH_SIZE_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 4E0h offset = 3070_04E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_TOTAL_HEIGHT																INPUT_TOTAL_WIDTH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_FETCH_SIZE_CH1 field descriptions

Field	Description
31–16 INPUT_TOTAL_ HEIGHT	actual total height -1
INPUT_TOTAL_ WIDTH	actual total width -1

13.6.12.70 PXP_HW_PXP_INPUT_FETCH_BACKGROUND_COLOR_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 4F0h offset = 3070_04F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BACKGROUND_COLOR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_FETCH_BACKGROUND_COLOR_CH0 field descriptions

Field	Description
BACKGROUND_COLOR	background color(in 32bpp format) for any pixels not within the bufffer range specified by the ULC/LRC

13.6.12.71 PXP_HW_PXP_INPUT_FETCH_BACKGROUND_COLOR_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 500h offset = 3070_0500h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BACKGROUND_COLOR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_FETCH_BACKGROUND_COLOR_CH1 field descriptions

Field	Description
BACKGROUND_COLOR	background color(in 32bpp format) for any pixels not within the bufffer range specified by the ULC/LRC

13.6.12.72 PXP_HW_PXP_INPUT_FETCH_PITCH

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 510h offset = 3070_0510h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH1_INPUT_PITCH																CH0_INPUT_PITCH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_FETCH_PITCH field descriptions

Field	Description
31–16 CH1_INPUT_ PITCH	This field indicates the channel 1 input pitch
CH0_INPUT_ PITCH	This field indicates the channel 0 input pitch

13.6.12.73 PXP_HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH0: 0x520

HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH0_SET: 0x524

HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH0_CLR: 0x528

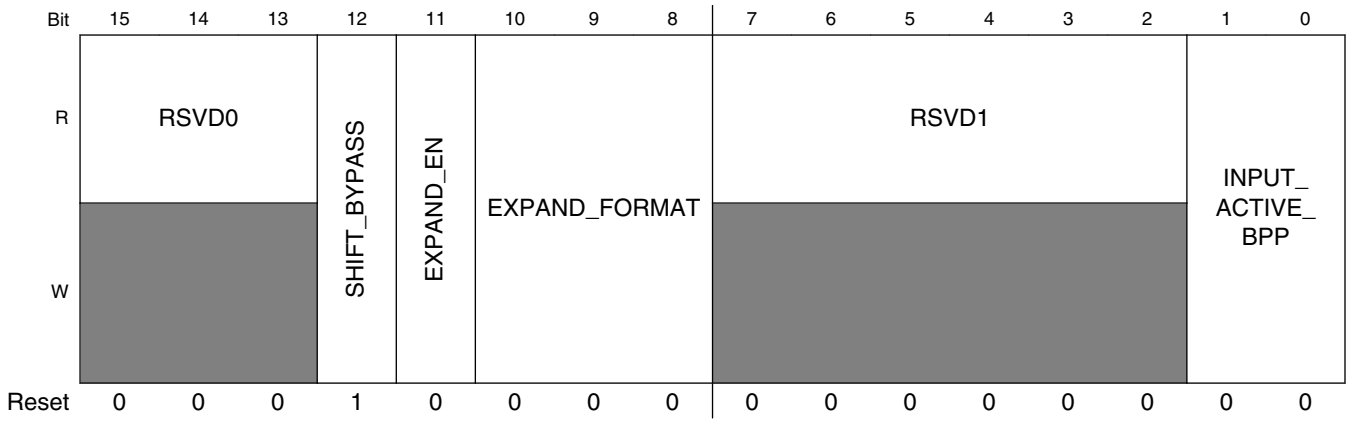
HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH0_TOG: 0x52C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 520h offset = 3070_0520h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



PXP_HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH0 field descriptions

Field	Description
31–13 RSVD0	Reserved, always set to zero.
12 SHIFT_BYPASS	0x0 **0** — channel0 data will do shift function 0x1 **1** — channel0 will bypass shift function
11 EXPAND_EN	0x0 **0** — channel0 format expanding disable 0x1 **1** — channel0 format expanding enable
10–8 EXPAND_ FORMAT	Select Pixel format 0x0 **0** — RGB 565 0x1 **1** — RGB 555 0x2 **2** — ARGB 1555 0x3 **3** — RGB 444 0x4 **4** — ARGB 4444 0x5 **5** — YUYV/YVYU 0x6 **6** — UYVY/VYUY 0x7 **7** — YUV422_2P
7–2 RSVD1	Reserved, always set to zero.
INPUT_ACTIVE_ BPP	0x0 **0** — 8 bits 0x1 **1** — 16 bits 0x2 **2** — 32 bits 0x3 **3** — 32 bits

13.6.12.74 PXP_HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH1: 0x530

HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH1_SET: 0x534

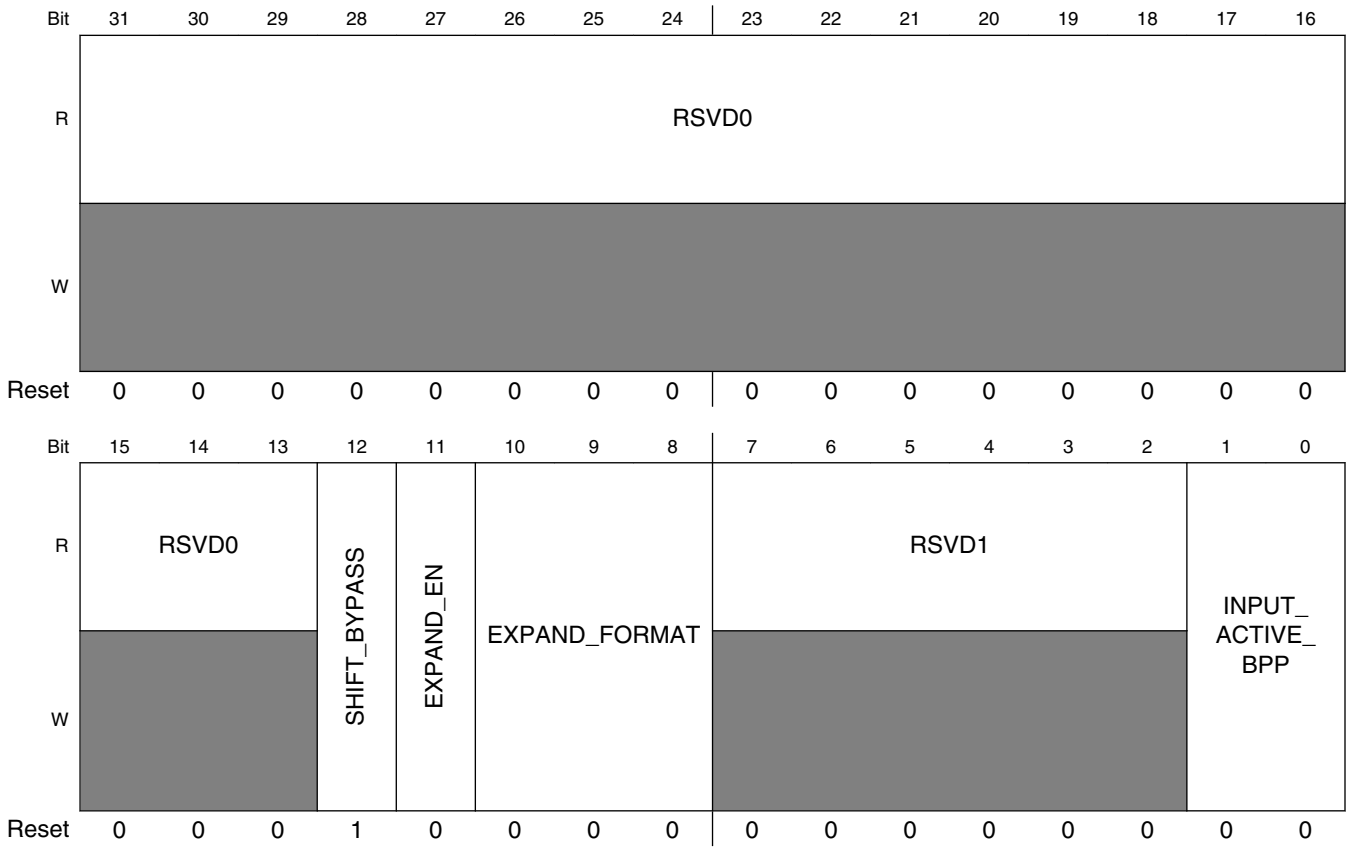
HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH1_CLR: 0x538

HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH1_TOG: 0x53C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 530h offset = 3070_0530h



PXP_HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH1 field descriptions

Field	Description
31–13 RSVD0	Reserved, always set to zero.
12 SHIFT_BYPASS	0x0 **0** — channel1 data will do shift function 0x1 **1** — channel1 will bypass shift function
11 EXPAND_EN	0x0 **0** — channel1 format expanding disable 0x1 **1** — channel1 format expanding enable
10–8 EXPAND_ FORMAT	Select Pixel format 0x0 **0** — RGB 565 0x1 **1** — RGB 555 0x2 **2** — ARGB 1555 0x3 **3** — RGB 444 0x4 **4** — ARGB 4444

Table continues on the next page...

PXP_HW_PXP_INPUT_FETCH_SHIFT_CTRL_CH1 field descriptions (continued)

Field	Description
	0x5 **5** — YUYV/YVYU 0x6 **6** — UYVY/VYUY 0x7 **7** — YUV422_2P
7–2 RSVD1	Reserved, always set to zero.
INPUT_ACTIVE_BPP	0x0 **0** — 8 bits 0x1 **1** — 16 bits 0x2 **2** — 32 bits 0x3 **3** — 32 bits

13.6.12.75 PXP_HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH0: 0x540

HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH0_SET: 0x544

HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH0_CLR: 0x548

HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH0_TOG: 0x54C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 540h offset = 3070_0540h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0			OFFSET3					RSVD1			OFFSET2					RSVD2			OFFSET1					RSVD3			OFFSET0				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH0 field descriptions

Field	Description
31–29 RSVD0	Reserved, always set to zero.
28–24 OFFSET3	Shift Offset for channel 0 componnent 3.
23–21 RSVD1	Reserved, always set to zero.
20–16 OFFSET2	Shift Offset for channel 0 componnent 2.

Table continues on the next page...

PXP_HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH0 field descriptions (continued)

Field	Description
15–13 RSVD2	Reserved, always set to zero.
12–8 OFFSET1	Shift Offset for channel 0 componnent 1.
7–5 RSVD3	Reserved, always set to zero.
OFFSET0	Shift Offset for channel 0 componnent 0.

13.6.12.76 PXP_HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH1: 0x550

HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH1_SET: 0x554

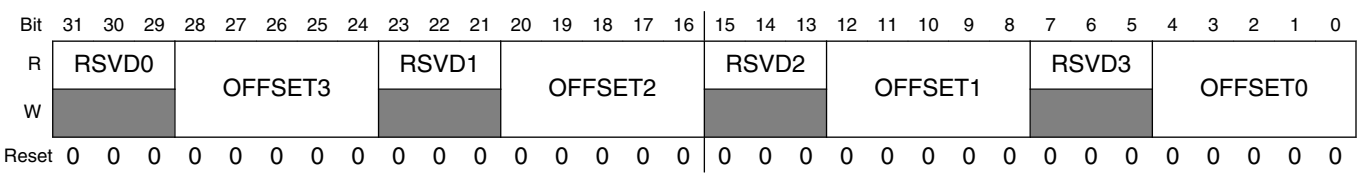
HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH1_CLR: 0x558

HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH1_TOG: 0x55C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 550h offset = 3070_0550h



PXP_HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH1 field descriptions

Field	Description
31–29 RSVD0	Reserved, always set to zero.
28–24 OFFSET3	Shift Offset for channel 1 componnent 3.
23–21 RSVD1	Reserved, always set to zero.
20–16 OFFSET2	Shift Offset for channel 1 componnent 2.
15–13 RSVD2	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_INPUT_FETCH_SHIFT_OFFSET_CH1 field descriptions (continued)

Field	Description
12–8 OFFSET1	Shift Offset for channel 1 componnent 1.
7–5 RSVD3	Reserved, always set to zero.
OFFSET0	Shift Offset for channel 1 componnent 0.

13.6.12.77 PXP_HW_PXP_INPUT_FETCH_SHIFT_WIDTH_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_INPUT_FETCH_SHIFT_WIDTH_CH0: 0x560

HW_PXP_INPUT_FETCH_SHIFT_WIDTH_CH0_SET: 0x564

HW_PXP_INPUT_FETCH_SHIFT_WIDTH_CH0_CLR: 0x568

HW_PXP_INPUT_FETCH_SHIFT_WIDTH_CH0_TOG: 0x56C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 560h offset = 3070_0560h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0																WIDTH3				WIDTH2				WIDTH1				WIDTH0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0

PXP_HW_PXP_INPUT_FETCH_SHIFT_WIDTH_CH0 field descriptions

Field	Description
31–16 RSVD0	Reserved, always set to zero.
15–12 WIDTH3	Shift Width for channel 0 componnent 3.
11–8 WIDTH2	Shift Width for channel 0 componnent 2.
7–4 WIDTH1	Shift Width for channel 0 componnent 1.
WIDTH0	Shift Width for channel 0 componnent 0.

13.6.12.78 PXP_HW_PXP_INPUT_FETCH_SHIFT_WIDTH_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_INPUT_FETCH_SHIFT_WIDTH_CH1: 0x570

HW_PXP_INPUT_FETCH_SHIFT_WIDTH_CH1_SET: 0x574

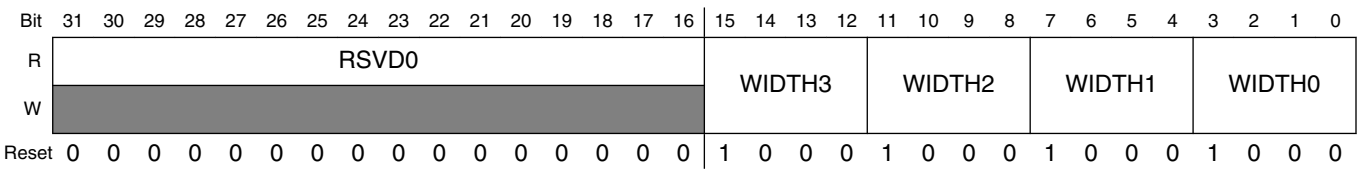
HW_PXP_INPUT_FETCH_SHIFT_WIDTH_CH1_CLR: 0x578

HW_PXP_INPUT_FETCH_SHIFT_WIDTH_CH1_TOG: 0x57C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 570h offset = 3070_0570h



PXP_HW_PXP_INPUT_FETCH_SHIFT_WIDTH_CH1 field descriptions

Field	Description
31–16 RSVD0	Reserved, always set to zero.
15–12 WIDTH3	Shift Width for channel 1 componnent 3.
11–8 WIDTH2	Shift Width for channel 1 componnent 2.
7–4 WIDTH1	Shift Width for channel 1 componnent 1.
WIDTH0	Shift Width for channel 1 componnent 0.

13.6.12.79 PXP_HW_PXP_INPUT_FETCH_ADDR_0_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 580h offset = 3070_0580h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_BASE_ADDR0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_FETCH_ADDR_0_CH0 field descriptions

Field	Description
INPUT_BASE_ADDR0	input base address0. For 2 channel, indicated the channel0 base address. For 1 channel and YUV422 2 plane, indicate the Y base address

13.6.12.80 PXP_HW_PXP_INPUT_FETCH_ADDR_1_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 590h offset = 3070_0590h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_BASE_ADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_FETCH_ADDR_1_CH0 field descriptions

Field	Description
INPUT_BASE_ADDR1	input base address1. For 2 channel, indicated the channel1 base address. For 1 channel and YUV422 2 plane, indicate the UV base address

13.6.12.81 PXP_HW_PXP_INPUT_FETCH_ADDR_0_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 5A0h offset = 3070_05A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_BASE_ADDR0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_FETCH_ADDR_0_CH1 field descriptions

Field	Description
INPUT_BASE_ADDR0	input base address0. For 2 channel, indicated the channel0 base address. For 1 channel and YUV422 2 plane, indicate the Y base address

13.6.12.82 PXP_HW_PXP_INPUT_FETCH_ADDR_1_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 5B0h offset = 3070_05B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_BASE_ADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_FETCH_ADDR_1_CH1 field descriptions

Field	Description
INPUT_BASE_ADDR1	input base address1. For 2 channel, indicated the channel1 base address. For 1 channel and YUV422 2 plane, indicate the UV base address

13.6.12.83 Store engine Control Channel 0 Register (PXP_HW_PXP_INPUT_STORE_CTRL_CH0)

This register defines the control bits for the pxp store_engine sub-block.

HW_PXP_INPUT_STORE_CTRL_CH0: 0x5C0

HW_PXP_INPUT_STORE_CTRL_CH0_SET: 0x5C4

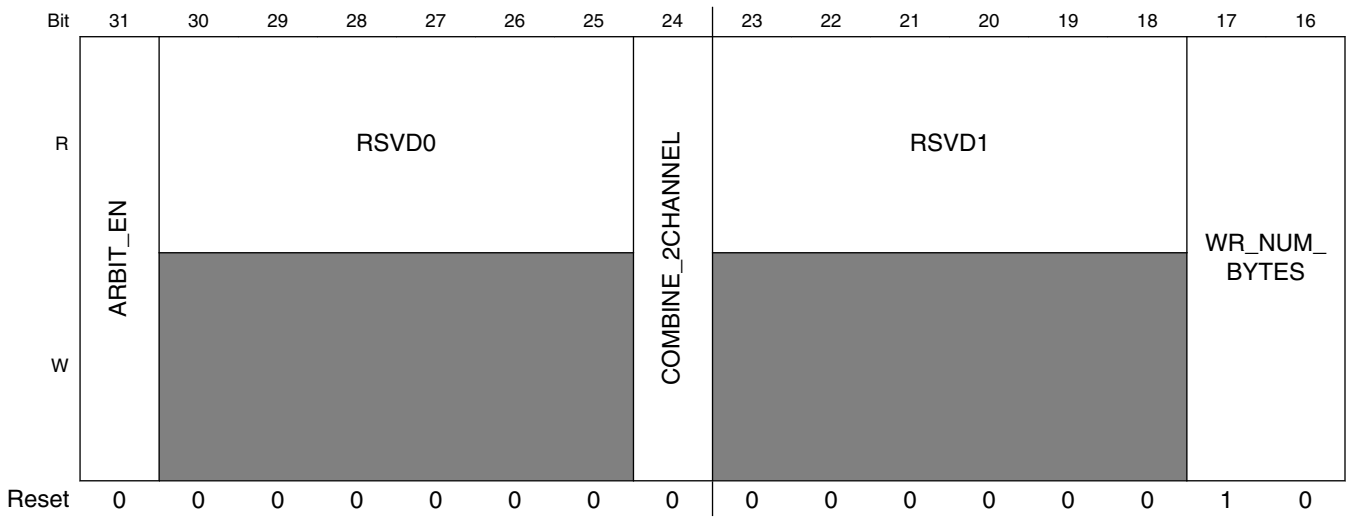
HW_PXP_INPUT_STORE_CTRL_CH0_CLR: 0x5C8

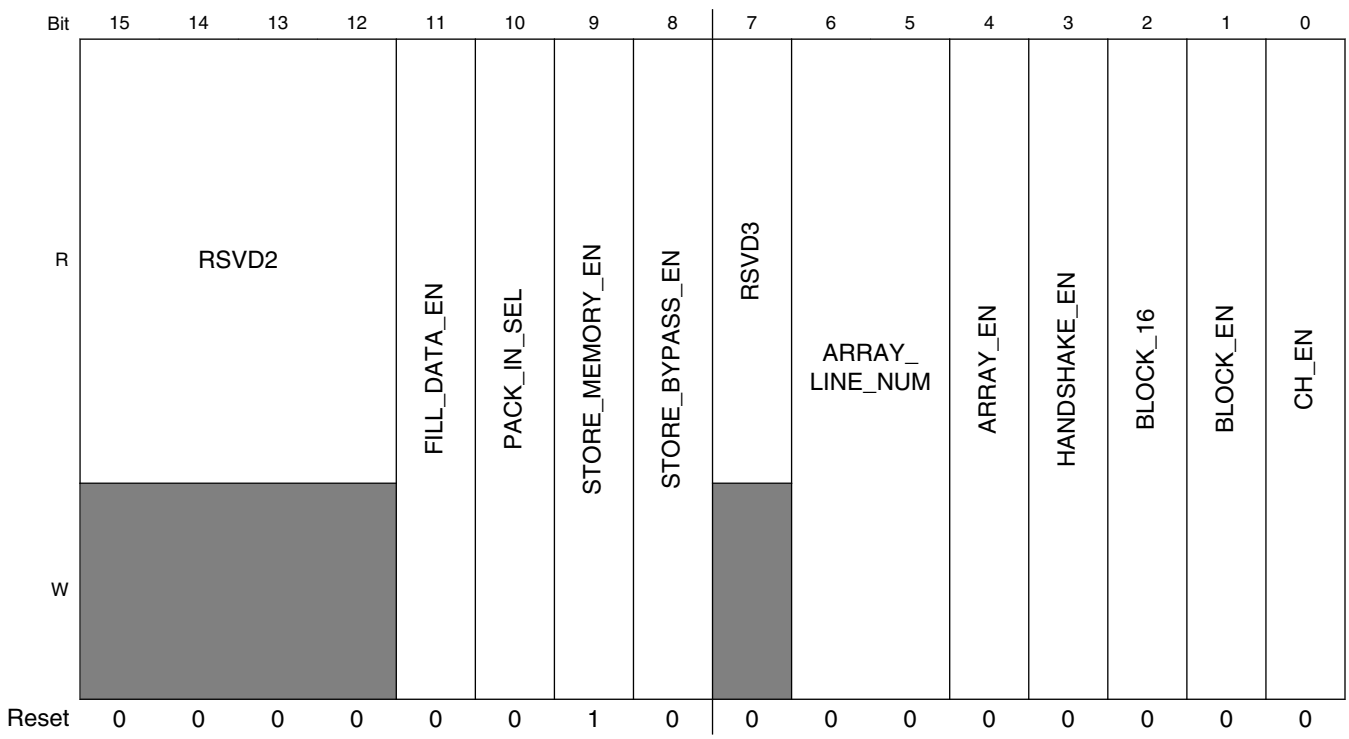
HW_PXP_INPUT_STORE_CTRL_CH0_TOG: 0x5CC

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 5C0h offset = 3070_05C0h





PXP_HW_PXP_INPUT_STORE_CTRL_CH0 field descriptions

Field	Description
31 ARBIT_EN	Arbitration Enable 0x0 **0** — Arbitration disable. If using 2 channels, will output 2 axi bus sets 0x1 **1** — Arbitration enable. If using 2 channel, will only output 1 axi bus sets
30–25 RSVD0	Reserved, always set to zero.
24 COMBINE_2CHANNEL	Combine 2 channel Enable

Table continues on the next page...

PXP_HW_PXP_INPUT_STORE_CTRL_CH0 field descriptions (continued)

Field	Description
	0x0 **0** — combine 2 channel disable 0x1 **1** — combine 2 channel enable
23–18 RSVD1	Reserved, always set to zero.
17–16 WR_NUM_ BYTES	Bytes in a write burst 0x0 **8_bytes** — 8 bytes 0x1 **16_bytes** — 16 bytes 0x2 **32_bytes** — 32 bytes 0x3 **64_bytes** — 64 bytes
15–12 RSVD2	Reserved, always set to zero.
11 FILL_DATA_EN	fill data enable 0x0 **0** — Fill data mode disable. 0x1 **1** — Fill data mode enable. When using fill_data mode, store_engine will store fixed data defined in fill_data register
10 PACK_IN_SEL	pack_in_sel 0x0 **0** — select 64 shift out data to pack 0x1 **1** — select low 32 bit shift out data to pack
9 STORE_ MEMORY_EN	store memory enable 0x0 **0** — store memory mode disable. 0x1 **1** — store memory mode enable. Data will store to memory
8 STORE_ BYPASS_EN	store bypass enable 0x0 **0** — store bypass mode disable. 0x1 **1** — store bypass mode enable. Data will bypass to store output.
7 RSVD3	Reserved, always set to zero.
6–5 ARRAY_LINE_ NUM	Selects Array Size 0x0 **0** — Using 1x1 Array 0x1 **1** — Using 3x3 Array 0x2 **2** — Using 5x5 Array 0x3 **3** — Using 5x5 Array
4 ARRAY_EN	0x0 **0** — Array Handshake Disabled 0x1 **1** — Array Handshake Enabled
3 HANDSHAKE_ EN	Enable bit for handshake with the store engine. 0x0 **0** — Handshake with the prefetch engine is disabled 0x1 **1** — Handshake with the prefetch engine is enabled
2 BLOCK_16	Determines the block sixe. 0x0 **8x8** — Block size is 8x8 0x1 **16x16** — Block size is 16x16

Table continues on the next page...

PXP_HW_PXP_INPUT_STORE_CTRL_CH0 field descriptions (continued)

Field	Description
1 BLOCK_EN	Choses the store mode. 0x0 **0** — Store in scan mode 0x1 **1** — Store in block mode
0 CH_EN	Channel enable. 0x0 **0** — Store function is disable 0x1 **1** — Store function is enable

13.6.12.84 Store engine Control Channel 1 Register (PXP_HW_PXP_INPUT_STORE_CTRL_CH1)

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_INPUT_STORE_CTRL_CH1: 0x5D0

HW_PXP_INPUT_STORE_CTRL_CH1_SET: 0x5D4

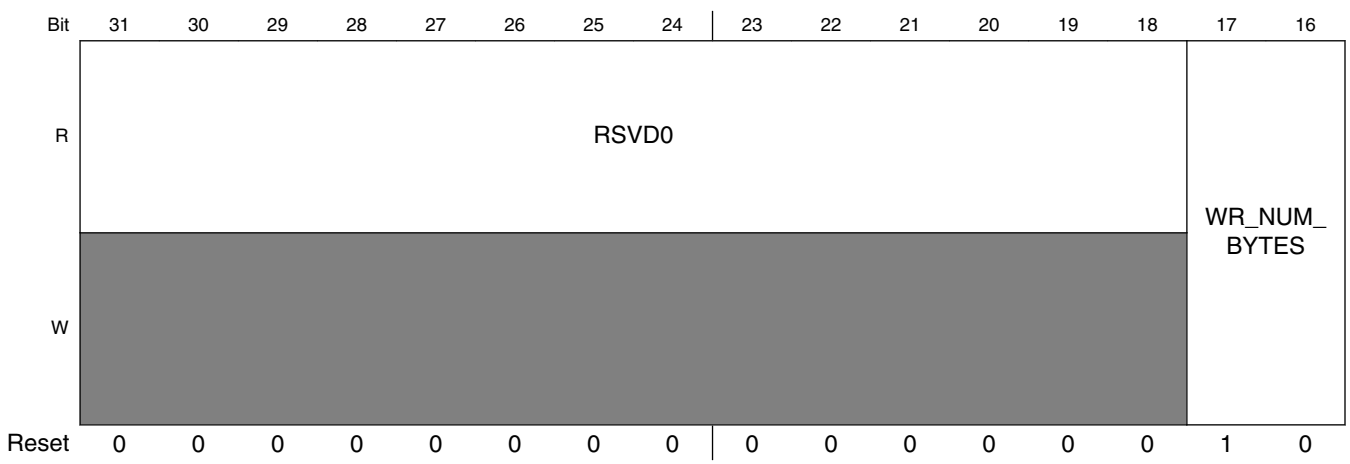
HW_PXP_INPUT_STORE_CTRL_CH1_CLR: 0x5D8

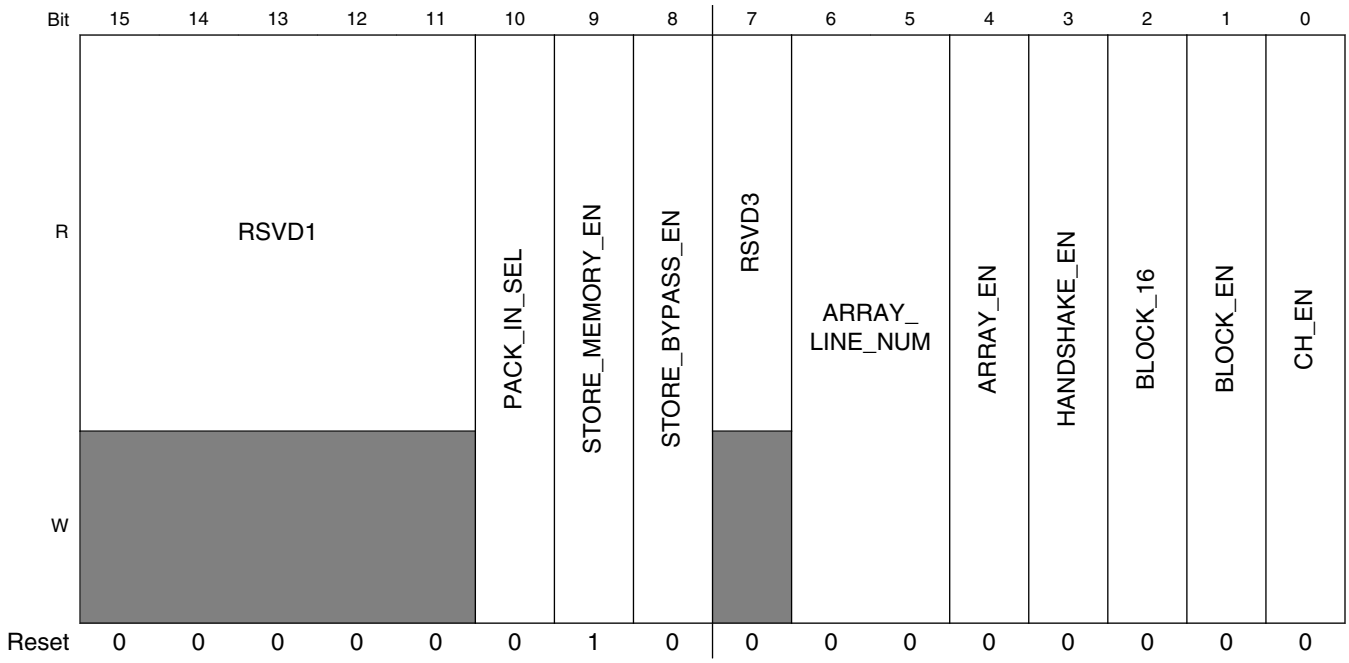
HW_PXP_INPUT_STORE_CTRL_CH1_TOG: 0x5DC

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 5D0h offset = 3070_05D0h





PXP_HW_PXP_INPUT_STORE_CTRL_CH1 field descriptions

Field	Description
31–18 RSVD0	Reserved, always set to zero.
17–16 WR_NUM_ BYTES	Bytes in a write burst 0x0 **8_bytes** — 8 bytes 0x1 **16_bytes** — 16 bytes 0x2 **32_bytes** — 32 bytes 0x3 **64_bytes** — 64 bytes
15–11 RSVD1	Reserved, always set to zero.
10 PACK_IN_SEL	pack_in_sel 0x0 **0** — select 64 shift out data to pack 0x1 **1** — select channel 0 high 32 bit shift out data to pack
9 STORE_ MEMORY_EN	store memory enable 0x0 **0** — store memory mode disable. 0x1 **1** — store memory mode enable. Data will store to memory.
8 STORE_ BYPASS_EN	enable bit for store bypass 0x0 **0** — store bypass mode disable. 0x1 **1** — store bypass mode enable. Data will bypass to store output.
7 RSVD3	Reserved, always set to zero.
6–5 ARRAY_LINE_ NUM	Selects Array Size 0x0 **0** — Using 1x1 Array

Table continues on the next page...

PXP_HW_PXP_INPUT_STORE_CTRL_CH1 field descriptions (continued)

Field	Description
	0x1 **1** — Using 3x3 Array 0x2 **2** — Using 5x5 Array 0x3 **3** — Using 5x5 Array
4 ARRAY_EN	0x0 **0** — Array Handshake Disabled 0x1 **1** — Array Handshake Enabled
3 HANDSHAKE_ EN	Enable bit for handshake with the fetch engine. 0x0 **0** — Handshake with the fetch engine is disabled 0x1 **1** — Handshake with the fetch engine is enabled
2 BLOCK_16	Determines the block sixe. 0x0 **8x8** — Block size is 8x8 0x1 **16x16** — Block size is 16x16
1 BLOCK_EN	Choses the store mode. 0x0 **0** — Store in scan mode 0x1 **1** — Store in block mode
0 CH_EN	Channel enable. 0x0 **0** — Store function is disable 0x1 **1** — Store function is enable

13.6.12.85 Store engine status Channel 0 Register (PXP_HW_PXP_INPUT_STORE_STATUS_CH0)

This register defines the status bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 5E0h offset = 3070_05E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STORE_BLOCK_Y																STORE_BLOCK_X															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_STATUS_CH0 field descriptions

Field	Description
31–16 STORE_ BLOCK_Y	When in scan mode, this field indicates the current Y coordinate of the frame. In block mode, it indicates the Y coordinate of the block currently being rendered.

Table continues on the next page...

PXP_HW_PXP_INPUT_STORE_STATUS_CH0 field descriptions (continued)

Field	Description
STORE_ BLOCK_X	When in scan mode, this field is always 0. In block mode, it indicates the X coordinate of the block currently being rendered.

13.6.12.86 Store engine status Channel 1 Register (PXP_HW_PXP_INPUT_STORE_STATUS_CH1)

This register defines the status bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 5F0h offset = 3070_05F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STORE_BLOCK_Y																STORE_BLOCK_X															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_STATUS_CH1 field descriptions

Field	Description
31–16 STORE_ BLOCK_Y	When in scan mode, this field indicates the current Y coordinate of the frame. In block mode, it indicates the Y coordinate of the block currently being rendered.
STORE_ BLOCK_X	When in scan mode, this field is always 0. In block mode, it indicates the X coordinate of the block currently being rendered.

13.6.12.87 PXP_HW_PXP_INPUT_STORE_SIZE_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 600h offset = 3070_0600h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OUT_HEIGHT																OUT_WIDTH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_SIZE_CH0 field descriptions

Field	Description
31–16 OUT_HEIGHT	actual output height -1
OUT_WIDTH	actual output width -1

13.6.12.88 PXP_HW_PXP_INPUT_STORE_SIZE_CH1

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 610h offset = 3070_0610h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OUT_HEIGHT																OUT_WIDTH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_SIZE_CH1 field descriptions

Field	Description
31–16 OUT_HEIGHT	actual output height -1
OUT_WIDTH	actual output width -1

13.6.12.89 PXP_HW_PXP_INPUT_STORE_PITCH

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 620h offset = 3070_0620h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH1_OUT_PITCH																CH0_OUT_PITCH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_PITCH field descriptions

Field	Description
31–16 CH1_OUT_ PITCH	This field indicates the channel 1 input pitch
CH0_OUT_ PITCH	This field indicates the channel 0 input pitch

13.6.12.90 PXP_HW_PXP_INPUT_STORE_SHIFT_CTRL_CH0

This register defines the control bits for the pxp store_engine sub-block.

HW_PXP_INPUT_STORE_SHIFT_CTRL_CH0: 0x630

HW_PXP_INPUT_STORE_SHIFT_CTRL_CH0_SET: 0x634

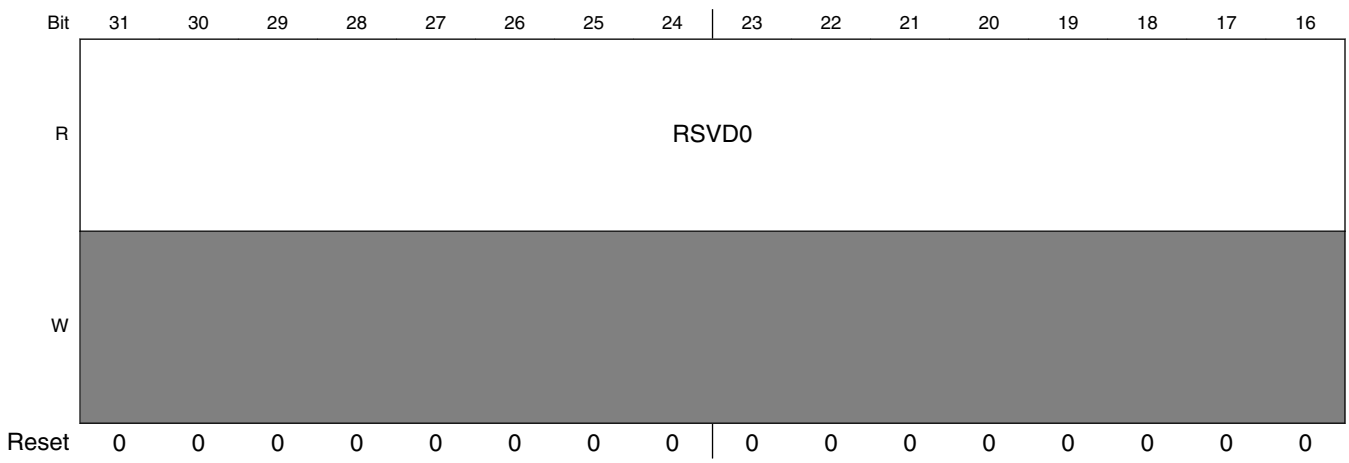
HW_PXP_INPUT_STORE_SHIFT_CTRL_CH0_CLR: 0x638

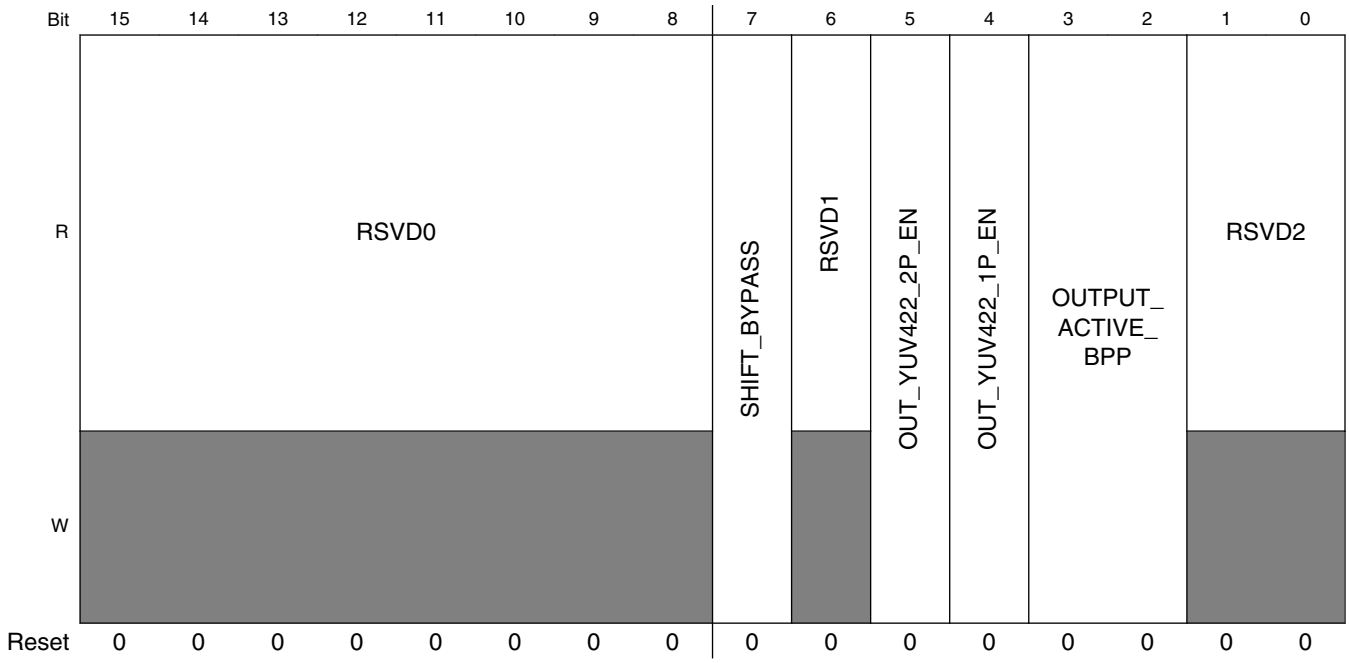
HW_PXP_INPUT_STORE_SHIFT_CTRL_CH0_TOG: 0x63C

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 630h offset = 3070_0630h





PXP_HW_PXP_INPUT_STORE_SHIFT_CTRL_CH0 field descriptions

Field	Description
31–8 RSVD0	Reserved, always set to zero.
7 SHIFT_BYPASS	CH0 shift bypass 0x0 **0** — data will do shift processing. 0x1 **1** — data will bypass shift module.
6 RSVD1	Reserved, always set to zero.
5 OUT_YUV422_ 2P_EN	Enable for YUV422 2 plane 0x0 **0** — YUYV422 2 plane disabled. 0x1 **1** — YUYV422 2 plane enabled.
4 OUT_YUV422_ 1P_EN	Enable for YUV422 1 plane 0x0 **0** — YUYV422 2 plane disabled. 0x1 **1** — YUYV422 2 plane enabled.
3–2 OUTPUT_ ACTIVE_BPP	0x0 **0** — 8 bits 0x1 **1** — 16 bits 0x2 **2** — 32 bits 0x3 **3** — 32 bits
RSVD2	Reserved, always set to zero.

13.6.12.91 PXP_HW_PXP_INPUT_STORE_SHIFT_CTRL_CH1

This register defines the control bits for the pxp store_engine sub-block.

HW_PXP_INPUT_STORE_SHIFT_CTRL_CH1: 0x640

HW_PXP_INPUT_STORE_SHIFT_CTRL_CH1_SET: 0x644

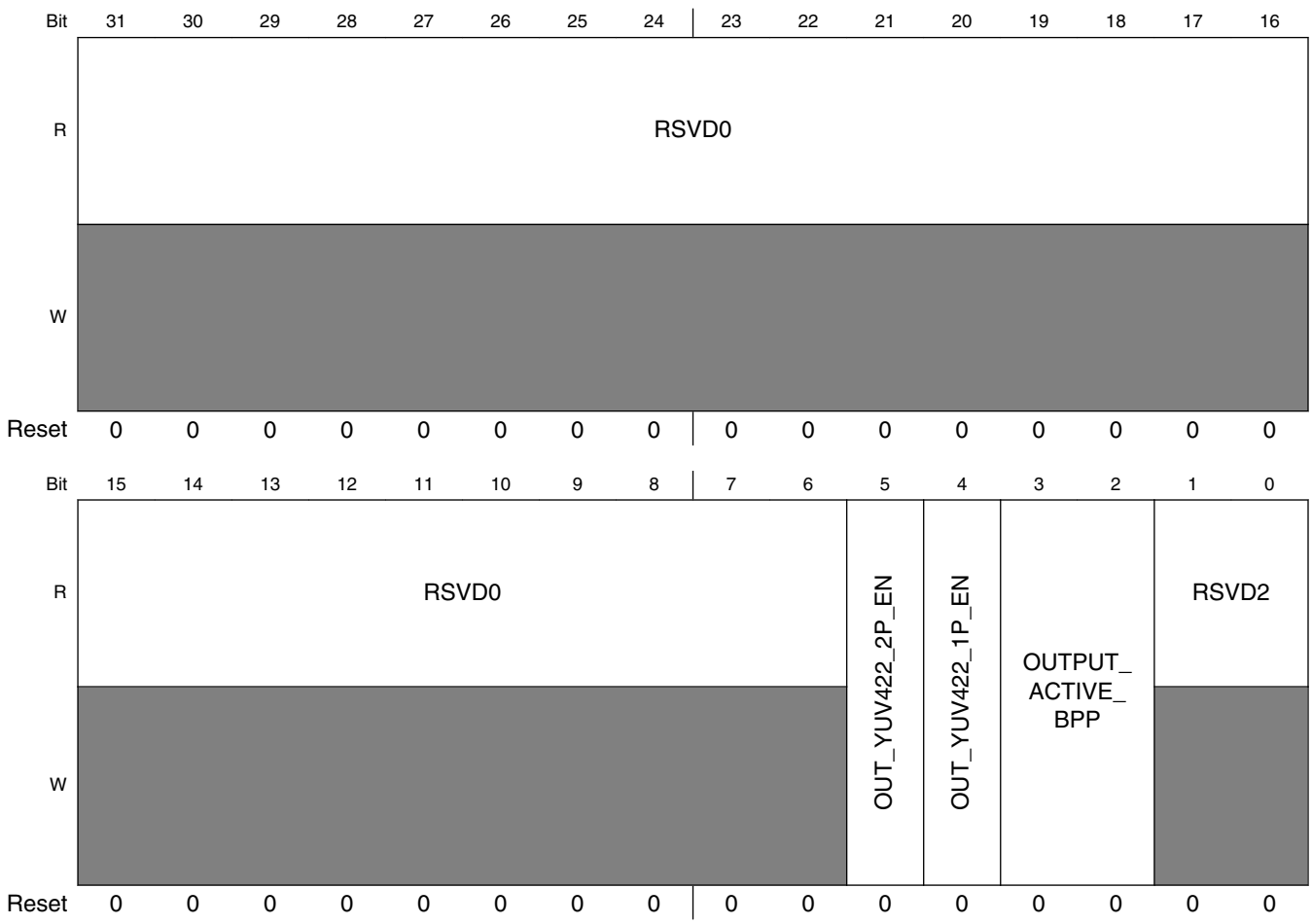
HW_PXP_INPUT_STORE_SHIFT_CTRL_CH1_CLR: 0x648

HW_PXP_INPUT_STORE_SHIFT_CTRL_CH1_TOG: 0x64C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 640h offset = 3070_0640h



PXP_HW_PXP_INPUT_STORE_SHIFT_CTRL_CH1 field descriptions

Field	Description
31–6 RSVD0	Reserved, always set to zero.
5 OUT_YUV422_2P_EN	Enable for YUV422 2 plane

Table continues on the next page...

PXP_HW_PXP_INPUT_STORE_SHIFT_CTRL_CH1 field descriptions (continued)

Field	Description
	0x0 **0** — YUYV422 2 plane disabled. 0x1 **1** — YUYV422 2 plane enabled.
4 OUT_YUV422_ 1P_EN	Enable for YUV422 1 plane 0x0 **0** — YUYV422 2 plane disabled. 0x1 **1** — YUYV422 2 plane enabled.
3–2 OUTPUT_ ACTIVE_BPP	0x0 **0** — 8 bits 0x1 **1** — 16 bits 0x2 **2** — 32 bits 0x3 **3** — 32 bits
RSVD2	Reserved, always set to zero.

13.6.12.92 PXP_HW_PXP_INPUT_STORE_ADDR_0_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 690h offset = 3070_0690h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OUT_BASE_ADDR0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_ADDR_0_CH0 field descriptions

Field	Description
OUT_BASE_ ADDR0	input base address0. For 2 channel, indicated the channel0 base address. For 1 channel and YUV422 2 plane, indicate the Y base address

13.6.12.93 PXP_HW_PXP_INPUT_STORE_ADDR_1_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 6A0h offset = 3070_06A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OUT_BASE_ADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_ADDR_1_CH0 field descriptions

Field	Description
OUT_BASE_ADDR1	input base address1. For 2 channel, indicated the channel 1 base address. For 1 channel and YUV422 2 plane, indicate the UV base address

13.6.12.94 PXP_HW_PXP_INPUT_STORE_FILL_DATA_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 6B0h offset = 3070_06B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FILL_DATA_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_FILL_DATA_CH0 field descriptions

Field	Description
FILL_DATA_CH0	when using fill_data mode,store engine channel0 will store the fill_data value defined here.

13.6.12.95 PXP_HW_PXP_INPUT_STORE_ADDR_0_CH1

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 6C0h offset = 3070_06C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OUT_BASE_ADDR0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_ADDR_0_CH1 field descriptions

Field	Description
OUT_BASE_ ADDR0	input base address0. For 2 channel, indicated the channel0 base address. For 1 channel and YUV422 2 plane, indicate the Y base address

13.6.12.96 PXP_HW_PXP_INPUT_STORE_ADDR_1_CH1

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 6D0h offset = 3070_06D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OUT_BASE_ADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_ADDR_1_CH1 field descriptions

Field	Description
OUT_BASE_ ADDR1	input base address1. For 2 channel, indicated the channel 1 base address. For 1 channel and YUV422 2 plane, indicate the UV base address

13.6.12.97 PXP_HW_PXP_INPUT_STORE_D_MASK0_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 6E0h offset = 3070_06E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK0_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK0_H_CH0 field descriptions

Field	Description
D_MASK0_H_ CH0	data mask0 high byte

13.6.12.98 PXP_HW_PXP_INPUT_STORE_D_MASK0_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 6F0h offset = 3070_06F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK0_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK0_L_CH0 field descriptions

Field	Description
D_MASK0_L_CH0	data mask0 low byte

13.6.12.99 PXP_HW_PXP_INPUT_STORE_D_MASK1_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 700h offset = 3070_0700h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK1_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK1_H_CH0 field descriptions

Field	Description
D_MASK1_H_CH0	data mask1 high byte

13.6.12.100 PXP_HW_PXP_INPUT_STORE_D_MASK1_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 710h offset = 3070_0710h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK1_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK1_L_CH0 field descriptions

Field	Description
D_MASK1_L_CH0	data mask1 low byte

13.6.12.101 PXP_HW_PXP_INPUT_STORE_D_MASK2_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 720h offset = 3070_0720h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK2_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK2_H_CH0 field descriptions

Field	Description
D_MASK2_H_CH0	data mask2 high byte

13.6.12.102 PXP_HW_PXP_INPUT_STORE_D_MASK2_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 730h offset = 3070_0730h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK2_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK2_L_CH0 field descriptions

Field	Description
D_MASK2_L_CH0	data mask2 low byte

13.6.12.103 PXP_HW_PXP_INPUT_STORE_D_MASK3_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 740h offset = 3070_0740h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK3_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK3_H_CH0 field descriptions

Field	Description
D_MASK3_H_CH0	data mask3 high byte

13.6.12.104 PXP_HW_PXP_INPUT_STORE_D_MASK3_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 750h offset = 3070_0750h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK3_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK3_L_CH0 field descriptions

Field	Description
D_MASK3_L_CH0	data mask3 low byte

13.6.12.105 PXP_HW_PXP_INPUT_STORE_D_MASK4_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 760h offset = 3070_0760h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK4_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK4_H_CH0 field descriptions

Field	Description
D_MASK4_H_CH0	data mask4 high byte

13.6.12.106 PXP_HW_PXP_INPUT_STORE_D_MASK4_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 770h offset = 3070_0770h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK4_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK4_L_CH0 field descriptions

Field	Description
D_MASK4_L_CH0	data mask4 low byte

13.6.12.107 PXP_HW_PXP_INPUT_STORE_D_MASK5_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 780h offset = 3070_0780h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK5_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK5_H_CH0 field descriptions

Field	Description
D_MASK5_H_CH0	data mask5 high byte

13.6.12.108 PXP_HW_PXP_INPUT_STORE_D_MASK5_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 790h offset = 3070_0790h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK5_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK5_L_CH0 field descriptions

Field	Description
D_MASK5_L_CH0	data mask5 low byte

13.6.12.109 PXP_HW_PXP_INPUT_STORE_D_MASK6_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 7A0h offset = 3070_07A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK6_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK6_H_CH0 field descriptions

Field	Description
D_MASK6_H_CH0	data mask6 high byte

13.6.12.110 PXP_HW_PXP_INPUT_STORE_D_MASK6_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 7B0h offset = 3070_07B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK6_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK6_L_CH0 field descriptions

Field	Description
D_MASK6_L_CH0	data mask6 low byte

13.6.12.111 PXP_HW_PXP_INPUT_STORE_D_MASK7_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 7C0h offset = 3070_07C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK7_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK7_H_CH0 field descriptions

Field	Description
D_MASK7_H_CH0	data mask7 high byte

13.6.12.112 PXP_HW_PXP_INPUT_STORE_D_MASK7_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 7E0h offset = 3070_07E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK7_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_D_MASK7_L_CH0 field descriptions

Field	Description
D_MASK7_L_CH0	data mask7 low byte

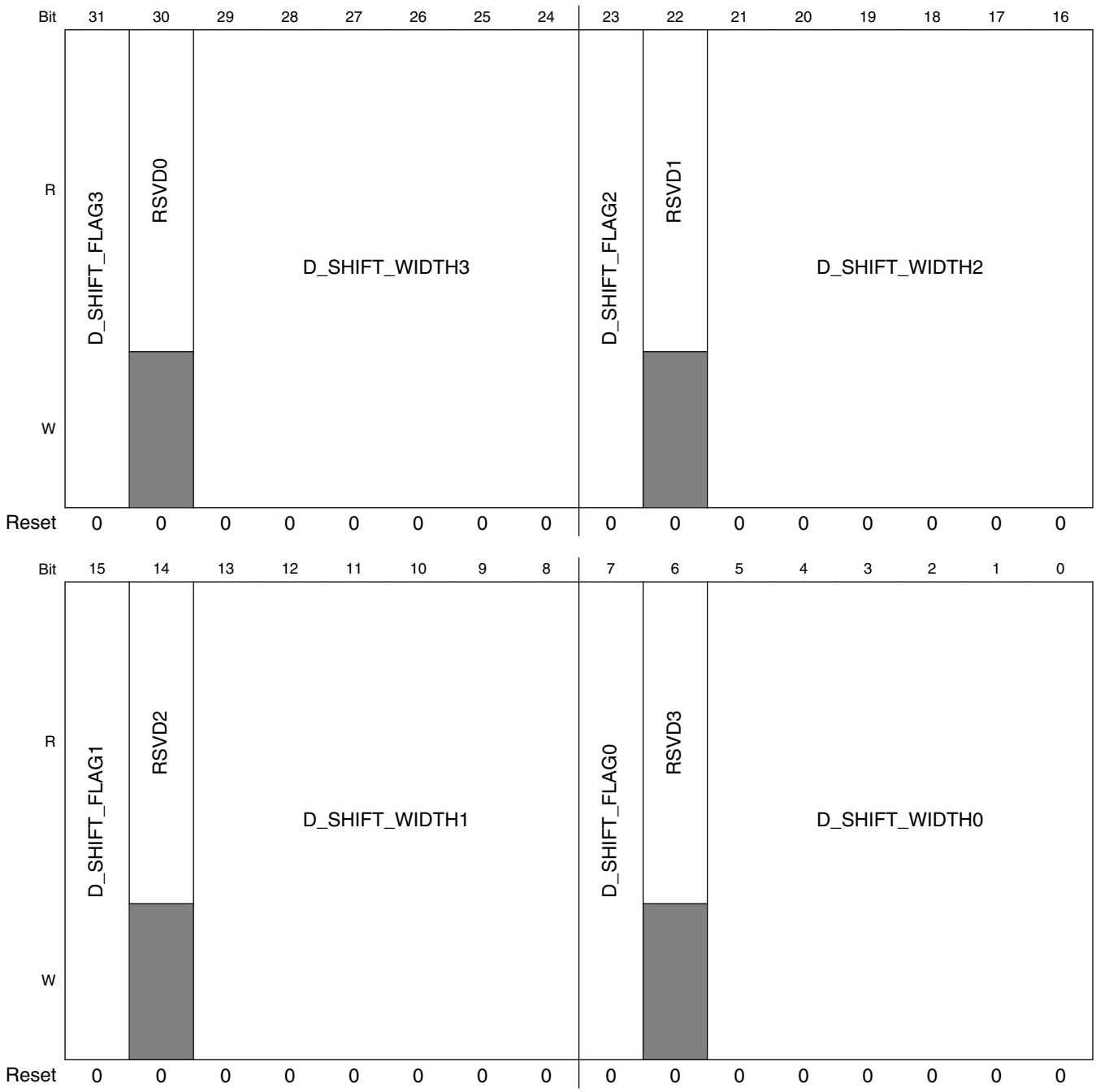
13.6.12.113 PXP_HW_PXP_INPUT_STORE_D_SHIFT_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 7F0h offset = 3070_07F0h



PXP_HW_PXP_INPUT_STORE_D_SHIFT_L_CH0 field descriptions

Field	Description
31 D_SHIFT_FLAG3	data shift flag 3
30 RSVD0	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_INPUT_STORE_D_SHIFT_L_CH0 field descriptions (continued)

Field	Description
29–24 D_SHIFT_ WIDTH3	data shift width 3
23 D_SHIFT_FLAG2	data shift flag 2
22 RSVD1	Reserved, always set to zero.
21–16 D_SHIFT_ WIDTH2	data shift width 2
15 D_SHIFT_FLAG1	data shift flag 1
14 RSVD2	Reserved, always set to zero.
13–8 D_SHIFT_ WIDTH1	data shift width 1
7 D_SHIFT_FLAG0	data shift flag 0
6 RSVD3	Reserved, always set to zero.
D_SHIFT_ WIDTH0	data shift width 0

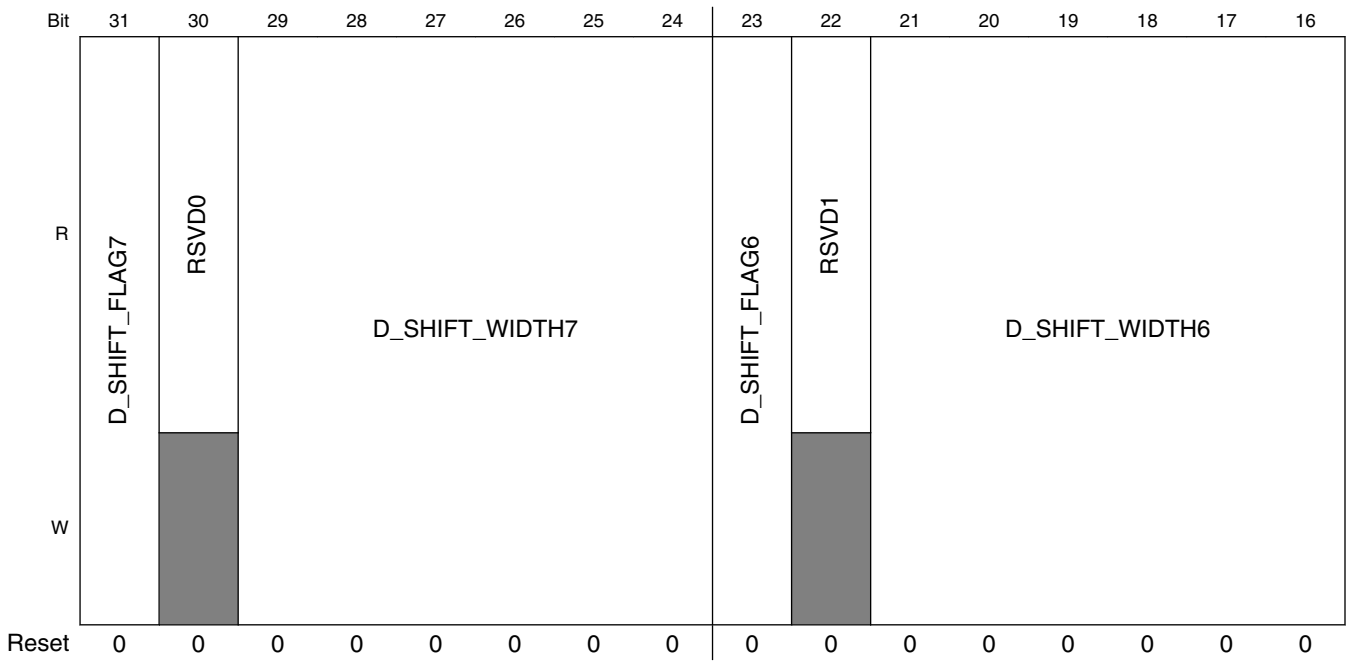
13.6.12.114 PXP_HW_PXP_INPUT_STORE_D_SHIFT_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 800h offset = 3070_0800h



PXP_HW_PXP_INPUT_STORE_D_SHIFT_H_CH0 field descriptions

Field	Description
31 D_SHIFT_FLAG7	data shift flag 7
30 RSVD0	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_INPUT_STORE_D_SHIFT_H_CH0 field descriptions (continued)

Field	Description
29–24 D_SHIFT_ WIDTH7	data shift width 3
23 D_SHIFT_FLAG6	data shift flag 6
22 RSVD1	Reserved, always set to zero.
21–16 D_SHIFT_ WIDTH6	data shift width 6
15 D_SHIFT_FLAG5	data shift flag 5
14 RSVD2	Reserved, always set to zero.
13–8 D_SHIFT_ WIDTH5	data shift width 5
7 D_SHIFT_FLAG4	data shift flag 4
6 RSVD3	Reserved, always set to zero.
D_SHIFT_ WIDTH4	data shift width 4

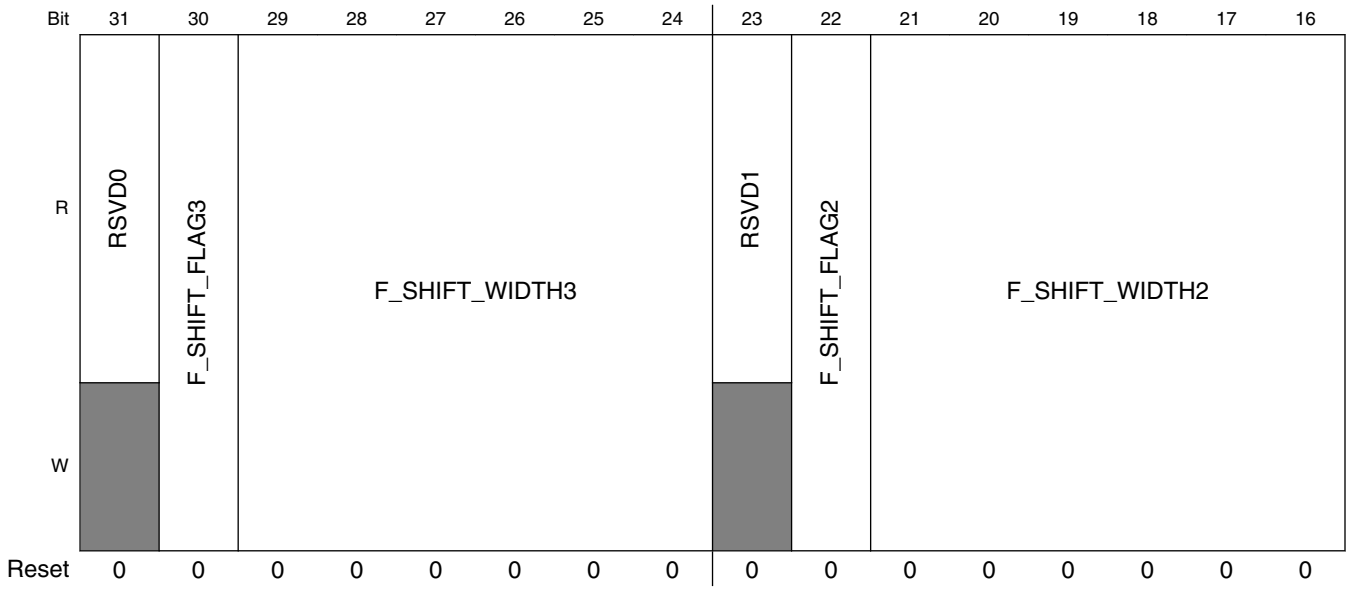
13.6.12.115 PXP_HW_PXP_INPUT_STORE_F_SHIFT_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 810h offset = 3070_0810h



PXP_HW_PXP_INPUT_STORE_F_SHIFT_L_CH0 field descriptions

Field	Description
31 RSVD0	Reserved, always set to zero.
30 F_SHIFT_FLAG3	flag shift flag3
29–24 F_SHIFT_ WIDTH3	flag shift width 3
23 RSVD1	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_INPUT_STORE_F_SHIFT_L_CH0 field descriptions (continued)

Field	Description
22 F_SHIFT_FLAG2	flag shift flag2
21–16 F_SHIFT_ WIDTH2	flag shift width 2
15 RSVD2	Reserved, always set to zero.
14 F_SHIFT_FLAG1	flag shift flag1
13–8 F_SHIFT_ WIDTH1	flag shift width 1
7 RSVD3	Reserved, always set to zero.
6 F_SHIFT_FLAG0	flag shift flag0
F_SHIFT_ WIDTH0	flag shift width 0

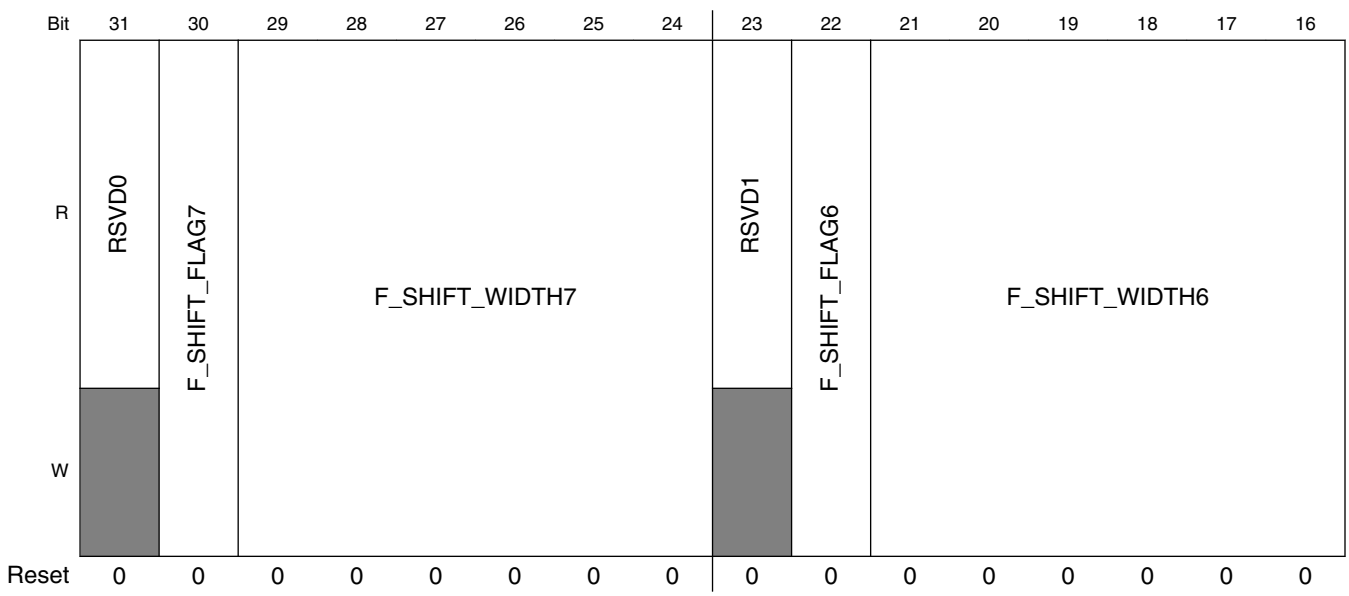
13.6.12.116 PXP_HW_PXP_INPUT_STORE_F_SHIFT_H_CH0

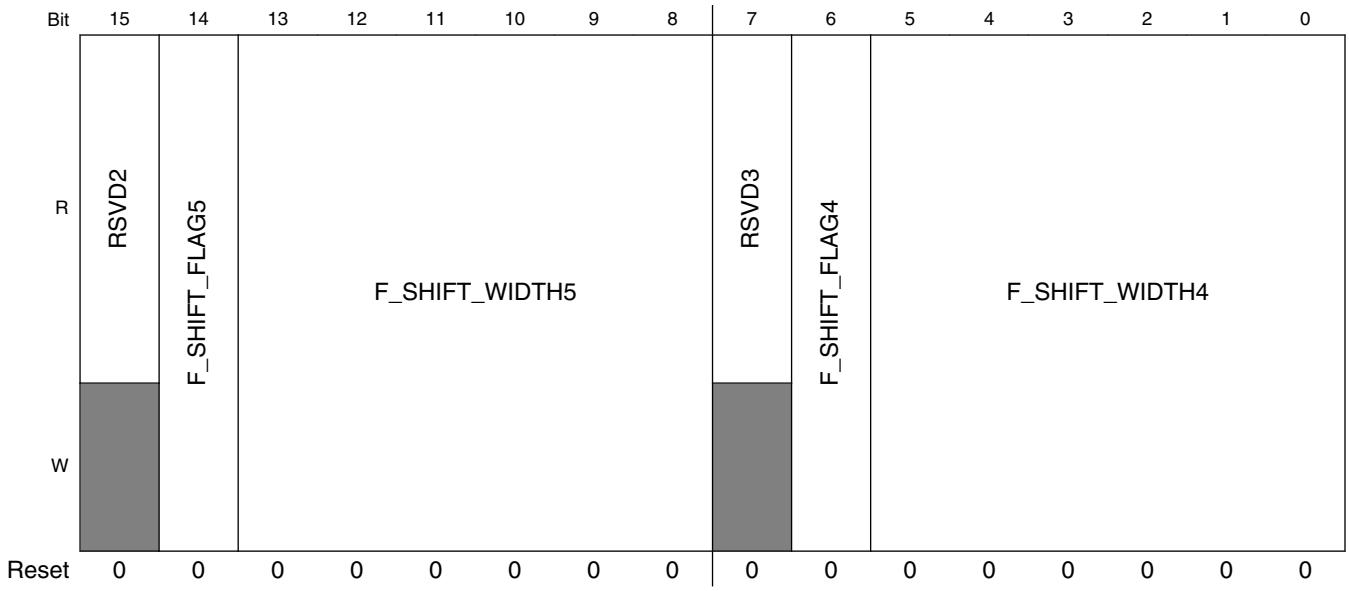
This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 820h offset = 3070_0820h





PXP_HW_PXP_INPUT_STORE_F_SHIFT_H_CH0 field descriptions

Field	Description
31 RSVD0	Reserved, always set to zero.
30 F_SHIFT_FLAG7	flag shift flag7
29–24 F_SHIFT_ WIDTH7	flag shift width 7
23 RSVD1	Reserved, always set to zero.
22 F_SHIFT_FLAG6	flag shift flag6
21–16 F_SHIFT_ WIDTH6	flag shift width 5
15 RSVD2	Reserved, always set to zero.
14 F_SHIFT_FLAG5	flag shift flag5
13–8 F_SHIFT_ WIDTH5	flag shift width 5
7 RSVD3	Reserved, always set to zero.
6 F_SHIFT_FLAG4	flag shift flag4
F_SHIFT_ WIDTH4	flag shift width 4

13.6.12.117 PXP_HW_PXP_INPUT_STORE_F_MASK_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 830h offset = 3070_0830h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	F_MASK3								F_MASK2								F_MASK1								F_MASK0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_F_MASK_L_CH0 field descriptions

Field	Description
31–24 F_MASK3	flag mask3
23–16 F_MASK2	flag mask2
15–8 F_MASK1	flag mask1
F_MASK0	flag mask0

13.6.12.118 PXP_HW_PXP_INPUT_STORE_F_MASK_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 840h offset = 3070_0840h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	F_MASK7								F_MASK6								F_MASK5								F_MASK4							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_INPUT_STORE_F_MASK_H_CH0 field descriptions

Field	Description
31–24 F_MASK7	flag mask7

Table continues on the next page...

PXP_HW_PXP_INPUT_STORE_F_MASK_H_CH0 field descriptions (continued)

Field	Description
23–16 F_MASK6	flag mask6
15–8 F_MASK5	flag mask5
F_MASK4	flag mask4

13.6.12.119 Pre-fetch engine Control Channel 0 Register (PXP_HW_PXP_DITHER_FETCH_CTRL_CH0)

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_DITHER_FETCH_CTRL_CH0: 0x850

HW_PXP_DITHER_FETCH_CTRL_CH0_SET: 0x854

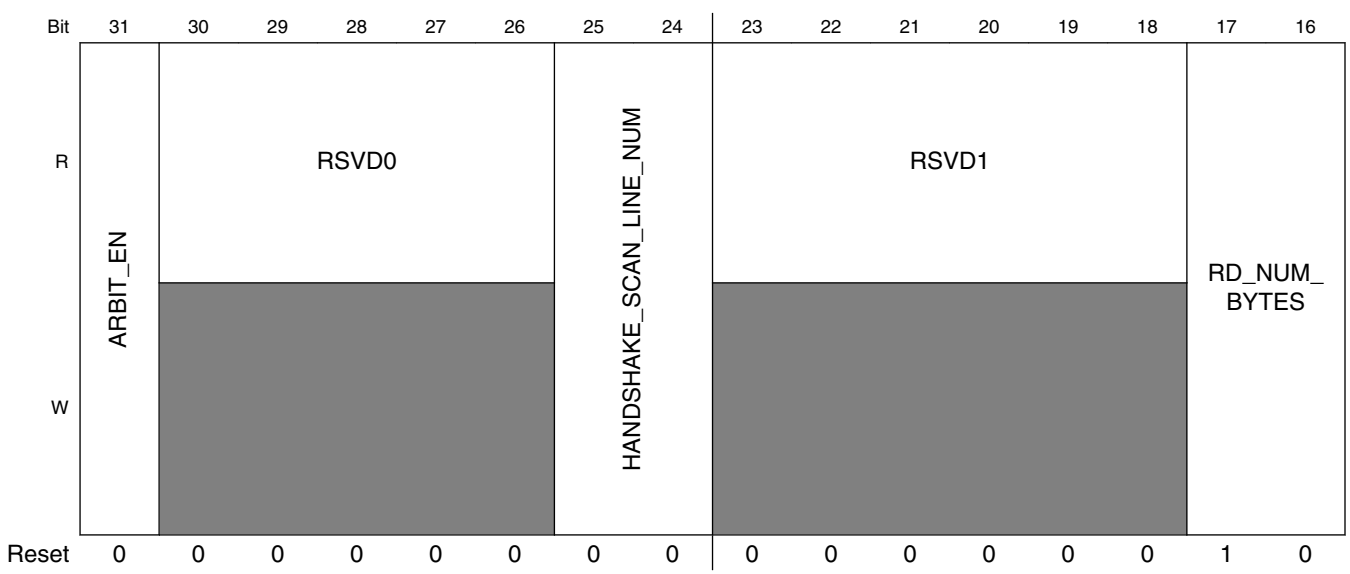
HW_PXP_DITHER_FETCH_CTRL_CH0_CLR: 0x858

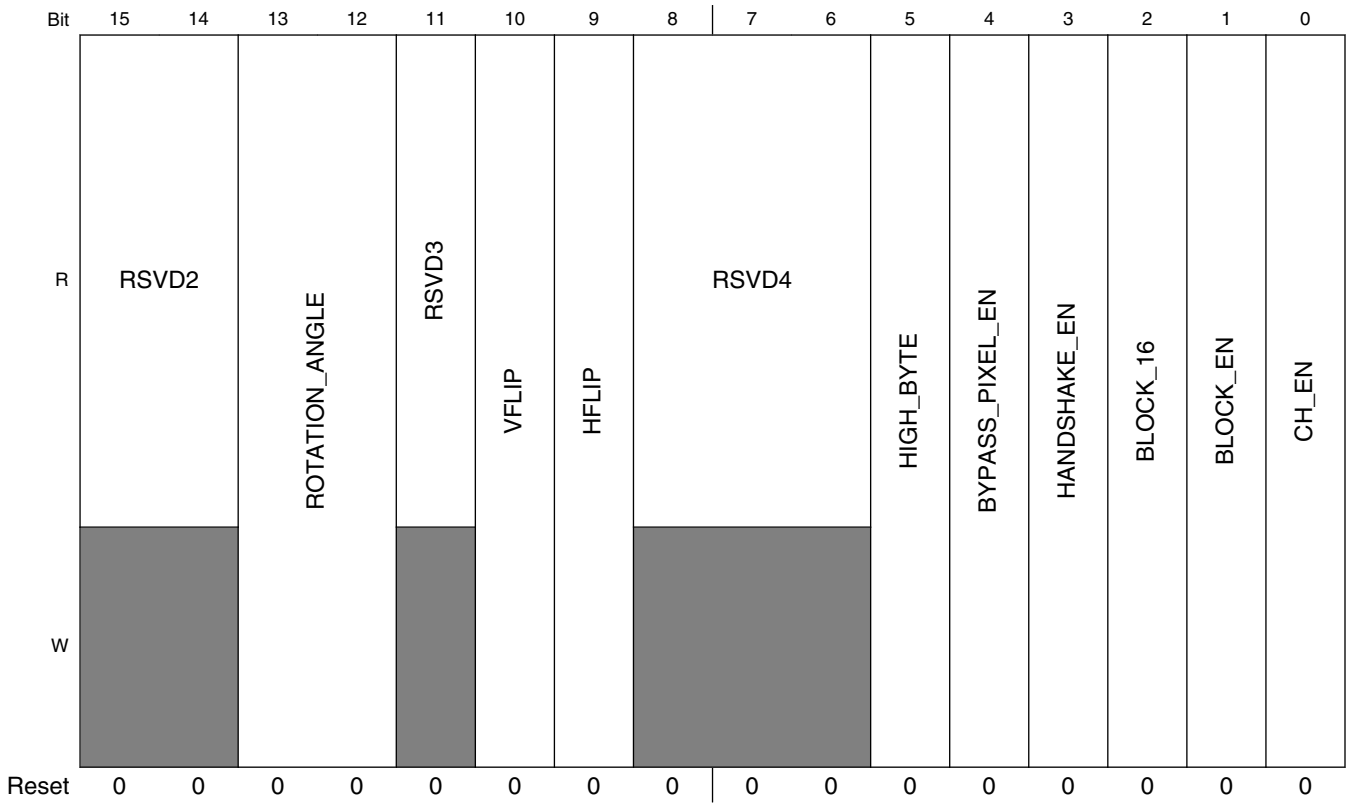
HW_PXP_DITHER_FETCH_CTRL_CH0_TOG: 0x85C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 850h offset = 3070_0850h





PXP_HW_PXP_DITHER_FETCH_CTRL_CH0 field descriptions

Field	Description
31 ARBIT_EN	Enables Arbitration 0x0 **0** — Arbitration disable. If using 2 channels, will output 2 axi bus sets. 0x1 **1** — Arbitration enable. If using 2 channel, will only output 1 axi bus sets
30–26 RSVD0	Reserved, always set to zero.
25–24 HANDSHAKE_ SCAN_LINE_ NUM	scan handshake line number 0x0 **0** — 1 line. 0x1 **1** — 8 lines 0x2 **2** — 16 lines 0x3 **3** — 16 lines
23–18 RSVD1	Reserved, always set to zero.
17–16 RD_NUM_ BYTES	Bytes in a read burst 0x0 **8_bytes** — 8 bytes. 0x1 **16_bytes** — 16 bytes. 0x2 **32_bytes** — 32 bytes. 0x3 **64_bytes** — 64 bytes.
15–14 RSVD2	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_DITHER_FETCH_CTRL_CH0 field descriptions (continued)

Field	Description
13–12 ROTATION_ ANGLE	0x0 **ROT_0** — Rotate image by 0 degrees. 0x1 **ROT_90** — Rotate image by 90 degrees. 0x2 **ROT_180** — Rotate image by 180 degrees. 0x3 **ROT_270** — Rotate image by 270 degrees.
11 RSVD3	Reserved, always set to zero.
10 VFLIP	Enables VFLIP 0x0 **0** — VFLIP disable 0x1 **1** — VFLIP enable
9 HFLIP	Enables HFLIP. 0x0 **0** — HFLIP disable 0x1 **1** — VFLIP enable
8–6 RSVD4	Reserved, always set to zero.
5 HIGH_BYTE	channel 0 high byte selection 0x0 **0** — In 64 bit mode, the output high byte will use channel1. 0x1 **1** — In 64 bit mode, the output high byte will use channel0
4 BYPASS_ PIXEL_EN	Selects Channel 0 pixel source 0x0 **0** — Channel 0 is from memory 0x1 **1** — Channel 0 is from previous process engine
3 HANDSHAKE_ EN	Enable bit for handshake with the store engine. 0x0 **0** — Handshake with the store engine is disabled 0x1 **1** — Handshake with the store engine is enabled
2 BLOCK_16	Determines the block sixe. 0x0 **8x8** — Block size is 8x8 0x1 **16x16** — Block size is 16x16
1 BLOCK_EN	Choses the prefetch mode. 0x0 **0** — Prefetch in scan mode 0x1 **1** — Prefetch in block mode
0 CH_EN	Channel enable. 0x0 **0** — Prefetch function is disable 0x1 **1** — Prefetch function is enable

13.6.12.120 Pre-fetch engine Control Channel 1 Register (PXP_HW_PXP_DITHER_FETCH_CTRL_CH1)

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_DITHER_FETCH_CTRL_CH1: 0x860

HW_PXP_DITHER_FETCH_CTRL_CH1_SET: 0x864

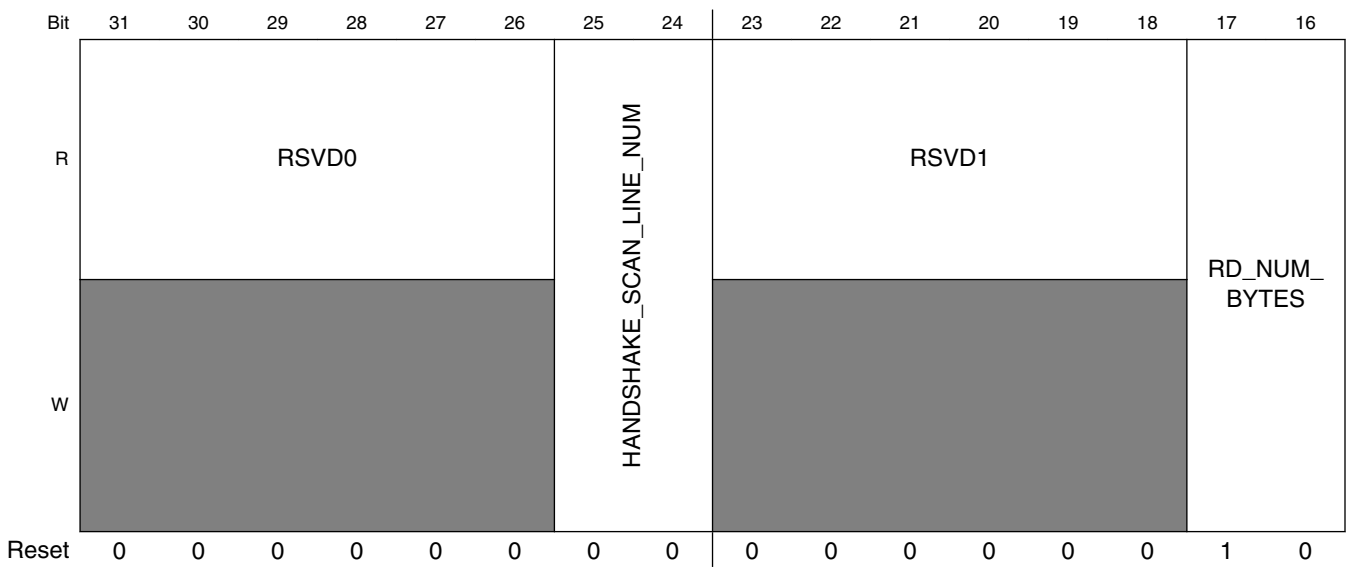
HW_PXP_DITHER_FETCH_CTRL_CH1_CLR: 0x868

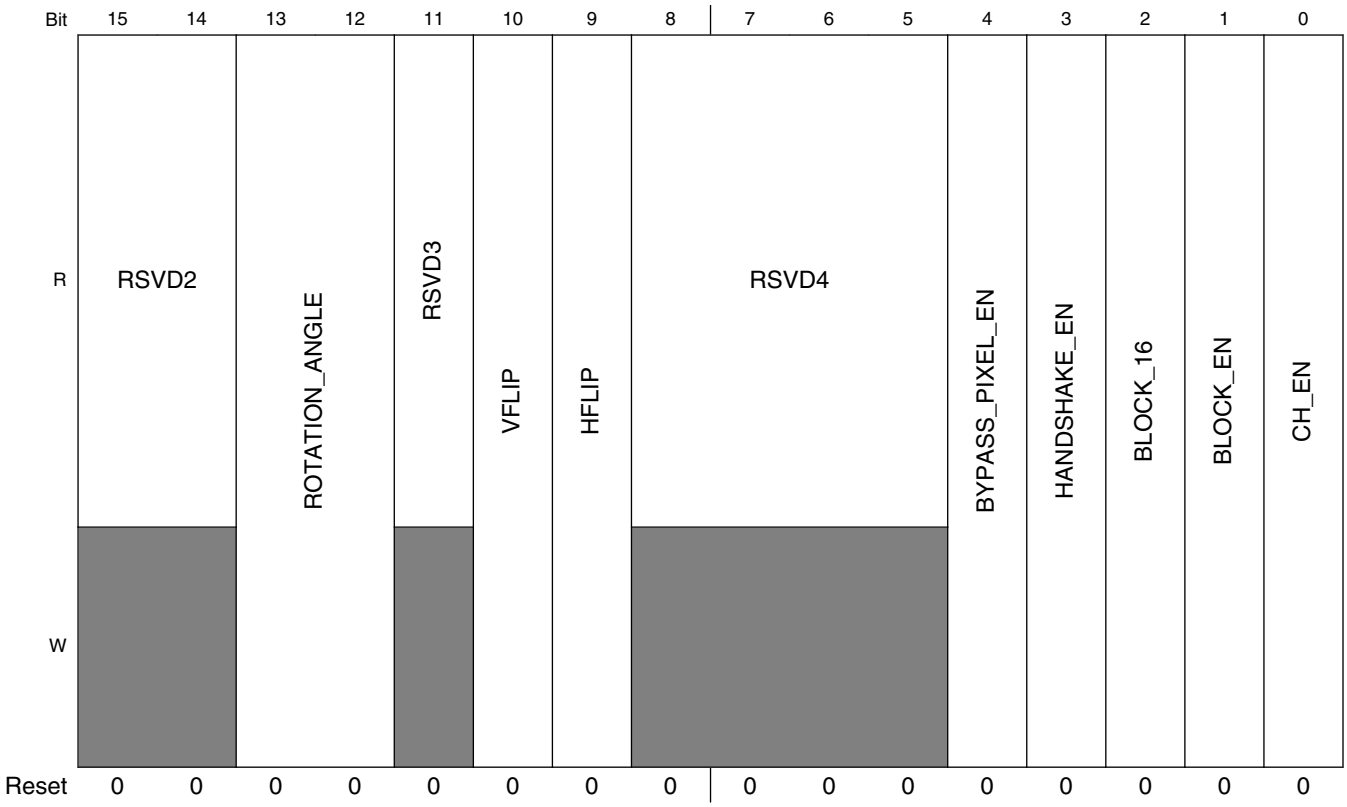
HW_PXP_DITHER_FETCH_CTRL_CH1_TOG: 0x86C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 860h offset = 3070_0860h





PXP_HW_PXP_DITHER_FETCH_CTRL_CH1 field descriptions

Field	Description
31–26 RSVD0	Reserved, always set to zero.
25–24 HANDSHAKE_ SCAN_LINE_ NUM	scan handshake line number 0x0 **0** — 1 line. 0x1 **1** — 8 lines 0x2 **2** — 16 lines 0x3 **3** — 16 lines
23–18 RSVD1	Reserved, always set to zero.
17–16 RD_NUM_ BYTES	Bytes in a read burst 0x0 **8_bytes** — 8 bytes. 0x1 **16_bytes** — 16 bytes. 0x2 **32_bytes** — 32 bytes. 0x3 **64_bytes** — 64 bytes.
15–14 RSVD2	Reserved, always set to zero.
13–12 ROTATION_ ANGLE	0x0 **ROT_0** — Rotate image by 0 degrees. 0x1 **ROT_90** — Rotate image by 90 degrees. 0x2 **ROT_180** — Rotate image by 180 degrees. 0x3 **ROT_270** — Rotate image by 270 degrees.

Table continues on the next page...

PXP_HW_PXP_DITHER_FETCH_CTRL_CH1 field descriptions (continued)

Field	Description
11 RSVD3	Reserved, always set to zero.
10 VFLIP	Enables VFLIP 0x0 **0** — VFLIP disable 0x1 **1** — VFLIP enable
9 HFLIP	Enables HFLIP. 0x0 **0** — HFLIP disable 0x1 **1** — VFLIP enable
8–5 RSVD4	Reserved, always set to zero.
4 BYPASS_ PIXEL_EN	Selects Channel 1 pixel source 0x0 **0** — Channel 1 is from memory 0x1 **1** — Channel 1 is from previous process engine
3 HANDSHAKE_ EN	Enable bit for handshake with the store engine. 0x0 **0** — Handshake with the store engine is disabled 0x1 **1** — Handshake with the store engine is enabled
2 BLOCK_16	Determines the block sixe. 0x0 **8x8** — Block size is 8x8 0x1 **16x16** — Block size is 16x16
1 BLOCK_EN	Choses the prefetch mode. 0x0 **0** — Prefetch in scan mode 0x1 **1** — Prefetch in block mode
0 CH_EN	Channel enable. 0x0 **0** — Prefetch function is disable 0x1 **1** — Prefetch function is enable

13.6.12.121 Pre-fetch engine status Channel 0 Register (PXP_HW_PXP_DITHER_FETCH_STATUS_CH0)

This register defines the status bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 870h offset = 3070_0870h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PREFETCH_BLOCK_Y																PREFETCH_BLOCK_X															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_STATUS_CH0 field descriptions

Field	Description
31–16 PREFETCH_ BLOCK_Y	When in scan mode, this field indicates the current Y coordinate of the frame. In block mode, it indicates the Y coordinate of the block currently being rendered.
PREFETCH_ BLOCK_X	When in scan mode, this field is always 0. In block mode, it indicates the X coordinate of the block currently being rendered.

13.6.12.122 Store engine status Channel 1 Register (PXP_HW_PXP_DITHER_FETCH_STATUS_CH1)

This register defines the status bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 880h offset = 3070_0880h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PREFETCH_BLOCK_Y																PREFETCH_BLOCK_X															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_STATUS_CH1 field descriptions

Field	Description
31–16 PREFETCH_ BLOCK_Y	When in scan mode, this field indicates the current Y coordinate of the frame. In block mode, it indicates the Y coordinate of the block currently being rendered.
PREFETCH_ BLOCK_X	When in scan mode, this field is always 0. In block mode, it indicates the X coordinate of the block currently being rendered.

13.6.12.123 PXP_HW_PXP_DITHER_FETCH_ACTIVE_SIZE_ULC_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 890h offset = 3070_0890h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ACTIVE_SIZE_ULC_Y																ACTIVE_SIZE_ULC_X															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_ACTIVE_SIZE_ULC_CH0 field descriptions

Field	Description
31–16 ACTIVE_SIZE_ ULC_Y	This field indicates the upper left Y-coordinate(in pixels) of the active surface of the total input memory
ACTIVE_SIZE_ ULC_X	This field indicates the upper left X-coordinate(in pixels) of the active surface of the total input memory

13.6.12.124 PXP_HW_PXP_DITHER_FETCH_ACTIVE_SIZE_LRC_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 8A0h offset = 3070_08A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ACTIVE_SIZE_LRC_Y																ACTIVE_SIZE_LRC_X															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_ACTIVE_SIZE_LRC_CH0 field descriptions

Field	Description
31–16 ACTIVE_SIZE_ LRC_Y	This field indicates the upper left Y-coordinate(in pixels) of the active surface of the total input memory
ACTIVE_SIZE_ LRC_X	This field indicates the upper left X-coordinate(in pixels) of the active surface of the total input memory

13.6.12.125 PXP_HW_PXP_DITHER_FETCH_ACTIVE_SIZE_ULC_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 8B0h offset = 3070_08B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ACTIVE_SIZE_ULC_Y																ACTIVE_SIZE_ULC_X															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_ACTIVE_SIZE_ULC_CH1 field descriptions

Field	Description
31–16 ACTIVE_SIZE_ ULC_Y	This field indicates the upper left Y-coordinate(in pixels) of the active surface of the total input memory
ACTIVE_SIZE_ ULC_X	This field indicates the upper left X-coordinate(in pixels) of the active surface of the total input memory

13.6.12.126 PXP_HW_PXP_DITHER_FETCH_ACTIVE_SIZE_LRC_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 8C0h offset = 3070_08C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ACTIVE_SIZE_LRC_Y																ACTIVE_SIZE_LRC_X															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_ACTIVE_SIZE_LRC_CH1 field descriptions

Field	Description
31–16 ACTIVE_SIZE_ LRC_Y	This field indicates the upper left Y-coordinate(in pixels) of the active surface of the total input memory
ACTIVE_SIZE_ LRC_X	This field indicates the upper left X-coordinate(in pixels) of the active surface of the total input memory

13.6.12.127 PXP_HW_PXP_DITHER_FETCH_SIZE_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 8D0h offset = 3070_08D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_TOTAL_HEIGHT																INPUT_TOTAL_WIDTH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_SIZE_CH0 field descriptions

Field	Description
31–16 INPUT_TOTAL_ HEIGHT	actual total height -1
INPUT_TOTAL_ WIDTH	actual total widht -1

13.6.12.128 PXP_HW_PXP_DITHER_FETCH_SIZE_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 8E0h offset = 3070_08E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_TOTAL_HEIGHT																INPUT_TOTAL_WIDTH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_SIZE_CH1 field descriptions

Field	Description
31–16 INPUT_TOTAL_ HEIGHT	acutal total height -1
INPUT_TOTAL_ WIDTH	actual_total_width -1

13.6.12.129 PXP_HW_PXP_DITHER_FETCH_BACKGROUND_COLOR_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 8F0h offset = 3070_08F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BACKGROUND_COLOR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_BACKGROUND_COLOR_CH0 field descriptions

Field	Description
BACKGROUND_COLOR	background color(in 32bpp format) for any pixels not within the bufffer range specified by the ULC/LRC

13.6.12.130 PXP_HW_PXP_DITHER_FETCH_BACKGROUND_COLOR_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 900h offset = 3070_0900h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BACKGROUND_COLOR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_BACKGROUND_COLOR_CH1 field descriptions

Field	Description
BACKGROUND_COLOR	background color(in 32bpp format) for any pixels not within the bufffer range specified by the ULC/LRC

13.6.12.131 PXP_HW_PXP_DITHER_FETCH_PITCH

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 910h offset = 3070_0910h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH1_INPUT_PITCH																CH0_INPUT_PITCH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_PITCH field descriptions

Field	Description
31–16 CH1_INPUT_ PITCH	This field indicates the channel 1 input pitch
CH0_INPUT_ PITCH	This field indicates the channel 0 input pitch

13.6.12.132 PXP_HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH0: 0x920

HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH0_SET: 0x924

HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH0_CLR: 0x928

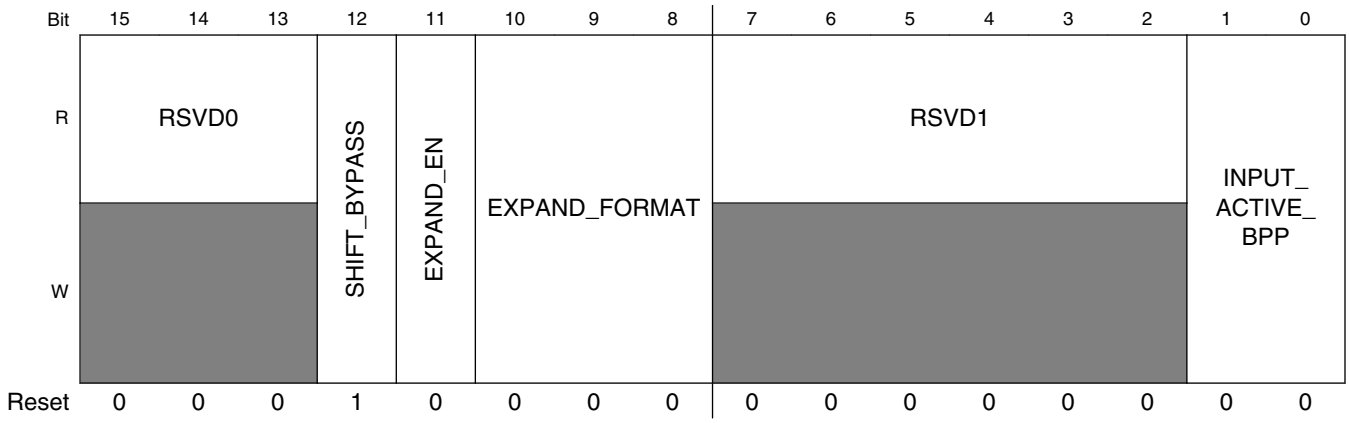
HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH0_TOG: 0x92C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 920h offset = 3070_0920h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



PXP_HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH0 field descriptions

Field	Description
31–13 RSVD0	Reserved, always set to zero.
12 SHIFT_BYPASS	0x0 **0** — channel0 data will do shift function 0x1 **1** — channel0 will bypass shift function
11 EXPAND_EN	0x0 **0** — channel0 format expanding disable 0x1 **1** — channel0 format expanding enable
10–8 EXPAND_ FORMAT	Select Pixel format 0x0 **0** — RGB 565 0x1 **1** — RGB 555 0x2 **2** — ARGB 1555 0x3 **3** — RGB 444 0x4 **4** — ARGB 4444 0x5 **5** — YUYV/YVYU 0x6 **6** — UYVY/VYUY 0x7 **7** — YUV422_2P
7–2 RSVD1	Reserved, always set to zero.
INPUT_ACTIVE_ BPP	0x0 **0** — 8 bits 0x1 **1** — 16 bits 0x2 **2** — 32 bits 0x3 **3** — 32 bits

13.6.12.133 PXP_HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH1: 0x930

HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH1_SET: 0x934

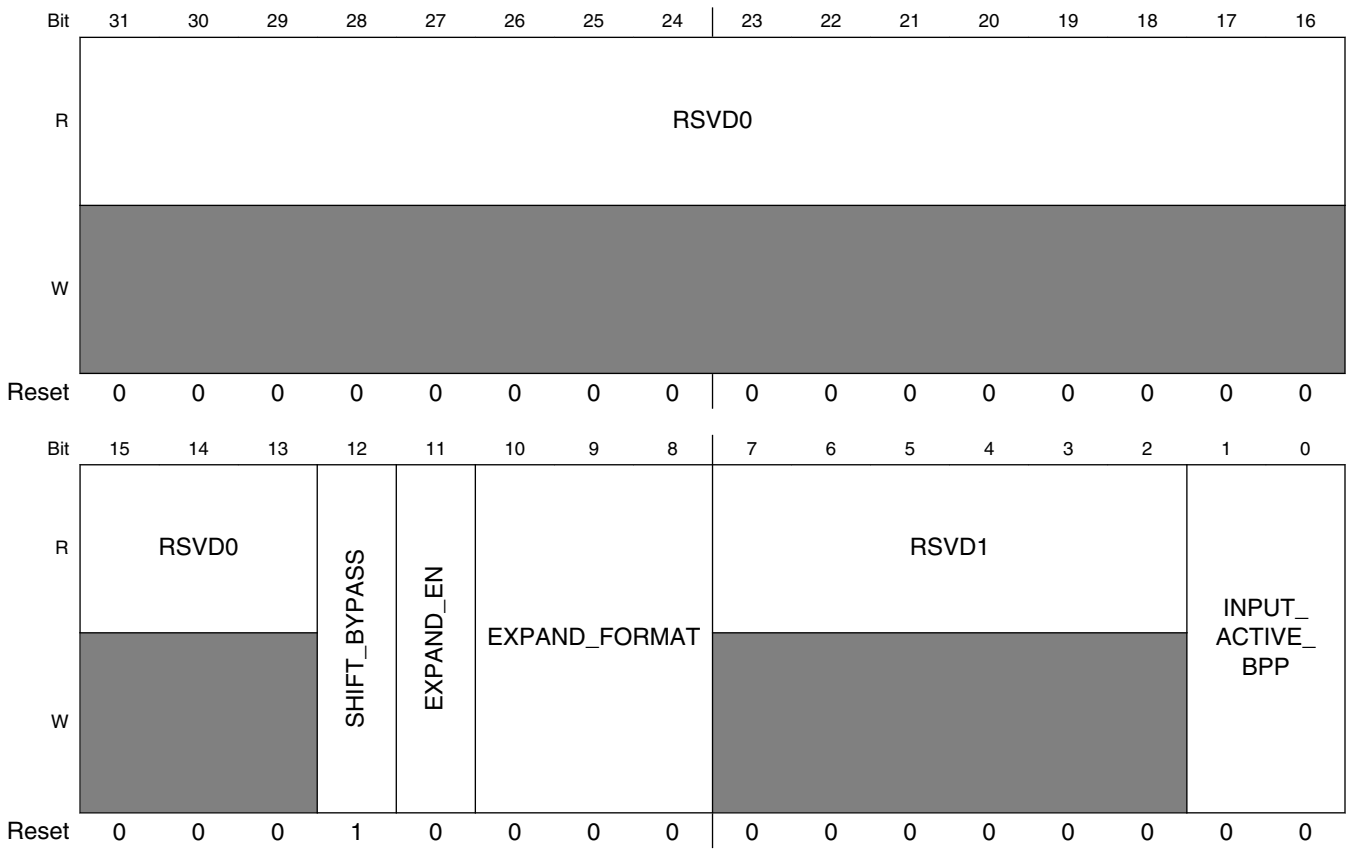
HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH1_CLR: 0x938

HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH1_TOG: 0x93C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 930h offset = 3070_0930h



PXP_HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH1 field descriptions

Field	Description
31–13 RSVD0	Reserved, always set to zero.
12 SHIFT_BYPASS	0x0 **0** — channel1 data will do shift function 0x1 **1** — channel1 will bypass shift function
11 EXPAND_EN	0x0 **0** — channel1 format expanding disable 0x1 **1** — channel1 format expanding enable
10–8 EXPAND_ FORMAT	Select Pixel format 0x0 **0** — RGB 565 0x1 **1** — RGB 555 0x2 **2** — ARGB 1555 0x3 **3** — RGB 444 0x4 **4** — ARGB 4444

Table continues on the next page...

PXP_HW_PXP_DITHER_FETCH_SHIFT_CTRL_CH1 field descriptions (continued)

Field	Description
	0x5 **5** — YUYV/YVYU 0x6 **6** — UYVY/VYUY 0x7 **7** — YUV422_2P
7–2 RSVD1	Reserved, always set to zero.
INPUT_ACTIVE_BPP	0x0 **0** — 8 bits 0x1 **1** — 16 bits 0x2 **2** — 32 bits 0x3 **3** — 32 bits

13.6.12.134 PXP_HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH0: 0x940

HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH0_SET: 0x944

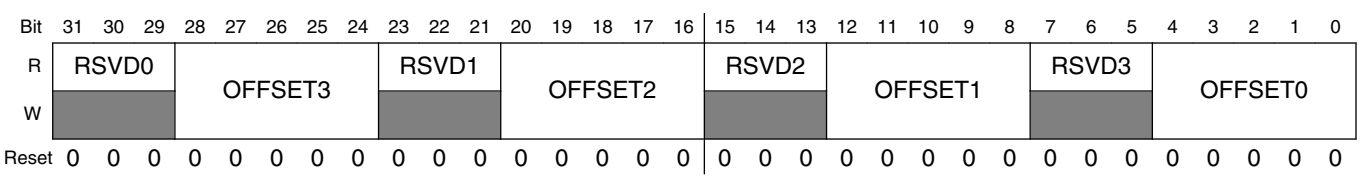
HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH0_CLR: 0x948

HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH0_TOG: 0x94C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 940h offset = 3070_0940h



PXP_HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH0 field descriptions

Field	Description
31–29 RSVD0	Reserved, always set to zero.
28–24 OFFSET3	Shift Offset for channel 0 componnent 3.
23–21 RSVD1	Reserved, always set to zero.
20–16 OFFSET2	Shift Offset for channel 0 componnent 2.

Table continues on the next page...

PXP_HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH0 field descriptions (continued)

Field	Description
15–13 RSVD2	Reserved, always set to zero.
12–8 OFFSET1	Shift Offset for channel 0 componnent 1.
7–5 RSVD3	Reserved, always set to zero.
OFFSET0	Shift Offset for channel 0 componnent 0.

13.6.12.135 PXP_HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH1: 0x950

HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH1_SET: 0x954

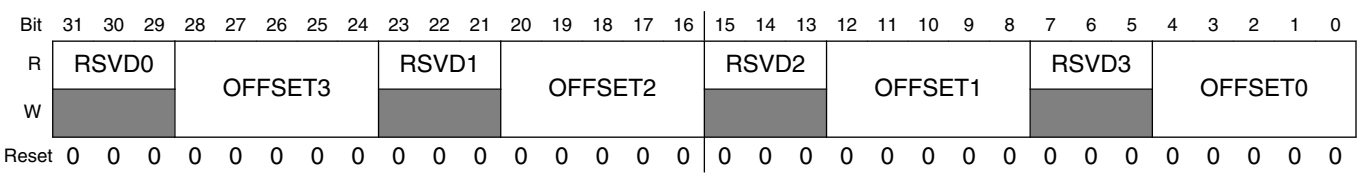
HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH1_CLR: 0x958

HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH1_TOG: 0x95C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 950h offset = 3070_0950h



PXP_HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH1 field descriptions

Field	Description
31–29 RSVD0	Reserved, always set to zero.
28–24 OFFSET3	Shift Offset for channel 1 componnent 3.
23–21 RSVD1	Reserved, always set to zero.
20–16 OFFSET2	Shift Offset for channel 1 componnent 2.
15–13 RSVD2	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_DITHER_FETCH_SHIFT_OFFSET_CH1 field descriptions (continued)

Field	Description
12–8 OFFSET1	Shift Offset for channel 1 componnent 1.
7–5 RSVD3	Reserved, always set to zero.
OFFSET0	Shift Offset for channel 1 componnent 0.

13.6.12.136 PXP_HW_PXP_DITHER_FETCH_SHIFT_WIDTH_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_DITHER_FETCH_SHIFT_WIDTH_CH0: 0x960

HW_PXP_DITHER_FETCH_SHIFT_WIDTH_CH0_SET: 0x964

HW_PXP_DITHER_FETCH_SHIFT_WIDTH_CH0_CLR: 0x968

HW_PXP_DITHER_FETCH_SHIFT_WIDTH_CH0_TOG: 0x96C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 960h offset = 3070_0960h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0																WIDTH3				WIDTH2				WIDTH1				WIDTH0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0

PXP_HW_PXP_DITHER_FETCH_SHIFT_WIDTH_CH0 field descriptions

Field	Description
31–16 RSVD0	Reserved, always set to zero.
15–12 WIDTH3	Shift Width for channel 0 componnent 3.
11–8 WIDTH2	Shift Width for channel 0 componnent 2.
7–4 WIDTH1	Shift Width for channel 0 componnent 1.
WIDTH0	Shift Width for channel 0 componnent 0.

13.6.12.137 PXP_HW_PXP_DITHER_FETCH_SHIFT_WIDTH_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_DITHER_FETCH_SHIFT_WIDTH_CH1: 0x970

HW_PXP_DITHER_FETCH_SHIFT_WIDTH_CH1_SET: 0x974

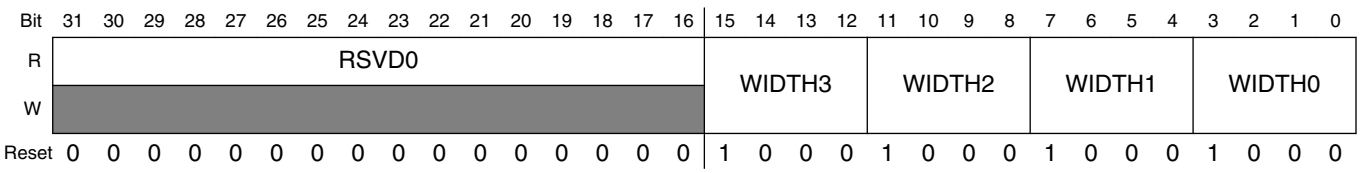
HW_PXP_DITHER_FETCH_SHIFT_WIDTH_CH1_CLR: 0x978

HW_PXP_DITHER_FETCH_SHIFT_WIDTH_CH1_TOG: 0x97C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 970h offset = 3070_0970h



PXP_HW_PXP_DITHER_FETCH_SHIFT_WIDTH_CH1 field descriptions

Field	Description
31–16 RSVD0	Reserved, always set to zero.
15–12 WIDTH3	Shift Width for channel 1 componnent 3.
11–8 WIDTH2	Shift Width for channel 1 componnent 2.
7–4 WIDTH1	Shift Width for channel 1 componnent 1.
WIDTH0	Shift Width for channel 1 componnent 0.

13.6.12.138 PXP_HW_PXP_DITHER_FETCH_ADDR_0_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 980h offset = 3070_0980h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_BASE_ADDR0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_ADDR_0_CH0 field descriptions

Field	Description
INPUT_BASE_ADDR0	input base address0. For 2 channel, indicated the channel0 base address. For 1 channel and YUV422 2 plane, indicate the Y base address

13.6.12.139 PXP_HW_PXP_DITHER_FETCH_ADDR_1_CH0

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 990h offset = 3070_0990h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_BASE_ADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_ADDR_1_CH0 field descriptions

Field	Description
INPUT_BASE_ADDR1	input base address1. For 2 channel, indicated the channel1 base address. For 1 channel and YUV422 2 plane, indicate the UV base address

13.6.12.140 PXP_HW_PXP_DITHER_FETCH_ADDR_0_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 9A0h offset = 3070_09A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_BASE_ADDR0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_ADDR_0_CH1 field descriptions

Field	Description
INPUT_BASE_ADDR0	input base address0. For 2 channel, indicated the channel0 base address. For 1 channel and YUV422 2 plane, indicate the Y base address

13.6.12.141 PXP_HW_PXP_DITHER_FETCH_ADDR_1_CH1

This register defines the control bits for the pxp prefetch_engine sub-block.

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 9B0h offset = 3070_09B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_BASE_ADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FETCH_ADDR_1_CH1 field descriptions

Field	Description
INPUT_BASE_ADDR1	input base address1. For 2 channel, indicated the channel1 base address. For 1 channel and YUV422 2 plane, indicate the UV base address

13.6.12.142 Store engine Control Channel 0 Register (PXP_HW_PXP_DITHER_STORE_CTRL_CH0)

This register defines the control bits for the pxp store_engine sub-block.

HW_PXP_DITHER_STORE_CTRL_CH0: 0x9C0

HW_PXP_DITHER_STORE_CTRL_CH0_SET: 0x9C4

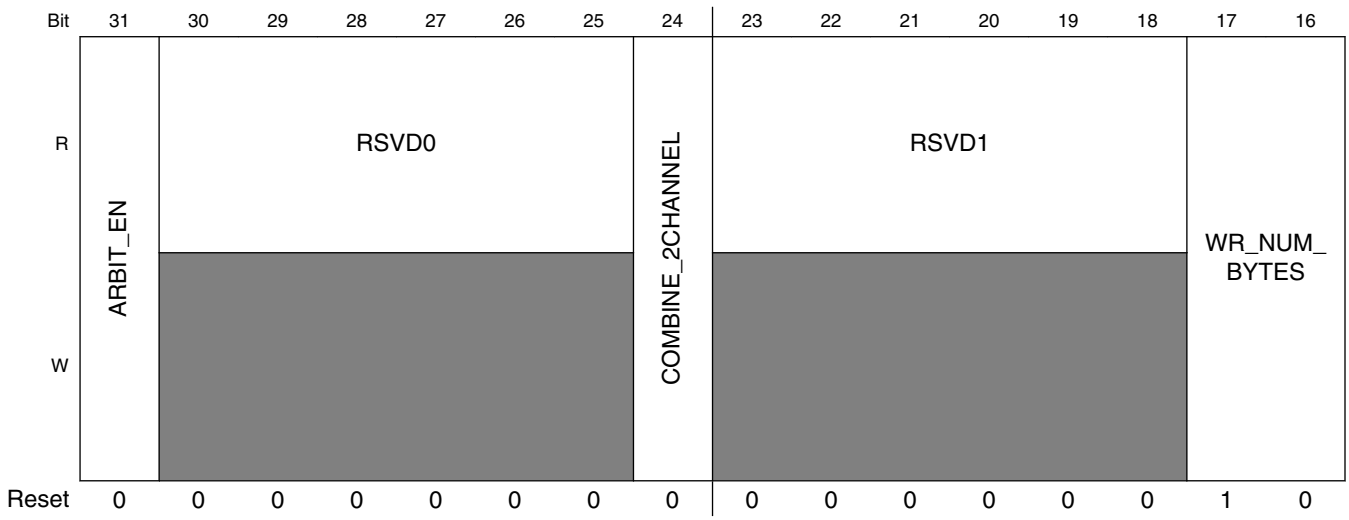
HW_PXP_DITHER_STORE_CTRL_CH0_CLR: 0x9C8

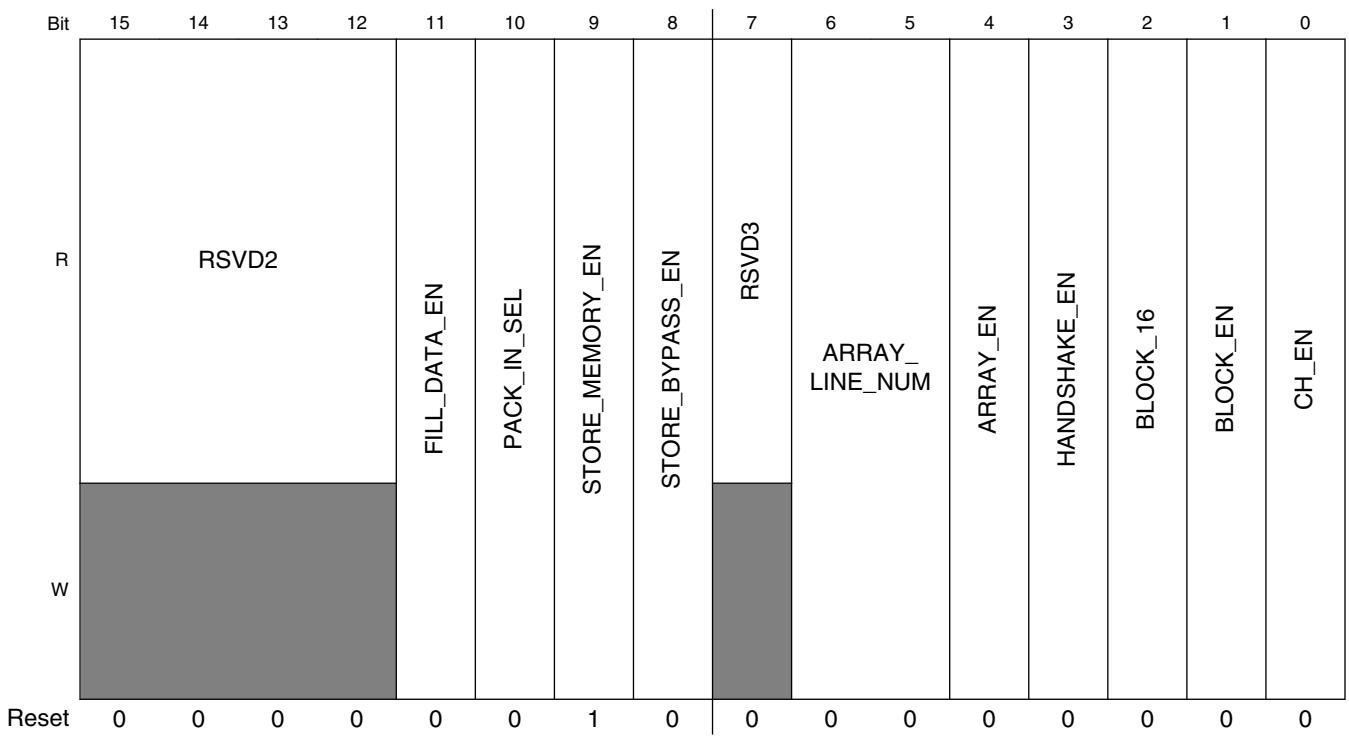
HW_PXP_DITHER_STORE_CTRL_CH0_TOG: 0x9CC

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 9C0h offset = 3070_09C0h





PXP_HW_PXP_DITHER_STORE_CTRL_CH0 field descriptions

Field	Description
31 ARBIT_EN	Arbitration Enable 0x0 **0** — Arbitration disable. If using 2 channels, will output 2 axi bus sets 0x1 **1** — Arbitration enable. If using 2 channel, will only output 1 axi bus sets
30–25 RSVD0	Reserved, always set to zero.
24 COMBINE_2CHANNEL	Combine 2 channel Enable

Table continues on the next page...

PXP_HW_PXP_DITHER_STORE_CTRL_CH0 field descriptions (continued)

Field	Description
	0x0 **0** — combine 2 channel disable 0x1 **1** — combine 2 channel enable
23–18 RSVD1	Reserved, always set to zero.
17–16 WR_NUM_ BYTES	Bytes in a write burst 0x0 **8_bytes** — 8 bytes 0x1 **16_bytes** — 16 bytes 0x2 **32_bytes** — 32 bytes 0x3 **64_bytes** — 64 bytes
15–12 RSVD2	Reserved, always set to zero.
11 FILL_DATA_EN	enable bit for fill data 0x0 **0** — Fill data mode disable. 0x1 **1** — Fill data mode enable. When using fill_data mode, store_engine will store fixed data defined in fill_data register
10 PACK_IN_SEL	pack_in_sel 0x0 **0** — select 64 shift out data to pack 0x1 **1** — select low 32 bit shift out data to pack
9 STORE_ MEMORY_EN	store memory enable 0x0 **0** — store memory mode disable. 0x1 **1** — store memory mode enable. Data will store to memory
8 STORE_ BYPASS_EN	enable bit for store bypass 0x0 **0** — store bypass mode disable. 0x1 **1** — store bypass mode enable. Data will bypass to store output.
7 RSVD3	Reserved, always set to zero.
6–5 ARRAY_LINE_ NUM	Selects Array Size 0x0 **0** — Using 1x1 Array 0x1 **1** — Using 3x3 Array 0x2 **2** — Using 5x5 Array 0x3 **3** — Using 5x5 Array
4 ARRAY_EN	0x0 **0** — Array Handshake Disabled 0x1 **1** — Array Handshake Enabled
3 HANDSHAKE_ EN	Enable bit for handshake with the store engine. 0x0 **0** — Handshake with the prefetch engine is disabled 0x1 **1** — Handshake with the prefetch engine is enabled
2 BLOCK_16	Determines the block sixe. 0x0 **8x8** — Block size is 8x8 0x1 **16x16** — Block size is 16x16

Table continues on the next page...

PXP_HW_PXP_DITHER_STORE_CTRL_CH0 field descriptions (continued)

Field	Description
1 BLOCK_EN	Choses the store mode. 0x0 **0** — Store in scan mode 0x1 **1** — Store in block mode
0 CH_EN	Channel enable. 0x0 **0** — Store function is disable 0x1 **1** — Store function is enable

13.6.12.143 Store engine Control Channel 1 Register (PXP_HW_PXP_DITHER_STORE_CTRL_CH1)

This register defines the control bits for the pxp prefetch_engine sub-block.

HW_PXP_DITHER_STORE_CTRL_CH1: 0x9D0

HW_PXP_DITHER_STORE_CTRL_CH1_SET: 0x9D4

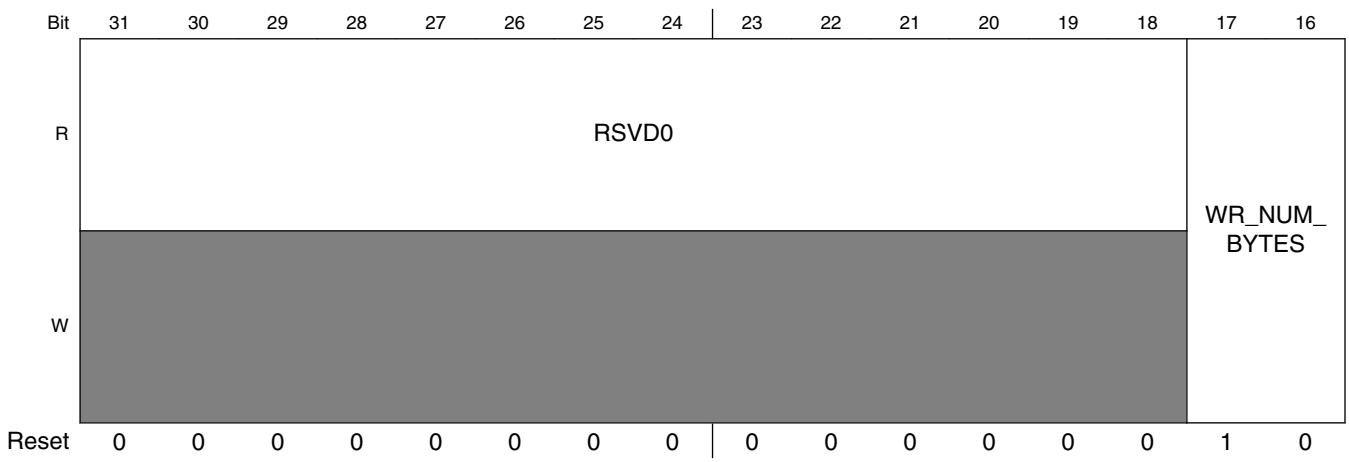
HW_PXP_DITHER_STORE_CTRL_CH1_CLR: 0x9D8

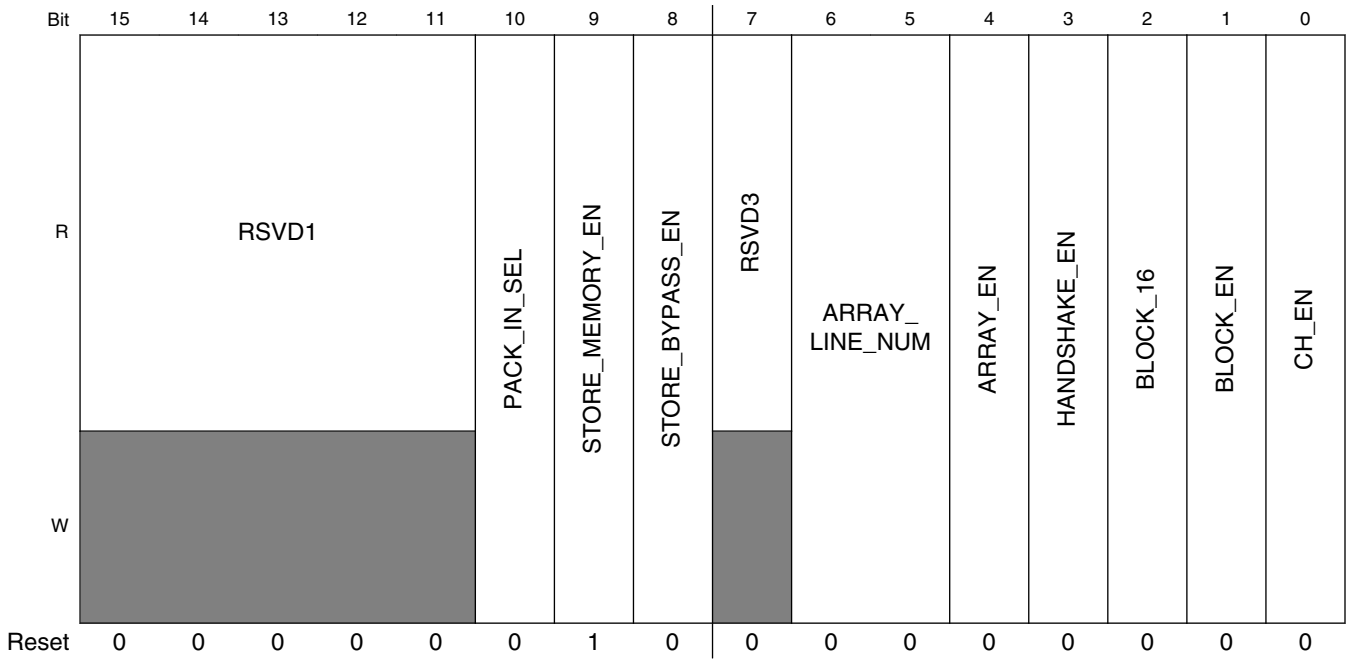
HW_PXP_DITHER_STORE_CTRL_CH1_TOG: 0x9DC

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 9D0h offset = 3070_09D0h





PXP_HW_PXP_DITHER_STORE_CTRL_CH1 field descriptions

Field	Description
31–18 RSVD0	Reserved, always set to zero.
17–16 WR_NUM_ BYTES	Bytes in a write burst 0x0 **8_bytes** — 8 bytes 0x1 **16_bytes** — 16 bytes 0x2 **32_bytes** — 32 bytes 0x3 **64_bytes** — 64 bytes
15–11 RSVD1	Reserved, always set to zero.
10 PACK_IN_SEL	pack_in_sel 0x0 **0** — select 64 shift out data to pack 0x1 **1** — select channel 0 high 32 bit shift out data to pack
9 STORE_ MEMORY_EN	store memory enable 0x0 **0** — store memory mode disable. 0x1 **1** — store memory mode enable. Data will store to memory
8 STORE_ BYPASS_EN	enable bit for store bypass 0x0 **0** — store bypass mode disable. 0x1 **1** — store bypass mode enable. Data will bypass to store output.
7 RSVD3	Reserved, always set to zero.
6–5 ARRAY_LINE_ NUM	Selects Array Size 0x0 **0** — Using 1x1 Array

Table continues on the next page...

PXP_HW_PXP_DITHER_STORE_CTRL_CH1 field descriptions (continued)

Field	Description
	0x1 **1** — Using 3x3 Array 0x2 **2** — Using 5x5 Array 0x3 **3** — Using 5x5 Array
4 ARRAY_EN	0x0 **0** — Array Handshake Disabled 0x1 **1** — Array Handshake Enabled
3 HANDSHAKE_ EN	Enable bit for handshake with the fetch engine. 0x0 **0** — Handshake with the fetch engine is disabled 0x1 **1** — Handshake with the fetch engine is enabled
2 BLOCK_16	Determines the block sixe. 0x0 **8x8** — Block size is 8x8 0x1 **16x16** — Block size is 16x16
1 BLOCK_EN	Choses the store mode. 0x0 **0** — Store in scan mode 0x1 **1** — Store in block mode
0 CH_EN	Channel enable. 0x0 **0** — Store function is disable 0x1 **1** — Store function is enable

13.6.12.144 Store engine status Channel 0 Register (PXP_HW_PXP_DITHER_STORE_STATUS_CH0)

This register defines the status bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 9E0h offset = 3070_09E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STORE_BLOCK_Y																STORE_BLOCK_X															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_STATUS_CH0 field descriptions

Field	Description
31–16 STORE_ BLOCK_Y	When in scan mode, this field indicates the current Y coordinate of the frame. In block mode, it indicates the Y coordinate of the block currently being rendered.

Table continues on the next page...

PXP_HW_PXP_DITHER_STORE_STATUS_CH0 field descriptions (continued)

Field	Description
STORE_ BLOCK_X	When in scan mode, this field is always 0. In block mode, it indicates the X coordinate of the block currently being rendered.

13.6.12.145 Store engine status Channel 1 Register (PXP_HW_PXP_DITHER_STORE_STATUS_CH1)

This register defines the status bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + 9F0h offset = 3070_09F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STORE_BLOCK_Y																STORE_BLOCK_X															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_STATUS_CH1 field descriptions

Field	Description
31–16 STORE_ BLOCK_Y	When in scan mode, this field indicates the current Y coordinate of the frame. In block mode, it indicates the Y coordinate of the block currently being rendered.
STORE_ BLOCK_X	When in scan mode, this field is always 0. In block mode, it indicates the X coordinate of the block currently being rendered.

13.6.12.146 PXP_HW_PXP_DITHER_STORE_SIZE_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + A00h offset = 3070_0A00h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OUT_HEIGHT																OUT_WIDTH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_SIZE_CH0 field descriptions

Field	Description
31–16 OUT_HEIGHT	actual output height -1
OUT_WIDTH	actual output width -1

13.6.12.147 PXP_HW_PXP_DITHER_STORE_SIZE_CH1

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + A10h offset = 3070_0A10h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OUT_HEIGHT																OUT_WIDTH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_SIZE_CH1 field descriptions

Field	Description
31–16 OUT_HEIGHT	actual output height -1
OUT_WIDTH	actual output width -1

13.6.12.148 PXP_HW_PXP_DITHER_STORE_PITCH

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + A20h offset = 3070_0A20h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH1_OUT_PITCH																CH0_OUT_PITCH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_PITCH field descriptions

Field	Description
31–16 CH1_OUT_ PITCH	This field indicates the channel 1 input pitch
CH0_OUT_ PITCH	This field indicates the channel 0 input pitch

13.6.12.149 PXP_HW_PXP_DITHER_STORE_SHIFT_CTRL_CH0

This register defines the control bits for the pxp store_engine sub-block.

HW_PXP_DITHER_STORE_SHIFT_CTRL_CH0: 0xA30

HW_PXP_DITHER_STORE_SHIFT_CTRL_CH0_SET: 0xA34

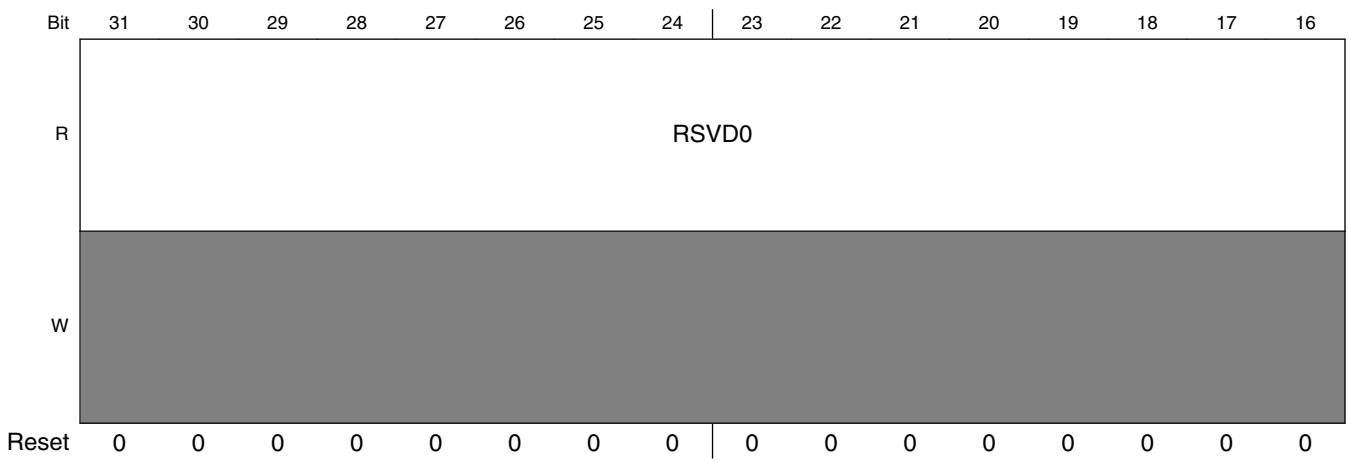
HW_PXP_DITHER_STORE_SHIFT_CTRL_CH0_CLR: 0xA38

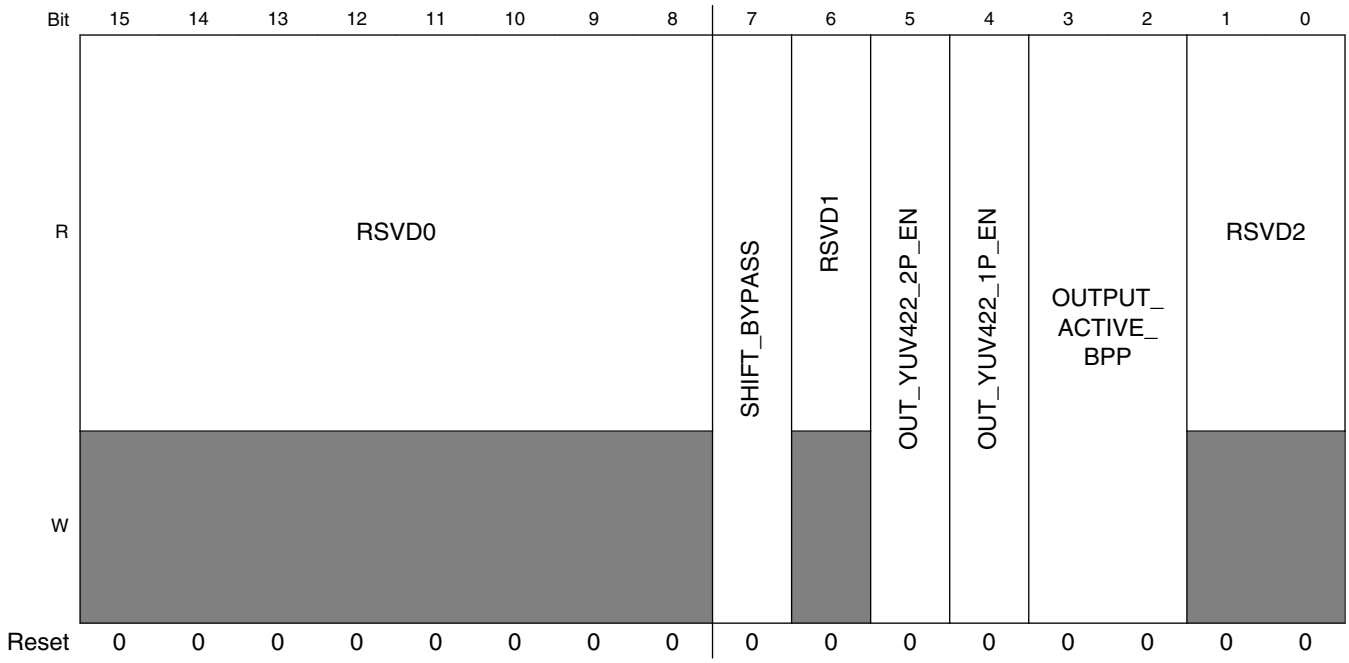
HW_PXP_DITHER_STORE_SHIFT_CTRL_CH0_TOG: 0xA3C

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + A30h offset = 3070_0A30h





PXP_HW_PXP_DITHER_STORE_SHIFT_CTRL_CH0 field descriptions

Field	Description
31–8 RSVD0	Reserved, always set to zero.
7 SHIFT_BYPASS	CH0 shift bypass 0x0 **0** — data will do shift processing. 0x1 **1** — data will bypass shift module.
6 RSVD1	Reserved, always set to zero.
5 OUT_YUV422_ 2P_EN	Enable for YUV422 2 plane 0x0 **0** — YUYV422 2 plane disabled. 0x1 **1** — YUYV422 2 plane enabled.
4 OUT_YUV422_ 1P_EN	Enable for YUV422 1 plane 0x0 **0** — YUYV422 2 plane disabled. 0x1 **1** — YUYV422 2 plane enabled.
3–2 OUTPUT_ ACTIVE_BPP	0x0 **0** — 8 bits 0x1 **1** — 16 bits 0x2 **2** — 32 bits 0x3 **3** — 32 bits
RSVD2	Reserved, always set to zero.

13.6.12.150 PXP_HW_PXP_DITHER_STORE_SHIFT_CTRL_CH1

This register defines the control bits for the pxp store_engine sub-block.

HW_PXP_DITHER_STORE_SHIFT_CTRL_CH1: 0xA40

HW_PXP_DITHER_STORE_SHIFT_CTRL_CH1_SET: 0xA44

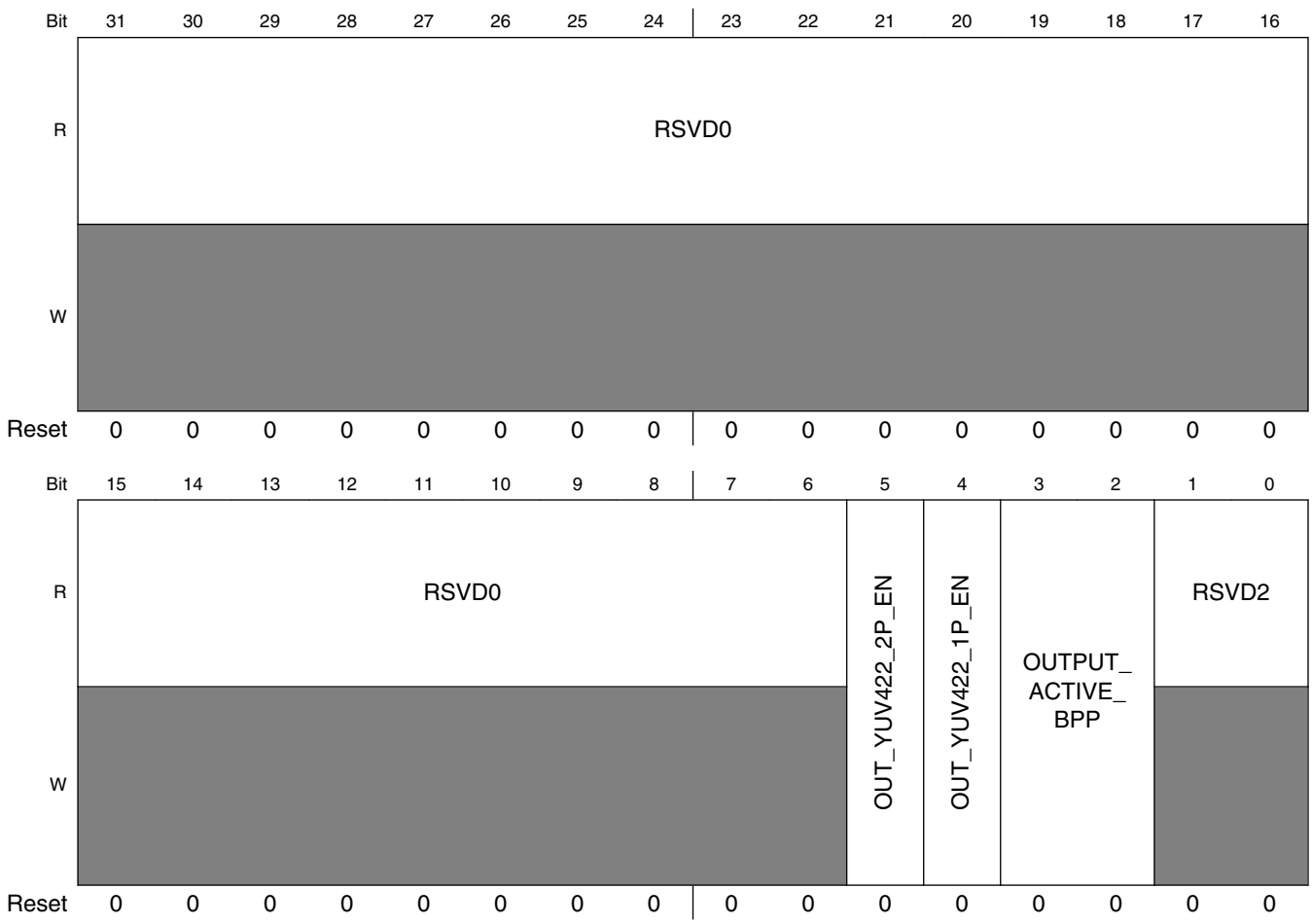
HW_PXP_DITHER_STORE_SHIFT_CTRL_CH1_CLR: 0xA48

HW_PXP_DITHER_STORE_SHIFT_CTRL_CH1_TOG: 0xA4C

The Control register contains the control bits for the pxp prefetch_engine sub-block.

EXAMPLE

Address: 3070_0000h base + A40h offset = 3070_0A40h



PXP_HW_PXP_DITHER_STORE_SHIFT_CTRL_CH1 field descriptions

Field	Description
31–6 RSVD0	Reserved, always set to zero.
5 OUT_YUV422_2P_EN	Enable for YUV422 2 plane

Table continues on the next page...

PXP_HW_PXP_DITHER_STORE_SHIFT_CTRL_CH1 field descriptions (continued)

Field	Description
	0x0 **0** — YUYV422 2 plane disabled. 0x1 **1** — YUYV422 2 plane enabled.
4 OUT_YUV422_ 1P_EN	Enable for YUV422 1 plane 0x0 **0** — YUYV422 2 plane disabled. 0x1 **1** — YUYV422 2 plane enabled.
3–2 OUTPUT_ ACTIVE_BPP	0x0 **0** — 8 bits 0x1 **1** — 16 bits 0x2 **2** — 32 bits 0x3 **3** — 32 bits
RSVD2	Reserved, always set to zero.

13.6.12.151 PXP_HW_PXP_DITHER_STORE_ADDR_0_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + A90h offset = 3070_0A90h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OUT_BASE_ADDR0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_ADDR_0_CH0 field descriptions

Field	Description
OUT_BASE_ ADDR0	input base address0. For 2 channel, indicated the channel0 base address. For 1 channel and YUV422 2 plane, indicate the Y base address

13.6.12.152 PXP_HW_PXP_DITHER_STORE_ADDR_1_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + AA0h offset = 3070_0AA0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OUT_BASE_ADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_ADDR_1_CH0 field descriptions

Field	Description
OUT_BASE_ADDR1	input base address1. For 2 channel, indicated the channel 1 base address. For 1 channel and YUV422 2 plane, indicate the UV base address

13.6.12.153 PXP_HW_PXP_DITHER_STORE_FILL_DATA_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + AB0h offset = 3070_0AB0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FILL_DATA_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_FILL_DATA_CH0 field descriptions

Field	Description
FILL_DATA_CH0	when using fill_data mode,store engine channel0 will store the fill_data value defined here.

13.6.12.154 PXP_HW_PXP_DITHER_STORE_ADDR_0_CH1

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + AC0h offset = 3070_0AC0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OUT_BASE_ADDR0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_ADDR_0_CH1 field descriptions

Field	Description
OUT_BASE_ ADDR0	input base address0. For 2 channel, indicated the channel0 base address. For 1 channel and YUV422 2 plane, indicate the Y base address

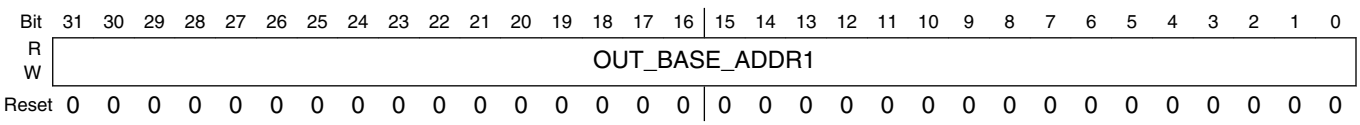
13.6.12.155 PXP_HW_PXP_DITHER_STORE_ADDR_1_CH1

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + AD0h offset = 3070_0AD0h



PXP_HW_PXP_DITHER_STORE_ADDR_1_CH1 field descriptions

Field	Description
OUT_BASE_ ADDR1	input base address1. For 2 channel, indicated the channel 1 base address. For 1 channel and YUV422 2 plane, indicate the UV base address

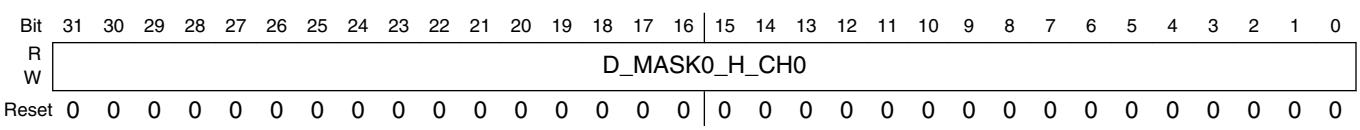
13.6.12.156 PXP_HW_PXP_DITHER_STORE_D_MASK0_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + AE0h offset = 3070_0AE0h



PXP_HW_PXP_DITHER_STORE_D_MASK0_H_CH0 field descriptions

Field	Description
D_MASK0_H_ CH0	data mask0 high byte

13.6.12.157 PXP_HW_PXP_DITHER_STORE_D_MASK0_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + AF0h offset = 3070_0AF0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK0_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK0_L_CH0 field descriptions

Field	Description
D_MASK0_L_CH0	data mask0 low byte

13.6.12.158 PXP_HW_PXP_DITHER_STORE_D_MASK1_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + B00h offset = 3070_0B00h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK1_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK1_H_CH0 field descriptions

Field	Description
D_MASK1_H_CH0	data mask1 high byte

13.6.12.159 PXP_HW_PXP_DITHER_STORE_D_MASK1_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + B10h offset = 3070_0B10h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK1_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK1_L_CH0 field descriptions

Field	Description
D_MASK1_L_CH0	data mask1 low byte

13.6.12.160 PXP_HW_PXP_DITHER_STORE_D_MASK2_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + B20h offset = 3070_0B20h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK2_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK2_H_CH0 field descriptions

Field	Description
D_MASK2_H_CH0	data mask2 high byte

13.6.12.161 PXP_HW_PXP_DITHER_STORE_D_MASK2_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + B30h offset = 3070_0B30h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK2_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK2_L_CH0 field descriptions

Field	Description
D_MASK2_L_CH0	data mask2 low byte

13.6.12.162 PXP_HW_PXP_DITHER_STORE_D_MASK3_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + B40h offset = 3070_0B40h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK3_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK3_H_CH0 field descriptions

Field	Description
D_MASK3_H_CH0	data mask3 high byte

13.6.12.163 PXP_HW_PXP_DITHER_STORE_D_MASK3_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + B50h offset = 3070_0B50h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK3_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK3_L_CH0 field descriptions

Field	Description
D_MASK3_L_CH0	data mask3 low byte

13.6.12.164 PXP_HW_PXP_DITHER_STORE_D_MASK4_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + B60h offset = 3070_0B60h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK4_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK4_H_CH0 field descriptions

Field	Description
D_MASK4_H_CH0	data mask4 high byte

13.6.12.165 PXP_HW_PXP_DITHER_STORE_D_MASK4_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + B70h offset = 3070_0B70h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK4_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK4_L_CH0 field descriptions

Field	Description
D_MASK4_L_CH0	data mask4 low byte

13.6.12.166 PXP_HW_PXP_DITHER_STORE_D_MASK5_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + B80h offset = 3070_0B80h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK5_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK5_H_CH0 field descriptions

Field	Description
D_MASK5_H_CH0	data mask5 high byte

13.6.12.167 PXP_HW_PXP_DITHER_STORE_D_MASK5_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + B90h offset = 3070_0B90h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK5_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK5_L_CH0 field descriptions

Field	Description
D_MASK5_L_CH0	data mask5 low byte

13.6.12.168 PXP_HW_PXP_DITHER_STORE_D_MASK6_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + BA0h offset = 3070_0BA0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK6_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK6_H_CH0 field descriptions

Field	Description
D_MASK6_H_CH0	data mask6 high byte

13.6.12.169 PXP_HW_PXP_DITHER_STORE_D_MASK6_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + BB0h offset = 3070_0BB0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK6_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK6_L_CH0 field descriptions

Field	Description
D_MASK6_L_CH0	data mask6 low byte

13.6.12.170 PXP_HW_PXP_DITHER_STORE_D_MASK7_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + BC0h offset = 3070_0BC0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK7_H_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK7_H_CH0 field descriptions

Field	Description
D_MASK7_H_CH0	data mask7 high byte

13.6.12.171 PXP_HW_PXP_DITHER_STORE_D_MASK7_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + BD0h offset = 3070_0BD0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_MASK7_L_CH0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_D_MASK7_L_CH0 field descriptions

Field	Description
D_MASK7_L_CH0	data mask7 low byte

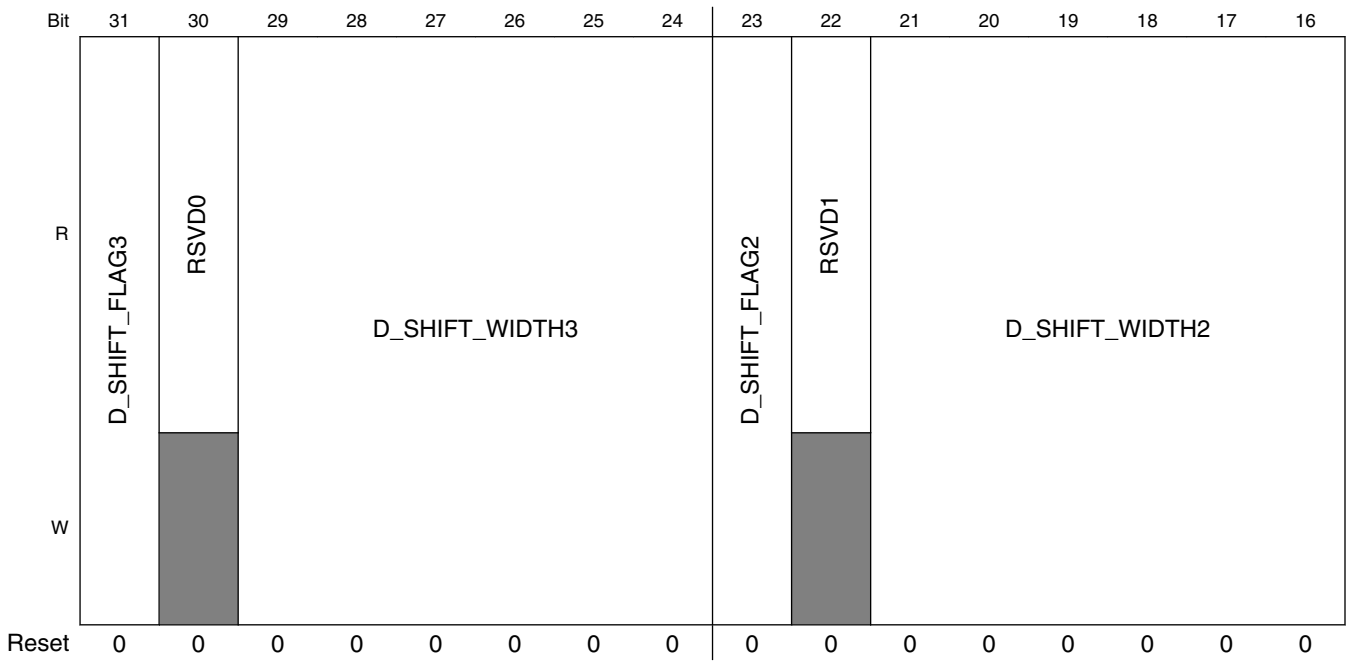
13.6.12.172 PXP_HW_PXP_DITHER_STORE_D_SHIFT_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + BE0h offset = 3070_0BE0h



PXP_HW_PXP_DITHER_STORE_D_SHIFT_L_CH0 field descriptions

Field	Description
31 D_SHIFT_FLAG3	data shift flag 3
30 RSVD0	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_DITHER_STORE_D_SHIFT_L_CH0 field descriptions (continued)

Field	Description
29–24 D_SHIFT_ WIDTH3	data shift width 3
23 D_SHIFT_FLAG2	data shift flag 2
22 RSVD1	Reserved, always set to zero.
21–16 D_SHIFT_ WIDTH2	data shift width 2
15 D_SHIFT_FLAG1	data shift flag 1
14 RSVD2	Reserved, always set to zero.
13–8 D_SHIFT_ WIDTH1	data shift width 1
7 D_SHIFT_FLAG0	data shift flag 0
6 RSVD3	Reserved, always set to zero.
D_SHIFT_ WIDTH0	data shift width 0

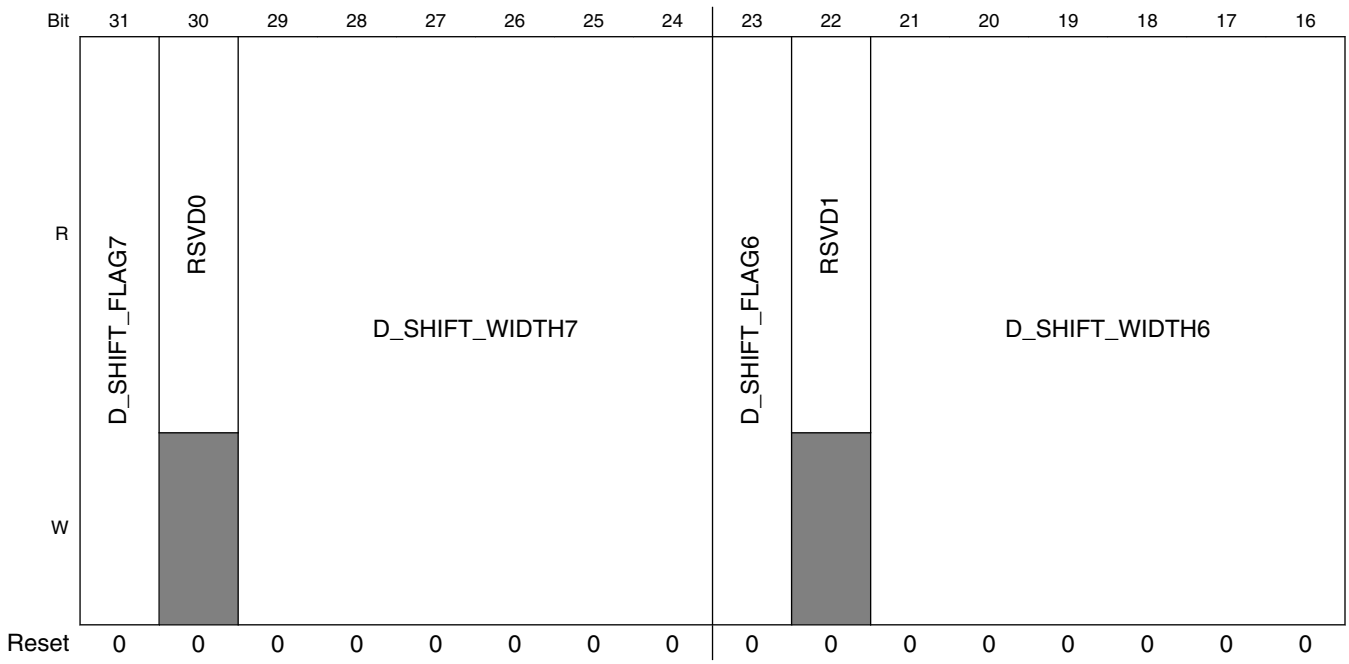
13.6.12.173 PXP_HW_PXP_DITHER_STORE_D_SHIFT_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + BF0h offset = 3070_0BF0h



PXP_HW_PXP_DITHER_STORE_D_SHIFT_H_CH0 field descriptions

Field	Description
31 D_SHIFT_FLAG7	data shift flag 7
30 RSVD0	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_DITHER_STORE_D_SHIFT_H_CH0 field descriptions (continued)

Field	Description
29–24 D_SHIFT_ WIDTH7	data shift width 3
23 D_SHIFT_FLAG6	data shift flag 6
22 RSVD1	Reserved, always set to zero.
21–16 D_SHIFT_ WIDTH6	data shift width 6
15 D_SHIFT_FLAG5	data shift flag 5
14 RSVD2	Reserved, always set to zero.
13–8 D_SHIFT_ WIDTH5	data shift width 5
7 D_SHIFT_FLAG4	data shift flag 4
6 RSVD3	Reserved, always set to zero.
D_SHIFT_ WIDTH4	data shift width 4

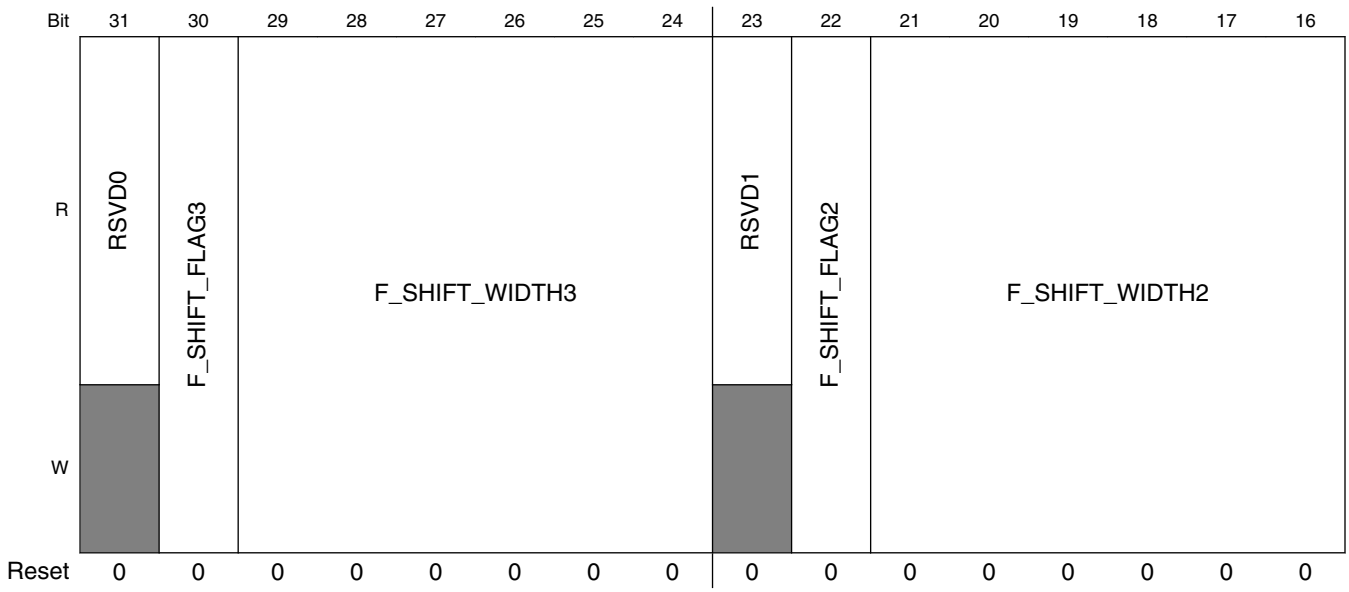
13.6.12.174 PXP_HW_PXP_DITHER_STORE_F_SHIFT_L_CH0

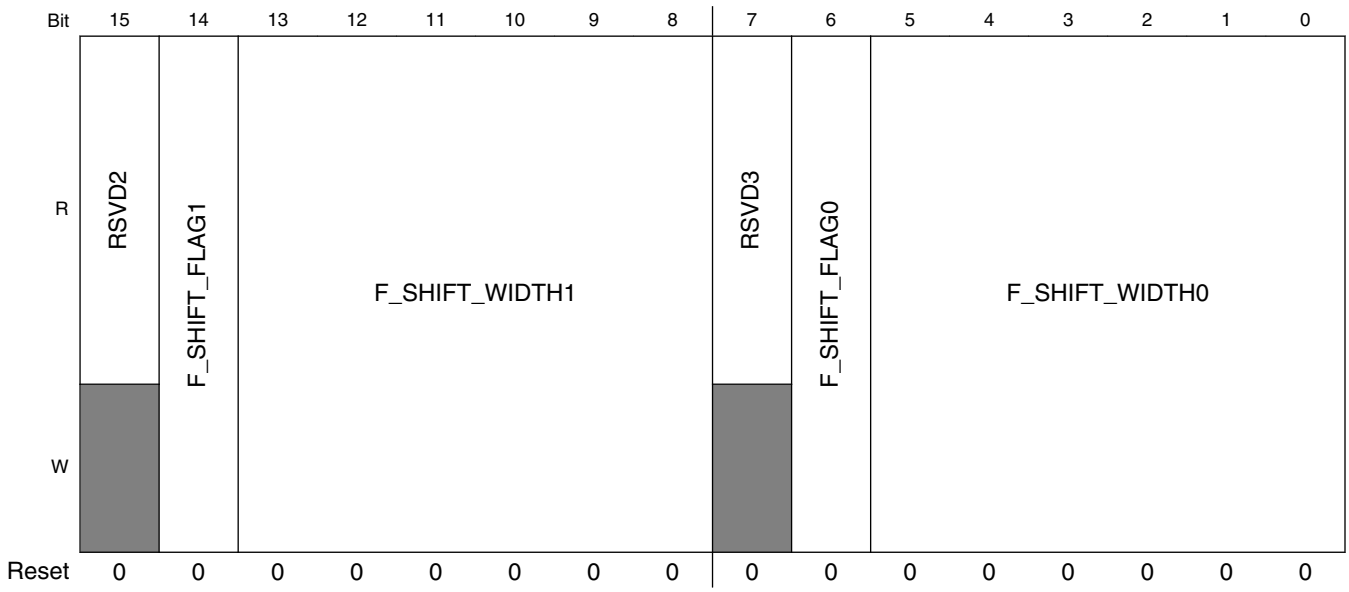
This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + C00h offset = 3070_0C00h





PXP_HW_PXP_DITHER_STORE_F_SHIFT_L_CH0 field descriptions

Field	Description
31 RSVD0	Reserved, always set to zero.
30 F_SHIFT_FLAG3	flag shift flag3
29–24 F_SHIFT_ WIDTH3	flag shift width 3
23 RSVD1	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_DITHER_STORE_F_SHIFT_L_CH0 field descriptions (continued)

Field	Description
22 F_SHIFT_FLAG2	flag shift flag2
21–16 F_SHIFT_ WIDTH2	flag shift width 2
15 RSVD2	Reserved, always set to zero.
14 F_SHIFT_FLAG1	flag shift flag1
13–8 F_SHIFT_ WIDTH1	flag shift width 1
7 RSVD3	Reserved, always set to zero.
6 F_SHIFT_FLAG0	flag shift flag0
F_SHIFT_ WIDTH0	flag shift width 0

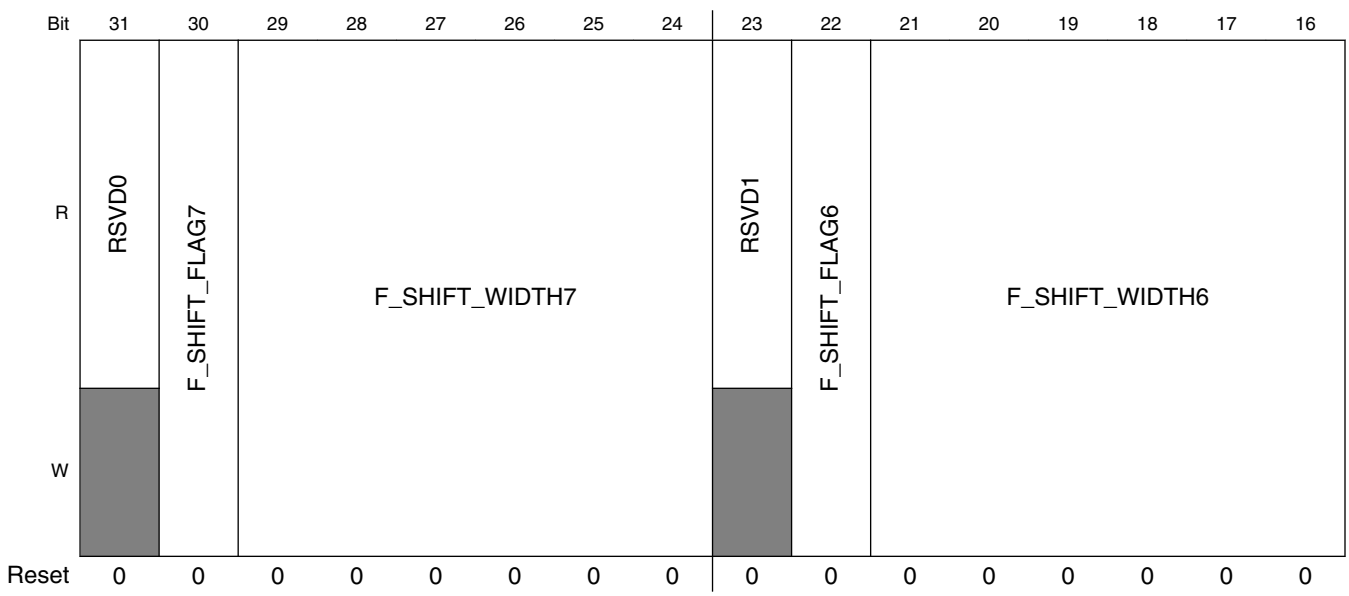
13.6.12.175 PXP_HW_PXP_DITHER_STORE_F_SHIFT_H_CH0

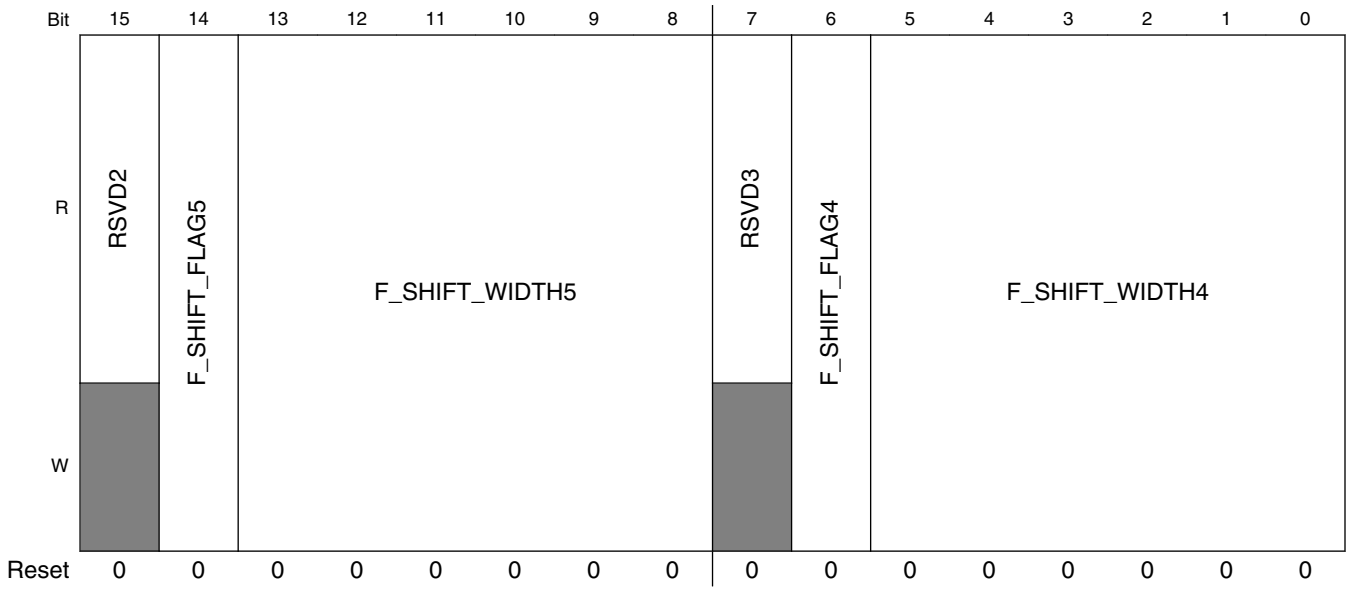
This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + C10h offset = 3070_0C10h





PXP_HW_PXP_DITHER_STORE_F_SHIFT_H_CH0 field descriptions

Field	Description
31 RSVD0	Reserved, always set to zero.
30 F_SHIFT_FLAG7	flag shift flag7
29–24 F_SHIFT_ WIDTH7	flag shift width 7
23 RSVD1	Reserved, always set to zero.
22 F_SHIFT_FLAG6	flag shift flag6
21–16 F_SHIFT_ WIDTH6	flag shift width 5
15 RSVD2	Reserved, always set to zero.
14 F_SHIFT_FLAG5	flag shift flag5
13–8 F_SHIFT_ WIDTH5	flag shift width 5
7 RSVD3	Reserved, always set to zero.
6 F_SHIFT_FLAG4	flag shift flag4
F_SHIFT_ WIDTH4	flag shift width 4

13.6.12.176 PXP_HW_PXP_DITHER_STORE_F_MASK_L_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + C20h offset = 3070_0C20h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	F_MASK3								F_MASK2								F_MASK1								F_MASK0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_F_MASK_L_CH0 field descriptions

Field	Description
31–24 F_MASK3	flag mask3
23–16 F_MASK2	flag mask2
15–8 F_MASK1	flag mask1
F_MASK0	flag mask0

13.6.12.177 PXP_HW_PXP_DITHER_STORE_F_MASK_H_CH0

This register defines the control bits for the pxp store_engine sub-block.

The Control register contains the control bits for the pxp store_engine sub-block.

EXAMPLE

Address: 3070_0000h base + C30h offset = 3070_0C30h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	F_MASK7								F_MASK6								F_MASK5								F_MASK4							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_STORE_F_MASK_H_CH0 field descriptions

Field	Description
31–24 F_MASK7	flag mask7

Table continues on the next page...

PXP_HW_PXP_DITHER_STORE_F_MASK_H_CH0 field descriptions (continued)

Field	Description
23–16 F_MASK6	flag mask6
15–8 F_MASK5	flag mask5
F_MASK4	flag mask4

13.6.12.178 Dither Control Register 0 (PXP_HW_PXP_DITHER_CTRL)

This register defines the control bits for the pxp dither sub-block.

HW_PXP_DITHER_CTRL: 0x1670

HW_PXP_DITHER_CTRL_SET: 0x1674

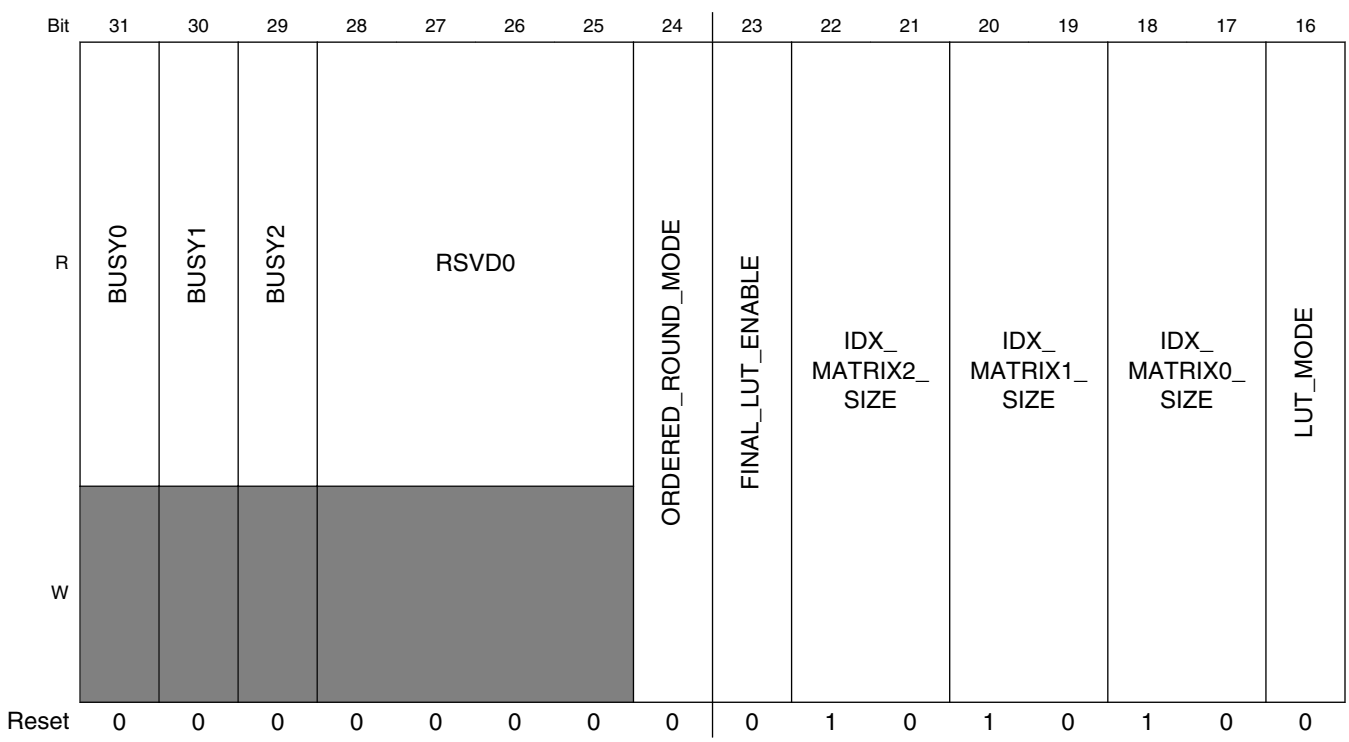
HW_PXP_DITHER_CTRL_CLR: 0x1678

HW_PXP_DITHER_CTRL_TOG: 0x167C

The Control register contains the primary controls for the PXP block.

EXAMPLE

Address: 3070_0000h base + 1670h offset = 3070_1670h



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LUT_MODE	NUM_QUANT_BIT			DITHER_MODE2			DITHER_MODE1			DITHER_MODE0			ENABLE2	ENABLE1	ENABLE0
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_CTRL field descriptions

Field	Description
31 BUSY0	When set indicates if the dither engine 0 is busy -- started but not finished processing all of the pixels in the current frame.
30 BUSY1	When set indicates if the dither engine 1 is busy -- started but not finished processing all of the pixels in the current frame.
29 BUSY2	When set indicates if the dither engine 2 is busy -- started but not finished processing all of the pixels in the current frame.
28–25 RSVD0	Reserved, always set to zero.
24 ORDERED_ ROUND_MODE	For test purposes. In ordered mode only, this field specifies to use rounding or truncation when calculating the ouptput pixel. 0x0 **0** — Use truncation method. 0x1 **1** — Use rounding method.
23 FINAL_LUT_ ENABLE	Enables a final stage register based LUT at the last stage before output. the lookup transform values come from register bits, not internal memory. Therefore they must be setup by the user by writing to the registers before processing. 0x0 **Disabled** — The dither engine 2 will not process any frames. 0x1 **Enabled** — The dither engine 2 is on and ready for processing
22–21 IDX_MATRIX2_ SIZE	For Dither Engine 2. Specify dimension (assumed square) of the index matrix in the LUT memory so proper indexing can occur. 0x0 **0** — 4x4 0x1 **1** — 8x8 0x2 **2** — 16x16 0x3 **3** — Input value of index
20–19 IDX_MATRIX1_ SIZE	For Dither Engine 1. Specify dimension (assumed square) of the index matrix in the LUT memory so proper indexing can occur. 0x0 **0** — 4x4 0x1 **1** — 8x8

Table continues on the next page...

PXP_HW_PXP_DITHER_CTRL field descriptions (continued)

Field	Description
	0x2 **2** — 16x16 0x3 **3** — Input value of index
18–17 IDX_MATRIX0_SIZE	For Dither Engine 0. Specify dimension (assumed square) of the index matrix in the LUT memory so proper indexing can occur. 0x0 **0** — 4x4 0x1 **1** — 8x8 0x2 **2** — 16x16 0x3 **3** — Input value of index
16–15 LUT_MODE	Specify to use memory lut to transform pixel. Lookup pre or post dithering cannot be used with Ordered dithering. This field only has reference to the internal memory based LUT function. There is a final stage LUT that is enabled through the FINAL_LUT_ENABLE field in this register. This final stage LUT can be enabled in any dither mode or LUT mode. 0x0 **0** — LUT mode off. 0x1 **1** — Use LUT at pre-diter stage. 0x2 **2** — Use LUT at post-dither stage. 0x3 **3** — Reserved`
14–12 NUM_QUANT_BIT	Number of bits to quantize down to. From 8 to (0-7). 0x0 **0** — Reserved. 0x1 **1** — Quantize down to 1 bit. 0x2 **2** — Quantize down to 2 bits. 0x3 **3** — Quantize down to 3 bits. 0x4 **4** — Quantize down to 4 bits. 0x5 **5** — Quantize down to 5 bits. 0x6 **6** — Quantize down to 6 bits. 0x7 **7** — Quantize down to 7 bits.
11–9 DITHER_MODE2	Dither mode. 0x0 **0** — Pass through. 0x1 **1** — Reserved. 0x2 **2** — Reserved. 0x3 **3** — Ordered. 0x4 **4** — No Dithering, quantization only. 0x5 **5** — Reserved. 0x6 **6** — Reserved. 0x7 **7** — Reserved.
8–6 DITHER_MODE1	Dither mode. 0x0 **0** — Pass through. 0x1 **1** — Reserved. 0x2 **2** — Reserved 0x3 **3** — Ordered. 0x4 **4** — No Dithering, quantization only. 0x5 **5** — Reserved. 0x6 **6** — Reserved. 0x7 **7** — Reserved.

Table continues on the next page...

PXP_HW_PXP_DITHER_CTRL field descriptions (continued)

Field	Description
5–3 DITHER_MODE0	Dither mode. 0x0 **0** — Pass through. 0x1 **1** — Floyd-Steinberg. 0x2 **2** — Atkinson. 0x3 **3** — Ordered. 0x4 **4** — No Dithering, quantization only. 0x5 **5** — Reserved. 0x6 **6** — Reserved. 0x7 **7** — Reserved.
2 ENABLE2	Enables the dither engine 2 0x0 **Disabled** — The dither engine 2 will not process any frames. 0x1 **Enabled** — The dither engine 2 is on and ready for processing
1 ENABLE1	Enables the dither engine 1 0x0 **Disabled** — The dither engine 1 will not process any frames. 0x1 **Enabled** — The dither engine 1 is on and ready for processing
0 ENABLE0	Enables the dither engine 0 0x0 **Disabled** — The dither engine 0 will not process any frames. 0x1 **Enabled** — The dither engine 0 is on and ready for processing

13.6.12.179 Final stage lookup value Register (PXP_HW_PXP_DITHER_FINAL_LUT_DATA0)

8 bit data values for the final stage dither LUT.

HW_PXP_DITHER_FINAL_LUT_DATA0: 0x1680

HW_PXP_DITHER_FINAL_LUT_DATA0_SET: 0x1684

HW_PXP_DITHER_FINAL_LUT_DATA0_CLR: 0x1688

HW_PXP_DITHER_FINAL_LUT_DATA0_TOG: 0x168C

This register contains lookup data values for the final stage register based dither LUT.

EXAMPLE

Address: 3070_0000h base + 1680h offset = 3070_1680h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA3								DATA2								DATA1								DATA0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FINAL_LUT_DATA0 field descriptions

Field	Description
31–24 DATA3	Final stage LUT data value.
23–16 DATA2	Final stage LUT data value.
15–8 DATA1	Final stage LUT data value.
DATA0	Final stage LUT data value.

13.6.12.180 Final stage lookup value Register (PXP_HW_PXP_DITHER_FINAL_LUT_DATA1)

8 bit data values for the final stage dither LUT.

HW_PXP_DITHER_FINAL_LUT_DATA1: 0x1690

HW_PXP_DITHER_FINAL_LUT_DATA1_SET: 0x1694

HW_PXP_DITHER_FINAL_LUT_DATA1_CLR: 0x1698

HW_PXP_DITHER_FINAL_LUT_DATA1_TOG: 0x169C

This register contains lookup data values for the final stage register based dither LUT.

EXAMPLE

Address: 3070_0000h base + 1690h offset = 3070_1690h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA7								DATA6								DATA5								DATA4							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FINAL_LUT_DATA1 field descriptions

Field	Description
31–24 DATA7	Final stage LUT data value.
23–16 DATA6	Final stage LUT data value.
15–8 DATA5	Final stage LUT data value.
DATA4	Final stage LUT data value.

13.6.12.181 Final stage lookup value Register (PXP_HW_PXP_DITHER_FINAL_LUT_DATA2)

8 bit data values for the final stage dither LUT.

HW_PXP_DITHER_FINAL_LUT_DATA2: 0x16A0

HW_PXP_DITHER_FINAL_LUT_DATA2_SET: 0x16A4

HW_PXP_DITHER_FINAL_LUT_DATA2_CLR: 0x16A8

HW_PXP_DITHER_FINAL_LUT_DATA2_TOG: 0x16AC

This register contains lookup data values for the final stage register based dither LUT.

EXAMPLE

Address: 3070_0000h base + 16A0h offset = 3070_16A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA11								DATA10								DATA9								DATA8							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_DITHER_FINAL_LUT_DATA2 field descriptions

Field	Description
31–24 DATA11	Final stage LUT data value.
23–16 DATA10	Final stage LUT data value.
15–8 DATA9	Final stage LUT data value.
DATA8	Final stage LUT data value.

13.6.12.182 Final stage lookup value Register (PXP_HW_PXP_DITHER_FINAL_LUT_DATA3)

8 bit data values for the final stage dither LUT.

HW_PXP_DITHER_FINAL_LUT_DATA3: 0x16B0

HW_PXP_DITHER_FINAL_LUT_DATA3_SET: 0x16B4

HW_PXP_DITHER_FINAL_LUT_DATA3_CLR: 0x16B8

HW_PXP_DITHER_FINAL_LUT_DATA3_TOG: 0x16BC

This register contains lookup data values for the final stage register based dither LUT.

EXAMPLE

Address: 3070_0000h base + 16B0h offset = 3070_16B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA15								DATA14								DATA13								DATA12							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

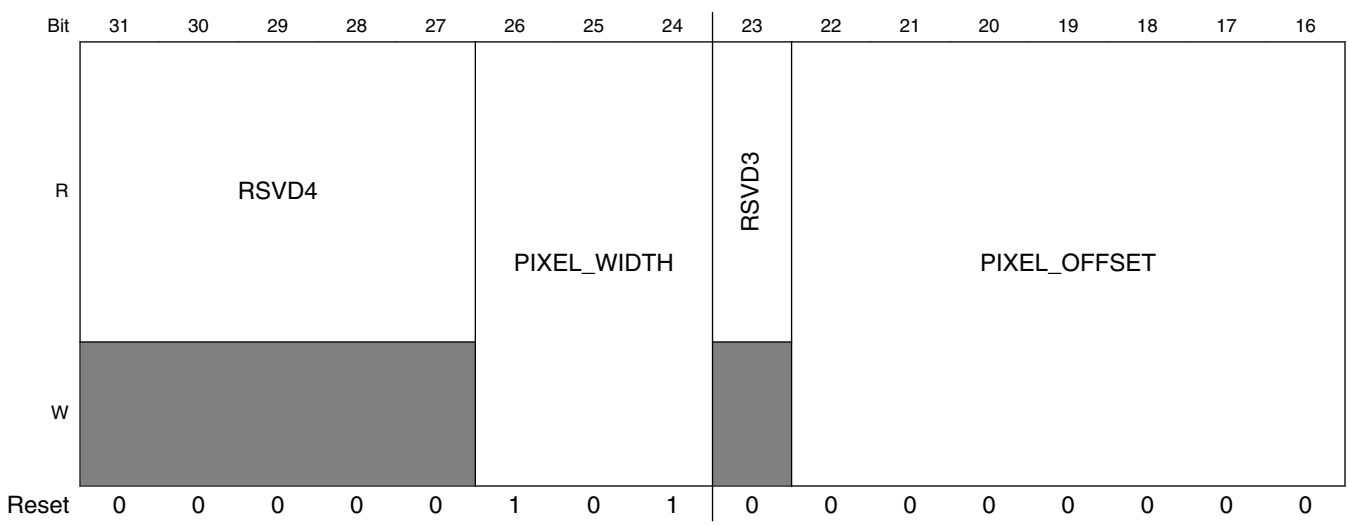
PXP_HW_PXP_DITHER_FINAL_LUT_DATA3 field descriptions

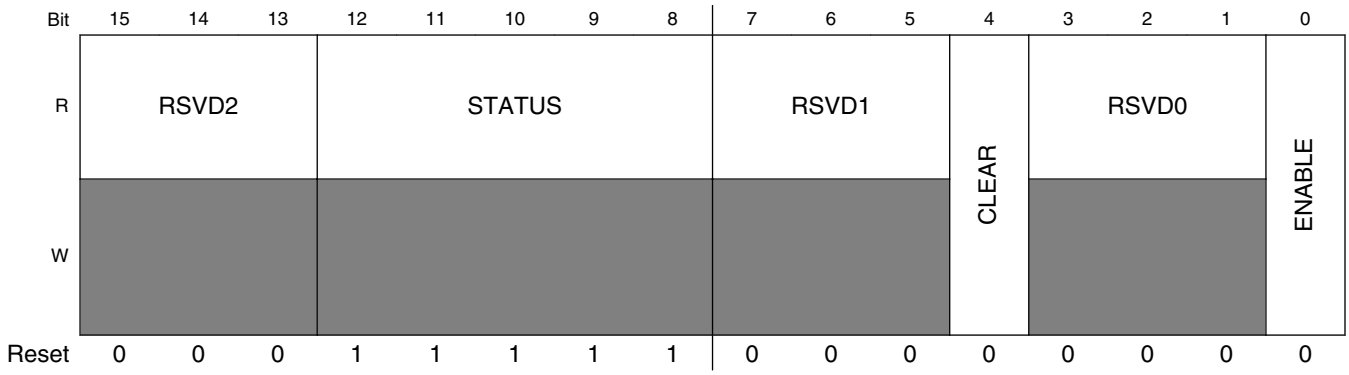
Field	Description
31–24 DATA15	Final stage LUT data value.
23–16 DATA14	Final stage LUT data value.
15–8 DATA13	Final stage LUT data value.
DATA12	Final stage LUT data value.

13.6.12.183 Histogram Control Register. (PXP_HW_PXP_HIST_A_CTRL)

Provides control and status registers for the PXP's histogram classification algorithm.

Address: 3070_0000h base + 2A00h offset = 3070_2A00h





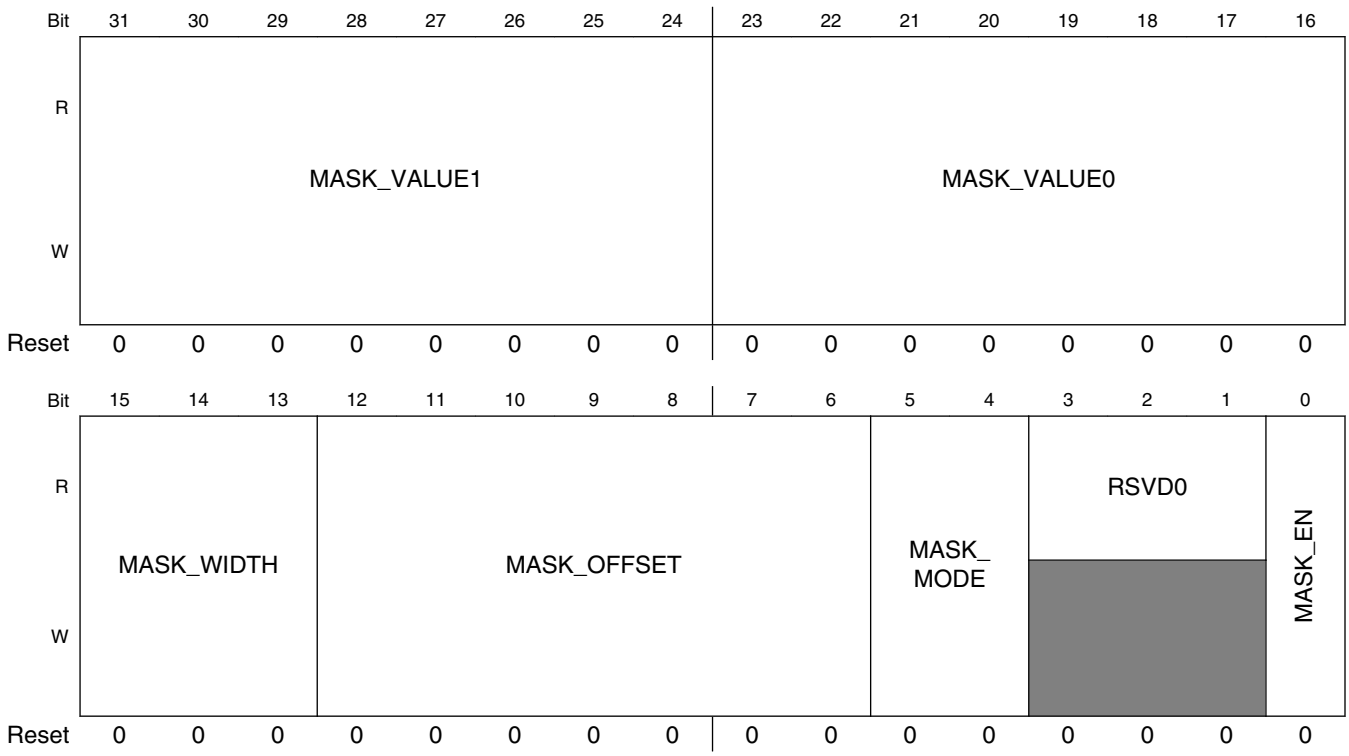
PXP_HW_PXP_HIST_A_CTRL field descriptions

Field	Description
31–27 RSVD4	Reserved, always set to zero.
26–24 PIXEL_WIDTH	The width of the pixel to be used for histogram calculation
23 RSVD3	Reserved, always set to zero.
22–16 PIXEL_OFFSET	The offset of the pixel to be used for histogram calculation
15–13 RSVD2	Reserved, always set to zero.
12–8 STATUS	Indicates which histogram matched the processed bitmap. Bit[0] indicates that the bitmap pixels were fully contained within the HIST2 (black / white) histogram. Bit[1] indicates that the bitmap pixels were fully contained within the HIST4 (2-bit grayscale) histogram. Bit[2] indicates that the bitmap pixels were fully contained within the HIST8 (3-bit grayscale) histogram. Bit[3] indicates that the bitmap pixels were fully contained within the HIST16 (4-bit grayscale) histogram. Bit[4] indicates that the bitmap pixels were fully contained within the HIST32 (5-bit grayscale) histogram.
7–5 RSVD1	Reserved, always set to zero.
4 CLEAR	Write 1 to clear the histogram result and will be self-clear after clear function finished
3–1 RSVD0	Reserved, always set to zero.
0 ENABLE	Enable the Histogram Engine

13.6.12.184 Histogram Pixel Mask Register. (PXP_HW_PXP_HIST_A_MASK)

Provides control registers for the pixel masking in PXP's histogram.

Address: 3070_0000h base + 2A10h offset = 3070_2A10h



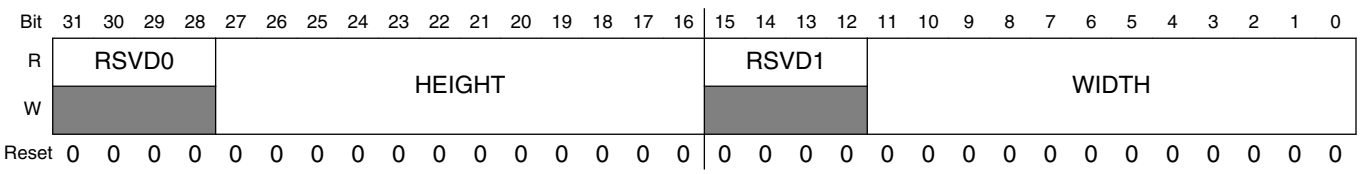
PXP_HW_PXP_HIST_A_MASK field descriptions

Field	Description
31–24 MASK_VALUE1	The value1 for mask condition checking
23–16 MASK_VALUE0	The value0 for mask condition checking
15–13 MASK_WIDTH	The width of the field to be checked against mask condition
12–6 MASK_OFFSET	The offset of the field to be checked against mask condition
5–4 MASK_MODE	Operation mode of pixel mask function 0x0 **EQUAL** — Run histogram for pixels equal to value0 0x1 **NOT_EQUAL** — Run histogram for pixels not equal to value0 0x2 **INSIDE** — Run histogram for pixels within the range of value0 to value1 0x3 **OUTSIDE** — Run histogram for pixels outside of the rang of value0 to value1
3–1 RSVD0	Reserved, always set to zero.
0 MASK_EN	Enable the Pixel Mask Function in Histogram

13.6.12.185 Histogram Pixel Buffer Size Register. (PXP_HW_PXP_HIST_A_BUF_SIZE)

This register defines the size of the buffer to be processed by the histogram engine.

Address: 3070_0000h base + 2A20h offset = 3070_2A20h



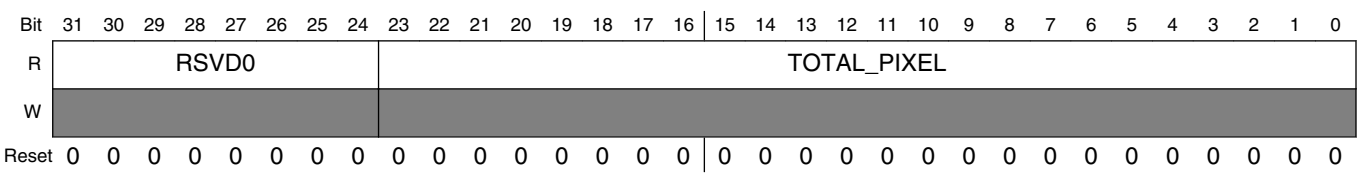
PXP_HW_PXP_HIST_A_BUF_SIZE field descriptions

Field	Description
31–28 RSVD0	Reserved. This field always reads 0.
27–16 HEIGHT	This indicate the buffer height in pixels
15–12 RSVD1	Reserved. This field always reads 0.
WIDTH	This indicate the buffer width in pixels

13.6.12.186 Total Number of Pixels Used by Histogram Engine. (PXP_HW_PXP_HIST_A_TOTAL_PIXEL)

This register shows the total number of pixels used by histogram engine

Address: 3070_0000h base + 2A30h offset = 3070_2A30h



PXP_HW_PXP_HIST_A_TOTAL_PIXEL field descriptions

Field	Description
31–24 RSVD0	Reserved. This field always reads 0.
TOTAL_PIXEL	Total number of pixels used by histogram engine, the pixels got masked will be skipped

13.6.12.187 The X Coordinate Offset for Active Area. (PXP_HW_PXP_HIST_A_ACTIVE_AREA_X)

This register shows the minimal and maximum X coordinate offset for the active area in histogram processing

Address: 3070_0000h base + 2A40h offset = 3070_2A40h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1				MAX_X_OFFSET												RSVD0				MIN_X_OFFSET												
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_HIST_A_ACTIVE_AREA_X field descriptions

Field	Description
31–28 RSVD1	Reserved. This field always reads 0.
27–16 MAX_X_OFFSET	Maximum X coordinate offset for the active area in histogram processing
15–12 RSVD0	Reserved. This field always reads 0.
MIN_X_OFFSET	Minimul X coordinate offset for the active area in histogram processing

13.6.12.188 The Y Coordinate Offset for Active Area. (PXP_HW_PXP_HIST_A_ACTIVE_AREA_Y)

This register shows the minimal and maximum Y coordinate offset for the active area in histogram processing

Address: 3070_0000h base + 2A50h offset = 3070_2A50h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1				MAX_Y_OFFSET												RSVD0				MIN_Y_OFFSET												
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_HIST_A_ACTIVE_AREA_Y field descriptions

Field	Description
31–28 RSVD1	Reserved. This field always reads 0.

Table continues on the next page...

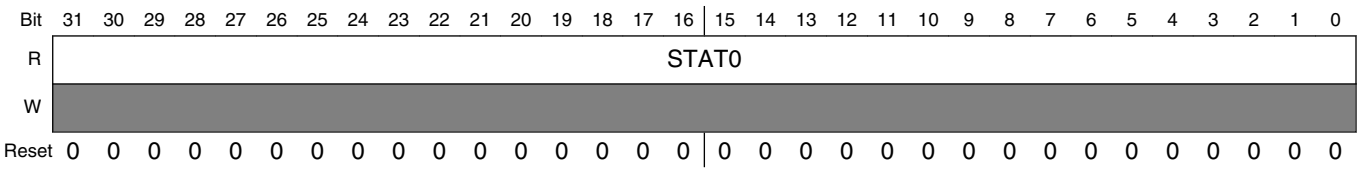
PXP_HW_PXP_HIST_A_ACTIVE_AREA_Y field descriptions (continued)

Field	Description
27–16 MAX_Y_OFFSET	Maximum Y coordinate offset for the active area in histogram processing
15–12 RSVD0	Reserved. This field always reads 0.
MIN_Y_OFFSET	Minimul Y coordinate offset for the active area in histogram processing

13.6.12.189 Histogram Result Based on RAW Pixel Value. (PXP_HW_PXP_HIST_A_RAW_STAT0)

This register shows the lower 32-bit of the histogram result based on raw pixel value

Address: 3070_0000h base + 2A60h offset = 3070_2A60h



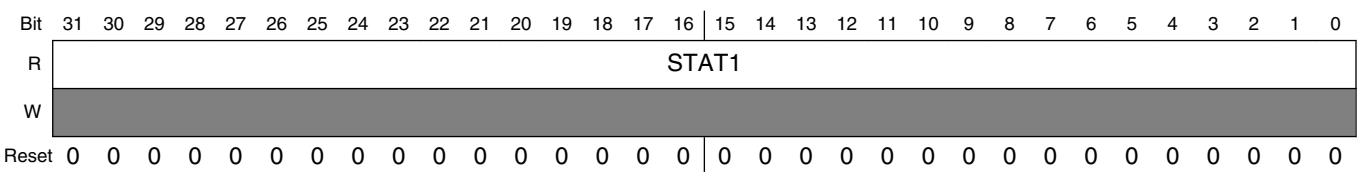
PXP_HW_PXP_HIST_A_RAW_STAT0 field descriptions

Field	Description
STAT0	Lower 32-bit result fo the histogram calculation

13.6.12.190 Histogram Result Based on RAW Pixel Value. (PXP_HW_PXP_HIST_A_RAW_STAT1)

This register shows the higher 32-bit of the histogram result based on raw pixel value

Address: 3070_0000h base + 2A70h offset = 3070_2A70h



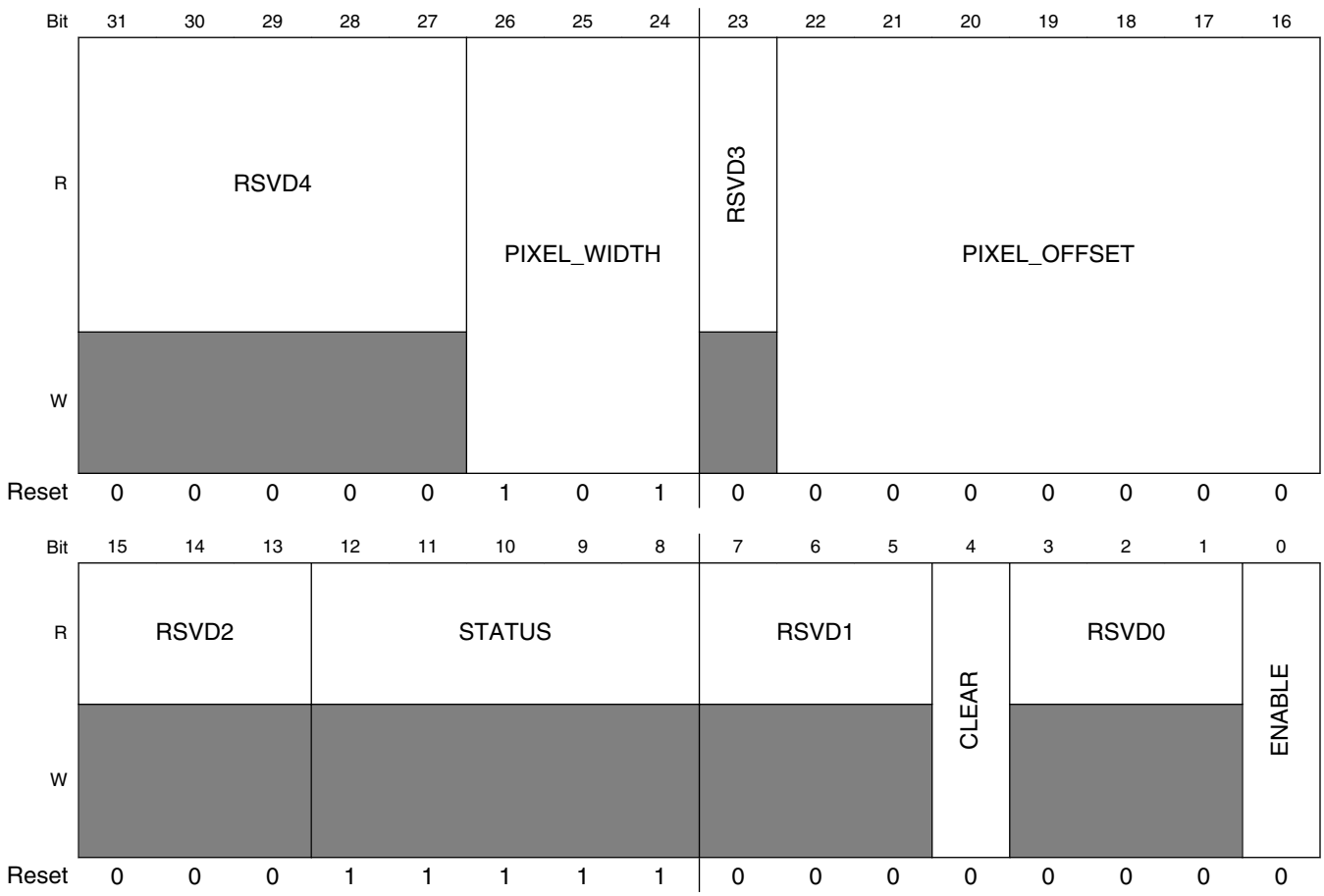
PXP_HW_PXP_HIST_A_RAW_STAT1 field descriptions

Field	Description
STAT1	Higher 32-bit result fo the histogram calculation

13.6.12.191 Histogram Control Register. (PXP_HW_PXP_HIST_B_CTRL)

Provides control and status registers for the PXP's histogram classification algorithm.

Address: 3070_0000h base + 2A80h offset = 3070_2A80h



PXP_HW_PXP_HIST_B_CTRL field descriptions

Field	Description
31–27 RSVD4	Reserved, always set to zero.
26–24 PIXEL_WIDTH	The width of the pixel to be used for histogram calculation
23 RSVD3	Reserved, always set to zero.
22–16 PIXEL_OFFSET	The offset of the pixel to be used for histogram calculation
15–13 RSVD2	Reserved, always set to zero.

Table continues on the next page...

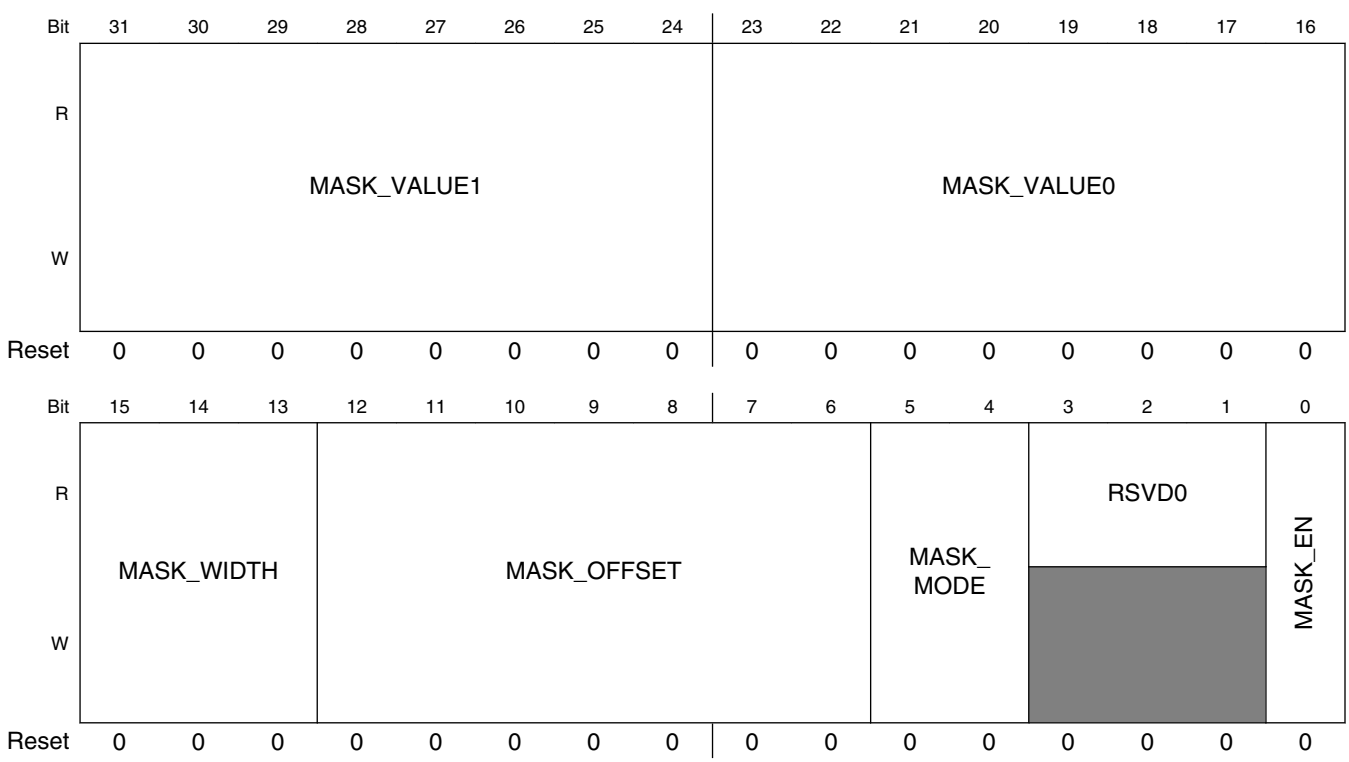
PXP_HW_PXP_HIST_B_CTRL field descriptions (continued)

Field	Description
12–8 STATUS	Indicates which histogram matched the processed bitmap. Bit[0] indicates that the bitmap pixels were fully contained within the HIST2 (black / white) histogram. Bit[1] indicates that the bitmap pixels were fully contained within the HIST4 (2-bit grayscale) histogram. Bit[2] indicates that the bitmap pixels were fully contained within the HIST8 (3-bit grayscale) histogram. Bit[3] indicates that the bitmap pixels were fully contained within the HIST16 (4-bit grayscale) histogram. Bit[4] indicates that the bitmap pixels were fully contained within the HIST32 (5-bit grayscale) histogram.
7–5 RSVD1	Reserved, always set to zero.
4 CLEAR	Write 1 to clear the histogram result and will be self-clear after clear function finished
3–1 RSVD0	Reserved, always set to zero.
0 ENABLE	Enable the Histogram Engine

13.6.12.192 Histogram Pixel Mask Register. (PXP_HW_PXP_HIST_B_MASK)

Provides control registers for the pixel masking in PXP's histogram.

Address: 3070_0000h base + 2A90h offset = 3070_2A90h



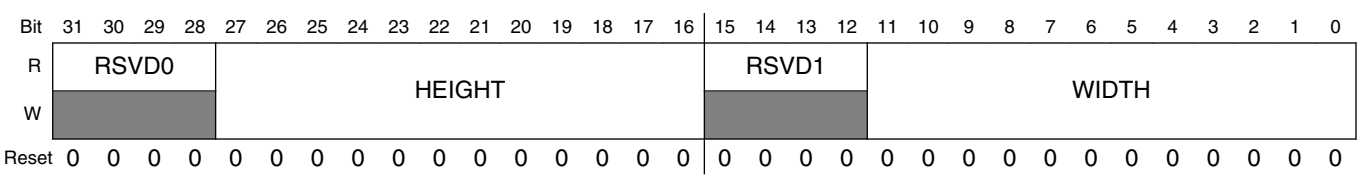
PXP_HW_PXP_HIST_B_MASK field descriptions

Field	Description
31–24 MASK_VALUE1	The value1 for mask condition checking
23–16 MASK_VALUE0	The value0 for mask condition checking
15–13 MASK_WIDTH	The width of the field to be checked against mask condition
12–6 MASK_OFFSET	The offset of the field to be checked against mask condition
5–4 MASK_MODE	Operation mode of pixel mask function 0x0 **EQUAL** — Run histogram for pixels equal to value0 0x1 **NOT_EQUAL** — Run histogram for pixels not equal to value0 0x2 **INSIDE** — Run histogram for pixels within the range of value0 to value1 0x3 **OUTSIDE** — Run histogram for pixels outside of the rang of value0 to value1
3–1 RSVD0	Reserved, always set to zero.
0 MASK_EN	Enable the Pixel Mask Function in Histogram

13.6.12.193 Histogram Pixel Buffer Size Register. (PXP_HW_PXP_HIST_B_BUF_SIZE)

This register defines the size of the buffer to be processed by the histogram engine.

Address: 3070_0000h base + 2AA0h offset = 3070_2AA0h



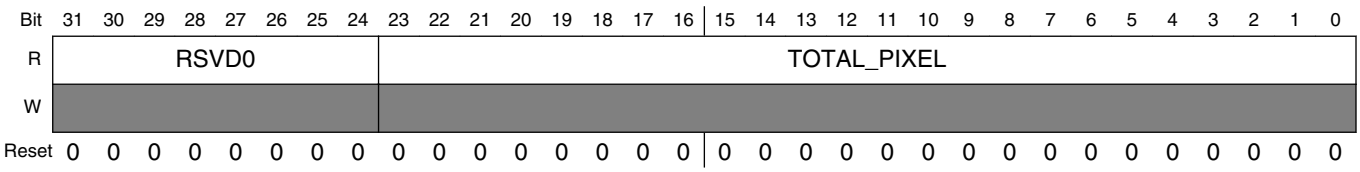
PXP_HW_PXP_HIST_B_BUF_SIZE field descriptions

Field	Description
31–28 RSVD0	Reserved. This field always reads 0.
27–16 HEIGHT	This indicate the buffer height in pixels
15–12 RSVD1	Reserved. This field always reads 0.
WIDTH	This indicate the buffer width in pixels

13.6.12.194 Total Number of Pixels Used by Histogram Engine. (PXP_HW_PXP_HIST_B_TOTAL_PIXEL)

This register shows the total number of pixels used by histogram engine

Address: 3070_0000h base + 2AB0h offset = 3070_2AB0h



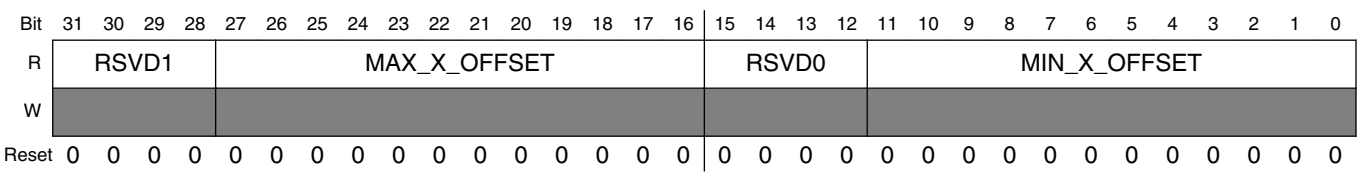
PXP_HW_PXP_HIST_B_TOTAL_PIXEL field descriptions

Field	Description
31–24 RSVD0	Reserved. This field always reads 0.
TOTAL_PIXEL	Total number of pixels used by histogram engine, the pixels got masked will be skipped

13.6.12.195 The X Coordinate Offset for Active Area. (PXP_HW_PXP_HIST_B_ACTIVE_AREA_X)

This register shows the minimal and maximum X coordinate offset for the active area in histogram processing

Address: 3070_0000h base + 2AC0h offset = 3070_2AC0h



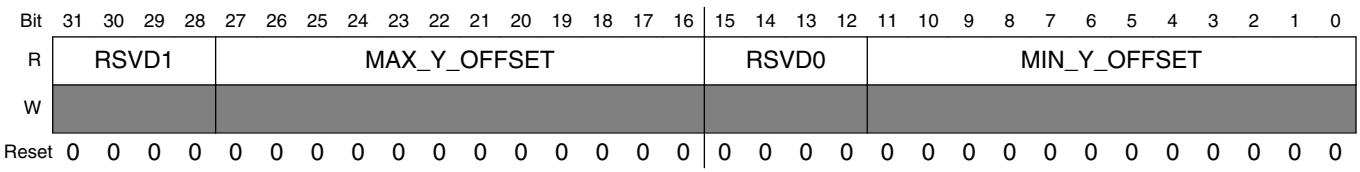
PXP_HW_PXP_HIST_B_ACTIVE_AREA_X field descriptions

Field	Description
31–28 RSVD1	Reserved. This field always reads 0.
27–16 MAX_X_OFFSET	Maximum X coordinate offset for the active area in histogram processing
15–12 RSVD0	Reserved. This field always reads 0.
MIN_X_OFFSET	Minimul X coordinate offset for the active area in histogram processing

13.6.12.196 The Y Coordinate Offset for Active Area. (PXP_HW_PXP_HIST_B_ACTIVE_AREA_Y)

This register shows the minimal and maximum Y coordinate offset for the active area in histogram processing

Address: 3070_0000h base + 2AD0h offset = 3070_2AD0h



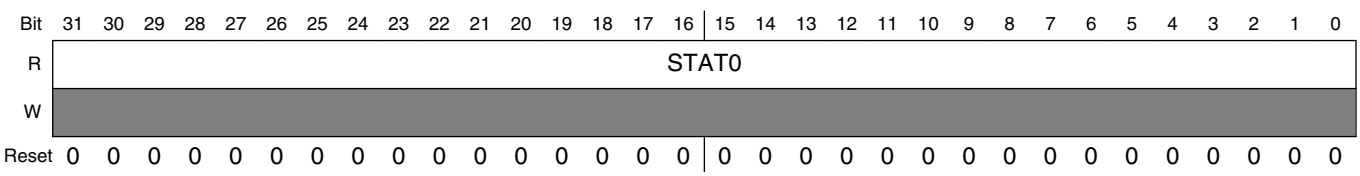
PXP_HW_PXP_HIST_B_ACTIVE_AREA_Y field descriptions

Field	Description
31–28 RSVD1	Reserved. This field always reads 0.
27–16 MAX_Y_OFFSET	Maximum Y coordinate offset for the active area in histogram processing
15–12 RSVD0	Reserved. This field always reads 0.
MIN_Y_OFFSET	Minimul Y coordinate offset for the active area in histogram processing

13.6.12.197 Histogram Result Based on RAW Pixel Value. (PXP_HW_PXP_HIST_B_RAW_STAT0)

This register shows the lower 32-bit of the histogram result based on raw pixel value

Address: 3070_0000h base + 2AE0h offset = 3070_2AE0h



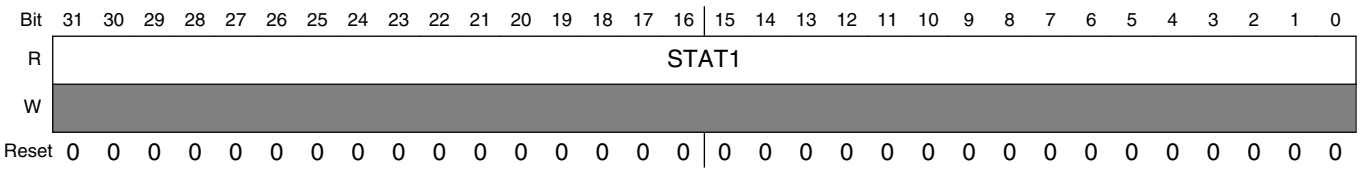
PXP_HW_PXP_HIST_B_RAW_STAT0 field descriptions

Field	Description
STAT0	Lower 32-bit result fo the histogram calculation

13.6.12.198 Histogram Result Based on RAW Pixel Value. (PXP_HW_PXP_HIST_B_RAW_STAT1)

This register shows the higher 32-bit of the histogram result based on raw pixel value

Address: 3070_0000h base + 2AF0h offset = 3070_2AF0h



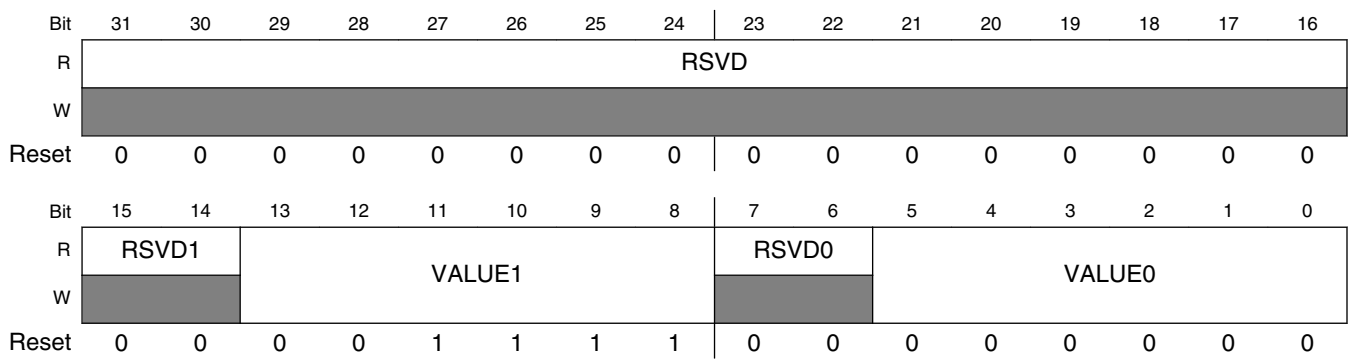
PXP_HW_PXP_HIST_B_RAW_STAT1 field descriptions

Field	Description
STAT1	Higher 32-bit result fo the histogram calculation

13.6.12.199 2-level Histogram Parameter Register. (PXP_HW_PXP_HIST2_PARAM)

This register specifies the valid values for a 2-level histogram. If all pixels in a bitmap match the 2-level histogram values, STATUS[0] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2B00h offset = 3070_2B00h



PXP_HW_PXP_HIST2_PARAM field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_HIST2_PARAM field descriptions (continued)

Field	Description
15–14 RSVD1	Reserved, always set to zero.
13–8 VALUE1	White value for 2-level histogram
7–6 RSVD0	Reserved, always set to zero.
VALUE0	Black value for 2-level histogram

13.6.12.200 4-level Histogram Parameter Register. (PXP_HW_PXP_HIST4_PARAM)

This register specifies the valid values for a 4-level histogram. If all pixels in a bitmap match the 4-level histogram values, STATUS[1] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2B10h offset = 3070_2B10h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3		VALUE3						RSVD2		VALUE2					
W																
Reset	0	0	0	0	1	1	1	1	0	0	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1		VALUE1						RSVD0		VALUE0					
W																
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

PXP_HW_PXP_HIST4_PARAM field descriptions

Field	Description
31–30 RSVD3	Reserved, always set to zero.
29–24 VALUE3	GRAY3 (White) value for 4-level histogram
23–22 RSVD2	Reserved, always set to zero.
21–16 VALUE2	GRAY2 value for 4-level histogram
15–14 RSVD1	Reserved, always set to zero.

Table continues on the next page...

PXP_HW_PXP_HIST4_PARAM field descriptions (continued)

Field	Description
13–8 VALUE1	GRAY1 value for 4-level histogram
7–6 RSVD0	Reserved, always set to zero.
VALUE0	GRAY0 (Black) value for 4-level histogram

13.6.12.201 8-level Histogram Parameter 0 Register. (PXP_HW_PXP_HIST8_PARAM0)

This register specifies four of the valid values for an 8-level histogram. If all pixels in a bitmap match the 8-level histogram values, STATUS[2] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2B20h offset = 3070_2B20h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3		VALUE3						RSVD2		VALUE2					
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1		VALUE1						RSVD0		VALUE0					
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_HIST8_PARAM0 field descriptions

Field	Description
31–30 RSVD3	Reserved, always set to zero.
29–24 VALUE3	GRAY3 value for 8-level histogram
23–22 RSVD2	Reserved, always set to zero.
21–16 VALUE2	GRAY2 value for 8-level histogram
15–14 RSVD1	Reserved, always set to zero.
13–8 VALUE1	GRAY1 value for 8-level histogram

Table continues on the next page...

PXP_HW_PXP_HIST8_PARAM0 field descriptions (continued)

Field	Description
7–6 RSVD0	Reserved, always set to zero.
VALUE0	GRAY0 (Black) value for 8-level histogram

13.6.12.202 8-level Histogram Parameter 1 Register. (PXP_HW_PXP_HIST8_PARAM1)

This register specifies four of the valid values for an 8-level histogram. If all pixels in a bitmap match the 8-level histogram values, STATUS[2] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2B30h offset = 3070_2B30h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD7		VALUE7						RSVD6		VALUE6					
W																
Reset	0	0	0	0	1	1	1	1	0	0	0	0	1	1	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD5		VALUE5						RSVD4		VALUE4					
W																
Reset	0	0	0	0	1	0	1	1	0	0	0	0	1	0	0	1

PXP_HW_PXP_HIST8_PARAM1 field descriptions

Field	Description
31–30 RSVD7	Reserved, always set to zero.
29–24 VALUE7	GRAY7 (White) value for 8-level histogram
23–22 RSVD6	Reserved, always set to zero.
21–16 VALUE6	GRAY6 value for 8-level histogram
15–14 RSVD5	Reserved, always set to zero.
13–8 VALUE5	GRAY5 value for 8-level histogram
7–6 RSVD4	Reserved, always set to zero.
VALUE4	GRAY4 value for 8-level histogram

13.6.12.203 16-level Histogram Parameter 0 Register. (PXP_HW_PXP_HIST16_PARAM0)

This register specifies four of the valid values for a 16-level histogram. If all pixels in a bitmap match the 16-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2B40h offset = 3070_2B40h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3		VALUE3						RSVD2		VALUE2					
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1		VALUE1						RSVD0		VALUE0					
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

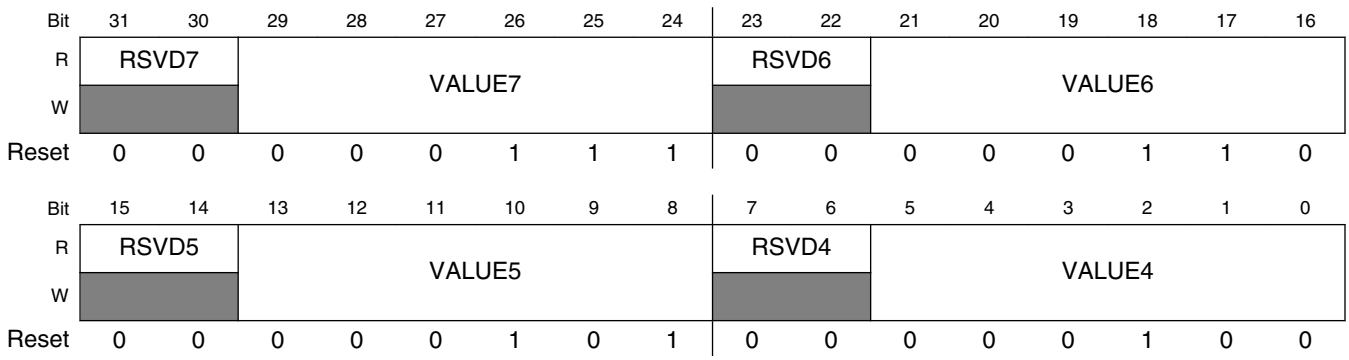
PXP_HW_PXP_HIST16_PARAM0 field descriptions

Field	Description
31–30 RSVD3	Reserved, always set to zero.
29–24 VALUE3	GRAY3 value for 16-level histogram
23–22 RSVD2	Reserved, always set to zero.
21–16 VALUE2	GRAY2 value for 16-level histogram
15–14 RSVD1	Reserved, always set to zero.
13–8 VALUE1	GRAY1 value for 16-level histogram
7–6 RSVD0	Reserved, always set to zero.
VALUE0	GRAY0 (Black) value for 16-level histogram

13.6.12.204 16-level Histogram Parameter 1 Register. (PXP_HW_PXP_HIST16_PARAM1)

This register specifies four of the valid values for a 16-level histogram. If all pixels in a bitmap match the 16-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2B50h offset = 3070_2B50h



PXP_HW_PXP_HIST16_PARAM1 field descriptions

Field	Description
31–30 RSVD7	Reserved, always set to zero.
29–24 VALUE7	GRAY7 value for 16-level histogram
23–22 RSVD6	Reserved, always set to zero.
21–16 VALUE6	GRAY6 value for 16-level histogram
15–14 RSVD5	Reserved, always set to zero.
13–8 VALUE5	GRAY5 value for 16-level histogram
7–6 RSVD4	Reserved, always set to zero.
VALUE4	GRAY4 value for 16-level histogram

13.6.12.205 16-level Histogram Parameter 2 Register. (PXP_HW_PXP_HIST16_PARAM2)

This register specifies four of the valid values for a 16-level histogram. If all pixels in a bitmap match the 16-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2B60h offset = 3070_2B60h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD11		VALUE11						RSVD10		VALUE10					
W																
Reset	0	0	0	0	1	0	1	1	0	0	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD9		VALUE9						RSVD8		VALUE8					
W																
Reset	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0

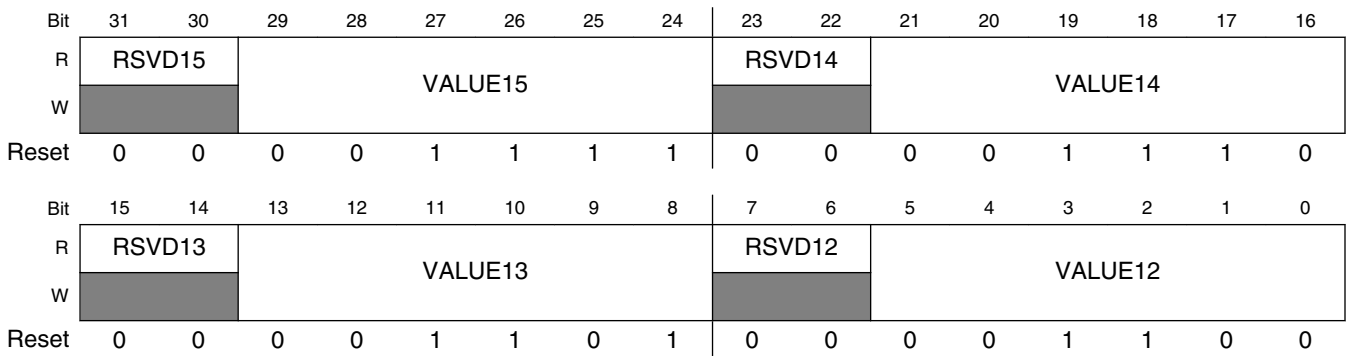
PXP_HW_PXP_HIST16_PARAM2 field descriptions

Field	Description
31–30 RSVD11	Reserved, always set to zero.
29–24 VALUE11	GRAY11 value for 16-level histogram
23–22 RSVD10	Reserved, always set to zero.
21–16 VALUE10	GRAY10 value for 16-level histogram
15–14 RSVD9	Reserved, always set to zero.
13–8 VALUE9	GRAY9 value for 16-level histogram
7–6 RSVD8	Reserved, always set to zero.
VALUE8	GRAY8 value for 16-level histogram

13.6.12.206 16-level Histogram Parameter 3 Register. (PXP_HW_PXP_HIST16_PARAM3)

This register specifies four of the valid values for a 16-level histogram. If all pixels in a bitmap match the 16-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2B70h offset = 3070_2B70h



PXP_HW_PXP_HIST16_PARAM3 field descriptions

Field	Description
31–30 RSVD15	Reserved, always set to zero.
29–24 VALUE15	GRAY15 (White) value for 16-level histogram
23–22 RSVD14	Reserved, always set to zero.
21–16 VALUE14	GRAY14 value for 16-level histogram
15–14 RSVD13	Reserved, always set to zero.
13–8 VALUE13	GRAY13 value for 16-level histogram
7–6 RSVD12	Reserved, always set to zero.
VALUE12	GRAY12 value for 16-level histogram

13.6.12.207 32-level Histogram Parameter 0 Register. (PXP_HW_PXP_HIST32_PARAM0)

This register specifies four of the valid values for a 32-level histogram. If all pixels in a bitmap match the 32-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2B80h offset = 3070_2B80h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3		VALUE3						RSVD2		VALUE2					
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1		VALUE1						RSVD0		VALUE0					
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

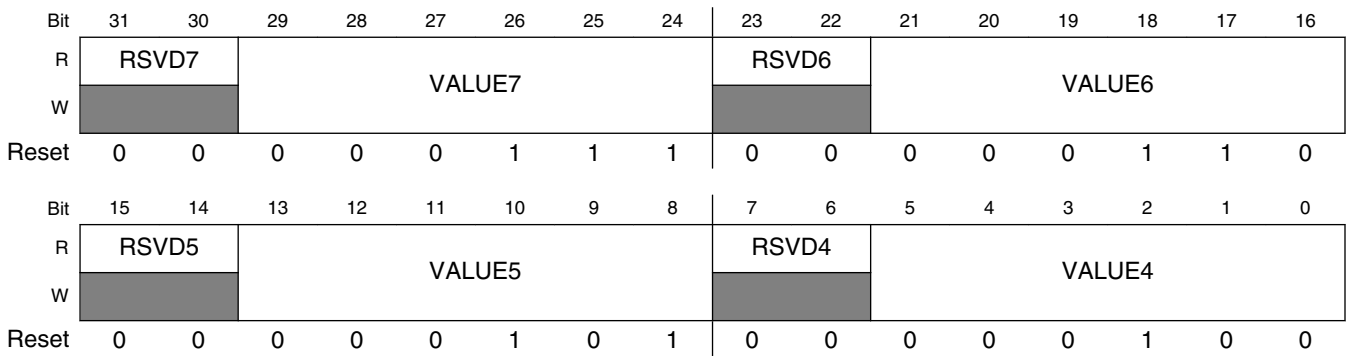
PXP_HW_PXP_HIST32_PARAM0 field descriptions

Field	Description
31–30 RSVD3	Reserved, always set to zero.
29–24 VALUE3	GRAY3 value for 32-level histogram
23–22 RSVD2	Reserved, always set to zero.
21–16 VALUE2	GRAY2 value for 32-level histogram
15–14 RSVD1	Reserved, always set to zero.
13–8 VALUE1	GRAY1 value for 32-level histogram
7–6 RSVD0	Reserved, always set to zero.
VALUE0	GRAY0 (Black) value for 32-level histogram

13.6.12.208 32-level Histogram Parameter 1 Register. (PXP_HW_PXP_HIST32_PARAM1)

This register specifies four of the valid values for a 32-level histogram. If all pixels in a bitmap match the 32-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2B90h offset = 3070_2B90h



PXP_HW_PXP_HIST32_PARAM1 field descriptions

Field	Description
31–30 RSVD7	Reserved, always set to zero.
29–24 VALUE7	GRAY7 value for 32-level histogram
23–22 RSVD6	Reserved, always set to zero.
21–16 VALUE6	GRAY6 value for 32-level histogram
15–14 RSVD5	Reserved, always set to zero.
13–8 VALUE5	GRAY5 value for 32-level histogram
7–6 RSVD4	Reserved, always set to zero.
VALUE4	GRAY4 value for 32-level histogram

13.6.12.209 32-level Histogram Parameter 2 Register. (PXP_HW_PXP_HIST32_PARAM2)

This register specifies four of the valid values for a 32-level histogram. If all pixels in a bitmap match the 32-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2BA0h offset = 3070_2BA0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD11		VALUE11						RSVD10		VALUE10					
W																
Reset	0	0	0	0	1	0	1	1	0	0	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD9		VALUE9						RSVD8		VALUE8					
W																
Reset	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0

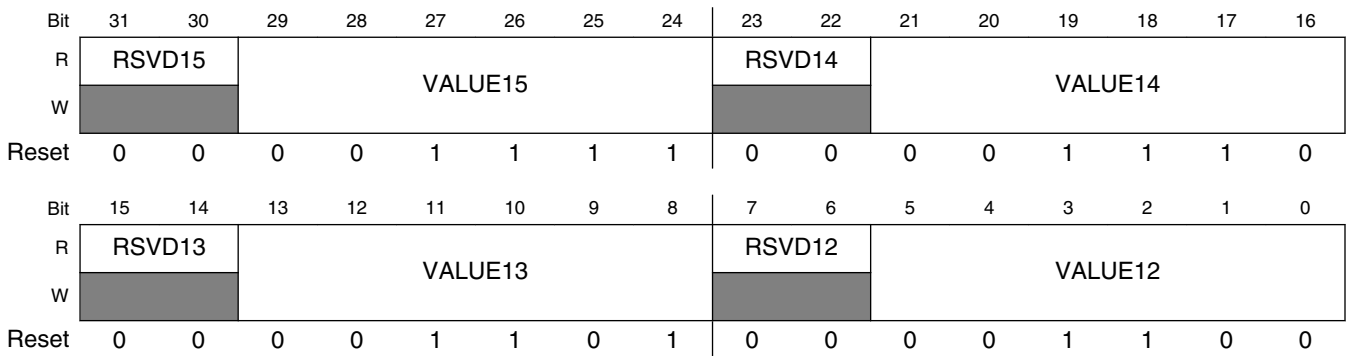
PXP_HW_PXP_HIST32_PARAM2 field descriptions

Field	Description
31–30 RSVD11	Reserved, always set to zero.
29–24 VALUE11	GRAY11 value for 32-level histogram
23–22 RSVD10	Reserved, always set to zero.
21–16 VALUE10	GRAY10 value for 32-level histogram
15–14 RSVD9	Reserved, always set to zero.
13–8 VALUE9	GRAY9 value for 32-level histogram
7–6 RSVD8	Reserved, always set to zero.
VALUE8	GRAY8 value for 32-level histogram

13.6.12.210 32-level Histogram Parameter 3 Register. (PXP_HW_PXP_HIST32_PARAM3)

This register specifies four of the valid values for a 32-level histogram. If all pixels in a bitmap match the 32-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2BB0h offset = 3070_2BB0h



PXP_HW_PXP_HIST32_PARAM3 field descriptions

Field	Description
31–30 RSVD15	Reserved, always set to zero.
29–24 VALUE15	GRAY15 (White) value for 32-level histogram
23–22 RSVD14	Reserved, always set to zero.
21–16 VALUE14	GRAY14 value for 32-level histogram
15–14 RSVD13	Reserved, always set to zero.
13–8 VALUE13	GRAY13 value for 32-level histogram
7–6 RSVD12	Reserved, always set to zero.
VALUE12	GRAY12 value for 32-level histogram

13.6.12.211 32-level Histogram Parameter 0 Register. (PXP_HW_PXP_HIST32_PARAM4)

This register specifies four of the valid values for a 32-level histogram. If all pixels in a bitmap match the 32-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2BC0h offset = 3070_2BC0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3		VALUE19						RSVD2		VALUE18					
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1		VALUE17						RSVD0		VALUE16					
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

PXP_HW_PXP_HIST32_PARAM4 field descriptions

Field	Description
31–30 RSVD3	Reserved, always set to zero.
29–24 VALUE19	GRAY19 value for 32-level histogram
23–22 RSVD2	Reserved, always set to zero.
21–16 VALUE18	GRAY18 value for 32-level histogram
15–14 RSVD1	Reserved, always set to zero.
13–8 VALUE17	GRAY17 value for 32-level histogram
7–6 RSVD0	Reserved, always set to zero.
VALUE16	GRAY16 (Black) value for 32-level histogram

13.6.12.212 32-level Histogram Parameter 1 Register. (PXP_HW_PXP_HIST32_PARAM5)

This register specifies four of the valid values for a 32-level histogram. If all pixels in a bitmap match the 32-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2BD0h offset = 3070_2BD0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD7		VALUE23						RSVD6		VALUE22					
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD5		VALUE21						RSVD4		VALUE20					
W																
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0

PXP_HW_PXP_HIST32_PARAM5 field descriptions

Field	Description
31–30 RSVD7	Reserved, always set to zero.
29–24 VALUE23	GRAY23 value for 32-level histogram
23–22 RSVD6	Reserved, always set to zero.
21–16 VALUE22	GRAY22 value for 32-level histogram
15–14 RSVD5	Reserved, always set to zero.
13–8 VALUE21	GRAY21 value for 32-level histogram
7–6 RSVD4	Reserved, always set to zero.
VALUE20	GRAY20 value for 32-level histogram

13.6.12.213 32-level Histogram Parameter 2 Register. (PXP_HW_PXP_HIST32_PARAM6)

This register specifies four of the valid values for a 32-level histogram. If all pixels in a bitmap match the 32-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2BE0h offset = 3070_2BE0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD11		VALUE27						RSVD10		VALUE26					
W																
Reset	0	0	0	0	1	0	1	1	0	0	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD9		VALUE25						RSVD8		VALUE24					
W																
Reset	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0

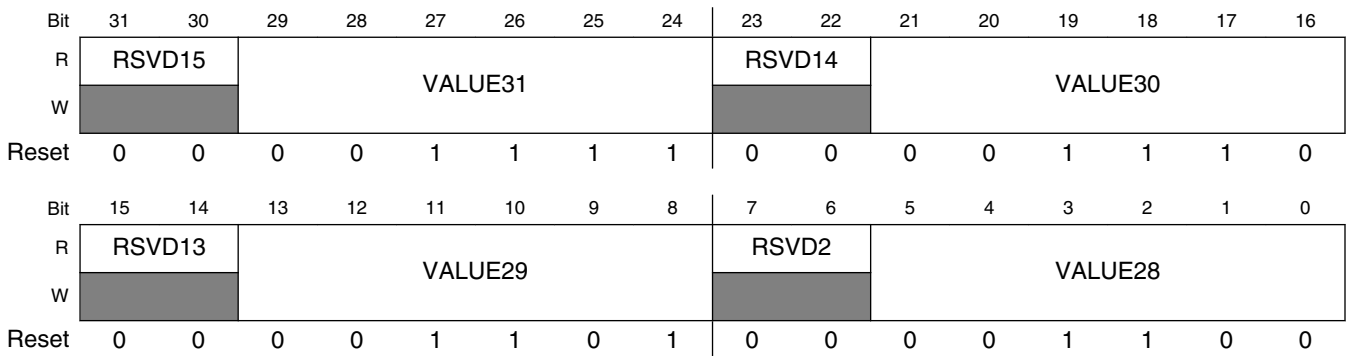
PXP_HW_PXP_HIST32_PARAM6 field descriptions

Field	Description
31–30 RSVD11	Reserved, always set to zero.
29–24 VALUE27	GRAY27 value for 32-level histogram
23–22 RSVD10	Reserved, always set to zero.
21–16 VALUE26	GRAY26 value for 32-level histogram
15–14 RSVD9	Reserved, always set to zero.
13–8 VALUE25	GRAY25 value for 32-level histogram
7–6 RSVD8	Reserved, always set to zero.
VALUE24	GRAY24 value for 32-level histogram

13.6.12.214 32-level Histogram Parameter 3 Register. (PXP_HW_PXP_HIST32_PARAM7)

This register specifies four of the valid values for a 32-level histogram. If all pixels in a bitmap match the 32-level histogram values, STATUS[3] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with the PANEL_MODE control field.

Address: 3070_0000h base + 2BF0h offset = 3070_2BF0h



PXP_HW_PXP_HIST32_PARAM7 field descriptions

Field	Description
31–30 RSVD15	Reserved, always set to zero.
29–24 VALUE31	GRAY31 (White) value for 32-level histogram
23–22 RSVD14	Reserved, always set to zero.
21–16 VALUE30	GRAY30 value for 32-level histogram
15–14 RSVD13	Reserved, always set to zero.
13–8 VALUE29	GRAY29 value for 32-level histogram
7–6 RSVD2	Reserved, always set to zero.
VALUE28	GRAY28 value for 32-level histogram

13.6.12.215 PXP_HW_PXP_COMP_CTRL

This register defines the control bits for the pxp compression sub-block.

HW_PXP_COMP_CTRL: 0x2C00

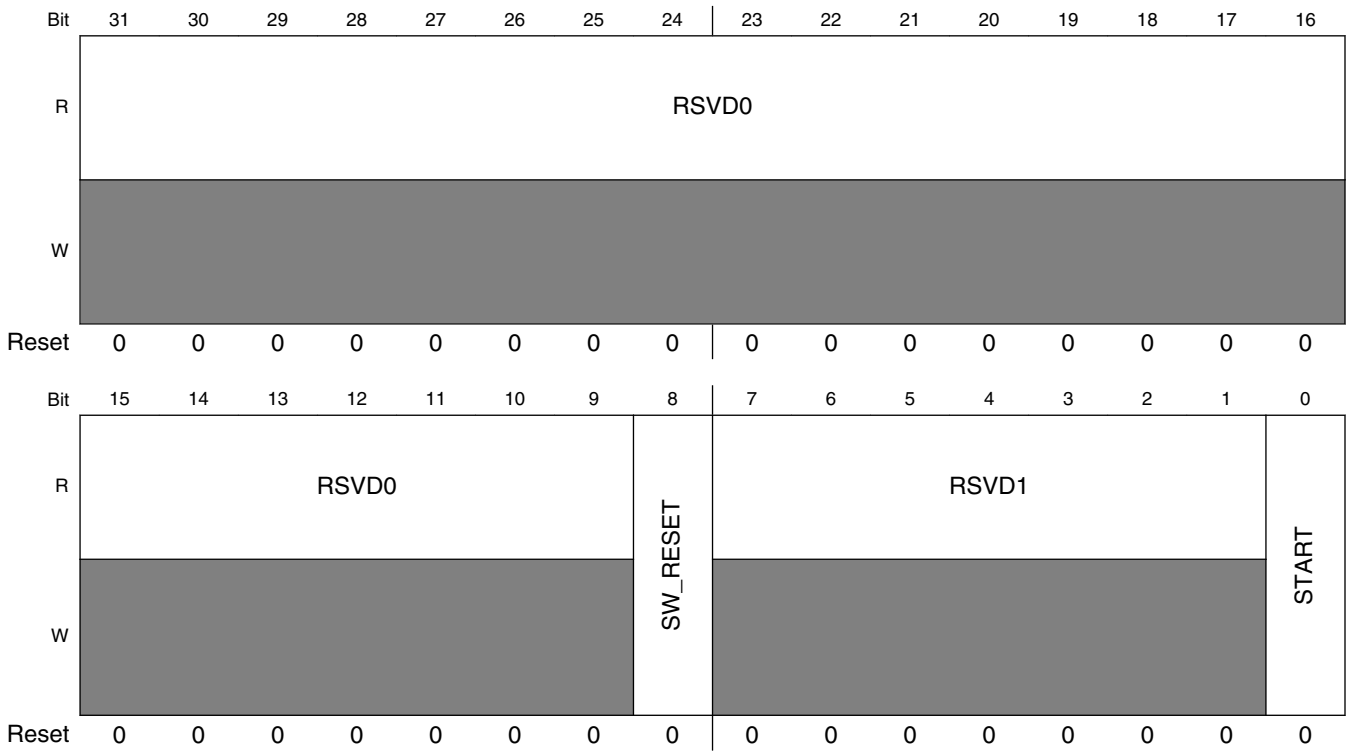
HW_PXP_COMP_CTRL_SET: 0x2C04

HW_PXP_COMP_CTRL_CLR: 0x2C08

HW_PXP_COMP_CTRL_TOG: 0x2C0C

EXAMPLE

Address: 3070_0000h base + 2C00h offset = 3070_2C00h



PXP_HW_PXP_COMP_CTRL field descriptions

Field	Description
31–9 RSVD0	Reserved. This field always reads 0.
8 SW_RESET	Write to 1 to do a software reset to the engine, self-clear.
7–1 RSVD1	Reserved. This field always reads 0.
0 START	Write to 1 to start operation, self-clear

13.6.12.216 PXP_HW_PXP_COMP_FORMAT0

This register defines the format bits for the pxp compression sub-block.

HW_PXP_COMP_FORMAT0: 0x2C10

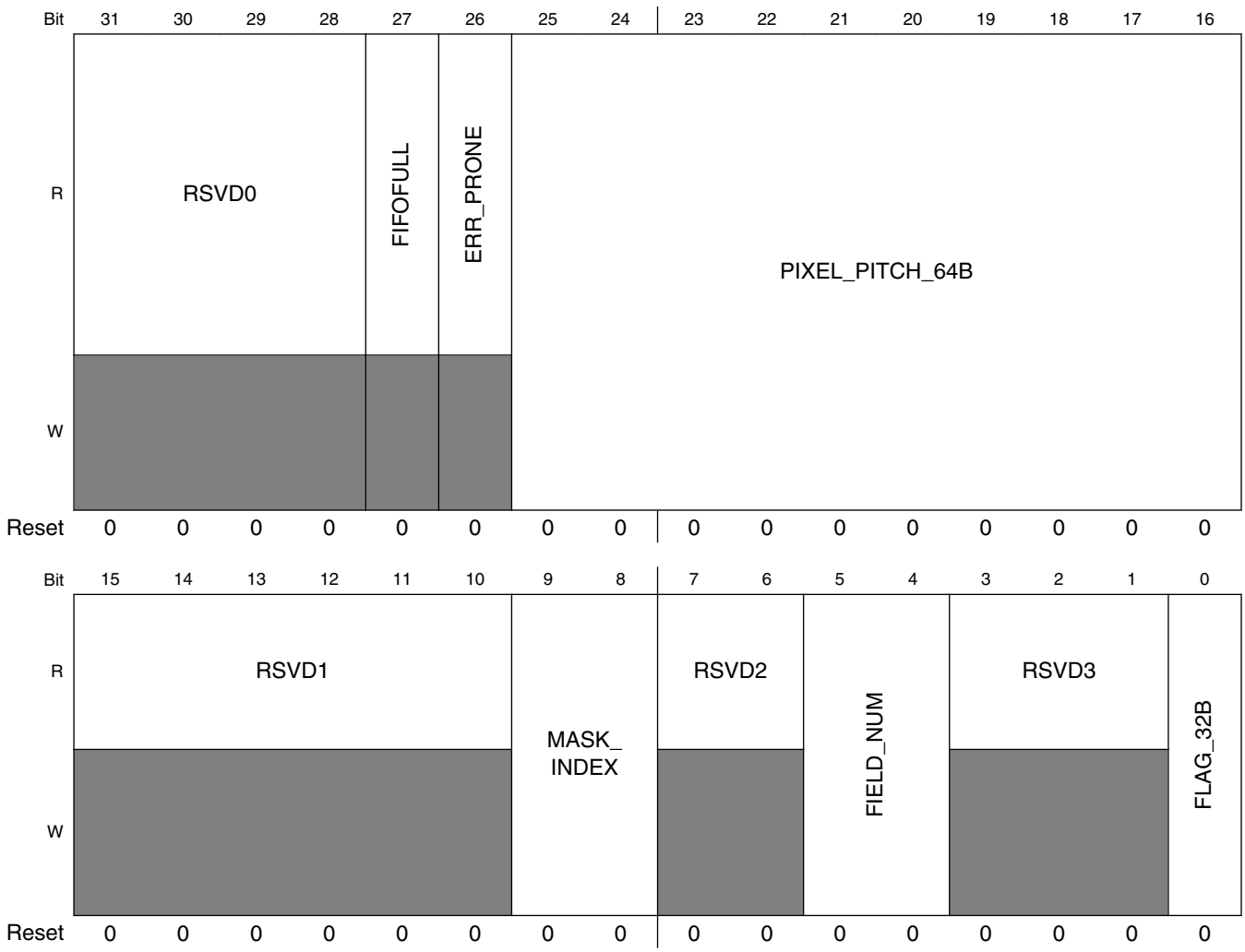
HW_PXP_COMP_FORMAT0_SET: 0x2C14

HW_PXP_COMP_FORMAT0_CLR: 0x2C18

HW_PXP_COMP_FORMAT0_TOG: 0x2C1C

EXAMPLE

Address: 3070_0000h base + 2C10h offset = 3070_2C10h



PXP_HW_PXP_COMP_FORMAT0 field descriptions

Field	Description
31–28 RSVD0	Reserved. This field always reads 0.
27 FIFOFULL	step1 write fifo full

Table continues on the next page...

PXP_HW_PXP_COMP_FORMAT0 field descriptions (continued)

Field	Description
26 ERR_PRONE	step1 write fifo full. If detected, this bit is 1, there is data error in current frame.
25–16 PIXEL_PITCH_64B	extend each line to be 64-bit aligned
15–10 RSVD1	Reserved. This field always reads 0.
9–8 MASK_INDEX	which field is the mask,0 for A, 3 for D
7–6 RSVD2	Reserved. This field always reads 0.
5–4 FIELD_NUM	indicate how many fields in one pixel,0 for only A;3 for ABCD
3–1 RSVD3	Reserved. This field always reads 0.
0 FLAG_32B	1 indicate 32-bit for one pixel, 0 for 16-bit

13.6.12.217 PXP_HW_PXP_COMP_FORMAT1

This register defines the format bits for the pxp compression sub-block.

EXAMPLE

Address: 3070_0000h base + 2C20h offset = 3070_2C20h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_LEN			D_OFFSET					C_LEN			C_OFFSET					B_LEN			B_OFFSET					A_LEN			A_OFFSET				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_COMP_FORMAT1 field descriptions

Field	Description
31–29 D_LEN	length of field D, 0 for 1 byte, 7 for 8 bytes, length of the field plus the length of RLE(*_runlen) should not more than 16-bit
28–24 D_OFFSET	offset for field D, 0 means D start from bit0
23–21 C_LEN	length of field C, 0 for 1 byte, 7 for 8 bytes, length of the field plus the length of RLE(*_runlen) should not more than 16-bit
20–16 C_OFFSET	offset for field C, 0 means C start from bit0
15–13 B_LEN	length of field B, 0 for 1 byte, 7 for 8 bytes, length of the field plus the length of RLE(*_runlen) should not more than 16-bit

Table continues on the next page...

PXP_HW_PXP_COMP_FORMAT1 field descriptions (continued)

Field	Description
12–8 B_OFFSET	offset for field B, 0 means B start from bit0
7–5 A_LEN	length of field A, 0 for 1 byte, 7 for 8 bytes, length of the field plus the length of RLE(*_runlen) should not more than 16-bit
A_OFFSET	offset for field A, 0 means A start from bit0

13.6.12.218 PXP_HW_PXP_COMP_FORMAT2

This register defines the format bits for the pxp compression sub-block.

EXAMPLE

Address: 3070_0000h base + 2C30h offset = 3070_2C30h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																D_RUNLEN				C_RUNLEN				B_RUNLEN				A_RUNLEN			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_COMP_FORMAT2 field descriptions

Field	Description
31–16 RSVD	Reserved. This field always reads 0.
15–12 D_RUNLEN	length of the RLE for field D, 12-bit(4095) max
11–8 C_RUNLEN	length of the RLE for field C, 12-bit(4095) max
7–4 B_RUNLEN	length of the RLE for field B, 12-bit(4095) max
A_RUNLEN	length of the RLE for field A, 12-bit(4095) max

13.6.12.219 PXP_HW_PXP_COMP_MASK0

This register defines the format bits for the pxp compress mask configure.

EXAMPLE

Address: 3070_0000h base + 2C40h offset = 3070_2C40h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VLD_MASK_LOW																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_COMP_MASK0 field descriptions

Field	Description
VLD_MASK_LOW	low 32bit of the valid mask, one of ABCD will be vld_flag,1 left shifted by vld_flag anded with vld_mask will be used to check whether this pixel is valid

13.6.12.220 PXP_HW_PXP_COMP_MASK1

This register defines the format bits for the pxp compress mask configure.

EXAMPLE

Address: 3070_0000h base + 2C50h offset = 3070_2C50h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VLD_MASK_HIGH																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_COMP_MASK1 field descriptions

Field	Description
VLD_MASK_HIGH	high 32bit of the valid mask

13.6.12.221 PXP_HW_PXP_COMP_BUFFER_SIZE

This register defines the format bits for the pxp compress mask configure.

EXAMPLE

Address: 3070_0000h base + 2C60h offset = 3070_2C60h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0			PIXEL_WIDTH													RSVD1			PIXEL_LENGTH												
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_COMP_BUFFER_SIZE field descriptions

Field	Description
31–29 RSVD0	Reserved. This field always reads 0.
28–16 PIXEL_WIDTH	pixel width of the input frame, 4096 max
15–13 RSVD1	Reserved. This field always reads 0.
PIXEL_LENGTH	pixel length of the input frame, 4096 max

13.6.12.222 PXP_HW_PXP_COMP_SOURCE

This register defines the pxp compress source buffer address register.

EXAMPLE

Address: 3070_0000h base + 2C70h offset = 3070_2C70h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SOURCE_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_COMP_SOURCE field descriptions

Field	Description
SOURCE_ADDR	source address of the input frame that located in the memory, should be 32-byte aligned

13.6.12.223 PXP_HW_PXP_COMP_TARGET

This register defines the pxp compress target buffer address register.

EXAMPLE

Address: 3070_0000h base + 2C80h offset = 3070_2C80h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TARGET_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_COMP_TARGET field descriptions

Field	Description
TARGET_ADDR	taget address of the output frmae that the pxp compress engine should write to the memory, should be 32-byte aligned

13.6.12.224 PXP_HW_PXP_COMP_BUFFER_A

This register defines the pxp compress field A buffer address register.

EXAMPLE

Address: 3070_0000h base + 2C90h offset = 3070_2C90h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	A_SRAM_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_COMP_BUFFER_A field descriptions

Field	Description
A_SRAM_ADDR	sram address used for inter-data saving of A filed,should be 32-bit aligned, SW should make sure the SRAM addr for 4 field will not overlap

13.6.12.225 PXP_HW_PXP_COMP_BUFFER_B

This register defines the pxp compress field B buffer address register.

EXAMPLE

Address: 3070_0000h base + 2CA0h offset = 3070_2CA0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	B_SRAM_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_COMP_BUFFER_B field descriptions

Field	Description
B_SRAM_ADDR	sram address used for inter-data saving of B filed,should be 32-bit aligned, SW should make sure the SRAM addr for 4 field will not overlap

13.6.12.226 PXP_HW_PXP_COMP_BUFFER_C

This register defines the pxp compress field C buffer address register.

EXAMPLE

Address: 3070_0000h base + 2CB0h offset = 3070_2CB0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	C_SRAM_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_COMP_BUFFER_C field descriptions

Field	Description
C_SRAM_ADDR	sram address used for inter-data saving of C filed,should be 32-bit aligned, SW should make sure the SRAM addr for 4 field will not overlap

13.6.12.227 PXP_HW_PXP_COMP_BUFFER_D

This register defines the pxp compress field D buffer address register.

EXAMPLE

Address: 3070_0000h base + 2CC0h offset = 3070_2CC0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_SRAM_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_HW_PXP_COMP_BUFFER_D field descriptions

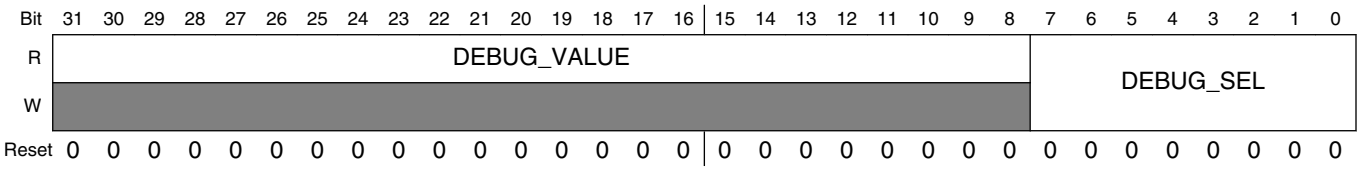
Field	Description
D_SRAM_ADDR	sram address used for inter-data saving of D filed,should be 32-bit aligned, SW should make sure the SRAM addr for 4 field will not overlap

13.6.12.228 PXP_HW_PXP_COMP_DEBUG

This register defines the pxp compress debug register.

EXAMPLE

Address: 3070_0000h base + 2CD0h offset = 3070_2CD0h



PXP_HW_PXP_COMP_DEBUG field descriptions

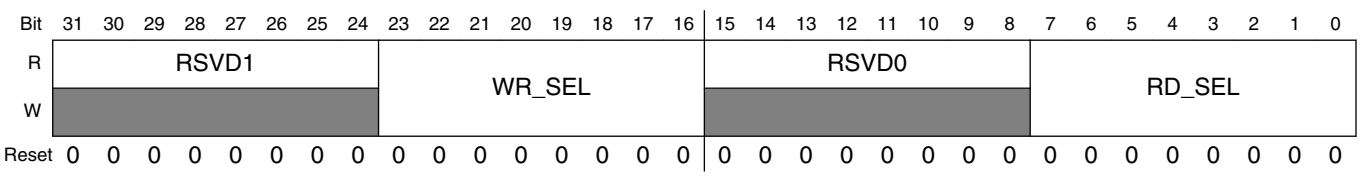
Field	Description
31–8 DEBUG_VALUE	value of selected debug signal
DEBUG_SEL	debug selection

13.6.12.229 PXP_HW_PXP_BUS_MUX

This register defines the pxp subblock bus mux on top level.

EXAMPLE

Address: 3070_0000h base + 2CE0h offset = 3070_2CE0h



PXP_HW_PXP_BUS_MUX field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
23–16 WR_SEL	Subblock BUS to AXI MUX, setting 0 to axi0 and setting 1 to axi1
15–8 RSVD0	Reserved, always set to zero.
RD_SEL	Subblock BUS to AXI MUX, setting 0 to axi0 and setting 1 to axi1

13.6.12.230 PXP_HW_PXP_HANDSHAKE_READY_MUX0

This register defines the pxp subblock handshake signals ready mux on top level.

EXAMPLE

Address: 3070_0000h base + 2CF0h offset = 3070_2CF0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HSK7				HSK6				HSK5				HSK4				HSK3				HSK2				HSK1				HSK0			
Reset	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

PXP_HW_PXP_HANDSHAKE_READY_MUX0 field descriptions

Field	Description
31–28 HSK7	Subblock double buffer handshake signals MUX
27–24 HSK6	Subblock double buffer handshake signals MUX
23–20 HSK5	Subblock double buffer handshake signals MUX
19–16 HSK4	Subblock double buffer handshake signals MUX
15–12 HSK3	Subblock double buffer handshake signals MUX
11–8 HSK2	Subblock double buffer handshake signals MUX
7–4 HSK1	Subblock double buffer handshake signals MUX
HSK0	Subblock double buffer handshake signals MUX 0: Ready signal source is from pxp_control; 1: Ready signal source is from pxp_store_wfe_B CH0; 2: Ready signal source is from pxp_store_wfe_B CH1; 3: Ready signal source is from pxp_store_pre_ditering CH0; 4: Ready signal source is from pxp_store_pre_ditering CH1; 5: Ready signal source is from pxp_store_dithering CH0; 6: Ready signal source is from pxp_store_dithering CH1; 7: Ready signal source is from pxp_store_wfe_a CH0; 8: Ready signal source is from pxp_store_wfe_a CH1; 9: Ready signal source is from cpu_fetch_sw0_ready; A: Ready signal source is from cpu_fetch_sw1_ready; B: Ready signal source is from cpu_store_sw0_ready; C: Ready signal source is from cpu_store_sw1_ready;

13.6.12.231 PXP_HW_PXP_HANDSHAKE_READY_MUX1

This register defines the pxp subblock handshake signals ready mux on top level.

EXAMPLE

Address: 3070_0000h base + 2D00h offset = 3070_2D00h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HSK15				HSK14				HSK13				HSK12				HSK11				HSK10				HSK9				HSK8			
Reset	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0

PXP_HW_PXP_HANDSHAKE_READY_MUX1 field descriptions

Field	Description
31–28 HSK15	Subblock double buffer handshake signals MUX
27–24 HSK14	Subblock double buffer handshake signals MUX
23–20 HSK13	Subblock double buffer handshake signals MUX
19–16 HSK12	Subblock double buffer handshake signals MUX
15–12 HSK11	Subblock double buffer handshake signals MUX
11–8 HSK10	Subblock double buffer handshake signals MUX
7–4 HSK9	Subblock double buffer handshake signals MUX
HSK8	Subblock double buffer handshake signals MUX

13.6.12.232 PXP_HW_PXP_HANDSHAKE_DONE_MUX0

This register defines the pxp subblock handshake signals done mux on top level.

EXAMPLE

Address: 3070_0000h base + 2D10h offset = 3070_2D10h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HSK7				HSK6				HSK5				HSK4				HSK3				HSK2				HSK1				HSK0			
Reset	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

PXP_HW_PXP_HANDSHAKE_DONE_MUX0 field descriptions

Field	Description
31–28 HSK7	Subblock double buffer handshake signals MUX
27–24 HSK6	Subblock double buffer handshake signals MUX
23–20 HSK5	Subblock double buffer handshake signals MUX

Table continues on the next page...

PXP_HW_PXP_HANDSHAKE_DONE_MUX0 field descriptions (continued)

Field	Description
19–16 HSK4	Subblock double buffer handshake signals MUX
15–12 HSK3	Subblock double buffer handshake signals MUX
11–8 HSK2	Subblock double buffer handshake signals MUX
7–4 HSK1	Subblock double buffer handshake signals MUX
HSK0	Subblock double buffer handshake signals MUX 0: Done signal source is from LCDIF; 1: Done signal source is from pxp_fetch_input CH0; 2: Done signal source is from pxp_fetch_input CH1; 3: Done signal source is from pxp_fetch_dithering CH0; 4: Done signal source is from pxp_fetch_dithering CH1; 5: Done signal source is from pxp_fetch_wfe_a CH0; 6: Done signal source is from pxp_fetch_wfe_a CH1; 7: Done signal source is from pxp_fetch_wfe_b CH0; 8: Done signal source is from pxp_fetch_wfe_b CH1; 9: Done signal source is from cpu_fetch_sw0_done; A: Done signal source is from cpu_fetch_sw1_done; B: Done signal source is from cpu_store_sw0_done; C: Done signal source is from cpu_store_sw1_done;

13.6.12.233 PXP_HW_PXP_HANDSHAKE_DONE_MUX1

This register defines the pxp subblock handshake signals done mux on top level.

EXAMPLE

Address: 3070_0000h base + 2D20h offset = 3070_2D20h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HSK15				HSK14				HSK13				HSK12				HSK11				HSK10				HSK9				HSK8			
Reset	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0

PXP_HW_PXP_HANDSHAKE_DONE_MUX1 field descriptions

Field	Description
31–28 HSK15	Subblock double buffer handshake signals MUX

Table continues on the next page...

PXP_HW_PXP_HANDSHAKE_DONE_MUX1 field descriptions (continued)

Field	Description
27–24 HSK14	Subblock double buffer handshake signals MUX
23–20 HSK13	Subblock double buffer handshake signals MUX
19–16 HSK12	Subblock double buffer handshake signals MUX
15–12 HSK11	Subblock double buffer handshake signals MUX
11–8 HSK10	Subblock double buffer handshake signals MUX
7–4 HSK9	Subblock double buffer handshake signals MUX
HSK8	Subblock double buffer handshake signals MUX

13.6.12.234 PXP_HW_PXP_HANDSHAKE_CPU_FETCH

This register defines the pxp software handshake signals with CPU.

HW_PXP_HANDSHAKE_CPU_FETCH: 0x2D30

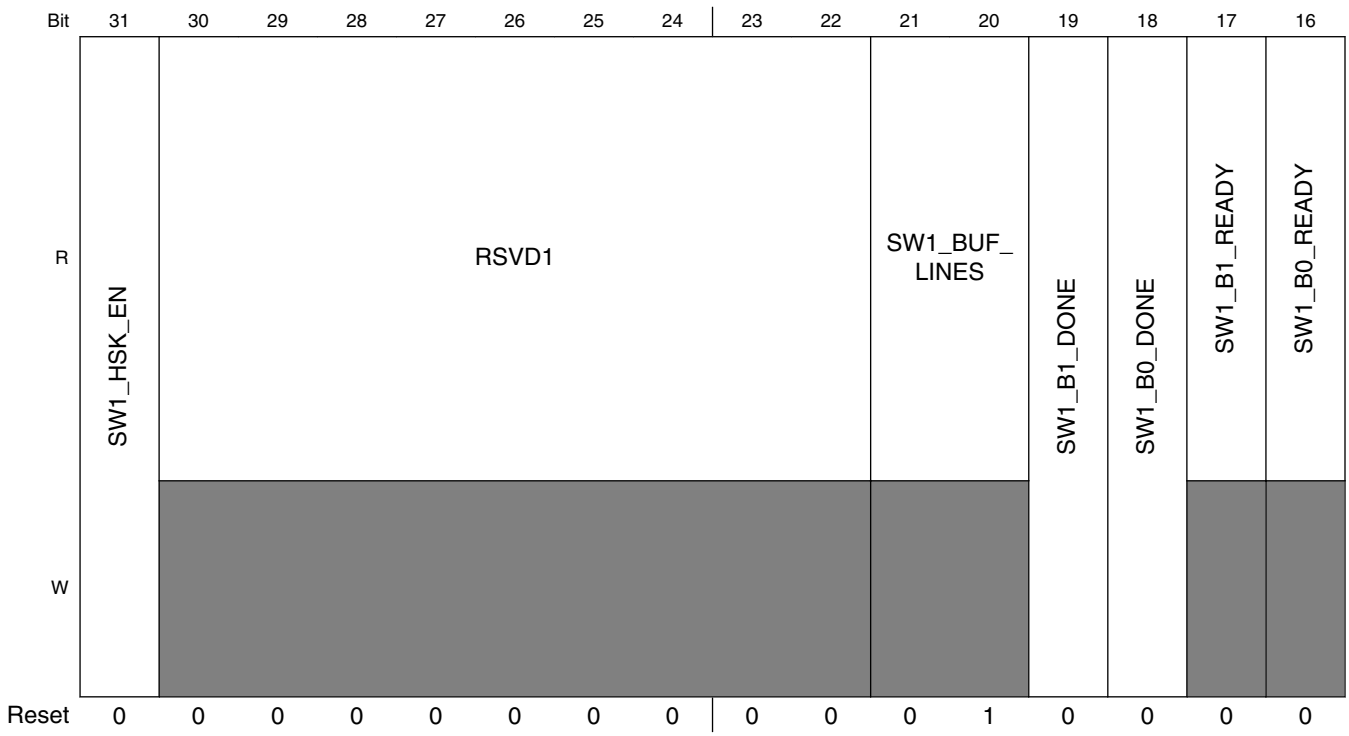
HW_PXP_HANDSHAKE_CPU_FETCH_SET: 0x2D34

HW_PXP_HANDSHAKE_CPU_FETCH_CLR: 0x2D38

HW_PXP_HANDSHAKE_CPU_FETCH_TOG: 0x2D3C

EXAMPLE

Address: 3070_0000h base + 2D30h offset = 3070_2D30h



PXP_HW_PXP_HANDSHAKE_CPU_FETCH field descriptions

Field	Description
31 SW1_HSK_EN	Enable software handshake 1 with CPU

Table continues on the next page...

PXP_HW_PXP_HANDSHAKE_CPU_FETCH field descriptions (continued)

Field	Description
30–22 RSVD1	Reserved, always set to zero.
21–20 SW1_BUF_ LINES	Buffer lines for software handshake 0x0 **LINE_4** — Buffer lines is 4 lines. 0x1 **LINE_8** — Buffer lines is 8 lines. 0x2 **LINE_16** — Buffer lines is 16 lines.
19 SW1_B1_DONE	CPU b1 buffer done to PXP
18 SW1_B0_DONE	CPU b0 buffer done to PXP
17 SW1_B1_ READY	PXP b1 buffer ready to CPU
16 SW1_B0_ READY	PXP b0 buffer ready to CPU
15 SW0_HSK_EN	Enable software handshake 0 with CPU
14–6 RSVD0	Reserved, always set to zero.
5–4 SW0_BUF_ LINES	Buffer lines for software handshake 0x0 **LINE_4** — Buffer lines is 4 lines. 0x1 **LINE_8** — Buffer lines is 8 lines. 0x2 **LINE_16** — Buffer lines is 16 lines.
3 SW0_B1_DONE	CPU b1 buffer done to PXP
2 SW0_B0_DONE	CPU b0 buffer done to PXP
1 SW0_B1_ READY	PXP b1 buffer ready to CPU
0 SW0_B0_ READY	PXP b0 buffer ready to CPU

13.6.12.235 PXP_HW_PXP_HANDSHAKE_CPU_STORE

This register defines the pxp software handshake signals with CPU.

HW_PXP_HANDSHAKE_CPU_STORE: 0x2D40

HW_PXP_HANDSHAKE_CPU_STORE_SET: 0x2D44

HW_PXP_HANDSHAKE_CPU_STORE_CLR: 0x2D48

HW_PXP_HANDSHAKE_CPU_STORE_TOG: 0x2D4C

EXAMPLE

Address: 3070_0000h base + 2D40h offset = 3070_2D40h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SW1_HSK_EN	RSVD1									SW1_BUF_LINES		SW1_B1_DONE	SW1_B0_DONE	SW1_B1_READY	SW1_B0_READY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SW0_HSK_EN	RSVD0									SW0_BUF_LINES		SW0_B1_DONE	SW0_B0_DONE	SW0_B1_READY	SW0_B0_READY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

PXP_HW_PXP_HANDSHAKE_CPU_STORE field descriptions

Field	Description
31 SW1_HSK_EN	Enable software handshake 1 with CPU
30–22 RSVD1	Reserved, always set to zero.
21–20 SW1_BUF_ LINES	Buffer lines for software handshake 0x0 **LINE_4** — Buffer lines is 4 lines. 0x1 **LINE_8** — Buffer lines is 8 lines. 0x2 **LINE_16** — Buffer lines is 16 lines.
19 SW1_B1_DONE	CPU b1 buffer done to PXP
18 SW1_B0_DONE	CPU b0 buffer done to PXP
17 SW1_B1_ READY	PXP b1 buffer ready to CPU
16 SW1_B0_ READY	PXP b0 buffer ready to CPU
15 SW0_HSK_EN	Enable software handshake 0 with CPU
14–6 RSVD0	Reserved, always set to zero.
5–4 SW0_BUF_ LINES	Buffer lines for software handshake 0x0 **LINE_4** — Buffer lines is 4 lines. 0x1 **LINE_8** — Buffer lines is 8 lines. 0x2 **LINE_16** — Buffer lines is 16 lines.
3 SW0_B1_DONE	CPU b1 buffer done to PXP
2 SW0_B0_DONE	CPU b0 buffer done to PXP
1 SW0_B1_ READY	PXP b1 buffer ready to CPU
0 SW0_B0_ READY	PXP b0 buffer ready to CPU

13.7 Synchronous Audio Interface (SAI)

13.7.1 Overview

The synchronous audio interface (SAI) supports full-duplex serial interfaces with frame synchronization such as I^2S, AC97, TDM, and codec/DSP interfaces.

13.7.1.1 Features

Note that some of the features are not supported across all SAI instances; see the chip-specific information in the first section of this chapter.

- Transmitter with independent bit clock and frame sync supporting 1 data line
- Receiver with independent bit clock and frame sync supporting 1 data line
- Maximum Frame Size of 32 words
- Word size of between 8-bits and 32-bits
- Word size configured separately for first word and remaining words in frame
- Asynchronous 32 × 32-bit FIFO for each transmit and receive channel
- Supports graceful restart after FIFO error

13.7.1.2 Block diagram

The following block diagram also shows the module clocks.

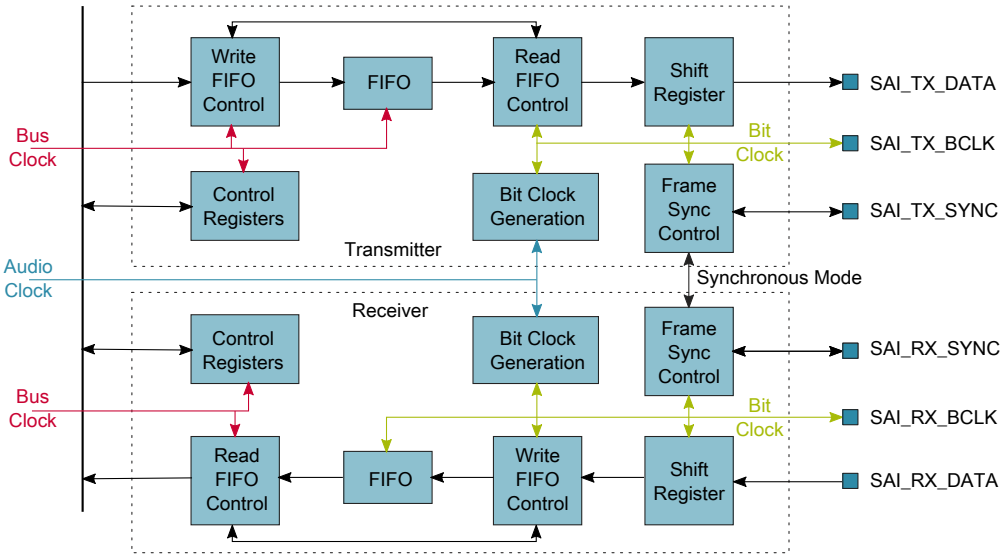


Figure 13-101. I²S/SAI block diagram

13.7.1.3 Modes of operation

The module operates in these power modes: Run mode, stop modes, and Debug mode.

13.7.1.3.1 Run mode

In Run mode, the SAI transmitter and receiver operate normally.

13.7.1.3.2 Stop modes

In Stop mode, the transmitter is disabled after completing the current transmit frame, and, the receiver is disabled after completing the current receive frame. Entry into Stop mode is prevented–not acknowledged–while waiting for the transmitter and receiver to be disabled at the end of the current frame.

13.7.1.3.3 Debug mode

In Debug mode, the SAI transmitter and/or receiver can continue operating provided the Debug Enable bit is set. When TCSR[DBGE] or RCSR[DBGE] bit is clear and Debug mode is entered, the SAI is disabled after completing the current transmit or receive frame. The transmitter and receiver bit clocks are not affected by Debug mode.

13.7.2 External Signals

The following table describes the external signals of SAI:

Table 13-40. SAI External Signals

Signal	Description	Pad	Mode	Direction
SAI1_MCLK	Audio Master Clock. The master clock is an input when externally generated and an output when internally generated.	GPIO1_IO01	ALT3	IO
		SAI1_MCLK	ALT0	
SAI1_RX_BCLK	Receive Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	ENET1_TXC	ALT2	IO
		SAI1_RXC	ALT0	
SAI1_RX_DATA	Receive Data. The receive data is sampled synchronously by the bit clock.	ENET1_TX_CLK	ALT2	I
		SAI1_RXD	ALT0	
SAI1_RX_SYNC	Receive Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	ENET1_TX_CTL	ALT2	IO
		SAI1_RXFS	ALT0	
SAI1_TX_BCLK	Transmit Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	ENET1_RX_CLK	ALT2	IO
		SAI1_TXC	ALT0	
SAI1_TX_DATA	Transmit Data. The transmit data is generated synchronously by the bit clock and is tristated whenever not transmitting a word.	ENET1_COL	ALT2	O
		SAI1_TXD	ALT0	
SAI1_TX_SYNC	Transmit Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	ENET1_CRS	ALT2	IO
		SAI1_TXFS	ALT0	
SAI2_MCLK	Audio Master Clock. The master clock is an input when externally generated and an output when internally generated.	GPIO1_IO02	ALT3	IO
		SAI1_MCLK	ALT2	
		SD2_RESET_B	ALT1	
SAI2_RX_BCLK	Receive Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	SAI1_RXC	ALT2	IO
		SD2_CMD	ALT1	
SAI2_RX_DATA	Receive Data. The receive data is sampled synchronously by the bit clock.	SAI2_RXD	ALT0	I
		SD2_DATA0	ALT1	
SAI2_RX_SYNC	Receive Frame Sync. The frame sync is an input sampled	SAI1_RXFS	ALT2	IO
		SD2_CLK	ALT1	

Table continues on the next page...

Table 13-40. SAI External Signals (continued)

Signal	Description	Pad	Mode	Direction
	synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.			
SAI2_TX_BCLK	Transmit Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	SAI2_TXC	ALT0	IO
		SD2_DATA1	ALT1	
SAI2_TX_DATA	Transmit Data. The transmit data is generated synchronously by the bit clock and is tristated whenever not transmitting a word.	SAI2_TXD	ALT0	O
		SD2_DATA3	ALT1	
SAI2_TX_SYNC	Transmit Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	SAI2_TXFS	ALT0	IO
		SD2_DATA2	ALT1	
SAI3_MCLK	Audio Master Clock. The master clock is an input when externally generated and an output when internally generated.	GPIO1_IO03	ALT3	IO
		SD1_RESET_B	ALT1	
		SD3_RESET_B	ALT3	
		UART1_TXD	ALT2	
SAI3_RX_BCLK	Receive Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	SD1_CMD	ALT1	IO
		SD3_CMD	ALT3	
		UART2_RXD	ALT2	
SAI3_RX_DATA	Receive Data. The receive data is sampled synchronously by the bit clock.	SD1_DATA0	ALT1	I
		SD3_DATA0	ALT3	
		UART2_TXD	ALT2	
SAI3_RX_SYNC	Receive Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	SD1_CLK	ALT1	IO
		SD3_CLK	ALT3	
		UART3_RXD	ALT2	
SAI3_TX_BCLK	Transmit Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	SD1_DATA1	ALT1	IO
		SD3_DATA1	ALT3	
		UART3_TXD	ALT2	
SAI3_TX_DATA	Transmit Data. The transmit data is generated synchronously by the bit clock and is tristated whenever not transmitting a word.	SD1_DATA3	ALT1	O
		SD3_DATA3	ALT3	
		UART3_RTS	ALT2	
SAI3_TX_SYNC	Transmit Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	SD1_DATA2	ALT1	IO
		SD3_DATA2	ALT3	
		UART3_CTS	ALT2	

13.7.3 Functional description

This section provides a complete functional description of the block.

13.7.3.1 SAI clocking

The SAI clocks include:

- The audio master clock
- The bit clock
- The bus clock

13.7.3.1.1 Audio master clock

The audio master clock is used to generate the bit clock when the receiver or transmitter is configured for an internally generated bit clock. The transmitter and receiver can independently select between the bus clock and up to three audio master clocks to generate the bit clock.

The audio master clock generation and selection is chip-specific. Refer to chip-specific clocking information about how the audio master clocks are generated.

13.7.3.1.2 Bit clock

The SAI transmitter and receiver support asynchronous free-running bit clocks that can be generated internally from an audio master clock or supplied externally. There is also the option for synchronous bit clock and frame sync operation between the receiver and transmitter.

Externally generated bit clocks must be:

- Enabled before the SAI transmitter or receiver is enabled
- Disabled after the SAI transmitter or receiver is disabled and completes its current frames

If the SAI transmitter or receiver is using an externally generated bit clock in asynchronous mode and that bit clock is generated by an SAI that is disabled in stop mode, then the transmitter or receiver should be disabled by software before entering stop mode. This issue does not apply when the transmitter or receiver is in a synchronous mode because all synchronous SAIs are enabled and disabled simultaneously.

13.7.3.1.3 Bus clock

The bus clock is used by the control and configuration registers and to generate synchronous interrupts and DMA requests.

NOTE

Although there is no specific minimum bus clock frequency specified, the bus clock frequency must be fast enough (relative to the bit clock frequency) to ensure that the FIFOs can be serviced, without generating either a transmitter FIFO underrun or receiver FIFO overflow condition.

13.7.3.2 SAI resets

The SAI is asynchronously reset on system reset. The SAI has a software reset and a FIFO reset.

13.7.3.2.1 Software reset

The SAI transmitter includes a software reset that resets all transmitter internal logic, including the bit clock generation, status flags, and FIFO pointers. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

The SAI receiver includes a software reset that resets all receiver internal logic, including the bit clock generation, status flags and FIFO pointers. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

13.7.3.2.2 FIFO reset

The SAI transmitter includes a FIFO reset that synchronizes the FIFO write pointer to the same value as the FIFO read pointer. This empties the FIFO contents and is to be used after TCSR[FEF] is set, and before the FIFO is re-initialized and TCSR[FEF] is cleared. The FIFO reset is asserted for one cycle only.

The SAI receiver includes a FIFO reset that synchronizes the FIFO read pointer to the same value as the FIFO write pointer. This empties the FIFO contents and is to be used after the RCSR[FEF] is set and any remaining data has been read from the FIFO, and before the RCSR[FEF] is cleared. The FIFO reset is asserted for one cycle only.

13.7.3.3 Synchronous modes

The SAI transmitter and receiver can operate synchronously to each other.

13.7.3.3.1 Synchronous mode

The SAI transmitter and receiver can be configured to operate with synchronous bit clock and frame sync.

If the transmitter bit clock and frame sync are to be used by both the transmitter and receiver:

- The transmitter must be configured for asynchronous operation and the receiver for synchronous operation.
- In synchronous mode, the receiver is enabled only when both the transmitter and receiver are enabled.
- It is recommended that the transmitter is the last enabled and the first disabled.

If the receiver bit clock and frame sync are to be used by both the transmitter and receiver:

- The receiver must be configured for asynchronous operation and the transmitter for synchronous operation.
- In synchronous mode, the transmitter is enabled only when both the receiver and transmitter are both enabled.
- It is recommended that the receiver is the last enabled and the first disabled.

When operating in synchronous mode, only the bit clock, frame sync, and transmitter/receiver enable are shared. The transmitter and receiver otherwise operate independently, although configuration registers must be configured consistently across both the transmitter and receiver.

13.7.3.4 Frame sync configuration

When enabled, the SAI continuously transmits and/or receives frames of data. Each frame consists of a fixed number of words and each word consists of a fixed number of bits. Within each frame, any given word can be masked causing the receiver to ignore that word and the transmitter to tri-state for the duration of that word.

The frame sync signal is used to indicate the start of each frame. A valid frame sync requires a rising edge (if active high) or falling edge (if active low) to be detected and the transmitter or receiver cannot be busy with a previous frame. A valid frame sync is also ignored (slave mode) or not generated (master mode) for the first four bit clock cycles after enabling the transmitter or receiver.

The transmitter and receiver frame sync can be configured independently with any of the following options:

- Externally generated or internally generated
- Active high or active low
- Assert with the first bit in frame or asserts one bit early
- Assert for a duration between 1 bit clock and the first word length
- Frame length from 1 to 32 words per frame
- Word length to support 8 to 32 bits per word
  - First word length and remaining word lengths can be configured separately
- Words can be configured to transmit/receive MSB first or LSB first

These configuration options cannot be changed after the SAI transmitter or receiver is enabled.

13.7.3.5 Data FIFO

Each transmit and receive channel includes a FIFO of size 32 × 32-bit. The FIFO data is accessed using the SAI Transmit/Receive Data Registers.

13.7.3.5.1 Data alignment

Data in the FIFO can be aligned anywhere within the 32-bit wide register through the use of the First Bit Shifted configuration field, which selects the bit index (between 31 and 0) of the first bit shifted.

Examples of supported data alignment and the required First Bit Shifted configuration are illustrated in Figure 13-102 for LSB First configurations and Figure 13-103 for MSB First configurations.

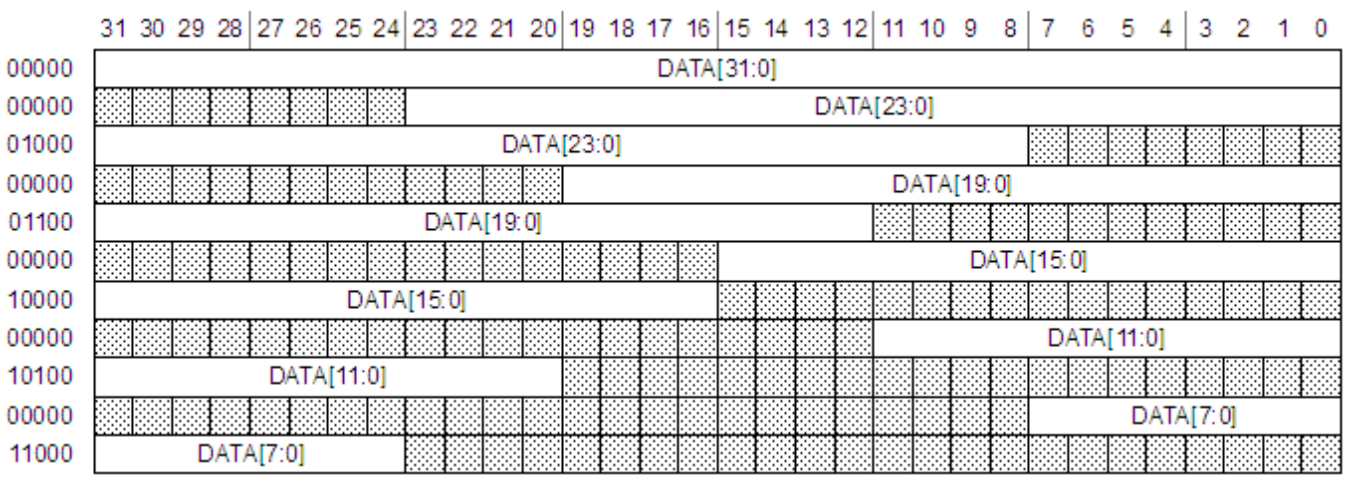


Figure 13-102. SAI first bit shifted, LSB first

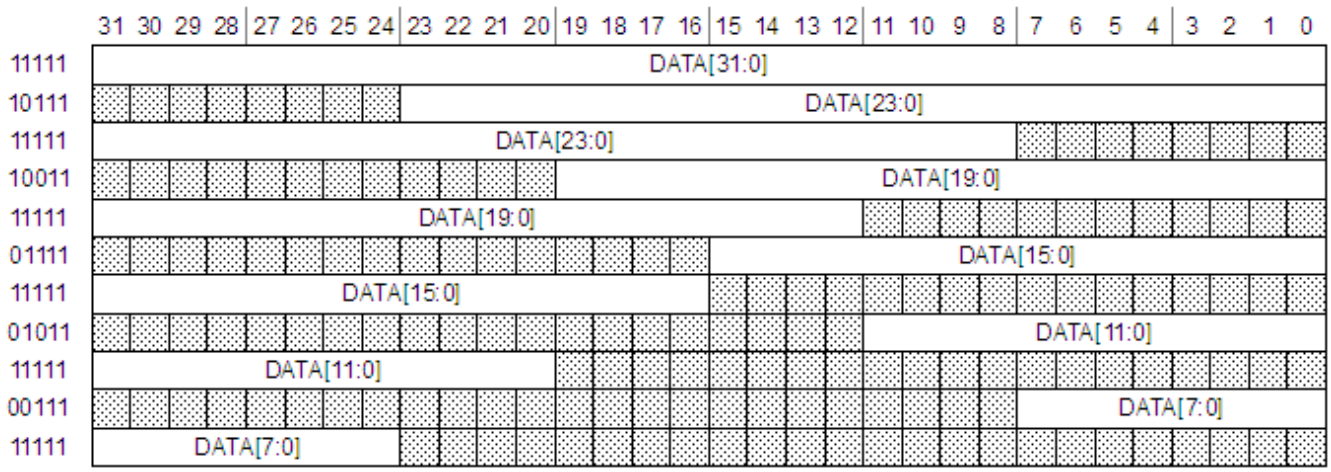


Figure 13-103. SAI first bit shifted, MSB first

13.7.3.5.2 FIFO pointers

When writing to a TDR, the WFP of the corresponding TFR increments after each valid write. The SAI supports 8-bit, 16-bit and 32-bit writes to the TDR and the FIFO pointer will increment after each individual write. Note that 8-bit writes should only be used when transmitting up to 8-bit data and 16-bit writes should only be used when transmitting up to 16-bit data.

Writes to a TDR are ignored if the corresponding bit of TCR3[TCE] is clear or if the FIFO is full. If the Transmit FIFO is empty, the TDR must be written at least three bit clocks before the start of the next unmasked word to avoid a FIFO underrun.

When reading an RDR, the RFP of the corresponding RFR increments after each valid read. The SAI supports 8-bit, 16-bit and 32-bit reads from the RDR and the FIFO pointer will increment after each individual read. Note that 8-bit reads should only be used when receiving up to 8-bit data and 16-bit reads should only be used when receiving up to 16-bit data.

Reads from an RDR are ignored if the corresponding bit of RCR3[RCE] is clear or if the FIFO is empty. If the Receive FIFO is full, the RDR must be read at least three bit clocks before the end of an unmasked word to avoid a FIFO overrun.

13.7.3.6 Word mask register

The SAI transmitter and receiver each contain a word mask register, namely TMR and RMR, that can be used to mask any word in the frame. Because the word mask register is double buffered, software can update it before the end of each frame to mask a particular word in the next frame.

The TMR causes the Transmit Data pin to be tri-stated for the length of each selected word and the transmit FIFO is not read for masked words.

The RMR causes the received data for each selected word to be discarded and not written to the receive FIFO.

13.7.3.7 Interrupts and DMA requests

The SAI transmitter and receiver generate separate interrupts and separate DMA requests, but support the same status flags.

13.7.3.7.1 FIFO request flag

The FIFO request flag is set based on the number of entries in the FIFO and the FIFO watermark configuration.

The transmit FIFO request flag is set when the number of entries in any of the enabled transmit FIFOs is less than or equal to the transmit FIFO watermark configuration and is cleared when the number of entries in each enabled transmit FIFO is greater than the transmit FIFO watermark configuration.

The receive FIFO request flag is set when the number of entries in any of the enabled receive FIFOs is greater than the receive FIFO watermark configuration and is cleared when the number of entries in each enabled receive FIFO is less than or equal to the receive FIFO watermark configuration.

The FIFO request flag can generate an interrupt or a DMA request.

13.7.3.7.2 FIFO warning flag

The FIFO warning flag is set based on the number of entries in the FIFO.

The transmit warning flag is set when the number of entries in any of the enabled transmit FIFOs is empty and is cleared when the number of entries in each enabled transmit FIFO is not empty.

The receive warning flag is set when the number of entries in any of the enabled receive FIFOs is full and is cleared when the number of entries in each enabled receive FIFO is not full.

The FIFO warning flag can generate an Interrupt or a DMA request.

13.7.3.7.3 FIFO error flag

The transmit FIFO error flag is set when the any of the enabled transmit FIFOs underflow. After it is set, all enabled transmit channels repeat the last valid word read from the transmit FIFO until TCSR[FEF] is cleared and the next transmit frame starts. All enabled transmit FIFOs must be reset and initialized with new data before TCSR[FEF] is cleared.

RCSR[FEF] is set when the any of the enabled receive FIFOs overflow. After it is set, all enabled receive channels discard received data until RCSR[FEF] is cleared and the next next receive frame starts. All enabled receive FIFOs should be emptied before RCSR[FEF] is cleared.

The FIFO error flag can generate only an interrupt.

13.7.3.7.4 Sync error flag

The sync error flag, TCSR[SEF] or RCSR[SEF], is set when configured for an externally generated frame sync and the external frame sync asserts when the transmitter or receiver is busy with the previous frame. The external frame sync assertion is ignored and the sync error flag is set. When the sync error flag is set, the transmitter or receiver continues checking for frame sync assertion when idle or at the end of each frame.

The sync error flag can generate an interrupt only.

13.7.3.7.5 Word start flag

The word start flag is set at the start of the second bit clock for the selected word, as configured by the Word Flag register field.

The word start flag can generate an interrupt only.

13.7.4 Memory map and register definition

A read or write access to an address from offset and above will result in a bus error.

I2S memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
308A_0000	SAI Transmit Control Register (I2S1_TCSR)	32	R/W	0000_0000h	13.7.4.1/ 4101
308A_0004	SAI Transmit Configuration 1 Register (I2S1_TCR1)	32	R/W	0000_0000h	13.7.4.2/ 4104

Table continues on the next page...

I2S memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
308A_0008	SAI Transmit Configuration 2 Register (I2S1_TCR2)	32	R/W	0000_0000h	13.7.4.3/ 4104
308A_000C	SAI Transmit Configuration 3 Register (I2S1_TCR3)	32	R/W	0000_0000h	13.7.4.4/ 4106
308A_0010	SAI Transmit Configuration 4 Register (I2S1_TCR4)	32	R/W	0000_0000h	13.7.4.5/ 4107
308A_0014	SAI Transmit Configuration 5 Register (I2S1_TCR5)	32	R/W	0000_0000h	13.7.4.6/ 4108
308A_0020	SAI Transmit Data Register (I2S1_TDR0)	32	W (always reads 0)	0000_0000h	13.7.4.7/ 4109
308A_0040	SAI Transmit FIFO Register (I2S1_TFR0)	32	R	0000_0000h	13.7.4.8/ 4110
308A_0060	SAI Transmit Mask Register (I2S1_TMR)	32	R/W	0000_0000h	13.7.4.9/ 4110
308A_0080	SAI Receive Control Register (I2S1_RCSR)	32	R/W	0000_0000h	13.7.4.10/ 4111
308A_0084	SAI Receive Configuration 1 Register (I2S1_RCR1)	32	R/W	0000_0000h	13.7.4.11/ 4114
308A_0088	SAI Receive Configuration 2 Register (I2S1_RCR2)	32	R/W	0000_0000h	13.7.4.12/ 4115
308A_008C	SAI Receive Configuration 3 Register (I2S1_RCR3)	32	R/W	0000_0000h	13.7.4.13/ 4116
308A_0090	SAI Receive Configuration 4 Register (I2S1_RCR4)	32	R/W	0000_0000h	13.7.4.14/ 4117
308A_0094	SAI Receive Configuration 5 Register (I2S1_RCR5)	32	R/W	0000_0000h	13.7.4.15/ 4119
308A_00A0	SAI Receive Data Register (I2S1_RDR0)	32	R	0000_0000h	13.7.4.16/ 4119
308A_00C0	SAI Receive FIFO Register (I2S1_RFR0)	32	R	0000_0000h	13.7.4.17/ 4120
308A_00E0	SAI Receive Mask Register (I2S1_RMR)	32	R/W	0000_0000h	13.7.4.18/ 4120
308B_0000	SAI Transmit Control Register (I2S2_TCSR)	32	R/W	0000_0000h	13.7.4.1/ 4101
308B_0004	SAI Transmit Configuration 1 Register (I2S2_TCR1)	32	R/W	0000_0000h	13.7.4.2/ 4104
308B_0008	SAI Transmit Configuration 2 Register (I2S2_TCR2)	32	R/W	0000_0000h	13.7.4.3/ 4104
308B_000C	SAI Transmit Configuration 3 Register (I2S2_TCR3)	32	R/W	0000_0000h	13.7.4.4/ 4106
308B_0010	SAI Transmit Configuration 4 Register (I2S2_TCR4)	32	R/W	0000_0000h	13.7.4.5/ 4107
308B_0014	SAI Transmit Configuration 5 Register (I2S2_TCR5)	32	R/W	0000_0000h	13.7.4.6/ 4108

Table continues on the next page...

I2S memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
308B_0020	SAI Transmit Data Register (I2S2_TDR0)	32	W (always reads 0)	0000_0000h	13.7.4.7/ 4109
308B_0040	SAI Transmit FIFO Register (I2S2_TFR0)	32	R	0000_0000h	13.7.4.8/ 4110
308B_0060	SAI Transmit Mask Register (I2S2_TMR)	32	R/W	0000_0000h	13.7.4.9/ 4110
308B_0080	SAI Receive Control Register (I2S2_RCSR)	32	R/W	0000_0000h	13.7.4.10/ 4111
308B_0084	SAI Receive Configuration 1 Register (I2S2_RCR1)	32	R/W	0000_0000h	13.7.4.11/ 4114
308B_0088	SAI Receive Configuration 2 Register (I2S2_RCR2)	32	R/W	0000_0000h	13.7.4.12/ 4115
308B_008C	SAI Receive Configuration 3 Register (I2S2_RCR3)	32	R/W	0000_0000h	13.7.4.13/ 4116
308B_0090	SAI Receive Configuration 4 Register (I2S2_RCR4)	32	R/W	0000_0000h	13.7.4.14/ 4117
308B_0094	SAI Receive Configuration 5 Register (I2S2_RCR5)	32	R/W	0000_0000h	13.7.4.15/ 4119
308B_00A0	SAI Receive Data Register (I2S2_RDR0)	32	R	0000_0000h	13.7.4.16/ 4119
308B_00C0	SAI Receive FIFO Register (I2S2_RFR0)	32	R	0000_0000h	13.7.4.17/ 4120
308B_00E0	SAI Receive Mask Register (I2S2_RMR)	32	R/W	0000_0000h	13.7.4.18/ 4120
308C_0000	SAI Transmit Control Register (I2S3_TCSR)	32	R/W	0000_0000h	13.7.4.1/ 4101
308C_0004	SAI Transmit Configuration 1 Register (I2S3_TCR1)	32	R/W	0000_0000h	13.7.4.2/ 4104
308C_0008	SAI Transmit Configuration 2 Register (I2S3_TCR2)	32	R/W	0000_0000h	13.7.4.3/ 4104
308C_000C	SAI Transmit Configuration 3 Register (I2S3_TCR3)	32	R/W	0000_0000h	13.7.4.4/ 4106
308C_0010	SAI Transmit Configuration 4 Register (I2S3_TCR4)	32	R/W	0000_0000h	13.7.4.5/ 4107
308C_0014	SAI Transmit Configuration 5 Register (I2S3_TCR5)	32	R/W	0000_0000h	13.7.4.6/ 4108
308C_0020	SAI Transmit Data Register (I2S3_TDR0)	32	W (always reads 0)	0000_0000h	13.7.4.7/ 4109
308C_0040	SAI Transmit FIFO Register (I2S3_TFR0)	32	R	0000_0000h	13.7.4.8/ 4110
308C_0060	SAI Transmit Mask Register (I2S3_TMR)	32	R/W	0000_0000h	13.7.4.9/ 4110

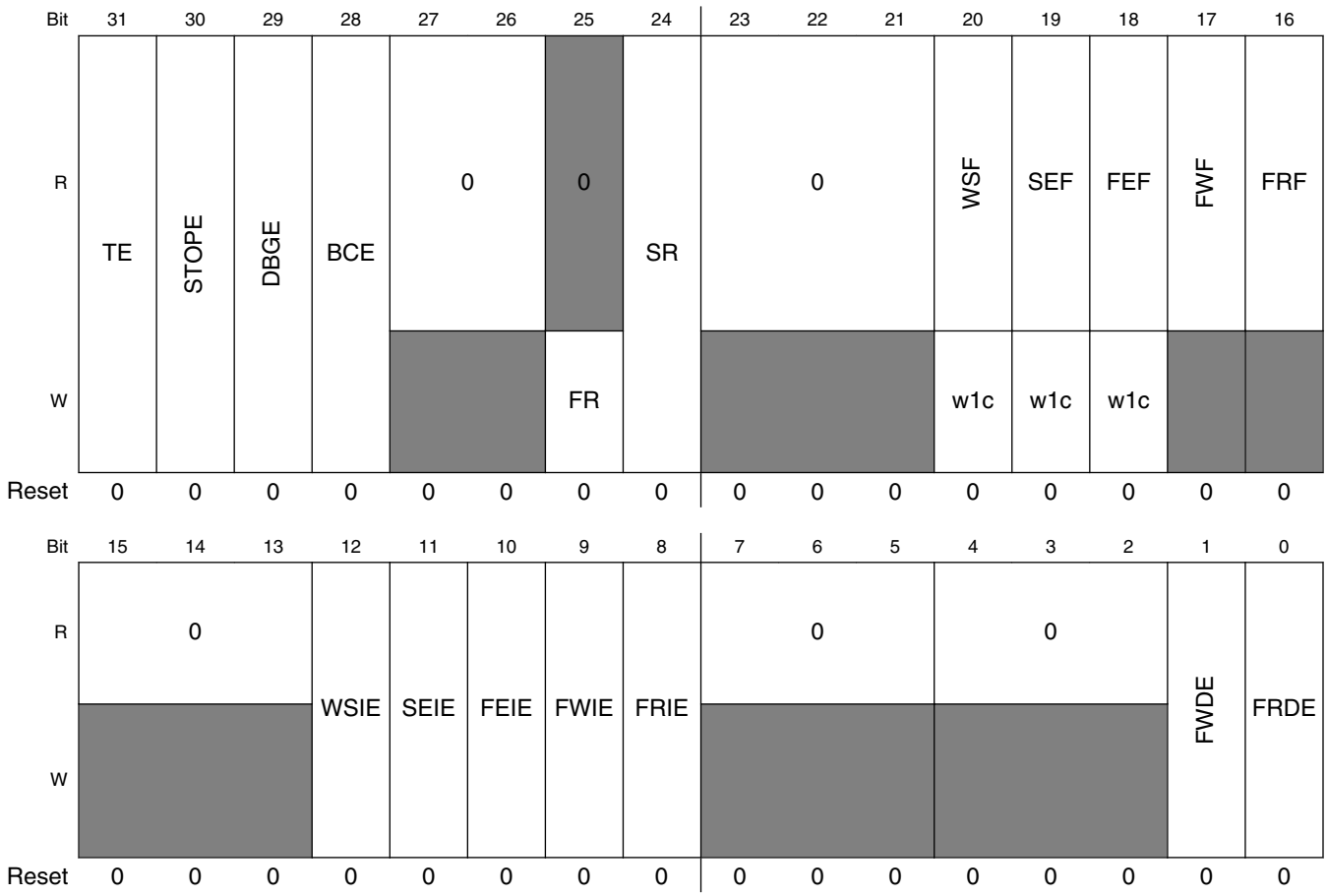
Table continues on the next page...

I2S memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
308C_0080	SAI Receive Control Register (I2S3_RCSR)	32	R/W	0000_0000h	13.7.4.10/ 4111
308C_0084	SAI Receive Configuration 1 Register (I2S3_RCR1)	32	R/W	0000_0000h	13.7.4.11/ 4114
308C_0088	SAI Receive Configuration 2 Register (I2S3_RCR2)	32	R/W	0000_0000h	13.7.4.12/ 4115
308C_008C	SAI Receive Configuration 3 Register (I2S3_RCR3)	32	R/W	0000_0000h	13.7.4.13/ 4116
308C_0090	SAI Receive Configuration 4 Register (I2S3_RCR4)	32	R/W	0000_0000h	13.7.4.14/ 4117
308C_0094	SAI Receive Configuration 5 Register (I2S3_RCR5)	32	R/W	0000_0000h	13.7.4.15/ 4119
308C_00A0	SAI Receive Data Register (I2S3_RDR0)	32	R	0000_0000h	13.7.4.16/ 4119
308C_00C0	SAI Receive FIFO Register (I2S3_RFR0)	32	R	0000_0000h	13.7.4.17/ 4120
308C_00E0	SAI Receive Mask Register (I2S3_RMR)	32	R/W	0000_0000h	13.7.4.18/ 4120

13.7.4.1 SAI Transmit Control Register (I2Sx_TCSR)

Address: Base address + 0h offset



I2Sx_TCSR field descriptions

Field	Description
31 TE	Transmitter Enable Enables/disables the transmitter. When software clears this field, the transmitter remains enabled, and this bit remains set, until the end of the current frame. 0 Transmitter is disabled. 1 Transmitter is enabled, or transmitter has been disabled and has not yet reached end of frame.
30 STOPE	Stop Enable Configures transmitter operation in Stop mode. This field is ignored and the transmitter is disabled in all stop modes. 0 Transmitter disabled in Stop mode. 1 Transmitter enabled in Stop mode.
29 DBGE	Debug Enable

Table continues on the next page...

I2Sx_TCSR field descriptions (continued)

Field	Description
	Enables/disables transmitter operation in Debug mode. The transmit bit clock is not affected by debug mode. 0 Transmitter is disabled in Debug mode, after completing the current frame. 1 Transmitter is enabled in Debug mode.
28 BCE	Bit Clock Enable Enables the transmit bit clock, separately from the TE. This field is automatically set whenever TE is set. When software clears this field, the transmit bit clock remains enabled, and this bit remains set, until the end of the current frame. 0 Transmit bit clock is disabled. 1 Transmit bit clock is enabled.
27–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 FR	FIFO Reset Resets the FIFO pointers. Reading this field will always return zero. FIFO pointers should only be reset when the transmitter is disabled or the FIFO error flag is set. 0 No effect. 1 FIFO reset.
24 SR	Software Reset When set, resets the internal transmitter logic including the FIFO pointers. Software-visible registers are not affected, except for the status registers. 0 No effect. 1 Software reset.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 WSF	Word Start Flag Indicates that the start of the configured word has been detected. Write a logic 1 to this field to clear this flag. 0 Start of word not detected. 1 Start of word detected.
19 SEF	Sync Error Flag Indicates that an error in the externally-generated frame sync has been detected. Write a logic 1 to this field to clear this flag. 0 Sync error not detected. 1 Frame sync error detected.
18 FEF	FIFO Error Flag Indicates that an enabled transmit FIFO has underrun. Write a logic 1 to this field to clear this flag. 0 Transmit underrun not detected. 1 Transmit underrun detected.

Table continues on the next page...

I2Sx_TCSR field descriptions (continued)

Field	Description
17 FWF	FIFO Warning Flag Indicates that an enabled transmit FIFO is empty. 0 No enabled transmit FIFO is empty. 1 Enabled transmit FIFO is empty.
16 FRF	FIFO Request Flag Indicates that the number of words in an enabled transmit channel FIFO is less than or equal to the transmit FIFO watermark. 0 Transmit FIFO watermark has not been reached. 1 Transmit FIFO watermark has been reached.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 WSIE	Word Start Interrupt Enable Enables/disables word start interrupts. 0 Disables interrupt. 1 Enables interrupt.
11 SEIE	Sync Error Interrupt Enable Enables/disables sync error interrupts. 0 Disables interrupt. 1 Enables interrupt.
10 FEIE	FIFO Error Interrupt Enable Enables/disables FIFO error interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
9 FWIE	FIFO Warning Interrupt Enable Enables/disables FIFO warning interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
8 FRIE	FIFO Request Interrupt Enable Enables/disables FIFO request interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FWDE	FIFO Warning DMA Enable

Table continues on the next page...

I2Sx_TCSR field descriptions (continued)

Field	Description
	Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.
0 FRDE	FIFO Request DMA Enable Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.

13.7.4.2 SAI Transmit Configuration 1 Register (I2Sx_TCR1)

Address: Base address + 4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																											TFW				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_TCR1 field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TFW	Transmit FIFO Watermark Configures the watermark level for all enabled transmit channels.

13.7.4.3 SAI Transmit Configuration 2 Register (I2Sx_TCR2)

This register must not be altered when TCSR[TE] is set.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SYNC		BCS	BCI	MSEL		BCP	BCD	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_TCR2 field descriptions

Field	Description
31–30 SYNC	Synchronous Mode Configures between asynchronous and synchronous modes of operation. When configured for a synchronous mode of operation, the receiver must be configured for asynchronous operation. 00 Asynchronous mode. 01 Synchronous with receiver.
29 BCS	Bit Clock Swap This field swaps the bit clock used by the transmitter. When the transmitter is configured in asynchronous mode and this bit is set, the transmitter is clocked by the receiver bit clock (SAI_RX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the transmitter continues to use the transmit frame sync (SAI_TX_SYNC). When the transmitter is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the transmitter bit clock (SAI_TX_BCLK) but use the receiver frame sync (SAI_RX_SYNC). 0 Use the normal bit clock source. 1 Swap the bit clock source.
28 BCI	Bit Clock Input When this field is set and using an internally generated bit clock in either synchronous or asynchronous mode, the bit clock actually used by the transmitter is delayed by the pad output delay (the transmitter is clocked by the pad input as if the clock was externally generated). This has the effect of decreasing the data input setup time, but increasing the data output valid time. The slave mode timing from the datasheet should be used for the transmitter when this bit is set. In synchronous mode, this bit allows the transmitter to use the slave mode timing from the datasheet, while the receiver uses the master mode timing. This field has no effect when configured for an externally generated bit clock . 0 No effect. 1 Internal logic is clocked as if bit clock was externally generated.
27–26 MSEL	MCLK Select Selects the audio Master Clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock. **NOTE:** Depending on the device, some Master Clock options might not be available. See the chip-specific information for the meaning of each option. 00 Master Clock (MCLK) 1 option selected. 01 Master Clock (MCLK) 1 option selected. 10 Master Clock (MCLK) 2 option selected. 11 Master Clock (MCLK) 3 option selected.
25 BCP	Bit Clock Polarity Configures the polarity of the bit clock. 0 Bit clock is active high with drive outputs on rising edge and sample inputs on falling edge. 1 Bit clock is active low with drive outputs on falling edge and sample inputs on rising edge.
24 BCD	Bit Clock Direction

Table continues on the next page...

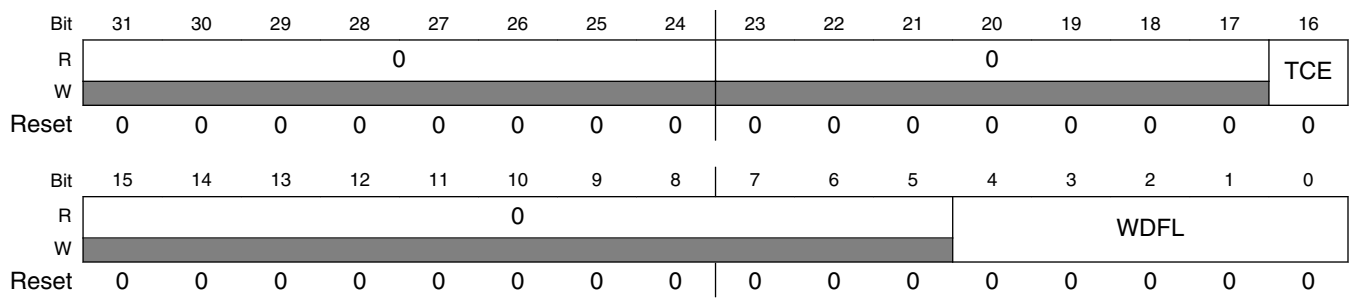
I2Sx_TCR2 field descriptions (continued)

Field	Description
	Configures the direction of the bit clock. 0 Bit clock is generated externally in Slave mode. 1 Bit clock is generated internally in Master mode.
23–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DIV	Bit Clock Divide Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2.

13.7.4.4 SAI Transmit Configuration 3 Register (I2Sx_TCR3)

This register must not be altered when TCSR[TE] is set.

Address: Base address + Ch offset



I2Sx_TCR3 field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 TCE	Transmit Channel Enable Enables the corresponding data channel for transmit operation. A channel must be enabled before its FIFO is accessed. 0 Transmit data channel N is disabled. 1 Transmit data channel N is enabled.
15–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
WDFL	Word Flag Configuration

Table continues on the next page...

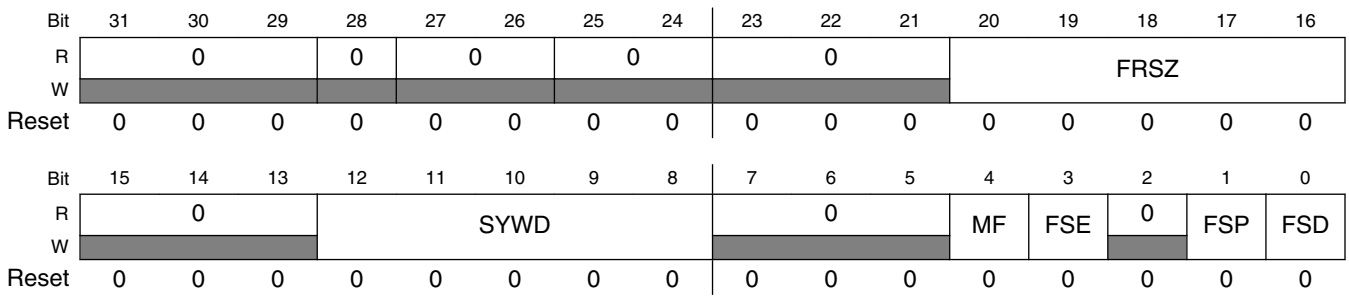
I2Sx_TCR3 field descriptions (continued)

Field	Description
	Configures which word sets the start of word flag. The value written must be one less than the word number. For example, writing 0 configures the first word in the frame. When configured to a value greater than TCR4[FRSZ], then the start of word flag is never set.

13.7.4.5 SAI Transmit Configuration 4 Register (I2Sx_TCR4)

This register must not be altered when TCSR[TE] is set.

Address: Base address + 10h offset



I2Sx_TCR4 field descriptions

Field	Description
31–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–16 FRSZ	Frame size Configures the number of words in each frame. The value written must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 32 words.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 SYWD	Sync Width Configures the length of the frame sync in number of bit clocks. The value written must be one less than the number of bit clocks. For example, write 0 for the frame sync to assert for one bit clock only. The sync width cannot be configured longer than the first word of the frame.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

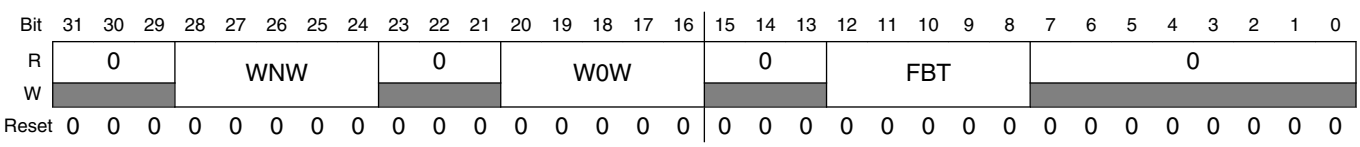
I2Sx_TCR4 field descriptions (continued)

Field	Description
4 MF	MSB First Configures whether the LSB or the MSB is transmitted first. 0 LSB is transmitted first. 1 MSB is transmitted first.
3 FSE	Frame Sync Early 0 Frame sync asserts with the first bit of the frame. 1 Frame sync asserts one bit before the first bit of the frame.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FSP	Frame Sync Polarity Configures the polarity of the frame sync. 0 Frame sync is active high. 1 Frame sync is active low.
0 FSD	Frame Sync Direction Configures the direction of the frame sync. 0 Frame sync is generated externally in Slave mode. 1 Frame sync is generated internally in Master mode.

13.7.4.6 SAI Transmit Configuration 5 Register (I2Sx_TCR5)

This register must not be altered when TCSR[TE] is set.

Address: Base address + 14h offset



I2Sx_TCR5 field descriptions

Field	Description
31–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–24 WNW	Word N Width Configures the number of bits in each word, for each word except the first in the frame. The value written must be one less than the number of bits per word. Word width of less than 8 bits is not supported.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

I2Sx_TCR5 field descriptions (continued)

Field	Description
20–16 W0W	Word 0 Width Configures the number of bits in the first word in each frame. The value written must be one less than the number of bits in the first word. Word width of less than 8 bits is not supported if there is only one word per frame.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 FBT	First Bit Shifted Configures the bit index for the first bit transmitted for each word in the frame. If configured for MSB First, the index of the next bit transmitted is one less than the current bit transmitted. If configured for LSB First, the index of the next bit transmitted is one more than the current bit transmitted. The value written must be greater than or equal to the word width when configured for MSB First. The value written must be less than or equal to 31-word width when configured for LSB First.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.7.4.7 SAI Transmit Data Register (I2Sx_TDR*n*)

Address: Base address + 20h offset + (4d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																															
W	TDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

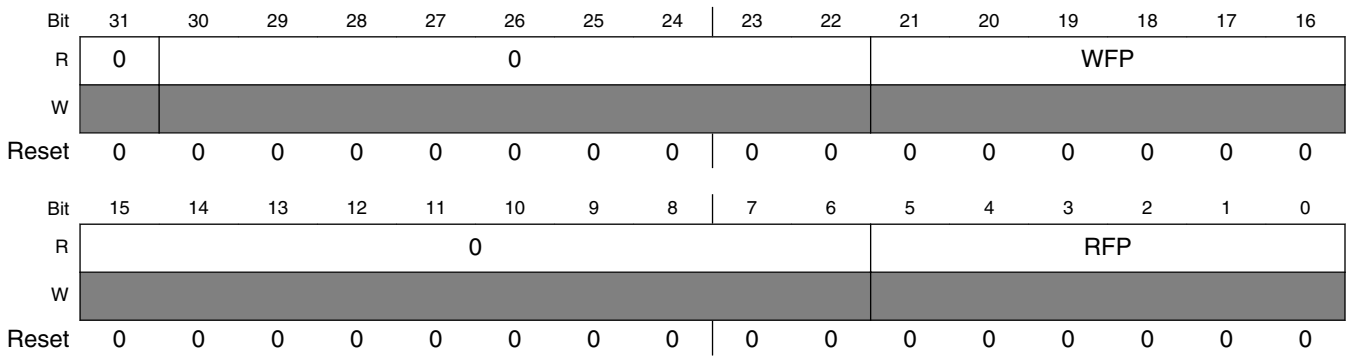
I2Sx_TDR*n* field descriptions

Field	Description
TDR	Transmit Data Register The corresponding TCR3[TCE] bit must be set before accessing the channel's transmit data register. Writes to this register when the transmit FIFO is not full will push the data written into the transmit data FIFO. Writes to this register when the transmit FIFO is full are ignored.

13.7.4.8 SAI Transmit FIFO Register (I2Sx_TFR*n*)

The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

Address: Base address + 40h offset + (4d × i), where i=0d to 0d



I2Sx_TFR*n* field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–16 WFP	Write FIFO Pointer FIFO write pointer for transmit data channel.
15–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RFP	Read FIFO Pointer FIFO read pointer for transmit data channel.

13.7.4.9 SAI Transmit Mask Register (I2Sx_TMR)

This register is double-buffered and updates:

1. When TCSR[TE] is first set
2. At the end of each frame.

This allows the masked words in each frame to change from frame to frame.

Address: Base address + 60h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TWM																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_TMR field descriptions

Field	Description
TWM	Transmit Word Mask Configures whether the transmit word is masked (transmit data pin tristated and transmit data not read from FIFO) for the corresponding word in the frame. 0 Word N is enabled. 1 Word N is masked. The transmit data pins are tri-stated when masked.

13.7.4.10 SAI Receive Control Register (I2Sx_RCSR)

Address: Base address + 80h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RE	STOPE	DBGE	BCE	0		0	SR	0			WSF	SEF	FEF	FWF	FRF
W							FR					w1c	w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			WSIE	SEIE	FEIE	FWIE	FRIE	0			0			FWDE	FRDE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_RCSR field descriptions

Field	Description
31 RE	Receiver Enable Enables/disables the receiver. When software clears this field, the receiver remains enabled, and this bit remains set, until the end of the current frame. 0 Receiver is disabled. 1 Receiver is enabled, or receiver has been disabled and has not yet reached end of frame.
30 STOPE	Stop Enable Configures receiver operation in Stop mode. This bit is ignored and the receiver is disabled in all stop modes. 0 Receiver disabled in Stop mode. 1 Receiver enabled in Stop mode.
29 DBGE	Debug Enable Enables/disables receiver operation in Debug mode. The receive bit clock is not affected by Debug mode. 0 Receiver is disabled in Debug mode, after completing the current frame. 1 Receiver is enabled in Debug mode.
28 BCE	Bit Clock Enable Enables the receive bit clock, separately from RE. This field is automatically set whenever RE is set. When software clears this field, the receive bit clock remains enabled, and this field remains set, until the end of the current frame. 0 Receive bit clock is disabled. 1 Receive bit clock is enabled.
27–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 FR	FIFO Reset Resets the FIFO pointers. Reading this field will always return zero. FIFO pointers should only be reset when the receiver is disabled or the FIFO error flag is set. 0 No effect. 1 FIFO reset.
24 SR	Software Reset Resets the internal receiver logic including the FIFO pointers. Software-visible registers are not affected, except for the status registers. 0 No effect. 1 Software reset.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 WSF	Word Start Flag Indicates that the start of the configured word has been detected. Write a logic 1 to this field to clear this flag.

Table continues on the next page...

I2Sx_RCSR field descriptions (continued)

Field	Description
	0 Start of word not detected. 1 Start of word detected.
19 SEF	Sync Error Flag Indicates that an error in the externally-generated frame sync has been detected. Write a logic 1 to this field to clear this flag. 0 Sync error not detected. 1 Frame sync error detected.
18 FEF	FIFO Error Flag Indicates that an enabled receive FIFO has overflowed. Write a logic 1 to this field to clear this flag. 0 Receive overflow not detected. 1 Receive overflow detected.
17 FWF	FIFO Warning Flag Indicates that an enabled receive FIFO is full. 0 No enabled receive FIFO is full. 1 Enabled receive FIFO is full.
16 FRF	FIFO Request Flag Indicates that the number of words in an enabled receive channel FIFO is greater than the receive FIFO watermark. 0 Receive FIFO watermark not reached. 1 Receive FIFO watermark has been reached.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 WSIE	Word Start Interrupt Enable Enables/disables word start interrupts. 0 Disables interrupt. 1 Enables interrupt.
11 SEIE	Sync Error Interrupt Enable Enables/disables sync error interrupts. 0 Disables interrupt. 1 Enables interrupt.
10 FEIE	FIFO Error Interrupt Enable Enables/disables FIFO error interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
9 FWIE	FIFO Warning Interrupt Enable Enables/disables FIFO warning interrupts.

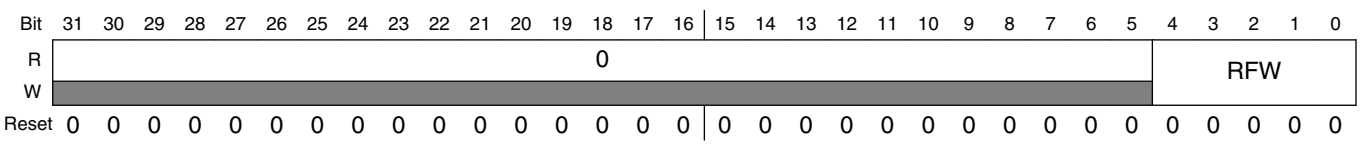
Table continues on the next page...

I2Sx_RCSR field descriptions (continued)

Field	Description
	0 Disables the interrupt. 1 Enables the interrupt.
8 FRIE	FIFO Request Interrupt Enable Enables/disables FIFO request interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FWDE	FIFO Warning DMA Enable Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.
0 FRDE	FIFO Request DMA Enable Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.

13.7.4.11 SAI Receive Configuration 1 Register (I2Sx_RCR1)

Address: Base address + 84h offset



I2Sx_RCR1 field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RFW	Receive FIFO Watermark Configures the watermark level for all enabled receiver channels.

13.7.4.12 SAI Receive Configuration 2 Register (I2Sx_RCR2)

This register must not be altered when RCSR[RE] is set.

Address: Base address + 88h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SYNC		BCS	BCI	MSEL		BCP	BCD	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_RCR2 field descriptions

Field	Description
31–30 SYNC	Synchronous Mode Configures between asynchronous and synchronous modes of operation. When configured for a synchronous mode of operation, the transmitter must be configured for asynchronous operation. 00 Asynchronous mode. 01 Synchronous with transmitter.
29 BCS	Bit Clock Swap This field swaps the bit clock used by the receiver. When the receiver is configured in asynchronous mode and this bit is set, the receiver is clocked by the transmitter bit clock (SAI_TX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the receiver continues to use the receiver frame sync (SAI_RX_SYNC). When the receiver is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the receiver bit clock (SAI_RX_BCLK) but use the transmitter frame sync (SAI_TX_SYNC). 0 Use the normal bit clock source. 1 Swap the bit clock source.
28 BCI	Bit Clock Input When this field is set and using an internally generated bit clock in either synchronous or asynchronous mode, the bit clock actually used by the receiver is delayed by the pad output delay (the receiver is clocked by the pad input as if the clock was externally generated). This has the effect of decreasing the data input setup time, but increasing the data output valid time. The slave mode timing from the datasheet should be used for the receiver when this bit is set. In synchronous mode, this bit allows the receiver to use the slave mode timing from the datasheet, while the transmitter uses the master mode timing. This field has no effect when configured for an externally generated bit clock . 0 No effect. 1 Internal logic is clocked as if bit clock was externally generated.
27–26 MSEL	MCLK Select

Table continues on the next page...

I2Sx_RCR2 field descriptions (continued)

Field	Description
	Selects the audio Master Clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock. **NOTE:** Depending on the device, some Master Clock options might not be available. See the chip-specific information for the availability and chip-specific meaning of each option. 00 Bus Clock selected. 01 Master Clock (MCLK) 1 option selected. 10 Master Clock (MCLK) 2 option selected. 11 Master Clock (MCLK) 3 option selected.
25 BCP	Bit Clock Polarity Configures the polarity of the bit clock. 0 Bit Clock is active high with drive outputs on rising edge and sample inputs on falling edge. 1 Bit Clock is active low with drive outputs on falling edge and sample inputs on rising edge.
24 BCD	Bit Clock Direction Configures the direction of the bit clock. 0 Bit clock is generated externally in Slave mode. 1 Bit clock is generated internally in Master mode.
23–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DIV	Bit Clock Divide Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2.

13.7.4.13 SAI Receive Configuration 3 Register (I2Sx_RCR3)

This register must not be altered when RCSR[RE] is set.

Address: Base address + 8Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								0							RCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0											WDFL				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_RCR3 field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 RCE	Receive Channel Enable Enables the corresponding data channel for receive operation. A channel must be enabled before its FIFO is accessed. 0 Receive data channel N is disabled. 1 Receive data channel N is enabled.
15–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
WDFL	Word Flag Configuration Configures which word the start of word flag is set. The value written should be one less than the word number (for example, write zero to configure for the first word in the frame). When configured to a value greater than the Frame Size field, then the start of word flag is never set.

13.7.4.14 SAI Receive Configuration 4 Register (I2Sx_RCR4)

This register must not be altered when RCSR[RE] is set.

Address: Base address + 90h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			0	0		0		0			FRSZ				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			SYWD					0			MF	FSE	0	FSP	FSD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_RCR4 field descriptions

Field	Description
31–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

I2Sx_RCR4 field descriptions (continued)

Field	Description
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–16 FRSZ	Frame Size Configures the number of words in each frame. The value written must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 32 words.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 SYWD	Sync Width Configures the length of the frame sync in number of bit clocks. The value written must be one less than the number of bit clocks. For example, write 0 for the frame sync to assert for one bit clock only. The sync width cannot be configured longer than the first word of the frame.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 MF	MSB First Configures whether the LSB or the MSB is received first. 0 LSB is received first. 1 MSB is received first.
3 FSE	Frame Sync Early 0 Frame sync asserts with the first bit of the frame. 1 Frame sync asserts one bit before the first bit of the frame.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FSP	Frame Sync Polarity Configures the polarity of the frame sync. 0 Frame sync is active high. 1 Frame sync is active low.
0 FSD	Frame Sync Direction Configures the direction of the frame sync. 0 Frame Sync is generated externally in Slave mode. 1 Frame Sync is generated internally in Master mode.

13.7.4.15 SAI Receive Configuration 5 Register (I2Sx_RCR5)

This register must not be altered when RCSR[RE] is set.

Address: Base address + 94h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			WNW					0			W0W					0			FBT					0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_RCR5 field descriptions

Field	Description
31–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–24 WNW	Word N Width Configures the number of bits in each word, for each word except the first in the frame. The value written must be one less than the number of bits per word. Word width of less than 8 bits is not supported.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–16 W0W	Word 0 Width Configures the number of bits in the first word in each frame. The value written must be one less than the number of bits in the first word. Word width of less than 8 bits is not supported if there is only one word per frame.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 FBT	First Bit Shifted Configures the bit index for the first bit received for each word in the frame. If configured for MSB First, the index of the next bit received is one less than the current bit received. If configured for LSB First, the index of the next bit received is one more than the current bit received. The value written must be greater than or equal to the word width when configured for MSB First. The value written must be less than or equal to 31-word width when configured for LSB First.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.7.4.16 SAI Receive Data Register (I2Sx_RDR*n*)

Reading this register introduces one additional peripheral clock wait state on each read.

Address: Base address + A0h offset + (4d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_RDRn field descriptions

Field	Description
RDR	Receive Data Register The corresponding RCR3[RCE] bit must be set before accessing the channel's receive data register. Reads from this register when the receive FIFO is not empty will return the data from the top of the receive FIFO. Reads from this register when the receive FIFO is empty are ignored.

13.7.4.17 SAI Receive FIFO Register (I2Sx_RFRn)

The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

Address: Base address + C0h offset + (4d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0										WFP					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0									RFP					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_RFRn field descriptions

Field	Description
31–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–16 WFP	Write FIFO Pointer FIFO write pointer for receive data channel.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RFP	Read FIFO Pointer FIFO read pointer for receive data channel.

13.7.4.18 SAI Receive Mask Register (I2Sx_RMR)

This register is double-buffered and updates:

1. When RCSR[RE] is first set
2. At the end of each frame

This allows the masked words in each frame to change from frame to frame.

Address: Base address + E0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RWM																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_RMR field descriptions

Field	Description
RWM	Receive Word Mask Configures whether the receive word is masked (received data ignored and not written to receive FIFO) for the corresponding word in the frame. 0 Word N is enabled. 1 Word N is masked.

13.8 Medium Quality Sound (MQS)

13.8.1 Overview

Medium quality sound (MQS) is used to generate medium quality audio via a standard GPIO in the pinmux, allowing the user to connect stereo speakers or headphones to a power amplifier without an additional DAC chip.

MQS accepts 2-channel, LSB-valid 16bit, MSB shift-out first; frame sync asserting with the first bit of the frame, data shifted with the posedge of bit clock, 44.1 kHz or 48 kHz signals from SAI1 in left_justified format; and it provides the SNR target as no more than 20dB for the signals below 10 kHz. The signals above 10 kHz will have worse THD+N values.

MQS provides only simple audio reproduction. No internal pop, click or distortion artifact reduction methods are provided.

13.8.2 Block Diagram

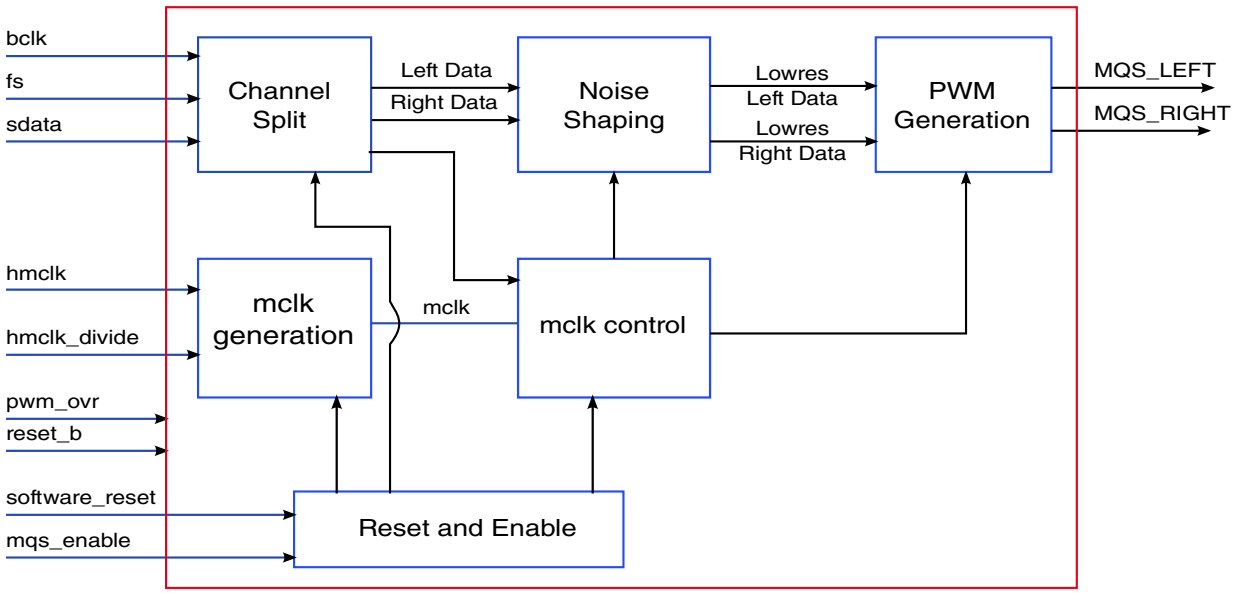


Figure 13-104. Block Diagram

MQS has the following sub-modules:

1. Channel Split: Splits the I2S signals into separate left channel and right channel audio data.
2. Noise Shaping: Uses the sigma-delta algorithm to generate low-resolution, very high sampling audio, while the audio sampling rate is increased.
3. PWM generation: Generates the bit stream to the GPIO, which is then used to drive the amplifier and then to drive the external speakers or headphones.
4. mclk generation: Used to generate the master clock (mclk). The frequency of mclk is determined by the final bit duration of PWM generation module.
5. mclk control:Used as a metronome to co-ordinate the different functional blocks working synchronously.
6. Reset and Enable: Used to generate the reset and enable logic to different clock domains.

13.8.3 External Signals

The following table describes the external signals of MQS:

Table 13-41. MQS External Signals

Signal	Description	Pad	Mode	Direction
MQS_LEFT	Left signal output	SAI1_RXC	ALT6	O
		SD2_CMD	ALT2	

Table continues on the next page...

Table 13-41. MQS External Signals (continued)

Signal	Description	Pad	Mode	Direction
MQS_RIGHT	Right signal output	SAI1_RXFS	ALT6	O
		SD2_CLK	ALT2	

13.8.4 Programmability

MQS has no internal programmable registers. But it does have some programmability from IOMUXC_GPR2.

Register Bits	Name	Description
IOMUXC_GPR2[26]	MQS_OVERSAMPLE	Used to control the PWM oversampling rate compared with mclk. 1—64, 0—32.
IOMUXC_GPR2[25]	MQS_EN	MQS enable. 1—Enable MQS, 0—Disable MQS
IOMUXC_GPR2[24]	MQS_SW_RST	MQS software reset. 1—Enable software reset for MQS 0—Exit software reset for MQS
IOMUXC_GPR2[23:16]	MQS_CLK_DIV[7:0]	Divider ration control for mclk from hmclk. 0—mclk frequency = hmclk frequency; 1—mclk frequency = ½*hmclk frequency; 2—mclk frequency = 1/3*hmclk frequency; …; n—mclk frequency = 1/(n+1)*hmclk frequency

13.8.5 Usage Model

The user needs to program SAI1 to output 2-channel LSB-16bit active left justified signal, and then to program the related IOMUXC_GPR2 bits.

Due to the different devices connected to MQS, and different high frequency behaviors of the connected analog circuits, customer need choose the appropriate MQS_CLK_DIV and MQS_OVERSAMPLE values for the best audible effects.

Chapter 14
ADC and Temperature Monitor

14.1 Analog-to-Digital Converter (ADC)

14.1.1 Overview

This block guide introduces the analog-to-digital converter (ADC). It discusses the architecture, functions and operating modes.

ADC is a 1.8 V 12-bit ADC with 16 channels analog input MUX and level-shifters for low-voltage digital interface. ADC supports conversion up to five logic groups (ChA / ChB / ChC / ChD / SW). Each group can select one channel from 0 - 15 physical channels.

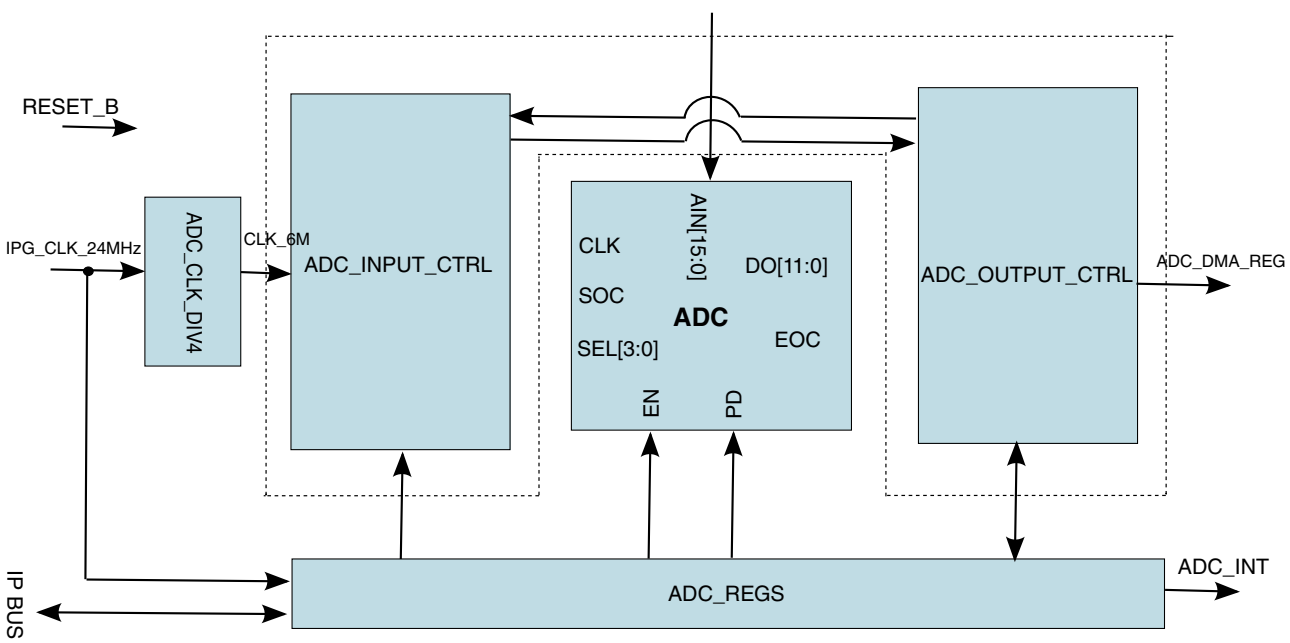


Figure 14-1. ADC Wrapper Hierarchy

14.1.1.1 Feature

The following list summarizes the key features of the ADC:

- Support up to 16 analog inputs
- Support five conversion pairs, can work simultaneously, with different conversion priority.
- Word size is 12-bits.
- Support Single and Continue conversion.
- Support Compare mode and channel auto disable if data match the requirement.
- Support Average conversion, Support flexible 4, 8, 16, 32 number of conversion data.
- Configurable sample time and conversion speed / power. The ADC core clock can vary from 300 kHz to 6 MHz, and the maximum sample rate is 1/6 ADC core clock.
- Conversion complete, hardware average complete, compare, DMA, time out flag and interrupt.
- Automatic compare with interrupt for less than, greater than, and equal to, within range, or out-of-range, programmable value.

14.1.2 Operation mode (ADC enable or disable)

By default, the ADC is in Disabled mode. In this state, no conversion or other actions occur. All of the ADC control registers are accessible in this state through an access bus interface. To enable the ADC, required configurations must be completed by programming the ADC configuration registers.

14.1.3 External signal description

The following table describes the external signals of ADC:

Table 14-1. ADC External Signals

Signal	Description	Pad	Mode	Direction
ADC1_IN0	Analog channel 1 input 0	ADC1_IN0	No muxing	I
ADC1_IN1	Analog channel 1 input 1	ADC1_IN1	No muxing	I
ADC1_IN2	Analog channel 1 input 2	ADC1_IN2	No muxing	I
ADC1_IN3	Analog channel 1 input 3	ADC1_IN3	No muxing	I
ADC1_VREFH	Voltage reference high	ADC1_VREFH	No muxing	I
ADC1_VREFL	Voltage reference low	ADC1_VREFL	No muxing	I

Table continues on the next page...

Table 14-1. ADC External Signals (continued)

Signal	Description	Pad	Mode	Direction
ADC2_IN0	Analog channel 2 input 0	ADC2_IN0	No muxing	I
ADC2_IN1	Analog channel 2 input 1	ADC2_IN1	No muxing	I
ADC2_IN2	Analog channel 2 input 2	ADC2_IN2	No muxing	I
ADC2_IN3	Analog channel 2 input 3	ADC2_IN3	No muxing	I
ADC2_VREFH	Voltage reference high	ADC2_VREFH	No muxing	I
ADC2_VREFL	Voltage reference low	ADC2_VREFL	No muxing	I

14.1.4 Clocks and timing

The following table describes the clock sources of ADC.

Table 14-2. ADC Clock Sources

Clock Name	Description
IPG_CLK_24MHZ	Global clock
IPG_CLK_24MHZ_S	Peripheral access clock

IPG_CLK_24MHZ can be divided by 4, 8, 16, 32, 64 or 128. Using the PRE_DIV bits to obtain the ADC analogue core clock. The ADC analogue core clock is vary from 300 kHz to 6 MHz.

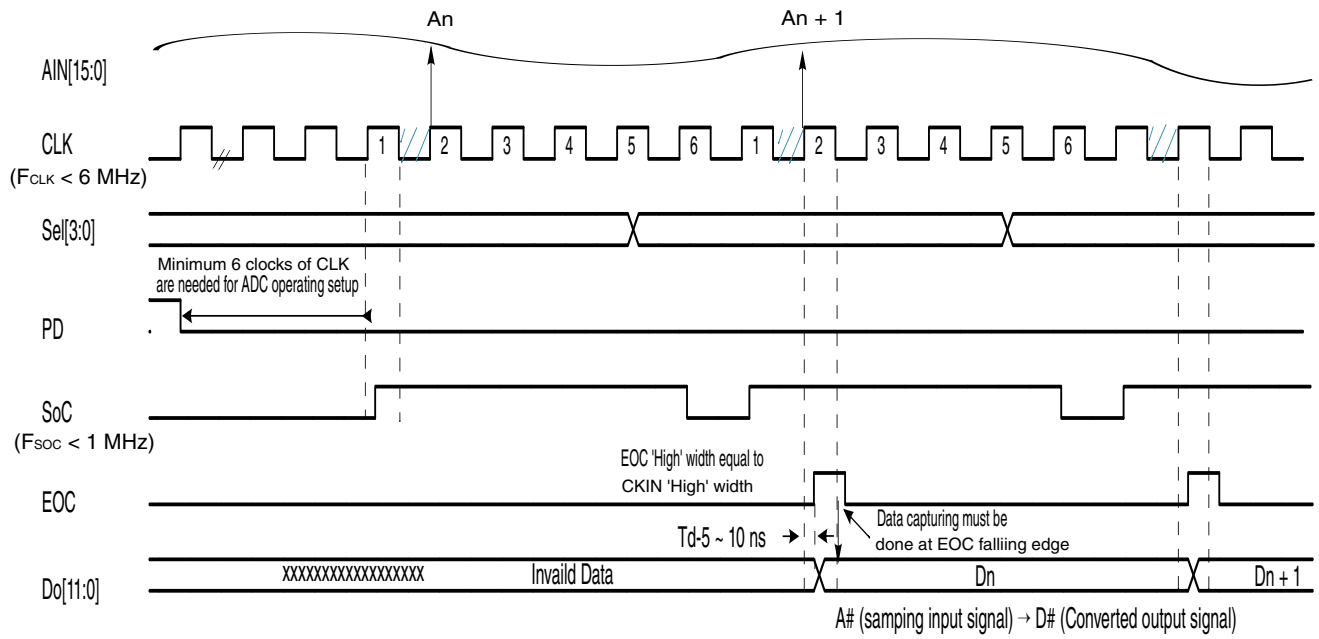


Figure 14-2. Timing

14.1.5 Functional description

This section provides a complete functional description of the ADC block.

14.1.5.1 Data FIFO

The size of ADC FIFO is 32 × 32-bit. The FIFO data are accessed by using the ADC FIFO Data Registers. Each data in the FIFO contains two ADC conversion data; each data is right aligned within the 12-bit wide.

When writing FIFO with an odd number of data, 0x0fff will be written into FIFO[31:16].

Table 14-3. Data FIFO

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

14.1.5.1.1 FIFO pointers

When writing of data into DMA_FIFO_DAT, the FIFO pointer increments one after every two valid write. Writing data of the DMA_FIFO_DAT will be ignored if FIFO is full.

When reading data from DMA_FIFO_DAT, the read FIFO pointer increments after each valid read. Reads from DMA_FIFO_DAT will be ignored if the FIFO is empty.

14.1.5.2 Interrupts and DMA requests

ADC generates interrupts or DMA requests when meet corresponding condition.

14.1.5.2.1 DMA requests

Only one channel can use DMA at a time. When more than one channel is enabled, it will be determined by its priority.

When the number of entries of ADC FIFO is less than or equal to the transmit FIFO watermark configuration, DMA request is generated. When the number of entries of FIFO is greater than FIFO watermark configuration, DMA request is cleared.

14.1.5.2.2 Flag and interrupt

Compare Flag

if enable the compare flag, the compare flag is set when conversion result matches the compare region. If enable compare interrupt, corresponding interrupt is generated.

Conversion Flag

If enable conversion flag, the Conversion Flag is set when one ADC conversion completed. If enable conversion interrupt, corresponding interrupt is generated.

Time Out Flag

If enable time out flag, the time out flag is set when conversion time out. If enable time out interrupt, corresponding interrupt is generated.

FIFO Overrun and Underrun

If enable fifo overrun or underrun flag, the fifo overrun or underrun flag is set when fifo overrun or underrun happens. If enable fifo overrun or underrun interrupt, corresponding interrupt is set.

14.1.5.2.3 Operating modes

Different modes can be programmed by several bits in the ADC register.

14.1.5.2.3.1 Single conversion

In Single conversion, ADC only starts a single conversion for enable channel. It stops when one conversion is finished.

14.1.5.2.3.2 Continue conversion

If continuous conversion is enabled, a new conversion is automatically initiated after the completion of the current conversion.

14.1.5.2.3.3 Average conversion

The hardware average function can be enabled (CH*_AVG_EN = 1) to perform a hardware average of multiple conversions. The number of conversions is determined by the CH*_AVG_NUMBER bit, which can average 4, 8, 16 or 32 conversions. If hardware averaging is enabled, a new conversion is automatically initiated after the completion of the current conversion until the correct number of conversions is completed.

If continuous conversion is also enabled, a new set of conversions (to be averaged) are initiated following the last of the selected number of conversions.

14.1.5.2.3.4 Compare mode

The compare function can be configured to check the result is less than, greater than, or equal to a single compare value, as well as the result falls within or outside a range determined by two compare values. The compare modes is determined by CH*_CMP_MODE. In compare mode, once one conversion is finished. ADC compares conversion with LOW_THRES and HIGH_THRES value to check whether the conversion result meet the requirement.

The following table shows six kinds of compare modes.

Table 14-4. Compare Modes

CH_*_CFG[31:29]	Compare Mode
X00	Disable compare
001	Compare true if the result is greater than **CHC_LOW_THRES**

Table continues on the next page...

Table 14-4. Compare Modes (continued)

CH_*_CFG[31:29]	Compare Mode
010	Compare true if the result is less than or equal to **CHC_LOW_THRES**
011	Compare true if the result is greater than **CHC_LOW_THRES** and less than **CHC_HIGH_THRES**
101	Compare true if the result is greater than or equal to **CHC_HIGH_THRES**
110	Compare true if the result is less than **CHC_HIGH_THRES**
111	Compare true if the result is less than or equal to **CHC_LOW_THRES**, or Compare true if the result is greater than or equal to **CHC_HIGH_THRES**

If compare mode and auto disable are enabled, the compare condition true can result in conversions discarded or conversion auto disabled.

14.1.5.2.3.5 Priority

When chA / chB / chC / chD / SW are all enabled, conflict may happen. To avoid this issue, ADC introduces priority mechanism. If chA / chB / chC / chD / SW start a conversion at the same time, chA will convert firstly, then ChB, ChC, chD, and SW at last. In worst condition, channel A blocks all chB / chC / chD / SW conversions.

The following table shows the priorities used in the conversion.

Table 14-5. Priority

Priority	Description
Highest	chA conversion
	chB conversion
	chC conversion
	chD conversion
Lowest	SW conversion

When ADC runs in Average and Priority mode simultaneously, ADC firstly consider the periodic, and then average time. When surplus timer between two chA conversion cannot complete a chB conversion, chB is blocked.

The same situation applies to other channels.

Constraints:

1. The conversion period of chA must be greater than total conversion time of chA + chB + chC + chD + SW. The conversion period of chB must be greater than total conversion time of chB + chC + chD + SW. The conversion period of channel with high priority must greater than total conversion time of low priority.

14.1.6 ADC Memory Map/Register Definition

ADC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3061_0000	Channel A configuration 1 (ADC1_CH_A_CFG1)	32	R/W	0000_0000h	14.1.6.1/ 4134
3061_0010	Channel A configuration 2 (ADC1_CH_A_CFG2)	32	R/W	0000_8000h	14.1.6.2/ 4136
3061_0020	ADC1_CH_B_CFG1	32	R/W	0000_0000h	14.1.6.3/ 4138
3061_0030	Channel B Configuration 2 (ADC1_CH_B_CFG2)	32	R/W	0000_8000h	14.1.6.4/ 4140
3061_0040	Channel C Configuration 1 (ADC1_CH_C_CFG1)	32	R/W	0000_0000h	14.1.6.5/ 4142
3061_0050	Channel C Configuration 2 (ADC1_CH_C_CFG2)	32	R/W	0000_8000h	14.1.6.6/ 4144
3061_0060	Channel D Configuration 1 (ADC1_CH_D_CFG1)	32	R/W	0000_0000h	14.1.6.7/ 4146
3061_0070	Channel D Configuration 2 (ADC1_CH_D_CFG2)	32	R/W	0000_8000h	14.1.6.8/ 4148
3061_0080	Channel Software Configuration (ADC1_CH_SW_CFG)	32	R/W	0000_0000h	14.1.6.9/ 4150
3061_0090	Timer Unit (ADC1_TIMER_UNIT)	32	R/W	0000_0000h	14.1.6.10/ 4151
3061_00A0	DMA FIFO (ADC1_DMA_FIFO)	32	R/W	0000_010Fh	14.1.6.11/ 4153
3061_00B0	FIFO Status (ADC1_FIFO_STATUS)	32	R/W	0000_0000h	14.1.6.12/ 4154
3061_00C0	ADC1_INT_SIG_EN	32	R/W	001F_1FCFh	14.1.6.13/ 4156
3061_00D0	Interrupt Enable (ADC1_INT_EN)	32	R/W	0000_00F0h	14.1.6.14/ 4160
3061_00E0	ADC1_INT_STATUS	32	R/W	0000_0000h	14.1.6.15/ 4164
3061_00F0	Channel A and B Conversion Result (ADC1_CHA_B_CNV_RSLT)	32	R/W	0000_0000h	14.1.6.16/ 4167
3061_0100	Channel C and D Conversion Result (ADC1_CHC_D_CNV_RSLT)	32	R/W	0000_0000h	14.1.6.17/ 4168
3061_0110	Channel Software Conversion Result (ADC1_CH_SW_CNV_RSLT)	32	R/W	0000_0000h	14.1.6.18/ 4168
3061_0120	DMA FIFO Data (ADC1_DMA_FIFO_DAT)	32	R/W	0000_0000h	14.1.6.19/ 4169
3061_0130	ADC Configuration (ADC1_ADC_CFG)	32	R/W	0000_0001h	14.1.6.20/ 4170

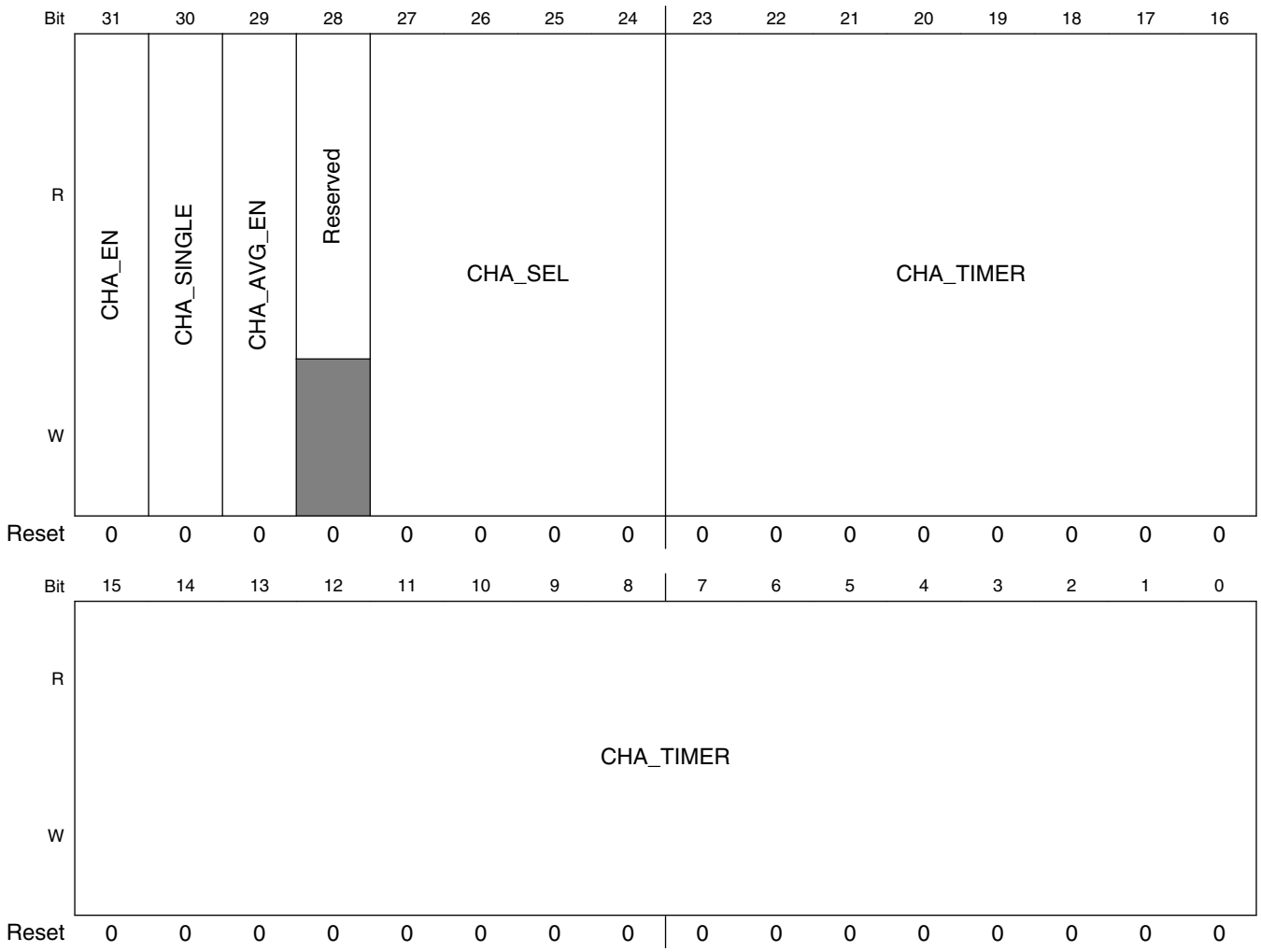
ADC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3062_0000	Channel A configuration 1 (ADC2_CH_A_CFG1)	32	R/W	0000_0000h	14.1.6.1/ 4134
3062_0010	Channel A configuration 2 (ADC2_CH_A_CFG2)	32	R/W	0000_8000h	14.1.6.2/ 4136
3062_0020	ADC2_CH_B_CFG1	32	R/W	0000_0000h	14.1.6.3/ 4138
3062_0030	Channel B Configuration 2 (ADC2_CH_B_CFG2)	32	R/W	0000_8000h	14.1.6.4/ 4140
3062_0040	Channel C Configuration 1 (ADC2_CH_C_CFG1)	32	R/W	0000_0000h	14.1.6.5/ 4142
3062_0050	Channel C Configuration 2 (ADC2_CH_C_CFG2)	32	R/W	0000_8000h	14.1.6.6/ 4144
3062_0060	Channel D Configuration 1 (ADC2_CH_D_CFG1)	32	R/W	0000_0000h	14.1.6.7/ 4146
3062_0070	Channel D Configuration 2 (ADC2_CH_D_CFG2)	32	R/W	0000_8000h	14.1.6.8/ 4148
3062_0080	Channel Software Configuration (ADC2_CH_SW_CFG)	32	R/W	0000_0000h	14.1.6.9/ 4150
3062_0090	Timer Unit (ADC2_TIMER_UNIT)	32	R/W	0000_0000h	14.1.6.10/ 4151
3062_00A0	DMA FIFO (ADC2_DMA_FIFO)	32	R/W	0000_010Fh	14.1.6.11/ 4153
3062_00B0	FIFO Status (ADC2_FIFO_STATUS)	32	R/W	0000_0000h	14.1.6.12/ 4154
3062_00C0	ADC2_INT_SIG_EN	32	R/W	001F_1FCFh	14.1.6.13/ 4156
3062_00D0	Interrupt Enable (ADC2_INT_EN)	32	R/W	0000_00F0h	14.1.6.14/ 4160
3062_00E0	ADC2_INT_STATUS	32	R/W	0000_0000h	14.1.6.15/ 4164
3062_00F0	Channel A and B Conversion Result (ADC2_CHA_B_CNV_RSLT)	32	R/W	0000_0000h	14.1.6.16/ 4167
3062_0100	Channel C and D Conversion Result (ADC2_CHC_D_CNV_RSLT)	32	R/W	0000_0000h	14.1.6.17/ 4168
3062_0110	Channel Software Conversion Result (ADC2_CH_SW_CNV_RSLT)	32	R/W	0000_0000h	14.1.6.18/ 4168
3062_0120	DMA FIFO Data (ADC2_DMA_FIFO_DAT)	32	R/W	0000_0000h	14.1.6.19/ 4169
3062_0130	ADC Configuration (ADC2_ADC_CFG)	32	R/W	0000_0001h	14.1.6.20/ 4170

14.1.6.1 Channel A configuration 1 (ADCx_CH_A_CFG1)

CH_A_CFG1 defines the functions of channel A. Channel A has the highest priority of all four channels.

Address: Base address + 0h offset



ADCx_CH_A_CFG1 field descriptions

Field	Description
31 CHA_EN	Channel A Enable This bit controls the working mode of logical channel A. 1 Enable logical channel A. Logical channel A will work. 0 Disable logical channel A. Prevent logical channel A from working.
30 CHA_SINGLE	Channel A Signal Conversion

Table continues on the next page...

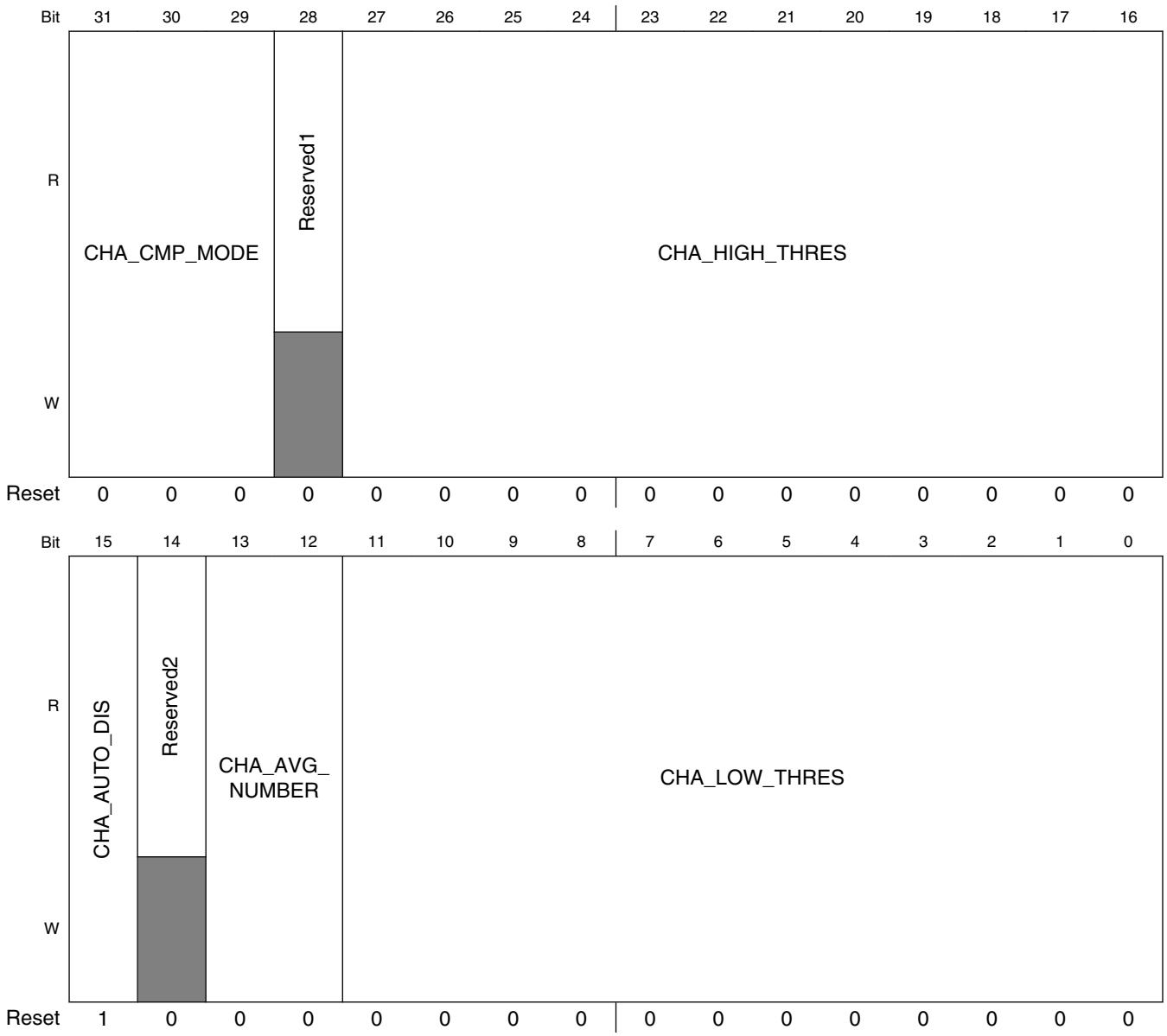
ADCx_CH_A_CFG1 field descriptions (continued)

Field	Description
	Start a single conversion from logical channel A. Switch between single and continuous conversion, it must disable CHA_EN first. When both CHA_EN and CHA_SINGLE are enabled, CHA_SIGNAL set CHA_EN to 0 and reset itself to 0. 1 Start a single conversion. Continuous conversion must stop. 0 No single conversion. Continuous conversion can start.
29 CHA_AVG_EN	Channel A Average Enable If enable this bit, channel A will do average for each conversion. 1 Enable average function. 0 Disable average function.
28 Reserved	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–24 CHA_SEL	Channel A Select Select input channel (from 16 channels) for the logical channel A. 0000 Channel 0 0001 Channel 1 0010 Channel 2 0011 Channel 3 0100 Channel 4 0101 Channel 5 0110 Channel 6 0111 Channel 7 1000 Channel 8 1001 Channel 9 1010 Channel 10 1011 Channel 11 1100 Channel 12 1101 Channel 13 1110 Channel 14 1111 Channel 15
CHA_TIMER	Channel A Timer This timer is for continuous conversion. The conversion rate = (CHA_TIMER + 1) times sample rate. 0x000 to 0xFFF timer value, every CHA_TIMER + 1 times sample rate, the ADC converts one data.

14.1.6.2 Channel A configuration 2 (ADCx_CH_A_CFG2)

CH_A_CFG2 defines the function of channel A. Channel A has the highest priority of all four channels.

Address: Base address + 10h offset



ADCx_CH_A_CFG2 field descriptions

Field	Description
31–29 CHA_CMP_ MODE	Channel A Compare Mode Select which compare mode the conversion result will take for channel A. X00 Disable compare function.

Table continues on the next page...

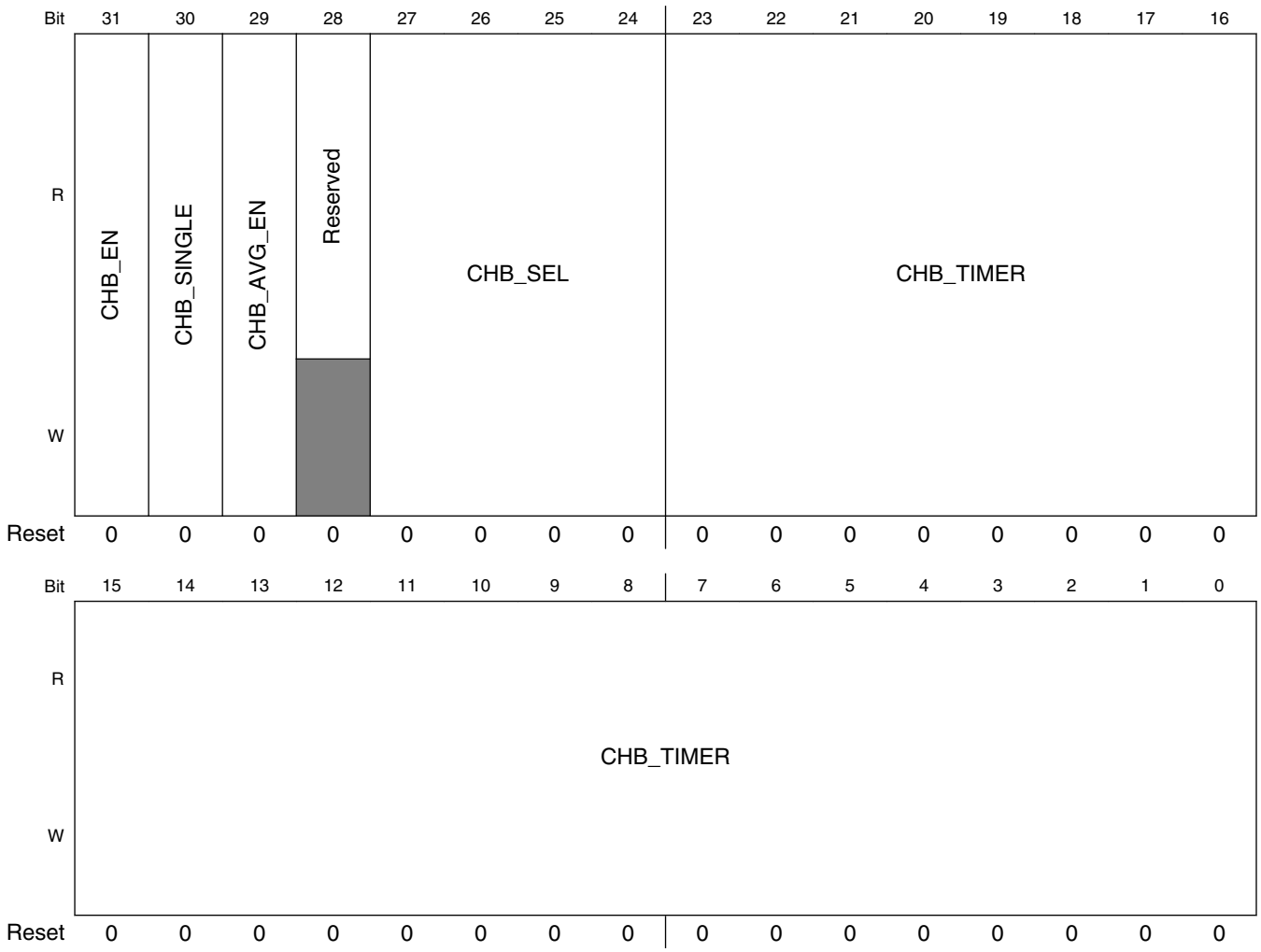
ADCx_CH_A_CFG2 field descriptions (continued)

Field	Description
	001 If channel A conversion result is greater than CHA_LOW_THRES, then an interrupt will be generated and the channel A flag will be set if they are enabled. 010 If channel A conversion result is less than or equal to CHA_LOW_THRES, then an interrupt will be generated and channel A flag will be set if they are enabled. 011 If channel A conversion result is greater than CHA_LOW_THRES and less than CHA_HIGH_THRES, then an interrupt will be generated and channel A flag will be set if they are enabled. 101 If channel A conversion result is greater than or equal to CHA_HIGH_THRES, then an interrupt will be generated and channel A flag will be set if they are enabled. 110 If channel A conversion result is less than CHA_HIGH_THRES, then an interrupt will be generated and channel A flag will be set if they are enabled. 111 If channel A conversion result is less than or equal to CHA_LOW_THRES, and at the same time, greater than or equal to CHA_HIGH_THRES, then an interrupt will be generated and channel A flag will be set if they are enabled.
28 Reserved1	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–16 CHA_HIGH_ THRES	Channel A High Threshold Value Contains the high compare value to compare with the conversion result when a compare mode has been selected.
15 CHA_AUTO_DIS	Channel A Auto Disable During continuous conversion, if compare match a converted result, the ADC checks this bit. If CHA_AUTO_DIS = 1, then channel A stops continuous conversion, if CHA_AUTO_DIS = 0, then channel A keeps continuous working, and the conversion result overwrites the result registers. The default value is 1.
14 Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
13–12 CHA_AVG_ NUMBER	Channel A Average Number If enable average function, this bit contains channel A average number. 00 Average number = 4 01 Average number = 8 10 Average number = 16 11 Average number = 32
CHA_LOW_ THRES	Channel A Low Threshold Value Contains the lower compare value to compare with the conversion result when a compare mode has been selected.

14.1.6.3 ADCx_CH_B_CFG1

CH_B_CFG1 defines the functions of channel B. Channel B has the second highest priority of all four channels.

Address: Base address + 20h offset



ADCx_CH_B_CFG1 field descriptions

Field	Description
31 CHB_EN	Channel B Enable Controls the usable of logical channel B. 1 Enable logical channel B. Logical channel B will work. 0 Disable logical channel B. Prevent logical channel B from working.
30 CHB_SINGLE	Channel B Single Conversion

Table continues on the next page...

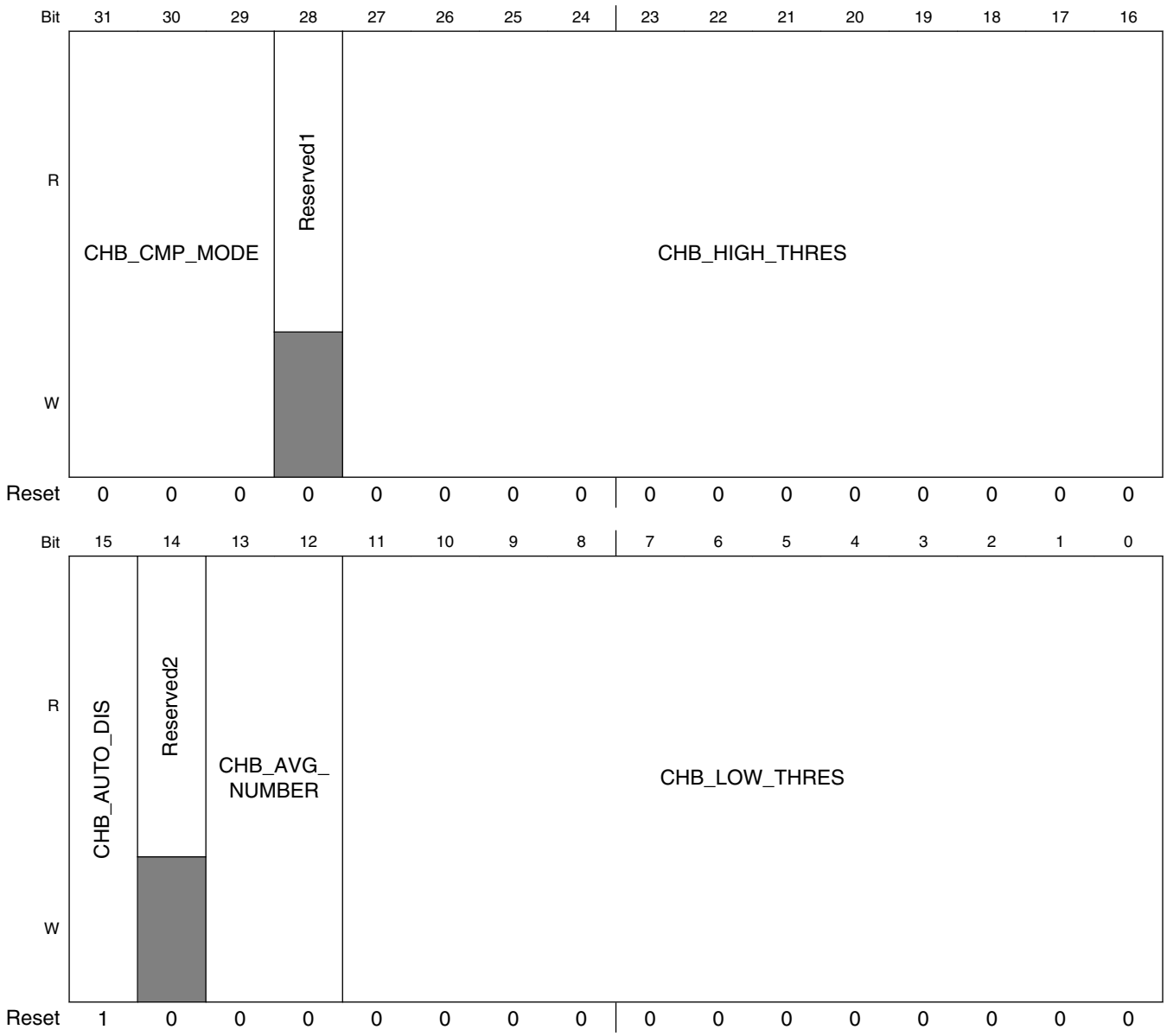
ADCx_CH_B_CFG1 field descriptions (continued)

Field	Description
	Start a single conversion from logical channel B. Switch between single and continuous conversion, it must disable CHB_EN first. When both CHB_EN and CHB_SINGLE are enabled, CHB_SIGNAL set CHB_EN to 0 and reset itself to 0. 1 Start a single conversion. Continuous conversion must stop. 0 No single conversion. Continuous conversion can start.
29 CHB_AVG_EN	Channel B Average Enable If enable this bit, channel B will do average for each conversion. 1 Enable average function. 0 Disable average function.
28 Reserved	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–24 CHB_SEL	Channel B Select Select input channel (from 16 channels) for the logical channel B. 0000 Channel 0 0001 Channel 1 0010 Channel 2 0011 Channel 3 0100 Channel 4 0101 Channel 5 0110 Channel 6 0111 Channel 7 1000 Channel 8 1001 Channel 9 1010 Channel 10 1011 Channel 11 1100 Channel 12 1101 Channel 13 1110 Channel 14 1111 Channel 15
CHB_TIMER	Channel B Timer This timer is for continuous conversion. The conversion rate = (CHB_TIMER + 1) times sample rate + offset. Offset is due to channel B obtains lower priority than channel A. In worst condition, all channel B's conversions are blocked by channel A's. 0x000 to 0xFFF timer value, every CHB_TIMER + 1 times sample rate + offset, the ADC converts one data.

14.1.6.4 Channel B Configuration 2 (ADCx_CH_B_CFG2)

CH_B_CFG2 defines the functions of channel B. Channel B has the second highest priority of all four channels.

Address: Base address + 30h offset



ADCx_CH_B_CFG2 field descriptions

Field	Description
31–29 CHB_CMP_ MODE	Channel B Compare Mode Select which compare mode the conversion result will take for channel B. X00 Disable compare function

Table continues on the next page...

ADCx_CH_B_CFG2 field descriptions (continued)

Field	Description
	001 If channel B conversion result is greater than CHB_LOW_THRES, then an interrupt will be generated and the channel B flag will be set if they are enabled. 010 If channel B conversion result is less than or equal to CHB_LOW_THRES, then an interrupt will be generated and channel B flag will be set if they are enabled. 011 If channel B conversion result is greater than CHB_LOW_THRES and less than CHB_HIGH_THRES, then an interrupt will be generated and channel B flag will be set if they are enabled. 101 If channel B conversion result is greater than or equal to CHB_HIGH_THRES, then an interrupt will be generated and channel B flag will be set if they are enabled. 110 If channel B conversion result is less than CHB_HIGH_THRES, then an interrupt will be generated and channel B flag will be set if they are enabled. 111 If channel B conversion result is less than or equal to CHB_LOW_THRES, and at the same time, is greater than or equal to CHB_HIGH_THRES, then an interrupt will be generated and channel B flag will be set.
28 Reserved1	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–16 CHB_HIGH_ THRES	Channel B High Threshold Value Contains the bigger compare value to compare with the conversion result when a compare mode has been selected.
15 CHB_AUTO_DIS	Channel B Auto Disable During continuous conversion, if compare match a converted result, the ADC checks this bit. If CHB_AUTO_DIS = 1, then channel B stops continuous conversion, if CHB_AUTO_DIS = 0, then channel B keeps continuous working, and the conversion result overwrites the result registers. The default value is 1.
14 Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
13–12 CHB_AVG_ NUMBER	Channel B Average Number If enable average function, this bit contains channel B average number. 00 Average number = 4 01 Average number = 8 10 Average number = 16 11 Average number = 32
CHB_LOW_ THRES	Channel B Low Threshold Value Contains the smaller compare value to compare with the conversion result when a compare mode has been selected.

14.1.6.5 Channel C Configuration 1 (ADCx_CH_C_CFG1)

Address: Base address + 40h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CHC_EN	CHC_SINGLE	CHC_AVG_EN	Reserved	CHC_SEL				CHC_TIMER							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CHC_TIMER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_CH_C_CFG1 field descriptions

Field	Description
31 CHC_EN	Channel C Enable Controls the usable of logical channel C. 1 Enable logical channel C. Logical channel C will work. 0 Disable logical channel C. Prevent logical channel C from working.
30 CHC_SINGLE	Channel C Single Conversion Start a single conversion from logical channel C. Switch between single and continuous conversion, it must disable CHC_EN. When both CHC_EN and CHC_SINGLE are enabled, CHC_SIGNAL set CHC_EN to 0 and reset itself to 0. Controls the usable of logical channel C. 1 Start a single conversion. Continuous conversion must stop. 0 No single conversion. Continuous conversion can start.
29 CHC_AVG_EN	Channel C Average Enable If enable this bit, channel C will do average for each conversion. 1 Enable average function. 0 Disable average function.
28 Reserved	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–24 CHC_SEL	Channel C Select Select input channel (from 16 channels) for the logical channel C.

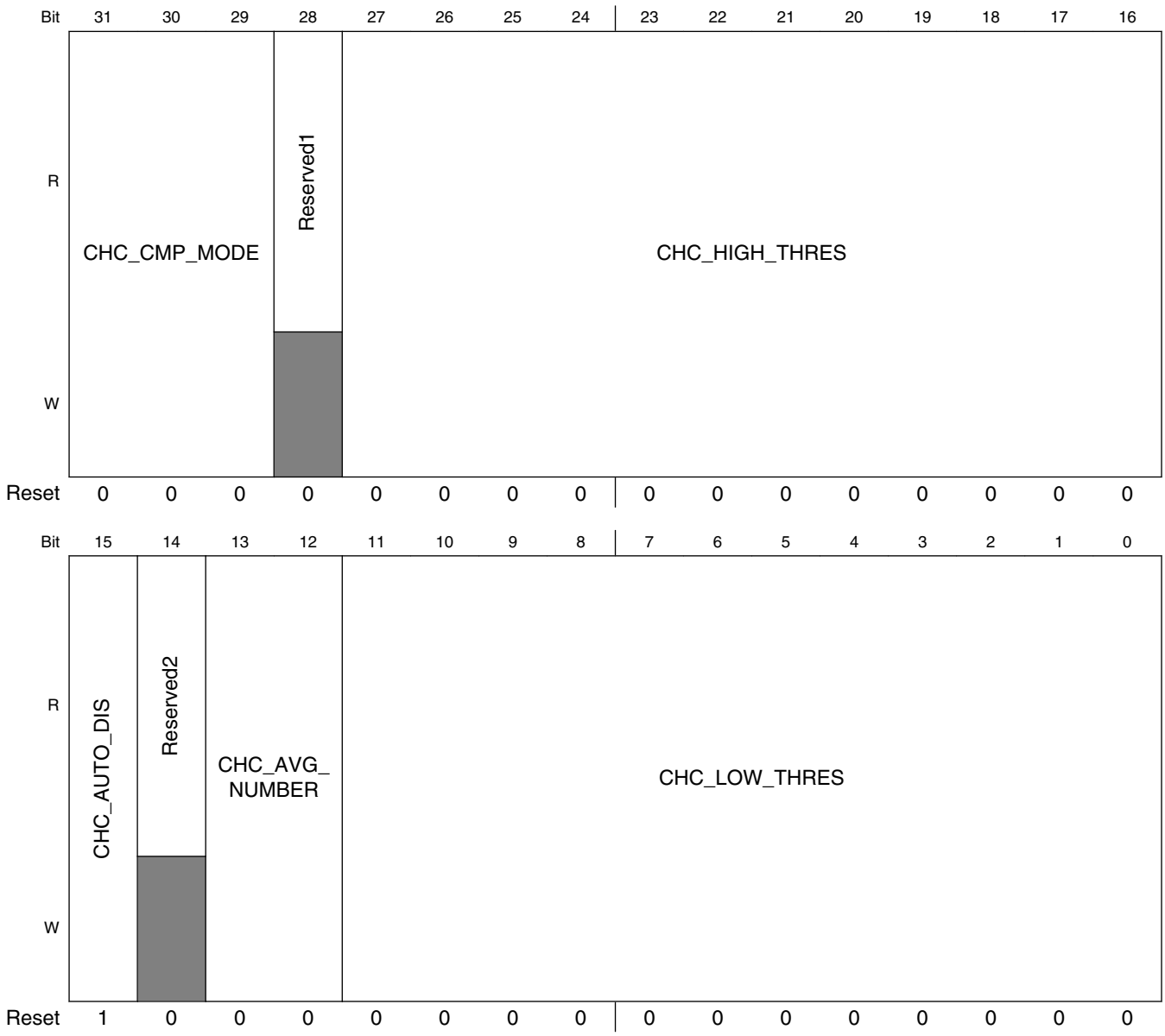
Table continues on the next page...

ADCx_CH_C_CFG1 field descriptions (continued)

Field	Description
	0000 Channel 0 0001 Channel 1 0010 Channel 2 0011 Channel 3 0100 Channel 4 0101 Channel 5 0110 Channel 6 0111 Channel 7 1000 Channel 8 1001 Channel 9 1010 Channel 10 1011 Channel 11 1100 Channel 12 1101 Channel 13 1110 Channel 14 1111 Channel 15
CHC_TIMER	Channel C Timer This timer is for continuous conversion. The conversion rate = (CHC_TIMER + 1) times sample rate + offset. The offset is due to channel C has lower priority than channel A and channel B. In worst condition, all channel C conversions are blocked by channel A and channel B's. 0x000 to 0xFFF timer value, every CHC_TIMER + 1 times sample rate + offset, the ADC converts one data.

14.1.6.6 Channel C Configuration 2 (ADCx_CH_C_CFG2)

Address: Base address + 50h offset



ADCx_CH_C_CFG2 field descriptions

Field	Description
31–29 CHC_CMP_ MODE	Channel C Compare Mode Select which compare mode the conversion result will take (for channel C). X00 Disable compare function. 001 If channel C conversion result bigger than CHC_LOW_THRES, then an interrupt will be generated and the channel C flag will be set if they are enabled.

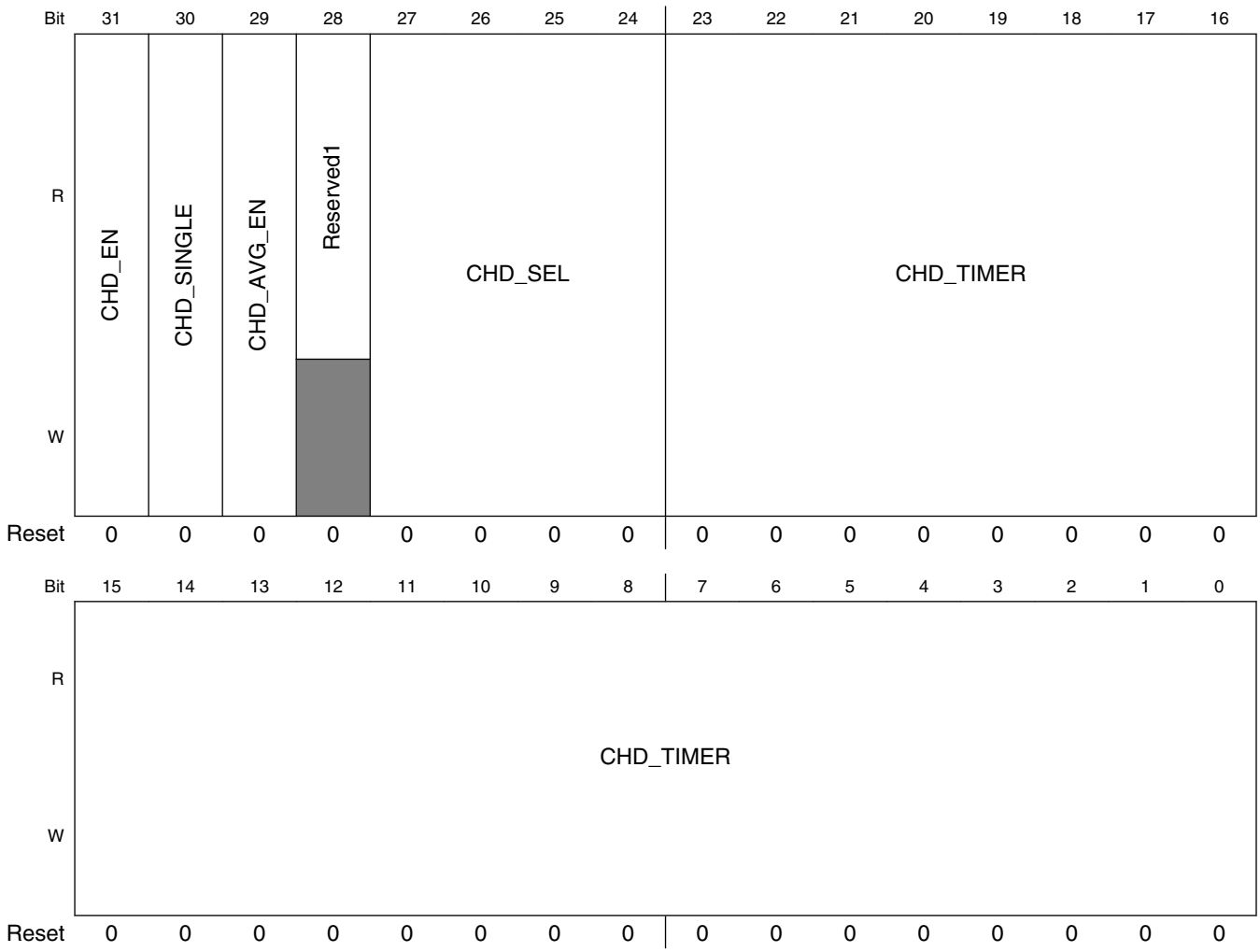
Table continues on the next page...

ADCx_CH_C_CFG2 field descriptions (continued)

Field	Description
	010 If channel C conversion result smaller or equal to CHC_LOW_THRES, then an interrupt will be generated and channel C flag will be set if they are enabled. 011 If channel C conversion result bigger than CHC_LOW_THRES and smaller than CHC_HIGH_THRES, then an interrupt will be generated and channel C flag will be set if they are enabled. 101 If channel C conversion result bigger or equal to CHC_HIGH_THRES, then an interrupt will be generated and channel C flag will be set if they are enabled. 110 If channel C conversion result smaller than CHC_HIGH_THRES, then an interrupt will be generated and channel C flag will be set if they are enabled. 111 If channel C conversion result smaller or equal to CHC_LOW_THRES, and at the same time, bigger or equal to CHC_HIGH_THRES, then an interrupt will be generated and channel C flag will be set if they are enabled.
28 Reserved1	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–16 CHC_HIGH_ THRES	Channel C High Threshold Value Contains the bigger compare value to compare with the conversion result when a compare mode has been selected.
15 CHC_AUTO_DIS	Channel C Auto Disable During continuous conversion, if compare match a converted result, the ADC checks this bit. If CHC_AUTO_DIS = 1, then channel C stops continuous conversion, if CHC_AUTO_DIS = 0, then channel C keeps continuous working, and the conversion result overwrites the result registers. The default value is 1.
14 Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
13–12 CHC_AVG_ NUMBER	Channel C Average Number If enable average function, this bit contains channel C average number. 00 Average number = 4 01 Average number = 8 10 Average number = 16 11 Average number = 32
CHC_LOW_ THRES	Channel C Low Threshold Value Contains the smaller compare value to compare with the conversion result when a compare mode has been selected.

14.1.6.7 Channel D Configuration 1 (ADCx_CH_D_CFG1)

Address: Base address + 60h offset



ADCx_CH_D_CFG1 field descriptions

Field	Description
31 CHD_EN	Channel D Enable Controls the usable of logical channel D. 1 Enable logical channel D. Logical channel D will work. 0 Disable logical channel D. Prevent logical channel D from working.
30 CHD_SINGLE	Channel D Single Conversion Start a single conversion from logical channel D. Switch between single and continuous conversion, it must disable channel D first. When both CHD_EN and CHD_SINGLE are enabled, CHD_SIGNAL set CHD_EN to 0 and reset itself to 0.

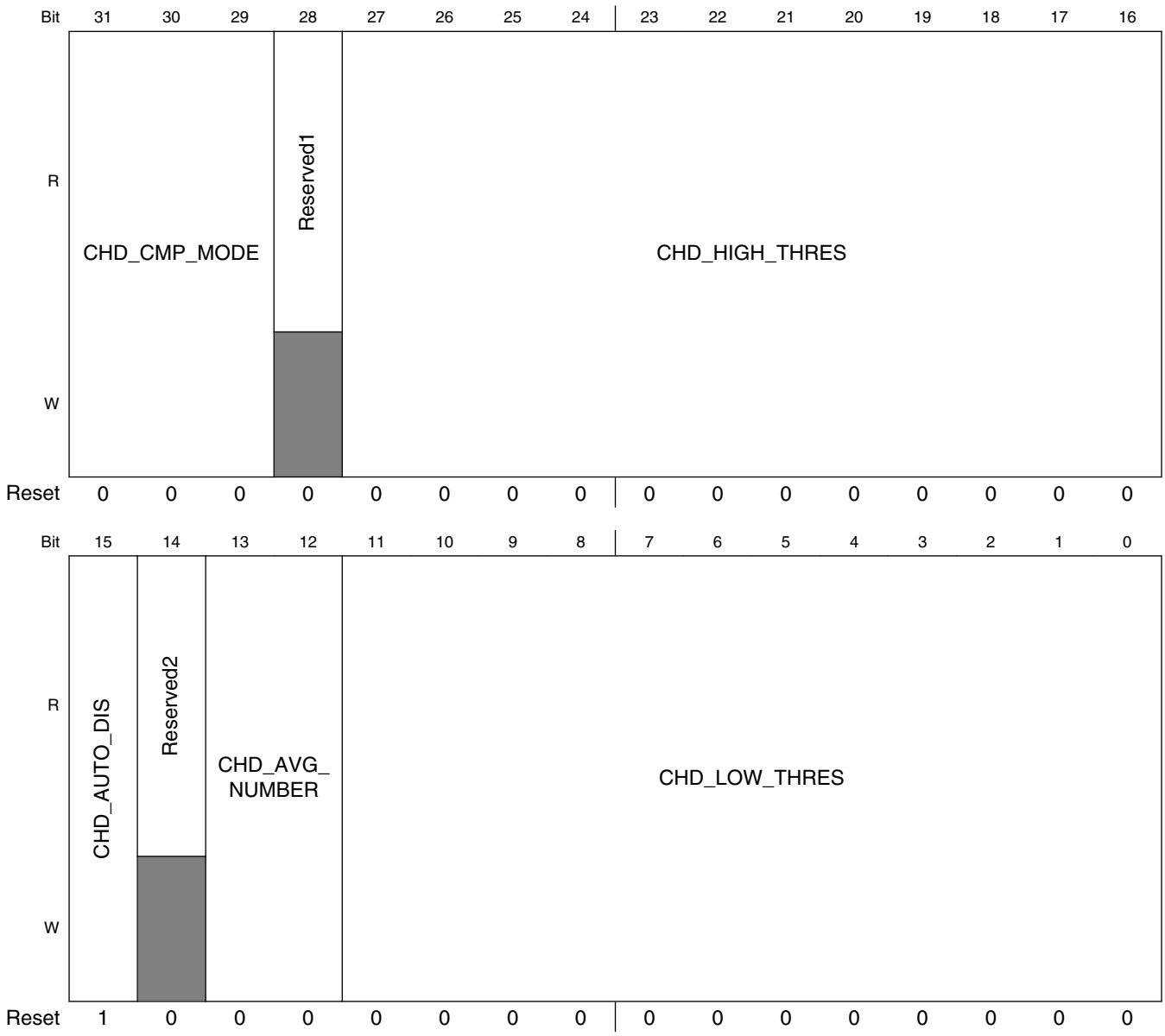
Table continues on the next page...

ADCx_CH_D_CFG1 field descriptions (continued)

Field	Description
	1 Start a single conversion. Continuous conversion must stop. 0 No single conversion. Continuous conversion can start.
29 CHD_AVG_EN	Channel D Average Enable If enable this bit, channel D will do average for each conversion. 1 Enable average function. 0 Disable average function.
28 Reserved1	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–24 CHD_SEL	Channel D Select Select input channel (from 16 channels) for the logical channel D. 0000 Channel 0 0001 Channel 1 0010 Channel 2 0011 Channel 3 0100 Channel 4 0101 Channel 5 0110 Channel 6 0111 Channel 7 1000 Channel 8 1001 Channel 9 1010 Channel 10 1011 Channel 11 1100 Channel 12 1101 Channel 13 1110 Channel 14 1111 Channel 15
CHD_TIMER	Channel D Timer This timer is for continuous conversion. The conversion rate = (CHD_TIMER + 1) times sample rate + offset. The offset is due to channel D has lower priority than Channel A, B, and C. In worst case, all channel D's conversion are blocked by channel A, channel B and channel C's. 0x000 to 0xFFF timer value, every CHD_TIMER + 1 times sample rate + offset, the ADC converts one data.

14.1.6.8 Channel D Configuration 2 (ADCx_CH_D_CFG2)

Address: Base address + 70h offset



ADCx_CH_D_CFG2 field descriptions

Field	Description
31–29 CHD_CMP_ MODE	Channel D Compare Mode Select which compare mode the conversion result will take (for channel D). X00 Disable compare function 001 If channel D conversion result bigger than CHD_LOW_THRES, then an interrupt will be generated and the channel D flag will be set if they are enabled.

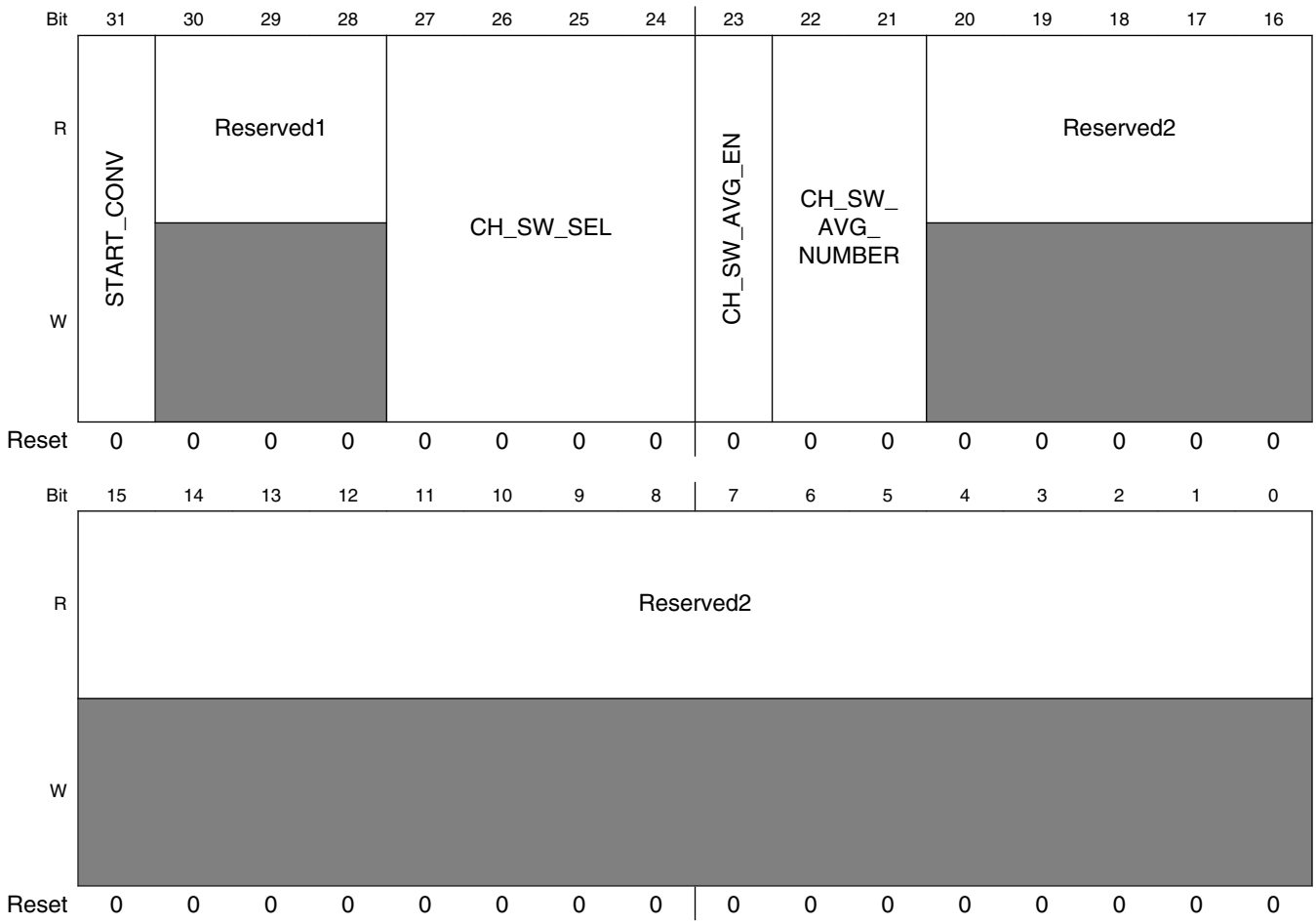
Table continues on the next page...

ADCx_CH_D_CFG2 field descriptions (continued)

Field	Description
	010 If channel D conversion result smaller or equal to CHD_LOW_THRES, then an interrupt will be generated and channel D flag will be set if they are enabled. 011 If channel D conversion result bigger than CHD_LOW_THRES and smaller than CHD_HIGH_THRES, then an interrupt will be generated and channel D flag will be set if they are enabled. 101 If channel D conversion result bigger or equal to CHD_HIGH_THRES, then an interrupt will be generated and channel D flag will be set if they are enabled. 110 If channel D conversion result smaller than CHD_HIGH_THRES, then an interrupt will be generated and channel D flag will be set if they are enabled. 111 If channel D conversion result smaller or equal to CHD_LOW_THRES, and at the same time, bigger or equal to CHD_HIGH_THRES, then an interrupt will be generated and channel D flag will be set if they are enabled.
28 Reserved1	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–16 CHD_HIGH_ THRES	Channel D High Threshold Value Contains the bigger compare value to compare with the conversion result when a compare mode has been selected.
15 CHD_AUTO_DIS	Channel D Auto Disable During continuous conversion, if compare match a converted result, the ADC checks this bit. If CHD_AUTO_DIS = 1, then channel D stops continuous conversion, if CHD_AUTO_DIS = 0, then channel D keeps continuous working, and the conversion result overwrites the result registers. The default value is 1.
14 Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
13–12 CHD_AVG_ NUMBER	Channel D Average Number If enable average function, this bit contains channel D average number. 00 Average number = 4 01 Average number = 8 10 Average number = 16 11 Average number = 32
CHD_LOW_ THRES	Channel D Low Threshold Value Contains the smaller compare value to compare with the conversion result when a compare mode has been selected.

14.1.6.9 Channel Software Configuration (ADCx_CH_SW_CFG)

Address: Base address + 80h offset



ADCx_CH_SW_CFG field descriptions

Field	Description
31 START_CONV	Start Software Trigger Conversion To start a software trigger, it must wait until the last software trigger finish. 0 Not start a new software trigger conversion. 1 Start a new software trigger conversion.
30–28 Reserved1	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–24 CH_SW_SEL	Software Trigger Channel Select Select 1 from 16 (physical) channels to be the software trigger channel. 0000 Channel 0

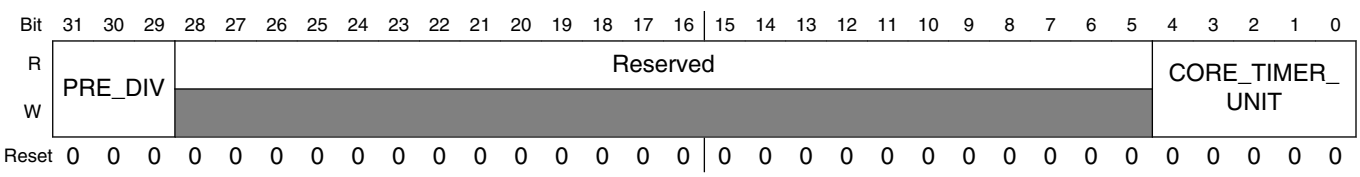
Table continues on the next page...

ADCx_CH_SW_CFG field descriptions (continued)

Field	Description
	0001 Channel 1 0010 Channel 2 0011 Channel 3 0100 Channel 4 0101 Channel 5 0110 Channel 6 0111 Channel 7 1000 Channel 8 1001 Channel 9 1010 Channel 10 1011 Channel 11 1100 Channel 12 1101 Channel 13 1110 Channel 14 1111 Channel 15
23 CH_SW_AVG_EN	Channel Software Average Enable If enable this bit, channel Software will do average for each conversion. 1 Enable average function. 0 Disable average function.
22–21 CH_SW_AVG_NUMBER	Channel Software Average Number If enable average function, this bit contains channel Software average number. 00 Average number = 4 01 Average number = 8 10 Average number = 16 11 Average number = 32
Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.

14.1.6.10 Timer Unit (ADCx_TIMER_UNIT)

Address: Base address + 90h offset

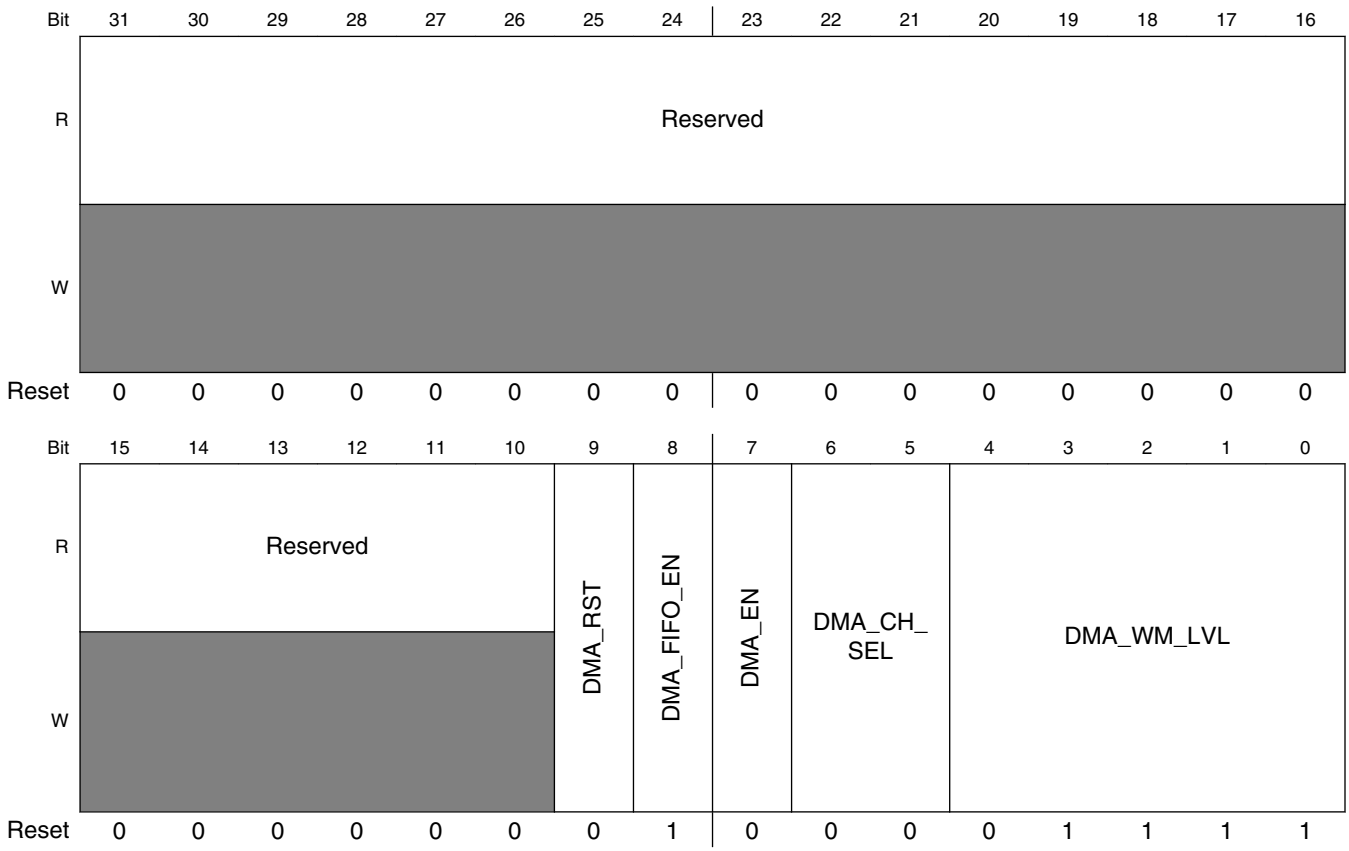


ADCx_TIMER_UNIT field descriptions

Field	Description
31–29 PRE_DIV	Pre-divide The ADC analogue core clock can vary from 300 kHz to 6 MHz. This parameter defines ADC core clock (and sample rate). The ADC analogue clock is default 1/4 of ADC digital input clock. All pre-divide is 1/4 ADC digital input clock. Warning: Before set this parameter, make sure analogue clock is from 300 kHz to 6 MHz. 000 No divide, analogue clock = 1/4 ADC digital input clock 001 Divide 2, analogue clock = 1/8 ADC digital input clock 010 Divide 4, analogue clock = 1/16 ADC digital input clock 011 Divide 8, analogue clock = 1/32 ADC digital input clock 100 Divide 16, analogue clock = 1/64 ADC digital input clock 101 Divide 32, analogue clock = 1/128 ADC digital input clock
28–5 Reserved	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
CORE_TIMER_ UNIT	Core_Timer_Unit This parameter defines sample rate. The maximum sample rate is 1/6 ADC analogue core clock. Sample rate = analogue core clock frequency (MHz) * 1 /((CORE_TIMER_UNIT + 1) * 6). If changing SAMPLE_RATE, please disable logical channel A, B, C, and D. Otherwise, the first A, B, C, and D conversion after change the sample rate may not follow exact sample rate interval.

14.1.6.11 DMA FIFO (ADC*x*_DMA_FIFO)

Address: Base address + A0h offset



ADC*x*_DMA_FIFO field descriptions

Field	Description
31–10 Reserved	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
9 DMA_RST	DMA Reset This bit contains reset information. If reset, the DMA and DMA FIFO return to its reset value. The DMA FIFO data is lost. 0 Not reset 1 Reset
8 DMA_FIFO_EN	DMA FIFO Enable If enable DMA_FIFO_EN, ADC conversion result can store into DMA FIFO. If disable, no ADC conversion result will store in DMA FIFO. 0 Disable DMA FIFO. 1 Enable DMA FIFO.

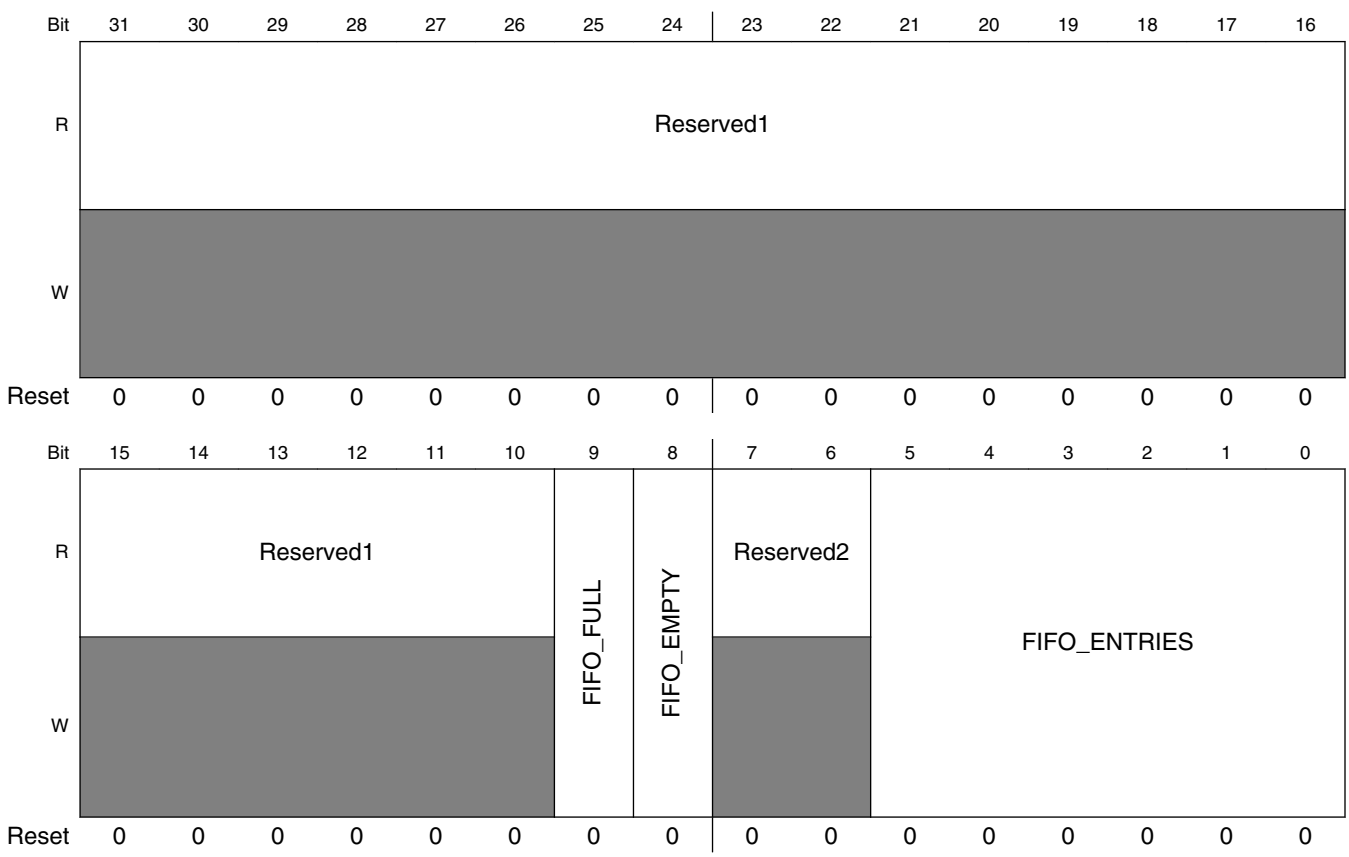
Table continues on the next page...

ADCx_DMA_FIFO field descriptions (continued)

Field	Description
7 DMA_EN	DMA Enable This bit defines data in DMA FIFO is moved by SDMA or moved by CPU. 0 Disable DMA, the data in DMA FIFO can only move by CPU. 1 Enable DMA, the data in DMA FIFO should move by SDMA.
6–5 DMA_CH_SEL	DMA Channel Select Select DMA FIFO source from Channel A, Channel B, Channel C, and Channel D. 00 Channel A 01 Channel B 10 Channel C 11 Channel D
DMA_WM_LVL	DMA Water Mark Level Once the DMA FIFO data reach water mark level, a DMA request or interrupt is generated (if enabled), and transfer data through DMA or CPU. 00000-11111 represents DMA water mark level value

14.1.6.12 FIFO Status (ADCx_FIFO_STATUS)

Address: Base address + B0h offset

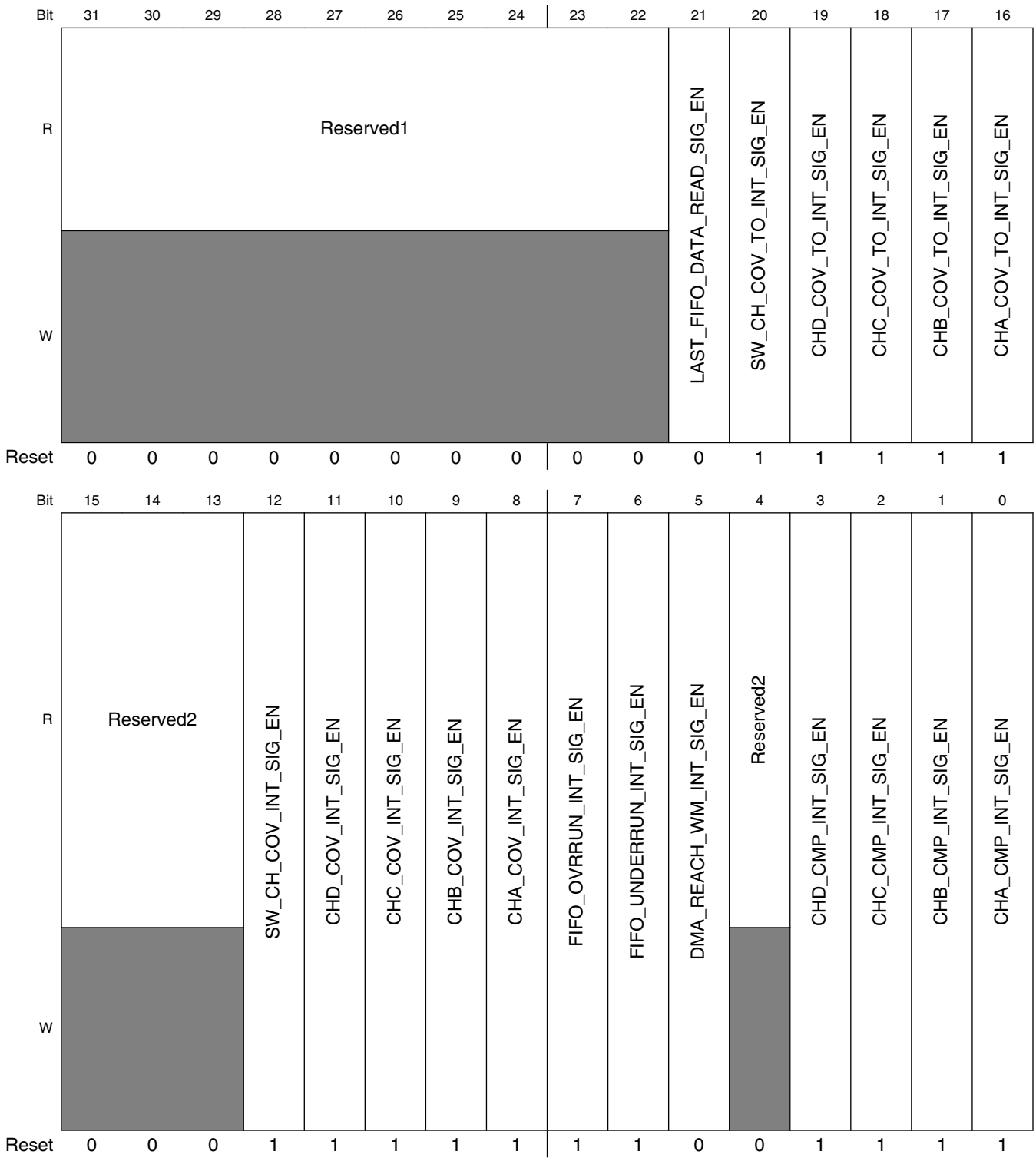


ADCx_FIFO_STATUS field descriptions

Field	Description
31–10 Reserved1	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
9 FIFO_FULL	This read-only bit represents the status of DMA FIFO. If FIFO is full, the DMA stop pushing data, set this bit, and generates an interrupt. 0 FIFO is not full. 1 FIFIO is full.
8 FIFO_EMPTY	FIFO Empty This read-only bit represents the status of DMA FIFO. If FIFO is empty, the DMA stops pop data, set this bit, and generates an interrupt. 0 FIFO is not empty. 1 FIFO is empty.
7–6 Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
FIFO_ENTRIES	FIFO Entries This read-only bit represents the numbers of data in DMA FIFO. 00000 to 11111 The numbers of data stored in DMA FIFO.

14.1.6.13 ADCx_INT_SIG_EN

Address: Base address + C0h offset



ADCx_INT_SIG_EN field descriptions

Field	Description
31–22 Reserved1	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
21 LAST_FIFO_ DATA_READ_ SIG_EN	Last FIFO Data Read Signal Enable Define whether enable flag when the last data read out of FIFO (the last of whole continuous result) 0 Disable last FIFO read signal enable 1 Enable last FIFO read signal enable
20 SW_CH_COV_ TO_INT_SIG_EN	Software Channel Conversion Time Out Interrupt Signal Enable Define whether software channel conversion time out interrupt signal is enabled. 0 Disable software channel conversion time out interrupt signal. 1 Enable software channel conversion time out interrupt signal.
19 CHD_COV_TO_ INT_SIG_EN	Channel D Conversion Time Out Interrupt Signal Enable Define whether logical channel D conversion time out interrupt signal is enabled. 0 Disable logical channel D conversion time out interrupt signal. 1 Enable logical channel D conversion time out interrupt signal.
18 CHC_COV_TO_ INT_SIG_EN	Channel C Conversion Time Out Interrupt Signal Enable Define whether logical channel C conversion time out interrupt signal is enabled. 0 Disable logical channel C conversion time out interrupt signal. 1 Enable logical channel C conversion time out interrupt signal.
17 CHB_COV_TO_ INT_SIG_EN	Channel B Conversion Time Out Interrupt Signal Enable Define whether logical channel B conversion time out interrupt signal is enabled. 0 Disable logical channel B conversion time out interrupt signal. 1 Enable logical channel B conversion time out interrupt signal.
16 CHA_COV_TO_ INT_SIG_EN	Channel A Conversion Time Out Interrupt Signal Enable Define whether logical channel A conversion time out interrupt signal is enabled. 0 Disable logical channel A conversion time out interrupt signal. 1 Enable logical channel A conversion time out interrupt signal.
15–13 Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12 SW_CH_COV_ INT_SIG_EN	Software Channel Conversion Interrupt Signal Enable Define whether software trigger interrupt flag is enabled. 0 Disable software channel conversion signal interrupt flag. 1 Enable software channel conversion signal interrupt flag.
11 CHD_COV_INT_ SIG_EN	Channel D Conversion Interrupt Signal Enable

Table continues on the next page...

ADCx_INT_SIG_EN field descriptions (continued)

Field	Description
	Define whether channel D conversion interrupt flag is enabled. If enable interrupt flag, an interrupt flag is set once Channel D finish a new conversion. 0 Disable channel D conversion Interrupt flag. 1 Enable channel D conversion Interrupt flag.
10 CHC_COV_INT_SIG_EN	Channel C Conversion Interrupt Signal Enable Define whether channel C conversion interrupt flag is enabled. If enable interrupt flag, an interrupt flag is set once Channel C finish a new conversion. 0 Disable channel C conversion Interrupt flag. 1 Enable channel C conversion Interrupt flag.
9 CHB_COV_INT_SIG_EN	Channel B Conversion Interrupt Signal Enable Define whether channel B conversion interrupt flag is enabled. If enable interrupt flag, an interrupt flag is generated once Channel B finish a new conversion. 0 Disable channel B conversion Interrupt flag. 1 Enable channel B conversion Interrupt flag.
8 CHA_COV_INT_SIG_EN	Channel A Conversion Interrupt Signal Enable Define whether channel A round conversion interrupt flag enable. If enable interrupt flag, an interrupt flag is generated once Channel A finish a new conversion. 0 Disable channel A conversion Interrupt flag. 1 Enable channel A conversion Interrupt flag.
7 FIFO_OVRRUN_INT_SIG_EN	FIFO overrun Interrupt Signal Enable Define whether DMA FIFO overrun interrupt flag is enabled. If DMA FIFO overrun interrupt flag enabled and DMA FIFO has overrun condition, then an interrupt flag is generated. 0 Enable FIFO Interrupt flag. 1 Disable FIFO Interrupt flag.
6 FIFO_UNDERRUN_INT_SIG_EN	FIFO Underrrun Interrupt Signal Enable Define whether DMA FIFO underrun interrupt flag is enabled. If DMA FIFO underrun interrupt flag is enabled and DMA FIFO has underrun condition, then an interrupt flag is generated. 0 Enable FIFO underrun Interrupt flag 1 Disable FIFO underrun Interrupt flag
5 DMA_REACH_WM_INT_SIG_EN	DMA Reach Watermark Level Interrupt Signal (Flag) Enable Define whether the number of data in DMA FIFO is greater than DMA watermark level. If it is greater, then a flag is set. 0 Enable DMA watermark level interrupt signal (flag). 1 Disable DMA watermark level interrupt signal (flag).
4 Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.

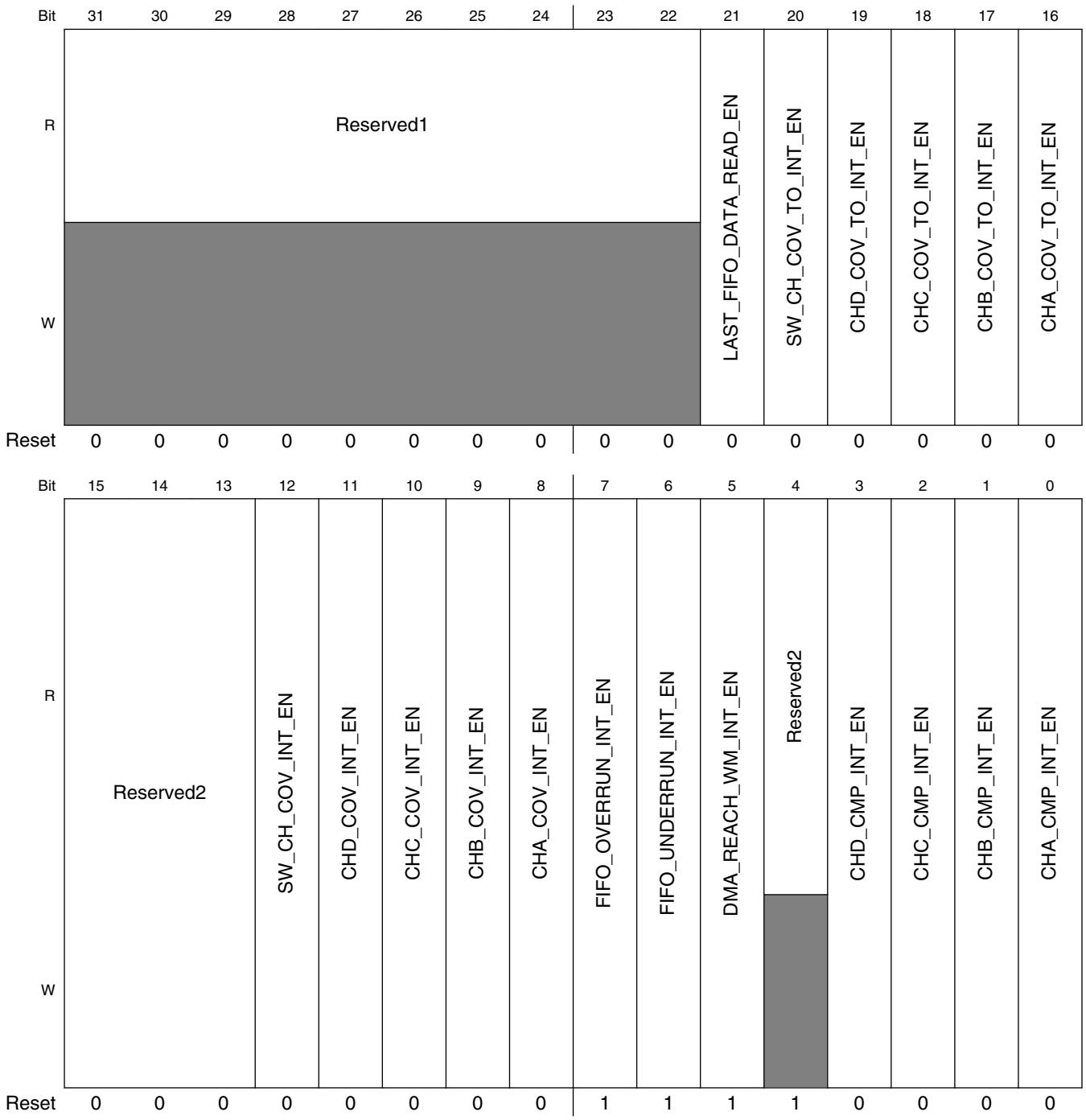
Table continues on the next page...

ADCx_INT_SIG_EN field descriptions (continued)

Field	Description
3 CHD_CMP_INT_SIG_EN	Channel D Compare Interrupt Signal Enable Define whether channel D compare interrupt flag is enabled. If enable the interrupt flag, an interrupt flag is set once channel D conversion result matches the compare region. 0 Disable Channel D Compare Interrupt flag 1 Enable Channel D Compare Interrupt flag
2 CHC_CMP_INT_SIG_EN	Channel C Compare Interrupt Signal Enable Define whether channel C compare interrupt flag is enabled. If enable the interrupt flag, an interrupt flag is set once channel C conversion result matches the compare region. 0 Disable Channel C Compare Interrupt flag. 1 Enable Channel C Compare Interrupt flag.
1 CHB_CMP_INT_SIG_EN	Channel B Compare Interrupt Signal Enable Define whether channel B compare interrupt flag is enabled. If enable the interrupt flag, an interrupt flag is set once channel B conversion result matches the compare region. 0 Disable Channel B Compare Interrupt flag. 1 Enable Channel B Compare Interrupt flag.
0 CHA_CMP_INT_SIG_EN	Channel A Compare Interrupt Signal Enable Define whether channel A compare interrupt flag is enabled. If enable the interrupt flag, an interrupt is set once channel A conversion result matches the compare region. 0 Disable Channel A Compare Interrupt flag. 1 Enable Channel A Compare Interrupt flag.

14.1.6.14 Interrupt Enable (ADCx_INT_EN)

Address: Base address + D0h offset



ADCx_INT_EN field descriptions

Field	Description
31–22 Reserved1	This field is reserved.

Table continues on the next page...

ADCx_INT_EN field descriptions (continued)

Field	Description
	This field is reserved. This read-only field is reserved and always has the value 0.
21 LAST_FIFO_ DATA_READ_EN	Last FIFO Data Read Enable Define whether generate an interrupt when the last data read out of FIFO (the last of whole continuous result) 0 Disable last FIFO read enable 1 Enable last FIFO read enable
20 SW_CH_COV_ TO_INT_EN	Software Channel Conversion Time Out Interrupt Enable Define whether software channel conversion time out interrupt is enabled. 0 Disable software channel conversion time out interrupt. 1 Enable software channel conversion time out interrupt.
19 CHD_COV_TO_ INT_EN	Channel D Conversion Time Out Interrupt Enable Define whether logical channel D conversion time out interrupt is enabled. 0 Disable logical channel D conversion time out interrupt. 1 Enable logical channel D conversion time out interrupt.
18 CHC_COV_TO_ INT_EN	Channel C Conversion Time Out Interrupt Enable Define whether logical channel C conversion time out interrupt is enabled. 0 Disable logical channel C conversion time out interrupt. 1 Enable logical channel C conversion time out interrupt.
17 CHB_COV_TO_ INT_EN	Channel B Conversion Time Out Interrupt Enable Define whether logical channel B conversion time out interrupt is enabled. 0 Disable logical channel B conversion time out interrupt. 1 Enable logical channel B conversion time out interrupt.
16 CHA_COV_TO_ INT_EN	Channel A Conversion Time Out Interrupt Enable Define whether logical channel A conversion time out interrupt is enabled. 0 Disable logical channel A conversion time out interrupt. 1 Enable logical channel A conversion time out interrupt.
15–13 Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12 SW_CH_COV_ INT_EN	Software Channel Conversion Interrupt Enable Define whether software trigger interrupt is enabled. 0 Disable software channel conversion interrupt. 1 Enable software channel conversion interrupt.
11 CHD_COV_INT_ EN	Channel D Conversion Interrupt Enable Define whether channel D conversion interrupt is enabled. If enable interrupt, an interrupt is generated once Channel D finish a new conversion.

Table continues on the next page...

ADCx_INT_EN field descriptions (continued)

Field	Description
	0 Disable channel D conversion Interrupt. 1 Enable channel D conversion Interrupt.
10 CHC_COV_INT_EN	Channel C Conversion Interrupt Enable Define whether channel C conversion interrupt is enabled. If enable interrupt, an interrupt is generated once Channel C finish a new conversion. 0 Disable channel C conversion Interrupt. 1 Enable channel C conversion Interrupt.
9 CHB_COV_INT_EN	Channel B Conversion Interrupt Enable Define whether channel B conversion interrupt is enabled. If enable interrupt, an interrupt is generated once Channel B finish a new conversion. 0 Disable channel B conversion Interrupt. 1 Enable channel B conversion Interrupt.
8 CHA_COV_INT_EN	Channel A Conversion Interrupt Enable Define whether channel A conversion interrupt is enabled. If enable interrupt, an interrupt is generated once Channel A finish a new conversion. 0 Disable channel A conversion Interrupt. 1 Enable channel A conversion Interrupt.
7 FIFO_OVERRUN_INT_EN	FIFO overrun Interrupt Enable Define whether DMA FIFO overrun interrupt is enabled. If DMA FIFO overrun interrupt is enabled and DMA FIFO has overrun condition, then an interrupt is generated. 0 Enable FIFO Interrupt. 1 Disable FIFO Interrupt.
6 FIFO_UNDERRUN_INT_EN	FIFO underrun Interrupt Enable Define whether DMA FIFO underrun interrupt is enabled. If DMA FIFO underrun interrupt is enabled and DMA FIFO has underrun condition, then an interrupt is generated. 0 Enable FIFO underrun Interrupt. 1 Disable FIFO underrun Interrupt.
5 DMA_REACH_WM_INT_EN	DMA Reach Watermark Level Interrupt Enable Define whether DMA watermark level interrupt is enabled. If enable the interrupt, an interrupt is generated. **NOTE:** This signal is for software read DMA FIFO. If SDMA read data, turn off both interrupt enable and signal (flag) enable. 0 Enable DMA reach watermark level interrupt. 1 Disable DMA reach watermark level interrupt.
4 Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.

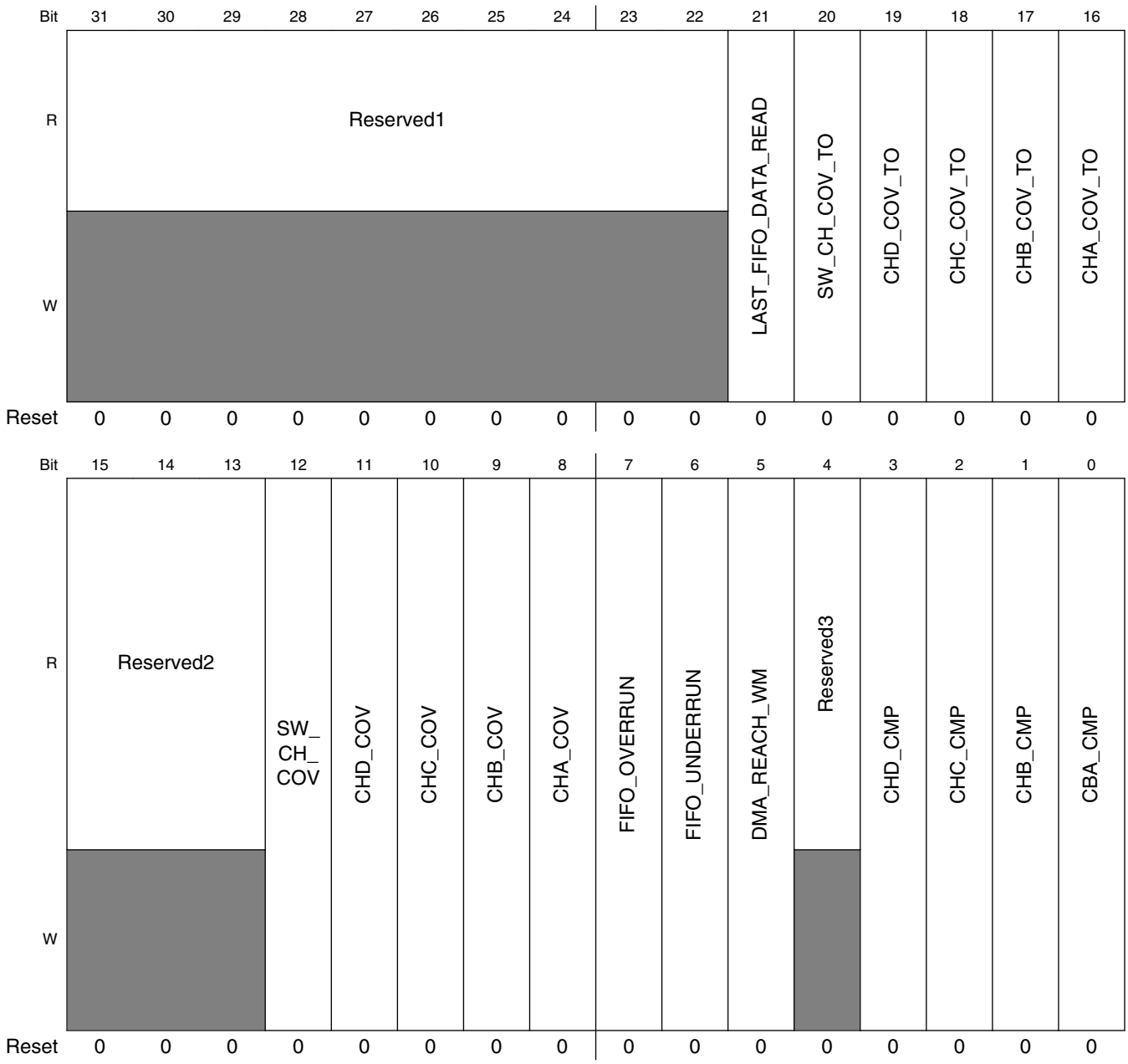
Table continues on the next page...

ADCx_INT_EN field descriptions (continued)

Field	Description
3 CHD_CMP_INT_EN	Channel D Compare Interrupt Enable Define whether channel D compare interrupt is enabled. If enable the interrupt, an interrupt is set once channel D conversion result matches the compare region. 0 Disable Channel D Compare Interrupt. 1 Enable Channel D Compare Interrupt.
2 CHC_CMP_INT_EN	Channel C Compare Interrupt Enable Define whether channel C compare interrupt is enabled. If enable the interrupt, an interrupt is set once channel C conversion result matches the compare region. 0 Disable Channel C Compare Interrupt. 1 Enable Channel C Compare Interrupt .
1 CHB_CMP_INT_EN	Channel B Compare Interrupt Enable Define whether channel B compare interrupt is enabled. If enable the interrupt, an interrupt is set once channel B conversion result matches the compare region. 0 Disable Channel B Compare Interrupt. 1 Enable Channel B Compare Interrupt.
0 CHA_CMP_INT_EN	Channel A Compare Interrupt Enable Define whether channel A compare interrupt is enabled. If enable the interrupt, an interrupt is set once channel A conversion result matches the compare region. 0 Disable Channel A Compare Interrupt. 1 Enable Channel A Compare Interrupt.

14.1.6.15 ADCx_INT_STATUS

Address: Base address + E0h offset



ADCx_INT_STATUS field descriptions

Field	Description
31–22 Reserved1	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ADCx_INT_STATUS field descriptions (continued)

Field	Description
21 LAST_FIFO_ DATA_READ	Last FIFO Data Read Represents whether the last data read out of FIFO (the last of whole continuous result) 0 Last FIFO data has not been read out 1 Last FIFO data has been read out
20 SW_CH_COV_ TO	Software Channel Conversion Time Out Represents whether software channel conversion time out exist. 0 Software channel conversion time out does not exist. 1 Exist software channel conversion time out. Software clear / clear signal enable.
19 CHD_COV_TO	Channel D Conversion Time Out Represent whether logical channel D conversion time out exist. 0 Logical channel D conversion time out does not exist. 1 Exist logical channel D conversion time out. Software clear / clear signal enable.
18 CHC_COV_TO	Channel C Conversion Time Out Represent whether logical channel C conversion time out exist. 0 Logical channel C conversion time out does not exist. 1 Exist logical channel C conversion time out. Software clear / clear signal enable.
17 CHB_COV_TO	Channel B Conversion Time Out Represent whether logical channel B conversion time out exist. 0 Logical channel B conversion time out exist. 1 Exist logical channel B conversion time out. Software clear/clear signal enable.
16 CHA_COV_TO	Channel A Conversion Time Out Represent whether logical channel A conversion time exist. 0 Logical channel A conversion time out does not exist. 1 Exist logical channel A conversion time out. Software clear / clear signal enable.
15–13 Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12 SW_CH_COV	Software Channel Conversion (Flag) This is the flag of software trigger. If SW_CH_COV_INT_SIG_EN is enabled and software trigger ADC conversion is finished, the SW_CH_COV (flag) is set. 0 Disable software channel conversion (Flag). 1 Enable software channel conversion (Flag).
11 CHD_COV	Channel D Conversion (Flag) This is the flag of channel D conversion. If CHD_COV_INT_SIG_EN is enabled and channel D ADC conversion is finished, the CHD_COV (flag) is set.

Table continues on the next page...

ADCx_INT_STATUS field descriptions (continued)

Field	Description
	0 Disable channel D conversion (Flag). 1 Enable channel D conversion (Flag).
10 CHC_COV	Channel C Conversion (Flag) This is the flag of channel C conversion. If CHC_COV_INT_SIG_EN is enabled and channel C ADC conversion is finished, the CHC_COV (flag) is set. 0 Disable channel C conversion (Flag). 1 Enable channel C conversion (Flag).
9 CHB_COV	Channel B Conversion (Flag) This is the flag of channel B conversion. If CHB_COV_INT_SIG_EN is enabled and channel B ADC conversion is finished, the CHB_COV (flag) is set. 0 Disable channel B conversion (Flag). 1 Enable channel B conversion (Flag).
8 CHA_COV	Channel A Conversion (Flag) This is the flag of channel A conversion. If CHA_COV_INT_SIG_EN is enabled and channel A ADC conversion is finished, the CHA_CONV (flag) is set. 0 Disable channel A conversion (Flag). 1 Enable channel A conversion (Flag).
7 FIFO_OVERRUN	FIFO Overrun (Flag) This is the flag of FIFO overrun. If FIFO_OVERRUN_INT_SIG_EN is enabled and FIFO overrun happens, the FIFO_OVERRUN (flag) is set. 0 Disable FIFO overrun (Flag). 1 Enable FIFO overrun (Flag).
6 FIFO_ UNDERRUN	FIFO Underrun (Flag) This is the flag of FIFO underrun. If FIFO_UNDERRUN_INT_SIG_EN is enabled and FIFO underrun happens, the FIFO_UNDERRUN (flag) is set. 0 Do not exist FIFO underrun (Flag). 1 Exist FIFO underrun (Flag). Software clear / clear signal enable.
5 DMA_REACH_ WM	DMA Reach Watermark Level (Flag) This is the flag of DMA reach watermark level. Setting this flag indicates the number of data in the DMA FIFO reach the watermark level. **NOTE:** This bit is for software read DMA FIFO, if SDMA read DMA FIFO, please disable both interrupt and interrupt signal (flag). 0 The numbers of data in DMA FIFO has not reach the watermark level. 1 The numbers of data in DMA FIFO has reach the watermark level. Software clear / clear signal enable.
4 Reserved3	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ADCx_INT_STATUS field descriptions (continued)

Field	Description
3 CHD_CMP	Channel D Compare (Flag) This is the flag of channel D compare. If CHD_CMP_INT_SIG_EN is enabled and channel D conversion result matches the compare region, the CHD_CMP (Flag) is set. 0 Disable Channel D Compare (Flag). 1 Enable Channel D Compare (Flag).
2 CHC_CMP	Channel C Compare (Flag) This is the flag of channel C compare. If CHC_CMP_INT_SIG_EN is enabled and channel C conversion result matches the compare region, the CHC_CMP (Flag) is set. 0 Disable Channel C Compare (Flag). 1 Enable Channel C Compare (Flag).
1 CHB_CMP	Channel B Compare (Flag) This is the flag of channel B compare. If CHB_CMP_INT_SIG_EN is enabled and channel B conversion result matches the compare region, the CHB_CMP (Flag) is set. 0 Disable Channel B Compare (Flag). 1 Enable Channel B Compare (Flag).
0 CBA_CMP	Channel A Compare (Flag) This is the flag of channel A compare. If CHA_CMP_INT_SIG_EN is enabled and channel A conversion result matches the compare region, the CHA_CMP (Flag) is set. 0 Disable Channel A Compare (Flag). 1 Enable Channel A Compare (Flag).

14.1.6.16 Channel A and B Conversion Result (ADCx_CHA_B_CNV_RSLT)

Address: Base address + F0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved1				CHB_CNV_RSLT												Reserved2				CHA_CNV_RSLT											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_CHA_B_CNV_RSLT field descriptions

Field	Description
31–28 Reserved1	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–16 CHB_CNV_RSLT	Channel B Conversion Result Channel B conversion result stores in this field.

Table continues on the next page...

ADCx_CHA_B_CNV_RSLT field descriptions (continued)

Field	Description
15–12 Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
CHA_CNV_RSLT	Channel A Conversion Result Channel A conversion result stores in this field.

14.1.6.17 Channel C and D Conversion Result (ADCx_CHC_D_CNV_RSLT)

Address: Base address + 100h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved1				CHD_CNV_RSLT												Reserved2				CHC_CNV_RSLT												
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_CHC_D_CNV_RSLT field descriptions

Field	Description
31–28 Reserved1	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–16 CHD_CNV_ RSLT	Channel D Conversion Result Channel D conversion result stores in this field.
15–12 Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
CHC_CNV_ RSLT	Channel C Conversion Result Channel C conversion result stores in this field.

14.1.6.18 Channel Software Conversion Result (ADCx_CH_SW_CNV_RSLT)

Address: Base address + 110h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																				CH_SW_CNV_RSLT												
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_CH_SW_CNV_RSLT field descriptions

Field	Description
31–12 Reserved	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
CH_SW_CNV_ RSLT	Channel Software Conversion Result Channel Software trigger conversion result stores in this field.

14.1.6.19 DMA FIFO Data (ADCx_DMA_FIFO_DAT)

Address: Base address + 120h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DAT2_FLAG		Reserved1		DMA_FIFO_1											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DAT1_FLAG		Reserved2		DMA_FIFO_0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_DMA_FIFO_DAT field descriptions

Field	Description
31–30 DAT2_FLAG	Data 2 Flag The flag defines the status of data 2. 00 Default 01 Valid Data. Indicate the data 2 is a valid data. 10 Last Data. Indicate the last the data of a continuous batch. 11 Invalid Data. Indicate the data 2 is a invalid data.
29–28 Reserved1	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–16 DMA_FIFO_1	The even number of data is in this field. This field is read-only
15–14 DAT1_FLAG	Data 1 Flag The flag defines the status of data 1. 00 Default 01 Valid Data. Indicate the data 1 is a valid data. 10 Last Data. Indicate the last the data of a continuous batch. 11 Invalid Data. Indicate the data 1 is a invalid data.

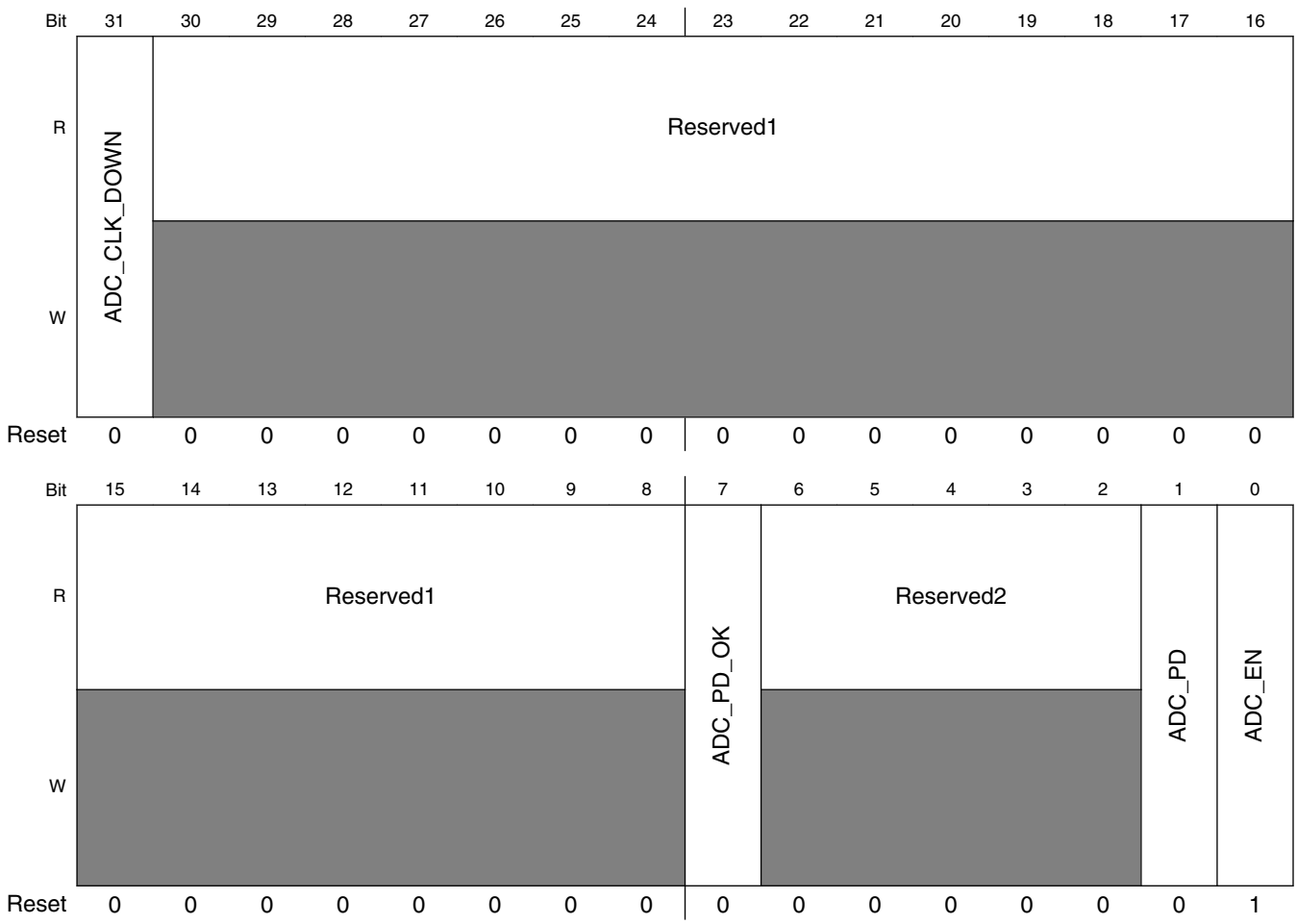
Table continues on the next page...

ADCx_DMA_FIFO_DAT field descriptions (continued)

Field	Description
13–12 Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
DMA_FIFO_0	The even number of data is in this field. This field is read-only

14.1.6.20 ADC Configuration (ADCx_ADC_CFG)

Address: Base address + 130h offset



ADCx_ADC_CFG field descriptions

Field	Description
31 ADC_CLK_ DOWN	ADC Clock Down This bit is for low power design. Set this bit to stop all digital part power. Warning: Before set this bit, all conversion must be finished.

Table continues on the next page...

ADCx_ADC_CFG field descriptions (continued)

Field	Description
	0 Clock running. 1 Clock down, no clock.
30–8 Reserved1	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
7 ADC_PD_OK	ADC Power Down OK This bit represents whether power down or to start conversion. After power up the ADC analogue core, system has to wait ADC_PD_OK = 1 before start a conversion or power down again. If violate this rule, the ADC may enter an unknown state. Clock cannot stop until this bit equals to 0. This bit will automatically set to 1 if ADC_PD = 0 and ADC_EN = 0 for the specific time period. 0 ADC has not power up completely. Cannot start a conversion. 1 ADC power up completely. Can start a conversion.
6–2 Reserved2	This field is reserved. This field is reserved. This read-only field is reserved and always has the value 0.
1 ADC_PD	ADC Power Down Power Down ADC analogue core. Before entering into stop-mode, power down ADC analogue core first. 0 Do not power down the ADC analogue core. 1 Power down the ADC analogue core.
0 ADC_EN	ADC Level Shifter Enable The level shifter provides 1.8 V to 1.1 V converter. The level shifter is embedded in ADC analogue. The ADC analogue ports voltage level must same with ADC digital part ports voltage level. During low power mode, also reset this bit to 1'b0 to save power. 0 Disable level shifter, the ADC analogue core ports using 1.8 V voltage level. 1 Enable level shifter, the ADC analogue core ports using 1.1 V voltage level.

14.2 Temperature Monitor (TEMPMON)

14.2.1 Overview

The temperature sensor module implements a temperature sensor/conversion function based on a temperature-dependent voltage to time conversion.

The high-level implementation of the temperature sensor is shown in the figure below.

Figure 14-3. High Level Temp Sensor System Diagram

As shown in the figure above, the temperature sensor uses and assumes that the bandgap reference, 480MHz PLL and 32KHz RTC modules are properly programmed and fully settled for correct operation.

14.2.2 Software Usage Guidelines

During normal system operation software can use the temperature sensor counter output (TEMP_CNT) in conjunction with the fused temperature calibration data to determine the on-die operational temperature or to set an over-temperature interrupt alarm to within a couple of °C.

Based on calibration, two sets of temperature and counter values will be available via fuses on the device. These data points will correspond to the points (N_1, T_1) and (N_2, T_2) in the curve below.

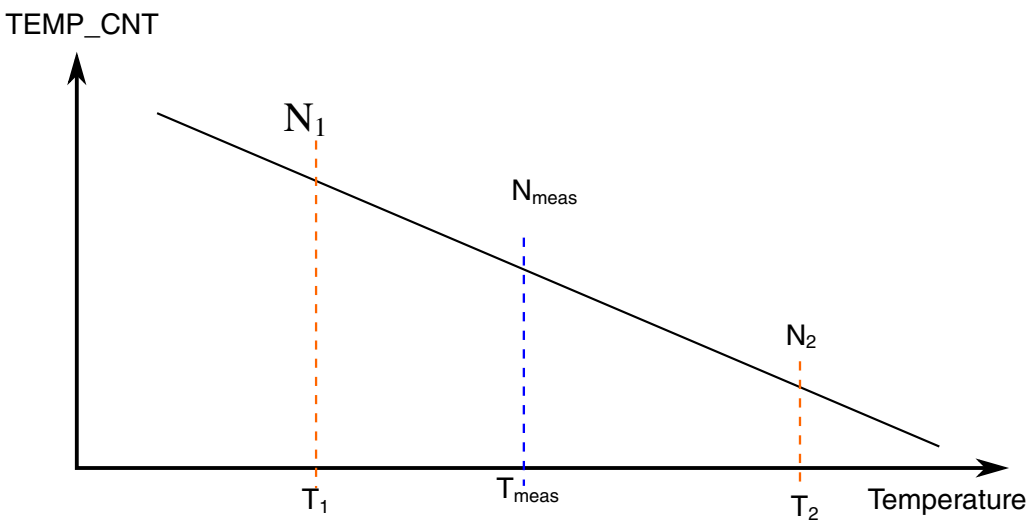


Figure 14-4. Temperature Measurement Cycle

After a temperature measurement cycle, software should use the calibration points in conjunction with the temperature code value in the TEMPMON_TEMPSENSE0[TEMP_CNT] bitfield to calculate the temperature for the device using the following equation:

$$T_{meas} = T_2 - (N_{meas} - N_2) * ((T_2 - T_1) / (N_1 - N_2))$$

Likewise, to determine the alarm counter value to be written in the TEMPMON_TEMPSENSE0 register for a temperature based interrupt, the above equation can be solved for the N_{meas} value that should be used based on the desired temperature trigger.

The temperature calibration point fuse values are available in the OCOTP_ANA1 register. The temperature calibration values are fused individually for each part in the product testing process. The fields of this register are described in the following table.

Table 14-6. OCOTP_ANA1 Temperature Sensor Calibration Data

Bit Range	Bit Mask	Name	Description
[31:20]	FFF0_0000h	ROOM_COUNT	Value of TEMPMON_TEMPSENSE0[TEMP_VALUE] after a measurement cycle at room temperature (25.0 °C).
[19:8]	000F_FF00h	HOT_COUNT	Value of TEMPMON_TEMPSENSE0[TEMP_VALUE] after a measurement cycle at the hot temperature, i.e. HOT_TEMP.
[7:0]	0000_00FFh	HOT_TEMP	The hot temperature test point. Each LSB equals 1 °C.

The points on the calibration curve are as follows.

- (N_1, T_1) = (ROOM_COUNT, 25.0)
- (N_2, T_2) = (HOT_COUNT, HOT_TEMP)
- (N_{meas}, T_{meas}) = (TEMP_CNT, T_{meas})

Substituting the fields from OCOTP_ANA1 into the earlier equation results in the following:

T_{meas} = HOT_TEMP - (N_{meas} - HOT_COUNT) * ((HOT_TEMP - 25.0) / (ROOM_COUNT – HOT_COUNT))

14.2.3 TEMPMON Memory Map/Register Definition

TEMPMON memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3036_0300	Anadig Tempsensor Control Register 0 (TEMPMON_HW_ANADIG_TEMPSENSE0)	32	R/W	07FF_FE00h	14.2.3.1/ 4174
3036_0304	Anadig Tempsensor Control Register 0 (TEMPMON_HW_ANADIG_TEMPSENSE0_SET)	32	R/W	07FF_FE00h	14.2.3.1/ 4174
3036_0308	Anadig Tempsensor Control Register 0 (TEMPMON_HW_ANADIG_TEMPSENSE0_CLR)	32	R/W	07FF_FE00h	14.2.3.1/ 4174
3036_030C	Anadig Tempsensor Control Register 0 (TEMPMON_HW_ANADIG_TEMPSENSE0_TOG)	32	R/W	07FF_FE00h	14.2.3.1/ 4174
3036_0310	Anadig Tempsensor Control Register 1 (TEMPMON_HW_ANADIG_TEMPSENSE1)	32	R/W	0001_0219h	14.2.3.2/ 4175
3036_0314	Anadig Tempsensor Control Register 1 (TEMPMON_HW_ANADIG_TEMPSENSE1_SET)	32	R/W	0001_0219h	14.2.3.2/ 4175

Table continues on the next page...

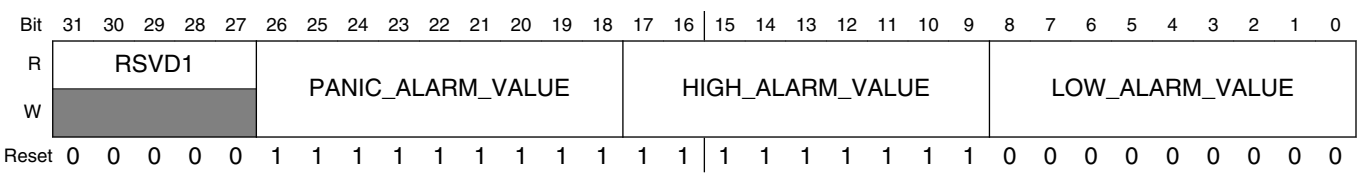
TEMPMON memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3036_0318	Anadig Tempsensor Control Register 1 (TEMPMON_HW_ANADIG_TEMPSENSE1_CLR)	32	R/W	0001_0219h	14.2.3.2/ 4175
3036_031C	Anadig Tempsensor Control Register 1 (TEMPMON_HW_ANADIG_TEMPSENSE1_TOG)	32	R/W	0001_0219h	14.2.3.2/ 4175
3036_0320	Anadig Tempsensor Trim Control Register (TEMPMON_HW_ANADIG_TEMPSENSE_TRIM)	32	R/W	C080_0010h	14.2.3.3/ 4176
3036_0324	Anadig Tempsensor Trim Control Register (TEMPMON_HW_ANADIG_TEMPSENSE_TRIM_SET)	32	R/W	C080_0010h	14.2.3.3/ 4176
3036_0328	Anadig Tempsensor Trim Control Register (TEMPMON_HW_ANADIG_TEMPSENSE_TRIM_CLR)	32	R/W	C080_0010h	14.2.3.3/ 4176
3036_032C	Anadig Tempsensor Trim Control Register (TEMPMON_HW_ANADIG_TEMPSENSE_TRIM_TOG)	32	R/W	C080_0010h	14.2.3.3/ 4176

14.2.3.1 Anadig Tempsensor Control Register 0 (TEMPMON_HW_ANADIG_TEMPSENSE0*n*)

This register defines the alarm and panic functions for the temperature sensor.

Address: 3036_0000h base + 300h offset + (4d × i), where i=0d to 3d



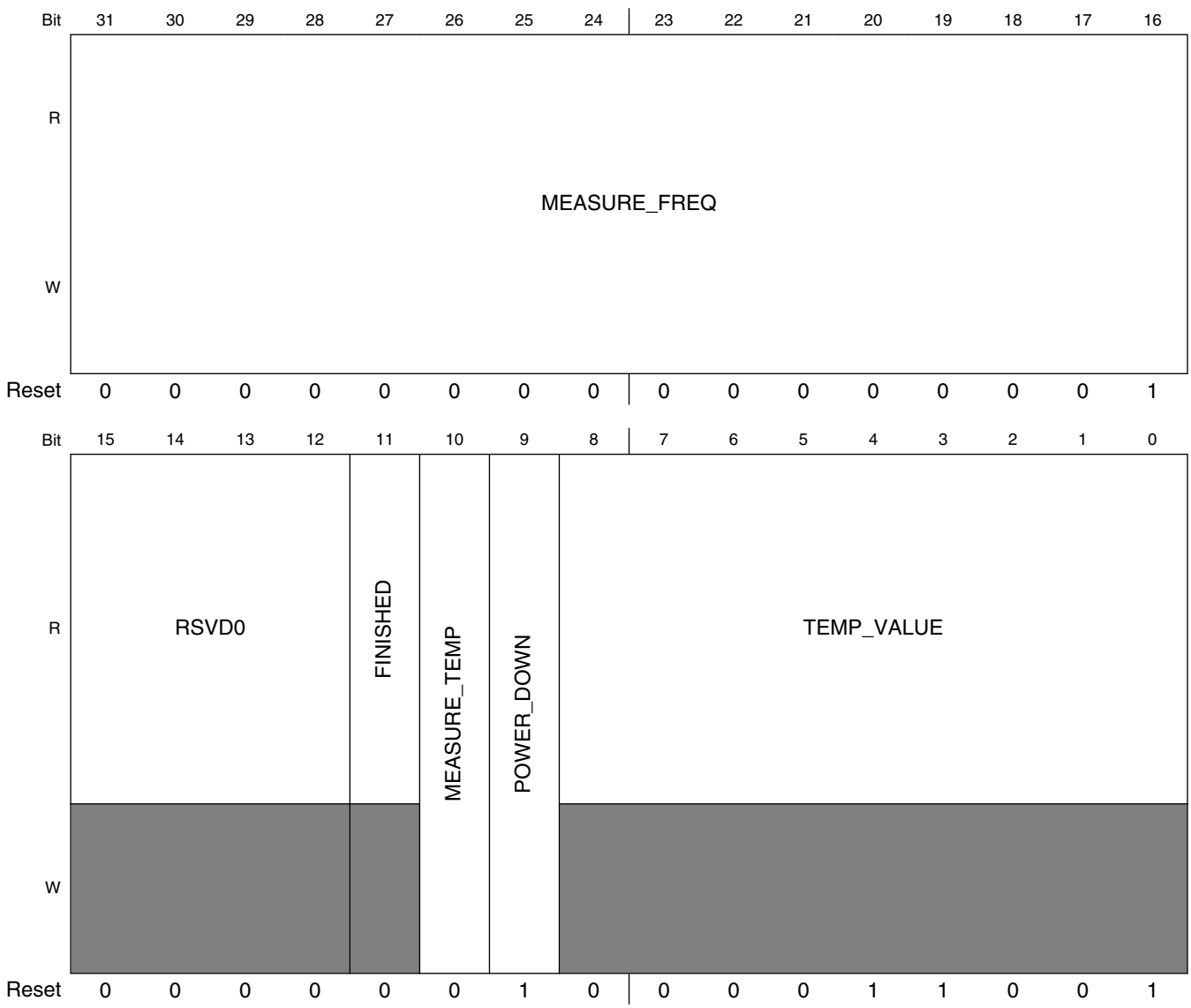
TEMPMON_HW_ANADIG_TEMPSENSE0*n* field descriptions

Field	Description
31–27 RSVD1	This bit field contains the last measured temperature.
26–18 PANIC_ALARM_ VALUE	This bit field contains the temperature measurement that will cause a panic and reset the chip.
17–9 HIGH_ALARM_ VALUE	This bit field contains the temperature measurement that will issue a high temperature interrupt
LOW_ALARM_ VALUE	This bit field contains the temperature measurement that will issue a low temperature interrupt.

14.2.3.2 Anadig Tempsensor Control Register 1 (TEMPMON_HW_ANADIG_TEMPSENSE1*n*)

This register defines the automatic repeat time of the temperature sensor.

Address: 3036_0000h base + 310h offset + (4d × i), where i=0d to 3d



TEMPMON_HW_ANADIG_TEMPSENSE1*n* field descriptions

Field	Description
31–16 MEASURE_ FREQ	This bits determines how many RTC clocks to wait before automatically repeating a temperature measurement. 0x0000 - defines a single measurement with no repeat. 0xFFFF determines a two second sample rate with a 32.768KHz RTC clock. Exact timings depend on the accuracy of the RTC clock.

Table continues on the next page...

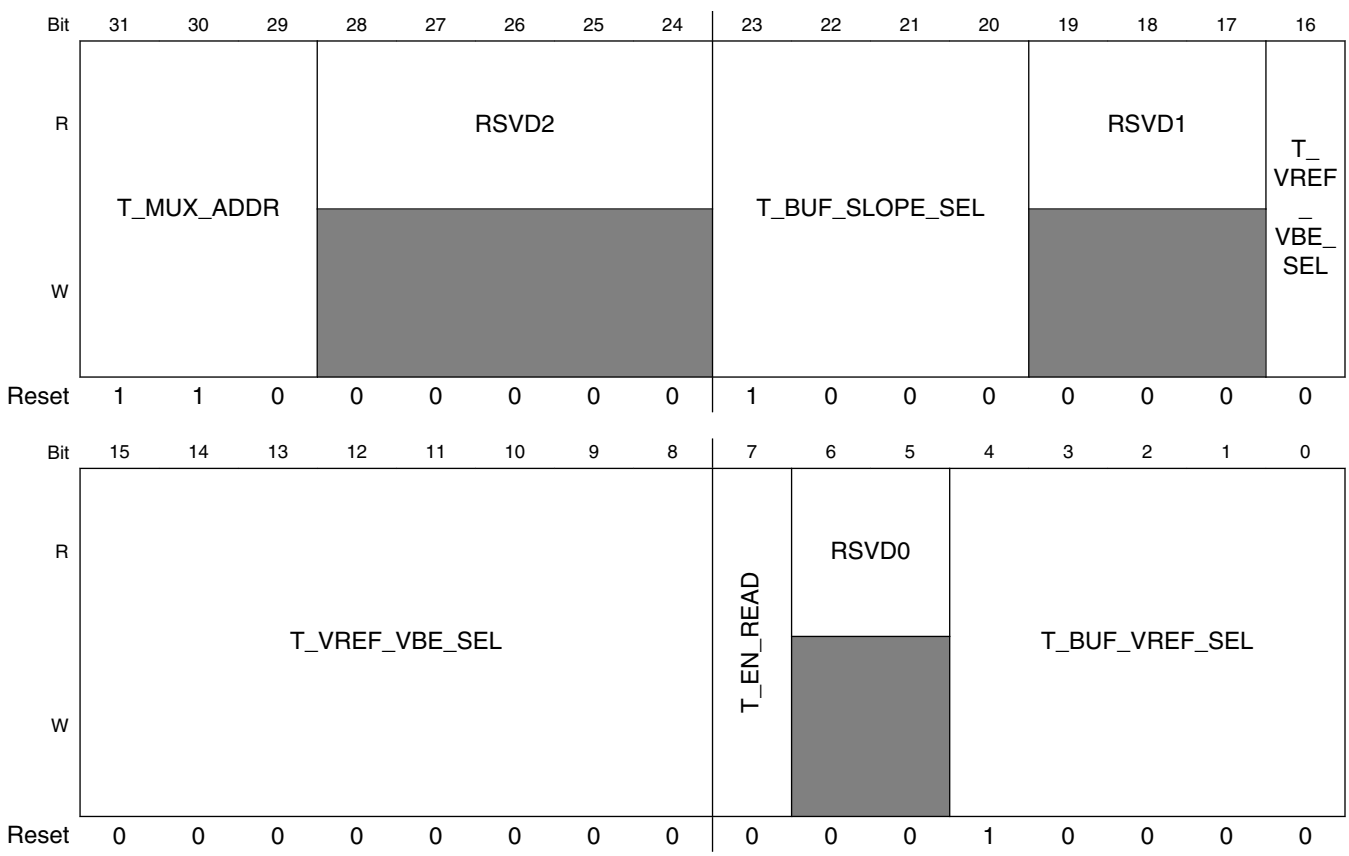
TEMPMON_HW_ANADIG_TEMPSENSE1*n* field descriptions (continued)

Field	Description
15–12 RSVD0	Always set to zero (0).
11 FINISHED	Indicates that the latest temp is valid. This bit should be cleared by the sensor after the start of each measurement.
10 MEASURE_ TEMP	Starts the measurement process. If the measurement frequency is zero in the tempsens_1 register. This results in a single conversion.
9 POWER_DOWN	This bit powers down the temperature sensor.
TEMP_VALUE	This bit field contains the temperature measurement.

14.2.3.3 Anadig Tempsensor Trim Control Register (TEMPMON_HW_ANADIG_TEMPSENSE_TRIM*n*)

This register defines the automatic repeat time of the temperature sensor.

Address: 3036_0000h base + 320h offset + (4d × i), where i=0d to 3d



TEMPMON_HW_ANADIG_TEMPSENSE_TRIM*n* field descriptions

Field	Description
31–29 T_MUX_ADDR	Test MUX address setting bits. Only for de-bugging purpose. Default: 110 Other Values : TBD
28–24 RSVD2	Always set to zero (0).
23–20 T_BUF_SLOPE_SEL	Amplifier gain setting bits. Default: 1000 This value can be changed by BJT characteristics. It will be fixed after temperature test.
19–17 RSVD1	Always set to zero (0).
16–8 T_VREF_VBE_SEL	Reference voltage setting bits for the am-plifier in the Positive-TC generator block. Default: 10000
7 T_EN_READ	Definition:TBD
6–5 RSVD0	Always set to zero (0).
T_BUF_VREF_SEL	Reference voltage setting bits for the am-plifier in the Positive-TC generator block. Default: 10000

Chapter 15
Low Speed Communication and Interconnects

15.1 Flexible Controller Area Network (FLEXCAN)

15.1.1 Overview

The Flexible Controller Area Network (FLEXCAN) module is a communication controller implementing the CAN protocol according to the CAN 2.0B protocol specification.

The CAN protocol was primarily designed to be used as a vehicle serial data bus meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth. The FLEXCAN module is a full implementation of the CAN protocol specification, which supports both standard and extended message frames. 64 Message Buffers are supported.

15.1.1.1 Block Diagram

A general block diagram is shown in the figure below,which describes the main sub-blocks implemented in the FLEXCAN module, including the associated memory for storing Mailboxes, Rx Global Mask Registers, Rx Individual Mask Registers, Rx FIFO and Rx FIFO ID Filters.

Support for 64 Mailboxes and 6-deep Rx FIFO is provided. The functions of the sub-modules are described in subsequent sections.

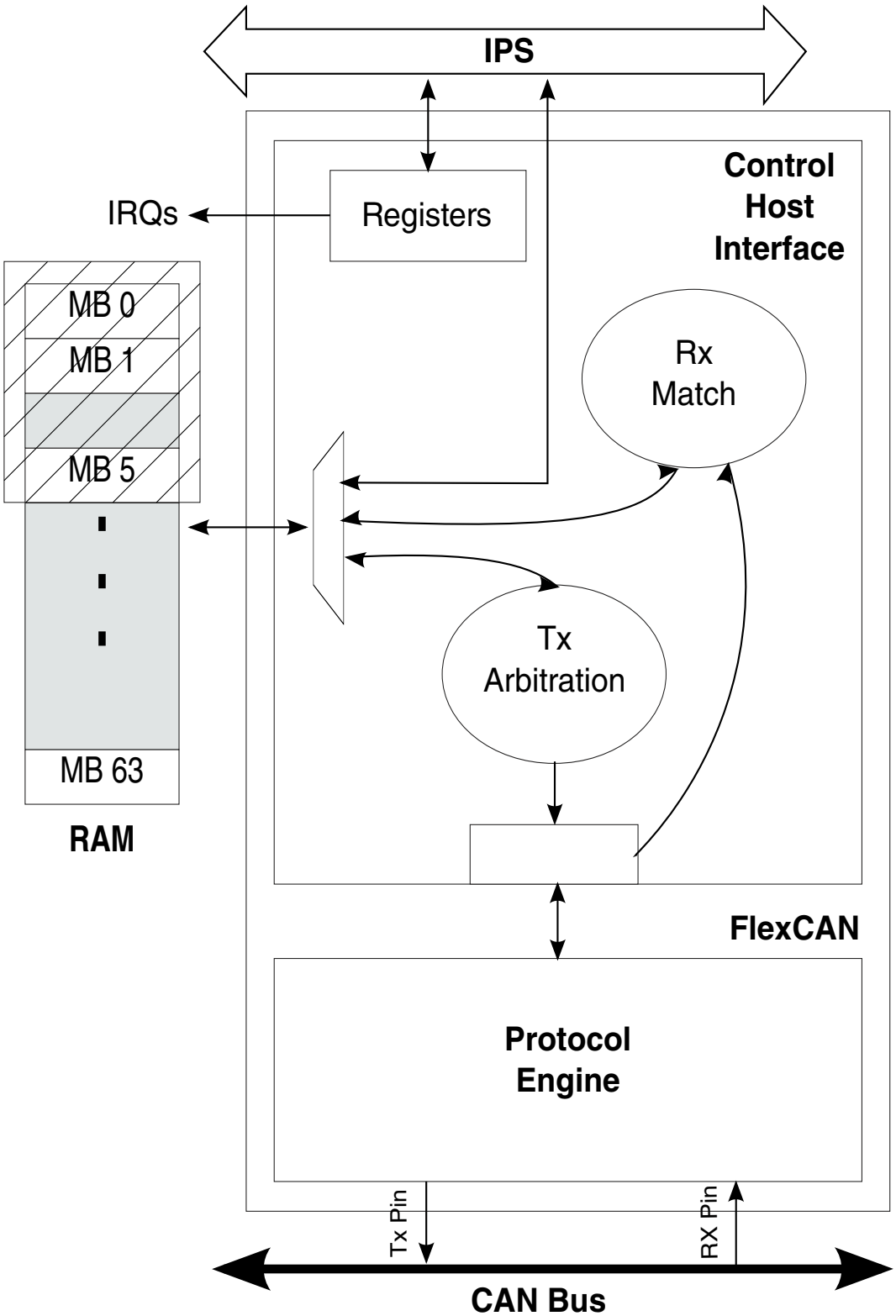


Figure 15-1. FLEXCAN Block Diagram

15.1.1.2 FLEXCAN Module Features

The FLEXCAN module includes these distinctive legacy features:

- Version 2.0B
  - Standard data and remote frames
  - Extended data and remote frames
  - Zero to eight bytes data length
  - Programmable bit rate up to 1 Mb/sec
  - Content-related addressing
- Flexible Mailboxes of eight bytes data length
- Each Mailbox is configurable as Rx or Tx, all supporting standard and extended messages
- Individual Rx Mask Registers per Mailbox
- Full featured Rx FIFO with storage capacity for 6 frames and internal pointer handling
- Transmission abort capability
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard or 512 partial (8 bits) IDs, with up to 32 individual masking capability
- 100% backwards compatibility with previous FLEXCAN version
- Unused structures space can be used as general purpose RAM space
- Listen only mode capability
- Programmable loop-back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number or highest priority
- Time Stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts independent of the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes, with programmable wake up on bus activity
- Configurable Glitch filter width to filter the noise on CAN bus when waking up
- Remote request frames may be handled automatically or by software.
- ID filter configuration in Normal Mode
- CAN bit time settings and configuration bits can only be written in Freeze Mode
- Tx mailbox status (Lowest priority buffer or empty buffer)
- SYNC bit status to inform that the module is synchronous with CAN bus
- CRC status for transmitted message
- Selectable priority between Mailboxes and Rx FIFO during matching process

15.1.1.3 Modes of Operation

The FLEXCAN module has four functional modes: Normal Mode (User and Supervisor), Freeze Mode, Listen-Only Mode and Loop-Back Mode. There are also two low power modes: Disable Mode and Stop Mode.

- Normal Mode (User or Supervisor):

In Normal Mode, the module operates receiving and/or transmitting message frames, errors are handled normally and all the CAN Protocol functions are enabled. User and Supervisor Modes differ in the access to some restricted control registers.

- Freeze Mode:

It is enabled when the FRZ bit in the MCR Register is asserted. If enabled, Freeze Mode is entered when the HALT bit in MCR is set or when Debug Mode is requested at MCU level and the FRZ_ACK bit in the MCR Register is asserted by the FlexCAN . In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Freeze Mode for more information.

- Listen-Only Mode:

The module enters this mode when the LOM bit in the Control Register is asserted. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FLEXCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

- Loop-Back Mode:

The module enters this mode when the LPB bit in the Control Register is asserted. In this mode, FLEXCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The FLEXCAN_RX input pin is ignored and the FLEXCAN_TX output goes to the recessive state (logic '1'). FLEXCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FLEXCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

- Module Disable Mode:

This low power mode is entered when the MDIS bit in the MCR Register is asserted and the LPM_ACK is asserted by the FlexCAN. When disabled, the module requests to disable the clocks to the CAN Protocol Engine and Controller Host Interface sub-modules. Exit from this mode is done by negating the MDIS bit in the MCR Register. See Module Disable Mode for more information.

- Stop Mode:

This low power mode is entered when Stop Mode is requested at ARM level and the LPM_ACK bit in the MCR Register is asserted by the FlexCAN. When in Stop Mode, the module puts itself in an inactive state and then informs the ARM that the clocks can be shut down globally. Exit from this mode happens when the Stop Mode request is removed or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled. See Stop Mode for more information.

15.1.2 External Signals

The FLEXCAN module has two I/O signals.

Table 15-1. FLEXCAN External Signals

Signal	Description	Pad	Mode	Direction
FLEXCAN1_RX	FLEXCAN receive pin. This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.	ENET1_RDATA2	ALT1	I
		GPIO1_IO12	ALT3	
		I2C1_SCL	ALT2	
		SAI1_RXD	ALT3	
		SD3_DATA7	ALT4	
FLEXCAN1_TX	FLEXCAN transmit pin. This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.	ENET1_RDATA3	ALT1	O
		GPIO1_IO13	ALT3	
		I2C1_SDA	ALT2	
		SAI1_TXC	ALT3	
		SD3_DATA5	ALT4	
FLEXCAN2_RX	FLEXCAN receive pin. This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.	ENET1_TDATA2	ALT1	I
		GPIO1_IO14	ALT3	
		I2C3_SCL	ALT2	
		SAI1_TXFS	ALT3	
		SD3_DATA4	ALT4	
FLEXCAN2_TX	FLEXCAN transmit pin. This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.	ENET1_TDATA3	ALT1	O
		GPIO1_IO15	ALT3	
		I2C3_SDA	ALT2	
		SAI1_TXD	ALT3	
		SD3_DATA6	ALT4	

15.1.3 Clocks

The table found here describes the clock sources for FLEXCAN.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 15-2. FLEXCAN Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_chi	ipg_clk_root	CHI clock
ipg_clk_pe	can_clk_root	Protocol Engine clock
ipg_clk_pe_nogate	can_clk_root	Protocol Engine clock (no gating)
ipg_clk_s	ipg_clk_root	Peripheral access clock
mem_ram_CLK	ipg_clk_root	RAM clock

15.1.4 Message Buffer Structure

Message Buffer Address: Base + 0x0080-0x047C

The Message Buffer structure used by the FLEXCAN module is represented in the figure found here.

Both Extended and Standard Frames (29-bit Identifier and 11-bit Identifier, respectively) used in the CAN specification are represented.

Table 15-3. Message Buffer Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0					CODE					SRR	IDE	RTR	DLC				TIME STAMP															
0x4	PRIO				ID Standard												ID Extended															
0x8	DATA BYTE 0								DATA BYTE 1								DATA BYTE 2								DATA BYTE 3							
0xC	DATA BYTE 4								DATA BYTE 5								DATA BYTE 6								DATA BYTE 7							

CODE - Message Buffer Code

This 4-bit field can be accessed (read or write) by the CPU and by the FLEXCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in the following tables. See Functional Description for additional information.

Table 15-4. Message Buffer Code for Rx buffers

CODE Description	Rx Code BEFORE receive New Frame	SRV[1]	Rx Code AFTER successful reception[2]	RRS[3]	Comment
0b0000: INACTIVE- MB is not active.	INACTIVE	-	-	-	MB does not participate in the matching process.
0b0100: EMPTY - MB is active and empty.	EMPTY	-	FULL	-	When a frame is received successfully (after move-in process. Refer to Move-in for details), the CODE field is automatically updated to FULL.
0b0010: FULL - MB is full.	FULL	Yes	FULL	-	The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a new frame is moved to the MB after the MB was serviced, the code still remains FULL. Refer to Matching Process for matching details related to FULL code.
		No	OVERRUN	-	If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. Refer to Matching Process for details about overrun behavior.
0b0110: OVERRUN - MB is being overwritten into a full buffer.	OVERRUN	Yes	FULL	-	If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB, the code returns to FULL.
		No	OVERRUN	-	If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. Refer to Matching Process for details about overrun behavior.
0b1010: RANSWER[4] - A frame was configured to recognize a Remote Request Frame and transmit a Response Frame in return.	RANSWER	-	TANSWER(0b1110)	0	A Remote Answer was configured to recognize a remote request frame received, after that a MB is set to transmit a response frame. The code is automatically changed to TANSWER (0b1110). Refer to Matching Process for details. If CTRL2[RRS] is negated, transmit a response frame whenever a remote request frame with the same ID is received.
			-	1	This code is ignored during matching and arbitration process. Refer to Matching Process for details.

Table continues on the next page...

Table 15-4. Message Buffer Code for Rx buffers (continued)

CODE Description	Rx Code BEFORE receive New Frame	SRV[1]	Rx Code AFTER successful reception[2]	RRS[3]	Comment
CODE[0]=1b1: BUSY - FlexCAN is updating the contents of the MB. The CPU must not access the MB.	BUSY[5]	-	FULL	-	Indicates that the MB is being updated, it will be negated automatically and does not interfere on the next CODE.
			OVERRUN	-	

1. SRV: Serviced MB. MB was read and unlocked by reading TIMER or other MB.
2. A frame is considered successful reception after the frame to be moved to MB (move-in process). Refer to Move-in for details)
3. Remote Request Stored bit from CTRL2 register. Refer to CTRL2 for details.
4. Code 4'b1010 is not considered as a Tx and a MB with this code should not to be aborted.
5. Note that for Tx MBs, the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register. If this bit is asserted, the corresponding MB does not participate in the matching process.

Table 15-5. Message Buffer Code for Tx buffers

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
0b1000: INACTIVE - MB is not active	INACTIVE	-	-	MB does not participate in the arbitration process.
0b1001: ABORT - MB is aborted	ABORT	-	-	MB does not participate in the arbitration process. .
0b1100: DATA - MB is a Tx Data Frame (MB RTR must be 0)	DATA	0	INACTIVE	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
0b1100: REMOTE - MB is a Tx Remote Request Frame (MB RTR must be 1)	REMOTE	1	EMPTY	Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID.
0b1110: TANSWER - MB is a Tx Response Frame from an incoming Remote Request Frame	TANSWER	-	RANSWER	This is an intermediate code that is automatically written to the MB by the CHI as a result of match to a remote request frame. The remote response frame will be transmitted unconditionally once and then the code will automatically return to RANSWER (0b1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. Refer to Matching Process and Arbitration process for details.

SRR - Substitute Remote Request

Fixed recessive bit, used only in extended format. It must be set to '1' by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FLEXCAN receives this bit as dominant, then it is interpreted as arbitration loss.

1= Recessive value is compulsory for transmission in Extended Format frames

0= Dominant is not a valid value for transmission in Extended Format frames

IDE - ID Extended Bit

This bit identifies whether the frame format is standard or extended. It is also used as part of the reception filter.

1= Frame format is extended

0= Frame format is standard

RTR - Remote Transmission Request

This bit affects the behavior of Remote Frames and is part of the reception filter. Refer to the tables above and RRS bit in Control 2 Register (FLEXCAN_CTRL2) for additional details.

If FLEXCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FLEXCAN module treats it as bit error. If the value received matches the value transmitted, it is considered as a successful bit transmission.

1= Indicates the current MB has a Remote Frame to be transmitted if MB is Tx. If the MB is Rx then incoming Remote Request Frames may be stored.

0= Indicates the current MB has a Data Frame to be transmitted. In Rx MB it may be considered in matching processes.

DLC - Length of Data in Bytes

This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x08 through 0x0F of the MB space (see the first table above). In reception, this field is written by the FLEXCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the ARM and corresponds to the DLC field value of the frame to be transmitted. When RTR=1, the Frame to be transmitted is a Remote Frame and does not include the data field, regardless of the Length field. The DLC field indicates which DATA BYTEs are valid as shown in the table below.

TIME STAMP - Free-Running Counter Time Stamp

This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.

PRIO - Local priority

This 3-bit field is only used when MCR[LPRIO_EN] bit is asserted and it only makes sense for Tx mailboxes. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Arbitration process.

ID - Frame Identifier

In Standard Frame format, only the 11 most significant bits (28 to 18) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In Extended Frame format, all bits are used for frame identification in both receive and transmit cases.

DATA BYTE 0-7 - Data Field

Up to eight bytes can be used for a data frame.

For Rx frames, the data is stored as it is received from the CAN bus. DATA BYTE (n) is valid only if *n* is less than DLC as shown in the table below.

For Tx frames, the CPU prepares the data field to be transmitted within the frame.

Table 15-6. DATA BYTEs validity

DLC	Valid DATA BYTEs
0	none
1	DATA BYTE 0
2	DATA BYTE 0-1
3	DATA BYTE 0-2
4	DATA BYTE 0-3
5	DATA BYTE 0-4
6	DATA BYTE 0-5
7	DATA BYTE 0-6
8	DATA BYTE 0-7

15.1.5 Rx FIFO Structure

When the MCR[RFEN] bit is set, the memory area from $80 to $DC (which is normally occupied by MBs 0 to 5) is used by the reception FIFO engine.

The region 0x80-0x8C contains the output of the FIFO which must be read by the CPU as a Message Buffer. This output contains the oldest message received and not read yet. The region 0x90-0xDC is reserved for internal use of the FIFO engine.

An additional memory area, that starts at 0xE0 and may extend up to 0x2DC (normally occupied by MBs 6 up to 37) depending on the CTRL2[RFFN] field setting, contains the ID Filter Table (configurable from 8 to 128 memory positions) that specifies filtering criteria for accepting frames into the FIFO.Table 15-7 shows the Rx FIFO data structure.

Each ID Filter Table Element occupies an entire 32-bit word and can be compounded by one, two or four Identifier Acceptance Filters (IDAF) depending on the MCR[IDAM] field setting. Table 15-8, Table 15-9 and Table 15-10 show the IDAF indexation. Table 15-11 show the three different formats that the IDAF can assume, depending on the MCR[IDAM] field setting. Note that all elements of the table must have the same format. See Rx FIFO for more information.

Out of reset, the ID Filter Table flexible memory area defaults to 0xE0 and only extends to 0xFC, which corresponds to MBs 6 to 7 for RFFN=0.

Table 15-7. Rx FIFO Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80										SRR	IDE	RTR	DLC				TIME STAMP															
0x84										ID Standard															ID Extended							
0x88	Data Byte 0												Data Byte 1												Data Byte 2							
0x8C	Data Byte 4												Data Byte 5												Data Byte 6							
0x90 to 0xDC	Reserved																															
0xE0	ID Filter Table Element 0																															
0xE4	ID Filter Table Element 1																															
0xE8 to 0x2D4	ID Filter Table Elements 2 through 125																															
0x2D8	ID Filter Table Element 126																															
0x2DC	ID Filter Table Element 127																															

Table 15-8. Position of ID Filter Table Elements in format A

Element	31																												0
0	IDAF0																												
1	IDAF1																												
2 through 125	Identifier Acceptance Filter 2 through 125																												
126	IDAF126																												
127	IDAF127																												

Table 15-9. Position of ID Filter Table Elements in format B

Element	31															16	15												0
0	IDAF0																IDAF1												
1	IDAF2																IDAF3												
2 through 125	Identifier Acceptance Filter 4 through 251																												
126	IDAF252																IDAF253												
127	IDAF254																IDAF255												

Table 15-10. Position of ID Filter Table Elements in format C

Element	31							24	23							16	15							8	7						0
0	IDAF0								IDAF1								IDAF2								IDAF3						
1	IDAF4								IDAF5								IDAF6								IDAF7						
2 through 125	Identifier Acceptance Filter 8 through 503																														
126	IDAF504								IDAF505								IDAF506								IDAF507						
127	IDAF508								IDAF509								IDAF510								IDAF511						

Table 15-11. Identifier Acceptance Filter Format A,B and C

Format	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	RTR	IDE	RXIDA (Standard = 29-19, Extended = 29-1)																													
B	RTR	IDE	RXIDB_0 (Standard = 29-19, Extended = 29-16)														RTR	IDE	RXIDB_1 (Standard = 13-3, Extended = 13-0)														
C	RXIDC_0								RXIDC_1								RXIDC_2								RXIDC_3							

Table 15-11. Identifier Acceptance Filter Format A,B and C

	(Std/Ext = 31-24)	(Std/Ext = 23-16)	(Std/Ext = 15-8)	(Std/Ext = 7-0)

RTR - Remote Frame

This bit specifies whether Remote Request Frames are accepted into the FIFO if they match the target ID in Formats A and B. If Format C is chosen the acceptance does not depend on whether the frame is a Remote Request Frame or not.

1= Remote Frames can be accepted and data frames are rejected

0= Remote Frames are rejected and data frames can be accepted

IDE - Extended Frame

Specifies if either Extended or Standard Format frames are accepted into the FIFO if they match the target ID in Formats A and B. If Format C is chosen the acceptance does not depend on whether the frame is of the Extended or Standard Format.

1= Extended frames can be accepted and standard frames are rejected

0= Extended frames are rejected and standard frames can be accepted

RXIDA - Rx Frame Identifier (Format A)

Specifies an ID to be used as acceptance criteria for the FIFO. In the Standard Format (IDAF's or incoming frame's IDE bit is negated), only the 11 most significant bits (29 to 19) are used for frame identification. In the Extended Format (both IDAF's and incoming frame's IDE are asserted), all bits are used.

RXIDB_0, RXIDB_1 - Rx Frame Identifier (Format B)

Specifies an ID to be used as acceptance criteria for the FIFO. In the Standard Format (IDAF's or incoming frame's IDE bit is negated), the 11 most significant bits (29 to 19 and 13 to 3) are used for frame identification. In the Extended Format (both IDAF's and incoming frame's IDE are asserted), all 14 bits of the field are compared with the 14 most significant bits of the Identifier of the incoming frame. The 15 least significant bits of the Identifier of an incoming Extended Format frame do not affect the acceptance.

RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 - Rx Frame Identifier (Format C)

Specifies an ID to be used as acceptance criteria for the FIFO. In both Standard Format and Extended Format, all 8 bits of the field are compared to the 8 most significant bits of the Identifier of the incoming frame. The 3 least significant bits of the Identifier of an incoming Standard Format frame and the 21 least significant bits of the Identifier of an incoming Extended Format frame do not affect the acceptance.

15.1.6 Functional Description

This section provides a complete functional description of the block.

15.1.6.1 Functional Overview

The FLEXCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames.

The mailbox system consists of a set of 64 Message Buffers (MB) that store configuration and control data, time stamp, message ID and data (see Message Buffer Structure). The memory corresponding to the first 38 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 128 extended IDs or 256 standard IDs or 512 8-bit ID slices), with individual mask register for up to 32 ID Filter Table elements. Simultaneous reception through FIFO and mailbox is supported. For mailbox reception, a *matching* algorithm makes it possible to store received frames only into MBs that have the same ID. A masking scheme makes it possible to match the ID programmed on the MB with a range of Identifiers on received CAN frames. For transmission, an *arbitration* algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be "active" at a given time if it can participate in both the Matching and Arbitration processes. An Rx MB with a 0b0000 code is inactive (refer to Table 15-4). Similarly, a Tx MB with a 0b1000 or 0b1001 code is also inactive (refer to Table 15-5).

15.1.6.2 Transmit Process

In order to transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the procedure found here.

1. Check if the respective interruption bit is set and clear it.
2. If the MB is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control and Status word to request an abortion of the transmission. Wait for the corresponding IFLAG to be asserted by polling the IFLAG register or by the interrupt request if enabled by the respective IMASK. Then read back the CODE field to check if the transmission was aborted or transmitted (see Transmission Abort Mechanism). If backwards compatibility is desired (MCR[AEN]

bit negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the MB but then the pending frame may be transmitted without notification (see Message Buffer Inactivation).
3. Write the ID word.
4. Write the data bytes.
5. Write the DLC, Control and Code fields of the Control and Status word to activate the MB.

Once the MB is activated, it will participate into the arbitration process and eventually be transmitted according to its priority.

At the end of the successful transmission, the value of the Free Running Timer at the time of the second bit of frame's Identifier field is written into the MB's Time Stamp field, the CODE field in the Control and Status word is updated, the CRC Register is updated, a status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new CODE field after transmission depends on the code that was used to activate the MB in step four (see Table 15-4 and Table 15-5 in Message Buffer Structure)

When the Abort feature is enabled (MCR[AEN] bit is asserted), after the Interrupt Flag is asserted for a Mailbox configured as transmit buffer, the Mailbox is blocked, therefore the CPU is not able to update it until it negates the Interrupt Flag. It means that the CPU must clear the corresponding IFLAG before starting to prepare this MB for a new transmission or reception.

15.1.6.3 Arbitration process

The arbitration process scans the Mailboxes searching the Tx one that holds the message to be sent in the next opportunity. This Mailbox is called the *arbitration winner*.

The scan starts from the lowest Mailbox number and runs toward the higher ones.

The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the CTRL2[TASD] field value. See Control 2 Register (FLEXCAN_CTRL2) for details.
- During the error delimiter field of the CAN frame
- During the Overload Delimiter field of a CAN frame.
- When the winner is inactivated and the CAN bus has still not reached the first bit of the Intermission field.
- When ARM write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the Intermission field.
- When CHI is in Idle state and ARM writes to the C/S word of any MB.

- When FlexCAN exits Bus Off state
- Upon leaving Freeze Mode or Low Power Mode

If the arbitration process does not manage to evaluate all Mailboxes before the CAN bus has reached the first bit of the Intermission field the temporary arbitration winner is invalidated and the FlexCAN will not compete for the CAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx Mailboxes at the end of the scan according to both CTRL1[LBUF] and MCR[LPRIO_EN] bits settings.

15.1.6.3.1 Lowest Mailbox number first

If CTRL1[LBUF] bit is asserted the first (lowest number) active Tx Mailbox found is the arbitration winner. MCR[LPRIO_EN] bit has no effect when CTRL1[LBUF] is asserted.

15.1.6.3.2 Highest Mailbox priority first

If CTRL1[LBUF] bit is negated then the arbitration process searches the active Tx Mailbox with the highest priority, and this Mailbox would have a higher probability to win the arbitration on CAN bus.

The sequence of bits considered for this arbitration is called the *arbitration value* of the Mailbox. The highest priority Tx Mailbox is the one that has the least arbitration value among all Tx Mailboxes.

If two or more Mailboxes have equivalent arbitration values the lowest Mailbox number is the arbitration winner.

The composition of the arbitration value depends on MCR[LPRIO_EN] bit setting.

15.1.6.3.2.1 Local Priority disabled

If MCR[LPRIO_EN] bit is negated the arbitration value is built in the exact sequence of bits as they would be transmitted in a CAN frame (see Table 15-12) in such a way that the Local Priority is disabled.

Table 15-12. Composition of the arbitration value when Local Priority is disabled

Format	Mailbox Arbitration Value (32 bits)				
Standard (IDE = 0)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

15.1.6.3.2.2 Local Priority enabled

If Local Priority is desired MCR[LPRIO_EN] must be asserted.

In this case the Mailbox PRIO field is included at the very left of the arbitration value (see the table below).

Table 15-13. Composition of the arbitration value when Local Priority is enabled

Format	Mailbox Arbitration Value (35 bits)					
Standard (IDE = 0)	PRIO (3 bits)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	PRIO (3 bits)	Extended ID[28:18](11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

As the PRIO field is the most significant part of the arbitration value Mailboxes with low PRIO values have higher priority than Mailboxes with high PRIO values regardless the rest of their arbitration values.

Note that the PRIO field is not part of the frame on the CAN bus. Its purpose is only to affect the internal arbitration process.

Once the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called "move-out" and after it is done, write access to the corresponding MB is blocked (if the AEN bit in MCR is asserted). The write access is released in the following events:

- After the MB is transmitted
- FlexCAN enters in Freeze Mode or Bus Off
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules. FlexCAN transmits up to eight data bytes, even if the DLC (Data Length Code) field value is greater than that.

Arbitration process can be triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. Arbitration start point depends on instantiation parameters NUMBER_OF_MB and TASD. Additionally, TASD value may be changed (see Control 2 Register (FLEXCAN_CTRL2)) to optimize the arbitration start point.
- During CAN Bus Off state from TX_ERR_CNT=124 to 128. Arbitration start point depends on instantiation parameters NUMBER_OF_MB and TASD. Additionally,

TASD value may be changed (see Control 2 Register (FLEXCAN_CTRL2)) to optimize the arbitration start point.

- During C/S write by CPU in Bus Idle. First C/S write starts arbitration process and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, Tx arbitration process is pending. If there is no arbitration winner after arbitration process has finished, then TX arbitration machine begins a new arbitration process.
- Arbitration winner deactivation during a valid arbitration window.
- Upon Leave Freeze Mode. If there is a re-synchronization during WaitForBusIdle arbitration process is restarted.

Arbitration process stops in the following situation:

- All Mailboxes were scanned.
- A Tx active Mailbox is found in case of Lowest Buffer feature enabled.
- Arbitration winner inactivation or abort during any arbitration process.
- There was not enough time to finish Tx arbitration process. For instance, a deactivation was performed near the end of frame). In this case arbitration process is pending.
- Error or Overload flag in the bus.
- Low Power or Freeze Mode request in Idle state

Arbitration is considered pending as described below:

- It was not possible to finish arbitration process in time.
- C/S write during arbitration if write is performed in a MB which number is lower than the Tx arbitration pointer.
- Any C/S write if there is no Tx Arbitration process in progress.
- Rx Match has just updated a Rx Code to Tx Code.
- Entering Bus off state.

C/S write during arbitration has the following effect:

- If C/S write is performed in the arbitration winner, a new process is restarted immediately.
- C/S write during arbitration if write is performed in a MB which number is higher than the Tx arbitration pointer.

15.1.6.4 Receive Process

To be able to receive CAN frames into a Mailbox, the CPU must prepare it for reception by executing the steps listed here.

1. If the Mailbox is active (either Tx or Rx) inactivate the Mailbox (see Message Buffer Inactivation), preferably with a *safe inactivation* (see Transmission Abort Mechanism);
2. Write the ID word;
3. Write the EMPTY code (0b0100) to the CODE field of the Control and Status word to activate the Mailbox.

Once the Mailbox is activated in the third step, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the Mailbox is updated by the *move-in process* (see Move-in) as follows:

1. The received Data field (8 bytes at most) is stored;
2. The received Identifier field is stored;
3. The value of the Free Running Timer at the time of the second bit of frame's Identifier field is written into the Mailbox Time Stamp field;
4. The received SRR, IDE, RTR and DLC fields are stored;
5. The CODE field in the Control and Status word is updated. (see Table 15-4 and Table 15-5 in Section Message Buffer Structure)
6. A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The recommended way for the CPU servicing (read) the frame received in a Mailbox is using the following procedure:

1. Read the Control and Status word of that Mailbox;
2. Check if the BUSY bit is deasserted, indicating that the Mailbox is locked. Repeat step 1) while it is asserted. See Message Buffer Lock Mechanism;
3. Read the contents of the Mailbox. Once Mailbox is locked now, its contents won't be modified by FlexCAN Move-in processes. See Move-in;
4. Acknowledge the proper flag at IFLAG registers;
5. Read the Free Running Timer. It is optional but recommended to unlock Mailbox as soon as possible and make it available for reception.

The CPU should synchronize to frame reception by the status flag bit for the specific Mailbox in one of the IFLAG Registers and not by the CODE field of that Mailbox. Polling the CODE field does not work because once a frame was received and the CPU services the Mailbox (by reading the C/S word followed by unlocking the Mailbox), the CODE field will not return to EMPTY. It will remain FULL. If the CPU tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the Mailbox without a prior *safe inactivation*, a newly received message matching the filter of that Mailbox may be lost.

In summary: never do polling by reading directly the C/S word of the Mailboxes. Instead, read the IFLAG registers.

Note that the received frame's Identifier field is always stored in the matching Mailbox, thus the contents of the ID field in a Mailbox may change if the match was due to masking. Note also that FlexCAN does receive frames transmitted by itself if there exists a matching Rx Mailbox, provided the MCR[SRX_DIS] bit is not asserted. If MCR[SRX_DIS] bit is asserted, FlexCAN will not store messages transmitted by itself in any MB, even if it contains a matching MB, and no interrupt flag or interrupt signal will be generated due to the frame reception.

To be able to receive CAN messages through the Rx FIFO, the CPU must enable and configure the Rx FIFO during Freeze Mode (see Rx FIFO). Upon receiving the Frames Available in Rx FIFO interrupt (see Interrupt Masks 1 Register (FLEXCAN_IMASK1), bit IFLAG[BUF5I] - Frames available in Rx FIFO), the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (optional - needed only if a mask was used for IDE and RTR bits);
2. Read the ID field (optional - needed only if a mask was used);
3. Read the Data field;
4. Read the RXFIR register (optional);
5. Clear the Frames Available in Rx FIFO interrupt by writing 1 to IFLAG[BUF5I] bit (mandatory - releases the MB and allows the CPU to read the next Rx FIFO entry)

15.1.6.5 Matching Process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus.

If the FIFO is enabled, the priority of scanning can be selected between Mailboxes and FIFO filters. In any case, the matching starts from the lowest number Message Buffer toward the higher ones. If no match is found within the first structure then the other is scanned subsequently. In the event that the FIFO is full, the matching algorithm will always look for a matching MB outside the FIFO region.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- if the received frame is a remote frame, the start point is the CRC field of the frame;
- if the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame;
- if the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame;

If a matching ID is found in the FIFO table or in one of the Mailboxes, the contents of the SMB will be transferred to the FIFO or to the matched Mailbox by the move-in process. If any CAN protocol error is detected then no match results will be transferred to the FIFO or to the matched Mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and active Rx Mailboxes (CODE is EMPTY, FULL, OVERRUN or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the CAN bus. The SMB has the same structure of a Mailbox. The reception structures (Rx FIFO or Mailboxes) associated with the matching elements that had a successful comparison are the *matched structures* . The *matching winner* is selected at the end of the scan among those matched structures and depends on conditions described ahead. Please, refer to the following table for details.

Table 15-14. Matching architecture

Structure	SMB[RTR]	CTRL2[RRS]	CTRL2[EACEN]	MB[IDE]	MB[RTR]	MB[ID][1]	MB[CODE]
Mailbox	0	-	0	cmp[2]	no_cmp[3]	cmp_msk[4]	EMPTY or FULL or OVERRUN
Mailbox	0	-	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	0	-	cmp	no_cmp	cmp	RANSWER
Mailbox	1	1	0	cmp	no_cmp	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	1	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
FIFO[5]	-	-	-	cmp_msk	cmp_msk	cmp_msk	-

1. For Mailbox structure, If SMB[IDE] is asserted, the ID is 29 bits (ID Standard + ID Extended). In case of SMB[IDE] to be negated, the ID is only 11 bits (ID Standard). Please, refer to Message Buffer Structure for ID details. For FIFO structure, the ID depends on IDAM. Please, refer to Rx FIFO Structure for IDAM details.
2. cmp: Compares the SMB contents with the MB contents regardless the masks.
3. no_cmp: The SMB contents are not compared with the MB contents.
4. cmp_msk: Compares the SMB contents with MB contents taking into account the masks.
5. SMB[IDE] and SMB[RTR] are not taken into account when IDAM is type C.

A reception structure is *free-to-receive* when any of the following conditions is satisfied:

- the CODE field of the Mailbox is EMPTY;
- the CODE field of the Mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the ARM and unlocked as described in Message Buffer Lock Mechanism);

- the CODE field of the Mailbox is either FULL or OVERRUN and an inactivation is performed. (see Message Buffer Inactivation)
- the Rx FIFO is not full.

The scan order for Mailboxes and Rx FIFO is from the matching element with lowest number to the higher ones.

The matching winner search for Mailboxes is affected by the MCR[IRMQ] bit. If it is negated the matching winner is the first matched Mailbox regardless if it is free-to-receive or not. If it is asserted, the matching winner is selected according to the priority below:

1. the first free-to-receive matched Mailbox;
2. the last non free-to-receive matched Mailbox.

It is possible to select the priority of scan between Mailboxes and Rx FIFO by the CTRL2[MRP] bit.

If the selected priority is Rx FIFO first:

- if the Rx FIFO is a matched structure and is free-to-receive then the Rx FIFO is the matching winner regardless of the scan for Mailboxes;
- otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched among Mailboxes as described above.

If the selected priority is Mailboxes first:

- if a free-to-receive matched Mailbox is found, it is the matching winner regardless the scan for Rx FIFO;
- if no matched Mailbox is found, then the matching winner is searched in the scan for the Rx FIFO;
- if both conditions above are not satisfied and a non free-to-receive matched Mailbox is found then the matching winner determination is conditioned by the MCR[IRMQ] bit:
  - if MCR[IRMQ] bit is negated the matching winner is the first matched Mailbox;
  - if MCR[IRMQ] bit is asserted the matching winner is the Rx FIFO if it is a free-to-receive matched structure, otherwise the matching winner is the last non free-to-receive matched Mailbox.

Please, refer to the table below for a summary of matching possibilities.

If a non-safe Mailbox inactivation (see Message Buffer Inactivation) occurs during matching process and the Mailbox inactivated is the temporary matching winner then the temporary matching winner is invalidated. The matching elements scan is not stopped nor

restarted, it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the FIFO is disabled, IRMQ is enabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm will find the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm will find MB number 2 again, but it is not "free-to-receive", so it will keep looking and find MB number 5 and store the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive", so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

Table 15-15. Matching Possibilities and Resulting Reception Structures

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception Structure	Description
No FIFO, only MB, match is always MB first						
0	0	X[1]	None[2]	-[3]	None	Frame lost by no match
0	0	X	Free[4]	-	FirstMB	
0	1	X	None	-	None	Frame lost by no match
0	1	X	Free	-	FirstMB	
0	1	X	NotFree	-	LastMB	Overrun
FIFO enabled, no match in FIFO is as if FIFO does not exist						
1	0	X	None	None[5]	None	Frame lost by no match
1	0	X	Free	None	FirstMB	
1	1	X	None	None	None	Frame lost by no match
1	1	X	Free	None	FirstMB	
1	1	X	NotFree	None	LastMB	Overrun
FIFO enabled, Queue disabled						
1	0	0	X	NotFull[6]	FIFO	
1	0	0	None	Full[7]	None	Frame lost by FIFO full (FIFO Overflow)
1	0	0	Free	Full	FirstMB	
1	0	0	NotFree	Full	FirstMB	
1	0	1	None	NotFull	FIFO	
1	0	1	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	0	1	Free	X	FirstMB	
1	0	1	NotFree	X	FirtsMB	Overrun
FIFO enabled, Queue enabled						
1	1	0	X	NotFull	FIFO	
1	1	0	None	Full	None	Frame lost by FIFO full (FIFO Overflow)

Table continues on the next page...

Table 15-15. Matching Possibilities and Resulting Reception Structures (continued)

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception Structure	Description
1	1	0	Free	Full	FirstMB	
1	1	0	NotFree	Full	LastMB	Overrun
1	1	1	None	NotFull	FIFO	
1	1	1	Free	X	FirstMB	
1	1	1	NotFree	NotFull	FIFO	
1	1	1	NotFree	Full	LastMB	Overrun

1. It is a don't care condition.
2. Matched in MB "None" means that the frame has not matched any MB (free-to-receive or non-free-to-receive).
3. It is a forbidden condition.
4. Matched in MB "Free" means that the frame matched at least one MB free-to-receive regardless it has matched MBs non-free-to-receive.
5. Matched in FIFO "None" means that the frame has not matched any filter in FIFO. It is as the FIFO didn't exist (CTRL2[RFEN]=0).
6. Matched in FIFO "NotFull" means that the frame has matched a FIFO filter and has empty slots to receive it.
7. Matched in FIFO "Full" means that the frame has matched a FIFO filter but couldn't store it because it has no empty slots to receive it.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for ARM to service the MBs. By programming more than one MB with the same ID, received messages will be queued into the MBs. ARM can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB. Please refer to Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK) . During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is "don't care". Please note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed while the module is in Freeze Mode , otherwise they are blocked by hardware.

FlexCAN also supports an alternate masking scheme with only four mask registers (RGXMASK, RX14MASK, RX15MASK and RXFGMASK) for backward compatibility. This alternate masking scheme is enabled when the IRMQ bit in the MCR Register is negated.

15.1.6.6 Move Process

There are two types of move process, namely move-in and move-out.

15.1.6.6.1 Move-in

The move-in process is the copy of a message received by an Rx SMB to a Rx Mailbox or FIFO that has matched it. If the move destination is the Rx FIFO, attributes of the message are also copied to the RXFIR FIFO. Each Rx SMB has its own move-in process, but only one is performed at a given time as described ahead.

The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (see Matching Process) and all of the following conditions are true:

- the CAN bus has reached or let past either:
  - the second bit of Intermission field next to the frame that carried the message that is in the Rx SMB;
  - the first bit of an overload frame next to the frame that carried the message that is in the Rx SMB;
- there is no ongoing matching process;
- the destination Mailbox is not locked by ARM;
- there is no ongoing move-in process from another Rx SMB. If more than one move-in processes are to be started at the same time both are performed and the newest substitutes the oldest.

The term *pending move-in* is used throughout the document and stands for a move-to-be that still does not satisfy all of the aforementioned conditions.

The move-in is cancelled and the Rx SMB is able to receive another message if any of the following conditions is satisfied:

- the destination Mailbox is inactivated after the CAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished;
- there is a previous pending move-in to the same destination Mailbox.
- the Rx SMB is receiving a frame transmitted by the FlexCAN itself and the self-reception is disabled;
- any CAN protocol error is detected.

Note that the pending move-in is not cancelled if the module enters in Freeze or Low Power Mode. It only stays on hold waiting for exiting Low Power Mode and to be unlocked. If an MB is unlocked during Freeze Mode, the move-in happens immediately.

The move-in process consists of the following steps:

1. if the message is destined to the Rx FIFO, push IDHIT into the RXFIR FIFO;
2. reads the words DATA0-3 and DATA4-7 from the Rx SMB;
3. writes it in the words DATA0-3 and DATA4-7 of the Rx Mailbox;
4. reads the words Control/Status and ID from the Rx SMB;

5. writes it in the words Control/Status and ID of the Rx Mailbox, updating the CODE field according to Table 15-4.

The move-in process is not atomic, in such a way that it is immediately cancelled by the inactivation of the destination Mailbox (see Message Buffer Inactivation) and in this case the Mailbox may be left partially updated, thus incoherent. The exception is if the move-in destination is an Rx FIFO Message Buffer, then the process cannot be cancelled.

The BUSY Bit (least significant bit of the CODE field) of the destination Message Buffer is asserted while the move-in is being performed in such a way that ARM beware that the Message Buffer content is temporarily incoherent.

15.1.6.6.2 Move-out

The move-out process is the copy of the content from a Tx Mailbox to the Tx SMB when a message for transmission is available (see Arbitration process).

The move-out occurs in the following conditions:

- the first bit of Intermission field;
- during Bus off field when TX Error Counter is in the 124 to 128 range;
- during BusIdle field
- during Wait For Bus Idle field

The move-out process is not atomic. Only ARM has priority to access the memory concurrently out of BusIdle state. In BusIdle, the move-out has the lowest priority to the concurrent memory accesses.

15.1.6.7 Data Coherence

In order to maintain data coherency and FlexCAN proper operation, the ARM must obey the rules described in

Any form of ARM accessing an MB structure within FlexCAN other than those specified may cause FlexCAN to behave in an unpredictable way.

15.1.6.7.1 Transmission Abort Mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform ARM if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

In order to abort a transmission, ARM must write a specific abort code (0b1001) to the CODE field of the Control and Status word. The active MBs configured as transmission must be aborted first and then they may be updated. If the abort code is written to a Mailbox that is currently being transmitted, or to a Mailbox that was already loaded into the SMB for transmission, the write operation is blocked and the MB is kept active, but the abort request is captured and kept pending until one of the following conditions are satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into Freeze Mode
- The module enters in BusOff state
- There is an overload frame

If none of conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the IFLAG register and an interrupt to the ARM is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. In the other hand, if one of the above conditions is reached, the frame is not transmitted, therefore the abort code is written into the CODE field, the interrupt flag is set in the IFLAG and an interrupt is (optionally) generated to ARM.

If ARM writes the ABORT code before the transmission begins internally, then the write operation is not blocked, therefore the MB is updated and the interrupt flag is set. In this way ARM just needs to read the abort code to make sure the active MB was *safely inactivated* . Although the AEN bit is asserted and ARM wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet. One Mailbox is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

1. ARM checks the corresponding IFLAG and clears it, if asserted.
2. ARM writes 0b1001 into the CODE field of the C/S word.
3. ARM waits for the corresponding IFLAG indicating that the frame was either transmitted or aborted.
4. ARM reads the CODE field to check if the frame was either transmitted (CODE=0b1000) or aborted (CODE=0b1001).
5. It is necessary to clear the corresponding IFLAG in order to allow the MB to be reconfigured.

15.1.6.7.2 Message Buffer Inactivation

Inactivation is a mechanism provided to protect the Mailbox against updates by the FlexCAN internal processes, thus allowing ARM to rely on Mailbox data coherence after having updated it, even in Normal Mode.

If a Mailbox is inactivated it does not participate neither in the arbitration nor in the matching process until it is reactivated. See Transmit Process and Receive Process for more detailed instruction on how to inactivate and reactivate a Mailbox.

In order to inactivate a Mailbox ARM must update its CODE field to INACTIVE (either 0b0000 or 0b1000).

As the user is not able to synchronize the CODE field update with the FlexCAN internal processes an inactivation can lead to undesirable results:

- a frame in the bus that matches the filtering of the inactivated Rx Mailbox may be lost without notice, even if there are other Mailboxes with the same filter;
- a frame containing the message within the inactivated Tx Mailbox may be transmitted without notice.

In order to eliminate such risk and perform a *safe inactivation* ARM must use the following mechanism along with the inactivation itself:

- for Tx Mailboxes, the Transmission Abort (see Transmission Abort Mechanism);

The inactivation automatically unlocks the Mailbox (see Message Buffer Lock Mechanism).

Message Buffers that are part of the Rx FIFO cannot be inactivated. There is no write protection on FIFO region by FlexCAN. ARM must keep the data coherence into FIFO region when RFEN is asserted.

15.1.6.7.3 Message Buffer Lock Mechanism

Besides MB inactivation, FlexCAN has another data coherence mechanism for the receive process. When ARM reads the Control and Status word of an Rx MB with codes FULL or OVERRUN, FlexCAN assumes that ARM wants to read the whole MB in an atomic operation, and thus it sets an internal lock flag for that MB. The lock is released when ARM reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB regardless of its code or when ARM writes into C/S word from locked MB. The MB locking is done to prevent a new frame to be written into the MB while ARM is reading it.

The locking mechanism only applies to Rx MBs that are not part of FIFO and have a code different than INACTIVE (0b0000) or EMPTY[1] (0b0100). Also, Tx MBs can not be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the ARM decides to read MB number 5 and at the same time another message with the same ID is arriving. When ARM reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no "free-to-receive" MBs, so it decides to override MB number 5. However, this MB is locked, so the new message can not be written there. It will remain in the SMB waiting for the MB to be unlocked, and only then will be written to the MB. If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the SMB and there will be no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved-in from the SMB to the MB, the BUSY bit on the CODE field is asserted. If ARM reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Inactivation takes precedence over locking. If ARM inactivates a locked Rx Mailbox, then its lock status is negated and the Mailbox is marked as invalid for the current matching round. Any pending message on the SMB will not be transferred anymore to the Mailbox.An MB is unlocked when ARM reads the Free Running Timer Register (see Free Running Timer Register (FLEXCAN_TIMER)), or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both Normal and Freeze modes.

An unlock during Normal or Freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during any of the low power modes (see in Modes of Operation specific information on Module Disable or Stop modes) and it will take place only when the module resumes to Normal or Freeze modes.

15.1.6.8 Rx FIFO

The receive-only FIFO is enabled by asserting the RFEN bit in the MCR.

1. In previous FlexCAN versions, reading the C/S word locks the MB even if it is EMPTY. This behavior is maintained when the IRMQ bit is negated.

The reset value of this bit is zero to maintain software backward compatibility with previous versions of the module that did not have the FIFO feature. The FIFO is 6-message deep, therefore when the FIFO is enabled, the memory region occupied by the first 6 Message Buffers is reserved for use of the FIFO engine (see Rx FIFO Structure). ARM can read the received messages sequentially, in the order they were received, by repeatedly reading a Message Buffer structure at the output of the FIFO.

The IFLAG[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, ARM can read the message (accessing the output of the FIFO as a Message Buffer) and the RXFIR register and then clear the interrupt. If there are more messages in the FIFO the act of clearing the interrupt updates the output of the FIFO with the next message and update the RXFIR with the attributes of that message, reissuing the interrupt to ARM. Otherwise, the flag remains negated. The output of the FIFO is only valid when the IFLAG[BUF5I] is asserted.

The IFLAG[BUF6I] (Rx FIFO Warning) is asserted when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. The flag remains asserted until ARM clears it.

The IFLAG[BUF7I] (Rx FIFO Overflow) is asserted when an incoming message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. The flag remains asserted until the ARM clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated if an IFLAG bit is asserted and the corresponding mask bit is asserted too.

A powerful filtering scheme is provided to accept only frames intended for the target application, thus reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to CTRL2[RFFN] setting, that can be configured to one of the following formats (see also Rx FIFO Structure):

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

Every frame available in the FIFO has a corresponding IDHIT (Identifier Acceptance Filter Hit Indicator) that can be read by accessing the RXFIR register. The RXFIR[IDHIT] field refers to the message at the output of the FIFO and is valid whilst

the IFLAG[BUF5I] flag is asserted. The RXFIR register must be read only before clearing the flag, which guarantees that the information refers to the correct frame within the FIFO.

Up to thirty two elements of the ID Filter Table are individually affected by the Individual Mask Registers (RXIMR0 - RXIMR31), according to CTRL2[RFFN] setting (refer to Control 2 Register (FLEXCAN_CTRL2)), allowing very powerful filtering criteria to be defined. If the MCR[IRMQ] bit is negated (or if the RXIMR are not available for the particular MCU), then the FIFO ID Filter Table is affected by RXFGMASK.

15.1.6.9 CAN Protocol Related Features

15.1.6.9.1 Remote Frames

Remote frame is a special kind of frame. The user can program a mailbox to be a Remote Request Frame by writing the mailbox as Transmit with the RTR bit set to '1'. After the remote request frame is transmitted successfully, the mailbox becomes a Receive Message Buffer, with the same ID as before.

When a remote request frame is received by FlexCAN, it can be treated in three ways, depending on Remote Request Storing (CTRL2[RRS]) and Rx FIFO Enable (MCR[RFEN]) bits:

- If RRS is negated the frame's ID is compared to the IDs of the Transmit Message Buffers with the CODE field 0b1010. If there is a matching ID, then this mailbox frame will be transmitted. Note that if the matching mailbox has the RTR bit set, then FlexCAN will transmit a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match. In the case that a remote request frame was received and matched a mailbox, this message buffer immediately enters the internal arbitration process, but is considered as normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.
- If RRS is asserted the frame's ID is compared to the IDs of the receive mailboxes with the CODE field 0b0100, 0b0010 or 0b0110. If there is a matching ID, then this mailbox will store the remote frame in the same fashion of a data frame. No

automatic remote response frame will be generated. The mask registers are used in the matching process.

- If RFEN is asserted FlexCAN will not generate an automatic response for remote request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the ARM. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID).Remote Request Frames are considered as normal frames, and generate a FIFO overflow when a successful reception occurs and the FIFO is already full.

15.1.6.9.2 Overload Frames

FLEXCAN does transmit overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

15.1.6.9.3 Time Stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of "move-in" in the TIME STAMP field, providing network behavior with respect to time.

Note that the Free Running Timer can be reset upon a specific frame reception, enabling network time synchronization. Refer to TSYN description in Control 1 Register (FLEXCAN_CTRL1).

15.1.6.9.4 Protocol Timing

The FLEXCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The Control Register has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW. See Control 1 Register (FLEXCAN_CTRL1) .

The PRESDIV field controls a prescaler that generates the Serial Clock (Sclock), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.

$$f_{Tq} = \frac{f_{CANCLK}}{(\text{Prescaler value})}$$

A bit time is subdivided into three segments[2] (reference Table 15-16):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section
- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL Register so that their sum (plus 2) is in the range of 4 to 16 time quanta
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL Register (plus 1) to be 2 to 8 time quanta long

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})}$$

2. For further explanation of the underlying concepts please refer to ISO/DIS 11519-1, Section 10.3. Reference also the Bosch CAN 2.0A/B protocol specification dated September 1991 for bit timing.

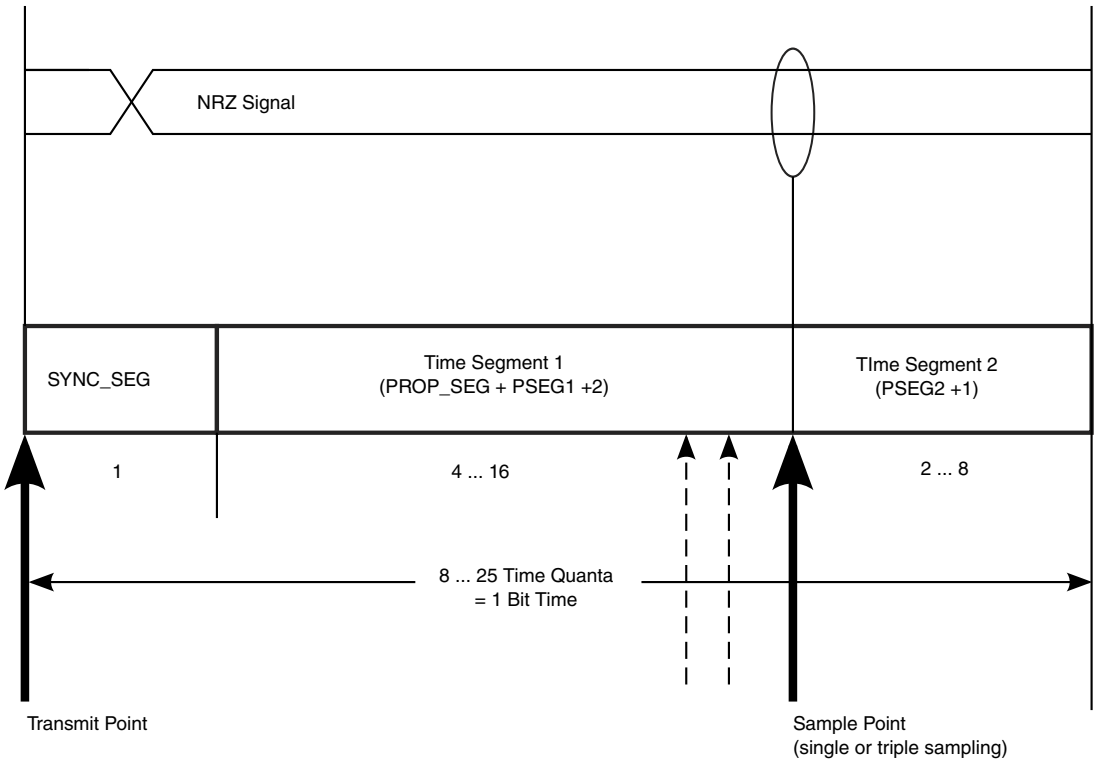


Figure 15-2. Segments within the Bit Time

Whenever CAN bit is used as a measure of duration (e.g. MCR[FRZ_ACK] and MCR[LPM_ACK] in Module Configuration Register (FLEXCAN_MCR)), the number of peripheral clocks in one CAN bit can be calculated as:

$$NCCP = \frac{f_{sys} \times [1 + (PSEG1 + 1) + (PSEG2 + 1) + (PROPSEG + 1)] \times (PRESDIV + 1)}{f_{CANCLK}}$$

where:

NCCP is the number of peripheral clocks in one CAN bit;

f_{CANCLK} is the Protocol Engine (PE) Clock in Hz;

f_{SYS} is the frequency of operation of the system (CHI) clock, in Hz;

PSEG1 is the value in CTRL1[PSEG1] field;

PSEG2 is the value in CTRL1[PSEG2] field;

PROPSEG is the value in CTRL1[PROPSEG] field;

PRESDIV is the value in CTRL1[PRESDIV] field.

For example, 180 CAN bits = 180 x NCCP peripheral clock periods.

Figure 15-2 gives an overview of the CAN compliant segment settings and the related parameter values.

Table 15-16. Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

Table 15-17. CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	Time Segment 2	Re-synchronization Jump Width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

15.1.6.9.5 Arbitration and Matching Timing

During normal reception and transmission of frames, the matching, arbitration, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in the following figures.

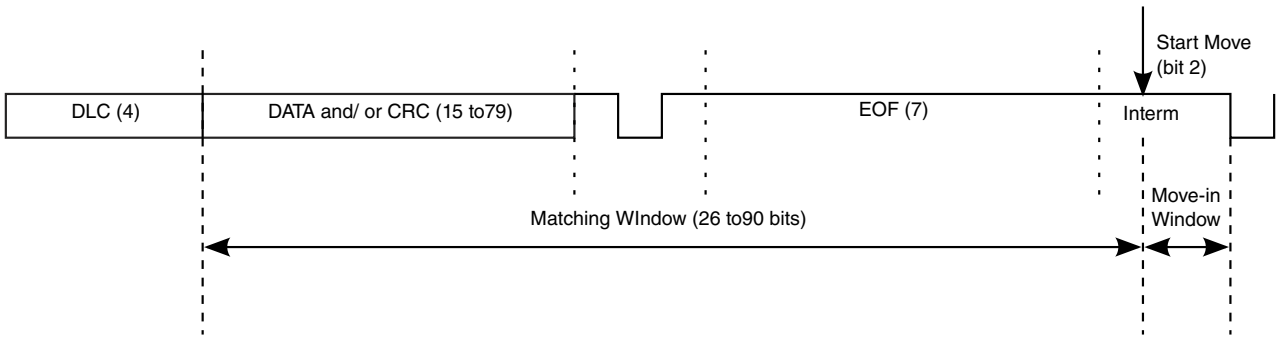


Figure 15-3. Matching and Move-In Time Windows

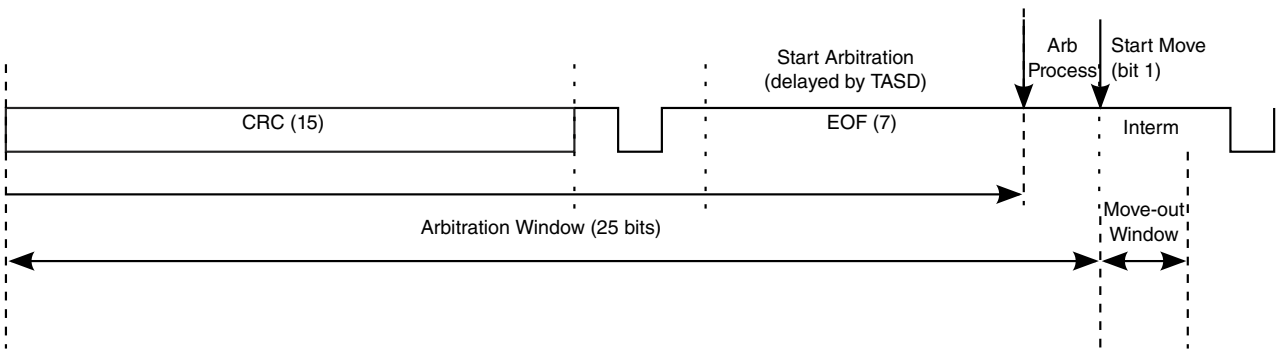


Figure 15-4. Arbitration and Move-Out Time Windows

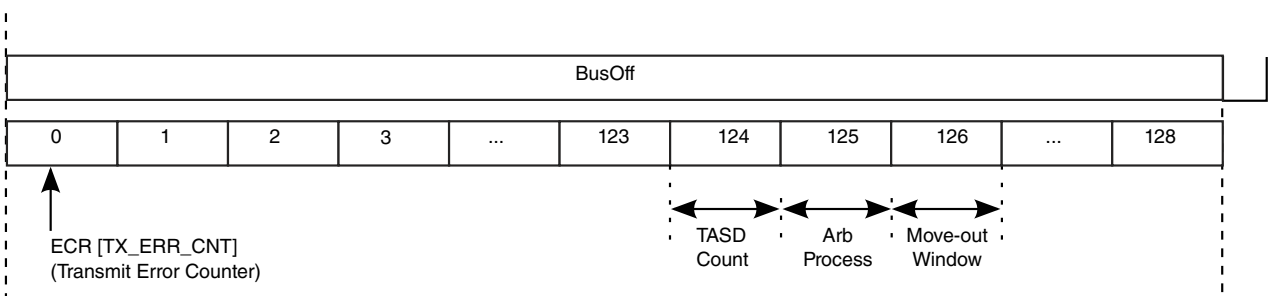


Figure 15-5. Arbitration at the end of Bus Off and Move-Out Time Windows

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the available time window. In order to have sufficient time to do that, the following requirements must be observed:

- A valid CAN bit timing must be programmed, as indicated in Table 15-17
- The peripheral clock frequency can not be smaller than the oscillator clock frequency, i.e. the PLL can not be programmed to divide down the oscillator clock
- There must be a minimum ratio between the peripheral clock frequency and the CAN bit rate, as specified in the following table.

Table 15-18. Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate

Number of Message Buffers	RFEN	Minimum Number of Peripheral Clocks per CAN bit
16 and 32	0	16
64	0	25
16	1	16
32	1	17
64	1	30

A direct consequence of the first requirement is that the minimum number of time quanta per CAN bit must be 8, so the oscillator clock frequency should be at least 8 times the CAN bit rate. The minimum frequency ratio specified in Table 15-18 can be achieved by choosing a high enough peripheral clock frequency when compared to the oscillator clock frequency, or by adjusting one or more of the bit timing parameters (PRESDIV, PROPSEG, PSEG1, PSEG2). As an example, taking the case of 64 MBs, if the oscillator and peripheral clock frequencies are equal and the CAN bit timing is programmed to have 8 time quanta per bit, then the prescaler factor (PRESDIV + 1) should be at least 2. For prescaler factor equal to one and CAN bit timing with 8 time quanta per bit, the ratio between peripheral and oscillator clock frequencies should be at least 2.

15.1.6.10 Modes of Operation Details

The FlexCAN module has four functional modes (Normal Mode, Freeze Mode, Listen-Only Mode and Loop-Back Mode) and two low power modes (Disable Mode and Stop Mode).

See in Modes of Operation an introductory description of all these modes of operation. The following sub-sections bring functional details on Freeze mode and the low power modes.

15.1.6.10.1 Freeze Mode

This mode is requested by ARM through the assertion of the HALT bit in the MCR Register or when the MCU is put into Debug Mode . In both cases it is also necessary that the FRZ bit is asserted in the MCR Register and the module is not in any of the low power modes (Disable, Stop). The acknowledgement is obtained through the assertion by the FlexCAN of FRZ_ACK bit in the same register. The ARM must only consider the FlexCAN in Freeze Mode when both request and acknowledgement conditions are satisfied.

When Freeze Mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off or Idle state
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. Pending move-in is not taken in account
- Ignores the FLEXCAN_RX input pin and drives the FLEXCAN_TX pin as recessive
- Stops the prescaler, thus halting all CAN protocol activities
- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in MCR

After requesting Freeze Mode, the user must wait for the FRZ_ACK bit to be asserted in MCR before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible, except for CTRL1[CLK_SRC] bit that can be read but cannot be written.

Exiting Freeze Mode is done in one of the following ways:

- ARM negates the FRZ bit in the MCR Register
- The ARM is removed from Debug Mode and the HALT bit is negated

The FRZ_ACK bit is negated after protocol engine recognizes the negation of freeze request. Once out of Freeze Mode, FlexCAN tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

15.1.6.10.2 Module Disable Mode

This low power mode is normally used to temporarily disable a complete FlexCAN block, with no power consumption. It is requested by the ARM through the assertion of the MDIS bit in the MCR Register and the acknowledgement is obtained through the assertion by the FlexCAN of the LPM_ACK bit in the same register. The ARM must only consider the FlexCAN in Disable Mode when both request and acknowledgement conditions are satisfied.

If the module is disabled during Freeze Mode, it requests to disable the clocks to the PE and CHI sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit. The ability to shut down the clocks depends on how FlexCAN is integrated into the MCU. If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. Pending move-in is not taken in account
- Ignores its FLEXCAN_RX input pin and drives its FLEXCAN_TX pin as recessive

- May shut down the clocks to the PE and CHI sub-modules, depending on how FlexCAN is integrated into the MCU
- Sets the NOT_RDY and LPM_ACK bits in MCR

The Bus Interface Unit continues to operate, enabling the ARM to access memory mapped registers, except the Rx Mailboxes Global Mask Registers, the Rx Buffer 14 Mask Register, the Rx Buffer 15 Mask Register, the Rx FIFO Global Mask Register. The Rx FIFO Information Register, the Message Buffers, the Rx Individual Mask Registers, and the reserved words within RAM may not be accessed when the module is in Disable Mode depending on how FlexCAN RAM is integrated into the ARM. Exiting from this mode is done by negating the MDIS bit by ARM, which make FlexCAN requests to resume the clocks and negates the LPM_ACK bit after CAN protocol engine recognizes the negation of disable mode requested by ARM.

15.1.6.10.3 Stop Mode

This is a system low power mode in which system clocks can be stopped for maximum power savings.. To enter stop mode, the CPU should manually assert a global Stop Mode request (see the CAN1_STOP_REQ and CAN2_STOP_REQ bit in the register IOMUXC_GPR4)and check the acknowledgement asserted by the FlexCAN (see the CAN1_STOP_ACK and CAN2_STOP_ACK in the register IOMUXC_GPR4) . The CPU must only consider the FlexCAN in Stop Mode when both request and acknowledgement conditions are satisfied.

If FlexCAN receives the global Stop Mode request during Freeze Mode, it sets the LPM_ACK bit, negates the FRZ_ACK bit and then sends the Stop Acknowledge signal to the CPU, in order to shut down the clocks globally. If Stop Mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish.Pending move-in is not taken in account
- Ignores its FLEXCAN_RX input pin and drives its FLEXCAN_TX pin as recessive
- Sets the NOT_RDY and LPM_ACK bits in MCR
- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Exiting Stop Mode is done in one of the following ways:

- ARM resuming the clocks and removing the Stop Mode request
- ARM resuming the clocks and Stop Mode request as a result of the Self Wake mechanism

In the Self Wake mechanism, if the SLF_WAK bit in MCR Register was set at the time FlexCAN entered Stop Mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN sets the WAK_INT bit in the ESR Register and, if enabled by the WAK_MSK bit in MCR, generates a Wake Up interrupt to the ARM. Upon receiving the interrupt, the ARM should resume the clocks and remove the Stop Mode request manually. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up.

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the FLEXCAN_RX input line while in Stop Mode. See the WAK_SRC bit in Module Configuration Register (FLEXCAN_MCR) . This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments, the glitch filter width can be set in Glitch Filter Width Register (FLEXCAN_GFWR).

15.1.6.11 Interrupts

The module can generate up to 70 interrupt sources (64 interrupts due to message buffers and 6 interrupts due to Ored interrupts from MBs, Bus Off, Error, Tx Warning, Rx Warning and Wake Up)).

The number of actual sources depends on the configured number of message buffers.

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has assigned a flag bit in the IFLAG Registers. The bit is set when the corresponding buffer completes a successful transmission/reception and is cleared when the ARM writes it to '1' (unless another interrupt is generated at the same time).

If the Rx FIFO is enabled (bit RFEN on MCR set), the interrupts corresponding to MBs 0 to 7 have a different behavior. Bit 7 of the IFLAG1 becomes the "FIFO Overflow" flag; bit 6 becomes the FIFO Warning flag, bit 5 becomes the "Frames Available in FIFO flag" and bits 4-0 are unused. See Interrupt Flags 1 Register (FLEXCAN_IFLAG1) for more information.

A combined interrupt for all MBs is also generated by an Or of all the interrupt sources from MBs. This interrupt gets generated when any of the Mailboxes or FIFO generates an interrupt. The ARM must read the IFLAG Registers to determine which MB or FIFO caused the interrupt.

The other 5 interrupt sources (Bus Off, Error, Tx Warning, Rx Warning and Wake Up generate interrupts like the MB ones, and can be read from both the Error and Status Register 1 and 2. The Bus Off, Error, Tx Warning and Rx Warning interrupt mask bits are located in the Control 1 Register and the Wake-Up interrupt mask bit is located in the MCR.

15.1.7 Initialization/Application Information

This section provide instructions for initializing the FLEXCAN module.

15.1.7.1 FLEXCAN Initialization Sequence

The FLEXCAN module may be reset in two ways:

- SOC level hard reset which resets all memory mapped registers asynchronously
- SOFT_RST bit in MCR, which resets some of the memory mapped registers synchronously

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The SOFT_RST bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in any of the low power modes. The low power mode should be exited and the clocks resumed before applying soft reset.

After the module is enabled (MDIS bit negated), FLEXCAN automatically goes to Freeze Mode. In Freeze Mode, FLEXCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in MCR Register are set, the internal state machines are disabled and the FRZ_ACK and NOT_RDY bits in the MCR Register are set. The FLEXCAN_TX pin is in recessive state and FLEXCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FLEXCAN is put into Freeze Mode. The following is a generic initialization sequence applicable to the FLEXCAN module:

- Initialize the Module Configuration Register
  - Enable the individual filtering per MB and reception queue features by setting the IRMQ bit
  - Enable the warning interrupts by setting the WRN_EN bit
  - If required, disable frame self reception by setting the SRX_DIS bit

    - Enable the FIFO by setting the RFEN bit
    - Enable the abort mechanism by setting the AEN bit
    - Enable the local priority feature by setting the LPRIO_EN bit
- Initialize the Control Register
    - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW
    - Determine the bit rate by programming the PRESDIV field
    - Determine the internal arbitration mode (LBUF bit)
- Initialize the Message Buffers
    - The Control and Status word of all Message Buffers must be initialized
    - If FIFO was enabled, the 8-entry ID table must be initialized
    - Other entries in each Message Buffer should be initialized as required
- Initialize the Rx Individual Mask Registers
- Set required interrupt mask bits in the IMASK Registers (for all MB interrupts), in CTRL Register (for Bus Off and Error interrupts) and in MCR Register for Wake-Up interrupt
- Negate the HALT bit in MCR

Starting with the last event, FLEXCAN attempts to synchronize to the CAN bus.

15.1.8 FLEXCAN Memory Map/Register Definition

The complete memory map for a FLEXCAN module with 64 MBs capability is shown in the following table. Each individual register is identified by its complete name and the corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor or Unrestricted access by programming the SUPV bit in the MCR Register. The MCR register allows only Supervisor access regardless the SUPV bit state.

The FLEXCAN module stores CAN messages for transmission and reception using a Mailboxes and Rx FIFO structure.

FLEXCAN memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30A0_0000	Module Configuration Register (FLEXCAN1_MCR)	32	R/W	5980_000Fh	15.1.8.1/ 4222
30A0_0004	Control 1 Register (FLEXCAN1_CTRL1)	32	R/W	0000_0000h	15.1.8.2/ 4227

Table continues on the next page...

FLEXCAN memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30A0_0008	Free Running Timer Register (FLEXCAN1_TIMER)	32	R/W	0000_0000h	15.1.8.3/ 4230
30A0_0010	Rx Mailboxes Global Mask Register (FLEXCAN1_RXMGMASK)	32	R/W	FFFF_FFFFh	15.1.8.4/ 4230
30A0_0014	Rx Buffer 14 Mask Register (FLEXCAN1_RX14MASK)	32	R/W	FFFF_FFFFh	15.1.8.5/ 4231
30A0_0018	Rx Buffer 15 Mask Register (FLEXCAN1_RX15MASK)	32	R/W	FFFF_FFFFh	15.1.8.6/ 4232
30A0_001C	Error Counter Register (FLEXCAN1_ECR)	32	R/W	0000_0000h	15.1.8.7/ 4233
30A0_0020	Error and Status 1 Register (FLEXCAN1_ESR1)	32	R/W	0000_0000h	15.1.8.8/ 4234
30A0_0024	Interrupt Masks 2 Register (FLEXCAN1_IMASK2)	32	R/W	0000_0000h	15.1.8.9/ 4238
30A0_0028	Interrupt Masks 1 Register (FLEXCAN1_IMASK1)	32	R/W	0000_0000h	15.1.8.10/ 4238
30A0_002C	Interrupt Flags 2 Register (FLEXCAN1_IFLAG2)	32	R/W	0000_0000h	15.1.8.11/ 4239
30A0_0030	Interrupt Flags 1 Register (FLEXCAN1_IFLAG1)	32	R/W	0000_0000h	15.1.8.12/ 4239
30A0_0034	Control 2 Register (FLEXCAN1_CTRL2)	32	R/W	0000_0000h	15.1.8.13/ 4241
30A0_0038	Error and Status 2 Register (FLEXCAN1_ESR2)	32	R	0000_0000h	15.1.8.14/ 4247
30A0_0044	CRC Register (FLEXCAN1_CRCR)	32	R	0000_0000h	15.1.8.15/ 4249
30A0_0048	Rx FIFO Global Mask Register (FLEXCAN1_RXFGMASK)	32	R/W	FFFF_FFFFh	15.1.8.16/ 4250
30A0_004C	Rx FIFO Information Register (FLEXCAN1_RXFIR)	32	R	0000_0000h	15.1.8.17/ 4251
30A0_0880	Rx Individual Mask Registers (FLEXCAN1_RXIMR0_RXIMR63)	32	R/W	0000_0000h	15.1.8.18/ 4252
30A0_09E0	Glitch Filter Width Registers (FLEXCAN1_GFWR)	32	R/W	0000_007Fh	15.1.8.19/ 4252
30A1_0000	Module Configuration Register (FLEXCAN2_MCR)	32	R/W	5980_000Fh	15.1.8.1/ 4222
30A1_0004	Control 1 Register (FLEXCAN2_CTRL1)	32	R/W	0000_0000h	15.1.8.2/ 4227
30A1_0008	Free Running Timer Register (FLEXCAN2_TIMER)	32	R/W	0000_0000h	15.1.8.3/ 4230
30A1_0010	Rx Mailboxes Global Mask Register (FLEXCAN2_RXMGMASK)	32	R/W	FFFF_FFFFh	15.1.8.4/ 4230
30A1_0014	Rx Buffer 14 Mask Register (FLEXCAN2_RX14MASK)	32	R/W	FFFF_FFFFh	15.1.8.5/ 4231

Table continues on the next page...

FLEXCAN memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30A1_0018	Rx Buffer 15 Mask Register (FLEXCAN2_RX15MASK)	32	R/W	FFFF_FFFFh	15.1.8.6/ 4232
30A1_001C	Error Counter Register (FLEXCAN2_ECR)	32	R/W	0000_0000h	15.1.8.7/ 4233
30A1_0020	Error and Status 1 Register (FLEXCAN2_ESR1)	32	R/W	0000_0000h	15.1.8.8/ 4234
30A1_0024	Interrupt Masks 2 Register (FLEXCAN2_IMASK2)	32	R/W	0000_0000h	15.1.8.9/ 4238
30A1_0028	Interrupt Masks 1 Register (FLEXCAN2_IMASK1)	32	R/W	0000_0000h	15.1.8.10/ 4238
30A1_002C	Interrupt Flags 2 Register (FLEXCAN2_IFLAG2)	32	R/W	0000_0000h	15.1.8.11/ 4239
30A1_0030	Interrupt Flags 1 Register (FLEXCAN2_IFLAG1)	32	R/W	0000_0000h	15.1.8.12/ 4239
30A1_0034	Control 2 Register (FLEXCAN2_CTRL2)	32	R/W	0000_0000h	15.1.8.13/ 4241
30A1_0038	Error and Status 2 Register (FLEXCAN2_ESR2)	32	R	0000_0000h	15.1.8.14/ 4247
30A1_0044	CRC Register (FLEXCAN2_CRCR)	32	R	0000_0000h	15.1.8.15/ 4249
30A1_0048	Rx FIFO Global Mask Register (FLEXCAN2_RXFGMASK)	32	R/W	FFFF_FFFFh	15.1.8.16/ 4250
30A1_004C	Rx FIFO Information Register (FLEXCAN2_RXFIR)	32	R	0000_0000h	15.1.8.17/ 4251
30A1_0880	Rx Individual Mask Registers (FLEXCAN2_RXIMR0_RXIMR63)	32	R/W	0000_0000h	15.1.8.18/ 4252
30A1_09E0	Glitch Filter Width Registers (FLEXCAN2_GFWR)	32	R/W	0000_007Fh	15.1.8.19/ 4252

15.1.8.1 Module Configuration Register (FLEXCANx_MCR)

This register defines global system configurations, such as the module operation mode (e.g., low power) and maximum message buffer configuration.

Address: Base address + 0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MDIS	FRZ	RFEN	HALT	NOT_RDY	WAK_MSK	SOFT_RST	FRZ_ACK	SUPV	SLF_WAK	WRN_EN	LPM_ACK	WAK_SRC	Reserved	SRX_DIS	IRMQ
W																
Reset	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		LPRIO_EN	AEN	Reserved		IDAM		Reserved	MAXMB						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

FLEXCANx_MCR field descriptions

Field	Description
31 MDIS	This bit controls whether FLEXCAN is enabled or not. When disabled, FLEXCAN shuts down the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules. This is the only bit in MCR not affected by soft reset. See Module Disable Mode for more information. 1 Disable the FLEXCAN module 0 Enable the FLEXCAN module
30 FRZ	The FRZ bit specifies the FLEXCAN behavior when the HALT bit in the MCR Register is set or when Debug Mode is requested at ARM level. When FRZ is asserted, FLEXCAN is enabled to enter Freeze Mode. Negation of this bit field causes FLEXCAN to exit from Freeze Mode. 1 Enabled to enter Freeze Mode 0 Not enabled to enter Freeze Mode
29 RFEN	This bit controls whether the Rx FIFO feature is enabled or not. When RFEN is set, MBs 0 to 5 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xDC) is used by the FIFO engine as well as additional MBs (up to 32, depending on CTRL2[RFFN] setting) which are used as Rx FIFO ID Filter Table elements.RFEN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table 15-18 (see Arbitration and Matching Timing).This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 FIFO enabled 0 FIFO not enabled
28 HALT	Assertion of this bit puts the FLEXCAN module into Freeze Mode. The ARM should clear it after initializing the Message Buffers and Control Register. No reception or transmission is performed by FLEXCAN before this bit is cleared. Freeze Mode can not be entered while FLEXCAN is in any of the low power modes.See Freeze Mode for more information 1 Enters Freeze Mode if the FRZ bit is asserted. 0 No Freeze Mode request.
27 NOT_RDY	This read-only bit indicates that FLEXCAN is either in Disable Mode, Stop Mode or Freeze Mode. It is negated once FLEXCAN has exited these modes. 1 FLEXCAN module is either in Disable Mode, Stop Mode or Freeze Mode 0 FLEXCAN module is either in Normal Mode, Listen-Only Mode or Loop-Back Mode
26 WAK_MSK	This bit enables the Wake Up Interrupt generation. 1 Wake Up Interrupt is enabled 0 Wake Up Interrupt is disabled
25 SOFT_RST	When this bit is asserted, FlexCAN resets its internal state machines and some of the memory mapped registers. The following registers are reset: MCR (except the MDIS bit), TIMER, ECR, ESR1, ESR2, IMASK1, IMASK2, IFLAG1, IFLAG2 and CRCR. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected: CTRL1, CTRL2, RXIMR0_RXIMR63, RXGMASK, RX14MASK, RX15MASK, RXFGMASK, RXFIR and all Message Buffers The SOFT_RST bit can be asserted directly by the ARM when it writes to the MCR Register. It may take some time to fully propagate its effect. The SOFT_RST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in any of the low power modes. The module should be first removed from low power mode, and then soft reset can be applied.

Table continues on the next page...

FLEXCANx_MCR field descriptions (continued)

Field	Description
	1 Reset the registers 0 No reset request
24 FRZ_ACK	This read-only bit indicates that FLEXCAN is in Freeze Mode and its prescaler is stopped. The Freeze Mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZ_ACK bit to know when FLEXCAN has actually entered Freeze Mode. If Freeze Mode request is negated, then this bit is negated once the FLEXCAN prescaler is running again. If Freeze Mode is requested while FLEXCAN is in any of the low power modes, then the FRZ_ACK bit will only be set when the low power mode is exited. See Freeze Mode for more information 1 FLEXCAN in Freeze Mode, prescaler stopped 0 FLEXCAN not in Freeze Mode, prescaler running
23 SUPV	This bit configures some of the FLEXCAN registers to be either in Supervisor or User Mode. Reset value of this bit is '1', so the affected registers start with Supervisor access allowance only. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 FlexCAN is in Supervisor Mode. Affected registers allow only Supervisor access. Unrestricted access behaves as though the access was done to an unimplemented register location 0 FlexCAN is in User Mode. Affected registers allow both Supervisor and Unrestricted accesses
22 SLF_WAK	This bit enables the Self Wake Up feature when FLEXCAN is in Stop Mode. If this bit had been asserted by the time FLEXCAN entered Stop Mode, then FLEXCAN will look for a recessive to dominant transition on the bus during these modes. If a transition from recessive to dominant is detected during Stop Mode, then FLEXCAN generates, if enabled to do so, a Wake Up interrupt to the ARM so that it can resume the clocks globally and FlexCAN can request to resume the clocks. This bit can not be written while the module is in Stop Mode. 1 FLEXCAN Self Wake Up feature is enabled 0 FLEXCAN Self Wake Up feature is disabled
21 WRN_EN	When asserted, this bit enables the generation of the TWRN_INT and RWRN_INT flags in the Error and Status Register. If WRN_EN is negated, the TWRN_INT and RWRN_INT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated.This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 TWRN_INT and RWRN_INT bits are set when the respective error counter transition from <96 to ≥ 96. 0 TWRN_INT and RWRN_INT bits are zero, independent of the values in the error counters.
20 LPM_ACK	This read-only bit indicates that FLEXCAN is either in Disable Mode or Stop Mode. Either of these low power modes can not be entered until all current transmission or reception processes have finished, so the ARM can poll the LPM_ACK bit to know when FLEXCAN has actually entered low power mode. See Module Disable Mode, and Stop Mode for more information 1 FLEXCAN is either in Disable Mode, or Stop mode 0 FLEXCAN not in any of the low power modes
19 WAK_SRC	This bit defines whether the integrated low-pass filter is applied to protect the FLEXCAN_RX input from spurious wake up. See Stop Mode for more information. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 FLEXCAN uses the filtered FLEXCAN_RX input to detect recessive to dominant edges on the CAN bus 0 FLEXCAN uses the unfiltered FLEXCAN_RX input to detect recessive to dominant edges on the CAN bus.
18 -	This field is reserved. Reserved

Table continues on the next page...

FLEXCANx_MCR field descriptions (continued)

Field	Description
17 SRX_DIS	This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Self reception disabled 0 Self reception enabled
16 IRMQ	This bit indicates whether Rx matching process will be based either on individual masking and queue or on masking scheme with RXMGMASK, RX14MASK and RX15MASK, RXFGMASK. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Individual Rx masking and queue feature are enabled. 0 Individual Rx masking and queue feature are disabled.For backward compatibility, the reading of C/S word locks the MB even if it is EMPTY.
15–14 -	This field is reserved. Reserved
13 LPRIO_EN	This bit is provided for backwards compatibility reasons. It controls whether the local priority feature is enabled or not. It is used to extend the ID used during the arbitration process. With this extended ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames.This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Local Priority enabled 0 Local Priority disabled
12 AEN	This bit is supplied for backwards compatibility reasons. When asserted, it enables the Tx abort feature. This feature guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. This bit can only be written in Freeze mode as it is blocked by hardware in other modes.Write Abort code into Rx Mailboxes can cause unpredictable results when the MCR[AEN] is asserted. 1 Abort enabled 0 Abort disabled
11–10 -	This field is reserved. Reserved
9–8 IDAM	This 2-bit field identifies the format of the elements of the Rx FIFO filter table, as shown below. Note that all elements of the table are configured at the same time by this field (they are all the same format). See Rx FIFO Structure. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 00 Format A One full ID (standard or extended) per ID filter Table element. 01 Format B Two full standard IDs or two partial 14-bit extended IDs per ID filter Table element. 10 Format C Four partial 8-bit IDs (standard or extended) per ID filter Table element. 11 Format D All frames rejected.
7 -	This field is reserved. Reserved
MAXMB	This 7-bit field defines the number of the last Message Buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to 16 MB configuration. This field can only be written in Freeze Mode as it is blocked by hardware in other modes Number of the last MB = MAXMB.

Table continues on the next page...

FLEXCANx_MCR field descriptions (continued)

Field	Description
	NOTE: Additionally, the value of MAXMB must encompass the FIFO size defined by CTRL2[RFFN] MAXMB also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table 15-18 (see Arbitration and Matching Timing).

15.1.8.2 Control 1 Register (FLEXCANx_CTRL1)

This register is defined for specific FLEXCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop Back Mode, Listen Only Mode, Bus Off recovery behavior and interrupt enabling (Bus-Off, Error, Warning). It also determines the Division Factor for the clock prescaler.

Address: Base address + 4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PRESDIV								RJW		PSEG1			PSEG2		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BOFF_MSK	ERR_MSK	Reserved	LPB	TWRN_MSK	RWRN_MSK	Reserved		SMP	BOFF_REC	TSYN	LBUF	LOM	PROP_SEG		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_CTRL1 field descriptions

Field	Description
31–24 PRESDIV	This 8-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The Maximum value of this register is 0xFF, that gives a minimum Sclock frequency equal to the PE clock frequency divided by 256.For more information refer to Protocol Timing. This field can only be written in Freeze mode as it is blocked by hardware in other modes.

Table continues on the next page...

FLEXCANx_CTRL1 field descriptions (continued)

Field	Description
	Sclock frequency = CPI clock frequency / (PRESDIV+1)
23–22 RJW	This 2-bit field defines the maximum number of time quanta[1] that a bit time can be changed by one re-synchronization. The valid programmable values are 0-3. This field can only be written in Freeze mode as it is blocked by hardware in other modes Resync Jump Width = RJW + 1.
21–19 PSEG1	This 3-bit field defines the length of Phase Buffer Segment 1 in the bit time. The valid programmable values are 0-7. This field can only be written in Freeze mode as it is blocked by hardware in other modes Phase Buffer Segment 1 = (PSEG1 + 1) x Time-Quanta.
18–16 PSEG2	This 3-bit field defines the length of Phase Buffer Segment 2 in the bit time. The valid programmable values are 1-7. This field can only be written in Freeze mode as it is blocked by hardware in other modes Phase Buffer Segment 2 = (PSEG2 + 1) x Time-Quanta.
15 BOFF_MSK	This bit provides a mask for the Bus Off Interrupt. 1 Bus Off interrupt enabled 0 Bus Off interrupt disabled
14 ERR_MSK	This bit provides a mask for the Error Interrupt. 1 Error interrupt enabled 0 Error interrupt disabled
13 -	This field is reserved. Reserved
12 LPB	This bit configures FlexCAN to operate in Loop-Back Mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The FLEXCAN_RX input pin is ignored and the FLEXCAN_TX output goes to the recessive state (logic '1'). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Loop Back enabled 0 Loop Back disabled
11 TWRN_MSK	This bit provides a mask for the Tx Warning Interrupt associated with the TWRN_INT flag in the Error and Status Register. This bit is read as zero when MCR[WRN_EN] bit is negated. This bit can only be written if MCR[WRN_EN] bit is asserted. 1 Tx Warning Interrupt enabled 0 Tx Warning Interrupt disabled
10 RWRN_MSK	This bit provides a mask for the Rx Warning Interrupt associated with the RWRN_INT flag in the Error and Status Register. This bit is read as zero when MCR[WRN_EN] bit is negated. This bit can only be written if MCR[WRN_EN] bit is asserted. 1 Rx Warning Interrupt enabled 0 Rx Warning Interrupt disabled
9–8 -	This field is reserved. Reserved
7 SMP	This bit defines the sampling mode of CAN bits at the FLEXCAN_RX. This bit can only be written in Freeze mode as it is blocked by hardware in other modes.

Table continues on the next page...

FLEXCANx_CTRL1 field descriptions (continued)

Field	Description
	1 Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples, a majority rule is used 0 Just one sample is used to determine the bit value
6 BOFF_REC	This bit defines how FLEXCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFF_REC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FLEXCAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFF_REC bit can be re-asserted again during Bus Off, but it will only be effective the next time the module enters Bus Off. If BOFF_REC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery. 1 Automatic recovering from Bus Off state disabled 0 Automatic recovering from Bus Off state enabled, according to CAN Spec 2.0 part B
5 TSYN	This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FLEXCAN stations with a special "SYNC" message (i.e., global network time). If the RFEN bit in MCR is set (FIFO enabled), the first available Mailbox, according to CTRL2[RFFN] setting, is used for timer synchronization instead of MB0.This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Timer Sync feature enabled 0 Timer Sync feature disabled
4 LBUF	This bit defines the ordering mechanism for Message Buffer transmission. When asserted, the LPRIO_EN bit does not affect the priority arbitration.This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Lowest number buffer is transmitted first 0 Buffer with highest priority is transmitted first
3 LOM	This bit configures FLEXCAN to operate in Listen Only Mode. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FLEXCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message. Listen-Only Mode acknowledgement can be obtained by the state of ESR1[FLT_CONF] field which is Passive Error when Listen-Only Mode is entered. There can be some delay between the Listen-Only Mode request and acknowledge. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 FLEXCAN module operates in Listen Only Mode 0 Listen Only Mode is deactivated
PROP_SEG	This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0-7. This field can only be written in Freeze mode as it is blocked by hardware in other modes Propagation Segment Time = (PROPSEG + 1) * Time-Quanta. Time-Quantum = one Sclock period.

1. One time quantum is equal to the Sclock period.

15.1.8.3 Free Running Timer Register (FLEXCANx_TIMER)

This register represents a 16-bit free running counter that can be read and written by the ARM. The timer starts from $0000 after Reset, counts linearly to $FFFF, and wraps around.

The timer is clocked by the FLEXCAN bit-clock (which defines the baud rate on the CAN bus). During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. During Freeze Mode, disable, and stop mode, the timer is not incremented.

The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

If bit CTRL1[TSYN] is asserted the Timer is reset whenever a message is received in the first available Mailbox, according to CTRL2[RFFN] setting.

ARM can write to this register anytime. However, if the write occurs at the same time that the Timer is being reset by a reception in the first Mailbox, then the write value is discarded.

Reading this register affects the Mailbox Unlocking procedure. For additional details, refer to Message Buffer Lock Mechanism.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																TIMER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_TIMER field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
TIMER	TIMER

15.1.8.4 Rx Mailboxes Global Mask Register (FLEXCANx_RXMGMASK)

RXMGMASK is provided for legacy support. Asserting the MCR[IRMQ] bit causes the RXMGMASK Register to have no effect on the module operation.

RXMGMASK is used to mask the filter fields of all Rx MBs, excluding MBs 14-15, which have individual mask registers.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Table 15-19. Rx Mailboxes Global Mask usage

SMB[RTR][1]	CTRL2[RRS]	CTRL2[EACEN]	Mailbox filter fields			
			MB[RTR]	MB[IDE]	MB[ID]	reserved
0	-	0	- Note[2]	- Note[3]	MG[28:0]	MG[31:29]
0	-	1	MG[31]	MG[30]	MG[28:0]	MG[29]
1	0	-	-	-	-	MG[31:0]
1	1	0	-	-	MG[28:0]	MG[31:29]
1	1	1	MG[31]	MG[30]	MG[28:0]	MG[29]

1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If CTRL2[EACEN] bit is negated the RTR bit of Mailbox is never compared with the RTR bit of the Incoming Frame (Rx SMB[RTR]).
3. If CTRL2[EACEN] bit is negated the IDE bit of Mailbox is always compared with the IDE bit of the Incoming Frame (Rx SMB[IDE]).

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MG31_MG0																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FLEXCANx_RXMGMASK field descriptions

Field	Description
MG31_MG0	These bits mask the Mailbox filter bits as shown in the figure above. Note that the alignment with the ID word of the Mailbox is not perfect as the two most significant MG bits affect the fields RTR and IDE which are located in the Control and Status word of the Mailbox. Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK) shows in detail which MG bits mask each Mailbox filter field. 1 The corresponding bit in the filter is checked against the one received 0 the corresponding bit in the filter is "don't care"

15.1.8.5 Rx Buffer 14 Mask Register (FLEXCANx_RX14MASK)

RX14MASK is provided for legacy support, asserting the MCR[IRMQ] bit causes the RX14MASK to have no effect on the module operation.

RX14MASK is used to mask the filter fields of Message Buffer 14.

This register can only be programmed while the module is in Freeze Mode as it is blocked by hardware in other modes.

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RX14M31_RX14M0																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FLEXCANx_RX14MASK field descriptions

Field	Description
RX14M31_ RX14M0	These bits mask Mailbox 14 filter bits in the same fashion as RXMGMASK masks other Mailboxes filters (see Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK)) 1 The corresponding bit in the filter is checked 0 the corresponding bit in the filter is "don't care"

15.1.8.6 Rx Buffer 15 Mask Register (FLEXCANx_RX15MASK)

RX15MASK is provided for legacy support, asserting the MCR[IRMQ] bit causes the RX15MASK Register to have no effect on the module operation.

RX15MASK is used to mask the filter fields of Message Buffer 15.

This register can only be programmed while the module is in Freeze Mode as it is blocked by hardware in other modes.

Address: Base address + 18h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RX15M31_RX15M0																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FLEXCANx_RX15MASK field descriptions

Field	Description
RX15M31_ RX15M0	These bits mask Mailbox 15 filter bits in the same fashion as RXMGMASK masks other Mailboxes filters (see Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK)). 1 The corresponding bit in the filter is checked 0 the corresponding bit in the filter is "don't care"

15.1.8.7 Error Counter Register (FLEXCANx_ECR)

This register has 2 8-bit fields reflecting the value of two FLEXCAN error counters: Transmit Error Counter (Tx_Err_Counter field) and Receive Error Counter (Rx_Err_Counter field). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FLEXCAN module. Both counters are read only except in Freeze Mode, where they can be written by the ARM.

FLEXCAN responds to any bus state as described in the protocol, e.g. transmit 'Error Active' or 'Error Passive' flag, delay its transmission start time ('Error Passive') and avoid any influence on the bus when in 'Bus Off' state. The following are the basic rules for FLEXCAN bus state transitions.

- If the value of Tx_Err_Counter or Rx_Err_Counter increases to be greater than or equal to 128, the FLT_CONF field in the Error and Status Register is updated to reflect 'Error Passive' state.
- If the FLEXCAN state is 'Error Passive', and either Tx_Err_Counter or Rx_Err_Counter decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLT_CONF field in the Error and Status Register is updated to reflect 'Error Active' state.
- If the value of Tx_Err_Counter increases to be greater than 255, the FLT_CONF field in the Error and Status Register is updated to reflect 'Bus Off' state, and an interrupt may be issued. The value of Tx_Err_Counter is then reset to zero.
- If FLEXCAN is in 'Bus Off' state, then Tx_Err_Counter is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, Tx_Err_Counter is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the Tx_Err_Counter. When Tx_Err_Counter reaches the value of 128, the FLT_CONF field in the Error and Status Register is updated to be 'Error Active' and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the Tx_Err_Counter value.
- If during system start-up, only one node is operating, then its Tx_Err_Counter increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACK_ERR bit in the Error and Status Register). After the transition to 'Error Passive' state, the Tx_Err_Counter does not increment anymore by acknowledge errors. Therefore the device never goes to the 'Bus Off' state.
- If the Rx_Err_Counter increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next

successful message reception, the counter is set to a value between 119 and 127 to resume to 'Error Active' state.

Address: Base address + 1Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																Rx_Err_Counter								Tx_Err_Counter							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_ECR field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 Rx_Err_Counter	Rx_Err_Counter
Tx_Err_Counter	Tx_Err_Counter

15.1.8.8 Error and Status 1 Register (FLEXCANx_ESR1)

This register reflects various error conditions, some general status of the device and it is the source of four interrupts to the ARM.

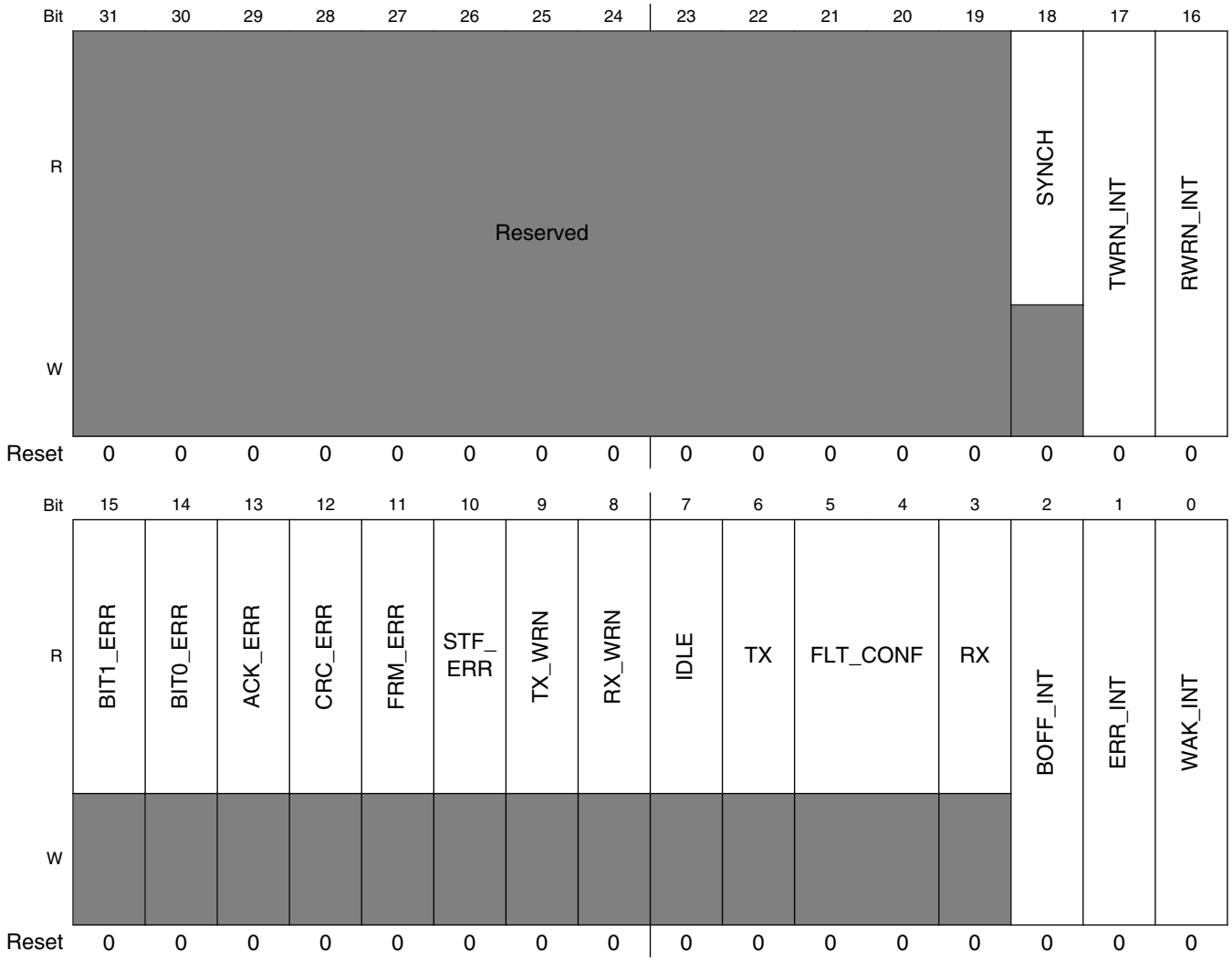
The ARM read action clears bits 15-10, therefore the reported *error conditions*(bits 15-10) are those that occurred since the last time the ARM read this register. Bits 9-3 are status bits.

Some bits in this register are read-only and some are not.

Table 15-20. FlexCAN State

SYNCH	IDLE	TX	RX	FlexCAN state
0	0	0	0	Not synchronized to CAN bus
1	1	x	x	Idle
1	0	1	0	Transmitting
1	0	0	1	Receiving
other combinations				Reserved

Address: Base address + 20h offset



FLEXCANx_ESR1 field descriptions

Field	Description
31–19 -	This field is reserved. Reserved
18 SYNCH	This read-only flag indicates whether the FlexCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the FlexCAN. Refer to Table 15-20 1 FlexCAN is synchronized to the CAN bus 0 FlexCAN is not synchronized to the CAN bus
17 TWRN_INT	If the WRN_EN bit in MCR is asserted, the TWRN_INT bit is set when the TX_WRN flag transition from '0' to '1', meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control Register (TWRN_MSK) is set, an interrupt is generated to the ARM. This bit is cleared by writing it to '1'. When WRN_EN is negated, this flag is masked. ARM must clear this flag before disabling the bit. Otherwise it will be set when the WRN_EN is set again. Writing '0' has no effect. This flag is not generated during "Bus Off" state. This bit is not updated during Freeze mode. 1 The Tx error counter transition from < 96 to >= 96 0 No such occurrence

Table continues on the next page...

FLEXCANx_ESR1 field descriptions (continued)

Field	Description
16 RWRN_INT	If the WRN_EN bit in MCR is asserted, the RWRN_INT bit is set when the RX_WRN flag transition from '0' to '1', meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control Register (RWRN_MSK) is set, an interrupt is generated to the ARM. This bit is cleared by writing it to '1'. When WRN_EN is negated, this flag is masked. ARM must clear this flag before disabling the bit. Otherwise it will be set when the WRN_EN is set again. Writing '0' has no effect. This bit is not updated during Freeze mode. 1 The Rx error counter transition from < 96 to >= 96 0 No such occurrence
15 BIT1_ERR	This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits. 1 At least one bit sent as recessive is received as dominant 0 No such occurrence
14 BIT0_ERR	This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 1 At least one bit sent as dominant is received as recessive 0 No such occurrence
13 ACK_ERR	This bit indicates that an Acknowledge Error has been detected by the transmitter node, i.e., a dominant bit has not been detected during the ACK SLOT. 1 An ACK error occurred since last read of this register 0 No such occurrence
12 CRC_ERR	This bit indicates that a CRC Error has been detected by the receiver node, i.e., the calculated CRC is different from the received. 1 A CRC error occurred since last read of this register. 0 No such occurrence
11 FRM_ERR	This bit indicates that a Form Error has been detected by the receiver node, i.e., a fixed-form bit field contains at least one illegal bit. 1 A Form Error occurred since last read of this register 0 No such occurrence
10 STF_ERR	This bit indicates that a Stuffing Error has been detected. 1 A Stuffing Error occurred since last read of this register. 0 No such occurrence.
9 TX_WRN	This bit indicates when repetitive errors are occurring during message transmission. 1 TX_Err_Counter ≥ 96 0 No such occurrence
8 RX_WRN	This bit indicates when repetitive errors are occurring during message reception. 1 Rx_Err_Counter ≥ 96 0 No such occurrence
7 IDLE	This bit indicates when CAN bus is in IDLE state.Refer to Table 15-20.

Table continues on the next page...

FLEXCANx_ESR1 field descriptions (continued)

Field	Description
	1 CAN bus is now IDLE 0 No such occurrence
6 TX	This bit indicates if FLEXCAN is transmitting a message.Refer to Table 15-20. 1 FLEXCAN is transmitting a message 0 FLEXCAN is receiving a message
5–4 FLT_CONF	If the LOM bit in the Control Register is asserted, after some delay that depends on the CAN bit timing the FLT_CONF field will indicate "Error Passive". The very same delay affects the way how FLT_CONF reflects an update to ECR register by the ARM. It may be necessary up to one CAN bit time to get them coherent again. Since the Control Register is not affected by soft reset, the FLT_CONF field will not be affected by soft reset if the LOM bit is asserted. This 2-bit field indicates the Confinement State of the FLEXCAN module, as shown in below: 00 Error Active 01 Error Passive 1x Bus off
3 RX	This bit indicates if FlexCAN is receiving a message. Refer to Table 15-20. 1 FLEXCAN is transmitting a message 0 FLEXCAN is receiving a message
2 BOFF_INT	This bit is set when FLEXCAN enters 'Bus Off' state. If the corresponding mask bit in the Control Register (BOFF_MSK) is set, an interrupt is generated to the ARM. This bit is cleared by writing it to '1'. Writing '0' has no effect. 1 FLEXCAN module entered 'Bus Off' state 0 No such occurrence
1 ERR_INT	This bit indicates that at least one of the Error Bits (bits 15-10) is set. If the corresponding mask bit in the Control Register (ERR_MSK) is set, an interrupt is generated to the ARM. This bit is cleared by writing it to '1'.Writing '0' has no effect. 1 Indicates setting of any Error Bit in the Error and Status Register 0 No such occurrence
0 WAK_INT	When FLEXCAN is Stop Mode and a recessive to dominant transition is detected on the CAN bus and if the WAK_MSK bit in the MCR Register is set, an interrupt is generated to the ARM. This bit is cleared by writing it to '1'. When SLF_WAK is negated, this flag is masked. ARM must clear this flag before disabling the bit. Otherwise it will be set when the SLF_WAK is set again. Writing '0' has no effect 1 Indicates a recessive to dominant transition received on the CAN bus when the FLEXCAN module is in Stop Mode 0 No such occurrence

15.1.8.9 Interrupt Masks 2 Register (FLEXCANx_IMASK2)

This register allows any number of a range of 32 Message Buffer Interrupts to be enabled or disabled. It contains one interrupt mask bit per buffer, enabling the ARM to determine which buffer generates an interrupt after a successful transmission or reception (i.e. when the corresponding IFLAG2 bit is set).

Address: Base address + 24h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF63M_BUF32M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_IMASK2 field descriptions

Field	Description
BUF63M_ BUF32M	Each bit enables or disables the respective FLEXCAN Message Buffer (MB32 to MB63) Interrupt. Setting or clearing a bit in the IMASK2 Register can assert or negate an interrupt request, if the corresponding IFLAG2 bit is set. 1 The corresponding buffer Interrupt is enabled 0 The corresponding buffer Interrupt is disabled

15.1.8.10 Interrupt Masks 1 Register (FLEXCANx_IMASK1)

This register allows to enable or disable any number of a range of 32 Message Buffer Interrupts. It contains one interrupt mask bit per buffer, enabling the ARM to determine which buffer generates an interrupt after a successful transmission or reception (i.e., when the corresponding IFLAG1 bit is set).

Address: Base address + 28h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF31M_BUF0M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_IMASK1 field descriptions

Field	Description
BUF31M_BUF0M	Each bit enables or disables the respective FLEXCAN Message Buffer (MB0 to MB31) Interrupt. Setting or clearing a bit in the IMASK1 Register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set

FLEXCANx_IMASK1 field descriptions (continued)

Field	Description
	1 The corresponding buffer Interrupt is enabled 0 The corresponding buffer Interrupt is disabled

15.1.8.11 Interrupt Flags 2 Register (FLEXCANx_IFLAG2)

This register defines the flags for 32 Message Buffer interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG2 bit. If the corresponding IMASK2 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing it to '1'. Writing '0' has no effect.Before updating MCR[MAXMB] field, ARM must treat the IFLAG2 bits which MB value is greater than the MCR[MAXMB] to be updated, otherwise they will keep set and be inconsistent with the amount of MBs available.

Address: Base address + 2Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF63I_BUF32I																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_IFLAG2 field descriptions

Field	Description
BUF63I_BUF32I	Each bit flags the respective FLEXCAN Message Buffer (MB32 to MB63) interrupt. 1 The corresponding buffer has successfully completed transmission or reception 0 No such occurrence

15.1.8.12 Interrupt Flags 1 Register (FLEXCANx_IFLAG1)

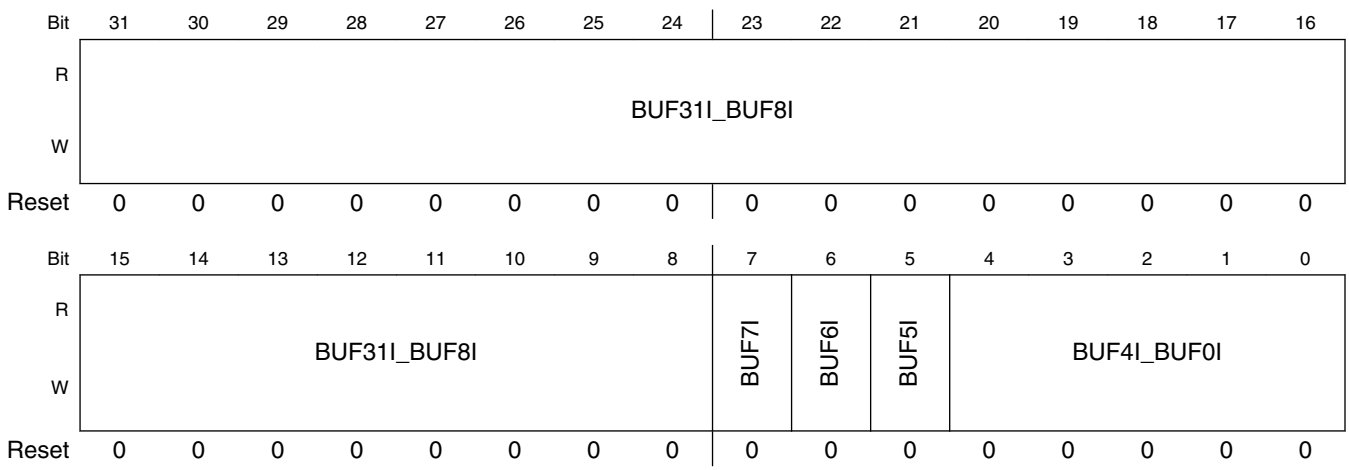
This register defines the flags for 32 Message Buffer interrupts and FIFO interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG1 bit. If the corresponding IMASK1 bit is set, an interrupt will be generated. The Interrupt flag must be cleared by writing it to '1'. Writing '0' has no effect.

When the RFEN bit in the MCR is set (Rx FIFO enabled), the function of the 8 least significant interrupt flags (BUF7I - BUF0I) is changed to support the FIFO operation. BUF7I, BUF6I and BUF5I indicate operating conditions of the FIFO, while BUF4I to BUF0I are not used. Before enabling the RFEN, ARM must service the IFLAGS asserted

in the Rx FIFO region (see Rx FIFO). Otherwise, these IFLAGS will mistakenly show the related MBs now belonging to FIFO as having contents to be serviced. When the RFEN is negated, the FIFO flags must be cleared. The same care must be taken when a RFFN value is selected extending Rx FIFO filters beyond MB7 (see Control 2 Register (FLEXCAN_CTRL2)). For example, when RFFN is 0x8, the MB0-23 range is occupied by Rx FIFO filters and related IFLAGS must be cleared.

Before updating MCR[MAXMB] field, ARM must service the IFLAG1 which MB value is greater than the MCR[MAXMB] to be updated, otherwise they will keep set and be inconsistent with the amount of MBs available.

Address: Base address + 30h offset



FLEXCANx_IFLAG1 field descriptions

Field	Description
31–8 BUF31I_BUF8I	Each bit flags the respective FLEXCAN Message Buffer (MB8 to MB31) interrupt. 1 The corresponding MB has successfully completed transmission or reception 0 No such occurrence
7 BUF7I	If the Rx FIFO is not enabled, this bit flags the interrupt for MB7. If the MCR[RFEN] bit is asserted, this flag indicates that a message was lost because Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by ARM writes. 1 MB7 completed transmission/reception or FIFO overflow 0 No such occurrence
6 BUF6I	If the Rx FIFO is not enabled, this bit flags the interrupt for MB6. If the MCR[RFEN] bit is asserted, this flag indicates when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. Note that if the flag is cleared while the number of unread messages is greater than 4 it will not assert again until the number of unread messages within the Rx FIFO is decreased to equal or less than 4. This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by ARM writes.

Table continues on the next page...

FLEXCANx_IFLAG1 field descriptions (continued)

Field	Description
	1 MB6 completed transmission/reception or FIFO almost full 0 No such occurrence
5 BUF5I	If the Rx FIFO is not enabled, this bit flags the interrupt for MB5. If the Rx FIFO is enabled, this flag indicates that at least one frame is available to be read from the Rx FIFO. This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by ARM writes. 1 MB5 completed transmission/reception or frames available in the FIFO 0 No such occurrence
BUF4I_BUF0I	If the Rx FIFO is not enabled, these bits flag the interrupts for MB0 to MB4. If the Rx FIFO is enabled, these flags are not used and must be considered as reserved locations. These flags are cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by ARM writes. 1 Corresponding MB completed transmission/reception 0 No such occurrence

15.1.8.13 Control 2 Register (FLEXCANx_CTRL2)

This register contains control bits for CAN errors, FIFO features and mode selection.

Table 15-21. Rx FIFO Filters

RFFN[3:0]	Number of Rx FIFO filters	Message Buffers occupied by Rx FIFO and ID Filter Table	Remaining Available Mailboxes[1]	Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks[2]	Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask[2]
0x0	8	MB 0-7	MB 8-63	Elements 0-7	none
0x1	16	MB 0-9	MB 10-63	Elements 0-9	Elements 10-15
0x2	24	MB 0-11	MB 12-63	Elements 0-11	Elements 12-23
0x3	32	MB 0-13	MB 14-63	Elements 0-13	Elements 14-31
0x4	40	MB 0-15	MB 16-63	Elements 0-15	Elements 16-39
0x5	48	MB 0-17	MB 18-63	Elements 0-17	Elements 18-47
0x6	56	MB 0-19	MB 20-63	Elements 0-19	Elements 20-55
0x7	64	MB 0-21	MB 22-63	Elements 0-21	Elements 22-63
0x8	72	MB 0-23	MB 24-63	Elements 0-23	Elements 24-71
0x9	80	MB 0-25	MB 26-63	Elements 0-25	Elements 26-79
0xA	88	MB 0-27	MB 28-63	Elements 0-27	Elements 28-87
0xB	96	MB 0-29	MB 30-63	Elements 0-29	Elements 30-95
0xC	104	MB 0-31	MB 32-63	Elements 0-31	Elements 32-103
0xD	112	MB 0-33	MB 34-63	Elements 0-31	Elements 32-111
0xE	120	MB 0-35	MB 36-63	Elements 0-31	Elements 32-119
0xF	128	MB 0-37	MB 38-63	Elements 0-31	Elements 32-127

1. The number of the last remaining available mailboxes is defined by the MCR[MAXMB] field.
2. If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.

Each group of eight filters occupies a memory space equivalent to two Message Buffers which means that the more filters are implemented the less Mailboxes will be available.

Considering that the Rx FIFO occupies the memory space originally reserved for MB0-5, RFFN should be programmed with a value corresponding to a number of filters not greater than the number of available memory words which can be calculated as follows:

$$(SETUP\_MB - 6) \times 4$$

where SETUP_MB is MAXMB.

The number of remaining Mailboxes available will be:

$$SETUP\_MB - 8 - (RFFN \times 2)$$

If the Number of Rx FIFO Filters programmed through RFFN exceeds the SETUP_MB value, the exceeding ones will not be functional.Unshaded regions in Table 15-22 indicate the valid combinations of MAXMB, RFEN and RFFN, shaded regions are not functional.

Table 15-22. Valid Combinations of MAXMB, RFEN and RFFN

RFFN	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RFEN	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
MAXMB																	
0 - 6																	
7 - 8																	
9 - 10																	
11 - 12																	
13 -14																	
15 - 16																	
17 -18																	
19 - 20																	
21 - 22																	

Table continues on the next page...

Table 15-22. Valid Combinations of MAXMB, RFEN and RFFN (continued)

RFFN	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RFEN	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
MAXMB																	
23 - 24																	
25 - 26																	
27 - 28																	
29 -30																	
31 - 32																	
33 - 34																	
35 - 36																	
37 - 63																	

Address: Base address + 34h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		Reserved		WRMFRZ	RFFN				TASD					MRP	RRS	EACEN
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_CTRL2 field descriptions

Field	Description
31 -	must be written as 0
30–29 -	This field is reserved. Reserved

Table continues on the next page...

FLEXCANx_CTRL2 field descriptions (continued)

Field	Description
28 WRMFRZ	Enable unrestricted write access to FlexCAN memory in Freeze mode. This bit can only be written in Freeze mode and has no effect out of Freeze mode. 1 Enable unrestricted write access to FlexCAN memory 0 Keep the write access restricted in some regions of FlexCAN memory
27–24 RFFN	This 4-bit field defines the number of Rx FIFO filters according to Table 15-21. The maximum selectable number of filters is determined by the ARM. This field can only be written in Freeze mode as it is blocked by hardware in other modes. RFFN defines a number of Message Buffers occupied by Rx FIFO and ID Filter (see Table 15-21) that **may not exceed** the number of available Mailboxes present in module, defined by MCR[MAXMB].Default RFFN value is 0x0, which leads to a total of 8 Rx FIFO filters, occupies the first 8 Message Buffers (MB 0-7) and makes available the next Message Buffers (MB 8-63) for Mailboxes. As a second example, when RFFN is set to 0xD, there will be 112 Rx FIFO filters, located in MB 0-33, and MB 34-63 are available for Mailboxes. Notice that, in this case, individual masks (RXIMR) will just cover Rx FIFO filters in 0-31 range, and filters 32-111 will use RXFGMASK. In case of reducing the number of last Message Buffers, MCR[MAXMB] (see Module Configuration Register (FLEXCAN_MCR)) can be adjusted by the application to minimum of 33, in order to give room to the Rx FIFO and its ID Filter Table defined by RFFN. On the contrary, if the application sets MCR[MAXMB] to 16, for instance, the maximum RFFN is limited to 0x4. RFFN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table 15-18 (see Arbitration and Matching Timing).
23–19 TASD	This 5-bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on CAN bus. This field can only be written in Freeze mode as it is blocked by hardware in other modes. This field is useful to optimize the transmit performance based on factors such as: peripheral/serial clock ratio, CAN bit timing and number of MBs. The duration of an arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and CAN baud rate and inversely proportional to the peripheral clock frequency. The optimal arbitration timing is that in which the last MB is scanned right before the first bit of the Intermission field of a CAN frame. Therefore, if there are few MBs and the system/serial clock ratio is high and the CAN baud rate is low then the arbitration can be delayed and vice-versa. If TASD is 0 then the arbitration start is not delayed, thus ARM has less time to configure a Tx MB for the next arbitration, but more time is reserved for arbitration. In the other hand, if TASD is 24 then ARM can configure a Tx MB later and less time is reserved for arbitration. If too little time is reserved for arbitration the FlexCAN may be not able to find winner MBs in time to compete with other nodes for the CAN bus. If the arbitration ends too much time before the first bit of Intermission field then there is a chance that ARM reconfigure some Tx MBs and the winner MB is not the best to be transmitted. The reset value is different on various platforms, according to their peripheral clock frequency, number of MBs and target CAN baud rate. The optimal configuration for TASD can be calculated as:

Table continues on the next page...

FLEXCANx_CTRL2 field descriptions (continued)

Field	Description
	$$TASD = 25 - \left\{ \frac{f_{CANCLK} \times [MAXMB + 3 - (RFEN \times 8) - (RFEN \times RFFN \times 2)] \times 2}{f_{sys} \times [1 + (PSEG1 + 1) + (PSEG2 + 1) + (PROPSEG + 1)] \times (PRESDIV + 1)} \right\}$$ where: f_{CANCLK} is the Protocol Engine (PE) Clock in Hz; PE clock is derrived from CAN_CLK_ROOT in CCM. See Clock Root Generator f_{SYS} is the peripheral clock in Hz; MAXMB is the value in CTRL1[MAXMB] field; RFEN is the value in CTRL1[RFEN] bit; RFFN is the value in CTRL2[RFFN] field; PSEG1 is the value in CTRL1[PSEG1] field; PSEG2 is the value in CTRL1[PSEG2] field; PROPSEG is the value in CTRL1[PROPSEG] field; PRESDIV is the value in CTRL1[PRESDIV] field. Please refer to Arbitration process and Protocol Timing for more details.
18 MRP	If this bit is set the matching process starts from the Mailboxes and if no match occurs the matching continues on the Rx FIFO. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Matching starts from Mailboxes and continues on Rx FIFO 0 Matching starts from Rx FIFO and continues on Mailboxes
17 RRS	If this bit is asserted Remote Request Frame is submitted to a matching process and stored in the corresponding Message Buffer in the same fashion of a Data Frame. No automatic Remote Response Frame will be generated. If this bit is negated the Remote Request Frame is submitted to a matching process and an automatic Remote Response Frame is generated if a Message Buffer with CODE=0b1010 is found with the same ID. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Remote Request Frame is stored 0 Remote Response Frame is generated
16 EACEN	This bit controls the comparison of IDE and RTR bits within Rx Mailboxes filters with their corresponding bits in the incoming frame by the matching process. This bit does not affect matching for Rx FIFO. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Enables the comparison of both Rx Mailbox filter's IDE and RTR bit with their corresponding bits within the incoming frame. Mask bits do apply. 0 Rx Mailbox filter's IDE bit is always compared and RTR is never compared despite mask bits.
-	This field is reserved.

Table continues on the next page...

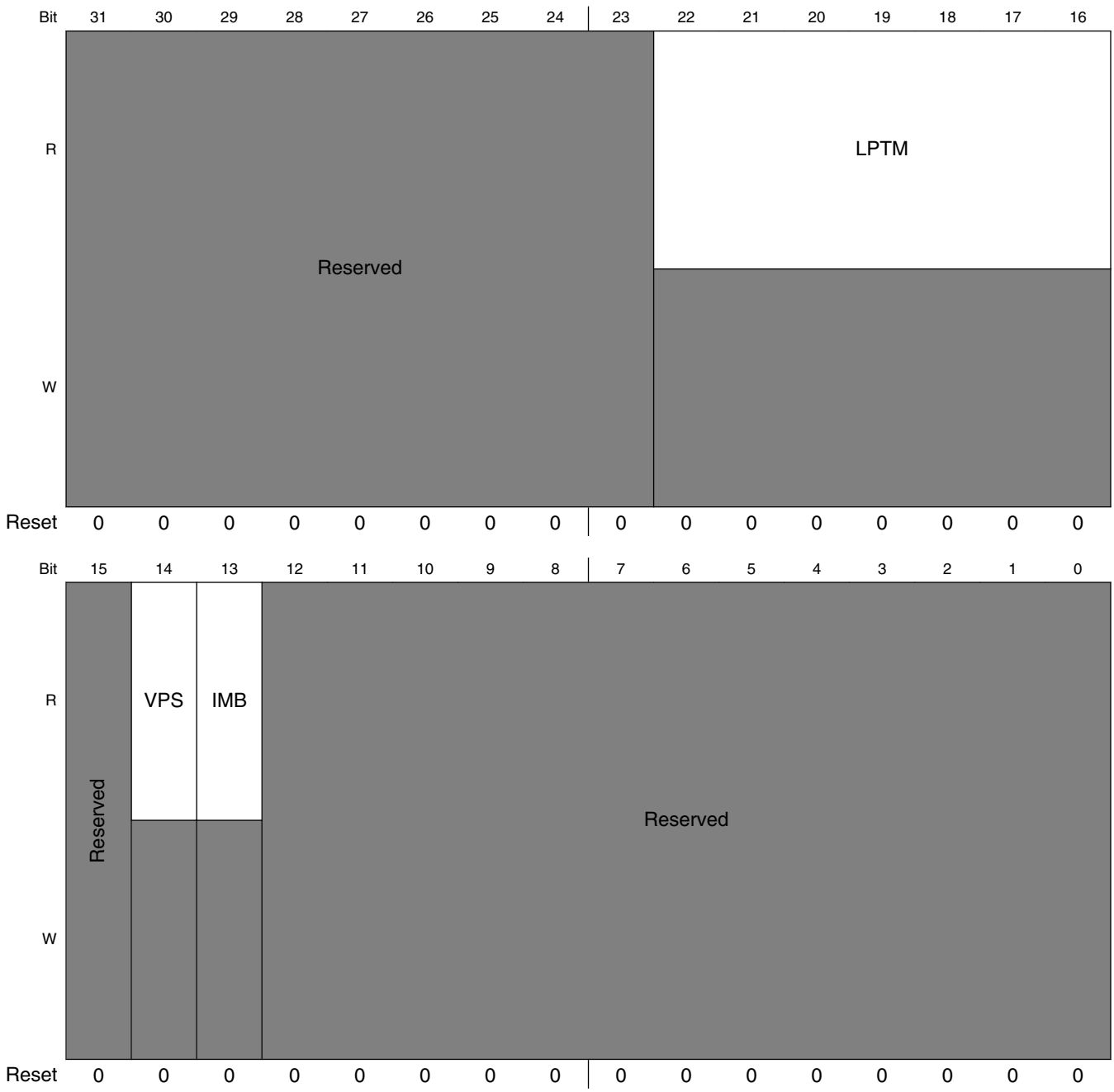
FLEXCANx_CTRL2 field descriptions (continued)

Field	Description
	Reserved

15.1.8.14 Error and Status 2 Register (FLEXCANx_ESR2)

This register reflects various interrupt flags and some general status.

Address: Base address + 38h offset



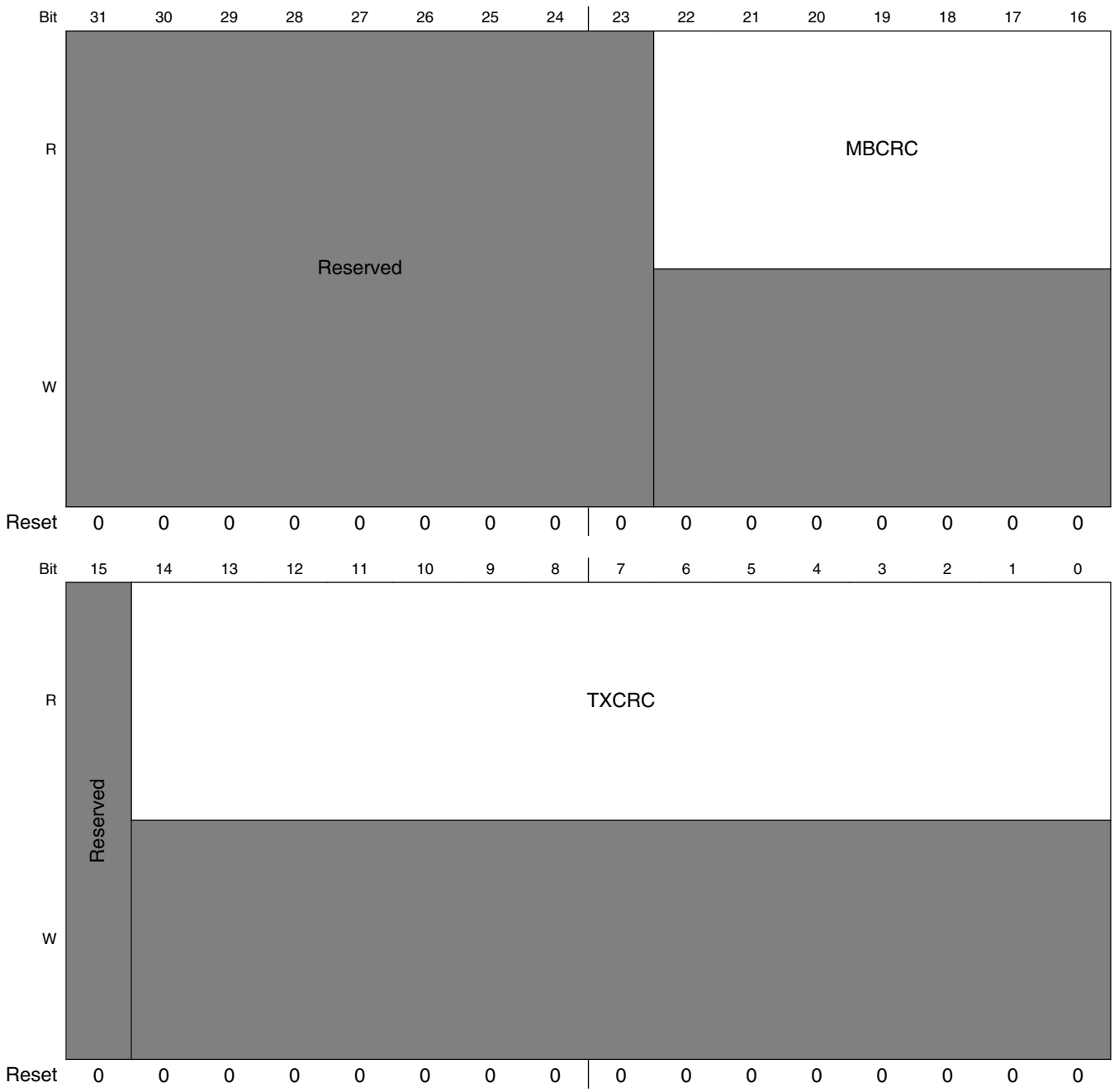
FLEXCANx_ESR2 field descriptions

Field	Description
31–23 -	This field is reserved. Reserved
22–16 LPTM	If ESR2[VPS] is asserted, his 7-bit field indicates the lowest number inactive Mailbox (refer to IMB bit description). If there is no inactive Mailbox then the Mailbox indicated depends on CTRL1[LBUF] bit value. If CTRL1[LBUF] bit is negated then the Mailbox indicated is the one which has the greatest arbitration value (see Highest Mailbox priority first). If CTRL1[LBUF] bit is asserted then the Mailbox indicated is the highest number active Tx Mailbox. If a Tx Mailbox is being transmitted it is not considered in LPTM calculation. If ESR2[IMB] is not asserted and a frame is transmitted successfully, LPTM is updated with its Mailbox number.
15 -	This field is reserved. Reserved
14 VPS	This bit indicates whether IMB and LPTM contents are currently valid or not. VPS is asserted upon every complete Tx arbitration process unless the ARM writes to Control and Status word of a Mailbox that has already been scanned (i.e. it is behind Tx Arbitration Pointer) during the Tx arbitration process. If there is no inactive Mailbox and only one Tx Mailbox which is being transmitted then VPS is not asserted. VPS is negated upon the start of every Tx arbitration process or upon a write to Control and Status word of any Mailbox.ESR2[VPS] is not affected by any ARM write into Control Status (C/S) of a MB which is blocked by abort mechanism. When MCR[AEN] is asserted, the abort code write in C/S of a MB that is been transmitted (pending abort), or any write attempt into a Tx MB with IFLAG set is blocked. 1 Contents of IMB and LPTM are valid 0 Contents of IMB and LPTM are invalid
13 IMB	If ESR2[VPS] is asserted, this bit indicates whether there is any inactive Mailbox (CODE field is either 0b1000 or 0b0000). This bit is asserted in the following cases: (1) During arbitration, if a LPTM is found and it is inactive. (2) If IMB is not asserted and a frame is transmitted successfully. (3) This bit is cleared in all start of arbitration (see Arbitration process). LPTM mechanism have the following behavior: if a MB is successfully transmitted and ESR2[IMB]=0 (no inactive Mailbox), then ESR2[VPS] and ESR2[IMB] are asserted and the index related to the MB just transmitted is loaded into ESR2[LPTM]. 1 If ESR2[VPS] is asserted, there is at least one inactive Mailbox. LPTM content is the number of the first one. 0 If ESR2[VPS] is asserted, the ESR2[LPTM] is not an inactive Mailbox.
-	This field is reserved. Reserved

15.1.8.15 CRC Register (FLEXCANx_CRCR)

This register provides information about the CRC of transmitted messages

Address: Base address + 44h offset



FLEXCANx_CRCR field descriptions

Field	Description
31–23 -	This field is reserved. Reserved
22–16 MBCRC	This field indicates the number of the Mailbox corresponding to the value in TXCRC field.
15 -	This field is reserved. Reserved
TXCRC	This field indicates the CRC value of the last message transmitted. This field is updated at the same time the Tx Interrupt Flag is asserted.

15.1.8.16 Rx FIFO Global Mask Register (FLEXCANx_RXFGMASK)

If Rx FIFO is enabled RXFGMASK is used to mask the Rx FIFO ID Filter Table elements that do not have a corresponding RXIMR according to CTRL2[RFFN] field setting.

This register can only be written in Freeze Mode as it is blocked by hardware in other modes.

Table 15-23. Rx FIFO Global Mask usage

Rx FIFO ID Filter Table Elements Format (MCR[IDAM])	Identifier Acceptance Filter fields					
	RTR	IDE	RXIDA	RXIDB	RXIDC	reserved
A	FGM[31]	FGM[30]	FGM[29:1]	-	-	FGM[0]
B	FGM[31] FGM[15]	FGM[30] FGM[14]	-	FGM[29:16] FGM[13:0] 1		-
C	-	-		-	FGM[31:24] FGM[23:16] FGM[15:8] FGM[7:0] 2	

1. If MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

Address: Base address + 48h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FGM31_FGM0																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FLEXCANx_RXFGMASK field descriptions

Field	Description
FGM31_FGM0	These bits mask the ID Filter Table elements bits in a perfect alignment.Rx FIFO Global Mask Register (FLEXCAN_RXFGMASK) shows in detail which FGM bits mask each IDAF field. Clear this register has the effect of disabling the ID Filter. 1 The corresponding bit in the filter is checked 0 The corresponding bit in the filter is "don't care"

15.1.8.17 Rx FIFO Information Register (FLEXCANx_RXFIR)

RXFIR provides information on Rx FIFO.

This register is the port through which ARM accesses the output of the RXFIR FIFO located in RAM. The RXFIR FIFO is written by the FlexCAN whenever a new message is moved into the Rx FIFO as well as its output is updated whenever the output of the Rx FIFO is updated with the next message. Refer to Rx FIFO to find instructions on reading this register.

Address: Base address + 4Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																							IDHIT								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_RXFIR field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
IDHIT	This 9-bit field indicates which Identifier Acceptance Filter (see Rx FIFO Structure) was hit by the received message that is in the output of the Rx FIFO. (refer to Rx FIFO for details) If multiple filters match the incoming message ID then the first matching IDAF found (lowest number) by the matching process is indicated. This field is valid only while the IFLAG[BUF5I] is asserted.

15.1.8.18 Rx Individual Mask Registers (FLEXCANx_RXIMR0_RXIMR63)

RXIMR are used as acceptance masks for ID filtering in Rx MBs and the Rx FIFO. If the Rx FIFO is not enabled, one mask register is provided for each available Mailbox, providing ID masking capability on a per Mailbox basis.

When the Rx FIFO is enabled (MCR[RFEN] bit is asserted), up to 32 Rx Individual Mask Registers can apply to the Rx FIFO ID Filter Table elements on a one-to-one correspondence depending on CTRL2[RFFN] setting. Refer to Control 2 Register (FLEXCAN_CTRL2) for details.

RXIMR can only be written by the ARM while the module is in Freeze Mode, otherwise they are blocked by hardware.

The Individual Rx Mask Registers are not affected by reset and must be explicitly initialized prior to any reception.

Address: Base address + 880h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MI31_MI0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_RXIMR0_RXIMR63 field descriptions

Field	Description
MI31_MI0	These bits mask both Mailbox filter and Rx FIFO ID Filter Table element in distinct ways. For Mailbox filter refer to Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK). For Rx FIFO ID Filter Table element refer to Rx FIFO Global Mask Register (FLEXCAN_RXFGMASK). 1 The corresponding bit in the filter is checked 0 the corresponding bit in the filter is "don't care"

15.1.8.19 Glitch Filter Width Registers (FLEXCANx_GFWR)

The Glitch Filter just takes effects when FLEXCAN enters the STOP mode.

Address: Base address + 9E0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																								GFWR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

FLEXCANx_GFWR field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value 0.
GFWR	It determines the Glitch Filter Width. The width will be divided from Oscillator clock by GFWR values. By default, it is 5.33 µs when the oscillator is 24 MHz. Filter Pulse Width = [(GFWR FIELD + 1) x (1 / Osc. Frequency)]

15.2 I2C Controller (I2C)

15.2.1 Overview

This chapter describes block-level operation and programming of I2C. The chapter is intended for a block-driver software developer. To understand how the block is integrated at the SoC level, a system software developer should see discussions of the block in the appropriate SoC-level chapter(s).

References: This document assumes an understanding of the following document:

- *The I2C Bus Specification*, Version 2.1, by Philips Semiconductor

The Inter IC (I2C) provides functionality of a standard I2C slave and master. The I2C is designed to be compatible with the standard NXP I2C bus protocol.

NOTE

independent I2C channels are available.

I2C is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange, minimizing the interconnection between devices. This bus is suitable for applications requiring occasional communications over a short distance between many devices. The flexible I2C standard allows additional devices to be connected to the bus for expansion and system development. See the connection diagram in the figure below.

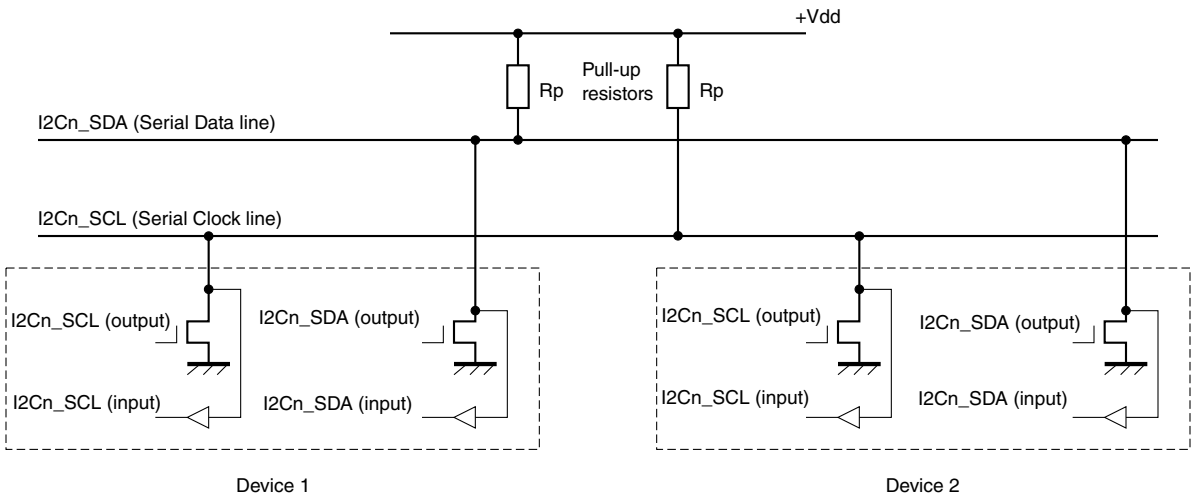


Figure 15-6. Connection of devices to I2C bus

The I2C interface speed is dependent on the I2C bus loading and timing characteristics. For pin requirement details, see *The I2C Bus Specification*. The I2C system is a true multimaster bus including arbitration and collision detection that prevents data corruption if multiple devices attempt to control the bus simultaneously. This feature supports complex applications with multiprocessor control and can be used for rapid testing and alignment of end products through external connections to an assembly-line computer. The figure below shows the block diagram of I2C.

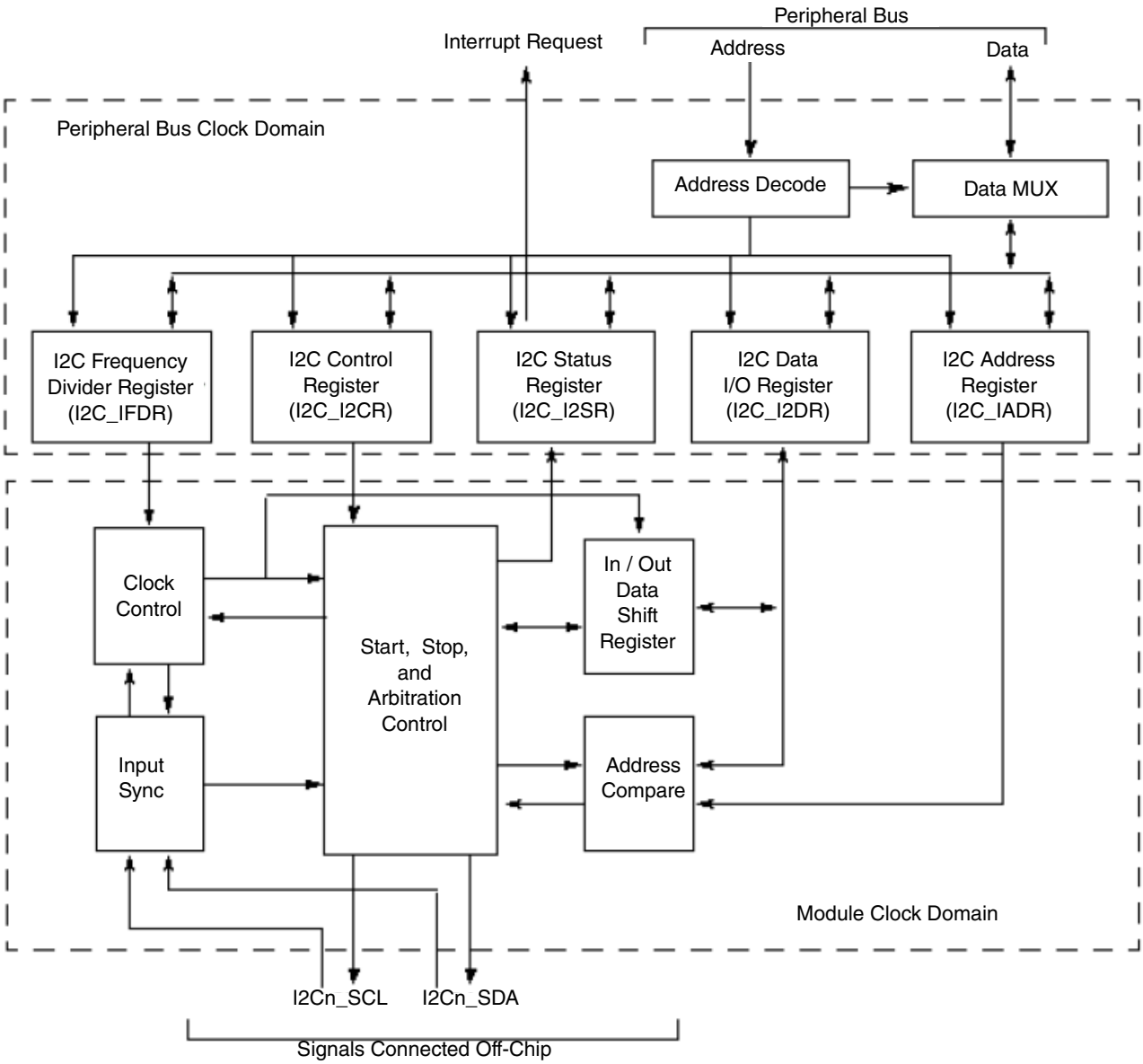


Figure 15-7. I2C block diagram

15.2.1.1 Features

The I2C has the following key features:

- Compatibility with I2C bus standard
- Multimaster operation
- Software programmability for one of 64 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven, byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave

- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated Start signal generation
- Acknowledge bit generation/detection
- Bus-busy detection

15.2.1.2 Modes and operations

The I2C operates primarily in two functional modes: Standard mode and Fast mode.

- In Standard mode, I2C supports the data transfer rates up to 100 kbits/s.
- In Fast mode, data transfer rates up to 400 kbits/s can be achieved. Per block operation, there is no special configuration required for Fast or Standard mode. It is the data transfer rate that distinguishes Standard and Fast mode.

15.2.2 External Signals

This section discusses I2C signals that connect off-chip.

For I2C compliance, all devices connected to the I2Cn_SCL and I2Cn_SDA signals must have open-drain or open-collector outputs. The logic AND function is implemented on both lines with external pull-up resistors.

Inputs of I2Cn_SCL and I2Cn_SDA also need to be manually enabled by setting the SION bit in the IOMUX after the corresponding PADs are selected as I2C function.

The table below describes all I2C signals that connect off-chip.

Table 15-24. I2C External Signals

Signal	Description	Pad	Mode	Direction
I2C1_SCL	Serial Clock	GPIO1_IO04	ALT4	IO
		I2C1_SCL	ALT0	
		UART1_RXD	ALT1	
I2C1_SDA	Serial Data	GPIO1_IO05	ALT4	IO
		I2C1_SDA	ALT0	
		UART1_TXD	ALT1	
I2C2_SCL	Serial Clock	GPIO1_IO06	ALT4	IO
		I2C2_SCL	ALT0	
		UART2_RXD	ALT1	
I2C2_SDA	Serial Data	GPIO1_IO07	ALT4	IO

Table continues on the next page...

Table 15-24. I2C External Signals (continued)

Signal	Description	Pad	Mode	Direction
		I2C2_SDA	ALT0	
		UART2_TXD	ALT1	
I2C3_SCL	Serial Clock	ENET1_RDATA0	ALT2	IO
		GPIO1_IO08	ALT4	
		I2C3_SCL	ALT0	
		LCD1_DATA20	ALT6	
		SD3_DATA3	ALT2	
I2C3_SDA	Serial Data	ENET1_RDATA1	ALT2	IO
		GPIO1_IO09	ALT4	
		I2C3_SDA	ALT0	
		LCD1_DATA21	ALT6	
		SD3_DATA2	ALT2	
I2C4_SCL	Serial Clock	ENET1_TDATA2	ALT3	IO
		GPIO1_IO10	ALT4	
		I2C4_SCL	ALT0	
		LCD1_DATA22	ALT6	
		SAI1_RXFS	ALT3	
I2C4_SDA	Serial Data	ENET1_TDATA3	ALT3	IO
		GPIO1_IO11	ALT4	
		I2C4_SDA	ALT0	
		LCD1_DATA23	ALT6	
		SAI1_RXC	ALT3	

15.2.3 Clocks

There are two input clocks for I2C.

The following table describes the clock sources for I2C. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 15-25. I2C Clocks

Clock name	Clock Root	Description
ipg_clk_patref	I2C_CLK_ROOT	Module clock
ipg_clk_s	I2C_CLK_ROOT	Peripheral access clock

- Peripheral clock: This clock is used for peripheral bus register read/writes.
- Module clock: This is the functional clock of the I2C. The serial bit clock frequency is derived from the module clock. The module clock and peripheral clocks are synchronous with each other. The minimum frequency of the module clock should be 12.8 MHz for Fast mode to achieve 400-kbps operation.

15.2.4 Functional description

This section provides a complete functional description of the block.

15.2.4.1 I2C system configuration

After a reset, the I2C defaults to Slave Receive operations. Thus, when not operating as a master or responding to a slave transmit address, the I2C defaults to the Slave Receive state.

For exceptions, see Initialization sequence.

NOTE

The I2C is designed to be compatible with the Philips™ I2C bus protocol. For information on system configuration, protocol, and restrictions, see the *I2C Bus Specification*, version 2.1, by Philips Semiconductors. The I2C supports Standard and Fast modes only.

15.2.4.2 Arbitration procedure

If multiple devices simultaneously request the bus, the bus clock is determined by a synchronization procedure in which the low period equals the longest clock-low period among the devices, and the high period equals the shortest. A data arbitration procedure determines the relative priority of competing devices.

A device loses arbitration if it sends logic high while another sends logic low; it immediately switches to Slave Receive mode and stops driving I2Cn_SDA. In this case, the transition from master to Slave mode does not generate a Stop condition. Meanwhile, hardware sets the arbitration lost bit in the I2C Status register (I2C_I2SR[IAL] to indicate loss of arbitration).

15.2.4.3 Clock synchronization

Because wire-AND logic is used, a high-to-low transition on SCL affects devices connected to the bus. Devices start counting their low period when the master drives SCL low. When a device clock goes low, it holds SCL low until the Clock High state is reached. However, the low-to-high change in this device clock may not change the state of SCL if another device clock is still in its low period. Therefore, the device with the longest low period holds the synchronized clock SCL low.

Devices with shorter low periods enter a High Wait state during this time (see Figure 15-8). When all devices involved have counted off their low periods, the synchronized clock SCL is released and pulled high. There is then no difference between device clocks and the state of SCL, so all of the devices start counting their high periods. The first device to complete its high period pulls SCL low again.

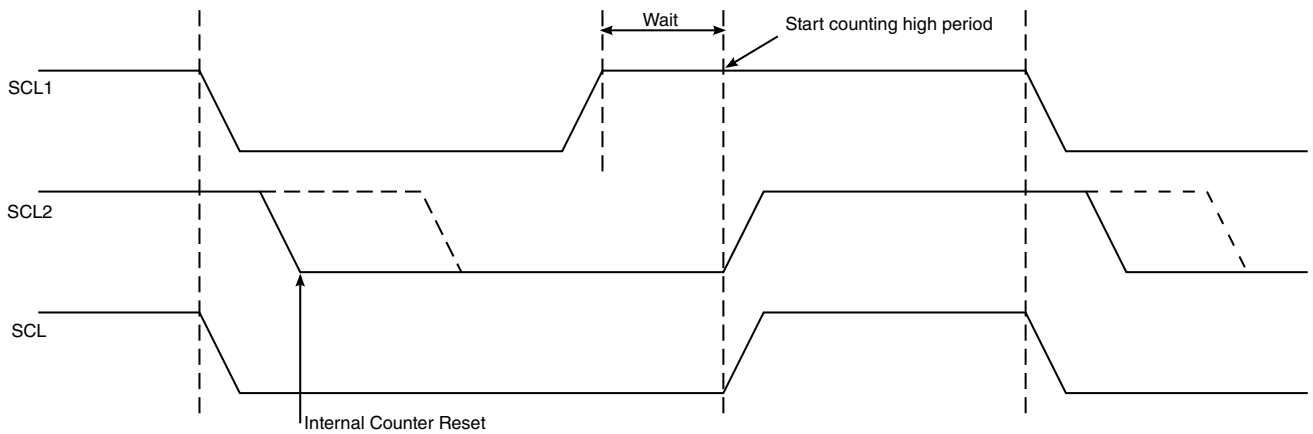


Figure 15-8. Synchronized clock SCL

15.2.4.4 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfers. Slave devices can hold SCL low after completing one byte transfer (9 bits). In such a case, the clock mechanism halts the bus clock and forces the master clock into a Wait state until the slave releases SCL.

15.2.4.5 Clock stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is longer than the master SCL low period, the resulting SCL bus signal low period is stretched.

15.2.4.6 Peripheral bus accesses

I2C is a 16-bit block. Only half-word accesses should be performed to the block.

15.2.4.7 Generation of transfer error on IP bus

If an address is received on the peripheral slave bus interface but it is not implemented, an access error is generated.

15.2.4.8 Reset

The I2C can be reset in the following ways:

- Global reset: A hard asynchronous reset of the whole I2C
- Software reset: An internal reset for the whole I2C (except for I2C_IADR and I2C_IFDR registers) initiated by deasserting the I2C_I2CR[IEN] bit

15.2.4.9 Interrupts

There is only one interrupt from the block, which is enabled by setting the I2C_I2CR[IIEN] bit.

The interrupt is generated in any one of the following conditions:

- One byte transfer is completed (the interrupt is set at the falling edge of the ninth clock).
- An address is received that matches its own specific address in Slave Receive mode.
- Arbitration is lost.

15.2.4.10 Byte order

The block only supports the Little-Endian mode.

15.2.5 Initialization

NOTE

Ensure the input select pins for IOMUXC are configured correctly for I2C.

15.2.5.1 Initialization sequence

Before the interface can transfer serial data, registers must be initialized, as listed here.

1. Set the data sampling rate (I2C_IFDR[IC] to obtain SCL frequency from the system bus clock.
2. Update the address in the (I2C_IADR) to define its slave address (address can range from 0 to 0x7f).
3. Set the I2C enable bit (I2C_I2CR[IEN]) to enable the I2C bus interface system.
4. Modify the bits in the I2C_I2CR to select Master/Slave mode, Transmit/Receive mode, and Interrupt-Enable or not.

15.2.5.2 Generation of Start

After completion of the initialization procedure, serial data can be transmitted by selecting the Master Transmit mode. On a multimaster bus system, the busy bus (I2C_I2SR[IBB]) must be tested to determine whether the serial bus is free. If the bus is free (IBB = 0), the Start signal and the first byte (the slave address) can be sent. The data written to the data register comprises the address of the desired slave and the LSB indicates the transfer direction.

The free time between a Stop and the next Start condition is built into the hardware that generates the Start cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I2C is not busy after writing the calling address to the data register (I2C_I2DR), before proceeding to load data into the data register (I2C_I2DR).

15.2.5.3 Post-transfer software response

Sending or receiving a byte sets the data transferring bit (I2C_I2SR[ICF]), which indicates one byte of communication is finished. Upon completion, the interrupt status (I2C_I2SR[IIF]) is also set. An external interrupt is generated if the interrupt enable (I2C_I2CR[IIEN]) is set. The software must first clear the interrupt status (I2C_I2SR[IIF]) in the interrupt routine.

See the flow chart in Figure 15-10.

The data transferring bit (I2C_I2SR[ICF]) is cleared either by reading from I2C_I2DR in Receive mode or by writing to this register in Transmit mode.

The software can service the I2C I/O in the main program by monitoring the interrupt status (I2C_I2SR[IIF]) if the interrupt enable is deasserted. In this case, the interrupt status should be polled in the data transferring bit (I2C_I2SR[ICF]) because the operation is different when arbitration is lost.

When an interrupt occurs at the end of the address cycle, the master is always in Transmit mode; that is, the address is sent. If Master Receive mode is required, then I2C_I2CR[MTX] should be toggled and a dummy read of the I2C_I2DR register must be executed to trigger receive data.

During Slave-mode address cycles (I2C_I2SR[IAAS] = 1), the slave read/write bit I2C_I2SR[SRW] is read to determine the direction of the next transfer. The transmit/receive bit (I2C_I2CR[MTX]) should also be programmed accordingly. For Slave-mode data cycles (IAAS = 0), SRW is invalid. MTX should be read to determine the current transfer direction.

15.2.5.4 Generation of Stop

A data transfer ends when the master signals a Stop, which can occur after all data is sent.

For a master receiver to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last data byte. This is done by setting the transmit acknowledge bit (I2C_I2CR[TXAK]) before reading the next-to-last byte. Before the last byte is read, a Stop signal must be generated.

15.2.5.5 Generation of Repeated Start

After the data transfer, if the master still requires the bus, it can signal another Start followed by another slave address without signaling a Stop.

15.2.5.6 Slave mode

In the slave interrupt service routine (see Figure 15-10), the block addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the Transmit/Receive mode select bit (I2C_I2CR[MTX]) according to the I2C_I2SR[SRW]. Writing to the I2C_I2CR clears the IAAS automatically. The only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred; interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer can now be initiated by writing information to I2C_I2DR for slave transmits, or read from I2C_I2DR in Slave Receive mode. A dummy read of I2C_I2DR in Slave Receive mode releases SCL, allowing the master to send data.

In the slave transmitter routine, the receive acknowledge bit (I2C_I2SR[RXAK]) must be tested before sending the next byte of data. Setting RXAK means an end-of-data signal from the master receiver, after which the software must switch it from Transmit to Receiver mode. Reading the data register (I2C_I2DR) then releases SCL so the master can generate a Stop signal.

15.2.5.7 Arbitration lost

If several devices try to engage the bus at the same time, one becomes master. Hardware immediately switches devices that lose arbitration to Slave Receive mode. Data output to 12Cn_SDA stops, but 12Cn_SCL is still generated until the end of the byte during which arbitration is lost. An interrupt occurs at the falling edge of the ninth clock of this transfer if the arbitration is lost (I2C_I2SR[IAL] = 1), and the Slave mode is selected (I2C_I2CR[MSTA] = 0).

See the flow chart in Figure 15-10.

If a device that is not a master tries to transmit or do a Start, hardware inhibits the transmission, clears MSTA without signaling a Stop, generates an interrupt to the ARM platform, and sets I2C_I2SR[IAL] to indicate a failed attempt to engage the bus. When considering these cases, the slave service routine should first test I2C_I2SR[IAL], and the software should clear it if it is set.

For Multimaster mode, when an I2C is enabled when the bus is busy and asserts Start, the I2C_I2SR[IAL] bit gets set only for 12Cn_SDA=0, 12Cn_SCL=0/1, 12Cn_SDA=1, and 12Cn_SCL=0; but not for 12Cn_SDA=1 and I2Cn_SCL=1, which is the equivalent of Bus Idle state.

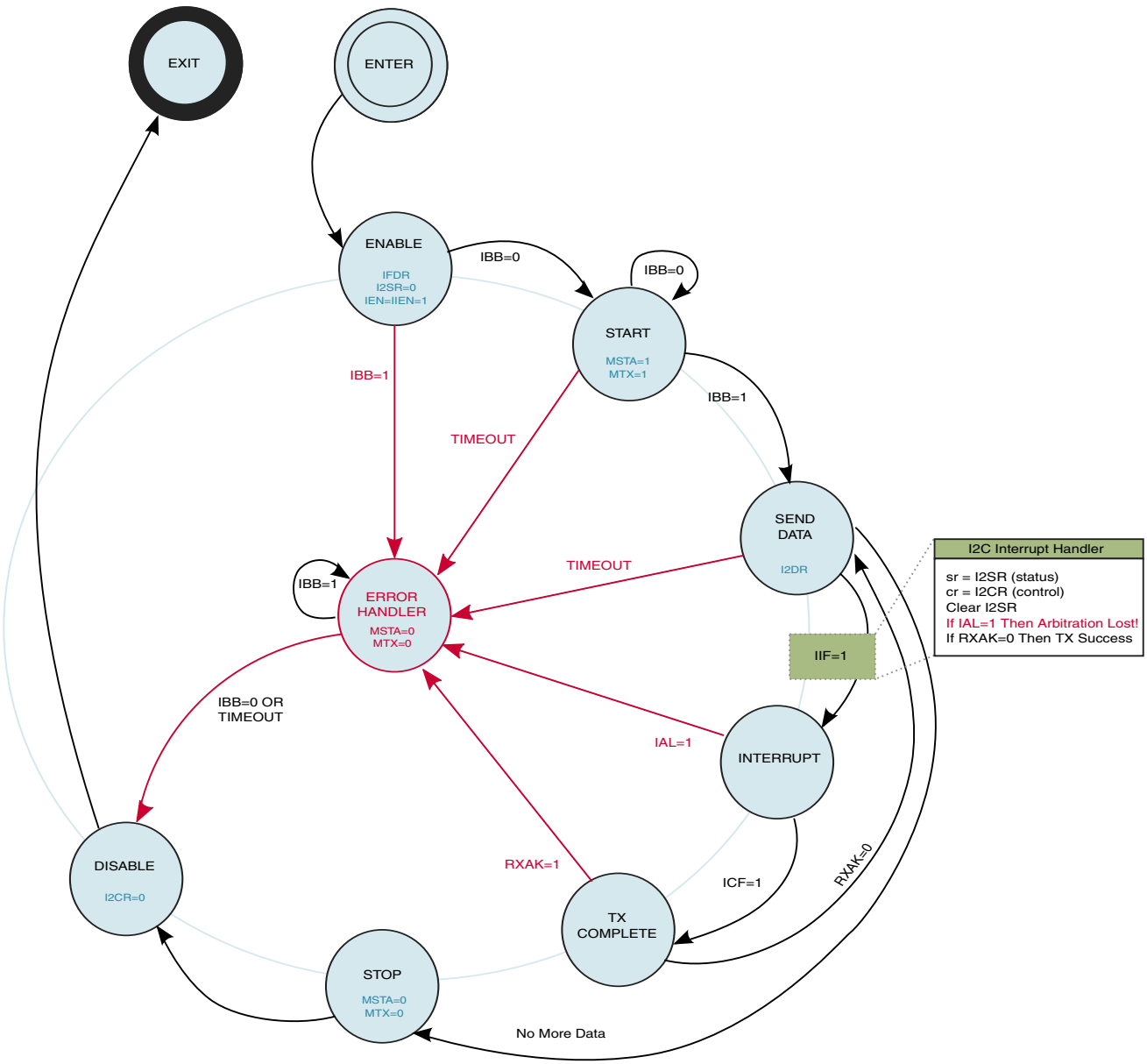


Figure 15-9. I2C Programming state diagram

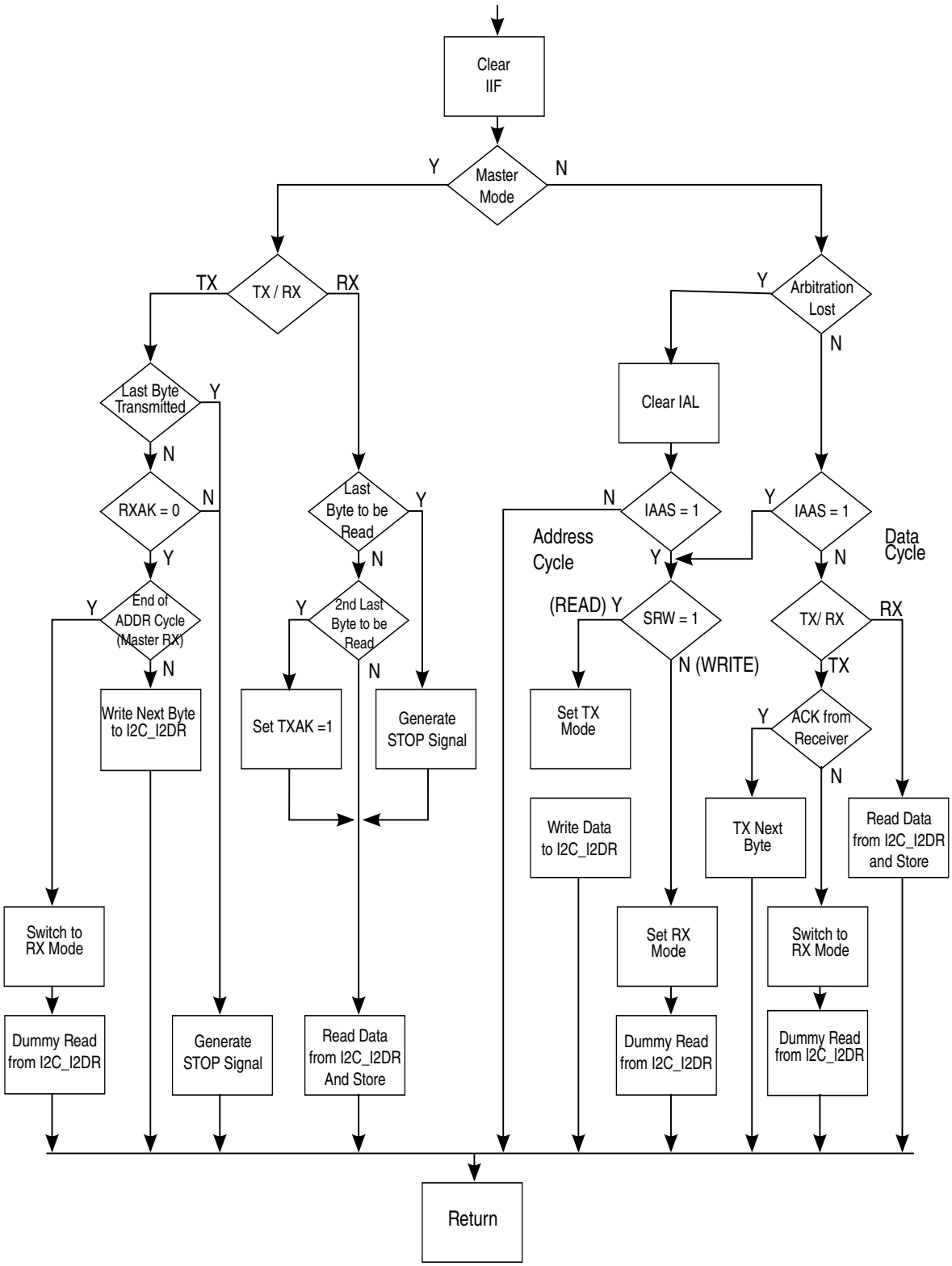


Figure 15-10. Flowchart of typical I2C interrupt routine

NOTE

For a Repeated Start only, the Stop-generation stage does not occur in Master mode. A loop repeats itself without stopping for the next start.

For Master Receive mode, I2C is programmed as Master Transmit during Address mode and after slave address transfer; the MTX bit should be cleared and a dummy read on the I2C_I2DR register should be performed so I2C can read the next receive data.

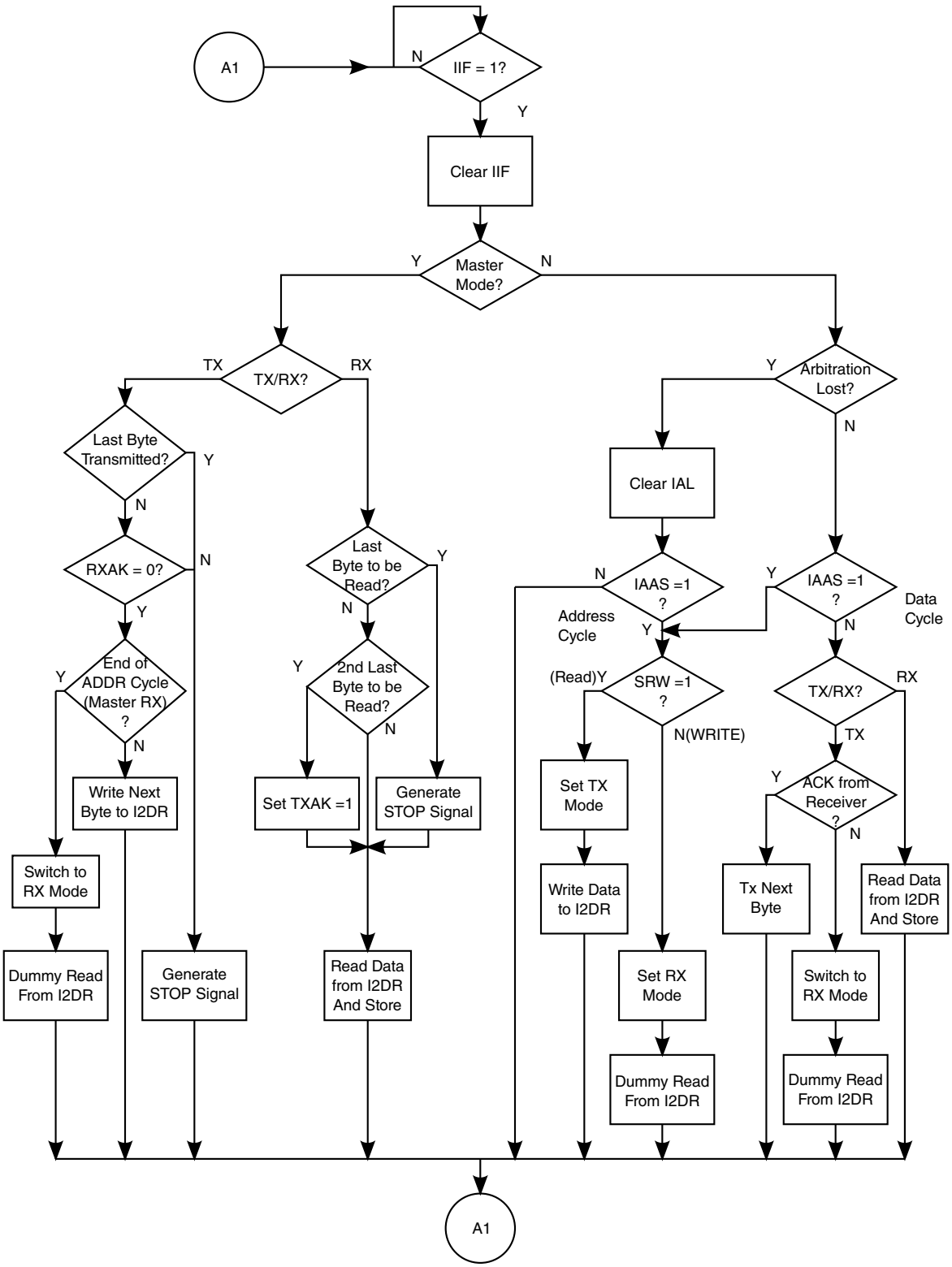


Figure 15-11. Flowchart for typical I2C polling routine

NOTE

The timeout value depends on the bus frequency at which I2C is operating. The minimum timeout for polling the IIF bit at a maximum I2C bus frequency of 400 kHz is T_{min} = 25 µs (=2.5 x 10 µs). This value can be calculated for any bus frequency. The formula is $T_{min} = 10/F_{SCL}$, where F_{SCL} is the frequency of the I2C clock (SCL).

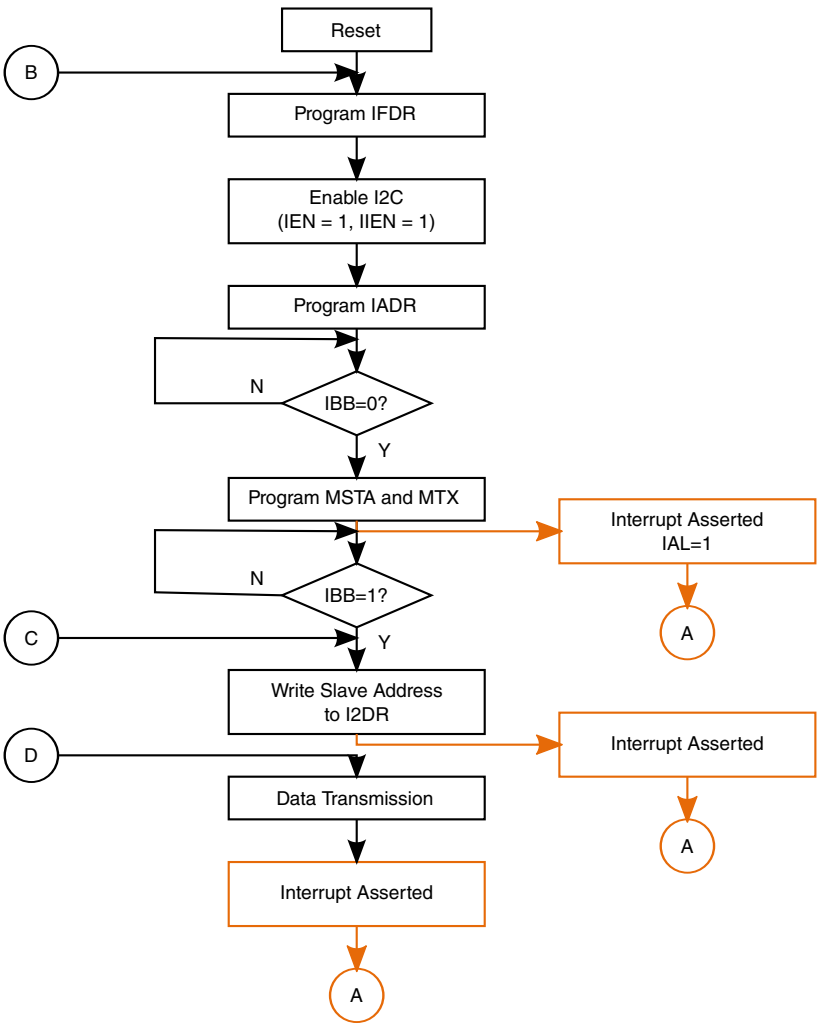


Figure 15-12. Detailed flowchart of a typical I2C Master Transmit mode, part 1

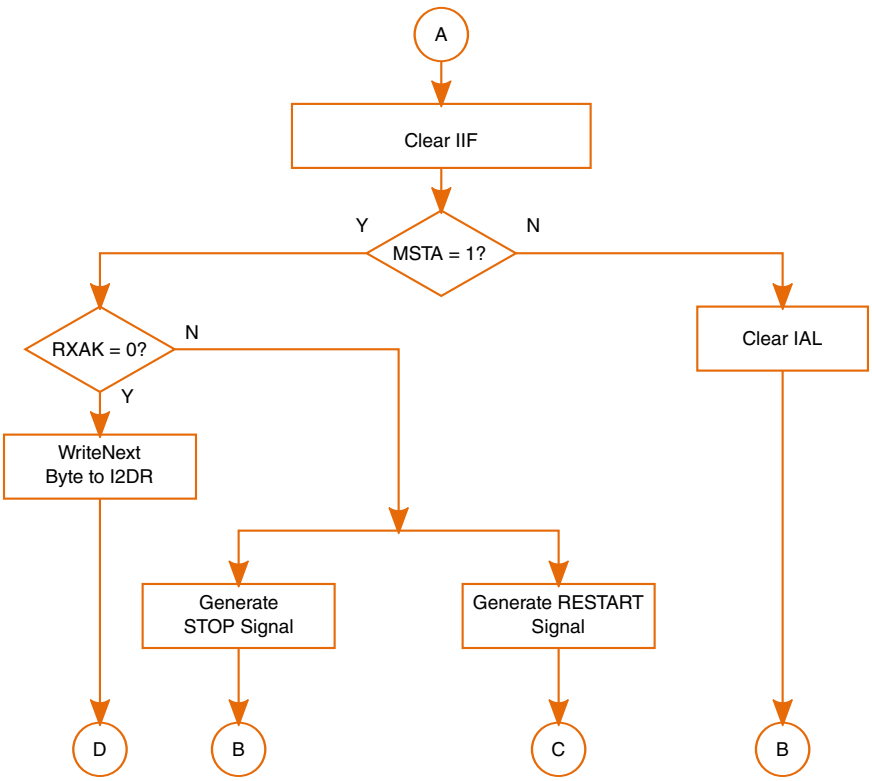


Figure 15-13. Detailed flowchart of a typical I2C Master Transmit mode, part 2

Figure 15-12 and Figure 15-13 show the Master Transmit mode operation with interrupt subroutine. If an interrupt is generated and the MSTA bit is 0, then bus arbitration is lost and IAL is set. Software can clear the IAL bit and reprogram I2C. If the MSTA bit is 1, then it is a transfer-generated interrupt. In this case, software can check the RXAK bit for a data receive acknowledgement by the slave and, accordingly, decide to do one of the following:

- Generate a STOP
- Generate a REPEATED START by writing to the I2C_I2CR register
- Perform the next data transfer by writing to the I2C_I2DR register

NOTE

The IBB bit is asserted by a Start condition on the bus, and it is deasserted by a Stop condition on the bus. Therefore, if arbitration is lost due to an unexpected Stop condition during transfer, then IBB is cleared. If arbitration is lost due to a data mismatch, then it is not cleared. Software should always clear the IEN bit and then set it if arbitration is lost.

15.2.6 Software restriction

Software should ensure that there is a delay of at least two module clock cycles after it sets the I2C_I2CR[RSTA] bit and before writing to the I2C_I2DR register. The maximum possible clock period of the module clock is 78 ns.

15.2.7 I2C Memory Map/Register Definition

The I2C contains five 16-bit registers.

NOTE

Registers at offsets 0x0002, 0x0006, 0x000A, and 0x000E are reserved for future additions.

I2C memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30A2_0000	I2C Address Register (I2C1_IADR)	16	R/W	0000h	15.2.7.1/ 4271
30A2_0004	I2C Frequency Divider Register (I2C1_IFDR)	16	R/W	0000h	15.2.7.2/ 4271
30A2_0008	I2C Control Register (I2C1_I2CR)	16	R/W	0000h	15.2.7.3/ 4273
30A2_000C	I2C Status Register (I2C1_I2SR)	16	R/W	0081h	15.2.7.4/ 4274
30A2_0010	I2C Data I/O Register (I2C1_I2DR)	16	R/W	0000h	15.2.7.5/ 4276
30A3_0000	I2C Address Register (I2C2_IADR)	16	R/W	0000h	15.2.7.1/ 4271
30A3_0004	I2C Frequency Divider Register (I2C2_IFDR)	16	R/W	0000h	15.2.7.2/ 4271
30A3_0008	I2C Control Register (I2C2_I2CR)	16	R/W	0000h	15.2.7.3/ 4273
30A3_000C	I2C Status Register (I2C2_I2SR)	16	R/W	0081h	15.2.7.4/ 4274
30A3_0010	I2C Data I/O Register (I2C2_I2DR)	16	R/W	0000h	15.2.7.5/ 4276
30A4_0000	I2C Address Register (I2C3_IADR)	16	R/W	0000h	15.2.7.1/ 4271
30A4_0004	I2C Frequency Divider Register (I2C3_IFDR)	16	R/W	0000h	15.2.7.2/ 4271

Table continues on the next page...

I2C memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30A4_0008	I2C Control Register (I2C3_I2CR)	16	R/W	0000h	15.2.7.3/ 4273
30A4_000C	I2C Status Register (I2C3_I2SR)	16	R/W	0081h	15.2.7.4/ 4274
30A4_0010	I2C Data I/O Register (I2C3_I2DR)	16	R/W	0000h	15.2.7.5/ 4276
30A5_0000	I2C Address Register (I2C4_IADR)	16	R/W	0000h	15.2.7.1/ 4271
30A5_0004	I2C Frequency Divider Register (I2C4_IFDR)	16	R/W	0000h	15.2.7.2/ 4271
30A5_0008	I2C Control Register (I2C4_I2CR)	16	R/W	0000h	15.2.7.3/ 4273
30A5_000C	I2C Status Register (I2C4_I2SR)	16	R/W	0081h	15.2.7.4/ 4274
30A5_0010	I2C Data I/O Register (I2C4_I2DR)	16	R/W	0000h	15.2.7.5/ 4276

15.2.7.1 I2C Address Register (I2Cx_IADR)

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								ADR							0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Cx_IADR field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value 0.
7–1 ADR	Slave address. Contains the specific slave address to be used by the I2C. Slave mode is the default I2C mode for an address match on the bus. **NOTE:** The I2C_IADR holds the address to which the I2C responds when addressed as a slave. The slave address is not the address sent on the bus during the address transfer. The register is not reset by a software reset.
0 Reserved	This read-only field is reserved and always has the value 0.

15.2.7.2 I2C Frequency Divider Register (I2Cx_IFDR)

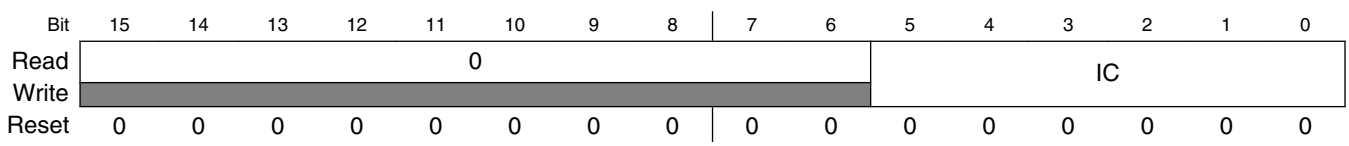
The I2C_IFDR provides a programmable prescaler to configure the clock for bit-rate selection. The register does not get reset by a software reset.

The following table describes the divider and register values for the register field "IC."

Table 15-26. I2C_IFDR Register Field Values

IC	Divider		IC	Divider		IC	Divider		IC	Divider
0x00	30		0x10	288		0x20	22		0x30	160
0x01	32		0x11	320		0x21	24		0x31	192
0x02	36		0x12	384		0x22	26		0x32	224
0x03	42		0x13	480		0x23	28		0x33	256
0x04	48		0x14	576		0x24	32		0x34	320
0x05	52		0x15	640		0x25	36		0x35	384
0x06	60		0x16	768		0x26	40		0x36	448
0x07	72		0x17	960		0x27	44		0x37	512
0x08	80		0x18	1152		0x28	48		0x38	640
0x09	88		0x19	1280		0x29	56		0x39	768
0x0A	104		0x1A	1536		0x2A	64		0x3A	896
0x0B	128		0x1B	1920		0x2B	72		0x3B	1024
0x0C	144		0x1C	2304		0x2C	80		0x3C	1280
0x0D	160		0x1D	2560		0x2D	96		0x3D	1536
0x0E	192		0x1E	3072		0x2E	112		0x3E	1792
0x0F	240		0x1F	3840		0x2F	128		0x3F	2048

Address: Base address + 4h offset



I2Cx_IFDR field descriptions

Field	Description
15–6 Reserved	This read-only field is reserved and always has the value 0.
IC	I2C clock rate. Prescales the clock for bit-rate selection. Due to potentially slow I2Cn_SCL and I2Cn_SDA rise and fall times, bus signals are sampled at the prescaler frequency. The serial bit clock frequency may be lower than IPG_CLK_ROOT divided by the divider shown in the I2C Data I/O Register. **NOTE:** The IC value should not be changed during the data transfer, however, it can be changed before a Repeat Start or Start programming sequence in I2C. The I2C protocol supports bit rates of up to 400 kbps. The IC bits need to be programmed in accordance with this constraint.

15.2.7.3 I2C Control Register (I2Cx_I2CR)

The I2C_I2CR is used to enable the I2C and the I2C interrupt. It also contains bits that govern operation as a slave or a master.

Address: Base address + 8h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	IEN	IIEN	MSTA	MTX	TXAK	0	0	
Write						RSTA		
Reset	0	0	0	0	0	0	0	0

I2Cx_I2CR field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value 0.
7 IEN	I2C enable. Also controls the software reset of the entire I2C. Resetting the bit generates an internal reset to the block. If the block is enabled in the middle of a byte transfer, Slave mode ignores the current bus transfer and starts operating when the next Start condition is detected. Master mode is not aware that the bus is busy, so initiating a start cycle may corrupt the current bus cycle, ultimately causing either the current master or the I2C to lose arbitration. Subsequently, bus operation returns to normal. 0 The block is disabled, but registers can still be accessed. 1 The I2C is enabled. This bit must be set before any other I2C_I2CR bits have an effect.
6 IIEN	I2C interrupt enable. **NOTE:** If data is written during the Start condition, that is, just after setting the I2C_I2CR[MSTA] and I2C_I2CR[MTX] bits, then the ICF bit is cleared at the falling edge of SCLK after Start. If data is written after the Start condition and falling edge of SCLK, then the ICF bit is cleared as soon as data is written. 0 I2C interrupts are disabled, but the status flag I2C_I2SR[IIF] continues to be set when an Interrupt condition occurs. 1 I2C interrupts are enabled. An I2C interrupt occurs if I2C_I2SR[IIF] is also set.
5 MSTA	Master/Slave mode select bit. If the master loses arbitration, MSTA is cleared without generating a Stop signal. **NOTE:** The module clock should be on for writing to the MSTA bit. **NOTE:** The MSTA bit is cleared by software to generate a Stop condition; it can also be cleared by hardware when the I2C loses the bus arbitration.

Table continues on the next page...

I2Cx_I2CR field descriptions (continued)

Field	Description
	0 Slave mode. Changing MSTA from 1 to 0 generates a Stop and selects Slave mode. 1 Master mode. Changing MSTA from 0 to 1 signals a Start on the bus and selects Master mode.
4 MTX	Transmit/Receive mode select bit. Selects the direction of master and slave transfers. 0 Receive. When a slave is addressed, the software should set MTX according to the slave read/write bit in the I2C status register (I2C_I2SR[SRW]). 1 Transmit. In Master mode, MTX should be set according to the type of transfer required. Therefore, for address cycles, MTX is always 1.
3 TXAK	Transmit acknowledge enable. Specifies the value driven onto I2Cn_SDA during acknowledge cycles for both master and slave receivers. **NOTE:** Writing TXAK applies only when the I2C bus is a receiver. 0 An acknowledge signal is sent to the bus at the ninth clock bit after receiving one byte of data. 1 No acknowledge signal response is sent (that is, the acknowledge bit = 1).
2 RSTA	Repeat start. Always reads as 0. Attempting a repeat start without bus mastership causes loss of arbitration. 0 No repeat start 1 Generates a Repeated Start condition
Reserved	This read-only field is reserved and always has the value 0.

15.2.7.4 I2C Status Register (I2Cx_I2SR)

The I2C_I2SR contains bits that indicate transaction direction and status.

Address: Base address + Ch offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	ICF	IAAS	IBB	IAL	0	SRW	IIF	RXAK
Write								
Reset	1	0	0	0	0	0	0	1

I2Cx_I2SR field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value 0.
7 ICF	Data transferring bit. While one byte of data is transferred, ICF is cleared. 0 Transfer is in progress. 1 Transfer is complete. This bit is set by the falling edge of the ninth clock of the last byte transfer.
6 IAAS	I2C addressed as a slave bit. The ARM platform is interrupted if the interrupt enable (I2C_I2CR[IIEN]) is set. The ARM platform must check the slave read/write bit (SRW) and set its Transfer/Receive mode accordingly. Writing to I2C_I2CR clears this bit. 0 Not addressed 1 Addressed as a slave. Set when its own address (I2C_IADR) matches the calling address.
5 IBB	I2C bus busy bit. Indicates the status of the bus. **NOTE:** When I2C is enabled (I2C_I2CR[IEN] = 1), it continuously polls the bus data (SDA) and clock (SCL) signals to determine a Start or Stop condition. 0 Bus is idle. If a Stop signal is detected, IBB is cleared. 1 Bus is busy. When Start is detected, IBB is set.
4 IAL	Arbitration lost. Set by hardware in the following circumstances (IAL must be cleared by software by writing a "0" to it at the start of the interrupt service routine): • I2Cn_SDA input samples low when the master drives high during an address or data-transmit cycle. • I2Cn_SDA input samples low when the master drives high during the acknowledge bit of a data-receive cycle. For the above two cases, the bit is set at the falling edge of the ninth I2Cn_SCL clock during the ACK cycle. • A Start cycle is attempted when the bus is busy. • A Repeated Start cycle is requested in Slave mode. • A Stop condition is detected when the master did not request it. **NOTE:** Software cannot set the bit. 0 No arbitration lost. 1 Arbitration is lost.
3 Reserved	This read-only field is reserved and always has the value 0.
2 SRW	Slave read/write. When the I2C is addressed as a slave, IAAS is set, and the slave read/write bit (SRW) indicates the value of the R/W command bit of the calling address sent from the master. SRW is valid only when a complete transfer has occurred, no other transfers have been initiated, and the I2C is a slave and has an address match. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
1 IIF	I2C interrupt. Must be cleared by the software by writing a "0" to it in the interrupt routine. **NOTE:** The software cannot set the bit. 0 No I2C interrupt pending. 1 An interrupt is pending.

Table continues on the next page...

I2Cx_I2SR field descriptions (continued)

Field	Description
	This causes a processor interrupt request (if the interrupt enable is asserted [IIEN = 1]). The interrupt is set when one of the following occurs: • One byte transfer is completed (the interrupt is set at the falling edge of the ninth clock). • An address is received that matches its own specific address in Slave Receive mode. • Arbitration is lost.
0 RXAK	Received acknowledge. This is the value received from the I2Cn_SDA input for the acknowledge bit during a bus cycle. 0 An "acknowledge" signal was received after the completion of an 8-bit data transmission on the bus. 1 A "No acknowledge" signal was detected at the ninth clock.

15.2.7.5 I2C Data I/O Register (I2Cx_I2DR)

In Master Receive mode, reading the data register allows a read to occur and initiates the next byte to be received. In Slave mode, the same function is available after it is addressed.

Address: Base address + 10h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								DATA							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Cx_I2DR field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value 0.
DATA	Data Byte. Holds the last data byte received or the next data byte to be transferred. Software writes the next data byte to be transmitted or reads the data byte received. **NOTE:** The core-written value in I2C_I2DR cannot be read back by the core. Only data written by the I2C bus side can be read.

15.3 Universal Asynchronous Receiver/Transmitter(UART)

15.3.1 Overview

Universal Asynchronous Receiver/Transmitter (UART) provides serial communication capability with external devices through a level converter and an RS-232 cable or through use of external circuitry that converts infrared signals to electrical signals (for reception) or transforms electrical signals to signals that drive an infrared LED (for transmission) to provide low speed IrDA compatibility.

UART supports NRZ encoding format , RS485 compatible 9 bit data format and IrDA-compatible infrared slow data rate (SIR) format.

Figure 15-14 is the UART block diagram.

The "Module Clock" is the UART_CLK which comes from CCM. The "Peripheral Clock" is the IPG_CLK which comes from CCM.

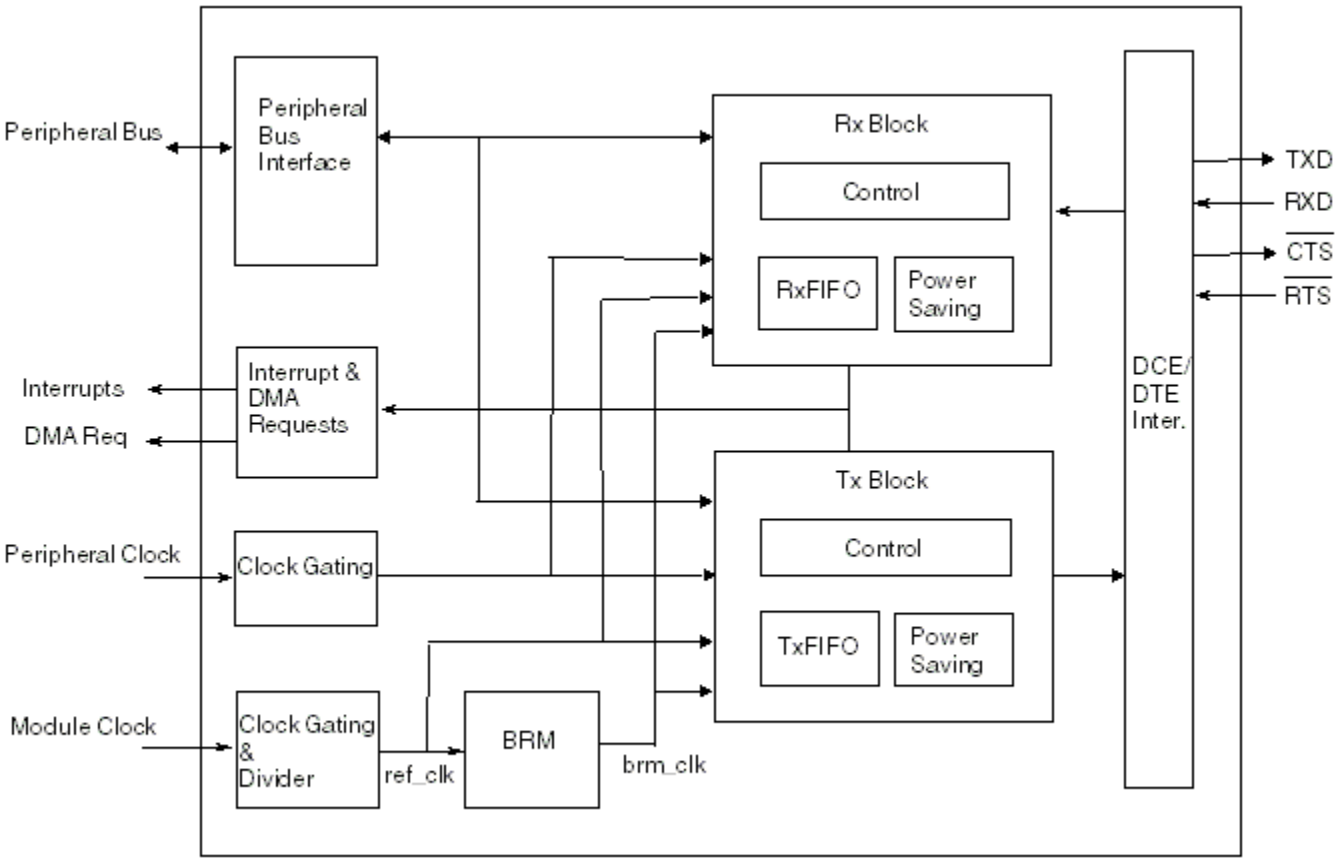


Figure 15-14. UART Block Diagram

15.3.1.1 Features

The UART includes the following features:

- High-speed TIA/EIA-232-F compatible, up to Mbit/s
- Serial IR interface low-speed, IrDA-compatible (up to 115.2 Kbit/s)
- 9-bit or Multidrop mode (RS-485) support (automatic slave address detection)
- 7 or 8 data bits for RS-232 characters, or 9 bit RS-485 format
- 1 or 2 stop bits
- Programmable parity (even, odd, and no parity)
- Hardware flow control support for request to send (RTS_B) and clear to send (CTS_B) signals
- RS-485 driver direction control via CTS_B signal
- Edge-selectable RTS_B and edge-detect interrupts
- Status flags for various flow control and FIFO states
- Voting logic for improved noise immunity (16x oversampling)
- Transmitter FIFO empty interrupt suppression

- UART internal clocks enable/disable
- Auto baud rate detection (up to 115.2 Kbit/s)
- Receiver and transmitter enable/disable for power saving
- RX_DATA input and TX_DATA output can be inverted respectively in RS-232/ RS-485 mode
- DCE/DTE capability
- RTS_B, IrDA asynchronous wake (AIRINT), receive asynchronous wake (AWAKE) interrupts wake the processor from STOP mode
- Maskable interrupts
- Two DMA Requests (TxFIFO DMA Request and RxFIFO DMA Request)
- Escape character sequence detection
- Software reset (SRST_B)
- Two independent, 32-entry FIFOs for transmit and receive
- The peripheral clock can be totally asynchronous with the module clock. The module clock determines baud rate. This allows frequency scaling on peripheral clock (such as during DVFS mode) while remaining the module clock frequency and baud rate.

15.3.1.2 Modes of operation

- Serial RS-232NRZ mode
- 9-bit RS-485 mode
- IrDA mode

To set UART in different modes, see the table below.

Table 15-27. UART mode definition

MDEN (UMCR[0])	IREN (UCR1[7])	UART Mode	Descriptioin
0	0	RS-232	RXD/TXD data is serial RS-232 NRZ format
0	1	IrDA (Interface)	RXD/TXD data is IrDA-compatible infrared slow data rate (SIR) format
1	0	RS-485	RXD/TXD data is RS485 compatible 9 bit data format
1	1	Undefined	Undefined

15.3.2 External Signals

The chip-level IOMUX modifies the direction and routing of the UART signals based on whether the UART is operating in DCE mode (UARTn_UFCR[DCEDTE]=0) or DTE mode (UARTn_UFCR[DCEDTE]=1). The routing of the external signals to the UART module is shown in the figure below.

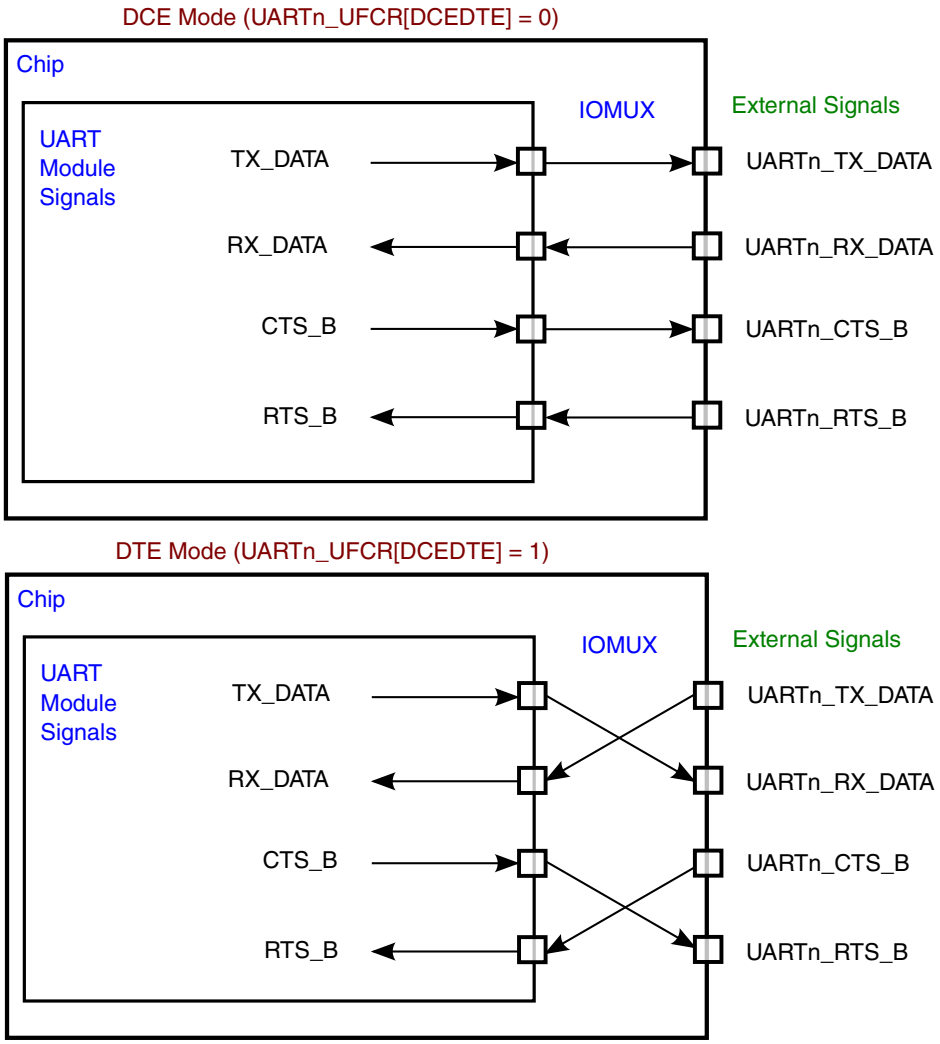


Figure 15-15. UART external signals to module signals routing with respect to DCE/DTE mode

The following table describes the external signals of UART:

Table 15-28. UART External Signals

Signal	Description	Pad	Mode	Direction
UART1_CTS_B	Clear to send	ENET1_RDATA0	ALT3	O
		SAI2_TXFS	ALT3	
UART1_RTS_B	Request to send	ENET1_RDATA1	ALT3	I
		SAI2_TXC	ALT3	
UART1_RX_DATA	Serial / infrared data receive	ENET1_RDATA2	ALT3	I
		UART1_RXD	ALT0	
UART1_TX_DATA	Serial / infrared data transmit	ENET1_RDATA3	ALT3	O
		UART1_TXD	ALT0	
UART2_CTS_B	Clear to send	LCD1_VSYNC	ALT4	O

Table continues on the next page...

Table 15-28. UART External Signals (continued)

Signal	Description	Pad	Mode	Direction
		SAI2_RXD	ALT3	
UART2_RTS_B	Request to send	LCD1_HSYNC	ALT4	I
		SAI2_TXD	ALT3	
UART2_RX_DATA	Serial / infrared data receive	LCD1_CLK	ALT4	I
		UART2_RXD	ALT0	
UART2_TX_DATA	Serial / infrared data transmit	LCD1_ENABLE	ALT4	O
		UART2_TXD	ALT0	
UART3_CTS_B	Clear to send	GPIO1_IO11	ALT3	O
		SD3_DATA7	ALT3	
		UART3_CTS	ALT0	
UART3_RTS_B	Request to send	GPIO1_IO10	ALT3	I
		SD3_DATA6	ALT3	
		UART3_RTS	ALT0	
UART3_RX_DATA	Serial / infrared data receive	GPIO1_IO08	ALT3	I
		SD3_DATA4	ALT3	
		UART3_RXD	ALT0	
UART3_TX_DATA	Serial / infrared data transmit	GPIO1_IO09	ALT3	O
		SD3_DATA5	ALT3	
		UART3_TXD	ALT0	
UART4_CTS_B	Clear to send	I2C1_SCL	ALT1	O
		SAI2_RXD	ALT2	
		SD2_DATA2	ALT2	
UART4_RTS_B	Request to send	I2C1_SDA	ALT1	I
		SAI2_TXD	ALT2	
		SD2_DATA3	ALT2	
UART4_RX_DATA	Serial / infrared data receive	I2C2_SCL	ALT1	I
		SAI2_TXFS	ALT2	
		SD2_DATA0	ALT2	
UART4_TX_DATA	Serial / infrared data transmit	I2C2_SDA	ALT1	O
		SAI2_TXC	ALT2	
		SD2_DATA1	ALT2	
UART5_CTS_B	Clear to send	GPIO1_IO04	ALT3	O
		I2C3_SCL	ALT1	
		SAI1_TXFS	ALT2	
UART5_RTS_B	Request to send	GPIO1_IO05	ALT3	I
		I2C3_SDA	ALT1	
		SAI1_TXD	ALT2	
UART5_RX_DATA	Serial / infrared data receive	GPIO1_IO06	ALT3	I
		I2C4_SCL	ALT1	

Table continues on the next page...

Table 15-28. UART External Signals (continued)

Signal	Description	Pad	Mode	Direction
		SAI1_RXD	ALT2	
UART5_TX_DATA	Serial / infrared data transmit	GPIO1_IO07	ALT3	O
		I2C4_SDA	ALT1	
		SAI1_TXC	ALT2	
UART6_CTS_B	Clear to send	ECSPI1_SS0	ALT1	O
		EPDC1_DATA11	ALT3	
		SD1_CLK	ALT2	
UART6_RTS_B	Request to send	ECSPI1_MISO	ALT1	I
		EPDC1_DATA10	ALT3	
		SD1_RESET_B	ALT2	
UART6_RX_DATA	Serial / infrared data receive	ECSPI1_SCLK	ALT1	I
		EPDC1_DATA08	ALT3	
		SD1_CD_B	ALT2	
UART6_TX_DATA	Serial / infrared data transmit	ECSPI1_MOSI	ALT1	O
		EPDC1_DATA09	ALT3	
		SD1_WP	ALT2	
UART7_CTS_B	Clear to send	ECSPI2_SS0	ALT1	O
		EPDC1_DATA15	ALT3	
		SD1_DATA2	ALT2	
UART7_RTS_B	Request to send	ECSPI2_MISO	ALT1	I
		EPDC1_DATA14	ALT3	
		SD1_DATA3	ALT2	
UART7_RX_DATA	Serial / infrared data receive	ECSPI2_SCLK	ALT1	I
		EPDC1_DATA12	ALT3	
		SD1_DATA0	ALT2	
UART7_TX_DATA	Serial / infrared data transmit	ECSPI2_MOSI	ALT1	O
		EPDC1_DATA13	ALT3	
		SD1_DATA1	ALT2	

The user must configure the input path to the UART by properly configuring the DAISY bits in the IOMUXC_UARTn_RX_DATA_INPUT and the IOMUXC_UARTn_UART_RTS_B_SELECT_INPUT registers.

For IOMUXC_UARTn_UART_RTS_B_SELECT_INPUT[DAISY]:

- Configurations that select UARTn_RTS_B for the pad are only valid when UARTn_UFCR[DCEDTE]=0 (DCE mode)
- Configurations that select UARTn_CTS_B for the pad are only valid when UARTn_UFCR[DCEDTE]=1 (DTE mode)

For IOMUXC_UARTn_UART_RX_DATA_B_SELECT_INPUT[DAISY]:

- Configurations that select UARTn_RX_DATA for the pad are only valid when UARTn_UFCR[DCEDTE]=0 (DCE mode)
- Configurations that select UARTn_TX_DATA for the pad are only valid when UARTn_UFCR[DCEDTE]=1 (DTE mode)

15.3.2.1 Detailed Signal Descriptions

15.3.2.1.1 Interrupt Signals

15.3.2.1.1.1 *interrupt_uart* - UART Interrupt

Output interrupt request.

15.3.2.1.2 DMA Request Signals

15.3.2.1.2.1 *dma_req_rx* - Receiver DMA Request

Output DMA Request signal for receiver interface.

15.3.2.1.2.2 *dma_req_tx* - Transmitter DMA Request

Output DMA Request signal for transmitter interface. Set at 0 when TXDMAEN (UCR1[3]) is at 1 and TRDY (USR1[13]) is also at 1.

15.3.2.1.3 Special Signals

15.3.2.1.3.1 *stop_req* - Stop Mode

Input stop mode. Indicates to UART that ARM platform is going to enter in Stop Mode and clocks are going to stop running.

See Low Power Modes for more information about Stop Mode.

15.3.2.1.3.2 *doze_req* - Doze Mode

Input doze mode. ARM platform requests UART to switch in doze mode (power saving mode).

See Low Power Modes for more information about Doze Mode.

15.3.2.1.3.3 *debug_req* - Debug Mode

Input debug mode. Indicates UART it has to enter in debug mode.

See UART Operation in System Debug State, for more information about Debug Mode.

15.3.3 Clocks

The table found here describes the clock sources for UART.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 15-29. UART Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock
ipg_perclk	uart_clk_root	Module clock

15.3.4 Functional Description

This section provides a complete functional description of the block.

15.3.4.1 Interrupts and DMA Requests

See the following table for the lists of all interrupt and DMA signals and associated interrupt and DMA sources of the UART. See register description section for explanation of interrupt/DMA enable and status.

Table 15-30. Interrupts and DMA

Interrupt/DMA Output	Interrupt/DMA Enable	Enable Register Location	Interrupt/DMA Flag	Flag Register Location
interrupt_uart	RRDYEN	UCR1 (bit 9)	RRDY	USR1 (bit 9)
	IDEN	UCR1 (bit 12)	IDLE	USR2 (bit 12)
	DREN	UCR4 (bit 0)	RDR	USR2 (bit 0)
	RXDSEN	UCR3 (bit 6)	RXDS	USR1 (bit 6)
	ATEN	UCR2 (bit 3)	AGTIM	USR1 (bit 8)
interrupt_uart	TXMPTYEN	UCR1 (bit 6)	TXFE	USR2 (bit 14)

Table continues on the next page...

Table 15-30. Interrupts and DMA (continued)

Interrupt/DMA Output	Interrupt/DMA Enable	Enable Register Location	Interrupt/DMA Flag	Flag Register Location
	TRDYEN	UCR1 (bit 13)	TRDY	USR1 (bit 13)
	TCEN	UCR4 (bit 3)	TXDC	USR2 (bit 3)
interrupt_uart	OREN	UCR4 (bit 1)	ORE	USR2 (bit 1)
	BKEN	UCR4 (bit 2)	BRCD	USR2 (bit 2)
	WKEN	UCR4 (bit 7)	WAKE	USR2 (bit 7)
	ADEN	UCR1 (bit 15)	ADET	USR2 (bit 15)
	ACIEN	UCR3 (bit 0)	ACST	USR2 (bit 11)
	ESCI	UCR2 (bit 15)	ESCF	USR1 (bit 11)
	ENIRI	UCR4 (bit 8)	IRINT	USR2 (bit 8)
	AIRINTEN	UCR3 (bit 5)	AIRINT	USR1 (bit 5)
	AWAKEN	UCR3 (bit 4)	AWAKE	USR1 (bit 4)
	FRAERREN	UCR3 (bit 11)	FRAERR	USR1 (bit 10)
	PARERREN	UCR3 (bit 12)	PARITYERR	USR1 (bit 15)
	RTSDEN	UCR1 (bit 5)	RTSD	USR1 (bit 12)
	RTSEN	UCR2 (bit 4)	RTSF	USR2 (bit 4)
	DTREN (DCE)	UCR3 (bit 13)	DTRF	USR2 (bit 13)
	RI (DTE)	UCR3 (bit 8)	RIDELT	USR2 (bit 10)
	DCD (DTE)	UCR3 (bit 9)	DCDDELT	USR2 (bit 6)
	DTRDEN	UCR3 (bit 3)	DTRD	USR1 (bit 7)
	SADEN	UMCR (bit 3)	SAD	USR1 (bit 3)
dma_req_rx	RXDMAEN	UCR1 (bit 8)	RRDY	USR1 (bit 9)
	ATDMAEN	UCR1 (bit 2)	AGTIM	USR1 (bit 8)
	IDDMAEN	UCR4 (bit 6)	IDLE	USR2 (bit 12)
dma_req_tx	TXDMAEN	UCR1 (bit 3)	TRDY	USR1 (bit 13)

15.3.4.2 Clocks

This section describes clocks and special clocking requirements of the UART.

15.3.4.2.1 Clock requirements

UART module receives 2 clocks, *peripheral_clock* and *module_clock*. The *peripheral_clock* is used as write clock of the TxFIFO, read clock of the RxFIFO and synchronization of the modem control input pins. It must always be running when UART is enabled. There is an exception in stop mode (see Clocking in Low-Power Modes).

The *module_clock* is for all the state machines, writing RxFIFO, reading TxFIFO, etc. It must always be running when UART is sending or receiving characters.This clock is used in order to allow frequency scaling on *peripheral_clock* without changing configuration of baud rate (*module_clock* staying at a fixed frequency).

The constraints on *peripheral_clock* and *module_clock* are as follows:

- *peripheral_clock* and *module_clock* can totally be asynchronous. They can also be synchronous.
- Due to the 16x oversampling of the incoming characters, *module_clock* frequency must always be greater or equal to 16x the maximum baud rate. For example, if max baud rate is 4 Mbit/s, *module_clock* must be greater or equal to 4 M x 16 = 64 MHz.

NOTE

The restriction that *peripheral_clock* frequency must be higher or equal to 16x baud rate has been removed. There is no limitation on *peripheral_clock* frequency to baud rate.

15.3.4.2.2 Maximum Baud Rate

The max baud rate the UART can support is determined by the max frequency of the module_clock.

For example, if the SoC can provide the fastest module_clock 66.5 MHz, the UART can transmit and receive serial data with the maximum baud rate 66.5M/16 = 4.15 Mbit/s.

The UART supports serial IR interface low speed. In the low speed IrDA mode, the max baud rate is 115.2 Kbit/s. To support the 115.2 Kbit/s, module_clock frequency must be higher or equal to 1.8432 MHz.

15.3.4.2.3 Clocking in Low-Power Modes

The UART supports 2 low-power modes: DOZE and STOP.

In STOP mode (input pin *stop_req* is at '1'), the UART doesn't need any clock. In this mode the UART can wake-up the ARM platform with the asynchronous interrupts (see Low Power Modes).

- If before entering in STOP mode the software has enabled RTSDEN interrupt, when RTS will change state (put at '0' by external device started to send), the asynchronous interrupt will wake-up the system, *peripheral_clock* and *module_clock* will be provided to the UART before first start bit, so that no data will be lost.
- If RTS doesn't change state (already at '0' before entering in STOP mode), then wake-up interrupt (AWAKE) will be sent at the arrival of first Start bit (on falling

edge). In this case, the UART must receive the *peripheral_clock* and *module_clock* during the first half of start bit to correctly receive this character (for example, at 115.2 Kbit/s, UART must receive *peripheral_clock* and *module_clock* at maximum 4.3 microseconds after falling edge of Start bit). If the UART receives *peripheral_clock* and *module_clock* too late, first character will be lost, and so should be dropped. Also, if autobaud detection is enabled, the first character won't be correctly received and another autobaud detection will need to be initiated.

In Doze mode, UART behavior is programmable through DOZE bit (UCR1[1]). If DOZE bit is set to '1', then UART is disabled in Doze mode, and in consequence, UART clocks can be switched-off (after being sure UART is not transmitting nor receiving). On the contrary, if DOZE bit is set to '0', UART is enabled and it must receive *peripheral_clock* and *module_clock* .

15.3.4.3 General UART Definitions

Definitions of terms that occurs the following discussions are given in this section.

- Bit Time-The period of time required to serially transmit or receive 1 bit of data (1 cycle of the baud rate frequency).
- Start bit-The bit time of a logic 0 that indicates the beginning of a data frame. A start bit begins with a 1-to-0 transition, and is preceded by at least 1 bit time of logic 1.
- Stop bit-1 bit time of logic 1 that indicates the end of a data frame.
- BREAK-A frame in which all of the data bits, including the stop bit, are logic 0. This type of frame is usually sent to signal the end of a message or the beginning of a new message.
- Mark - When no data is being sent, the serial port's transmit pin's voltage is 1 and is said to be in a MARK state.
- Space - The serial port can also be forced to keep the transmit pin at a 0 and is said to be the SPACE or BREAK state.
- Frame-A start bit followed by a specified number of data or information bits and terminated by a stop bit. The number of data or information bits depends on the format specified and must be the same for the transmitting device and the receiving device. The most common frame format is 1 start bit followed by 8 data bits (least significant bit first) and terminated by 1 stop bit. An additional stop bit and a parity bit also can be included.
- Framing Error-An error condition that occurs when the stop bit of a received frame is missing, usually when the frame boundaries in the received bit stream are not synchronized with the receiver bit counter. Framing errors can go undetected if a data bit in the expected stop bit time happens to be a logic 1. A framing error is always present on the receiver side when the transmitter is sending BREAKs. However,

when the UART is programmed to expect 2 stop bits and only the first stop bit is received, this is not a framing error by definition.
- Parity Error-An error condition that occurs when the calculated parity of the received data bits in a frame does not match the parity bit received on the RX_DATA input. Parity error is calculated only after an entire frame is received.
- Idle-One in NRZ encoding format and selectable polarity in IrDA mode.
- Overrun Error-An error condition that occurs when the latest character received is ignored to prevent overwriting a character already present in the UART receive buffer (RxFIFO). An overrun error indicates that the software reading the buffer (RxFIFO) is not keeping up with the actual reception of characters on the RX_DATA input.

15.3.4.3.1 RTS_B - UART Request To Send

The UART Request To Send input controls the transmitter. The modem or other terminal equipment signals the UART when it is ready to receive by setting '0' on the RTS_B pin.

Normally, the transmitter waits until this signal is active (low) before transmitting a character, however when the Ignore RTS (IRTS) bit is set, the transmitter sends a character as soon as it is ready to transmit. An interrupt (RTSD) can be posted on any transition of this pin and can wake the ARM platform from STOP mode on its assertion. When RTS_B is set to '1' during a transmission, the UART transmitter finishes transmitting the current character and shuts off. The contents of the TxFIFO (characters to be transmitted) remain undisturbed. The operation of this input is the same regardless of whether the UART is in DTE or DCE mode.

15.3.4.3.2 RTS Edge Triggered Interrupt

The input to the RTS_B pin can be programmed to generate an interrupt on a selectable edge.

See the table below for summary of the operation of the RTS edge triggered interrupt (RTSF).

To enable the RTS_B pin to generate an interrupt, set the request to send interrupt enable (RTSEN) bit (UCR2[4]) to 1. Writing 1 to the RTS_B edge triggered interrupt flag (RTSF) bit (USR2[4]) clears the interrupt flag. The interrupt can occur on the rising edge, falling edge, or either edge of the RTS_B input. The request to send edge control (RTEC) field (UCR2[10:9]) programs the edge that generates the interrupt. When RTEC is set to 0x00 and RTSEN = 1, the interrupt occurs on the rising edge (default). When RTEC is set to 0x01 and RTSEN = 1, the interrupt occurs on the falling edge. When RTEC is set to 0x1X and RTSEN = 1, the interrupt occurs on either edge. This is a synchronous interrupt. The RTSF bit is cleared by writing 1 to it. Writing 0 to RTSF has no effect.

Table 15-31. RTS_B Edge Triggered Interrupt Truth Table

RTS_B	RTSEN	RTEC [1]	RTEC [0]	RTSF	Interrupt Occurs On…	interrupt_uart
X	0	X	X	0	Interrupt disabled	1
1->0	1	0	0	0	Rising edge	1
0->1	1	0	0	1	Rising edge	0
1->0	1	0	1	1	Falling edge	0
0->1	1	0	1	0	Falling edge	1
1->0	1	1	X	1	Either edge	0
0->1	1	1	X	1	Either edge	0

There is another RTS_B interrupt that is not programmable. The status bit RTSD asserts the *interrupt_uart* interrupt when the RTS_B delta interrupt enable = 1. This is an asynchronous interrupt. The RTSD bit is cleared by writing 1 to it. Writing 0 to the RTSD bit has no effect.

15.3.4.3.3 CTS_B - Clear To Send

This output pin serves two purposes. Normally, the receiver indicates that it is ready to receive data by asserting this pin (low). When the CTS_B trigger level is programmed to trigger at 32 characters received and the receiver detects the valid start bit of the 33 character, it de-asserts this pin. The operation of this output is the same regardless of whether the UART is in DTE or DCE mode.

15.3.4.3.4 Programmable CTS_B Deassertion

The CTS_B output can also be programmed to deassert when the RxFIFO reaches a certain level. Setting the CTS trigger level (UCR4[15:10]) at any value less than 32 deasserts the CTS_B pin on detection of the valid start bit of the N + 1 character (where N is the trigger level setting). However, the receiver continues to receive characters until the RxFIFO is full.

15.3.4.3.5 TX_DATA - UART Transmit

This is the transmitter serial output. When operating in RS-232/RS-485 mode, NRZ encoded data is transmitted, and the data can be inverted (controlled by INVT (UCR3[1])) before transmitted. When operating in infrared mode, a 3/16 bit-period pulse is output for each 0 bit transmitted, and no pulse is output for each 1 bit transmitted.

For RS-232/RS-485 applications, this pin must be connected to an RS-232/RS-485 transmitter. The operation of this output is the same regardless of whether the UART is in DTE or DCE mode. See Figure 15-16.

15.3.4.3.6 RX_DATA - UART Receive

This is the receiver serial input. When operating in RS-232/RS-485 mode, NRZ encoded data is expected, and the data can be inverted (controlled by INVR (UCR4[9])) before sampled. When operating in infrared mode, a narrow pulse is expected for each 0 bit received and no pulse is expected for each 1 bit received.

External circuitry must convert the IR signal to an electrical signal. RS-232/RS-485 applications require an external RS-232/RS-485 receiver to convert voltage levels. The operation of this input is the same regardless of whether the UART is in DTE or DCE mode. See the figure below.

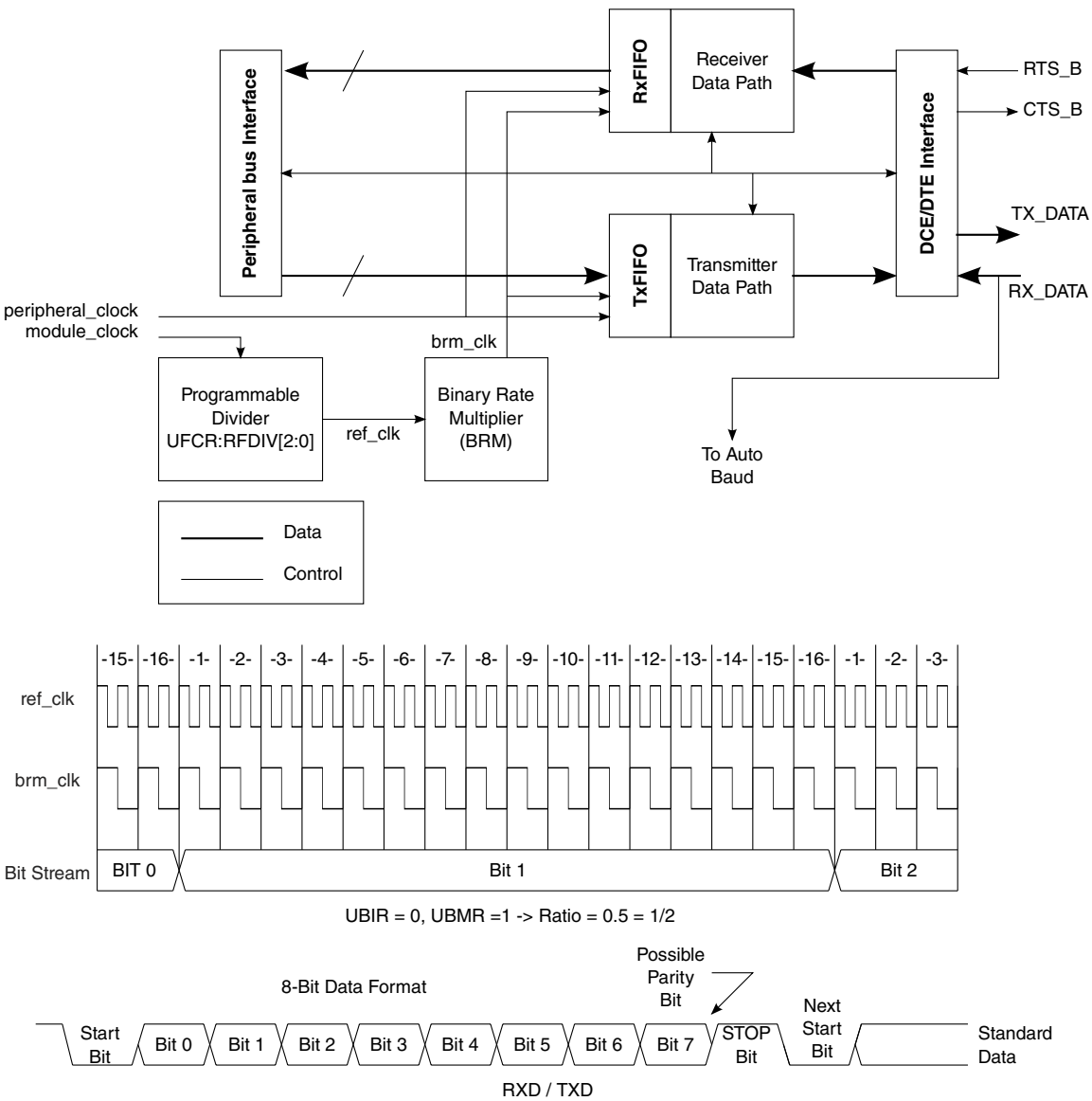


Figure 15-16. UART Simplified Block and Clock Generation Diagrams

15.3.4.4 UART Ports Mapping in DCE/DTE Mode

The table below shows UART signal mapping in DCE and DTE mode.

IO Pads Name	DCE mode		DTE mode	
	IO direction	Uart function port	IO direction	Uart function port
UART_CTS_B	O	CTS_B	I	RTS_B
UART_RTS_B	I	RTS_B	O	CTS_B
UART_TXD	O	TXD	I	RXD
UART_RXD	I	RXD	O	TXD

15.3.4.5 Transmitter

The transmitter accepts a parallel character from the ARM platform and transmits it serially. The start, stop, and parity (when enabled) bits are added to the character.

When the ignore RTS bit (IRTS) is set, the transmitter sends a character as soon as it is ready to transmit. RTS_B can be used to provide flow-control of the serial data. When RTS_B is set to '1', the transmitter finishes sending the character in progress (if any), stops, and waits for RTS_B to be set to '0' again. Generation of BREAK characters and parity errors (for debugging purposes) is supported. The transmitter operates from the clock provided by the Binary Rate Multiplier(BRM). Normal NRZ encoded data is transmitted when the IR interface is disabled.

The transmitter FIFO (TxFIFO) contains 32 bytes. The data is written to TxFIFO by writing to the UTXD register with the byte data to the [7:0] bits. The data is written consecutively if the TxFIFO is not full. It is read (internally) consecutively if the TxFIFO is not empty. TXFULL bit (UTS[4]) can be used to control whether TXFIFO is full or not. The TxFIFO can be written regardless of the transmitter is disabled or enabled. If the UART is disabled, user can still write data into the TxFIFO correctly. But in this case the write access will yield to a transfer error.

15.3.4.5.1 Transmitter FIFO Empty Interrupt Suppression

The transmitter FIFO empty interrupt suppression logic suppresses the TXFE interrupt between writes to the TxFIFO.

When TxFIFO is empty, the software can either send one or several characters. If the software sends one character, it would write the character into the UTXD register, then that character is immediately transferred to the transmitter shift register, assuming the

transmitter is already enabled. Without interrupt suppression logic, the TXFE interrupt flag would be set immediately. But, with this logic, the interrupt flag is set when the last bit of the character has been transmitted, for example, before the transmission of the parity bit (if exists) and the stop bit(s).

So, the suppression logic doesn't immediately send the TXFE interrupt flag. It allows the software to write another character to the TxFIFO before the interrupt flag is asserted.

When the transmitter shift register empties before another character is written to the TxFIFO, the interrupt flag is asserted. Writing data to the TxFIFO would release the interrupt flag. The interrupt flag is asserted on the following conditions:

- System Reset
- UART software reset
- When a single character has been written to Transmitter FIFO and then the Transmitter FIFO and the Transmitter Shift Register become empty until another character is written to the Transmitter FIFO
- The last character in the TxFIFO is transferred to the shift register, when TxFIFO contains two or more characters. See the figure below.

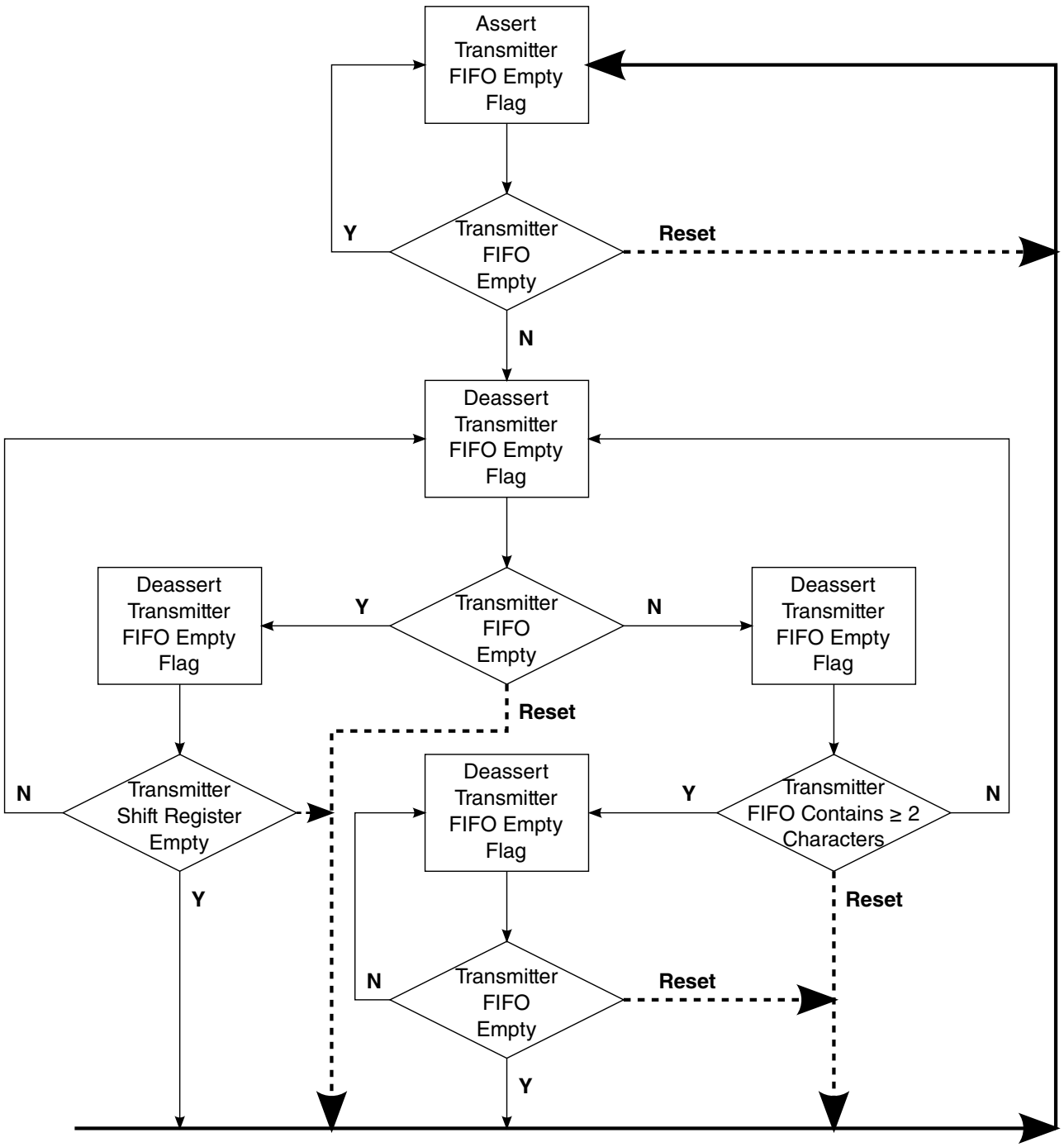


Figure 15-17. Transmitter FIFO Empty Interrupt Suppression Flow Chart

15.3.4.5.2 Transmitting a Break Condition

Asserting SNDBRK bit of the UCR1 Register forces the transmitter to send a break character (continuous zeros). The transmitter will finish sending the character in progress (if any) before sending break until this bit is reset.

The user is responsible to ensure that this bit is high for long enough to generate a valid BREAK. The transmitter samples SNDBRK after every bit is transmitted. Following completion of the BREAK transmission, the UART will transmit two mark bits. The user can continue to fill the FIFO and any character remaining will be transmitted when the break is terminated.

15.3.4.6 Receiver

See the figure below for the receiver flow chart.

The receiver accepts a serial data stream and converts it into parallel characters. When enabled, it searches for a start bit, qualifies it, and samples the following data bits at the bit-center.

Jitter tolerance and noise immunity are provided by sampling at a 16x rate and using voting techniques to clean up the samples. Once the start bit is found, the data bits, parity bit (if enabled), and stop bits (either 1 or 2 depending on user selection) are shifted in. Parity is checked and its status reported in the URXD register when parity is enabled. Frame errors and BREAKs are also checked and reported. When a new character is ready to be read by the ARM platform from the RxFIFO, the receive data ready (RDR = USR2[0]) bit is asserted and an interrupt is posted (if DREN = UCR4[0] = 1). If the receiver trigger level is set to 2 (RXTL[5:0] = UFCR[5:0] = 2), and 2 chars have been received into RxFIFO, the receiver ready interrupt flag (RRDY = USR1[9]) is asserted and an interrupt is posted if the receiver ready interrupt enable bit is set (RRDYEN = UCR1[9] = 1). If the UART Receiver Register (URXD) is read once, and in consequence there is only 1 character in the RxFIFO, the interrupt generated by the RDR bit is automatically cleared. The RRDY bit is cleared when the data in the RxFIFO falls below the programmed trigger level.

Normal NRZ encoded data is expected when the IR interface is disabled. The RxFIFO contains 32 half-word entries. Characters received are written consecutively into this FIFO. If the FIFO is full and a 33rd characters is received, this character will be ignored and the USR2[ORE] bit will be set.

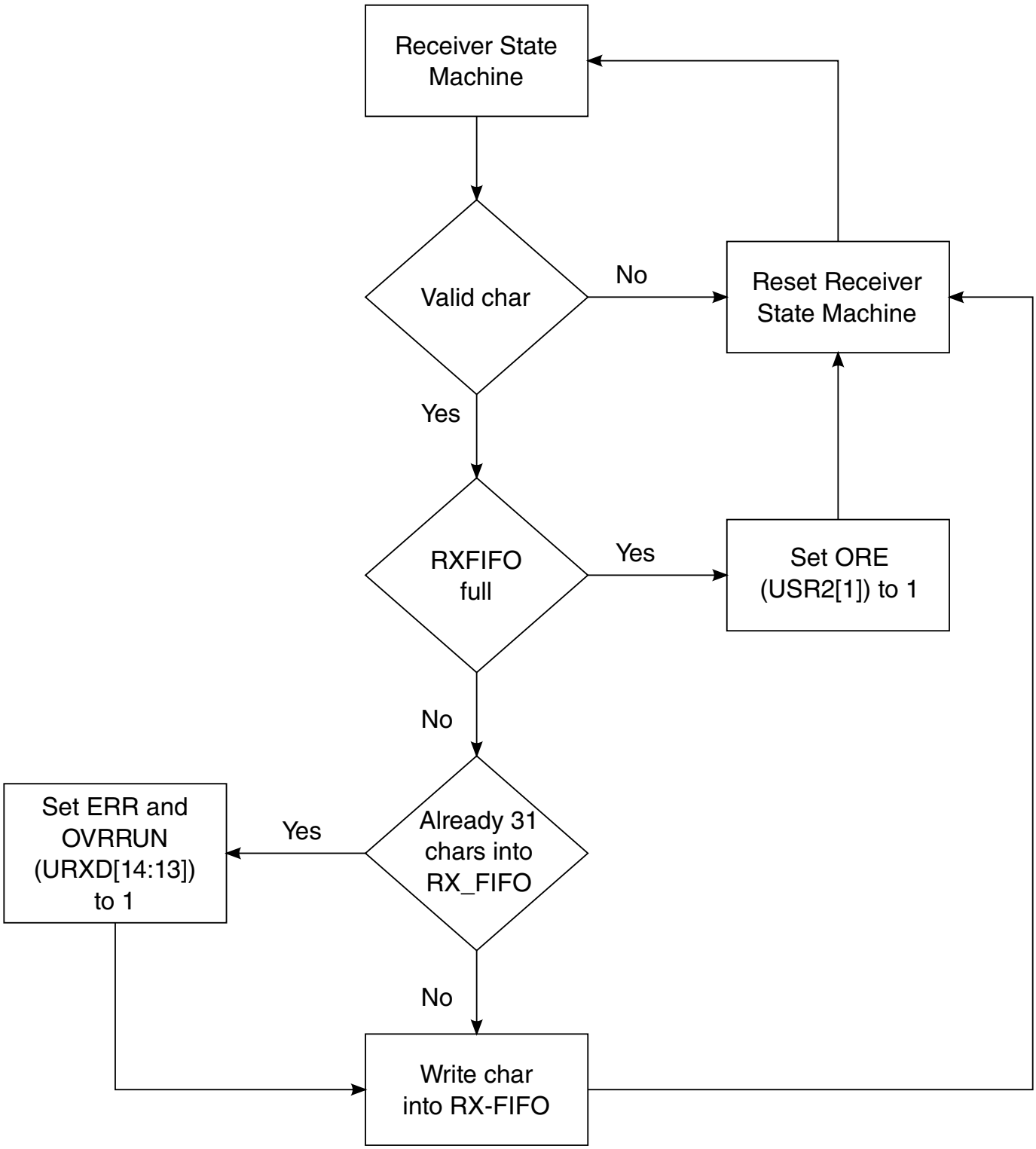


Figure 15-18. Receiver Flow Chart

15.3.4.6.1 Idle Line Detect

The receiver logic block includes the ability to detect an idle line. Idle lines indicate the end or the beginning of a message.

For an idle condition to occur:

- RxFIFO must be empty and
- RX_DATA pin must be idle for more than a configured number of frames (ICD[1:0] = UCR1[11:10]).

When the idle condition detected interrupt enable (IDEN = UCR1[12]) is set and the line is idle for 4 (default), 8, 16, or 32 (maximum) frames, the detection of an idle condition flags an interrupt (see the table below). When an idle condition is detected, the IDLE (USR2[12]) bit is set. Clear the IDLE bit by writing 1 to it. Writing 0 to the IDLE bit has no effect.

Table 15-32. Detection Truth Table

IDEN	ICD [1]	ICD [0]	IDLE	*interrupt_uart*
0	X	X	0	1
1	0	0	asserted after 4 idle frames	asserted after 4 idle frames
1	0	1	asserted after 8 idle frames	asserted after 8 idle frames
1	1	0	asserted after 16 idle frames	asserted after 16 idle frames
1	1	1	asserted after 32 idle frames	asserted after 32 idle frames
NOTE: This table assumes that no other interrupt is set at the same time this interrupt is set for the *interrupt_uart* signal. This table shows how this interrupt affects the *interrupt_uart* signal.				

During a normal message there is no idle time between frames. When all of the information bits in a frame are logic 1s, the start bit ensures that at least one logic 0 bit time occurs for each frame so that the IDLE bit is not asserted.

15.3.4.6.2 Aging Character Detect

The receiver block also includes the possibility to detect when at least one character has been sitting into the RxFIFO for a time corresponding to 8 characters. This aging character capability allows the UART to inform the ARM platform that there is less character into the RxFIFO than the Rx trigger and, no new character has been detected on the RXD line.

The aging capability is a timer which starts to count as soon as the RxFIFO is not empty and its trigger level is not reached (RRDY=0). This counter is reset when either a RxFIFO read is performed or another character starts to present on the RXD line. If none of those two events occurs, the bit AGTIM (USR1[8]) is set when the counter has

measured a time corresponding to 8 characters. AGTIM is cleared by writing a 1 to it. AGTIM can flag an interrupt to ARM platform on *interrupt_uart* if ATEN (UCR2[3]) has been set.

To summarize, AGTIM is set when:

- There is at least one character into RxFIFO.
- No read has occurred on RxFIFO and RXD line has stayed high, for a time corresponding to 8 characters.
- The RxFIFO trigger is not reached (RRDY=0)

15.3.4.6.3 Receiver Wake

The WAKE bit (USR2[7]) is set when the receiver detects a qualified Start bit. For this, two conditions must be fulfilled, firstly a falling edge on RX_DATA line must be detected and secondly the RX_DATA line must stays at low level for more than a half-bit duration.

When the wake interrupt enable WKEN (UCR4[7]) bit is enabled, the receiver flags an interrupt (*interrupt_uart*) if the WAKE status bit is set. The WAKE bit is cleared by writing 1 to it. Writing 0 to the WAKE bit has no effect. The WAKE status bit can be asserted in either serial RS-232 mode or IR mode. The generation of the WAKE interrupt needs the clock *module_clock* .

When the asynchronous wake interrupt (AWAKE) is enabled (AWAKEN = UCR3[4] = 1), and the ARM platform is in STOP mode, and UART clocks have been shut-off, then a falling edge detected on the receive pin (RX_DATA) asserts the AWAKE bit (USR1[4]) and the *interrupt_uart* interrupt to wake the ARM platform from STOP mode. Re-enable UART clocks and clear the AWAKE bit by writing 1 to it. Writing 0 to the AWAKE bit has no effect. When IR interface is enabled (UCR1[7]=1), the AWAKE bit is always not asserted. The generation of the asynchronous AWAKE interrupt does not need any clocks.

In IR mode, if the asynchronous IR WAKE interrupt is enabled (AIRINTEN = UCR3[5] = 1), and if the ARM platform is in STOP mode (UART clocks are off when ARM platform in STOP mode), then the detection of a falling edge on the receive pin (RXD_IR), asserts the AIRINT bit (USR1[5]), and the *interrupt_uart* interrupt. This interrupt wakes the ARM platform from STOP mode. Software re-enables UART clocks and clear the AIRINT bit by writing 1 to it. Writing 0 to the AIRINT bit has no effect. When IR interface is disabled (UCR1[7]=0), the AIRINT bit is always not asserted. The generation of the asynchronous AIRINT interrupt does not need any clocks.

Recommended procedure for programming the asynchronous interrupts is to first clear them by writing 1 to the appropriate bit in the UART Status Register 1 (USR1). Poll or enable the interrupt for the Receiver IDLE Interrupt Flag (RXDS) in the USR1. When asserted, the RXDS bit indicates to the software that the receiver state machine is in the idle state, the next state is idle, and the RX_DATA pin is idle (high). After following this procedure, enable the asynchronous interrupt and enter STOP mode.

15.3.4.6.4 Receiving a BREAK Condition

A BREAK condition is received when the receiver detects all 0s (including a 0 during the bit time of the stop bit) in a frame. The BREAK condition asserts the BRCD bit (USR2[2]) and writes only the first BREAK character to the RxFIFO. Clear the BRCD bit by writing 1 to it. Writing 0 to the BRCD bit has no effect.

Asserting BRCD would generate an interrupt on *interrupt_uart*. The interrupt generation can be masked using the control bit BKEN (UCR4[2]). Receiving a break condition will also effect the following bits in the receiver register URXD:

URXD(11) = BRK. While high this bit indicates that the current char was detected as a break.

URXD(12) = FRMERR. The frame error bit will always be set when BRK is set.

URXD(10) = PRERR. If odd parity was selected the parity error bit will also be set when BRK is set.

URXD(14) = ERR. The error detect bit indicates that the character present in the rx data field has an error status. This can be asserted by a break.

15.3.4.6.5 Vote Logic

The vote logic block provides jitter tolerance and noise immunity by sampling with respect to a 16x clock (*brm_clk*) and using voting techniques to clean up the samples. The voting is implemented by sampling the incoming signal constantly on the rising edge of the *brm_clk*.

See Figure 15-19. The receiver is provided with the majority vote value, which is 2 out of the 3 samples. For examples of the majority vote results of the vote logic, see the following table.

Table 15-33. Majority Vote Results

Samples	Vote
000	0

Table continues on the next page...

Table 15-33. Majority Vote Results (continued)

Samples	Vote
101	1
001	0
111	1

The vote logic captures a sample on every rising edge of *brm_clk*, however the receiver uses 16x oversampling to take its value in the middle of the sample character.

The receiver starts to count when the Start bit is set however it does not capture the contents of the RxFIFO at the time the Start bit is set. The start bit is validated when 0s are received for 7 consecutive 1/16 of bit times following the 1-to-0 transition. Once the counter reaches 0xF, it starts counting on the next bit and captures it in the middle of the sampling frame (see Table 15-33). All data bits are captured in the same manner. Once the stop bit is detected, the receiver shift register (SIPO_OUT) data is parallel shifted to the RxFIFO.

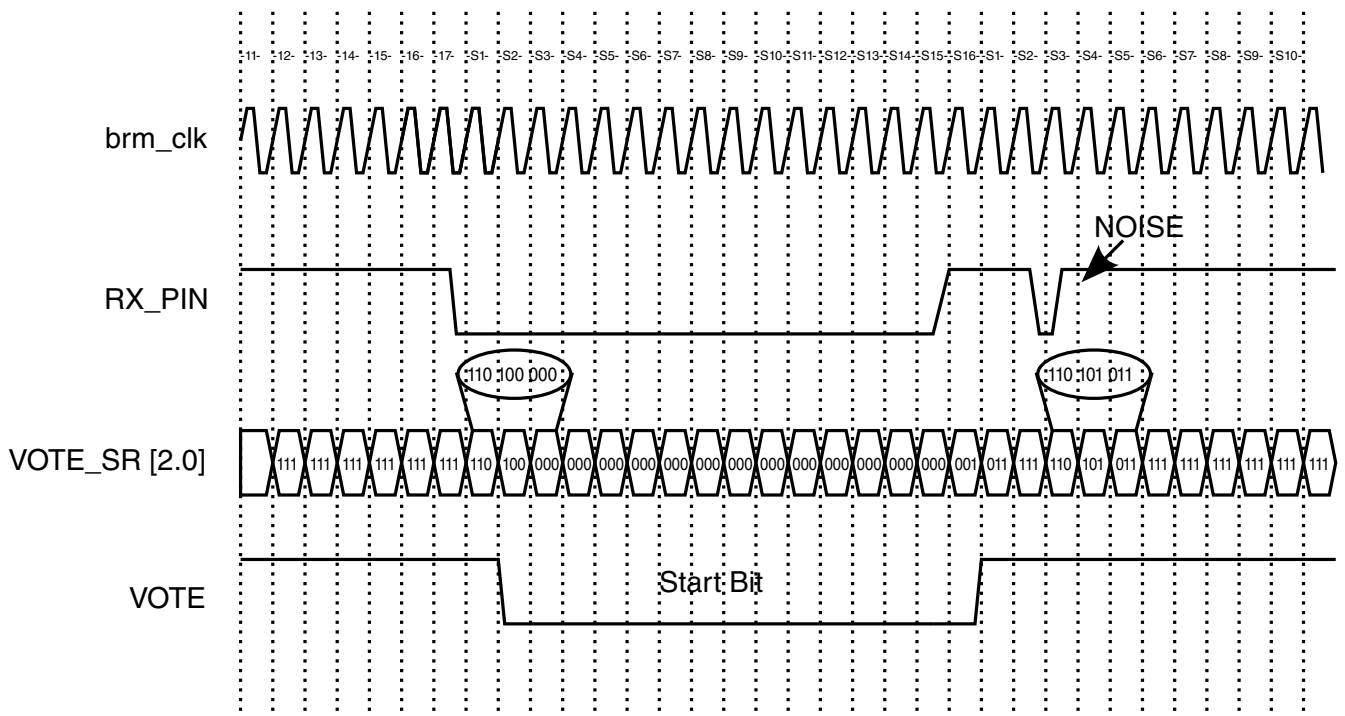


Figure 15-19. Majority Vote Results

A new feature has been recently implemented, it allows to re-synchronize the counter on each edge of RX_DATA line. This is automatic and allows to improve the immunity of UART against signal distortion.

There is a special case when the *brm_clk* frequency is too low and is unable to capture a 0 pulse in IrDA. In this case, the software must set the IRSC (UCR4[5]) bit so that the reference clock (after internal divider) is used for the voting logic. The pulse is validated by counting the length of the pulse.

Refer to Infrared Interface for more details.

15.3.4.6.6 Baud Rate Automatic Detection Logic

When the baud rate automatic detection logic is enabled, the UART locks onto the incoming baud rate. To enable this feature, set the automatic detection of baud rate bit (ADBR = UCR1[14] = 1) and write 1 to the ADET bit (USR2[15]) to clear it.

When ADET=0 and ADBR =1, the detection starts. Then, once the beginning of start bit (transition from 1-to-0 of RX_DATA) has been detected, UART starts a counter (UBRC) working at reference frequency. Once the end of start bit is detected (transition from 0-to-1 of RX_DATA), the value of UBRC - 1 is directly copied into UBMR register. UBIR register is filled with 0x000F.

So, at the end of start bit, registers gets following values:

```
UBRC = number of reference clock periods (after divider) during Start bit.
UBIR = 0x000F
UBMR = UBRC - 1
```

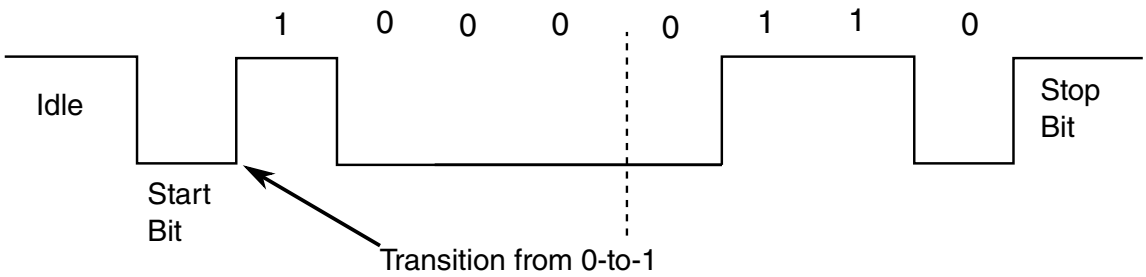
The updated values of the 3 registers can be read.

See Table 15-34 for list of parameters for baud rate detection and Figure 15-20 for baud rate detection protocol diagram.

If any of the UART BRM registers are simultaneously written by the baud rate automatic detection logic and by the peripheral data bus, the peripheral data bus would have lower priority.

Table 15-34. Baud Rate Automatic Detection

ADBR	ADET	Baud Rate Detection	*interrupt_uart*
0	X	Manual Configuration	1
1	0	Auto Detection Started	1
1	1	Auto Detection Complete	0
NOTE: This table assumes that no other interrupt is set at the same time this interrupt is set for the *interrupt_uart* signal.			



Note: LSB Transmitted first.

Figure 15-20. Baud Rate Detection Protocol Diagram

15.3.4.6.6.1 Baud Rate Automatic Detection Protocol

The receiver must receive an ASCII character "A" or "a" to verify proper detection of the incoming baud rate. When an ASCII character "A" (0x41) or "a" (0x61) is received and no error occurs, the Automatic Detect baud rate bit is set (ADET=1) and if the interrupt is enabled (ADEN=UCR1[15]=1), an interrupt *interrupt_uart* is generated.

When an ASCII character "A" or "a" is not received (because of a bit error or the reception of another character), the auto detection sequence restarts and waits for another 1-to-0 transition.

As long as ADET = 0 and ADBR = 1, the UART continues to try to lock onto the incoming baud rate. Once the ASCII character "A" or "a" is detected and the ADET bit is set, the receiver ignores the ADBR bit and continues normal operation with the calculated baud rate.

The UART interrupt is active (*interrupt_uart* = 0) as long as ADET = 1 and ADBR = 1. This can be disabled by clearing the automatic baud rate detection interrupt enable bit (ADEN = 0). Before starting an automatic baud rate detection sequence, set ADET = 0 and ADBR = 1.

The RxFIFO must contain the ASCII character "A" or "a" following the automatic baud rate detection interrupt.

The 16-bit UART Baud Rate Count Register (UBRC) is reset to 4 and stays at 0xFFFF when an overflow occurs. The UBRC register counts (measures) the duration of start bit. When the start bit is detected and counted, the UART Baud Rate Count Register retains its value until the next automatic baud rate detection sequence is initiated.

The Baud Rate Count Register counts only when auto detection is enabled.

15.3.4.6.6.2 New Baud Rate Determination

In order to fight against the problems caused by the distortion and the noise on the RX_DATA line, the duration of the baud rate measurement has been extended.

Previously, as described above, this determination was based on the measurement of the START bit duration. Now, this measurement is based on the duration of START bit + bit0. Bit0 is the first bit following the START bit. In fact, the counter which is started at the falling edge of START bit is no longer stopped at next rising edge (end of START bit), but it is stopped at the next falling edge (end of bit0). As the character sent is always a "A" (41h) or a "a" (61h), this second falling edge will always be present and it will indicate the end of bit0. Once this counter is stopped, the result is divided by 2 and used by the BRM to determine the incoming baud rate.

NOTE

UBRC register contains the result of this division by two, in consequence it reflects the measurement of the duration of one bit.

15.3.4.6.6.2.1 New Autobaud Counter Stopped bit and Interrupt

A new bit has been added in USR2 register: ACST (USR2[11]). This bit is set immediately after the determination of the baud rate.

So,

- if ADNIMP is not set (default), ACST is set to 1 after the end of bit0,
- If ADNIMP is set to 1, ACST is set to 1 at the end of START bit.

If ACIEN (UCR3[0]) is set to 1, ACST will flag an interrupt on *interrupt_uart* signal. This interrupt informs the ARM platform that the BRM has just been set with the result of the bit length measurement. If needed, the ARM platform can perform a read of UBMR (or UBRC) register and determine by itself the baud rate measured. Then the ARM platform has the possibility to correct the BRM registers with the nearest standardized baud rate.

NOTE

ACST is set only if ADBR is set to 1, for example, the UART is autobauding.

Clear the ACST bit by writing 1 to it. Writing 0 to the ACST bit has no effect.

15.3.4.7 Escape Sequence Detection

An escape sequence typically consists of 3 characters entered in rapid succession (such as +++). Because these are valid characters by themselves, the time between characters determines if it is a valid escape sequence.

Too much time between two of the "+" characters is interpreted as two "+" characters, and not part of an escape sequence.

The software chooses the escape character and writes its value to the UART Escape Character Register (UESC). The software must also enable escape detection feature by setting ESCEN (UCR2[11]) to 1. The hardware compares this value to incoming characters in the RxFIFO. When an escape character is detected, the internal escape timer starts to count. The software specifies a time-out value for the maximum allowable time between 2 successive escape characters (see the table below). The escape timer is programmable in intervals of 2 ms to a maximum interval of 8.192 seconds.

Table 15-35. Escape Timer Scaling

UTIM Register	Maximum Time Between Specified Escape Characters
0x000	2 ms
0x001	4 ms
0x002	6 ms
0x003	8 ms
0x004	10 ms
...	...
0F8	498 ms
0F9	500 ms
...	...
9C3	5 s
...	...
FFD	8.188 s
FFE	8.190 s
FFF	8.192 s
NOTE: To calculate the time interval: (UTIM_Value + 1) x 0.002 = Time_Interval Example: (09C3 + 1) x 0.002 = 5 s.	

The escape sequence detection feature is available for all the reference frequencies. Before using Escape Sequence Detection, the user must fill the ONEMS register. This 24-bit register must contain the value of the UART internal frequency divided by 1000. The internal frequency is obtained after the UART internal divider which is applied on *module_clock* clock.

Example I:

- If the input clock *module_clock* frequency is 66.5 MHz.
- And if the input clock *module_clock* is divided by 2 with the internal divider: UFCR[9:7] = 3'b100

$$ONEMS = \frac{66.5 \times 10^6}{2 \times 1000} = 33250 = 81E2h$$

Figure 15-21. Calculation of Frequency for ONEMS Register

Example II:

- If the input clock *module_clock* frequency is 66.5 MHz.
- And if the input clock *module_clock* is divided by 1with the internal divider: UFCR[9:7] = 3'b101

$$ONEMS = \frac{66.5 \times 10^6}{1000} = 66500 = 103C4h$$

Figure 15-22. Calculation of Frequency for ONEMS Register

The escape sequence detection interrupt is asserted when the escape sequence interrupt enable (ESCI) bit is set and an escape sequence is detected (ESCF set). Clear the ESCF bit by writing 1 to it. Writing 0 to the ESCF bit has no effect.

15.3.5 Binary Rate Multiplier (BRM)

The BRM sub-block receives *ref_clk* (*module_clock* clock after divider). From this clock, and with integer and non-integer division, BRM generates a 16x baud rate clock .

The UART transmitte will shift data out based on this 16x baud rate clock. The UART receiver will sample the serial data line based on this 16x baud rate clock. The input and output frequency ratio is programmed in the UART BRM Incremental Register (UBIR) and UART BRM MOD Register (UBMR). The output frequency is divided by the input frequency to produce this ratio. For integer division, set the UBIR = 0x000F and write the divisor to the UBMR register. All values written to these registers must be one less than the actual value to eliminate division by 0 (undefined), and to increase the maximum range of the registers.

Updating the BRM registers requires writing to both registers. The UBIR register must be written before writing to the UBMR register. If only one register is written to by the software, the BRM continues to use the previous values.

The following examples show how to determine what values are to be programmed into UBIR and UBMR for a given reference frequency and desired baud rate. The following equation can be used to help determine these values:

$$\text{BaudRate} = \frac{\text{Ref Freq}}{\left(16 \times \frac{\text{UBMR} + 1}{\text{UBIR} + 1} \right)}$$

Figure 15-23. Frequency and Baud Rate for UBIR and UBMR

With:

Reference Frequency (Hz): UART Reference Frequency (*module_clock* after RFDIV divider)

Baud Rate (bit/s): Desired baud rate.

Integer Division ÷ 21

Reference Frequency = 19.44 MHz

UBIR = 0x000F

UBMR = 0x0014

Baud Rate = 925.7 kbit/s

NOTE

Observe that each value written to the registers is one less than the actual value.

Non-Integer Division

```
Reference Frequency = 16 MHz
Desired Baud Rate = 920 Kbits/s
```

$$\frac{UBMR + 1}{UBIR + 1} = \frac{RefFreq}{16 \times BaudRate} = \frac{16 \times 10^6}{16 \times 920 \times 10^3} = 1.087$$

```
Ratio = 1.087 = 1087 / 1000
UBIR = 999 (decimal)= 0x3E7
UBMR = 1086 (decimal)= 0x43E
Non-Integer Division
Reference Frequency = 25 MHz
Desired Baud Rate = 920 kbit/s
Ratio = 1.69837 = 625 / 368
UBIR = 367 (decimal)= 0x16F
UBMR = 624 (decimal)= 0x270
```

Non-Integer Division

```
Reference Frequency: 30 MHz
Desired Baud Rate = 115.2 kbit/s
Ratio = 16.276043 = 65153 / 4003
UBIR = 4002 (decimal) = 0x0FA2
UBMR = 65152 (decimal) = 0xFE80
```

15.3.6 Infrared Interface

15.3.6.1 Generalities-Infrared

The Infrared interface is selected when IREN (UCR1[7]) is set to 1.

The Infrared Interface is compatible with IrDA Serial Infrared Physical Layer Specification. In this specification, a "zero" is represented by a positive pulse, and a "one" is represented by no pulse (line remains low).

In the UART:

In TX: For each "zero" to be transmitted, a narrow positive pulse which is 3/16 of a bit time is generated. For each "one" to be transmitted no pulse is generated (output is low). External circuitry has to be provided to drive an Infrared LED.

In RX: When receiving, a narrow negative pulse is expected for each "zero" transmitted while no pulse is expected for each "one" transmitted (input is high).

NOTE

Rx part of IR block expects to receive an inverted signal compared to IrDA specification. Circuitry external to the IC transforms the Infrared signal to an electrical signal.

The IR interface has an edge triggered interrupt (IRINT). This interrupt validates a zero bit being received. This interrupt is enabled by writing a "one" to ENIRI bit.

The behavior of Infrared Interface is determined by 3 bits INVT (UCR3[1]), INVR (UCR4[9]) and IRSC (UCR4[5]).

15.3.6.2 Inverted Transmission and Reception bits (INVT & INVR)

The values of INVT and INVR depend of the IrDA transceiver connected on the TXD_IR and RXD_IR pins of the UART. If this transceiver is not inverting on both paths Tx and Rx, a Zero is represented by a positive pulse and a One is represented by no pulse (line remains low). In this case, the bit INVT must be set to 0 and the bit INVR must be set to 1 (because Rx IR block expects an inverted signal).

On the contrary user must set INVT=1 and INVR=0 if both paths of the transceiver are inverting, that is, a Zero is represented as a negative pulse and a One is represented by no pulse (line remains high). The transceiver can also be inverting on only one path (Tx or Rx), in this case INVT and INVR must be together equal to 1 or to 0, depending on which path is inverted.

15.3.6.3 InfraRed Special Case (IRSC) Bit

The value to apply to IRSC bit is based on 2 parameters: the baud rate and the Minimum Pulse Duration (MPD) of the transceiver.

According to IrDA Standard Specification, for SIR (Serial IR) baud rates from 2.4 Kbit/s to 115.2 Kbit/s this nominal pulse duration is equal to 3/16 of a bit duration (at the selected baud rate). But, for all the baud rates a Minimum Pulse Duration is also specified. According to IrDA Standard, a Zero is represented by a light pulse, so the IrDA transceiver can't emit a light pulse shorter than the MPD. For SIR, the MPD is constant and equal to 1.41 µs.

But user must take into account the electrical MPD associated with the transceiver on the receiver path. Typically this value is 2.0 µs, but for some manufacturers MPD can go down to 1.0 µs.

In order to understand the meaning of IRSC bit, one must understand how the RX path works in IrDA mode.

When the UART is in IrDA mode, a Zero is not only detected by the state of the RXD_IR line, but also with the duration of the pulse. This pulse duration can be measured with 2 different clocks. In this case, clock is selected with the IRSC bit.

- If IRSC = 0, the clock used is the BRM clock.
- If IRSC = 1, the clock used is the UART internal clock (UART clock after the divider (RFDIV)).

In normal operation, IRSC=0. This means that at any time, the user must ensure that the frequency of BRM_clock is high enough to measure the pulse. The pulse must last at least 2 BRM clock cycles. If this condition is not fulfilled, IRSC must be set to 1.

Let's examine two examples, for a Minimum Pulse Duration equal to the MPD from the IrDA SIR specification (i.e., 1.41 µs).

1: Calculation of BRM Clock Period (Clock Period < 1.41 µs)

The user wants to receive IrDA data at 115.2 Kbit/s. The UBIR and UBMR registers are set in order to create the BRM_clock with a frequency of 16*baud rate = 16 * 115.2K = 1.843 MHz. But at the same time, in order to correctly detect the pulse, the user must be sure that 2* BRM_clock period is lower than 1.41 µs. Lets check:

BRM_clock period = 1/1843000 = 542 ns

So 2*BRM_clock period = 1.09 µs < 1.41 µs. It is fine.

2: Calculation of BRM Clock Period (Clock Period > 1.41 µs)

This time the user wants to receive at 19.2 Kbit/s. So, the BRM_clock is set to 16*19200 = 307.2 kHz. Let's check if 2* BRM_clock period < 1.41 µs:

1. BRM_clock period =1/307200 = 3.25 µs

   So 2*BRM_clock period = 6.50 µs >> 1.41 µs. It doesn't work.

   So, in this case, the BRM clock can't be used to measure the pulse duration and the user must select the UART internal clock by setting IRSC =1.

NOTE

Like for Escape character detection, when IR Special Case is enabled (IRSC=1), the UART must measure a duration. In order to do that, the user must fill the ONEMS register. Refer to Escape Sequence Detection.

15.3.6.4 IrDA interrupt

Serial infrared mode (SIR) uses an edge triggered interrupt flag IRINT (USR2[8]). When INVR =0, detection of a falling edge on the RXD pin asserts the IRINT bit. When INVR=1, detection of a rising edge on the RXD pin asserts the IRINT bit. When IRINT and ENIRI bits are both asserted, the *interrupt_uart* interrupt is asserted. Clear the IRINT bit by writing 1 to it. Writing 0 to the IRINT bit has no effect.

15.3.6.5 Conclusion about IrDA

Before using the UART in IrDA, the baud rate limit must be calculated. This baud rate limit will inform the user if IRSC bit has to be set or not.

Let's determine this limit:

As already described, if IRSC = 0, the following condition must always be fulfilled

$$2 \times BRMClockPeriod < MinPulseDuration$$

Figure 15-24. Calculation of Baud Rate

So,

$$BRMClockFrequency > \frac{2}{MPD}$$

So, knowing BRM_clock frequency = 16 * Baud Rate, we get:

$$BaudRate > \frac{1}{8 \times MinPulseDuration}$$

So, the user needs to set IRSC = 0 when:

- If Minimum Pulse Duration = 2.5 us and Baud Rate > 50 Kbit/s.
- If Minimum Pulse Duration = 2.0 us and Baud Rate > 62.5 Kbit/s.
- If Minimum Pulse Duration = 1.41 us and Baud Rate > 88.6 Kbit/s.

NOTE

For baud rates lower than the limit, IRSC must be set to1.

15.3.6.6 Programming IrDA Interface

15.3.6.6.1 High Speed

As an example, the following sequence can be used to program the IrDA interface in order to send and receive characters at 115.2 Kbit/s.

Assumptions:

- Input UART clock = 90 MHz
- Internal clock divider = 3 (divide Input UART clock by 3)
- Baud rate = 115.2 Kbit/s
- IrDA transceiver is not inverting on both channels: for Tx and Rx, a Zero is represented by a positive pulse, and a One is represented by no pulse (line stays low).
- Interrupt: Sent to ARM platform when 1 char is received into the Rx FIFO (RDR)

Registers values and Programming orders:

```
UCR1 = 0x0085
UCR1[7] = IREN = 1: Enable IR interface
UCR1[0] = UARTEN = 1: Enable UART
UTS = 0x0000
UFCR = 0x0981
TXTL[5:0] = 0x02: Default value
RFDIV[2:0] = 0x3: Divide Input UART clock by 3 (resulting internal clock  is 30 MHz)
RXTL[5:0] = 0x01: Default value
UBIR = 0x0202
UBMR = 0x20BE Baud rate = 115.2 kbit/s with internal clock = 30 MHz
UCR2 = 0x4027
UCR2[14] = IRTS = 1: Ignore level of RTS input signal
UCR2[5] = WS = 1: Characters are 8-bit length
UCR2[2] = TXEN = 1: Enable Rx path
UCR2 [1] = RXEN = 1: Enable Tx path
UCR2[0] = SRST_B = 1: No software reset
UCR3 = 0x0000

         UCR4 = 0x8201
CTSTL[5:0] = 0x20: Default value
UCR4[9] = INVR = 1: Inverted Infrared Reception (because IrDA transceiver is not inverting)
UCR4[1] = DREN = 1: To enable RDR interrupt (sent when one char is received)
```

The UART is ready to send a character as soon as there is a write into UTXD register. And an interrupt will be sent to ARM platform when a character is received.

15.3.6.6.2 Low Speed

This time, we keep the same assumptions but the speed is now 9.6 Kbit/s. So, this baud rate is below the limit (even with a Min. Pulse Duration of 2.5 us) and thus IRSC must be set to 1.

Assumptions:

- Input UART clock = 90 MHz
- Internal clock divider = 3 (divide Input UART clock by 3)
- Baud rate = 9.6 Kbit/s
- IrDA transceiver is not inverting on both channels: for Tx and Rx, a Zero is represented by a positive pulse, and a One is represented by no pulse (line stays low).
- Interrupt: Sent to ARM platform when 1 char is received into the Rx FIFO (RDR).

Registers values and Programming orders:

```
UCR1 = 0x0085
UCR1[7] = IREN = 1: Enable IR interface
UCR1[0] = UARTEN = 1: Enable UART
UFCR = 0x0981
UFCR[15:10] = TXTL[5:0] = 0x02: Default value
RFDIV[2:0] = 0x3: Divide Input UART clock by 3 (resulting internal clock  is 30 MHz)
UFCR[5:0] = RXTL[5:0] = 0x01: Default value
UBIR = 0x00FF
UBMR = 0xC354 Baud rate = 9.6 kbit/s with internal clock = 30 MHz
UCR2 = 0x4027
UCR2[14] = IRTS = 1: Ignore level of RTS input signal
UCR2[5] = WS = 1: Characters are 8-bit length
UCR2[2] = TXEN = 1: Enable Rx path
UCR2 [1] = RXEN = 1: Enable Tx path
UCR2[0] = SRST_B = 1: No software reset
UCR3 = 0x0000
UCR3[1] = INVT = 0: Positive pulse represents 0.
UCR4 = 0x8221
UCR4[15:10] = CTSTL[5:0] = 0x20: Default value
UCR4[9] = INVR = 1: Inverted Infrared Reception (because IrDA transceiver is not inverting)
UCR4[5] = IRSC = 1: Because data rate is below the limit and thus the UART internal clock is
used to measure the pulse duration.
UCR4[1] = DREN = 1: To enable RDR interrupt (sent when one char is received)
```

The UART is now ready to send a character as soon as there is a write into UTXD register. An interrupt will be sent to ARM platform when a character is received.

15.3.7 9-bit RS-485 Mode

15.3.7.1 Generalities

The UART provides a 9-bit mode to facilitate multidrop (RS-485) network communication. To enable this mode, set MDEN bit in the UMCR register to 1. When 9-bit RS-485 mode is enabled, UART transmitter can transmit the ninth bit (9^{th} bit) set by TXB8, and UART receiver can differentiate between data frames (9^{th} bit = 0) and address frames (9^{th} bit = 1).

The CTS_B pin can be used to control RS-485 output driver outside the chip.

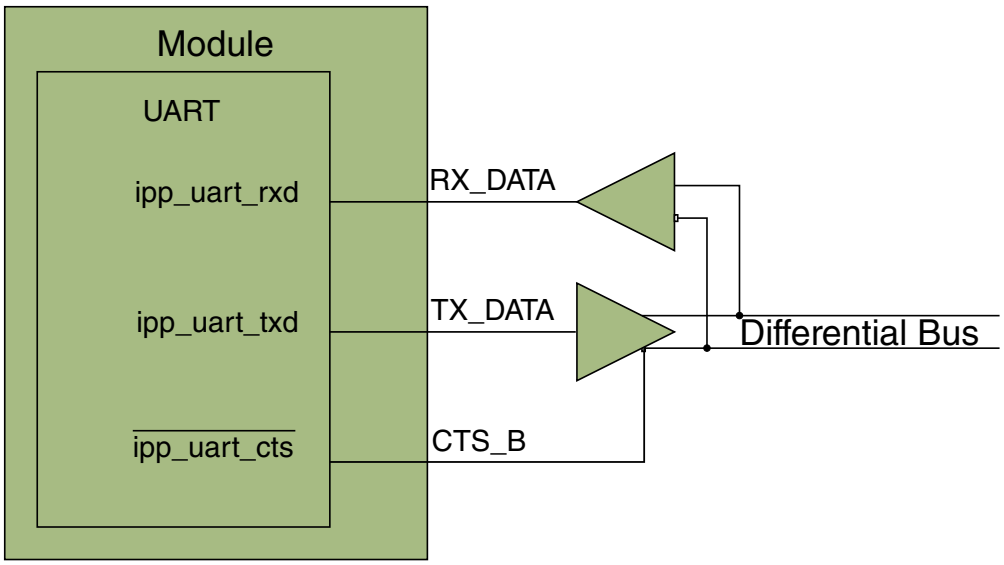


Figure 15-25. RS-485 driver connection (UART in DCE mode)

15.3.7.2 Transmit 9-bit RS-485 frames

To transmit 9-bit RS-485 frames, user need to enable parity (PREN=1) to enable trasmitting the ninth data bit, set 8-bit data word size (WS=1), and write TXB8 (UMCR[2]) as the 9th bit (bit [8]) to be transmitted (write '0' to TXB8 to transmit a data frame, write '1' to transmit a address frame). The other data bit [7:0] is written to TxFIFO by writing to the UTXD same as normal RS-232 operation.

15.3.7.3 Receive 9-bit RS-485 frames

To receive 9-bit RS-485 frames, user need to enable parity (PREN=1) to enable receiving the ninth data bit, set 8-bit data word size (WS=1). The receiver will save the 9-bit data to RxFIFO, and user should read the 9th databit (bit [8]) by reading the PRERR (URXD[10]) bit, and read data bit [7:0] by reading the RX_DATA (URXD[7:0]).

There are two slave address detect modes, normal detect mode and automatic detect mode, and can be selected by SLAM (UMCR[1]).

15.3.7.3.1 RS-485 Slave Address Normal Detect Mode

To enable Normal Detect mode, clear SLAM (UMCR[1] to 0). The receiver ignores all data frames (9th bit = 0) until an address frame is received (9th bit = 1). At that time, the slave address detected (SAD = USR1[3]) bit is asserted and the *interrupt_uart* interrupt is

generated (if SADEN = UMCR[3] = 1). The address byte and sebsequent bytes are all put into RxFIFO along with their 9th bit. The UART will also generate DMA request *dma_req_rx* when the RxFIFO reaches the selected threshold (controlled by RXTL) if receive ready DMA (RXDMAEN = UCR1[8]) request is enabled.

User should read the 9th databit (bit [8]) by reading the PRERR (URXD[10]) bit, and read data bit [7:0] by reading the RX_DATA (URXD[7:0]).

In this mode, once the UART has detected a 9th bit is equal to '1', it will always save the subsequent frames to RxFIFO. So the software must decide whether the address and data in RxFIFO are needed or not.

15.3.7.3.2 RS-485 Slave Address Automatic Detect Mode

To enable Automatic Detect Mode, set SLAM (UMCR[1]) to 1. The receiver tries to detect an address byte (frame 9th bit = 1) that matches the programmed SLADDR (UMCR[15:8]) character. If the received byte is a data or an address byte that does not match the programmed SLADDR character, the receiver will discard these data.

Once the UART receives a matching address byte, it will assert the slave address detected (SAD = USR1[3]) bit and the *interrupt_uart* interrupt will be generated (if SADEN = UMCR[3] = 1). The address byte and sebsequent bytes are all put into RxFIFO along with their 9th bit. If receive ready DMA(RXDMAEN = UCR1[8]) request is enabled, the UART will also generate DMA request *dma_req_rx* when the RxFIFO reaches the selected threshold (controlled by RXTL).

If another address byte is received and this address byte does not match SLADDR character, the receiver will discard the address byte and subsequent data byte. If the address byte again matches SLADDR character, the receiver will put this address byte and subsequent data byte in the RxFIFO along with their 9th bit.

User should read the 9th databit (bit [8]) by reading the PRERR (URXD[10]) bit, and read data bit [7:0] by reading the RX_DATA (URXD[7:0]).

See Initialization for 9-bit RS-485 programming guide.

15.3.8 Low Power Modes

These modes are controlled by the signals *doze_req* and *stop_req*. The control/status/data registers won't change when getting in/out of low power modes.

Table 15-36. UART Low Power State Operation

	Normal State (*doze_req* = 1'b0 & *stop_req* = 1'b0)	Doze State (*doze_req* = 1'b1)		Stop State (*stop_req* = 1'b1)
		DOZE bit = 0	DOZE bit = 1	
UART-Clock	ON	ON	ON	OFF
UART Serial / IrDA	ON	ON	OFF	OFF

15.3.8.1 UART Operation in System Doze Mode

While in Doze State (when *doze_req* input pin is set to 1'b1), the UART behavior depends on the DOZE (UCR1[1]) control bit.

While the DOZE bit is negated, the UART serial interface is enabled. While the system is in the Doze State, and the DOZE bit is asserted, the UART is disabled. If the Doze State is entered with the DOZE bit asserted while the UART serial interface was receiving or transmitting data, it will complete the receive/transmit of the current character and signal to the far-end transmitter/receiver to stop sending/receiving.

15.3.8.2 UART Operation in System Stop Mode

The internal baud rate clocks of the transmitter and receiver are gated off if the *stop_req* signal to UART is asserted. Even though the clocks at the input of the UART continue to run during system Stop mode, the UART will not do any transmission or reception.

The following UART interrupts wake the ARM platform processor from STOP mode:

- RTS (RTSD)
- IrDA Asynchronous WAKE (AIRINT)
- Asynchronous WAKE (AWAKE)
- RI (RIDELT in DTE mode only)
- DCD (DCDDELT in DTE mode only)
- DTR (DTRD in DCE mode only)
- DSR (DTRD in DTE mode only)

When an asynchronous WAKE (awake) interrupt exits the ARM platform from STOP mode, make sure that a dummy character is sent first because the first character may not be received correctly.

15.3.8.3 Power Saving Method in UART

The RXEN (UCR2[1]), TXEN (UCR2[2]) and UARTEN (UCR1[0]) bits are set by the user and provide software control of low-power modes.

Setting the UARTEN (UCR1[0]) bit to 0 shuts off the receiver and transmitter logic and the associated clocks.

If the UART is used only in transmit mode, UARTEN and TXEN must be set to 1. If the UART is used only in receive mode, UARTEN and RXEN must be set to 1. Setting TXEN or RXEN to 0 allows to save a lot of power.

15.3.9 UART Operation in System Debug State

The bit UTS [11] controls whether the UART will respond to the input signal *debug_req*, or whether it will continue to run as normal.

If the UART is programmed to respond to *debug_req*:

1. The UART will halt all operations upon detecting the *debug_req* input.
2. A transfer in progress, either to/from a core (using the IP Bus interface) or to/from an external device, will be completed before halting. This means a single byte/word transfer, not an entire FIFO. Reception of any further data from an external device will be disabled.
3. Internal registers will continue to be writable and readable using the IP Bus interface. A read will leave the contents unaffected.
4. The RX FIFO is affected in debug mode in the following way:
   - All writes into the RX FIFO are prevented.
   - The bit RXDBG (UTS[9]) is used to select the readability of the RX FIFO during debug mode:

     RXDBG = 0: hold the read pointer at the location it had upon entering debug mode, and URXD register returns only the data value at that location, no matter how many reads attempted.

RXDBG = 1, selectable at any time: Allow to read the characters received in Rx FIFO. It will not be possible to re-read previously read locations, nor will it be possible to readjust the read pointer to the value it had prior to entering debug mode.

15.3.10 Reset

This section describes how to reset the block and explains special requirements related to reset.

15.3.10.1 Hardware reset

All of registers, FIFOs, state machines and sequential elements can be reset to their initial values by hardware reset or power on reset.

15.3.10.2 Software reset

The status registers USR1 and USR2, BRM registers UBIR and UBMR, TxFIFO and RxFIFO, and transmitter and receiver state machines can be reset by software reset. Internal logic will keep the software reset asserted for about 4 *module_clock* cycles.

Programmer can follow the following software reset sequence:

1. Clear the SRST_B bit (UCR2[0])
2. Wait for software reset complete: poll SOFTRST bit (UTS[0]) until it is 0.
3. Re-program baud rate registers: Re-write UBIR and UBMR.

15.3.11 Transfer Error

The UART can generate a transfer error on the peripheral bus in the following cases:

- Core is writing into a read-only register.
- Core is accessing (read or write) an unused location within the assigned address space reserved to UART.
- Core is writing into UTXD register with transmit interface disabled (TXEN=0 or UARTEN=0)
- Core is reading URXD register with receive interface disabled (RXEN=0 or UARTEN=0)

15.3.12 Functional Timing

This section includes timing diagrams for functional signaling.

15.3.12.1 IrDA Mode

According to IrDA specification, the low speed (115.2Kbit/s and below) IR frame format is compatible with UART frame.

In this figure, an example data 0x65 is used.

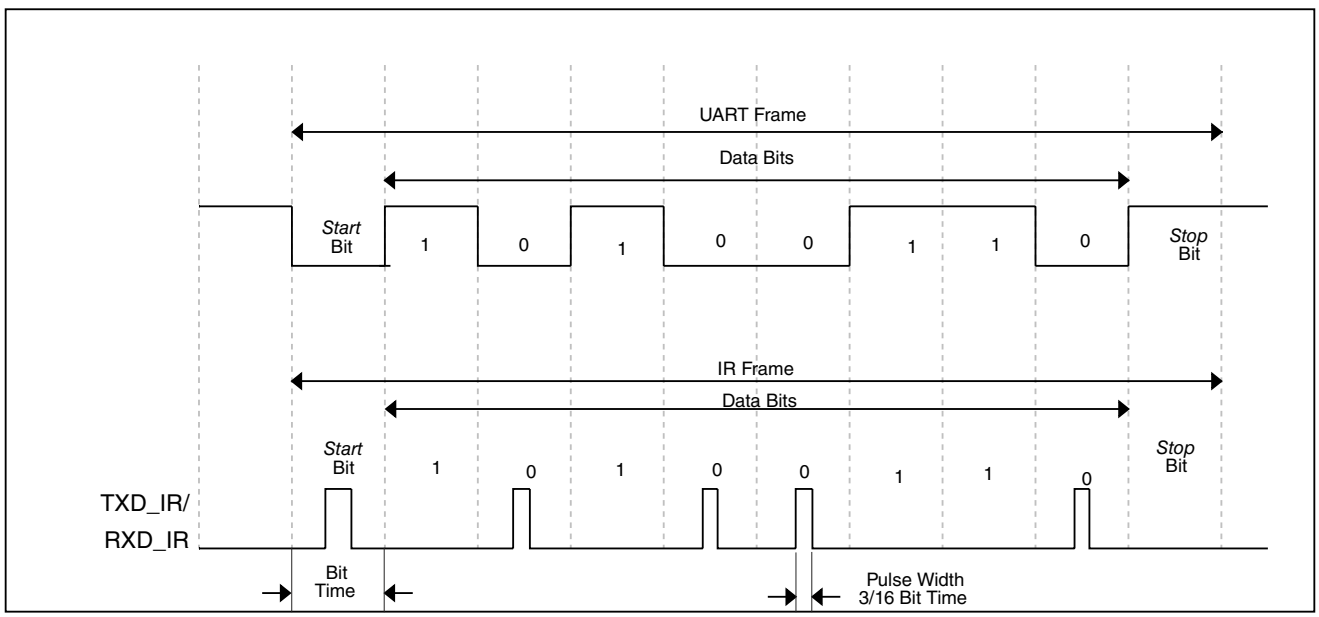


Figure 15-26. Timing diagram of Low Speed IR (<=115.2 Kbit/s) Data Line

15.3.13 Initialization

15.3.13.1 Programming the UART in RS-232 mode

As an example, the following sequence can be used to program the UART in order to send and receive characters in RS-232 mode.

Assumptions:

- Input uart clock = 100 MHz
- Baud rate = 921.6 Kbps

- Data bits = 8 bits
- Parity = Even
- Stop bits = 1 bit
- Flow control = Hardware

Main program:

1. UCR1 = 0x0001

   Enable the UART.

2. UCR2 = 0x2127

   Set hardware flow control, data format and enable transmitter and receiver.

3. UCR3 = 0x0704

   Set UCR3[RXDMUXSEL] = 1.

4. UCR4 = 0x7C00

   Set CTS trigger level to 31,

5. UFCR = 0x089E

   Set internal clock divider = 5 (divide input uart clock by 5). So the reference clock is 100 MHz/5 = 20 MHz.

   Set TXTL = 2 and RXTL = 30.

6. UBIR = 0x08FF
7. UBMR = 0x0C34

   In the above two steps, set baud rate to 921.6Kbps based on the 20MHz reference clock.

8. UCR1 = 0x2201

   Enable the TRDY and RRDY interrupts.

9. UMCR = 0x0000

   UMCR stay at default value 0x0000

Interrupt service routine for the transmitter:

- Write characters into UTXD

The TRDY interrupt will be automatically de-asserted when the data level of the TxFIFO exceeds the TXTL=2. Note: For the first time the interrupt may be de-asserted after 4 characters are written into the TxFIFO because of the shift register.

Interrupt service routine for the receiver:

- Read characters from URXD

The RRDY interrupt will be automatically de-asserted when the data level of the RxFIFO is below the RXTL=30.

15.3.13.2 Programming the UART in 9-bit RS-485 mode

As an example, the following sequence can be used to program the UART in order to send and receive frames in RS-485 mode.

Assumptions:

- Input uart clock = 100 MHz
- Baud rate = 5 Mbps

Main program:

1. UCR1 = 0x0001

   Enable the UART.

2. UCR2 = 0x4127

   Set software flow control ($\overline{\text{CTS}}$ pin is controlled by UCR2[12]), enable parity(enable 9^{th} bit rxd/txd), 8-bit word size , and enable transmitter and receiver.

3. UCR4 = 0x7C00

   Set CTS trigger level to 31,

4. UFCR = 0x0A9E

   Set RFDIV = 5 (divide input uart clock by 1), so the reference clock is 100 MHz. Set UART in DCE mode (RS-485 driver connection outside the chip is the same as Figure 15-25)

   Set TXTL = 2 and RXTL = 30.

5. UBIR = 0x0003
6. UBMR = 0x0004

   In the above two steps, set baud rate to 5 Mbps based on the 100 MHz reference clock.

7. UCR1 = 0x2001 when UART as a master ,

   or UCR1 = 0x0201 (or 0x0101) when UART as a slave.

Enable TRDY interrupt when UART as a master, enable RRDY interrupt or DMA request when UART as a slave.

8. UMCR = 0xA50B

   Enable 9-bit RS-485 mode, enable SAD interrupt, set automatic slave address detect mode, set slave address is 0xA5.

Interrupt service routine for the transmitter:

- Transmit data: write its ninth bit (bit[8]) to UMCR[2], write its bit [7:0] into UTXD[7:0]

The TRDY interrupt will be automatically de-asserted when the data level of the TxFIFO exceeds the TXTL=2.

Note: For the first time the interrupt may be de-asserted after 4 characters are written into the TxFIFO because of the shift register.

Interrupt service routine for the receiver:

- Receive data: read its ninth bit (bit[8]) from URXD[10] , read its bit [7:0] from URXD[7:0].

Note: in RS-485 mode, URXD[10] bit is not the parity error, instead it holds the ninth bit (bit[8]) of the received data.

The SAD interrupt can not de-assert automatically, it needs MCU write 1 to USR1[3] to clear it . The RRDY interrupt or DMA request will be automatically de-asserted when the data level of the RxFIFO is below the RXTL=30.

15.3.14 References

- EIA/TIA-232-F Interface Standard

  http://www.eia.org, http://www.tiaonline.org/standards

- IrDA Standard

  http://www.irda.org

15.3.15 UART Memory Map/Register Definition

UART supports 8-bit, 16-bit and 32-bit accesses to 32-bit memory-mapped addresses. Any access to unmapped memory location will yield a transfer error.

All registers except the ONEMS described in this section are 16-bit registers. The ONEMS register is a 24-bit register.

- For 32-bit write accesses, the upper two bytes will not be taken into account.
- For 32-bit read accesses the upper two bytes will return 0.

The ONEMS register is expanded from 16 bits to 24 bits in order to support the high frequency of the BRM internal clock *ref_clk* (*module_clock* after divider). The ONEMS register can be accessed as 8 bits, 16 bits or 32 bits.

- For 32-bit write accesses, the most significant byte of the ONEMS will be discarded.
- For 32-bit read accesses, the most significant byte of the ONEMS will be read as 0.

UART memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3086_0000	UART Receiver Register (UART1_URXD)	32	R	0000_0000h	15.3.15.1/ 4327
3086_0040	UART Transmitter Register (UART1_UTXD)	32	W	0000_0000h	15.3.15.2/ 4329
3086_0080	UART Control Register 1 (UART1_UCR1)	32	R/W	0000_0000h	15.3.15.3/ 4330
3086_0084	UART Control Register 2 (UART1_UCR2)	32	R/W	0000_0001h	15.3.15.4/ 4332
3086_0088	UART Control Register 3 (UART1_UCR3)	32	R/W	0000_0700h	15.3.15.5/ 4335
3086_008C	UART Control Register 4 (UART1_UCR4)	32	R/W	0000_8000h	15.3.15.6/ 4337
3086_0090	UART FIFO Control Register (UART1_UFCR)	32	R/W	0000_0801h	15.3.15.7/ 4339
3086_0094	UART Status Register 1 (UART1_USR1)	32	R/W	0000_2040h	15.3.15.8/ 4341
3086_0098	UART Status Register 2 (UART1_USR2)	32	R/W	0000_4028h	15.3.15.9/ 4344
3086_009C	UART Escape Character Register (UART1_UESC)	32	R/W	0000_002Bh	15.3.15.10/ 4346
3086_00A0	UART Escape Timer Register (UART1_UTIM)	32	R/W	0000_0000h	15.3.15.11/ 4346
3086_00A4	UART BRM Incremental Register (UART1_UBIR)	32	R/W	0000_0000h	15.3.15.12/ 4347
3086_00A8	UART BRM Modulator Register (UART1_UBMR)	32	R/W	0000_0000h	15.3.15.13/ 4347
3086_00AC	UART Baud Rate Count Register (UART1_UBRC)	32	R	0000_0004h	15.3.15.14/ 4348

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3086_00B0	UART One Millisecond Register (UART1_ONEMS)	32	R/W	0000_0000h	15.3.15.15/ 4349
3086_00B4	UART Test Register (UART1_UTS)	32	R/W	0000_0060h	15.3.15.16/ 4350
3086_00B8	UART RS-485 Mode Control Register (UART1_UMCR)	32	R/W	0000_0000h	15.3.15.17/ 4351
3088_0000	UART Receiver Register (UART3_URXD)	32	R	0000_0000h	15.3.15.1/ 4327
3088_0040	UART Transmitter Register (UART3_UTXD)	32	W	0000_0000h	15.3.15.2/ 4329
3088_0080	UART Control Register 1 (UART3_UCR1)	32	R/W	0000_0000h	15.3.15.3/ 4330
3088_0084	UART Control Register 2 (UART3_UCR2)	32	R/W	0000_0001h	15.3.15.4/ 4332
3088_0088	UART Control Register 3 (UART3_UCR3)	32	R/W	0000_0700h	15.3.15.5/ 4335
3088_008C	UART Control Register 4 (UART3_UCR4)	32	R/W	0000_8000h	15.3.15.6/ 4337
3088_0090	UART FIFO Control Register (UART3_UFCR)	32	R/W	0000_0801h	15.3.15.7/ 4339
3088_0094	UART Status Register 1 (UART3_USR1)	32	R/W	0000_2040h	15.3.15.8/ 4341
3088_0098	UART Status Register 2 (UART3_USR2)	32	R/W	0000_4028h	15.3.15.9/ 4344
3088_009C	UART Escape Character Register (UART3_UESC)	32	R/W	0000_002Bh	15.3.15.10/ 4346
3088_00A0	UART Escape Timer Register (UART3_UTIM)	32	R/W	0000_0000h	15.3.15.11/ 4346
3088_00A4	UART BRM Incremental Register (UART3_UBIR)	32	R/W	0000_0000h	15.3.15.12/ 4347
3088_00A8	UART BRM Modulator Register (UART3_UBMR)	32	R/W	0000_0000h	15.3.15.13/ 4347
3088_00AC	UART Baud Rate Count Register (UART3_UBRC)	32	R	0000_0004h	15.3.15.14/ 4348
3088_00B0	UART One Millisecond Register (UART3_ONEMS)	32	R/W	0000_0000h	15.3.15.15/ 4349
3088_00B4	UART Test Register (UART3_UTS)	32	R/W	0000_0060h	15.3.15.16/ 4350
3088_00B8	UART RS-485 Mode Control Register (UART3_UMCR)	32	R/W	0000_0000h	15.3.15.17/ 4351
3089_0000	UART Receiver Register (UART2_URXD)	32	R	0000_0000h	15.3.15.1/ 4327
3089_0040	UART Transmitter Register (UART2_UTXD)	32	W	0000_0000h	15.3.15.2/ 4329

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3089_0080	UART Control Register 1 (UART2_UCR1)	32	R/W	0000_0000h	15.3.15.3/ 4330
3089_0084	UART Control Register 2 (UART2_UCR2)	32	R/W	0000_0001h	15.3.15.4/ 4332
3089_0088	UART Control Register 3 (UART2_UCR3)	32	R/W	0000_0700h	15.3.15.5/ 4335
3089_008C	UART Control Register 4 (UART2_UCR4)	32	R/W	0000_8000h	15.3.15.6/ 4337
3089_0090	UART FIFO Control Register (UART2_UFCR)	32	R/W	0000_0801h	15.3.15.7/ 4339
3089_0094	UART Status Register 1 (UART2_USR1)	32	R/W	0000_2040h	15.3.15.8/ 4341
3089_0098	UART Status Register 2 (UART2_USR2)	32	R/W	0000_4028h	15.3.15.9/ 4344
3089_009C	UART Escape Character Register (UART2_UESC)	32	R/W	0000_002Bh	15.3.15.10/ 4346
3089_00A0	UART Escape Timer Register (UART2_UTIM)	32	R/W	0000_0000h	15.3.15.11/ 4346
3089_00A4	UART BRM Incremental Register (UART2_UBIR)	32	R/W	0000_0000h	15.3.15.12/ 4347
3089_00A8	UART BRM Modulator Register (UART2_UBMR)	32	R/W	0000_0000h	15.3.15.13/ 4347
3089_00AC	UART Baud Rate Count Register (UART2_UBRC)	32	R	0000_0004h	15.3.15.14/ 4348
3089_00B0	UART One Millisecond Register (UART2_ONEMS)	32	R/W	0000_0000h	15.3.15.15/ 4349
3089_00B4	UART Test Register (UART2_UTS)	32	R/W	0000_0060h	15.3.15.16/ 4350
3089_00B8	UART RS-485 Mode Control Register (UART2_UMCR)	32	R/W	0000_0000h	15.3.15.17/ 4351
30A6_0000	UART Receiver Register (UART4_URXD)	32	R	0000_0000h	15.3.15.1/ 4327
30A6_0040	UART Transmitter Register (UART4_UTXD)	32	W	0000_0000h	15.3.15.2/ 4329
30A6_0080	UART Control Register 1 (UART4_UCR1)	32	R/W	0000_0000h	15.3.15.3/ 4330
30A6_0084	UART Control Register 2 (UART4_UCR2)	32	R/W	0000_0001h	15.3.15.4/ 4332
30A6_0088	UART Control Register 3 (UART4_UCR3)	32	R/W	0000_0700h	15.3.15.5/ 4335
30A6_008C	UART Control Register 4 (UART4_UCR4)	32	R/W	0000_8000h	15.3.15.6/ 4337
30A6_0090	UART FIFO Control Register (UART4_UFCR)	32	R/W	0000_0801h	15.3.15.7/ 4339

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30A6_0094	UART Status Register 1 (UART4_USR1)	32	R/W	0000_2040h	15.3.15.8/ 4341
30A6_0098	UART Status Register 2 (UART4_USR2)	32	R/W	0000_4028h	15.3.15.9/ 4344
30A6_009C	UART Escape Character Register (UART4_UESC)	32	R/W	0000_002Bh	15.3.15.10/ 4346
30A6_00A0	UART Escape Timer Register (UART4_UTIM)	32	R/W	0000_0000h	15.3.15.11/ 4346
30A6_00A4	UART BRM Incremental Register (UART4_UBIR)	32	R/W	0000_0000h	15.3.15.12/ 4347
30A6_00A8	UART BRM Modulator Register (UART4_UBMR)	32	R/W	0000_0000h	15.3.15.13/ 4347
30A6_00AC	UART Baud Rate Count Register (UART4_UBRC)	32	R	0000_0004h	15.3.15.14/ 4348
30A6_00B0	UART One Millisecond Register (UART4_ONEMS)	32	R/W	0000_0000h	15.3.15.15/ 4349
30A6_00B4	UART Test Register (UART4_UTS)	32	R/W	0000_0060h	15.3.15.16/ 4350
30A6_00B8	UART RS-485 Mode Control Register (UART4_UMCR)	32	R/W	0000_0000h	15.3.15.17/ 4351
30A7_0000	UART Receiver Register (UART5_URXD)	32	R	0000_0000h	15.3.15.1/ 4327
30A7_0040	UART Transmitter Register (UART5_UTXD)	32	W	0000_0000h	15.3.15.2/ 4329
30A7_0080	UART Control Register 1 (UART5_UCR1)	32	R/W	0000_0000h	15.3.15.3/ 4330
30A7_0084	UART Control Register 2 (UART5_UCR2)	32	R/W	0000_0001h	15.3.15.4/ 4332
30A7_0088	UART Control Register 3 (UART5_UCR3)	32	R/W	0000_0700h	15.3.15.5/ 4335
30A7_008C	UART Control Register 4 (UART5_UCR4)	32	R/W	0000_8000h	15.3.15.6/ 4337
30A7_0090	UART FIFO Control Register (UART5_UFCR)	32	R/W	0000_0801h	15.3.15.7/ 4339
30A7_0094	UART Status Register 1 (UART5_USR1)	32	R/W	0000_2040h	15.3.15.8/ 4341
30A7_0098	UART Status Register 2 (UART5_USR2)	32	R/W	0000_4028h	15.3.15.9/ 4344
30A7_009C	UART Escape Character Register (UART5_UESC)	32	R/W	0000_002Bh	15.3.15.10/ 4346
30A7_00A0	UART Escape Timer Register (UART5_UTIM)	32	R/W	0000_0000h	15.3.15.11/ 4346
30A7_00A4	UART BRM Incremental Register (UART5_UBIR)	32	R/W	0000_0000h	15.3.15.12/ 4347

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30A7_00A8	UART BRM Modulator Register (UART5_UBMR)	32	R/W	0000_0000h	15.3.15.13/ 4347
30A7_00AC	UART Baud Rate Count Register (UART5_UBRC)	32	R	0000_0004h	15.3.15.14/ 4348
30A7_00B0	UART One Millisecond Register (UART5_ONEMS)	32	R/W	0000_0000h	15.3.15.15/ 4349
30A7_00B4	UART Test Register (UART5_UTS)	32	R/W	0000_0060h	15.3.15.16/ 4350
30A7_00B8	UART RS-485 Mode Control Register (UART5_UMCR)	32	R/W	0000_0000h	15.3.15.17/ 4351
30A8_0000	UART Receiver Register (UART6_URXD)	32	R	0000_0000h	15.3.15.1/ 4327
30A8_0040	UART Transmitter Register (UART6_UTXD)	32	W	0000_0000h	15.3.15.2/ 4329
30A8_0080	UART Control Register 1 (UART6_UCR1)	32	R/W	0000_0000h	15.3.15.3/ 4330
30A8_0084	UART Control Register 2 (UART6_UCR2)	32	R/W	0000_0001h	15.3.15.4/ 4332
30A8_0088	UART Control Register 3 (UART6_UCR3)	32	R/W	0000_0700h	15.3.15.5/ 4335
30A8_008C	UART Control Register 4 (UART6_UCR4)	32	R/W	0000_8000h	15.3.15.6/ 4337
30A8_0090	UART FIFO Control Register (UART6_UFCR)	32	R/W	0000_0801h	15.3.15.7/ 4339
30A8_0094	UART Status Register 1 (UART6_USR1)	32	R/W	0000_2040h	15.3.15.8/ 4341
30A8_0098	UART Status Register 2 (UART6_USR2)	32	R/W	0000_4028h	15.3.15.9/ 4344
30A8_009C	UART Escape Character Register (UART6_UESC)	32	R/W	0000_002Bh	15.3.15.10/ 4346
30A8_00A0	UART Escape Timer Register (UART6_UTIM)	32	R/W	0000_0000h	15.3.15.11/ 4346
30A8_00A4	UART BRM Incremental Register (UART6_UBIR)	32	R/W	0000_0000h	15.3.15.12/ 4347
30A8_00A8	UART BRM Modulator Register (UART6_UBMR)	32	R/W	0000_0000h	15.3.15.13/ 4347
30A8_00AC	UART Baud Rate Count Register (UART6_UBRC)	32	R	0000_0004h	15.3.15.14/ 4348
30A8_00B0	UART One Millisecond Register (UART6_ONEMS)	32	R/W	0000_0000h	15.3.15.15/ 4349
30A8_00B4	UART Test Register (UART6_UTS)	32	R/W	0000_0060h	15.3.15.16/ 4350
30A8_00B8	UART RS-485 Mode Control Register (UART6_UMCR)	32	R/W	0000_0000h	15.3.15.17/ 4351

UART memory map (continued)

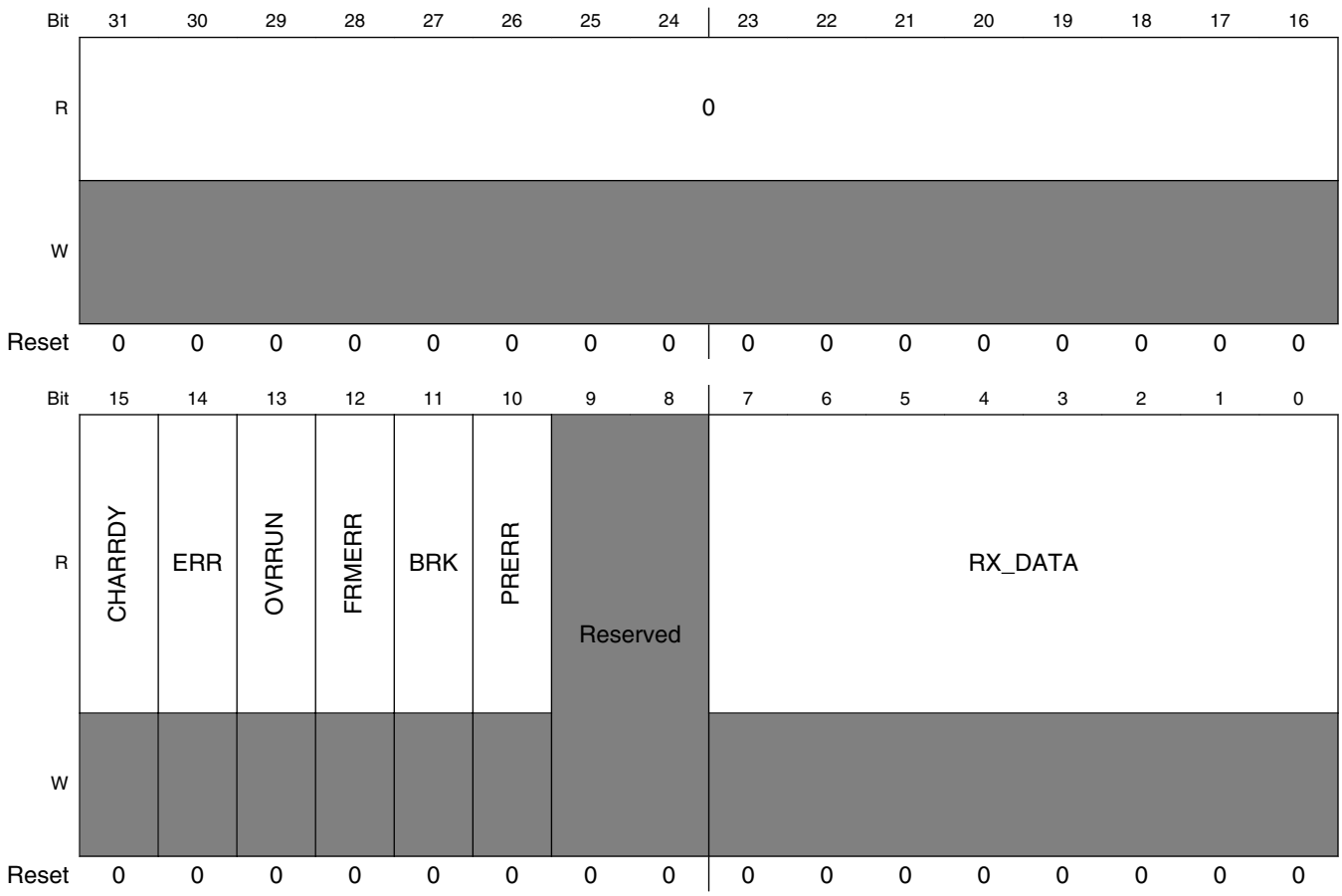
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30A9_0000	UART Receiver Register (UART7_URXD)	32	R	0000_0000h	15.3.15.1/ 4327
30A9_0040	UART Transmitter Register (UART7_UTXD)	32	W	0000_0000h	15.3.15.2/ 4329
30A9_0080	UART Control Register 1 (UART7_UCR1)	32	R/W	0000_0000h	15.3.15.3/ 4330
30A9_0084	UART Control Register 2 (UART7_UCR2)	32	R/W	0000_0001h	15.3.15.4/ 4332
30A9_0088	UART Control Register 3 (UART7_UCR3)	32	R/W	0000_0700h	15.3.15.5/ 4335
30A9_008C	UART Control Register 4 (UART7_UCR4)	32	R/W	0000_8000h	15.3.15.6/ 4337
30A9_0090	UART FIFO Control Register (UART7_UFCR)	32	R/W	0000_0801h	15.3.15.7/ 4339
30A9_0094	UART Status Register 1 (UART7_USR1)	32	R/W	0000_2040h	15.3.15.8/ 4341
30A9_0098	UART Status Register 2 (UART7_USR2)	32	R/W	0000_4028h	15.3.15.9/ 4344
30A9_009C	UART Escape Character Register (UART7_UESC)	32	R/W	0000_002Bh	15.3.15.10/ 4346
30A9_00A0	UART Escape Timer Register (UART7_UTIM)	32	R/W	0000_0000h	15.3.15.11/ 4346
30A9_00A4	UART BRM Incremental Register (UART7_UBIR)	32	R/W	0000_0000h	15.3.15.12/ 4347
30A9_00A8	UART BRM Modulator Register (UART7_UBMR)	32	R/W	0000_0000h	15.3.15.13/ 4347
30A9_00AC	UART Baud Rate Count Register (UART7_UBRC)	32	R	0000_0004h	15.3.15.14/ 4348
30A9_00B0	UART One Millisecond Register (UART7_ONEMS)	32	R/W	0000_0000h	15.3.15.15/ 4349
30A9_00B4	UART Test Register (UART7_UTS)	32	R/W	0000_0060h	15.3.15.16/ 4350
30A9_00B8	UART RS-485 Mode Control Register (UART7_UMCR)	32	R/W	0000_0000h	15.3.15.17/ 4351

15.3.15.1 UART Receiver Register (UARTx_URXD)

NOTE

The UART will yield a transfer error on the peripheral bus when core is reading URXD register with receive interface disabled (RXEN=0 or UARTEN=0).

Address: Base address + 0h offset



UARTx_URXD field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 CHARRDY	Character Ready. This read-only bit indicates an invalid read when the FIFO becomes empty and software tries to read the same old data. This bit should not be used for polling for data written to the RX FIFO. 0 Character in RX_DATA field and associated flags are invalid. 1 Character in RX_DATA field and associated flags valid and ready for reading.

Table continues on the next page...

UARTx_URXD field descriptions (continued)

Field	Description
14 ERR	**Error Detect.** Indicates whether the character present in the RX_DATA field has an error (OVRRUN, FRMERR, BRK or PRERR) status. The ERR bit is updated and valid for each received character. 0 No error status was detected 1 An error status was detected
13 OVRRUN	**Receiver Overrun.** This read-only bit, when HIGH, indicates that the corresponding character was stored in the last position (32nd) of the Rx FIFO. Even if a 33rd character has not been detected, this bit will be set to '1' for the 32nd character. 0 No RxFIFO overrun was detected 1 A RxFIFO overrun was detected
12 FRMERR	**Frame Error.** Indicates whether the current character had a framing error (a missing stop bit) and is possibly corrupted. FRMERR is updated for each character read from the RxFIFO. 0 The current character has no framing error 1 The current character has a framing error
11 BRK	**BREAK Detect.** Indicates whether the current character was detected as a BREAK character. The data bits and the stop bit are all 0. The FRMERR bit is set when BRK is set. When odd parity is selected, PRERR is also set when BRK is set. BRK is valid for each character read from the RxFIFO. 0 The current character is not a BREAK character 1 The current character is a BREAK character
10 PRERR	**In RS-485 mode, it holds the ninth data bit (bit [8]) of received 9-bit RS-485 data** **In RS232/IrDA mode, it is the Parity Error flag**. Indicates whether the current character was detected with a parity error and is possibly corrupted. PRERR is updated for each character read from the RxFIFO. When parity is disabled, PRERR always reads as 0. 0 = No parity error was detected for data in the RX_DATA field 1 = A parity error was detected for data in the RX_DATA field
9–8 -	This field is reserved. **Reserved**
RX_DATA	**Received Data**. Holds the received character. In 7-bit mode, the most significant bit (MSB) is forced to 0. In 8-bit mode, all bits are active.

15.3.15.2 UART Transmitter Register (UARTx_UTXD)

NOTE

The UART will yield a transfer error on the peripheral bus when core is writing into UART_URXD register with transmit interface disabled (TXEN=0 or UARTEN=0).

Memory space between UART_URXD and UART_UTXD registers is reserved. Any read or write access to this space will be considered as an invalid access and yield a transfer error.

Address: Base address + 40h offset

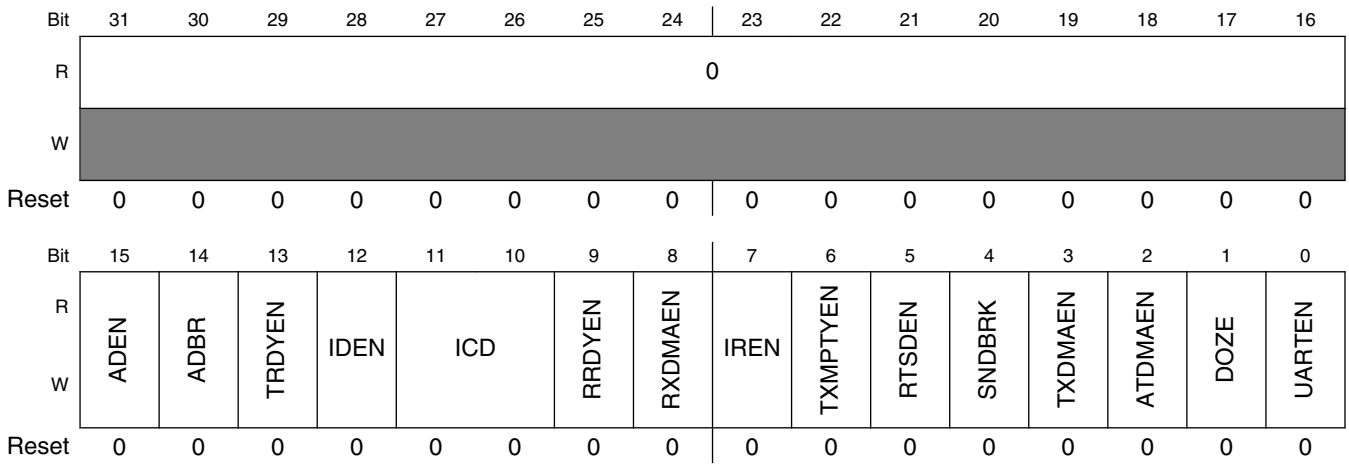
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																0								TX_DATA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UARTx_UTXD field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–8 Reserved	This read-only field is reserved and always has the value 0.
TX_DATA	**Transmit Data**. Holds the parallel transmit data inputs. In 7-bit mode, D7 is ignored. In 8-bit mode, all bits are used. Data is transmitted least significant bit (LSB) first. A new character is transmitted when the TX_DATA field is written. The TX_DATA field must be written only when the TRDY bit is high to ensure that corrupted data is not sent.

15.3.15.3 UART Control Register 1 (UARTx_UCR1)

Address: Base address + 80h offset



UARTx_UCR1 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 ADEN	**Automatic Baud Rate Detection Interrupt Enable**. Enables/Disables the automatic baud rate detect complete (ADET) bit to generate an interrupt (*interrupt_uart* = 0). 0 Disable the automatic baud rate detection interrupt 1 Enable the automatic baud rate detection interrupt
14 ADBR	**Automatic Detection of Baud Rate**. Enables/Disables automatic baud rate detection. When the ADBR bit is set and the ADET bit is cleared, the receiver detects the incoming baud rate automatically. The ADET flag is set when the receiver verifies that the incoming baud rate is detected properly by detecting an ASCII character "A" or "a" (0x41 or 0x61). 0 Disable automatic detection of baud rate 1 Enable automatic detection of baud rate
13 TRDYEN	**Transmitter Ready Interrupt Enable**. Enables/Disables the transmitter Ready Interrupt (TRDY) when the transmitter has one or more slots available in the TxFIFO. The fill level in the TXFIFO at which an interrupt is generated is controlled by TxTL bits. When TRDYEN is negated, the transmitter ready interrupt is disabled. **NOTE:** An interrupt will be issued as long as TRDYEN and TRDY are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TRDY interrupt. 0 Disable the transmitter ready interrupt 1 Enable the transmitter ready interrupt
12 IDEN	**Idle Condition Detected Interrupt Enable**. Enables/Disables the IDLE bit to generate an interrupt (*interrupt_uart* = 0). 0 Disable the IDLE interrupt 1 Enable the IDLE interrupt

Table continues on the next page...

UARTx_UCR1 field descriptions (continued)

Field	Description
11–10 ICD	**Idle Condition Detect**. Controls the number of frames RXD is allowed to be idle before an idle condition is reported. 00 Idle for more than 4 frames 01 Idle for more than 8 frames 10 Idle for more than 16 frames 11 Idle for more than 32 frames
9 RRDYEN	**Receiver Ready Interrupt Enable**. Enables/Disables the RRDY interrupt when the RxFIFO contains data. The fill level in the RxFIFO at which an interrupt is generated is controlled by the RXTL bits. When RRDYEN is negated, the receiver ready interrupt is disabled. 0 Disables the RRDY interrupt 1 Enables the RRDY interrupt
8 RXDMAEN	**Receive Ready DMA Enable**. Enables/Disables the receive DMA request *dma_req_rx* when the receiver has data in the RxFIFO. The fill level in the RxFIFO at which a DMA request is generated is controlled by the RXTL bits. When negated, the receive DMA request is disabled. 0 Disable DMA request 1 Enable DMA request
7 IREN	**Infrared Interface Enable**. Enables/Disables the IR interface. See the IR interface description in Infrared Interface, for more information. Note: MDEN(UMCR[0]) must be cleared to 0 when using IrDA interface. See Table 15-27 0 Disable the IR interface 1 Enable the IR interface
6 TXMPTYEN	**Transmitter Empty Interrupt Enable**. Enables/Disables the transmitter FIFO empty (TXFE) interrupt. *interrupt_uart*. When negated, the TXFE interrupt is disabled. **NOTE:** An interrupt will be issued as long as TXMPTYEN and TXFE are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TXFE interrupt. 0 Disable the transmitter FIFO empty interrupt 1 Enable the transmitter FIFO empty interrupt
5 RTSDEN	**RTS Delta Interrupt Enable**. Enables/Disables the RTSD interrupt. The current status of the RTS_B pin is read in the RTSS bit. 0 Disable RTSD interrupt 1 Enable RTSD interrupt
4 SNDBRK	**Send BREAK**. Forces the transmitter to send a BREAK character. The transmitter finishes sending the character in progress (if any) and sends BREAK characters until SNDBRK is reset. Because the transmitter samples SNDBRK after every bit is transmitted, it is important that SNDBRK is asserted high for a sufficient period of time to generate a valid BREAK. After the BREAK transmission completes, the UART transmits 2 mark bits. The user can continue to fill the TxFIFO and any characters remaining are transmitted when the BREAK is terminated. 0 Do not send a BREAK character 1 Send a BREAK character (continuous 0s)
3 TXDMAEN	**Transmitter Ready DMA Enable**. Enables/Disables the transmit DMA request *dma_req_tx* when the transmitter has one or more slots available in the TxFIFO. The fill level in the TxFIFO that generates the *dma_req_tx* is controlled by the TXTL bits.

Table continues on the next page...

UARTx_UCR1 field descriptions (continued)

Field	Description
	NOTE: A DMA request will be issued as long as TXDMAEN and TRDY are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the transmit DMA request. 0 Disable transmit DMA request 1 Enable transmit DMA request
2 ATDMAEN	**Aging DMA Timer Enable**. Enables/Disables the receive DMA request *dma_req_rx* for the aging timer interrupt (triggered with AGTIM flag in USR1[8]). 0 Disable AGTIM DMA request 1 Enable AGTIM DMA request
1 DOZE	**DOZE**. Determines the UART enable condition in the DOZE state. When *doze_req* input pin is at '1', (the ARM Platform executes a doze instruction and the system is placed in the Doze State), the DOZE bit affects operation of the UART. While in the Doze State, if this bit is asserted, the UART is disabled. See the description in Low Power Modes. 0 The UART is enabled when in DOZE state 1 The UART is disabled when in DOZE state
0 UARTEN	**UART Enable**. Enables/Disables the UART. If UARTEN is negated in the middle of a transmission, the transmitter stops and pulls the TXD line to a logic 1. UARTEN must be set to 1 before any access to UTXD and URXD registers, otherwise a transfer error is returned. This bit can be set to 1 along with other bits in this register. There is no restriction to the sequence of programing this bit and other control registers. 0 Disable the UART 1 Enable the UART

15.3.15.4 UART Control Register 2 (UARTx_UCR2)

Address: Base address + 84h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ESCI	IRTS	CTSC	CTS	ESCEN	RTEC		PREN	PROE	STPB	WS	RTSEN	ATEN	TXEN	RXEN	SRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

UARTx_UCR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 ESCI	**Escape Sequence Interrupt Enable**. Enables/Disables the ESCF bit to generate an interrupt. 0 Disable the escape sequence interrupt 1 Enable the escape sequence interrupt
14 IRTS	**Ignore RTS Pin**. Forces the RTS input signal presented to the transmitter to always be asserted (set to low), effectively ignoring the external pin. When in this mode, the RTS pin serves as a general purpose input. 0 Transmit only when the RTS pin is asserted 1 Ignore the RTS pin
13 CTSC	CTS **Pin Control**. Controls the operation of the CTS_B module output. When CTSC is asserted, the CTS_B module output is controlled by the receiver. When the RxFIFO is filled to the level of the programmed trigger level and the start bit of the overflowing character (TRIGGER LEVEL + 1) is validated, the CTS_B module output is negated to indicate to the far-end transmitter to stop transmitting. When the trigger level is programmed for less than 32, the receiver continues to receive data until the RxFIFO is full. When the CTSC bit is negated, the CTS_B module output is controlled by the CTS bit. On reset, because CTSC is cleared to 0, the CTS_B pin is controlled by the CTS bit, which again is cleared to 0 on reset. This means that on reset the CTS_B signal is negated. 0 The CTS_B pin is controlled by the CTS bit 1 The CTS_B pin is controlled by the receiver
12 CTS	**Clear to Send**. Controls the CTS_B pin when the CTSC bit is negated. CTS has no function when CTSC is asserted. 0 The CTS_B pin is high (inactive) 1 The CTS_B pin is low (active)
11 ESCEN	**Escape Enable**. Enables/Disables the escape sequence detection logic. 0 Disable escape sequence detection 1 Enable escape sequence detection
10–9 RTEC	**Request to Send Edge Control**. Selects the edge that triggers the RTS interrupt. This has no effect on the RTS delta interrupt. RTEC has an effect only when RTSEN = 1 (see Table 15-31). 00 Trigger interrupt on a rising edge 01 Trigger interrupt on a falling edge 1X Trigger interrupt on any edge
8 PREN	**Parity Enable**. Enables/Disables the parity generator in the transmitter and parity checker in the receiver. When PREN is asserted, the parity generator and checker are enabled, and disabled when PREN is negated. 0 Disable parity generator and checker 1 Enable parity generator and checker
7 PROE	**Parity Odd/Even**. Controls the sense of the parity generator and checker. When PROE is high, odd parity is generated and expected. When PROE is low, even parity is generated and expected. PROE has no function if PREN is low. 0 Even parity 1 Odd parity

Table continues on the next page...

UARTx_UCR2 field descriptions (continued)

Field	Description
6 STPB	**Stop**. Controls the number of stop bits after a character. When STPB is low, 1 stop bit is sent. When STPB is high, 2 stop bits are sent. STPB also affects the receiver. 0 The transmitter sends 1 stop bit. The receiver expects 1 or more stop bits. 1 The transmitter sends 2 stop bits. The receiver expects 2 or more stop bits.
5 WS	**Word Size**. Controls the character length. When WS is high, the transmitter and receiver are in 8-bit mode. When WS is low, they are in 7-bit mode. The transmitter ignores bit 7 and the receiver sets bit 7 to 0. WS can be changed in-between transmission (reception) of characters, however not when a transmission (reception) is in progress, in which case the length of the current character being transmitted (received) is unpredictable. 0 7-bit transmit and receive character length (not including START, STOP or PARITY bits) 1 8-bit transmit and receive character length (not including START, STOP or PARITY bits)
4 RTSEN	**Request to Send Interrupt Enable.** Controls the RTS edge sensitive interrupt. When RTSEN is asserted and the programmed edge is detected on the RTS_B pin (the RTSF bit is asserted), an interrupt will be generated on the *interrupt_uart* pin. (See Table 15-31.) 0 Disable request to send interrupt 1 Enable request to send interrupt
3 ATEN	**Aging Timer Enable**. This bit is used to enable the aging timer interrupt (triggered with AGTIM) 0 AGTIM interrupt disabled 1 AGTIM interrupt enabled
2 TXEN	**Transmitter Enable**. Enables/Disables the transmitter. When TXEN is negated the transmitter is disabled and idle. When the UARTEN and TXEN bits are set the transmitter is enabled. If TXEN is negated in the middle of a transmission, the UART disables the transmitter immediately, and starts marking 1s. The transmitter FIFO cannot be written when this bit is cleared. 0 Disable the transmitter 1 Enable the transmitter
1 RXEN	**Receiver Enable**. Enables/Disables the receiver. When the receiver is enabled, if the RXD input is already low, the receiver does not recognize BREAK characters, because it requires a valid 1-to-0 transition before it can accept any character. 0 Disable the receiver 1 Enable the receiver
0 SRST	**Software Reset**. Once the software writes 0 to SRST_B, the software reset remains active for 4 *module_clock* cycles before the hardware deasserts SRST_B. The software can only write 0 to SRST_B. Writing 1 to SRST_B is ignored. 0 Reset the transmit and receive state machines, all FIFOs and register USR1, USR2, UBIR, UBMR, UBRC , URXD, UTXD and UTS[6-3]. 1 No reset

15.3.15.5 UART Control Register 3 (UARTx_UCR3)

Address: Base address + 88h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DPEC		DTREN	PARERREN	FRAERREN	DSR	DCD	RI	ADNIMP	RXDSEN	AIRINTEN	AWAKEN	DTRDEN	RXDMUXSEL	INVT	ACIEN
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

UARTx_UCR3 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–14 DPEC	This bit is not used in this chip.
13 DTREN	This bit is not used in this chip.
12 PARERREN	**Parity Error Interrupt Enable. Enables/Disables the interrupt. When asserted, PARERREN causes the PARITYERR bit to generate an interrupt.** 0 Disable the parity error interrupt 1 Enable the parity error interrupt
11 FRAERREN	**Frame Error Interrupt Enable. Enables/Disables the interrupt. When asserted, FRAERREN causes the FRAMERR bit to generate an interrupt.** 0 Disable the frame error interrupt 1 Enable the frame error interrupt
10 DSR	This bit is not used in this chip.
9 DCD	This bit is not used in this chip.
8 RI	This bit is not used in this chip.
7 ADNIMP	**Autobaud Detection Not Improved-. Disables new features of autobaud detection (See Baud Rate Automatic Detection Protocol, for more details).** 0 Autobaud detection new features selected 1 Keep old autobaud detection mechanism

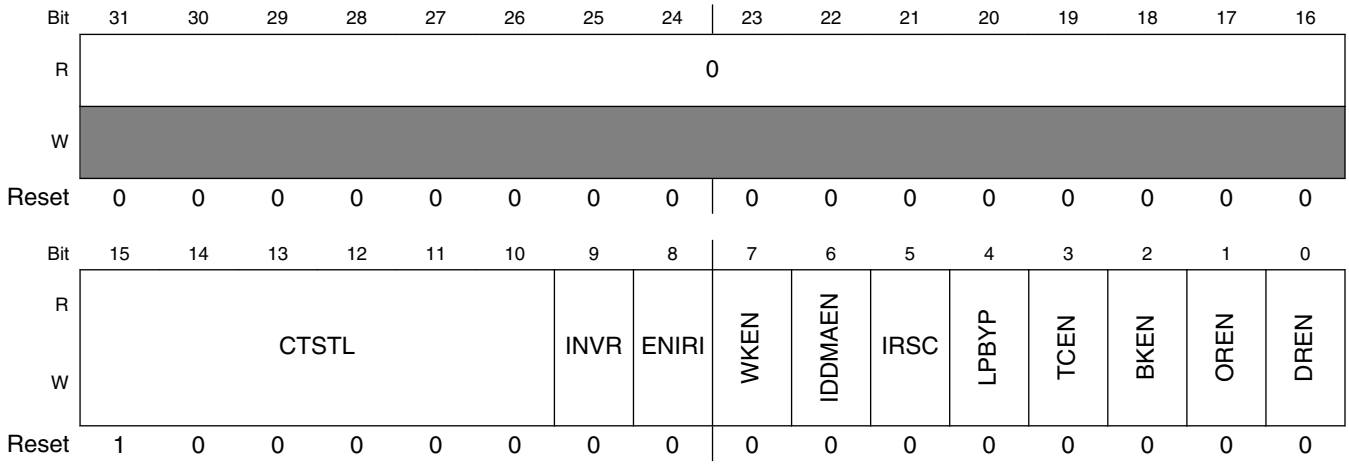
Table continues on the next page...

UARTx_UCR3 field descriptions (continued)

Field	Description
6 RXDSEN	Receive Status Interrupt Enable. Controls the receive status interrupt (*interrupt_uart*). When this bit is enabled and RXDS status bit is set, the interrupt *interrupt_uart* will be generated. 0 Disable the RXDS interrupt 1 Enable the RXDS interrupt
5 AIRINTEN	Asynchronous IR WAKE Interrupt Enable. Controls the asynchronous IR WAKE interrupt. An interrupt is generated when AIRINTEN is asserted and a pulse is detected on the RXD pin. 0 Disable the AIRINT interrupt 1 Enable the AIRINT interrupt
4 AWAKEN	Asynchronous WAKE Interrupt Enable. Controls the asynchronous WAKE interrupt. An interrupt is generated when AWAKEN is asserted and a falling edge is detected on the RXD pin. 0 Disable the AWAKE interrupt 1 Enable the AWAKE interrupt
3 DTRDEN	This bit is not used in this chip.
2 RXDMUXSEL	RXD Muxed Input Selected. Selects proper input pins for serial and Infrared input signal. **NOTE:** In this chip, UARTs are used in MUXED mode, so that this bit should always be set.
1 INVT	Invert TXD output in RS-232/RS-485 mode, set TXD active level in IrDA mode. In RS232/RS-485 mode(UMCR[0] = 1), if this bit is set to 1, the TXD output is inverted before transmitted. In **IrDA mode**, when INVT is cleared, the infrared logic block transmits a positive IR 3/16 pulse for all 0s and 0s are transmitted for 1s. When INVT is set (INVT = 1), the infrared logic block transmits an active low or negative infrared 3/16 pulse for all 0s and 1s are transmitted for 1s. 0 **TXD is not inverted** 1 **TXD is inverted** 0 **TXD** Active low transmission 1 **TXD** Active high transmission
0 ACIEN	**Autobaud Counter Interrupt Enable.** This bit is used to enable the autobaud counter stopped interrupt (triggered with ACST (USR2[11]). 0 ACST interrupt disabled 1 ACST interrupt enabled

15.3.15.6 UART Control Register 4 (UARTx_UCR4)

Address: Base address + 8Ch offset



UARTx_UCR4 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–10 CTSTL	**CTS Trigger Level**. Controls the threshold at which the CTS_B pin is deasserted by the RxFIFO. After the trigger level is reached and the CTS_B pin is deasserted, the RxFIFO continues to receive data until it is full. The CTSTL bits are encoded as shown in the Settings column. Settings 0 to 32 are in use. All other settings are Reserved. 000000 0 characters received 000001 1 characters in the RxFIFO ... — ... — 100000 32 characters in the RxFIFO (maximum)
9 INVR	**Invert RXD input in RS-232/RS-485 Mode, d**etermine **RXD input** logic level being sampled in **In IrDA mode.** **In RS232/RS-485 Mode(UMCR[0] = 1), if this bit is set to 1, the RXD input is inverted before sampled.** **In IrDA mode**,when cleared, the infrared logic block expects an active low or negative IR 3/16 pulse for 0s and 1s are expected for 1s. When INVR is set (INVR 1), the infrared logic block expects an active high or positive IR 3/16 pulse for 0s and 0s are expected for 1s. 0 **RXD input is not inverted** 1 **RXD input is inverted** 0 **RXD** active low detection 1 **RXD** active high detection
8 ENIRI	**Serial Infrared Interrupt Enable**. Enables/Disables the serial infrared interrupt. 0 Serial infrared Interrupt disabled 1 Serial infrared Interrupt enabled

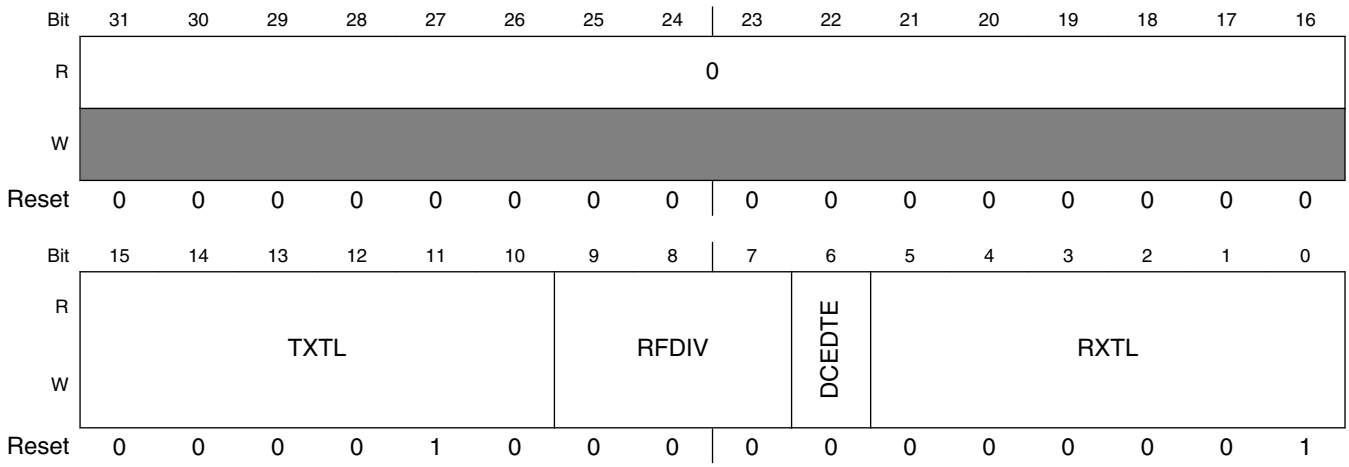
Table continues on the next page...

UARTx_UCR4 field descriptions (continued)

Field	Description
7 WKEN	**WAKE Interrupt Enable**. Enables/Disables the WAKE bit to generate an interrupt. The WAKE bit is set at the detection of a start bit by the receiver. 0 Disable the WAKE interrupt 1 Enable the WAKE interrupt
6 IDDMAEN	**DMA IDLE Condition Detected Interrupt Enable** Enables/Disables the receive DMA request *dma_req_rx* for the IDLE interrupt (triggered with IDLE flag in USR2[12]). 0 DMA IDLE interrupt disabled 1 DMA IDLE interrupt enabled
5 IRSC	**IR Special Case**. Selects the clock for the vote logic. When set, IRSC switches the vote logic clock from the sampling clock to the UART reference clock. The IR pulses are counted a predetermined amount of time depending on the reference frequency. See InfraRed Special Case (IRSC) Bit. 0 The vote logic uses the sampling clock (16x baud rate) for normal operation 1 The vote logic uses the UART reference clock
4 LPBYP	**Low Power Bypass**. Allows to bypass the low power new features in UART. To use during debug phase. 0 Low power features enabled 1 Low power features disabled
3 TCEN	**TransmitComplete Interrupt Enable**. Enables/Disables the TXDC bit to generate an interrupt (*interrupt_uart* = 0) **NOTE:** An interrupt will be issued as long as TCEN and TXDC are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TXDC interrupt. 0 Disable TXDC interrupt 1 Enable TXDC interrupt
2 BKEN	**BREAK Condition Detected Interrupt Enable**. Enables/Disables the BRCD bit to generate an interrupt. 0 Disable the BRCD interrupt 1 Enable the BRCD interrupt
1 OREN	**Receiver Overrun Interrupt Enable**. Enables/Disables the ORE bit to generate an interrupt. 0 Disable ORE interrupt 1 Enable ORE interrupt
0 DREN	**Receive Data Ready Interrupt Enable**. Enables/Disables the RDR bit to generate an interrupt. 0 Disable RDR interrupt 1 Enable RDR interrupt

15.3.15.7 UART FIFO Control Register (UARTx_UFCR)

Address: Base address + 90h offset



UARTx_UFCR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–10 TXTL	**Transmitter Trigger Level**. Controls the threshold at which a maskable interrupt is generated by the TxFIFO. A maskable interrupt is generated whenever the data level in the TxFIFO falls below the selected threshold. The bits are encoded as shown in the Settings column. Settings 0 to 32 are in use. All other settings are Reserved. 000000 Reserved 000001 Reserved 000010 TxFIFO has 2 or fewer characters ... — ... — 011111 TxFIFO has 31 or fewer characters 100000 TxFIFO has 32 characters (maximum)
9–7 RFDIV	Reference Frequency Divider. Controls the divide ratio for the reference clock. The input clock is *module_clock*. The output from the divider is *ref_clk* which is used by BRM to create the 16x baud rate oversampling clock (*brm_clk*). 000 Divide input clock by 6 001 Divide input clock by 5 010 Divide input clock by 4 011 Divide input clock by 3 100 Divide input clock by 2 101 Divide input clock by 1 110 Divide input clock by 7 111 Reserved
6 DCEDTE	**DCE/DTE mode select**. Select UART as data communication equipment (DCE mode) or as data terminal equipment (DTE mode).

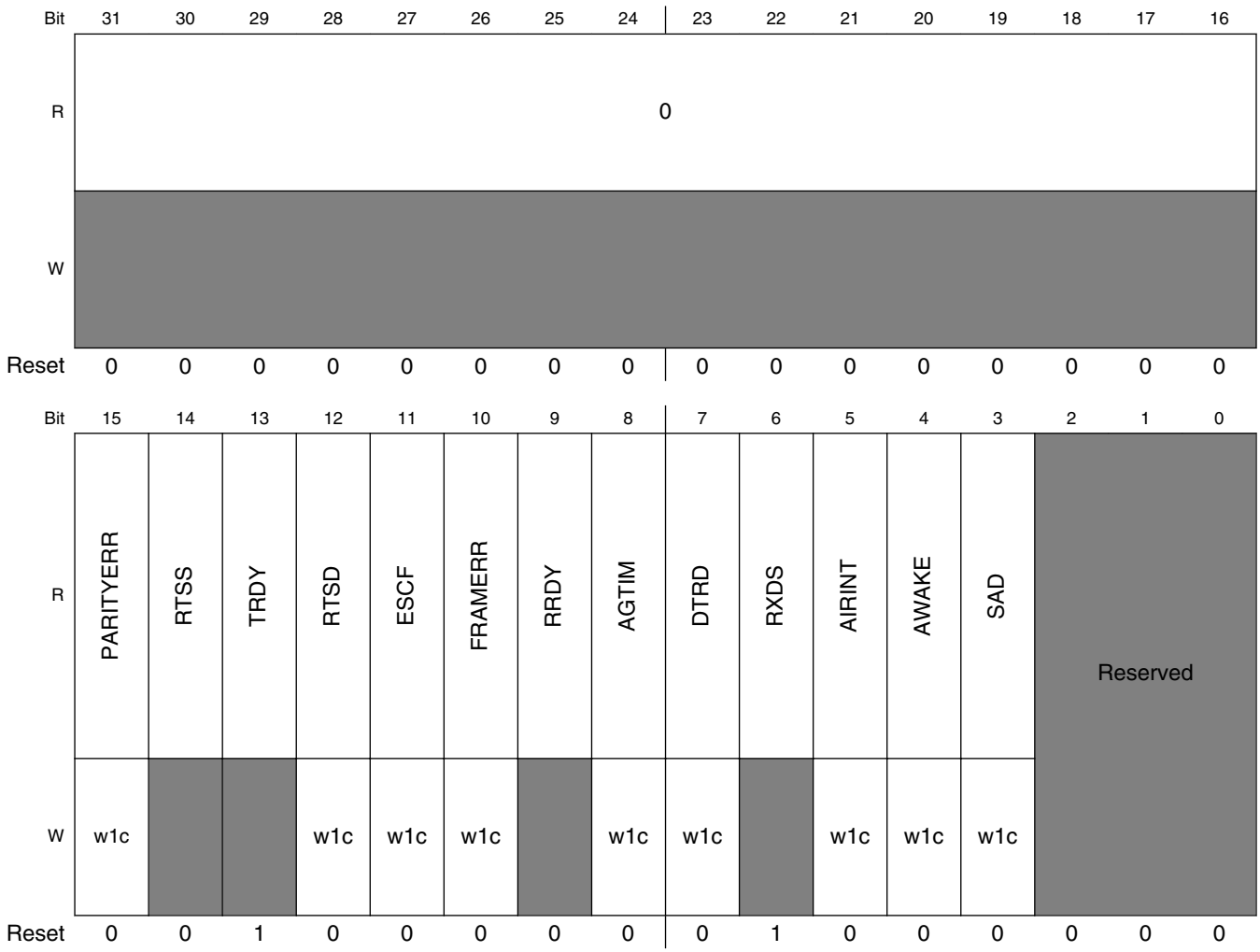
Table continues on the next page...

UARTx_UFCR field descriptions (continued)

Field	Description
	0 DCE mode selected 1 DTE mode selected
RXTL	**Receiver Trigger Level**. Controls the threshold at which a maskable interrupt is generated by the RxFIFO. A maskable interrupt is generated whenever the data level in the RxFIFO reaches the selected threshold. The RXTL bits are encoded as shown in the Settings column. Setting 0 to 32 are in use. All other settings are Reserved. 000000 0 characters received 000001 RxFIFO has 1 character ... — ... — 011111 RxFIFO has 31 characters 100000 RxFIFO has 32 characters (maximum)

15.3.15.8 UART Status Register 1 (UARTx_USR1)

Address: Base address + 94h offset



UARTx_USR1 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 PARITYERR	**Parity Error Interrupt Flag**. Indicates a parity error is detected. PARITYERR is cleared by writing 1 to it. Writing 0 to PARITYERR has no effect. When parity is disabled, PARITYERR always reads 0. At reset, PARITYERR is set to 0. 0 No parity error detected 1 Parity error detected (write 1 to clear)
14 RTSS	RTS_B **Pin Status**. Indicates the current status of the RTS_B pin. A "snapshot" of RTS_B is taken immediately before RTSS is presented to the data bus. RTSS cannot be cleared because all writes to RTSS are ignored. At reset, RTSS is set to 0.

Table continues on the next page...

UARTx_USR1 field descriptions (continued)

Field	Description
	0 The RTS_B module input is high (inactive) 1 The RTS_B module input is low (active)
13 TRDY	**Transmitter Ready Interrupt / DMA Flag**. Indicates that the TxFIFO emptied below its target threshold and requires data. TRDY is automatically cleared when the data level in the TxFIFO exceeds the threshold set by TXTL bits. At reset, TRDY is set to 1. 0 The transmitter does not require data 1 The transmitter requires data (interrupt posted)
12 RTSD	**RTS Delta.** Indicates whether the RTS_B pin changed state. It (RTSD) generates a maskable interrupt. When in STOP mode, RTS assertion sets RTSD and can be used to wake the processor. The current state of the RTS_B pin is available on the RTSS bit. Clear RTSD by writing 1 to it. Writing 0 to RTSD has no effect. At reset, RTSD is set to 0. 0 RTS_B pin did not change state since last cleared 1 RTS_B pin changed state (write 1 to clear)
11 ESCF	**Escape Sequence Interrupt Flag**. Indicates if an escape sequence was detected. ESCF is asserted when the ESCEN bit is set and an escape sequence is detected in the RxFIFO. Clear ESCF by writing 1 to it. Writing 0 to ESCF has no effect. 0 No escape sequence detected 1 Escape sequence detected (write 1 to clear).
10 FRAMERR	**Frame Error Interrupt Flag**. Indicates that a frame error is detected. The *interrupt_uart* interrupt will be generated if a frame error is detected and the interrupt is enabled. Clear FRAMERR by writing 1 to it. Writing 0 to FRAMERR has no effect. 0 No frame error detected 1 Frame error detected (write 1 to clear)
9 RRDY	**Receiver Ready Interrupt / DMA Flag**. Indicates that the RxFIFO data level is above the threshold set by the RXTL bits. (See the RXTL bits description in UART FIFO Control Register (UART_UFCR) for setting the interrupt threshold.) When asserted, RRDY generates a maskable interrupt or DMA request. RRDY is automatically cleared when data level in the RxFIFO goes below the set threshold level. At reset, RRDY is set to 0. 0 No character ready 1 Character(s) ready (interrupt posted)
8 AGTIM	Ageing Timer Interrupt Flag. Indicates that data in the RxFIFO has been idle for a time of 8 character lengths (where a character length consists of 7 or 8 bits, depending on the setting of the WS bit in UCR2, with the bit time corresponding to the baud rate setting) and FIFO data level is less than RxFIFO threshold level (RXTL in the UFCR). Clear by writing a 1 to it. 0 AGTIM is not active 1 AGTIM is active (write 1 to clear)
7 DTRD	This bit is not used in this chip.
6 RXDS	Receiver IDLE Interrupt Flag. Indicates that the receiver state machine is in an IDLE state, the next state is IDLE, and the receive pin is high. RXDS is automatically cleared when a character is received. RXDS is active only when the receiver is enabled. 0 Receive in progress 1 Receiver is IDLE

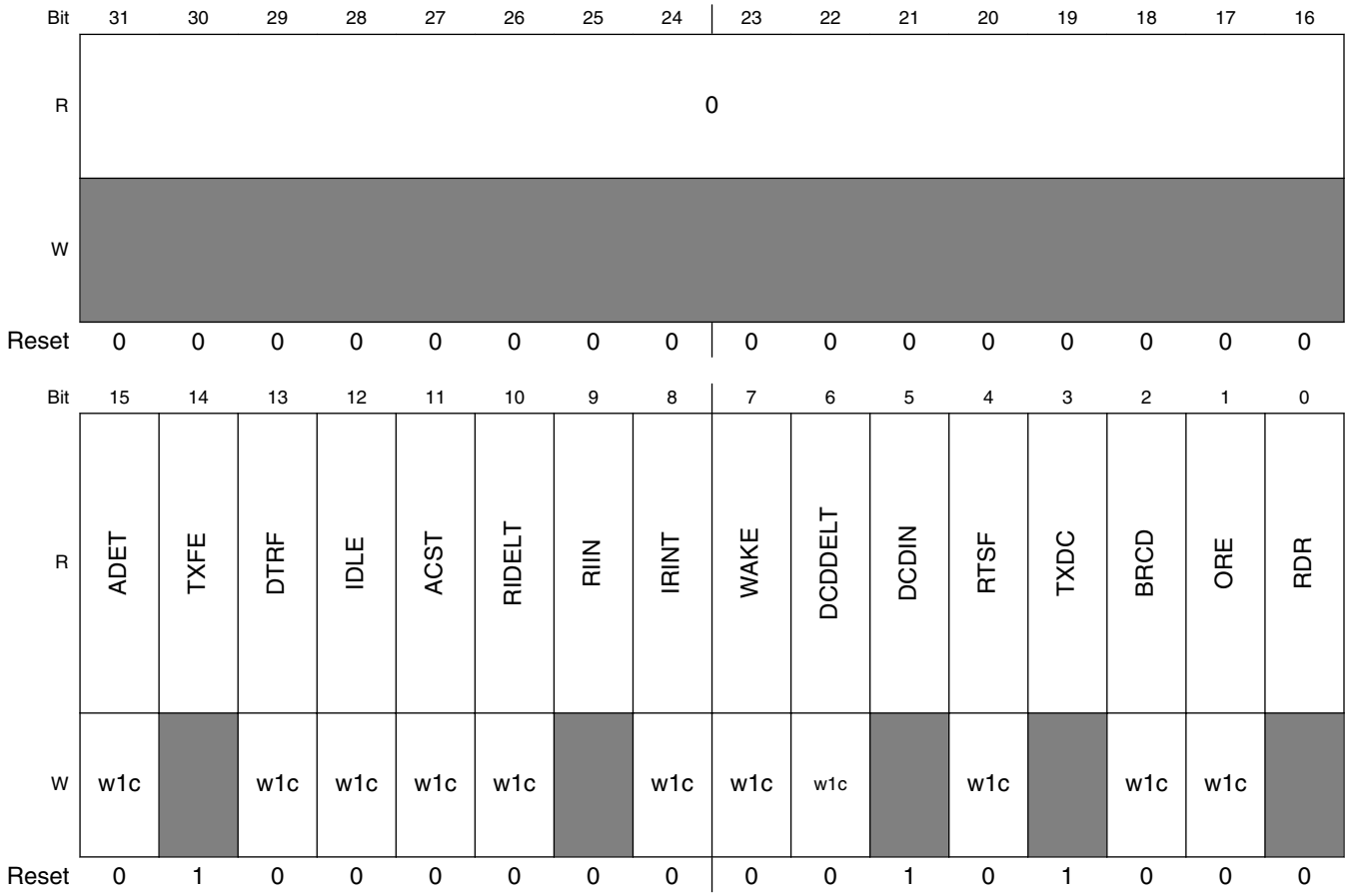
Table continues on the next page...

UARTx_USR1 field descriptions (continued)

Field	Description
5 AIRINT	Asynchronous IR WAKE Interrupt Flag. Indicates that the IR WAKE pulse was detected on the RXD pin. Clear AIRINT by writing 1 to it. Writing 0 to AIRINT has no effect. 0 No pulse was detected on the RXD IrDA pin 1 A pulse was detected on the RXD IrDA pin
4 AWAKE	Asynchronous WAKE Interrupt Flag. Indicates that a falling edge was detected on the RXD pin. Clear AWAKE by writing 1 to it. Writing 0 to AWAKE has no effect. 0 No falling edge was detected on the RXD Serial pin 1 A falling edge was detected on the RXD Serial pin
3 SAD	RS-485 Slave Address Detected Interrupt Flag. Indicates if RS-485 Slave Address was detected . SAD was asserted in RS-485 mode when the SADEN bit is set and Slave Address is detected in RxFIFO (in Nomal Address Detect Mode, the 9th data bit = 1; in Automatic Address Detect Mode, the received charater matches the programmed SLADDR). 0 No slave address detected 1 Slave address detected
-	This field is reserved. Reserved

15.3.15.9 UART Status Register 2 (UARTx_USR2)

Address: Base address + 98h offset



UARTx_USR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 ADET	**Automatic Baud Rate Detect Complete**. Indicates that an "A" or "a" was received and that the receiver detected and verified the incoming baud rate. Clear ADET by writing 1 to it. Writing 0 to ADET has no effect. 0 ASCII "A" or "a" was not received 1 ASCII "A" or "a" was received (write 1 to clear)
14 TXFE	**Transmit Buffer FIFO Empty**. Indicates that the transmit buffer (TxFIFO) is empty. TXFE is cleared automatically when data is written to the TxFIFO. Even though TXFE is high, the transmission might still be in progress. 0 The transmit buffer (TxFIFO) is not empty 1 The transmit buffer (TxFIFO) is empty
13 DTRF	This bit is not used in this chip.

Table continues on the next page...

UARTx_USR2 field descriptions (continued)

Field	Description
12 IDLE	**Idle Condition**. Indicates that an idle condition has existed for more than a programmed amount frame (see Idle Line Detect. An interrupt can be generated by this IDLE bit if IDEN (UCR1[12]) is enabled. IDLE is cleared by writing 1 to it. Writing 0 to IDLE has no effect. 0 No idle condition detected 1 Idle condition detected (write 1 to clear)
11 ACST	**Autobaud Counter Stopped**. In autobaud detection (ADBR=1), indicates the counter which determines the baud rate was running and is now stopped. This means either START bit is finished (if ADNIMP=1), or Bit 0 is finished (if ADNIMP=0). See New Autobaud Counter Stopped bit and Interrupt, for more details. An interrupt can be flagged on *interrupt_uart* if ACIEN=1. 0 Measurement of bit length not finished (in autobaud) 1 Measurement of bit length finished (in autobaud). (write 1 to clear)
10 RIDELT	This bit is not used in this chip.
9 RIIN	This bit is not used in this chip.
8 IRINT	**Serial Infrared Interrupt Flag.** When an edge is detected on the RXD pin during SIR Mode, this flag will be asserted. This flag can cause an interrupt which can be masked using the control bit ENIRI: UCR4 [8]. 0 no edge detected 1 valid edge detected (write 1 to clear)
7 WAKE	**Wake**. Indicates the start bit is detected. WAKE can generate an interrupt that can be masked using the WKEN bit. Clear WAKE by writing 1 to it. Writing 0 to WAKE has no effect. 0 start bit not detected 1 start bit detected (write 1 to clear)
6 DCDDELT	This bit is not used in this chip.
5 DCDIN	This bit is not used in this chip.
4 RTSF	**RTS Edge Triggered Interrupt Flag. Indicates if** a programmed edge is detected on the RTS_B pin. The RTEC bits select the edge that generates an interrupt (see Table 15-31). RTSF can generate an interrupt that can be masked using the RTSEN bit. Clear RTSF by writing 1 to it. Writing 0 to RTSF has no effect. 0 Programmed edge not detected on RTS_B 1 Programmed edge detected on RTS_B (write 1 to clear)
3 TXDC	**Transmitter Complete**. Indicates that the transmit buffer (TxFIFO) and Shift Register is empty; therefore the transmission is complete. TXDC is cleared automatically when data is written to the TxFIFO. 0 Transmit is incomplete 1 Transmit is complete
2 BRCD	**BREAK Condition Detected**. Indicates that a BREAK condition was detected by the receiver. Clear BRCD by writing 1 to it. Writing 0 to BRCD has no effect. 0 No BREAK condition was detected 1 A BREAK condition was detected (write 1 to clear)
1 ORE	**Overrun Error**. When set to 1, ORE indicates that the receive buffer (RxFIFO) was full (32 chars inside), and a 33rd character has been fully received. This 33rd character has been discarded. Clear ORE by writing 1 to it. Writing 0 to ORE has no effect.

Table continues on the next page...

UARTx_USR2 field descriptions (continued)

Field	Description
	0 No overrun error 1 Overrun error (write 1 to clear)
0 RDR	**Receive Data Ready**-Indicates that at least 1 character is received and written to the RxFIFO. If the URXD register is read and there is only 1 character in the RxFIFO, RDR is automatically cleared. 0 No receive data ready 1 Receive data ready

15.3.15.10 UART Escape Character Register (UARTx_UESC)

Address: Base address + 9Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																								ESC_CHAR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1

UARTx_UESC field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value 0.
ESC_CHAR	**UART Escape Character**. Holds the selected escape character that all received characters are compared against to detect an escape sequence.

15.3.15.11 UART Escape Timer Register (UARTx_UTIM)

Address: Base address + A0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																				TIM											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UARTx_UTIM field descriptions

Field	Description
31–12 Reserved	This read-only field is reserved and always has the value 0.
TIM	UART Escape Timer. Holds the maximum time interval (in ms) allowed between escape characters. The escape timer register is programmable in intervals of 2 ms. See Escape Sequence Detection and Table 15-35 for more information on the UART escape sequence detection. Reset value 0x000 = 2 ms up to 0xFFF = 8.192 s.

15.3.15.12 UART BRM Incremental Register (UARTx_UBIR)

This register can be written by both software and hardware. When enabling the automatic baud rate detection feature hardware can write 0x000F value into the UBIR after finishing detecting baud rate. Hardware has higher priority when both software and hardware try to write it at the same cycle[8].

Please note software reset will reset the register to its reset value.

Address: Base address + A4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																INC															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UARTx_UBIR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
INC	Incremental Numerator. Holds the numerator value minus one of the BRM ratio (see Binary Rate Multiplier (BRM)). The UBIR register MUST be updated before the UBMR register for the baud rate to be updated correctly. If only one register is written to by software, the BRM will ignore this data until the other register is written to by software. Updating this field using byte accesses is not recommended and is undefined.

15.3.15.13 UART BRM Modulator Register (UARTx_UBMR)

This register can be written by both software and hardware. When enabling the automatic baud rate detection feature hardware can write a proper value into the UBMR based on detected baud rate. Hardware has higher priority when both software and hardware try to write it at the same cycle[9].

Please note software reset will reset the register to its reset value.

Address: Base address + A8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																MOD															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

8. Note: The write priority in the new design is not same as the original UART. In the orginal design, software has higher priority than hardware when writing this register at the same time.
9. Note: The write priority in the new design is not same as the original UART. In the orginal design, software has higher priority than hardware when writing this register at the same time.

UARTx_UBMR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
MOD	**Modulator Denominator.** Holds the value of the denominator minus one of the BRM ratio (see Binary Rate Multiplier (BRM)). The UBIR register MUST be updated before the UBMR register for the baud rate to be updated correctly. If only one register is written to by software, the BRM will ignore this data until the other register is written to by software. Updating this register using byte accesses is not recommended and undefined.

15.3.15.14 UART Baud Rate Count Register (UARTx_UBRC)

Address: Base address + ACh offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																BCNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

UARTx_UBRC field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
BCNT	**Baud Rate Count Register.** This read only register is used to count the start bit of the incoming baud rate (if ADNIMP=1), or start bit + bit0 (if ADNIMP=0). When the measurement is done, the Baud Rate Count Register contains the number of UART internal clock cycles (clock after divider) present in an incoming bit. BCNT retains its value until the next Automatic Baud Rate Detection sequence has been initiated. The 16 bit Baud Rate Count register is reset to 4 and stays at hex FFFF in the case of an overflow.

15.3.15.15 UART One Millisecond Register (UARTx_ONEMS)

NOTE

This register has been expanded from 16 bits to 24 bits. In previous versions, the 16-bit ONEMS can only support the maximum 65.535MHz (0xFFFFx1000) *ref_clk*. To support 4Mbps Bluetooth application with 66.5MHz *module_clock*, the value 0x103C4 (66.5M/1000) should be written into this register. In this case, the 16 bits are not enough to contain the 0x103C4. So this register was expanded to 24 bits to support high frequency of the *ref_clk*.

Address: Base address + B0h offset

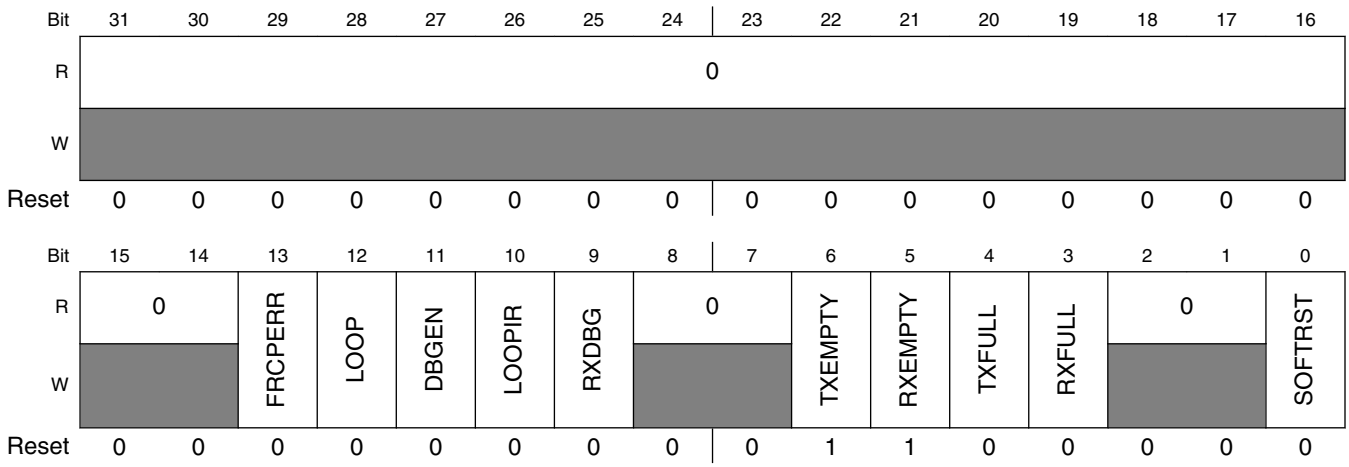
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								ONEMS																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UARTx_ONEMS field descriptions

Field	Description
31–24 Reserved	This read-only field is reserved and always has the value 0.
ONEMS	**One Millisecond Register.** This 24-bit register must contain the value of the UART internal frequency (*ref_clk* in Figure 15-14) divided by 1000. The internal frequency is obtained after the UART BRM internal divider (F (*ref_clk*) = F(*module_clock*) / RFDIV). In fact this register contains the value corresponding to the number of UART BRM internal clock cycles present in one millisecond. The ONEMS (and UTIM) registers value are used in the escape character detection feature (Escape Sequence Detection) to count the number of clock cycles left between two escape characters. The ONEMS register is also used in infrared special case mode (IRSC = UCR4[5] = 1'b1), see InfraRed Special Case (IRSC) Bit.

15.3.15.16 UART Test Register (UARTx_UTS)

Address: Base address + B4h offset



UARTx_UTS field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
13 FRCPERR	Force Parity Error. Forces the transmitter to generate a parity error if parity is enabled. FRCPERR is provided for system debugging. 0 Generate normal parity 1 Generate inverted parity (error)
12 LOOP	Loop TX and RX for Test. Controls loopback for test purposes. When LOOP is high, the receiver input is internally connected to the transmitter and ignores the RXD pin. The transmitter is unaffected by LOOP. If RXDMUXSEL (UCR3[2]) is set to 1, the loopback is applied on serial and IrDA signals. If RXDMUXSEL is set to 0, the loopback is only applied on serial signals. 0 Normal receiver operation 1 Internally connect the transmitter output to the receiver input
11 DBGEN	This bit is not used in this chip. debug_enable_B**.** This bit controls whether to respond to the *debug_req* input signal. 0 UART will go into debug mode when *debug_req* is HIGH 1 UART will not go into debug mode even if *debug_req* is HIGH
10 LOOPIR	**Loop TX and RX for IR Test (LOOPIR)**. This bit controls loopback from transmitter to receiver in the InfraRed interface. 0 No IR loop 1 Connect IR transmitter to IR receiver
9 RXDBG	This bit is not used in this chip. **RX_fifo_debug_mode.** This bit controls the operation of the RX fifo read counter when in debug mode. 0 rx fifo read pointer does not increment 1 rx_fifo read pointer increments as normal

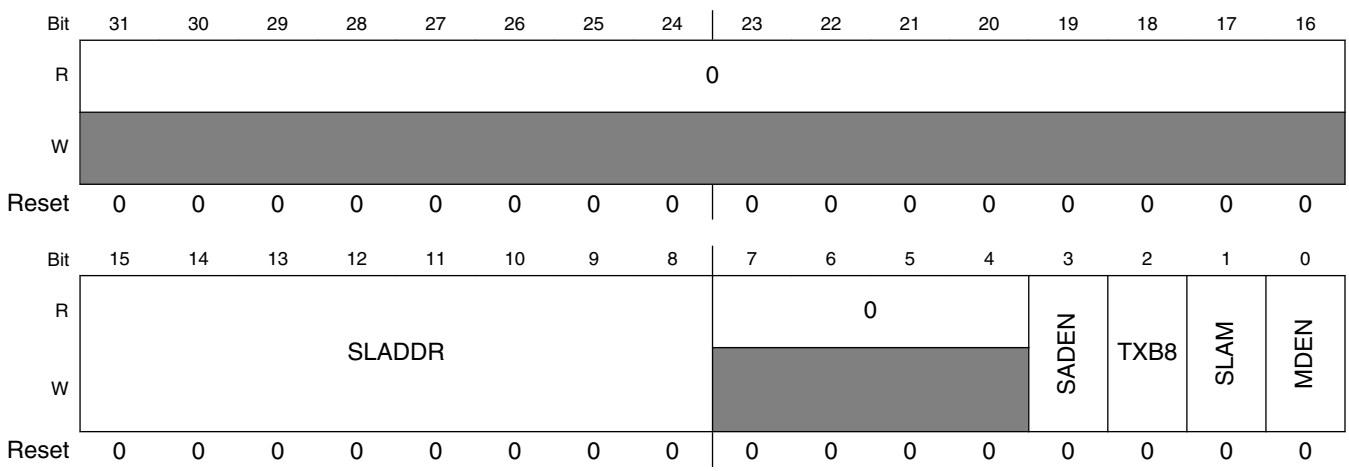
Table continues on the next page...

UARTx_UTS field descriptions (continued)

Field	Description
8–7 Reserved	This read-only field is reserved and always has the value 0.
6 TXEMPTY	TxFIFO Empty. Indicates that the TxFIFO is empty. 0 The TxFIFO is not empty 1 The TxFIFO is empty
5 RXEMPTY	RxFIFO Empty. Indicates the RxFIFO is empty. 0 The RxFIFO is not empty 1 The RxFIFO is empty
4 TXFULL	TxFIFO FULL. Indicates the TxFIFO is full. 0 The TxFIFO is not full 1 The TxFIFO is full
3 RXFULL	RxFIFO FULL. Indicates the RxFIFO is full. 0 The RxFIFO is not full 1 The RxFIFO is full
2–1 Reserved	This read-only field is reserved and always has the value 0.
0 SOFTRST	Software Reset. Indicates the status of the software reset (SRST_B bit of UCR2). 0 Software reset inactive 1 Software reset active

15.3.15.17 UART RS-485 Mode Control Register (UARTx_UMCR)

Address: Base address + B8h offset



UARTx_UMCR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–8 SLADDR	RS-485 Slave Address Character. Holds the selected slave adress character that the receiver wil try to detect.
7–4 Reserved	This read-only field is reserved and always has the value 0.
3 SADEN	RS-485 Slave Address Detected Interrupt Enable. 0 Disable RS-485 Slave Address Detected Interrupt 1 Enable RS-485 Slave Address Detected Interrupt
2 TXB8	Transmit RS-485 bit 8 (the ninth bit or 9th bit). In RS-485 mode, software writes TXB8 bit as the 9th data bit to be transmitted. 0 0 will be transmitted as the RS485 9th data bit 1 1 will be transmitted as the RS485 9th data bit
1 SLAM	RS-485 Slave Address Detect Mode Selection. 0 Select Normal Address Detect mode 1 Select Automatic Address Detect mode
0 MDEN	9-bit data or Multidrop Mode (RS-485) Enable. 0 Normal RS-232 or IrDA mode, see Table 15-27 for detail. 1 Enable RS-485 mode, see Table 15-27 for detail

15.4 Keypad Port (KPP)

15.4.1 Overview

The Keypad Port (KPP) is a 16-bit peripheral that can be used as a keypad matrix interface or as general purpose input/output (I/O).

The figure below shows the KPP block diagram. The KPP provides interface for the keypad matrix with 2-point contact or 3-point contact keys. The KPP is designed to simplify the software task of scanning a keypad matrix. With appropriate software support, the KPP is capable of detecting, debouncing, and decoding one or multiple keys pressed simultaneously on the keypad.

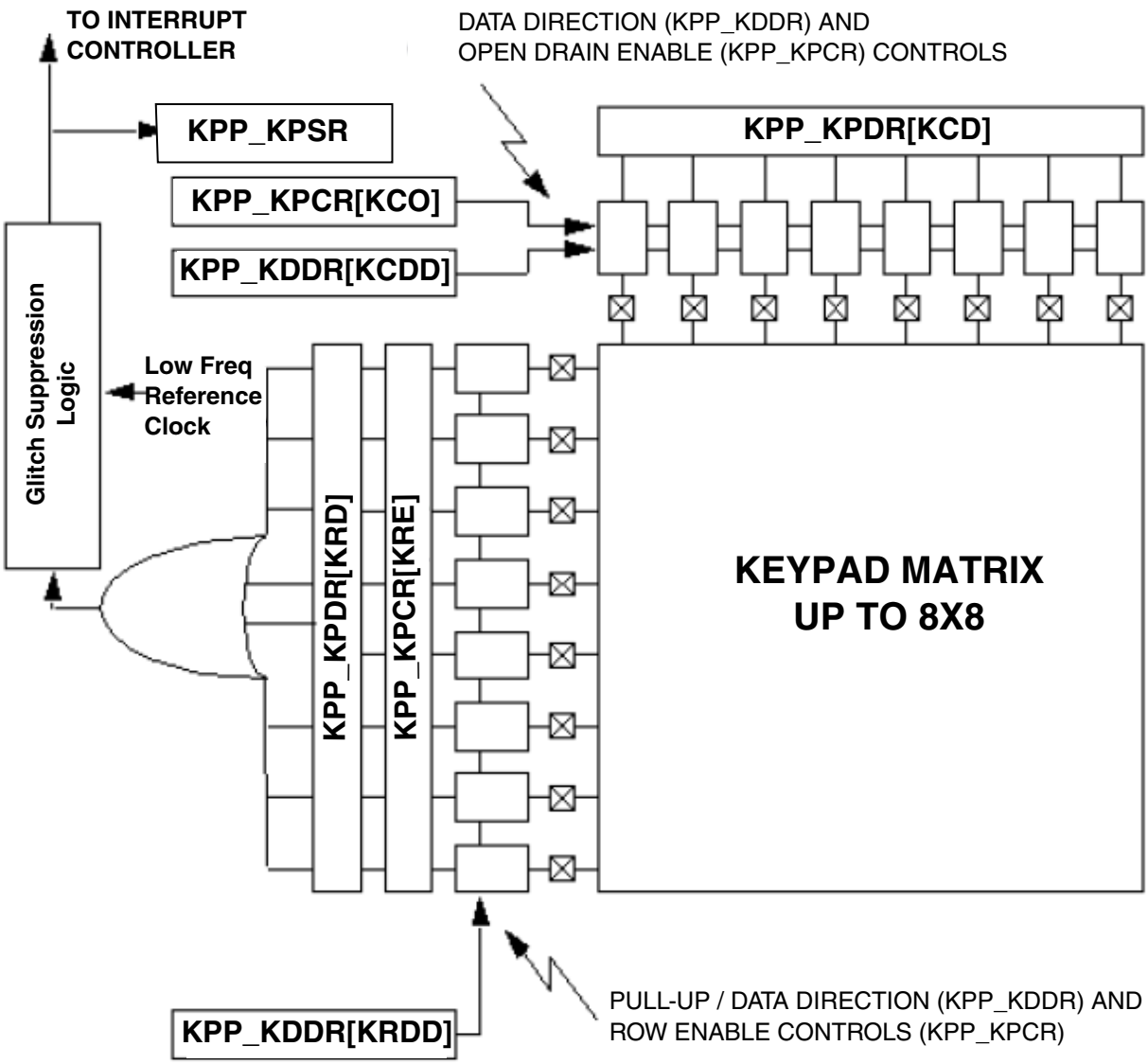


Figure 15-27. KPP Peripheral Block Diagram

15.4.1.1 Features

The KPP includes these distinctive features:

- Supports up to an 8 x 8 external key pad matrix
- Port pins can be used as general purpose I/O
- Open drain design
- Glitch suppression circuit design
- Multiple-key detection
- Long key-press detection
- Standby key-press detection
- Synchronizer chain clear
- Supports a 2-point and 3-point contact key matrix

15.4.1.2 Modes and Operations

This block supports the following modes:

- Run Mode-This is the normal functional mode in which the KPP can detect any key press event.
- Low Power Mode-The keypad can detect any key press even in low power modes (when there is no MCU clock).

15.4.2 Clocks

The table found here describes the clock sources for KPP.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 15-37. KPP Clocks

Clock name	Clock Root	Description
ipg_clk_32k	ckil_sync_clk_root	Low-frequency reference clock (32 kHz)
ipg_clk_s	ipg_clk_root	Peripheral access clock

15.4.3 External Signals

There are several pins dedicated to the KPP. Keypads of any configuration up to eight rows and eight columns are supported through the software configuration of the peripheral pins. Any pins not used for the keypad are available as general purpose I/O. The registers are configured such that the pins can be treated as an I/O port up to 16 bits wide.

See the table below for the list of external signals.

Table 15-38. KPP External Signals

Signal	Description	Pad	Mode	Direction
KEY_COL0	Column input or output pin, from chip	ENET1_TDATA1	ALT6	IO
		EPDC1_DATA07	ALT3	
KEY_COL1	Column input or output pin, from chip	ENET1_RXC	ALT6	IO
		EPDC1_DATA05	ALT3	
KEY_COL2	Column input or output pin, from chip	ENET1_RDATA3	ALT6	IO
		EPDC1_DATA03	ALT3	
KEY_COL3	Column input or output pin, from chip	ENET1_RDATA1	ALT6	IO
		EPDC1_DATA01	ALT3	
KEY_COL4	Column input or output pin, from chip	EPDC1_SDLE	ALT3	IO
		GPIO1_IO07	ALT6	
KEY_COL5	Column input or output pin, from chip	EPDC1_SDOE	ALT3	IO
		GPIO1_IO08	ALT6	
KEY_COL6	Column input or output pin, from chip	EPDC1_SDCE2	ALT3	IO
		GPIO1_IO10	ALT6	
KEY_COL7	Column input or output pin, from chip	EPDC1_GDCLK	ALT3	IO
		SAI2_RXD	ALT6	
KEY_ROW0	Row input or output pin, from chip	ENET1_TDATA0	ALT6	IO
		EPDC1_DATA06	ALT3	
KEY_ROW1	Row input or output pin, from chip	ENET1_RX_CTL	ALT6	IO
		EPDC1_DATA04	ALT3	
KEY_ROW2	Row input or output pin, from chip	ENET1_RDATA2	ALT6	IO
		EPDC1_DATA02	ALT3	
KEY_ROW3	Row input or output pin, from chip	ENET1_RDATA0	ALT6	IO
		EPDC1_DATA00	ALT3	
KEY_ROW4	Row input or output pin, from chip	EPDC1_SDCLK	ALT3	IO
		GPIO1_IO06	ALT6	
KEY_ROW5	Row input or output pin, from chip	EPDC1_SDSHR	ALT3	IO
		GPIO1_IO09	ALT6	
KEY_ROW6	Row input or output pin, from chip	EPDC1_SDCE3	ALT3	IO

Table continues on the next page...

Table 15-38. KPP External Signals (continued)

Signal	Description	Pad	Mode	Direction
		GPIO1_IO11	ALT6	
KEY_ROW7	Row input or output pin, from chip	EPDC1_GDOE	ALT3	IO
		SAI2_TXD	ALT6	

15.4.3.1 Input Pins

Any of the 16 pins associated with the KPP can be configured as inputs by writing a "0" to the appropriate bits in the KPP_KDDR. Additionally, the least significant 8 bits (ROW inputs) corresponding to KPP_KDDR[KRDD] have internal pull-ups, which are enabled when the pin is used as an input.

15.4.3.2 Output Pins

Any of the 16 pins associated with the KPP can be configured as outputs by writing the appropriate bits in the KPP_KDDR to a "1". Additionally, the 8 most significant bits (15-8) can be designated as open drain outputs by writing a "1" to the appropriate bits in the KPP_KPCR. The lower 8 bits (7-0) are always in "totem pole" style, driven when configured as outputs.

See the table below.

Table 15-39. Keypad Port Column Modes

KPP_KDDR (15:8)	KPP_KPCR (15:8)	Pin Function
0	x	Input
1	0	Totem-Pole Output
1	1	Open-Drain Output

NOTE

Totem pole capability should be provided for column pins. Totem pole configuration helps for a faster discharge of keypad capacitance when all columns need to be quickly brought to a "1" during the scan routine. With this configuration, delay between the scanning of two subsequent columns is reduced.

15.4.3.3 Generation of Transfer Error Signal on Peripheral Bus

If there is an access to an address which is not implemented, then the KPP asserts a transfer error signal on Peripheral Bus.

15.4.4 Functional Description

The Keypad Port (KPP) is designed to simplify the software task of scanning a keypad matrix. With appropriate software support and matrix organization, the KPP is capable of detecting, debouncing, and decoding one or more keys pressed simultaneously on the keypad.

Logic in the KPP is capable of detecting a key press even while the processor is in one of the low power standby modes provided that a low frequency reference clock is on. The KPP may generate an ARM platform interrupt any time a key press or key release is detected. This interrupt is capable of forcing the processor out of a low power mode.

15.4.4.1 Keypad Matrix Construction

The KPP is designed to interface to a keypad matrix, which shorts the intersecting row and column lines together whenever a key is depressed. The interface is not optimized for any other switch configuration.

15.4.4.2 Keypad Port Configuration

The software must initialize the KPP for the size of the keypad matrix. Pins connected to the keypad columns should be configured as open-drain outputs. Pins connected to the keypad rows should be configured as inputs. On-chip, pull-up resistors should be implemented for active keypad rows.

In addition to enabled row inputs in the Keypad Control register, corresponding interrupt (depress or/and release) must also be enabled to generate an interrupt.

Discrete switches that are not part of the matrix may be connected to any unused row inputs. The second terminal of the discrete switch is connected to ground. The hardware detects closures of these switches without the need for software polling.

15.4.4.3 Keypad Matrix Scanning

Keypad scanning is performed by a software loop that walks a zero across each of the keypad columns, reading the value on the rows at each step. The process is repeated several times in succession, with the results of each pass optionally compared to those from the previous pass. When several (3 or 4) consecutive scans yield the same key closures, a valid key press has been detected. Software then can decode exactly which switch was depressed and pass the value up to the next higher software layer.

The basic debouncing period, which must be defined in the software routine, may be controlled with an internal timer. The basic period is the period between the scan of two consecutive columns, so the debouncing time between two consecutive scans of the whole matrix shall be the number of columns multiplied by the basic period.

15.4.4.4 Keypad Standby

There is no need for the ARM platform to continually scan the keypad. Between key presses, the keypad can be left in a state that requires no software intervention until the next key press is detected. To place the keypad in a standby state, software should write all column outputs low. Row inputs are left enabled. At this point, the ARM platform can attend to other tasks or revert to a low power standby mode. The KPP will interrupt the ARM platform if any key is pressed.

Upon receiving a keypad interrupt, the ARM platform should set all the column strobes high, and begin a normal keypad scanning routine to determine which key was pressed. It is important that open-drain drivers be used when scanning to prevent a possible DC path between power and ground through two or more switches.

15.4.4.5 Glitch Suppression on Keypad Inputs

A glitch suppression circuit qualifies the keypad inputs to prevent noise from inadvertently interrupting the ARM platform. The circuit is a 4-state synchronizer clocked from a low frequency reference clock source.

This clock must continue to run in any low power mode where the keypad is a wake-up source, as the ARM platform interrupt is generated from the synchronized input. An interrupt is not generated until all four synchronizer stages have latched a valid key assertion. This guarantees the filtering out of any noise less than three clock periods in duration of a low frequency reference clock. Noise filtering of the duration between three to four clock periods cannot be guaranteed. The interrupt output is latched in an S-R latch and remains asserted until cleared by the software. The Set input of the latch is rising-edge clocked. See the figure below.

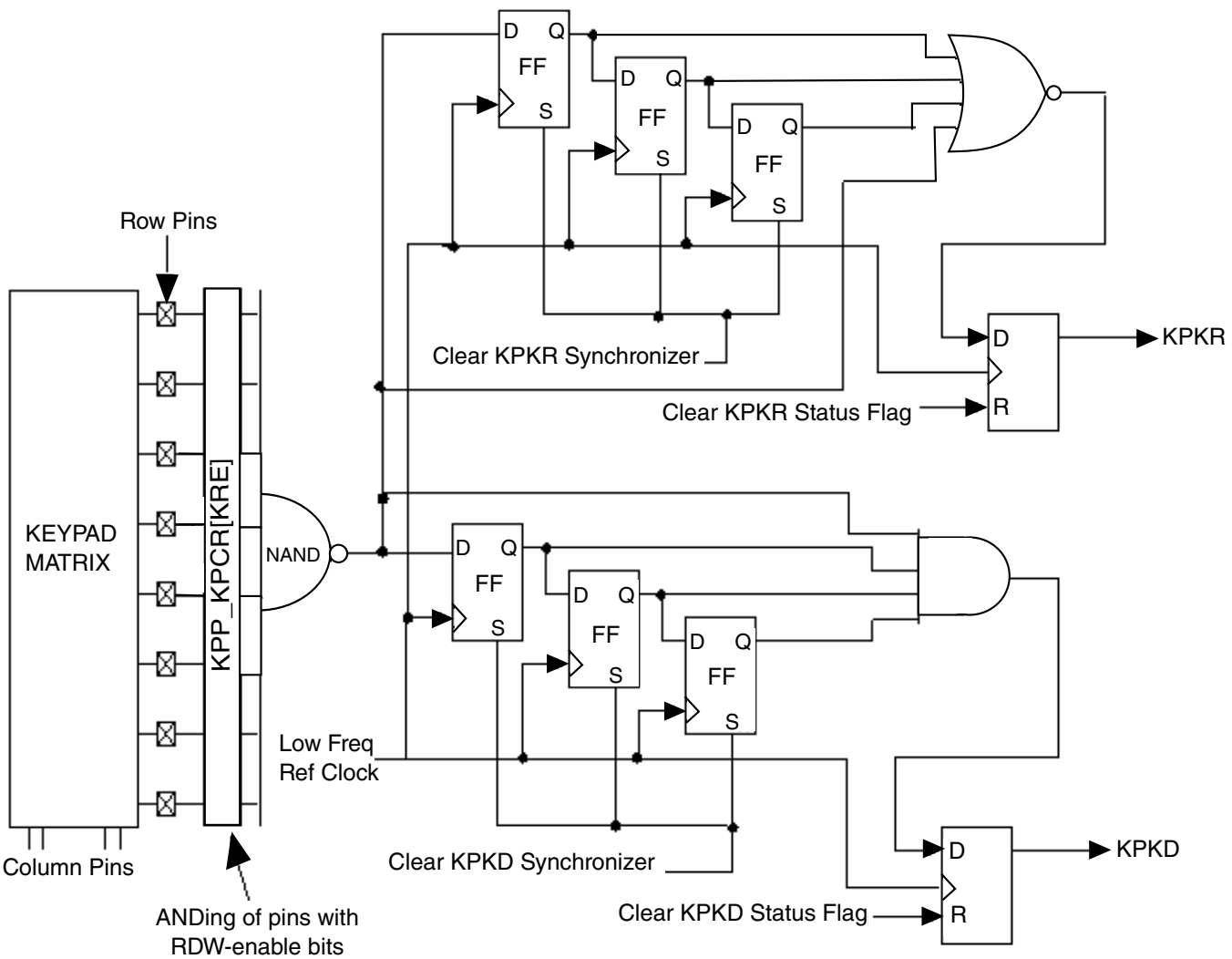


Figure 15-28. Keypad Synchronizer Functional Diagram

15.4.4.6 Multiple Key Closures

Using the key press and Key release interrupts, the software can detect multiple keys or achieve n key rollover. The key scanning routine can be programmed accordingly.

See the following figures for illustrations of the interfacing of a 2-contact keypad matrix with the KPP controller. With proper enabling of row lines and the performing scan-routine, multiple key presses can be detected. When keys present on the same row are pressed, corresponding row lines (multiple lines) become low when the column is driven low during a scan-routine. By reading the data-register, pressed keys can be detected. Similarly, when keys present on same row line are pressed, the corresponding row line (only one line) becomes low when logic "0" is driven on the column line during a scan-routine.

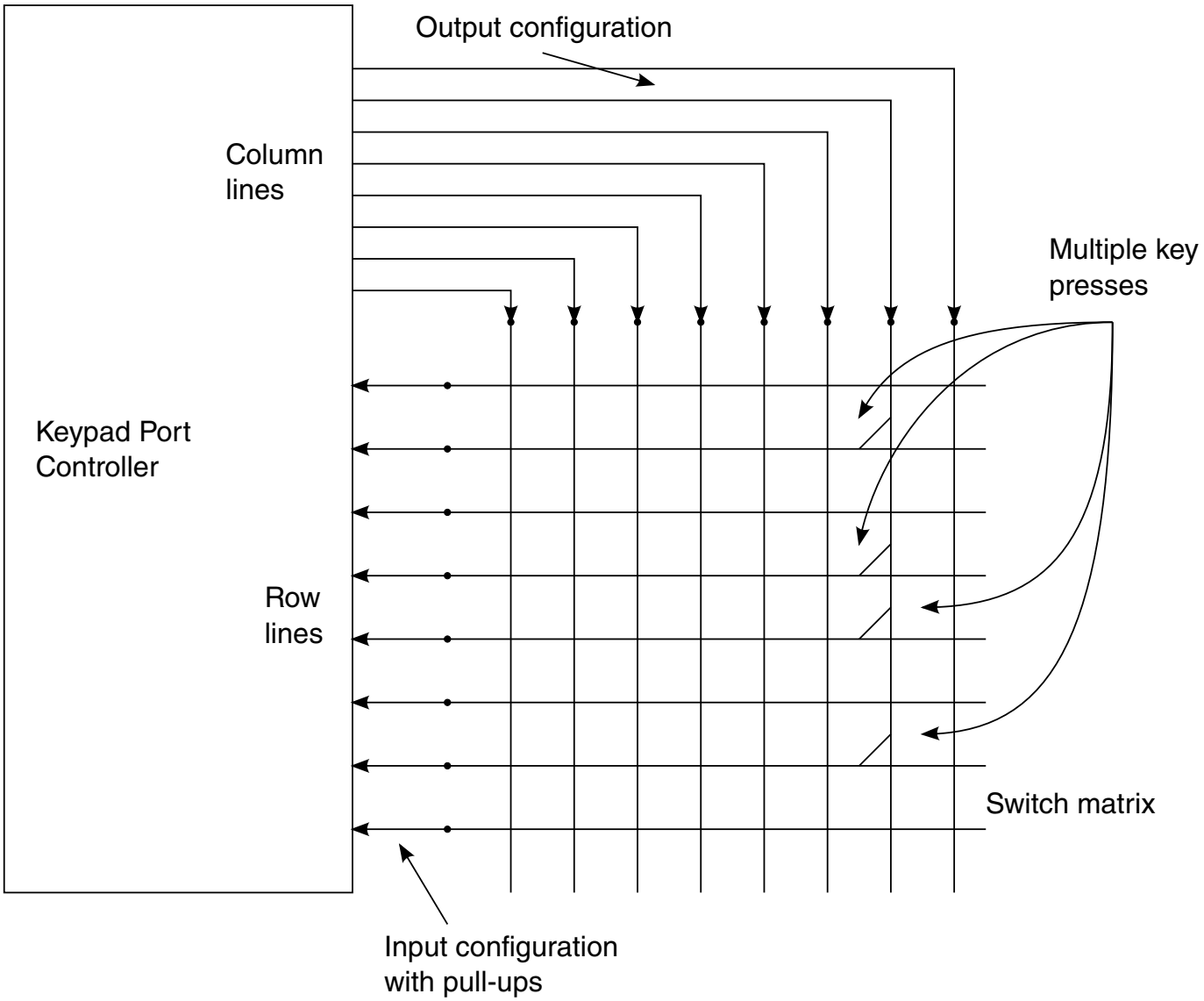


Figure 15-29. Multiple Key Presses on Same Column Line (Simplified View)

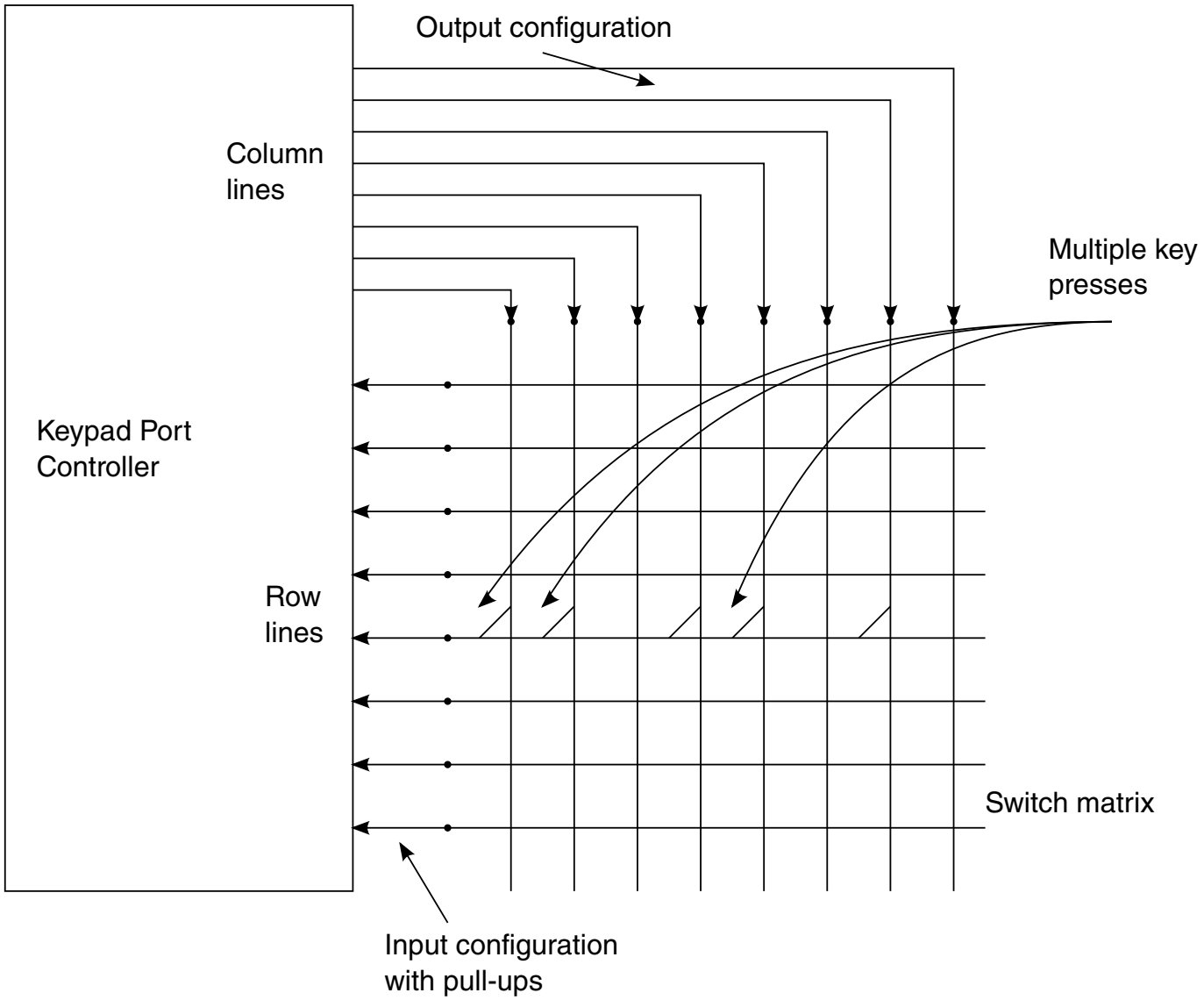


Figure 15-30. Multiple Key Presses on Same Row Line (Simplified View)

NOTE

An n key rollover is a technique with which the system can recognize the order in which keys are pressed.

15.4.4.6.1 Ghost Key Problem and Correction

The KPP detects if one or multiple keys are pressed or released. In the case where a simple keypad matrix with two-contact switches is used, there is a chance of "ghost" key detection when three or more keys are pressed. This is a limitation imposed by such a keypad matrix.

As can be seen in Figure 15-31, three keys pressed simultaneously can cause a short between the column currently "scanned" by the software and another column. Depending on the location of the third key pressed, a "ghost" key press may be detected.

However, this can be corrected by using a keypad matrix that provides "ghost" key protection. Such a matrix implements a one-way "diode" at all keypad points between rows and columns. This way, the multiple pressing of three keys will not cause a short at a fourth key (see Figure 15-32).

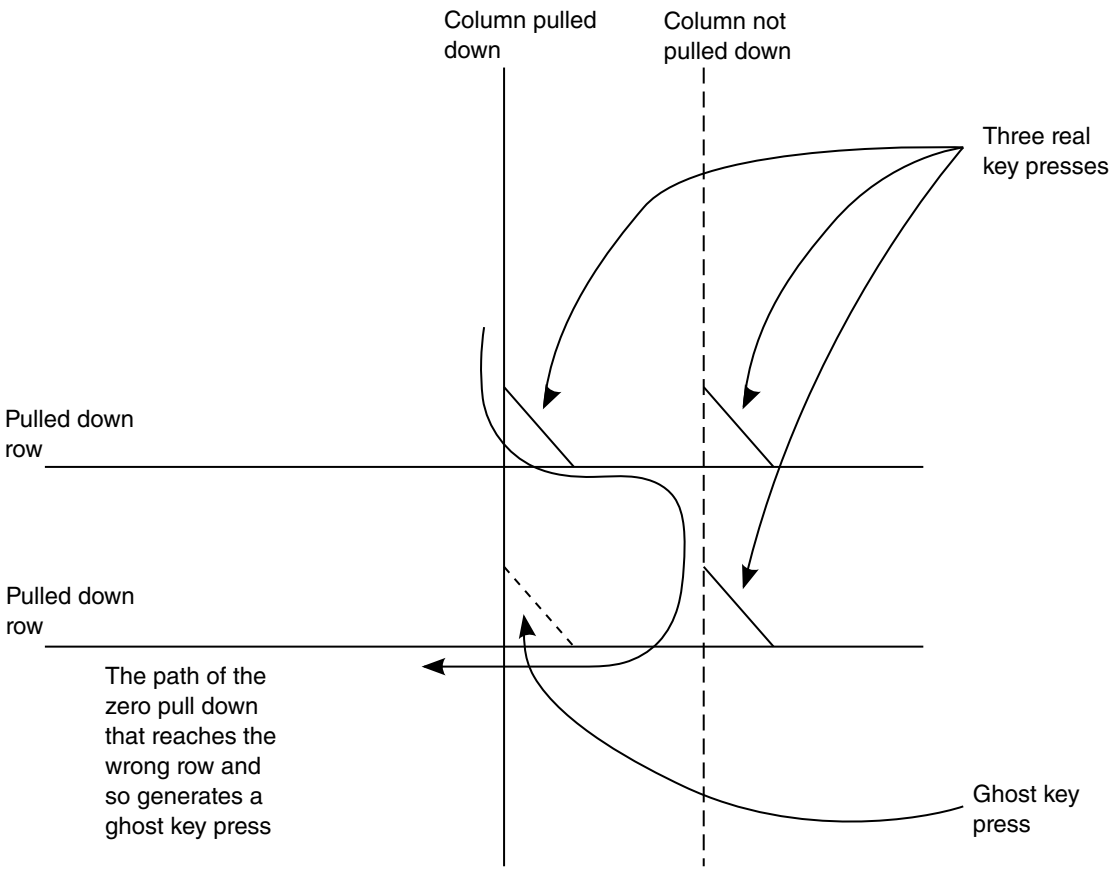


Figure 15-31. Decoding Wrong Three- Key-Presses

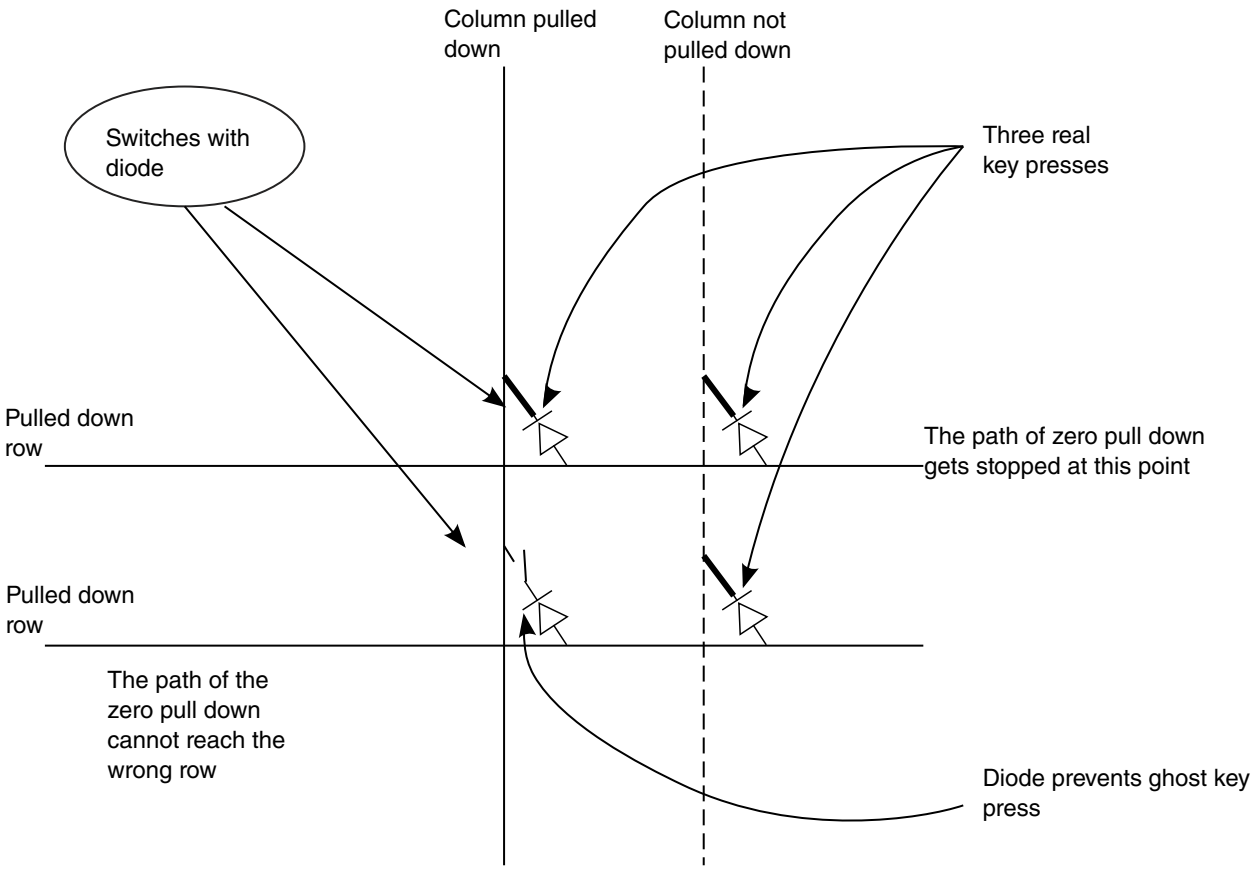


Figure 15-32. Matrix with "Ghost" Key Protections

15.4.4.7 3-Point Contact Keys Support

The KPP supports interfacing to a matrix consisting of 3-point contact keys. As shown in Figure 15-33, two points of such a key are connected to keypad lines, while a third point is connected to ground (low logic).

The keypad lines should be configured as input and a pull-up should to be present on these lines. When such a key is pressed, corresponding keypad lines go low and an interrupt is generated. There is no need to perform a scanning routine for identification of pressed key as it can be done by reading the keypad data-register. A limitation with such a matrix is that for every key at least one keypad row line should be used.

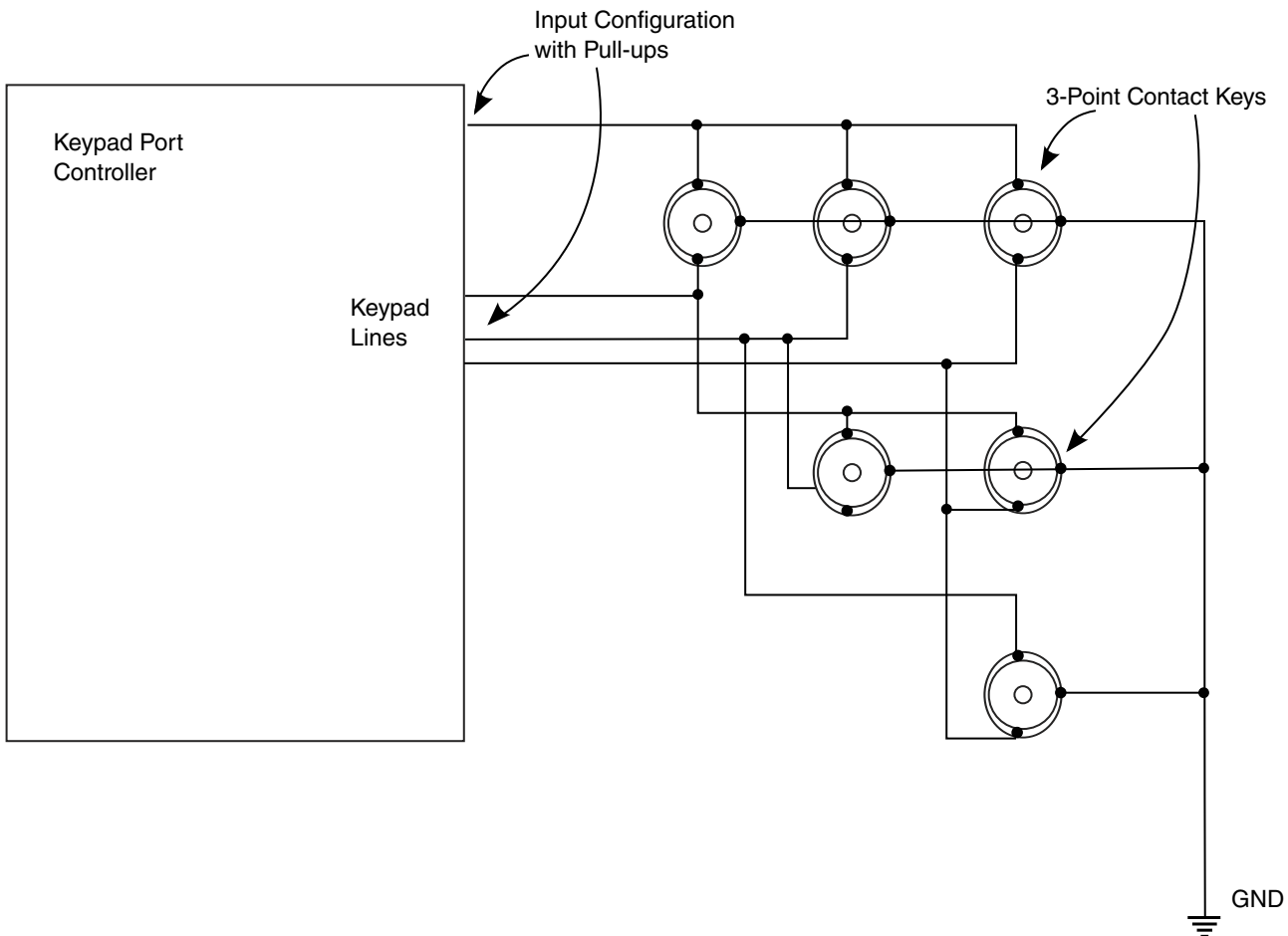


Figure 15-33. KPP Interface with 3-point Contact Key Matrix (Simplified View)

15.4.5 Initialization/Application Information

15.4.5.1 Typical Keypad Configuration and Scanning Sequence

Perform the following steps to configure the keypad:

1. Enable the number of rows in the keypad (KPP_KPCR[KRE]).
2. Write 0s to KPP_KPDR[KCD].
3. Configure the keypad columns as open-drain (KPP_KPCR[KCO]).
4. Configure columns as output (KPP_KDDR[KCDD]) and rows as input (KPP_KDDR[KRDD]).
5. Clear the KPKD Status Flag and Synchronizer chain.
6. Set the KDIE control bit, and clear the KRIE control bit (avoid false release events).
7. (The system is now in standby mode, and awaiting a key press.)

15.4.5.2 Key Press Interrupt Scanning Sequence

Perform the following steps to perform a keypad scanning routine:

1. Disable both (depress and release) keypad interrupts.
2. Write 1s to KPP_KPDR[KCD], setting column data to 1s.
3. Configure columns as totem pole outputs (for quick discharging of keypad capacitance).
4. Configure columns as open-drain.
5. Write a single column to 0, and other columns to 1.
6. Sample row inputs and save data. Multiple key presses can be detected on a single column.
7. Repeat Steps 2-6 for remaining columns.
8. Return all columns to 0 in preparation for standby mode.
9. Clear KPKD and KPKR status bit(s) by writing to a "1"; set the KPKR synchronizer chain by writing a "1" to the KPP_KRSS register; and clear the KPKD synchronizer chain by writing a "1" to the KDSC register.
10. Re-enable the appropriate keypad interrupt(s) so that the KDIE detects a key hold condition, or the KRIE detects a key-release event.

15.4.5.3 Additional Comments

The order of key press detection can be done in software only. Therefore, the software may need to run the scan routines at very short intervals of time per the application's demands. The reason that such functionality cannot be put in the KPP is that the block is limited by the number of external pins.

For the keys that require a very precise order (such as game keys), individual GPIO pins may be more useful.

15.4.6 KPP Memory Map/Register Definition

The KPP contains four registers.

KPP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3032_0000	Keypad Control Register (KPP_KPCR)	16	R/W	0000h	15.4.6.1/ 4367
3032_0002	Keypad Status Register (KPP_KPSR)	16	R/W	0400h	15.4.6.2/ 4368
3032_0004	Keypad Data Direction Register (KPP_KDDR)	16	R/W	0000h	15.4.6.3/ 4370
3032_0006	Keypad Data Register (KPP_KPDR)	16	R/W	0000h	15.4.6.4/ 4370

15.4.6.1 Keypad Control Register (KPP_KPCR)

The Keypad Control Register determines which of the eight possible column strobes are to be open drain when configured as outputs, and which of the eight row sense lines are considered in generating an interrupt to the core.

It is up to the programmer to ensure that pins being used for functions other than the keypad are properly disabled. The KPP_KPCR register is byte- or half-word-addressable.

Address: 3032_0000h base + 0h offset = 3032_0000h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	KCO								KRE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

KPP_KPCR field descriptions

Field	Description
15–8 KCO	Keypad Column Strobe Open-Drain Enable. Setting a column open-drain enable bit (KCO7-KCO0) disables the pull-up driver on that pin. Clearing the bit allows the pin to drive to the high state. This bit has no effect when the pin is configured as an input. **NOTE:** Configuration of external port control logic (for example, IOMUX) should be done properly so that the KPP controls an open-drain enable of the pin.

Table continues on the next page...

KPP_KPCR field descriptions (continued)

Field	Description
	0 **TOTEM_POLE** — Column strobe output is totem pole drive. 1 **OPEN_DRAIN** — Column strobe output is open drain.
KRE	Keypad Row Enable. Setting a row enable control bit in this register enables the corresponding row line to participate in interrupt generation. Likewise, clearing a bit disables that row from being used to generate an interrupt. This register is cleared by a reset, disabling all rows. The row-enable logic is independent of the programmed direction of the pin. Writing a "0" to the data register of the pins configured as outputs will cause a keypad interrupt to be generated if the row enable associated with that bit is set. 0 Row is not included in the keypad key press detect. 1 Row is included in the keypad key press detect.

15.4.6.2 Keypad Status Register (KPP_KPSR)

The Keypad Status Register reflects the state of the key press detect circuit. The KPP_KPSR register is byte- or half-word-addressable.

Address: 3032_0000h base + 2h offset = 3032_0002h

Bit	15	14	13	12	11	10	9	8
Read	0						KRIE	KDIE
Write								
Reset	0	0	0	0	0	1	0	0

Bit	7	6	5	4	3	2	1	0
Read	0				0	0	KPKR	KPKD
Write					KRSS	KDSC	w1c	w1c
Reset	0	0	0	0	0	0	0	0

KPP_KPSR field descriptions

Field	Description
15–10 Reserved	This read-only field is reserved and always has the value 0.
9 KRIE	Keypad Release Interrupt Enable. The software should ensure that the interrupt for a Key Release event is masked until it has entered the key pressed state, and vice versa, unless this activity is desired (as might be the case when a repeated interrupt is to be generated). The synchronizer chains are capable of being initialized to detect repeated key presses or releases. If they are not initialized when the corresponding event flag is cleared, false interrupts may be generated for depress (or release) events shorter than the length of the corresponding chain. 0 No interrupt request is generated when KPKR is set. 1 An interrupt request is generated when KPKR is set.
8 KDIE	Keypad Key Depress Interrupt Enable. Software should ensure that the interrupt for a Key Release event is masked until it has entered the key pressed state, and vice-versa, unless this activity is desired (as might be the case when a repeated interrupt is to be generated). The synchronizer chains are capable of

Table continues on the next page...

KPP_KPSR field descriptions (continued)

Field	Description
	being initialized to detect repeated key presses or releases. If they are not initialized when the corresponding event flag is cleared, false interrupts may be generated for depress (or release) events shorter than the length of the corresponding chain. 0 No interrupt request is generated when KPKD is set. 1 An interrupt request is generated when KPKD is set.
7–4 Reserved	This read-only field is reserved and always has the value 0.
3 KRSS	Key Release Synchronizer Set. Self-clear bit. The Key release synchronizer is set by writing a logic one into this bit. Reads return a value of "0". 0 No effect 1 Set bits which sets keypad release synchronizer chain
2 KDSC	Key Depress Synchronizer Clear. Self-clear bit. The Key depress synchronizer is cleared by writing a logic "1" into this bit. Reads return a value of "0". 0 No effect 1 Set bits that clear the keypad depress synchronizer chain
1 KPKR	Keypad Key Release. The keypad key release (KPKR) status bit is set when all enabled rows are detected high after synchronization (the KPKR status bit will be set when cleared by a reset). The KPKR bit may be used to generate a maskable key release interrupt. The key release synchronizer may be set high by software after scanning the keypad to ensure a known state. Due to the logic function of the release and depress synchronizer chains, it is possible to see the re-assertion of a status flag (KPKD or KPKR) if it is cleared by software prior to the system exiting the state it represents. Reset value of register is "0" as long as reset is asserted. However when reset is de-asserted, the value of the register depends upon the external row pins and can become "1". 0 No key release detected 1 All keys have been released
0 KPKD	Keypad Key Depress. The keypad key depress (KPKD) status bit is set when one or more enabled rows are detected low after synchronization. The KPKD status bit remains set until cleared by the software. The KPKD bit may be used to generate a maskable key depress interrupt. If desired, the software may clear the key press synchronizer chain to allow a repeated interrupt to be generated while a key remains pressed. In this case, a new interrupt will be generated after the synchronizer delay (4 cycles of the low frequency reference clock elapses if a key remains pressed. This functionality can be used to detect a long key press. This allows detection of additional key presses of the same key or other keys. Due to the logic function of the release and depress synchronizer chains, it is possible to see the re-assertion of a status flag (KPKD or KPKR) if it is cleared by the software prior to the system exiting the state it represents. 0 No key presses detected 1 A key has been depressed

15.4.6.3 Keypad Data Direction Register (KPP_KDDR)

The bits in the KPP_KDDR control the direction of the keypad port pins. The upper eight bits in the register affect the pins designated as column strobes, while the lower eight bits affect the row sense pins. Setting any bit in this register configures the corresponding pin as an output. Clearing any bit in this register configures the corresponding port pin as an input. For the Keypad Row DDR, an internal pull-up is enabled if the corresponding bit is clear. This register is cleared by a reset, configuring all pins as inputs. The KPP_KDDR register is byte- or half-word addressable.

NOTE

When a pin is used as row pin for keypad purposes, all corresponding pull-ups should be enabled at the upper level (for example, IOMUX) when the bit in KRDD is cleared.

Address: 3032_0000h base + 4h offset = 3032_0004h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	KCDD								KRDD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

KPP_KDDR field descriptions

Field	Description
15–8 KCDD	Keypad Column Data Direction Register. Setting a bit configures the corresponding COL*n* pin as an output (where *n* = 7 through 0). 0 **INPUT** — COLn pin is configured as an input. 1 **OUTPUT** — COLn pin is configured as an output.
KRDD	Keypad Row Data Direction. Setting a bit configures the corresponding ROW*n* pin as an output (where *n* = 7 through 0). 0 **INPUT** — ROWn pin configured as an input. 1 **OUTPUT** — ROWn pin configured as an output.

15.4.6.4 Keypad Data Register (KPP_KPDR)

This 16-bit register is used to access the column and row data. Data written to this register is stored in an internal latch, and for each pin configured as an output, the stored data is driven onto the pin. A read of this register returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register.

The KPP_KPDR register is byte- or half-word addressable. This register is not initialized by a reset. Valid data should be written to this register before any bits are configured as outputs.

Address: 3032_0000h base + 6h offset = 3032_0006h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	KCD								KRD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

KPP_KPDR field descriptions

Field	Description
15–8 KCD	Keypad Column Data. A read of these bits returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register. 0 Read/Write "0" from/to column ports 1 Read/Write "1" from/to column ports
KRD	Keypad Row Data. A read of these bits returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register. 0 Read/Write "0" from/to row ports 1 Read/Write "1" from/to row ports

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

NXP reserves the right to make changes without further notice to any products herein. NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Document Number: IMX7SRM
Rev. 0.1, 08/2016