

To our customers,

---

## Old Company Name in Catalogs and Other Documents

---

On April 1<sup>st</sup>, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: <http://www.renesas.com>

April 1<sup>st</sup>, 2010  
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (<http://www.renesas.com>)

Send any inquiries to <http://www.renesas.com/inquiry>.

## Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

"Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

To all our customers

---

## **Regarding the change of names mentioned in the document, such as Hitachi Electric and Hitachi XX, to Renesas Technology Corp.**

---

The semiconductor operations of Mitsubishi Electric and Hitachi were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Hitachi, Hitachi, Ltd., Hitachi Semiconductors, and other Hitachi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Renesas Technology Home Page: <http://www.renesas.com>

Renesas Technology Corp.  
Customer Support Dept.  
April 1, 2003

## Cautions

Keep safety first in your circuit designs!

1. Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.  
Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
2. Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.  
The information described here may contain technical inaccuracies or typographical errors.  
Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.  
Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (<http://www.renesas.com>).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.  
Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.

# SH7011

## Hardware Manual

# Cautions

1. Hitachi neither warrants nor grants licenses of any rights of Hitachi's or any third party's patent, copyright, trademark, or other intellectual property rights for information contained in this document. Hitachi bears no responsibility for problems that may arise with third party's rights, including intellectual property rights, in connection with use of the information contained in this document.
2. Products and product specifications may be subject to change without notice. Confirm that you have received the latest product standards or specifications before final design, purchase or use.
3. Hitachi makes every attempt to ensure that its products are of high quality and reliability. However, contact Hitachi's sales office before using the product in an application that demands especially high quality and reliability or where its failure or malfunction may directly threaten human life or cause risk of bodily injury, such as aerospace, aeronautics, nuclear power, combustion control, transportation, traffic, safety equipment or medical equipment for life support.
4. Design your application so that the product is used within the ranges guaranteed by Hitachi particularly for maximum rating, operating supply voltage range, heat radiation characteristics, installation conditions and other characteristics. Hitachi bears no responsibility for failure or damage when used beyond the guaranteed ranges. Even within the guaranteed ranges, consider normally foreseeable failure rates or failure modes in semiconductor devices and employ systemic measures such as fail-safes, so that the equipment incorporating Hitachi product does not cause bodily injury, fire or other consequential damage due to operation of the Hitachi product.
5. This product is not designed to be radiation resistant.
6. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without written approval from Hitachi.
7. Contact Hitachi's sales office for any questions regarding this document or Hitachi semiconductor products.

# Contents

Section 1	SH7011 Overview .....	1
1.1	SH7011 Overview .....	1
1.1.1	SH7011 Features .....	1
1.2	Block Diagram.....	4
1.3	Pin Arrangement and Pin Functions.....	5
1.3.1	Pin Arrangement .....	5
1.3.2	Pin Functions.....	6
Section 2	CPU.....	9
2.1	Register Configuration .....	9
2.1.1	General Registers (Rn).....	9
2.1.2	Control Registers.....	10
2.1.3	System Registers .....	11
2.1.4	Initial Values of Registers.....	11
2.2	Data Formats.....	12
2.2.1	Data Format in Registers.....	12
2.2.2	Data Format in Memory .....	12
2.2.3	Immediate Data Format.....	12
2.3	Instruction Features.....	13
2.3.1	RISC-Type Instruction Set .....	13
2.3.2	Addressing Modes.....	16
2.3.3	Instruction Format .....	20
2.4	Instruction Set by Classification.....	23
2.5	Processing States .....	36
2.5.1	State Transitions .....	36
Section 3	Power-Down State.....	39
3.1	Overview.....	39
3.1.1	Power-Down State.....	39
3.2	Sleep Mode.....	40
3.2.1	Transition to Sleep Mode .....	40
3.2.2	Exit from Sleep Mode .....	40
Section 4	Clock Pulse Generator (CPG).....	41
4.1	Overview.....	41
4.2	Clock Source.....	41
4.2.1	Crystal Resonator Connection.....	41
4.2.2	External Clock Input .....	43
4.3	Usage Notes .....	43

Section 5	Exception Processing .....	45
5.1	Overview.....	45
5.1.1	Types of Exception Processing and Priority .....	45
5.1.2	Exception Processing Operations .....	46
5.1.3	Exception Processing Vector Table.....	47
5.2	Resets.....	49
5.2.1	Reset .....	49
5.2.2	Power-On Reset.....	49
5.3	Address Errors .....	50
5.3.1	Address Error Exception Processing .....	50
5.4	Interrupts.....	51
5.4.1	Interrupt Priority Level.....	51
5.4.2	Interrupt Exception Processing .....	52
5.5	Exceptions Triggered by Instructions.....	52
5.5.1	Trap Instructions .....	53
5.5.2	Illegal Slot Instructions .....	53
5.5.3	General Illegal Instructions .....	53
5.6	When Exception Sources Are Not Accepted.....	54
5.6.1	Immediately after a Delayed Branch Instruction.....	54
5.6.2	Immediately after an Interrupt-Disabled Instruction.....	54
5.7	Stack Status after Exception Processing Ends.....	55
5.8	Notes on Use.....	56
5.8.1	Value of Stack Pointer (SP).....	56
5.8.2	Value of Vector Base Register (VBR) .....	56
5.8.3	Address Errors Caused by Stacking of Address Error Exception Processing.....	56
Section 6	Interrupt Controller (INTC).....	57
6.1	Overview.....	57
6.1.1	Features .....	57
6.1.2	Block Diagram.....	58
6.1.3	Pin Configuration .....	59
6.1.4	Register Configuration .....	59
6.2	Interrupt Sources.....	60
6.2.1	NMI Interrupts.....	60
6.2.2	IRQ Interrupts .....	60
6.2.3	On-Chip Peripheral Module Interrupts .....	61
6.2.4	Interrupt Exception Vectors and Priority Rankings .....	61
6.3	Description of Registers .....	64
6.3.1	Interrupt Priority Registers A–H (IPRA–IPRH) .....	64
6.3.2	Interrupt Control Register (ICR) .....	65
6.3.3	IRQ Status Register (ISR).....	66
6.4	Interrupt Operation .....	68
6.4.1	Interrupt Sequence.....	68



6.4.2	Stack after Interrupt Exception Processing .....	70
6.5	Interrupt Response Time.....	70
<b>Section 7</b>	<b>Bus State Controller (BSC) .....</b>	<b>73</b>
7.1	Overview.....	73
7.1.1	Features .....	73
7.1.2	Block Diagram.....	74
7.1.3	Pin Configuration .....	75
7.1.4	Register Configuration .....	75
7.1.5	Address Map .....	76
7.2	Description of Registers .....	77
7.2.1	Bus Control Register 2 (BCR2).....	77
7.2.2	Wait Control Register 1 (WCR1).....	80
7.3	Accessing Ordinary Space.....	82
7.3.1	Basic Timing .....	82
7.3.2	Wait State Control.....	83
7.3.3	$\overline{CS}$ Assert Period Extension.....	85
7.4	Waits between Access Cycles .....	86
7.4.1	Prevention of Data Bus Conflicts.....	86
7.4.2	Simplification of Bus Cycle Start Detection .....	87
7.5	Memory Connection Examples .....	88
<b>Section 8</b>	<b>Multifunction Timer Pulse Unit (MTU).....</b>	<b>89</b>
8.1	Overview.....	89
8.1.1	Features .....	89
8.1.2	Block Diagram.....	92
8.1.3	Pin Configuration .....	93
8.1.4	Register Configuration .....	94
8.2	MTU Register Descriptions.....	95
8.2.1	Timer Control Register (TCR) .....	95
8.2.2	Timer Mode Register (TMDR) .....	99
8.2.3	Timer I/O Control Register (TIOR) .....	100
8.2.4	Timer Interrupt Enable Register (TIER) .....	107
8.2.5	Timer Status Register (TSR).....	109
8.2.6	Timer Counters (TCNT).....	111
8.2.7	Timer General Register (TGR) .....	112
8.2.8	Timer Start Register (TSTR).....	112
8.2.9	Timer Synchro Register (TSYR).....	113
8.3	Bus Master Interface.....	114
8.3.1	16-Bit Registers.....	114
8.3.2	8-Bit Registers .....	114
8.4	Operation .....	116
8.4.1	Overview .....	116

8.4.2	Basic Functions .....	116
8.4.3	Synchronous Operation .....	122
8.4.4	Buffer Operation .....	124
8.4.5	Cascade Connection Mode .....	127
8.4.6	PWM Mode .....	128
8.5	Interrupts.....	133
8.5.1	Interrupt Sources and Priority Ranking.....	133
8.5.2	A/D Converter Activation .....	134
8.6	Operation Timing .....	135
8.6.1	Input/Output Timing .....	135
8.6.2	Interrupt Signal Timing.....	139
8.7	Notes and Precautions.....	141
8.8	MTU Output Pin Initialization .....	150
8.8.1	Operating Modes .....	150
8.8.2	Reset Start Operation.....	151
8.8.3	Operation in Case of Re-Setting Due to Error During Operation, Etc.....	151
8.8.4	Overview of Initialization Procedures and Mode Transitions in Case of Error During Operation, Etc. ....	152
Section 9 8-Bit Timer 1 (TIM1).....		163
9.1	Overview.....	163
9.1.1	Features .....	163
9.1.2	Block Diagram.....	164
9.1.3	Register Configuration .....	165
9.2	Register Descriptions.....	165
9.2.1	Timer 1 Counter (T1CNT) .....	165
9.2.2	Timer 1 Control/Status Register (T1CSR) .....	166
9.2.3	Notes on Register Access .....	167
9.3	Operation .....	168
9.3.1	Interval Timer Operation.....	168
9.3.2	Timing of Overflow Flag (OVF) Setting.....	169
9.4	Usage Notes .....	170
9.4.1	Contention between Timer 1 Counter (TCNT) Write and Increment.....	170
9.4.2	Rewriting Bits CKS2 to CKS0.....	170
Section 10 8-Bit Timer 2 (TIM2).....		171
10.1	Overview.....	171
10.1.1	Features .....	171
10.1.2	Block Diagram.....	172
10.1.3	Register Configuration .....	172
10.2	Register Descriptions.....	173
10.2.1	Timer 2 Control/Status Register (T2CSR) .....	173
10.2.2	Timer 2 Counter (T2CNT) .....	174

10.2.3	Timer 2 Constant Register (T2COR) .....	175
10.3	Operation .....	176
10.3.1	Cyclic Count Operation.....	176
10.3.2	T2CNT Count Timing.....	176
10.4	Interrupts.....	177
10.4.1	Interrupt Source.....	177
10.4.2	Timing of Compare Match Flag Setting .....	177
10.4.3	Timing of Compare Match Flag Clearing .....	178
<b>Section 11</b>	<b>Compare Match Timer (CMT) .....</b>	<b>179</b>
11.1	Overview.....	179
11.1.1	Features .....	179
11.1.2	Block Diagram.....	180
11.1.3	Register Configuration .....	181
11.2	Register Descriptions.....	182
11.2.1	Compare Match Timer Start Register (CMSTR) .....	182
11.2.2	Compare Match Timer Control/Status Register (CMCSR).....	183
11.2.3	Compare Match Timer Counter (CMCNT).....	184
11.2.4	Compare Match Timer Constant Register (CMCOR).....	185
11.3	Operation .....	185
11.3.1	Period Count Operation.....	185
11.3.2	CMCNT Count Timing .....	186
11.4	Interrupts.....	186
11.4.1	Interrupt Sources .....	186
11.4.2	Compare Match Flag Set Timing .....	186
11.4.3	Compare Match Flag Clear Timing.....	187
11.5	Notes on Use.....	188
11.5.1	Contention between CMCNT Write and Compare Match .....	188
11.5.2	Contention between CMCNT Word Write and Incrementation .....	189
11.5.3	Contention between CMCNT Byte Write and Incrementation .....	190
<b>Section 12</b>	<b>Serial Communication Interface (SCI) .....</b>	<b>191</b>
12.1	Overview.....	191
12.1.1	Features .....	191
12.1.2	Block Diagram.....	192
12.1.3	Pin Configuration .....	192
12.1.4	Register Configuration .....	193
12.2	Register Descriptions.....	193
12.2.1	Receive Shift Register (RSR).....	193
12.2.2	Receive Data Register (RDR) .....	194
12.2.3	Transmit Shift Register (TSR).....	194
12.2.4	Transmit Data Register (TDR).....	194
12.2.5	Serial Mode Register (SMR).....	195

12.2.6	Serial Control Register (SCR).....	197
12.2.7	Serial Status Register (SSR).....	199
12.2.8	Bit Rate Register (BRR).....	203
12.3	Operation .....	211
12.3.1	Overview .....	211
12.3.2	Operation in Asynchronous Mode.....	212
12.3.3	Multiprocessor Communication .....	221
12.4	Interrupt .....	228
12.5	Notes on Use.....	229
Section 13	A/D Converter (A/D).....	231
13.1	Overview.....	231
13.1.1	Features .....	231
13.1.2	Block Diagram.....	232
13.1.3	Pin Configuration .....	233
13.1.4	Register Configuration .....	234
13.2	Register Descriptions.....	234
13.2.1	A/D Data Registers A–D (ADDRA–ADDRD).....	234
13.2.2	A/D Control/Status Register (ADCSR).....	235
13.2.3	A/D Control Register (ADCR).....	237
13.3	CPU Interface .....	238
13.4	Operation .....	240
13.4.1	Single Mode (SCAN = 0).....	240
13.4.2	Scan Mode (SCAN = 1) .....	242
13.4.3	Input Sampling and A/D Conversion Time.....	244
13.4.4	MTU Trigger Input Timing.....	246
13.5	A/D Conversion Precision Definitions.....	246
13.6	Notes on Use.....	248
13.6.1	Analog Voltage Settings.....	248
13.6.2	Handling of Analog Input Pins.....	248
Section 14	Pin Function Controller .....	251
14.1	Overview.....	251
14.2	Register Configuration .....	251
14.3	Register Descriptions.....	252
14.3.1	Port A I/O Register H (PAIORH) .....	252
14.3.2	Port E I/O Register (PEIOR).....	252
14.3.3	Port E Control Register 2 (PECR2).....	253
Section 15	I/O Ports (I/O).....	255
15.1	Overview.....	255
15.2	Port A.....	255
15.2.1	Register Configuration .....	255

15.2.2	Port A Data Register H (PADRH).....	256
15.3	Port E.....	257
15.3.1	Register Configuration .....	257
15.3.2	Port E Data Register (PEDR) .....	258
<b>Section 16</b>	<b>RAM.....</b>	<b>261</b>
16.1	Overview.....	261
<b>Section 17</b>	<b>Electrical Characteristics.....</b>	<b>263</b>
17.1	Absolute Maximum Ratings.....	263
17.2	DC Characteristics.....	264
17.3	AC Characteristics.....	266
17.3.1	Clock Timing.....	266
17.3.2	Control Signal Timing.....	268
17.3.3	Bus Timing .....	270
17.3.4	Multifunction Timer Pulse Unit Timing .....	274
17.3.5	I/O Port Timing .....	275
17.3.6	Serial Communication Interface Timing.....	276
17.3.7	A/D Converter Timing .....	277
17.3.8	Measurement Conditions for AC Characteristic .....	278
17.4	A/D Converter Characteristics .....	279
<b>Appendix A</b>	<b>On-Chip Supporting Module Registers .....</b>	<b>281</b>
<b>Appendix B</b>	<b>Pin States.....</b>	<b>285</b>
B.1	Pin States .....	285
B.2	Bus Related Signal Pin States.....	286
<b>Appendix C</b>	<b>Package Dimensions .....</b>	<b>287</b>

# Section 1 SH7011 Overview

## 1.1 SH7011 Overview

The SH7011 CMOS single-chip microprocessors integrate a Hitachi-original architecture, high-speed CPU with peripheral functions required for system configuration.

The CPU has a RISC-type instruction set. Most instructions can be executed in one clock cycle, which greatly improves instruction execution speed. In addition, the 32-bit internal-bus architecture enhances data processing power. With this CPU, it has become possible to assemble low cost, high performance/high-functioning systems, even for applications that were previously impossible with microprocessors, such as real-time control, which demands high speeds.

In addition, the SH7011 includes on-chip peripheral functions necessary for system configuration, such as large-capacity ROM, timers, a serial communication interface (SCI), an A/D converter, an interrupt controller, and I/O ports. Memory or peripheral LSIs can be connected efficiently with an external memory access support function. This greatly reduces system cost.

### 1.1.1 SH7011 Features

#### CPU:

- Original Hitachi architecture
- 32-bit internal data bus
- General-register machine
  - Sixteen 32-bit general registers
  - Three 32-bit control registers
  - Four 32-bit system registers
- RISC-type instruction set
  - Instruction length: 16-bit fixed length for improved code efficiency
  - Load-store architecture (basic operations are executed between registers)
  - Delayed branch instructions reduce pipeline disruption during branch
  - Instruction set based on C language
- Instruction execution time: one instruction/cycle (50 ns/instruction at 20-MHz operation)
- Address space: Architecture supports 4 Gbytes
- On-chip multiplier: multiplication operations (32 bits  $\times$  32 bits  $\rightarrow$  64 bits) and multiplication/accumulation operations (32 bits  $\times$  32 bits + 64 bits  $\rightarrow$  64 bits) executed in two to four cycles
- Five-stage pipeline

### **Interrupt Controller (INTC):**

- Nine external interrupt pins (NMI,  $\overline{\text{IRQ0}}$  to  $\overline{\text{IRQ7}}$ )
- Twenty-two internal interrupt sources
- Sixteen programmable priority levels

### **Bus State Controller (BSC):**

- Supports external extended memory access
  - 16-bit external data bus
- Memory address space divided into four areas (four areas of SRAM space) with the following settable features:
  - Number of wait cycles (0 to 3 cycles)
  - Outputs chip-select signals for each area
- Wait cycles can be inserted using an external WAIT signal

### **Multifunction Timer Pulse Unit (MTU):**

- Maximum 6 types of waveform output or maximum 6 types of pulse I/O processing possible based on 16-bit timer, 3 channels
- 8 dual-use output compare/input capture registers
- 8 independent comparators
- 6 types of counter input clock
- Input capture function
- Pulse output mode
  - One shot, toggle, PWM
- Multiple counter synchronization function

### **Compare Match Timer (CMT) (Two Channels):**

- 16-bit free-running counter
- One compare register
- Generates an interrupt request upon compare match

### **8-Bit Timers (TIM1, TIM2) (Two Channels):**

- 8-bit interval timer function
- Interrupt generated on counter overflow (TIM1)
- Interrupt generated on compare match (TIM2)

### **Serial Communication Interface (SCI) (One Channel):**

- Asynchronous mode
- Can transmit and receive simultaneously (full duplex)
- On-chip dedicated baud rate generator
- Multiprocessor communication function

### **I/O Ports:**

- Input/output: 11

### **A/D Converter:**

- 10 bits × 7 channels
- Sample & hold function

### **Large Capacity On-Chip Memory:**

- RAM: 4 kbytes

### **Operating Modes:**

- Processing states
  - Program execution state
  - Exception processing state
- Power-down modes
  - Sleep mode

### **Clock Pulse Generator (CPG):**

- On-chip clock pulse generator

### **Package:**

- 100-pin plastic TQFP (TFP-100B)



## 1.2 Block Diagram

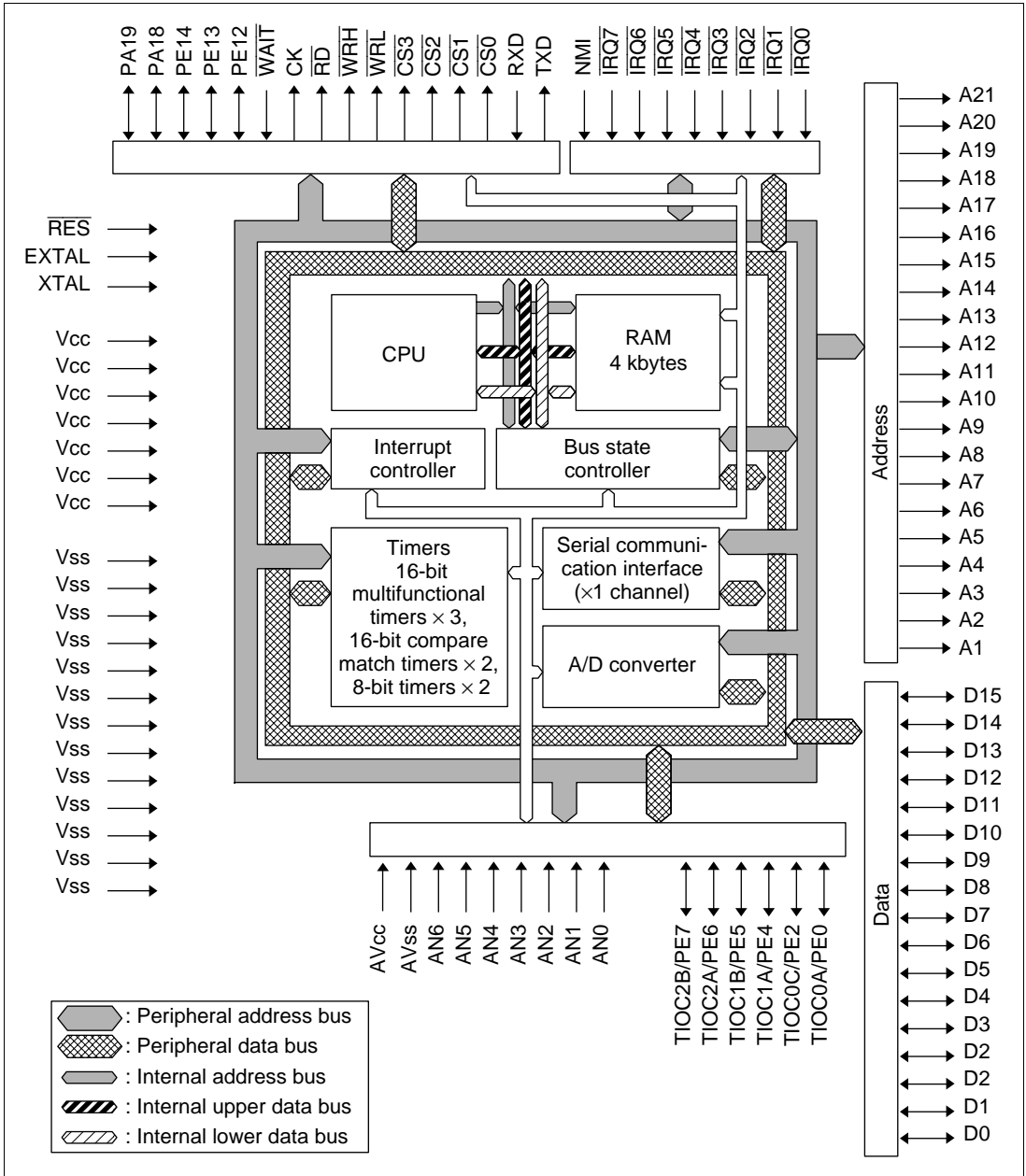
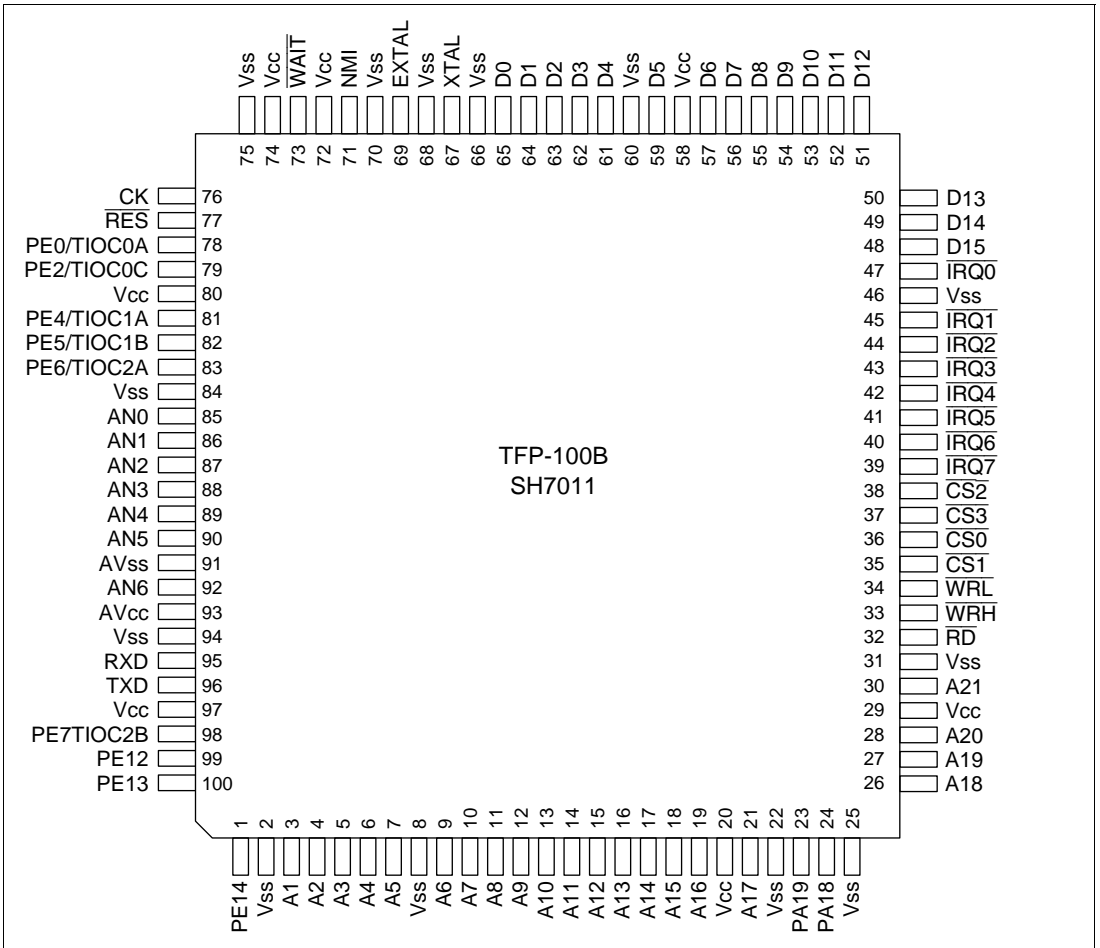


Figure 1.1 Block Diagram of the SH7011

# 1.3 Pin Arrangement and Pin Functions

## 1.3.1 Pin Arrangement



**Figure 1.2 SH7011 Pin Arrangement (TFP-100: Top View)**

## 1.3.2 Pin Functions

Table 1.1 lists the pin functions.

**Table 1.1 Pin Functions**

Classification	Symbol	I/O	Name	Function
Power supply	$V_{CC}$	I	Supply	Connects to power supply. Connect all $V_{CC}$ pins to the system supply. No operation will occur if there are any open pins.
	$V_{SS}$	I	Ground	Connects to ground. Connect all $V_{SS}$ pins to the system ground. No operation will occur if there are any open pins.
Clock	EXTAL	I	External clock	Connect a crystal oscillator. Also, an external clock can be input to the EXTAL pin.
	XTAL	I	Crystal	Connect a crystal oscillator.
	CK	O	System clock	Supplies the system clock to peripheral devices.
System control	$\overline{RES}$	I	Power-on reset	Power-on reset when low
Interrupts	NMI	I	Non-maskable interrupt	Non-maskable interrupt request pin. Enables selection of whether to accept on the rising or falling edge.
	$\overline{IRQ0}$ – $\overline{IRQ7}$	I	Interrupt requests 0–7	Maskable interrupt request pins. Allows selection of level input and edge input.
Address bus	A1–A21	O	Address bus	Outputs addresses.
Data bus	D0–D15	I/O	Data bus	16-bit bidirectional data bus.
Bus control	$\overline{CS0}$ to $\overline{CS3}$	O	Chip selects 0–3	Chip select signals for external memory or devices.
	$\overline{RD}$	O	Read	Indicates reading from an external device.
	$\overline{WRH}$	O	Upper write	Indicates writing the upper 8 bits (15–8) of external data.
	$\overline{WRL}$	O	Lower write	Indicates writing the lower 8 bits (7–0) of external data.
	$\overline{WAIT}$	I	Wait	Input causes insertion of wait cycles into the bus cycle during external space access.

**Table 1.1 Pin Functions (cont)**

<b>Classification</b>	<b>Symbol</b>	<b>I/O</b>	<b>Name</b>	<b>Function</b>
Multifunction timer/pulse unit (MTU)	TIOC0A	I/O	MTU input capture/output compare (channel 0)	Channel 0 input capture input/output compare output/PWM output pins.
	TIOC0C			
	TIOC1A	I/O	MTU input capture/output compare (channel 1)	Channel 1 input capture input/output compare output/PWM output pins.
	TIOC1B			
	TIOC2A	I/O	MTU input capture/output compare (channel 2)	Channel 2 input capture input/output compare output/PWM output pins.
	TIOC2B			
Serial communication interface (SCI)	TxD	O	Transmit data	Transmit data output pins.
	RxD	I	Receive data	Receive data input pins.
A/D Converter	AV <sub>CC</sub>	I	Analog supply	Analog supply; connected to V <sub>CC</sub> .
	AV <sub>SS</sub>	I	Analog ground	Analog supply; connected to V <sub>SS</sub> .
	AN0 to AN6	I	Analog input	Analog signal input pins.
I/O ports	PA18, 19	I/O	Port output enable	Input pin for port pin drive control when general use ports are established as output.
	PE0, 2, 4, to 7, 12 to 14	I/O	General purpose port	General purpose input/output port pins. Each bit can be designated for input/output.

# Section 2 CPU

## 2.1 Register Configuration

The register set consists of sixteen 32-bit general registers, three 32-bit control registers and four 32-bit system registers.

### 2.1.1 General Registers (Rn)

The sixteen 32-bit general registers (Rn) are numbered R0–R15. General registers are used for data processing and address calculation. R0 is also used as an index register. Several instructions have R0 fixed as their only usable register. R15 is used as the hardware stack pointer (SP). Saving and recovering the status register (SR) and program counter (PC) in exception processing is accomplished by referencing the stack using R15. Figure 2.1 shows the general registers.

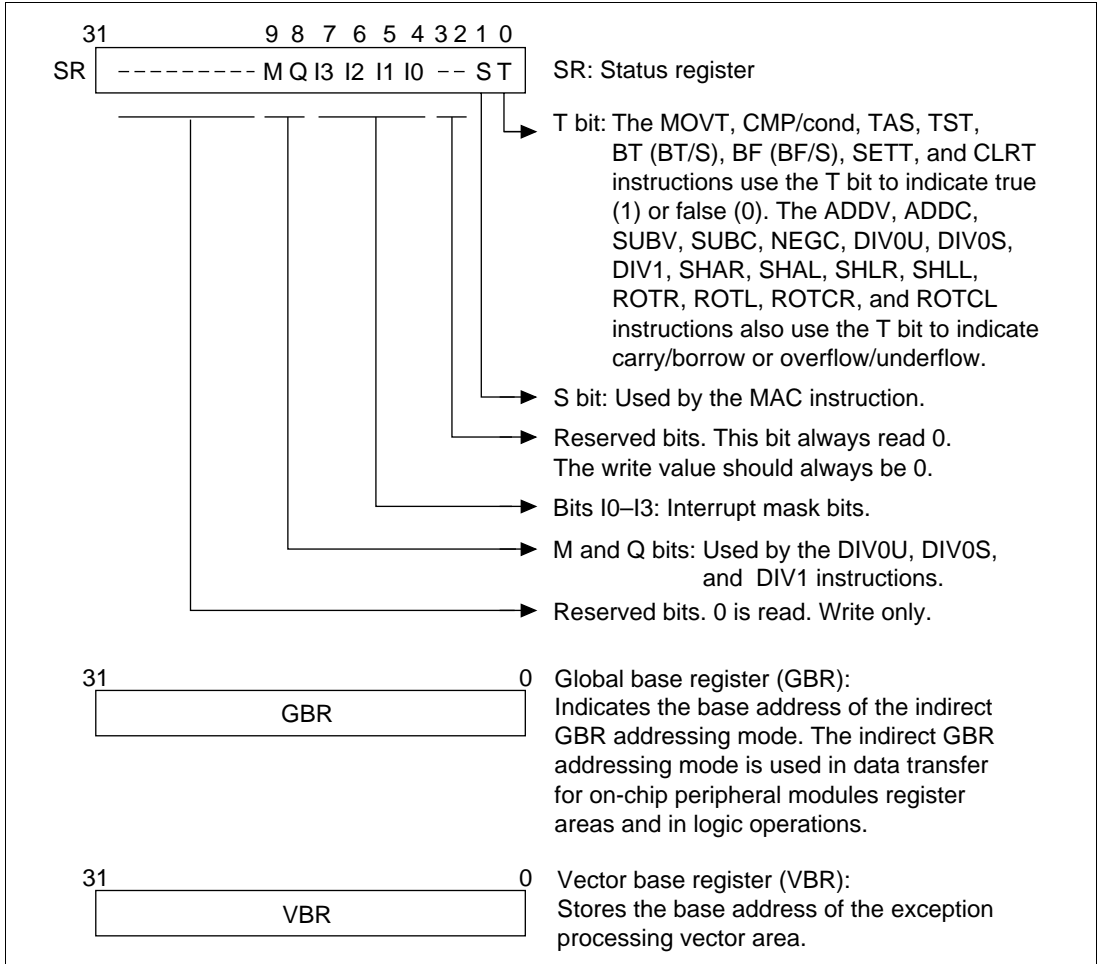
31	0
R0*1	
R1	
R2	
R3	
R4	
R5	
R6	
R7	
R8	
R9	
R10	
R11	
R12	
R13	
R14	
R15, SP (hardware stack pointer)*2	

- Notes:
1. R0 functions as an index register in the indirect indexed register addressing mode and indirect indexed GBR addressing mode. In some instructions, R0 functions as a fixed source register or destination register.
  2. R15 functions as a hardware stack pointer (SP) during exception processing.

**Figure 2.1 General Registers**

## 2.1.2 Control Registers

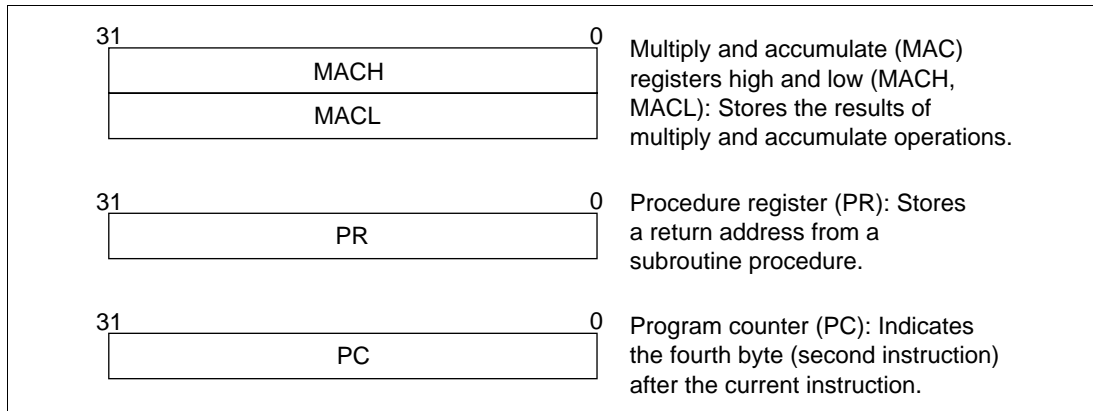
The 32-bit control registers consist of the 32-bit status register (SR), global base register (GBR), and vector base register (VBR). The status register indicates processing states. The global base register functions as a base address for the indirect GBR addressing mode to transfer data to the registers of on-chip peripheral modules. The vector base register functions as the base address of the exception processing vector area (including interrupts). Figure 2.2 shows a control register.



**Figure 2.2 Control Registers**

### 2.1.3 System Registers

System registers consist of four 32-bit registers: high and low multiply and accumulate registers (MACH and MACL), the procedure register (PR), and the program counter (PC). The multiply and accumulate registers store the results of multiply and accumulate operations. The procedure register stores the return address from the subroutine procedure. The program counter stores program addresses to control the flow of the processing. Figure 2.3 shows a system register.



**Figure 2.3 System Registers**

### 2.1.4 Initial Values of Registers

Table 2.1 lists the values of the registers after reset.

**Table 2.1 Initial Values of Registers**

<b>Classification</b>	<b>Register</b>	<b>Initial Value</b>
General registers	R0–R14	Undefined
	R15 (SP)	Value of the stack pointer in the vector address table
Control registers	SR	Bits I3–I0 are 1111 (H'F), reserved bits are 0, and other bits are undefined
	GBR	Undefined
	VBR	H'00000000
System registers	MACH, MACL, PR	Undefined
	PC	Value of the program counter in the vector address table

## 2.2 Data Formats

### 2.2.1 Data Format in Registers

Register operands are always longwords (32 bits). When the memory operand is only a byte (8 bits) or a word (16 bits), it is sign-extended into a longword when loaded into a register (figure 2.4).

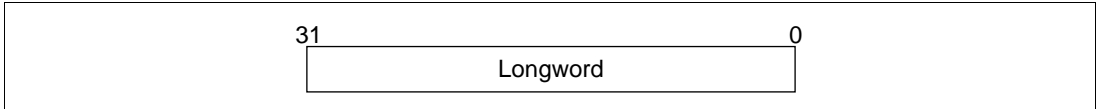


Figure 2.4 Longword Operand

### 2.2.2 Data Format in Memory

Memory data formats are classified into bytes, words, and longwords. Byte data can be accessed from any address, but an address error will occur if you try to access word data starting from an address other than  $2n$  or longword data starting from an address other than  $4n$ . In such cases, the data accessed cannot be guaranteed. The hardware stack area, referred to by the hardware stack pointer (SP, R15), uses only longword data starting from address  $4n$  because this area holds the program counter and status register (figure 2.5).

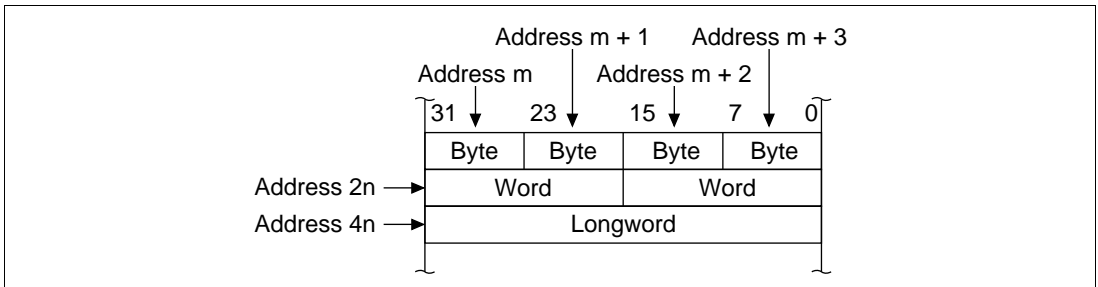


Figure 2.5 Byte, Word, and Longword Alignment

### 2.2.3 Immediate Data Format

Byte (8 bit) immediate data resides in an instruction code. Immediate data accessed by the MOV, ADD, and CMP/EQ instructions is sign-extended and handled in registers as longword data. Immediate data accessed by the TST, AND, OR, and XOR instructions is zero-extended and handled as longword data. Consequently, AND instructions with immediate data always clear the upper 24 bits of the destination register.



Word or longword immediate data is not located in the instruction code, but instead is stored in a memory table. An immediate data transfer instruction (MOV) accesses the memory table using the PC relative addressing mode with displacement.

## 2.3 Instruction Features

### 2.3.1 RISC-Type Instruction Set

All instructions are RISC type. This section details their functions.

**16-Bit Fixed Length:** All instructions are 16 bits long, increasing program code efficiency.

**One Instruction per Cycle:** The microprocessor can execute basic instructions in one cycle using the pipeline system. Instructions are executed in 50 ns at 20 MHz.

**Data Length:** Longword is the standard data length for all operations. Memory can be accessed in bytes, words, or longwords. Byte or word data accessed from memory is sign-extended and handled as longword data. Immediate data is sign-extended for arithmetic operations or zero-extended for logic operations. It also is handled as longword data (table 2.2).

**Table 2.2 Sign Extension of Word Data**

SH7011 CPU	Description	Example of Conventional CPU
MOV.W @ (disp, PC), R1	Data is sign-extended to 32 bits, and R1 becomes H'00001234. It is next operated upon by an ADD instruction.	ADD.W #H'1234, R0
ADD R1, R0		
..... .DATA.W H'1234		

Note: @ (disp, PC) accesses the immediate data.

**Load-Store Architecture:** Basic operations are executed between registers. For operations that involve memory access, data is loaded to the registers and executed (load-store architecture). Instructions such as AND that manipulate bits, however, are executed directly in memory.

**Delayed Branch Instructions:** Unconditional branch instructions are delayed. Executing the instruction that follows the branch instruction and then branching reduces pipeline disruption during branching (table 2.3). There are two types of conditional branch instructions: delayed branch instructions and ordinary branch instructions.

**Table 2.3 Delayed Branch Instructions**

SH7011 CPU		Description	Example of Conventional CPU	
BRA	TRGET	Executes an ADD before branching to TRGET	ADD.W	R1,R0
ADD	R1,R0		BRA	TRGET

**Multiplication/Accumulation Operation:** 16-bit  $\times$  16-bit  $\rightarrow$  32-bit multiplication operations are executed in one to two cycles. 16-bit  $\times$  16-bit + 64-bit  $\rightarrow$  64-bit multiplication/accumulation operations are executed in two to three cycles. 32-bit  $\times$  32-bit  $\rightarrow$  64-bit and 32-bit  $\times$  32-bit + 64bit  $\rightarrow$  64-bit multiplication/accumulation operations are executed in two to four cycles.

**T Bit:** The T bit in the status register changes according to the result of the comparison, and in turn is the condition (true/false) that determines if the program will branch. The number of instructions that change the T bit is kept to a minimum to improve the processing speed (table 2.4).

**Table 2.4 T Bit**

SH7011 CPU		Description	Example of Conventional CPU	
CMP/GE	R1,R0	T bit is set when R0 $\geq$ R1. The program branches to TRGET0 when R0 $\geq$ R1 and to TRGET1 when R0 < R1.	CMP.W	R1,R0
BT	TRGET0		BGE	TRGET0
BF	TRGET1		BLT	TRGET1
ADD	#1,R0	T bit is not changed by ADD. T bit is set when R0 = 0. The program branches if R0 = 0.	SUB.W	#1,R0
CMP/EQ	#0,R0		BEQ	TRGET
BT	TRGET			

**Immediate Data:** Byte (8 bit) immediate data resides in instruction code. Word or longword immediate data is not input via instruction codes but is stored in a memory table. An immediate data transfer instruction (MOV) accesses the memory table using the PC relative addressing mode with displacement (table 2.5).

**Table 2.5 Immediate Data Accessing**

Classification	SH7011 CPU		Example of Conventional CPU
8-bit immediate	MOV	#H' 12,R0	MOV.B #H' 12,R0
16-bit immediate	MOV.W	@(disp,PC),R0 .....	MOV.W #H' 1234,R0
	.DATA.W	H' 1234	
32-bit immediate	MOV.L	@(disp,PC),R0 .....	MOV.L #H' 12345678,R0
	.DATA.L	H' 12345678	

Note: @(disp, PC) accesses the immediate data.

**Absolute Address:** When data is accessed by absolute address, the value already in the absolute address is placed in the memory table. Loading the immediate data when the instruction is executed transfers that value to the register and the data is accessed in the indirect register addressing mode (table 2.6).

**Table 2.6 Absolute Address Accessing**

Classification	SH7011 CPU		Example of Conventional CPU
Absolute address	MOV.L	@(disp,PC),R1	MOV.B @H' 12345678,R0
	MOV.B	@R1,R0 .....	
	.DATA.L	H' 12345678	

Note: @(disp,PC) accesses the immediate data.

**16-Bit/32-Bit Displacement:** When data is accessed by 16-bit or 32-bit displacement, the pre-existing displacement value is placed in the memory table. Loading the immediate data when the instruction is executed transfers that value to the register and the data is accessed in the indirect indexed register addressing mode (table 2.7).

**Table 2.7 Displacement Accessing**


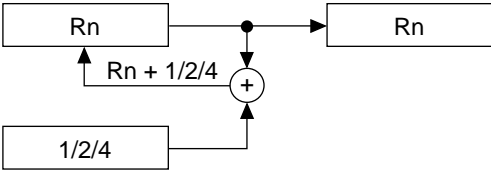
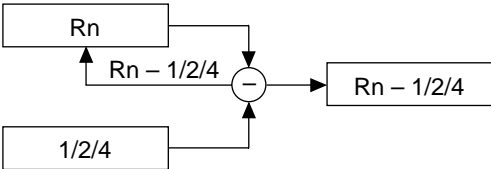
Classification	SH7011 CPU		Example of Conventional CPU
16-bit displacement	MOV.W	@(disp,PC),R0	MOV.W @(H' 1234,R1),R2
	MOV.W	@(R0,R1),R2 .....	
	.DATA.W	H' 1234	

Note: @(disp,PC) accesses the immediate data.

## 2.3.2 Addressing Modes

Table 2.8 describes addressing modes and effective address calculation.

**Table 2.8 Addressing Modes and Effective Addresses**

Addressing Mode	Instruction Format	Effective Addresses Calculation	Equation
Direct register addressing	Rn	The effective address is register Rn. (The operand is the contents of register Rn.)	—
Indirect register addressing	@Rn	The effective address is the content of register Rn. 	Rn
Post-increment indirect register addressing	@Rn+	The effective address is the content of register Rn. A constant is added to the content of Rn after the instruction is executed. 1 is added for a byte operation, 2 for a word operation, and 4 for a longword operation. 	Rn (After the instruction executes) Byte: Rn + 1 → Rn Word: Rn + 2 → Rn Longword: Rn + 4 → Rn
Pre-decrement indirect register addressing	@-Rn	The effective address is the value obtained by subtracting a constant from Rn. 1 is subtracted for a byte operation, 2 for a word operation, and 4 for a longword operation. 	Byte: Rn - 1 → Rn Word: Rn - 2 → Rn Longword: Rn - 4 → Rn (Instruction executed with Rn after calculation)

**Table 2.8 Addressing Modes and Effective Addresses (cont)**

Addressing Mode	Instruction Format	Effective Addresses Calculation	Equation
Indirect register addressing with displacement	@(disp:4, Rn)	<p>The effective address is Rn plus a 4-bit displacement (disp). The value of disp is zero-extended, and remains the same for a byte operation, is doubled for a word operation, and is quadrupled for a longword operation.</p>	<p>Byte: <math>Rn + disp</math></p> <p>Word: <math>Rn + disp \times 2</math></p> <p>Longword: <math>Rn + disp \times 4</math></p>
Indirect indexed register addressing	@(R0, Rn)	<p>The effective address is the Rn value plus R0.</p>	$Rn + R0$
Indirect GBR addressing with displacement	@(disp:8, GBR)	<p>The effective address is the GBR value plus an 8-bit displacement (disp). The value of disp is zero-extended, and remains the same for a byte operation, is doubled for a word operation, and is quadrupled for a longword operation.</p>	<p>Byte: <math>GBR + disp</math></p> <p>Word: <math>GBR + disp \times 2</math></p> <p>Longword: <math>GBR + disp \times 4</math></p>

**Table 2.8 Addressing Modes and Effective Addresses (cont)**

Addressing Mode	Instruction Format	Effective Addresses Calculation	Equation
Indirect indexed GBR addressing	@(R0, GBR)	The effective address is the GBR value plus the R0.	$GBR + R0$
PC relative addressing with displacement	@(disp:8, PC)	The effective address is the PC value plus an 8-bit displacement (disp). The value of disp is zero-extended, and is doubled for a word operation, and quadrupled for a longword operation. For a longword operation, the lowest two bits of the PC value are masked.	Word: $PC + disp \times 2$ Longword: $PC \& H'FFFFFFC + disp \times 4$

**Table 2.8 Addressing Modes and Effective Addresses (cont)**

Addressing Mode	Instruction Format	Effective Addresses Calculation	Equation
PC relative addressing	disp:8	The effective address is the PC value sign-extended with an 8-bit displacement (disp), doubled, and added to the PC value.	$PC + disp \times 2$
	disp:12	The effective address is the PC value sign-extended with a 12-bit displacement (disp), doubled, and added to the PC value.	$PC + disp \times 2$
	Rn	The effective address is the register PC value plus Rn.	$PC + Rn$
Immediate addressing	#imm:8	The 8-bit immediate data (imm) for the TST, AND, OR, and XOR instructions are zero-extended.	—
	#imm:8	The 8-bit immediate data (imm) for the MOV, ADD, and CMP/EQ instructions are sign-extended.	—
	#imm:8	The 8-bit immediate data (imm) for the TRAPA instruction is zero-extended and is quadrupled.	—

### 2.3.3 Instruction Format

Table 2.9 lists the instruction formats for the source operand and the destination operand. The meaning of the operand depends on the instruction code. The symbols are used as follows:

- xxxx: Instruction code
- mmmm: Source register
- nnnn: Destination register
- iiiii: Immediate data
- dddd: Displacement

**Table 2.9 Instruction Formats**

Instruction Formats	Source Operand	Destination Operand	Example
0 format <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 5px auto;"> <span style="float: left;">15</span> <span style="float: right;">0</span> <div style="display: flex; justify-content: space-around; width: 100%;"> <span>xxxx</span> <span>xxxx</span> <span>xxxx</span> <span>xxxx</span> </div> </div>	—	—	NOP
n format <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 5px auto;"> <span style="float: left;">15</span> <span style="float: right;">0</span> <div style="display: flex; justify-content: space-around; width: 100%;"> <span>xxxx</span> <span>nnnn</span> <span>xxxx</span> <span>xxxx</span> </div> </div>	—	nnnn: Direct register	MOVT Rn
	Control register or system register	nnnn: Direct register	STS MACH, Rn
	Control register or system register	nnnn: Indirect pre-decrement register	STC.L SR, @-Rn
m format <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 5px auto;"> <span style="float: left;">15</span> <span style="float: right;">0</span> <div style="display: flex; justify-content: space-around; width: 100%;"> <span>xxxx</span> <span>mmmm</span> <span>xxxx</span> <span>xxxx</span> </div> </div>	mmmm: Direct register	Control register or system register	LDC Rm, SR
	mmmm: Indirect post-increment register	Control register or system register	LDC.L @Rm+, SR
	mmmm: Direct register	—	JMP @Rm
	mmmm: PC relative using Rm	—	BRAF Rm



**Table 2.9 Instruction Formats (cont)**

Instruction Formats	Source Operand	Destination Operand	Example				
nm format	mmmm: Direct register	nnnn: Direct register	ADD Rm, Rn				
15 <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="width: 25px; text-align: center;">xxxx</td> <td style="width: 25px; text-align: center;">nnnn</td> <td style="width: 25px; text-align: center;">mmmm</td> <td style="width: 25px; text-align: center;">xxxx</td> </tr> </table> 0	xxxx	nnnn	mmmm	xxxx	mmmm: Direct register	nnnn: Indirect register	MOV.L Rm, @Rn
xxxx	nnnn	mmmm	xxxx				
	mmmm: Indirect post-increment register (multiply/accumulate)	MACH, MACL	MAC.W @Rm+, @Rn+				
	nnnn*: Indirect post-increment register (multiply/accumulate)						
	mmmm: Indirect post-increment register	nnnn: Direct register	MOV.L @Rm+, Rn				
	mmmm: Direct register	nnnn: Indirect pre-decrement register	MOV.L Rm, @-Rn				
	mmmm: Direct register	nnnn: Indirect indexed register	MOV.L Rm, @(R0, Rn)				
md format	mmmmdddd: indirect register with displacement	R0 (Direct register)	MOV.B @(disp, Rm), R0				
15 <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="width: 25px; text-align: center;">xxxx</td> <td style="width: 25px; text-align: center;">xxxx</td> <td style="width: 25px; text-align: center;">mmmm</td> <td style="width: 25px; text-align: center;">dddd</td> </tr> </table> 0	xxxx	xxxx	mmmm	dddd	R0 (Direct register)	nnnndddd: Indirect register with displacement	MOV.B R0, @(disp, Rn)
xxxx	xxxx	mmmm	dddd				
nd4 format							
15 <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="width: 25px; text-align: center;">xxxx</td> <td style="width: 25px; text-align: center;">xxxx</td> <td style="width: 25px; text-align: center;">nnnn</td> <td style="width: 25px; text-align: center;">dddd</td> </tr> </table> 0	xxxx	xxxx	nnnn	dddd			
xxxx	xxxx	nnnn	dddd				
nmd format	mmmm: Direct register	nnnndddd: Indirect register with displacement	MOV.L Rm, @(disp, Rn)				
15 <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="width: 25px; text-align: center;">xxxx</td> <td style="width: 25px; text-align: center;">nnnn</td> <td style="width: 25px; text-align: center;">mmmm</td> <td style="width: 25px; text-align: center;">dddd</td> </tr> </table> 0	xxxx	nnnn	mmmm	dddd	mmmmdddd: Indirect register with displacement	nnnn: Direct register	MOV.L @(disp, Rm), Rn
xxxx	nnnn	mmmm	dddd				

Note: In multiply/accumulate instructions, nnnn is the source register.

**Table 2.9 Instruction Formats (cont)**

Instruction Formats	Source Operand	Destination Operand	Example
d format 15 <span style="float: right;">0</span> <div style="border: 1px solid black; padding: 5px; width: fit-content;">             xxxx    xxxx    dddd    dddd           </div>	ddddddd: Indirect GBR with displacement	R0 (Direct register)	MOV.L @(disp,GBR),R0
	R0(Direct register)	ddddddd: Indirect GBR with displacement	MOV.L R0,@(disp,GBR)
	ddddddd: PC relative with displacement	R0 (Direct register)	MOVA @(disp,PC),R0
	ddddddd: PC relative	—	BF    label
d12 format 15 <span style="float: right;">0</span> <div style="border: 1px solid black; padding: 5px; width: fit-content;">             xxxx    dddd    dddd    dddd           </div>	dddddddddd: PC relative	—	BRA    label (label = disp + PC)
nd8 format 15 <span style="float: right;">0</span> <div style="border: 1px solid black; padding: 5px; width: fit-content;">             xxxx    nnnn    dddd    dddd           </div>	ddddddd: PC relative with displacement	nnnn: Direct register	MOV.L @(disp,PC),Rn
i format 15 <span style="float: right;">0</span> <div style="border: 1px solid black; padding: 5px; width: fit-content;">             xxxx    xxxx    iiii    iiii           </div>	iiiiii: Immediate	Indirect indexed GBR	AND.B #imm,@(R0,GBR)
	iiiiii: Immediate	R0 (Direct register)	AND    #imm,R0
	iiiiii: Immediate	—	TRAPA    #imm
ni format 15 <span style="float: right;">0</span> <div style="border: 1px solid black; padding: 5px; width: fit-content;">             xxxx    nnnn    iiii    iiii           </div>	iiiiii: Immediate	nnnn: Direct register	ADD    #imm,Rn

## 2.4 Instruction Set by Classification

**Table 2.10 Classification of Instructions**

Classification	Types	Operation		No. of Instructions
		Code	Function	
Data transfer	5	MOV	Data transfer, immediate data transfer, peripheral module data transfer, structure data transfer	39
		MOVA	Effective address transfer	
		MOVT	T bit transfer	
		SWAP	Swap of upper and lower bytes	
		XTRCT	Extraction of the middle of registers connected	
Arithmetic operations	21	ADD	Binary addition	33
		ADDC	Binary addition with carry	
		ADDV	Binary addition with overflow check	
		CMP/cond	Comparison	
		DIV1	Division	
		DIV0S	Initialization of signed division	
		DIV0U	Initialization of unsigned division	
		DMULS	Signed double-length multiplication	
		DMULU	Unsigned double-length multiplication	
		DT	Decrement and test	
		EXTS	Sign extension	
		EXTU	Zero extension	
		MAC	Multiply/accumulate, double-length multiply/accumulate operation	
		MUL	Double-length multiply operation	
		MULS	Signed multiplication	
		MULU	Unsigned multiplication	
		NEG	Negation	
		NEGC	Negation with borrow	
		SUB	Binary subtraction	
		SUBC	Binary subtraction with borrow	
SUBV	Binary subtraction with underflow			

**Table 2.10 Classification of Instructions (cont)**

Classification	Types	Operation		No. of Instructions
		Code	Function	
Logic operations	6	AND	Logical AND	14
		NOT	Bit inversion	
		OR	Logical OR	
		TAS	Memory test and bit set	
		TST	Logical AND and T bit set	
		XOR	Exclusive OR	
Shift	10	ROTL	One-bit left rotation	14
		ROTR	One-bit right rotation	
		ROTCL	One-bit left rotation with T bit	
		ROTCR	One-bit right rotation with T bit	
		SHAL	One-bit arithmetic left shift	
		SHAR	One-bit arithmetic right shift	
		SHLL	One-bit logical left shift	
		SHLLn	n-bit logical left shift	
		SHLR	One-bit logical right shift	
		SHLRn	n-bit logical right shift	
Branch	9	BF	Conditional branch, conditional branch with delay (Branch when T = 0)	11
		BT	Conditional branch, conditional branch with delay (Branch when T = 1)	
		BRA	Unconditional branch	
		BRAF	Unconditional branch	
		BSR	Branch to subroutine procedure	
		BSRF	Branch to subroutine procedure	
		JMP	Unconditional branch	
		JSR	Branch to subroutine procedure	
		RTS	Return from subroutine procedure	

**Table 2.10 Classification of Instructions (cont)**

Classification	Types	Operation		No. of Instructions
		Code	Function	
System control	11	CLRT	T bit clear	31
		CLRMAC	MAC register clear	
		LDC	Load to control register	
		LDS	Load to system register	
		NOP	No operation	
		RTE	Return from exception processing	
		SETT	T bit set	
		SLEEP	Shift into power-down mode	
		STC	Storing control register data	
		STS	Storing system register data	
		TRAPA	Trap exception handling	
Total:		62		142

Table 2.11 shows the format used in tables 2.12 to 2.17, which list instruction codes, operation, and execution states in order by classification.

**Table 2.11 Instruction Code Format**

Item	Format	Explanation
Instruction	OP.Sz SRC,DEST	OP: Operation code Sz: Size (B: byte, W: word, or L: longword) SRC: Source DEST: Destination Rm: Source register Rn: Destination register imm: Immediate data disp: Displacement* <sup>1</sup>
Instruction code	MSB ↔ LSB	m m m m: Source register n n n n: Destination register 0000: R0 0001: R1 . . . 1111: R15 iiii: Immediate data dddd: Displacement
Operation	→, ←	Direction of transfer
	(xx)	Memory operand
	M/Q/T	Flag bits in the SR
	&	Logical AND of each bit
		Logical OR of each bit
	^	Exclusive OR of each bit
	~	Logical NOT of each bit
	<<n	n-bit left shift
	>>n	n-bit right shift
Execution cycles	—	Value when no wait states are inserted* <sup>2</sup>
T bit	—	Value of T bit after instruction is executed. An em-dash (—) in the column means no change.

Notes: 1. Depending on the operand size, displacement is scaled  $\times 1$ ,  $\times 2$ , or  $\times 4$ . For details, see the *SH-1/SH-2/SH-DSP Programming Manual*.

2. Instruction execution cycles: The execution cycles shown in the table are minimums. The actual number of cycles may be increased when (1) contention occurs between instruction fetches and data access, or (2) when the destination register of the load instruction (memory → register) and the register used by the next instruction are the same.

**Table 2.12 Data Transfer Instructions**

Instruction	Instruction Code	Operation	Execution Cycles	T Bit
MOV #imm,Rn	1110nnnniiiiiii	#imm → Sign extension → Rn	1	—
MOV.W @(disp,PC),Rn	1001nnnnddddddd	(disp × 2 + PC) → Sign extension → Rn	1	—
MOV.L @(disp,PC),Rn	1101nnnnddddddd	(disp × 4 + PC) → Rn	1	—
MOV Rm,Rn	0110nnnnnnmm0011	Rm → Rn	1	—
MOV.B Rm,@Rn	0010nnnnnnmm0000	Rm → (Rn)	1	—
MOV.W Rm,@Rn	0010nnnnnnmm0001	Rm → (Rn)	1	—
MOV.L Rm,@Rn	0010nnnnnnmm0010	Rm → (Rn)	1	—
MOV.B @Rm,Rn	0110nnnnnnmm0000	(Rm) → Sign extension → Rn	1	—
MOV.W @Rm,Rn	0110nnnnnnmm0001	(Rm) → Sign extension → Rn	1	—
MOV.L @Rm,Rn	0110nnnnnnmm0010	(Rm) → Rn	1	—
MOV.B Rm,@-Rn	0010nnnnnnmm0100	Rn-1 → Rn, Rm → (Rn)	1	—
MOV.W Rm,@-Rn	0010nnnnnnmm0101	Rn-2 → Rn, Rm → (Rn)	1	—
MOV.L Rm,@-Rn	0010nnnnnnmm0110	Rn-4 → Rn, Rm → (Rn)	1	—
MOV.B @Rm+,Rn	0110nnnnnnmm0100	(Rm) → Sign extension → Rn, Rm + 1 → Rm	1	—
MOV.W @Rm+,Rn	0110nnnnnnmm0101	(Rm) → Sign extension → Rn, Rm + 2 → Rm	1	—
MOV.L @Rm+,Rn	0110nnnnnnmm0110	(Rm) → Rn, Rm + 4 → Rm	1	—
MOV.B R0,@(disp,Rn)	1000000nnnndddd	R0 → (disp + Rn)	1	—
MOV.W R0,@(disp,Rn)	10000001nnnndddd	R0 → (disp × 2 + Rn)	1	—
MOV.L Rm,@(disp,Rn)	0001nnnnnnmmddd	Rm → (disp × 4 + Rn)	1	—
MOV.B @(disp,Rm),R0	10000100nnmmddd	(disp + Rm) → Sign extension → R0	1	—
MOV.W @(disp,Rm),R0	10000101mmmmddd	(disp × 2 + Rm) → Sign extension → R0	1	—
MOV.L @(disp,Rm),Rn	0101nnnnnnmmddd	(disp × 4 + Rm) → Rn	1	—
MOV.B Rm,@(R0,Rn)	0000nnnnnnmm0100	Rm → (R0 + Rn)	1	—

**Table 2.12 Data Transfer Instructions (cont)**

<b>Instruction</b>	<b>Instruction Code</b>	<b>Operation</b>	<b>Execution Cycles</b>	<b>T Bit</b>
MOV.W Rm,@(R0,Rn)	0000nnnnmmmm0101	Rm → (R0 + Rn)	1	—
MOV.L Rm,@(R0,Rn)	0000nnnnmmmm0110	Rm → (R0 + Rn)	1	—
MOV.B @(R0,Rm),Rn	0000nnnnmmmm1100	(R0 + Rm) → Sign extension → Rn	1	—
MOV.W @(R0,Rm),Rn	0000nnnnmmmm1101	(R0 + Rm) → Sign extension → Rn	1	—
MOV.L @(R0,Rm),Rn	0000nnnnmmmm1110	(R0 + Rm) → Rn	1	—
MOV.B R0,@(disp,GBR)	11000000dddddddd	R0 → (disp + GBR)	1	—
MOV.W R0,@(disp,GBR)	11000001dddddddd	R0 → (disp × 2 + GBR)	1	—
MOV.L R0,@(disp,GBR)	11000010dddddddd	R0 → (disp × 4 + GBR)	1	—
MOV.B @(disp,GBR),R0	11000100dddddddd	(disp + GBR) → Sign extension → R0	1	—
MOV.W @(disp,GBR),R0	11000101dddddddd	(disp × 2 + GBR) → Sign extension → R0	1	—
MOV.L @(disp,GBR),R0	11000110dddddddd	(disp × 4 + GBR) → R0	1	—
MOVA @(disp,PC),R0	11000111dddddddd	disp × 4 + PC → R0	1	—
MOVT Rn	0000nnnn00101001	T → Rn	1	—
SWAP.B Rm,Rn	0110nnnnmmmm1000	Rm → Swap the bottom two bytes → Rn	1	—
SWAP.W Rm,Rn	0110nnnnmmmm1001	Rm → Swap two consecutive words → Rn	1	—
XTRCT Rm,Rn	0010nnnnmmmm1101	Rm: Middle 32 bits of Rn → Rn	1	—



**Table 2.13 Arithmetic Operation Instructions**

Instruction		Instruction Code	Operation	Execution Cycles	T Bit
ADD	Rm, Rn	0011nnnnnnmmmm1100	$Rn + Rm \rightarrow Rn$	1	—
ADD	#imm, Rn	0111nnnnniiiiiii	$Rn + imm \rightarrow Rn$	1	—
ADDC	Rm, Rn	0011nnnnnnmmmm1110	$Rn + Rm + T \rightarrow Rn$ , Carry $\rightarrow T$	1	Carry
ADDV	Rm, Rn	0011nnnnnnmmmm1111	$Rn + Rm \rightarrow Rn$ , Overflow $\rightarrow T$	1	Overflow
CMP/EQ	#imm, R0	10001000iiiiiii	If $R0 = imm$ , $1 \rightarrow T$	1	Comparison result
CMP/EQ	Rm, Rn	0011nnnnnnmmmm0000	If $Rn = Rm$ , $1 \rightarrow T$	1	Comparison result
CMP/HS	Rm, Rn	0011nnnnnnmmmm0010	If RnRm with unsigned data, $1 \rightarrow T$	1	Comparison result
CMP/GE	Rm, Rn	0011nnnnnnmmmm0011	If Rn Rm with signed data, $1 \rightarrow T$	1	Comparison result
CMP/HI	Rm, Rn	0011nnnnnnmmmm0110	If $Rn > Rm$ with unsigned data, $1 \rightarrow T$	1	Comparison result
CMP/GT	Rm, Rn	0011nnnnnnmmmm0111	If $Rn > Rm$ with signed data, $1 \rightarrow T$	1	Comparison result
CMP/PL	Rn	0100nnnn00010101	If $Rn > 0$ , $1 \rightarrow T$	1	Comparison result
CMP/PZ	Rn	0100nnnn00010001	If $Rn = 0$ , $1 \rightarrow T$	1	Comparison result
CMP/STR	Rm, Rn	0010nnnnnnmmmm1100	If Rn and Rm have an equivalent byte, $1 \rightarrow T$	1	Comparison result
DIV1	Rm, Rn	0011nnnnnnmmmm0100	Single-step division (Rn/Rm)	1	Calculation result
DIV0S	Rm, Rn	0010nnnnnnmmmm0111	MSB of Rn $\rightarrow Q$ , MSB of Rm $\rightarrow M$ , $M \wedge Q \rightarrow T$	1	Calculation result
DIV0U		0000000000011001	$0 \rightarrow M/Q/T$	1	0

**Table 2.13 Arithmetic Operation Instructions (cont)**

Instruction	Instruction Code	Operation	Execution Cycles	T Bit
DMULS.L Rm, Rn	0011nnnnnnmm1101	Signed operation of $Rn \times Rm \rightarrow MACH, MACL$ $32 \times 32 \rightarrow 64$ bit	2 to 4*	—
DMULU.L Rm, Rn	0011nnnnnnmm0101	Unsigned operation of $Rn \times Rm \rightarrow MACH, MACL$ $32 \times 32 \rightarrow 64$ bit	2 to 4*	—
DT Rn	0100nnnn00010000	$Rn - 1 \rightarrow Rn$ , when $Rn$ is 0, $1 \rightarrow T$ . When $Rn$ is nonzero, $0 \rightarrow T$	1	Comparison result
EXTS.B Rm, Rn	0110nnnnnnmm1110	A byte in $Rm$ is sign-extended $\rightarrow Rn$	1	—
EXTS.W Rm, Rn	0110nnnnnnmm1111	A word in $Rm$ is sign-extended $\rightarrow Rn$	1	—
EXTU.B Rm, Rn	0110nnnnnnmm1100	A byte in $Rm$ is zero-extended $\rightarrow Rn$	1	—
EXTU.W Rm, Rn	0110nnnnnnmm1101	A word in $Rm$ is zero-extended $\rightarrow Rn$	1	—
MAC.L @Rm+, @Rn+	0000nnnnnnmm1111	Signed operation of $(Rn) \times (Rm) + MAC \rightarrow MAC$ $32 \times 32 \rightarrow 64$ bit	3/(2 to 4)*	—
MAC.W @Rm+, @Rn+	0100nnnnnnmm1111	Signed operation of $(Rn) \times (Rm) + MAC \rightarrow MAC$ $16 \times 16 + 64 \rightarrow 64$ bit	3/(2)*	—
MUL.L Rm, Rn	0000nnnnnnmm0111	$Rn \times Rm \rightarrow MACL$ , $32 \times 32 \rightarrow 32$ bit	2 to 4*	—
MULS.W Rm, Rn	0010nnnnnnmm1111	Signed operation of $Rn \times Rm \rightarrow MAC$ $16 \times 16 \rightarrow 32$ bit	1 to 3*	—
MULU.W Rm, Rn	0010nnnnnnmm1110	Unsigned operation of $Rn \times Rm \rightarrow MAC$ $16 \times 16 \rightarrow 32$ bit	1 to 3*	—
NEG Rm, Rn	0110nnnnnnmm1011	$0 - Rm \rightarrow Rn$	1	—
NEGC Rm, Rn	0110nnnnnnmm1010	$0 - Rm - T \rightarrow Rn$ , Borrow $\rightarrow T$	1	Borrow

**Table 2.13 Arithmetic Operation Instructions (cont)**

Instruction		Instruction Code	Operation	Execution Cycles	T Bit
SUB	Rm, Rn	0011nnnnmmmm1000	$Rn - Rm \rightarrow Rn$	1	—
SUBC	Rm, Rn	0011nnnnmmmm1010	$Rn - Rm - T \rightarrow Rn$ , Borrow $\rightarrow T$	1	Borrow
SUBV	Rm, Rn	0011nnnnmmmm1011	$Rn - Rm \rightarrow Rn$ , Underflow $\rightarrow T$	1	Overflow

Note: The normal minimum number of execution cycles. (The number in parentheses is the number of cycles when there is contention with following instructions.)

**Table 2.14 Logic Operation Instructions**

Instruction		Instruction Code	Operation	Execution Cycles	T Bit
AND	Rm, Rn	0010nnnnmmmm1001	$Rn \& Rm \rightarrow Rn$	1	—
AND	#imm, R0	11001001iiiiiii	$R0 \& imm \rightarrow R0$	1	—
AND.B	#imm, @(R0, GBR)	11001101iiiiiii	$(R0 + GBR) \& imm \rightarrow$ $(R0 + GBR)$	3	—
NOT	Rm, Rn	0110nnnnmmmm0111	$\sim Rm \rightarrow Rn$	1	—
OR	Rm, Rn	0010nnnnmmmm1011	$Rn   Rm \rightarrow Rn$	1	—
OR	#imm, R0	11001011iiiiiii	$R0   imm \rightarrow R0$	1	—
OR.B	#imm, @(R0, GBR)	11001111iiiiiii	$(R0 + GBR)   imm \rightarrow$ $(R0 + GBR)$	3	—
TAS.B	@Rn	0100nnnn00011011	If (Rn) is 0, $1 \rightarrow T$ ; $1 \rightarrow$ MSB of (Rn)	4	Test result
TST	Rm, Rn	0010nnnnmmmm1000	$Rn \& Rm$ ; if the result is 0, $1 \rightarrow T$	1	Test result
TST	#imm, R0	11001000iiiiiii	$R0 \& imm$ ; if the result is 0, $1 \rightarrow T$	1	Test result
TST.B	#imm, @(R0, GBR)	11001100iiiiiii	$(R0 + GBR) \& imm$ ; if the result is 0, $1 \rightarrow T$	3	Test result
XOR	Rm, Rn	0010nnnnmmmm1010	$Rn \wedge Rm \rightarrow Rn$	1	—
XOR	#imm, R0	11001010iiiiiii	$R0 \wedge imm \rightarrow R0$	1	—
XOR.B	#imm, @(R0, GBR)	11001110iiiiiii	$(R0 + GBR) \wedge imm \rightarrow$ $(R0 + GBR)$	3	—

**Table 2.15 Shift Instructions**

Instruction		Instruction Code	Operation	Execution Cycles	T Bit
ROTL	Rn	0100nnnn00000100	$T \leftarrow Rn \leftarrow \text{MSB}$	1	MSB
ROTR	Rn	0100nnnn00000101	$\text{LSB} \rightarrow Rn \rightarrow T$	1	LSB
ROTCL	Rn	0100nnnn00100100	$T \leftarrow Rn \leftarrow T$	1	MSB
ROTCL	Rn	0100nnnn00100101	$T \rightarrow Rn \rightarrow T$	1	LSB
SHAL	Rn	0100nnnn00100000	$T \leftarrow Rn \leftarrow 0$	1	MSB
SHAR	Rn	0100nnnn00100001	$\text{MSB} \rightarrow Rn \rightarrow T$	1	LSB
SHLL	Rn	0100nnnn00000000	$T \leftarrow Rn \leftarrow 0$	1	MSB
SHLR	Rn	0100nnnn00000001	$0 \rightarrow Rn \rightarrow T$	1	LSB
SHLL2	Rn	0100nnnn00001000	$Rn \ll 2 \rightarrow Rn$	1	—
SHLR2	Rn	0100nnnn00001001	$Rn \gg 2 \rightarrow Rn$	1	—
SHLL8	Rn	0100nnnn00011000	$Rn \ll 8 \rightarrow Rn$	1	—
SHLR8	Rn	0100nnnn00011001	$Rn \gg 8 \rightarrow Rn$	1	—
SHLL16	Rn	0100nnnn00101000	$Rn \ll 16 \rightarrow Rn$	1	—
SHLR16	Rn	0100nnnn00101001	$Rn \gg 16 \rightarrow Rn$	1	—

**Table 2.16 Branch Instructions**

<b>Instruction</b>	<b>Instruction Code</b>	<b>Operation</b>	<b>Exec. Cycles</b>	<b>T Bit</b>
BF label	10001011dddddddd	If T = 0, disp × 2 + PC → PC; if T = 1, nop	3/1*	—
BF/S label	10001111dddddddd	Delayed branch, if T = 0, disp × 2 + PC → PC; if T = 1, nop	3/1*	—
BT label	10001001dddddddd	If T = 1, disp × 2 + PC → PC; if T = 0, nop	3/1*	—
BT/S label	10001101dddddddd	Delayed branch, if T = 1, disp × 2 + PC → PC; if T = 0, nop	2/1*	—
BRA label	1010dddddddddddd	Delayed branch, disp × 2 + PC → PC	2	—
BRAF Rm	0000rrrrrrm00100011	Delayed branch, Rm + PC → PC	2	—
BSR label	1011dddddddddddd	Delayed branch, PC → PR, disp × 2 + PC → PC	2	—
BSRF Rm	0000rrrrrrm00000011	Delayed branch, PC → PR, Rm + PC → PC	2	—
JMP @Rm	0100rrrrrrm00101011	Delayed branch, Rm → PC	2	—
JSR @Rm	0100rrrrrrm00001011	Delayed branch, PC → PR, Rm → PC	2	—
RTS	0000000000001011	Delayed branch, PR → PC	2	—

Note: One state when it does not branch.

**Table 2.17 System Control Instructions**

Instruction	Instruction Code	Operation	Exec. Cycles	T Bit
CLRT	0000000000001000	0 → T	1	0
CLRMAC	0000000000101000	0 → MACH, MACL	1	—
LDC Rm, SR	0100mmmm00001110	Rm → SR	1	LSB
LDC Rm, GBR	0100mmmm00011110	Rm → GBR	1	—
LDC Rm, VBR	0100mmmm00101110	Rm → VBR	1	—
LDC.L @Rm+, SR	0100mmmm00000111	(Rm) → SR, Rm + 4 → Rm	3	LSB
LDC.L @Rm+, GBR	0100mmmm00010111	(Rm) → GBR, Rm + 4 → Rm	3	—
LDC.L @Rm+, VBR	0100mmmm00100111	(Rm) → VBR, Rm + 4 → Rm	3	—
LDS Rm, MACH	0100mmmm00001010	Rm → MACH	1	—
LDS Rm, MACL	0100mmmm00011010	Rm → MACL	1	—
LDS Rm, PR	0100mmmm00101010	Rm → PR	1	—
LDS.L @Rm+, MACH	0100mmmm00000110	(Rm) → MACH, Rm + 4 → Rm	1	—
LDS.L @Rm+, MACL	0100mmmm00010110	(Rm) → MACL, Rm + 4 → Rm	1	—
LDS.L @Rm+, PR	0100mmmm00100110	(Rm) → PR, Rm + 4 → Rm	1	—
NOP	0000000000001001	No operation	1	—
RTE	0000000000101011	Delayed branch, stack area → PC/SR	4	—
SETT	000000000011000	1 → T	1	1
SLEEP	000000000011011	Sleep	3*	—
STC SR, Rn	0000nnnn00000010	SR → Rn	1	—
STC GBR, Rn	0000nnnn00010010	GBR → Rn	1	—
STC VBR, Rn	0000nnnn00100010	VBR → Rn	1	—
STC.L SR, @-Rn	0100nnnn00000011	Rn-4 → Rn, SR → (Rn)	2	—
STC.L GBR, @-Rn	0100nnnn00010011	Rn-4 → Rn, GBR → (Rn)	2	—
STC.L VBR, @-Rn	0100nnnn00100011	Rn-4 → Rn, BR → (Rn)	2	—
STS MACH, Rn	0000nnnn00001010	MACH → Rn	1	—
STS MACL, Rn	0000nnnn00011010	MACL → Rn	1	—
STS PR, Rn	0000nnnn00101010	PR → Rn	1	—

**Table 2.17 System Control Instructions (cont)**

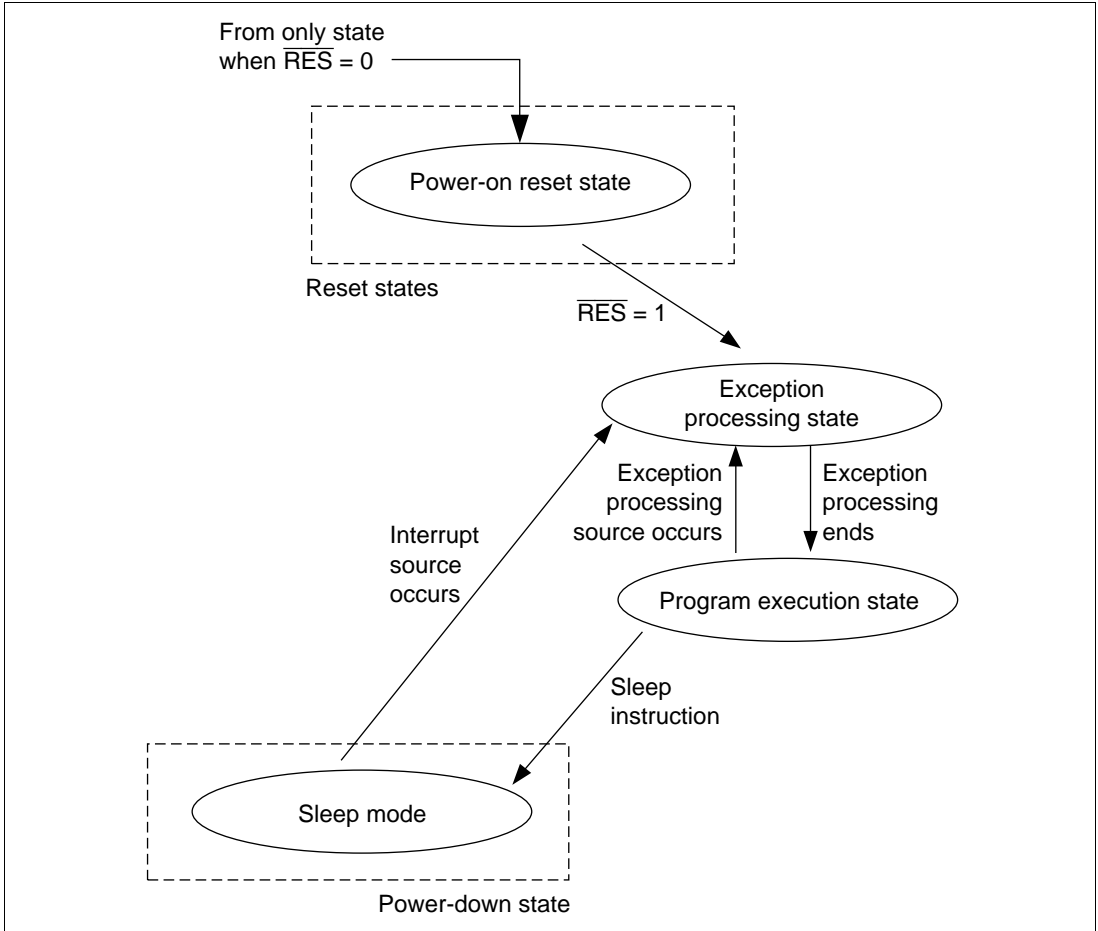
<b>Instruction</b>	<b>Instruction Code</b>	<b>Operation</b>	<b>Exec. Cycles</b>	<b>T Bit</b>
STS.L MACH,@-Rn	0100nnnn00000010	Rn-4 → Rn, MACH → (Rn)	1	—
STS.L MACL,@-Rn	0100nnnn00010010	Rn-4 → Rn, MACL → (Rn)	1	—
STS.L PR,@-Rn	0100nnnn00100010	Rn-4 → Rn, PR → (Rn)	1	—
TRAPA #imm	11000011iiiiiiii	PC/SR → stack area, (imm) → PC	8	—

Note: The number of execution cycles before the chip enters sleep mode.

## 2.5 Processing States

### 2.5.1 State Transitions

The CPU has four processing states: reset, exception processing, program execution and power-down. Figure 2.6 shows the transitions between the states.



**Figure 2.6 Transitions between Processing States**

**Reset State:** The CPU resets in the reset state. When the  $\overline{RES}$  pin level goes low, a power-on reset results.

**Exception Processing State:** The exception processing state is a transient state that occurs when exception processing sources such as resets or interrupts alter the CPU's processing state flow.



For a reset, the initial values of the program counter (PC) (execution start address) and stack pointer (SP) are fetched from the exception processing vector table and stored; the CPU then branches to the execution start address and execution of the program begins.

For an interrupt, the stack pointer (SP) is accessed and the program counter (PC) and status register (SR) are saved to the stack area. The exception service routine start address is fetched from the exception processing vector table; the CPU then branches to that address and the program starts executing, thereby entering the program execution state.

**Program Execution State:** In the program execution state, the CPU sequentially executes the program.

**Power-Down State:** In the power-down state, the CPU operation halts and power consumption declines. The SLEEP instruction places the CPU in the power-down state.

# Section 3 Power-Down State

## 3.1 Overview

In the power-down state, CPU functions are halted, greatly reducing the power consumption of the chip.

### 3.1.1 Power-Down State

The power-down state consists of a sleep mode.

Table 3.1 shows the conditions for entering sleep mode from the program execution state, the state of the CPU and on-chip peripheral functions in sleep mode, and the methods of exiting sleep mode.

**Table 3.1 Power-Down State**

Mode	Entering Conditions	State						
		Clock	CPU	On-Chip Peripheral Modules	CPU Registers	On-Chip RAM	I/O Ports	Exiting Methods
Sleep	Execution of SLEEP instruction	Active	Halted	Active	Held	Held	Held	1. Interrupt 2. Power-on reset

## 3.2 Sleep Mode

### 3.2.1 Transition to Sleep Mode

When the SLEEP instruction is executed, the chip makes a transition from the program execution state to sleep mode. Immediately after execution of the SLEEP instruction the CPU halts, but the contents of its internal registers are retained. On-chip peripheral modules continue to operate.

### 3.2.2 Exit from Sleep Mode

Sleep mode is exited by an interrupt or a power-on reset.

**Exit by Interrupt:** When an interrupt is generated, sleep mode is exited and interrupt exception processing is executed. If the priority level of the generated interrupt is not higher than the mask level set in the CPU's status register (SR), or if an interrupt by an on-chip peripheral module is disabled on the module side, the interrupt request will not be accepted and sleep mode will not be exited.

**Exit by Power-On Reset:** When the  $\overline{\text{RES}}$  pin is driven low, the chip exits sleep mode and enters the power-on reset state.

# Section 4 Clock Pulse Generator (CPG)

## 4.1 Overview

The clock pulse generator (CPG) supplies clock pulses within the SH7011 and to external devices. The SH7011's CPG operates the SH7011 at a frequency equal to the oscillation frequency of the crystal resonator. The CPG is composed of an oscillator and a duty adjustment circuit (figure 4.1). There are two ways of generating a clock with the CPG: by connecting a crystal resonator, or by inputting an external clock.

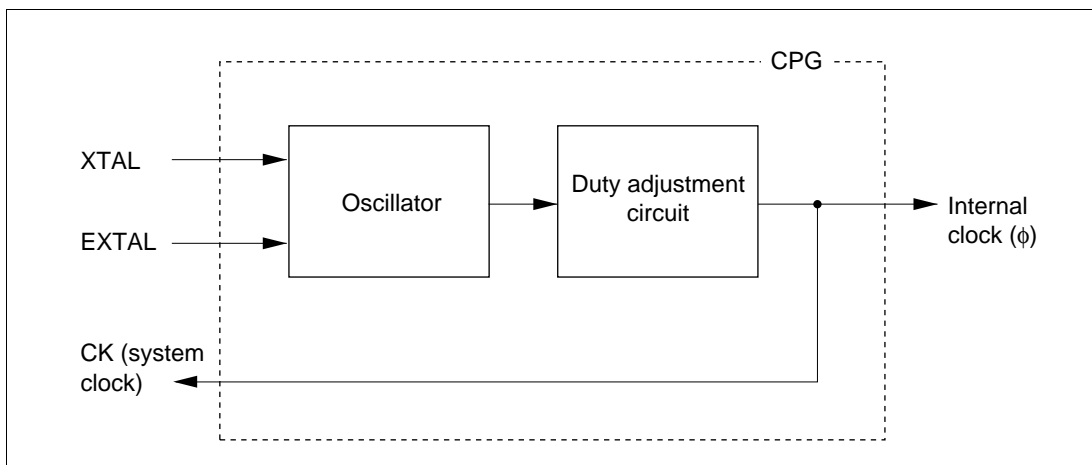


Figure 4.1 CPG Block Diagram

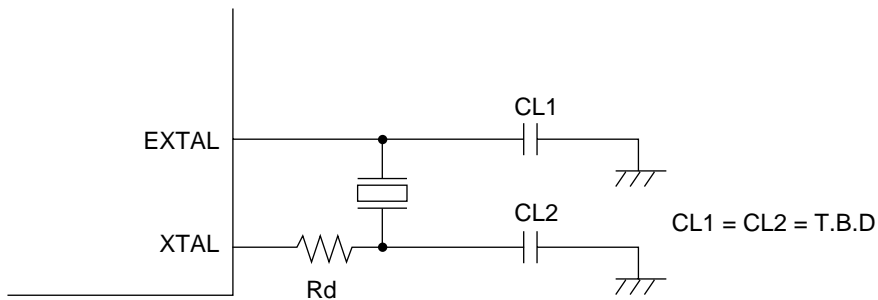
## 4.2 Clock Source

Either a crystal resonator or an external clock can be selected as the clock pulse source.

### 4.2.1 Crystal Resonator Connection

**Circuit Configuration:** Figure 4.2 shows the method of connecting a crystal resonator. Use the damping resistance ( $R_d$ ) shown in table 4.1. An AT-cut parallel-resonance type crystal resonator with the same frequency as the system clock (CK) should be used. Load capacitors ( $C_{L1}$ ,  $C_{L2}$ ) must be connected as shown in the figure.

The clock pulses generated by the crystal resonator and internal oscillator are sent to the duty adjustment circuit. After the duty has been adjusted, the pulses are supplied within the SH7011 chip and to external devices.

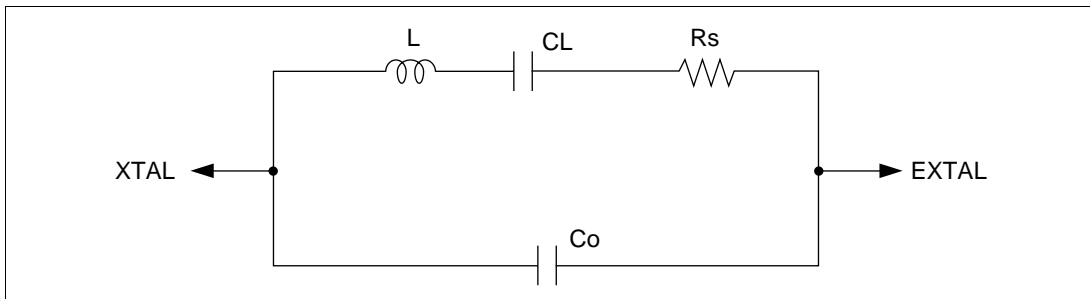


**Figure 4.2 Example of Crystal Resonator Connection**

**Table 4.1 Damping Resistance Value**

Frequency (MHz)	20
Rd (Ω)	T.B.D.

**Crystal Resonator:** Figure 4.3 shows an equivalent circuit for the crystal resonator. Use a crystal resonator with the characteristics shown in table 4.2.



**Figure 4.3 Crystal Resonator Equivalent Circuit**

**Table 4.2 Crystal Resonator Characteristics**

Parameter	Frequency (MHz)
Rs max (Ω)	T.B.D.
Co max (pF)	T.B.D.

## 4.2.2 External Clock Input

Input the external clock to the EXTAL pin and leave the XTAL pin open (figure 4.4.). The external clock frequency should be the same as that of the system clock (CK).

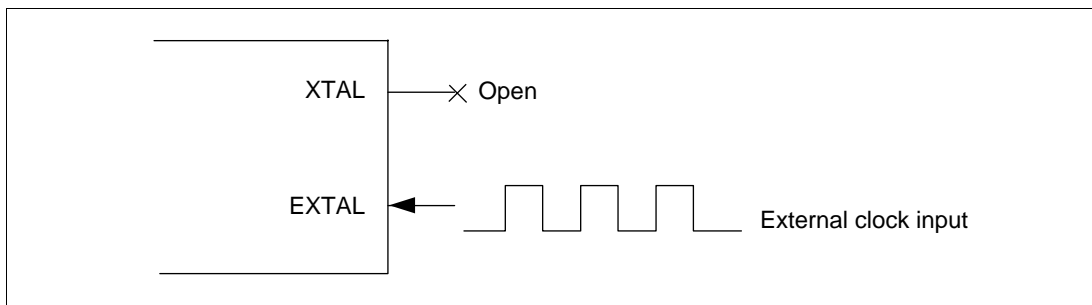


Figure 4.4 External Clock Input

## 4.3 Usage Notes

**Note on Board Design:** Place the crystal resonator and load capacitors as close as possible to the EXTAL and XTAL pins. To prevent induction from interfering with correct oscillation, ensure that other signal lines do not cross the EXTAL and XTAL pin signal lines.

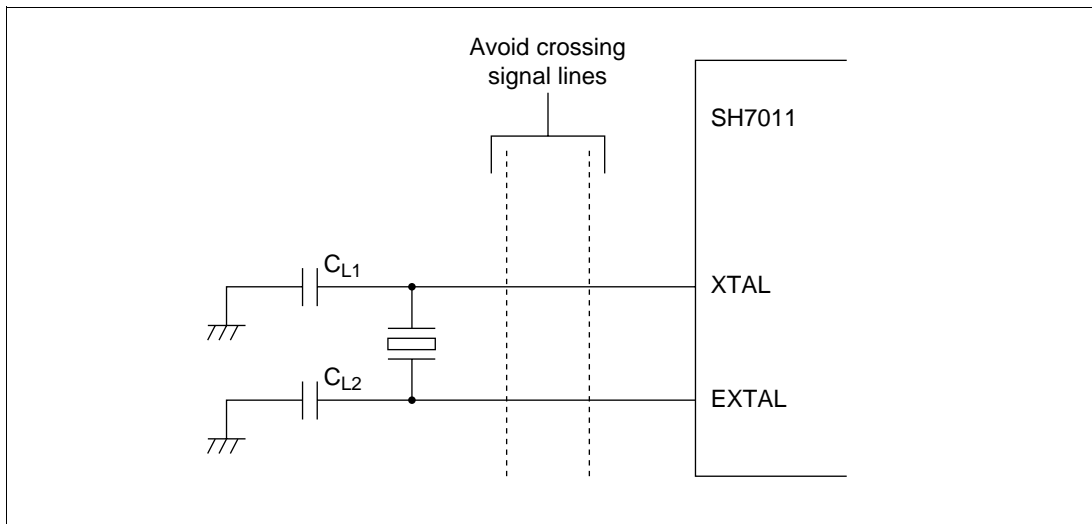
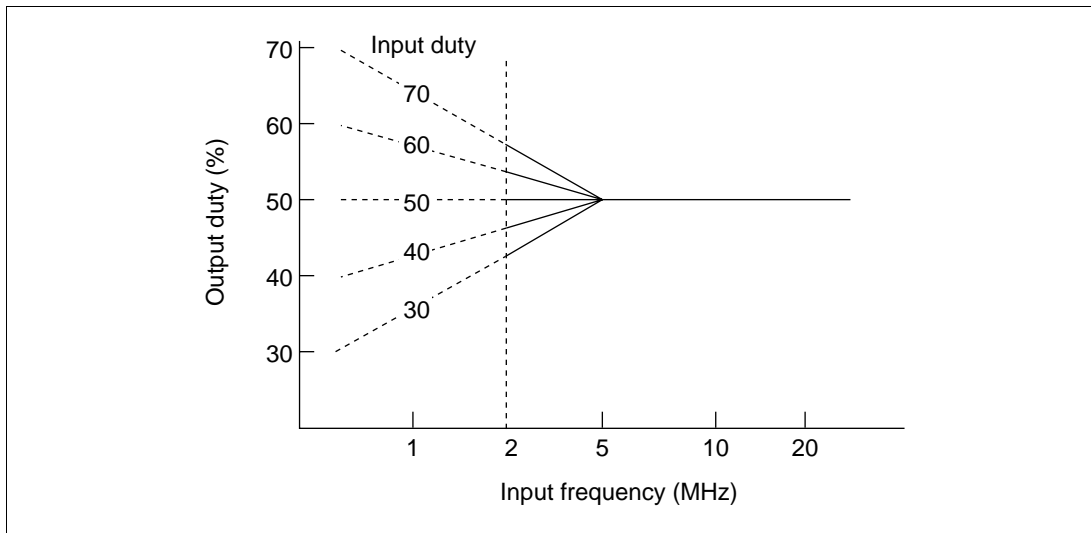


Figure 4.5 Note on Board Design

**Notes on Duty Adjustment:** Duty adjustment circuit is performed on an input clock of 5 MHz or higher. With a frequency of less than 5 MHz, duty adjustment may not be performed, but AC characteristics  $t_{CH}$  (clock high-level width) and  $t_{CL}$  (clock low-level width) are satisfied, and there is no problem with SH7011 internal operation. Figure 4.6 shows the basic characteristics of the duty adjustment circuit.

The duty adjustment circuit does not correct for transient fluctuations or jitter in the input clock. Thus, several tens of  $\mu$ s are required until duty adjustment is performed and a stable clock is obtained.



**Figure 4.6 Duty Adjustment Circuit Characteristics**

# Section 5 Exception Processing

## 5.1 Overview

### 5.1.1 Types of Exception Processing and Priority

Exception processing is started by four sources: resets, address errors, interrupts and instructions and have the priority shown in table 5.1. When several exception processing sources occur at once, they are processed according to the priority shown.

**Table 5.1 Types of Exception Processing and Priority Order**

Exception	Source	Priority
Reset	Power-on reset	High
Address error	CPU address error	
Interrupt	NMI	
	IRQ	
	On-chip peripheral modules:	<ul style="list-style-type: none"><li>• Multifunction timer/pulse unit (MTU)</li><li>• Serial communications interface (SCI)</li><li>• A/D converter (A/D)</li><li>• Compare match timer (CMT)</li><li>• 8-bit timer 1 (TIM1)</li><li>• 8-bit timer 2 (TIM2)</li></ul>
Instructions	Trap instruction (TRAPA instruction)	
	General illegal instructions (undefined code)	
	Illegal slot instructions (undefined code placed directly after a delay branch instruction* <sup>1</sup> or instructions that rewrite the PC* <sup>2</sup> )	Low

- Notes:
1. Delayed branch instructions: JMP, JSR, BRA, BSR, RTS, RTE, BF/S, BT/S, BSRF, BRAF.
  2. Instructions that rewrite the PC: JMP, JSR, BRA, BSR, RTS, RTE, BT, BF, TRAPA, BF/S, BT/S, BSRF, BRAF.



## 5.1.2 Exception Processing Operations

The exception processing sources are detected and begin processing according to the timing shown in table 5.2.

**Table 5.2 Timing of Exception Source Detection and the Start of Exception Processing**

Exception	Source	Timing of Source Detection and Start of Processing
Power-on reset		Starts when the $\overline{\text{RES}}$ pin changes from low to high.
Address error		Detected when instruction is decoded and starts when the previous executing instruction finishes executing.
Interrupts		Detected when instruction is decoded and starts when the previous executing instruction finishes executing.
Instructions	Trap instruction	Starts from the execution of a TRAPA instruction.
	General illegal instructions	Starts from the decoding of undefined code anytime except after a delayed branch instruction (delay slot).
	Illegal slot instructions	Starts from the decoding of undefined code placed in a delayed branch instruction (delay slot) or of instructions that rewrite the PC.

When exception processing starts, the CPU operates as follows:

1. Exception processing triggered by reset:

The initial values of the program counter (PC) and stack pointer (SP) are fetched from the exception processing vector table (PC and SP are respectively the H'00000000 and H'00000004 addresses). See section 5.1.3, Exception Processing Vector Table, for more information. 0 is then written to the vector base register (VBR) and 1111 is written to the interrupt mask bits (I3–I0) of the status register (SR). The program begins running from the PC address fetched from the exception processing vector table.

2. Exception processing triggered by address errors, interrupts and instructions:

SR and PC are saved to the stack indicated by R15. For interrupt exception processing, the interrupt priority level is written to the SR's interrupt mask bits (I3–I0). For address error and instruction exception processing, the I3–I0 bits are not affected. The start address is then fetched from the exception processing vector table and the program begins running from that address.

### 5.1.3 Exception Processing Vector Table

Before exception processing begins running, the exception processing vector table must be set in memory. The exception processing vector table stores the start addresses of exception service routines. (The reset exception processing table holds the initial values of PC and SP.)

All exception sources are given different vector numbers and vector table address offsets, from which the vector table addresses are calculated. During exception processing, the start addresses of the exception service routines are fetched from the exception processing vector table, which indicated by this vector table address.

Table 5.3 shows the vector numbers and vector table address offsets. Table 5.4 shows how vector table addresses are calculated.

**Table 5.3 Exception Processing Vector Table**

Exception Sources		Vector Numbers	Vector Table Address Offset
Power-on reset	PC	0	H'00000000–H'00000003
	SP	1	H'00000004–H'00000007
(Reserved by system)		2	H'00000008–H'0000000F
		3	
General illegal instruction		4	H'00000010–H'00000013
(Reserved by system)		5	H'00000014–H'00000017
Slot illegal instruction		6	H'00000018–H'0000001B
(Reserved by system)		7	H'0000001C–H'0000001F
(Reserved by system)		8	H'00000020–H'00000023
CPU address error		9	H'00000024–H'00000027
(Reserved by system)		10	H'00000028–H'0000002B
Interrupts	NMI	11	H'0000002C–H'0000002F
(Reserved by system)		12	H'00000030–H'00000033
		:	:
Trap instruction (user vector)		31	H'0000007C–H'0000007F
		:	:
		32	H'00000080–H'00000083
		:	:
		63	H'000000FC–H'000000FF

**Table 5.3 Exception Processing Vector Table (cont)**

<b>Exception Sources</b>	<b>Vector Numbers</b>	<b>Vector Table Address Offset</b>
Interrupts	IRQ0	64
	IRQ1	65
	IRQ2	66
	IRQ3	67
	IRQ4	68
	IRQ5	69
	IRQ6	70
	IRQ7	71
	On-chip peripheral module*	72 : 255

Note: The vector numbers and vector table address offsets for each on-chip peripheral module interrupt are given in section 6, Interrupt Controller, table 6.3, Interrupt Exception Processing Vectors and Priorities.

**Table 5.4 Calculating Exception Processing Vector Table Addresses**

<b>Exception Source</b>	<b>Vector Table Address Calculation</b>
Resets	Vector table address = (vector table address offset) = (vector number) × 4
Address errors, interrupts, instructions	Vector table address = VBR + (vector table address offset) = VBR + (vector number) × 4

- Notes: 1. VBR: Vector base register  
2. Vector table address offset: See table 5.3.  
3. Vector number: See table 5.3.

## 5.2 Resets

### 5.2.1 Reset

A reset has the highest priority of any exception source. As shown in table 5.5, a power-on reset initializes the internal state of the CPU and the on-chip peripheral module registers.

**Table 5.5** Types of Resets

Type	Conditions for Transition to Reset Status	Internal Status	
	$\overline{\text{RES}}$	CPU	On-Chip Peripheral Module
Power-on reset	Low	Initialized	Initialized

### 5.2.2 Power-On Reset

When the  $\overline{\text{RES}}$  pin is driven low, the LSI does a power-on reset. To reliably reset the LSI, the  $\overline{\text{RES}}$  pin should be kept at low for at least the duration of the oscillation settling time when applying power or when in standby mode (when the clock circuit is halted) or at least  $20 t_{\text{cyc}}$  (when the clock circuit is running). During power-on reset, CPU internal status and all registers of on-chip peripheral modules are initialized. See Appendix B, Pin Status, for the status of individual pins during the power-on reset status.

In the power-on reset status, power-on reset exception processing starts when the  $\overline{\text{RES}}$  pin is first driven low for a set period of time and then returned to high. The CPU will then operate as follows:

1. The initial value (execution start address) of the program counter (PC) is fetched from the exception processing vector table.
2. The initial value of the stack pointer (SP) is fetched from the exception processing vector table.
3. The vector base register (VBR) is cleared to H'00000000 and the interrupt mask bits (I3–I0) of the status register (SR) are set to H'F (1111).
4. The values fetched from the exception processing vector table are set in the program counter (PC) and SP and the program begins executing.

Be certain to always perform power-on reset processing when turning the system power on.

## 5.3 Address Errors

Address errors occur when instructions are fetched or data read or written, as shown in table 5.6.

**Table 5.6 Bus Cycles and Address Errors**

<b>Bus Cycle</b>		
<b>Type</b>	<b>Bus Cycle Description</b>	<b>Address Errors</b>
Instruction fetch	Instruction fetched from even address	None (normal)
	Instruction fetched from odd address	Address error occurs
	Instruction fetched from other than on-chip peripheral module space*	None (normal)
	Instruction fetched from on-chip peripheral module space*	Address error occurs
Data read/write	Word data accessed from even address	None (normal)
	Word data accessed from odd address	Address error occurs
	Longword data accessed from other than a longword boundary	Address error occurs
	Byte or word data accessed in on-chip peripheral module space*	None (normal)
	Longword data accessed in 16-bit on-chip peripheral module space*	None (normal)
	Longword data accessed in 8-bit on-chip peripheral module space*	Address error occurs

Note: See section 7, Bus State Controller.

### 5.3.1 Address Error Exception Processing

When an address error occurs, the bus cycle in which the address error occurred ends. When the executing instruction then finishes, address error exception processing starts up. The CPU operates as follows:

1. The status register (SR) is saved to the stack.
2. The program counter (PC) is saved to the stack. The PC value saved is the start address of the instruction to be executed after the last executed instruction.
3. The exception service routine start address is fetched from the exception processing vector table that corresponds to the address error that occurred and the program starts executing from that address. The jump that occurs is not a delayed branch.

## 5.4 Interrupts

Table 5.7 shows the sources that start up interrupt exception processing. These are divided into NMI, IRQ and on-chip peripheral modules.

**Table 5.7 Interrupt Sources**

Type	Request Source	Number of Sources
NMI	NMI pin (external input)	1
IRQ	$\overline{\text{IRQ0}}\text{--}\overline{\text{IRQ7}}$ (external input)	8
On-chip peripheral module	Multifunction timer/pulse unit (MTU)	11
	Serial communications interface (SCI)	4
	A/D converter	1
	Compare match timer (CMT)	2
	8-bit timer 1	1
	8-bit timer 2	1

Each interrupt source is allocated a different vector number and vector table offset. See section 6, Interrupt Controller, table 6.3, Interrupt Exception Processing Vectors and Priorities, for more information on vector numbers and vector table address offsets.

### 5.4.1 Interrupt Priority Level

The interrupt priority order is predetermined. When multiple interrupts occur simultaneously (overlap), the interrupt controller (INTC) determines their relative priorities and starts up processing according to the results.

The priority order of interrupts is expressed as priority levels 0–16, with priority 0 the lowest and priority 16 the highest. The NMI interrupt has priority 16 and cannot be masked, so it is always accepted. The user break interrupt priority level is 15. IRQ interrupts and on-chip peripheral module interrupt priority levels can be set freely using the INTC's interrupt priority level setting registers A through H (IPRA to IPRH) as shown in table 5.8. The priority levels that can be set are 0–15. Level 16 cannot be set. See section 6.3.1, Interrupt Priority Registers A-H (IPRA-IPRH), for more information on IPRA to IPRH.

**Table 5.8 Interrupt Priority Order**

Type	Priority Level	Comment
NMI	16	Fixed priority level. Cannot be masked.
IRQ	0–15	Set with interrupt priority level setting registers A through H (IPRA to IPRH).
On-chip peripheral module	0–15	Set with interrupt priority level setting registers A through H (IPRA to IPRH).

## 5.4.2 Interrupt Exception Processing

When an interrupt occurs, its priority level is ascertained by the interrupt controller (INTC). NMI is always accepted, but other interrupts are only accepted if they have a priority level higher than the priority level set in the interrupt mask bits (I3–I0) of the status register (SR).

When an interrupt is accepted, exception processing begins. In interrupt exception processing, the CPU saves SR and the program counter (PC) to the stack. The priority level value of the accepted interrupt is written to SR bits I3–I0. For NMI, however, the priority level is 16, but the value set in I3–I0 is H'F (level 15). Next, the start address of the exception service routine is fetched from the exception processing vector table for the accepted interrupt, that address is jumped to and execution begins. See section 6.4, Interrupt Operation, for more information on the interrupt exception processing.

## 5.5 Exceptions Triggered by Instructions

Exception processing can be triggered by trap instructions, general illegal instructions, and illegal slot instructions, as shown in table 5.9.

**Table 5.9 Types of Exceptions Triggered by Instructions**

Type	Source Instruction	Comment
Trap instructions	TRAPA	—
Illegal slot instructions	Undefined code placed immediately after a delayed branch instruction (delay slot) and instructions that rewrite the PC	Delayed branch instructions: JMP, JSR, BRA, BSR, RTS, RTE, BF/S, BT/S, BSRF, BRAF Instructions that rewrite the PC: JMP, JSR, BRA, BSR, RTS, RTE, BT, BF, TRAPA, BF/S, BT/S, BSRF, BRAF
General illegal instructions	Undefined code anywhere besides in a delay slot	—

### 5.5.1 Trap Instructions

When a TRAPA instruction is executed, trap instruction exception processing starts up. The CPU operates as follows:

1. The status register (SR) is saved to the stack.
2. The program counter (PC) is saved to the stack. The PC value saved is the start address of the instruction to be executed after the TRAPA instruction.
3. The exception service routine start address is fetched from the exception processing vector table that corresponds to the vector number specified in the TRAPA instruction. That address is jumped to and the program starts executing. The jump that occurs is not a delayed branch.

### 5.5.2 Illegal Slot Instructions

An instruction placed immediately after a delayed branch instruction is said to be placed in a delay slot. When the instruction placed in the delay slot is undefined code, illegal slot exception processing starts up when that undefined code is decoded. Illegal slot exception processing also starts up when an instruction that rewrites the program counter (PC) is placed in a delay slot. The processing starts when the instruction is decoded. The CPU handles an illegal slot instruction as follows:

1. The status register (SR) is saved to the stack.
2. The program counter (PC) is saved to the stack. The PC value saved is the jump address of the delayed branch instruction immediately before the undefined code or the instruction that rewrites the PC.
3. The exception service routine start address is fetched from the exception processing vector table that corresponds to the exception that occurred. That address is jumped to and the program starts executing. The jump that occurs is not a delayed branch.

### 5.5.3 General Illegal Instructions

When undefined code placed anywhere other than immediately after a delayed branch instruction (i.e., in a delay slot) is decoded, general illegal instruction exception processing starts up. The CPU handles general illegal instructions the same as illegal slot instructions. Unlike processing of illegal slot instructions, however, the program counter value stored is the start address of the undefined code.



## 5.6 When Exception Sources Are Not Accepted

When an address error or interrupt is generated after a delayed branch instruction or interrupt-disabled instruction, it is sometimes not accepted immediately but stored instead, as shown in table 5.10. When this happens, it will be accepted when an instruction that can accept the exception is decoded.

**Table 5.10 Generation of Exception Sources Immediately after a Delayed Branch Instruction or Interrupt-Disabled Instruction**

Point of Occurrence	Exception Source	
	Address Error	Interrupt
Immediately after a delayed branch instruction* <sup>1</sup>	Not accepted	Not accepted
Immediately after an interrupt-disabled instruction* <sup>2</sup>	Accepted	Not accepted

Notes: 1. Delayed branch instructions: JMP, JSR, BRA, BSR, RTS, RTE, BF/S, BT/S, BSRF, BRAF

2. Interrupt-disabled instructions: LDC, LDC.L, STC, STC.L, LDS, LDS.L, STS, STS.L

### 5.6.1 Immediately after a Delayed Branch Instruction

When an instruction placed immediately after a delayed branch instruction (delay slot) is decoded, neither address errors nor interrupts are accepted. The delayed branch instruction and the instruction located immediately after it (delay slot) are always executed consecutively, so no exception processing occurs during this period.

### 5.6.2 Immediately after an Interrupt-Disabled Instruction

When an instruction immediately following an interrupt-disabled instruction is decoded, interrupts are not accepted. Address errors are accepted.

## 5.7 Stack Status after Exception Processing Ends

The status of the stack after exception processing ends is as shown in table 5.11.

**Table 5.11 Types of Stack Status After Exception Processing Ends**

Types	Stack Status
Address error	
Trap instruction	
General illegal instruction	
Interrupt	
Illegal slot instruction	

## 5.8 Notes on Use

### 5.8.1 Value of Stack Pointer (SP)

The value of the stack pointer must always be a multiple of four. If it is not, an address error will occur when the stack is accessed during exception processing.

### 5.8.2 Value of Vector Base Register (VBR)

The value of the vector base register must always be a multiple of four. If it is not, an address error will occur when the stack is accessed during exception processing.

### 5.8.3 Address Errors Caused by Stacking of Address Error Exception Processing

When the stack pointer is not a multiple of four, an address error will occur during stacking of the exception processing (interrupts, etc.) and address error exception processing will start up as soon as the first exception processing is ended. Address errors will then also occur in the stacking for this address error exception processing. To ensure that address error exception processing does not go into an endless loop, no address errors are accepted at that point. This allows program control to be shifted to the address error exception service routine and enables error processing.

When an address error occurs during exception processing stacking, the stacking bus cycle (write) is executed. During stacking of the status register (SR) and program counter (PC), the SP is -4 for both, so the value of SP will not be a multiple of four after the stacking either. The address value output during stacking is the SP value, so the address where the error occurred is itself output. This means the write data stacked will be undefined.

# Section 6 Interrupt Controller (INTC)

## 6.1 Overview

The interrupt controller (INTC) ascertains the priority of interrupt sources and controls interrupt requests to the CPU. The INTC has registers for setting the priority of each interrupt which can be used by the user to order the priorities in which the interrupt requests are processed.

### 6.1.1 Features

The INTC has the following features:

- 16 levels of interrupt priority: By setting the eight interrupt-priority level registers, the priorities of IRQ interrupts and on-chip peripheral module interrupts can be set in 16 levels for different request sources.
- NMI noise canceler function: NMI input level bits indicate the NMI pin status. By reading these bits with the interrupt exception service routine, the pin status can be confirmed, enabling it to be used as a noise canceler.

## 6.1.2 Block Diagram

Figure 6.1 is a block diagram of the INTC.

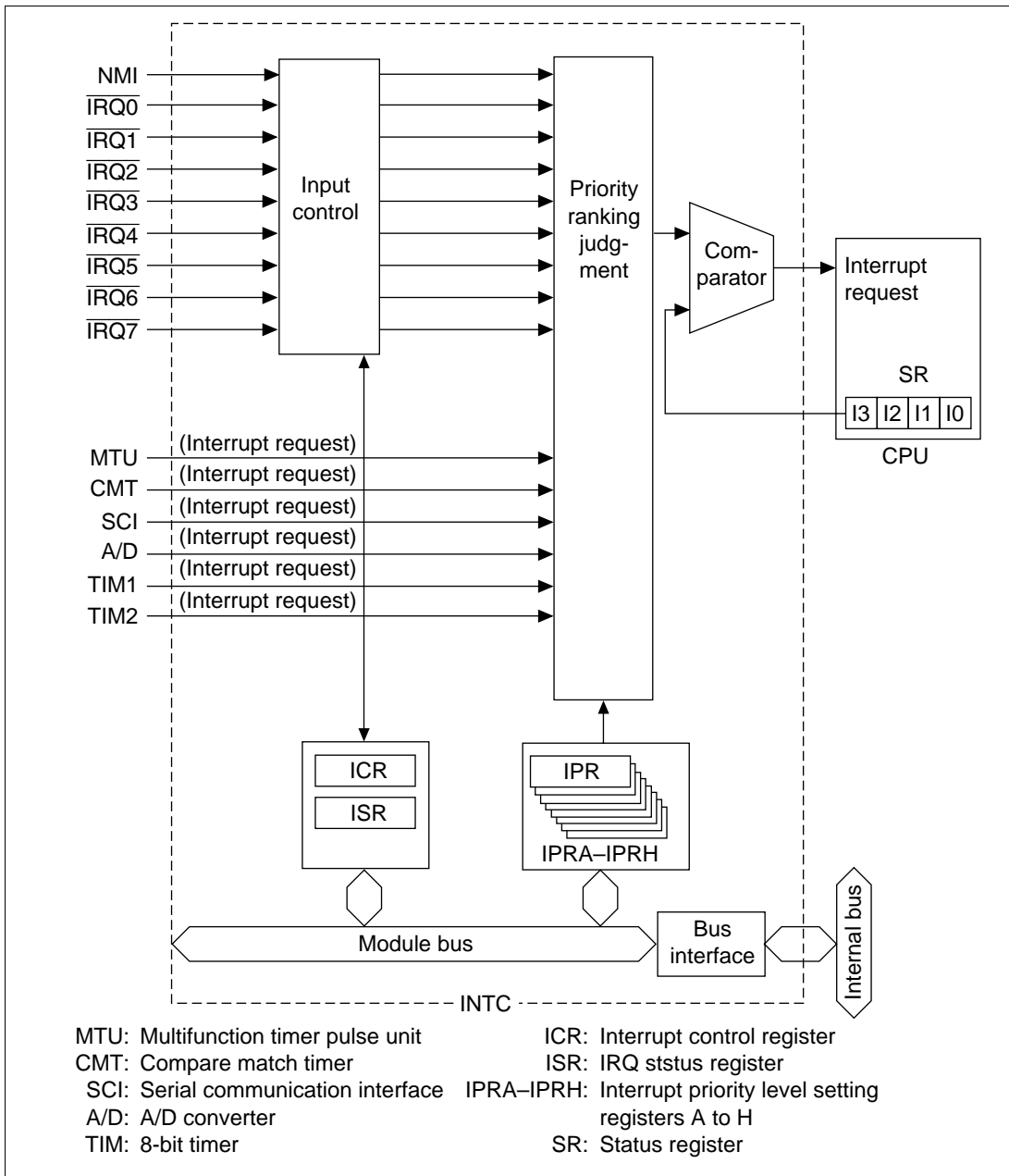


Figure 6.1 INTC Block Diagram

### 6.1.3 Pin Configuration

Table 6.1 shows the INTC pin configuration.

**Table 6.1 Pin Configuration**

Name	Abbreviation	I/O	Function
Non-maskable interrupt input pin	NMI	I	Input of non-maskable interrupt request signal
Interrupt request input pins	$\overline{\text{IRQ0}}\text{--}\overline{\text{IRQ7}}$	I	Input of maskable interrupt request signals

### 6.1.4 Register Configuration

The INTC has the 10 registers shown in table 6.2. These registers set the priority of the interrupts and control external interrupt input signal detection.

**Table 6.2 Register Configuration**

Name	Abbr.	R/W	Initial Value	Address	Access Sizes
Interrupt priority register A	IPRA	R/W	H'0000	H'FFFF8348	8, 16, 32
Interrupt priority register B	IPRB	R/W	H'0000	H'FFFF834A	8, 16, 32
Interrupt priority register C	IPRC	R/W	H'0000	H'FFFF834C	8, 16, 32
Interrupt priority register D	IPRD	R/W	H'0000	H'FFFF834E	8, 16, 32
Interrupt priority register E	IPRE	R/W	H'0000	H'FFFF8350	8, 16, 32
Interrupt priority register F	IPRF	R/W	H'0000	H'FFFF8352	8, 16, 32
Interrupt priority register G	IPRG	R/W	H'0000	H'FFFF8354	8, 16, 32
Interrupt priority register H	IPRH	R/W	H'0000	H'FFFF8356	8, 16, 32
Interrupt control register	ICR	R/W	*1	H'FFFF8358	8, 16, 32
IRQ status register	ISR	R(W)*2	H'0000	H'FFFF835A	8, 16, 32

Notes: 1. The value when the NMI pin is high is H'8000; when the NMI pin is low, it is H'0000.  
2. Only 0 can be written, in order to clear flags.

## 6.2 Interrupt Sources

There are four types of interrupt sources: NMI, user breaks, IRQ, and on-chip peripheral modules. Each interrupt has a priority expressed as a priority level (0 to 16, with 0 the lowest and 16 the highest). Giving an interrupt a priority level of 0 masks it.

### 6.2.1 NMI Interrupts

The NMI interrupt has priority 16 and is always accepted. Input at the NMI pin is detected by edge. Use the NMI edge select bit (NMIE) in the interrupt control register (ICR) to select either the rising or falling edge. NMI interrupt exception processing sets the interrupt mask level bits (I3–I0) in the status register (SR) to level 15.

### 6.2.2 IRQ Interrupts

IRQ interrupts are requested by input from pins  $\overline{\text{IRQ0}}\text{--}\overline{\text{IRQ7}}$ . Set the IRQ sense select bits (IRQ0S–IRQ7S) of the interrupt control register (ICR) to select low level detection or falling edge detection for each pin. The priority level can be set from 0 to 15 for each pin using the interrupt priority registers A and B (IPRA–IPRB).

When IRQ interrupts are set to low level detection, an interrupt request signal is sent to the INTC during the period the IRQ pin is low level. Interrupt request signals are not sent to the INTC when the IRQ pin becomes high level. Interrupt request levels can be confirmed by reading the IRQ flags (IRQ0F–IRQ7F) of the IRQ status register (ISR).

When IRQ interrupts are set to falling edge detection, interrupt request signals are sent to the INTC upon detecting a change on the IRQ pin from high to low level. IRQ interrupt request detection results are maintained until the interrupt request is accepted. Confirmation that IRQ interrupt requests have been detected is possible by reading the IRQ flags (IRQ0F–IRQ7F) of the IRQ status register (ISR), and by writing a 0 after reading a 1, IRQ interrupt request detection results can be withdrawn.

In IRQ interrupt exception processing, the interrupt mask bits (I3–I0) of the status register (SR) are set to the priority level value of the accepted IRQ interrupt.

### 6.2.3 On-Chip Peripheral Module Interrupts

On-chip peripheral module interrupts are interrupts generated by the following on-chip peripheral modules:

- Multifunction timer pulse unit (MTU)
- Compare match timer (CMT)
- Serial communications interface (SCI)
- A/D converter (A/D)
- 8-bit timer 1 (TIM1)
- 8-bit timer 2 (TIM2)

A different interrupt vector is assigned to each interrupt source, so the exception service routine does not have to decide which interrupt has occurred. Priority levels between 0 and 15 can be assigned to individual on-chip peripheral modules in interrupt priority registers C–H (IPRC–IPRH).

On-chip peripheral module interrupt exception processing sets the interrupt mask level bits (I3–I0) in the status register (SR) to the priority level value of the on-chip peripheral module interrupt that was accepted.

### 6.2.4 Interrupt Exception Vectors and Priority Rankings

Table 6.3 lists interrupt sources and their vector numbers, vector table address offsets and interrupt priorities.

Each interrupt source is allocated a different vector number and vector table address offset. Vector table addresses are calculated from vector numbers and address offsets. In interrupt exception processing, the exception service routine start address is fetched from the vector table indicated by the vector table address. See section 5 Exception Processing, table 5.4, Calculating Exception Processing Vector Table Addresses.

IRQ interrupts and on-chip peripheral module interrupt priorities can be set freely between 0 and 15 for each pin or module by setting interrupt priority registers A–H (IPRA–IPRH). The ranking of interrupt sources for IPRC–IPRH, however, must be the order listed under Priority Order Within IPR Setting Range in table 6.3 and cannot be changed. A power-on reset assigns priority level 0 to IRQ interrupts and on-chip peripheral module interrupts. If the same priority level is assigned to two or more interrupt sources and interrupts from those sources occur simultaneously, their priority order is the default priority order indicated at the right in table 6.3.



**Table 6.3 Interrupt Exception Processing Vectors and Priorities**

Interrupt Source	Interrupt Vector		Interrupt Priority (Initial Value)	Corresponding IPR (Bits)	Priority within IPR Setting Range	Default Priority
	Vector No.	Vector Table Address Offset				
NMI	11	H'0000002C to H'0000002F	16	—	—	High
IRQ0	64	H'00000100 to H'00000103	0 to 15 (0)	IPRA (15–12)	—	
IRQ1	65	H'00000104 to H'00000107	0 to 15 (0)	IPRA (11–8)	—	
IRQ2	66	H'00000108 to H'0000010B	0 to 15 (0)	IPRA (7–4)	—	
IRQ3	67	H'0000010C to H'0000010F	0 to 15 (0)	IPRA (3–0)	—	
IRQ4	68	H'00000110 to H'00000113	0 to 15 (0)	IPRB (15–12)	—	
IRQ5	69	H'00000114 to H'00000117	0 to 15 (0)	IPRB (11–8)	—	
IRQ6	70	H'00000118 to H'0000011B	0 to 15 (0)	IPRB (7–4)	—	
IRQ7	71	H'0000011C to H'0000011F	0 to 15 (0)	IPRB (3–0)	—	
MTU0	TGI0A	88	H'00000160 to H'00000163	0 to 15 (0)	IPRD (15–12)	High
	TGI0B	89	H'00000164 to H'00000167	0 to 15 (0)		
	TGI0C	90	H'00000168 to H'0000016B	0 to 15 (0)		
	TGI0D	91	H'0000016C to H'0000016F	0 to 15 (0)	Low	
	TCI0V	92	H'00000170 to H'00000173	0 to 15 (0)		IPRD (11–8)

**Table 6.3 Interrupt Exception Processing Vectors and Priorities (cont)**

Interrupt Source		Interrupt Vector		Interrupt Priority (Initial Value)	Corresponding IPR (Bits)	Priority within IPR Setting Range	Default Priority
		Vector No.	Vector Table Address Offset				
MTU1	TGI1A	96	H'00000180 to H'00000183	0 to 15 (0)	IPRD (7–4)	High	High
	TGI1B	97	H'00000184 to H'00000187			0 to 15 (0)	
	TCI1V	100	H'00000190 to H'00000193	0 to 15 (0)	IPRD (3–0)	—	
MTU2	TGI2A	104	H'000001A0 to H'000001A3	0 to 15 (0)	IPRE (15–12)	High	
	TGI2B	105	H'000001A4 to H'000001A7			0 to 15 (0)	
	TCI2V	108	H'000001B0 to H'000001B3	0 to 15 (0)	IPRE (11–8)	—	
SCI	ERI	132	H'00000210 to H'00000213	0 to 15 (0)	IPRF (3–0)	High	
	RXI	133	H'00000214 to H'00000217				
	TXI	134	H'00000218 to H'0000021B				
	TEI	135	H'0000021C to H'0000021F		Low		
A/D	ADI	138	H'00000228 to H'0000022B	0 to 15 (0)	IPRG (15–12)	—	
CMT0	CMIO	144	H'00000240 to H'00000243	0 to 15 (0)	IPRG (7–4)	—	
CMT1	CMI1	148	H'00000250 to H'00000253	0 to 15 (0)	IPRG (3–0)	—	
TIM1	ITI	152	H'00000260 to H'00000263	0 to 15 (0)	IPRH (15–12)	High	
TIM2	CMI	153	H'00000264 to H'00000267			Low	Low

## 6.3 Description of Registers

### 6.3.1 Interrupt Priority Registers A–H (IPRA–IPRH)

Interrupt priority registers A–H (IPRA–IPRH) are 16-bit readable/writable registers that set priority levels from 0 to 15 for IRQ interrupts and on-chip peripheral module interrupts. Correspondence between interrupt request sources and each of the IPRA–IPRH bits is shown in table 6.4.

Bit:	15	14	13	12	11	10	9	8
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit:	7	6	5	4	3	2	1	0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**Table 6.4 Interrupt Request Sources and IPRA–IPRH**

Register	Bits			
	15–12	11–8	7–4	3–0
Interrupt priority register A	IRQ0	IRQ1	IRQ2	IRQ3
Interrupt priority register B	IRQ4	IRQ5	IRQ6	IRQ7
Interrupt priority register C	Reserved	Reserved	Reserved	Reserved
Interrupt priority register D	MTU0	MTU0	MTU1	MTU1
Interrupt priority register E	MTU2	MTTU2	Reserved	Reserved
Interrupt priority register F	Reserved	Reserved	Reserved	SCI
Interrupt priority register G	A/D	Reserved	CMT0	CMT1
Interrupt priority register H	TIM1, 2	Reserved	Reserved	Reserved

As indicated in table 6.4, four  $\overline{\text{IRQ}}$  pins or groups of 4 on-chip peripheral modules are allocated to each register. Each of the corresponding interrupt priority ranks are established by setting a value from H'0 (0000) to H'F (1111) in each of the four-bit groups 15–12, 11–8, 7–4 and 3–0. Interrupt priority rank becomes level 0 (lowest) by setting H'0, and level 15 (highest) by setting H'F. 8-bit timers 1 and 2 are set to the same priority rank.

IPRA–IPRH are initialized to H'0000 by a power-on reset. Reserved bits always return 0 if read, and the write value for these bits should always be 0.

### 6.3.2 Interrupt Control Register (ICR)

The ICR is a 16-bit register that sets the input signal detection mode of the external interrupt input pin NMI and  $\overline{\text{IRQ0}}-\overline{\text{IRQ7}}$  and indicates the input signal level to the NMI pin. A power-on reset initializes ICR.

	Bit:	15	14	13	12	11	10	9	8
		NMIL	—	—	—	—	—	—	NMIE
Initial value:		*	0	0	0	0	0	0	0
R/W:		R	R	R	R	R	R	R	R/W
	Bit:	7	6	5	4	3	2	1	0
		IRQ0S	IRQ1S	IRQ2S	IRQ3S	IRQ4S	IRQ5S	IRQ6S	IRQ7S
Initial value:		0	0	0	0	0	0	0	0
R/W:		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: When NMI input is high: 1; when NMI input is low: 0

- Bit 15—NMI Input Level (NMIL): Sets the level of the signal input at the NMI pin. This bit can be read to determine the NMI pin level. This bit cannot be modified.

Bit 15: NMIL	Description
0	NMI input level is low
1	NMI input level is high

- Bits 14 to 9—Reserved: These bits always read as 0. The write value should always be 0.
- Bit 8—NMI Edge Select (NMIE)

Bit 8: NMIE	Description
0	Interrupt request is detected on falling edge of NMI input (initial value)
1	Interrupt request is detected on rising edge of NMI input

- Bits 7 to 0—IRQ0–IRQ7 Sense Select (IRQ0S–IRQ7S): These bits set the IRQ0–IRQ7 interrupt request detection mode.

#### Bits 7-0: IRQ0S–IRQ7S Description

Bit	Description
0	Interrupt request is detected on low level of IRQ input (initial value)
1	Interrupt request is detected on falling edge of IRQ input

### 6.3.3 IRQ Status Register (ISR)

The ISR is a 16-bit register that indicates the interrupt request status of the external interrupt input pins  $\overline{\text{IRQ0}}\text{--}\overline{\text{IRQ7}}$ . When IRQ interrupts are set to edge detection, held interrupt requests can be withdrawn by writing a 0 to IRQnF after reading an  $\text{IRQnF} = 1$ .

A power-on reset initializes ISR.

Bit:	15	14	13	12	11	10	9	8
	—	—	—	—	—	—	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R

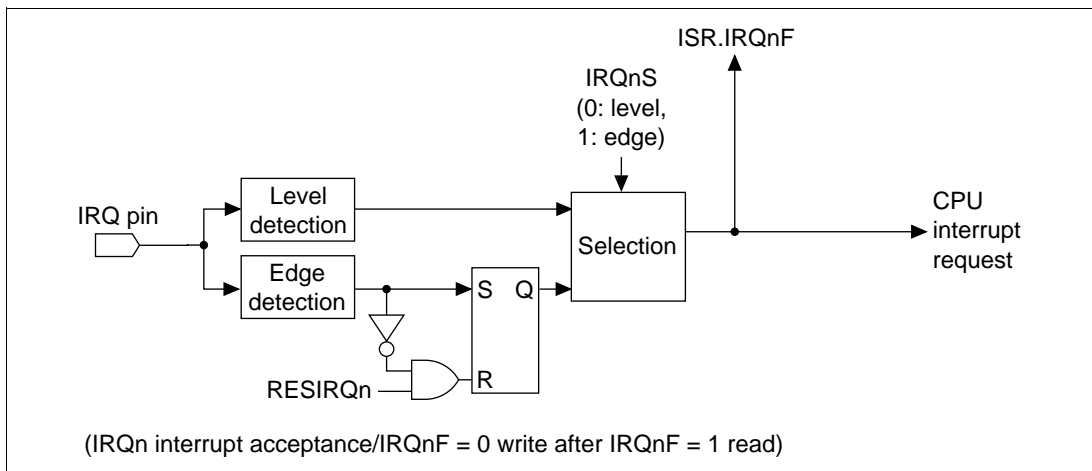
Bit:	7	6	5	4	3	2	1	0
	IRQ0F	IRQ1F	IRQ2F	IRQ3F	IRQ4F	IRQ5F	IRQ6F	IRQ7F
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- Bits 15 to 8—Reserved: These bits always read as 0. The write value should always be 0.

- Bits 7 to 0—IRQ0–IRQ7 Flags (IRQ0F–IRQ7F): These bits display the IRQ0–IRQ7 interrupt request status.

### Bits 7-0:

IRQ0F–IRQ7F	Detection Setting	Description
0	Level detection	No IRQn interrupt request exists. Clear conditions: When $\overline{\text{IRQn}}$ input is high level
	Edge detection	No IRQn interrupt request was detected. (initial value) Clear conditions: <ol style="list-style-type: none"> <li>1. When a 0 is written after reading <math>\text{IRQnF} = 1</math> status</li> <li>2. When IRQn interrupt exception processing has been executed</li> </ol>
1	Level detection	An IRQn interrupt request exists. Set conditions: When $\overline{\text{IRQn}}$ input is low level
	Edge detection	An IRQn interrupt request was detected. Set conditions: When a falling edge occurs at an $\overline{\text{IRQn}}$ input



**Figure 6.2 External Interrupt Process**

## 6.4 Interrupt Operation

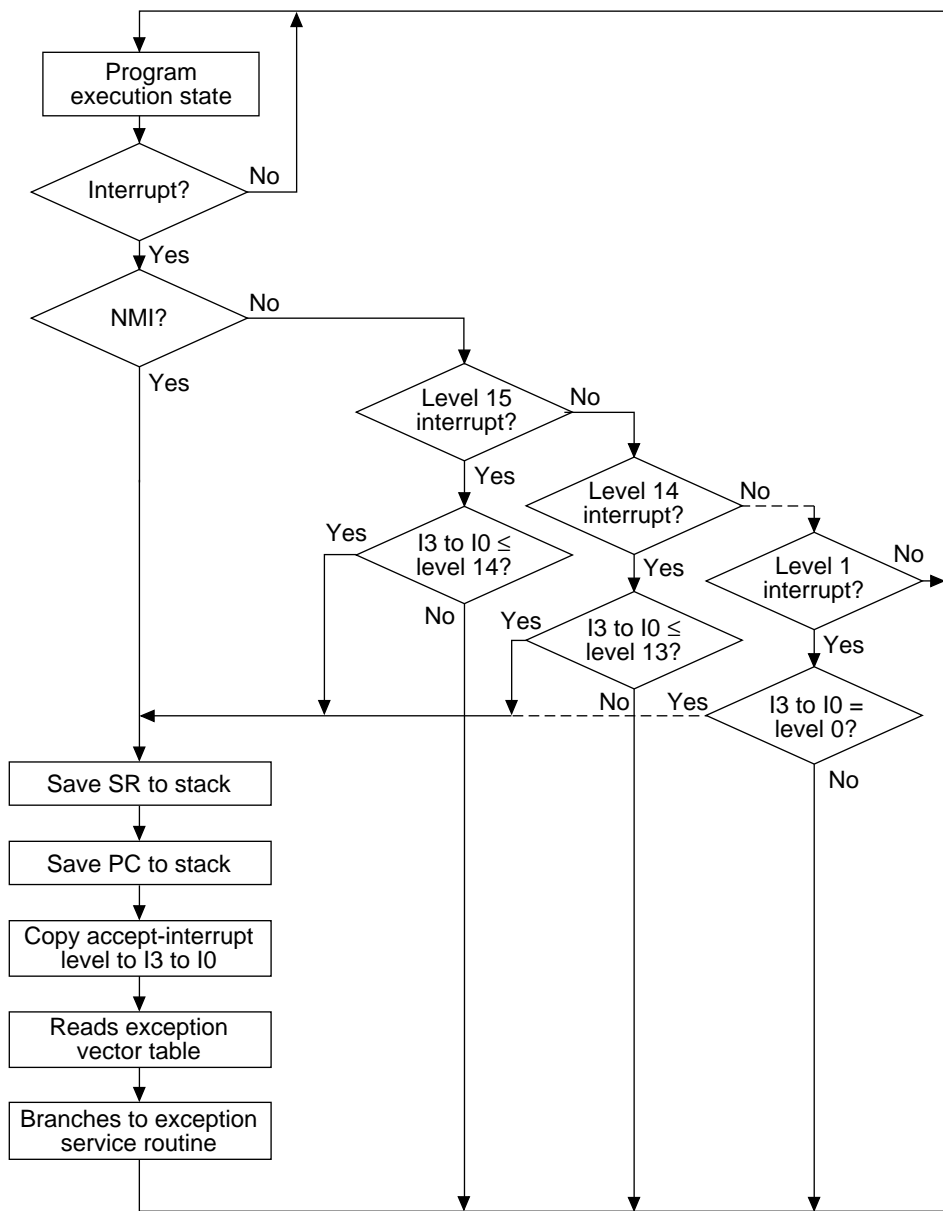
### 6.4.1 Interrupt Sequence

The sequence of interrupt operations is explained below. Figure 6.3 is a flowchart of the operations.

1. The interrupt request sources send interrupt request signals to the interrupt controller.
2. The interrupt controller selects the highest priority interrupt in the interrupt requests sent, following the priority levels set in interrupt priority level setting registers A–H (IPRA–IPRH). Lower-priority interrupts are ignored\*. If a number of interrupts with the same priority level occur, or if multiple interrupts occur within a single module, the interrupt with the highest default priority or the highest priority within its IPR setting range (as indicated in table 6.3) is selected.
3. The interrupt controller compares the priority level of the selected interrupt request with the interrupt mask bits (I3–I0) in the CPU's status register (SR). If the request priority level is equal to or less than the level set in I3–I0, the request is ignored. If the request priority level is higher than the level in bits I3–I0, the interrupt controller accepts the interrupt and sends an interrupt request signal to the CPU.
4. The interrupt controller detects the interrupt request sent from the interrupt controller when it decodes the next instruction to be executed. Instead of executing the decoded instruction, the CPU starts interrupt exception processing (figure 6.4).
5. The status register (SR) and program counter (PC) are saved onto the stack.
6. The priority level of the accepted interrupt is written to bits I3–I0 in SR.
7. The CPU reads the start address of the exception service routine from the exception vector table for the accepted interrupt, jumps to that address, and starts executing the program there. This jump is not a delay branch.

Note: An interrupt request for which edge detection has been set is held pending until it is accepted. However, an IRQ interrupt can be cleared by an IRQ status register (ISR) access. For details see section 6.2.3, IRQ Interrupts.

Pending edge-detected interrupts are cleared by a power-on reset.



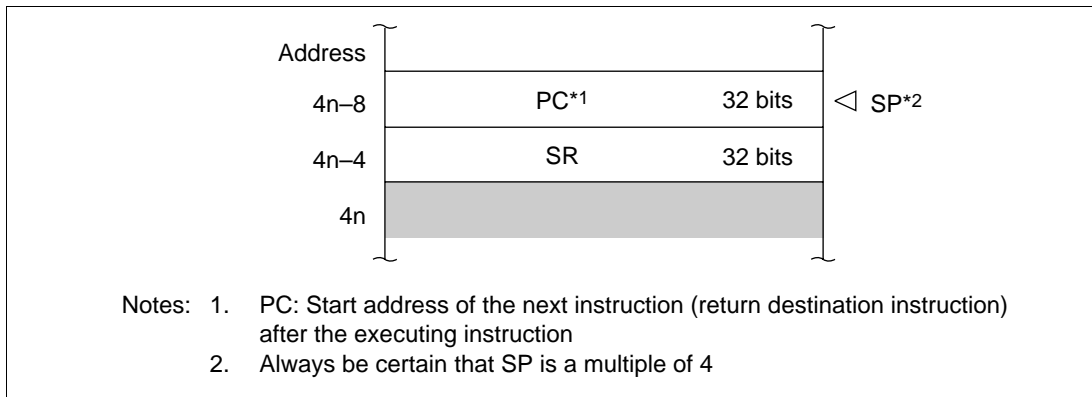
I3 to I0: Interrupt mask bits of status register

**Figure 6.3 Interrupt Sequence Flowchart**



## 6.4.2 Stack after Interrupt Exception Processing

Figure 6.4 shows the stack after interrupt exception processing.



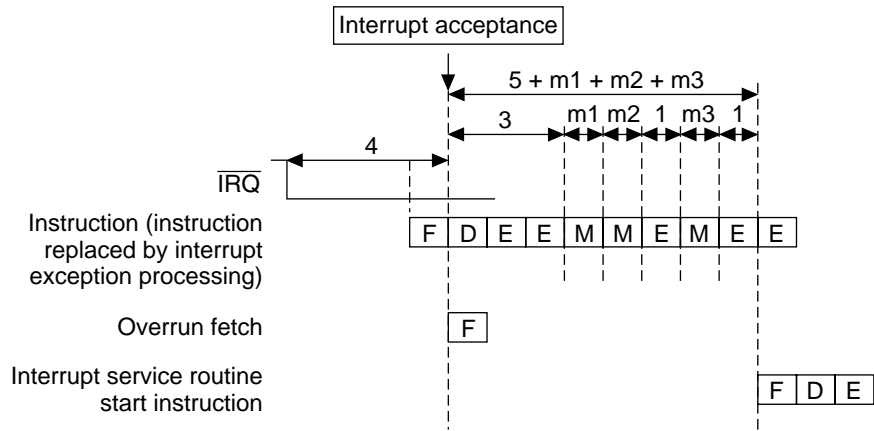
**Figure 6.4 Stack after Interrupt Exception Processing**

## 6.5 Interrupt Response Time

Table 6.5 indicates the interrupt response time, which is the time from the occurrence of an interrupt request until the interrupt exception processing starts and fetching of the first instruction of the interrupt service routine begins. Figure 6.5 shows the pipeline when an IRQ interrupt is accepted.

**Table 6.5 Interrupt Response Time**

Item	Number of States		Notes
	NMI, Peripheral Module	IRQ	
Compare identified interrupt priority with SR mask level	2 or 3	4	
Wait for completion of sequence currently being executed by CPU	$X (\geq 0)$		The longest sequence is for interrupt or address-error exception processing ( $X = 4 + m1 + m2 + m3 + m4$ ). If an interrupt-masking instruction follows, however, the time may be even longer.
Time from start of interrupt exception processing until fetch of first instruction of exception service routine starts	$5 + m1 + m2 + m3$		Performs the PC and SR saves and vector address fetch.
Interrupt response time	Total:	$7 + m1 + m2 + m3$	$9 + m1 + m2 + m3$
	Minimum:	10	12
	Maximum:	$12 + 2 (m1 + m2 + m3) + m4$	$13 + 2 (m1 + m2 + m3) + m4$
Note: When $m1 = m2 = m3 = m4 = 1$ $m1$ – $m4$ are the number of states needed for the following memory accesses. $m1$ : SR save (longword write) $m2$ : PC save (longword write) $m3$ : Vector address read (longword read) $m4$ : Fetch first instruction of interrupt service routine			
Response time	Minimum	$7 + m1 + m2 + m3$	$9 + m1 + m2 + m3$
		$0.5 \mu\text{s}^*$	$0.6 \mu\text{s}^*$
	Maximum	$12 + 2 (m1 + m2 + m3) + m4$	$13 + 2 (m1 + m2 + m3) + m4$
		$0.95 \mu\text{s}^*$	$1.0 \mu\text{s}^*$
Note: 20 MHz operation, $m1 = m2 = m3 = m4 = 1$			



- F: Instruction fetch (instruction fetched from memory where program is stored).
- D: Instruction decoding (fetched instruction is decoded).
- E: Instruction execution (data operation and address calculation is performed according to the results of decoding).
- M: Memory access (data in memory is accessed).

**Figure 6.5 Pipeline when an IRQ Interrupt is Accepted**

# Section 7 Bus State Controller (BSC)

## 7.1 Overview

The bus state controller (BSC) divides up the address spaces and outputs control for various types of memory. This enables memories like SRAM, and ROM to be linked directly to the LSI without external circuitry.

### 7.1.1 Features

- Address space is divided into four spaces
  - A maximum linear 4 Mbytes for each of address spaces CS0-CS3
  - 16-bit bus width
  - Wait states can be inserted by software for each space (0-3 waits)
  - In external memory space access, wait states can be inserted by the  $\overline{\text{WAIT}}$  pin
  - Outputs control signals for each space according to the type of memory connected
- RAM interface
  - On-chip RAM access of 32 bits in 1 state

## 7.1.2 Block Diagram

Figure 7.1 shows the BSC block diagram.

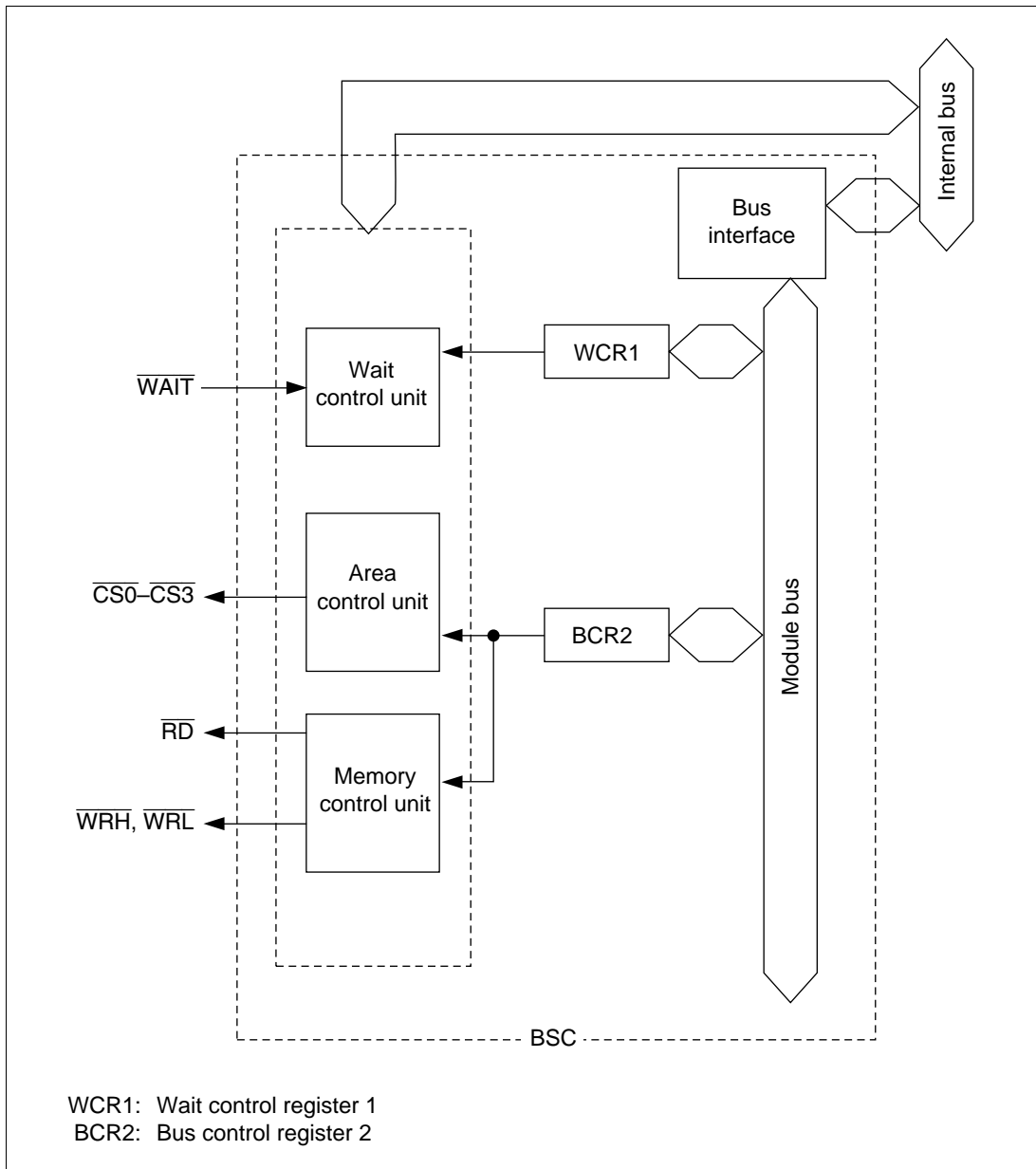


Figure 7.1 BSC Block Diagram

### 7.1.3 Pin Configuration

Table 7.1 shows the bus state controller pin configuration.

**Table 7.1 Pin Configuration**

Pin Name	I/O	Function
A21–A1	Output	Address output
D15–D0	I/O	16-bit data bus
$\overline{\text{CS0}}\text{--}\overline{\text{CS3}}$	Output	Chip select
$\overline{\text{RD}}$	Output	Strobe that indicates a read cycle for ordinary space/multiplex I/O
$\overline{\text{WRH}}$	Output	Strobe that indicates an upper byte (D15–D8) write cycle
$\overline{\text{WRL}}$	Output	Strobe that indicates a lower byte (D7–D0) write cycle
$\overline{\text{WAIT}}$	Input	Wait state request signal

### 7.1.4 Register Configuration

The bus state controller has two registers. The functions of these registers include control of wait states and interfaces with memories such as ROM and SRAM. The registers are summarized in figure table 7.2.

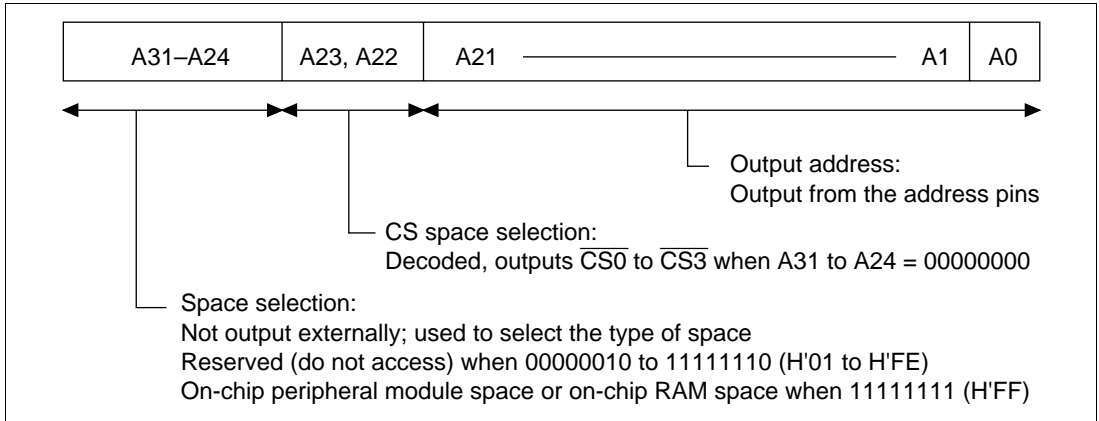
Both registers are 16 bits in size, and are initialized by a power-on reset.

**Table 7.2 Register Configuration**

Name	Abbr.	R/W	Initial Value	Address	Access Size
Bus control register 2	BCR2	R/W	H'FFFF	H'FFFF8622	8, 16
Wait state control register 1	WCR1	R/W	H'FFFF	H'FFFF8624	8, 16

## 7.1.5 Address Map

Figure 7.2 shows the address format used by the SH7092.



**Figure 7.2 Address Format**

This LSI uses 32-bit addresses:

- A31 to A24 are used to select the type of space and are not output externally.
- Bits A23 and A22 are decoded and output as chip select signals ( $\overline{CS0}$  to  $\overline{CS3}$ ) for the corresponding areas when bits A31 to A24 are 00000000.
- A21 to A1 are output externally.

Table 7.3 shows an address map.

**Table 7.3 Address Map**

Address	Space	Memory	Size	Bus Width
H'00200000 to H'003FFFFFFF	CS0 space	Ordinary space	4 Mbytes	16 bits
H'00400000 to H'007FFFFFFF	CS1 space	Ordinary space	4 Mbytes	16 bits
H'00800000 to H'00BFFFFFFF	CS2 space	Ordinary space	4 Mbytes	16 bits
H'00C00000 to H'00FFFFFFF	CS3 space	Ordinary space	4 Mbytes	16 bits
H'01000000 to H'FFFFFF7FFF	Reserved	Reserved		
H'FFFFFF8000 to H'FFFFFF87FF	On-chip peripheral module	On-chip peripheral module	2 kbytes	8/16 bits
H'FFFFFF8800 to H'FFFFFFE7FF	Reserved	Reserved		
H'FFFFFFE800 to H'FFFFFFFFFF	On-chip RAM	On-chip RAM	6 kbytes	32 bits

Note: Do not access reserved spaces. Operation cannot be guaranteed if they are accessed.

## 7.2 Description of Registers

### 7.2.1 Bus Control Register 2 (BCR2)

BCR2 is a 16-bit read/write register that specifies the number of idle cycles and  $\overline{CS}$  signal assert extension of each CS space.

BCR2 is initialized by power-on resets to H'FFFF.

Bit:	15	14	13	12	11	10	9	8
	IW31	IW30	IW21	IW20	IW11	IW10	IW01	IW00
Initial value:	1	1	1	1	1	1	1	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit:	7	6	5	4	3	2	1	0
	CW3	CW2	CW1	CW0	SW3	SW2	SW1	SW0
Initial value:	1	1	1	1	1	1	1	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- Bits 15–8—Idles between Cycles (IW31, IW30, IW21, IW20, IW11, IW10, IW01, IW00):  
These bits specify idle cycles inserted between consecutive accesses when the second one is to a different CS area after a read. Idles are used to prevent data conflict between ROM (and other memories, which are slow to turn the read data buffer off), fast memories, and I/O interfaces. Even when access is to the same area, idle cycles must be inserted when a read access is followed immediately by a write access. The idle cycles to be inserted comply with the area specification of the previous access. Refer to section 7.4, Waits between Access Cycles, for details.

IW31, IW30 specify the idle between cycles for CS3 space; IW21, IW20 specify the idle between cycles for CS2 space; IW11, IW10 specify the idle between cycles for CS1 space and IW01, IW00 specify the idle between cycles for CS0 space.

Bit 15 (IW31)	Bit 14 (IW30)	Description
0	0	No idle cycle after accessing CS3 space
	1	Inserts one idle cycle
1	0	Inserts two idle cycles
	1	Inserts three idle cycles (initial value)



Bit 13 (IW21)	Bit 12 (IW20)	Description
0	0	No idle cycle after accessing CS2 space
	1	Inserts one idle cycle
1	0	Inserts two idle cycles
	1	Inserts three idle cycles (initial value)

Bit 11 (IW11)	Bit 10 (IW10)	Description
0	0	No idle cycle after accessing CS1 space
	1	Inserts one idle cycle
1	0	Inserts two idle cycles
	1	Inserts three idle cycles (initial value)

Bit 9 (IW01)	Bit 8 (IW00)	Description
0	0	No idle cycle after accessing CS0 space
	1	Inserts one idle cycle
1	0	Inserts two idle cycles
	1	Inserts three idle cycles (initial value)

- Bits 7–4—Idle Specification for Continuous Access (CW3, CW2, CW1, CW0): The continuous access idle specification makes insertions to clearly delineate the bus intervals by once negating the  $\overline{CS}_n$  signal when doing consecutive accesses of the same CS space. When a write immediately follows a read, the number of idle cycles inserted is the larger of the two values specified by IW and CW. Refer to section 7.6, Waits between Access Cycles, for details.

CW3 specifies the continuous access idles for CS3 space; CW2 specifies the continuous access idles for CS2 space; CW1 specifies the continuous access idles for CS1 space and CW0 specifies the continuous access idles for CS0 space.

Bit 7 (CW3)	Description
0	No CS3 space continuous access idle cycles
1	One CS3 space continuous access idle cycle (initial value)

Bit 6 (CW2)	Description
0	No CS2 space continuous access idle cycles
1	One CS2 space continuous access idle cycle (initial value)

Bit 5 (CW1)	Description
0	No CS1 space continuous access idle cycles
1	One CS1 space continuous access idle cycle (initial value)

Bit 4 (CW0)	Description
0	No CS0 space continuous access idle cycles
1	One CS0 space continuous access idle cycle (initial value)

- Bits 3–0— $\overline{CS}$  Assert Extension Specification (SW3, SW2, SW1, SW0): The CS assert cycle extension specification is for making insertions to prevent extension of the  $\overline{RD}$  signal or  $\overline{WRx}$  signal assert period beyond the length of the  $\overline{CSn}$  signal assert period. Extended cycles insert one cycle before and after each bus cycle, which simplifies interfaces with external devices and also has the effect of extending write data hold time. Refer to section 7.3.3,  $\overline{CS}$  Assert Extension Function, for details.

SW3 specifies the  $\overline{CS}$  assert extension for CS3 space access; SW2 specifies the  $\overline{CS}$  assert extension for CS2 space access; SW1 specifies the  $\overline{CS}$  assert extension for CS1 space access and SW0 specifies the  $\overline{CS}$  assert extension for CS0 space access.

Bit 3 (SW3)	Description
0	No CS3 space $\overline{CS}$ assert extension
1	CS3 space $\overline{CS}$ assert extension (initial value)

Bit 2 (SW2)	Description
0	No CS2 space $\overline{CS}$ assert extension
1	CS2 space $\overline{CS}$ assert extension (initial value)

Bit 1 (SW1)	Description
0	No CS1 space $\overline{CS}$ assert extension
1	CS1 space $\overline{CS}$ assert extension (initial value)

Bit 0 (SW0)	Description
0	No CS0 space $\overline{CS}$ assert extension
1	CS0 space $\overline{CS}$ assert extension (initial value)

## 7.2.2 Wait Control Register 1 (WCR1)

Wait control register 1 (WCR1) is a 16-bit read/write register that specifies the number of wait cycles (0-3) for each CS space.

WCR1 is initialized to H'FFFF by power-on resets.

Bit:	15	14	13	12	11	10	9	8
	—	—	W31	W30	—	—	W21	W20
Initial value:	1	1	1	1	1	1	1	1
R/W:	R	R	R/W	R/W	R	R	R/W	R/W
Bit:	7	6	5	4	3	2	1	0
	—	—	W11	W10	—	—	W01	W00
Initial value:	1	1	1	1	1	1	1	1
R/W:	R	R	R/W	R/W	R	R	R/W	R/W

- Bits 15 and 14—Reserved. These bits always read 1. The write value should always be 1.
- Bits 13 and 12—CS3 Space Wait Specification (W31, W30): These bits specify the number of waits for CS3 space accesses.

Bit 13 (W31)	Bit 12 (W30)	Description
0	0	No wait (external wait input disabled)
	1	1-wait external wait input enabled
1	0	2-wait external wait input enabled
	1	3-wait external wait input enabled (initial value)

- Bits 11 and 10—Reserved. These bits always read 1. The write value should always be 1.
- Bits 9 and 8—CS2 Space Wait Specification (W21, W20): These bits specify the number of waits for CS2 space accesses.

Bit 9 (W21)	Bit 8 (W20)	Description
0	0	No wait (external wait input disabled)
	1	1-wait external wait input enabled
1	0	2-wait external wait input enabled
	1	3-wait external wait input enabled (initial value)

- Bits 7 and 6—Reserved. These bits always read 1. The write value should always be 1.
- Bits 5 and 4—CS1 Space Wait Specification (W11, W10): These bits specify the number of waits for CS1 space accesses.

Bit 5 (W11)	Bit 4 (W10)	Description
0	0	No wait (external wait input disabled)
	1	1-wait external wait input enabled
1	0	2-wait external wait input enabled
	1	3-wait external wait input enabled (initial value)

- Bits 3 and 2—Reserved. These bits always read 1. The write value should always be 1.
- Bits 1 and 0—CS0 Space Wait Specification (W01, W00): These bits specify the number of waits for CS0 space accesses.

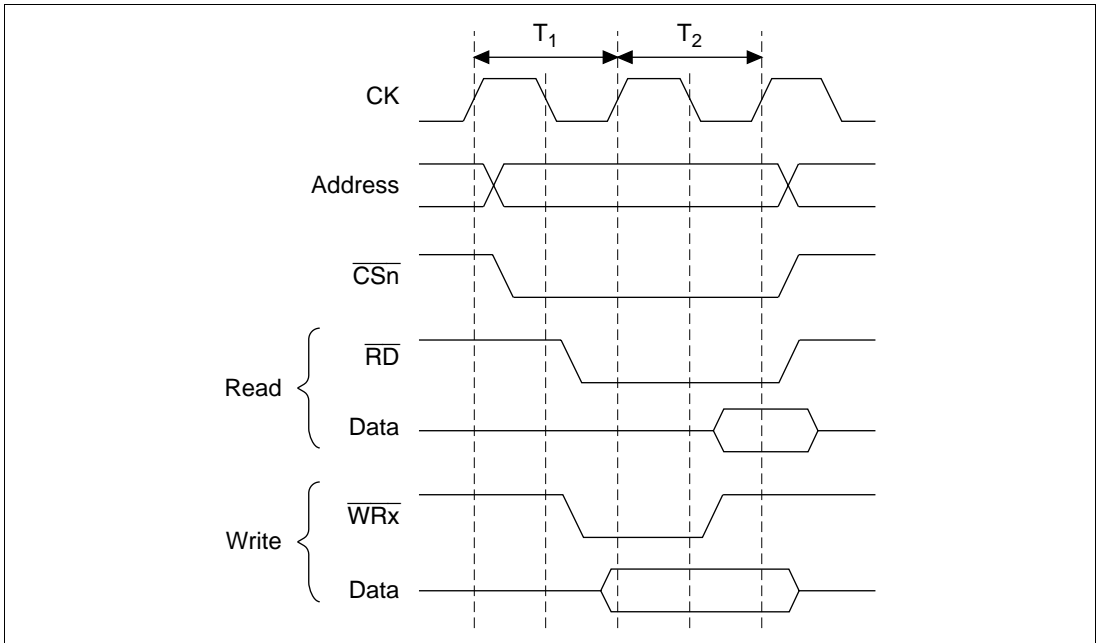
Bit 1 (W01)	Bit 0 (W00)	Description
0	0	No wait (external wait input disabled)
	1	1-wait external wait input enabled
1	0	2-wait external wait input enabled
	1	3-wait external wait input enabled (initial value)

## 7.3 Accessing Ordinary Space

A strobe signal is output by ordinary space accesses to provide primarily for SRAM or ROM direct connections.

### 7.3.1 Basic Timing

Figure 7.3 shows the basic timing of ordinary space accesses. Ordinary access bus cycles are performed in 2 states.



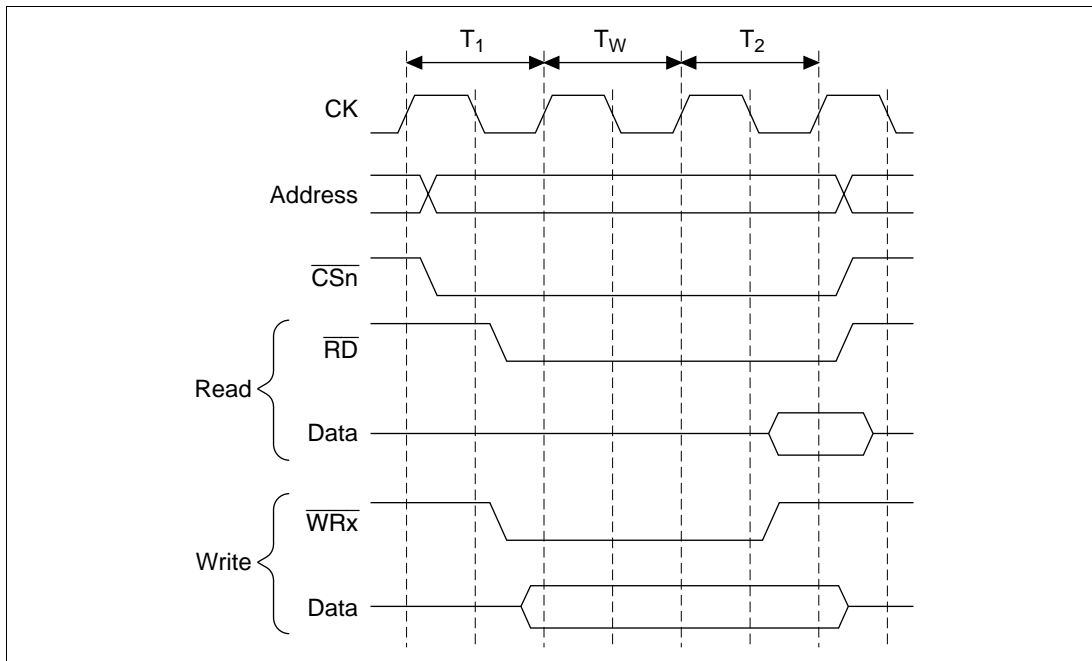
**Figure 7.3 Basic Timing of Ordinary Space Access**

During a read, irrespective of operand size, all bits in the data bus width for the access space (address) are fetched by the LSI on  $\overline{RD}$ , using the required byte locations.

During a write, the following signals are associated with transfer of these actual byte locations:  $\overline{WRH}$  (bits 15–8) and  $\overline{WRL}$  (bits 7–0).

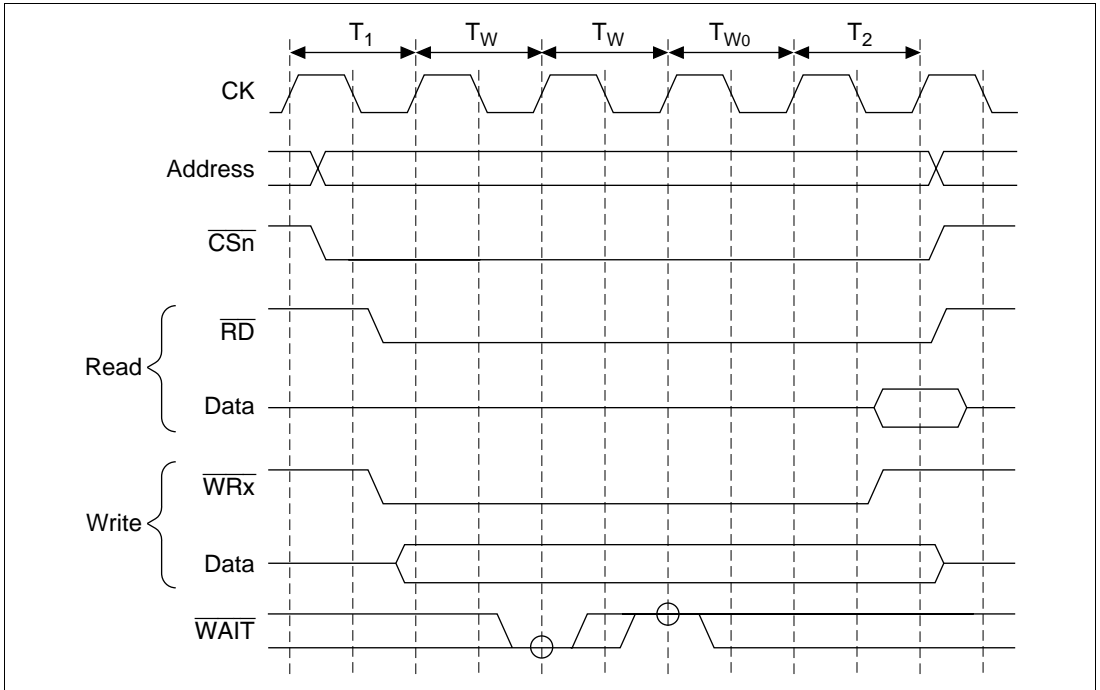
### 7.3.2 Wait State Control

The number of wait states inserted into ordinary space access states can be controlled using the WCR settings. The specified number of  $T_w$  cycles (0–3 waits) are inserted as software wait cycles with the timing shown in figure 7.4.



**Figure 7.4 Wait Timing of Ordinary Space Access (Software Wait Only)**

When the wait is specified by software using WCR, the wait input  $\overline{\text{WAIT}}$  signal from outside is sampled. Figure 7.5 shows the  $\overline{\text{WAIT}}$  signal sampling. The  $\overline{\text{WAIT}}$  signal is sampled at the clock rise one cycle before the clock rise when  $T_w$  state shifts to  $T_2$  state.



**Figure 7.5 Wait State Timing of Ordinary Space Access (Wait States from Software Wait 2 State +  $\overline{\text{WAIT}}$  Signal)**

### 7.3.3 $\overline{CS}$ Assert Period Extension

Idle cycles can be inserted to prevent extension of the  $\overline{RD}$  signal or  $\overline{WRx}$  signal assert period beyond the length of the  $\overline{CSn}$  signal assert period by setting the SW3–SW0 bits of BCR2. This allows for flexible interfaces with external circuitry. The timing is shown in figure 7.6.  $T_h$  and  $T_f$  cycles are added respectively before and after the ordinary cycle. Only  $\overline{CSn}$  is asserted in these cycles;  $\overline{RD}$  and  $\overline{WRx}$  signals are not. Further, data is extended up to the  $T_f$  cycle, which is effective for gate arrays and the like, which have slower write operations.

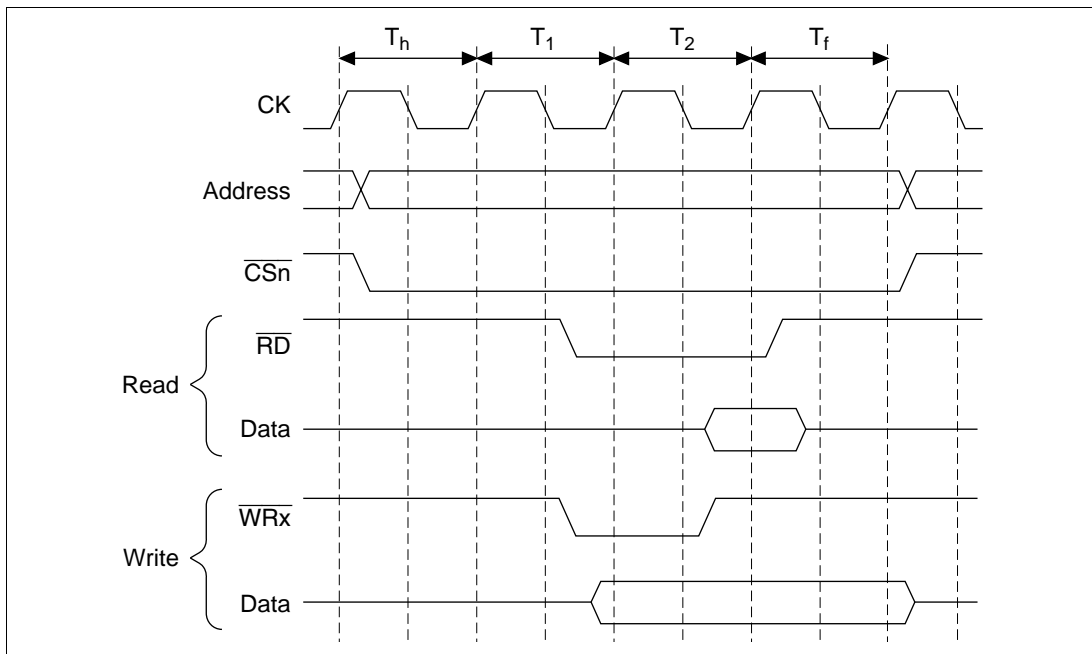


Figure 7.6  $\overline{CS}$  Assert Period Extension Function



## 7.4 Waits between Access Cycles

When a read from a slow device is completed, data buffers may not go off in time to prevent data conflicts with the next access. If there is a data conflict during memory access, the problem can be solved by inserting a wait in the access cycle.

To enable detection of bus cycle starts, waits can be inserted between access cycles during continuous accesses of the same CS space by negating the  $\overline{CSn}$  signal once.

### 7.4.1 Prevention of Data Bus Conflicts

For the two cases of write cycles after read cycles, and read cycles for a different area after read cycles, waits are inserted so that the number of idle cycles specified by the IW31 to IW00 bits of the BCR2. When idle cycles already exist between access cycles, only the number of empty cycles remaining beyond the specified number of idle cycles are inserted.

Figure 7.7 shows an example of idles between cycles. In this example, 1 idle between  $\overline{CSn}$  space cycles has been specified, so when a  $\overline{CSm}$  space write immediately follows a  $\overline{CSn}$  space read cycle, 1 idle cycle is inserted.

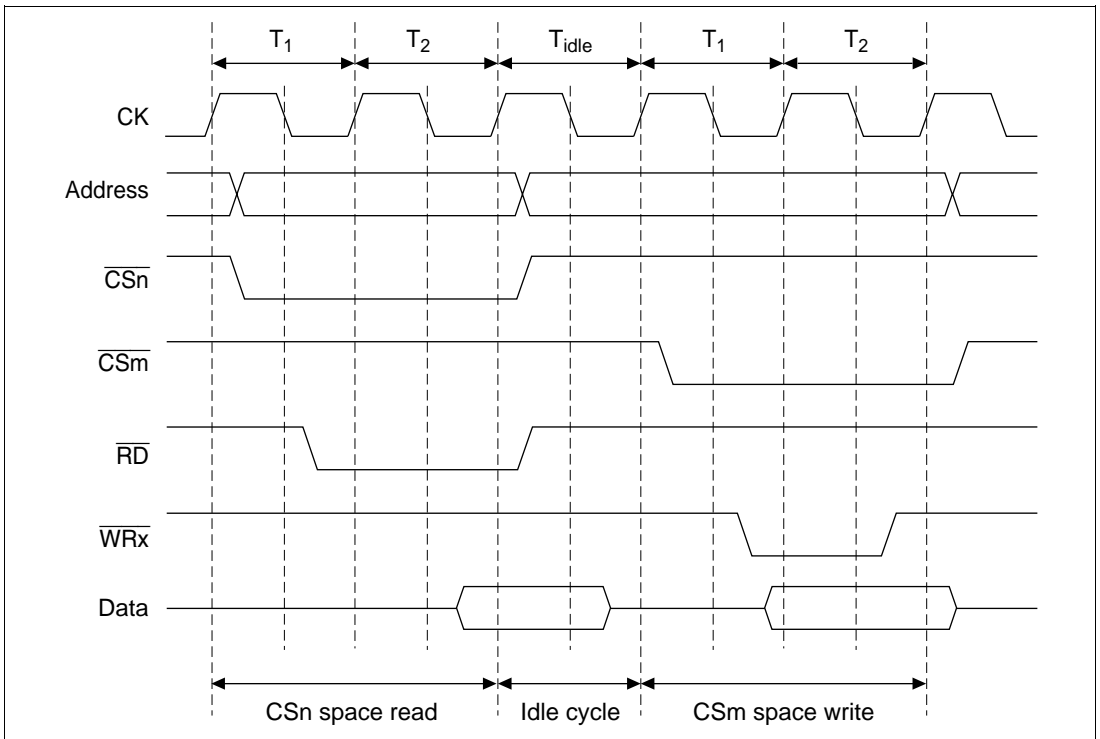


Figure 7.7 Idle Cycle Insertion Example

IW31 and IW30 specify the number of idle cycles required after a CS3 space read either to read other external spaces, or for this LSI, to do write accesses. In the same manner, IW21 and IW20 specify the number of idle cycles after a CS2 space read, IW11 and IW10, the number after a CS1 space read, and IW01 and IW00, the number after a CS0 space read.

0 to 3 cycles can be specified for CS space.

#### 7.4.2 Simplification of Bus Cycle Start Detection

For consecutive accesses of the same CS space, waits are inserted so that the number of idle cycles designated by the CW3 to CW0 bits of the BCR2 occur. However, for write cycles after reads, the number of idle cycles inserted will be the larger of the two values defined by the IW and CW bits. When idle cycles already exist between access cycles, waits are not inserted. Figure 7.8 shows an example. A continuous access idle is specified for CSn space, and CSn space is consecutively write accessed.

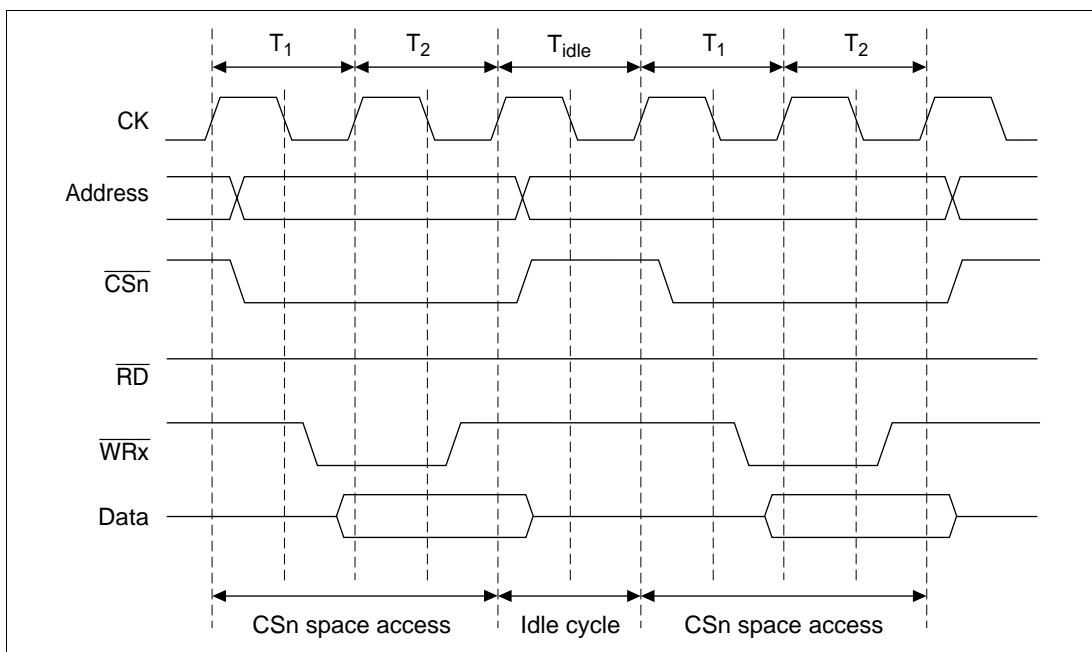


Figure 7.8 Same Space Consecutive Access Idle Cycle Insertion Example

## 7.5 Memory Connection Examples

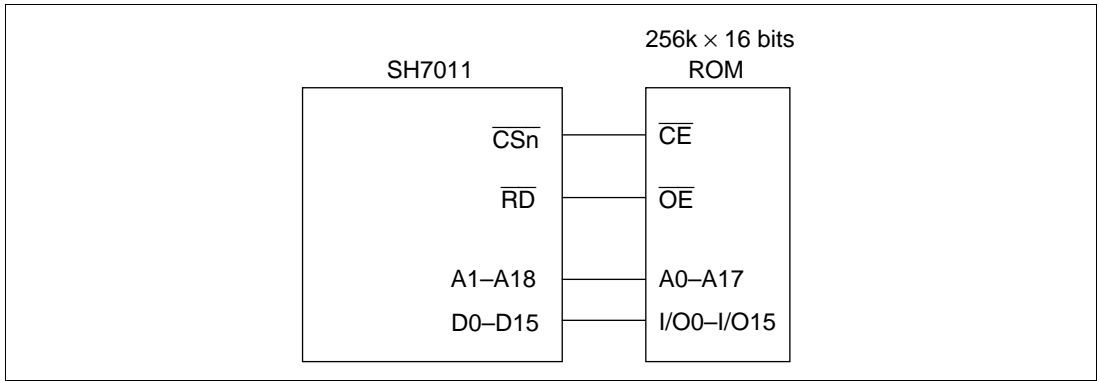


Figure 7.9 16-Bit Data Bus Width ROM Connection

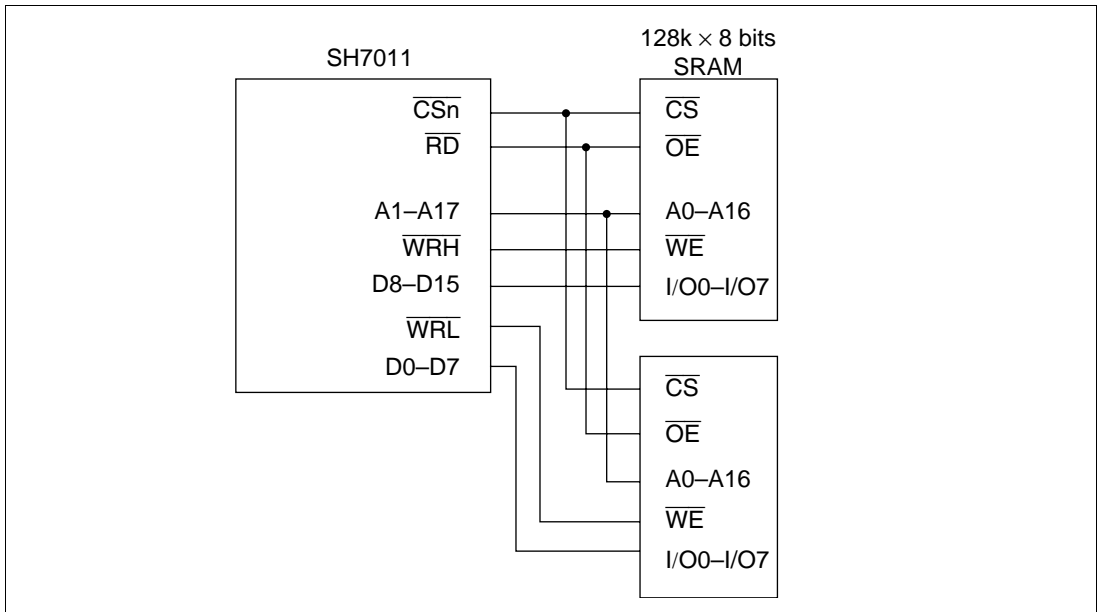


Figure 7.10 16-Bit Data Bus Width SRAM Connection

# Section 8 Multifunction Timer Pulse Unit (MTU)

## 8.1 Overview

The SH microprocessor has an on-chip 16-bit multifunction timer pulse unit (MTU) with three channels of 16-bit timers.

### 8.1.1 Features

- Can process a maximum of six different pulse outputs and inputs.
- Has eight timer general registers (TGR), four for channel 0 and two each for channels 1 and 2, that can be set to function independently as output compare registers or (except for TGR0B and TGR0D of channel 0) as input capture registers. The channel 0 TGRC and TGRD registers can be used as buffer registers.
- Can select six counter input clock sources for all channels
- All channels can be set for the following operating modes:
  - Compare match waveform output: 0 output/1 output/toggle output selectable.
  - Input capture function: Selectable rising edge, falling edge, or both rising and falling edge detection.
  - Counter clearing function: Counters can be cleared by a compare-match or input capture.
  - Synchronizing mode: Two or more timer counters (TCNT) can be written to simultaneously. Two or more timer counters can be simultaneously cleared by a compare-match or input capture. Counter synchronization functions enable synchronized register input/output.
  - PWM mode: PWM output can be provided with any duty cycle. When combined with the counter synchronizing function, enables up to four-phase\* PWM output.

Note: \* When channels 0 to 2 are set to PWM mode 1

- Channel 0 can be set for buffer operation
  - Input capture register double buffer configuration possible
  - Output compare register automatic re-write possible
- Cascade connection operation
  - Can be operated as a 32-bit counter by using the channel 2 input clock for channel 1 overflow/underflow
- High speed access via internal 16-bit bus
- Eleven interrupt sources
  - Channel 0 has two dual-function compare-match/input capture interrupts, two compare-match interrupts, and one overflow interrupt, which can be requested independently.

- Channels 1 and 2 have two compare-match/input capture interrupts, one overflow interrupt, and one underflow interrupt which can be requested independently.
- A/D converter conversion start trigger can be generated
  - Channel 0 to 2 compare-match/input capture signals can be used as A/D converter conversion start triggers.

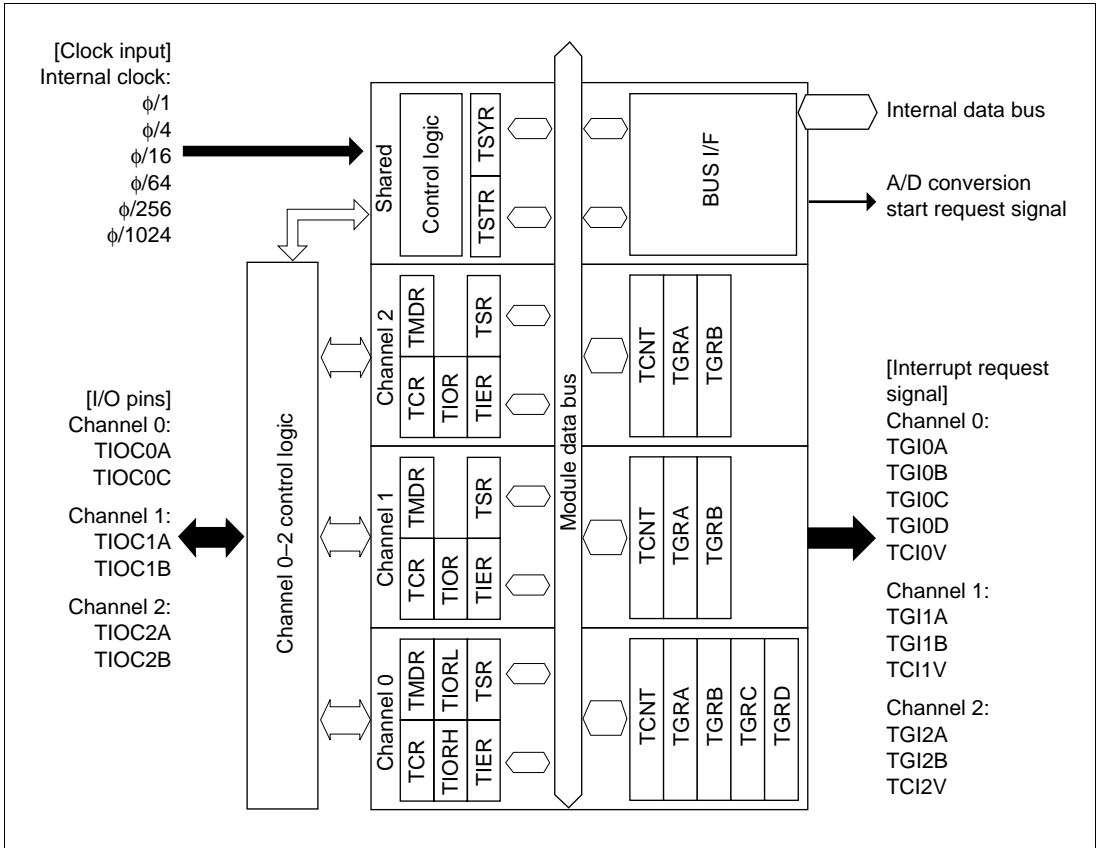
Table 8.1 summarizes the MTU functions.

**Table 8.1 MTU Functions**

Item	Channel 0	Channel 1	Channel 2
Counter clocks	Internal: $\phi/1$ , $\phi/4$ , $\phi/16$ , $\phi/64$ , $\phi/256$ , $\phi/1024$ Six to each channel		
General registers	TGR0A	TGR1A	TGR2A
	TGR0B	TGR1B	TGR2B
General registers/buffer registers	TGR0C	No	No
	TGR0D		
Input/output pins	TIOC0A	TIOC1A	TIOC2A
	TIOC0C		
Counter clear function	TGR compare-match or input capture	TGR compare-match or input capture	TGR compare-match or input capture
Compare match output	0	Yes	Yes
	1	Yes	Yes
	Toggle	Yes	Yes
Input capture function	Yes	Yes	Yes
Synchronization	Yes	Yes	Yes
Buffer operation	Yes	No	No
PWM mode 1	Yes	Yes	Yes
PWM mode 2	Yes	Yes	Yes
A/D conversion start trigger	TGR0A compare match or input capture	TGR1A compare match or input capture	TGR2A compare match or input capture
Interrupt sources	Compare match/input capture 0A	Compare match/input capture 1A	Compare match/input capture 2A
	Compare match 0B	Compare match/input capture 1B	Compare match/input capture 2B
	Compare match/input capture 0C	Overflow	Overflow
	Compare match 0D	—	—
	Overflow	—	—

## 8.1.2 Block Diagram

Figure 8.1 is the block diagram of the MTU.



**Figure 8.1 MTU Block Diagram**

### 8.1.3 Pin Configuration

Table 8.2 summarizes the MTU pins.

**Table 8.2 Pin Configuration**

Channel	Name	Pin Name	I/O	Function
0	Input capture/output compare-match 0A	TIOC0A	I/O	TGR0A input capture input/output compare output/PWM output pin
	Input capture/output compare-match 0C	TIOC0C	I/O	TGR0C input capture input/output compare output/PWM output pin
1	Input capture/output compare-match 1A	TIOC1A	I/O	TGR1A input capture input/output compare output/PWM output pin
	Input capture/output compare-match 1B	TIOC1B	I/O	TGR1B input capture input/output compare output/PWM output pin
2	Input capture/output compare-match 2A	TIOC2A	I/O	TGR2A input capture input/output compare output/PWM output pin
	Input capture/output compare-match 2B	TIOC2B	I/O	TGR2B input capture input/output compare output/PWM output pin

Note: The TIOC pins output undefined values when they are set to input capture and timer output by the pin function controller (PFC).



## 8.1.4 Register Configuration

Table 8.3 summarizes the MTU register configuration.

**Table 8.3 Register Configuration**

Channel	Name	Abbreviation	R/W	Initial Value	Address	Access Size (Bits) *1	
Shared	Timer start register	TSTR	R/W	H'00	H'FFFF8240	8, 16	
	Timer synchro register	TSYR	R/W	H'00	H'FFFF8241		
0	Timer control register 0	TCR0	R/W	H'00	H'FFFF8260	8, 16, 32	
	Timer mode register 0	TMDR0	R/W	H'C0	H'FFFF8261		
	Timer I/O control register 0H	TIOR0H	R/W	H'00	H'FFFF8262		
	Timer I/O control register 0L	TIOR0L	R/W	H'00	H'FFFF8263		
	Timer interrupt enable register 0	TIER0	R/W	H'40	H'FFFF8264		
	Timer status register 0	TSR0	R/(W)*2	H'C0	H'FFFF8265		
	Timer counter 0	TCNT0	R/W	H'0000	H'FFFF8266	16, 32	
	General register 0A	TGR0A	R/W	H'FFFF	H'FFFF8268		
	General register 0B	TGR0B	R/W	H'FFFF	H'FFFF826A		
	General register 0C	TGR0C	R/W	H'FFFF	H'FFFF826C		
	General register 0D	TGR0D	R/W	H'FFFF	H'FFFF826E		
	1	Timer control register 1	TCR1	R/W	H'00	H'FFFF8280	8, 16
		Timer mode register 1	TMDR1	R/W	H'C0	H'FFFF8281	
Timer I/O control register 1		TIOR1	R/W	H'00	H'FFFF8282	8	
Timer interrupt enable register 1		TIER1	R/W	H'40	H'FFFF8284	8, 16, 32	
Timer status register 1		TSR1	R/(W)*2	H'C0	H'FFFF8285		
Timer counter 1		TCNT1	R/W	H'0000	H'FFFF8286	16, 32	
General register 1A		TGR1A	R/W	H'FFFF	H'FFFF8288		
General register 1B		TGR1B	R/W	H'FFFF	H'FFFF828A		

**Table 8.3 Register Configuration (cont)**

Channel	Name	Abbreviation	R/W	Initial Value	Address	Access Size (Bits) *1
2	Timer control register 2	TCR2	R/W	H'00	H'FFFF82A0	8, 16
	Timer mode register 2	TMDR2	R/W	H'C0	H'FFFF82A1	
	Timer I/O control register 2	TIOR2	R/W	H'00	H'FFFF82A2	8
	Timer interrupt enable register 2	TIER2	R/W	H'40	H'FFFF82A4	8, 16, 32
	Timer status register 2	TSR2	R/(W)*2	H'C0	H'FFFF82A5	
	Timer counter 2	TCNT2	R/W	H'0000	H'FFFF82A6	16, 32
	General register 2A	TGR2A	R/W	H'FFFF	H'FFFF82A8	
	General register 2B	TGR2B	R/W	H'FFFF	H'FFFF82AA	

Notes: Do not access empty addresses.

1. 16-bit registers (TCNT, TGR) cannot be read or written in 8-bit units.
2. Write 0 to clear flags.

## 8.2 MTU Register Descriptions

### 8.2.1 Timer Control Register (TCR)

The TCR is an 8-bit read/write register for controlling the TCNT counter for each channel. The MTU has three TCR registers, one for each of the channels 0 to 2. TCR is initialized to H'00 by a power-on reset or the standby mode.

#### Channel 0: TCR0

Bit:	7	6	5	4	3	2	1	0
	CCLR2	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

#### Channels 1, 2: TCR1, TCR2

Bit:	7	6	5	4	3	2	1	0
	—	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- Bits 7–5—Counter Clear 2, 1, 0 (CCLR2, CCLR1, CCLR0): Select the counter clear source for the TCNT counter.

### Channels 0:

Bit 7: CCLR2	Bit 6: CCLR1	Bit 5: CCLR0	Description
0	0	0	TCNT clear disabled (initial value)
		1	TCNT is cleared by TGRA compare-match or input capture
	1	0	TCNT is cleared by TGRB compare-match
		1	Synchronizing clear: TCNT is cleared in synchronization with clear of other channel counters operating in sync.* <sup>1</sup>
1	0	0	TCNT clear disabled
		1	TCNT is cleared by TGRC compare-match or input capture* <sup>2</sup>
	1	0	TCNT is cleared by TGRD compare-match* <sup>2</sup>
		1	Synchronizing clear: TCNT is cleared in synchronization with clear of other channel counters operating in sync* <sup>1</sup>

- Notes: 1. Setting the SYNC bit of the TSYR to 1 sets the synchronization.  
 2. When TGRC or TGRD are functioning as buffer registers, TCNT is not cleared because the buffer registers have priority and compare-match/input captures do not occur.

### Channels 1, 2:

Bit 7: Reserved* <sup>1</sup>	Bit 6: CCLR1	Bit 5: CCLR0	Description
0	0	0	TCNT clear disabled (initial value)
		1	TCNT is cleared by TGRA compare-match or input capture
	1	0	TCNT is cleared by TGRB compare-match or input capture
		1	Synchronizing clear: TCNT is cleared in synchronization with clear of other channel counters operating in sync* <sup>2</sup>

- Notes: 1. The bit 7 of channels 1 and 2 is reserved. It always reads 0, and cannot be modified.  
 2. Setting the SYNC bit of the TSYR to 1 sets the synchronization.

- Bits 4–3—Clock Edge 1, 0 (CKEG1 and CKEG0): CKEG1 and CKEG0 select the input clock edges. When counting is done on both edges of the internal clock the input clock frequency becomes 1/2 (Example: both edges of  $\phi/4 =$  rising edge of  $\phi/2$ ).

**Bit 4: CKEG1**   **Bit 3: CKEG0**   **Description**

0	0	Count on rising edges (initial value)
	1	Count on falling edges
1	X	Count on both rising and falling edges

Notes: 1. X: 0 or 1, don't care.

2. Internal clock edge selection is effective when the input clock is  $\phi/4$  or slower. These settings are ignored when  $\phi/1$ , or the overflow of another channel is selected for the input clock.

- Bits 2–0—Timer Prescaler 2–0 (TPSC2–TPSC0): TPSC2–TPSC0 select the counter clock source for the TCNT. An independent clock source can be selected for each channel. Table 8.4 shows the possible settings for each channel.

**Table 8.4 MTU Clock Sources**

Channel	Internal Clock					Other Channel Overflow	
	$\phi/1$	$\phi/4$	$\phi/16$	$\phi/64$	$\phi/256$		$\phi/1024$
0	O	O	O	O	X	X	X
1	O	O	O	O	O	X	O
2	O	O	O	O	X	O	X

Note: Symbols: O: Setting possible      X: Setting not possible

**Channel 0:**

**Bit 2: TPSC2**   **Bit 1: TPSC1**   **Bit 0: TPSC0**   **Description**

0	0	0	Internal clock: count with $\phi/1$ (initial value)
		1	Internal clock: count with $\phi/4$
	1	0	Internal clock: count with $\phi/16$
		1	Internal clock: count with $\phi/64$
1	0	0	Reserved (Do not set)
		1	Reserved (Do not set)
	1	0	Reserved (Do not set)
		1	Reserved (Do not set)

## Channel 1:

Bit 2: TPSC2	Bit 1: TPSC1	Bit 0: TPSC0	Description
0	0	0	Internal clock: count with $\phi/1$ (initial value)
		1	Internal clock: count with $\phi/4$
	1	0	Internal clock: count with $\phi/16$
		1	Internal clock: count with $\phi/64$
1	0	0	Reserved (Do not set)
		1	Reserved (Do not set)
	1	0	Internal clock: count with $\phi/256$
		1	Count with the TCNT2 overflow

## Channel 2:

Bit 2: TPSC2	Bit 1: TPSC1	Bit 0: TPSC0	Description
0	0	0	Internal clock: count with $\phi/1$ (initial value)
		1	Internal clock: count with $\phi/4$
	1	0	Internal clock: count with $\phi/16$
		1	Internal clock: count with $\phi/64$
1	0	0	Reserved (Do not set)
		1	Reserved (Do not set)
	1	0	Reserved (Do not set)
		1	Internal clock: count with $\phi/1024$

## 8.2.2 Timer Mode Register (TMDR)

The TMDR is an 8-bit read/write register that sets the operating mode for each channel. The MTU has three TMDR registers, one for each channel. TMDR is initialized to H'C0 by a power-on reset.

### Channel 0: TMDR0

Bit:	7	6	5	4	3	2	1	0
	—	—	BFB	BFA	MD3	MD2	MD1	MD0
Initial value:	1	1	0	0	0	0	0	0
R/W:	R	R	R/W	R/W	R/W	R/W	R/W	R/W

### Channels 1, 2: TMDR1, TMDR2

Bit:	7	6	5	4	3	2	1	0
	—	—	—	—	MD3	MD2	MD1	MD0
Initial value:	1	1	0	0	0	0	0	0
R/W:	R	R	R	R	R/W	R/W	R/W	R/W

- Bits 7, 6—Reserved: These bits are reserved. They always read as 1, and cannot be modified.
- Bit 5—Buffer Operation B (BFB): Designates whether to use the TGRB register for normal operation, or buffer operation in combination with the TGRD register. When using TGRD as a buffer register, no TGRD register input capture/output compares are generated.

This bit is reserved in channels 1 and 2, which have no TGRD registers. It is always read as 0, and cannot be modified.

Bit 5: BFB	Description
------------	-------------

0	TGRB operates normally (initial value)
---	--

1	TGRB and TGRD buffer operation
---	--------------------------------

- Bit 4—Buffer Operation A (BFA): Designates whether to use the TGRA register for normal operation, or buffer operation in combination with the TGRC register. When using TGRC as a buffer register, no TGRC register input capture/output compares are generated.

This bit is reserved in channels 1 and 2, which have no TGRC registers. It is always read as 0, and cannot be modified.

Bit 4: BFA	Description
------------	-------------

0	TGRA operates normally (initial value)
---	--

1	TGRA and TGRC buffer operation
---	--------------------------------

- Bits 3–0—Modes 3–0 (MD3–MD0): These bits set the timer operation mode.

Bit 3: MD3	Bit 2: MD2	Bit 1: MD1	Bit 0: MD0	Description
0	0	0	0	Normal operation (initial value)
			1	Reserved (do not set)
		1	0	PWM mode 1
			1	PWM mode 2
0	1	*	*	Reserved (Do not set)
1	*	*	*	Reserved (Do not set)

\*: Don't care

### 8.2.3 Timer I/O Control Register (TIOR)

The TIOR is a register that controls the TGR. The MTU has four TIOR registers, two for channels 0, and one each for channels 1 and 2. TIOR is initialized to H'00 by a power-on reset.

#### Channel 0: TIOR0H

Bit:	7	6	5	4	3	2	1	0
	—	—	—	—	IOA3	IOA2	IOA1	IOA0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R/W	R/W	R/W	R/W

#### Channels 1, 2: TIOR1, TIOR2

Bit:	7	6	5	4	3	2	1	0
	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bits 7–4—I/O Control B3–B0 (IOB3–IOB0): These bits set the TGRB register function.

(TIOR1 and TIOR2 only. TIOR0H is a reserved bit: it always reads 0 and its write value should always be 0.)

Bits 3–0—I/O Control A3–B0 (IOA3–IOA0): These bits set the TGRA register function.

## Channel 0: TIOR0L

Bit:	7	6	5	4	3	2	1	0
	—	—	—	—	IOC3	IOC2	IOC1	IOC0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R/W	R/W	R/W	R/W

Note: When the TGRC or TGRD registers are set for buffer operation, these settings become ineffective and the operation is as a buffer register.

- Bits 7–4—Reserved. These bits always read 0. The write value should always be 0.
- Bits 3–0—I/O Control C3–C0 (IOC3–IOC0): These bits set the TGRC register function.

## Channel 0 (TIOR0H Register):

- Bits 7–4—Reserved. These bits always read 0. The write value should always be 0.
- Bits 3–0—I/O Control A3–A0 (IOA3–IOA0): These bits set the TGR0A register function.

### Bit 3: IOA3    Bit 2: IOA2    Bit 1: IOA1    Bit 0: IOA0

IOA3	IOA2	IOA1	IOA0	Description
0	0	0	0	TGR0A Output disabled (initial value)
			1	is an Initial output is 0
		1	0	output compare register
			1	Output 0 on compare-match Output 1 on compare-match Toggle output on compare-match
	1	0	0	Output disabled
			1	Initial output is 1
		1	0	Output 0 on compare-match Output 1 on compare-match Toggle output on compare-match
			1	
1	0	0	0	TGR0A Capture input source is the TIOC0A pin
			1	Input capture on rising edge Input capture on falling edge Input capture on both edges
		1	0	capture register
			1	Input capture on TCNT1 count up/count down
	1	0	0	Capture input source is channel 1/ count clock
			1	
		1	0	
			1	



### Channel 0 (TIOR0L Register):

- Bits 7–4—Reserved. These bits always read 0. The write value should always be 0.
- Bits 3–0—I/O Control C3–C0 (IOC3–IOC0): These bits set the TGR0C register function.

Bit 3: IOC3	Bit 2: IOC2	Bit 1: IOC1	Bit 0: IOC0	Description			
0	0	0	0	TGR0C	Output disabled (initial value)		
			1	is an	Initial	Output 0 on compare-match	
		1	0	output	output	Output 1 on compare-match	
			1	compare register	is 0	Toggle output on compare-match	
	1	0	0		Output disabled		
			1		Initial	Output 0 on compare-match	
		1	0		output	Output 1 on compare-match	
			1		is 1	Toggle output on compare-match	
1	0	0	0	TGR0C	Capture	Input capture on rising edge	
			1	is an	input source	Input capture on falling edge	
		1	0	input	is the	Input capture on both edges	
			1	capture register	TIOC0C pin		
	1	0	0		Capture	Input capture	
			1		input source	on TCNT1	
		1	0		is channel 1/	count up/count down	
			1		count clock		

Note: When the BFA bit of TMDR0 is set to 1 and TGR0C is being used as a buffer register, these settings become ineffective and input capture/output compares do not occur.

## Channel 1 (TIOR1 Register):

- Bits 7–4—I/O Control B3–B0 (IOB3–IOB0): These bits set the TGR1B register function.

Bit 7: IOB3	Bit 6: IOB2	Bit 5: IOB1	Bit 4: IOB0	Description			
0	0	0	0	TGR1B	Output disabled (initial value)		
			1	is an	Initial	Output 0 on compare-match	
			1	output	output	Output 1 on compare-match	
	1	0	0	0	compare	Toggle output on compare-match	
				1	register		
				1		Output disabled	
1	0	0	1		Initial	Output 0 on compare-match	
			1		output	Output 1 on compare-match	
			1		is 1	Toggle output on compare-match	
1	0	0	0	TGR1B	Capture	Input capture on rising edge	
			1	is an	input source	Input capture on falling edge	
			1	input	is the	Input capture on both edges	
	1	0	0	0	capture	TIOC1B pin	
				1	register		
				1		Capture input	Input capture
1	0	0	1		source TGR0C	on channel TGR0C	
			1		compare/match	compare-match/input	
			1		input capture	capture generation	

- Bits 3–0—I/O Control A3–A0 (IOA3–IOA0): These bits set the TGR1A register function.

Bit 3: IOA3	Bit 2: IOA2	Bit 1: IOA1	Bit 0: IOA0	Description			
0	0	0	0	TGR1A	Output disabled (initial value)		
			1	is an	Initial	Output 0 on compare-match	
		1	0	output	output	Output 1 on compare-match	
			1	compare register	is 0	Toggle output on compare-match	
	1	0	0		Output disabled		
			1		Initial	Output 0 on compare-match	
		1	0		output	Output 1 on compare-match	
			1		is 1	Toggle output on compare-match	
1	0	0	0	TGR1A	Capture	Input capture on rising edge	
			1	is an	input source	Input capture on falling edge	
		1	0	input	is the	TIOC1A pin	Input capture on both edges
			1	capture register			
	1	0	0		Capture input	Input capture	
			1		source is TGR0A	on channel 0/TGR0A	
		1	0		compare-match/input	capture	compare-match/input capture generation
			1				

## Channel 2 (TIOR2 Register):

- Bits 7–4—I/O Control B3–B0 (IOB3–IOB0): These bits set the TGR2B register function.

Bit 7: IOB3	Bit 6: IOB2	Bit 5: IOB1	Bit 4: IOB0	Description			
0	0	0	0	TGR2B	Output disabled (initial value)		
			1	is an	Initial	Output 0 on compare-match	
		1	0	output	output	Output 1 on compare-match	
			1	compare	is 0	Toggle output on compare-match	
	1	0	0	register	Output disabled		
			1		Initial	Output 0 on compare-match	
		1	0		output	Output 1 on compare-match	
			1		is 1	Toggle output on compare-match	
1	0	0	0	TGR2B	Capture	Input capture on rising edge	
			1	is an		Input capture on falling edge	
		1	0	input		is the	Input capture on both edges
			1	capture		TIOC2B pin	
	1	0	0	register	Input capture on rising edge		
			1		Input capture on falling edge		
		1	0		Input capture on both edges		
			1				

- Bits 3–0—I/O Control A3–A0 (IOA3–IOA0): These bits set the TGR2A register function.

Bit 3: IOA3	Bit 2: IOA2	Bit 1: IOA1	Bit 0: IOA0	Description		
0	0	0	0	TGR2A is an output compare register	Output disabled (initial value)	
			1		Initial output is 0	Output 0 on compare-match
		1	0		Output 1 on compare-match	
			1		Toggle output on compare-match	
	1	0	0	Output disabled		
			1	Initial output is 1	Output 0 on compare-match	
		1	0	Output 1 on compare-match		
			1	Toggle output on compare-match		
1	0	0	0	TGR2A is an input capture register	Capture input source is the TIOC2A pin	Input capture on rising edge
			1		Input capture on falling edge	
		1	0		Input capture on both edges	
			1			
	1	0	0	Input capture on rising edge		
			1	Input capture on falling edge		
		1	0	Input capture on both edges		
			1			

## 8.2.4 Timer Interrupt Enable Register (TIER)

The TIER is an 8-bit register that controls the enable/disable of interrupt requests for each channel. The MTU has three TIER registers, one each for channel. TIER is initialized to H'40 by a power-on reset.

### Channel 0: TIER0

Bit:	7	6	5	4	3	2	1	0
	TTGE	—	—	TCIEV	TGIED	TGIEC	TGIEB	TGIEA
Initial value:	0	1	0	0	0	0	0	0
R/W:	R/W	R	R	R/W	R/W	R/W	R/W	R/W

### Channels 1, 2: TIER1, TIER2

Bit:	7	6	5	4	3	2	1	0
	TTGE	—	—	TCIEV	—	—	TGIEB	TGIEA
Initial value:	0	1	0	0	0	0	0	0
R/W:	R/W	R	R	R/W	R	R	R/W	R/W

- Bit 7—A/D Conversion Start Request Enable (TTGE): Enables or disables generation of an A/D conversion start request by a TGRA register input capture/compare-match.

Bit 7: TTGE	Description
0	Disable A/D conversion start requests generation (initial value)
1	Enable A/D conversion start request generation

- Bit 6—Reserved: This bit is reserved. It always reads as 1, and cannot be modified.
- Bit 5—Reserved. This bit always reads 0. The write value should always be 0.
- Bit 4—Overflow Interrupt Enable (TCIEV): Enables or disables interrupt requests when the overflow flag TCFV of the timer status register (TSR) is set to 1.

Bit 4: TCIEV	Description
0	Disable TCFV interrupt requests (TCIV) (initial value)
1	Enable TCFV interrupt requests (TCIV)

- Bit 3—TGR Interrupt Enable D (TGIED): Enables or disables interrupt TGFD requests when the TGFD bit of the channel 0 of the TSR register is set to 0.

This bit is reserved for channels 1 and 2. It always reads as 0. The write value should always be 0.

Bit 3: TGIED	Description
0	Disable interrupt requests (TGID) due to the TGFD bit (initial value)
1	Enable interrupt requests (TGID) due to the TGFD bit

- Bit 2—TGR Interrupt Enable C (TGIEC): Enables or disables TGFC interrupt requests when the TGFC bit of the Channel 0 of the TSR register is set to 1.

This bit is reserved for channels 1 and 2. It always reads as 0. The write value should always be 0.

Bit 2: TGIEC	Description
0	Disable interrupt requests (TGIC) due to the TGFC bit (initial value)
1	Enable interrupt requests (TGIC) due to the TGFC bit

- Bit 1—TGR Interrupt Enable B (TGIEB): Enables or disables TGFB interrupt requests when the TGFB bit of the TSR register is set to 1.

Bit 1: TGIEB	Description
0	Disable interrupt requests (TGIB) due to the TGFB bit (initial value)
1	Enable interrupt requests (TGIB) due to the TGFB bit

- Bit 0—TGR Interrupt Enable A (TGIEA): Enables or disables TGFA interrupt requests when the TGFA bit of the TSR register is set to 1.

Bit 0: TGIEA	Description
0	Disable interrupt requests (TGIA) due to the TGFA bit (initial value)
1	Enable interrupt requests (TGIA) due to the TGFA bit

## 8.2.5 Timer Status Register (TSR)

The timer status register (TSR) is an 8-bit register that indicates the status of each channel. The MTU has three TSR registers, one each for channel. TSR is initialized to H'00 by a power-on reset.

### Channel 0: TSR0

Bit:	7	6	5	4	3	2	1	0
	—	—	—	TCFV	TGFD	TGFC	TGFB	TGFA
Initial value:	1	1	0	0	0	0	0	0
R/W:	R	R	R	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*

Note: Only 0 writes to clear the flags are possible.

### Channels 1, 2: TSR1, TSR2

Bit:	7	6	5	4	3	2	1	0
	—	—	—	TCFV	—	—	TGFB	TGFA
Initial value:	1	1	0	0	0	0	0	0
R/W:	R	R	R	R/(W)*	R	R	R/(W)*	R/(W)*

Note: Only 0 writes to clear the flags are possible.

- Bits 7 and 6—Reserved. These bits always read 1. The write value should always be 1.
- Bit 5—Reserved. This bit always reads 0. The write value should always be 0.
- Bit 4—Overflow Flag (TCFV): This status flag indicates the occurrence of a TCNT counter overflow.

Bit 4: TCFV	Description
0	Clear condition: With TCFV =1, a 0 write to TCFV after reading it (initial value)
1	Set condition: When the TCNT value overflows (H'FFFF → H'0000)



- Bit 3—Output Compare Flag D (TGFD): This status flag indicates the occurrence of a channel 0 TGRD register compare-match.

This bit is reserved in channels 1 and 2: it always reads 0 and the write value should always be 0.

Bit 3: TGFD	Description
0	Clear condition: With TGFD = 1, a 0 write to TGFD following a read (initial value)
1	Set condition: When TCNT = TGRD while TGRD is functioning as an output compare register

- Bit 2—Input Capture/Output Compare Flag C (TGFC): This status flag indicates the occurrence of a Channel 0 TGRC register input capture or compare-match.

This bit is reserved for channels 1 and 2. It always reads as 0. The write value should always be 0.

Bit 2: TGFC	Description
0	Clear condition: With TGFC = 1, a 0 write to TGFC following a read (initial value)
1	Set conditions: <ul style="list-style-type: none"> <li>• When TGRC is functioning as an output compare register (TCNT = TGRC)</li> <li>• When TGRC is functioning as input capture (the TCNT value is sent to TGRC by the input capture signal)</li> </ul>

- Bit 1—Output Compare Flag B (TGFB): This status flag indicates the occurrence of a TGRB register compare-match.

Bit 1: TGFB	Description
0	Clear condition: With TGFB = 1, a 0 write to TGFB following a read (initial value)
1	Set conditions: When TGRB is functioning as an output compare register (TCNT = TGRB)

- Bit 0—Input Capture/Output Compare Flag A (TGFA): This status flag indicates the occurrence of a TGRA register input capture or compare-match.

Bit 0: TGFA	Description
0	Clear condition: With TGFA = 1, a 0 write to TGFA following a read (initial value)
1	Set conditions: <ul style="list-style-type: none"> <li>• When TGRA is functioning as an output compare register (TCNT = TGRA)</li> <li>• When TGRA is functioning as input capture (the TCNT value is sent to TGRA by the input capture signal)</li> </ul>

## 8.2.6 Timer Counters (TCNT)

The timer counters (TCNT) are 16-bit counters, with one for each channel, for a total of three.

The TCNT are initialized to H'0000 by a power-on reset. Accessing the TCNT counters in 8-bit units is prohibited. Always access in 16-bit units.

### Channel 0: TCNT0

### Channel 1: TCNT1

### Channel 2: TCNT2

Bit:	15	14	13	12	11	10	9	8
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit:	7	6	5	4	3	2	1	0
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

## 8.2.7 Timer General Register (TGR)

Each timer general register (TGR) is a 16-bit register that can function as either an output compare register or an input capture register. There are a total of eight TGR, four each for channels 0 and two each for channels 1 and 2. The TGRC and TGRD of channels 0 can be set to operate as buffer registers. The TGR register and buffer register combinations are TGRA with TGRC, and TGRB with TGRD.

The TGRs are initialized to H'FFFF by a power-on reset. Accessing of the TGRs in 8-bit units is disabled; they may only be accessed in 16-bit units.

Bit:	15	14	13	12	11	10	9	8
Initial value:	1	1	1	1	1	1	1	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit:	7	6	5	4	3	2	1	0
Initial value:	1	1	1	1	1	1	1	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

## 8.2.8 Timer Start Register (TSTR)

The timer start register (TSTR) is an 8-bit read/write register that starts and stops the timer counters (TCNT) of channels 0–2. TSTR is initialized to H'00 upon power-on reset.

Bit:	7	6	5	4	3	2	1	0
	—	—	—	—	—	CST2	CST1	CST0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R/W	R/W	R/W

- Bits 7–3—Reserved. These bits always read 0. The write value should always be 0.
- Bits 2–0—Counter Start 2-0 (CST2-CST0): Select starting and stopping of the timer counters (TCNT). The corresponding between bits and channels is as follows:

CST2: Channel 2 (TCNT2)

CST1: Channel 1 (TCNT1)

CST0: Channel 0 (TCNT0)

Bit n: CSTn	Description
0	TCNTn count is halted (initial value)
1	TCNTn counts

Note: n = 2 to 0.

If 0 is written to a CST bit during operation with the TIOC pin in the output state, the counter stops, but the TIOC pin output compare output level is maintained. If a write is performed on the TIOR register while a CST bit is 0, the pin output level is updated to the set initial output value.

## 8.2.9 Timer Synchro Register (TSYR)

The timer synchro register (TSYR) is an 8-bit read/write register that selects independent or synchronous TCNT counter operation for channels 0–2. Channels for which 1 is set in the corresponding bit will be synchronized. TSYR is initialized to H'00 upon power-on reset.

Bit:	7	6	5	4	3	2	1	0
	—	—	—	—	—	SYNC2	SYNC1	SYNC0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R/W	R/W	R/W

- Bits 7–3—Reserved. These bits always read 0. The write value should always be 0.
- Bits 2–0—Timer Synchronization 2–0 (SYNC2–SYNC0): Selects operation independent of, or synchronized to, other channels. Synchronous operation allows synchronous clears due to multiple TCNT synchronous presets and other channel counter clears. A minimum of two channels must have SYNC bits set to 1 for synchronous operation. For synchronization clearing, it is necessary to set the TCNT counter clear sources (the CCLR2–CCLR0 bits of the TCR register), in addition to the SYNC bit. The counter start to channel and bit-to-channel correspondence are indicated in the tables below.

SYNC2: Channel 2 (TCNT2)

SYNC1: Channel 1 (TCNT1)

SYNC0: Channel 0 (TCNT0)

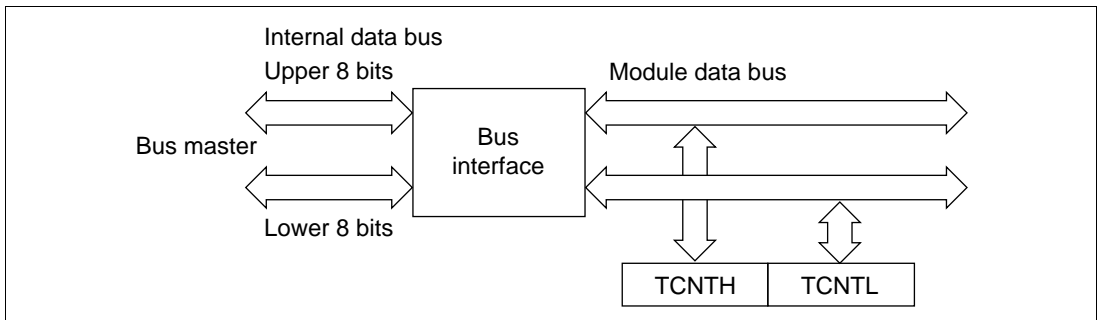
Bit n: SYNCn	Description
0	Timer counter (TCNTn) independent operation (initial value) (TCNTn preset/clear unrelated to other channels)
1	Timer counter synchronous operation* <sup>1</sup> TCNTn synchronous preset/ synchronous clear* <sup>2</sup> possible

- Note:
1. Minimum of two channel SYNC bits must be set to 1 for synchronous operation.
  2. TCNT counter clear sources (CCLR2–CCLR0 bits of the TCR register) must be set in addition to the SYNC bit in order to have clear synchronization.  
n = 2 to 0.

## 8.3 Bus Master Interface

### 8.3.1 16-Bit Registers

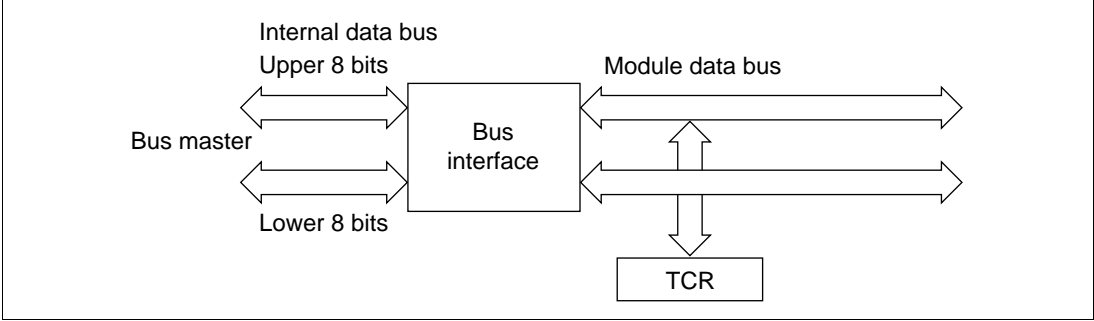
The timer counters (TCNT) and general registers (TGR) are 16-bit registers. A 16-bit data bus to the bus master enables 16-bit read/writes. 8-bit read/write is not possible. Always access in 16-bit units. Figure 8.3 shows an example of 16-bit register access operation.



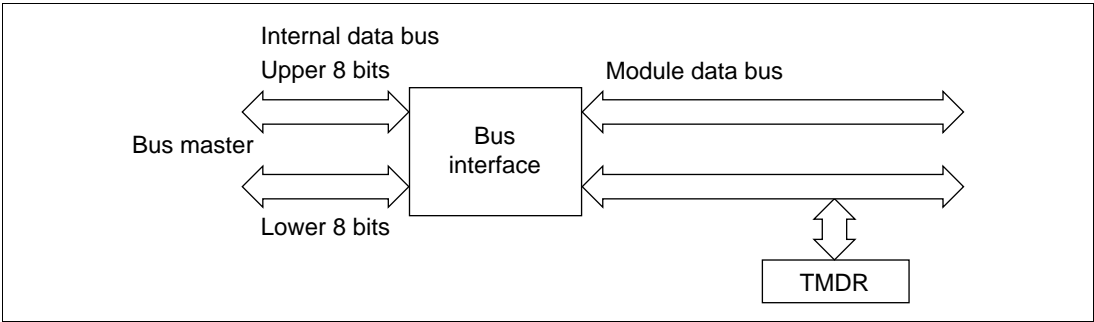
**Figure 8.2 16-Bit Register Access Operation (Bus Master ↔ TCNT (16 Bit))**

### 8.3.2 8-Bit Registers

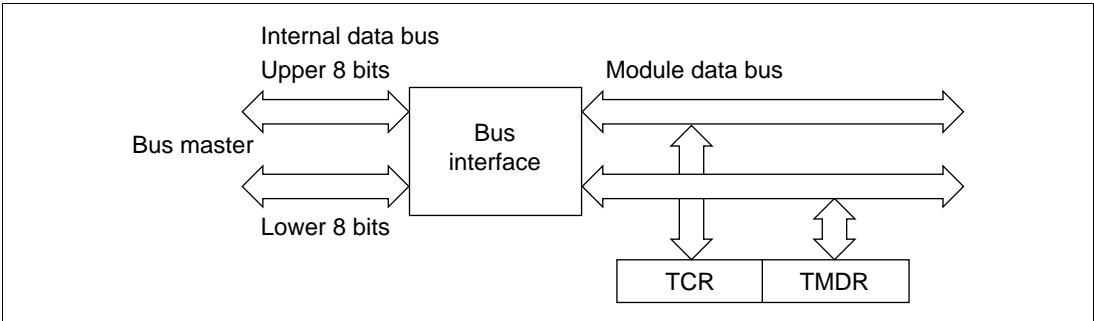
All registers other than the TCNT and general registers (TGR) are 8-bit registers. These are connected to the CPU by a 16-bit data bus, so 16-bit read/writes and as 8-bit read/writes are both possible (figures 8.3, 8.4, 8.5).



**Figure 8.3 8-Bit Register Access Operation (Bus Master ↔ TCR (Upper 8 Bits))**



**Figure 8.4 8-Bit Register Access Operation (Bus Master ↔ TMDR (Lower 8 Bits))**



**Figure 8.5 8-Bit Register Access Operation (Bus Master ↔ TCR, TMDR (16 Bit))**

## 8.4 Operation

### 8.4.1 Overview

The operation modes are described below.

**Ordinary Operation:** Each channel has a timer counter (TCNT) and general register (TGR). The TCNT is an upcounter and can also operate as a free-running counter, periodic counter or external event counter. General registers (TGR) can be used as output compare registers or input capture registers.

**Synchronized Operation:** The TCNT of a channel set for synchronized operation does a synchronized preset. When any TCNT of a channel operating in the synchronized mode is rewritten, the TCNTs of other channels are simultaneously rewritten as well. The timer synchronization bits of the TSYR registers of multiple channels set for synchronous operation can be set to clear the TCNTs simultaneously.

**Buffer Operation:** When TGR is an output compare register, the buffer register value of the corresponding channel is transferred to the TGR when a compare-match occurs. When TGR is an input capture register, the TCNT counter value is transferred to the TGR when an input capture occur simultaneously the value previously stored in the TGR is transferred to the buffer register.

**Cascade Connection Operation:** The channel 1 and channel 2 counters (TCNT1 and TCNT2) can be connected together to operate as a 32-bit counter.

**PWM Mode:** In PWM mode, a PWM waveform is output. The output level can be set by the TIOR register. Each TGR can be set for PWM waveform output with a duty cycle between 0% and 100%.

### 8.4.2 Basic Functions

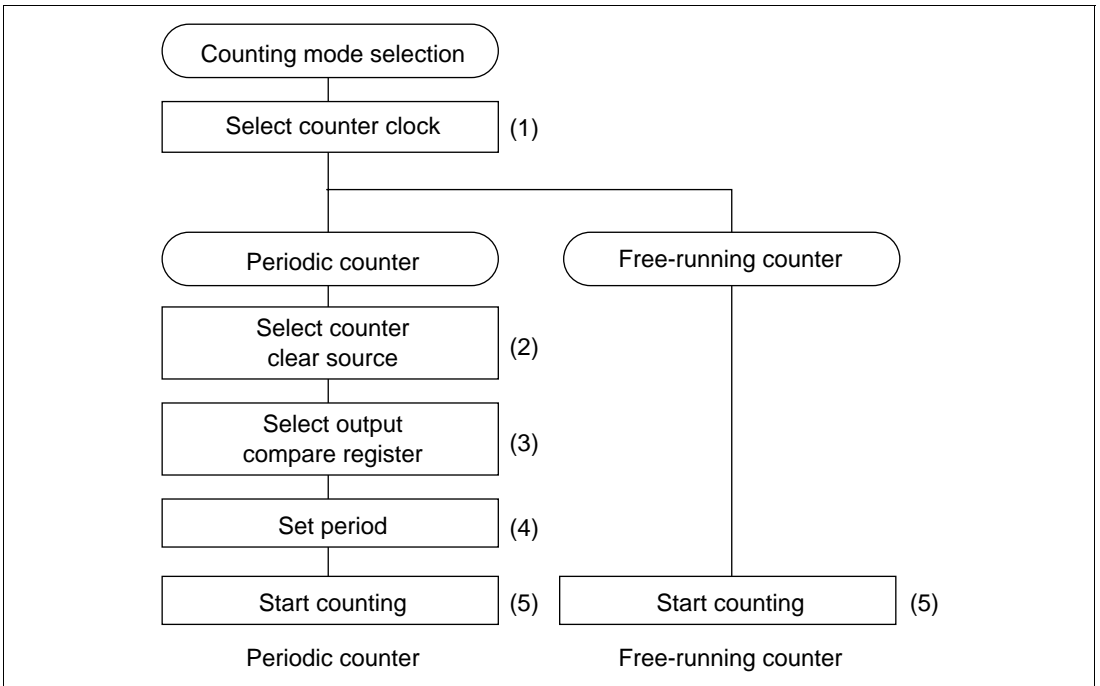
Always select MTU external pin set function using the pin function controller (PFC).

**Counter Operation:** When a start bit (CST0–CST2) in the timer start register (TSTR) is set to 1, the corresponding timer counter (TCNT) starts counting. There are two counting modes: a free-running mode and a periodic mode.

To select the counting operation (figure 8.6):

1. Set bits TPSC2–TPSC0 in the TCR to select the counter clock. At the same time, set bits CKEG1 and CKEG0 in the TCR to select the desired edge of the input clock.
2. To operate as a periodic counter, set the CCLR2–CCLR0 bits in the TCR to select TGR as a clearing source for the TCNT.

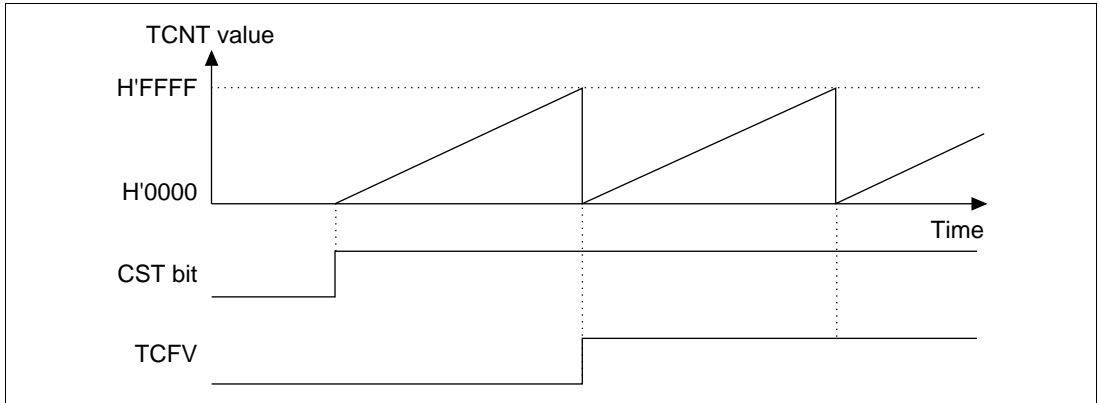
3. Set the TGR selected in step 2 as an output compare register using the timer I/O control register (TIOR).
4. Write the desired cycle value in the TGR selected in step 2.
5. Set the CST bit in the TSTR to 1 to start counting.



**Figure 8.6 Procedure for Selecting the Counting Operation**

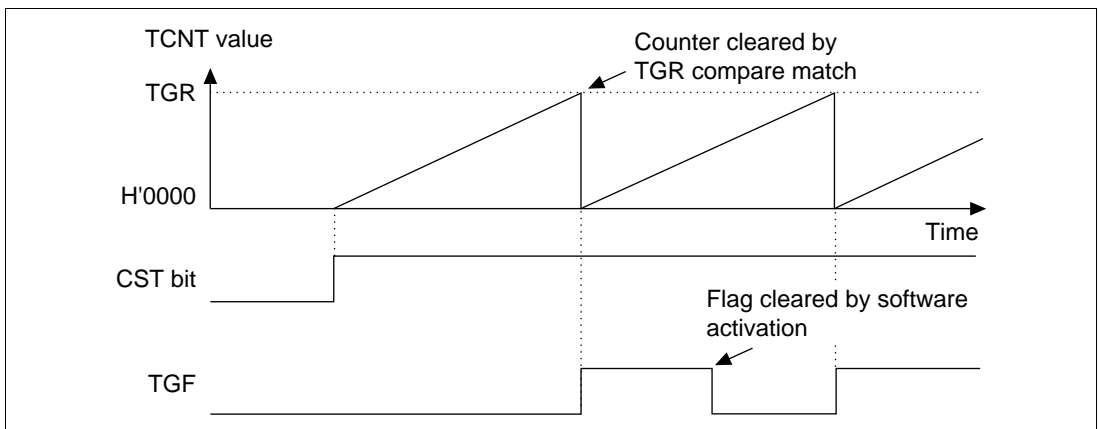
**Free-Running Counter Operation Example:** A reset of the MTU timer counters (TCNT) leaves them all in the free-running mode. When a bit in the TSTR is set to 1, the corresponding timer counter operates as a free-running counter and begins to increment. When the count overflows from H'FFFF–H'0000, the TCFV bit in the timer status register (TSR) is set to 1. If the TCIEV bit in the timer's corresponding timer interrupt enable register (TIER) is set to 1, the MTU will make an interrupt request to the interrupt controller. After the TCNT overflows, counting continues from H'0000. Figure 8.7 shows an example of free-running counter operation.





**Figure 8.7 Free-Running Counter Operation**

**Periodic Counter Operation Example:** Periodic counter operation is obtained for a given channel's TCNT by selecting compare-match as a TCNT clear source. Set the TGR register for period setting to output compare register and select counter clear upon compare-match using the CCLR2–CCLR0 bits of the timer control register (TCR). After these settings, the TCNT begins incrementing as a periodic counter when the corresponding bit of TSTR is set to 1. When the count matches the TGR register value, the TGF bit in the TSR is set to 1 and the counter is cleared to H'0000. If the TGIE bit of the corresponding TIER is set to 1 at this point, the MTU will make an interrupt request to the interrupt controller. After the compare-match, TCNT continues counting from H'0000. Figure 8.8 shows an example of periodic counting.

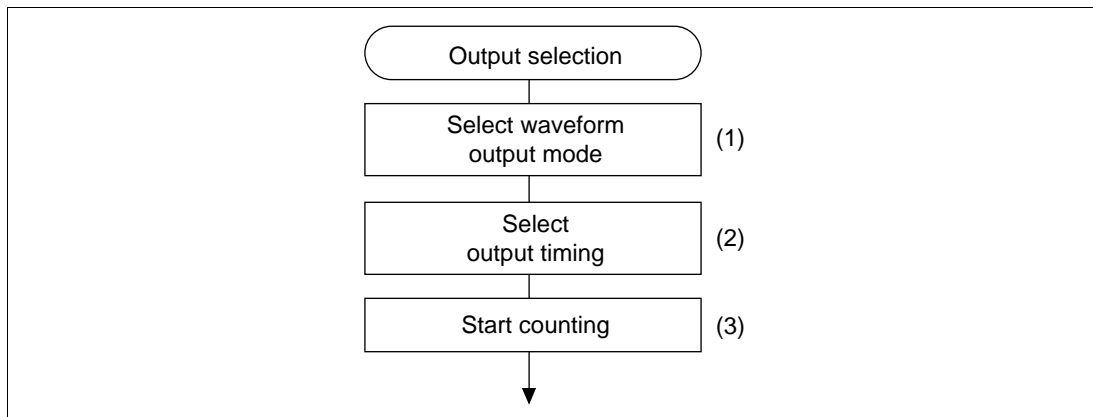


**Figure 8.8 Periodic Counter Operation**

**Compare-Match Waveform Output Function:** The MTU can output 0 level, 1 level, or toggle output from the corresponding output pins upon compare-matches.

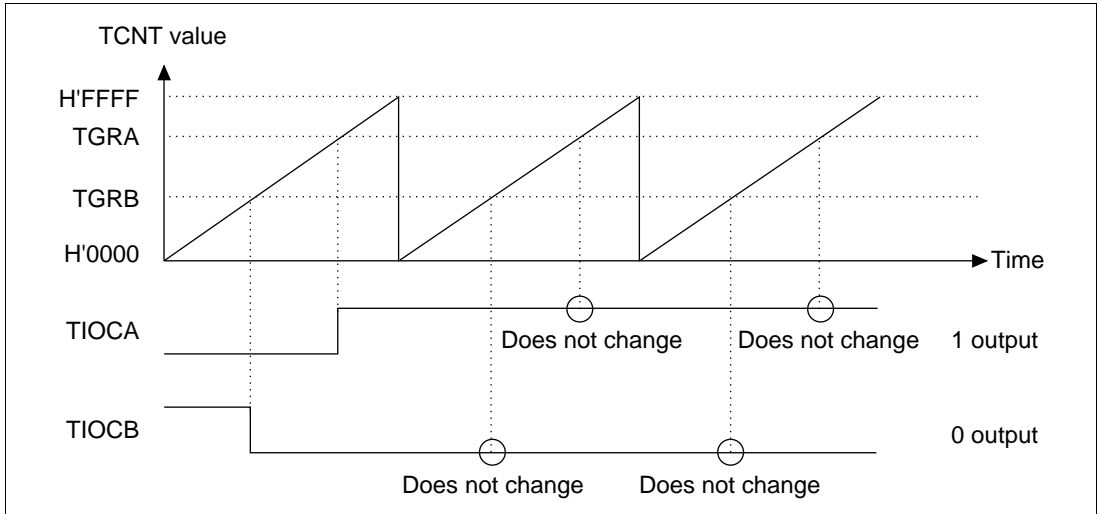
Procedure for selecting the compare-match waveform output operation (figure 8.9):

1. Set the TIOR to select 0 output or 1 output for the initial value, and 0 output, 1 output, or toggle output for compare-match output. The TIOC pin will output the set initial value until the first compare-match occurs.
2. Set a value in the TGR to select the compare-match timing.
3. Set the CST bit in the TSTR to 1 to start counting.



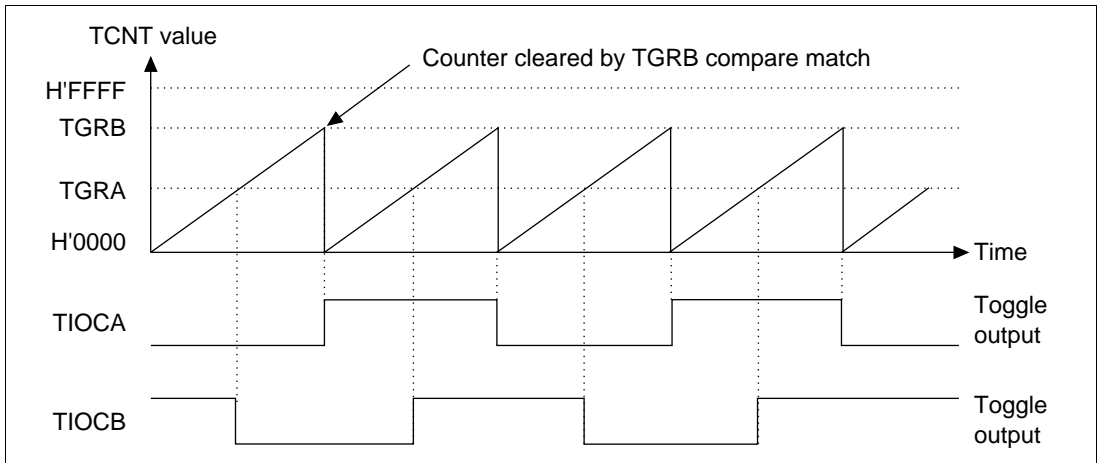
**Figure 8.9 Procedure for Selecting Compare Match Waveform Output Operation**

**Waveform Output Operation (0 Output/1 Output):** Figure 8.10 shows 0 output/1 output. In the example, TCNT is a free-running counter, 1 is output upon compare-match A and 0 is output upon compare-match B. When the pin level matches the set level, the pin level does not change.



**Figure 8.10 Example of 0 Output/1 Output**

**Waveform Output Operation (Toggle Output):** Figure 8.11 shows the toggle output. In the example, the TCNT operates as a periodic counter cleared by compare-match B, with toggle output at both compare-match A and compare-match B.



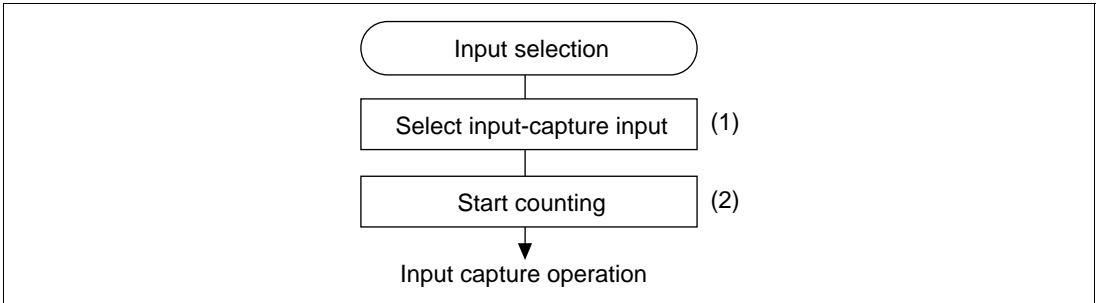
**Figure 8.11 Example of Toggle Output**

**Input Capture Function:** In the input capture mode, the TCNT value is transferred into the TGR register when the input edge is detected at the input capture/output compare pin (TIOC).

Detection can take place on the rising edge, falling edge, or both edges. Channels 0 and 1 can use other channel counter input clocks or compare-match signals as input capture sources.

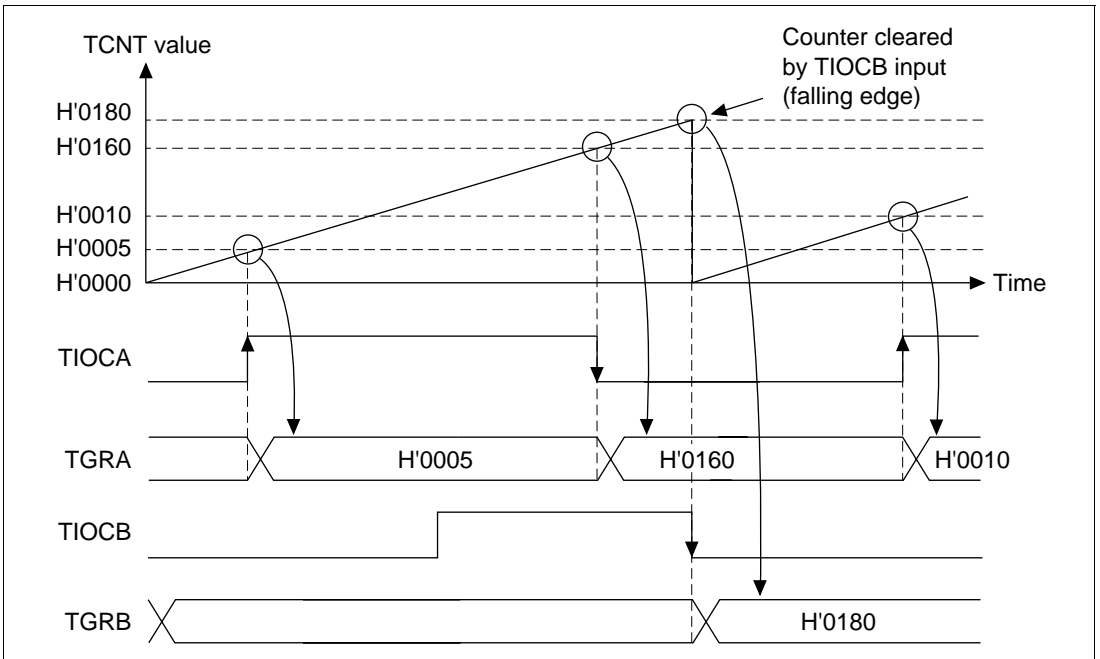
The procedure for selecting the input capture operation (figure 8.12) is:

1. Set the TIOR to select the input capture function of the TGR, then select the input capture source, and rising edge, falling edge, or both edges as the input edge.
2. Set the CST bit in the TSTR to 1 to start the TCNT counting.



**Figure 8.12 Procedure for Selecting Input Capture Operation**

**Input Capture Operation:** Figure 8.13 shows input capture. The falling edge of TIOCB and both edges of TIOCA are selected as input capture input edges. In the example, TCNT is set to clear at the input capture of the TGRB register.



**Figure 8.13 Input Capture Operation**

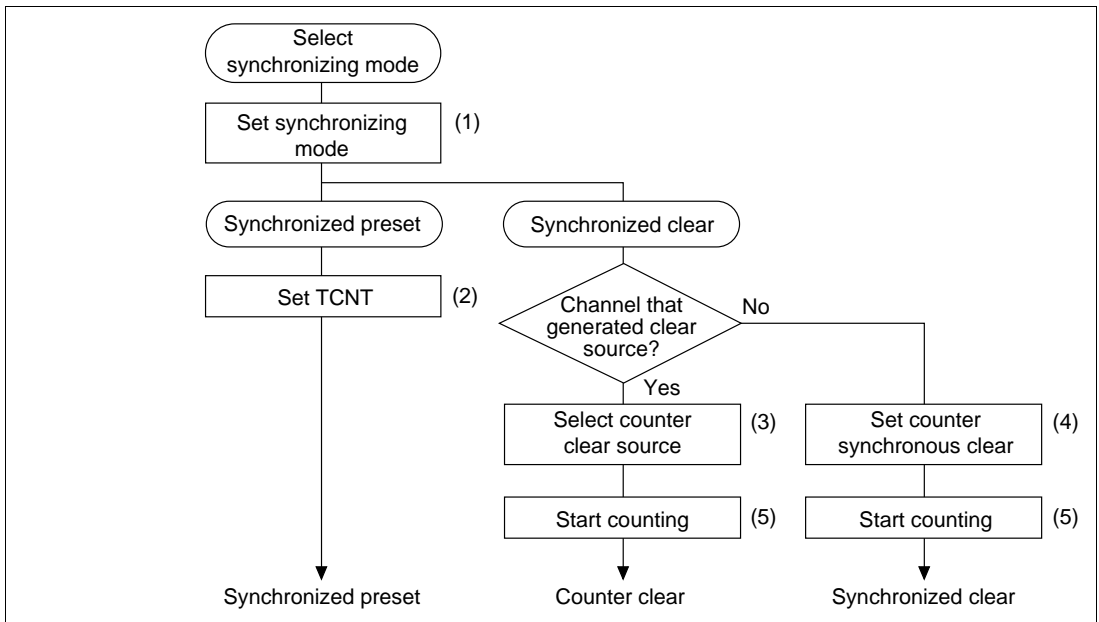
### 8.4.3 Synchronous Operation

There are two kinds of synchronous operation, synchronized preset and synchronized clear. The synchronized preset operation allows multiple timer counters (TCNT) to be rewritten simultaneously, while the synchronized clear operation allows multiple TCNT counters to be cleared simultaneously using timer control register (TCR) settings.

The synchronizing mode can increase the number of TGR registers for a single time base. All five channels can be set for synchronous operation.

#### Procedure for Selecting the Synchronizing Mode (Figure 8.14):

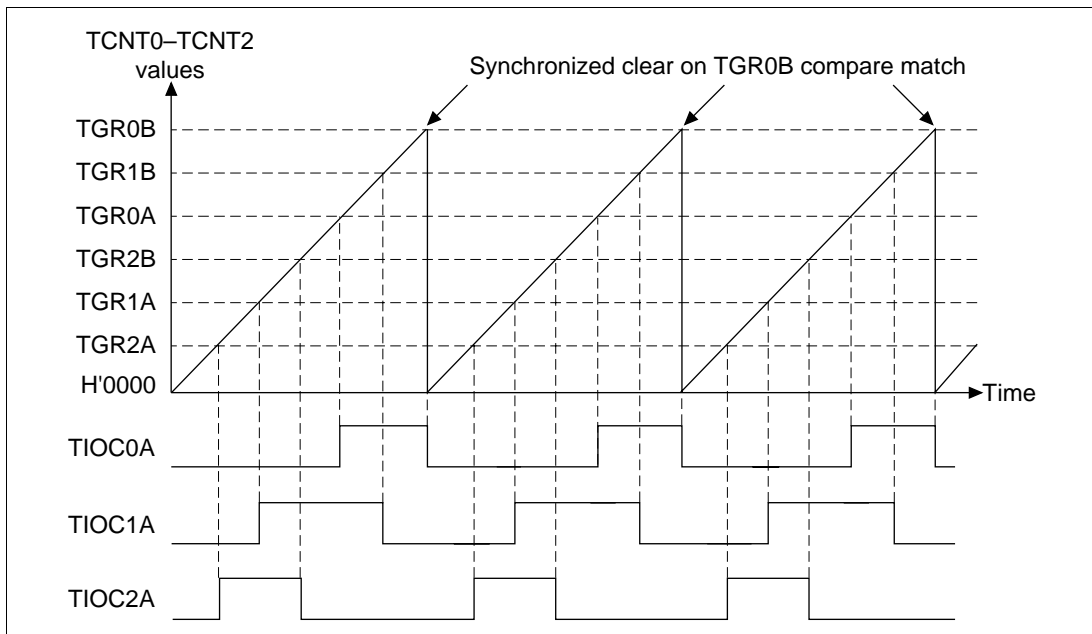
1. Set 1 in the SYNC bit of the timer synchro register (TSYR) to use the corresponding channel in the synchronizing mode.
2. When a value is written in the TCNT in any of the synchronized channels, the same value is simultaneously written in the TCNT in the other channels.
3. Set the counter to clear with output compare/input capture using bits CCLR2–CCLR0 in the TCR.
4. Set the counter clear source to synchronized clear using the CCLR2–CCLR0 bits of the TCR.
5. Set the CST bits for the corresponding channels in the TSTR to 1 to start counting in the TCNT.



**Figure 8.14 Procedure for Selecting Synchronizing Operation**

**Synchronized Operation:** Figure 8.15 shows an example of synchronized operation. Channels 0, 1, and 2 are set to synchronized operation and PWM mode 1. Channel 0 is set for a counter clear upon compare-match with TGR0B. Channels 1 and 2 are set for synchronous counter clears by synchronous presets and TGR0B register compare-matches. Accordingly, a three-phase PWM waveform with the data set in the TGR0B register as its PWM period is output from the TIOC0A, TIOC1A, and TIOC2A pins.

See section 8.4.6, PWM Mode, for details on the PWM mode.



**Figure 8.15 Synchronized Operation Example**

## 8.4.4 Buffer Operation

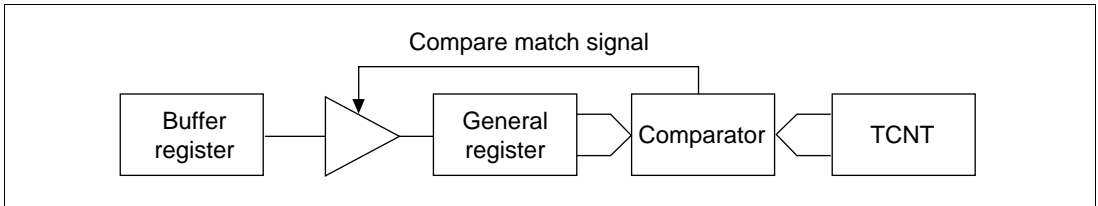
Buffer operation is a function of channel 0. TGRC and TGRD can be used as buffer registers. Table 8.5 shows the register combinations for buffer operation.

**Table 8.5 Register Combinations**

Channel	General Register	Buffer Register
0	TGR0A	TGR0C
	TGR0B	TGR0D

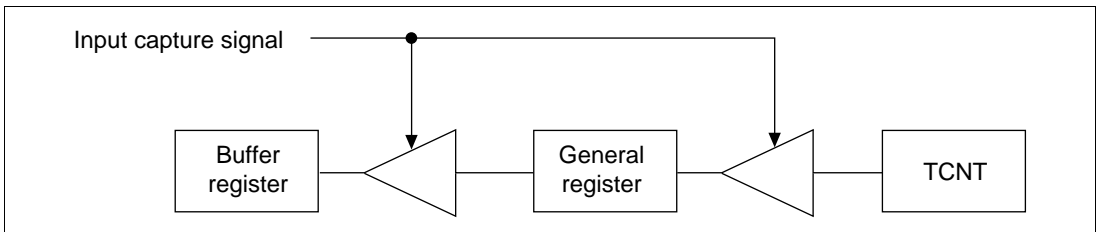
The buffer operation differs, depending on whether the TGR has been set as an input capture register or an output compare register.

**When TGR Is an Output Compare Register:** When a compare-match occurs, the corresponding channel buffer register value is transferred to the general register. Figure 8.16 shows an example.



**Figure 8.16 Compare Match Buffer Operation**

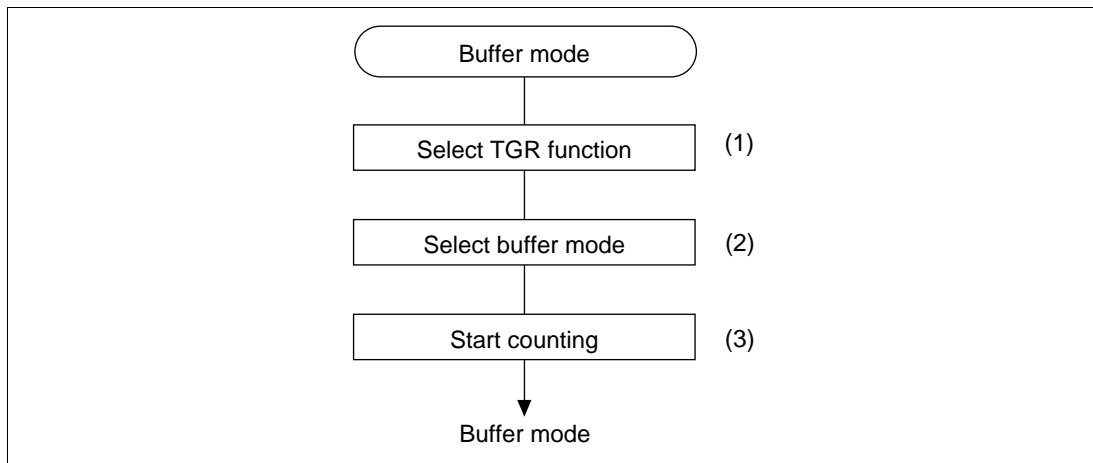
**When TGR Is an Input Capture Register:** When an input capture occurs, the timer counter (TCNT) value is transferred to the general register (TGR), and the value that had been held up to that time in the TGR is transferred to the buffer register (figure 8.17).



**Figure 8.17 Input Capture Buffer Operation**

### Procedure for Setting Buffer Mode (Figure 8.18):

1. Use the timer I/O control register (TIOR) to set the TGR as either an input capture or output compare register.
2. Use the timer mode register (TMDR) BFA, and BFB bits to set the TGR for buffer mode.
3. Set the CST bit in the TSTR to 1 to start the count operation.



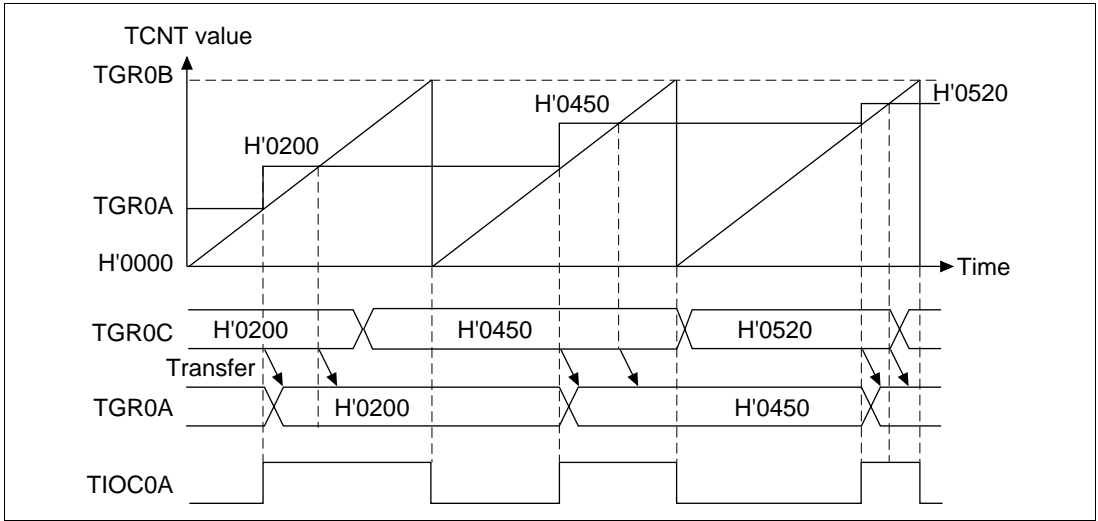
**Figure 8.18 Buffer Operation Setting Procedure**

**Buffer Operation Examples—when TGR Is an Output Compare Register:** Figure 8.19 shows an example of channel 0 set to PWM mode 1, and the TGRA and TGRC registers set for buffer operation.

The TCNT counter is cleared by a compare-match B, and the output is a 1 upon compare-match A and 0 output upon compare-match B. Because buffer mode is selected, a compare-match A changes the output, and the buffer register TGRC value is simultaneously transferred to the general register TGRA. This operation is repeated with each occurrence of a compare-match A.

See section 8.4.6, PWM Mode, for details on the PWM mode.

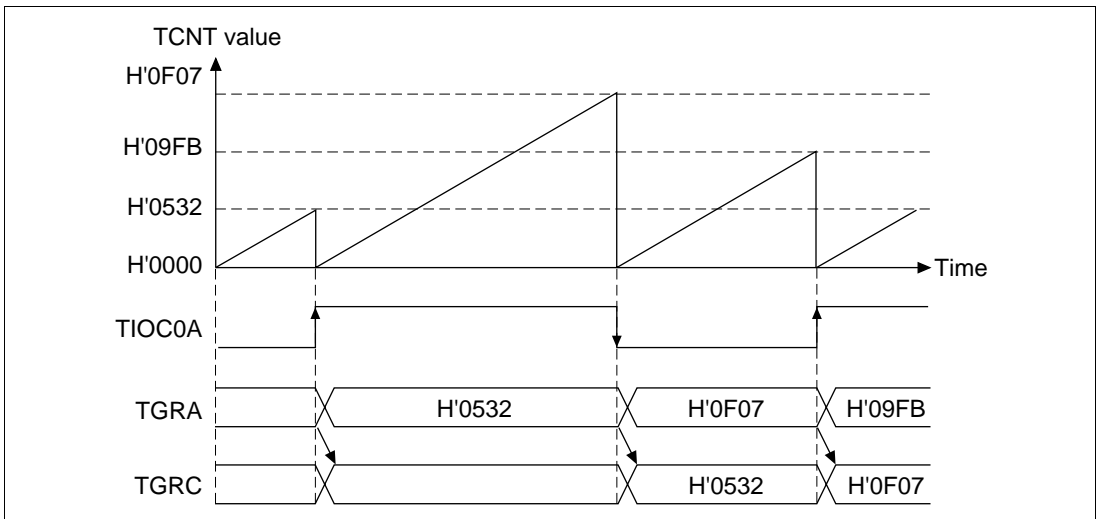




**Figure 8.19 Buffer Operation Example (Output Compare Register)**

**Buffer Operation Examples—when TGR Is an Input Capture Register:** Figure 8.20 shows an example of TGRA set as an input capture register with the TGRA and TGRB registers set for buffer operation.

The TCNT counter is cleared by a TGRA register input capture, and the TIOCA pin input capture input edge is selected as both rising and falling edge. Because buffer mode is selected, an input capture A causes the TCNT counter value to be stored in the TGRA register, and the value that was stored in the TGRA up until that time is simultaneously transferred to the TGRC register.



**Figure 8.20 Buffer Operation Example (Input Capture Register)**

## 8.4.5 Cascade Connection Mode

Cascade connection mode is a function that connects the 16-bit counters of two channels together to act as a 32-bit counter.

This function operates by using the TPSC2–TPSC0 bits of the TCR register to set the channel 1 counter clock to count by TCNT2 counter overflow.

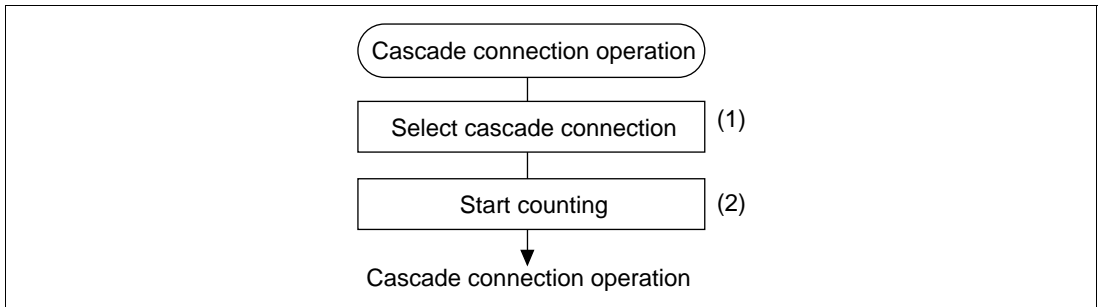
Table 8.6 shows the cascade connection combinations.

**Table 8.6 Cascade Connection Combinations**

Combination	Upper 16 Bits	Lower 16 Bits
Channel 1, channel 2	TCNT1	TCNT2

### Procedure for Setting Cascade Connection Mode (Figure 8.21):

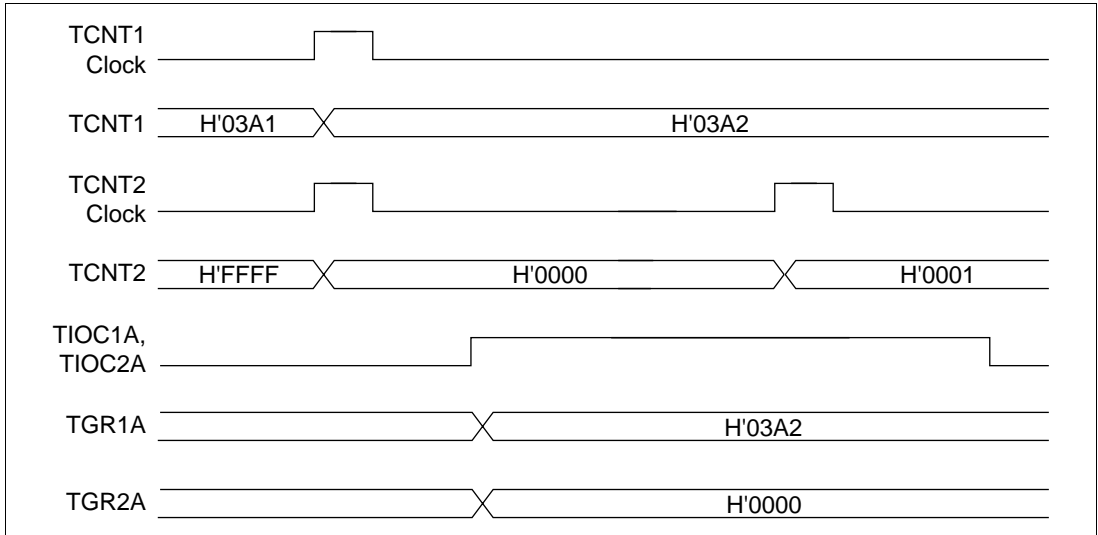
1. Set the TPSC2–TPSC 0 bits of the channel 1 timer control register (TCR) to B'111 to select “count by TCNT2 overflow.”
2. Set the CST bits corresponding to the upper and lower 16 bits in the TSTR to 1 to start the count operation.



**Figure 8.21 Procedure for Selecting Cascade Connection Mode**

**Cascade Connection Mode Examples—Input Capture:** Figure 8.22 shows an example of operation when the TCNT1 counter is set to count on TCNT2 overflow, the TGR1A and TGR2A registers are set as input capture registers, and the TIOC pin rising edge is selected.

Through simultaneous input of the rising edge to the TIOC1A and TIOC2A pins, 32-bit data is transferred, with the upper 16 bits to the TGR1A register and the lower 16 bits to the TGR2A register.



**Figure 8.22 Cascade Connection Operation Example (Input Capture)**

#### 8.4.6 PWM Mode

PWM mode outputs the various PWM waveforms from output pins. Output levels of 0 output, 1 output, or toggle output can be selected as the output level for the compare-match of each TGR.

A period can be set for a register by using the TGR compare-match as a counter clear source. All five channels can be independently set to PWM mode. Synchronous operation is also possible.

There are two PWM modes:

- **PWM mode 1**  
Generates PWM output using the TGRA and TGRB registers, and TGRC and TGRD registers as pairs. The initial output values are those established in the TGRA and TGRC registers. When the values set in TGR registers being used as a pair are equal, output values will not change even if a compare-match occurs.  
A maximum of 4-phase PWM output is possible for PWM mode 1.
- **PWM mode 2**  
Generates PWM output using one TGR register as a period register and another as a duty cycle register. The output value of each pin upon a counter clear is the initial value established by the TIOR register. When the values set in the period register and duty register are equal, output values will not change even if a compare-match occurs.

Table 8.7 lists the combinations of PWM output pins and registers.

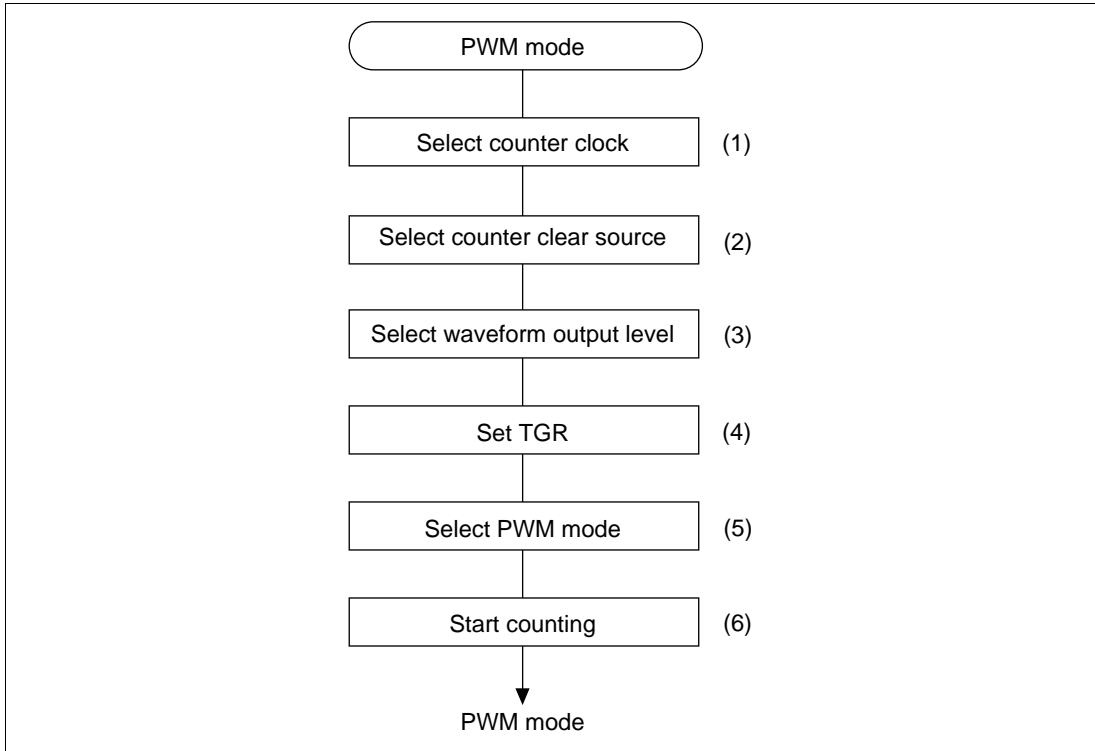
**Table 8.7 Combinations of PWM Output Pins and Registers**

Channel	Register	Output Pin	
		PWM Mode 1	PWM Mode 2
0 (AB pair)	TGR0A	TIOC0A	TIOC 0A
	TGR0B		TIOC 0B
0 (CD pair)	TGR0C	TIOC0C	TIOC 0C
	TGR0D		
1	TGR1A	TIOC1A	TIOC 1A
	TGR1B		TIOC 1B
2	TGR2A	TIOC2A	TIOC 2A
	TGR2B		TIOC 2B

Note: PWM output of the period setting TGR is not possible in PWM mode 2.

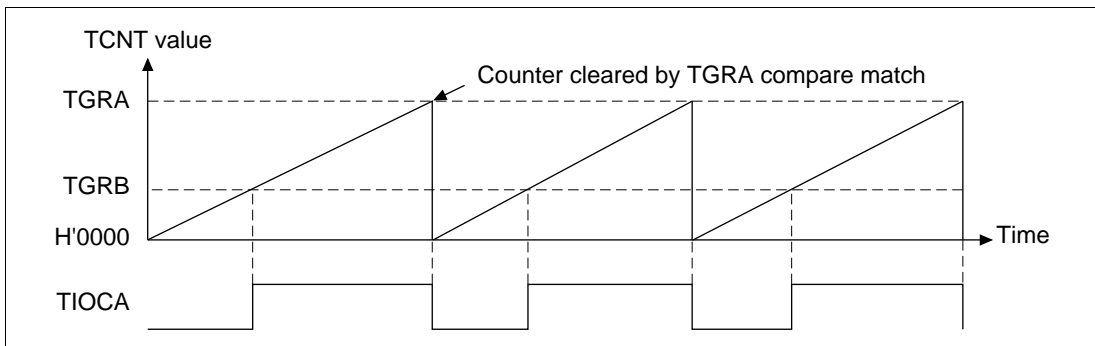
**Procedure for Selecting the PWM Mode (Figure 8.23):**

1. Set bits TPSC2–TPSC0 in the TCR to select the counter clock source. At the same time, set bits CKEG1 and CKEG0 in the TCR to select the desired edge of the input clock.
2. Set bits CCLR2 to CCLR0 in the TCR to select the TGR to be used as a counter clear source.
3. Set the period in the TGR selected in step 2, and the duty cycle in another TGR.
4. Using the timer I/O control register (TIOR), set the TGR selected in step 3 to act as an output compare register, and select the initial value and output value.
5. Set the MD3–MD 0 bits in TMDR to select the PWM mode.
6. Set the CST bit in the TSTR to 1 to let the TCNT start counting.



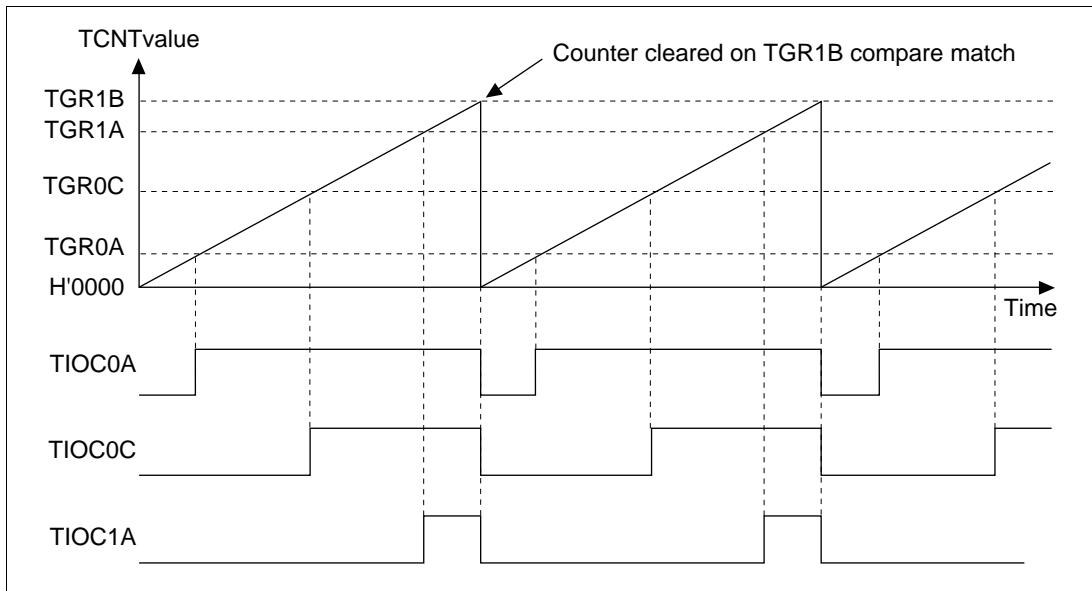
**Figure 8.23 Procedure for Selecting the PWM Mode**

**PWM Mode Operation Examples—PWM Mode 1 (Figure 8.24):** A TGRA register compare-match is used as a TCNT counter clear source, the TGRA register initial output value and output compare output value are both 0, and the TGRB register output compare output value is a 1. In this example, the value established in the TGRA register becomes the period and the value established in the TGRB register becomes the duty cycle.



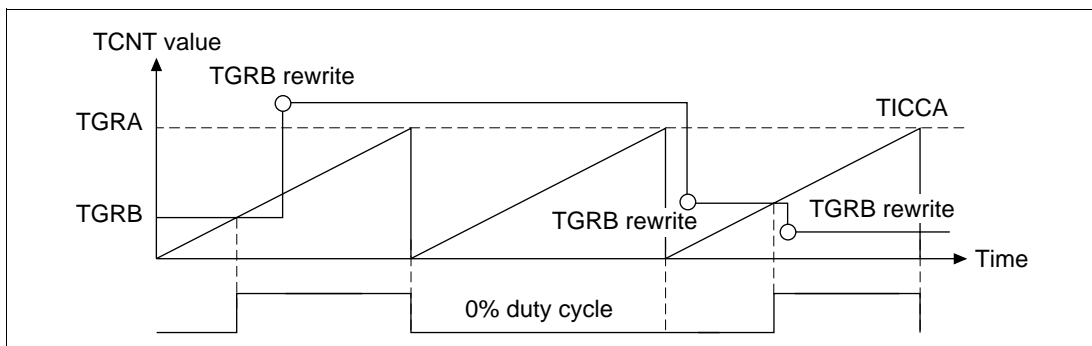
**Figure 8.24 PWM Mode Operation Example (Mode 1)**

**PWM Mode Operation Examples—PWM Mode 2 (Figure 8.25):** Channels 0 and 1 are set for synchronous operation, TGR1B register compare-match is used as a TCNT counter clear source, the other TGR register initial output value is 0 and output compare output value is 1, and a 3-phase PWM waveform is output. In this example, the value established in the TGR1B register becomes the period and the value established in the other TGR register becomes the duty cycle.



**Figure 8.25 PWM Mode Operation Example (Mode 2)**

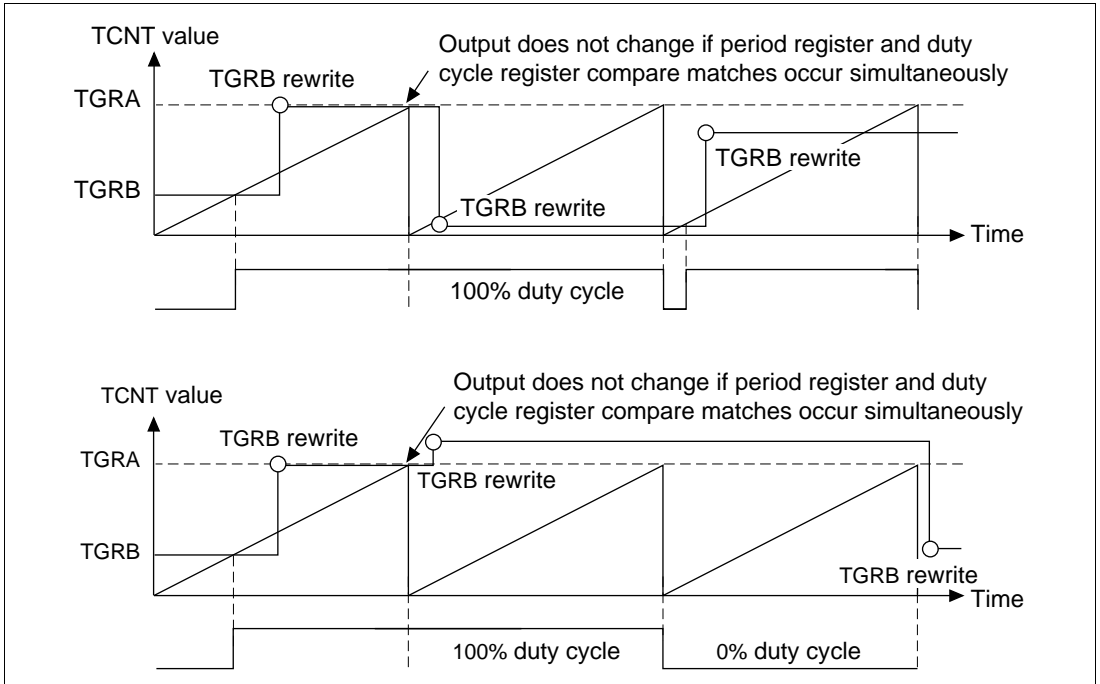
**0% Duty Cycle:** Figure 8.26 shows an example of a 0% duty cycle PWM waveform output in PWM mode.



**Figure 8.26 PWM Mode Operation Example (0% Duty Cycle)**

**100% Duty Cycle:** Figure 8.27 shows an example of a 100% duty cycle PWM waveform output in PWM mode.

In PWM mode, when setting cycle = duty cycle the output waveform does not change, nor is there a change of waveform for the first pulse immediately after clearing the counter.



**Figure 8.27 PWM Mode Operation Example (100% Duty Cycle)**

## 8.5 Interrupts

### 8.5.1 Interrupt Sources and Priority Ranking

The MTU has two interrupt sources: TGR register compare-match/input captures, TCNT counter overflows. Because each of these three types of interrupts are allocated its own dedicated status flag and enable/disable bit, the issuing of interrupt request signals to the interrupt controller can be independently enabled or disabled.

When an interrupt source is generated, the corresponding status flag in the timer status register (TSR) is set to 1. If the corresponding enable/disable bit in the timer input enable register (TIER) is set to 1 at this time, the MTU makes an interrupt request of the interrupt controller. The interrupt request is canceled by clearing the status flag to 0.

The channel priority order can be changed with the interrupt controller. The priority ranking within a channel is fixed. For more information, see section 6, Interrupt Controller.

Table 8.8 lists the MTU interrupt sources.

**Input Capture/Compare Match Interrupts:** If the TGIE bit of the timer input enable register (TIER) is already set to 1 when the TGF flag in the timer status register (TSR) is set to 1 by a TGR register input capture/compare-match of any channel, an interrupt request is sent to the interrupt controller. The interrupt request is canceled by clearing the TGF flag to 0. The MTU has 8 input capture/compare-match interrupts; four each for channel 0, and two each for channels 1 and 2.

**Overflow Interrupts:** If the TCIEV bit of the TIER is already set to 1 when the TCFV flag in the TSR is set to 1 by a TCNT counter overflow of any channel, an interrupt request is sent to the interrupt controller. The interrupt request is canceled by clearing the TCFV flag to 0. The MTU has three overflow interrupts, one for each channel.



**Table 8.8 MTU Interrupt Sources**

Channel	Interrupt Source	Description	Priority*
0	TGI0A	TGR0A input capture/compare-match	High
	TGI0B	TGR0B input capture/compare-match	
	TGI0C	TGR0C input capture/compare-match	
	TGI0D	TGR0D input capture/compare-match	
	TCI0V	TCNT0 overflow	
1	TGI1A	TGR1A input capture/compare-match	
	TGI1B	TGR1B input capture/compare-match	
	TCI1V	TCNT1 overflow	
2	TGI2A	TGR2A input capture/compare-match	
	TGI2B	TGR2B input capture/compare-match	
	TCI2V	TCNT2 overflow	
	TCI2U	TCNT2 underflow	

Note: Indicates the initial status following reset. The ranking of channels can be altered using the interrupt controller.

### 8.5.2 A/D Converter Activation

The TGRA register input capture/compare-match of any channel can be used to activate the on-chip A/D converter.

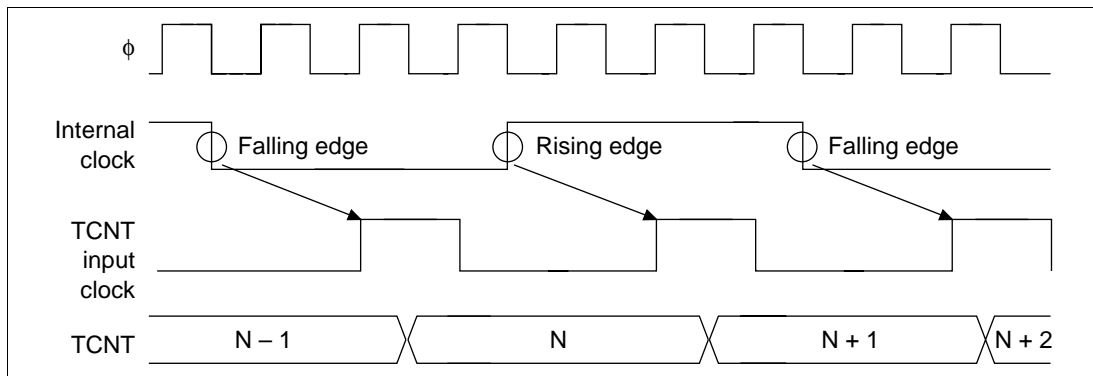
If the TTGE bit of the TIER is already set to 1 when the TGFA flag in the TSR is set to 1 by a TGRA register input capture/compare-match of any of the channels, an A/D conversion start request is sent to the A/D converter. If the MTU conversion start trigger is selected at such a time on the A/D converter side when this happens, the A/D conversion starts.

The MTU has 3 TGRA register input capture/compare-match interrupts, one for each channel, that can be used as A/D converter activation sources.

## 8.6 Operation Timing

### 8.6.1 Input/Output Timing

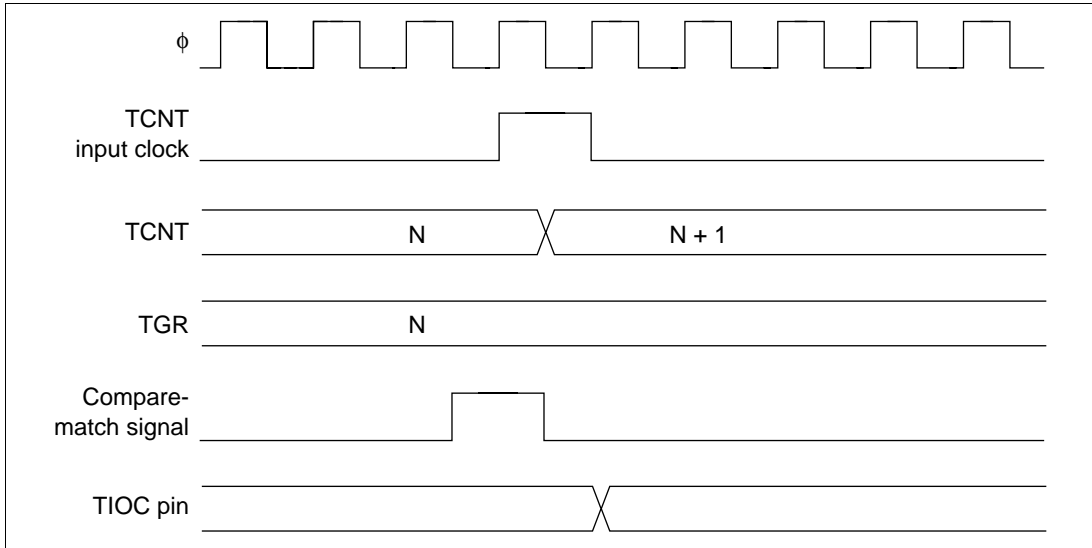
**TCNT Count Timing:** Count timing for the TCNT counter with internal clock operation is shown in figure 8.28.



**Figure 8.28 TCNT Count Timing during Internal Clock Operation**

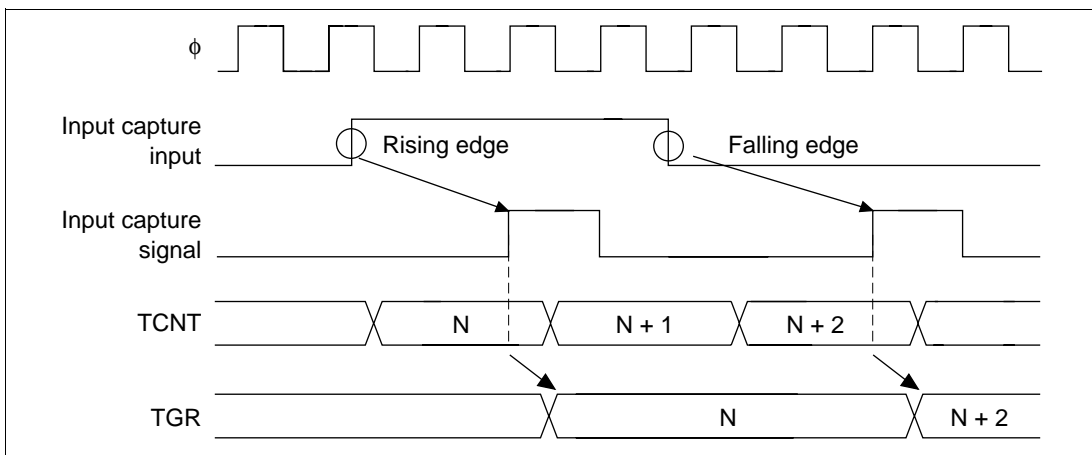
**Output Compare Output Timing:** The compare-match signal is generated at the final state of TCNT and TGR matching. When a compare-match signal is issued, the output value set in TIOR or TOCR is output to the output compare output pin (TIOC pin). After TCNT and TGR matching, a compare-match signal is not issued until immediately before the TCNT input clock.

Output compare output timing (normal mode and PWM mode) is shown in figure 8.29.



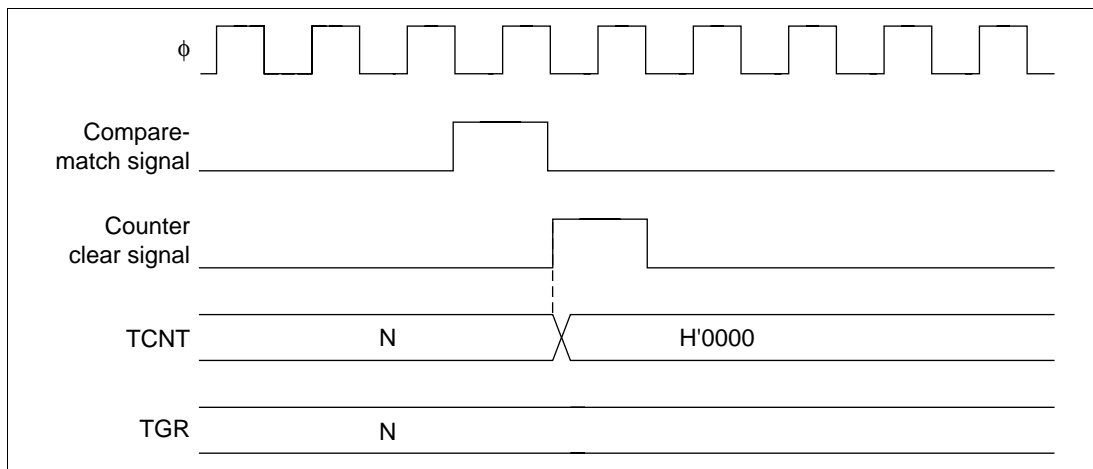
**Figure 8.29 Output Compare Output Timing (Normal Mode/PWM Mode)**

**Input Capture Signal Timing:** Figure 8.30 illustrates input capture timing.

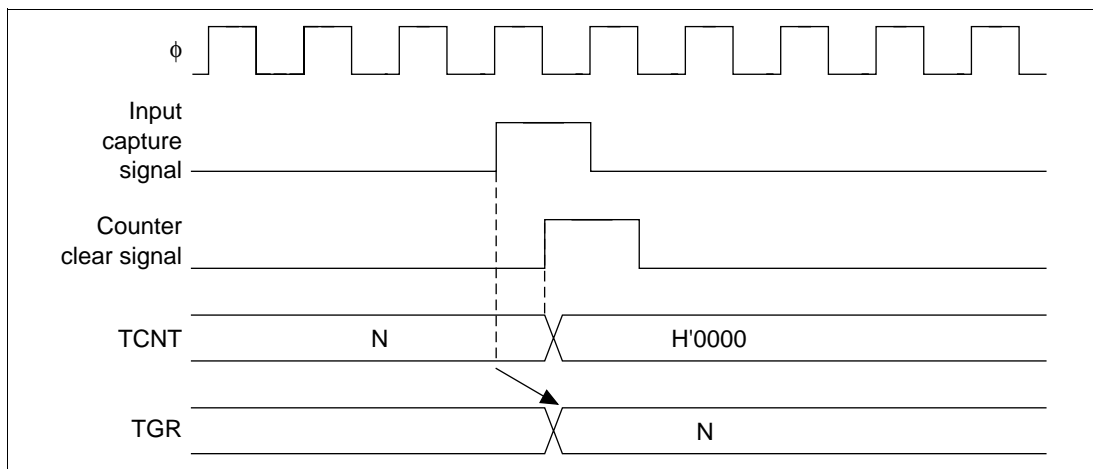


**Figure 8.30 Input Capture Input Signal Timing**

**Counter Clearing Timing Due to Compare-Match/Input Capture:** Timing for counter clearing due to compare-match is shown in figure 8.31. Figure 8.32 shows the timing for counter clearing due to input capture.

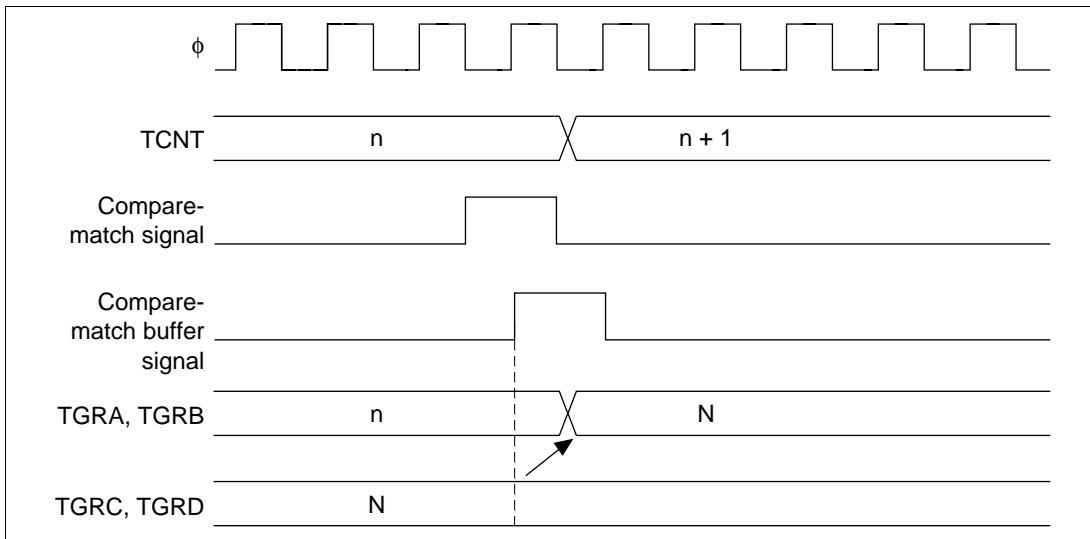


**Figure 8.31 Counter Clearing Timing (Compare-Match)**

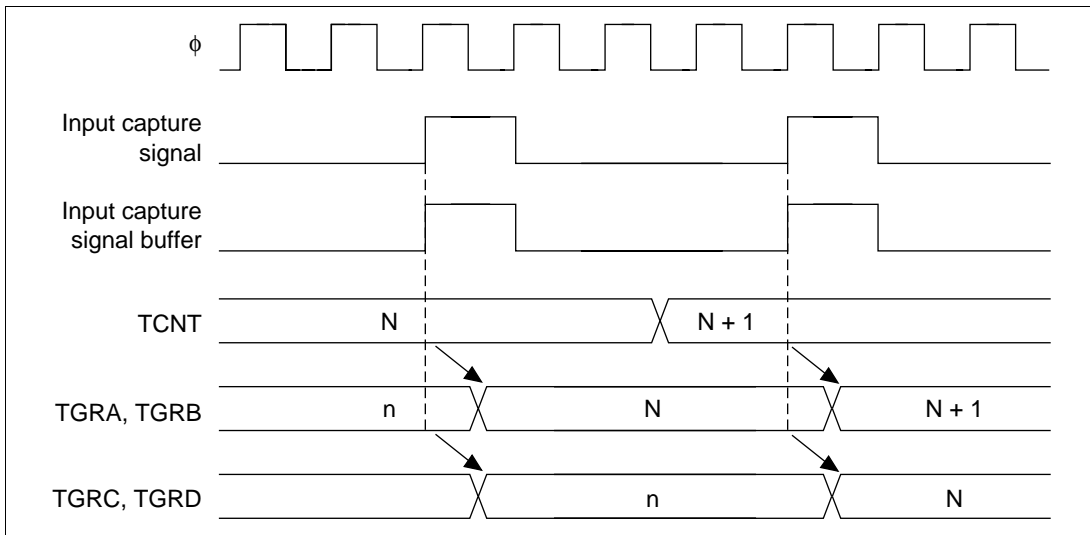


**Figure 8.32 Counter Clearing Timing (Input Capture)**

**Buffer Operation Timing:** Compare-match buffer operation timing is shown in figure 8.33. Figure 8.34 shows input capture buffer operation timing.



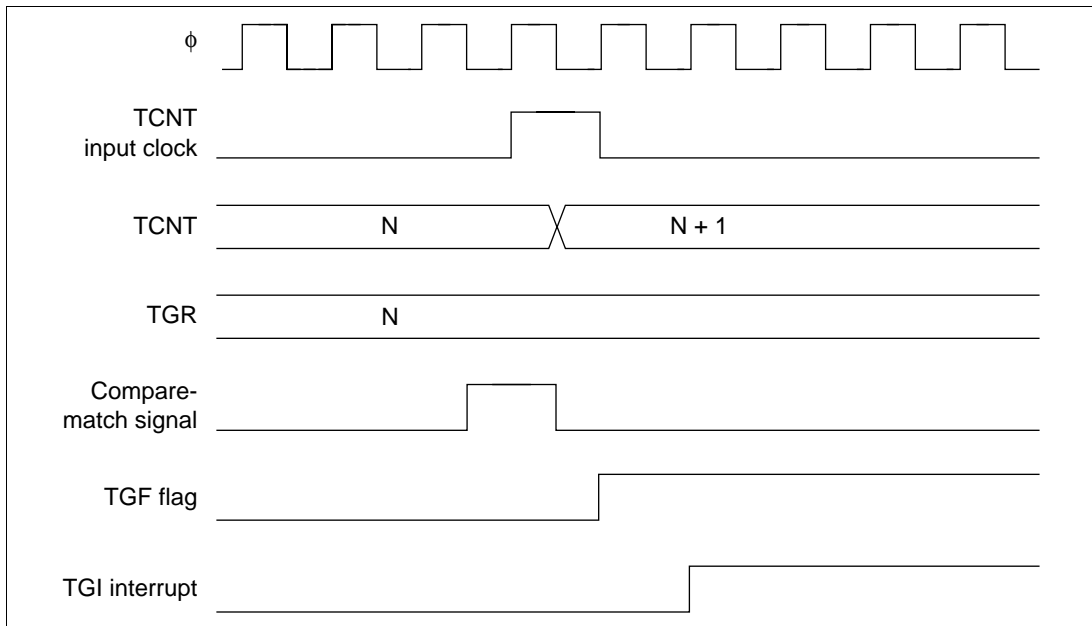
**Figure 8.33 Buffer Operation Timing (Compare-Match)**



**Figure 8.34 Buffer Operation Timing (Input Capture)**

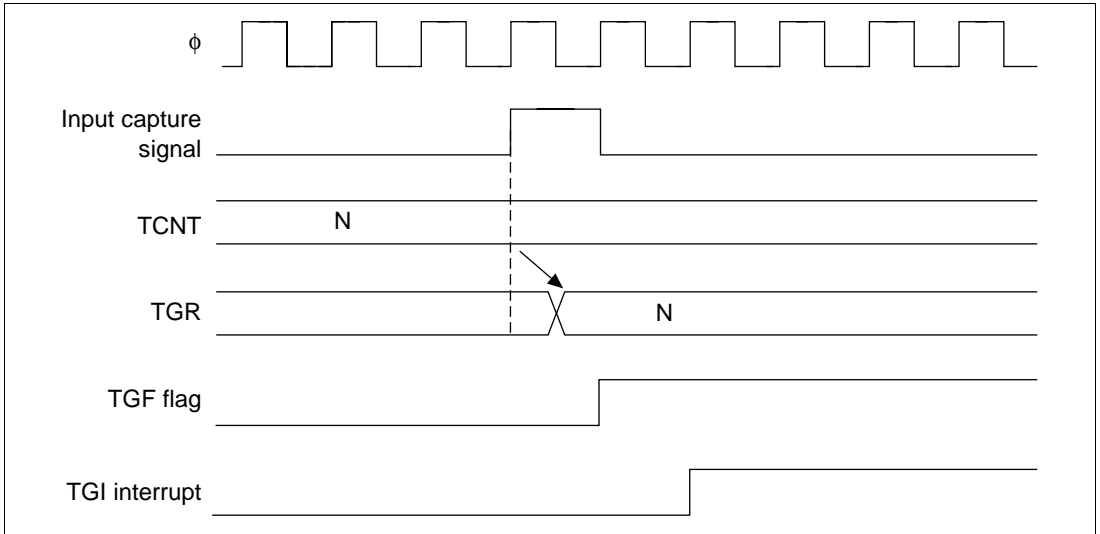
## 8.6.2 Interrupt Signal Timing

**Setting TGF Flag Timing during Compare-Match:** Figure 8.35 shows timing for the TGF flag of the timer status register (TSR) due to compare-match, as well as TGI interrupt request signal timing.



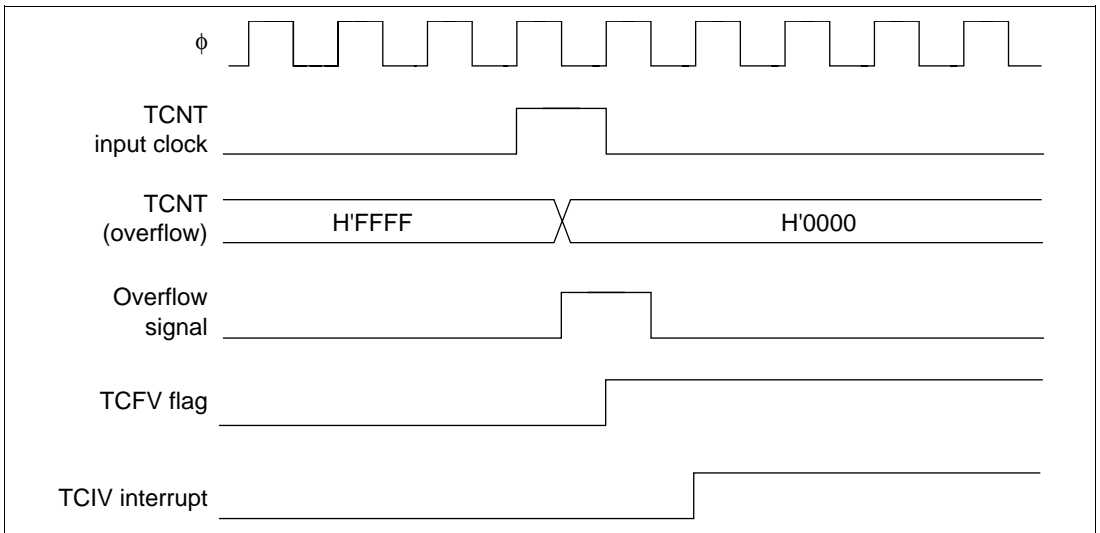
**Figure 8.35 TGI Interrupt Timing (Compare Match)**

**Setting TGF Flag Timing during Input Capture:** Figure 8.36 shows timing for the TGF flag of the timer status register (TSR) due to input capture, as well as TGI interrupt request signal timing.



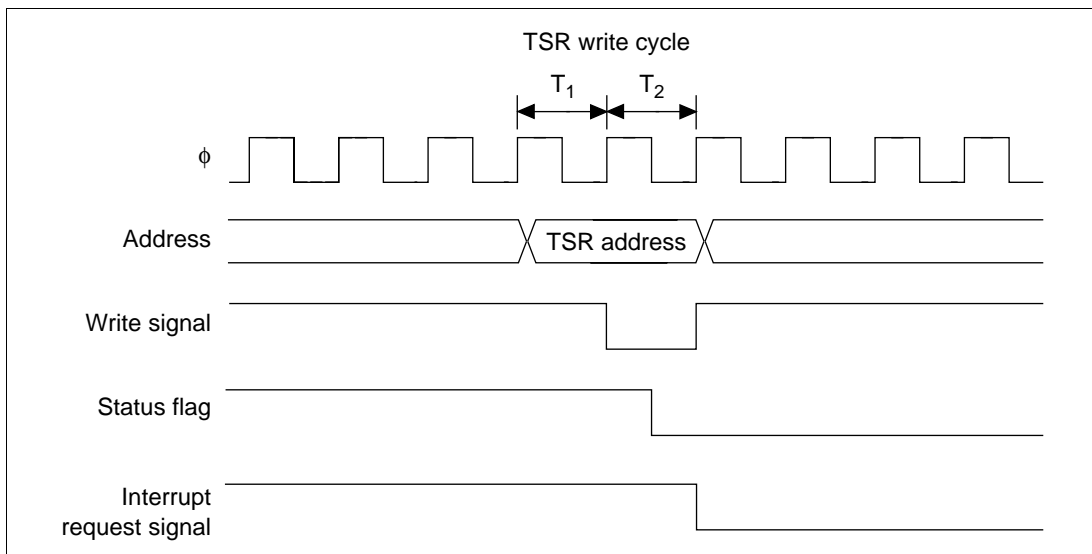
**Figure 8.36 TGI Interrupt Timing (Input Capture)**

**Setting Timing for Overflow Flag (TCFV):** Figure 8.37 shows timing for the TCFV flag of the timer status register (TSR) due to overflow, as well as TCIV interrupt request signal timing.



**Figure 8.37 TCIV Interrupt Setting Timing**

**Status Flag Clearing Timing:** The status flag is cleared when the CPU reads a 1 status followed by a 0 write. Figure 8.38 shows the timing for status flag clearing by the CPU.



**Figure 8.38 Timing of Status Flag Clearing by the CPU**

## 8.7 Notes and Precautions

This section describes contention and other matters requiring special attention during MTU operations.

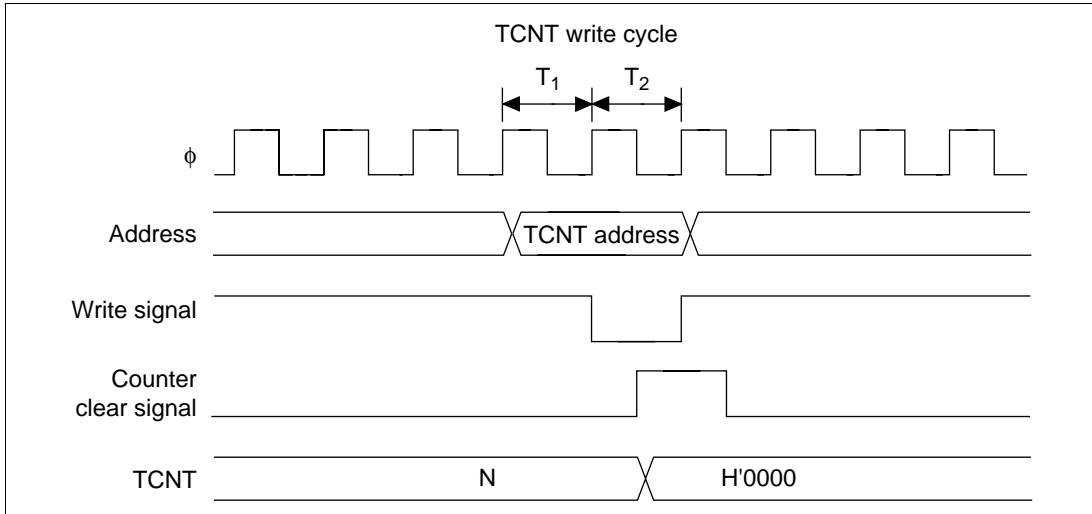
**Note on Cycle Setting:** When setting a counter clearing by compare-match, clearing is done in the final state when TCNT matches the TGR value (update timing for count value on TCNT match). The actual number of states set in the counter is given by the following equation:

$$f = \frac{\phi}{(N + 1)}$$

(f: counter frequency,  $\phi$ : operating frequency, N: value set in the TGR)

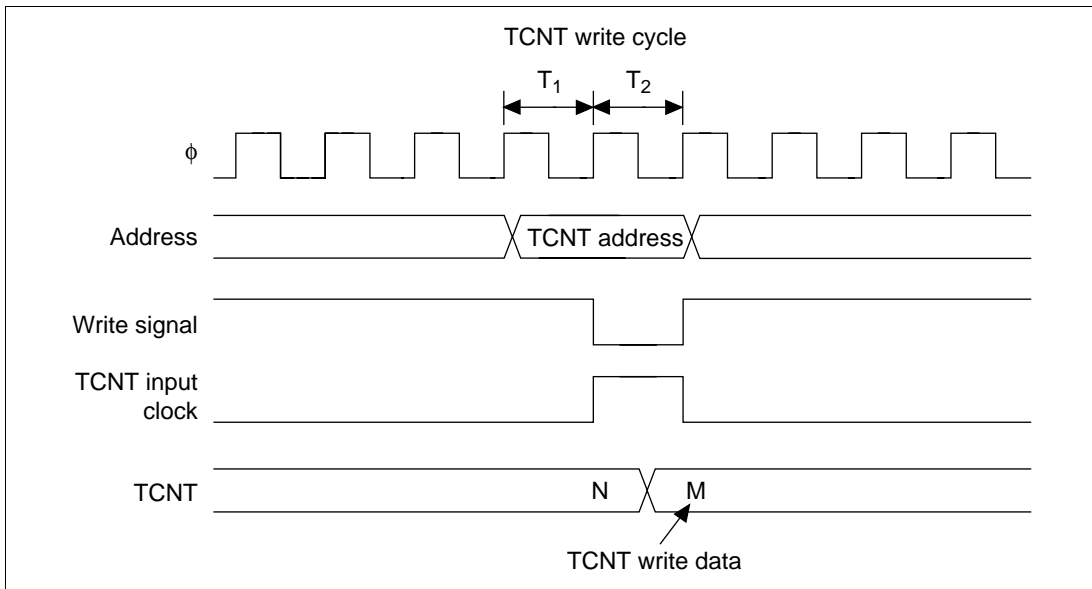
**Contention between TCNT Write and Clear:** If a counter clear signal is issued in the  $T_2$  state during the TCNT write cycle, TCNT clearing has priority, and TCNT write is not conducted (figure 8.39).





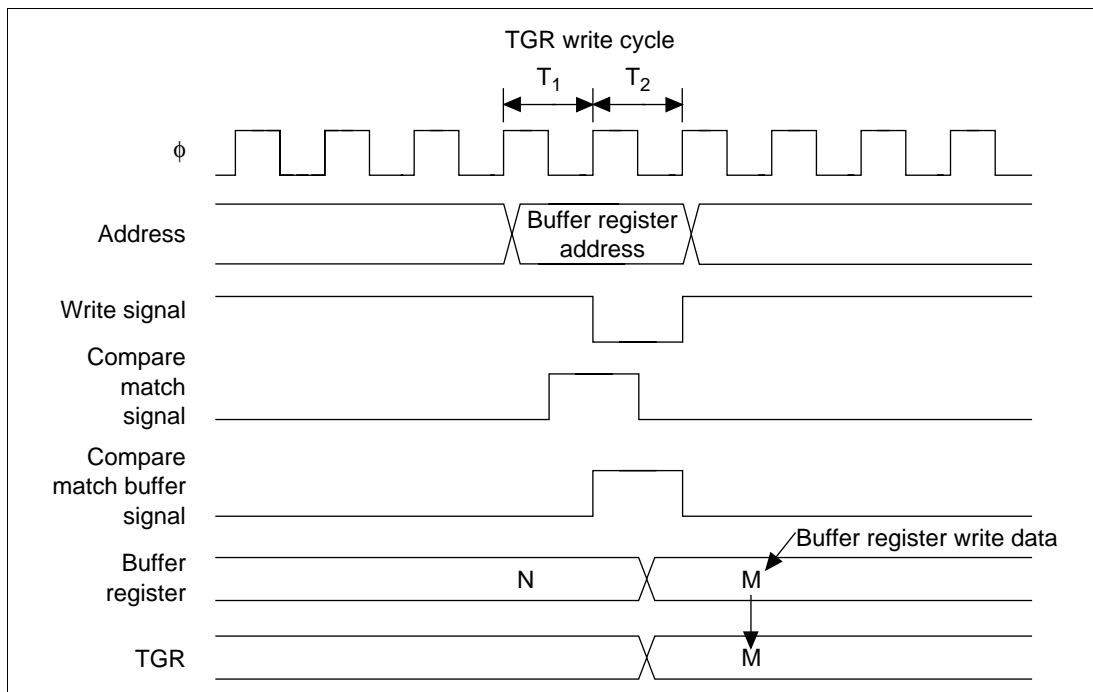
**Figure 8.39 TCNT Write and Clear Contention**

**Contention between TCNT Write and Increment:** If a count-up signal is issued in the  $T_2$  state during the TCNT write cycle, TCNT write has priority, and the counter is not incremented (figure 8.40).



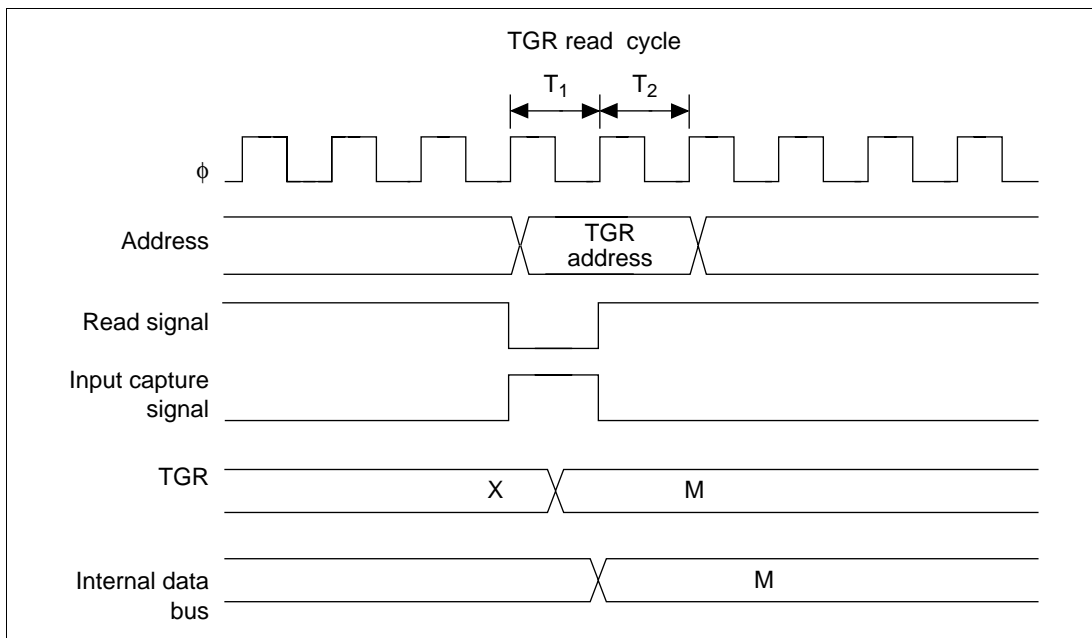
**Figure 8.40 TCNT Write and Increment Contention**

**Contention between Buffer Register Write and Compare Match:** If a compare-match occurs in the  $T_2$  state of the TGR write cycle, data is transferred by the buffer operation from the buffer register to the TGR. On channel 0, the data to be transferred is that after the write (figure 8.41).



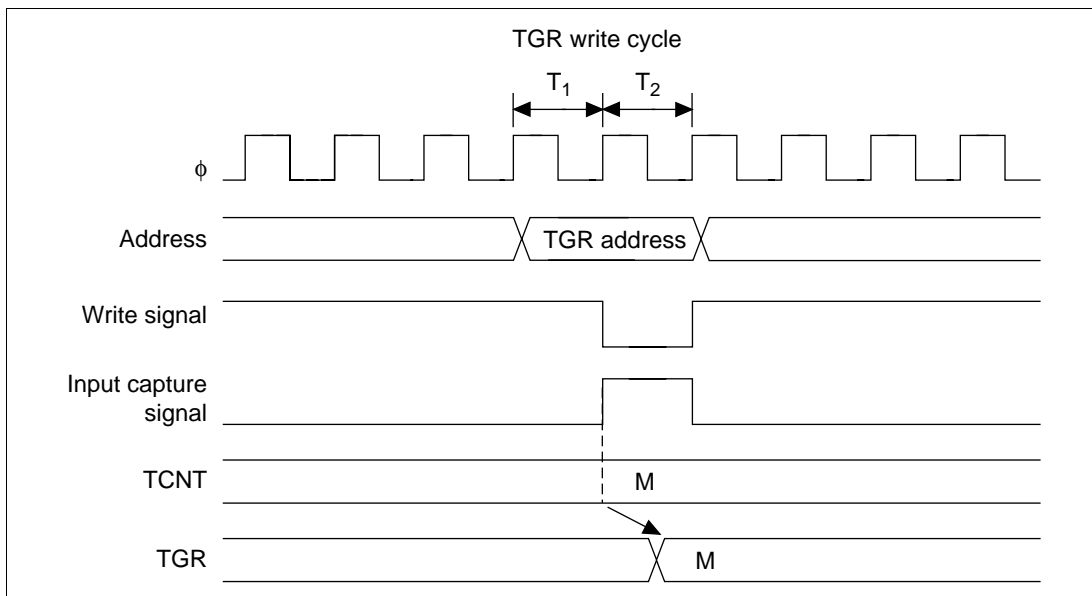
**Figure 8.41 TGR Write and Compare-Match Contention (Channel 0)**

**Contention between TGR Read and Input Capture:** If an input capture signal is issued in the  $T_1$  state of the TGR read cycle, the read data is that after input capture transfer (figure 8.42).



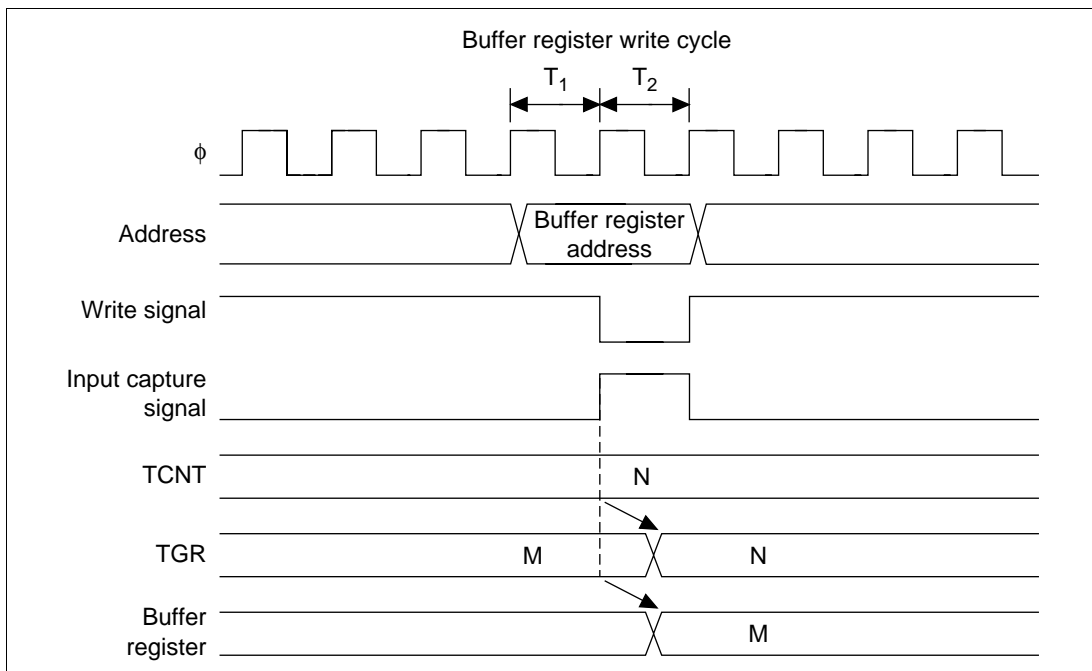
**Figure 8.42 TGR Read and Input Capture Contention**

**Contention between TGR Write and Input Capture:** If an input capture signal is issued in the  $T_2$  state of the TGR read cycle, input capture has priority, and TGR write does not occur (figure 8.43).



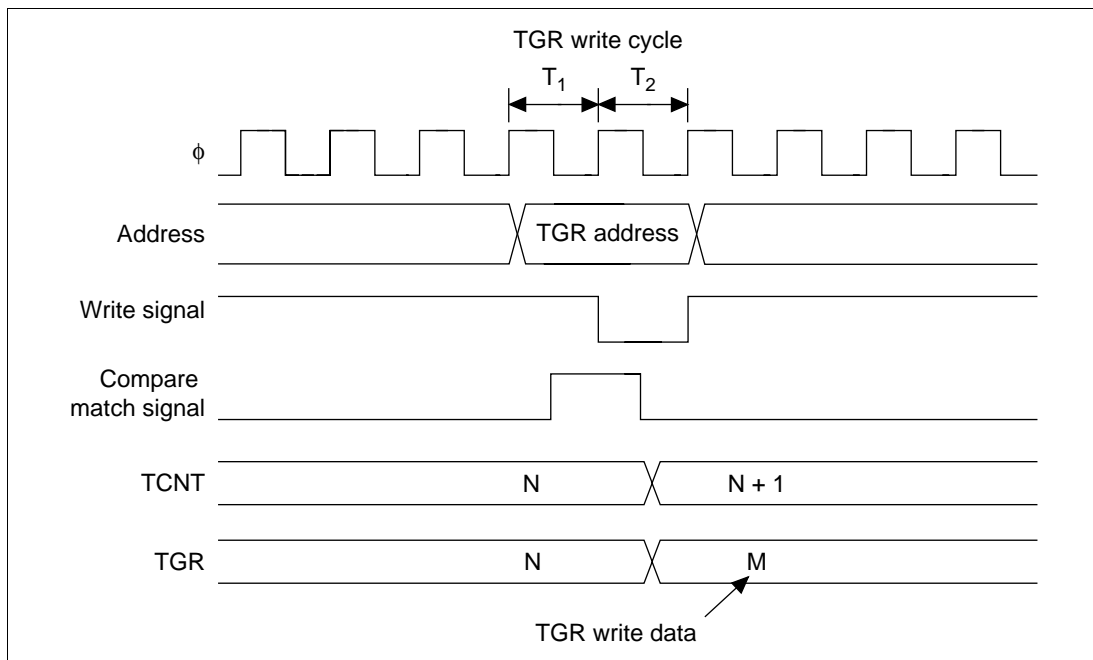
**Figure 8.43 TGR Write and Input Capture Contention**

**Contention between Buffer Register Write and Input Capture:** If an input capture signal is issued in the  $T_2$  state of the buffer write cycle, write to the buffer register does not occur, and buffer operation takes priority (figure 8.44).



**Figure 8.44 Buffer Register Write and Input Capture Contention**

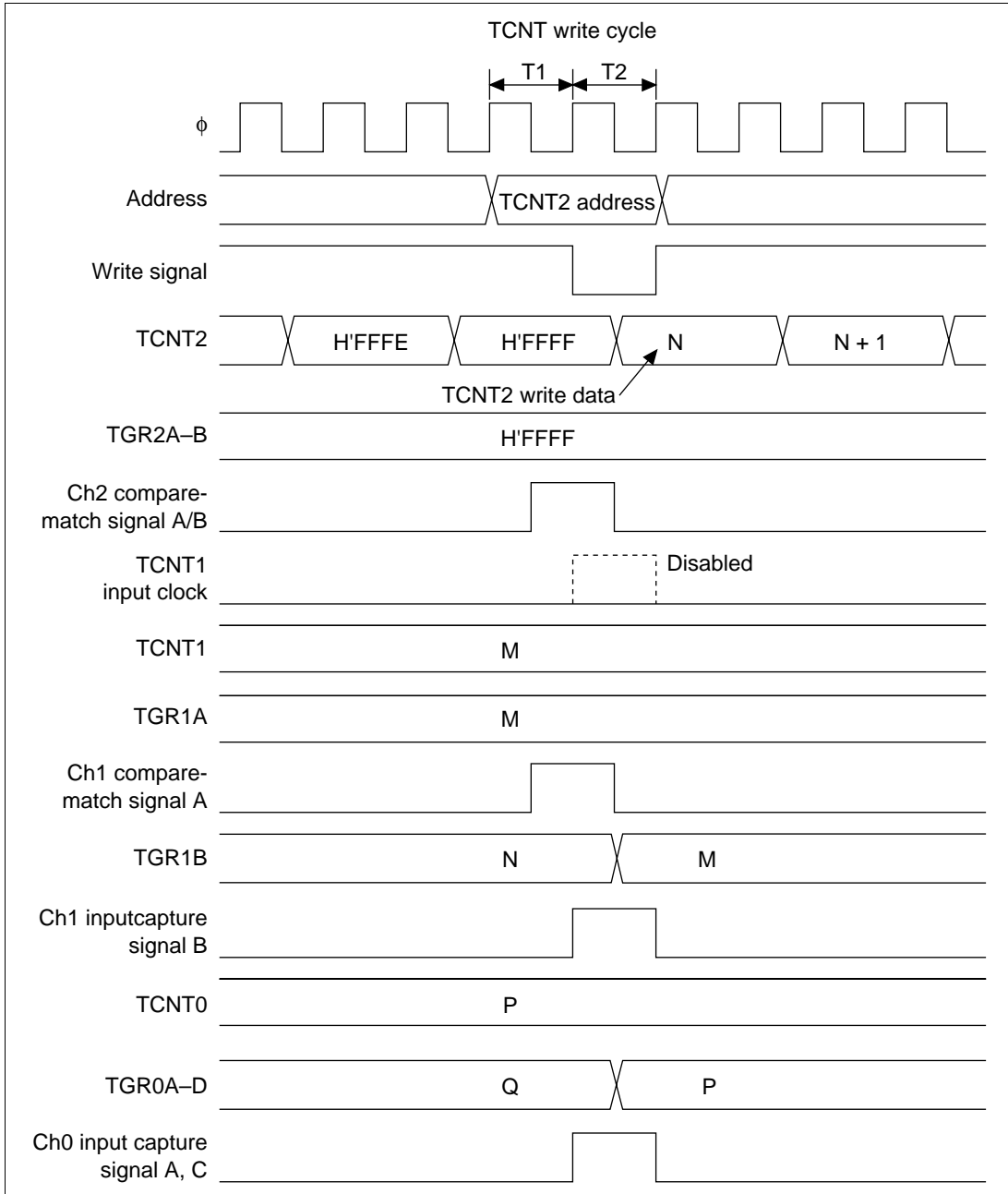
**Contention Between TGR Write and Compare Match:** If a compare-match occurs in the  $T_2$  state of the TGR write cycle, data is written to the TGR and a compare-match signal is issued (figure 8.45).



**Figure 8.45 TGR Write and Compare Match Contention**

**TCNT2 Write and Overflow Contention in Cascade Connection:** With timer counters TCNT1 and TCNT2 in a cascade connection, when a contention occurs during TCNT1 count (during a TCNT2 overflow) in the  $T_2$  state of the TCNT2 write cycle, the write to TCNT2 is conducted, and the TCNT1 count signal is prohibited. At this point, if there is match with TGR1A and the TCNT1 value, a compare signal is issued. When the TCNT1 count clock is selected as the channel 0 input capture source, TGR0A and TGR0C operate as input capture registers. When TGR0C compare-match/input capture is selected as the TGR1B input capture source, TGR1B operates as an input capture register. The timing is shown in figure 8.46.

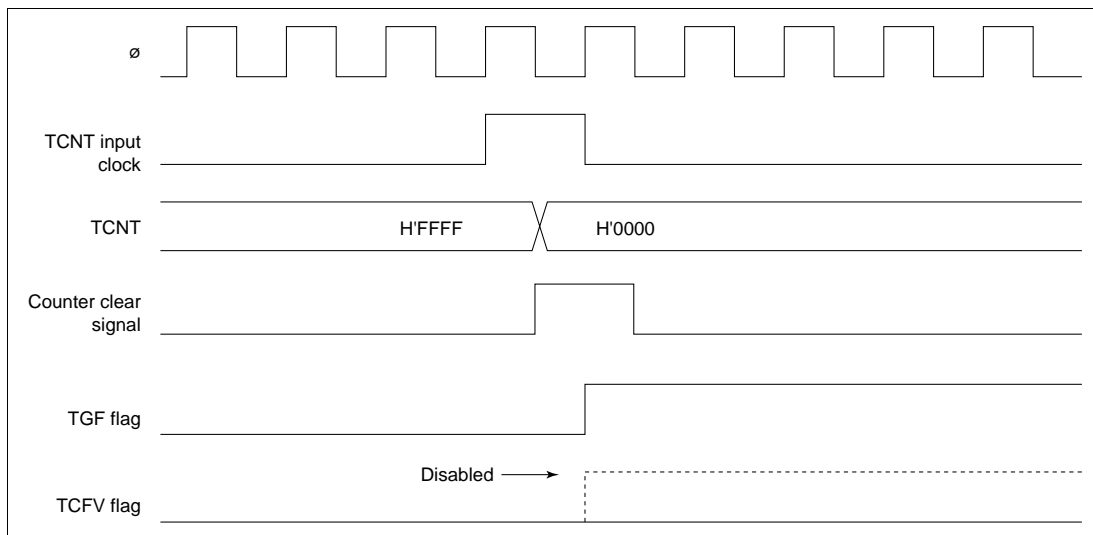
For cascade connections, be sure to synchronize settings for channels 1 and 2 when setting TCNT clearing.



**Figure 8.46 TCNT2 Write and Overflow Contention with Cascade Connection**

**Contention between Overflow and Counter Clearing:** If overflow and counter clearing occur simultaneously, the TCFV flag in TSR is not set and TCNT clearing takes precedence.

Figure 8.47 shows the operation timing when a TGR compare-match is specified as the clearing source, and H'FFFF is set in TGR.

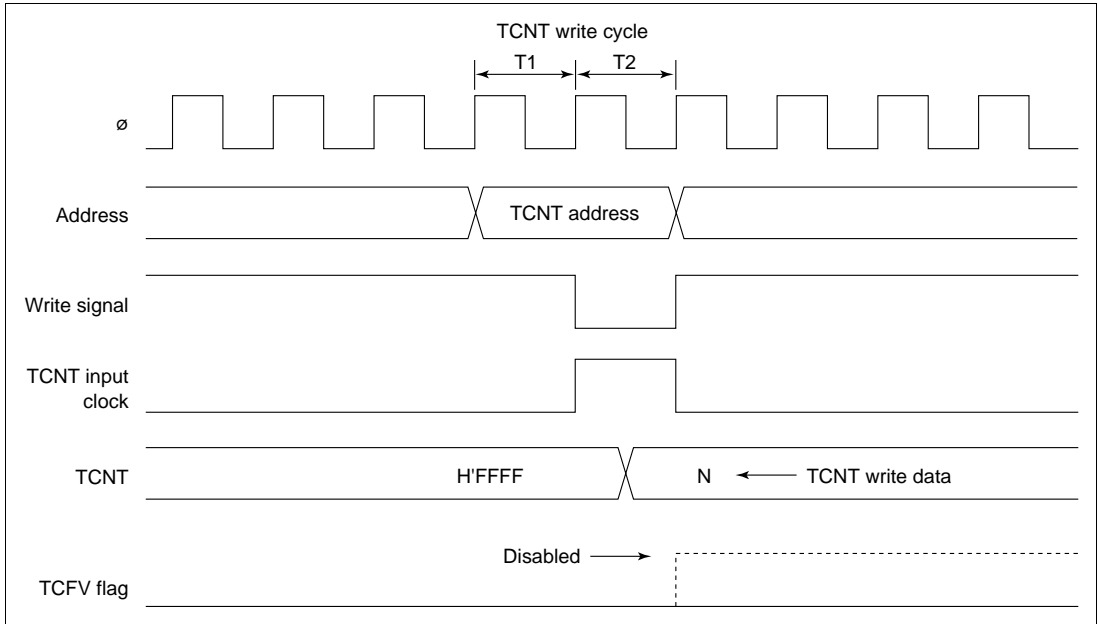


**Figure 8.47 Contention between Overflow and Counter Clearing**

**Contention between TCNT Write and Overflow:** If there is an up-count in the T2 state of a TCNT write cycle, and overflow occurs, the TCNT write takes precedence and the TCFV flag in TSR is not set .

Figure 8.48 shows the operation timing in this case.





**Figure 8.48 Contention between TCNT Write and Overflow**

## 8.8 MTU Output Pin Initialization

### 8.8.1 Operating Modes

The MTU has the following three operating modes. Waveform output is possible in all of these modes.

- Normal mode (channels 0 to 2)
- PWM mode 1 (channels 0 to 2)
- PWM mode 2 (channels 0 to 2)

The MTU output pin initialization method for each of these modes is described in this section.

## 8.8.2 Reset Start Operation

The MTU output pins (TIOC\*) are initialized low by a reset and in standby mode. Since MTU pin function selection is performed by the pin function controller (PFC), when the PFC is set, the MTU pin states at that point are output to the ports. When MTU output is selected by the PFC immediately after a reset, the MTU output initial level, low, is output directly at the port. When the active level is low, the system will operate at this point, and therefore the PFC setting should be made after initialization of the MTU output pins is completed.

Note: \* represents the channel number and port symbol.

## 8.8.3 Operation in Case of Re-Setting Due to Error During Operation, Etc.

If an error occurs during MTU operation, MTU output should be cut by the system. Cutoff is performed by switching the pin output to port output with the PFC and outputting the inverse of the active level. The pin initialization procedures for re-setting due to an error during operation, etc., and the procedures for restarting in a different mode after re-setting, are shown below.

The MTU has three operating modes, as stated above. There are thus 9 mode transition combinations. Possible mode transition combinations are shown in table 8.9.

**Table 8.9 Mode Transition Combinations**

Before	After		
	Normal	PWM1	PWM2
Normal	(1)	(2)	(3)
PWM1	(4)	(5)	(6)
PWM2	(7)	(8)	(9)

Legend:

Normal: Normal mode

PWM1: PWM1 mode

PWM2: PWM2 mode

The above abbreviations are used in some places in the following descriptions.

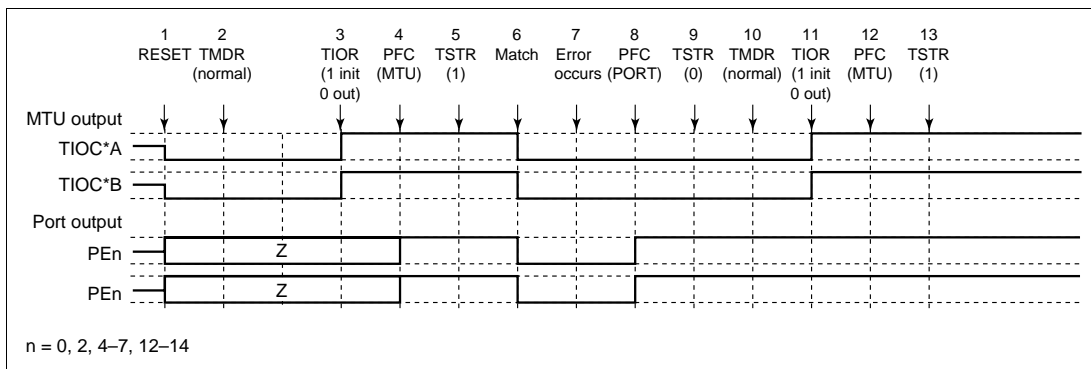
#### 8.8.4 Overview of Initialization Procedures and Mode Transitions in Case of Error During Operation, Etc.

- When making a transition to a mode (Normal, PWM1, PWM2) in which the pin output level is selected by the timer I/O control register (TIOR) setting, initialize the pins by means of a TIOR setting.
- In PWM mode 1, since a waveform is not output to the TIOC\*B pin, setting TIOR will not initialize the pins. If initialization is required, carry it out in normal mode, then switch to PWM mode 1.
- In PWM mode 2, since a waveform is not output to the cycle register pin, setting TIOR will not initialize the pins. If initialization is required, carry it out in normal mode, then switch to PWM mode 2.
- In normal mode or PWM mode 2, if TGRC and TGRD operate as buffer registers, setting TIOR will not initialize the buffer register pins. If initialization is required, clear buffer mode, carry out initialization, then set buffer mode again.
- In PWM mode 1, if either TGRC or TGRD operates as a buffer register, setting TIOR will not initialize the TGRC pin. To initialize the TGRC pin, clear buffer mode, carry out initialization, then set buffer mode again.

Note: An asterisk in this section represents the channel number.

Pin initialization procedures are described below for the numbered combinations in table 8.9. The active level is assumed to be low.

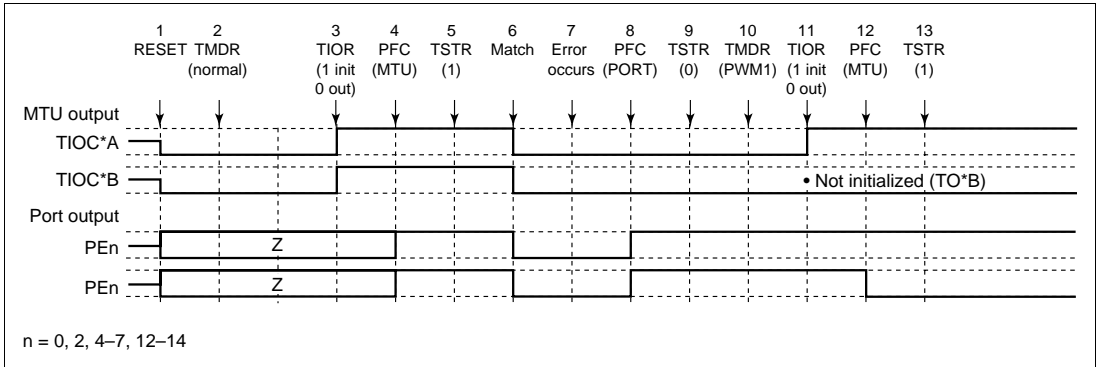
**(1) Operation when Error Occurs during Normal Mode Operation, and Operation is Restarted in Normal Mode:** Figure 8.49 shows an explanatory diagram of the case where an error occurs in normal mode and operation is restarted in normal mode after re-setting.



**Figure 8.49 Error Occurrence in Normal Mode, Recovery in Normal Mode**

1. After a reset, MTU output is low and ports are in the high-impedance state.
2. After a reset, the TMDR setting is for normal mode.
3. Initialize the pins with TIOR. (The example shows initial high output, with low output on compare-match occurrence.)
4. Set MTU output with the PFC.
5. The count operation is started by TSTR.
6. Output goes low on compare-match occurrence.
7. An error occurs.
8. Set port output with the PFC and output the inverse of the active level.
9. The count operation is stopped by TSTR.
10. Not necessary when restarting in normal mode.
11. Initialize the pins with TIOR.
12. Set MTU output with the PFC.
13. Operation is restarted by TSTR.

**(2) Operation when Error Occurs during Normal Mode Operation, and Operation is Restarted in PWM Mode 1:** Figure 8.50 shows an explanatory diagram of the case where an error occurs in normal mode and operation is restarted in PWM mode 1 after re-setting.



**Figure 8.50 Error Occurrence in Normal Mode, Recovery in PWM Mode 1**

1 to 9 are the same as in figure 8.49.

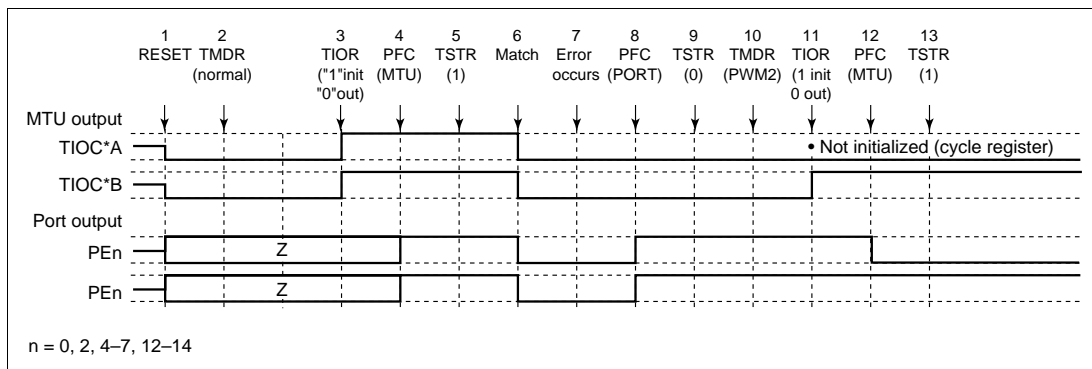
10. Set PWM mode 1.

11. Initialize the pins with TIOR. (In PWM mode 1, the TO\*B side is not initialized. If initialization is required, initialize in normal mode, then switch to PWM mode 1.)

12. Set MTU output with the PFC.

13. Operation is restarted by TSTR.

**(3) Operation when Error Occurs during Normal Mode Operation, and Operation is Restarted in PWM Mode 2:** Figure 8.51 shows an explanatory diagram of the case where an error occurs in normal mode and operation is restarted in PWM mode 2 after re-setting.



**Figure 8.51 Error Occurrence in Normal Mode, Recovery in PWM Mode 2**

1 to 9 are the same as in figure 8.49.

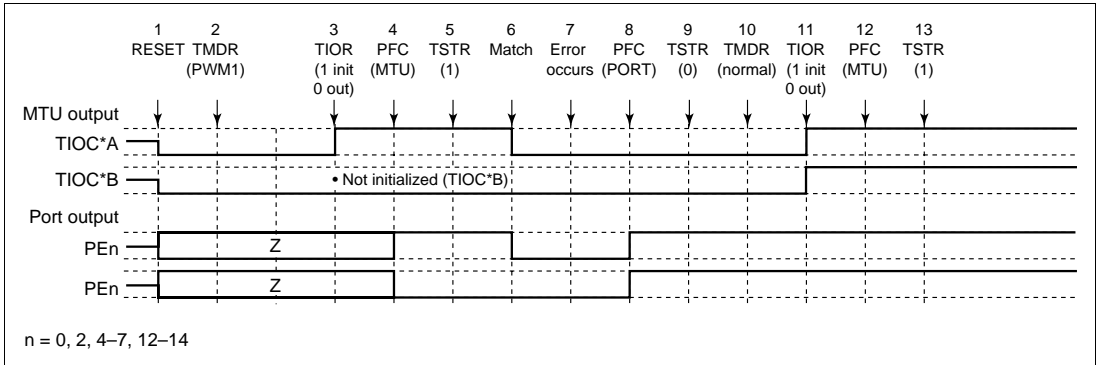
10. Set PWM mode 2.

11. Initialize the pins with TIOR. (In PWM mode 2, the cycle register pins are not initialized. If initialization is required, initialize in normal mode, then switch to PWM mode 2.)

12. Set MTU output with the PFC.

13. Operation is restarted by TSTR.

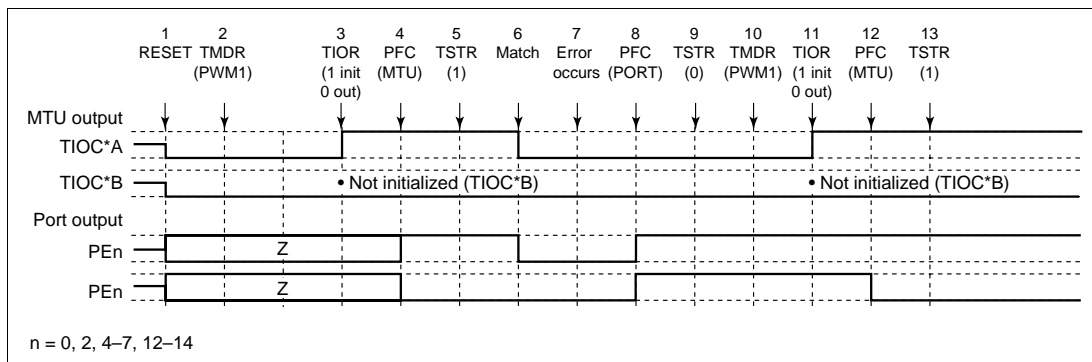
**(4) Operation when Error Occurs during PWM Mode 1 Operation, and Operation is Restarted in Normal Mode:** Figure 8.52 shows an explanatory diagram of the case where an error occurs in PWM mode 1 and operation is restarted in normal mode after re-setting.



**Figure 8.52 Error Occurrence in PWM Mode 1, Recovery in Normal Mode**

1. After a reset, MTU output is low and ports are in the high-impedance state.
2. Set PWM mode 1.
3. Initialize the pins with TIOR. (The example shows initial high output, with low output on compare-match occurrence. In PWM mode 1, the TIOC\*B side is not initialized.)
4. Set MTU output with the PFC.
5. The count operation is started by TSTR.
6. Output goes low on compare-match occurrence.
7. An error occurs.
8. Set port output with the PFC and output the inverse of the active level.
9. The count operation is stopped by TSTR.
10. Set normal mode.
11. Initialize the pins with TIOR.
12. Set MTU output with the PFC.
13. Operation is restarted by TSTR.

**(5) Operation when Error Occurs during PWM Mode 1 Operation, and Operation is Restarted in PWM Mode 1:** Figure 8.53 shows an explanatory diagram of the case where an error occurs in PWM mode 1 and operation is restarted in PWM mode 1 after re-setting.



**Figure 8.53 Error Occurrence in PWM Mode 1, Recovery in PWM Mode 1**

1 to 9 are the same as in figure 8.52.

10. Not necessary when restarting in PWM mode 1.

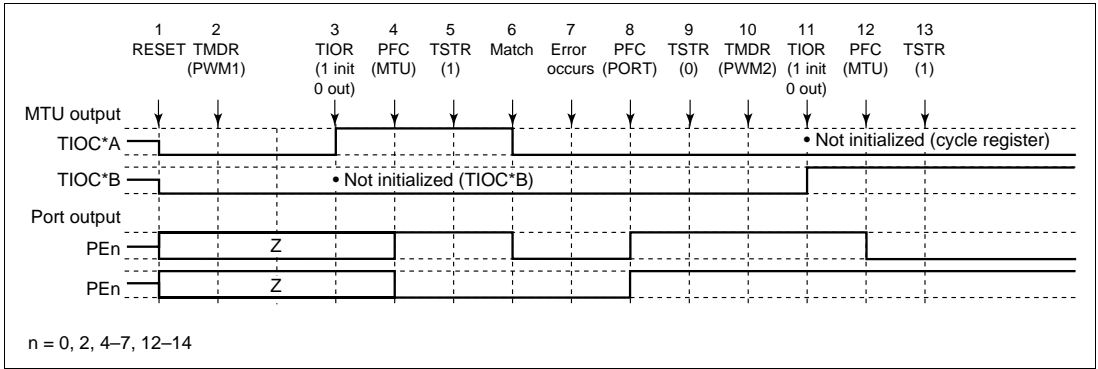
11. Initialize the pins with TIOR. (In PWM mode 1, the TIOC\*B side is not initialized.)

12. Set MTU output with the PFC.

13. Operation is restarted by TSTR.



**(6) Operation when Error Occurs during PWM Mode 1 Operation, and Operation is Restarted in PWM Mode 2:** Figure 8.54 shows an explanatory diagram of the case where an error occurs in PWM mode 1 and operation is restarted in PWM mode 2 after re-setting.



**Figure 8.54 Error Occurrence in PWM Mode 1, Recovery in PWM Mode 2**

1 to 9 are the same as in figure 8.52.

10. Set PWM mode 2.

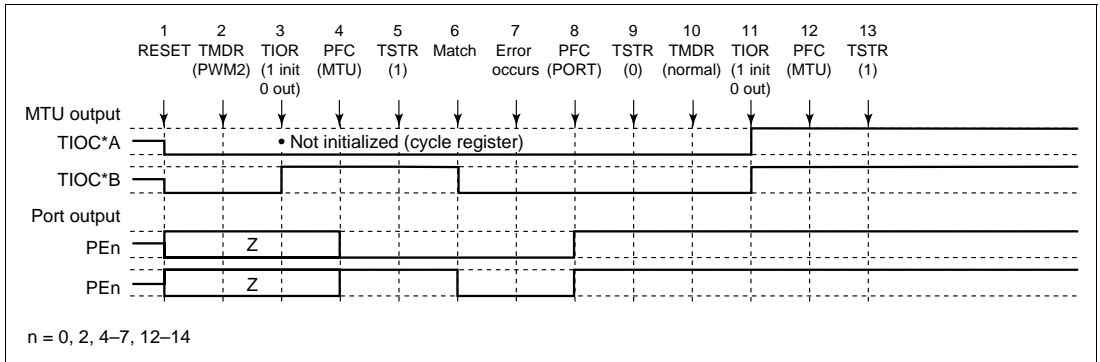
11. Initialize the pins with TIOR. (In PWM mode 2, the cycle register pins are not initialized.)

12. Set MTU output with the PFC.

13. Operation is restarted by TSTR.

### (7) Operation when Error Occurs during PWM Mode 2 Operation, and Operation is

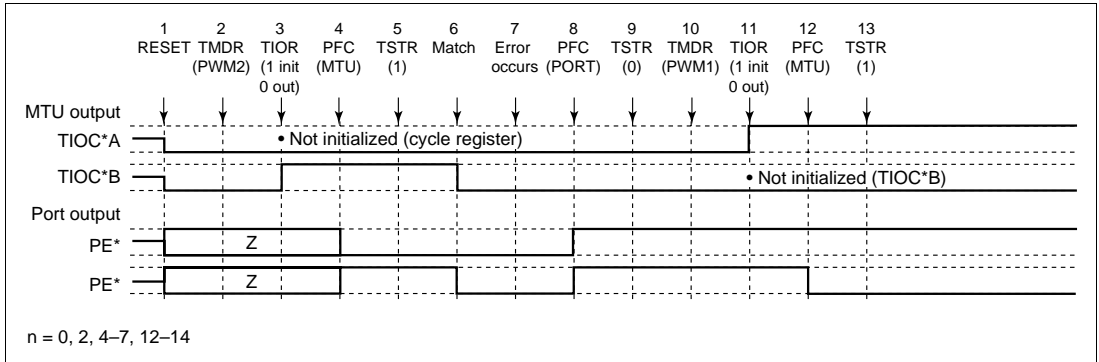
**Restarted in Normal Mode:** Figure 8.55 shows an explanatory diagram of the case where an error occurs in PWM mode 2 and operation is restarted in normal mode after re-setting.



**Figure 8.55 Error Occurrence in PWM Mode 2, Recovery in Normal Mode**

1. After a reset, MTU output is low and ports are in the high-impedance state.
2. Set PWM mode 2.
3. Initialize the pins with TIOR. (The example shows initial high output, with low output on compare-match occurrence. In PWM mode 2, the cycle register pins are not initialized. In the example, TIOC\*A is the cycle register.)
4. Set MTU output with the PFC.
5. The count operation is started by TSTR.
6. Output goes low on compare-match occurrence.
7. An error occurs.
8. Set port output with the PFC and output the inverse of the active level.
9. The count operation is stopped by TSTR.
10. Set normal mode.
11. Initialize the pins with TIOR.
12. Set MTU output with the PFC.
13. Operation is restarted by TSTR.

**(8) Operation when Error Occurs during PWM Mode 2 Operation, and Operation is Restarted in PWM Mode 1:** Figure 8.56 shows an explanatory diagram of the case where an error occurs in PWM mode 2 and operation is restarted in PWM mode 1 after re-setting.



**Figure 8.56 Error Occurrence in PWM Mode 2, Recovery in PWM Mode 1**

1 to 9 are the same as in figure 8.55.

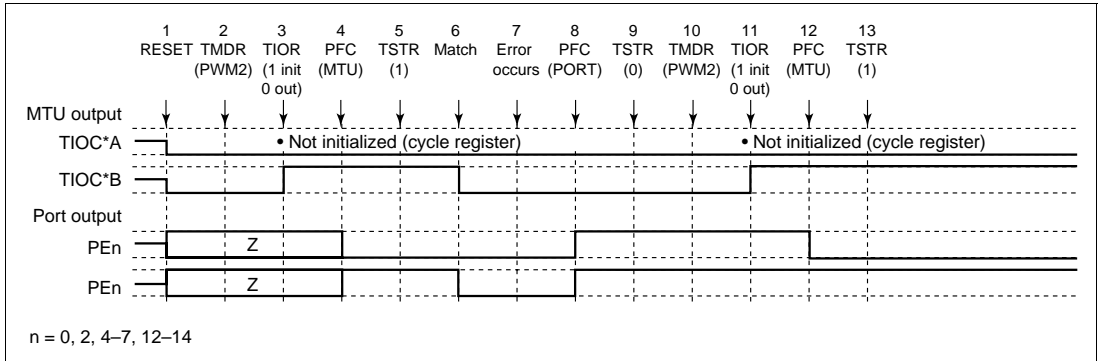
10. Set PWM mode 1.

11. Initialize the pins with TIOR. (In PWM mode 1, the TIOC\*B side is not initialized.)

12. Set MTU output with the PFC.

13. Operation is restarted by TSTR.

**(9) Operation when Error Occurs during PWM Mode 2 Operation, and Operation is Restarted in PWM Mode 2:** Figure 8.57 shows an explanatory diagram of the case where an error occurs in PWM mode 2 and operation is restarted in PWM mode 2 after re-setting.



**Figure 8.57 Error Occurrence in PWM Mode 2, Recovery in PWM Mode 2**

1 to 9 are the same as in figure 8.55.

10. Not necessary when restarting in PWM mode 2.

11. Initialize the pins with TIOR. (In PWM mode 2, the cycle register pins are not initialized.)

12. Set MTU output with the PFC.

13. Operation is restarted by TSTR.

# Section 9 8-Bit Timer 1 (TIM1)

## 9.1 Overview

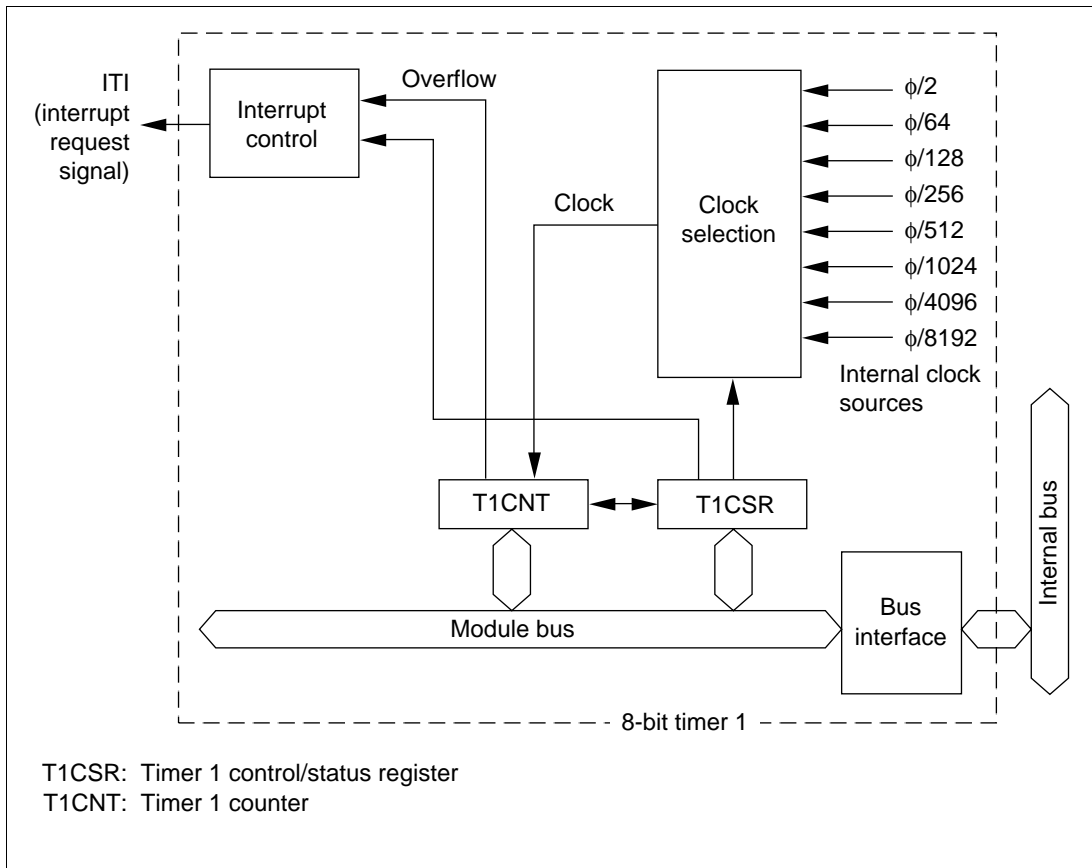
8-bit timer 1 (TIM1) is a single-channel interval timer that generates an interval timer interrupt each time the counter overflows.

### 9.1.1 Features

- 8-bit interval timer
- Generates interval timer interrupts  
An interval timer interrupt is generated each time the counter overflows.
- Selection of eight counter input clock sources

## 9.1.2 Block Diagram

Figure 9.1 shows a block diagram of 8-bit timer 1 (TIM1).



**Figure 9.1 Block Diagram of 8-Bit Timer 1**

### 9.1.3 Register Configuration

8-bit timer 1 (TIM1) has two registers, as shown in table 9.1. These registers perform clock selection and other functions.

**Table 9.1 8-Bit Timer 1 Registers**

Name	Abbreviation	R/W	Initial Value	Address	
				Write* <sup>1</sup>	Read* <sup>2</sup>
Timer 1 control/ status register	T1CSR	R/(W)* <sup>3</sup>	H18	H'FFFF8610	H'FFFF8610
Timer 1 counter	T1CNT	R/W	H'00		H'FFFF8611

- Notes:
1. Use word-length writes. Byte and longword writes cannot be used.
  2. Use byte-length reads. A word or longword read will not return the correct value.
  3. Only 0 can be written in bit 7, to clear the flag.

## 9.2 Register Descriptions

### 9.2.1 Timer 1 Counter (T1CNT)

The timer 1 counter (T1CNT) is an 8-bit readable/writable\* up-counter. When the timer enable bit (TME) is set to 1 in the timer 1 control/status register (T1CSR), T1CNT starts incrementing on the internal clock selected by bits CKS2—CKS0 in T1CSR. When the T1CNT value overflows (from H'FF to H'00), an interval timer interrupt (ITI) is generated.

T1CNT is initialized to H'00 by a hardware reset or when the TME bit is 0.

Note: The method for writing to T1CNT is different from that for general registers to prevent inadvertent overwriting. For details, see section 9.2.3, Notes on Register Access.

Bit:	7	6	5	4	3	2	1	0
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

## 9.2.2 Timer 1 Control/Status Register (T1CSR)

The timer 1 control/status register (T1CSR) is an 8-bit readable/writable\* register that selects the clock to be input to the timer 1 counter (T1CNT) and the timer mode.

Bits 7, 5, and 2 through 0 are initialized to 0 by a power-on reset.

Note: The method for writing to T1CSR is different from that for general registers to prevent inadvertent overwriting. For details, see section 9.2.3, Notes on Register Access.

Bit:	7	6	5	4	3	2	1	0
	OVF	—	TME	—	—	CKS2	CKS1	CKS0
Initial value:	0	0	0	1	1	0	0	0
R/W:	R/(W)*	R	R/W	R	R	R/W	R/W	R/W

- Bit 7—Overflow Flag (OVF): Indicates that T1CNT has overflowed from H'FF to H'00.

### Bit 7: OVF Description

0	No T1CNT overflow (initial value) [Clearing condition] Cleared by reading OVF then writing 0 in OVF
1	[Setting condition] Set when T1CNT overflows

- Bit 6—Reserved: This bit always reads 0 and must only be written with 0.
- Bit 5—Timer Enable (TME): Selects whether TCNT runs or is halted.

### Bit 5: TME Description

0	Timer disabled: T1CNT is initialized to H'00 and halted (initial value)
1	Timer enabled: T1CNT starts counting, and an interrupt is generated when T1CNT overflows

- Bits 4 and 3—Reserved: These bits always read 1 and must only be written with 1.
- Bits 2 to 0—Clock Select 2 to 0 (CKS2–CKS0): These bits select one of eight internal clock sources, obtained by dividing the system clock ( $\phi$ ), for input to T1CNT.



Bit 2: CKS2	Bit 1: CKS1	Bit 0: CKS0	Description	
			Clock	Overflow Period* (when $\phi = 20.0$ MHz)
0	0	0	$\phi/2$ (initial value)	25.6 $\mu$ s
		1	$\phi/64$	819.2 $\mu$ s
	1	0	$\phi/128$	1.6384 ms
		1	$\phi/256$	3.2768 ms
1	0	0	$\phi/512$	6.5536 ms
		1	$\phi/1024$	13.1072 ms
	1	0	$\phi/4096$	52.4288 ms
		1	$\phi/8192$	104.8576 ms

Note: The overflow period is the time from when T1CNT starts counting up from H'00 until overflow occurs.

### 9.2.3 Notes on Register Access

The method for writing to the timer 1 counter (T1CNT) and the timer 1 control/status register (T1CSR) is different from that for general registers to prevent inadvertent overwriting. The procedures for writing to and reading these registers are given below.

**Writing to T1CNT and T1CSR:** These registers must be written to with a word transfer instruction. They cannot be written to with a byte instruction.

T1CNT and T1CSR both have the same write address. For a write to T1CNT, the upper byte of the written word must contain H'5A and the lower byte must contain the write data. For a write to T1CSR, the upper byte of the written word must contain H'A5 and the lower byte must contain the write data. This transfers the write data from the lower byte to T1CNT or T1CSR. (See figure 9.2).

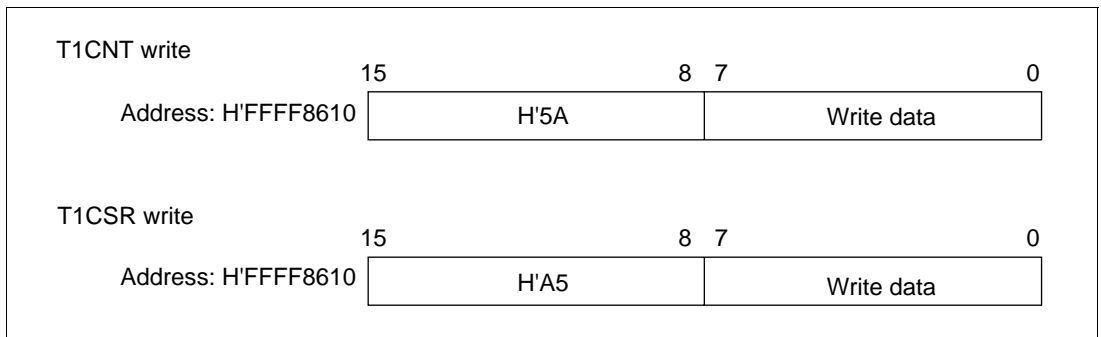


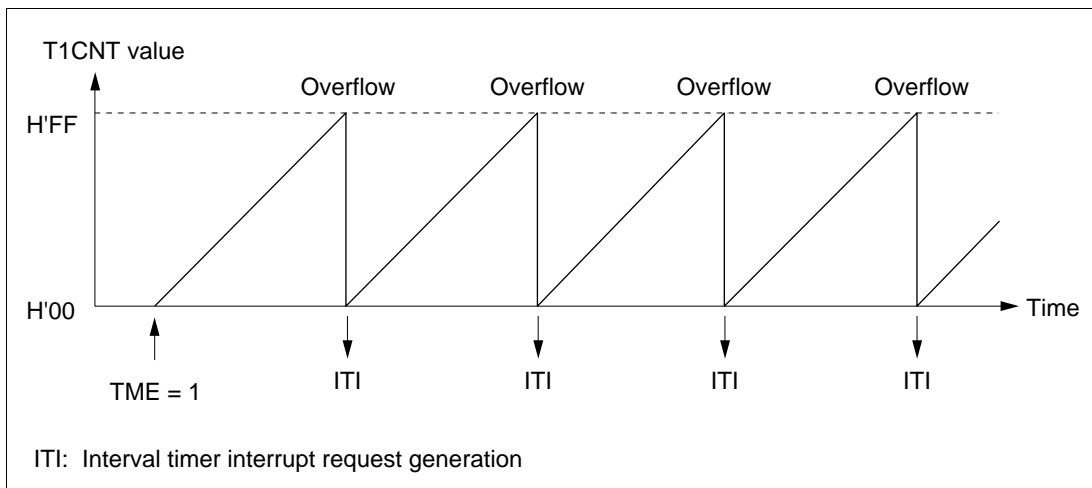
Figure 9.2 Writing to T1CNT and T1CSR

**Reading T1CNT and T1CSR:** These registers are read in the same way as other registers. The read addresses are H'FFFF8610 for T1CSR and H'FFFF8611 for T1CNT. A byte transfer instruction must be used to read these registers.

## 9.3 Operation

### 9.3.1 Interval Timer Operation

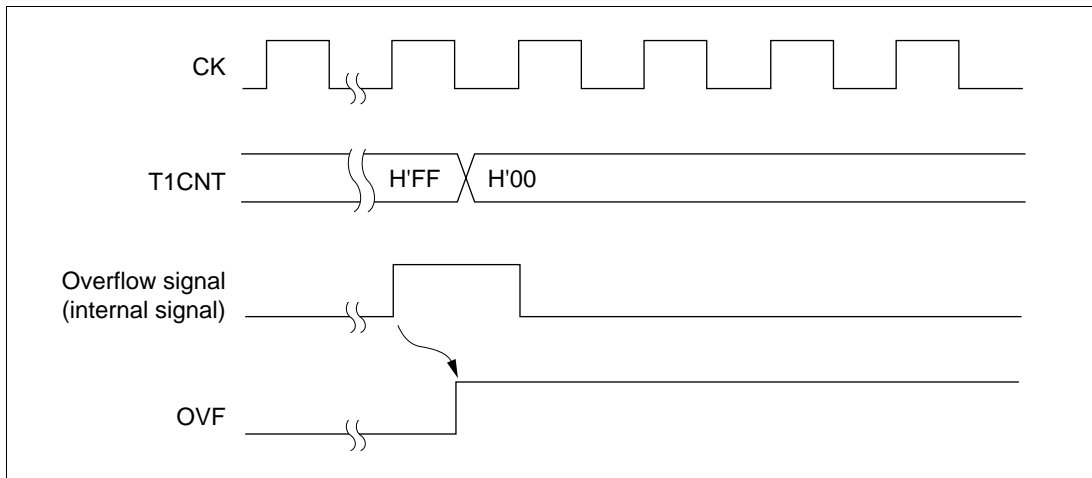
To use the interval timer function, set the TME bit to 1 in the timer 1 control/status register (T1CSR). An interval timer interrupt (ITI) is generated each time the timer 1 counter (T1CNT) overflows, as shown in figure 9.3. This function can be used to generate interrupts at regular intervals.



**Figure 9.3 Interval Timer Operation**

### 9.3.2 Timing of Overflow Flag (OVF) Setting

When the timer 1 counter (T1CNT) overflows, the OVF bit is set to 1 in the timer 1 serial control register (T1CSR) and, at the same time, an interval timer interrupt (ITI) is requested. The timing is shown in figure 9.4.

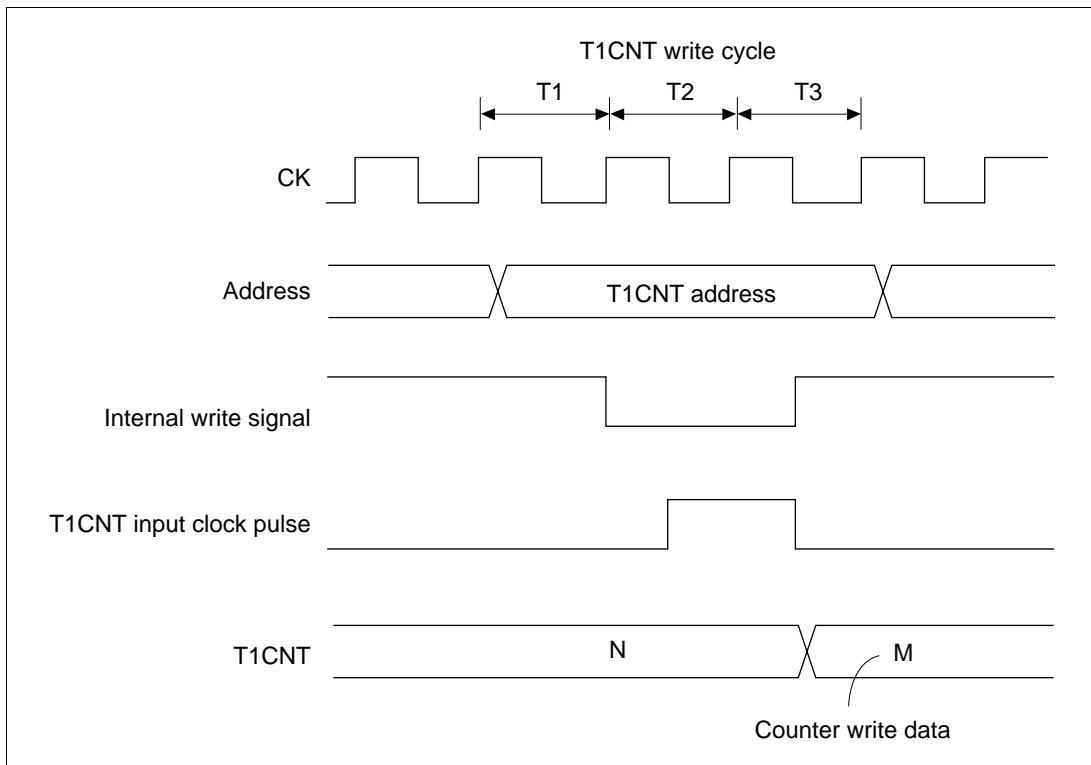


**Figure 9.4 Timing of Overflow Flag (OVF) Setting**

## 9.4 Usage Notes

### 9.4.1 Contention between Timer 1 Counter (TCNT) Write and Increment

If a timer 1 counter clock pulse is generated during the T3 state of a timer 1 counter (TCNT) write cycle, the data write to T1CNT takes priority and the timer counter is not incremented. Figure 9.5 shows the operation in this case.



**Figure 9.5 Contention between T1CNT Write and Increment**

### 9.4.2 Rewriting Bits CKS2 to CKS0

If bits CKS2 to CKS0 in the timer 1 control/status register (T1CSR) are rewritten while 8-bit timer 1 (TIM1) is running, the timer counter may not increment correctly. TIM1 must therefore be stopped (by clearing the TME bit to 0) before rewriting bits CKS2 to CKS0.

# Section 10 8-Bit Timer 2 (TIM2)

## 10.1 Overview

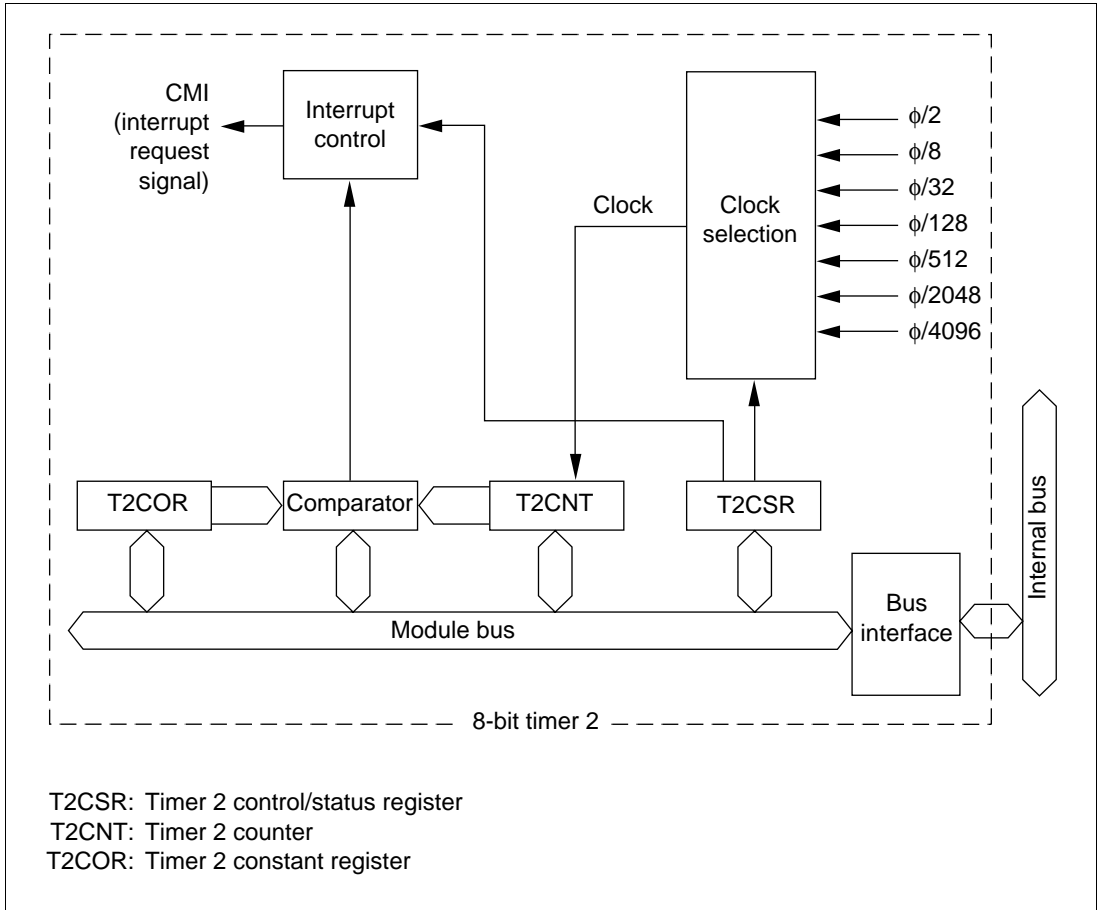
8-bit timer 2 (TIM2) is a single-channel interval timer that generates compare match interrupts.

### 10.1.1 Features

- 8-bit interval timer
- Generates compare match interrupts  
A compare match interrupt is generated by a counter compare match.
- Selection of seven counter input clock sources

## 10.1.2 Block Diagram

Figure 10.1 shows a block diagram of 8-bit timer 2 (TIM2).



**Figure 10.1 Block Diagram of 8-Bit Timer 2**

## 10.1.3 Register Configuration

8-bit timer 2 (TIM2) has three registers for compare match cycle setting, clock selection, and other functions. The register configuration is shown in table 10.1.

All the registers are 16 bits in size, and are initialized by a power-on reset.

**Table 10.1 8-Bit Timer 2 Registers**

Name	Abbreviation	R/W	Initial Value	Address	Access Size
Timer 2 control/status register	T2CSR	R/W	H'0000	H'FFFF862C	8, 16, 32
Timer 2 counter	T2CNT	R/W	H'0000	H'FFFF862E	8, 16, 32
Timer 2 constant register	T2COR	R/W	H'0000	H'FFFF8630	8, 16

## 10.2 Register Descriptions

### 10.2.1 Timer 2 Control/Status Register (T2CSR)

The timer 2 control/status register (T2CSR) is a 16-bit readable/writable\* register that selects the clock to be input to the timer 2 counter (T2CNT) and controls compare match interrupts (CMI).

T2CSR is initialized to H'0000 by a power-on reset.

Bit:	15	14	13	12	11	10	9	8
	—	—	—	—	—	—	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R
Bit:	7	6	5	4	3	2	1	0
	—	CMF	CMIE	CKS2	CKS1	CKS0	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R/W	R/W	R/W	R/W	R/W	R	R

- Bits 15 to 7—Reserved: These bits always read 0 and must only be written with 0.
- Bit 6—Compare Match Flag (CMF): Status flag that indicates a match between the values of T2CNT and T2COR. The setting and clearing conditions for this flag are shown below.

Bit 6: CMF	Description
0	[Clearing condition] Cleared by reading T2CSR when CMF = 1, then writing 0 in CMF (initial value)
1	[Setting condition] Set when T2CNT = T2COR*

Note: When T2CNT and T2COR still contain their initial values (when the initial values have not been changed or when the T2CNT value has not been incremented), CMF is not set even though the T2CNT and T2COR values are the same (H'0000).

- Bit 5—Compare Match Interrupt Enable (CMIE): Enables or disables interrupt requests initiated by the CMF flag when set to 1 in T2CSR.

Bit 5: CMIE	Description
0	Interrupt request by CMF flag disabled (initial value)
1	Interrupt request by CMF flag enabled

- Bits 4 to 2—Clock Select 2 to 0 (CKS2–CKS0): These bits select one of seven internal clock sources, obtained by dividing the system clock ( $\phi$ ), for input to T2CNT.

Bit 4: CKS2	Bit 3: CKS1	Bit 2: CKS0	Description
0	0	0	Up-count stopped (initial value)
		1	$\phi/2$
	1	0	$\phi/8$
		1	$\phi/32$
1	0	0	$\phi/128$
		1	$\phi/512$
	1	0	$\phi/2048$
		1	$\phi/4096$

- Bits 1 and 0—Reserved: These bits always read 0 and must only be written with 0.

### 10.2.2 Timer 2 Counter (T2CNT)

The timer 2 counter (T2CNT) is a 16-bit readable/writable register used as an 8-bit up-counter.

T2CNT increments on the internal clock selected by bits CKS2–CKS0 in T2CSR. The T2CNT value can be read or written by the CPU at all times. When the T2CNT value matches the value in the timer 2 constant register (T2COR), T2CNT is cleared to H'0000 and the CMF flag is set to 1 in T2CSR. If the CMIE bit in T2CSR is set to 1 at this time, a compare match interrupt (CMI) is generated.

Bits 15 to 8 are reserved and have no counter function. These bits always read 0 and must only be written with 0.

T2CNT is initialized to H'0000 by a power-on reset



Bit:	15	14	13	12	11	10	9	8
	—	—	—	—	—	—	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R
Bit:	7	6	5	4	3	2	1	0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

### 10.2.3 Timer 2 Constant Register (T2COR)

The timer 2 constant register (T2COR) is a 16-bit readable/writable register that is used to set the T2CNT compare match cycle. The values in T2COR and T2CNT are continually compared, and when the values match the CMF flag is set in T2CSR and T2CNT is cleared to 0. If the CMIE bit in T2CSR is set to 1, an interrupt request is sent to the interrupt controller in response to the match signal. The interrupt request is output continuously until the CMF flag in T2CSR is cleared.

Bits 15 to 8 are reserved and are not used in the cycle setting. These bits always read 0.

T2COR is initialized to H'0000 by a power-on reset.

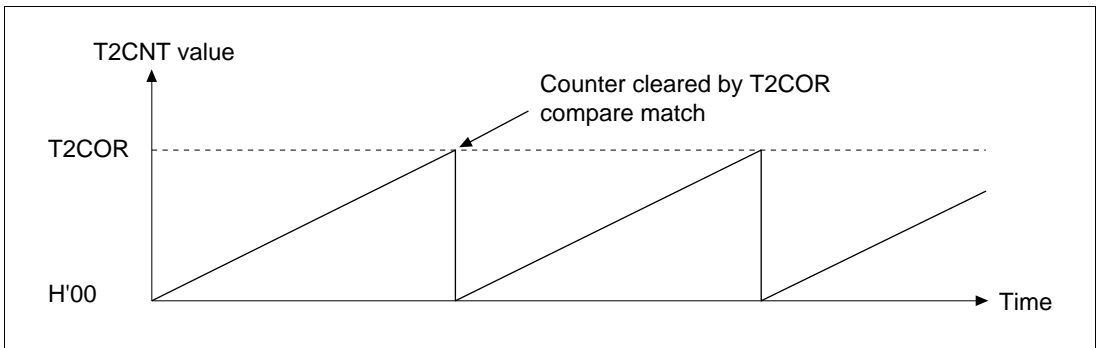
Bit:	15	14	13	12	11	10	9	8
	—	—	—	—	—	—	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R
Bit:	7	6	5	4	3	2	1	0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

## 10.3 Operation

### 10.3.1 Cyclic Count Operation

When a clock is selected with bits CKS2–CKS0 in the T2CSR register, the T2CNT counter starts incrementing on the selected clock. When the T2CNT counter value matches the value in the timer 2 constant register (T2COR), the T2CNT counter is cleared to H'00, and the CMF flag is set to 1 in the T2CSR register. If the CMIE bit in T2CSR is set to 1 at this time, a compare match interrupt (CMI) is requested. The T2CNT counter then starts incrementing again from H'00.

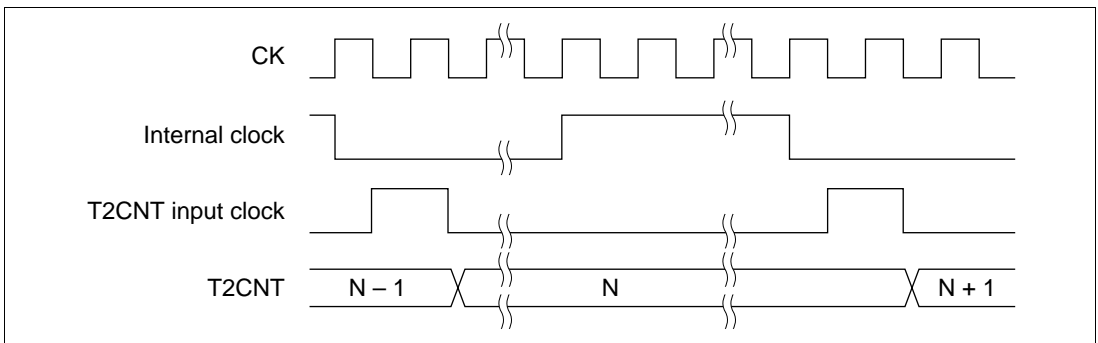
The compare match counter operation is shown in figure 10.2.



**Figure 10.2 Counter Operation**

### 10.3.2 T2CNT Count Timing

Any of seven internal clocks ( $\phi/2$ ,  $\phi/8$ ,  $\phi/32$ ,  $\phi/128$ ,  $\phi/512$ ,  $\phi/2048$ , or  $\phi/4096$ ) divided from the system clock (CK) can be selected with bits CKS2–CKS0 in T2CSR. The count timing is shown in figure 10.3.



**Figure 10.3 Count Timing**

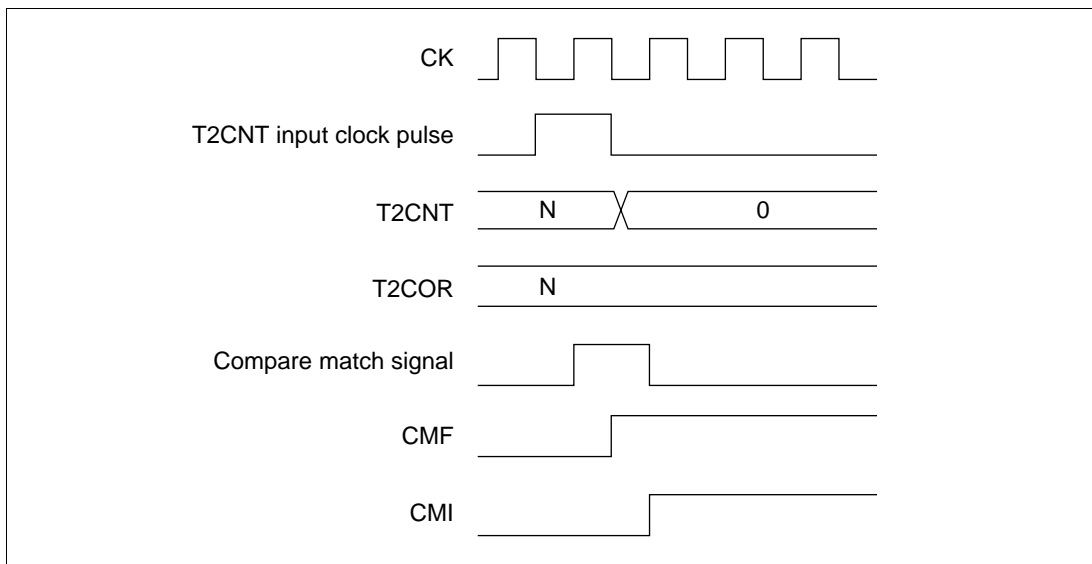
## 10.4 Interrupts

### 10.4.1 Interrupt Source

When interrupt request flag CMF is set to 1, and interrupt enable bit CMIE is also 1, the corresponding interrupt request is output.

### 10.4.2 Timing of Compare Match Flag Setting

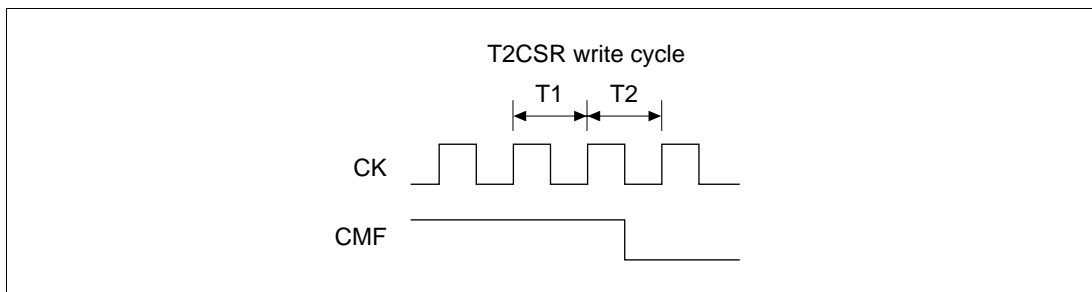
The CMF bit in the T2CSR register is set to 1 by the compare match signal generated when the T2COR register and T2CNT counter values match. The compare match signal is generated in the last state in which the match is true (when the value at which the T2CNT counter match occurred is about to be updated). Therefore, after a match between the T2CNT counter and the T2COR register, the compare match signal is not generated until the next T2CNT counter input clock pulse. Figure 10.4 shows the timing of CMF bit setting.



**Figure 10.4 Timing of CMF Setting**

### 10.4.3 Timing of Compare Match Flag Clearing

The CMF bit in the T2CSR register is cleared by reading the bit when it is set to 1, then writing 0 in it. Figure 10.5 shows the timing of CMF bit clearing by the CPU.



**Figure 10.5** Timing of CMF Clearing by CPU

# Section 11 Compare Match Timer (CMT)

## 11.1 Overview

The SH7011 has an on-chip compare match timer (CMT) configured of 16-bit timers for two channels. The CMT has 16-bit counters and can generate interrupts at set intervals.

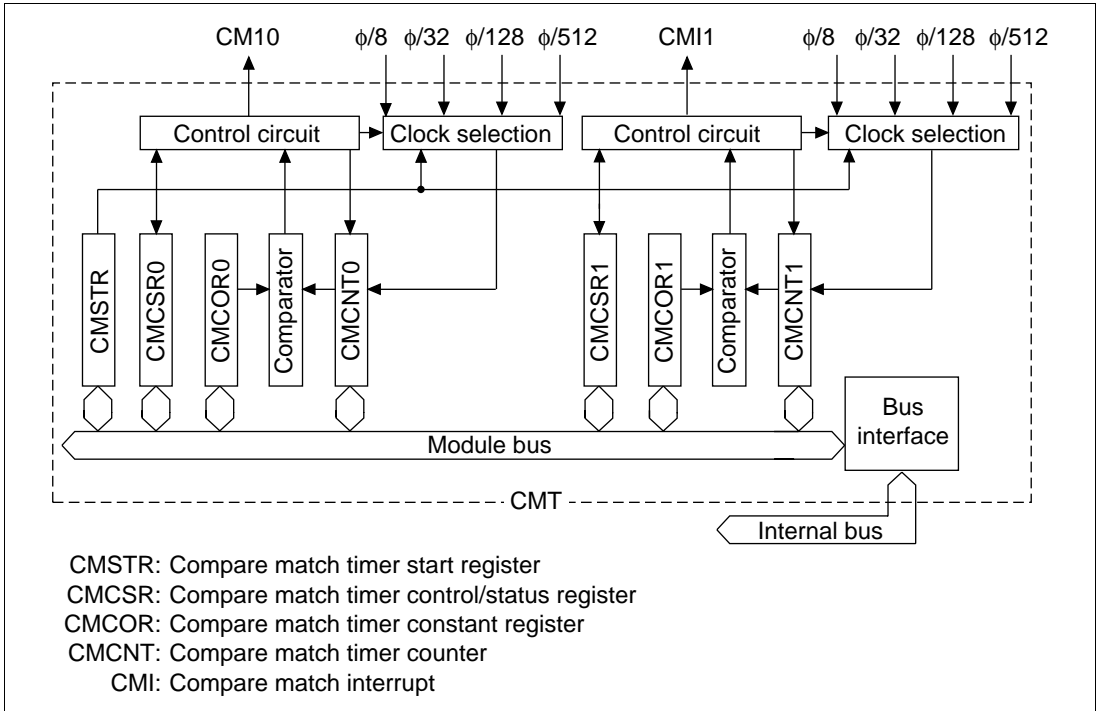
### 11.1.1 Features

The CMT has the following features:

- Four types of counter input clock can be selected
  - One of four internal clocks ( $\phi/8$ ,  $\phi/32$ ,  $\phi/128$ ,  $\phi/512$ ) can be selected independently for each channel.
- Interrupt sources
  - A compare match interrupt can be requested independently for each channel.

## 11.1.2 Block Diagram

Figure 11.1 shows a block diagram of the CMT.



**Figure 11.1 CMT Block Diagram**

### 11.1.3 Register Configuration

Table 11.1 summarizes the CMT register configuration.

**Table 11.1 Register Configuration**

Channel	Name	Abbreviation	R/W	Initial Value	Address	Access Size (Bits)
Shared	Compare match timer start register	CMSTR	R/W	H'0000	H'FFFF83D0	8, 16, 32
0	Compare match timer control/status register 0	CMCSR0	R/(W)*	H'0000	H'FFFF83D2	8, 16, 32
	Compare match timer counter 0	CMCNT0	R/W	H'0000	H'FFFF83D4	8, 16, 32
	Compare match timer constant register 0	CMCOR0	R/W	H'FFFF	H'FFFF83D6	8, 16, 32
1	Compare match timer control/status register 1	CMCSR1	R/(W)*	H'0000	H'FFFF83D8	8, 16, 32
	Compare match timer counter 1	CMCNT1	R/W	H'0000	H'FFFF83DA	8, 16, 32
	Compare match timer constant register 1	CMCOR1	R/W	H'FFFF	H'FFFF83DC	8, 16

Note: The only value that can be written to the CMCSR0 and CMCSR1 CMF bits is a 0 to clear the flags.

## 11.2 Register Descriptions

### 11.2.1 Compare Match Timer Start Register (CMSTR)

The compare match timer start register (CMSTR) is a 16-bit register that selects whether to operate or halt the channel 0 and channel 1 counters (CMCNT). It is initialized to H'0000 by power-on resets.

Bit:	15	14	13	12	11	10	9	8
	—	—	—	—	—	—	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R
Bit:	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	STR1	STR0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R/W	R/W

- Bits 15–2—Reserved: These bits always read as 0. The write value should always be 0.
- Bit 1—Count Start 1 (STR1): Selects whether to operate or halt compare match timer counter 1.

Bit 1: STR1	Description
0	CMCNT1 count operation halted (initial value)
1	CMCNT1 count operation

- Bit 0—Count Start 0 (STR0): Selects whether to operate or halt compare match timer counter 0.

Bit 0: STR0	Description
0	CMCNT0 count operation halted (initial value)
1	CMCNT0 count operation



## 11.2.2 Compare Match Timer Control/Status Register (CMCSR)

The compare match timer control/status register (CMCSR) is a 16-bit register that indicates the occurrence of compare matches, sets the enable/disable of interrupts, and establishes the clock used for incrementation. It is initialized to H'0000 by power-on resets.

Bit:	15	14	13	12	11	10	9	8
	—	—	—	—	—	—	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R
Bit:	7	6	5	4	3	2	1	0
	CMF	CMIE	—	—	—	—	CKS1	CKS0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R/W	R	R	R	R	R/W	R/W

Note: The only value that can be written is a 0 to clear the flag.

- Bits 15–8 and 5–2—Reserved: These bits always read as 0. The write value should always be 0.
- Bit 7—Compare Match Flag (CMF): This flag indicates whether or not the CMCNT and CMCOR values have matched.

Bit 7: CMF	Description
0	CMCNT and CMCOR values have not matched (initial status) Clear condition: Write a 0 to CMF after reading a 1 from it
1	CMCNT and CMCOR values have matched

- Bit 6—Compare Match Interrupt Enable (CMIE): Selects whether to enable or disable a compare match interrupt (CMI) when the CMCNT and CMCOR values have matched (CMF = 1).

Bit 6: CMIE	Description
0	Compare match interrupts (CMI) disabled (initial status)
1	Compare match interrupts (CMI) enabled

- Bits 1, 0—Clock Select 1, 0 (CKS1, CKS0): These bits select the clock input to the CMCNT from among the four internal clocks obtained by dividing the system clock ( $\phi$ ). When the STR bit of the CMSTR is set to 1, the CMCNT begins incrementing with the clock selected by CKS1 and CKS0.

Bit 1: CKS1	Bit 0: CKS0	Description
0	0	$\phi/8$ (initial status)
	1	$\phi/32$
1	0	$\phi/128$
	1	$\phi/512$

### 11.2.3 Compare Match Timer Counter (CMCNT)

The compare match timer counter (CMCNT) is a 16-bit register used as an upcounter for generating interrupt requests.

When an internal clock is selected with the CKS1, CKS0 bits of the CMCSR register and the STR bit of the CMSTR is set to 1, the CMCNT begins incrementing with that clock. When the CMCNT value matches that of the compare match timer constant register (CMCOR), the CMCNT is cleared to H'0000 and the CMF flag of the CMCSR is set to 1. If the CMIE bit of the CMCSR is set to 1 at this time, a compare match interrupt (CMI) is requested.

The CMCNT is initialized to H'0000 by power-on resets.

Bit:	15	14	13	12	11	10	9	8
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit:	7	6	5	4	3	2	1	0
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

## 11.2.4 Compare Match Timer Constant Register (CMCOR)

The compare match timer constant register (CMCOR) is a 16-bit register that sets the compare match period with the CMCNT.

The CMCOR is initialized to H'FFFF by power-on resets.

Bit:	15	14	13	12	11	10	9	8
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Initial value:	1	1	1	1	1	1	1	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit:	7	6	5	4	3	2	1	0
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Initial value:	1	1	1	1	1	1	1	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

## 11.3 Operation

### 11.3.1 Period Count Operation

When an internal clock is selected with the CKS1, CKS0 bits of the CMCSR register and the STR bit of the CMSTR is set to 1, the CMCNT begins incrementing with the selected clock. When the CMCNT counter value matches that of the compare match constant register (CMCOR), the CMCNT counter is cleared to H'0000 and the CMF flag of the CMCSR register is set to 1. If the CMIE bit of the CMCSR register is set to 1 at this time, a compare match interrupt (CMI) is requested. The CMCNT counter begins counting up again from H'0000.

Figure 11.2 shows the compare match counter operation.

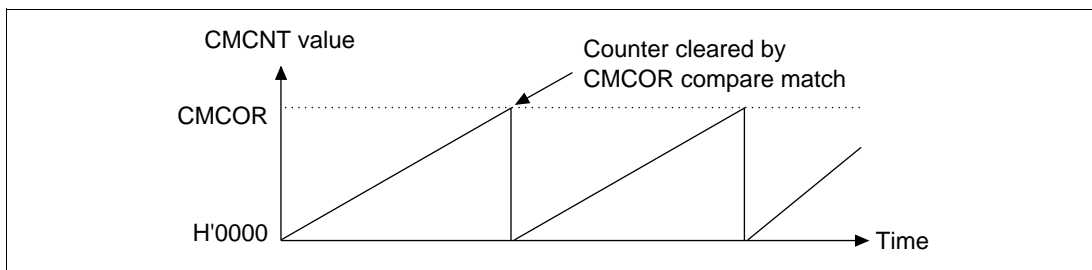


Figure 11.2 Counter Operation

### 11.3.2 CMCNT Count Timing

One of four clocks ( $\phi/8$ ,  $\phi/32$ ,  $\phi/128$ ,  $\phi/512$ ) obtained by dividing the system clock (CK) can be selected by the CKS1, CKS0 bits of the CMCSR. Figure 11.3 shows the timing.

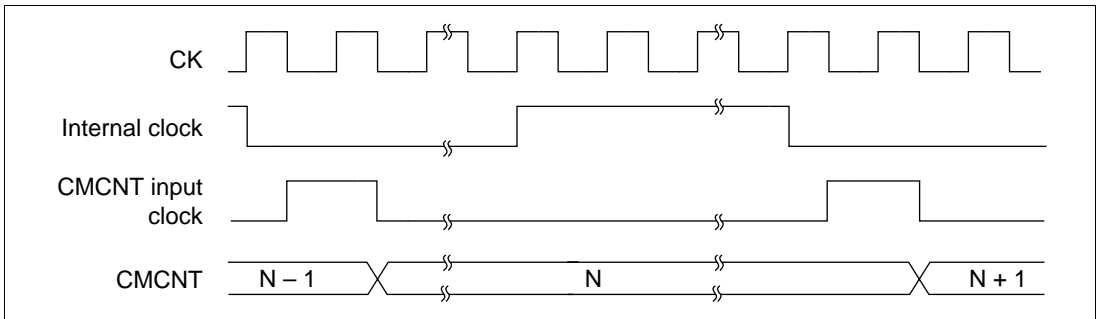


Figure 11.3 Count Timing

## 11.4 Interrupts

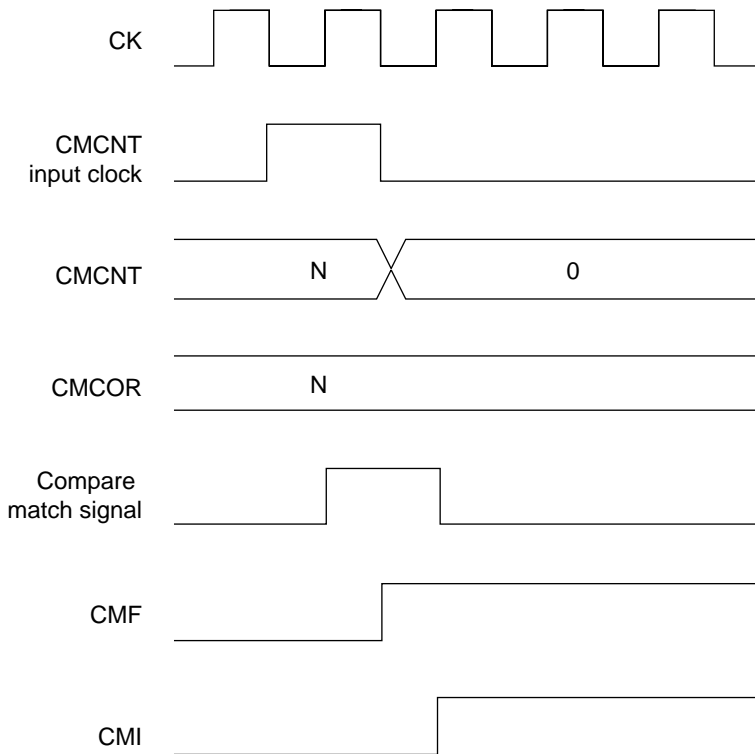
### 11.4.1 Interrupt Sources

The CMT has a compare match interrupt for each channel, with independent vector addresses allocated to each of them. The corresponding interrupt request is output when the interrupt request flag CMF is set to 1 and the interrupt enable bit CMIE has also been set to 1.

When activating CPU interrupts by interrupt request, the priority between the channels can be changed by using the interrupt controller settings. See section 6, Interrupt Controller, for details.

### 11.4.2 Compare Match Flag Set Timing

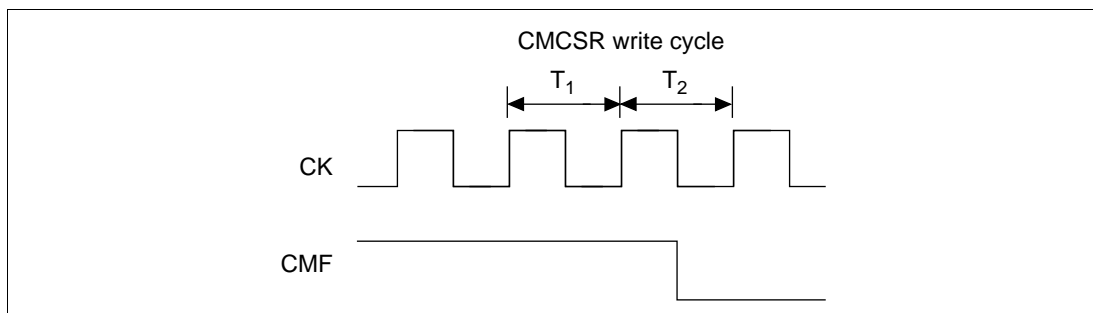
The CMF bit of the CMCSR register is set to 1 by the compare match signal generated when the CMCOR register and the CMCNT counter match. The compare match signal is generated upon the final state of the match (timing at which the CMCNT counter matching count value is updated). Consequently, after the CMCOR register and the CMCNT counter match, a compare match signal will not be generated until a CMCNT counter input clock occurs. Figure 11.4 shows the CMF bit set timing.



**Figure 11.4 CMF Set Timing**

### 11.4.3 Compare Match Flag Clear Timing

The CMF bit of the CMCSR register is cleared either by writing a 0 to it after reading a 1. Figure 11.5 shows the timing when the CMF bit is cleared by the CPU.



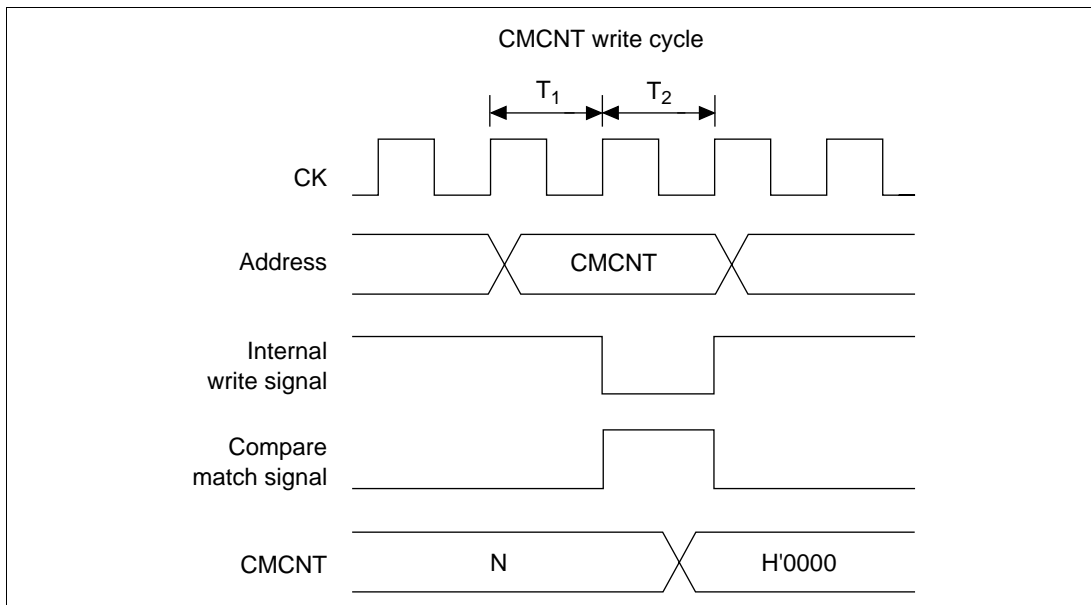
**Figure 11.5 Timing of CMF Clear by the CPU**

## 11.5 Notes on Use

Take care that the contentions described in sections 11.5.1–11.5.3 do not arise during CMT operation.

### 11.5.1 Contention between CMCNT Write and Compare Match

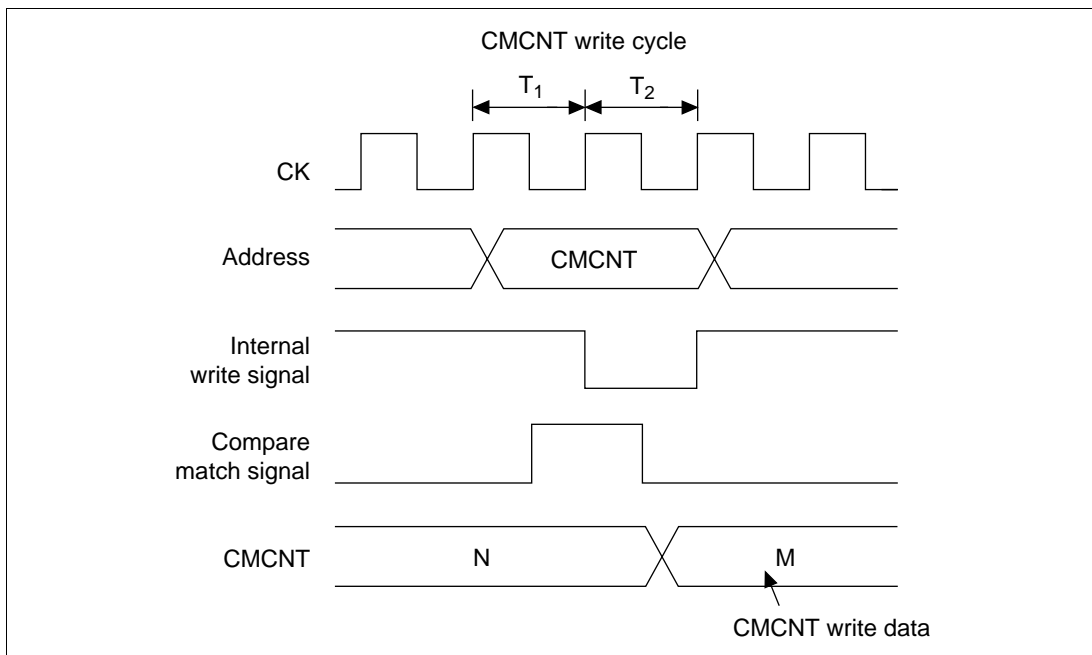
If a compare match signal is generated during the  $T_2$  state of the CMCNT counter write cycle, the CMCNT counter clear has priority, so the write to the CMCNT counter is not performed. Figure 11.6 shows the timing.



**Figure 11.6 CMCNT Write and Compare Match Contention**

## 11.5.2 Contention between CMCNT Word Write and Incrementation

If an increment occurs during the  $T_2$  state of the CMCNT counter word write cycle, the counter write has priority, so no increment occurs. Figure 11.7 shows the timing.

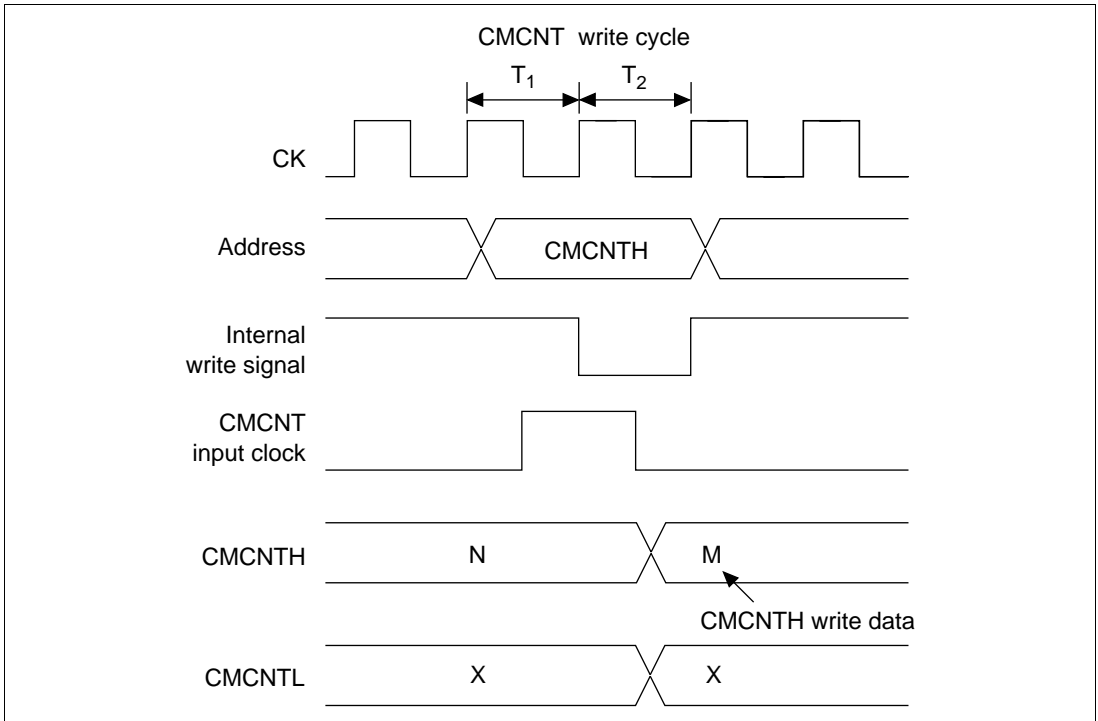


**Figure 11.7 CMCNT Word Write and Increment Contention**

### 11.5.3 Contention between CMCNT Byte Write and Incrementation

If an increment occurs during the  $T_2$  state of the CMCNT byte write cycle, the counter write has priority, so no increment of the write data results on the writing side. The byte data on the side not performing the writing is also not incremented, so the contents are those before the write.

Figure 11.8 shows the timing when an increment occurs during the  $T_2$  state of the CMCNTH write cycle.



**Figure 11.8 CMCNT Byte Write and Increment Contention**



# Section 12 Serial Communication Interface (SCI)

## 12.1 Overview

The SH7011 has a serial communication interface (SCI) with one channel.

The SCI supports asynchronous serial communication. It also has a multiprocessor communication function for serial communication among two or more processors.

### 12.1.1 Features

- Select asynchronous or clock synchronous as the serial communications mode.
  - Asynchronous mode: Serial data communications are synched by start-stop in character units. The SCI can communicate with a universal asynchronous receiver/transmitter (UART), an asynchronous communication interface adapter (ACIA), or any other chip that employs a standard asynchronous serial communication. It can also communicate with two or more other processors using the multiprocessor communication function. There are twelve selectable serial data communication formats.
  - Data length: seven or eight bits
  - Stop bit length: one or two bits
  - Parity: even, odd, or none
  - Multiprocessor bit: one or none
  - Receive error detection: parity, overrun, and framing errors
  - Break detection: by reading the RxD level directly when a framing error occurs
- Full duplex communication: The transmitting and receiving sections are independent, so the SCI can transmit and receive simultaneously. Both sections use double buffering, so continuous data transfer is possible in both the transmit and receive directions.
- On-chip baud rate generator with selectable bit rates.
- Internal transmit/receive clock source: baud rate generator (internal).
- Four types of interrupts: Transmit-data-empty, transmit-end, receive-data-full, and receive-error interrupts are requested independently.

## 12.1.2 Block Diagram

Figure 12.1 shows a block diagram of the SCI.

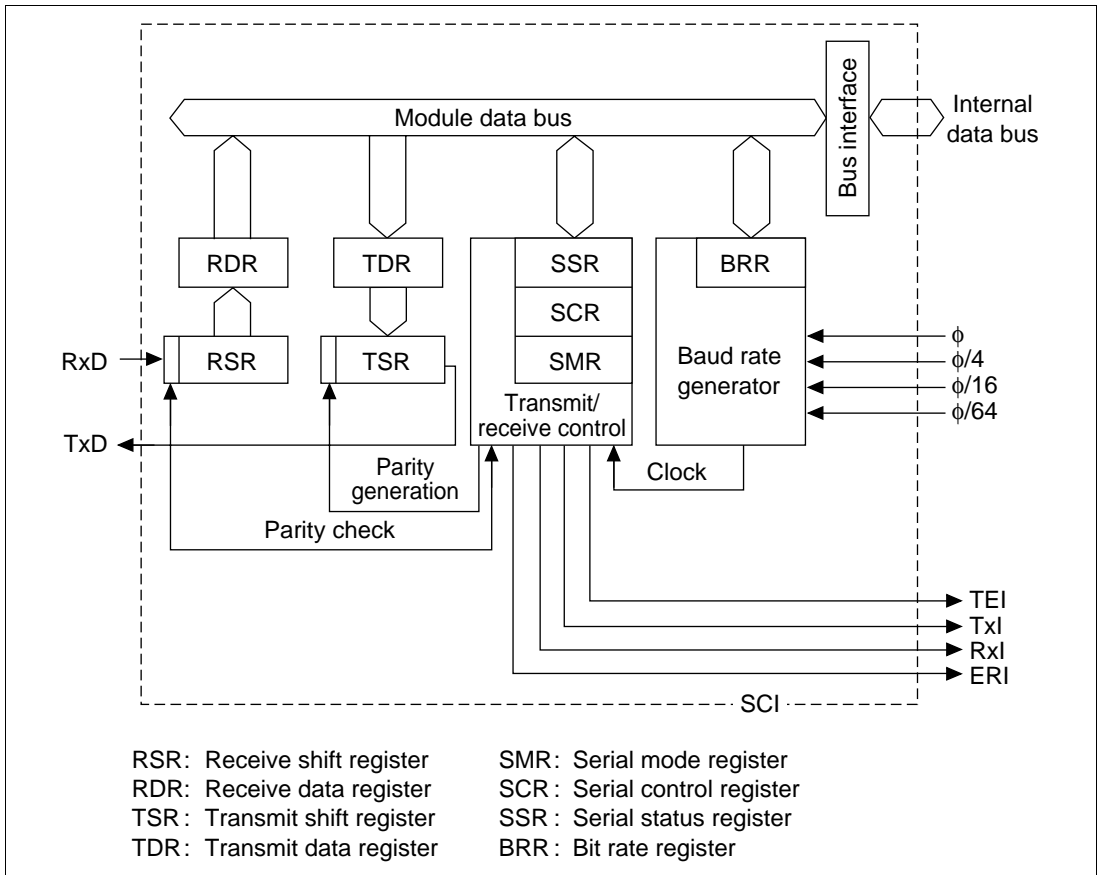


Figure 12.1 SCI Block Diagram

## 12.1.3 Pin Configuration

Table 12.1 summarizes the SCI pins by channel.

Table 12.1 SCI Pins

Pin Name	Abbreviation	Input/Output	Function
Receive data pin	RxD	Input	SCI receive data input
Transmit data pin	TxD	Output	SCI transmit data output

## 12.1.4 Register Configuration

Table 12.2 summarizes the SCI internal registers. These registers specify the data format and bit rate, and control the transmitter and receiver sections.

**Table 12.2 Registers**

Name	Abbreviation	R/W	Initial Value	Address	Access Size
Serial mode register	SMR	R/W	H'00	H'FFFF81B0	8, 16
Bit rate register	BRR	R/W	H'FF	H'FFFF81B1	8, 16
Serial control register	SCR	R/W	H'00	H'FFFF81B2	8, 16
Transmit data register	TDR	R/W	H'FF	H'FFFF81B3	8, 16
Serial status register	SSR	R/(W)*	H'84	H'FFFF81B4	8, 16
Receive data register	RDR	R	H'00	H'FFFF81B5	8, 16

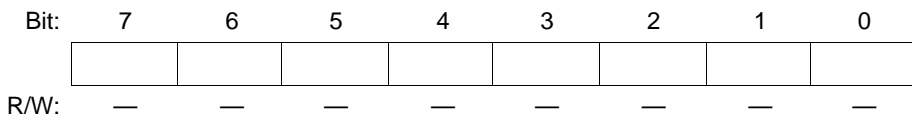
Note: The only value that can be written is a 0 to clear the flags..

## 12.2 Register Descriptions

### 12.2.1 Receive Shift Register (RSR)

The receive shift register (RSR) receives serial data. Data input at the Rx D pin is loaded into the RSR in the order received, LSB (bit 0) first, converting the data to parallel form. When one byte has been received, it is automatically transferred to the RDR.

The CPU cannot read or write the RSR directly.



## 12.2.2 Receive Data Register (RDR)

The receive data register (RDR) stores serial receive data. The SCI completes the reception of one byte of serial data by moving the received data from the receive shift register (RSR) into the RDR for storage. The RSR is then ready to receive the next data. This double buffering allows the SCI to receive data continuously.

The CPU can read but not write the RDR. The RDR is initialized to H'00 by a power-on reset.

Bit:	7	6	5	4	3	2	1	0
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R

## 12.2.3 Transmit Shift Register (TSR)

The transmit shift register (TSR) transmits serial data. The SCI loads transmit data from the transmit data register (TDR) into the TSR, then transmits the data serially from the TxD pin, LSB (bit 0) first. After transmitting one data byte, the SCI automatically loads the next transmit data from the TDR into the TSR and starts transmitting again. If the TDRE bit of the SSR is 1, however, the SCI does not load the TDR contents into the TSR.

The CPU cannot read or write the TSR directly.

Bit:	7	6	5	4	3	2	1	0
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
R/W:	—	—	—	—	—	—	—	—

## 12.2.4 Transmit Data Register (TDR)

The transmit data register (TDR) is an 8-bit register that stores data for serial transmission. When the SCI detects that the transmit shift register (TSR) is empty, it moves transmit data written in the TDR into the TSR and starts serial transmission. Continuous serial transmission is possible by writing the next transmit data in the TDR during serial transmission from the TSR.

The CPU can always read and write the TDR. The TDR is initialized to H'FF by a power-on reset.

Bit:	7	6	5	4	3	2	1	0
Initial value:	1	1	1	1	1	1	1	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

### 12.2.5 Serial Mode Register (SMR)

The serial mode register (SMR) is an 8-bit register that specifies the SCI serial communication format and selects the clock source for the baud rate generator.

The CPU can always read and write the SMR. The SMR is initialized to H'00 by a power-on reset.

Bit:	7	6	5	4	3	2	1	0
	—	CHR	PE	O/ $\bar{E}$	STOP	MP	CKS1	CKS0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- Bit 7—Reserved. This bit always reads 0. The write value should always be 0.
- Bit 6—Character Length (CHR): Selects 7-bit or 8-bit data in the asynchronous mode.

Bit 6: CHR	Description
0	Eight-bit data (initial value)
1	Seven-bit data. (When 7-bit data is selected, the MSB (bit 7) of the transmit data register is not transmitted.)

- Bit 5—Parity Enable (PE): Selects whether to add a parity bit to transmit data and to check the parity of receive data.

Bit 5: PE	Description
0	Parity bit not added or checked (initial value)
1	Parity bit added and checked. When PE is set to 1, an even or odd parity bit is added to transmit data, depending on the parity mode (O/ $\bar{E}$ ) setting. Receive data parity is checked according to the even/odd (O/ $\bar{E}$ ) mode setting.

- Bit 4—Parity Mode ( $O/\bar{E}$ ): Selects even or odd parity when parity bits are added and checked. The  $O/\bar{E}$  setting is used only when the parity enable bit (PE) is set to 1 to enable parity addition and check. The  $O/\bar{E}$  setting is ignored when parity addition and check is disabled.

Bit 4: $O/\bar{E}$	Description
0	Even parity (initial value). If even parity is selected, the parity bit is added to transmit data to make an even number of 1s in the transmitted character and parity bit combined. Receive data is checked to see if it has an even number of 1s in the received character and parity bit combined.
1	Odd parity. If odd parity is selected, the parity bit is added to transmit data to make an odd number of 1s in the transmitted character and parity bit combined. Receive data is checked to see if it has an odd number of 1s in the received character and parity bit combined.

- Bit 3—Stop Bit Length (STOP): Selects one or two bits as the stop bit length. In receiving, only the first stop bit is checked, regardless of the STOP bit setting. If the second stop bit is 1, it is treated as a stop bit, but if the second stop bit is 0, it is treated as the start bit of the next incoming character.

Bit 3: STOP	Description
0	One stop bit (initial value). In transmitting, a single bit of 1 is added at the end of each transmitted character.
1	Two stop bits. In transmitting, two bits of 1 are added at the end of each transmitted character.

- Bit 2—Multiprocessor Mode (MP): Selects multiprocessor format. When multiprocessor format is selected, settings of the parity enable (PE) and parity mode ( $O/\bar{E}$ ) bits are ignored. For the multiprocessor communication function, see section 12.3.3, Multiprocessor Communication.

Bit 2: MP	Description
0	Multiprocessor function disabled (initial value)
1	Multiprocessor format selected

- Bits 1 and 0—Clock Select 1 and 0 (CKS1 and CKS0): These bits select the internal clock source of the on-chip baud rate generator. Four clock sources are available;  $\phi$ ,  $\phi/4$ ,  $\phi/16$ , or  $\phi/64$ . For further information on the clock source, bit rate register settings, and baud rate, see section 12.2.8, Bit Rate Register.

Bit 1: CKS1	Bit 0: CKS0	Description
0	0	$\phi$ (initial value)
	1	$\phi/4$
1	0	$\phi/16$
	1	$\phi/64$

## 12.2.6 Serial Control Register (SCR)

The serial control register (SCR) operates the SCI transmitter/receiver, enables/disables interrupt requests. The CPU can always read and write the SCR. The SCR is initialized to H'00 by a power-on reset.

Bit:	7	6	5	4	3	2	1	0
	TIE	RIE	TE	RE	MPIE	TEIE	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R	R

- Bit 7—Transmit Interrupt Enable (TIE): Enables or disables the transmit-data-empty interrupt (TxI) requested when the transmit data register empty bit (TDRE) in the serial status register (SSR) is set to 1 by transfer of serial transmit data from the TDR to the TSR.

Bit 7: TIE	Description
0	Transmit-data-empty interrupt request (TxI) is disabled (initial value). The TxI interrupt request can be cleared by reading TDRE after it has been set to 1, then clearing TDRE to 0, or by clearing TIE to 0.
1	Transmit-data-empty interrupt request (TxI) is enabled

- Bit 6—Receive Interrupt Enable (RIE): Enables or disables the receive-data-full interrupt (RxI) requested when the receive data register full bit (RDRF) in the serial status register (SSR) is set to 1 by transfer of serial receive data from the RSR to the RDR. It also enables or disables receive-error interrupt (ERI) requests.

Bit 6: RIE	Description
0	Receive-data-full interrupt (RxI) and receive-error interrupt (ERI) requests are disabled (initial value). RxI and ERI interrupt requests can be cleared by reading the RDRF flag or error flag (FER, PER, or ORER) after it has been set to 1, then clearing the flag to 0, or by clearing RIE to 0.
1	Receive-data-full interrupt (RxI) and receive-error interrupt (ERI) requests are enabled.

- Bit 5—Transmit Enable (TE): Enables or disables the SCI serial transmitter.

Bit 5: TE	Description
0	Transmitter disabled (initial value). The transmit data register empty bit (TDRE) in the serial status register (SSR) is locked at 1.
1	Transmitter enabled. Serial transmission starts when the transmit data register empty (TDRE) bit in the serial status register (SSR) is cleared to 0 after writing of transmit data into the TDR. Select the transmit format in the SMR before setting TE to 1.

- Bit 4—Receive Enable (RE): Enables or disables the SCI serial receiver.

Bit 4: RE	Description
0	Receiver disabled (initial value). Clearing RE to 0 does not affect the receive flags (RDRF, FER, PER, ORER). These flags retain their previous values.
1	Receiver enabled. Serial reception starts when a start bit is detected in the asynchronous mode, or synchronous clock input is detected in the clock synchronous mode. Select the receive format in the SMR before setting RE to 1.

- Bit 3—Multiprocessor Interrupt Enable (MPIE): Enables or disables multiprocessor interrupts. The MPIE setting is used only if the multiprocessor mode bit (MP) in the serial mode register (SMR) is set to 1 during reception.

Bit 3: MPIE	Description
0	Multiprocessor interrupts are disabled (normal receive operation) (initial value). MPIE is cleared when the MPIE bit is cleared to 0, or the multiprocessor bit (MPB) is set to 1 in receive data.
1	Multiprocessor interrupts are enabled. Receive-data-full interrupt requests (Rxl), receive-error interrupt requests (ERI), and setting of the RDRF, FER, and ORER status flags in the serial status register (SSR) are disabled until data with the multiprocessor bit set to 1 is received.  The SCI does not transfer receive data from the RSR to the RDR, does not detect receive errors, and does not set the RDRF, FER, and ORER flags in the serial status register (SSR). When it receives data that includes MPB = 1, MPB is set to 1, and the SCI automatically clears MPIE to 0, generates Rxl and ERI interrupts (if the TIE and RIE bits in the SCR are set to 1), and allows the FER and ORER bits to be set.



- Bit 2—Transmit-End Interrupt Enable (TEIE): Enables or disables the transmit-end interrupt (TEI) requested if TDR does not contain valid transmit data when the MSB is transmitted.

Bit 2: TEIE	Description
0	Transmit-end interrupt (TEI) requests are disabled* (initial value)
1	Transmit-end interrupt (TEI) requests are enabled.*

Note: The TEI request can be cleared by reading the TDRE bit in the serial status register (SSR) after it has been set to 1, then clearing TDRE to 0 and clearing the transmit end (TEND) bit to 0; or by clearing the TEIE bit to 0.

- Bits 1 and 0—Reserved. These bits always read 0. The write value should always be 0.

### 12.2.7 Serial Status Register (SSR)

The serial status register (SSR) is an 8-bit register containing multiprocessor bit values, and status flags that indicate SCI operating status.

The CPU can always read and write the SSR, but cannot write 1 in the status flags (TDRE, RDRF, ORER, PER, and FER). These flags can be cleared to 0 only if they have first been read (after being set to 1). Bits 2 (TEND) and 1 (MPB) are read-only bits that cannot be written. The SSR is initialized to H'84 by a power-on reset.

Bit:	7	6	5	4	3	2	1	0
	TDRE	RDRF	ORER	FER	PER	TEND	MPB	MPBT
Initial value:	1	0	0	0	0	1	0	0
R/W:	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R	R	R/W

Note: The only value that can be written is a 0 to clear the flag.

- Bit 7—Transmit Data Register Empty (TDRE): Indicates that the SCI has loaded transmit data from the TDR into the TSR and new serial transmit data can be written in the TDR.

Bit 7: TDRE	Description
0	TDR contains valid transmit data TDRE is cleared to 0 when software reads TDRE after it has been set to 1, then writes 0 in TDRE
1	TDR does not contain valid transmit data (initial value) TDRE is set to 1 when the chip is power-on reset the TE bit in the serial control register (SCR) is cleared to 0, or TDR contents are loaded into TSR, so new data can be written in TDR

- Bit 6—Receive Data Register Full (RDRF): Indicates that RDR contains received data.

Bit 6: RDRF	Description
0	RDR does not contain valid received data (initial value) RDRF is cleared to 0 when the chip is power-on reset, software reads RDRF after it has been set to 1, then writes 0 in RDRF
1	RDR contains valid received data RDRF is set to 1 when serial data is received normally and transferred from RSR to RDR

Note: The RDR and RDRF are not affected by detection of receive errors or by clearing of the RE bit to 0 in the serial control register. They retain their previous contents. If RDRF is still set to 1 when reception of the next data ends, an overrun error (ORER) occurs and the received data is lost.

- Bit 5—Overrun Error (ORER): Indicates that data reception ended abnormally due to an overrun error.

Bit 5: ORER	Description
0	Receiving is in progress or has ended normally (initial value). Clearing the RE bit to 0 in the serial control register does not affect the ORER bit, which retains its previous value. ORER is cleared to 0 when the chip is power-on reset or software reads ORER after it has been set to 1, then writes 0 in ORER
1	A receive overrun error occurred. RDR continues to hold the data received before the overrun error, so subsequent receive data is lost. Serial receiving cannot continue while ORER is set to 1. ORER is set to 1 if reception of the next serial data ends when RDRF is set to 1

- Bit 4—Framing Error (FER): Indicates that data reception ended abnormally due to a framing error.

Bit 4: FER	Description
0	<p>Receiving is in progress or has ended normally (initial value). Clearing the RE bit to 0 in the serial control register does not affect the FER bit, which retains its previous value.</p> <p>FER is cleared to 0 when the chip is power-on reset or software reads FER after it has been set to 1, then writes 0 in FER</p>
1	<p>A receive framing error occurred. When the stop bit length is two bits, only the first bit is checked to see if it is a 1. The second stop bit is not checked. When a framing error occurs, the SCI transfers the receive data into the RDR but does not set RDRF. Serial receiving cannot continue while FER is set to 1.</p> <p>FER is set to 1 if the stop bit at the end of receive data is checked and found to be 0</p>

- Bit 3—Parity Error (PER): Indicates that data reception (with parity) ended abnormally due to a parity error.

Bit 3: PER	Description
0	<p>Receiving is in progress or has ended normally (initial value). Clearing the RE bit to 0 in the serial control register does not affect the PER bit, which retains its previous value.</p> <p>PER is cleared to 0 when the chip is power-on reset or software reads PER after it has been set to 1, then writes 0 in PER</p>
1	<p>A receive parity error occurred. When a parity error occurs, the SCI transfers the receive data into the RDR but does not set RDRF. Serial receiving cannot continue while PER is set to 1.</p> <p>PER is set to 1 if the number of 1s in receive data, including the parity bit, does not match the even or odd parity setting of the parity mode bit (O/<math>\bar{E}</math>) in the serial mode register (SMR)</p>

- **Bit 2—Transmit End (TEND):** Indicates that when the last bit of a serial character was transmitted, the TDR did not contain valid data, so transmission has ended. TEND is a read-only bit and cannot be written.

<b>Bit 2: TEND</b>	<b>Description</b>
0	Transmission is in progress TEND is cleared to 0 when software reads TDRE after it has been set to 1, then writes 0 in TDRE
1	End of transmission (initial value) TEND is set to 1 when the chip is power-on reset, TE is cleared to 0 in the serial control register (SCR), or TDRE is 1 when the last bit of a one-byte serial character is transmitted.

- **Bit 1—Multiprocessor Bit (MPB):** Stores the value of the multiprocessor bit in receive data when a multiprocessor format is selected for receiving. The MPB is a read-only bit and cannot be written.

<b>Bit 1: MPB</b>	<b>Description</b>
0	Multiprocessor bit value in receive data is 0 (initial value). If RE is cleared to 0 when a multiprocessor format is selected, the MPB retains its previous value.
1	Multiprocessor bit value in receive data is 1

- **Bit 0—Multiprocessor Bit Transfer (MPBT):** Stores the value of the multiprocessor bit added to transmit data when a multiprocessor format is selected for transmitting.

<b>Bit 0: MPBT</b>	<b>Description</b>
0	Multiprocessor bit value in transmit data is 0 (initial value)
1	Multiprocessor bit value in transmit data is 1

## 12.2.8 Bit Rate Register (BRR)

The bit rate register (BRR) is an 8-bit register that, together with the baud rate generator clock source selected by the CKS1 and CKS0 bits in the serial mode register (SMR), determines the serial transmit/receive bit rate.

The CPU can always read and write the BRR. The BRR is initialized to H'FF by a power-on reset.

Bit:	7	6	5	4	3	2	1	0
Initial value:	1	1	1	1	1	1	1	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 12.3 lists examples of BRR settings in the asynchronous mode.

**Table 12.3 Bit Rates and BRR Settings**

Bit Rate (Bits/s)	$\phi$ (MHz)								
	4			4.9152			6		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	2	70	0.03	2	86	0.31	2	106	-0.44
150	1	207	0.16	1	255	0.00	2	77	0.16
300	1	103	0.16	1	127	0.00	1	155	0.16
600	0	207	0.16	0	255	0.00	1	77	0.16
1200	0	103	0.16	0	127	0.00	0	155	0.16
2400	0	51	0.16	0	63	0.00	0	77	0.16
4800	0	25	0.16	0	31	0.00	0	38	0.16
9600	0	12	0.16	0	15	0.00	0	19	-2.34
14400	0	8	-3.55	0	10	-3.03	0	12	0.16
19200	0	6	-6.99	0	7	0.00	0	9	-2.34
28800	0	3	8.51	0	4	6.67	0	6	-6.99
31250	0	3	0.00	0	4	-1.70	0	5	0.00
38400	0	2	8.51	0	3	0.00	0	4	-2.34

**Table 12.3 Bit Rates and BRR Settings (cont)**

Bit Rate (Bits/s)	$\phi$ (MHz)								
	7.3728			8			9.8304		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	2	130	-0.07	2	141	0.03	2	174	-0.26
150	2	95	0.00	2	103	0.16	2	127	0.00
300	1	191	0.00	1	207	0.16	1	255	0.00
600	1	95	0.00	1	103	0.16	1	127	0.00
1200	0	191	0.00	0	207	0.16	0	255	0.00
2400	0	95	0.00	0	103	0.16	0	127	0.00
4800	0	47	0.00	0	51	0.16	0	63	0.00
9600	0	23	0.00	0	25	0.16	0	31	0.00
14400	0	15	0.00	0	16	2.12	0	20	1.59
19200	0	11	0.00	0	12	0.16	0	15	0.00
28800	0	7	0.00	0	8	-3.55	0	10	-3.03
31250	0	6	5.33	0	7	0.00	0	9	-1.70
38400	0	5	0.00	0	6	-6.99	0	7	0.00

**Table 12.3 Bit Rates and BRR Settings (cont)**

Bit Rate (Bits/s)	$\phi$ (MHz)								
	10			11.0592			12		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	2	177	-0.25	2	195	0.19	2	212	0.03
150	2	129	0.16	2	143	0.00	2	155	0.16
300	2	64	0.16	2	71	0.00	2	77	0.16
600	1	129	0.16	1	143	0.00	1	155	0.16
1200	1	64	0.16	1	71	0.00	1	77	0.16
2400	0	129	0.16	0	143	0.00	0	155	0.16
4800	0	64	0.16	0	71	0.00	0	77	0.16
9600	0	32	-1.36	0	35	0.00	0	38	0.16
14400	0	21	-1.36	0	23	0.00	0	25	0.16
19200	0	15	1.73	0	17	0.00	0	19	-2.34
28800	0	10	-1.36	0	11	0.00	0	12	0.16
31250	0	9	0.00	0	10	0.54	0	11	0.00
38400	0	7	1.73	0	8	0.00	0	9	-2.34

**Table 12.3 Bit Rates and BRR Settings (cont)**

Bit Rate (Bits/s)	$\phi$ (MHz)								
	12.288			14			14.7456		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	2	217	0.08	2	248	-0.17	3	64	0.70
150	2	159	0.00	2	181	0.16	2	191	0.00
300	2	79	0.00	2	90	0.16	2	95	0.00
600	1	159	0.00	1	181	0.16	1	191	0.00
1200	1	79	0.00	1	90	0.16	1	95	0.00
2400	0	159	0.00	0	181	0.16	0	191	0.00
4800	0	79	0.00	0	90	0.16	0	95	0.00
9600	0	39	0.00	0	45	-0.93	0	47	0.00
14400	0	26	-1.23	0	29	1.27	0	31	0.00
19200	0	19	0.00	0	22	-0.93	0	23	0.00
28800	0	12	2.56	0	14	1.27	0	15	0.00
31250	0	11	2.40	0	13	0.00	0	14	-1.70
38400	0	9	0.00	0	10	3.57	0	11	0.00



**Table 12.3 Bit Rates and BRR Settings (cont)**

Bit Rate (Bits/s)	$\phi$ (MHz)								
	16			17.2032			18		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	3	70	0.03	3	75	0.48	3	79	-0.12
150	2	207	0.16	2	223	0.00	2	233	0.16
300	2	103	0.16	2	111	0.00	2	116	0.16
600	1	207	0.16	1	223	0.00	1	233	0.16
1200	1	103	0.16	1	111	0.00	1	116	0.16
2400	0	207	0.16	0	223	0.00	0	233	0.16
4800	0	103	0.16	0	111	0.00	0	116	0.16
9600	0	51	0.16	0	55	0.00	0	58	-0.69
14400	0	34	-0.79	0	36	0.90	0	38	0.16
19200	0	25	0.16	0	27	0.00	0	28	1.02
28800	0	16	2.12	0	18	-1.75	0	19	-2.34
31250	0	15	0.00	0	16	1.20	0	17	0.00
38400	0	12	0.16	0	13	0.00	0	14	-2.34

**Table 12.3 Bit Rates and BRR Settings (cont)**

Bit Rate (Bits/s)	$\phi$ (MHz)								
	18.432			19.6608			20		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	3	81	-0.22	3	86	0.31	3	88	-0.25
150	2	239	0.00	2	255	0.00	3	64	0.16
300	2	119	0.00	2	127	0.00	2	129	0.16
600	1	239	0.00	1	255	0.00	2	64	0.16
1200	1	119	0.00	1	127	0.00	1	129	0.16
2400	0	239	0.00	0	255	0.00	1	64	0.16
4800	0	119	0.00	0	127	0.00	0	129	0.16
9600	0	59	0.00	0	63	0.00	0	64	0.16
14400	0	39	0.00	0	42	-0.78	0	42	0.94
19200	0	29	0.00	0	31	0.00	0	32	-1.36
28800	0	19	0.00	0	20	1.59	0	21	-1.36
31250	0	17	2.40	0	19	-1.70	0	19	0.00
38400	0	14	0.00	0	15	0.00	0	15	1.73

The BRR setting is calculated as follows:

$$N = \frac{\phi}{64 \times 2^{2n-1} \times B} \times 10^6 - 1$$

B: Bit rate (bit/s)

N: Baud rate generator BRR setting (0 N 255)

$\phi$ : Operating frequency (MHz)

n: Baud rate generator input clock (n = 0 to 3)

(See the following table for the clock sources and value of n.)

n	Clock	SMR Settings	
		CKS1	CKS2
0	$\phi$	0	0
1	$\phi/4$	0	1
2	$\phi/16$	1	0
3	$\phi/64$	1	1

The bit rate error in asynchronous mode is calculated as follows:

$$\text{Error (\%)} = \left\{ \frac{\phi \times 10^6}{(N + 1) \times B \times 64 \times 2^{2n-1}} - 1 \right\} \times 100$$

Table 12.4 shows the maximum bit rates for various frequencies.

**Table 12.4 Maximum Bit Rates for Various Frequencies with Baud Rate Generator**

$\phi$ (MHz)	Maximum Bit Rate (Bits/s)	Settings	
		n	N
4	125000	0	0
4.9152	153600	0	0
6	187500	0	0
7.3728	230400	0	0
8	250000	0	0
9.8304	307200	0	0
10	312500	0	0
11.0592	345600	0	0
12	375000	0	0
12.288	384000	0	0
14	437500	0	0
14.7456	460800	0	0
16	500000	0	0
17.2032	537600	0	0
18	562500	0	0
18.432	576000	0	0
19.6608	614400	0	0
20	625000	0	0

## 12.3 Operation

### 12.3.1 Overview

The SCI can perform serial communication in asynchronous mode, in which characters are synchronized individually. The asynchronous mode transmission format is selected in the serial mode register (SMR), as shown in table 12.5.

#### Asynchronous Mode:

- Data length is selectable: seven or eight bits.
- Parity and multiprocessor bits are selectable, as well as the stop bit length (one or two bits). These selections determine the transmit/receive format and character length.
- In receiving, it is possible to detect framing errors (FER), parity errors (PER), overrun errors (ORER), and the break state.
- SCI clock source: internal clock  
The SCI operates using the on-chip baud rate generator.

**Table 12.5 Serial Mode Register Settings and SCI Communication Formats**

Mode	SMR Settings				SCI Communication Format				
	Bit 6 CHR	Bit 5 PE	Bit 2 MP	Bit 3 STOP	Data Length	Parity Bit	Multipro- cessor Bit	Stop Bit Length	
Asynchronous	0	0	0	0	8-bit	Not set	Not set	1 bit	
				1				2 bits	
				0				1 bit	
				1				2 bits	
	1	0	0	0	7-bit	Not set		1 bit	
				1				2 bits	
				0				1 bit	
				1				2 bits	
Asynchronous (multiprocessor format)	0	*	1	0	8-bit	Not set	Set	1 bit	
				1				2 bits	
				0				7-bit	1 bit
				1				2 bits	

Note: Asterisks (\*) in the table indicate don't-care bits.

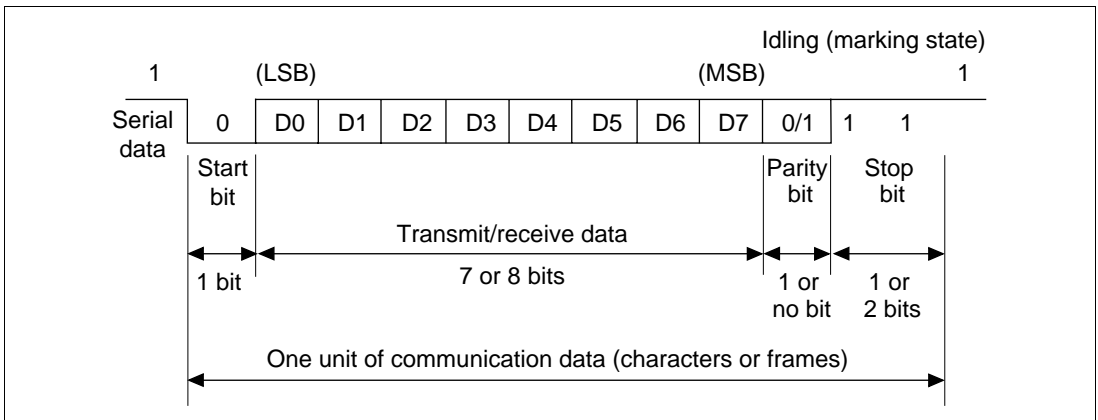
### 12.3.2 Operation in Asynchronous Mode

In the asynchronous mode, each transmitted or received character begins with a start bit and ends with a stop bit. Serial communication is synchronized one character at a time.

The transmitting and receiving sections of the SCI are independent, so full duplex communication is possible. The transmitter and receiver are both double buffered, so data can be written and read while transmitting and receiving are in progress, enabling continuous transmitting and receiving.

Figure 12.2 shows the general format of asynchronous serial communication. In asynchronous serial communication, the communication line is normally held in the marking (high) state. The SCI monitors the line and starts serial communication when the line goes to the space (low) state, indicating a start bit. One serial character consists of a start bit (low), data (LSB first), parity bit (high or low), and stop bit (high), in that order.

When receiving in the asynchronous mode, the SCI synchronizes on the falling edge of the start bit. The SCI samples each data bit on the eighth pulse of a clock with a frequency 16 times the bit rate. Receive data is latched at the center of each bit.



**Figure 12.2 Data Format in Asynchronous Communication (Example: 8-bit Data with Parity and Two Stop Bits)**

**Transmit/Receive Formats:** Table 12.6 shows the 12 communication formats that can be selected in the asynchronous mode. The format is selected by settings in the serial mode register (SMR).

**Table 12.6 Serial Communication Formats (Asynchronous Mode)**

SMR Bits				Serial Transmit/Receive Format and Frame Length												
Bit 6: CHR	Bit 2: MP	Bit 5: PE	Bit 3: STOP	1	2	3	4	5	6	7	8	9	10	11	12	
0	0	0	0	START	8-Bit data								STOP			
0	0	0	1	START	8-Bit data								STOP	STOP		
0	0	1	0	START	8-Bit data								P	STOP		
0	0	1	1	START	8-Bit data								P	STOP	STOP	
1	0	0	0	START	7-Bit data							STOP				
1	0	0	1	START	7-Bit data							STOP	STOP			
1	0	1	0	START	7-Bit data							P	STOP			
1	0	1	1	START	7-Bit data							P	STOP	STOP		
0	1	—	0	START	8-Bit data								MPB	STOP		
0	1	—	1	START	8-Bit data								MPB	STOP	STOP	
1	1	—	0	START	7-Bit data							MPB	STOP			
1	1	—	1	START	7-Bit data							MPB	STOP	STOP		

—: Don't care bits.

Note: START: Start bit  
 STOP: Stop bit  
 P: Parity bit  
 MPB: Multiprocessor bit

**Clock:** The SCI's transmit/receive clock is an internal clock generated by the on-chip baud rate generator.

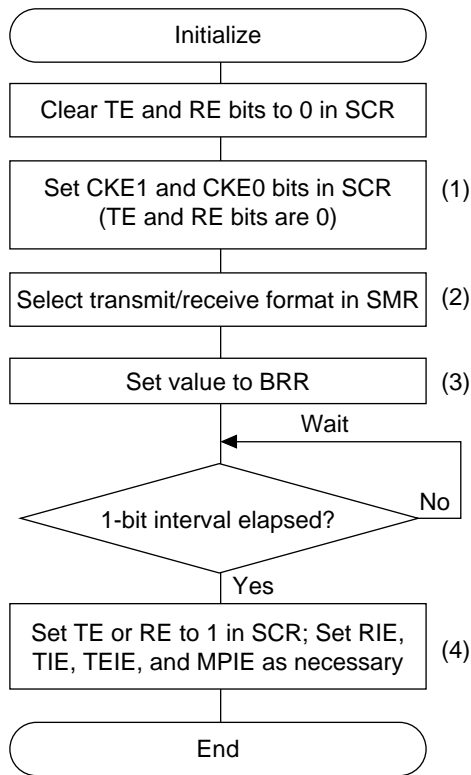
**SCI Initialization (Asynchronous Mode):** Before transmitting or receiving, clear the TE and RE bits to 0 in the serial control register (SCR), then initialize the SCI as follows.

When changing the operation mode or communication format, always clear the TE and RE bits to 0 before following the procedure given below. Clearing TE to 0 sets TDRE to 1 and initializes the transmit shift register (TSR). Clearing RE to 0, however, does not initialize the RDRF, PER, FER, and ORER flags and receive data register (RDR), which retain their previous contents.

Figure 12.3 is a sample flowchart for initializing the SCI. The procedure is as follows (the steps correspond to the numbers in the flowchart):

1. Select the clock source in the serial control register (SCR). Leave RIE, TIE, TEIE, MPIE, TE and RE cleared to 0.
2. Select the communication format in the serial mode register (SMR).
3. Write the value corresponding to the bit rate in the bit rate register (BRR).
4. Wait for at least the interval required to transmit or receive one bit, then set TE or RE in the serial control register (SCR) to 1. Also set RIE, TIE, TEIE and MPIE as necessary. Setting TE or RE enables the SCI to use the TxD or RxD pin. The initial states are the marking transmit state, and the idle receive state (waiting for a start bit).

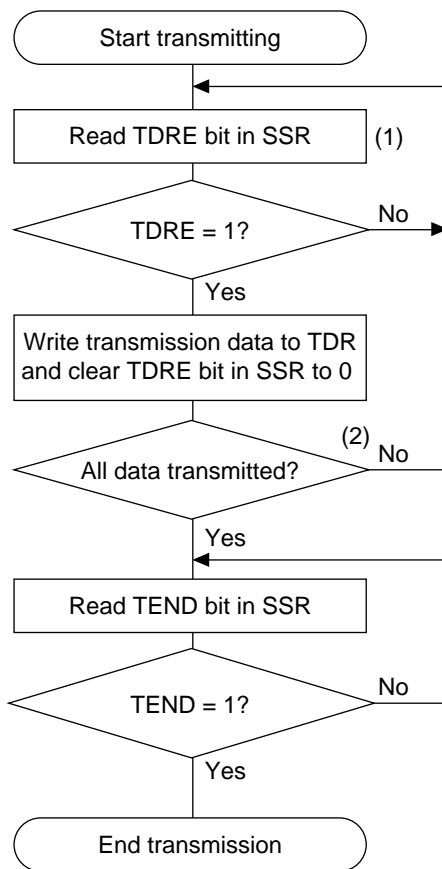




**Figure 12.3 Sample Flowchart for SCI Initialization**

**Transmitting Serial Data (Asynchronous Mode):** Figure 12.4 shows a sample flowchart for transmitting serial data. The procedure is as follows (the steps correspond to the numbers in the flowchart):

1. SCI status check and transmit data write: Read the serial status register (SSR), check that the TDRE bit is 1, then write transmit data in the transmit data register (TDR) and clear TDRE to 0.
2. To continue transmitting serial data: Read the TDRE bit to check whether it is safe to write (if it reads 1); if so, write data in TDR, then clear TDRE to 0.



**Figure 12.4 Sample Flowchart for Transmitting Serial Data**

In transmitting serial data, the SCI operates as follows:

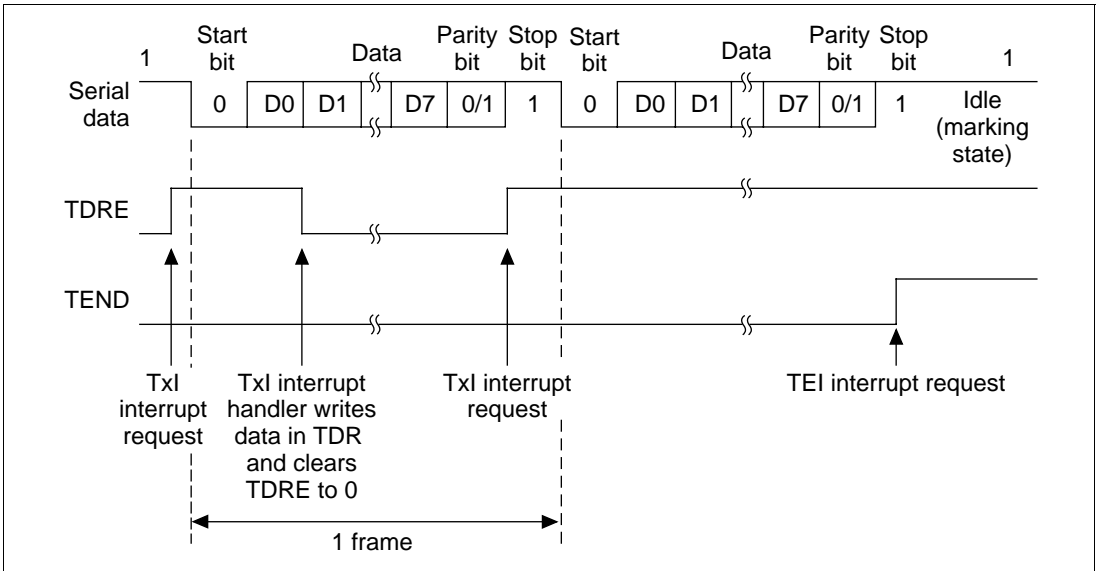
1. The SCI monitors the TDRE bit in the SSR. When TDRE is cleared to 0, the SCI recognizes that the transmit data register (TDR) contains new data, and loads this data from the TDR into the transmit shift register (TSR).
2. After loading the data from the TDR into the TSR, the SCI sets the TDRE bit to 1 and starts transmitting. If the transmit-data-empty interrupt enable bit (TIE) is set to 1 in the SCR, the SCI requests a transmit-data-empty interrupt (TxI) at this time.

Serial transmit data is transmitted in the following order from the TxD pin:

- a. Start bit: one 0 bit is output.
- b. Transmit data: seven or eight bits of data are output, LSB first.
- c. Parity bit or multiprocessor bit: one parity bit (even or odd parity) or one multiprocessor bit is output. Formats in which neither a parity bit nor a multiprocessor bit is output can also be selected.

- d. Stop bit: one or two 1 bits (stop bits) are output.
  - e. Marking: output of 1 bits continues until the start bit of the next transmit data.
3. The SCI checks the TDRE bit when it outputs the stop bit. If TDRE is 0, the SCI loads new data from the TDR into the TSR, outputs the stop bit, then begins serial transmission of the next frame. If TDRE is 1, the SCI sets the TEND bit to 1 in the SSR, outputs the stop bit, then continues output of 1 bits (marking). If the transmit-end interrupt enable bit (TEIE) in the SCR is set to 1, a transmit-end interrupt (TEI) is requested.

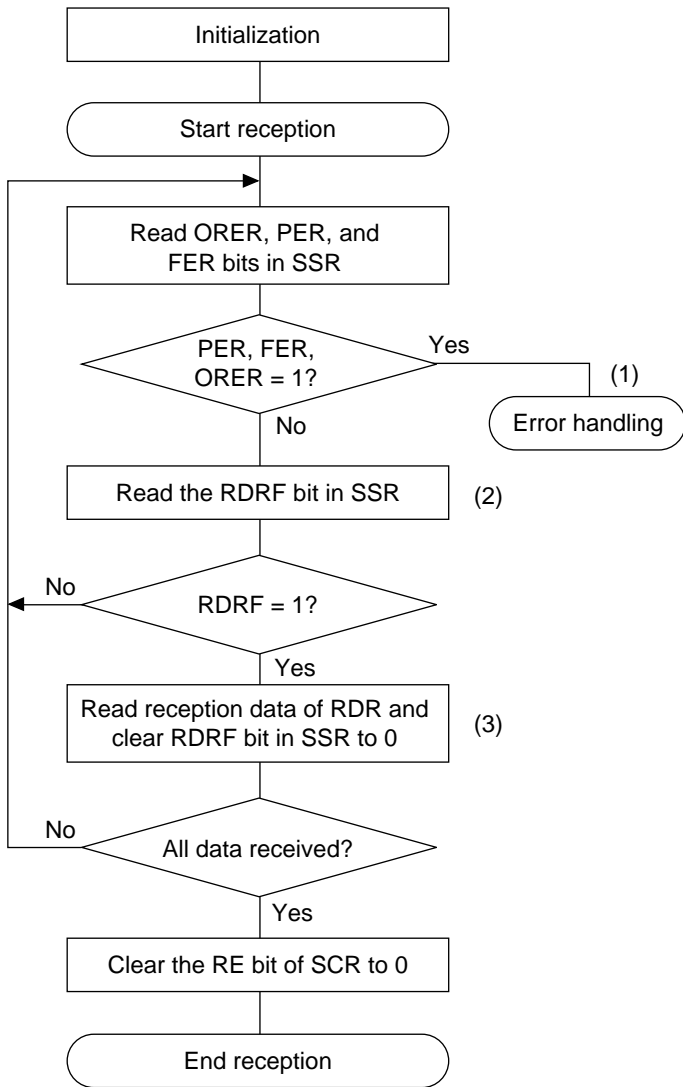
Figure 12.5 shows an example of SCI transmit operation.



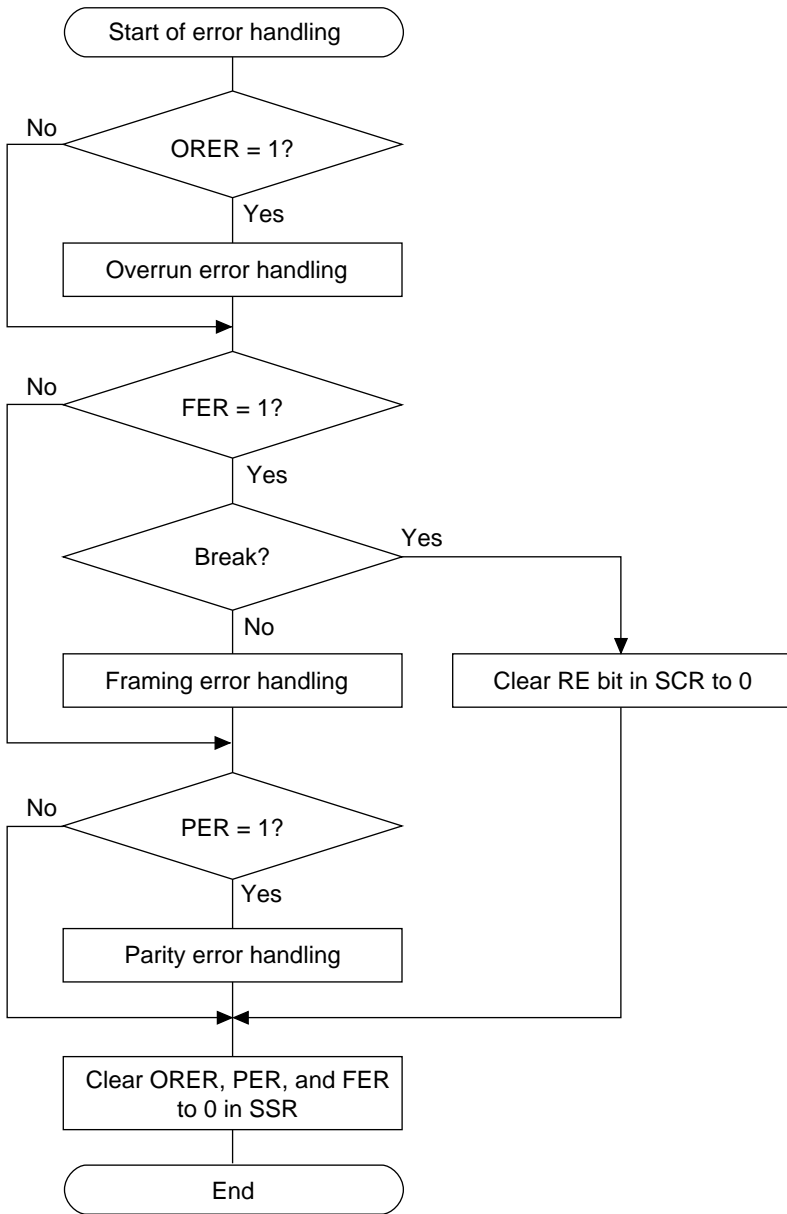
**Figure 12.5 SCI Transmit Operation in Asynchronous Mode (8-Bit Data with Parity and One Stop Bit)**

**Receiving Serial Data:** Figures 12.6 show a sample flowchart for receiving serial data. The procedure is as follows (the steps correspond to the numbers in the flowchart).

1. Receive error handling and break detection: If a receive error occurs, read the ORER, PER, and FER bits of the SSR to identify the error. After executing the necessary error handling, clear ORER, PER, and FER all to 0. Receiving cannot resume if ORER, PER or FER remain set to 1. When a framing error occurs, the RxD pin can be read to detect the break state.
2. SCI status check and receive-data read: Read the serial status register (SSR), check that RDRF is set to 1, then read receive data from the receive data register (RDR) and clear RDRF to 0. The RxI interrupt can also be used to determine if the RDRF bit has changed from 0 to 1.
3. Continue receiving serial data: Read the RDR and RDRF bit and clear RDRF to 0 before the stop bit of the current frame is received.



**Figure 12.6 Sample Flowchart for Receiving Serial Data**



**Figure 12.6 Sample Flowchart for Receiving Serial Data (cont)**

In receiving, the SCI operates as follows:

1. The SCI monitors the communication line. When it detects a start bit (0), the SCI synchronizes internally and starts receiving.
2. Receive data is shifted into the RSR in order from the LSB to the MSB.
3. The parity bit and stop bit are received. After receiving these bits, the SCI makes the following checks:
  - a. Parity check. The number of 1s in the receive data must match the even or odd parity setting of the  $O/\bar{E}$  bit in the SMR.
  - b. Stop bit check. The stop bit value must be 1. If there are two stop bits, only the first stop bit is checked.
  - c. Status check. RDRF must be 0 so that receive data can be loaded from the RSR into the RDR.

If the data passes these checks, the SCI sets RDRF to 1 and stores the received data in the RDR. If one of the checks fails (receive error), the SCI operates as indicated in table 12.16.

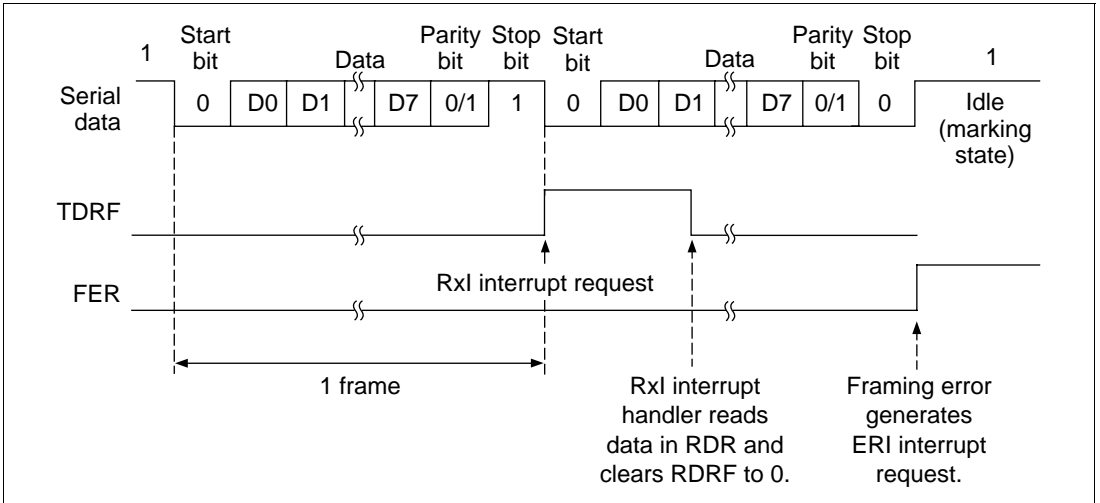
Note: When a receive error occurs, further receiving is disabled. While receiving, the RDRF bit is not set to 1, so be sure to clear the error flags.

4. After setting RDRF to 1, if the receive-data-full interrupt enable bit (RIE) is set to 1 in the SCR, the SCI requests a receive-data-full interrupt (RxI). If one of the error flags (ORER, PER, or FER) is set to 1 and the receive-data-full interrupt enable bit (RIE) in the SCR is also set to 1, the SCI requests a receive-error interrupt (ERI).

Figure 12.7 shows an example of SCI receive operation in the asynchronous mode.

**Table 12.7 Receive Error Conditions and SCI Operation**

<b>Receive Error</b>	<b>Abbreviation</b>	<b>Condition</b>	<b>Data Transfer</b>
Overrun error	ORER	Receiving of next data ends while RDRF is still set to 1 in SSR	Receive data not loaded from RSR into RDR
Framing error	FER	Stop bit is 0	Receive data loaded from RSR into RDR
Parity error	PER	Parity of receive data differs from even/odd parity setting in SMR	Receive data loaded from RSR into RDR



**Figure 12.7 SCI Receive Operation (8-Bit Data with Parity and One Stop Bit)**

### 12.3.3 Multiprocessor Communication

The multiprocessor communication function enables several processors to share a single serial communication line for sending and receiving data. The processors communicate in the asynchronous mode using a format with an additional multiprocessor bit (multiprocessor format).

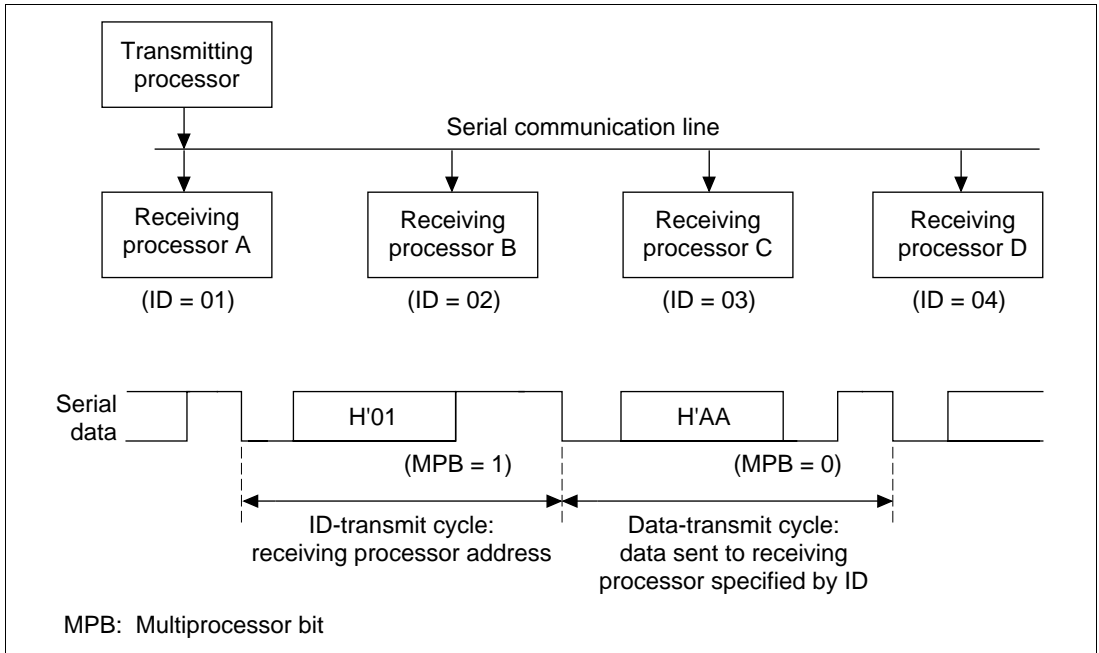
In multiprocessor communication, each receiving processor is addressed by a unique ID. A serial communication cycle consists of an ID-sending cycle that identifies the receiving processor, and a data-sending cycle. The multiprocessor bit distinguishes ID-sending cycles from data-sending cycles. The transmitting processor starts by sending the ID of the receiving processor with which it wants to communicate as data with the multiprocessor bit set to 1. Next the transmitting processor sends transmit data with the multiprocessor bit cleared to 0.

Receiving processors skip incoming data until they receive data with the multiprocessor bit set to 1. When they receive data with the multiprocessor bit set to 1, receiving processors compare the data with their IDs. The receiving processor with a matching ID continues to receive further incoming data. Processors with IDs not matching the received data skip further incoming data until they again receive data with the multiprocessor bit set to 1. Multiple processors can send and receive data in this way.

Figure 12.8 shows the example of communication among processors using the multiprocessor format.

**Communication Formats:** Four formats are available. Parity-bit settings are ignored when the multiprocessor format is selected. For details see table 12.5.

**Clock:** See the description in the asynchronous mode section.

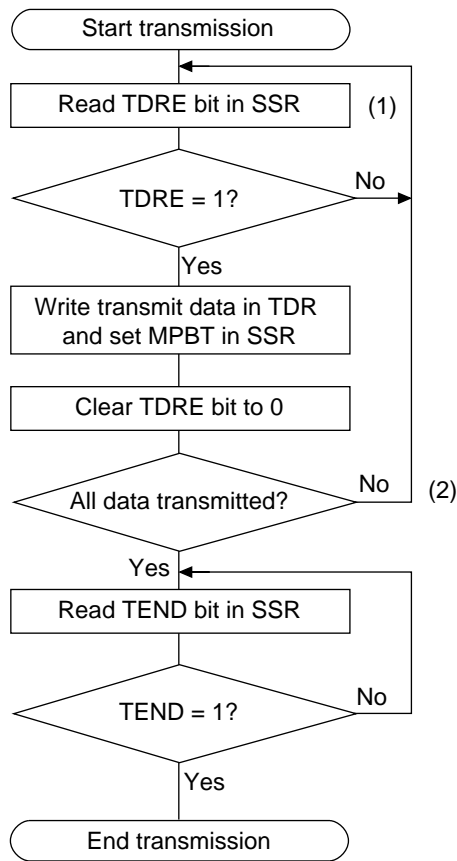


**Figure 12.8 Communication Among Processors Using Multiprocessor Format (Sending Data H'AA to Receiving Processor A)**

**Transmitting Multiprocessor Serial Data:** Figure 12.9 shows a sample flowchart for transmitting multiprocessor serial data. The procedure is as follows (the steps correspond to the numbers in the flowchart):

1. SCI status check and transmit data write: Read the serial status register (SSR), check that the TDRE bit is 1, then write transmit data in the transmit data register (TDR). Also set MPBT (multiprocessor bit transfer) to 0 or 1 in SSR. Finally, clear TDRE to 0.
2. Continue transmitting serial data: Read the TDRE bit to check whether it is safe to write (if it reads 1); if so, write data in TDR, then clear TDRE to 0.





**Figure 12.9 Sample Flowchart for Transmitting Multiprocessor Serial Data**

In transmitting serial data, the SCI operates as follows:

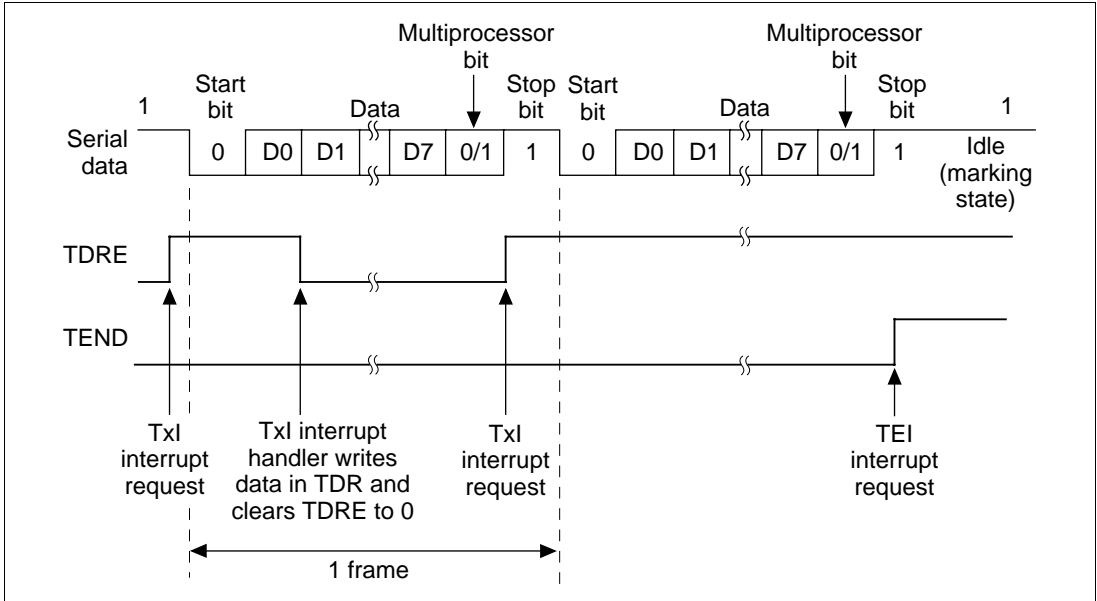
1. The SCI monitors the TDRE bit in the SSR. When TDRE is cleared to 0 the SCI recognizes that the transmit data register (TDR) contains new data, and loads this data from the TDR into the transmit shift register (TSR).
2. After loading the data from the TDR into the TSR, the SCI sets the TDRE bit to 1 and starts transmitting. If the transmit-data-empty interrupt enable bit (TIE) in the SCR is set to 1, the SCI requests a transmit-data-empty interrupt (TxI) at this time.

Serial transmit data is transmitted in the following order from the TxD pin:

- a. Start bit: one 0 bit is output.
- b. Transmit data: seven or eight bits are output, LSB first.
- c. Multiprocessor bit: one multiprocessor bit (MPBT value) is output.
- d. Stop bit: one or two 1 bits (stop bits) are output.
- e. Marking: output of 1 bits continues until the start bit of the next transmit data.

- The SCI checks the TDRE bit when it outputs the stop bit. If TDRE is 0, the SCI loads data from the TDR into the TSR, outputs the stop bit, then begins serial transmission of the next frame. If TDRE is 1, the SCI sets the TEND bit in the SSR to 1, outputs the stop bit, then continues output of 1 bits in the marking state. If the transmit-end interrupt enable bit (TEIE) in the SCR is set to 1, a transmit-end interrupt (TEI) is requested at this time.

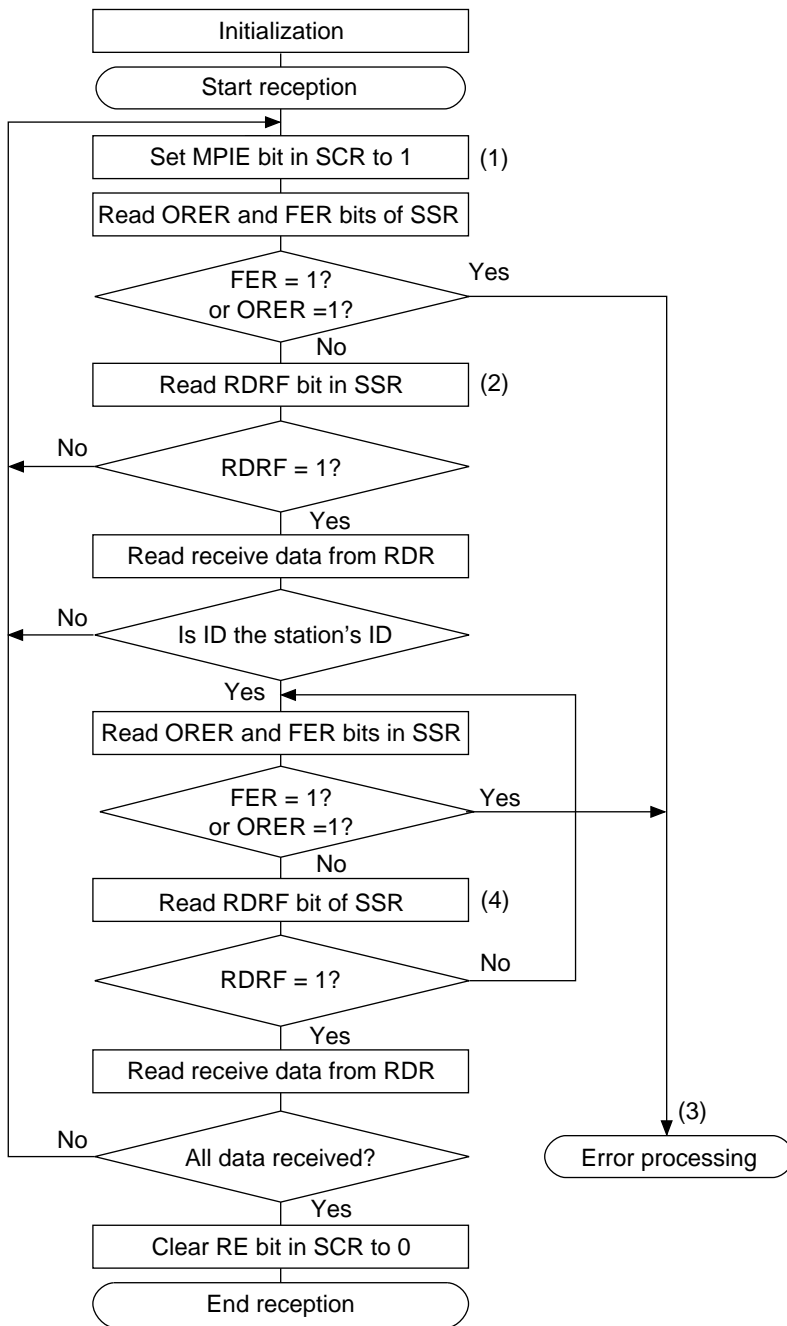
Figure 12.10 shows an example of SCI receive operation in the multiprocessor format.



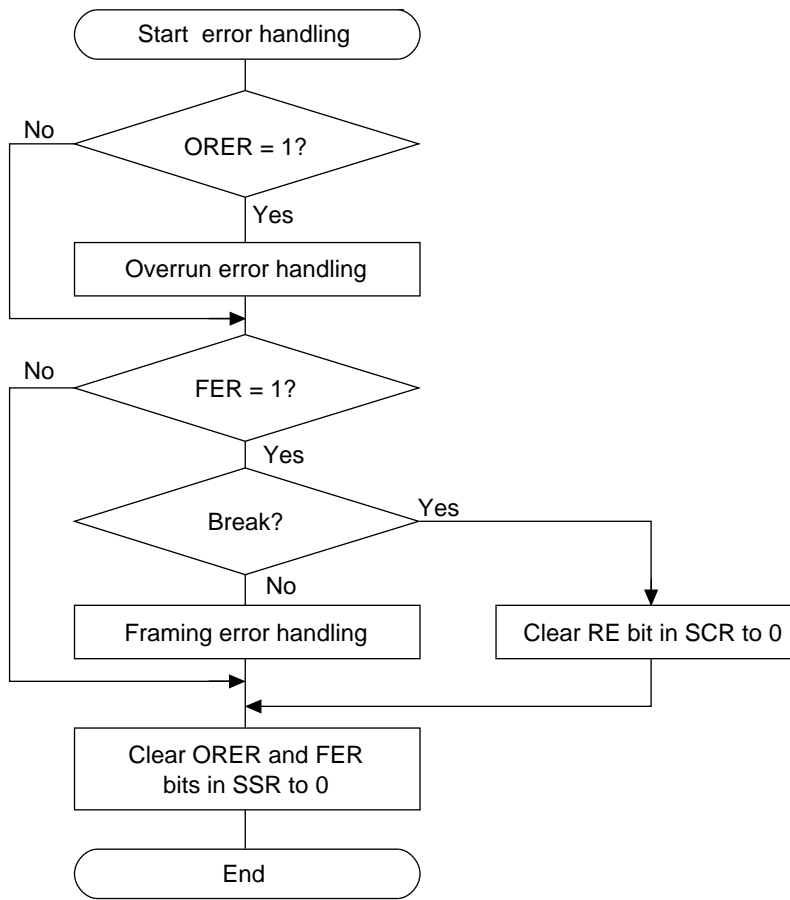
**Figure 12.10 SCI Multiprocessor Transmit Operation (8-Bit Data with Multiprocessor Bit and One Stop Bit)**

**Receiving Multiprocessor Serial Data:** Figure 12.11 shows a sample flowchart for receiving multiprocessor serial data. The procedure for receiving multiprocessor serial data is listed below.

- ID receive cycle: Set the MPIE bit in the serial control register (SCR) to 1.
- SCI status check and compare to ID reception: Read the serial status register (SSR), check that RDRF is set to 1, then read data from the receive data register (RDR) and compare with the processor's own ID. If the ID does not match the receive data, set MPIE to 1 again and clear RDRF to 0. If the ID matches the receive data, clear RDRF to 0.
- Receive error handling and break detection: If a receive error occurs, read the ORER and FER bits in SSR to identify the error. After executing the necessary error processing, clear both ORER and FER to 0. Receiving cannot resume if ORER or FER remain set to 1. When a framing error occurs, the RxD pin can be read to detect the break state.
- SCI status check and data receiving: Read SSR, check that RDRF is set to 1, then read data from the receive data register (RDR).



**Figure 12.11 Sample Flowchart for Receiving Multiprocessor Serial Data**



**Figure 12.11 Sample Flowchart for Receiving Multiprocessor Serial Data (cont)**

Figures 12.12 show examples of SCI receive operation using a multiprocessor format.

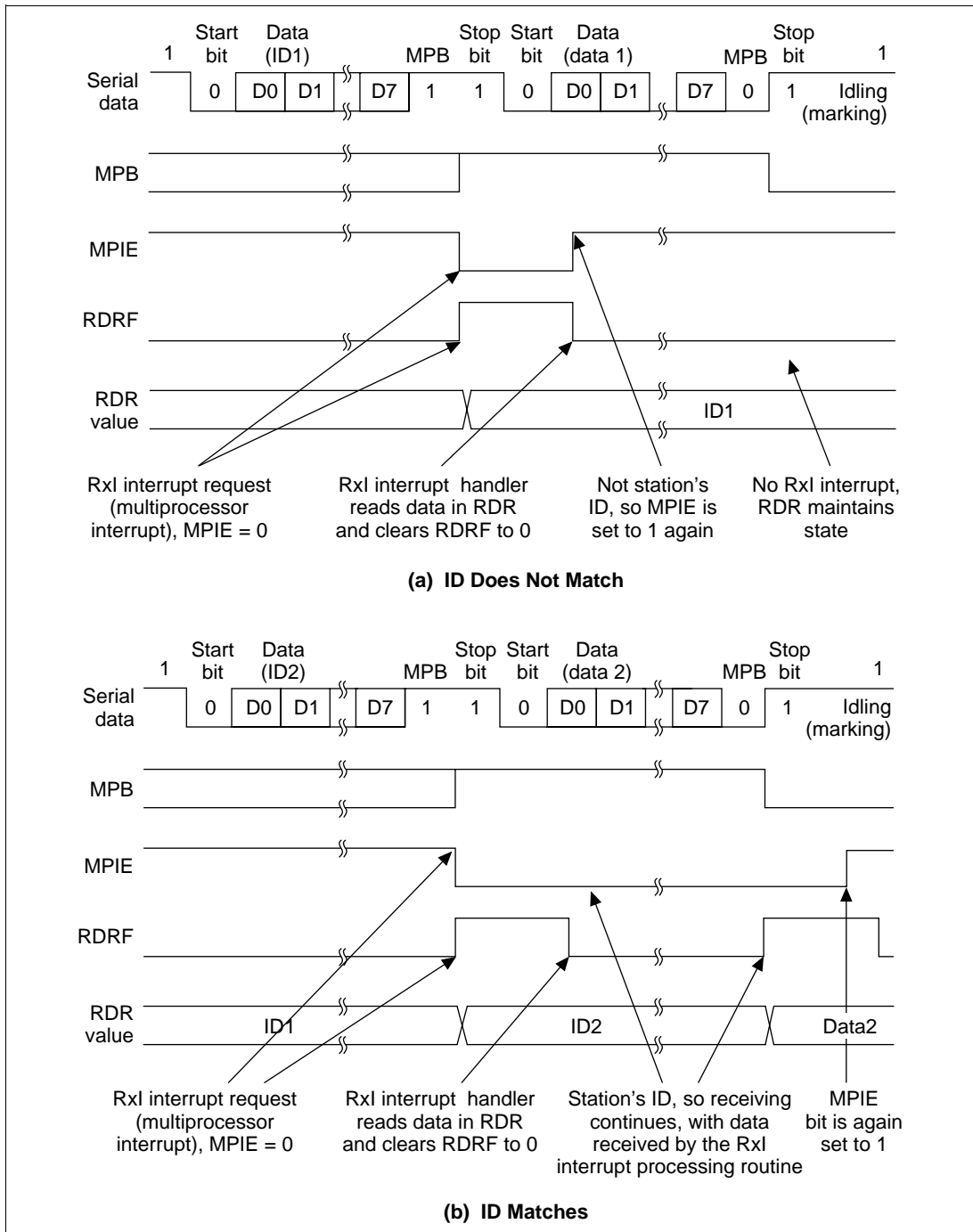


Figure 12.12 SCI Receive Operation (8-Bit Data with Multiprocessor Bit and One Stop Bit)

## 12.4 Interrupt

The SCI has four interrupt sources: transmit-end (TEI), receive-error (ERI), receive-data-full (RxI), and transmit-data-empty (TxI). Table 12.8 lists the interrupt sources and indicates their priority. These interrupts can be enabled and disabled by the TIE, RIE, and TEIE bits in the serial control register (SCR). Each interrupt request is sent separately to the interrupt controller.

TxI is requested when the TDRE bit in the serial status register (SSR) is set to 1. TDRE is automatically cleared to 0 when a write to the transmit data register (TDR) is performed.

RxI is requested when the RDRF bit in the SSR is set to 1.

ERI is requested when the ORER, FER, or PER bit in the SSR is set to 1.

TEI is requested when the TEND bit in the SSR is set to 1.

Where the TxI interrupt indicates that transmit data writing is enabled, the TEI interrupt indicates that the transmit operation has ended.

**Table 12.8 SCI Interrupt Sources**

<b>Interrupt Source</b>	<b>Description</b>	<b>Priority</b>
ERI	Receive error (ORER, PER, or FER)	High
RxI	Receive data full (RDRF)	
TxI	Transmit data empty (TDRE)	
TEI	Transmit end (TEND)	Low

## 12.5 Notes on Use

The following points should be noted when using the SCI.

**TDR Write and TDRE Flags:** The TDRE bit in the serial status register (SSR) is a status flag indicating loading of transmit data from TDR into TSR. The SCI sets TDRE to 1 when it transfers data from TDR to TSR. Data can be written to TDR regardless of the TDRE bit status. If new data is written in TDR when TDRE is 0, however, the old data stored in TDR will be lost because the data has not yet been transferred to the TSR. Before writing transmit data to the TDR, be sure to check that TDRE is set to 1.

**Simultaneous Multiple Receive Errors:** Table 12.9 indicates the state of the SSR status flags when multiple receive errors occur simultaneously. When an overrun error occurs, the RSR contents cannot be transferred to the RDR, so receive data is lost.

**Table 12.9 SSR Status Flags and Transfer of Receive Data**

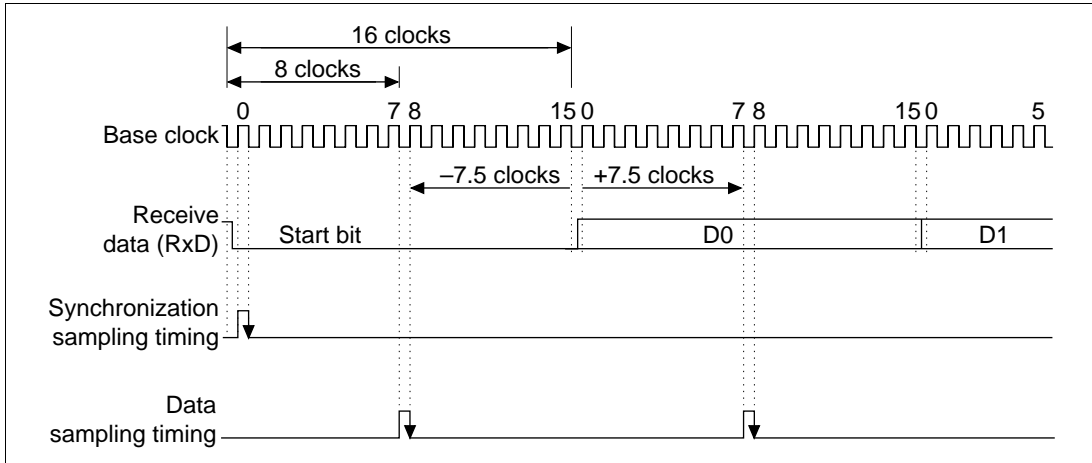
Receive Error Status	SSR Status Flags				Receive Data Transfer
	RDRF	ORER	FER	PER	RSR → RDR
Overrun error	1	1	0	0	X
Framing error	0	0	1	0	O
Parity error	0	0	0	1	O
Overrun error + framing error	1	1	1	0	X
Overrun error + parity error	1	1	0	1	X
Framing error + parity error	0	0	1	1	O
Overrun error + framing error + parity error	1	1	1	1	X

Note: O = Receive data is transferred from RSR to RDR.

X = Receive data is not transferred from RSR to RDR.

**Break Detection and Processing:** Break signals can be detected by reading the RxD pin directly when a framing error (FER) is detected. In the break state, the input from the RxD pin consists of all 0s, so FER is set and the parity error flag (PER) may also be set. In the break state, the SCI receiver continues to operate, so if the FER bit is cleared to 0, it will be set to 1 again.

**Receive Data Sampling Timing and Receive Margin:** The SCI operates on a base clock of 16 times the bit rate frequency. In receiving, the SCI synchronizes internally with the falling edge of the start bit, which it samples on the base clock. Receive data is latched on the rising edge of the eighth base clock pulse (figure 12.13).



**Figure 12.13 Receive Data Sampling Timing**

The receive margin in the asynchronous mode can therefore be expressed as:

$$M = \left| \left( 0.5 - \frac{1}{2N} \right) - (L - 0.5) F - \frac{|D - 0.5|}{N} (1 + F) \right| \times 100\%$$

M : Receive margin (%)

N : Ratio of clock frequency to bit rate (N = 16)

D : Clock duty cycle (D = 0–1.0)

L : Frame length (L = 9–12)

F : Absolute deviation of clock frequency

From the equation above, if F = 0 and D = 0.5 the receive margin is 46.875%:

$$D = 0.5, F = 0$$

$$M = \left( 0.5 - \frac{1}{2 \times 16} \right) \times 100\%$$

$$= 46.875\%$$

This is a theoretical value. A reasonable margin to allow in system designs is 20–30%.



# Section 13 A/D Converter (A/D)

## 13.1 Overview

The A/D converter has 10-bit resolution, and can select from a maximum of seven channels of analog inputs.

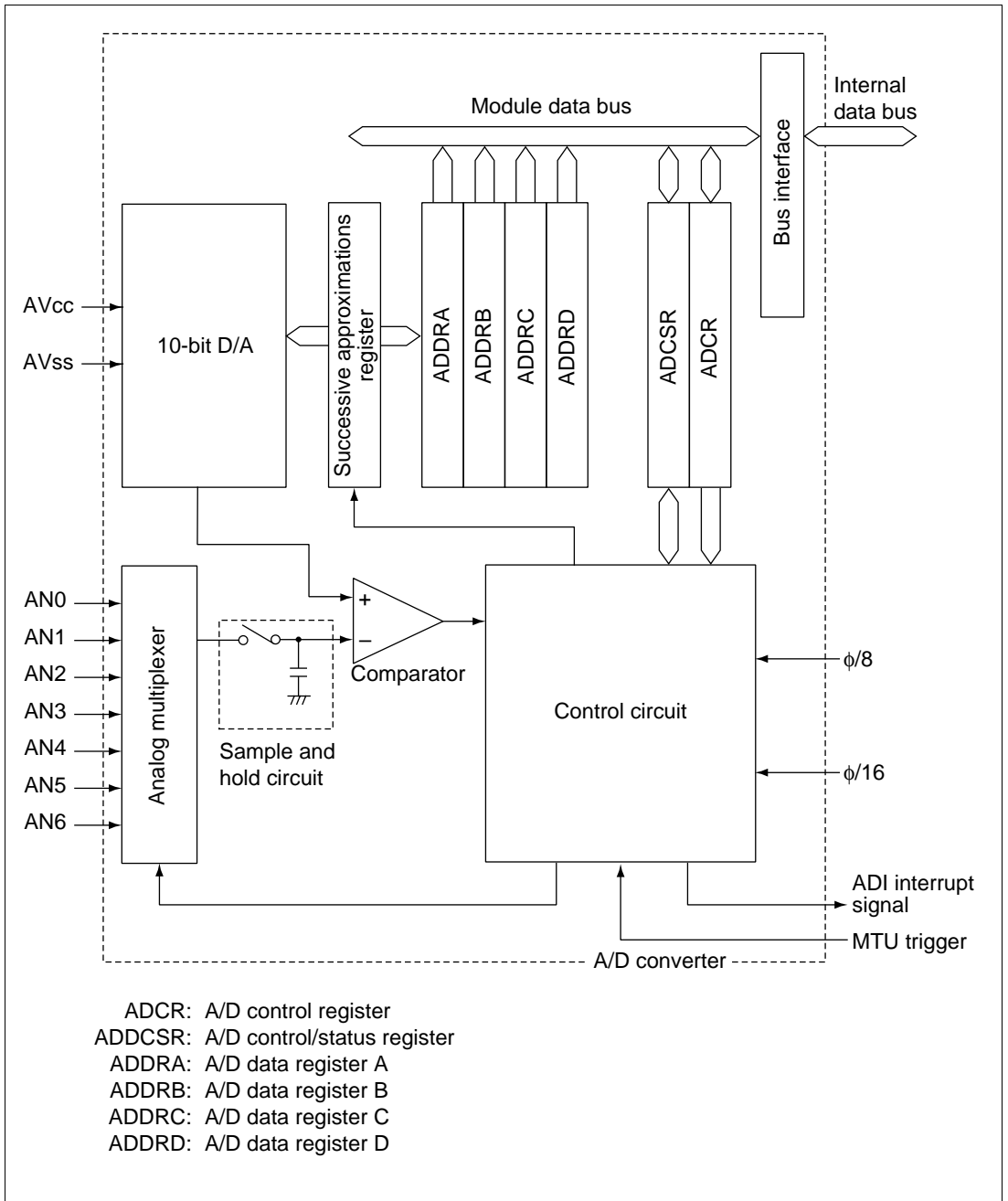
### 13.1.1 Features

The A/D converter has the following features:

- 10-bit resolution
- Seven input channels
- High-speed conversion
  - Minimum conversion time: 6.7  $\mu$ s per channel (for 20-MHz operation)
- Two operating modes: single mode or scan mode
  - Single mode: A/D conversion on one channel
  - Scan mode: Continuous A/D conversion on one to four channels
- Four 16-bit data registers  
Conversion results transferred to and stored in data registers corresponding to each channel.
- Sample and hold function
- A/D conversion end interrupt generation  
An A/D conversion end interrupt (ADI) request can be generated on completion of A/D conversion.
- A/D conversion can be started by MTU trigger input.

### 13.1.2 Block Diagram

Figure 13.1 is the block diagram of the A/D converter.



**Figure 13.1 A/D Converter Block Diagram**

### 13.1.3 Pin Configuration

Table 13.1 shows the input pins used by the A/D converter.

The seven analog input pins are divided into two groups: group 0, comprising analog input pins 0–3 (AN0–AN3), and group 1, comprising analog input pins 4–6 (AN4–AN6).

The  $AV_{CC}$  and  $AV_{SS}$  pins are for the A/D converter internal analog section power supply.

**Table 13.1 Pin Configuration**

Pin	Abbreviation	I/O	Function
Analog supply	$AV_{CC}$	I	Analog section power supply
Analog ground	$AV_{SS}$	I	Analog section ground and A/D conversion reference voltage
Analog input 0	AN0	I	Analog input group 0
Analog input 1	AN1	I	
Analog input 2	AN2	I	
Analog input 3	AN3	I	
Analog input 4	AN4	I	Analog input group 1
Analog input 5	AN5	I	
Analog input 6	AN6	I	

### 13.1.4 Register Configuration

Table 13.2 shows the configuration of the A/D converter registers.

**Table 13.2 Register Configuration**

Name	Abbreviation	R/W	Initial Value	Address	Access Size
A/D data register AH	ADDRAH	R	H'00	H'FFFF8420	8,16
A/D data register AL	ADDRAL	R	H'00	H'FFFF8421	16
A/D data register BH	ADDRBH	R	H'00	H'FFFF8422	8,16
A/D data register BL	ADDRBL	R	H'00	H'FFFF8423	16
A/D data register CH	ADDRCH	R	H'00	H'FFFF8424	8,16
A/D data register CL	ADDRCL	R	H'00	H'FFFF8425	16
A/D data register DH	ADDRDH	R	H'00	H'FFFF8426	8,16
A/D data register DL	ADDRDL	R	H'00	H'FFFF8427	16
A/D control/status register	ADCSR	R/(W)*	H'00	H'FFFF8428	8,16
A/D control register	ADCR	R/W	H'7F	H'FFFF8429	8,16

Note: Only 0 can be written to bit 7 to clear the flag.

## 13.2 Register Descriptions

### 13.2.1 A/D Data Registers A–D (ADDRA–ADDRD)

The A/D data registers (ADDR) are 16-bit read-only registers for storing A/D conversion results. There are four of these registers, ADDRA through ADDRD.

The A/D-converted data is 10-bit data which is transferred to the ADDR for the selected channel for storage. The upper 8 bits of the converted data correspond to the upper byte of the ADDR, and the lower 2 bits correspond to the lower byte. Bits 5–0 of the lower byte of the ADDR are reserved, and always read 0. Table 13.3 shows the correspondence between the analog input channels and the ADDR registers.

The ADDR registers can be read by the CPU at all times. The upper byte is read directly, but the lower byte data is transferred via a temporary register (TEMP). For details, see section 13.3, CPU Interface.

The ADDR registers are initialized to H'0000 by a power-on reset.

Bit:	15	14	13	12	11	10	9	8
ADDRn:	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R
Bit:	7	6	5	4	3	2	1	0
ADDRn:	AD1	AD0	—	—	—	—	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R

(n = A-D)

**Table 13.3 Correspondence between Analog Input Channels and ADDRA-ADDRD**

Analog Input Channel		A/D Data Register
Group 0	Group 1	
AN0	AN4	ADDRA
AN1	AN5	ADDRB
AN2	AN6	ADDRC
AN3	—	ADDRD

### 13.2.2 A/D Control/Status Register (ADCSR)

The ADCSR is an 8-bit read/write register used for A/D conversion operation control and to indicate status.

The ADCSR is initialized to H'00 by power-on reset.

Bit:	7	6	5	4	3	2	1	0
	ADF	ADIE	ADST	SCAN	CKS	CH2	CH1	CH0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: The only value that can be written is a 0 to clear the flag.

- Bit 7—A/D End Flag (ADF): This status flag indicates that A/D conversion has ended.

Bit 7: ADF	Description
0	Clear conditions (initial value) With ADF = 1, by reading the ADF flag then writing 0 in ADF
1	Set conditions <ul style="list-style-type: none"> <li>• Single mode: When A/D conversion ends after conversion for all designated channels</li> <li>• Scan mode: After one round of A/D conversion for all specified channels</li> </ul>

- Bit 6—A/D Interrupt Enable (ADIE): Enables or disables interrupt requests (ADI) after A/D conversion ends.

Bit 6: ADIE	Description
0	Disables interrupt requests (ADI) after A/D conversion ends (initial value)
1	Enables interrupt requests (ADI) after A/D conversion ends

- Bit 5—A/D Start (ADST): Selects start or stop for A/D conversion.  
The ADST bit remains set to 1 during A/D conversion. It can also be set to 1 by MTU trigger input.

Bit 5: ADST	Description
0	A/D conversion halted (initial value)
1	Single mode: Start A/D conversion. Automatically cleared to 0 after conversion for the designated channel ends. Scan mode: Start A/D conversion. Continuous conversion until 0 cleared by software, and by power-on reset.

- Bit 4—Scan Mode (SCAN): Selects single mode or scan mode for A/D conversion. For details of the operation in single mode and scan mode, see section 13.4, Operation. Change the mode only when ADST = 0.

Bit 4: SCAN	Description
0	Single mode (initial value)
1	Scan mode

- Bit 3—Clock Select (CKS): Sets the A/D conversion time. Change the conversion time only when ADST = 0.

Bit 3: CKS	Description
0	Conversion time = 266 states (max.) (initial value)
1	Conversion time = 134 states (max.)

- Bits 2–0—Channel Select 2–0 (CH2–CH0): These bits, along with the SCAN bit, select the analog input channel.

Change the channel selection only when ADST = 0.

Bit 2: CH2	Bit 1: CH1	Bit 0: CH0	Description	
			Single Mode	Scan Mode
0	0	0	AN0 (initial value)	AN0 (initial value)
0	0	1	AN1	AN0–AN1
0	1	0	AN2	AN0–AN2
0	1	1	AN3	AN0–AN3
1	0	0	AN4	AN4
1	0	1	AN5	AN4, AN5
1	1	0	AN6	AN4–AN6
1	1	1	Reserved*	Reserved*

Note: This setting is reserved and must not be used.

### 13.2.3 A/D Control Register (ADCR)

The A/D control register (ADCR) is an 8-bit read/write register that enables or disables starting of A/D conversion by MTU trigger input. The ADCR is initialized to H'7F by a power-on reset or in standby mode.

Bit:	7	6	5	4	3	2	1	0
	TRGE	—	—	—	—	—	—	—
Initial value:	0	1	1	1	1	1	1	1
R/W:	R/W	R	R	R	R	R	R	R

- Bit 7—Trigger Enable (TRGE): Enables or disables starting of A/D conversion by MTU trigger input.

Bit 7: TRGE	Description
0	Disables A/D conversion start by MTU trigger input (initial value)
1	A/D conversion is started by MTU trigger

- Bits 6–0—Reserved: These bits always read 1. The write value should always be 1.

### 13.3 CPU Interface

ADDRA–ADDRD are 16-bit registers, but they are connected to the CPU by an 8-bit data bus. Therefore, while the upper byte is accessed directly by the CPU, the lower byte is accessed via an 8-bit temporary register (TEMP).

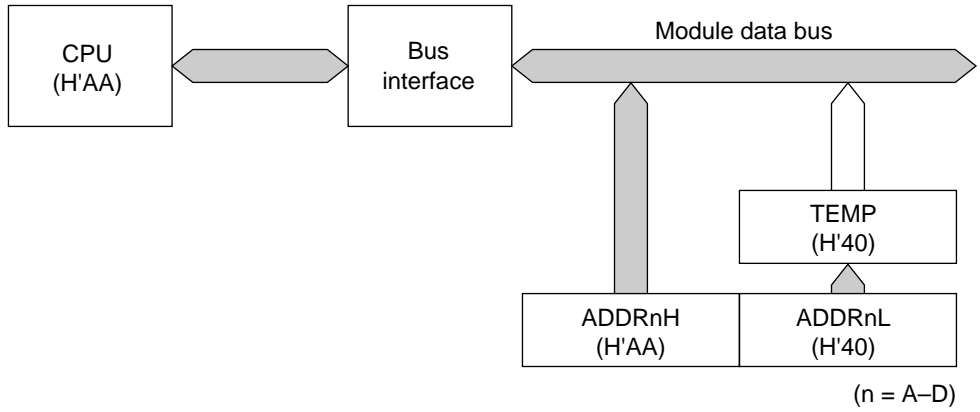
Data is read from an ADDR register as follows. When the upper byte is read, the upper-byte value is transferred directly to the CPU and the lower-byte value is transferred into TEMP. Next, when the lower byte is read, the TEMP contents are transferred to the CPU.

When reading an ADDR register, always read the upper byte before the lower byte. This operation can be performed by reading ADDR from the upper byte address using a word transfer instruction (such as MOV.W). It is possible to read only the upper byte, but if only the lower byte is read, incorrect data may be obtained.

Figure 13.2 shows the data flow for access to an ADDR register.



Upper-byte read



Lower-byte read

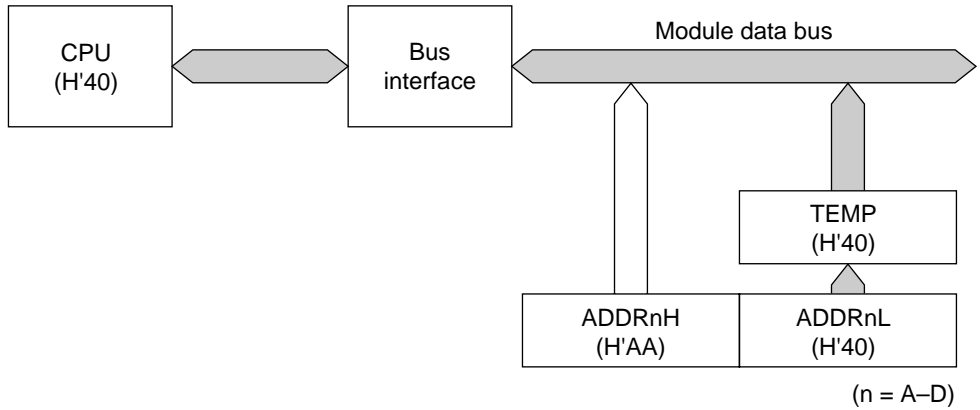


Figure 13.2 ADDR Access Operation (Reading H'AA40)

## 13.4 Operation

The A/D converter operates by successive approximations with 10-bit resolution. It has two operating modes: single mode and scan mode.. The operation in these two modes is described below.

### 13.4.1 Single Mode (SCAN = 0)

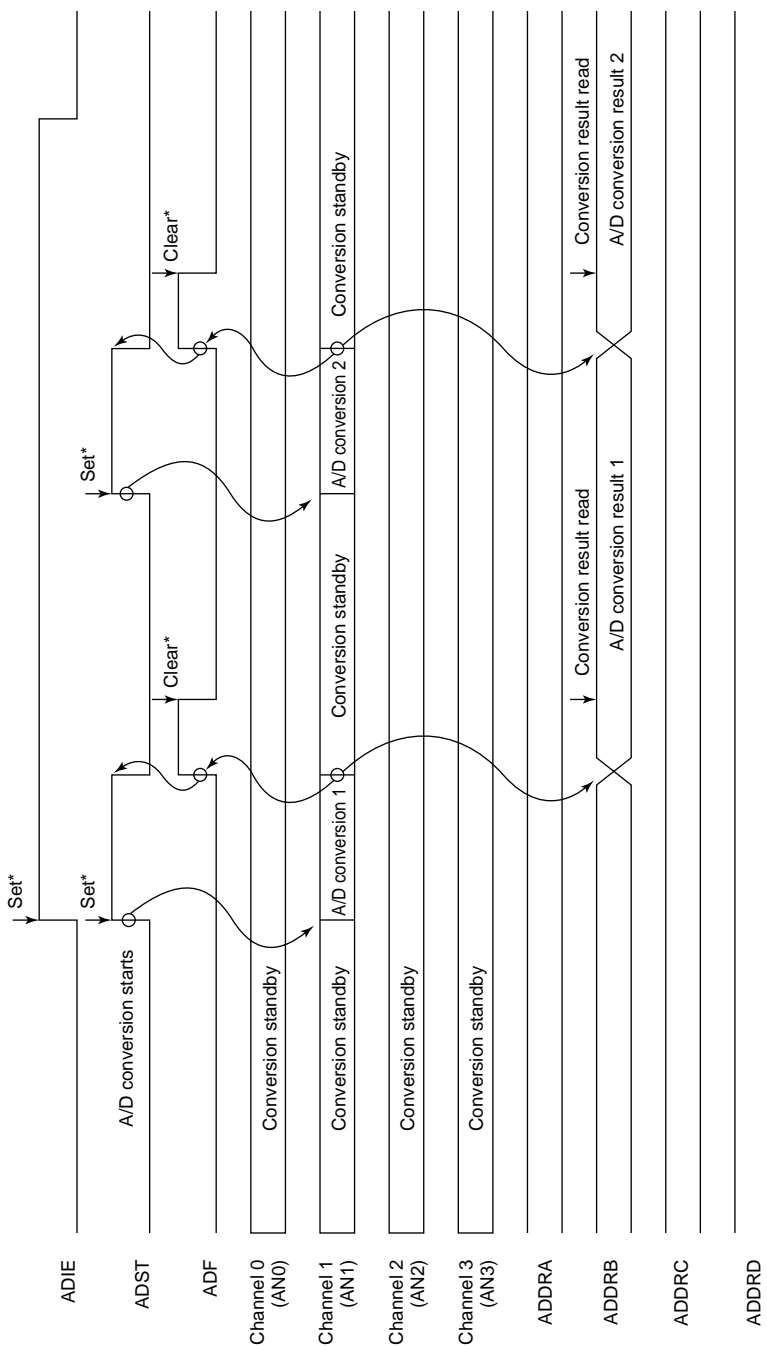
Single mode should be selected for A/D conversion on only one channel. A/D conversion starts when the ADST bit in the A/D control/status register (ADCSR) is set to 1 by software or MTU trigger input. The ADST bit remains set to 1 during A/D conversion, and is automatically cleared to 0 when conversion ends.

When conversion ends, the ADF bit in ADCSR is set to 1. If the ADIE bit in ADCSR is also 1, an ADI interrupt is requested. To clear the ADF bit, first read ADF when set to 1, then write 0 in ADF.

To prevent incorrect operation, A/D conversion should be halted by clearing the ADST bit to 0 before changing the mode or analog input channel. After the change is made, A/D conversion is restarted by setting the ADST bit to 1 (the mode or channel change and setting of the ADST bit can be carried out simultaneously).

An example of the operation when analog input channel 1 (AN1) is selected and A/D conversion is performed in single mode is described below. Figure 13.3 shows a timing diagram for this example.

1. Single mode is selected (SCAN = 0), input channel AN1 is selected (CH2 = CH1 = 0, CH0 = 1), the A/D interrupt request is enabled (ADIE = 1), and A/D conversion is started (ADST = 1).
2. When A/D conversion is completed, the result is transferred to ADDR0. At the same time ADF is set to 1, ADST is cleared to 0, and the A/D converter becomes idle.
3. Since ADF = 1 and ADIE = 1, an ADI interrupt is requested.
4. The A/D interrupt service routine is started.
5. The routine reads ADF set to 1, then writes 0 in ADF.
6. The routine reads and processes the conversion result (ADDR0).
7. Execution of the A/D interrupt service routine ends. After this, if the ADST bit is set to 1, A/D conversion starts again and steps 2 to 7 are repeated.



Note: Vertical arrows (↓) indicate instructions executed by software.

Figure 13.3 Example of A/D Converter Operation (Single Mode, Channel 1 Selected)

### 13.4.2 Scan Mode (SCAN = 1)

Scan mode is useful for monitoring analog inputs in a group of one or more channels. When the ADST bit in the A/D control/status register (ADCSR) is set to 1 by software or MTU trigger input, A/D conversion starts on the first channel in the group (AN0 when CH2 = 0; AN4 when CH1 = 1).

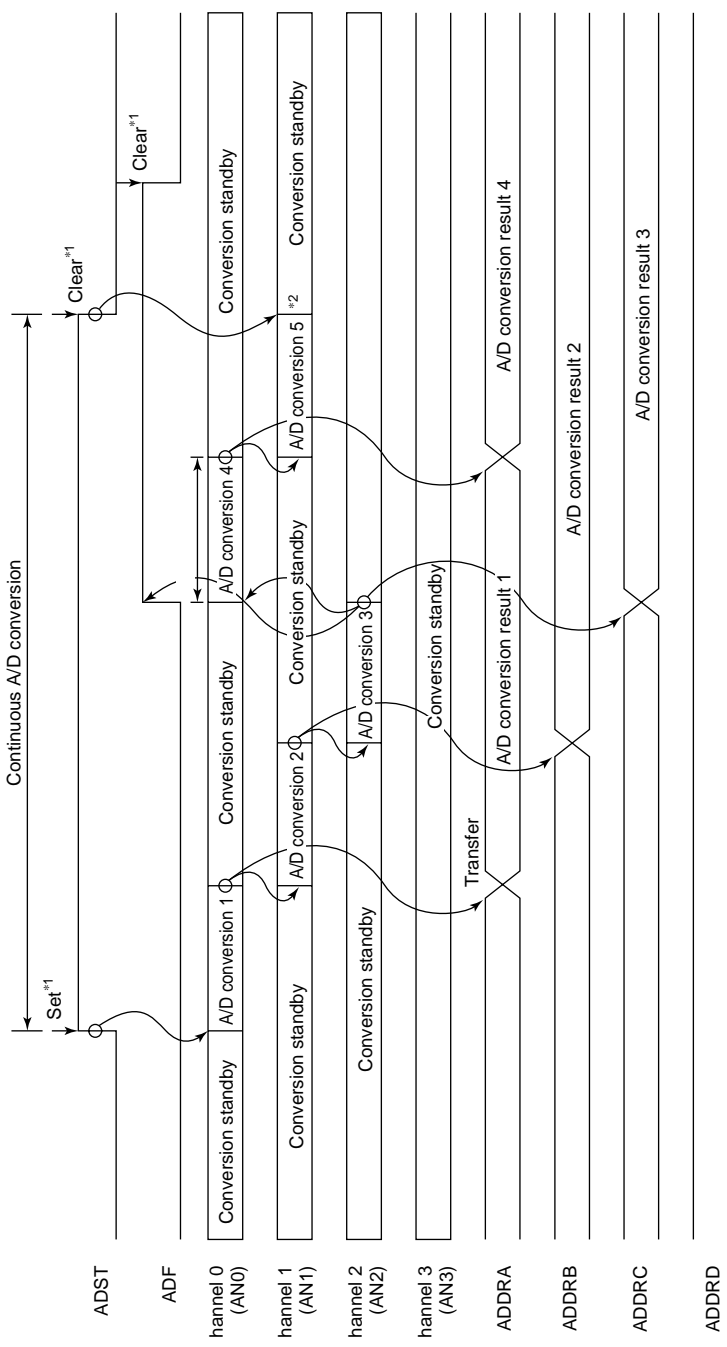
When more than one channel has been selected, A/D conversion starts on the second channel (AN1 or AN5) as soon as conversion ends on the first channel.

A/D conversion is performed repeatedly on all the selected channels until the ADST bit is cleared to 0. The conversion results are transferred to and stored in the ADDR register for each channel.

To prevent incorrect operation, A/D conversion should be halted by clearing the ADST bit to 0 before changing the mode or analog input channels. After the change is made, the first channel is selected and A/D conversion is restarted by setting the ADST bit to 1 (the mode or channel change and setting of the ADST bit can be carried out simultaneously).

An example of the A/D conversion operation in scan mode when three channels (AN0–AN2) in group 0 are selected is described below. Figure 13.4 shows a timing diagram for this example.

1. Scan mode is selected (SCAN = 1), group 0 is selected as the scan group (CH2 = 0), analog input channels AN0-AN2 are selected (CH1 = 1, CH0 = 0), and A/D conversion is started (ADST = 1).
2. A/D conversion starts on the first channel (AN0), and when completed, the result is transferred to ADDR. Next, conversion of the second channel (AN1) starts automatically.
3. Conversion proceeds in the same way through the third channel (AN2).
4. When conversion is completed for all the selected channels (AN0–AN2), ADIF is set to 1, the first channel (AN0) is selected again, and conversion is performed on that channel. If the ADIE bit is also 1, an ADI interrupt is requested when conversion is completed.
5. Steps 2 to 4 are repeated as long as the ADST bit remains set to 1. When the ADST bit is cleared to 0, A/D conversion stops. After this, if the ADST bit is set to 1, A/D conversion starts again from the first channel (AN0).



- Notes:
1. Vertical arrows (↓) indicate instructions executed by software.
  2. Data currently being converted is ignored.

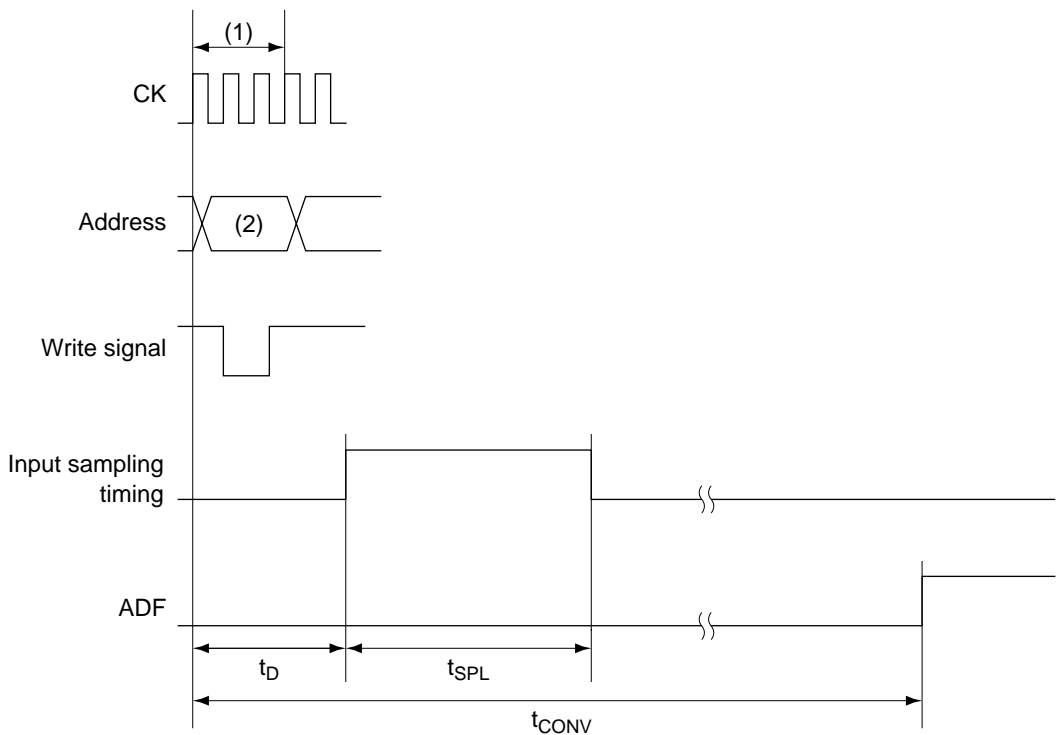
Figure 13.4 Example of A/D Converter Operation (Scan Mode, Channels AN0–AN2 Selected)

### 13.4.3 Input Sampling and A/D Conversion Time

The A/D converter has a built-in sample and hold circuit. The A/D converter samples the analog input at time  $t_D$  after the ADST bit is set to 1 in the A/D control/status register (ADCSR), then starts conversion. Figure 13.5 shows the A/D conversion timing, and table 13.4 shows A/D conversion times.

As shown in figure 13.5, A/D conversion time  $t_{CONV}$  consists of A/D conversion start delay time  $t_D$  and analog input sampling time  $t_{SPL}$ . The length of  $t_D$  is not fixed, but is determined by the timing of the write to ADSCR. The total conversion time therefore varies within the ranges shown in table 13.4.

In scan mode, the  $t_{CONV}$  values given in table 13.4 apply to the first conversion. In the second and subsequent conversions,  $t_{CONV}$  is fixed at 256 states when  $CKS = 0$  or 128 states when  $CKS = 1$ .



- (1): ADCSR write cycle
- (2): ADCSR address
- $t_D$ : A/D conversion start delay time
- $t_{SPL}$ : Input sampling time
- $t_{CONV}$ : A/D conversion time

**Figure 13.5 A/D Conversion Timing**

**Table 13.4 A/D Conversion Times (Single Mode)**

	Symbol	CKS = 0			CKS = 1		
		Min	Typ	Max	Min	Typ	Max
A/D conversion start delay time	$t_D$	10	—	17	6	—	9
Input sampling time	$t_{SPL}$	—	64	—	—	32	—
A/D conversion time	$t_{CCNV}$	259	—	266	131	—	134

Note: Unit: states ( $t_{cyc}$ )

### 13.4.4 MTU Trigger Input Timing

A/D conversion can also be started by MTU trigger input. When the TRGE bit is set to 1 in the A/D control register (ADCR), input from the MTU functions as trigger input. When an MTU trigger is detected, the ADST bit is set to 1 in the A/D control/status register (ADST), and the A/D converter is started.

Other operations, for both single mode and scan mode, are the same as when the ADST bit is set to 1 by software, .

Figure 13.6 shows the timing for MTU trigger input.

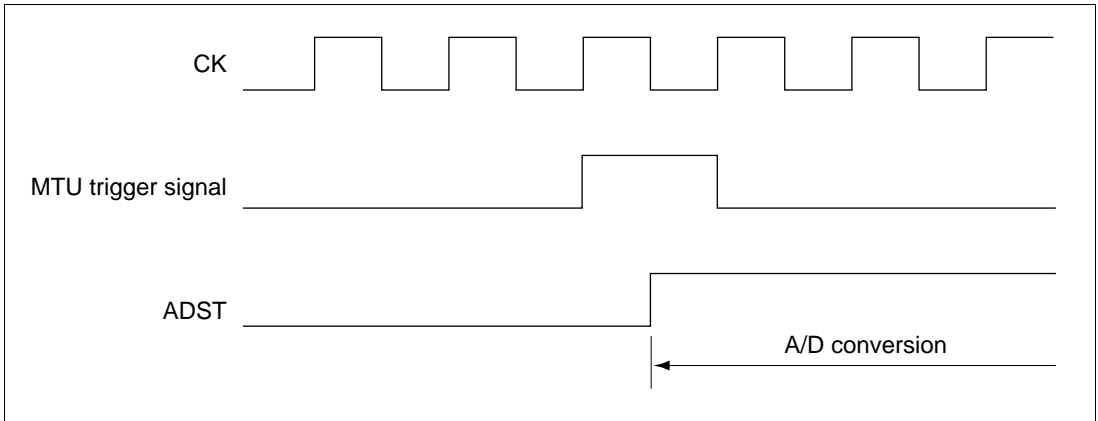


Figure 13.6 External Trigger Input Timing

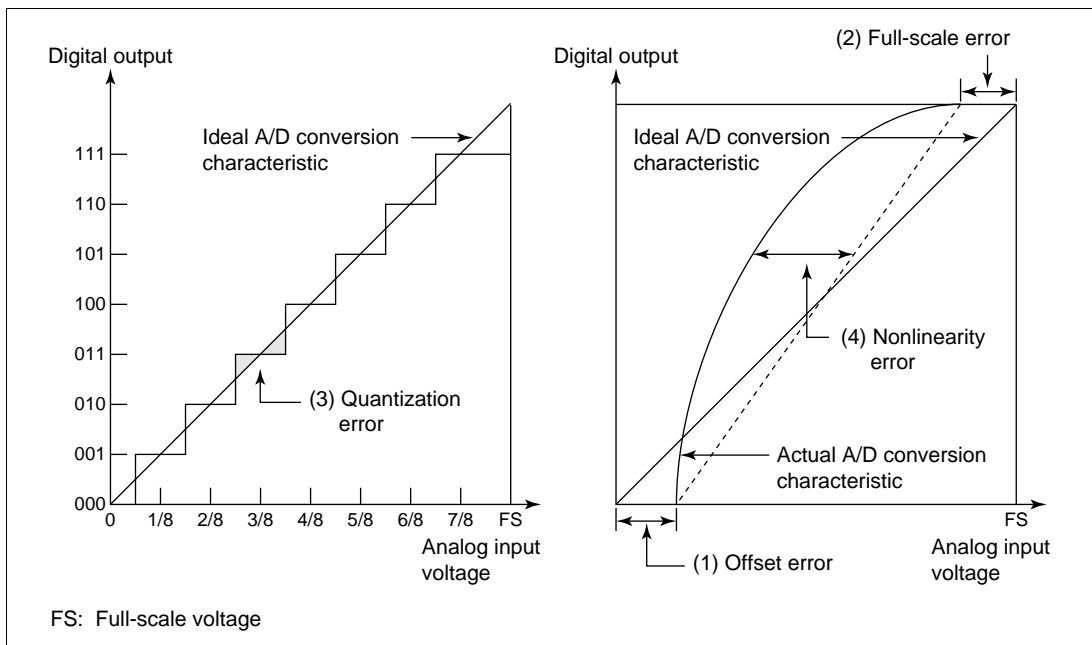
### 13.5 A/D Conversion Precision Definitions

The A/D converter converts analog values input from analog input channels to 10-bit digital values by comparing them with an analog reference voltage. In this operation, the absolute precision of the A/D conversion (i.e. the deviation between the input analog value and the output digital value) includes the following kinds of error.

1. Offset error
2. Full-scale error
3. Quantization error
4. Nonlinearity error



The above four kinds of error are described below with reference to figure 13.7. For the sake of clarity, this figure shows 3-bit A/D conversion rather than 10-bit A/D conversion. Offset error (see figure 13.7 (1)) is the deviation between the actual A/D conversion characteristic and the ideal A/D conversion characteristic when the digital output value changes from the minimum value (zero voltage) of 0000000000 (000 in the figure) to 0000000001 (001 in the figure). Full-scale error (see figure 13.7 (2)) is the deviation between the actual A/D conversion characteristic and the ideal A/D conversion characteristic when the digital output value changes from 1111111110 (110 in the figure) to the maximum value (full-scale voltage) of 1111111111 (111 in the figure). Quantization error is the deviation inherent in the A/D converter, given by 1/2 LSB (see figure 13.7 (3)). Nonlinearity error is the deviation between the actual A/D conversion characteristic and the ideal A/D conversion characteristic from zero voltage to full-scale voltage (see figure 13.7 (4)). This does not include offset error, full-scale error, and quantization error.



**Figure 13.7 A/D Conversion Precision Definitions**

## 13.6 Notes on Use

The following points should be noted when using the A/D converter.

### 13.6.1 Analog Voltage Settings

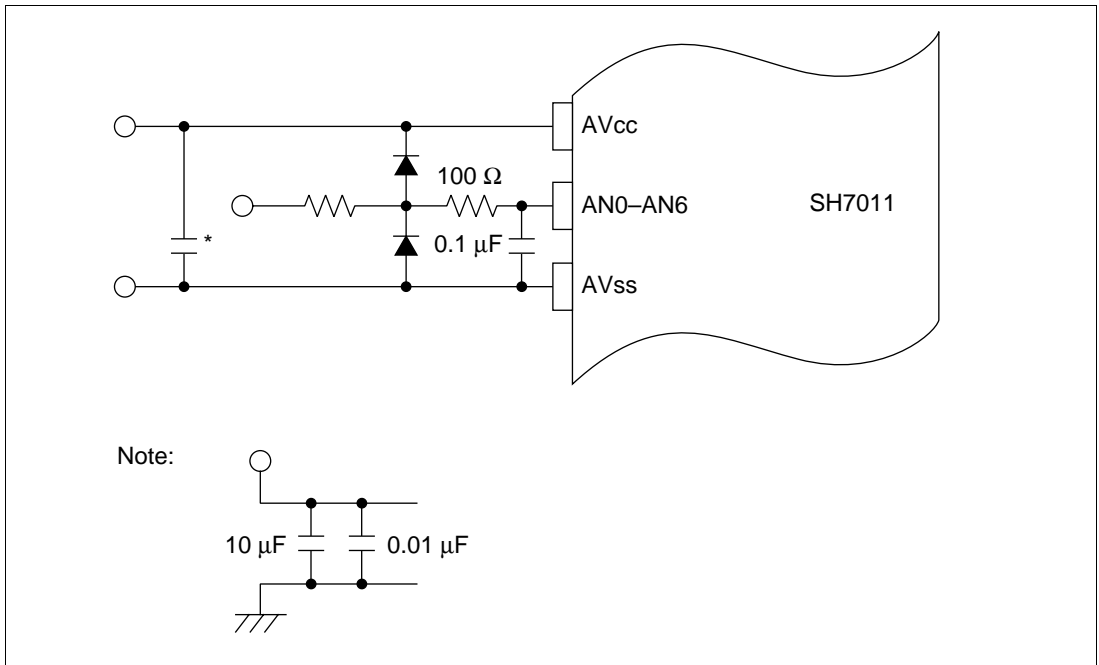
**Analog Input Voltage Range:** The voltage applied to analog input pins during A/D conversion should be in the range  $AV_{SS} \leq AN_n \leq AV_{CC}$  ( $n = 0$  to  $6$ ).

**$AV_{CC}$  and  $AV_{SS}$  input voltages:** For the  $AV_{CC}$  and  $AV_{SS}$  input voltages, set  $AV_{CC} = 3.3 \text{ V} \pm 10\%$ , and  $AV_{SS} = V_{SS}$ . When the A/D converter is not used, set  $AV_{CC} = V_{CC}$  and  $AV_{SS} = V_{SS}$ .

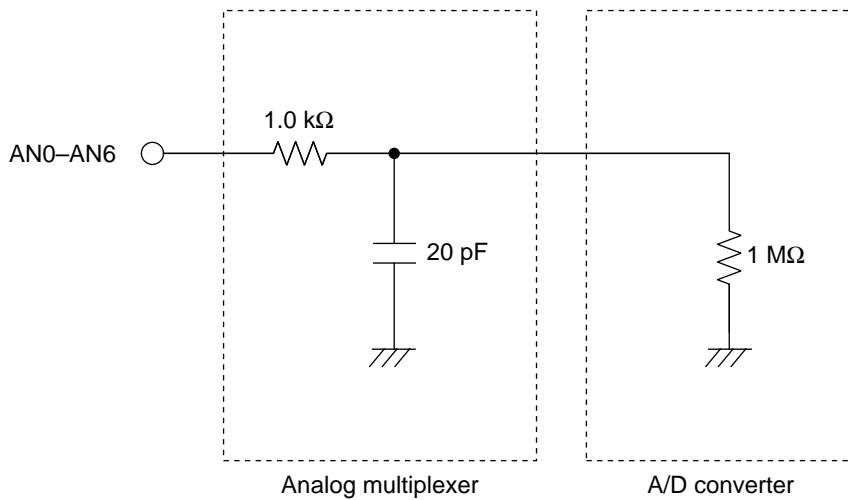
### 13.6.2 Handling of Analog Input Pins

To prevent damage from surges and other abnormal voltages at the analog input pins (AN0–AN6), connect a protection circuit such as that shown in figure 13.8. This circuit also includes a CR filter function that suppresses error due to noise. The circuit shown here is only a design example; circuit constants must be decided on the basis of the actual operating conditions.

Figure 13.9 shows an equivalent circuit for the analog input pins, and table 13.5 summarizes the analog input pin specifications.



**Figure 13.8 Example of Analog Input Pin Protection Circuit**



Note: Values are reference values.

**Figure 13.9 Analog Input Pin Equivalent Circuit**

**Table 13.5 Analog Input Pin Specifications**

Item	Min	Max	Unit
Analog input capacitance	—	20	pF
Permitted signal source impedance	—	3	k

# Section 14 Pin Function Controller

## 14.1 Overview

The pin function controller (PFC) is composed of registers for selecting the function of multiplexed pins and the direction of input/output. Table 14.1 lists the SH7011's multiplexed pins. These multiplexed pins are initially set as input ports after a power-on reset.

**Table 14.1 Multiplexed Pins**

Port	Function 1 (Related Module)	Function 2 (Related Module)	Pin
E	PE7 I/O (port)	TIOC2B I/O (MTU)	98
E	PE6 I/O (port)	TIOC2A I/O (MTU)	83
E	PE5 I/O (port)	TIOC1B I/O (MTU)	82
E	PE4 I/O (port)	TIOC1A I/O (MTU)	81
E	PE2 I/O (port)	TIOC0C I/O (MTU)	79
E	PE0 I/O (port)	TIOC0A I/O (MTU)	78

## 14.2 Register Configuration

Table 14.2 summarizes the registers of the pin function controller.

**Table 14.2 Register Configuration**

Name	Abbreviation	R/W	Initial Value	Address	Access Size
Port A I/O register H	PAIORH	R/W	H'0000	H'FFFF8384 H'FFFF8385	8, 16
Port E I/O register	PEIOR	R/W	H'0000	H'FFFF83B4 H'FFFF83B5	8, 16
Port E control register 2	PECR2	R/W	H'0000	H'FFFF83BA H'FFFF83BB	8, 16

## 14.3 Register Descriptions

### 14.3.1 Port A I/O Register H (PAIORH)

Port A I/O register H (PAIORH) is a 16-bit read/write register that selects input or output for pins PA19 and PA18 in port A. A pin in port A is an output pin if its corresponding PAIORH bit is set to 1, and an input pin if the bit is cleared to 0.

PAIORH is initialized to H'0000 by a power-on reset; however, it is not initialized in sleep mode, so the previous data is maintained.

Bit:	15	14	13	12	11	10	9	8
	—	—	—	—	—	—	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R

Bit:	7	6	5	4	3	2	1	0
	—	—	—	—	PA19 IOR	PA18 IOR	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R/W	R/W	R	R

### 14.3.2 Port E I/O Register (PEIOR)

The port E I/O register (PEIOR) is a 16-bit read/write register that selects input or output for nine pins in port E. Bits PE17IOR–PE4IOR, PE2IOR, and PE0IOR correspond to multiplexed pins.

When the port E pin functions are PEX, a pin in port E is an output pin if its corresponding PEIOR bit is set to 1, and an input pin if the bit is cleared to 0.

PEIOR is initialized to H'0000 by a power-on reset; however, it is not initialized in sleep mode, so the previous data is maintained.

Bit:	15	14	13	12	11	10	9	8
	—	PE14 IOR	PE13 IOR	PE12 IOR	—	—	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R/W	R/W	R/W	R	R	R	R

Bit:	7	6	5	4	3	2	1	0
	PE7 IOR	PE6 IOR	PE5 IOR	PE4 IOR	—	PE2 IOR	—	PE0 IOR
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R	R/W	R	R/W

### 14.3.3 Port E Control Register 2 (PECR2)

Port E control register 2 (PECR2) is a 16-bit read/write register that selects the functions of the six multiplexed pins in port E.

PECR2 is initialized to H'0000 by a power-on reset; however, it is not initialized in sleep mode, so the previous data is maintained.

Bit:	15	14	13	12	11	10	9	8
	—	PE7MD	—	PE6MD	—	PE5MD	—	PE4MD
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R/W	R	R/W	R	R/W	R	R/W

Bit:	7	6	5	4	3	2	1	0
	—	—	—	PE2 MD0	—	—	—	PE0 MD0
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R/W	R	R	R	R/W

- Bit 15—Reserved: This bit always reads as 0. The write value should always be 0.
- Bit 14—PE7 Mode (PE7MD): Selects the function of the PE7/TIOC2B pin.

#### Bit 14: PE7MD Description

0	General input/output (PE7) (initial value)
1	MTU input capture input/output compare output (TIOC2B)

- Bit 13 —Reserved: This bit always reads as 0. The write value should always be 0.

- Bit 12—PE6 Mode (PE6MD): Selects the function of the PE6/TIOC2A pin.

Bit 12: PE6MD	Description
---------------	-------------

0	General input/output (PE6) (initial value)
1	MTU input capture input/output compare output (TIOC2A)

- Bit 11—Reserved: This bit always reads as 0. The write value should always be 0.
- Bit 10—PE5 Mode (PE5MD): Selects the function of the PE5/TIOC1B pin.

Bit 10: PE5MD	Description
---------------	-------------

0	General input/output (PE5) (initial value)
1	MTU input capture input/output compare output (TIOC1B)

- Bit 9—Reserved: This bit always reads as 0. The write value should always be 0.
- Bit 8—PE4 Mode (PE4MD): Selects the function of the PE4/TIOC1A pin.

Bit 8: PE4MD	Description
--------------	-------------

0	General input/output (PE4) (initial value)
1	MTU input capture input/output compare output (TIOC1A)

- Bits 7–5—Reserved. These bits always read 0. The write value should always be 0.
- Bit 4—PE2 Mode 0 (PE2MD0): Selects the function of the PE2/TIOC0C pin.

Bit 4: PE2MD0	Description
---------------	-------------

0	General input/output (PE2) (initial value)
1	MTU input capture input/output compare output (TIOC0C)

- Bits 3–1—Reserved. These bits always read 0. The write value should always be 0.
- Bit 0—PE0 Mode 0 (PE0MD0): Selects the function of the PE0/TIOC0A pin.

Bit 0: PE0MD0	Description
---------------	-------------

0	General input/output (PE0) (initial value)
1	MTU input capture input/output compare output (TIOC0A)

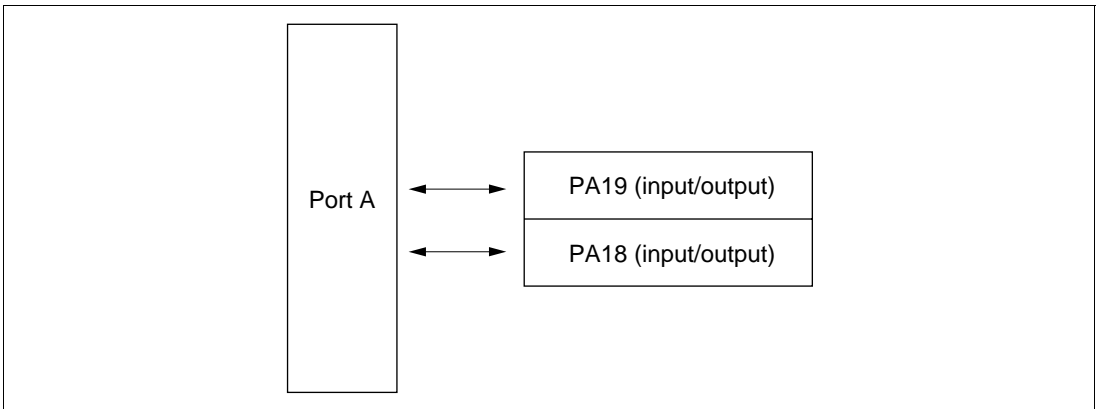
# Section 15 I/O Ports (I/O)

## 15.1 Overview

There are two ports, A and E. The pin function controller (PFC) is used to select the function of multiplexed pins. The ports each have one data register for storing pin data. All pins are initially set as input port pins after a power-on reset.

## 15.2 Port A

Port A is a 2-pin input/output port, as shown in figure 15.1.



**Figure 15.1 Port A**

### 15.2.1 Register Configuration

Table 15.1 summarizes the port A register.

**Table 15.1 Port A Register**

Name	Abbreviation	R/W	Initial Value	Address	Access Size
Port A data register H	PADRH	R/W	H'0000	H'FFFF8380 H'FFFF8381	8, 16



## 15.2.2 Port A Data Register H (PADRH)

PADRH is a 16-bit read/write register that stores data for port A. The bits PA19DR–PA18DR correspond to the PA19 pin and PA18 pin. When the pins are used as ordinary outputs, they will output whatever value is written in the PADRH; when PADRH is read, the register value will be output regardless of the pin status. When the pins are used as ordinary inputs, the pin status rather than the register value is read directly when PADRH is read. When a value is written to PADRH, that value can be written into PADRH, but it will not affect the pin status. Table 15.2 shows the read/write operations of the port A data register.

PADRH is initialized by an external power-on reset. However, PADRH is not initialized in sleep mode, so the previous data is maintained.

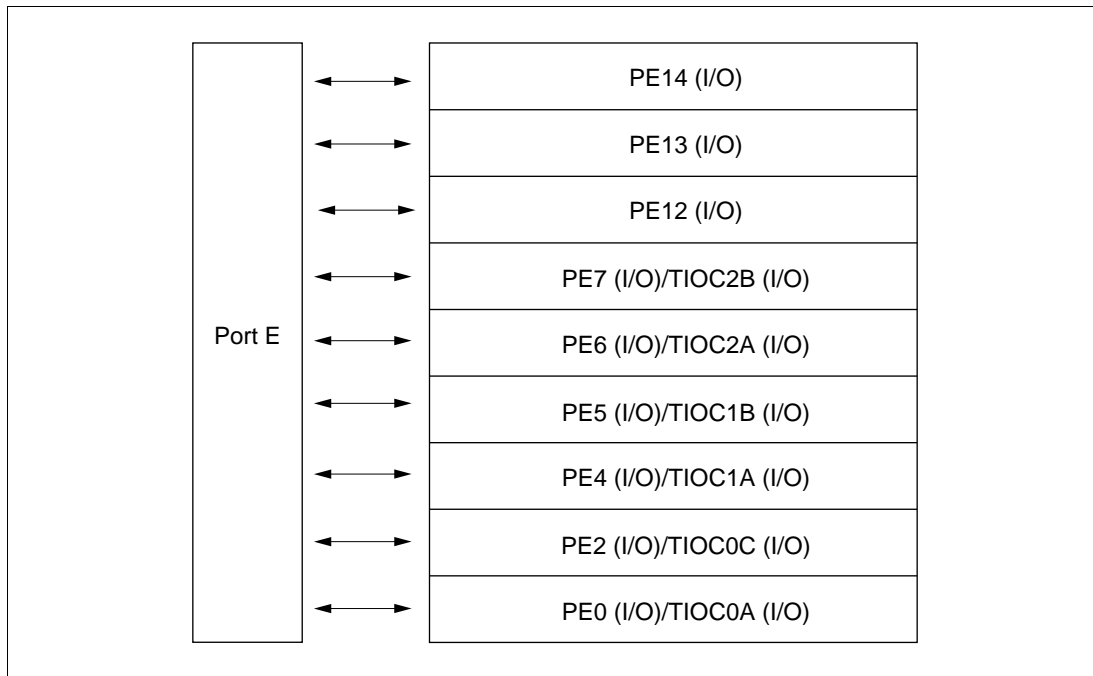
Bit:	15	14	13	12	11	10	9	8
	—	—	—	—	—	—	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R
Bit:	7	6	5	4	3	2	1	0
	—	—	—	—	PA19DR	PA18DR	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R/W	R/W	R	R

**Table 15.2 Port A Data Register (PADR) Read/Write Operations**

PAIOR	Pin Function	Read	Write
0	Input	Pin status	Can write to PADR, but it has no effect on pin status
1	Output	PADR value	Value written is output by pin

## 15.3 Port E

Port E is a 9-pin input/output port, as shown in table 15.2.



**Figure 15.2 Port E**

### 15.3.1 Register Configuration

Table 15.3 summarizes the port E register.

**Table 15.3 Port E Register**

Name	Abbreviation	R/W	Initial Value	Address	Access Size
Port E data register	PEDR	R/W	H'0000	H'FFFF83B0 H'FFFF83B1	8, 16

### 15.3.2 Port E Data Register (PEDR)

PEDR is a 16-bit read/write register that stores data for port E. Table 15.5 shows the correspondence between PEDR bits and port E pins. When the pins are used as ordinary outputs, they will output whatever value is written in the PEDR; when PEDR is read, the register value will be read regardless of the pin status. When the pins are used as ordinary inputs, the pin status rather than the register value is read directly when PEDR is read. When a value is written to PEDR, that value can be written into PEDR, but it will not affect the pin status. Table 15.4 shows the read/write operations of the port E data register.

PEDR is initialized by a power-on reset. However, PEDR is not initialized for a sleep mode, so the previous data is retained.

Bit:	15	14	13	12	11	10	9	8
	—	PE14DR	PE13DR	PE12DR	—	—	—	—
Initial value:	0	0	0	0	0	0	0	0
R/W:	R	R/W	R/W	R/W	R	R	R	R
Bit:	7	6	5	4	3	2	1	0
	PE7DR	PE6DR	PE5DR	PE4DR	—	PE2DR	—	PE0DR
Initial value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R	R/W	R	R/W

**Table 15.4 Read/Write Operation of the Port E Data Register (PEDR)**

PEIOR	Pin Status	Read	Write
0	Ordinary input	Pin status	Can write to PEDR, but it has no effect on pin status
	Other function	Pin status	Can write to PEDR, but it has no effect on pin status
1	Ordinary output	PEDR value	Value written is output by pin
	Other function	PEDR value	Can write to PEDR, but it has no effect on pin status

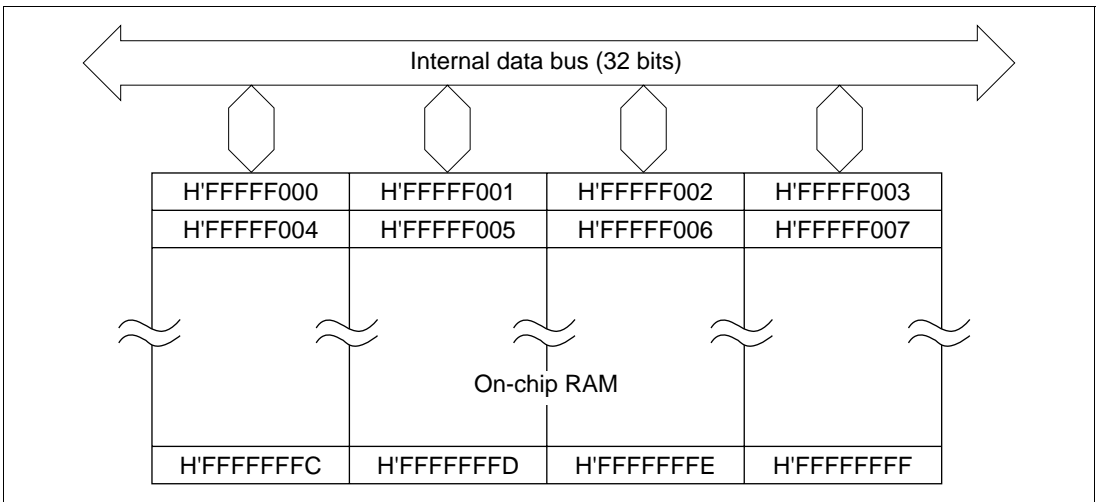
**Table 15.5 Correspondence between Port E Data Register (PEDR) Bits and Port E Pins**

<b>PEDR Bit</b>	<b>Port E Pin</b>
PE14DR	PE14
PE13DR	PE13
PE12DR	PE12
PE7DR	PE7/TIOC2B
PE6DR	PE6/TIOC2A
PE5DR	PE5/TIOC1B
PE4DR	PE4/TIOC1A
PE2DR	PE2/TIOC0C
PE0DR	PE0/TIOC0A

# Section 16 RAM

## 16.1 Overview

The SH7011 has 4 kbytes of on-chip RAM. The on-chip RAM is connected to the CPU by a 32-bit data bus (figure 16.1). The CPU can access data in the on-chip RAM in 8, 16, or 32 bit widths. On-chip RAM data can always be accessed in one state, making the RAM ideal for use as a program area, stack area, or data area, which require high-speed access. The contents of the on-chip RAM are held in sleep mode. Memory area addresses H'FFFFFFE000 to H'FFFFFFF000 are allocated to the on-chip RAM.



**Figure 16.1 Block Diagram of RAM**

## 17.1 Absolute Maximum Ratings

**Table 17.1 Absolute Maximum Ratings**

Item	Symbol	Rating	Unit
Power supply voltage	$V_{CC}$	-0.3 to +7.0	V
Input voltage (other than A/D ports)	$V_{in}$	-0.3 to $V_{CC} + 0.3$	V
Input voltage (A/D ports)	$V_{in}$	-0.3 to $AV_{CC} + 0.3$	V
Analog supply voltage	$AV_{CC}$	-0.3 to +7.0	V
Analog input voltage	$V_{AN}$	-0.3 to $AV_{CC} + 0.3$	V
Operating temperature	$T_{opr}$	-20 to +75	°C
Storage temperature	$T_{stg}$	-55 to +125	°C

Note: Operating the LSI in excess of the absolute maximum ratings may result in permanent damage.

## 17.2 DC Characteristics

**Table 17.2 DC Characteristics (Conditions:  $V_{CC} = 3.0$  to  $3.6V$ ,  $AV_{CC} = 3.0$  to  $5.5V$ ,  $AV_{CC} V_{CC}, V_{SS} = AV_{SS} = 0V$ ,  $T_a = D20$  to  $+75^{\circ}C$ )**

Item	Pin	Symbol	Min	Typ	Max	Measurement	
						Unit	Conditions
Input high-level voltage	$\overline{RES}$ , NMI, PE0, PE2, PE4 to PE7, PE12 to PE14	$V_{IH}$	$V_{CC} - 0.7$	—	$V_{CC} + 0.3$	V	
	EXTAL		$V_{CC} \times 0.7$	—	$V_{CC} + 0.3$	V	
	A/D port		2.2	—	$AV_{CC} + 0.3$	V	
	Other input pins		2.2	—	$V_{CC} + 0.3$	V	
Input low-level voltage	$\overline{RES}$ , NMI, PE0, PE2, PE4 to PE7, PE12 to PE14	$V_{IL}$	-0.3	—	$V_{CC} \times 0.1$	V	
	Other input pins		-0.3	—	$V_{CC} \times 0.2$	V	
Schmitt trigger input voltage	PE0, PE2, PE4 to PE7, PE12 to PE14	$V_T^+$	$V_{CC} \times 0.9$	—	—	V	
		$V_T^-$	—	—	$V_{CC} \times 0.2$	V	
		$V_T^+ - V_T^-$	$V_{CC} \times 0.07$	—	—	V	
Input leak current	$\overline{RES}$ , NMI, PE0, PE2, PE4 to PE7, PE12 to PE14	$ I_{in} $	—	—	1.0	$\mu A$	$V_{in} = 0.5$ to $V_{CC} - 0.5V$
	A/D port		—	—	1.0	$\mu A$	$V_{in} = 0.5$ to $AV_{CC} - 0.5V$
	Other input pins		—	—	1.0	$\mu A$	$V_{in} = 0.5$ to $V_{CC} - 0.5V$
Three-state leak current (while off)	A21–A1, D15–D0, CS3–CS0, WRx, RD, Port A, E	$ I_{TSI} $	—	—	1.0	$\mu A$	$V_{in} = 0.5$ to $V_{CC} - 0.5V$

**Table 17.2 DC Characteristics (Conditions:  $V_{CC} = 3.0$  to  $3.6V$ ,  $AV_{CC} = 3.0$  to  $5.5V$ ,  $AV_{CC} V_{CC}$ ,  $V_{SS} = AV_{SS} = 0V$ ,  $T_a = \text{D20 to } +75^{\circ}C$ ) (cont)**

Item	Pin	Symbol	Min	Typ	Max	Unit	Measurement Conditions
Output high-level voltage	All output pins	$V_{OH}$	$V_{CC} - 0.5$	—	—	V	$I_{OH} = -200 \mu A$
			$V_{CC} - 1.0$	—	—	V	$I_{OH} = -1mA$
Output low-level voltage	All output pins	$V_{OL}$	—	—	0.4	V	$I_{OL} = 1.6mA$
Input capacitance	$\overline{RES}$	$C_{in}$	—	—	80	pF	$V_{in} = 0V$
	NMI		—	—	50	pF	$f = 1 \text{ MHz}$
	All other input pins		—	—	20	pF	$T_a = 25^{\circ}C$
Current consumption	During normal operations	$I_{CC}$	—	80	130	mA	$f = 20MHz$
	During sleep mode		—	70	110	mA	$f = 20MHz$
Analog supply current		$AI_{CC}$	—	4	8	mA	$f = 20MHz$

- Notes:
1. Do not release  $AV_{CC}$  and  $AV_{SS}$  pins when not using the A/D converter. Connect  $AV_{CC}$  pin to  $V_{CC}$  and  $AV_{SS}$  pin to  $V_{SS}$ .
  2. The value for consumed current is with conditions of  $V_{IHmin} = V_{CC} - 0.5V$  and  $V_{ILmax} = 0.5V$ , with no burden on any of the output pins.



**Table 17.3 Permitted Output Current Values (Conditions:  $V_{CC} = 3.0$  to  $3.6V$ ,  $AV_{CC} = 3.0$  to  $5.5V$ ,  $AV_{CC} V_{CC}$ ,  $V_{SS} = AV_{SS} = 0V$ ,  $T_a = \text{D20 to } +75^{\circ}C$ )**

Item	Symbol	Min	Typ	Max	Unit
Output low-level permissible current (per pin)	$I_{OL}$	—	—	2.0	mA
Output low-level permissible current (total)	$I_{OL}$	—	—	80	mA
Output high-level permissible current (per pin)	$-I_{OH}$	—	—	2.0	mA
Output high-level permissible current (total)	$(-I_{OH})$	—	—	25	mA

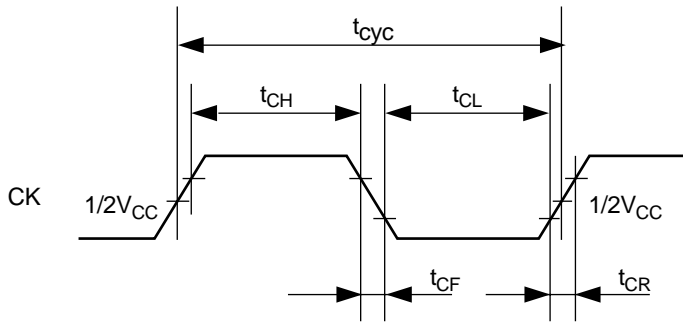
Note: To assure LSI reliability, do not exceed the output values listed in this table.

## 17.3 AC Characteristics

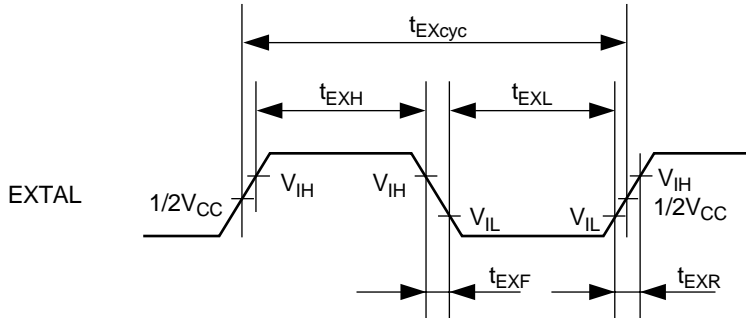
### 17.3.1 Clock Timing

**Table 17.4 Clock Timing (Conditions:  $V_{CC} = 3.0$  to  $3.6V$ ,  $AV_{CC} = 3.0$  to  $5.5V$ ,  $AV_{CC} V_{CC}$ ,  $V_{SS} = AV_{SS} = 0V$ ,  $T_a = \text{D20 to } +75^{\circ}C$ )**

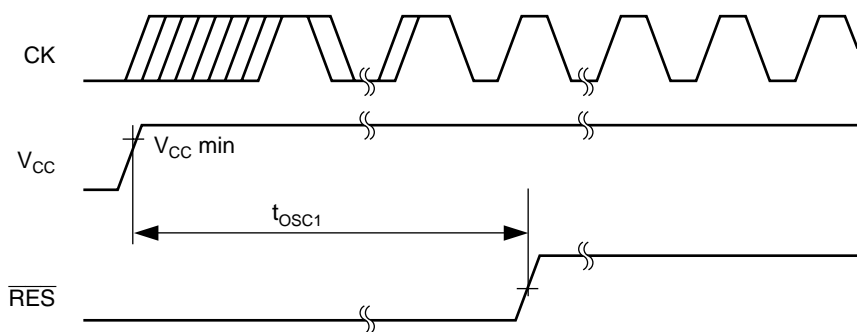
Item	Symbol	Min	Max	Unit	Figures
Operating frequency	$f_{OP}$	4	20	MHz	17.1
Clock cycle time	$t_{cyc}$	50	250	ns	
Clock low-level pulse width	$t_{CL}$	10	—	ns	
Clock high-level pulse width	$t_{CH}$	10	—	ns	
Clock rise time	$t_{CR}$	—	10	ns	
Clock fall time	$t_{CF}$	—	10	ns	17.2
EXTAL clock input frequency	$f_{EX}$	4	20	MHz	
EXTAL clock input cycle time	$t_{EXcyc}$	50	250	ns	
EXTAL clock low-level input pulse width	$t_{EXL}$	10	—	ns	
EXTAL clock high-level input pulse width	$t_{EXH}$	10	—	ns	
EXTAL clock input rise time	$t_{EXR}$	—	5	ns	
EXTAL clock input fall time	$t_{EXF}$	—	5	ns	
Reset oscillation settling time	$t_{OSC1}$	20	—	ms	17.3



**Figure 17.1 System Clock Timing**



**Figure 17.2 EXTAL Clock Input Timing**



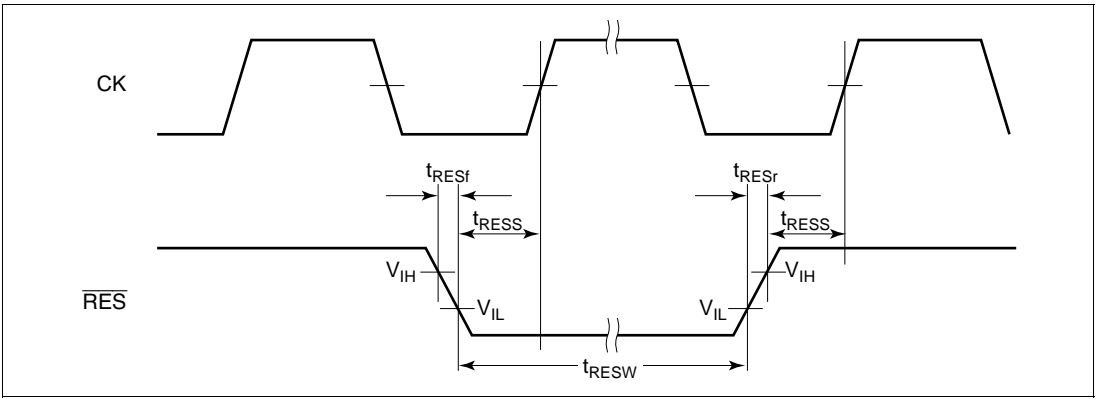
**Figure 17.3 Oscillation Settling Time**

### 17.3.2 Control Signal Timing

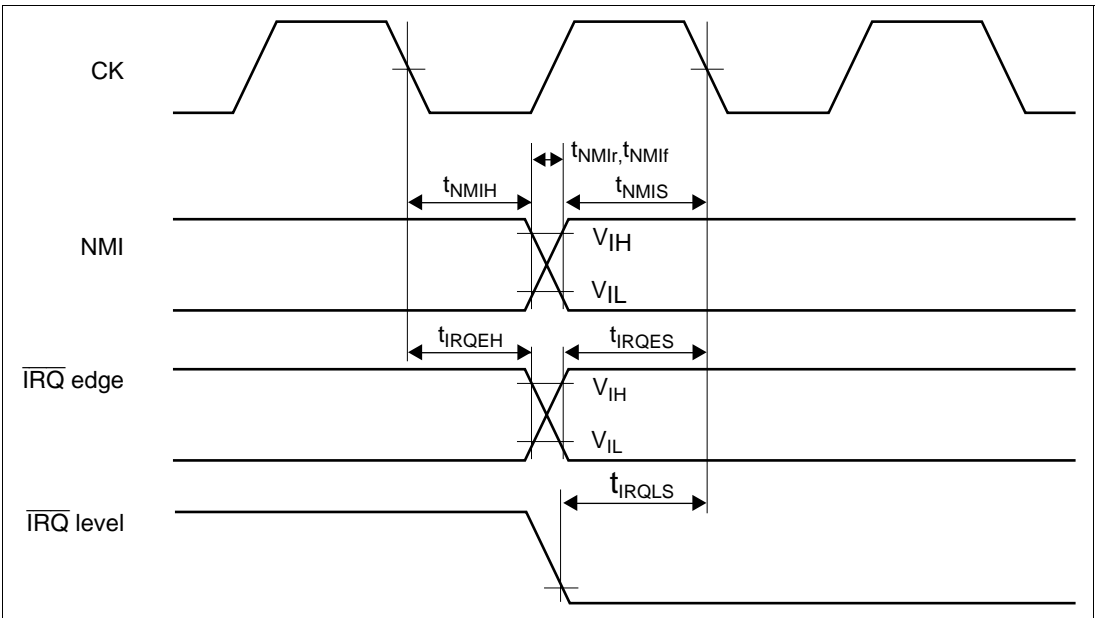
**Table 17.5 Control Signal Timing (Conditions:  $V_{CC} = 3.0$  to  $3.6V$ ,  $AV_{CC} = 3.0$  to  $5.5V$ ,  $AV_{CC} = V_{CC}$ ,  $V_{SS} = AV_{SS} = 0V$ ,  $T_a = \text{D20 to } +75^{\circ}\text{C}$ )**

Item	Symbol	Min	Max	Unit	Figure
$\overline{\text{RES}}$ rise/fall	$t_{\text{RESr}}, t_{\text{RESf}}$	—	200	ns	17.4
$\overline{\text{RES}}$ pulse width	$t_{\text{RESW}}$	40	—	$t_{\text{cyc}}$	
NMI rise/fall	$t_{\text{NMIr}}, t_{\text{NMIf}}$	—	200	ns	17.5
$\overline{\text{RES}}$ setup time *	$t_{\text{RESS}}$	100	—	ns	17.4
NMI setup time (during edge detection)	$t_{\text{NMIS}}$	100	—	ns	17.5
$\overline{\text{IRQ7}}\text{--}\overline{\text{IRQ0}}$ setup time (edge detection)	$t_{\text{IRQES}}$	100	—	ns	
$\overline{\text{IRQ7}}\text{--}\overline{\text{IRQ0}}$ setup time (level detection)	$t_{\text{IRQLS}}$	100	—	ns	
NMI hold time	$t_{\text{NMIH}}$	50	—	ns	17.5
$\overline{\text{IRQ7}}\text{--}\overline{\text{IRQ0}}$ hold time	$t_{\text{IRQEH}}$	50	—	ns	

Note: The  $\overline{\text{RES}}$ , NMI, and  $\overline{\text{IRQ7}}\text{--}\overline{\text{IRQ0}}$  signals are asynchronous inputs, but when the setup times shown here are provided, the signals are considered to have produced changes at clock rise (for  $\overline{\text{RES}}$ ) or clock fall (for NMI and  $\overline{\text{IRQ7}}\text{--}\overline{\text{IRQ0}}$ ). If the setup times are not provided, recognition is delayed until the next clock rise or fall.



**Figure 17.4 Reset Input Timing**



**Figure 17.5 Interrupt Signal Input Timing**

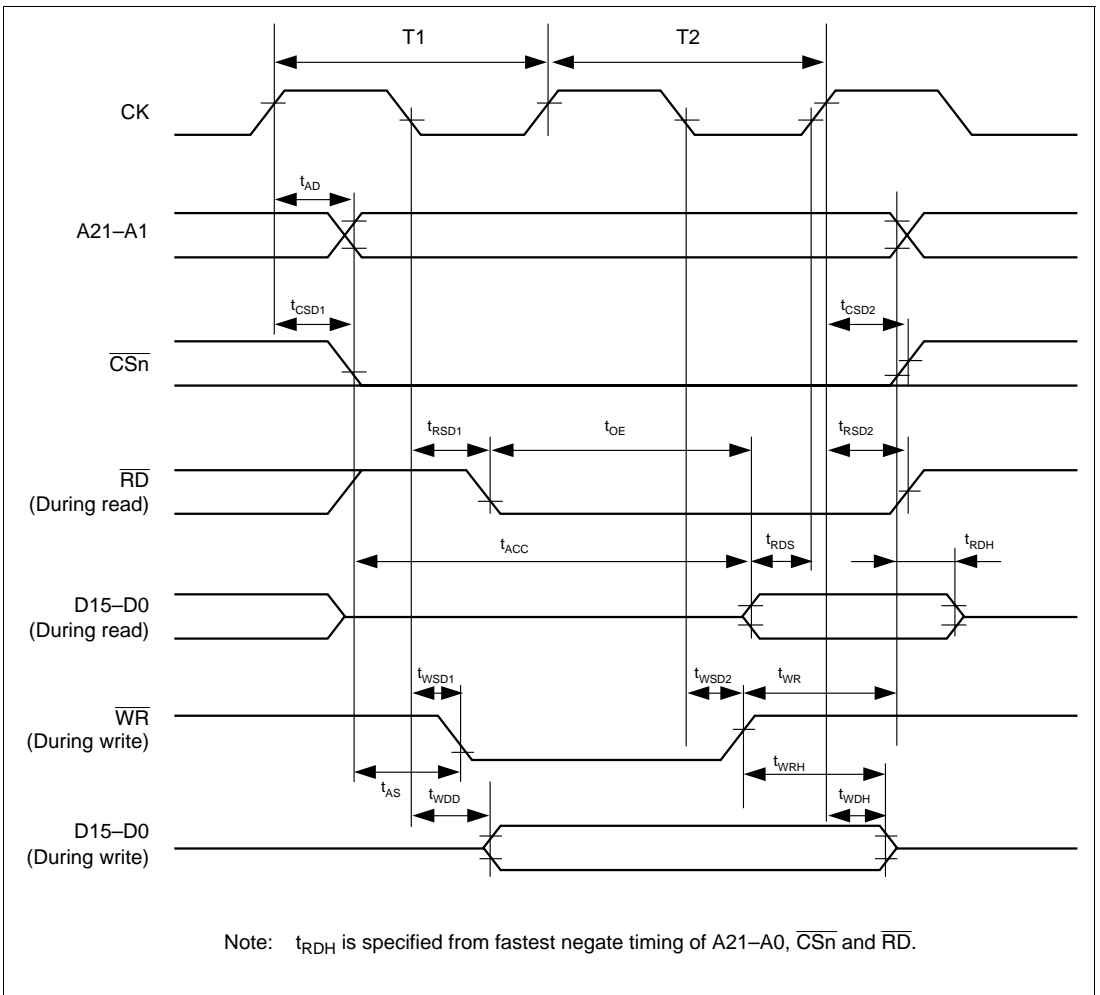
### 17.3.3 Bus Timing

**Table 17.6 Bus Timing (Conditions:  $V_{CC} = 3.0$  to  $3.6V$ ,  $AV_{CC} = 3.0$  to  $5.5V$ ,  $AV_{CC} = V_{CC}$ ,  $V_{SS} = AV_{SS} = 0V$ ,  $T_a = \text{D20 to } +75^{\circ}C$ )**

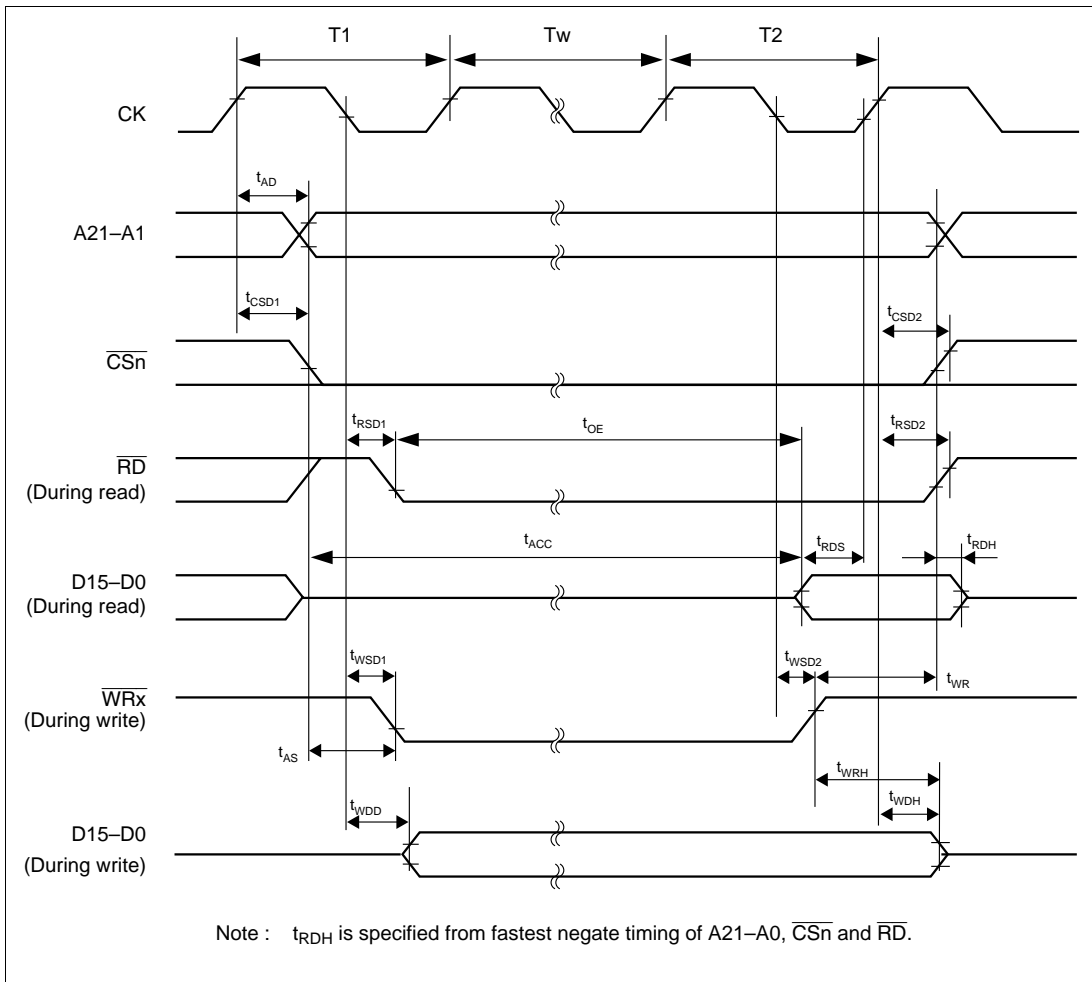
Item	Symbol	Min	Max	Unit	Figure
Address delay time	$t_{AD}$	$3^{*3}$	40	ns	17.6, 7
$\overline{CS}$ delay time 1	$t_{CSD1}$	$3^{*3}$	40	ns	
$\overline{CS}$ delay time 2	$t_{CSD2}$	$3^{*3}$	40	ns	
Read strobe delay time 1	$t_{RSD1}$	$3^{*3}$	40	ns	
Read strobe delay time 2	$t_{RSD2}$	$3^{*3}$	40	ns	
Read data setup time	$t_{RDS}^{*4}$	45	—	ns	
Read data hold time	$t_{RDH}$	0	—	ns	
Write strobe delay time 1	$t_{WSD1}$	$3^{*3}$	40	ns	
Write strobe delay time 2	$t_{WSD2}$	$3^{*3}$	40	ns	
Write data delay time	$t_{WDD}$	—	50	ns	
Write data hold time	$t_{WDH}$	0	$30^{*2}$	ns	
$\overline{WAIT}$ setup time	$t_{WTS}$	20	—	ns	17.8
$\overline{WAIT}$ hold time	$t_{WTH}$	0	—	ns	
Read data access time	$t_{ACC}^{*1 *5}$	$t_{cyc} \times (n+2) - 75$	—	ns	17.6, 7
Access time from read strobe	$t_{CE}^{*1}$	$t_{cyc} \times (n+1.5) - 75$	—	ns	
Write address setup time with respect to $\overline{WR}$ fall	$t_{AS}$	0	—	ns	
Write address hold time with respect to $\overline{WR}$ fall	$t_{WR}$	5	—	ns	
Write data hold time with respect to $\overline{WR}$ fall	$t_{WRH}$	0	—	ns	

Notes: n is the wait number.

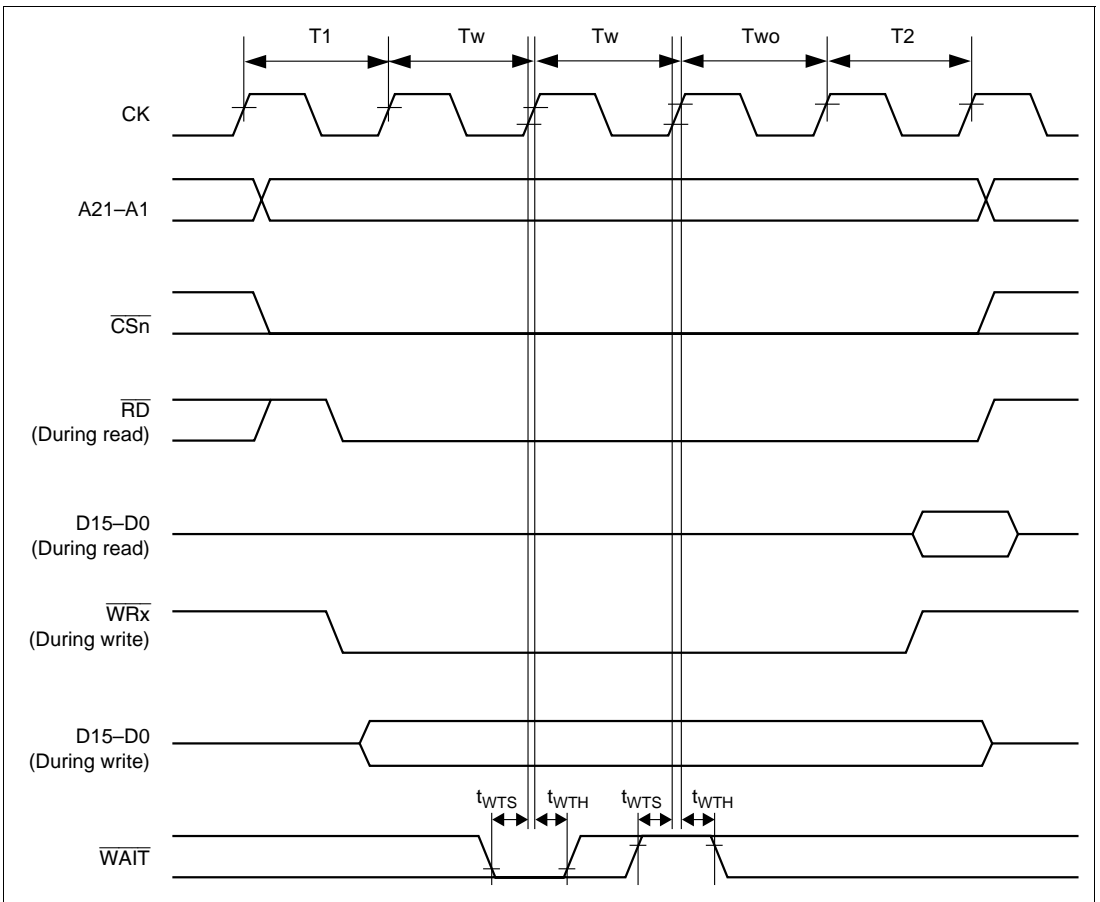
1. If the access time is satisfied, then the  $t_{RDS}$  need not be satisfied.
2.  $t_{WDH}$  (max) is a reference value.
3. The delay time min values are reference values (typ).
4.  $t_{RDS}$  is a reference value.
5. Depending on the operating frequency of the chip, there may be no memory that can be connected in no-wait mode. In this case, a wait should be inserted.



**Figure 17.6 Basic Cycle (No Waits)**



**Figure 17.7 Basic Cycle (Software Waits)**



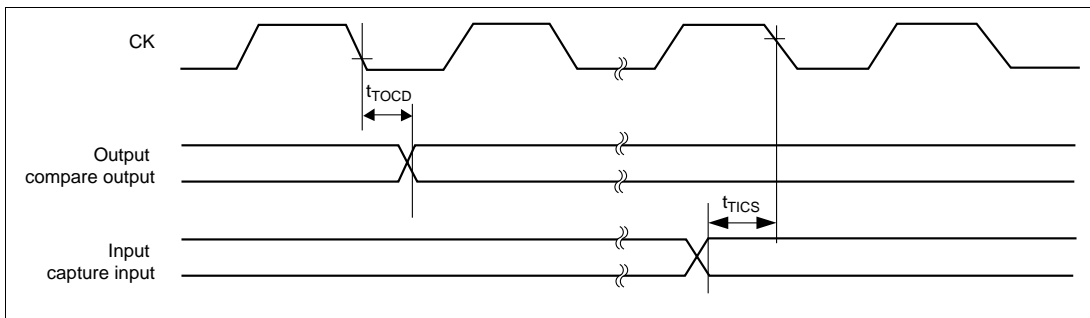
**Figure 17.8 Basic Cycle (2 Software Waits + Wait due to  $\overline{\text{WAIT}}$  Signal)**



### 17.3.4 Multifunction Timer Pulse Unit Timing

**Table 17.7 Multifunction Timer Pulse Unit Timing (Conditions:  $V_{CC} = 3.0$  to  $3.6V$ ,  $AV_{CC} = 3.0$  to  $5.5V$ ,  $AV_{CC} = V_{CC}$ ,  $V_{SS} = AV_{SS} = 0V$ ,  $T_a = \text{D20 to } +75^{\circ}C$ )**

Item	Symbol	Min	Max	Unit	Figure
Output compare output delay time	$t_{TOCD}$	—	100	ns	17.9
Input capture input setup time	$t_{TICS}$	100	—	ns	

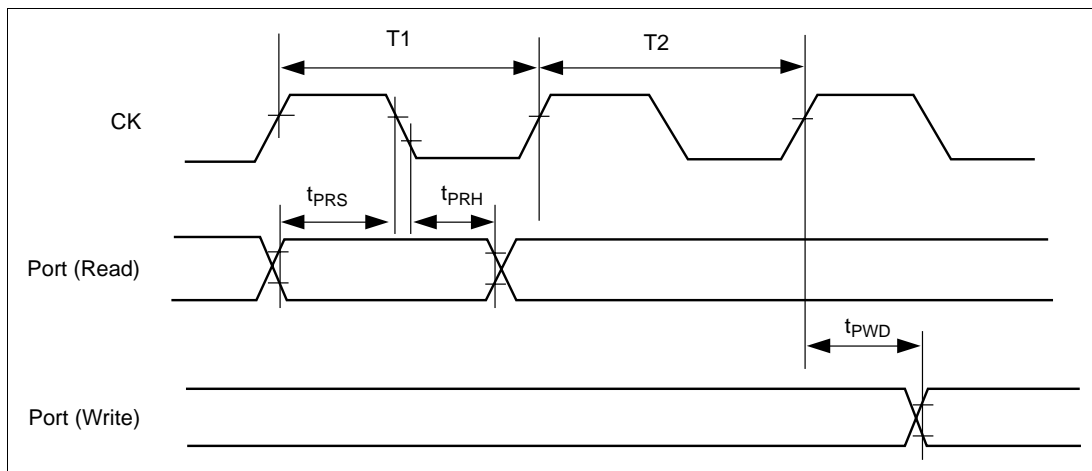


**Figure 17.9 MTU I/O Timing**

### 17.3.5 I/O Port Timing

**Table 17.8 I/O Port Timing (Conditions:  $V_{CC} = 3.0$  to  $3.6V$ ,  $AV_{CC} = 3.0$  to  $5.5V$ ,  $AV_{CC} = V_{CC}$ ,  $V_{SS} = AV_{SS} = 0V$ ,  $T_a = \text{Đ20 to } +75^{\circ}C$ )**

Item	Symbol	Min	Max	Unit	Figure
Port output data delay time	$t_{PWD}$	—	100	ns	17.10
Port input hold time	$t_{PRH}$	100	—	ns	
Port input setup time	$t_{PRS}$	100	—	ns	

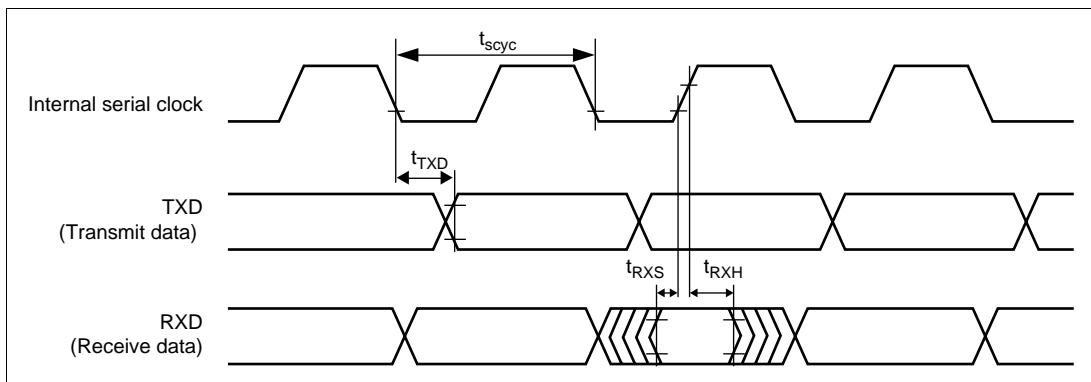


**Figure 17.10 I/O Port I/O Timing**

### 17.3.6 Serial Communication Interface Timing

**Table 17.9 Serial Communication Interface Timing (Conditions:  $V_{CC} = 3.0$  to  $3.6V$ ,  $AV_{CC} = 3.0$  to  $5.5V$ ,  $AV_{CC} = V_{CC}$ ,  $V_{SS} = AV_{SS} = 0V$ ,  $T_a = \text{Đ20 to } +75^{\circ}C$ )**

Item	Symbol	Min	Max	Unit	Figure
Transmit data delay time (clock sync)	$t_{TXD}$	—	100	ns	17.11
Receive data setup time (clock sync)	$t_{RXS}$	100	—	ns	
Receive data hold time (clock sync)	$t_{RXH}$	100	—	ns	

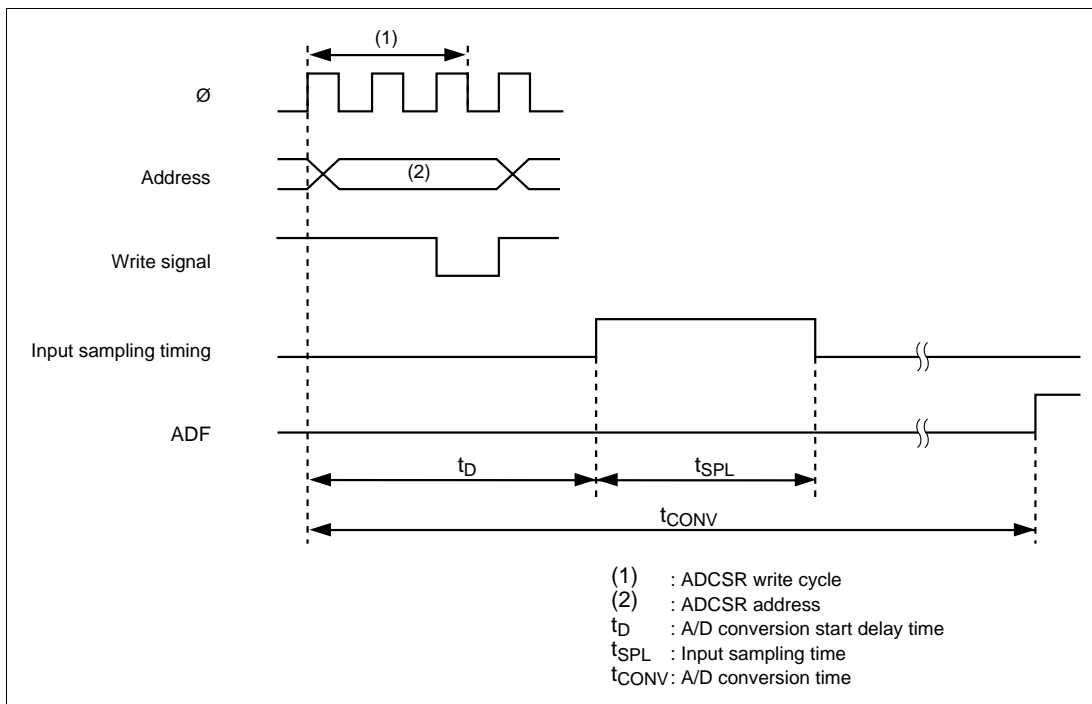


**Figure 17.11 SCI I/O Timing (Clock Sync Mode)**

### 17.3.7 A/D Converter Timing

**Table 17.10 A/D Converter Timing (Conditions:  $V_{CC} = 3.0$  to  $3.6V$ ,  $AV_{CC} = 3.0$  to  $5.5V$ ,  $AV_{CC} = V_{CC}$ ,  $V_{SS} = AV_{SS} = 0V$ ,  $T_a = \text{Đ20 to } +75^{\circ}C$ )**

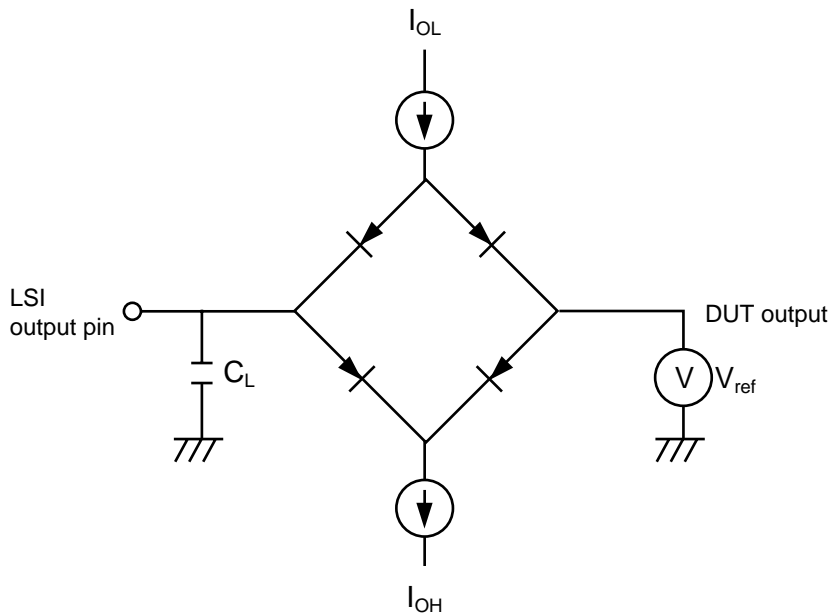
Item	Symbol	Min	Typ	Max	Unit	Figure
A/D conversion start delay time	$CKS = 0$ $t_D$	10	—	17	$t_{cyc}$	17.12
	$CKS = 1$	6	—	9		
Input sampling time	$CKS = 0$ $t_{RXS}$	—	64	—		
	$CKS = 1$	—	32	—		
A/D conversion time	$CKS = 0$ $t_{RXH}$	259	—	266		
	$CKS = 1$	131	—	134		



**Figure 17.12 Analog Conversion Timing**

### 17.3.8 Measurement Conditions for AC Characteristic

- Input reference levels:
  - High level: 2.2 V
  - Low level: 0.8 V
- Output reference levels:
  - High level: 2.0 V
  - Low level: 0.8 V



Note:  $C_L$  is set with the following pins, including the total capacitance of the measurement equipment etc:

30 pF: CK,  $\overline{CS0}$ – $\overline{CS3}$

50 pF: A21–A1, D15–D0,  $\overline{RD}$ ,  $\overline{WRx}$

70 pF: Port output and peripheral module output pins other than the above.

$I_{OL}$ ,  $I_{OH}$ : See table 17.3, Permitted Output Current Values.

**Figure 17.13 Output Added Circuit**

## 17.4 A/D Converter Characteristics

**Table 17.11 A/D Converter Characteristics (Conditions:  $V_{CC} = 3.0$  to  $3.6V$ ,  $AV_{CC} = 3.0$  to  $5.5V$ ,  $V_{CC} = V_{CC}$ ,  $V_{SS} = AV_{SS} = 0V$ ,  $T_a = \text{D20 to } +75^{\circ}C$ ,  $CKS = 0$ )**

Item	20 MHz			Unit
	Min	Typ	Max	
Resolution	10	10	10	bit
Conversion time	—	—	13.4	$\mu s$
Analog input capacity	—	—	20	pF
Permission signal source impedance	—	—	1	k $\Omega$
Non-linearity error*	—	—	$\pm 3$	LSB
Offset error*	—	—	$\pm 3$	LSB
Full scale error*	—	—	$\pm 3$	LSB
Quantize error*	—	—	$\pm 0.5$	LSB
Absolute error	—	—	$\pm 4$	LSB

Note: \* Reference values

**Table 17.12 A/D Converter Characteristics (Conditions:  $V_{CC} = 3.0$  to  $3.6V$ ,  $AV_{CC} = 3.0$  to  $5.5V$ ,  $V_{CC} = V_{CC}$ ,  $V_{SS} = AV_{SS} = 0V$ ,  $T_a = \text{D20 to } +75^{\circ}C$ ,  $CKS = 1$ )**

Item	20 MHz			Unit
	min	typ	max	
Resolution	10	10	10	bit
Conversion time	—	—	6.7	$\mu s$
Analog input capacity	—	—	20	pF
Permission signal source impedance	—	—	1	k $\Omega$
Non-linearity error*	—	—	$\pm 5$	LSB
Offset error*	—	—	$\pm 5$	LSB
Full scale error*	—	—	$\pm 5$	LSB
Quantize error*	—	—	$\pm 0.5$	LSB
Absolute error	—	—	$\pm 6$	LSB

Note: \* Reference values

# Appendix A On-Chip Supporting Module Registers

**Table A.1 On-Chip Supporting Module Registers**

Address H'FFFxxxx	Register Abbr.	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
81B0	SMR	—	CHR	PE	O/E	STOP	MP	CKS1	CKS0	SCI
81B1	BRR									
81B2	SCR	TIE	RIE	TE	RE	MPIE	TEIE	—	—	
81B3	TDR									
81B4	SSR	TDRE	RDRF	ORER	FER	PER	TEND	MPB	MPBT	
81B5	RDR									
8240	TSTR	—	—	—	—	—	CST2	CST1	CST0	Common MTU
8241	TSYR	—	—	—	—	—	SYNC2	SYNC1	SYNC0	
8260	TCR0	CCLR2	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	Ch0
8261	TMDR0	—	—	BFB	BFA	MD3	MD2	MD1	MD0	
8262	TIOR0H	—	—	—	—	IOA3	IOA2	IOA1	IOA0	
8263	TIOR0L	—	—	—	—	IOC3	IOC2	IOC1	IOC0	
8264	TIER0	TTGE	—	—	TCIEV	TGIED	TGIEC	TGIEB	TGIEA	
8265	TSR0	—	—	—	TCFV	TGFD	TGFC	TGFB	TGFA	
8266	TCNT0									
8267										
8268	TGR0A									
8269										
826A	TGR0B									
826B										
826C	TGR0C									
826D										
826E	TGR0D									
826F										

**Table A.1 On-Chip Supporting Module Registers (cont)**

Address H'FFFFxxxx	Register Abbr.	Bit Names								Module	
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0		
8280	TCR1	—	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	Ch1	MTU
8281	TMDR1	—	—	—	—	MD3	MD2	MD1	MD0		
8282	TIOR1	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0		
8284	TIER1	TTGE	—	—	TCIEV	—	—	TGIEB	TGIEA		
8285	TSR1	—	—	—	TCFV	—	—	TGFB	TGFA		
8286	TCNT1										
8287											
8288	TGR1A										
8289											
828A	TGR1B										
828B											
82A0	TCR2	—	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	Ch2	
82A1	TMDR2	—	—	—	—	MD3	MD2	MD1	MD0		
82A2	TIOR2	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0		
82A4	TIER2	TTGE	—	—	TCIEV	—	—	TGIEB	TGIEA		
82A5	TSR2	—	—	—	TCFV	—	—	TGFB	TGFA		
82A6	TCNT2										
82A7											
82A8	TGR2A										
82A9											
82AA	TGR2B										
82AB											
8348	IPRA	(IRQ0)	(IRQ0)	(IRQ0)	(IRQ0)	(IRQ1)	(IRQ1)	(IRQ1)	(IRQ1)	INTC	
8349		(IRQ2)	(IRQ2)	(IRQ2)	(IRQ2)	(IRQ3)	(IRQ3)	(IRQ3)	(IRQ3)		
834A	IPRB	(IRQ4)	(IRQ4)	(IRQ4)	(IRQ4)	(IRQ5)	(IRQ5)	(IRQ5)	(IRQ5)		
834B		(IRQ6)	(IRQ6)	(IRQ6)	(IRQ6)	(IRQ7)	(IRQ7)	(IRQ7)	(IRQ7)		
834C	IPRC	—	—	—	—	—	—	—	—		
834D		—	—	—	—	—	—	—	—		
834E	IPRD	(MTU0)	(MTU0)	(MTU0)	(MTU0)	(MTU0)	(MTU0)	(MTU0)	(MTU0)		
834F		(MTU1)	(MTU1)	(MTU1)	(MTU1)	(MTU1)	(MTU1)	(MTU1)	(MTU1)		
8350	IPRE	(MTU2)	(MTU2)	(MTU2)	(MTU2)	(MTU2)	(MTU2)	(MTU2)	(MTU2)		
8351		—	—	—	—	—	—	—	—		
8352	IPRF	—	—	—	—	—	—	—	—		
8353		—	—	—	—	(SCI)	(SCI)	(SCI)	(SCI)		



**Table A.1 On-Chip Supporting Module Registers (cont)**

Address H'FFFFxxxx	Register Abbr.	Bit Names								Module	
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0		
8354	IPRG	(A/D)	(A/D)	(A/D)	(A/D)	—	—	—	—	INTC	
8355		(CMT0)	(CMT0)	(CMT0)	(CMT0)	(CMT1)	(CMT1)	(CMT1)	(CMT1)		
8356	IPRH	(TIM1,2)	(TIM1,2)	(TIM1,2)	(TIM1,2)	—	—	—	—		
8357		—	—	—	—	—	—	—	—		
8358	ICR	NMIL	—	—	—	—	—	—	—	NMIE	
8359		IRQ0S	IRQ1S	IRQ2S	IRQ3S	IRQ4S	IRQ5S	IRQ6S	IRQ7S		
835A	ISR	—	—	—	—	—	—	—	—		
835B		IRQ0F	IRQ1F	IRQ2F	IRQ3F	IRQ4F	IRQ5F	IRQ6F	IRQ7F		
8380	PADRH	—	—	—	—	—	—	—	—	I/O	Port A
8381		—	—	—	—	PA19DR	PA18DR	—	—		
8384	PAIORH	—	—	—	—	—	—	—	—	PFC	
8385		—	—	—	—	PA19IOR	PA18IOR	—	—		
83B0	PEDR	—	PE14DR	PE13DR	PE12DR	—	—	—	—	I/O	Port E
83B1		PE7DR	PE6DR	PE5DR	PE4DR	—	PE2DR	—	PE0DR		
83B4	PEIOR	—	PE14IOR	PE13IOR	PE12IOR	—	—	—	—	PFC	
83B5		PE7IOR	PE6IOR	PE5IOR	PE4IOR	—	PE2IOR	—	PE0IOR		
83BA	PECR2	—	PE7MD	—	PE6MD	—	PE5MD	—	PE4MD		
83BB		—	—	—	PE2MD0	—	—	—	PE0MD0		
83D0	CMSTR	—	—	—	—	—	—	—	—	Common	CMT
83D1		—	—	—	—	—	—	STR1	STR0		
83D2	CMCSR0	—	—	—	—	—	—	—	—	Ch0	
83D3		CMF	CMIE	—	—	—	—	CKS1	CKS0		
83D4	CMCNT0	—	—	—	—	—	—	—	—		
83D5		—	—	—	—	—	—	—	—		
83D6	CMCOR0	—	—	—	—	—	—	—	—		
83D7		—	—	—	—	—	—	—	—		
83D8	CMCSR1	—	—	—	—	—	—	—	—	Ch1	
83D9		CMF	CMIE	—	—	—	—	CKS1	CKS0		
83DA	CMCNT1	—	—	—	—	—	—	—	—		
83DB		—	—	—	—	—	—	—	—		
83DC	CMCOR1	—	—	—	—	—	—	—	—		
83DD		—	—	—	—	—	—	—	—		

**Table A.1 On-Chip Supporting Module Registers (cont)**

Address H'FFFFxxxx	Register Abbr.	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
8420	ADDRAH	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	A/D
8421	ADDRAL	AD1	AD0	—	—	—	—	—	—	
8422	ADDRBH	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	
8423	ADDRBL	AD1	AD0	—	—	—	—	—	—	
8424	ADDRCH	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	
8425	ADDRCL	AD1	AD0	—	—	—	—	—	—	
8426	ADDRDH	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	
8427	ADDRDL	AD1	AD0	—	—	—	—	—	—	
8428	ADCSR	ADF	ADIE	ADST	SCAN	CKS	CH2	CH1	CH0	
8429	ADCR	TRGE	—	—	—	—	—	—	—	
8610	T1CSR	OVF	—	TME	—	—	CKS2	CKS1	CKS0	TIM1
8610	T1CNT (WR)									
8611	T1CNT (RD)									
8622	BCR2	IW31	IW30	IW21	IW20	IW11	IW10	IW01	IW00	BSC
8623		CW3	CW2	CW1	CW0	SW3	SW2	SW1	SW0	
8624	WCR1	—	—	W31	W30	—	—	W21	W20	
8625		—	—	W11	W10	—	—	W01	W00	
862C	T2CSR	—	—	—	—	—	—	—	—	TIM2
862D		—	CMF	CMIE	CKS2	CKS1	CKS0	—	—	
862E	T2CNT	—	—	—	—	—	—	—	—	
862F										
8630	T2COR	—	—	—	—	—	—	—	—	
8631										

# Appendix B Pin States

## B.1 Pin States

**Table B.1 Pin States during Reset, and Power-Down**

Pin Function		Pin Modes	
		Reset	Power-Down
Class	Pin Name	Power-On	Sleep
Clock	CK	O	O
System control	$\overline{\text{RES}}$	I	I
Interrupt	NMI	I	I
	$\overline{\text{IRQ0}}\text{--}\overline{\text{IRQ7}}$	Z	I
Address bus	A1–A21	O	O
Data bus	D0–D15	Z	I/O
Bus control	$\overline{\text{WAIT}}$	Z	I
	$\overline{\text{RD}}$	H	H
	$\overline{\text{CS0}}, \overline{\text{CS1}}$	H	H
	$\overline{\text{CS2}}, \overline{\text{CS3}}$	Z	H
	$\overline{\text{WRH}}, \overline{\text{WRL}}$	H	H
MTU	TIOC0A, TIOC0C, TIOC1A, TIOC1B, TIOC2A, TIOC2B	Z	I/O
SCI	TxD	Z	O
	RxD	Z	I
A/D converter	AN0–AN6	Z	I
I/O port	PA18, PA19	Z	K
	PE0, PE2, PE4–PE7, PE12–PE14		

Legend:

I: Input

O: Output

H: High-level output

L: Low-level output

Z: High impedance

K: Input pin with high impedance, output pin mode maintained.

## B.2 Bus Related Signal Pin States

Table B.2 Bus Related Signal Pin States

Pin Name	On-Chip Supporting Module						External Normal Space		
	Internal RAM Space	8-Bit Space	16-Bit Space			16-Bit Space			
			Upper Byte	Lower Byte	Word/ Longword	Upper Byte	Lower Byte	Word/ Longword	
$\overline{CS0}$ – $\overline{CS3}$	H	H	H	H	H	Valid	Valid	Valid	
$\overline{RD}$	R	H	H	H	H	H	L	L	L
	W	H	H	H	H	H	H	H	H
$\overline{WRH}$	R	H	H	H	H	H	H	H	H
	W	H	H	H	H	H	L	L	L
$\overline{WRL}$	R	H	H	H	H	H	H	H	H
	W	H	H	H	H	H	H	L	L
A21–A1	Address	Address	Address	Address	Address	Address	Address	Address	
D15–D8	Z	Z	Z	Z	Z	Data	Z	Data	
D7–D0	Z	Z	Z	Z	Z	Z	Data	Data	

Legend:

R: Read

W: Write

H: High-level output

L: Low-level output

Valid: Chip select signal corresponding with accessed area is low; chip select signal in other cases is high.

# Appendix C Package Dimensions

Package dimensions of the SH7011 (TFP-100B) are shown in figure C.1.

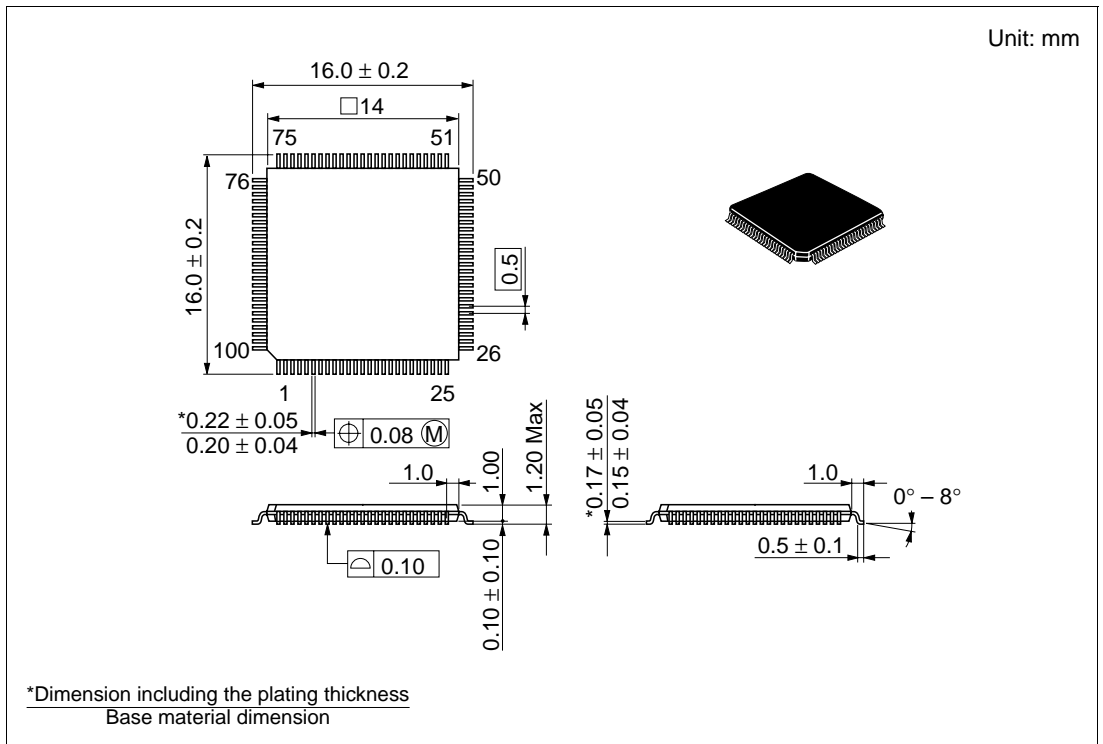


Figure C.1 Package Dimensions (TFP-100B)

---

## **SH7011 Hardware Manual**

Publication Date: 1st Edition, December 1998

Published by: Electronic Devices Sales & Marketing Group  
Semiconductor & Integrated Circuits Group  
Hitachi, Ltd.

Edited by: Technical Documentation Group  
UL Media Co., Ltd.

Copyright © Hitachi, Ltd., 1998. All rights reserved. Printed in Japan.