

# BHA250 / BHA250B

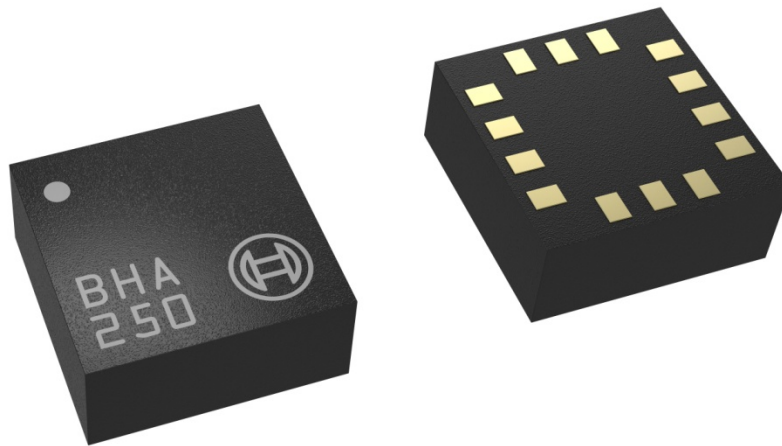
Ultra low-power sensor hub incl. integrated Accel

Bosch Sensortec



**BOSCH**

Invented for life



## Data Sheet

Document revision	1.2
Document release date	Mar 2017
Document number	BST-BHA250(B)-DS000-01
Technical reference code(s)	BHA250: 0 273 141 231    BHA250B: 0 273 141 310

Notes      Data in this document are subject to change without notice. Product photos and pictures are for illustration purposes only and may differ from the real product's appearance.

### Features

- All-in-one smart-hub solution for always-on motion sensing at a fraction of current consumption which is commonly required using discrete components.
- 32-bit floating-point microcontroller (Fuser Core). Optimized for data fusion, motion sensing and activity recognition at ultra low power consumption. All in order to offload the power hungry data processing from the main application processor to the smart-hub.
- Powerful BSX sensor fusion library integrated in ROM for lowest design-in effort and fastest time-to-market.
- Additional software and algorithms for RAM processing, provided as ready to use FW patch files. Visit our web site to check available downloads.
- Onboard calculation power for data fusion, 3D- and absolute orientation, rotation vector, quaternions and Euler angles.
- Gesture recognition of significant motion, tilt, pickup, wake up and glance. Enabling customer specific gesture based HMI interfaces for smartphones and wearables.
- Activity recognition of standing, walking, running, biking and in vehicle. Enabling health & fitness applications or any other use case where highly accurate and reliable detection and/or monitoring of user activities is required.
- Step detection and step counting.
- Android 5 / L / Lollipop & Android 6 / M / Marshmallow (non-HiFi) support, incl. batching with dual FIFO buffer for wakeup and non-wakeup events. Implements the full Android sensor stack although an Android OS or any other Android environment is not required.
- High speed I2C interface, with data rates up to 3.4 MBit/s for power-efficient data transfer.
- Highly configurable internal RAM for either feature extension and/or FIFO data buffering.
- SW / FW based functionality. Can be updated, optimized, customized or upgraded with totally new features to support future requirements.
- Smart-hub plus microcontroller, MEMS sensors and software all highly integrated in one 2.2x2.2x0.95 mm<sup>3</sup> LGA package with extension interface for additional sensors.

### Implemented Sensor Types

#### With integrated acceleration sensor only:

Accelerometer, Step counter, Step detector, Significant motion, Tilt gesture, Pickup gesture, Wake up gesture, Glance gesture, Activity recognition

#### With attached gyroscope:

Gravity, Linear acceleration, Gyroscope, Gyroscope uncalibrated, Game rotation vector

#### With attached magnetometer:

Geomagnetic field, Magnetic field uncalibrated, Orientation, Rotation vector, Geomagnetic rotation vector

### General Description

The BHA250(B) is a small, low-power smart-hub with an integrated three axis accelerometer plus a programmable microcontroller, all specifically designed to enable always-on motion sensing. On top it contains software and algorithms for motion-step-, gesture- and activity recognition. The overall concept perfectly matches the requirements of smartphones, wearables or any other application which demands highly accurate, real-time motion data at very low power consumption.

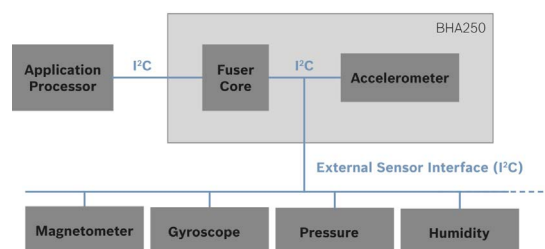
The device integrates our millionfold proven 14bit acceleration sensor with a microcontroller – the new Bosch Sensortec Fuser core. It is bringing you the full Android sensor stack inside your devices – even without having an Android OS or an Android environment. Combining this with the built in computing power and the highly configurable on-board memory the BHA smart-hub offers you a low power solution for motion sensing and data processing.

### Target applications

- Activity recognition of standing, walking, running, biking or in vehicle
- HMI interfaces incl. gesture detection of motion, tilt, pickup, wake up and glance
- Step detection and step counting
- Indoor navigation, PDR
- Augmented reality, immersive gaming
- Tilt compensated eCompass and orientation

### Target devices

- Mobile phones and tablets
- Wearables such as smart watches, wrist- or neck-bands
- Smart-sports and smart-fitness devices
- Hearables, smart earphones and other head worn devices
- Smart-TV- or AR/VR controllers
- Smart-pens



# Table of Contents

<b>1. SPECIFICATION.....</b>	<b>7</b>
1.1 ELECTRICAL SPECIFICATION.....	7
1.2 ELECTRICAL AND PHYSICAL CHARACTERISTICS, MEASUREMENT PERFORMANCE.....	7
1.3 ABSOLUTE MAXIMUM RATINGS .....	9
<b>2. PIN CONNECTIONS AND DESCRIPTION.....</b>	<b>10</b>
2.1 CONNECTION DIAGRAM .....	11
<b>3. OVERVIEW.....</b>	<b>12</b>
<b>4. PHYSICAL INTERFACES.....</b>	<b>14</b>
4.1 HOST INTERFACE.....	14
4.2 SENSOR INTERFACE .....	15
<b>5. DATA INTERFACE.....</b>	<b>16</b>
5.1 GENERAL OVERVIEW.....	16
5.2 REGISTER MAP.....	17
<b>6. DEVICE INITIALIZATION AND STARTUP.....</b>	<b>19</b>
6.1 RESET.....	19
6.2 BOOT MODE .....	19
6.3 MAIN EXECUTION MODE .....	19
<b>7. DEVICE CONFIGURATION .....</b>	<b>21</b>
<b>8. FIFOS AND EVENTS.....</b>	<b>22</b>
<b>9. FUNCTIONAL DESCRIPTION.....</b>	<b>23</b>
9.1 DATAFLOW OF SENSOR FUSION.....	23
9.2 SUPPORTED DATA RATES OF BSX SENSOR FUSION ENGINE .....	23
9.3 GESTURE RECOGNITION .....	24
9.4 POWER MODES AND CURRENT CONSUMPTION .....	24
9.5 VIRTUAL SENSORS .....	25
9.6 VIRTUAL SENSOR DATA TYPES .....	27
9.7 SENSOR CONFIGURATION.....	28
9.8 SENSOR STATUS INFORMATION .....	29
9.9 FIFOS.....	30
9.10 NON-BATCH MODE .....	31
9.11 BATCH MODE .....	31

<b>10. REGISTER MAP DESCRIPTION.....</b>	<b>32</b>
10.1 BUFFER_OUT[0:49] .....	32
10.2 FIFO_FLUSH.....	32
10.3 CHIP_CONTROL.....	33
10.4 HOST_STATUS .....	33
10.5 INT_STATUS .....	34
10.6 CHIP_STATUS .....	34
10.7 BYTES_REMAINING[0:1] .....	35
10.8 PARAMETER_ACKNOWLEDGE .....	35
10.9 PARAMETER_READ_BUFFER[0:15].....	36
10.10 GP[20:24] .....	36
10.11 PARAMETER_PAGE_SELECT .....	36
10.12 HOST_INTERFACE_CONTROL .....	38
10.13 GP[31:36].....	40
10.14 PARAMETER_WRITE_BUFFER[0:7] .....	40
10.15 PARAMETER_REQUEST .....	40
10.16 GP[46:52].....	41
10.17 ROM_VERSION[0:1].....	41
10.18 RAM_VERSION[0:1] .....	41
10.19 PRODUCT_ID .....	42
10.20 REVISION_ID.....	42
10.21 UPLOAD_ADDRESS[0:1] .....	43
10.22 UPLOAD_DATA .....	43
10.23 UPLOAD_CRC[0:3] .....	43
10.24 RESET_REQUEST .....	44
<b>11. PARAMETER I/O DESCRIPTION.....</b>	<b>45</b>
11.1 PARAMETER PAGE 1: SYSTEM .....	45
11.2 PARAMETER PAGE 3: SENSORS.....	50
11.3 SENSOR INFORMATION STRUCTURE .....	53
11.4 SENSOR CONFIGURATION STRUCTURE.....	54
11.5 PARAMETER PAGE 15: SOFT PASS-THROUGH .....	55
<b>12. SENSOR DATA TYPES AND OUTPUT FORMAT .....</b>	<b>57</b>
12.1 QUATERNION+.....	58
12.2 VECTOR+ .....	59
12.3 VECTOR_UNCALIBRATED.....	60



12.4 SCALAR DATA.....	60
12.5 SENSOR EVENT DATA (PARAMETERLESS SENSORS).....	61
12.6 ACTIVITY RECOGNITION DATA (SENSOR_ACTIVITY_REC_DATA).....	61
12.7 DEBUG .....	61
12.8 SENSOR DATA SCALING .....	62
12.9 META EVENTS .....	63
12.9.1 SELF-TEST RESULTS.....	64
12.9.2 INITIALIZED.....	65
<b>13. READING FIFO DATA.....</b>	<b>66</b>
13.1 HOST INTERRUPT BEHAVIOR.....	66
13.2 PAUSE AND RESUME MECHANISM.....	67
13.3 FIFO OVERFLOW HANDLING.....	68
13.4 HOST SUSPEND PROCEDURE .....	68
13.5 HOST WAKEUP PROCEDURE.....	68
13.6 NON-COMPLIANT HOSTS .....	68
13.7 RECOVERY FROM LOSS OF SYNC.....	69
13.8 PADDING DATA .....	69
13.9 ABORTING A TRANSFER .....	69
13.10 FIFO PARSING EXAMPLES.....	69
13.10.1 ACCELEROMETER & STEP COUNTER.....	69
<b>14. PACKAGE.....</b>	<b>72</b>
14.1 OUTLINE DIMENSIONS.....	72
14.2 SENSING AXES ORIENTATION AND AXIS REMAPPING.....	72
14.3 LANDING PATTERN RECOMMENDATION.....	74
14.4 MARKING.....	75
14.4.1 MASS PRODUCTION.....	75
14.4.2 ENGINEERING SAMPLES .....	75
14.5 SOLDERING GUIDELINES .....	76
14.6 HANDLING INSTRUCTIONS .....	77
14.7 TAPE AND REEL SPECIFICATION .....	77
14.7.1 ORIENTATION WITHIN THE REEL.....	78
14.8 ENVIRONMENTAL SAFETY .....	78
14.9 HALOGEN CONTENT .....	78
14.10 MULTIPLE SOURCING.....	78
<b>15. LEGAL DISCLAIMER.....</b>	<b>79</b>
15.1 ENGINEERING SAMPLES .....	79

15.2 PRODUCT USE .....	79
15.3 APPLICATION EXAMPLES AND HINTS .....	79
<b>16. DOCUMENT HISTORY AND MODIFICATIONS.....</b>	<b>80</b>

# 1. Specification

## 1.1 Electrical specification

Table 1: Operating Conditions

OPERATING CONDITIONS BHA						
Parameter	Symbol	Condition	Min	Typ	Max	Unit
Supply Voltage Internal Domains	$V_{DD}$		1.62	2.4	3.6	V
Supply Voltage I/O Domain	$V_{DDIO}$		1.6	2.4	3.3	V
Voltage Input Low Level	$V_{IL,a}$		0		$0.3V_{DDIO}$	V
Voltage Input High Level	$V_{IH,a}$		$0.7V_{DDIO}$		$V_{DDIO}$	V
Voltage Output Low Level	$V_{OL,a}$	$I_{OL}=1mA$			0.3V	V
Voltage Output High Level	$V_{OH,a}$	$I_{OH}=-1mA,$	$V_{DDIO}-0.3V$			V
Operating Temperature	$T_A$		-40		+85	°C

## 1.2 Electrical and physical characteristics, measurement performance

All parameters defined for operating conditions (unless otherwise specified)

Table 2: Electrical characteristics Fuser Core

OPERATING CONDITIONS FUSER CORE						
Parameter	Symbol	Condition	Min	Typ	Max	Units
REGULATOR OUTPUT VOLTAGE	$V_{REG}$		1.0	1.1	1.2	V
POWER ON RESET THRESHOLD	$V_{POR}$	$V_{REG} > V_{POR}$		$V_{REG} - 125mV$		V
CURRENT CONSUMPTION, RUN1	$I_{RUN}$	0°C TO +40°C, (1)		800		UA
CURRENT CONSUMPTION, NORMAL OPERATION2	$I_{OPER}$	0°C TO +40°C, (2)		350		UA
CURRENT CONSUMPTION, SLEEP3	$I_{SLEEP}$	0°C TO +40°C, (3)		40		UA
CURRENT CONSUMPTION, DEEP SLEEP4	$I_{DSLEEP}$	0°C TO +40°C, (4)		7		UA
CURRENT CONSUMPTION, IDLE5	$I_{IDLE}$	0°C TO +40°C, (5)		6		UA

Notes:

- (1) Current consumption when CPU is running and executing from ROM.

- (2) Current consumption in normal operation is average consumption for 9DoF Sensor Fusion with ODR of 100 Hz
- (3) Sleep mode is entered when CPU and I2C are idle and timer and system clock are enabled
- (4) In Deep Sleep mode, only timer is enabled while system clock is disabled
- (5) In Idle mode, no operations are performed, all oscillators are disabled

Table 3: Electrical characteristics accelerometer

OPERATING CONDITIONS ACCELEROMETER						
Parameter	Symbol	Condition	Min	Typ	Max	Units
Acceleration Range	$g_{FS2g}$	Selectable via serial digital interface		$\pm 2$		g
	$g_{FS4g}$			$\pm 4$		g
	$g_{FS8g}$			$\pm 8$		g
	$g_{FS16g}$			$\pm 16$		g

OUTPUT SIGNAL ACCELEROMETER						
Parameter	Symbol	Condition	Min	Typ	Max	Units
Resolution				14		bit
Sensitivity	$S_{2g}$	$g_{FS2g}$ , $T_A=25^\circ\text{C}$		4096		LSB/g
	$S_{4g}$	$g_{FS4g}$ , $T_A=25^\circ\text{C}$		2048		LSB/g
	$S_{8g}$	$g_{FS8g}$ , $T_A=25^\circ\text{C}$		1024		LSB/g
	$S_{16g}$	$g_{FS16g}$ , $T_A=25^\circ\text{C}$		512		LSB/g
Sensitivity Temperature Drift	$TCS_a$	$g_{FS2g}$ , Nominal $V_{DD}$ supplies		$\pm 0.02$		%/K
Sensitivity Supply Volt. Drift	$S_{VDD,a}$	$g_{FS2g}$ , $T_A=25^\circ\text{C}$ , $V_{DD\_min} \leq V_{DD} \leq V_{DD\_max}$		0.05		%/V
Zero-g Offset	Off	$g_{FS2g}$ , $T_A=25^\circ\text{C}$ , nominal $V_{DD}$ supplies, over life-time		$\pm 80$		mg
Zero-g Offset Temperature Drift	$TCO_a$	$g_{FS2g}$ , Nominal $V_{DD}$ supplies		$\pm 1$		mg/K
Zero-g Offset Supply Volt. Drift	$Off_{VDD,a}$	$g_{FS2g}$ , $T_A=25^\circ\text{C}$ , $V_{DD\_min} \leq V_{DD} \leq V_{DD\_max}$		0.5		mg/V
Nonlinearity	$NL_A$	Best fit straight line, $g_{FS2g}$		$\pm 0.5$		%FS
Output Noise Density	$n_{rms,a}$	$g_{FS2g}$ , $T_A=25^\circ\text{C}$ , nominal $V_{DD}$ supplies, Normal mode		150		$\mu\text{g}/\sqrt{\text{Hz}}$

### 1.3 Absolute maximum ratings

Table 4: Absolute maximum ratings

PARAMETER	Condition	Min	Max	Units
Voltage at Supply Pin	V <sub>DD</sub> Pin	-0.3	4.25	V
	V <sub>DDIO</sub> Pin	-0.3	3.6	V
Voltage at any Logic Pin	Non-Supply Pin	-0.3	V <sub>DDIO</sub> +0.3	V
Passive Storage Temp. Range	≤65% rel. H.	-50	+150	°C
None-volatile memory (NVM) Data Retention	T = 85°C, after 15 cycles	10		y
Mechanical Shock	Duration 200 μs, half sine		10,000	g
	Duration 1.0 ms, half sine		2,000	g
	Free fall onto hard surfaces		1.8	m
ESD	HBM, at any Pin		2	kV
	CDM		500	V
	MM		100	V

**NOTE:** Stress above these limits may cause damage to the device. Exceeding the specified electrical limits may affect the device reliability or cause malfunction.

## 2. Pin Connections and description

Figure 1: Pin Connections

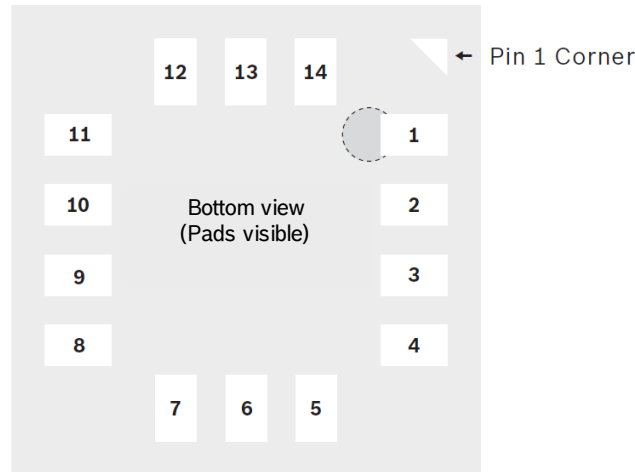


Table 5: Pin description

Pin	Name	Description
1	INT	Host interrupt
2	SCK	I <sup>2</sup> C serial clock (Host interface)
3	ASCK	I <sup>2</sup> C Master serial clock, for connecting to external sensors
4	ASDA	I <sup>2</sup> C master serial data, for connecting to external sensors
5	VREG	Regulator filter capacitor connection
6	GPIO1	Application specific I/O pin <sup>1)</sup>
7	RESV1	Do not connect pin (reserved)
8	GPIO2	Application specific I/O pin
9	GND	Analog power supply ground
10	SA_GPIO7	Select I <sup>2</sup> C address & Application specific I/O pin refer to section 4.1 page 14
11	GNDIO	Digital I/O power supply ground
12	VDD	Analog power supply voltage (1.71V ... 3.6V)
13	VDDIO	Digital I/O power supply voltage (1.6 ... 3.3 V)
14	SDA	I <sup>2</sup> C serial data (Host interface)

- 1) GPIO1 is driven low at power up until firmware download is completed and BHA is initialized.

## 2.1 Connection Diagram

Figure 2: Reference Diagram

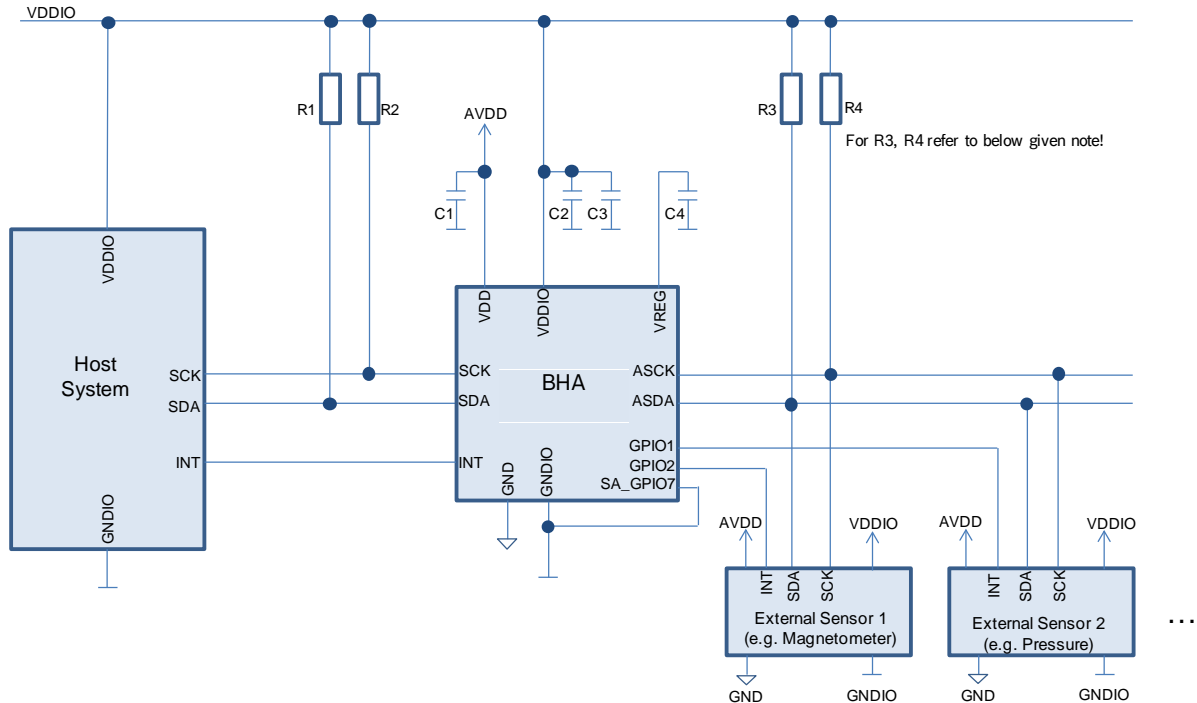


Table 6: Typical values for external circuit components

Component	Value	Remarks
R1	4.7 kΩ	Pull-up resistor for SDA, Host Interface
R2	4.7 kΩ	Pull-up resistor for SCK, Host Interface
R3	4.7 kΩ	Pull-up resistor for ASDA, Aux Interface
R4	4.7 kΩ	Pull-up resistor for ASCK, Aux Interface
C1	100nF	Filter capacitor AVDD
C2	1 μF	Filter capacitor VDDIO
C3	100 nF	Filter capacitor VDDIO
C4	470 nF	Filter capacitor VREG

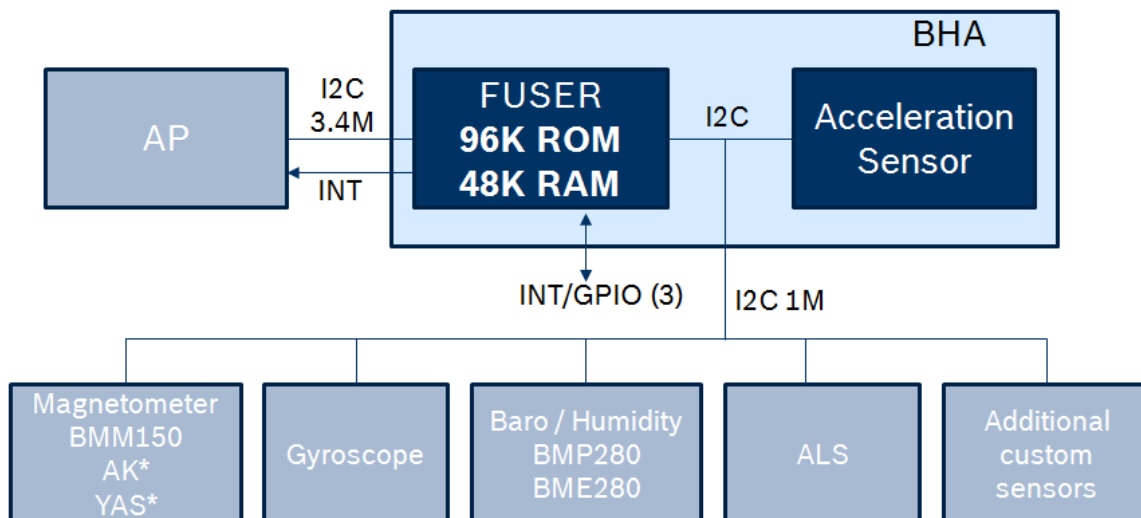
**NOTE:** R3 and R4 are mandatory, even if no external sensor is attached.

### 3. Overview

The BHA Sensor hub is a small multichip system in a LGA package consisting of

- a 32-bit floating-point microcontroller (Fuser Core) optimized for sensor fusion and activity recognition
- 96 KByte of ROM including the BSX sensor fusion library
- 48 KByte of RAM for
  - feature extension (e.g. for additional drivers of externally attached sensors)
  - local data buffering (implementing a wake-up and a non-wake-up FIFO as defined in Android)
  - feature updates (allowing the updates of features implemented in RAM or ROM to meet future requirement)
- a high speed I<sup>2</sup>C host interface, with data rates up to 3.4 MBit/s and a host interrupt line
- a fast I<sup>2</sup>C sensor interface, with data rates up to 1 MBit/s for connection of external sensors
- up to 3 additional GPIO pins

Figure 3: Block Diagram



With these integrated hardware and software features, the low power consumption and the sensor extension interface, the BHA provides an ideal all-in-one solution for always-on sensor applications.



Without any additionally attached sensors the BHA provides a three degrees of freedom (3-DoF) acceleration sensor out of the box, implementing the following Android<sup>1</sup> sensor types:

- Accelerometer
- Step counter
- Step detector
- Significant motion
- Tilt detector
- Pickup gesture
- Wake up gesture
- Glance gesture
- Activity recognition<sup>2</sup> of standing, walking, running, biking, in vehicle

By attaching an external magnetometer to the sensor interface (and configuring the RAM firmware patch to include the sensor driver for the magnetometer) the BHA provides additionally the following sensor types:

- Gravity
- Linear acceleration
- Geomagnetic field
- Magnetic field uncalibrated
- Orientation
- Rotation vector
- Geomagnetic rotation vector

offering a robust eCompass solution to the user.

With further attachment of additional sensors to the sensor interface, as e.g.

- Gyroscope
- Barometric pressure
- Humidity
- Ambient temperature
- Proximity
- Ambient Light

the BHA can provide the full Android sensor stack to the application.

---

<sup>1</sup> See <http://source.android.com/devices/sensors/sensor-types.html> for details on defined Android Sensor Types.

<sup>2</sup> Activity recognition is also implemented as a Sensor Type in BHA250, despite not being defined in Android's "sensors.h", but in "activity\_recognition.h".

## 4. Physical Interfaces

### 4.1 Host interface

According to the interface concept introduced from Android 5 onwards, the BHA provides a high speed I<sup>2</sup>C interface and a single interrupt line as main interface to the application processor.

The available GPIO pins can be used to implement additional interrupt lines, in case this is necessary for specific applications.

The host interface is implemented as an I<sup>2</sup>C slave interface, as described in the I<sup>2</sup>C bus specification created from NXP<sup>3</sup> and implements data transfer rates up to 3.4 Mbit/s in the high-speed mode.

The I<sup>2</sup>C bus consists of 2 wires, SCK (Serial Clock) and SDA (Serial Data). Both bus lines are bi-directional. The BHA250 can be connected to this bus via SDA and SCL pads with open drain drivers within the device. The bus lines must be externally connected to a positive supply voltage (VDDIO) via a pull-up resistor or current-source.

A data transfer via the I<sup>2</sup>C slave interface is always initiated by the host. The I<sup>2</sup>C slave interface can operate as either a transmitter or receiver only, if a valid device address has been received from the host.

The BHA250 responds to device addresses, depending on the logic level applied on the SA\_GPIO7. To select the corresponding I<sup>2</sup>C address keep the desired level for min 10 ns after reset release as described in Table 7. By default there are 2 application specific I/O pins GPIO1 and GPIO2 available and recommended. Special cases might require additional I/O pins. Therefore SA\_GPIO was designed to be operated as a third application specific I/O pin, once the I<sup>2</sup>C address was successfully selected. For details and technical support please refer to corresponding application notes or contact our regional offices, distributors and sales representatives.

Table 7: I<sup>2</sup>C address selection

SA_GPIO7	I <sup>2</sup> C address
HIGH	0x29
LOW	0x28

The address and data are transferred between master and slave serially through the data line (SDA) in an 8-bit oriented transfer format. The transfer is synchronized by the serial clock line (SCK). The supported transfer formats are single byte read, multiple byte read, single byte write, multiple byte write. The data line (SDA) can be driven either by the host or the BHA. The serial clock line (SCK) is driven by the host only.

Figure 3 illustrates an example of how to write data to registers in single-byte or multiple-byte mode.

Figure 4: I<sup>2</sup>C write example

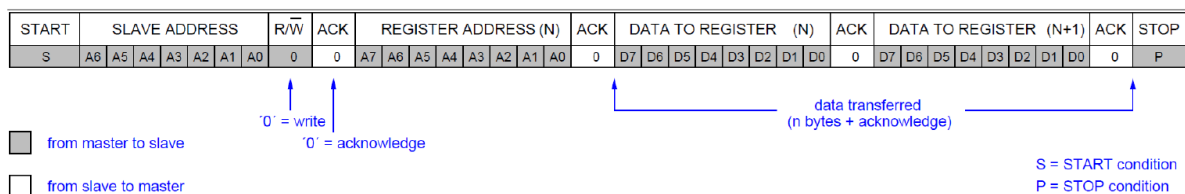
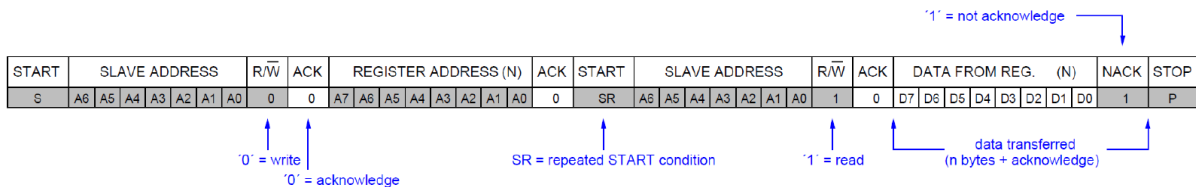


Figure 4 illustrates an example of how to read data to registers in single-byte or multiple-byte mode.

<sup>3</sup> See [http://www.nxp.com/documents/user\\_manual/UM10204.pdf](http://www.nxp.com/documents/user_manual/UM10204.pdf) for details

Figure 5: I<sup>2</sup>C read example


## 4.2 Sensor interface

The BHA implements a fast-mode plus I<sup>2</sup>C master interface for connections of external sensors. This sensor interface is directly connected to the internal acceleration sensor and also available on the auxiliary serial clock (ASCK) and auxiliary serial data (ASDA) pins of the BHA.

The bus lines must be externally connected to a positive supply voltage (VDDIO) via a pull-up resistor or current-source, even if no additional external sensor is attached to the device, in order to enable the proper I<sup>2</sup>C communication between the Fuser core and the integrated BMA2x2 acceleration sensor.

A common use-case of the sensor interface is the connection of an external magnetometer. The following external magnetometers are currently supported:

Table 8: Supported Magnetometers

Vendor	Device
Bosch Sensortec	BMM150
Asahi Kasei	AK09911/12
Yamaha	YAS532/537

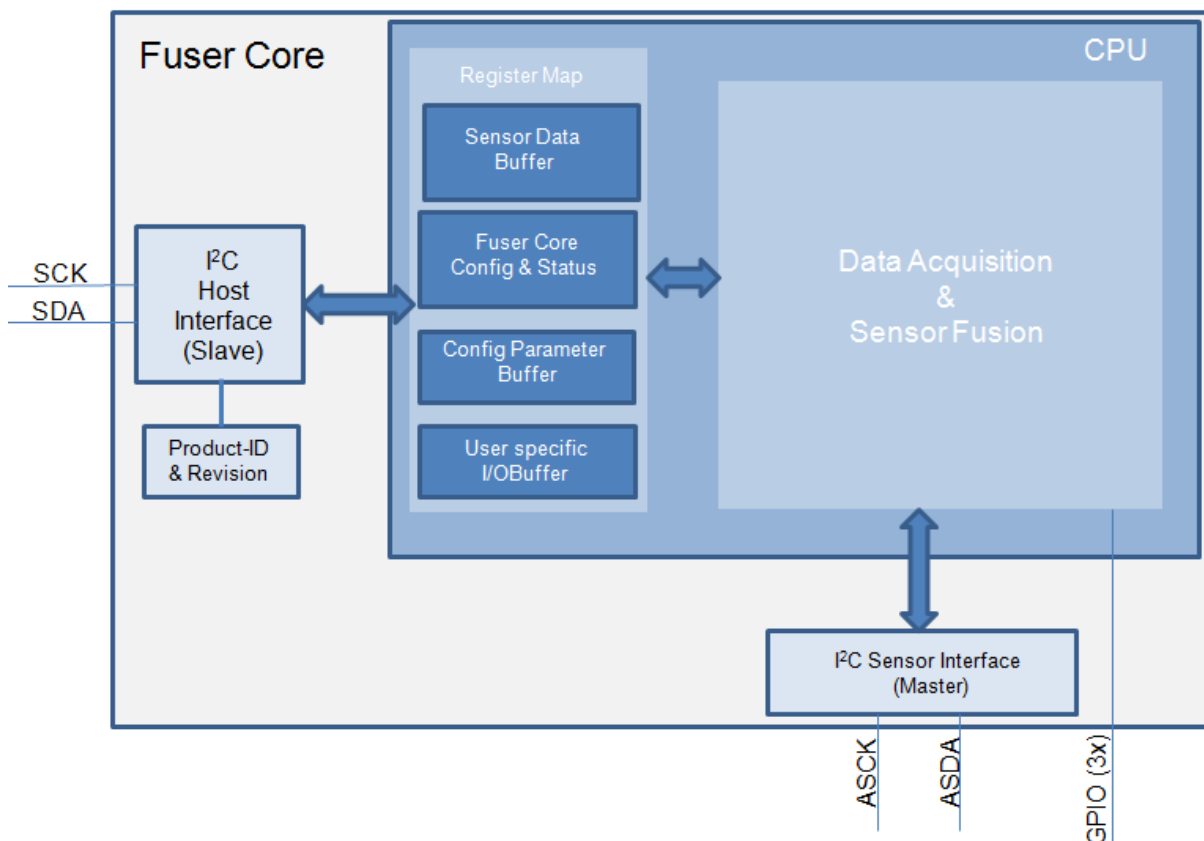
Alternative magnetometers can be supported on customer request.

## 5. Data interface

### 5.1 General overview

Figure 5 provides a general overview of the BHA data interface. The software running on the Fuser core obtains the raw sensor from the I<sup>2</sup>C sensor interface, performs the necessary computations and provides the results into a register map, which forms the main I/O interface to the host from a programmer's point of view.

Figure 6: BHA data interface



The register map consists of 4 main sections:

- Sensor Data Buffer
- Fuser Core Config & Status Buffer
- Configuration Parameter I/O
- User specific I/O Buffer

The **Sensor Data Buffer** consists of 50 register (I<sup>2</sup>C addresses 0x00:0x31) providing an interface to the Fuser Core's internal Event FIFOs which contain the sensor event data.

Per default the data of both FIFOs (the wake-up and the non-wake-up FIFO) will be mapped to the Sensor Data Buffer, so that the host can read all available data in a burst and identify and separate the data afterwards.

The FIFO\_FLUSH register of the Fuser Core Config & Status Buffer can be used to adjust the behavior of the sensor data buffer in a more specific way.

The **Fuser Core & Status Buffer** consist of a register set, which allows the host to control the fundamental behavior of the fuser core as well as getting information on the current status.

**The Configuration Parameter I/O interface** provides a window in the various configuration options of the sensor system. It consists of the 16 Byte deep Par\_Read\_Buffer (I<sup>2</sup>C addresses 0x3B:0x4A) and the 8 Byte deep Par\_Write\_Buffer (I<sup>2</sup>C addresses 0x5C:0x63) and some additional registers for selecting and mapping the desired parameters into these 2 buffers.

The **User Specific I/O Buffer** is reserved for application specific purposes and can be used to serve the needs of individual applications. It consists of 3 different I<sup>2</sup>C address areas (0x4B:0x4F, 0x56:0x5B and 0x65:0x6B) where the first one is read-only, while the others are read-write for the host.

## 5.2 Register Map

Table 9: BHA Register Map

I2C Address	Register Name	Access mode	Map section
0x00 – 0x31	Buffer Out[00:49]	Read only	Sensor Data Buffer
0x32	FIFO Flush	Read write	Fuser Core Config & Status
0x33	<i>Reserved</i>		
0x34 0x35 0x36 0x37	Chip Control Host Status Int Status Chip Status	Read write Read only Read only Read only	Fuser Core Config & Status
0x38 0x39	Bytes Remaining LSB Bytes Remaining MSB	Read only Read only	Sensor Data Buffer
0x3A	Parameter Acknowledge	Read only	Config Parameter I/O Interface
0x3B – 0x4A	Parameter Read Buffer[0:15]	Read only	Config Parameter I/O Interface
0x4B – 0x4F	GP20 – GP24	Read only	User specific I/O
0x50 – 0x53	<i>Reserved</i>		
0x54	Parameter Page Select	Read write	Config Parameter I/O Interface
0x55	Host Interface Control	Read write	Fuser Core Config & Status
0x56 – 0x5B	GP31 – GP36	Read write	User specific I/O
0x5C – 0x63	Parameter Write Buffer[0:7]	Read write	Config Parameter I/O Interface
0x64	Parameter Request	Read write	Config Parameter I/O Interface
0x65 – 0x6B	GP46 – GP52	Read write	User specific I/O
0x6C – 0x6F	Host IRQ Timestamp	Read only	Fuser Core Config & Status
0x70 – 0x71 0x72 – 0x73	ROM Version RAM Version	Read only	Fuser Core Config & Status
0x74 – 0x8F	<i>Reserved</i>		

0x90 0x91	Product ID Revision ID	Read only Read only	Chip specific IDs
0x92 – 0x93	<i>Reserved</i>		
0x94 – 0x95 0x96 0x97 – 0x9A	Upload Address Upload Data Upload CRC	Read write Read write Read only	Fuser Core Config & Status (Firmware upload interface)
0x9B	Reset Request	Read write	Fuser Core Config & Status

## 6. Device Initialization and Startup

The procedure in order to initialization and startup the BHA until it reaches its normal operation mode consist mainly of the following steps:

1. Power on or reset the device
2. Wait for Interrupt
3. Upload the Firmware (RAM patch)
4. Switch into main execution mode
5. Wait for Interrupt
6. Configure the sensors and meta events
7. Configure the FIFO buffers
8. Configure the host interrupt setting

Once this procedure is successfully finished, the host can go into sleep mode and wait for the BHA's interrupt, according to the defined conditions (see step 6).

If the host receives an interrupt request from the BHA250, it can simply read out the FIFO buffer and parse the obtained data. (See section 13 for details on how to read the FIFO buffer)

### 6.1 Reset

The BHA does not provide a specific hardware pin for a reset. A reset can be triggered due to

- Power On Reset
- Watchdog Reset
- Host initiated Reset Request

In order to trigger a reset request, the host has to write a 1 into the Reset\_Request register (Address 0x9B in the register map). This bit automatically clears to 0 after reset.

### 6.2 Boot Mode

The ROM is split into two parts, a small boot loader and the larger set of libraries and drivers which can be used by a RAM-based firmware or "patch."

It is this latter part of the ROM which provides most of the functionality required for sensor fusion, host interface interactions, data batching, and so on. However, without a RAM patch, none of these more advanced behaviors can occur. This is where boot loading comes in.

When the BHA first comes out of reset it executes the ROM boot loader. The boot loader performs the default initialization of the BHA, apply factory trim values, initialization of the host interrupt line, etc, generates an interrupt request to the host and goes into halt mode.

In halt mode, the host may directly load a RAM patch using the firmware update interface registers (Address 0x94-0x9A in the register map) in the Fuser Core Config & Status block.

After the firmware upload procedure is finished successfully, the host can switch the BHA250 into the main execution mode by writing a 1 to bit 0 (CPU\_Run\_Request) of the Chip Control register (Address 0x34). A successful execution of the CPU\_Run\_Request can be detected by checking the RAM Version registers (Address 0x72-0x73). Before execution of the RAM patch, the RAM Version registers will contain 0.

### 6.3 Main Execution Mode

Once in this mode, the full Android host interface and sensor suite is available. The BHA indicates its readiness by inserting an initialized meta event in the FIFO. The host should wait for this before attempting to query or configure sensors or other features.

If an incorrect RAM patch has been loaded (for example, is built for a different sensor suite), the FIFO will instead contain one or more Sensor Error or Error meta events.

In the nominal case, however, the host is now free to query which sensors are present by reading the Sensor Status bits, learn the details of each sensor by querying the Sensor Information parameters, load any Warm Start values using the Algorithm Warm Start parameters, and/or configure sensors to start generating output using the Sensor Configuration parameters.

The host may also wish to configure which meta events will appear in the FIFOs, such as FIFO Overflow, Watermark, or many others. It can specify whether certain meta events can cause an immediate host interrupt, or are batched until later.

Finally, the host may wish to configure the optional Watermark values using the FIFO Control parameter. This allows the host to be informed that either one or both of the FIFOs have reached a level at which the host shall read its contents to avoid data is loss. This is especially useful when the Application Processor is asleep.



## 7. Device Configuration

A set of registers (the Fuser Core Config & Status block) can be used to configure the fundamental behavior of the CPU core and the host interface (see section 10 for a detailed description of the specific registers). Besides this basic configuration, the full flexibility of the BHA is offered through the Configuration Parameter I/O interface.

The Configuration Parameter I/O interface, is provided through registers of the BHA and consists of

- Parameter\_Read\_Buffer[0:15] (0x3B – 0x4A)  
in order to read a specific parameter set out the BHA's config parameter area
- Parameter\_Write\_Buffer[0:7] (0x5C – 0x63)  
in order to write a specific parameter set into the BHA's config parameter area
- Parameter\_Page\_Select (0x54), Parameter\_Request (0x64), Parameter\_Acknowledge (0x3A)  
for the required handshaking.

In general, the Configuration Parameter I/O interface basically copies a specific parameter set either from the parameter area into the read buffer (read access) or from the write buffer into the specified parameter area (write access).

The procedure for a **read** access works a follow:

In order to get the a copy of the desired parameter inside the read buffer, the host requests a parameter set by writing the requested page into the Parameter\_Page\_Select register and the desired parameter set into the Parameter\_Request register.

Afterwards the host waits for an acknowledgement, by polling the Parameter\_Acknowledge register until it matches the desired parameter number (or indicates an error). The acknowledgment indicates that the Parameter\_Read\_Buffer has been updated with the values of the requested parameter.

The host can read more parameters within the same page by writing a new Parameter Request register value, polling for a match in the Parameter Acknowledge register, then reading the new parameter's value from the Parameter Read Buffer area.

The host ends the parameter transfer procedure by writing the Parameter Page Select register with 0.

The procedure for a **write** access works a follow:

The host writes the new data for a specific parameter set into the Parameter\_Write\_Buffer. In order to address the specific dataset it writes the desired parameter page into the Parameter\_Page\_Select register and the specific parameter set into the Parameter\_Request register.

Afterwards the host waits for an acknowledgement, by polling the Parameter\_Acknowledge register until it matches the desired parameter number (or indicates an error). The acknowledgment indicates that the Parameter\_Write\_Buffer has been copied inside the addressed parameter set.

The host may write another parameter in the same page by repeating the procedure.

The host ends the parameter transfer by writing a 0 into the Parameter\_Request register.

A detailed description of the various parameters and their organization into several parameter pages is given in section 11 Parameter I/O Description.

## 8. FIFOs and Events

Understanding the concept of FIFOs and Events is fundamental for proper operation of the BHA. Both elements are implemented into the device in order to meet the requirements of Android.

Every piece of information the BHA delivers to the host is treated as an Event and placed into a FIFO.

### FIFOs

BHA provides two FIFOs: a wakeup and a non-wakeup FIFO.

The **non-wakeup FIFO** will never trigger an interrupt request to the host when the host is in sleep mode (in default configuration).

(The host should inform the BHA about using the AP\_SUSPENDED bit (bit 5) in the Host\_Interface\_Control register (0x55))

If the non-wakeup FIFO is full, while the host is in sleep mode, the non-wakeup FIFO is allowed to overflow, discarding the oldest data to make room for new data as they arrive.

The **wakeup FIFO** may trigger an interrupt request, depending on the current configuration, for different reasons, even if the host is in sleep mode. One obvious reason is to avoid, that the wakeup FIFO overflows (configured by the wakeup FIFO watermark level setting) or the events in the FIFO become too old (configured by the max report latency setting).

There are more reasons, see section 11 and 13 for further details.

### Events

In order to implement a generalized and efficient handling mechanism for sensor data the concept of sensor events is used within the BHA.

A sensor event consists of the sensor ID of the virtual sensor generating the event and, the data according to the data type of the specific sensor. It is placed into a FIFO, when it occurs.

Events can be generated continuously, e.g. if a virtual sensor is setup to produce data samples on a configured data rate, or as single events, e.g. when a step or significant motion is detected.

To make use of the event concept in a generalized way, the virtual sensor IDs – which are originating from (and thus are identical to) the virtual sensor definitions in the Android CDD – are extended by additional IDs not necessary related to virtual sensors.

In a first step, each virtual sensor gets a second sensor ID in order to distinguish wakeup from non-wakeup events.

In a second step, additional event IDs are introduced in order to handle non sensor related information, like timestamps and meta events.

A detailed description of all available event IDs is provided in the following sections.

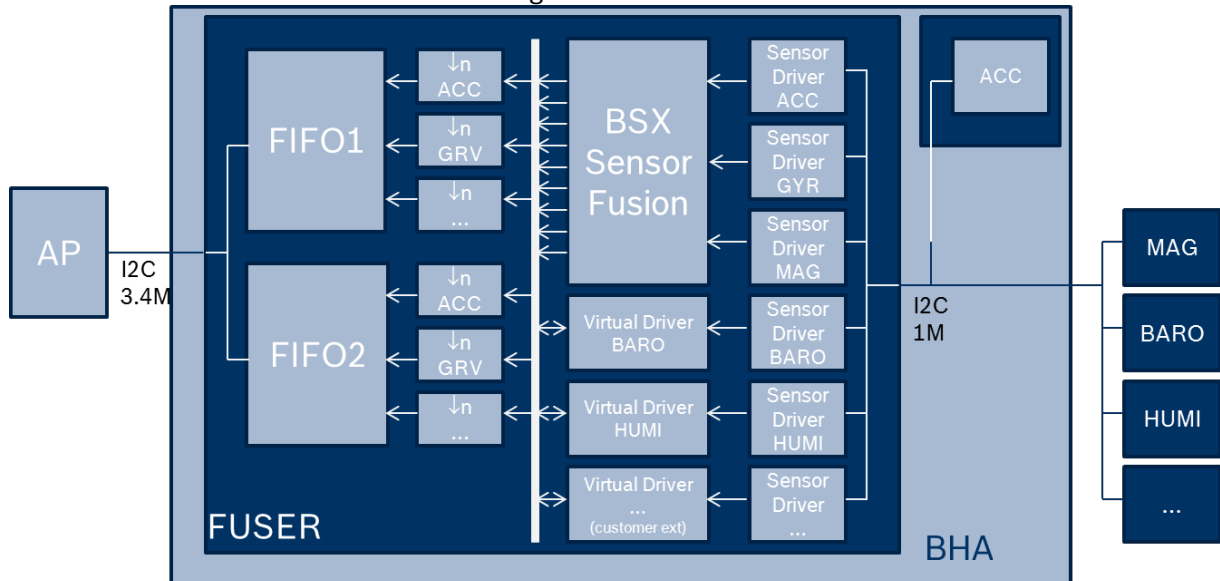
Using this event concept, the BHA's output data will be sent in a continuous stream, with each event (e.g. a sensor sample) uniquely identified. Because many sensors produce output at the same time, the timestamp event is only introduced once into the FIFOs at the start of a series of sensor samples that occurred at the same time. This saves space in the buffer.

## 9. Functional description

### 9.1 Dataflow of Sensor Fusion

The integrated Fuser Core receives *raw sensor data* from the connected sensors and provides *calibrated (virtual) sensor data* to the application processor. The raw sensor data flows from the sensor with a maximum ODR 200 Hz to the fuser core. The BSX library runs sensor fusion (when required) using this high speed data, to avoid loss in signal quality. The results are subsampled to the ODR required by the host processor.

Figure 7: Dataflow



### 9.2 Supported data rates of BSX Sensor Fusion Engine

The following output data rates configuration can be selected by the host processor, these are support in both sensor only and data fusion operating modes:

Table 10: Supported BSX output data rates

BSX output data rate
200Hz
100Hz
50Hz
25Hz
12.5Hz

The actual output data rate requested by Android will be provided according to the Android requirements and derived from the above mentioned internal data rates. I.e., the actual output data rate will be in the range of 90%...210% of the requested data rate. Output samples are generated by subsampling from a suitable data rate from Table 10.

If multiple virtual sensors with different output data rates are requested by Android, the internal data rate will be selected such that all output data rates can be generated according to the Android requirements.

### 9.3 Gesture recognition

Android defines 3 gestures have to be implemented in the system, but leaves the functional implementation open to the device provider. The BHA allows individual gestures to be implemented and mapped on the above mentioned system gestures. By default the firmware of the BHA performs the following mapping of gestures to the virtual gesture sensors:

- **Wakeup Gesture**

Double Tap on the device

- **Pickup Gesture**

Pickup the device from a surface (table) and hold it in a 45° angle in relation to gravity

- **Glance Gesture**

Move (Slide) the phone left and right on the surface (table) without lifting it up

### 9.4 Power Modes and Current Consumption

The power modes of the Fuser Core and the connected sensors are configured automatically, depending on which virtual sensors are requested by the host, and the resulting fusion mode.

After startup, the list of requested sensors is empty, i.e. the firmware switches the connected sensors into standby mode and then also sets the processor to sleep. Once a virtual sensor has been requested by the host, the requested physical sensors is enabled and the BSX library is set into a working mode that supports the requested sensors.

In general, the uC can be in operation or in sleep. Each interrupt will wake the uC out of sleep, e.g. to process a new data sample from a sensor. When the processing is complete the uC returns to sleep again.

The current consumption of the Fuser Core in sleep mode is ~7µA, in full operation it is ~800µA. The actual average current consumption therefore depends of the amount of time the uC is in full operation mode, which in turn depends on the selected operation mode.

#### Current Consumption per operation mode

The following exemplary average current consumptions can be reached in the various operation mode of the BHA, within an isolated use case consideration. The total value includes both the processing in the Fuser Core and the MEMS Sensor power consumption (including the intrinsic ASIC and an estimate for the external magnetometer sensors, where required).

Please note that there is no linear addition of the exemplary values given in the following table if the use cases are not isolated but combined. In this case the resulting total current consumption is always lower than its single fractions.

Table 11: Power consumption vs. operating mode

		<b>Current Consumption (µA Typical)</b>
<b>Use Case</b>	<b>Device</b>	<b>BHA250</b>
Significant Motion	Accelerometer	50
	Fuser Core	50
	<b>Total</b>	<b>100</b>
Step Counting	Accelerometer	50
	Fuser Core	50
	<b>Total</b>	<b>100</b>
Activity recognition	Accelerometer	50
	Fuser Core	150
	<b>Total</b>	<b>200</b>
Rotation Vector eCompass	Accelerometer	200
	Magnetometer	300
	Fuser Core	300
	<b>Total</b>	<b>800</b>
Standby	Accelerometer	3
	Magnetometer	1
	Fuser Core	7
	<b>Total</b>	<b>11</b>

## 9.5 Virtual Sensors

Virtual sensors are the interface to the Android application layer and are directly requested from there. The BHA supports all Virtual Sensors as defined in the Android CDD. Based on the Android specification virtual sensor are completely independent. So each virtual sensor has its own data rate (delay), type, and trigger mode.

Virtual sensors are implemented as software modules within the firmware running on the Fuser Core. Each virtual sensor SW module may access a physical sensor via its sensor driver, or it may use other SW modules (e.g. the BSX library) in order to derive processed data based on physical sensors.

The virtual sensors generate sensor events, which may be continuously (e.g. samples at a configured data rate) or single events (on-change, one-shot, or special; e.g. when a step or significant motion has been detected). These events are represented as data packets of a specific virtual sensor data type and are put into the output FIFO of the BHA.

Each of the sensors will be supported as wakeup and non-wakeup sensor and has a fixed ID which allows distinguishing a wakeup from a non-wakeup version. Each version of a sensor has independent sample rate and report latency values.

The supported virtual sensors are listed the following table:

Table 12: virtual sensor IDs

Virtual Sensor	ID (Non Wakeup)	ID (Wakeup)	VS Data Type	In ROM Library
VS_TYPE_ACCELEROMETER	1	33	Vector+	X
VS_TYPE_GEOMAGNETIC_FIELD	2	34	Vector+	X
VS_TYPE_ORIENTATION	3	35	Vector+	X
VS_TYPE_GYROSCOPE	4	36	Vector+	X
VS_TYPE_LIGHT	5	37	Abs_Scalar	
VS_TYPE_PRESSURE	6	38	Abs_Scalar_long	
VS_TYPE_TEMPERATURE	7	39	Abse_Scalar	
VS_TYPE_PROXIMITY	8	40	Abs_Scalar	
VS_TYPE_GRAVITY	9	41	Vector+	X
VS_TYPE_LINEAR_ACCELERATION	10	42	Vector+	X
VS_TYPE_ROTATION_VECTOR	11	43	Quaternion+	X
VS_TYPE_RELATIVE_HUMIDITY	12	44	Abs_Scalar	
VS_TYPE_AMBIENT_TEMPERATURE	13	45	Abs_Scalar	
VS_TYPE_MAGNETIC_FIELD_UNCALIBRATED	14	46	Vector_Uncalibrated	X
VS_TYPE_GAME_ROTATION_VECTOR	15	47	Quaternion+	X
VS_TYPE_GYROSCOPE_UNCALIBRATED	16	48	Vector_Uncalibrated	X
VS_TYPE_SIGNIFICANT_MOTION	17	49	Sensor_Event_Data	X
VS_TYPE_STEP_DETECTOR	18	50	Sensor_Event_Data	X
VS_TYPE_STEP_COUNTER	19	51	Abs_Scalar	X
VS_TYPE_GEOMAGNETIC_ROTATION_VECTOR	20	52	Quaternion+	X
VS_TYPE_HEART_RATE	21	53	Abs_Scalar_short	
VS_TYPE_TILT	22	54	Sensor_Event_Data	X
VS_TYPE_WAKEUP	23	55	Sensor_Event_Data	
VS_TYPE_GLANCE	24	56	Sensor_Event_Data	
VS_TYPE_PICKUP	25	57	Sensor_Event_Data	
VS_TYPE_ACTIVITY_RECOGNITION	31	63	Sensor_Activity_Rec_Data	X

Virtual sensors as defined in Table 12 can be customized and optimized by a firmware update even if they are originated from the write protected ROM library. Furthermore, an extension by additional virtual sensors which are not already included in the library can be added to the RAM by a firmware upgrade, if the necessary physical sensors are available on the PCB and connected to external I2C interface. A complete description of the sensor types, including non-sensor related IDs (meta events, timestamps), is provided in section 12. For details and technical support please refer to corresponding application notes or contact our regional offices, distributors and sales representatives.

## 9.6 Virtual Sensor Data Types

Depending on the nature of the sensor and the Android specification different data types are used to represent the sensor data efficiently. These sensor data types are:

Table 13: Virtual sensor data types

Virtual Sensor Data Type	Data Values	Format
Quaternion+	X Y Z W Estimated Accuracy	See 12.1 for detailed description
Vector+	X, scaled Y, scaled Z, scaled Meas Accuracy Status	See 12.2 for detailed description
Vector_Uncalibrated	X Uncalibrated, scaled Y Uncalibrated, scaled Z Uncalibrated, scaled X Bias Y Bias Z Bias Meas Accuracy Status	See 12.3 for detailed description
Scalar	Signed scalar value (16 bit signed int)	See 12.4 for detailed description
Scalar	Absolute scalar value, scaled (16 bit signed int)	See 12.4 for detailed description
Abs_Scalar	Absolute Scalar value (16 bit unsigned int)	See 12.4 for detailed description
Abs_Scalar_short	Absolute scalar value (8 bit unsigned int)	See 12.4 for detailed description
Abs_Scalar_long	Absolute long scalar value, scaled (24 bit unsigned int)	See 12.4 for detailed description
Sensor_Event_Data	Sensor_Event (Parameterless)	See 12.5 for detailed description
Sensor_Activity_Rec_Data	Activity_State	See 12.6 for detailed description

A detailed description of the available sensor IDs and the data formats is provided section 12. This includes information on the scale factors of the scaled data.

## 9.7 Sensor Configuration

Each sensor needs to be configured first, before it can be used in the system. The configuration includes at least the activation of the sensor, but there are more options available:

- Activate – enable or disable the sensor: Boolean
- Batch – enable by setting the maximum report latency or disable by setting to zero: Integer (nanosecond resolution or larger)
- Delay (Sample Period) – must be equal or less than the requested value, but no smaller than the Delay / 2: Integer (nanosecond resolution or larger)
- Flush – send all batched samples (if any) immediately, then send a special meta data type indicating the flush is complete
- Poll – read a specified number of samples: Integer
- Wakeup – if a sensor is opened as a wakeup sensor (using a flag), then it is allowed to interrupt the host, either when each sample is available or after the maximum report latency. If opened as a non-wakeup sensor, then they must not interrupt the host. Data for the wakeup vs. non-wakeup sensors go in separate FIFOs.

These are combined into the following settings in the BHA hardware:

- Sample rate: set to 0 to disable the sensor: unsigned 16 bit integer, in Hz; this is 1 / sample period; *on-change, one-shot, and special sensors should be set to a non-zero value to enable them or 0 to disable.* The sample rate of the non-wake-up and wake-up sensor of the same type can be set independently.
- Max report latency: set to 0 for non-batch mode, nonzero for batch mode; if nonzero, sample period must also be nonzero: unsigned 16 bit integer, in milliseconds. The max report latencies of the non-wake-up and wake-up sensor of the same type can be set independently.
- Wakeup vs. non-wakeup: this is implemented using a special bit in the Sensor ID (IDs > 32 are wakeup).
- Flush sensor: flush the samples for a specific sensor's FIFO or all sensor data. See 10.2 for more details.
- Change sensitivity: unsigned 16 bits; same scaling as the sensor's corresponding data value (for future Win8/10 compatibility)
- Dynamic range: unsigned 16 bits; specified in terms of commonly used units such as g-s for accelerometers and degrees / second for gyroscopes (for custom use)

The configuration of the sensor is performed using the Config Parameter I/O (sensors parameter page) interface and is described in section 11.2 in detail. Notes:

- Lollipop does not require the ability to set the dynamic range or resolution of any sensors. The only configurable items at the HAL to sensor driver interface are below. However, the BHA does provide a dynamic range setting for custom use.
- Since the HAL / driver will be given the sample rate (period) and max report latency at the same time, and since they map to 8 bytes of data, they can be sent to the BHA in a single 8 byte parameter write. Each unique sensor (physical or virtual) would be assigned a unique Parameter number.
- Writing this Parameter modifies the values; reading this Parameter returns the actual sample rate and actual report latency. The actual sample rate will be in the range of 90% ... 210% of the requested sample rate. While this ability to query the actual sample rate is not required by Lollipop, it is useful for external testing, debugging, and non-Lollipop applications. If the requested sensor is not present, the returned sample rate and report latency will be 0.
- The flush mechanism is done using a single 8 bit GP register. Android OS defines a special meta data value to be placed in the buffer to indicate the flush of a specific sensor has been completed. So this meta data information goes into FIFO.



## 9.8 Sensor Status Information

Android sensor drivers need to make the following information available about each supported sensor. This information are provided by the BHA via a sensor status information structure, available also through the Config Parameter I/O (sensors parameter page) interface (see section 11.2 for details).

The available sensor status information parameters are:

- Name – unique; if there are multiple sensors in the system of the same type, each one must have a unique name; this can be derived from the BHA sensor's driver ID and slave I2C address
- Vendor – vendor of the underlying HW; this can be derived from the BHA sensor's driver ID
- Version – version of the HW + driver; must change when the driver's output changes in some way. It is derived from driver version reported by BHA plus driver-specific version information
- Type – sensor type, e.g., `SENSOR_TYPE_ROTATION_VECTOR`; same as sensor data packet type
- maxRange – the maximum possible sensor value in SI units (e.g. an accelerometer set for a 4g range would report 4g here); derived from sensor dynamic range
- resolution – smallest difference between two values reported by this sensor derived from sensor dynamic range and bits of resolution (e.g., an accelerometer at 4g range and 16 bit signed values could report  $4g/32767 = 1.2 \times 10^{-4} g$ )
- power – rough estimate of sensor's power consumption in 0.1 mA; this appears to be only queried at reboot by Android, and is defined to be the maximum power consumed when in use
- minDelay – continuous sensors report minimum period in microseconds; on-change sensors report 0; one-shot sensors report -1
- maxDelay – continuous sensors report maximum period in microseconds
- fifoReservedEventCount – number of events reserved for this sensor in the batch FIFO; since a single FIFO for all sensors is used in the BHA, this means that no area is reserved specially for any one sensor, so this returns 0
- fifoMaxEventCount – maximum number of events that could be batched; since the FIFO is shared, this is the size of the FIFO in bytes divided by the number of bytes per sample
- flags - only one is defined -- wakeup

In the BHA, these are combined into the following fields:

- Sensor Type: unsigned 8 bits
- Driver ID: unsigned 8 bits
- Driver Version: unsigned 8 bits
- Max Range: signed 16 bits; scaled the same as the sensor's corresponding data value
- Resolution: signed 16 bits; number of bits per sensor (axis) sample
- Power: unsigned 8 bits; multiples of 0.1 milliamps
- Max Rate: unsigned 16 bits; rate in Hz
- Min Rate: unsigned 8 bits; rate in Hz
- FIFO Reserved: unsigned 16 bits; 0
- FIFO Max: unsigned 16 bits; total FIFO size in bytes divided by size of data value

- Status Bits: unsigned 8 bits; while not required by Lollipop, for testing and debugging the BHA provides bit flags indicating:
  - data\_available
  - i2c\_nack
  - device\_id\_error
  - transient\_error (e.g., magnetic transient)
  - data\_lost (FIFO overflow)
  - sensor\_power\_mode (shutdown, standby, low power active, high power active)

**NOTE:** the Status Bits will change dynamically at run time, but the other fields are fixed and only read by the Android HAL once at boot. Therefore, these per-sensor Status Bits are available through a separate mechanism than the static information, via the sensor status banks, described in section 11.1.

The remaining fields add up to 15 bytes of information. These values, plus the size of a given sensor's sample in the FIFO, can be read by the host by reading the Sensor Information structure for a specific sensor, as described in section 11.3.

Since not all of the possible sensor types will be present in all builds (e.g., one mobile device might have a barometer but another may not), the host can query this data for all possible sensors, and based on the returned data, know which sensors are present. If a sensor is not present, all status fields will be returned with a value of 0. Alternatively, the host can query the Sensor Status Bits for each sensor; those sensors that are not available will return 0 for the power mode, indicating sensor not present.

## 9.9 FIFOs

Since sensors in wakeup and non-wakeup versions are supported from Android Lollipop onwards, the BHA provides two FIFOs.

Each FIFO has independent watermarks, interrupt control, and flush requests.

Under certain conditions, i.e. based on the host interrupt configuration settings (e.g. watermark level of the FIFO, max delay of a sensor has exceeded; see section 11 for further details), the BHA will raise an interrupt request to the host processor. The driver of the host processor can then fetch data from the BHA FIFO.

Data will be delivered to the host as a single burst, starting with the events of the wake-up FIFO, followed by the events of the non-wake-up FIFO, unless the output was triggered by a flush request, in which case, the type of sensor requested in the flush determines which FIFO is delivered.

**Meta events** not related to a specific sensor will only be placed in the non-wakeup FIFO. These meta events are

- Error
- Self-Test Results
- Initialized.

However, these meta events are enabled by default and they are configured by default to trigger host interrupt request.

Meta events related to a specific physical sensor, such as the

- Sample Rate Changed
- Power Mode Changed
- Dynamic Range Changed

are always placed to the non-wakeup FIFO.

**Timestamps** will be inserted in both FIFOs and maybe in a non-continuous manner, i.e. for certain cases a timestamp with a lower value can follow a timestamp with a higher value.

## 9.10 Non-Batch Mode

Any sensor with zero latency or batching timeout will be reported as soon as it is detected. This will of necessity result in many small FIFO transfers and an interrupt rate as high as the fastest non-batched sensor's sample rate. This may in reality be even more often than one might expect, due to slight variations in sensor sample rates.

The host can minimize interrupts while ensuring timely transfer of sensor data by setting all but the fastest enabled sensor to have non-zero latency, large enough to not timeout before the next sample of the fastest sensor. In this configuration, data transfers from the FIFO will occur at the rate of the fastest sensor, with all slower sensors transferred together.

For example, if the Accelerometer is set for 60 Hz, and the Gyroscope and Magnetometer are set to 20 Hz, and the Gyroscope and Magnetometer are each set with a latency timeout of 50 ms, host transfers will occur like this:

1. Accel 1
2. Accel 2
3. Accel 3, Gyro 1, Mag 1
4. Accel 4
5. Accel 5
6. Accel 6, Gyro 2, Mag 2
7. ...

## 9.11 Batch Mode

The BHA fully supports Android Lollipop & Marshmallow (non-HiFi) batching requirements. Each sensor type has an independently settable latency or batching timeout.

The batched sensor data and other meta data is stored in a RAM-based FIFO. The size of the FIFO depends on the remaining available RAM after uploading of the RAM patch into the BHA.

The BHA implements a single shared wakeup FIFO for wakeup sensors, and a single shared non-wakeup FIFO for non-wakeup sensors. As such, whichever sensor's batching timeout expires first will cause all events in the FIFOs to be sent to the host.

The BHA supports a host-settable watermark value for each FIFO, which is used to ensure that batched data is not lost due to FIFO overflow, when the AP is outside of the suspend mode.

When the host is in suspend mode, the non-wakeup FIFO is allowed to overflow. The BHA will discard the oldest data to make room for new data as it arrives. As soon as the host leaves suspend mode, the BHA will request a transfer of the entire contents of both FIFOs to the host.

## 10. Register Map Description

### 10.1 Buffer\_Out[0:49]

#### Register Address (0x00 .. 0x31) – Read only

This range of 8 bit registers is used for data transfers from the FIFO to the host. Access to this area must be done in a specific manner as described in section 13. The general procedure is, however, that the host must read the Bytes\_Remaining register to determine the current number of pending bytes in the FIFO and reads afterwards this amount of bytes from this Buffer\_Out register area.

(0x00 .. 0x31) Bit	Name	Description
Bit 7..0	Buffer_out	These registers are used as a freeform streaming output buffer area for reading FIFO data

### 10.2 FIFO\_Flush

#### Register Address (0x32) – Read write

This allows the host to request that a single sensor's FIFO (batch mode) be flushed, or all sensors. This is an optional mechanism; if the host does not use this register, then the watermark, sensor latency, and wakeup and non-wakeup FIFO interrupt disable bits, as well as which sensors are enabled, determine when the host interrupt occurs and whether the data stream will include both the wakeup and non-wakeup FIFOs.

(0x32) Bit	Name	Description
Bit 7..0	FIFO_Flush	Sensor ID, or special value 0xFF to flush all

#### FIFO\_Flush - Enumerated Values

Value	Name	Description
0x00	NOP	No operation
0xFF	FLUSH_ALL	flush all samples for both FIFOs

### 10.3 Chip\_Control

#### Register (0x34) – Read write

This register provides bits that control fundamental behavior of the chip.

(0x34) Bit	Name	Description
Bit 7..2	-	(fixed to 0)
Bit 1	HOST_UPLOAD_ENABLE	controls the RAM patch upload mechanism
Bit 0	CPU_RUN_REQUEST	controls whether the CPU is running or not

### 10.4 Host\_Status

#### Register (0x35) – Read only

Provides status information to the host.

(0x35) Bit	Name	Description
Bit 7..5	ALGORITHM_ID	Algorithm ID
Bit 4..2	HOST_IF_ID	Host Interface ID: 0 = Android K 1 = Android L (et sqq.)
Bit 1	ALGORITHM_STANDBY	Algorithm Standby will be set to confirm that the host's previous write of a 1 to the Algorithm Standby Request bit in the Host Interface Control register has taken effect.
Bit 0	RESET	Reset is set after power-on reset or reset invoked by means of the Reset Request register.

#### ALGORITHM\_ID - Enumerated Values

Value	Name	Description
0	BSX	Bosch Sensortec BSX Fusion Library

## 10.5 Int\_Status

### Register (0x36) – Read only

This provides an alternative way for the host to determine the host interrupt status of the device, if the physical interrupt line is not used.

**NOTE:** the time at which the host interrupt was asserted can be queried via the Host IRQ Timestamp parameter of the System parameter page.

The Host Interrupt bit reflects the state of the host interrupt GPIO pin. The Wakeup and Non-Wakeup Watermark bits are set if the watermark for their respective FIFOs was reached. The Wakeup and Non-Wakeup latency bits are set if a timeout on a sensor in their respective FIFOs expired. The Wakeup and Non-Wakeup Immediate bits are set if a sensor event has occurred which was configured with no latency.

(0x36) Bit	Name	Description
Bit 7	Reserved	
Bit 6	Non-Wakeup Immediate	
Bit 5	Non-Wakeup Latency	
Bit 4	Non-Wakeup Watermark	
Bit 3	Wakeup Immediate	
Bit 2	Wakeup Latency	
Bit 1	Wakeup Watermark	
Bit 0	Host Interrupt	

## 10.6 Chip\_Status

### Register (0x37) – Read only

This register reflects fundamental behavior of the chip during boot up.

(0x37) Bit	Name	Description
Bit 7..5	-	(fixed to 0)
Bit 4	NO_EEPROM	No EEPROM
Bit 3	FIRMWARE_IDLE	Firmware Idle (halted)
Bit 2	EE_UPLOAD_ERROR	EEUploadError
Bit 1	EE_UPLOAD_DONE	EEUploadDone
Bit 0	EEPROM_DETECTED	EEPROM Detected

## 10.7 Bytes\_Remaining[0:1]

### Registers (0x38 - 0x39) – Read only

This 2x 8 bit register pair indicates how many bytes are available in the FIFO buffer. It forms a 16 bit value and shall be read in one access in order to get the correct result. The value can vary from the size of the smallest single FIFO event (a sensor sample of other event type) to the combined size of both FIFOs. The maximum FIFO sizes can be queried using the FIFO Control Parameter in the System Parameter Page.

The value of this register pair is updated by the BHA only at the following times:

1. Immediately prior to asserting the host interrupt
2. Upon demand, i.e. after the host writes a 1 to the Update Transfer Count bit of the Host Interface Control register.

During normal operation, i.e. when the host receives an interrupt from the BHA, the host should read these Bytes\_Remaining registers, and use the provided value to read the amount of bytes from the FIFO.

If all bytes are read, the BHA will de-assert the host interrupt line, in order to acknowledge that all data announced by the Bytes\_Remaining register to the host, have been read.

If new data arrive in the FIFOs, while the host is reading the FIFO the BHA, will update the Bytes\_Remaining registers and reassert the host interrupt (depending on the configured settings for creating a host interrupt). This could occur immediately after the acknowledge, or later in time.

(0x38 - 0x39) Bit	Name	Description
Bit 15..0	Bytes_Remaining	Available Bytes in FIFOs

## 10.8 Parameter\_Acknowledge

### Register (0x3A) – Read only

This register is used to acknowledge a parameter read/write request, from the host. I.e. a host write to the Parameter\_Page\_Select and the Parameter\_Request register. The host should poll the Parameter\_Acknowledge register, until it matches the Parameter\_Request register, or it indicates an error providing the value 0x80. The error value means that the requested parameter page or parameter number is unsupported.

(0x3A) Bit	Name	Description
Bit 7..0	Parameter_Acknowledge	Parameter Acknowledge

## 10.9 Parameter\_Read\_Buffer[0:15]

### Registers (0x3B .. 0x4A) – Read only

This 8 bit register area provides an interface to the host for reading requested parameter out of the various BHA's parameter pages.

**NOTE:** The Parameter\_Read\_Buffer register area is large enough to report an entire sensor status structure in one transfer.

(0x3B .. 0x4A) Bit	Name	Description
Bit 7..0	Parameter Read Buffer	Parameter Read Buffer

## 10.10 GP[20:24]

### Registers (0x4B .. 0x4F) – Read only

This is a read only register area available for custom specific extensions. They are all read-only from the I2C host but writable from the Fuser Core MCU.

## 10.11 Parameter\_Page\_Select

### Register (0x54) – Read write

This register is used to select a parameter page for read/write access.

The least significant nibble contains the parameter page number, described below.

The most significant nibble contains the desired transfer size in bytes.

If 0 is selected for the transfer size, the max values for the transfer size (16 bytes for reading, 8 bytes for writing) are selected. The size will be limited to the max values, in case the host specifies larger values.

(0x54) Bit	Name	Description
Bit 7..4	PARAMETER_SIZE	desired transfer size in bytes or 0 for max size (16 bytes read, 8 bytes write)
Bit 3..0	PARAMETER_PAGE	parameter page number



**PARAMETER\_PAGE - Enumerated Values**

Value	Name	Description
0	PAGE_0	The host must write this value, after finishing an access on the Algorithm Parameter Page, as an Acknowledgment for the BHA, that it is safe to copy back the algorithm data structures.
1	SYSTEM	This page contains parameters which affect the whole system, such as meta event enables, sensor status, FIFO watermark control, etc.
2	ALGORITHM	This page contains all the original algorithm coefficients and knobs. When this is first selected, the CPU makes a safe copy of all necessary algorithm data structures that may be modified using Parameter I/O to this page.
3	SENSORS	This page contains parameters for every sensor (real or virtual), both for reading their status and for configuring their operation.
12	CUSTOM_12	These can be used by customers for any purpose. See appendix A.
13	CUSTOM_13	These can be used by customers for any purpose. See appendix A.
14	CUSTOM_14	These can be used by customers for any purpose. See appendix A.

## 10.12 Host\_Interface\_Control

### Register (0x55) – Read only

This register can be used by the host in order to control miscellaneous features of the BHA250, as described in the following table.

**NOTE:** Abort Transfer and Update Transfer Count bits do not auto-clear. It is up to the host to set these two bits correctly every time it writes this register. However, due to possible race conditions, it should not clear any of these bits immediately after setting.

(0x55) Bit	Name	Description
Bit 7	NON_WAKEUP_FIFO_HOST_INTERRUPT_DISABLE	
Bit 6	REQUEST_SENSOR_SELF_TEST	Is used by the host to inform the BHA, that a self-test should be performed when transitioning out of standby. Any physical sensor driver, that implement self-test control, will request it and report a Self-Test Results meta event with the results
Bit 5	AP_SUSPENDED	Affects the BHA behavior in issuing a host interrupt. When true, only wakeup sensor events may wake the AP. When false, any sensor event may trigger a host interrupt according to the configured conditions.
Bit 4	NED_COORDINATES	Selects the North East Down coordinate system instead of the default Android East North Up (ENU) system
Bit 3	WAKEUP_FIFO_HOST_INTERRUPT_DISABLE	Is a master interrupt disable bit; setting this bit de-asserts the host interrupt and prevents further interrupts, while clearing this bit (the default state) allows it to be asserted whenever a proper condition occur. This controls interrupt generation due to the wakeup FIFO.
Bit 2	UPDATE_TRANSFER_COUNT	Can be used by the host to request a new value to be written to the Bytes Remaining registers, such that data that has arrived

		<p>since the last time Bytes Remaining was written, and data that has been removed, shall be accounted for. However, this does not extend the length of any pending or on-going transfer. It is merely an approximation of how much more there is in the FIFO.</p>
Bit 1	ABORT_TRANSFER	<p>Indicates the host does not intend to complete reading out the FIFO; all pending data is discarded, as well as any partial sensor sample that remains. The host interrupt line is deasserted and the Bytes Remaining is set to 0. If there is more data in the FIFO, the BHA will soon request another transfer. It is up to the host to recover properly from this request.</p>
Bit 0	ALGORITHM_STANDBY_REQUEST	<p>Requests the algorithm to prepare itself to pause (if required by the implemented algorithm), then shuts down all sensors in order to save power. When this bit is deasserted, any sensors previously enabled by the host will be restarted, and the operation of the algorithm will resume. This is a simpler way to temporarily conserve power without requiring the host to disable all active virtual sensors individually.</p>

### 10.13 GP[31:36]

#### Registers (0x56 .. 0x5B) – Read write

This is a read/write register area available for custom specific extensions. This range is writeable from the I2C host and readable by the Fuser Core MCU.

### 10.14 Parameter\_Write\_Buffer[0:7]

#### Registers (0x5C .. 0x63) – Read write

This 8 bit register area provides an interface to the host for writing specific parameter into one of the various BHA's parameter sets.

In order to write a specific parameter, the host should follow the procedure as already written in the description of the Parameter\_Read\_Buffer.

**NOTE:** The configuration data for each sensor takes 8 bytes.

(0x5C .. 0x63) Bit	Name	Description
Bit 7..0	Parameter_Write_Buffer	Parameter Write Buffer area

### 10.15 Parameter\_Request

#### Register (0x64) – Read write

This register is used to read or write parameter from or to the BHA. In order to read or write a specific parameter set, the host should follow the procedure as already written in the description of the Parameter\_Read\_Buffer.

**NOTE:** Having the Parameter Acknowledge reset to 0, allows the host to determine on the next parameter I/O request whether the request was successful.

(0x64) Bit	Name	Description
Bit 7	Request	Direction of the operation
Bit 6..0	Parameter	Parameter page select for saving or writing

#### Request - Enumerated Values

Value	Name	Description
0	Read	Read parameter page
1	Write	Write parameter page

## 10.16 GP[46:52]

### Registers (0x65 .. 0x6B) – Read write

This is a read/write register area available for custom specific extensions. This block is writeable by the I2C host and readable by the Fuser Core MCU.

## 10.17 ROM\_Version[0:1]

### Registers (0x70 - 0x71) – Read only

This 2x 8 bit register pair contains the software version number corresponding to the code placed in ROM and in the RAM firmware patch, if any. If none is present, this will read back 0.

(0x70 - 0x71) Bit	Name	Description
Bit 15..0	Rom_Version	ROM version number

### Rom\_Version – Enumerated Values

Value	Name	Description
0x2112	FUSER1_C2	FUSER1_C2 BHA250
0x2DAD	FUSER1_C3	FUSER1_C3 BHA250B

## 10.18 RAM\_Version[0:1]

### Registers (0x72 - 0x73) – Read only

This 2x 8 bit register pair contains the software version number corresponding to the RAM firmware patch, if any. If none is present, this will read back 0.

(0x72 - 0x73) Bit	Name	Description
Bit 15..0	Ram_FW_Version	RAM patch number

## 10.19 Product\_ID

### Register (0x90) – Read only

This contains the product number of the device.

(0x90) Bit	Name	Description
Bit 7..0	Product_ID	

### Product\_ID - Enumerated Values

Value	Name	Description
0x83	FUSER1_C2	FUSER1_C2, BHA250
0x83	FUSER1_C3	FUSER1_C3, BHA250B

## 10.20 Revision\_ID

### Register (0x91) – Read only

This identifies the hardware revision for the chip.

(0x91) Bit	Name	Description
Bit 7..0	Revision_ID	

### Revision\_ID - Enumerated Values

Value	Name	Description
0x01	val_0x01, di01	FUSER1_C2 BHA250
0x03	val_0x03, di03	FUSER1_C3, BHA250B

## 10.21 Upload\_Address[0:1]

### Register (0x94) – Read write

This 16 bit register lets the host specify the starting address for a RAM patch. By default it is 0. After a RAM upload, it will not be 0, so a subsequent RAM upload procedure will need to start by writing this to 0.

As an exception, this register is Big Endian (i.e. MSB on address 0x94, LSB on address 0x95).

(0x94) Bit	Name	Description
Bit 7..0	Upload_Address_12_8	Upload Address

(0x95) Bit	Name	Description
Bit 7..0	Upload_Address_7_0	Upload Address

## 10.22 Upload\_Data

### Register (0x96) – Read write

Once the host has entered upload mode by writing a 1 to the Host Upload Enable bit of the Chip Control register, it may burst the RAM image to this register.

**NOTE:** The RAM patch file format starts with a 16 byte header which must be skipped. Every 4 bytes of data in the file after the header must be byte swapped before upload to this register.

(0x96) Bit	Name	Description
Bit 7..0	Upload_Data	Upload Data

## 10.23 Upload\_CRC[0:3]

### Registers (0x97 .. 0x9A) – Read only

After the host has transferred all data from the RAM patch file via the Upload Data register into the BHA, the Data CRC register will contain a 32 bit CRC of the data. The host should compare this to a calculated CRC to determine whether the upload was successful.

If the upload was successful, the host should disable upload mode and start firmware execution by writing a 0 to the Host Upload Enable bit and a 1 to the CPU Run Request bit of the Chip Control register.

(0x97 .. 0x9A) Bit	Name	Description
Bit 7..0	Data_CRC	Data CRC

## 10.24 Reset\_Request

### Register (0x9B) – Read write

The host writes a 1 to this register to trigger a hardware reset of the BHA's internal CPU. This bit automatically clears to 0.

(0x9B) Bit	Name	Description
Bit 7..0	Reset_Request	Reset Request



## 11. Parameter I/O Description

Except for a few features which the host can request using the Chip Control and Host Interface Control registers, or which it can query from other registers, the primary control channel for configuring and querying the state of the system and the sensors is done using Parameter I/O.

Parameter I/O in the BHA is implemented using a mail box protocol. This protocol includes a full handshake between the host and the BHA to synchronize access to the data transfer registers; these registers are used to carry control information to the BHA from the host or status information to the host from the BHA.

The following sections describe the parameter pages, relevant to the user of the BHA.

### 11.1 Parameter Page 1: System

These parameters control general system-wide features.

Status banks 0 and 1 are for the non-wakeup sensors, and 2 and 3 are for the wakeup sensors. Meta Event Control parameter 1 is for non-wakeup FIFO meta events, and Meta Event Control parameter 29 is for wakeup FIFO meta events.

Table 14: Parameter Page 1 - System

Parameter Number	Parameter Name	Read Data Buffer	Write Data Buffer
1	Meta Event Control		
2	FIFO Control		
3	Sensor Status Bank 0	Status Bits Sensors 1-16	Not used
4	Sensor Status Bank 1	Status Bits Sensors 17-32	Not used
5	Sensor Status Bank 2	Status Bits Sensors 33-48	Not used
6	Sensor Status Bank 3	Status Bits Sensors 49-64	Not used
7-28	Reserved		
29	Meta Event Control for Wakeup FIFO		
30	Host IRQ Timestamp		
31	Physical Sensor Status	Physical Sensor Status	Not used

## Meta Event Control

The 8 bytes in this writeable parameter will be divided into two bit sections. Each section controls whether the corresponding Meta Event will be enabled (so that it will appear in the output FIFO when it occurs), as well as whether that will lead to an immediate host interrupt.

The MSB in each Bit Range is the event enable, and each LSB is the event interrupt enable.

Table 15:Meta Event Control

Load Parameter Byte	Enable Bit	Int Enable Bit	Meta Event
0	1	0	Meta Event 1
0	3	2	Meta Event 2
0	5	4	Meta Event 3
0	7	6	Meta Event 4
...			
7	1	0	Meta Event 29
7	3	2	Meta Event 30
7	5	4	Meta Event 31
7	7	6	Meta Event 32

## FIFO Control

This parameter provides a mechanism for the host to set a target number of bytes the Wakeup FIFO buffer and/or the Non-Wakeup FIFO buffer should contain before they assert the host interrupt signal.

Set this value to 0 to disable this feature, or a non-zero value to set the watermark. Any value larger than the size of the FIFO will be treated the same as a value exactly equal to the size of the FIFO.

Data loss will likely occur with watermark values that are too high. It is up to the customer to determine, in their application, based on the maximum I2C host rate and host interrupt response time, what a safe maximum watermark level might be.

**NOTE:** a non-zero Watermark has **no effect** if all enabled sensors have 0 latencies (batch timeouts) and the AP is active (the Host Interface Control register's AP Suspend bit is 0). As soon as any one sensor generates a sample, and that sensor's latency is 0, a host interrupt will be generated. The Watermark is only useful when all enabled continuous output sensors are configured with non-zero latencies, or the AP is suspended, and no wakeup events occur.

The size of the FIFO can be retrieved by the host by reading this same parameter; it is returned in bytes 2 and 3 for the Wakeup FIFO and bytes 6 and 7 for the Non-Wakeup FIFO. This size is determined at compile time of the RAM patch; additions of customer code or additional features or bug fixes will reduce the amount of RAM available for the FIFOs.

Table 16:FIFO Control

Parameter Byte	FIFO	Field Name	Description	Direction
0	Wakeup	Watermark LSB	Number of bytes in FIFO before interrupt is asserted	Read/write
1		Watermark MSB		
2		FIFO Size LSB	Size of FIFO in bytes	
3		FIFO Size MSB		
4	Non-Wakeup	Watermark LSB	Number of bytes in FIFO before interrupt is asserted	Read/write
5		Watermark MSB		
6		FIFO Size LSB	Size of FIFO in bytes	
7		FIFO Size MSB		

## Sensor Status Banks

Each byte in a 16 byte Sensor Status Bank corresponds to a specific sensor type, starting with 1 (since sensor type 0 is reserved).

Table 17:Sensor Status Bank

Sensor Status Bank	Saved Parameter Byte	Sensor Type
0	0	Sensor Type 1
0	...	
0	15	Sensor Type 16
1	0	Sensor Type 17
1	...	
1	15	Sensor Type 32
2	0	Sensor Type 33
2	...	
2	15	Sensor Type 48
3	0	Sensor Type 49
3	...	
3	15	Sensor Type 64

Each of these bytes contains the Sensor Status Bits for a given sensor. These reflect various pieces of information about a sensor that used to be scattered among a number of different registers.

Table 18:Sensor Status Bits

Bit	Field Name	Description
0	Data Available	One or more samples in output buffer
1	I2C NACK	Sensor did not acknowledge transfer
2	Device ID Error	WHO_AM_I register mismatch
3	Transient Error	e.g., magnetic transient
4	Data Lost	FIFO overflow
5-7	Sensor Power Mode	Shutdown, standby, low power active, high power active

The Sensor Power Mode values in bits 5-7 are:

- 0: Sensor Not Present
- 1: Power Down
- 2: Suspend
- 3: Self-Test
- 4: Interrupt Motion
- 5: One Shot
- 6: Low Power Active
- 7: Active

## Timestamps

In order to provide a mechanism for the host to translate sensor data timestamps to host-relative timestamps, this parameter may be read to determine the time at which the last host-interrupt was asserted, as well as the current system time.

**NOTE:** the Host IRQ Timestamp can be read more efficiently from the Host IRQ Timestamp registers 0x6C-0x6F.

Table 19:Timestamp Data

Parameter Byte	Field Name	Description	Direction
0	Host IRQ Timestamp LSB	Time (in units of 1/32000 seconds) that the last host interrupt was triggered	Read only
1	Host IRQ Timestamp B2		
2	Host IRQ Timestamp B3		
3	Host IRQ Timestamp MSB		
4	Current Timestamp LSB	Time (in units of 1/32000 seconds) for current system time	
5	Current Timestamp B2		
6	Current Timestamp B3		
7	Current Timestamp MSB		

## Physical Sensor Status

This parameter is provided for debugging. The host can read the current underlying physical sensor settings such as sample rate, dynamic range, interrupt enable, and power mode.

Table 20: Physical Sensor Status

Parameter Byte	Field Name	Type	Description
0	Accel Sample Rate	Unsigned 16 bit	Actual sample rate in Hz
1			
2	Accel Dynamic Range	Unsigned 16 bit	Actual dynamic range in gs
3			
4	Accel Flags	Unsigned 8 bit	bit 0: interrupt enable bits 5-7: Sensor Power Mode (same as Sensor Status Bits 5-7)
5	Gyro Sample Rate	Unsigned 16 bit	Actual sample rate in Hz
6			
7	Gyro Dynamic Range	Unsigned 16 bit	Actual dynamic range in gs
8			
9	Gyro Flags	Unsigned 8 bit	bit 0: interrupt enable bits 5-7: Sensor Power Mode (same as Sensor Status Bits 5-7)
10	Mag Sample Rate	Unsigned 16 bit	Actual sample rate in Hz
11			
12	Mag Dynamic Range	Unsigned 16 bit	Actual dynamic range in gs
13			
14	Mag Flags	Unsigned 8 bit	bit 0: interrupt enable bits 5-7: Sensor Power Mode (same as Sensor Status Bits 5-7)

## Physical Sensors Present

This parameter contains a 64 bit bitmap, where a set bit indicates the corresponding physical sensor is present in the system.

For example, if a physical accelerometer, magnetometer, and humidity sensor were the only physical sensors present, the bit map would have bits set for sensor ID 1 (accelerometer), 2 (magnetometer) and 12 (humidity). In this example, the bit map would be:

```

Byte 0: 0000 0110      (binary; left most bit is bit 7, right most is bit 0)
Byte 1: 0001 0000      (left most is bit 15; right most is bit 8)
Byte 2: 0000 0000
Byte 3: 0000 0000
Byte 4: 0000 0000
Byte 5: 0000 0000
Byte 6: 0000 0000
Byte 7: 0000 0000
    
```

## Physical Sensor Information

This structure is returned for any parameters 33-96 when the corresponding physical sensor is present. If not present, this structure returns all 0s.

This is an enhanced version of the earlier Physical Sensor Status structure. This new structure also provides access to the orientation matrix, for those sensors that include them, such as 3 axis accelerometers, magnetometers, and gyroscopes.

Table 21: Physical Sensor Information

Parameter Byte	Field Name	Type	Description
0	Sensor Type	Unsigned 8 bit	Same as parameter number - 32
1	Driver ID	Unsigned 8 bit	Unique per driver / vendor / part number
2	Driver Version	Unsigned 8 bit	Denotes notable change in behavior
3	Current	Unsigned 8 bit	0.1 mA
4-5	Current Range	Unsigned 16 bit	Current dynamic range of sensor in SI units
6	Flags	Unsigned 8 bit	Bit 0: IRQ enabled Bits 1-4: reserved Bits 5-7: power mode (see 5.1.3)
7	Reserved		
8-9	Current Rate	Unsigned 16 bit	Current Sample Rate in Hz
10	Number of Axes	Unsigned 8 bit	Number of Axes (e.g., X/Y/Z = 3)
11-15	Orientation Matrix	4 bits per element	See below

The matrix is used to align the orientation of physical sensor axes to match the required ENU (east north up) orientation required by Android. The calculation is performed as:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = |X_s \ Y_s \ Z_s| \cdot \begin{bmatrix} C_0 C_1 C_2 \\ C_3 C_4 C_5 \\ C_6 C_7 C_8 \end{bmatrix}$$

The calibration matrix is output in the same order as the elements are listed in the board .cfg file used to generate a firmware image (.fw file). Each matrix element is stored in successive nibbles.

For example, if the board .cfg file contains:

```
#DriverID,Addr,GPIO,C0,C1,C2,C3,C4,C5,C6,C7,C8,Off0,Off1,Off2,Range
a9, 24, 3, 1, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0, 0, 0
```

Then bytes 11-15 of the Physical Sensor Information structure would be:

Byte 11: 01 (hexadecimal)  
 Byte 12: 00  
 Byte 13: 0F  
 Byte 14: 00  
 Byte 15: 0F

Bytes 11-15 would show the same result if the stuffelf utility were used to generate the .fw file by using the command line: `stuffelf outerloop.elf -a -d24 -p3 -c1,0,0,0,-1,0,0,0,-1`

There are several possibilities to adjust the orientation matrix according to the needs of the application, all covered within a separate application note. For details please refer to section 14.2.

### 11.2 Parameter Page 3: Sensors

The table below is split into two logical areas: Access to the Sensor Information structure, which is read only, and access to the Sensor Configuration structure which is readable and writeable. The results of a read access are provided in the Parameter\_Read\_Buffer, while the input for a write access has to be provided in the Parameter\_Write\_Buffer.

The table describes, how to address the individual Information structures, while the next 2 following sections provide detailed information on these structures.

In the first area, Config Numbers 1-31, is for reading the Sensor Information structure of a specific non-wakeup sensor. Config Numbers 33-63, is for reading the Sensor Information structure of a specific wakeup sensor.

In the second area, Config numbers 65-95, is for writing the Sensor Configuration structure of a specific non-wakeup sensor, or for reading the actual sample rates, latencies, dynamic ranges, and sensitivities. Config numbers 97-127, is for writing the Sensor Configuration structure of a specific wakeup sensor or for reading the actual settings.

**NOTE:** The index can be calculated as: Sensor ID + (32 if wakeup) + (64 if Sensor Configuration). In other words, the Config Number bits 0-4 specify Sensor ID; bit 5 specifies wakeup if 1, non-wakeup if 0; and bit 6 specifies Sensor Information if 0, Sensor Configuration if 1.

Table 22: Parameter Page 3 - Sensors

Config Number	Config Name	Value in Parameter_Read_Buffer	Value in Parameter_Write_Buffer
<b>Non-Wakeup Sensor Information</b>			
1	Accelerometer	Sensor Information Structure	Not used
2	Geomagnetic Field	Sensor Information	Not used
3	Orientation	Sensor Information	Not used
4	Gyroscope	Sensor Information	Not used
5	Light	Sensor Information	Not used
6	Pressure	Sensor Information	Not used
7	Temperature	Sensor Information	Not used
8	Proximity	Sensor Information	Not used

9	Gravity	Sensor Information	Not used
10	Linear Acceleration	Sensor Information	Not used
11	Rotation Vector	Sensor Information	Not used
12	Humidity	Sensor Information	Not used
13	Ambient Temperature	Sensor Information	Not used
14	Magnetic Field Uncalibrated	Sensor Information	Not used
15	Game Rotation Vector	Sensor Information	Not used
16	Gyroscope Uncalibrated	Sensor Information	Not used
17	Significant Motion	Sensor Information	Not used
18	Step Detector	Sensor Information	Not used
19	Step Counter	Sensor Information	Not used
20	Geomagnetic Rotation Vector	Sensor Information	Not used
21	Heart Rate	Sensor Information	Not used
22	Tilt Detector	Sensor Information	Not used
23	Wake Gesture	Sensor Information	Not used
24	Glance Gesture	Sensor Information	Not used
25	Pick Up Gesture	Sensor Information	Not used
26-30	Reserved		
31	Activity Recognition	Sensor Information	Not used
<b>Wakeup Sensor Information</b>			
32	Reserved		
33	Accelerometer	Sensor Information	Not used
34	Geomagnetic Field	Sensor Information	Not used
35	Orientation	Sensor Information	Not used
36	Gyroscope	Sensor Information	Not used
37	Light	Sensor Information	Not used
38	Pressure	Sensor Information	Not used
39	Temperature	Sensor Information	Not used
40	Proximity	Sensor Information	Not used
41	Gravity	Sensor Information	Not used
42	Linear Acceleration	Sensor Information	Not used
43	Rotation Vector	Sensor Information	Not used
44	Humidity	Sensor Information	Not used
45	Ambient Temperature	Sensor Information	Not used
46	Magnetic Field Uncalibrated	Sensor Information	Not used
47	Game Rotation Vector	Sensor Information	Not used
48	Gyroscope Uncalibrated	Sensor Information	Not used
49	Significant Motion	Sensor Information	Not used
50	Step Detector	Sensor Information	Not used
51	Step Counter	Sensor Information	Not used
52	Geomagnetic Rotation Vector	Sensor Information	Not used
53	Heart Rate	Sensor Information	Not used
54	Tilt Detector	Sensor Information	Not used
55	Wake Gesture	Sensor Information	Not used
56	Glance Gesture	Sensor Information	Not used
57	Pick Up Gesture	Sensor Information	Not used
58-62	Reserved		
63	Activity	Sensor Information	Not used
<b>Non-Wakeup Sensor Configuration</b>			
64	Reserved		
65	Accelerometer	Actual Configuration	New Configuration
66	Geomagnetic Field	Actual Configuration	New Configuration
67	Orientation	Actual Configuration	New Configuration
68	Gyroscope	Actual Configuration	New Configuration
69	Light	Actual Configuration	New Configuration
70	Pressure	Actual Configuration	New Configuration

71	Temperature	Actual Configuration	New Configuration
72	Proximity	Actual Configuration	New Configuration
73	Gravity	Actual Configuration	New Configuration
74	Linear Acceleration	Actual Configuration	New Configuration
75	Rotation Vector	Actual Configuration	New Configuration
76	Humidity	Actual Configuration	New Configuration
77	Ambient Temperature	Actual Configuration	New Configuration
78	Magnetic Field Uncalibrated	Actual Configuration	New Configuration
79	Game Rotation Vector	Actual Configuration	New Configuration
80	Gyroscope Uncalibrated	Actual Configuration	New Configuration
81	Significant Motion	Actual Configuration	New Configuration
82	Step Detector	Actual Configuration	New Configuration
83	Step Counter	Actual Configuration	New Configuration
84	Geomagnetic Rotation Vector	Actual Configuration	New Configuration
85	Heart Rate	Actual Configuration	New Configuration
86	Tilt Detector	Actual Configuration	New Configuration
87	Wake Gesture	Actual Configuration	New Configuration
88	Glance Gesture	Actual Configuration	New Configuration
89	Pick Up Gesture	Actual Configuration	New Configuration
90-94	Reserved		
95	Activity	Actual Configuration	New Configuration
<b>Wakeup Sensor Configuration</b>			
96	Reserved		
97	Accelerometer	Actual Configuration	New Configuration
98	Geomagnetic Field	Actual Configuration	New Configuration
99	Orientation	Actual Configuration	New Configuration
100	Gyroscope	Actual Configuration	New Configuration
101	Light	Actual Configuration	New Configuration
102	Pressure	Actual Configuration	New Configuration
103	Temperature	Actual Configuration	New Configuration
104	Proximity	Actual Configuration	New Configuration
105	Gravity	Actual Configuration	New Configuration
106	Linear Acceleration	Actual Configuration	New Configuration
107	Rotation Vector	Actual Configuration	New Configuration
108	Humidity	Actual Configuration	New Configuration
109	Ambient Temperature	Actual Configuration	New Configuration
110	Magnetic Field Uncalibrated	Actual Configuration	New Configuration
111	Game Rotation Vector	Actual Configuration	New Configuration
112	Gyroscope Uncalibrated	Actual Configuration	New Configuration
113	Significant Motion	Actual Configuration	New Configuration
114	Step Detector	Actual Configuration	New Configuration
115	Step Counter	Actual Configuration	New Configuration
116	Geomagnetic Rotation Vector	Actual Configuration	New Configuration
117	Heart Rate	Actual Configuration	New Configuration
118	Tilt Detector	Actual Configuration	New Configuration
119	Wake Gesture	Actual Configuration	New Configuration
120	Glance Gesture	Actual Configuration	New Configuration
121	Pick Up Gesture	Actual Configuration	New Configuration
122-126	Reserved		
127	Activity	Actual Configuration	New Configuration



### 11.3 Sensor Information Structure

This structure reports everything that Android needs to know about a sensor type. If the requested sensor is not supported by the current firmware image, then all fields must be reported as zero.

For physical sensors, the Max Range field will be set to the maximum possible range that the sensor can attain if set to its highest range setting. This is a constant value that does not change based on the current Dynamic Range setting. It is stored in Android-appropriate units (m/s<sup>2</sup>, radians/s,  $\mu$ T). As this is a 16 bit integer field, the value is rounded up to the next nearest integer.

The Resolution field is provided so that the host can determine the “smallest difference between two values reported by this sensor.” It contains the number of bits of resolution. With that, the host can determine the floating point resolution value in SI units by dividing the Max Range or current Dynamic Range (in SI units) by  $2^{\text{Resolution}}$ .

Table 23: Sensor Information Structure

Parameter Byte	Field Name	Type	Description
0	Sensor Type	Unsigned 8 bit	Defensive programming measure – repeat the requested sensor type <b>NOTE:</b> bit 4 of Sensor Type is 1 for Non-Wakeup sensors
1	Driver ID	Unsigned 8 bit	Unique per driver / vendor / part number
2	Driver Version	Unsigned 8 bit	Denotes notable change in behavior
3	Power	Unsigned 8 bit	0.1 mA/LSB
4	Max Range	Unsigned 16 bit	Maximum range of sensor data in SI units
5			
6	Resolution	Unsigned 16 bit	Number of bits of resolution of underlying sensor
7			
8	Max Rate	Unsigned 16 bit	Hz
9			
10	FIFO Reserved	Unsigned 16 bit	FIFO size in bytes reserved for this sensor divided by data packet size in bytes; if a single shared FIFO, this can be 0
11			
12	FIFO Max	Unsigned 16 bit	Entire FIFO size in bytes divided by data packet size in bytes
13			
14	Event Size	8 bit	Number of bytes for sensor data packet (including Sensor Type)
15	Min Rate	Unsigned 8 bit	Hz

## 11.4 Sensor Configuration Structure

The following table describes the parameter set of the sensor configuration structure:

Table 24: Sensor Configuration Structure

Parameter Byte	Field Name	Type	Description
0	Sample Rate	Unsigned 16 bit	Rate in Hz; $1 \div$ Android sample period; reads back actual sensor rate
1			
2	Max Report Latency	Unsigned 16 bit	0 for non-batch mode; if nonzero, Sample Rate must also be nonzero; milliseconds; reads back actual latency
3			
4	Change Sensitivity	Unsigned 16 bit	Scaled same as sensor's data value; for future Win8/10 support
5			
6	Dynamic Range	Unsigned 16 bit	range setting for physical setting in appropriate units
7			

The meaning of each field below is slightly different depending on whether this is being written or read; see the description for details.

Changes to the Sample Rate field take effect quickly, but not immediately. If the host wishes to know when the rate change is complete, it can enable and wait for the Sample Rate Changed meta event. The actual rate selected is within the range of 90% ... 210% of the requested range.

Changes to the Max Report Latency take effect immediately. If a timer for the sensor using a different Max Report Latency is running, it will be modified. It is possible due to timing for a change to be slightly too late to effect the current timer, but will affect subsequent samples. If Max Report Latency is set to 0 when it was previously not 0, then this will be treated the same as a flush request.

The Dynamic Range field for the virtual Accelerometer, Gyroscope, and Magnetometer sensors will be able to control the actual dynamic range settings in the corresponding physical sensors. A value of 0 requests the default. The algorithm will be informed of the request to change the dynamic range, and, the corresponding scale factor for the sensor data outputs will change accordingly. The host may then read back the Sensor Configuration structure to determine the actual current dynamic range.

**NOTE:** the host should enable and watch for the Dynamic Range Changed meta event so it can apply the correct scale factor before and after the dynamic range change, if the change is made while the sensor is already enabled.

Reading back the parameter is especially important if the host sets a dynamic range for other virtual sensors that share the same underlying physical sensor. The BHA will select the largest requested dynamic range of all virtual sensors that share that physical sensor.

For instance, the virtual Accelerometer, Gravity, and Linear Acceleration sensors all share the physical accelerometer. The virtual Gyroscope and Uncalibrated Gyroscope both share the physical gyroscope. The virtual Magnetometer and Uncalibrated Magnetometer share the physical magnetometer. If the host does not specify a dynamic range for a specific virtual sensor (by setting it to 0), then only the virtual sensors with non-zero dynamic range requests from the host will be considered in selecting the actual dynamic range for the physical sensor.

For example, setting the dynamic range for the Accelerometer sensor to 156.93 m/s<sup>2</sup> while setting the dynamic range of the Linear Acceleration sensor to 78.46 m/s<sup>2</sup> and the Gravity sensor to 0 results in an actual dynamic range for all three sensors as well as the physical sensor of 156.93 m/s<sup>2</sup>, which is the same as 16 Earth g-s.

The dynamic range will determine the scale factor for the sensor data, based on the number of bits and signed-ness of the data.

The units of measurement for dynamic range here are not the same as the Android units for those sensors. We have chosen to use units that are commonly used by sensor manufacturers in their data sheets when discussing range settings.

1. Accelerometer: Earth g-s
2. Gyroscope: degrees/second
3. Magnetometer:  $\mu$ T

### 11.5 Parameter Page 15: Soft Pass-Through

This parameter page can be used during normal operation to read and write registers on devices attached to the sensor I2C bus.

Parameter 1: Soft Pass-Through register read single multi-byte transfer

Parameter 3: Soft Pass-Through register read multiple single byte transfers

Parameter 5: Soft Pass-Through register read multiple single byte transfers with 0.5ms delays

Table 25: Parameter Page 15 – Soft-Pass-Through 1

Parameter Byte	Parameter Name	Description	Direction
0	I2C Slave address	Slave address for the sensor	Read/Write
1	Start Register	The first register address to read	Read/Write
2	Read Length	Register length to read (maximum 4)	Read/Write
3	Completion Status	To judge if the register read has finished	Read
4	Reg Value Byte 1	Returned register value	Read
5	Reg Value Byte 2	Returned register value	Read
6	Reg Value Byte 3	Returned register value	Read
7	Reg Value Byte 4	Returned register value	Read

On read bytes 1 through 3 returns the last time value written for validation purpose.

Parameter 2: Soft Pass-Through register write single multi-byte transfer

Parameter 4: Soft Pass-Through register write multiple single byte transfers

Parameter 6: Soft Pass-Through register write multiple single byte transfers with 0.5ms delays

Table 26: Parameter Page 15 – Soft-Pass-Through 2

Parameter Byte	Parameter Name	Description	Direction
0	I2C Slave address	Slave address for the sensor	Read/Write
1	Start Register	The first register address to write	Read/Write
2	Write Length	Register length to write (maximum 4)	Read/Write
3	Completion Status	To judge if the register write has finished	Read
4	Reg Value Byte 1	Register value to write	Write
5	Reg Value Byte 2	Register value to write	Write
6	Reg Value Byte 3	Register value to write	Write
7	Reg Value Byte 4	Register value to write	Write

On read bytes 1 through 3 returns the last time value written for validation purpose. On write bytes 3 is ignored.

Parameters 1 and 2 perform fast single transfers of multiple bytes. Some sensor devices do not support this. Alternatively, parameters 3 and 4 simulate a multi byte transfer by doing a series of single byte transfers with an incrementing register address; this is useful for some devices that do not support multi-byte transfers. Finally, parameters 5 and 6 are similar to 3 and 4, except a delay of 0.5ms is added between each byte transfer. Some sensors require slow writes in certain modes, for example.

Completion status values:

- 0 = transfer in progress (or none ever issued yet)
- 1 = transfer successful
- 2 = I2C NACK or I2C error

## 12. Sensor Data Types and Output Format

Sensor IDs will match the Android numbering; these are currently numbered 1 through 31. Any new sensor IDs that represent unique custom sensors will be numbered starting at 254, decreasing towards the Android numbers.

If the host and BHA become out of sync, the host can regain synchronization by flushing the buffer (reading everything in it). Any data that arrives after that will start with a whole sensor data packet, with the Sensor Type as the first byte.

Table 27: Sensor IDs, Data Types and Output Format

Sensor ID Non Wakeup	Sensor ID Wakeup	Size in FIFO	Contents	Scale Factor	Sensor Value	Format
		1	Padding			Unsigned 8 bit value 0 means NOP
11 15 20	43 47 52	11	Rotation Vector Game Rotation Vector Geomagnetic Rotation Vector	214	X, Y, Z, W Quaternion Estimated Accuracy (radians)	16 bit signed fixed point  16 bit signed fixed point integer
1 2 3 4 9 10	33 34 35 36 41 42	8	Accelerometer Magnetometer Orientation* Gyroscope Gravity Linear Acceleration	dynamic dynamic 360 ° / 215 dynamic dynamic dynamic	X, Y, Z Vector Status	16 bit signed integer, scaled to current dynamic range; 8 bit unsigned integer indicating accuracy of measurement
5 8 12	37 40 44	3	Light Proximity Humidity	10000Lux / 216 100cm / 216 1%RH	Absolute Scalar	16 bit unsigned integer, scaled to maximize dynamic range
19	51	3	Step Counter	None	Absolute Scalar	16 bit unsigned integer
7 13	39 45	3	Temperature Ambient Temperature	500LSB/°C centered at 24°C	Scalar	16 bit signed integer, scaled to maximize dynamic range
6	38	4	Barometer	1/128 Pa	Absolute Scalar	24 bit unsigned integer, scaled as required
17 18 22 23 24 25	49 50 54 55 56 57	2	Significant Motion Step Detector Tilt Detector Wake Gesture Glance Gesture Pick Up Gesture	None	Event	None
14 16	46 48	14	Uncalibrated Magnetometer Uncalibrated Gyroscope	dynamic dynamic	Uncalib X, Y, Z; bias X, Y, Z Status	16 bit signed integer, scaled to current dynamic range 8 bit unsigned integer indicating accuracy of measurement (low, medium, high, unreliable)
21	53	2	Heart Rate	None	Scalar	8 bit beats per minute

31	63	3	Activity	None	Scalar	Bits 0-7 specify the activity change off, bits 8-15 specify the activity change on; see below for bit definitions
245	n/a	14	Debug	None	Structure	Byte 1: Bit 7: reserved Bit 6: binary fmt (string if 0, binary if 1) Bits 5-0: valid bytes  Bytes 2-13: debug data
249 250 251	n/a	17	BSX_C BSX_B BSX_A	None	Accel, Gyro, Mag: X, Y, Z, Timestamp	Accel, Gyro, Mag: 32 bit signed integer (C=raw gyro, B=raw mag, A=raw accel); 32 bit timestamp of current sensor sample
252	246	3	Timestamp LSW	1/32000 seconds	Time	16 bit unsigned integer; counts at a 32KHz rate; applies to all following sensor samples; wraps every 2 seconds
253	247	3	Timestamp MSW (Overflow)	65536/32000 seconds	Time	16 bit unsigned integer; counts at Timestamp overflow rate; wraps every 36 hours
254	248	4	Meta Events	None	Event	8 bit unsigned integer event number 8 bit unsigned integer sensor type 8 bit unsigned integer event-specific value

**\*NOTE:** X = azimuth = 0° to 360° unsigned, Y = pitch = +/- 180° signed, Z = roll = +/- 90° signed

## 12.1 Quaternion+

For the three rotation vectors (Rotation Vector, Game Rotation Vector, Geomagnetic Rotation Vector), the following format is used.

The host can convert the X, Y, Z, W, and Estimated Accuracy fields to floating point numbers like this:

```

unsigned char event[11];
... read in the event to the array above...
float x = ((float)(event[1] + ((unsigned int)event[2]) << 8)) /
16384;
... convert other fields...
  
```

The Estimated Accuracy in Radians is reported as 0 for the Game Rotation Vector.

Table 28: Quaternion+ Data Format

Byte Number	Contents	Format
0	Sensor ID (Rotation Vector, etc.)	
1	X LSB	Signed 16 bit fixed point integer
2	X MSB	
3	Y LSB	Signed 16 bit fixed point integer
4	Y MSB	
5	Z LSB	Signed 16 bit fixed point integer
6	Z MSB	
7	W LSB	Signed 16 bit fixed point integer
8	W MSB	
9	ESTIMATED ACCURACY, RADIANS, LSB	Signed 16 bit fixed point integer
10	ESTIMATED ACCURACY, MSB	

## 12.2 Vector+

For the many 3 axis sensors (Accelerometer, Magnetometer, Orientation, Gyroscope, Gravity, Linear Acceleration), the following layout is used.

Table 29: Vector+ Data Format

Byte Number	Contents	Format
0	Sensor ID (Accelerometer, etc.)	
1	X LSB	Signed 16 bit integer
2	X MSB	
3	Y LSB	Signed 16 bit integer
4	Y MSB	
5	Z LSB	Signed 16 bit integer
6	Z MSB	
7	STATUS	Accuracy of measurement

Table 30: Vector+ Status description

Status Value	Meaning
<b>0</b>	Unreliable
<b>1</b>	Accuracy Low
<b>2</b>	Accuracy Medium
<b>3</b>	Accuracy High

### 12.3 Vector\_Uncalibrated

For the two uncalibrated 3 axis sensors (Uncalibrated Magnetometer, Uncalibrated Gyroscope), the following layout is used.

Table 31: Vector\_Uncalibrated Data Format

Byte Number	Contents	Format
0	Sensor ID (Uncalibrated Magnetometer, etc.)	
1	X LSB	Signed 16 bit integer
2	X MSB	
3	Y LSB	Signed 16 bit integer
4	Y MSB	
5	Z LSB	Signed 16 bit integer
6	Z MSB	
7	X BIAS LSB	Signed 16 bit integer
8	X BIAS MSB	
9	Y BIAS LSB	Signed 16 bit integer
10	Y BIAS MSB	
11	Z BIAS LSB	Signed 16 bit integer
12	Z BIAS MSB	
13	STATUS	Accuracy of measurement

Table 32: Vector\_Uncalibrated Status description

Status Value	Meaning
<b>0</b>	Unreliable
<b>1</b>	Accuracy Low
<b>2</b>	Accuracy Medium
<b>3</b>	Accuracy High

### 12.4 Scalar Data

All of the scalar sensors (Light, Proximity, Humidity, Step Counter, Heart Rate, Temperature, Ambient Temperature, and Barometer) share a similar format. While there are differences in signed vs. unsigned and scale, the layout is similar for all (as listed in section 9.6) of them.

Table 33: Scalar\_Data Data Format

Byte Number	Contents	Format
0	Sensor ID (Light, etc.)	
1	LSB	Signed or Unsigned 16 bit integer (LSB only for 8bit)
2	MSB	

Byte Number	Contents	Format
0	Barometer	
1	LSB	Unsigned 24 bit integer
2	MIB	
3	MSB	



## 12.5 Sensor Event Data (Parameterless Sensors)

Some sensors (Significant Motion, Step Detector, Tilt Detector, Padding) generate no actual data, so the FIFO will simply contain the sensor ID itself.

Table 34: Sensor\_Event Data Format

Byte Number	Contents	Format
0	Sensor ID (Significant Motion, etc.)	

## 12.6 Activity Recognition Data (Sensor\_Activity\_Rec\_Data)

The activity sensor outputs 16 bits of data whenever there is a change detected in activity. Bits are provided to indicate both the onset of an activity and the end of it.

Table 35: Activity Recognition Data Format

Bit	Contents
0	Still activity ended
1	Walking activity ended
2	Running activity ended
3	On Bicycle activity ended
4	In Vehicle activity ended
5	Tilting activity ended
6	Reserved
7	Reserved
8	Still activity started
9	Walking activity started
10	Running activity started
11	On Bicycle activity started
12	In Vehicle activity started
13	Tilting activity started
14	Reserved
15	Reserved

## 12.7 Debug

Customers using the SDK can use `printf()` to send ASCII strings to the host in the non-wakeup FIFO, or `fwrite()` to send custom binary data. The data will be sent with Sensor ID = Debug.

The lower 6 bits include the number of valid bytes in the packet with the range from 0 up to 12. For example, if a 20 character long string is printed out: "ABCDEFGHIJKLMNQRST", the FIFO will contain:

Debug (=245), 0x0C, "ABCDEFGHIJKLMNQRST"

Debug, 0x08, "MNQRST"

Table 36: Debug

Byte	Bit	Contents
0	0-7	Sensor ID (Debug)
1	0-5	Amount of payload bytes (0-12)
	6	Payload data format (0: ASCII, 1: Binary)
	7	Reserved
2-13	0-7	Debug Payload (Bytes)

## 12.8 Sensor Data scaling

The table of sensor types and their data formats above indicates a fixed scale factor for each sensor's data for many sensors. The selection of this scale factor is meant to ensure the full dynamic range of a sensor can fit within the number of bits provided. The exceptions are the accelerometer-, gyroscope-, and magnetometer-derived sensors. Those sensors' scale factors are set dynamically based on the largest dynamic range setting for any related virtual sensors (see section 11.4).

### Default Scale Factors

The default scale factor specified for the intrinsic Accel, Gravity, and Linear Acceleration allows for a 4g range (39.24 m/s<sup>2</sup>).

The default scale factor for an externally attached Gyro allows for a 2000° / s range (34.9 radians / s).

The default scale factor for the magnetometer is based on a dynamic range of 1000μT, despite the fact that many sensors can read larger values than this. The AK8963, for example, can measure fields up to 4912μT. In return, there will be 2 more bits of resolution for normal readings, keeping in mind that the Earth's magnetic field ranges from 25μT to 65μT. Lollipop does not require any particular range for the magnetometer beyond this.

These sensors all use a scale factor that adjusts automatically to fit the actual dynamic range of each sensor, if changed from the defaults. The host needs to query the actual dynamic range and then perform a calculation to determine the scale factor as dynamic range / 2<sup>(number of bits)</sup>, where the number of bits is 15 for signed 16 bit integers and so on.

**NOTE:** The accelerometer and gyroscope dynamic range settings, as discussed in section 11.4, need to be converted to m/s<sup>2</sup> and radians / s, before being used to scale the output data as described below. For example, the Accelerometer produces 16 bit signed data for the X, Y, and Z axes. The scale factor for a dynamic range of 156.93 m/s<sup>2</sup> would be:

Accel Scale Factor = Dynamic Range / Maximum Positive Value = 156.93 / 32767 = 4.789e<sup>-3</sup> m/s<sup>2</sup>.

## 12.9 Meta Events

These indicate asynchronous, low periodicity events. These can be individually enabled or disabled in the Meta Event Control Parameter in the System Parameter Page. They can also be configured to enable or disable a host interrupt when they occur; this would occur even if there were no pending samples. The Initialized Meta Event will be the first event in the FIFO (following the current timestamp) after initialization. Once the host receives this, it is safe to interact with the parameter I/O system to configure the chip. See section 9.9 for more details.

Table 37: Meta Events

Meta Event Type	Name	Byte 1	Byte 2	Wake FIFO Default Enable State	Wake FIFO Default Int Enable State	Non Wake FIFO Default Enable State	Non Wake FIFO Default Int Enable State
0	Not used						
1	Flush Complete	Sensor Type from FIFO_FLUSH register	Not used	Enabled	Disabled	Enabled	Disabled
2	Sample Rate Changed	Sensor Type	Not used	Enabled	Disabled	Enabled	Disabled
3	Power Mode Changed	Sensor Type	Power Mode	Disabled	Disabled	Disabled	Disabled
4	Error	Error Register	Debug State	Enabled	Enabled	Enabled	Enabled
5-10	Reserved						
11	Sensor Error	Sensor Type	Sensor Status Bits	Enabled	Enabled	Enabled	Enabled
12	FIFO Overflow	Loss Count LSB	Loss Count MSB	Enabled	Disabled	Disabled	Disabled
13	Dynamic Range Changed	Sensor Type	Not used	Enabled	Disabled	Enabled	Disabled
14	FIFO Watermark	Bytes Remaining LSB	Bytes Remaining MSB	Disabled	Disabled	Disabled	Disabled
15	Self-Test Results	Sensor Type	Test Result	Enabled	Disabled	Enabled	Enabled
16	Initialized	RAM Ver LSB	RAM Ver MSB	Enabled	Enabled	Enabled	Enabled

### Flush Complete

This meta event is the only official Android meta event. It will be inserted in the FIFO after a Flush FIFO request, whether there is any data in the FIFO or not. Flushing in Android means “transfer to host”, not “discard.”

### Sample Rate Changed

This meta event occurs when a given sensor's sample rate has been set for the first time, and/or when a requested change to the rate actually occurs. Byte 1 indicates the sensor type whose rate changed.

### Power Mode Changed

This meta event indicates when a given sensor powers up or down; the sensor type is passed in byte 1.

### Error

The error meta event will only occur when unexpected internal firmware errors occur. It is important for the host to log such events, including byte 1 (error register) and byte 2 (debug state), to help with troubleshooting. The only proper recovery from this is to reset the BHA250, reload the RAM patch, and reinitialize the sensors to the desired rates, latencies, and so on.

### Sensor Error

This meta event occurs when there is either a sensor mismatch between the RAM patch loaded and the hardware available, or, when a sensor fails to respond when expected. Byte 1 specifies the sensor type, and byte 2 specifies the sensor status bits for that sensor. This will indicate whether the error was due to an I2C NACK or Device ID mismatch. The Device ID mismatch error should never occur in a properly configured system in production. The I2C NACK error indicates a hardware failure.

### FIFO Overflow

This meta event indicates when data loss has occurred due to the host being unable to read out FIFO data quickly enough. This may be intentional, such as when the AP is suspended, or it may be due to having too many sensors on at high sample rates with a slow host I2C rate or slow driver implementation. It reports in bytes 1 and 2 a saturating count of lost bytes. Following this meta event, the BHA will insert an accurate Timestamp MSW and Timestamp LSW event, to ensure the host will be able to report accurate timestamps after the area of data loss.

The BHA, when it detects a FIFO overflow, automatically discards approximately 200 bytes of FIFO data in order to make room for more data, make room for the FIFO Overflow and Timestamp events, and ensure that at least some new data will appear in the FIFO between FIFO Overflow events, rather than become saturated with such events in worst case conditions.

### Dynamic Range Changed

This event will be placed in the FIFO as soon as a requested change in dynamic range has occurred. The host may wish to wait for this event before changing the scale factor, in the event that a sensor whose dynamic range was changed was already on. Otherwise, the host could apply the wrong scale factor on some samples, and report invalid data as a result.

### FIFO Watermark

This event occurs within a few dozen bytes of the specified watermark level.

#### 12.9.1 Self-Test Results

Byte 1 indicates which sensor is reporting self-test results.

The Sensor Type field values for each physical sensor type that can perform self-test are:

- Accelerometer = ID 1
- Uncalibrated Gyroscope = ID 16
- Uncalibrated Magnetometer = ID 14

Byte 2 indicates the test status:

- 0: Test Passed
- 1: X Axis Failed
- 2: Y Axis Failed
- 3: X & Y Axes Failed
- 4: Z Axis Failed
- 5: X & Z Axes Failed
- 6: Y & Z Axes Failed
- 7: All Axes Failed or Single Test Failed (if testing of each axis cannot be done)

### 12.9.2 Initialized

This is the first meta event reported after reset. The RAM version is reported in bytes 1 and 2, for convenience.

### Timestamp LSW

The BHA outputs the Timestamp LSW event before every sensor sample whose timestamp is different from the previous Timestamp LSW placed in the FIFO. This event contains the least significant 16 bits of a sensor sample's 32 bit timestamp. If a sensor sample is about to be inserted in the FIFO which shares the same timestamp, this event is not inserted, in order to conserve FIFO space.

### Timestamp MSW

This event will be output when the most significant 16 bits of a sensor sample timestamp differs from the previous Timestamp MSW event placed in the FIFO. It, and the Timestamp LSW event, will be placed in the FIFO after a FIFO overflow as well.

## 13. Reading FIFO Data

As mentioned previously, registers 0x00 to 0x31 form the data transfer area. Unlike the other registers, these have strict protocols for host access.

When the host receives the host interrupt from the BHA, it should first read the Bytes Remaining registers to determine the number of bytes currently in the FIFO. It should then read all those bytes in one or more consecutive I2C read operations.

This area is double buffered. FIFO data is placed in the two buffers by the FUSER core, as needed, to transfer the number of bytes in the Bytes Remaining registers. The host only needs to start reading at register 0, and continue reading until the entire transfer is complete. The register address being read auto-resets when reading past 0x31, which also causes the double buffers to swap; when this happens, the FUSER core fills the just emptied buffer with the next set of FIFO data.

In this manner, the host will be able to set up a single I2C transfer for the entire Bytes Remaining value.

The transfer should be started as:

```
<START>  
<BHA Address + W>  
<0x00 (register address 0x00)>  
<REPEATED_START>  
<BHA Address + R>  
<read data...>  
<STOP>
```

Due to the fact that the double buffers are 50 bytes in size, it is recommended that if the host needs to break up a long FIFO read into multiple smaller individual I2C reads, it do so in multiples of 50 bytes, and start reading each one starting at register address 0.

The current hardware implementation cannot tell the FUSER core if the host has read less than 50 bytes in any given buffer. As a result, it is up to the host to: read in multiples of 50 bytes; read the entire buffer in one transfer; or follow the Pause and Resume Mechanism below to begin each transfer at the proper I2C address.

**NOTE:** reading more data than the Bytes Remaining registers indicate can lead to data corruption; always read the amount specified and no more.

### 13.1 Host Interrupt Behavior

The host interrupt asserts high on reset or power up, and is cleared by loading a RAM patch or reading the Host Status register (0x35).

During normal operation, it will also assert high when data is available in the FIFO and it is time to notify the host.

A host interrupt is generated when:

1. An enabled sensor has a zero max report latency (timeout) and has generated a sample
2. A sensor has a non-zero max report latency, it has a sample in the FIFO, and it has timed out before any other sensor with a shorter latency or zero latency generates a sample
3. Same as 2, but the FIFO Watermark is non-zero, and it has been exceeded before the latencies timed out
4. A meta event has occurred which has its interrupt enable set (by default, only internal firmware errors or sensor hardware errors can generate interrupts)

5. The AP is in suspend mode, one or more wakeup sensors are enabled, and one or more wakeup events have occurred (no other event except unexpected reset will generate an interrupt in this state), and the wakeup FIFO interrupt disable bit is clear in the Host Interface Control register
6. The AP is in suspend mode, the wakeup FIFO watermark is non-zero, and all enabled sensors have non-zero latency; when the watermark is reached, the AP will be woken up

The host interrupt will remain asserted until the host has emptied the FIFO or has aborted the transfer with the Abort Transfer bit in the Host Interface Control register (0x55). It may then reassert immediately depending on the notification criteria above.

The host interrupt will always go low for a minimum of a few  $\mu$ s between the time the host receives it and empties the FIFO and any subsequent transfer. This is to ensure that either level or edge-triggered interrupts can be used in the AP.

The time at which the rising edge of the interrupt occurred will be accessible as the Host IRQ Timestamp parameter of the System Parameter Page. This timestamp is a 32 bit counter, incremented at 32 KHz, and synchronized with the real time counter used to timestamp sensor data.

### 13.2 Pause and Resume Mechanism

The transfer can be paused by ending the I2C transfer with a <STOP> condition, and resumed later by issuing:

```
<START>  
<BHA Address + W>  
<restart_register_address>  
<REPEATED_START>  
<BHA Address + R>  
<read data...>  
<STOP>
```

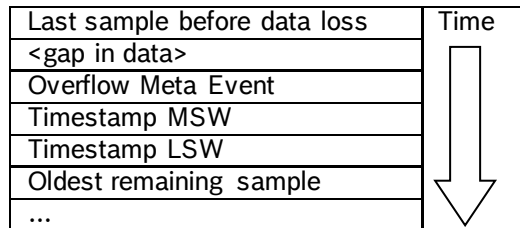
The restart\_register\_address must be calculated by the host as:  
 $\text{restart\_register\_address} = \text{bytes\_transferred\_so\_far} \text{ MOD } 50$

The Update Transfer Count bit in the Host Interface Control register can be used to get an updated value in registers 0x38/0x39 (Bytes Remaining). The BHA cannot detect the pause in transfer so it cannot automatically update the transfer count itself. However, this updated count will not affect the length of the current transfer; it simply advises the host of how much will eventually be transferred between the current and next host interrupt. Again, reading more data from the FIFO than the original Bytes Remaining value is not recommended.

### 13.3 FIFO Overflow Handling

In the event that the FIFO overflows, due too excessively high sample rates and/or slow reading by the host (or even the host being asleep), we will discard the oldest data first.

Once the host begins reading again, we will insert an Overflow Meta Event (if enabled) into the buffer at the point of data loss, followed immediately by the Timestamp MSW and Timestamp LSW values for the continuation point, followed finally by the oldest remaining data. In this way, the host can know that data was lost and know an accurate timestamp for the data that continues after the gap – after the lost data.



See the FIFO examples in section 13.9 for more details.

### 13.4 Host Suspend Procedure

When the application processor goes into suspend mode, it should tell the BHA by first writing a 1 to the AP Suspend bit in the Host Interface Control register, so that only wake-up sensors issue a host interrupt. It is recommended that the host precede this with a FIFO Flush and complete emptying of the FIFO, so that the host is not immediately woken by mistake.

### 13.5 Host Wakeup Procedure

When the AP wakes, it should notify the BHA by clearing the AP Suspend bit in the Host Interface Control register, so that all enabled sensors can (if so configured) issue a host interrupt as needed. The BHA will detect this action and automatically update the Bytes Remaining registers if there is a pending host interrupt.

If the AP did not notify the BHA of entering and leaving suspend mode, but instead masked the host interrupt line, the AP may find the host-interrupt line already asserted when it comes out of suspend. The Bytes Remaining registers will contain an old value, which, while valid, will likely be smaller than the current number of bytes in the FIFO. Once the host has transferred this smaller amount of bytes, the next host transfer will cover the remaining amount to be transferred.

### 13.6 Non-Compliant Hosts

Hosts must read the Bytes Remaining registers prior to starting the FIFO emptying protocol in order to read buffered data. Hosts must read at least that number of bytes of data; if it reads more, pad bytes (0s) will be returned. Hosts which read less than this will not receive a new host interrupt until they finish reading the remainder of the buffer, or until they request a transfer abort (which will cause loss of data). Hosts could read in fixed size blocks until, after parsing each block, it detects one or more pad bytes in the stream.

**NOTE:** Any data read after the first pad byte should be discarded, as it may be a repetition of previous data. The host should stop reading as soon as this first pad byte is detected.



### 13.7 Recovery from Loss of Sync

If the host cannot make sense of the data it is reading from the FIFO, it may have lost sync. While this should never occur, if it does, the host should abort the transfer by writing a 1 to the Abort Transfer bit in the Host Interface Control register. The next transfer will start with a whole (not partial) block of data. Alternatively, the host could simply finish the current transfer, discard what it is receiving, and then wait for the next transfer as signaled by a host interrupt. By definition, the start of a FIFO read will always begin at the start of a valid sensor event.

### 13.8 Padding Data

FIFO reads beyond the value previous read from the Bytes Remaining registers will result in the host reading 0s, which indicate end of data. A 0 sensor ID is defined to be a single byte NOP sensor event.

**NOTE:** due to hardware limitations, if the amount of FIFO data to transfer modulus 50 is greater than 0 and less than 3, the BHA will add enough padding bytes to bring up the size of the final transfer to 4 bytes minimum.

### 13.9 Aborting a Transfer

The abort transfer action can be used to recover from loss of sync or for fast emergency shutdown. As mentioned previously, there will likely be approximately 100 bytes of FIFO data that are discarded when the transfer is aborted, including any partial sensor sample or event that started in the 100 byte transfer registers and was scheduled to be finished in the next buffer.

If the host wishes to respond to the next host interrupt and parse out the remaining FIFO data, it will see a valid sample at the start, but, the BHA makes no attempt to provide an accurate starting timestamp in this case. So, the host needs to read, parse, and discard events up until the first Timestamp LSW event and/or Timestamp MSW event.

If there was one or more continuous output sensors enabled, then the host need only discard up to the first Timestamp LSW event; if the timestamp value is greater than the previous value seen before the abort, the host's knowledge of the timestamps of sensor samples that follow is accurate. However, if the first Timestamp LSW value seen after the abort is less than the previous value, the MSW has incremented, and the Timestamp MSW value was lost in the 100+ bytes that were discarded.

If there were only on-change, special, or one-shot sensors enabled, the host cannot be guaranteed to know the correct 32 bit timestamps for any events it sees after the abort until the next Timestamp MSW is read and parsed. It may still want to report the events it does see, as they may be important to the user, but the host will need to estimate the timestamps.

### 13.10 FIFO Parsing Examples

#### 13.10.1 Accelerometer & Step Counter

This example assumes the FIFO watermark is disabled, the Accelerometer is configured for 50Hz and 40 millisecond latency; the only other sensor on is the Step Counter, which is set for 0 latency. This is not the first sample; the last known Timestamp MSW = 0x0010 (~32 seconds after startup). At 50Hz, the timestamp will increment approximately 640 ticks between samples ( $640/32000 = 20\text{ms} = 1/50\text{Hz}$ ).

Steps are listed in time order from the host's perspective.

1. Host interrupt received from BHA
2. Host reads registers 0x38-0x39; value read is 25
3. Host reads registers 0x00-0x18; data received:

Byte Number	Value	Meaning
1	252	Timestamp LSW Event
2	0xF8	LSB
3	0xFF	MSB; Time = $0x0010FFF8 / 32000 = 34.81575$ seconds
4	1	Accelerometer Event
5	0xFE	X LSB
6	0xFF	X MSB; X value = $0xFFFE = -2 = -0.009578 \text{ m/s}^2$
7	0x05	Y LSB
8	0x00	Z MSB; Y value = $0x0005 = 5 = 0.023945 \text{ m/s}^2$
9	0x69	Z LSB
10	0x08	Z MSB; Z value = $0x0869 = 2153 = 10.310717 \text{ m/s}^2$
11	0x02	Status = medium accuracy
12	253	Timestamp MSW Event
13	0x11	LSB
14	0x00	MSB
15	252	Timestamp LSW Event
16	0x78	LSB
17	0x02	MSB; Time = $0x00110278 / 32000 = 34.83575$ seconds
18	1	Accelerometer Event
19	0xFD	
20	0xFF	X value = $-3 = -0.014367 \text{ m/s}^2$
21	0x08	
22	0x00	Y value = $8 = 0.038312 \text{ m/s}^2$
23	0xFC	
24	0x07	Z value = $0x07FC = 2044 = 9.798 \text{ m/s}^2$
25	0x02	Status = medium accuracy

4. The host decodes this, and as indicated in the meaning column, learns that:
  - a. The FIFO contained the least significant word (LSW) of the timestamp; using that and the previous timestamp MSW gives us an actual timestamp for the next sample as 34.81575 seconds
  - b. There are two samples of the Accelerometer, which makes sense because the sample latency divided by the sample period is 2
  - c. The samples are separated by a full timestamp (both the MSW and the LSW) because the LSW overflowed between samples
  - d. Using the default scale factor of  $4.789e^{-3} \text{ m/s}^2$ , it can calculate the actual values of the Accelerometer X, Y, and Z axes
5. The host now waits for the next interrupt
6. The human was stepping during this; right after the next accelerometer sample, the Step Counter outputs a new step count
 

**NOTE:** The accelerometer data is just dummy data; real data taken when a person is walking will of course change constantly on all axes more than this data does.
7. Host interrupt received from BHA
8. Host reads registers 0x38-0x39; value read is 14
9. Host reads registers 0x00-0x0D; data received:

Byte Number	Value	Meaning
1	252	Timestamp LSW Event
2	0xF8	LSB
3	0x04	MSB; Time = $0x001104F8 / 32000 = 34.85575$ seconds
4	1	Accelerometer Event
5	0xFF	X LSB
6	0xFF	X MSB; X value = $0xFFFE = -1 = -0.004789$ m/s <sup>2</sup>
7	0x11	Y LSB
8	0x00	Z MSB; Y value = $0x0011 = 17 = 0.081413$ m/s <sup>2</sup>
9	0x82	Z LSB
10	0x07	Z MSB; Z value = $0x0782 = 1922 = 9.204458$ m/s <sup>2</sup>
11	0x02	Status = medium accuracy
12	19	Step Counter Event
13	0x01	LSB
14	0x00	MSB = 1 <sup>st</sup> step

10. The host decodes this, and learns that:
  - a. There is a new timestamp, and the sample time is now 34.85575 seconds
  - b. There is one accelerometer sample
  - c. There is a step counter sample
  - d. The step counter, since it had a latency of zero, flushed the FIFO earlier than it would have been by the accelerometer's latency of 40ms
11. Approximately 40 milliseconds later (the latency for the accelerometer), another host interrupt followed by reading of two accelerometer samples will occur (not shown)

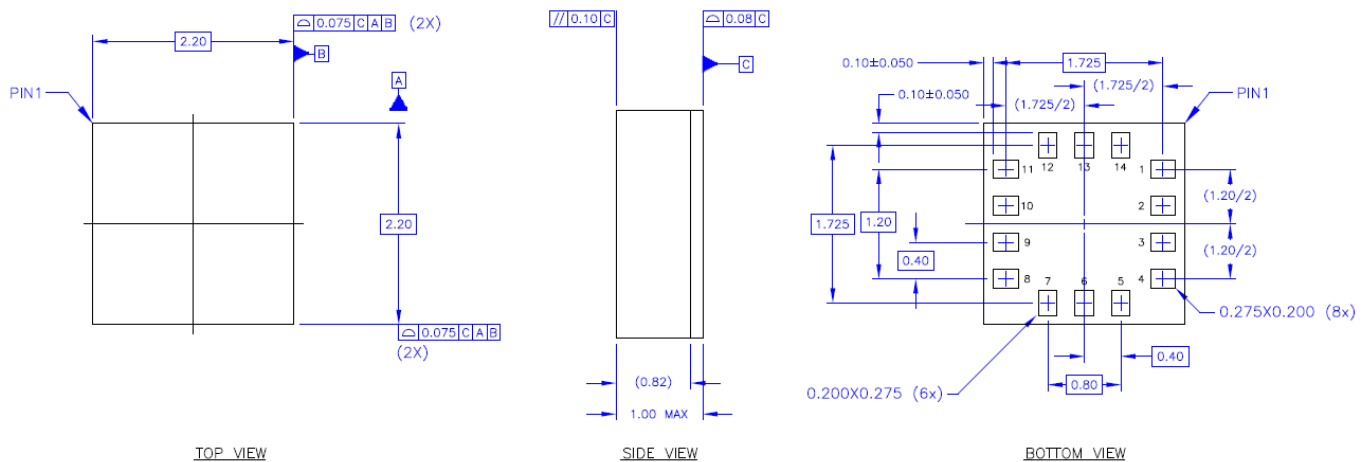
## 14. Package

### 14.1 Outline Dimensions

The sensor package is a standard LGA package; dimensions are shown in the following diagram. Units are in mm.

**NOTE:** Unless otherwise specified tolerance = decimal  $\pm 0.1\text{mm}$ .

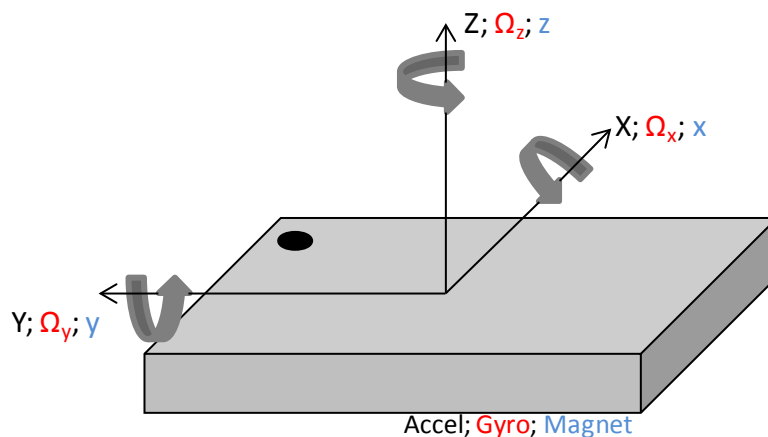
Figure 8: BHA250 Package outline in LGA14, 2.2x2.2x0.95mm<sup>3</sup>



### 14.2 Sensing Axes Orientation and Axis Remapping

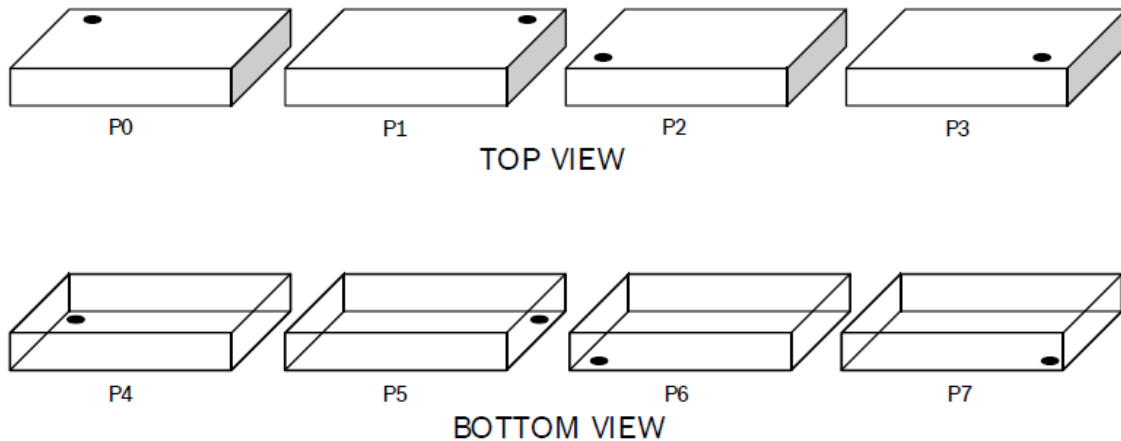
The default device axis orientation is shown in Figure 9: below. This orientation is valid for all sensor outputs (physical and virtual). “Accel” is already integrated in the BHA according to this axis orientation. If there are other sensors connected to the hub, e.g. “Gyro” and/or “Magnet” the physical alignment of the corresponding sensors and its axis should be according to Figure 9 as well to guarantee correct 9DoF data fusion.

Figure 9: Device sensing axis orientation



In case the default axis orientation of the BHA and/or its sensor extensions does not match to the target device coordinate system an axis remapping can be performed to reassign the sensing axes of the BHA so, that they match to the axes defined by the target device coordinate system. Possible placement and remapping options are given in Figure 10.

Figure 10: Placement options of BHA with respect to device orientation



If the sensing axis of the sensor do not match to the coordinate system of the target device or to sensor extensions which are potentially connected to the hub, several tools can be used to remap the sensing assignment in the firmware file. This enables a subsequent axis alignment of all virtual and physically integrated and/or attached sensors to match the target coordinate system.

For details regarding the orientation matrix refer to Table 21: Physical Sensor Information; page 49

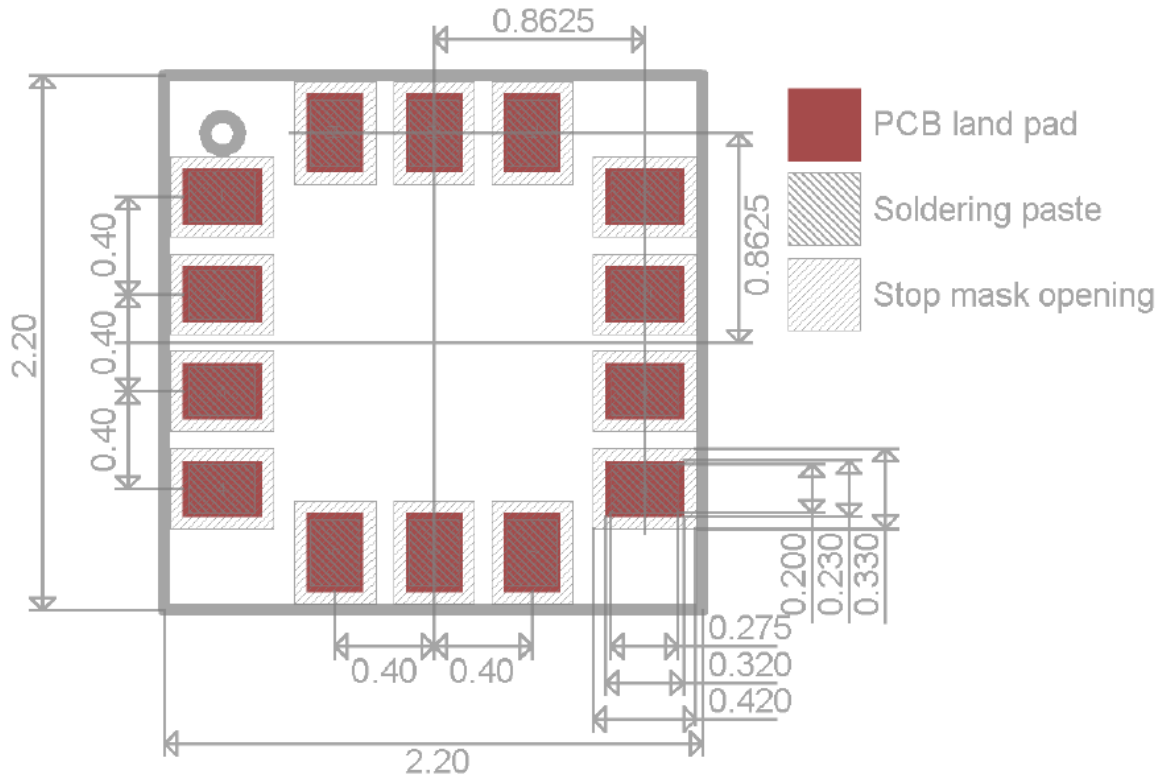
For details and technical support regarding the remapping of this matrix please refer to the Bosch Sensortec document BST-BHY1-AN001-00 “Axes remapping” placed within the section “Application notes”, available on

[https://www.bosch-sensortec.com/bst/support\\_tools/downloads/overview\\_downloads](https://www.bosch-sensortec.com/bst/support_tools/downloads/overview_downloads)

or contact our regional offices, distributors and sales representatives.

### 14.3 Landing pattern recommendation

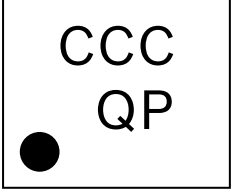
Figure 11: Landing pattern



## 14.4 Marking

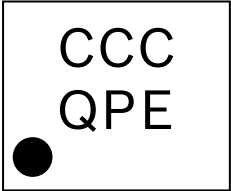
### 14.4.1 Mass production

Table 38: Marking of mass production material

Labeling	Name	Symbol	Remark
	Counter ID	CCC	3 alphanumeric digits, variable to generate trace-code
	First letter of second row	R	Product identifier Q = "R" denoting BHA250, 0.273.141.231 Q = "S" denoting BHA250B, 0.273.141.310
	Second letter of second row	P	Internal use
	Pin 1 identifier	●	--

### 14.4.2 Engineering samples

Table 39: Marking of engineering samples

Labeling	Name	Symbol	Remark
	Counter ID	CCC	3 alphanumeric digits, variable to generate trace-code
	First letter of second row	R	Product identifier Q = "R" denoting BHA250, 0.273.141.231 Q = "S" denoting BHA250B, 0.273.141.310
	Second letter of second row	P	Internal use
	Third letter of second row	E	"E" denotes engineering samples
	Pin 1 identifier	●	--

## 14.5 Soldering guidelines

The moisture sensitivity level of the BHA sensors corresponds to JEDEC Level 1, see also

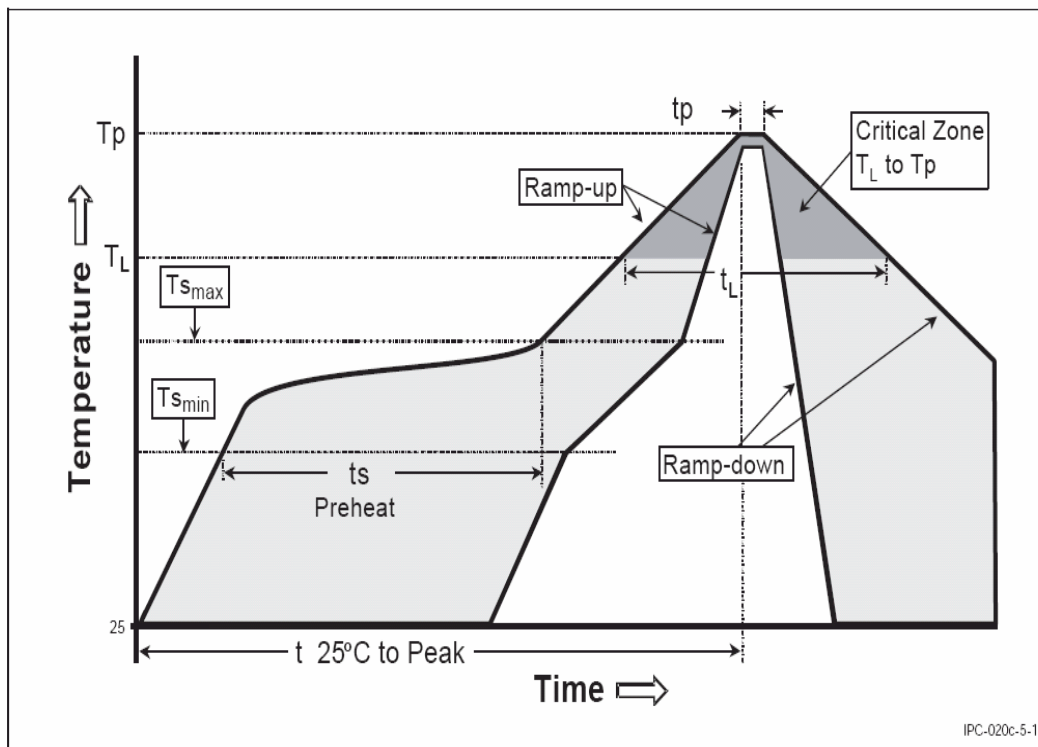
- IPC/JEDEC J-STD-020C “Joint Industry Standard: Moisture/Reflow Sensitivity Classification for non-hermetic Solid State Surface Mount Devices”
- IPC/JEDEC J-STD-033A “Joint Industry Standard: Handling, Packing, Shipping and Use of Moisture/Reflow Sensitive Surface Mount Devices”

The sensor fulfils the lead-free soldering requirements of the above-mentioned IPC/JEDEC standard, i.e. reflow soldering with a peak temperature up to 260°C.

Figure 12: Soldering profile

Profile Feature	Pb-Free Assembly
Average Ramp-Up Rate ( $T_{S_{max}}$ to $T_p$ )	3° C/second max.
<b>Preheat</b> – Temperature Min ( $T_{S_{min}}$ ) – Temperature Max ( $T_{S_{max}}$ ) – Time ( $t_{s_{min}}$ to $t_{s_{max}}$ )	150 °C 200 °C 60-180 seconds
Time maintained above: – Temperature ( $T_L$ ) – Time ( $t_L$ )	217 °C 60-150 seconds
Peak/Classification Temperature ( $T_p$ )	≤ 260 °C
Time within 5 °C of actual Peak Temperature ( $t_p$ )	20-40 seconds
Ramp-Down Rate	6 °C/second max.
Time 25 °C to Peak Temperature	8 minutes max.

**Note 1:** All temperatures refer to topside of the package, measured on the package body surface.





## 14.6 Handling instructions

Micromechanical sensors are designed to sense acceleration with high accuracy even at low amplitudes and contain highly sensitive structures inside the sensor element. The MEMS sensor can tolerate mechanical shocks up to several thousand g's. However, these limits might be exceeded in conditions with extreme shock loads such as e.g. hammer blow on or next to the sensor, dropping of the sensor onto hard surfaces etc.

We recommend avoiding g-forces beyond the specified limits during transport, handling and mounting of the sensors in a defined and qualified installation process.

This device has built-in protections against high electrostatic discharges or electric fields (e.g. 2kV HBM); however, anti-static precautions should be taken as for any other CMOS component. Unless otherwise specified, proper operation can only occur when all terminal voltages are kept within the supply voltage range. Unused inputs must always be tied to a defined logic voltage level.

For more details on recommended handling, soldering and mounting please contact your local Bosch Sensortec sales representative and ask for the "Handling, soldering and mounting instructions" document.

## 14.7 Tape and reel specification

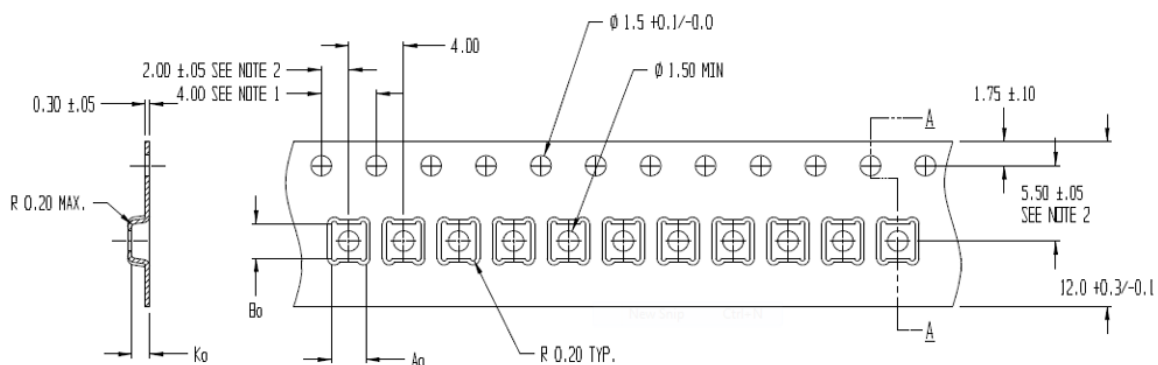
The BHA is shipped in standard cardboard box.

The box dimension for one reel is: L x W x H = 34.5 cm x 34.5 cm x 4.5 cm.

BHA quantity: 10,000 pcs per reel, please handle with care.

The following picture describes the dimensions of the tape used for shipping the BHA sensor device. The material of the tape is made of conductive polysterene (IV).

Figure 13: Tape and reel dimensions in mm



**NOTES:**

1. 10 SPROCKET HOLE PITCH CUMULATIVE TOLERANCE  $\pm 0.2$
2. POCKET POSITION RELATIVE TO SPROCKET HOLE MEASURED AS TRUE POSITION OF POCKET, NOT POCKET HOLE
3.  $A_0$  AND  $B_0$  ARE CALCULATED ON A PLANE AT A DISTANCE "R" ABOVE THE BOTTOM OF THE POCKET.

$$A_0 = 2.40 \pm 0.2 / -0.0$$

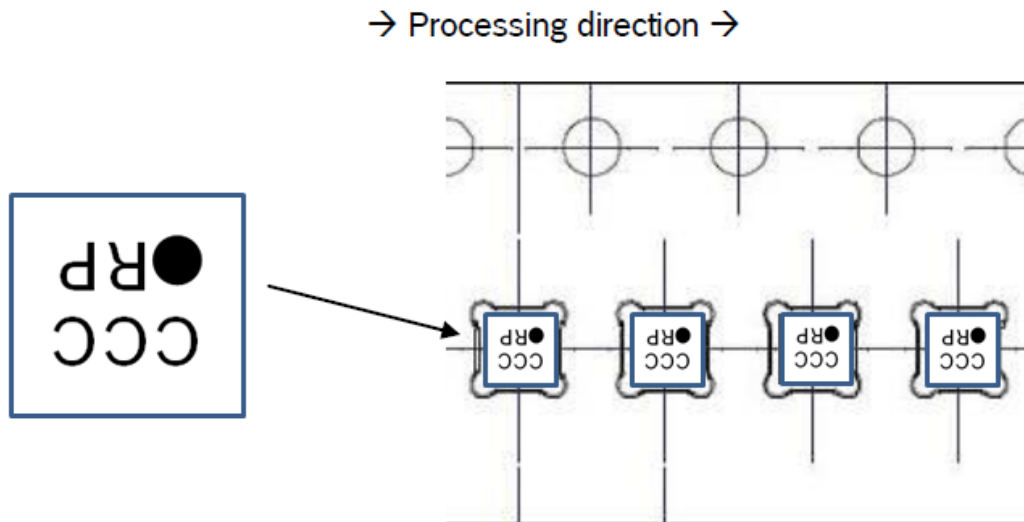
$$B_0 = 2.40 \pm 0.2 / -0.0$$

$$K_0 = 1.30$$

Note:  $K_0$  has the default tolerance of  $\pm 0.2$ .

### 14.7.1 Orientation within the reel

Figure 14: Orientation of the BHA devices relative to the tape



## 14.8 Environmental safety

The BHA sensor meets the requirements of the EC restriction of hazardous substances (RoHS) directive, see also:

*Directive 2011/65/EU und (EU) 2015/863 of the European Parliament and of the Council on the restriction of the use of certain hazardous substances in electrical and electronic equipment.*

## 14.9 Halogen content

The BHA is halogen-free. For more details on the analysis results please contact your Bosch Sensortec representative.

## 14.10 Multiple sourcing

Within the scope of Bosch Sensortec's ambition to improve its products and secure the mass product supply, Bosch Sensortec employs multiple sources in the supply chain.

While Bosch Sensortec takes care that all of technical parameters are described above are 100% identical for all sources, there can be differences in device marking and bar code labeling.

However, as secured by the extensive product qualification process of Bosch Sensortec, this has no impact to the usage or to the quality of the product.

## 15. Legal disclaimer

### 15.1 Engineering samples

Engineering Samples are marked with an asterisk (\*) or (e) or (E). Samples may vary from the valid technical specifications of the product series contained in this data sheet. They are therefore not intended or fit for resale to third parties or for use in end products. Their sole purpose is internal client testing. The testing of an engineering sample may in no way replace the testing of a product series. Bosch Sensortec assumes no liability for the use of engineering samples. The Purchaser shall indemnify Bosch Sensortec from all claims arising from the use of engineering samples.

### 15.2 Product use

Bosch Sensortec products are developed for the consumer goods industry. They may only be used within the parameters of this product data sheet. They are not fit for use in life-sustaining or security sensitive systems. Security sensitive systems are those for which a malfunction is expected to lead to bodily harm or significant property damage. In addition, they are not fit for use in products which interact with motor vehicle systems.

The resale and/or use of products are at the purchaser's own risk and his own responsibility. The examination of fitness for the intended use is the sole responsibility of the Purchaser.

The purchaser shall indemnify Bosch Sensortec from all third party claims arising from any product use not covered by the parameters of this product data sheet or not approved by Bosch Sensortec and reimburse Bosch Sensortec for all costs in connection with such claims.

The purchaser must monitor the market for the purchased products, particularly with regard to product safety, and inform Bosch Sensortec without delay of all security relevant incidents.

### 15.3 Application examples and hints

With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Bosch Sensortec hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights or copyrights of any third party. The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. They are provided for illustrative purposes only and no evaluation regarding infringement of intellectual property rights or copyrights or regarding functionality, performance or error has been made.

## 16. Document history and modifications

Rev. No	Chapter	Description of modification/changes	Date	
1.0	All	1 <sup>st</sup> Release version	Jul 2016	
1.1	All 10.17 10.19 10.20 14.4	Introduced new tech. ref. code BHA250B: 0.273.141.310	Dec 2016	
	All 9.11 10.4			Added Android 6 / M / Marshmallow (non HiFi)
	11.1 14.2			Updated explanation of byte 11-15 "orientation matrix" Introduced BST-BHY1-AN001-00 "Axes remapping".
	10.12			Corrected Type in register description 0x55, Bit 7
1.2	11.5 14.8	Added description for soft-pass-through mode Updated reference to ROHS directive	Mar 2017	

Bosch Sensortec GmbH  
 Gerhard-Kindler-Strasse 9  
 72770 Reutlingen / Germany

contact@bosch-sensortec.com  
 www.bosch-sensortec.com

Modifications reserved | Printed in Germany  
 Specifications subject to change without notice  
 Document number: BST-BHA250(B)-DS000-01  
 Revision\_1.2\_160317