BHI385 – Extended range, and Self-learning AI

Programmable, High-g Inertial Measurement Unit with On-Device AI and Sensor Fusion

BHI385 Datasheet

Document revision	1.0
Document release date	August, 2025
Document number	BST-BHI385-DS001-01
Sales part numbers	BHI385: 0 273 017 068
Notes	Data and descriptions 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 appearance.

BHI385 Programmable, High-g Inertial Measurement Unit with On-Device AI and Sensor Fusion

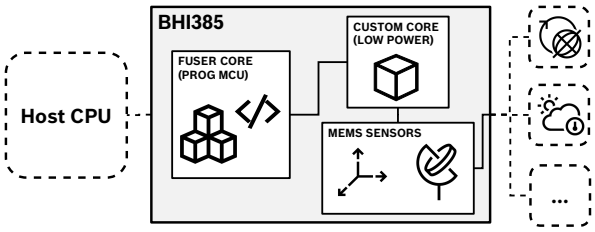


Figure 1: BHI385 Simplified

General description

The BHI385 is a highly integrated, ultra-low-power, programmable smart sensor system consisting of:

- A best-in-class 6-axis IMU with extended g-range
- A programmable 32-bit microcontroller (Fuser2)
- An additional ultra-low-power MCU
- A powerful software framework and SDK Pre-installed sensor data processing algorithms

The BHI385 is integrated in a 2.5 mm X 3 mm LGA package, pin-to-pin backward compatible with many Bosch Sensortec IMUs for easy replacement and platform design. Its unique combination of high-g sensing, on-device AI, and ultra-low power consumption makes it the ideal solution for sophisticated always-on activity and sports tracking.

Integrated sensor (6-DoF IMU)

- 16-bit 3-axis accelerometer with up to 32g full-scale[1)]
- 16-bit 3-axis gyroscope

Integrated programmable MCU (Fuser2)

- ARC EM4 CPU (up to 3.6 CoreMark/MHz)
- Long Run mode: 950 µA @ 20 MHz running CoreMark®
- Turbo mode: 2.8 mA @ 50 MHz running CoreMark®
- Floating Point Unit (FPU)
- Memory Protection Unit (MPU)
- 4-channel micro DMA Controller
- 256 kByte on-chip SRAM
- 144 kByte on-chip ROM preloaded with software

Integrated ultra-low-power MCU (Custom Core)

- Optimized for accelerometer-based always-on algorithms

Connectivity

- Host interface configurable as SPI or I2C
- Two secondary master interfaces (one I²C interface, and one selectable SPI or I²C)
- Up to 8 available GPIOs
- Fast I/O operations (SPI, GPIOs up to 50 MHz ; I2C up to 3.4 MHz)

Software Features BHI385

- Open & Programmable Platform: An event-driven software framework and open SDK enable the development of custom, embedded algorithms
- Advanced Sensor Fusion: The BSX library provides robust 6DoF/9DoF orientation, gravity vector, and auto-calibration across multiple power modes
- On-Chip AI Capabilities: Features a self-learning AI model and supports custom neural networks, easily deployed with Bosch Motion AI Studio
- Pre-installed algorithms for step counting, tap detection, and gesture recognition run on a dedicated core to minimize power consumption

Target applications and devices

- Wearable devices like smartwatches, hearables and smart glasses
- Smartphones, tablets and other mobile communication devices
- 24/7 always-on sensor data processing at ultra-low power consumption

[1] When the sensor is configured for the 32g range, its output may not change with accelerations exceeding the limits in Table 131: Operating conditions accelerometer. The maximum measurable acceleration is typically +-28g.

Table of contents

List of figures

List of tables

1 Overview

The BHI385 belongs to the BHI family of highly integrated, ultra-low power programmable smart sensor systems. The BHI385 integrates the Fuser2 processor, which is based on the 32-Bit ARC™ EM4™ floating point RISC processor, an integrated Inertial Measurement Unit (6DoF IMU) and a powerful Event-Driven Software Framework specifically designed for signal data processing and comes with pre-installed sensor fusion and other sensor data processing algorithms.

The BHI385 offers two secondary high speed master interfaces with I2C and/or SPI capability, and up to 8 GPIOs which can be configured in a flexible way (e.g. Chip Select or Interrupt lines). In this way, the BHI385 can be used as a “Sensor Hub” connected to external sensors and devices, such as magnetic field sensors, pressure sensors and many more. All sensor data from both integrated MEMS IMU and external sensors can be efficiently integrated, synchronized and processed by the integrated Fuser2.

The BHI385 is mainly intended to be used as coprocessor, offloading the main CPU from any sensor data processing related tasks, like sensor fusion, data batching, position tracking, activity recognition and gesture detection with high precision and low latency while significantly reducing the overall system power consumption. When used as a coprocessor, the BHI385 supports a wide variety of host CPU devices, ranging from a small MCU up to multicore application processors. The BHI385 communicates with the Host CPU through its primary high speed I2C or SPI interface.

The BHI385 can be used as a co-processor in different applications:

- **As a high-end Smart Sensor with integrated features**

  The BHI385 can be used as a ready-to-use solution, delivering advanced functionality out-of-the-box. It comes with a powerful sensor fusion library (BSX) for precise orientation and includes a suite of pre-configured virtual sensors for standard gesture and activity recognition. Uniquely, the BHI385 also features an integrated Self-Learning AI core. This allows the device to intelligently learn and recognize custom cyclic movements directly on the chip, making it especially suited for detailed workout and exercise evaluation without requiring any initial programming or effort-intensive model training. Bosch Sensortec offers firmware versions for a variety of use cases (e.g. wearables, hearables).

- **As an open programmable Smart Sensor**

  The BHI385 functionality can be customized and extended by the customer. Bosch Sensortec offers a Software Development Kit (SDK) and a sophisticated Event-Driven Software Framework that enable the development of additional algorithms (virtual sensors). The framework provides powerful services including power management, FIFO management, and event synchronization. To simplify AI development, Bosch offers the Motion AI Studio, a powerful tool that enables the easy integration and deployment of custom-trained neural networks directly onto the BHI385. This allows developers to create highly specific and efficient motion-based AI solutions with minimal effort.

2 Hardware Features

2.1 Fuser2 MCU Core

- 32-bit Synopsys DesignWare™ ARC™ EM4 CPU
- ARCv2 16/32-bit RISC instruction set architecture
- CPU provides up to 1.6 DMIPS/MHz, 3.6 CoreMark/MHz
- Operating frequency: 20 MHz (Long Run mode) and 50 MHz (Turbo mode)
- Maximum CPU power consumption in Long Run mode: 950 µA (47.5 µA/MHz)
- Maximum CPU power consumption in Turbo mode: 2.8 mA (56 µA/MHz)
- Harvard architecture with closely coupled memories for instructions (ICCM) and data (DCCM)
- Single-precision FPU with IEEE 754 compliance
- Memory protection unit (MPU)
- 4-channel micro-DMA controller
- Timer and core watchdog
- Action-point support for debugging
- Hardware support for fast CRC32 calculation
- Programmable by customers and partners

2.2 Integrated Physical Sensors

- Low power, low noise inertial measurement unit (6DoF IMU) consisting of:
  - 16-bit digital triaxial gyroscope
  - 16-bit digital triaxial accelerometer with up to 32g full-scale[1)]
- Supports data rates up to 1600 Hz (Accelerometer), 6400 Hz (Gyroscope)
- Built in power management unit and enhanced interrupt engine (for Fuser2 wake-up)
- Auxiliary master interface:
  - for direct magnetometer attachment (upgrade to 9DoF IMU)
  - as OIS data interface

2.3 Integrated Low-Power Custom Core (Bosch Sensortec Core)

- Integrated low-power custom MCU (Bosch Sensortec IP)
- Optimized for simple always-on algorithms
- Internal simple interrupt engine
- Comes with integrated preprocessing algorithms from Bosch Sensortec
- Not programmable by customers
- Used in combination with Fuser2 for a more efficient power use

2.4 System Control Blocks

- Brown-out reset
- Voltage regulator for digital core
- On-chip system oscillator: 20 MHz, 50 MHz
- On-chip timer oscillator: 128 kHz

[1] When the sensor is configured for the 32g range, its output may not change with accelerations exceeding the limits in Table 131: Operating conditions accelerometer. The maximum measurable acceleration is typically +-28g.

2.4.1 Reset options

Available reset sources:

- Power-on reset
- Reset pad
- Host reset command
- Core watchdog

2.4.2 Oscillators & Clocks

Available clock domains[1]:

- Host interface clock (up to 50 MHz in SPI mode)
- System clock (20 MHz or 50 MHz) provided by internal system oscillator
- Timer clock (64 kHz derived from 128 kHz internal timer oscillator)
- External clock (input to universal timer, from GPIO up to 1 MHz)
- JTAG debug clock (up to system clock / 8)

2.5 Host Interface

- Pin-configurable as either I2C (up to 3.4 MHz) or SPI (up to 50 MHz) slave interface
- I2C slave interface monitored by I2C watchdog
- Configurable (latched/non-latched, push-pull or open drain) host interrupt output
- Dual DMA enabled command I/O FIFO for efficient command and status handling
- Dual data FIFO sensor event batching

2.6 Memory Subsystem

2.6.1 On-Chip SRAM

256 kByte of SRAM in total, organized in nine individual RAM banks:

- 16 kByte dedicated to ICCM space
- 16 kByte dedicated to DCCM space
- 7x32 kByte RAM banks in a RAM pool, configurable to either ICCM space, DCCM space or powered off

2.6.2 ROM with Integrated Software

144 kByte of program ROM in fast-access ICCM space, containing:

- Bootloader
- BSX sensor fusion library
- Functions of standard C and math library
- OPENRTOS kernel
- SHA256 and ECDSA (NIST FIPS PUB 186-4) digital signature libraries

2.6.3 OTP

- 128 Byte OTP memory for Bosch internal factory calibration and secure boot key storage (not customer programmable)

[1] These clock domains are asynchronous to each other.

2.7 Peripheral Subsystem

2.7.1 Secondary Master Interface 1

- Master Interface 1 (M1) configured as SPI for connection of the integrated IMU sensor, operating at 10 MHz by default.

2.7.2 Secondary Master Interface 2

- Master Interface 2 (M2) configurable as I2C (up to 1 MHz) or SPI (up to 50 MHz) for connection of additional physical sensors to the BHI385 (see Configuration of Master Interfaces)
- Various chip selects can be associated to M2 in SPI mode
- For more Information about the Master Interface Configuration, see Section 9.1.

2.7.3 Secondary Master Interface 3

- Master Interface 3 (M3) configurable as I2C (up to 1 MHz) for connection of additional physical sensors to the BHI385 (see Configuration of Master Interfaces)
- For more Information about the Master Interface Configuration, see Section 9.1.

2.7.4 GPIOs

- Up to 8 software configurable GPIOs available for use as CS lines, Interrupt lines or other purposes. Supported configurations:
  - Input: internal configurable pull-ups, High-Z, selectable as event interrupt source
  - Output: Push/Pull, High-Z, Open Drain, selectable drive strength.

For more Information about the GPIOs configuration, see Section 9.2.

2.7.5 Universal Timers & Counter

- One 32-bit real-time counter incremented by the Timer Clock running at 64 kHz (used for time stamp generation)
- One 32-bit interval timer for generation of periodic interrupts, software counters, etc. It is driven by the timer clock running at 64 kHz
- One 16-bit universal timer operating on an external clock input or the internal timer clock. It can trigger internal interrupts or create external output signals, e.g. a PWM signal.
- Software API to access/program all timers & counter

2.7.6 Interrupt Sources

Various sources can generate interrupt requests to the MCU in the BHI385 (Fuser2) for which interrupt handlers can be registered. For some interrupt sources, default handlers are already registered to ensure proper operation of the Event-Driven Software Framework.

Interrupt requests can originate from:

- Host interface
  - Host writes or read into specific host interface registers
  - Host interface detects end of data transmission
  - Buffer overflow or underflow during data transmission
- I2C and SPI master interfaces
  - Master interface signals end of a data transmission
  - GPIO events

For more detailed information about the interrupt configuration, see Section 10 Event and Interrupt Configuration.

2.7.7 Event Subsystem

- Advanced event subsystem supporting effort free time synchronization between external events and internally handled (sensor) data
- Up to 9 event channels, which can be mapped to a variety of GPIO pins
- Any GPIO, associated to an event channel, can serve as external interrupt source
- Event triggered hardware logic for capture and storage of real-time counter value for automatic time synchronization between external events and internal time base

For more detailed information about the interrupt configuration, see Section 10 Event and Interrupt Configuration.

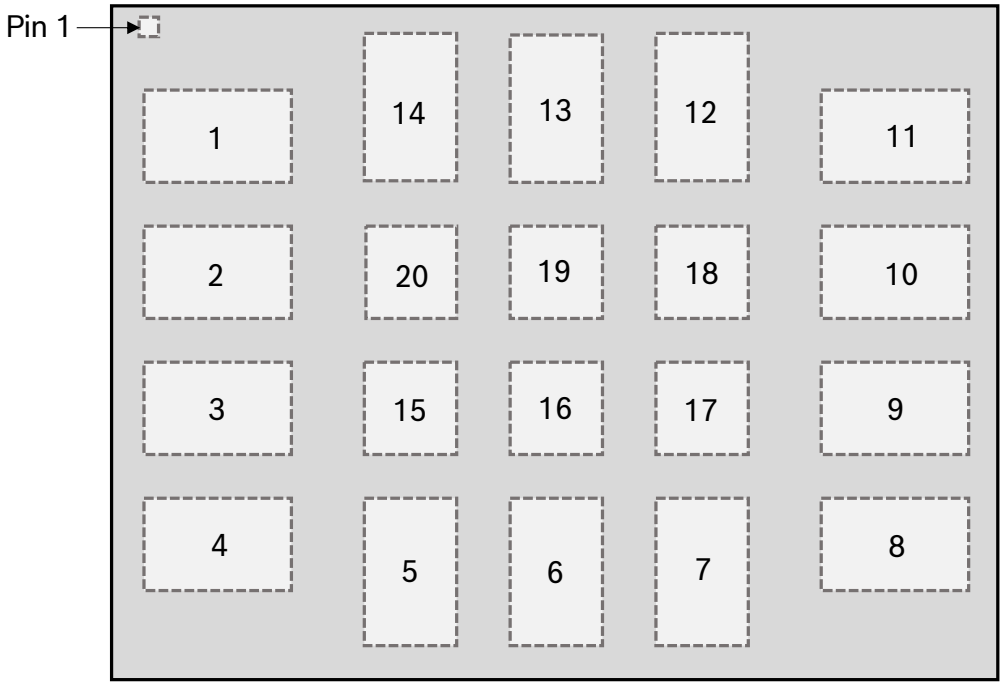
2.8 Debug Interface

- 2-wire JTAG interface
- Supports direct read/write of CPU data and aux registers as well as ICCM/DCCM memory locations
- Hardware breakpoint and single step execution
- Supported with Metaware Debugger for ARC and OpenOCD for GCC

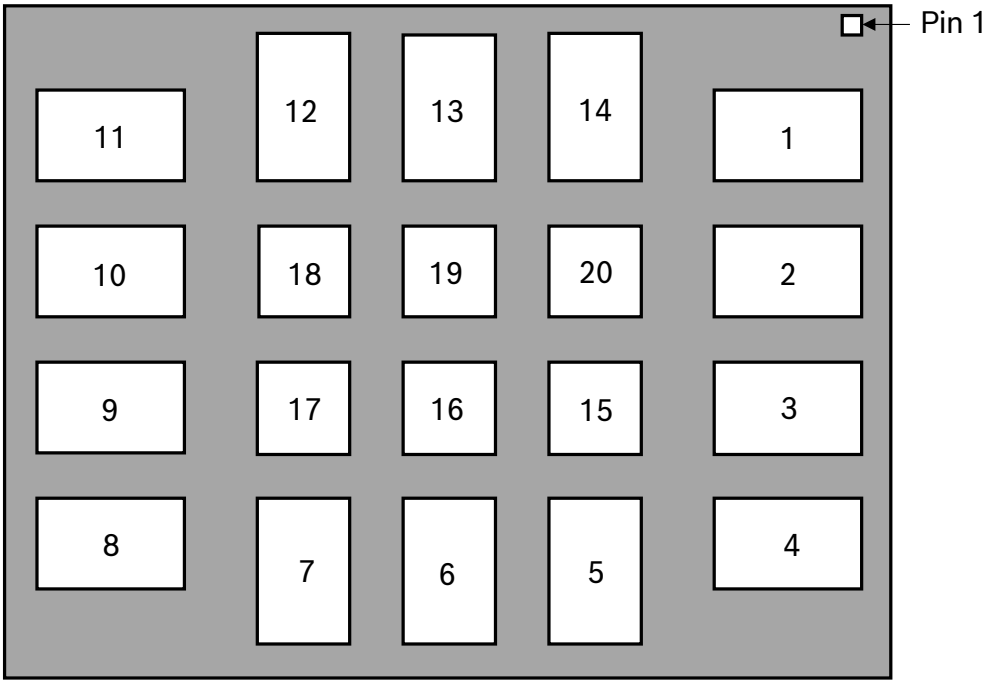
3 Pad Connections and Description

3.1 Pad Description

Figure 2: Pad Description



Pin-out top view



Pin-out bottom view

Table 1: Pin-out and pin connections

Pad #	Pad Name	Function Group	Reset Value	Description
1	HSDO	Host Interface	Input, Pull-down	Host Interface: SPI mode: MISO I2C mode: address select
2	ASDX	IMU Auxiliary Interface, Master Interface 2	Input, Pull-up	Sensor Aux Interface: OIS mode: MOS I2C mode: SDA Master 2: SPI MOSI / I2C SDA
3	ASCX	IMU Auxiliary Interface, Master Interface 2	Input, Pull-up	Sensor Aux Interface: OIS mode: Clock I2C mode: SCL Master 2: SPI SCK / I2C SCL
4	HIRQ	Host Interface	Input, Pull-up	Host Interrupt Signal (Output)
5	VDDIO	Supply		Digital IO and Fuser2 Supply
6	GNDIO	Ground		Digital IO and Fuser2 Ground
7	GND	Ground		Analog Sensor Ground
8	VDD	Supply		Analog Sensor Supply
9	VREG	Supply		Voltage regulator output
10	OCSB	IMU Auxiliary Interface, GPIO	Input, Pull-up	Sensor Aux Interface: OIS mode: Chip Select I2C mode: unused Master 2: SPI Chip Select 1
11	OSDO	IMU Auxiliary Interface, Master Interface 2	Input, Pull-up	Sensor Aux Interface: OIS mode: MISO I2C mode: unused Master 2: SPI MISO / I2C unused
12	HCSB	Host Interface	Input, Pull-up	Host Interface: SPI mode: Chip Select I2C: Switch to SPI mode* (to work in I2C mode, this line should be maintained as High)
13	HSCX	Host Interface	Input, Pull-up	Host Interface: SPI SCK / I2C SCL
14	HSDX	Host Interface	Input, Pull-up	Host Interface Serial Data SPI MOSI / I2C SDA
15	M3SCL	Master Interface 3, Debug	Input, Pull-up	Master 3 I2C SCL (I2C Serial Clock), Fuser2 Debug Clock (JTAG_CLK)
16	JTAG_DIO	Debug	Input, Pull-up	Fuser2 Debug Data (JTAG_DIO)
17	RESETN	System Control	Input, Pull-up	Reset input, active low
18	M3SDA	Master Interface 3	Input, Pull-up	Master 3: I2C SDA (I2C Serial Data)
19	RESV2	reserved	Input, Pull-up	Reserved: do not connect (internal IMU interrupt)
20	RESV1	reserved	Input, Pull-up	Reserved: do not connect (internal IMU interrupt)

3.2 Setup of Multifunction pins

Almost all pads of the device can have multiple functions assigned. Next to their primary function (Host Interface, Master Interfaces, Debug, OIS and Aux Interface), most pads can also serve as General Purpose IOs (GPIOS), Event Interrupt Inputs, or timer/capture inputs/outputs. The following diagram depicts the multiple assignments of the pads.

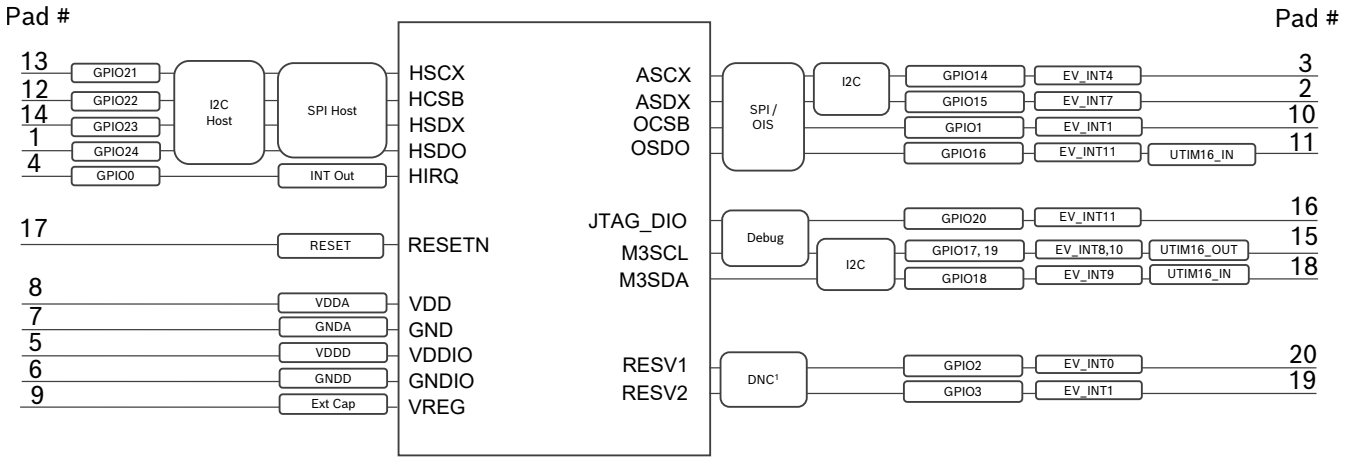


Figure 3: Pinout BHI385

The selection of the function of each pad depends on the boot state of the device and the configuration within the firmware. The following table describes the available functions of each pad and how they are configured.

[1]RESV1 and RESV2 are used for internal connections between the sensor and Fuser2 and shall be left open in the application. They are marked therefore as DNC (Do not connect) for function 1.

Table 2: Pad functions and initial options

Pad #	Pad Name	Primary Function Group	Function 1	Function 2	Function 3	Function 4 Event Interrupt[3]		Function 5 Universal Timer	
			SPI etc.		GPIO	Config 1	Config 2	Config 1	Config 2
17	RESETN	System Control	Fuser2 Reset, active low						
13	HSCX	Host Interface 1	Host SPI SCK	Host I2C SCL	GPIO21[1]				
12	HCSB		Host SPI chip sel	Protocol sel: keep high for I2C	GPIO22[1]				
14	HSDX		Host SPI MOSI	Host I2C SDA	GPIO23[1]				
1	HSDO		Host SPI MISO	Host I2C adr0	GPIO24[1]				
4	HIRQ		Host interrupt[2]	Host interrupt[2]	GPIO0				
3	ASCX	Master Interface 2	SPI SCK	I2C SCL	GPIO14		EVINT4		
			OIS SPI SCK	Auxiliary I2C SCL					
2	ASDX		SPI MOSI	I2C SDA	GPIO15		EVINT7		
			OIS SPI MOSI	Auxiliary I2C SDA					
11	OSDO		SPI MISO		GPIO16	EVINT11		IN	
			OIS SPI MISO						
10	OCSB		SPI chip select 1		GPIO1		EVINT1		
			OIS SPI chip select						
15	M3SCL	Master Interface 3, Debug	I2C SCL		GPIO17		EVINT8		OUT
			JTAG Clock		GPIO19		EVINT10		
18	M3SDA		I2C SDA		GPIO18		EVINT9		IN
16	JTAG_ DIO		JTAG Data IO		GPIO20		EVINT11		
19	RESV2		(Integrated sensor IRQ 2)		GPIO3	EVINT1			
20	RESV1		(Integrated sensor IRQ 1)		GPIO2	EVINT0	EVINT0		

[1] Function 3 is locked after initial selection. This is not valid for HIRQ.
[2] Function 1 is defined by firmware. In hardware these pins are standard GPIOs.
[3] GPIO must be configured for correct direction – event interrupts are inputs.

Table 3: Pad functions and initial conditions

Pad #	Pad Name	Primary Function Group	Function after:		
			Reset	**Bootloader completed**	**Firmware load**
17	RESETN	System Control	Function 1	Function 1	Function 1
13	HSCX	Host Interface[1)]	Function 2	Function 1 or 2 defined by HCSB	Function 1 or 2 defined by HCSB
12	HCSB				
14	HSDX				
1	HSDO				
4	HIRQ		Function 3		
3	ASCX	Master Interface 2, IMU Auxiliary Interface	Function 3	Function 3	Defined by FW: SIF0: Function 1 SIF1: Function 2
2	ASDX				
11	OSDO				
15	M3SCL	Master Interface 3, Debug	Function 3	Function 3	1 or 3 defined by FW
18	M3SDA				
10	OCSB	Master Interface 2 Chip Select, IMU Auxiliary Interface	Function 3	Function 3	1 to 5 defined by FW
20	RESV1				defined by FW
19	RESV2				defined by FW
16	JTAG_DIO	Debug[1)]	Function 3	Function 1 or 3: defined by product type	Function 1 or 3: defined by FW

[1] Primary Function is locked after initial selection. This is not valid for HIRQ.
[2] Function 1 is defined by firmware. In hardware these pins are standard GPIOs.

4 System Configuration

4.1 Connection Diagrams

4.1.1 Primary IF = I2C; Secondary IF = None

Figure 4: BHI385 connection diagram (primary IF = I2C; secondary IF = none) shows a possible connection of BHI385 in I2C host protocol mode, with no additional sensor connected to the BHI385.

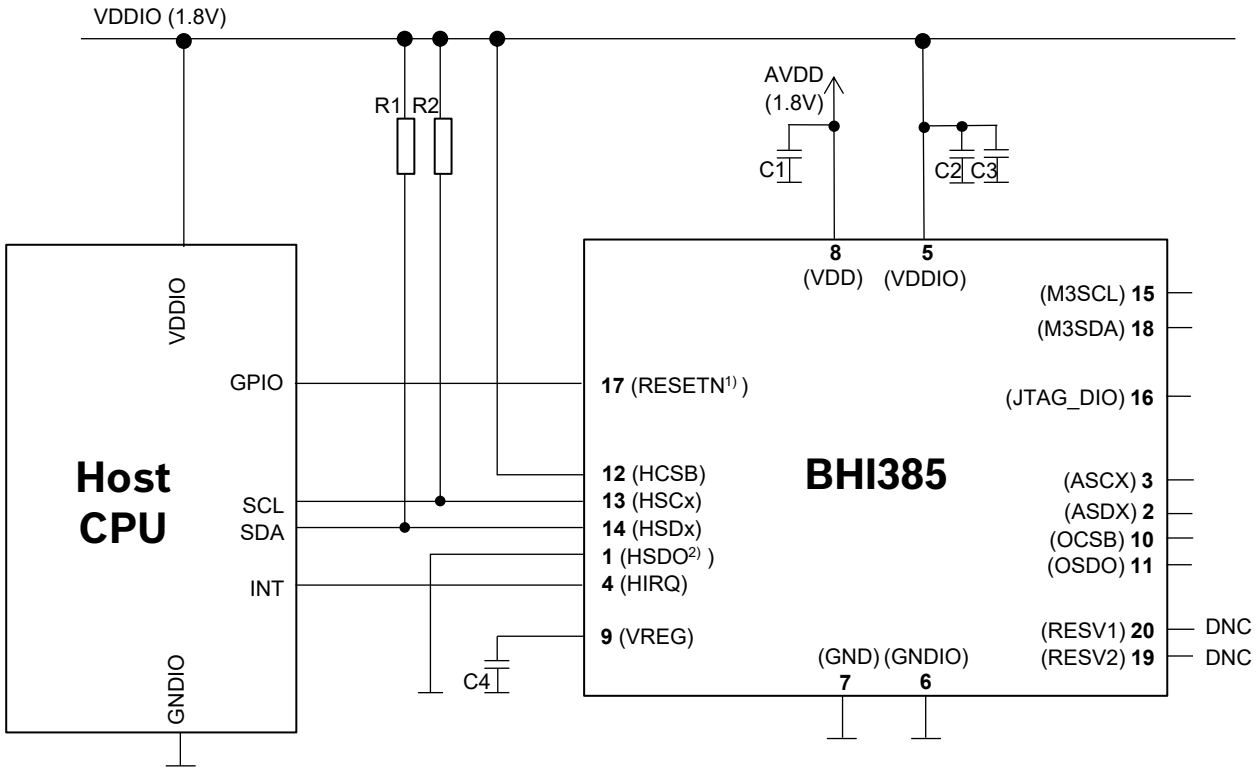


Figure 4: BHI385 connection diagram (primary IF = I2C; secondary IF = none)

Table 4: Typical vales for external circuit components

Component	Value	Remarks
R1	4.7kΩ	Pull-up resistor for SDA, only for host interface in I2C mode
R2	4.7kΩ	Pull-up resistor for SCL, only for host interface in I2C mode
C1	100nF	Bypass capacitor AVDD to GND
C2	100nF	Bypass capacitor VDDIO to GNDIO
C3	1µF	Bypass capacitor VDDIO to GNDIO
C4	220nF	Regulator capacitor VREG

[1] Recommendation: Connect to VDDIO, if not needed.
[2] HSDO can be used for device address selection, if needed.

4.1.2 Primary IF = I2C; Secondary IF = I2C Slave device

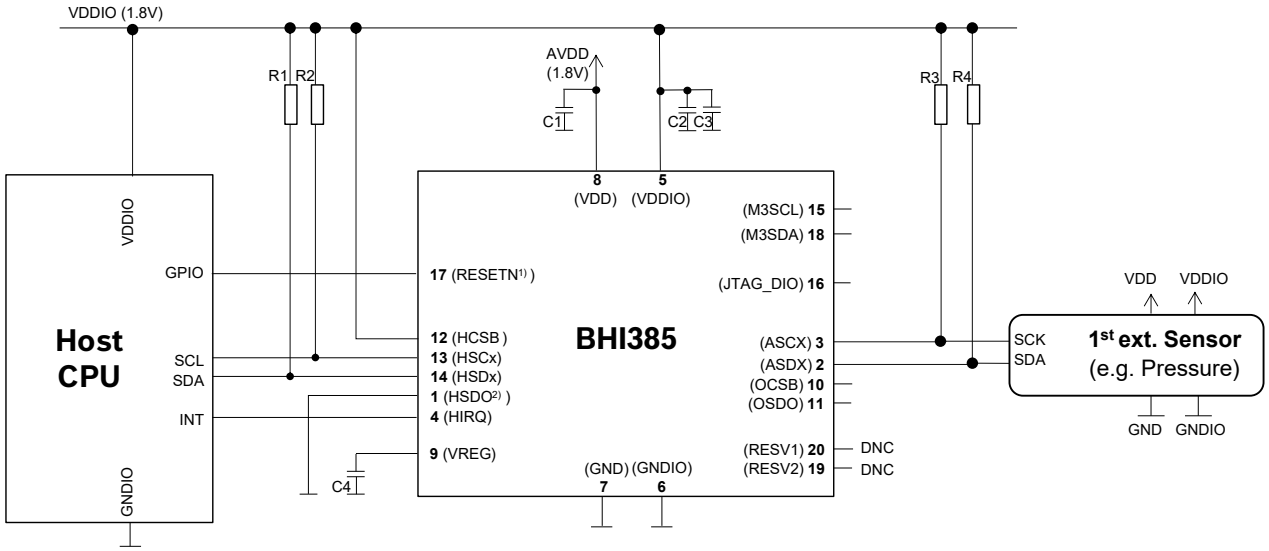


Figure 5: BHI385 connection diagram (primary IF = I2C; secondary IF = I2C slave device)

Table 5: Typical vales for external circuit components

Component	Value	Remarks
R1	4.7kΩ	Pull-up resistor for SDA, only for host interface in I2C mode
R2	4.7kΩ	Pull-up resistor for SCL, only for host interface in I2C mode
R3	4.7kΩ	Pull-up resistor for ASCX, only for M2 interface configured in I2C mode
R4	4.7kΩ	Pull-up resistor for ASDX, only for M2 interface configured in I2C mode
C1	100nF	Bypass capacitor AVDD to GND
C2	100nF	Bypass capacitor VDDIO to GNDIO
C3	1µF	Bypass capacitor VDDIO to GNDIO
C4	220nF	Regulator capacitor VREG

[1] Recommendation: Connect to VDDIO, if not needed .
[2] HSDO can be used for device address selection, if needed.

4.1.3 Primary IF = SPI; Secondary IF = SPI Slave device

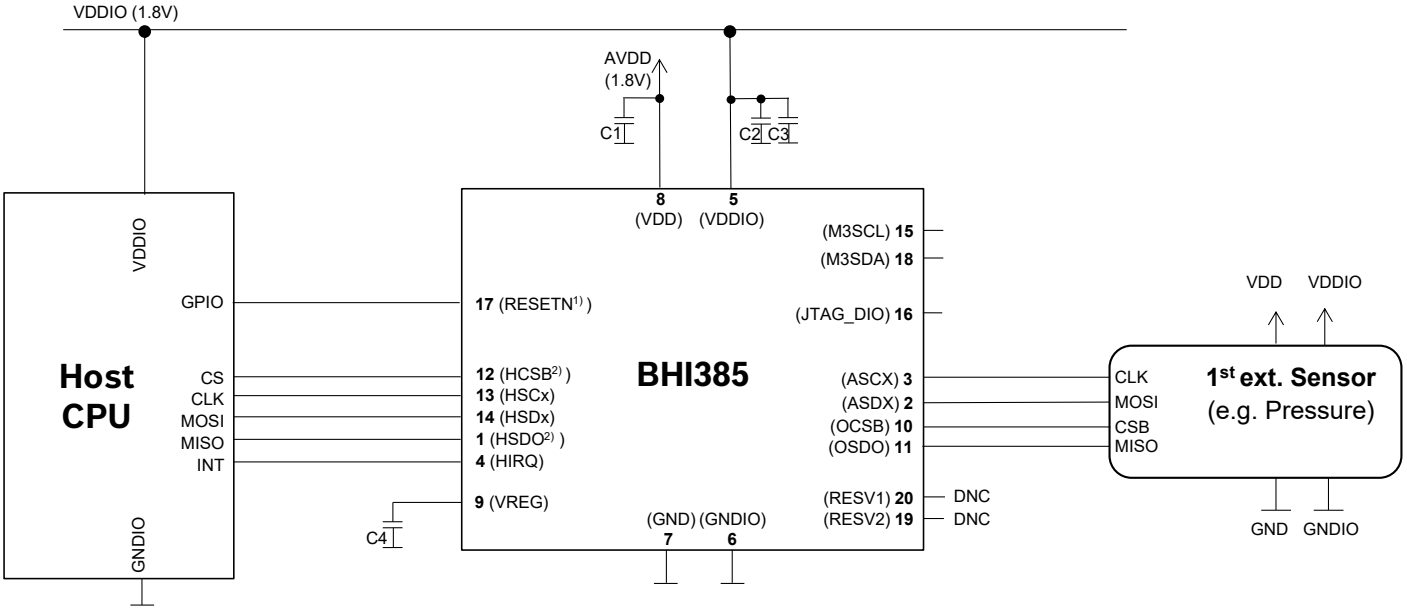


Figure 6: BHI385 connection diagram (primary IF = SPI ; secondary IF = SPI slave device)

Table 6: Typical vales for external circuit components

Component	Value	Remarks
C1	100nF	Bypass capacitor AVDD to GND
C2	100nF	Bypass capacitor VDDIO to GNDIO
C3	1µF	Bypass capacitor VDDIO to GNDIO
C4	220nF	Regulator capacitor VREG

4.2 PCB Considerations for Backwards-Compatibility with Bosch Sensortec IMUs

The BHI385 has been designed to provide schematics and PCB layout compatibility with a number of Bosch Sensortec IMUs, like the BMI320, BMI260, BMI270, or BMI160. This allows the creation of applications with the ability to provide different sensor options with the same PCB design. Also, with some forehand considerations, it is possible to upgrade a PCB designed for an IMU to the BHI385 later.

The following items shall be considered when designing such a compatible PCB:

1. VDDIO voltage
   The VDDIO voltage has to be selected compliant to BHI385, which is nominal 1.8V, see Section 17.2 Table 129.
2. Application of Pad 9 (VREG / INT2)
   On BHI385 the pad 9 has a different function compared to IMUs. It has to be used for connecting an external decoupling capacitor for the on-chip voltage regulator of the Fuser2. In an IMU application this capacitor shall not be mounted. At the same time, in this case the pad 9 has the functionality of an interrupt output from the IMU and may be connected to the host processor via a 0R bridge. This bridge shall be not mounted in a BHI385 application, since the regulator voltage shall not be connected to a CMOS input.
3. It is recommended to add a 1µF external bypass capacitor on the VDDIO rail in case the BHI385 is used.
4. BHI385 has additional pads 15...20 inside the outer pad ring. These pads provide optional functionality and might not be needed for many applications. For a PCB that is compatible with an IMU, these pads can be left unconnected.

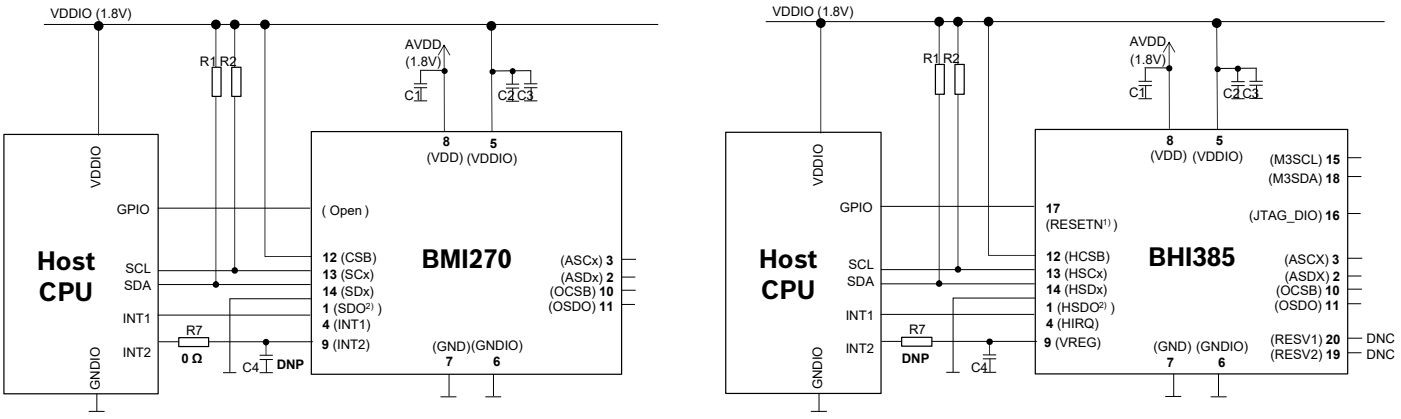
[1] Recommendation: Connect to VDDIO, if not needed.
[2] HSDO can be used for device address selection, if needed.

If an application uses these pads on the PCB, it is still possible to mount an IMU package, which would leave these pads disconnected.

5. Package height
   BHI385 has a slightly higher package height compared to IMUs. This has to be taken into account when designing the application housing, and also for the setup of the PCB assembling equipment.

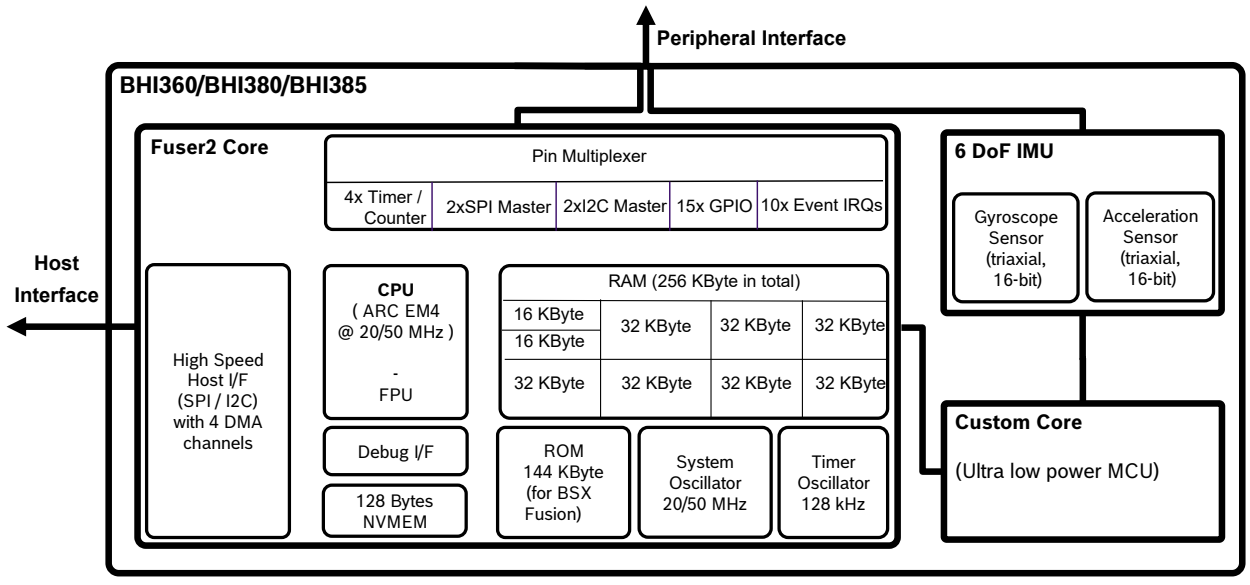
Figure 7shows an example of a connection diagram for a PCB allowing both BHI385 and compatible IMUs to be used.

Figure 7: BHI385 connection diagram for best compatibility with IMUs (Primary IF = I2C; Secondary IF = None)



4.3 Block Diagram

Figure 8: Block diagram of BHI385



4.4 Physical Primary Host Interface

The BHI385 provides a high-speed data interface and a single interrupt line (the Host Interrupt Signal HIRQ) as the main interface to the host processor.

The host interface can be operated in either I2C or SPI mode. The default protocol after power-up or reset is I2C. Pulling the HCSB (Host Chip Select) line to low at any point in time switches the host interface to SPI mode, where it remains until reset or a new power-on cycle.

In both interface modes, all data is treated MSbit first. The host interface provides access to the host register map of BHI385, which provides all necessary functionality to operate the device.

4.4.1 Host Interrupt

The interrupt line is implemented based on one of the device's GPIO pins (GPIO0) and therefore is as flexible in its configuration options (e.g. active high, active low, open-drain or push pull, level or pulse, with selectable drive-strength) as any other GPIO pin.

When using a default firmware image, the interrupt line[1)] is configured to be active high, push pull, level, low drive strength.

4.4.2 Operation in I2C Mode

In I2C mode, the host interface is implemented as an I2C slave interface as described in the I2C bus specification from NXP (see Section 20, Reference 1) and implements data transfer rate up to 3.4 Mbit/s in in high-speed mode.

The I2C bus consists of two wires, HSCX (Host Serial Clock) and HSDX (Host Serial Data). For connections to the host, both bus lines must be externally connected to a positive supply voltage (VDDIO) via pull-up resistors or current source.

A data transfer is always initiated from the host. The BHI385 can operate as either a transmitter or receiver only, if a valid device address has been received from the host. The valid device address can be selected by adjusting the state on the HSDO pin as shown in Table 7:

Table 7: I2C device address selection options

HSDO	I2C Device Address
HIGH	0x29
LOW	0x28

All I2C transactions start with the host sending a START bit followed by the slave device address (7-bit only) and the read-write indication bit, where a logical "0" (LOW) defines the transaction as a write. If the received slave device address matches the defined device address, the BHI385 is selected (i.e. ready for communication) and responds with an ACK (driving HSDX low). In case of a non-matching device address, the BHI385 is not selected and responds with a NAK (leaving HSDX high).

After the ACK, the host must provide the host interface register address next and so the very first host transaction must be a write operation. The MSbit of the register address is ignored. The BHI385 always responds with an ACK (even for reserved registers). If the host wishes to write into the selected internal register, it has to append the write data to the register address within the same transaction.

The host can read the selected internal register in one of two ways:

- in the combined format (i.e. issue a repeat-START bit, slave address and read indication). See Figure 10.
- in a separate transaction (i.e. issue a STOP bit and start a new read transaction with START bit, slave device address and read indication). See Figures 11 and 12.

During multiple writes, the BHI385 acknowledges all data bytes with an ACK. During reads, the host responds with an ACK to all data bytes except the last one, which is acknowledged with a NAK. Write and read transactions are illustrated in the figures below. All communication is MSbit first. The I2C host interface does not stretch the clock.

[1] It is also possible to operate the host interface in polling mode and re-use the interrupt line as GPIO pad.

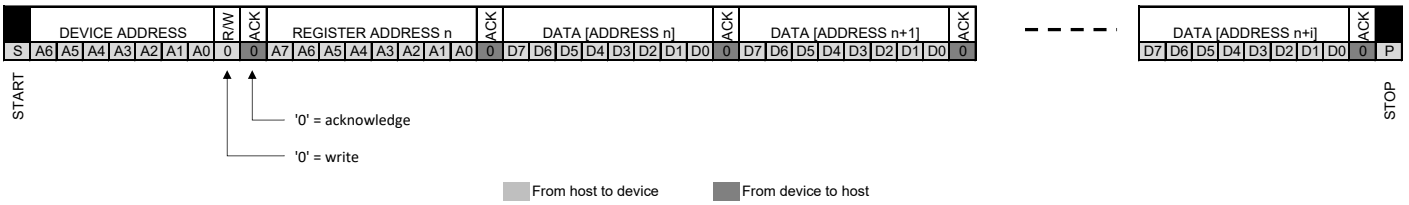


Figure 9: I2C write transaction

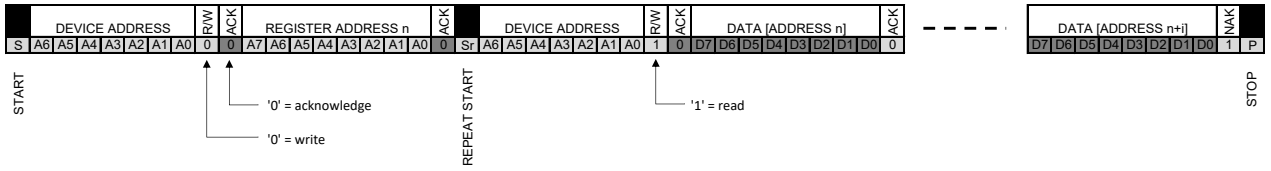


Figure 10: I2C read transaction - combined format (with repeat start)

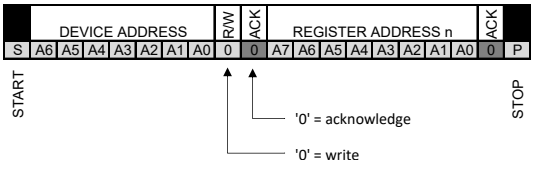


Figure 11: I2C read transaction - split format (1)

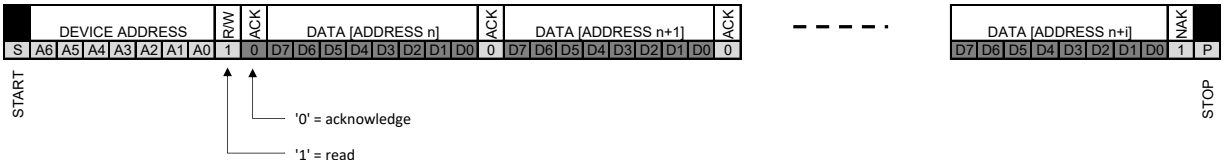


Figure 12: I2C read transaction - split format (2)

An I2C watchdog is available in the BHI385 to monitor the activity of the I2C protocol engine. It can be enabled and configured via the Host Control register in Section 11.1.12.

When the I2C watchdog is enabled, if a transaction is in progress but the I2C protocol engine does not carry out any transaction activity for a configured amount of time (either 1ms or 50ms), the transaction is considered to be stalled and the I2C protocol engine aborts the stalled transaction by assuming idle state. Conditions that may cause a stalled transaction include:

- The I2C master is reset during a transaction and the clock stops while the slave is transmitting a “0”.
- The slave is transmitting data; due to spurious clocks, it gets out-of-sync with the master and waits for an acknowledgement that never arrives.

4.4.3 Operation in SPI Mode

In SPI mode, the host interface is implemented as an SPI slave interface and implements data transfer rate up to 20 Mbit/s (in Long Run mode) or up to 50 Mbit/s (in Turbo mode).

By default, the SPI interface consists of four wires (4-wire SPI protocol), an active-low chip select HCSB (Host Chip Select), a clock line HSCX (Host Serial Clock), a data input line HSDX (Host Serial Data), and a data output line HSDO (Host Serial Data Output).

The SPI interface can be configured to operate also only with three wires (3-wire SPI protocol), where the HSDX line is used as bidirectional data line while the HSDO line is omitted. To enable this mode, the respective bit in the Host Control register needs to be set.

An SPI transaction starts with the host driving the HCSB to logical “0” (LOW). The host first sends the internal register address to be accessed, where the first bit (MSbit A7) indicates whether the access is intended to be a read or write access (with A7= “0” indicating a write and A7= “1” indicating a read) and the following 7 bits specify the actual address of the register.

In case of a write access the host has to follow the address byte with the data byte on the HSDX line in the same sequence.

In case of a read access, the BHI385 provides data with each additional clock pulse on HSCX either on the HSDO line (4-wire SPI) or on the HSDX (3-wire SPI) depending on the SPI mode setting in the Host Control register.

- If the sequence is interrupted by pulling HCSB to HIGH, the next SPI transaction starting with a new HCSB falling to LOW, will require a new register address transmission as described above, before any data can be read or written successfully.

Two CPOL/CPHA configurations are supported: “0/0” and “1/1”.

- With CPOL=CPHA=0 HSCX has to be LOW, when a sequence is initiated with a falling edge of HCSN.
- With CPOL=CPHA =1 HSCX has to be HIGH, when a sequence is initiated with a falling edge of HCSN.

In both supported configurations, data is presented on the falling edge of the clock and sampled on the rising edge.

The following three figures illustrate the protocol. The corresponding timing parameters are listed in Section 17.5.1.

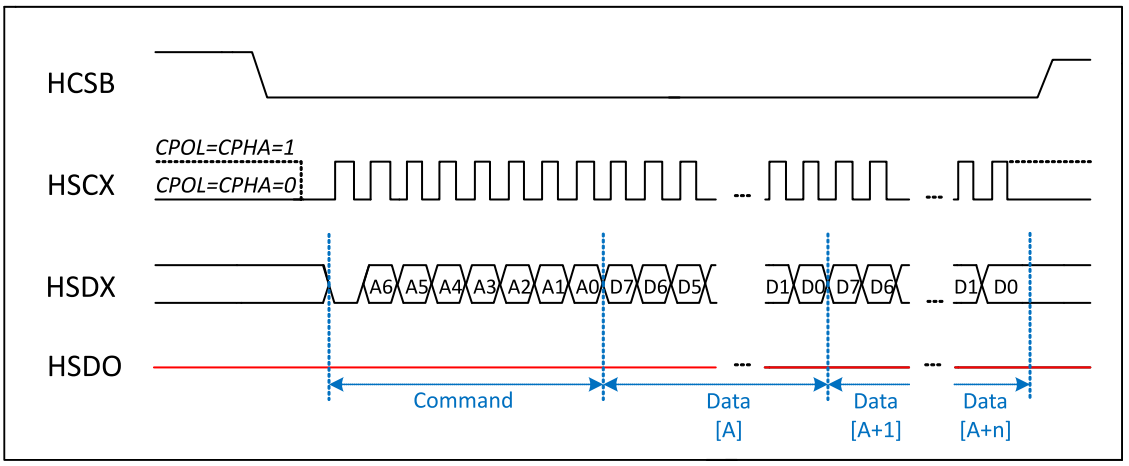


Figure 13: SPI write transaction in 4-wire and 3-wire mode

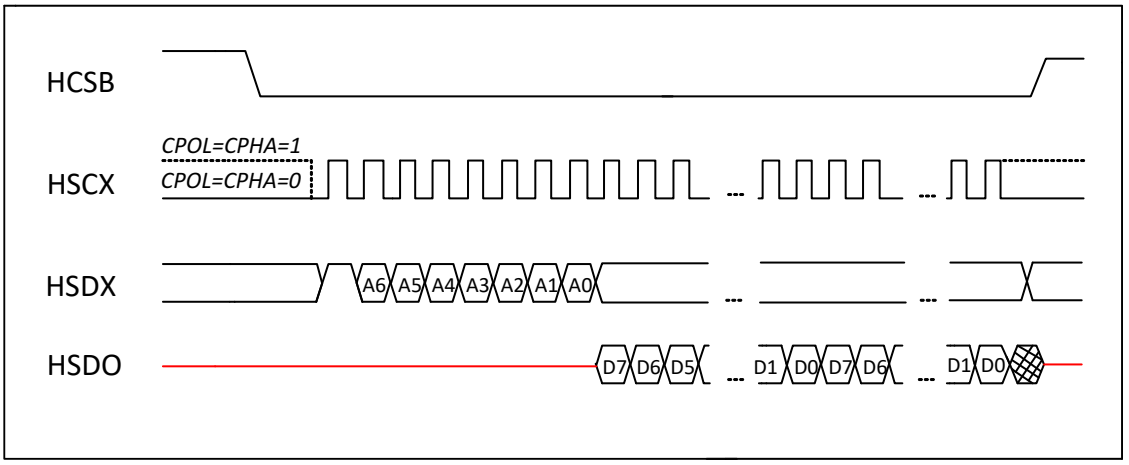


Figure 14: SPI read transaction in 4-wire mode

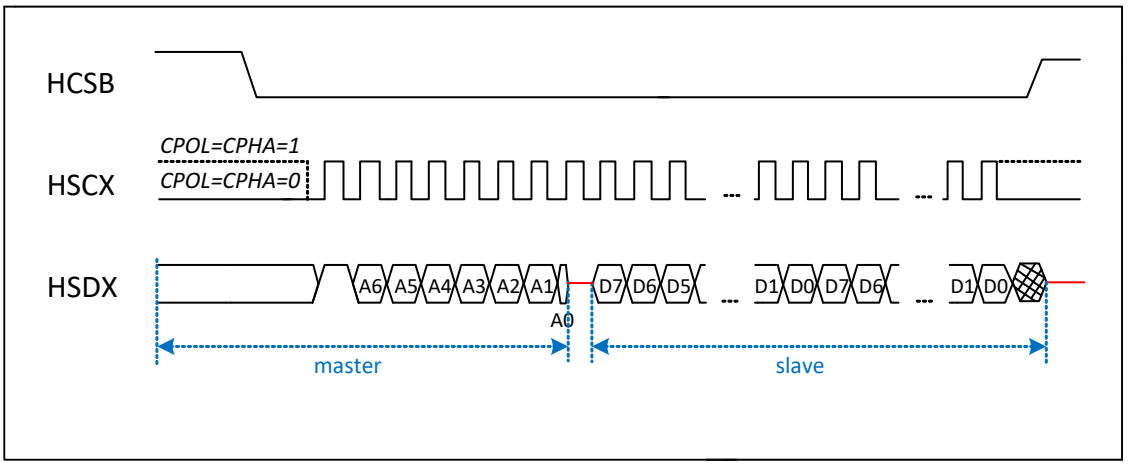


Figure 15: SPI read transaction in 3-wire mode

4.4.4 Burst Mode Operation

A data transmission sequence can cover the read/write transmission of multiple data bytes. In this burst mode operation, the transmission of multiple bytes is supported by two different features of the BHI385, as a burst mode operation on DMA-enabled registers or a burst mode operation on regular registers.

4.4.4.1 Burst mode operation on DMA enabled registers

The register addresses 0x00:0x03 are linked to four individual DMA engines (one for each address), which allow for efficient transfer of bigger data portions between the BHI385 and the host, by directly mapping these register addresses to the defined memory regions of the BHI385 internal closely coupled memories (ICCM and/or DCCM).

The DMA engine at address 0x00 is designed to enable write operations into the BHI385, while the engines at the other three addresses are designed to enable read operations from the BHI385 device.

The usage of the four channels is described below.

Table 8: Overview of DMA channels

Register Address	DMA Channel	Operation Mode	Function
0x00	0	Write	Send commands to the BHI385 with variable length (including firmware upload)
0x01	1	Read	FIFO for reading Wake-Up sensor data
0x02	2	Read	FIFO for reading Non-Wake-Up sensor data
0x03	3	Read	FIFO for reading Status and Debug information

Reading more bytes from a channel than available results in zeros being read. Writing more bytes than expected is considered as an overflow and discards the whole transmitted data within the session.

4.4.5 Burst mode operation on regular registers

Burst mode operations to all other registers are supported by auto-incrementing the register address so that consecutive registers can be read or written within one sequence. Read/write from/to reserved registers can cause unpredictable effects and shall be avoided.

4.5 Host Data Interface

4.5.1 Host Data Access

The BHI385 can be controlled entirely by the host through reading and writing its host interface (HIF) registers. This can be for example, booting the device, resetting the device, enabling/disabling virtual sensors, and changing parameters at runtime. These registers are summarized in Figure 16 and described in detail in Section 11.

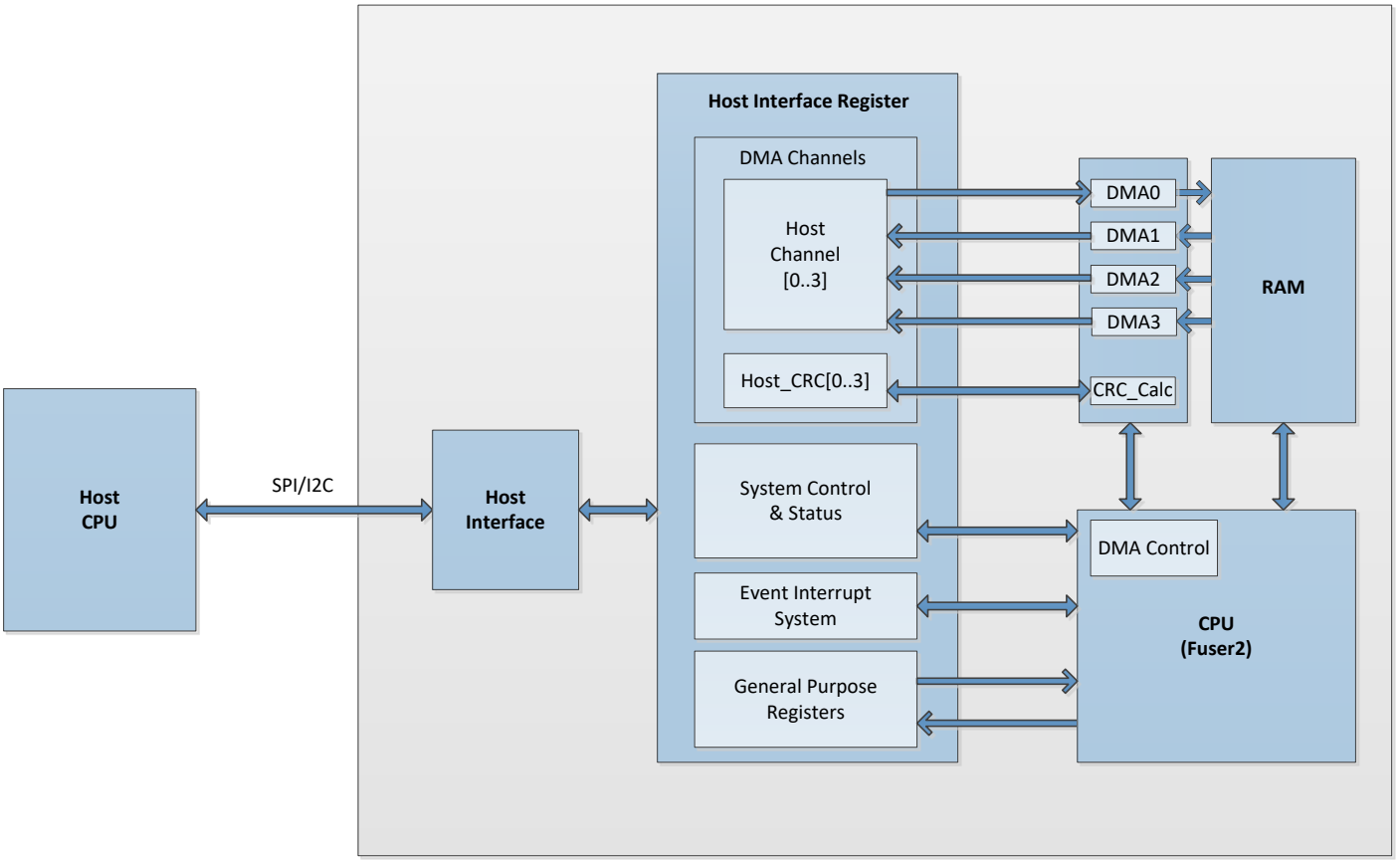


Figure 16: Overview of host data interface

Table 9: Basic overview of the BHI385 host interface registers

Section	Host Register Name	Address	Main Function
DMA Channels	Host_Channel[0]	0x00	Input for the command interface
	Host_Channel[1]	0x01	Output for wake-up virtual sensor data
	Host_Channel[2]	0x02	Output for non-wake-up virtual sensor data
	Host_Channel[3]	0x03	Output for the command interface & debug
	Host_CRC[0...3]	0x18 – 0x1B	Calculated CRC for last channel used
System Control & Status	Chip Control	0x050x06	Basic chip control (debug, turbo-mode)
	Host Interface Control		Host interface & DMA control
	Host Interrupt Control	0x07	Enable & disable interrupts
	Reset Request	0x14	Host controlled software reset
	Host Control	0x16	SPI & I2C mode settings
	Host Status	0x17	Host interface status information
	Fuser2_Product_ID	0x1C	Fuser2 product specific number
	Fuser2_Revision_ID	0x1D	Fuser2 revision specific number
	ROM Version[0...1]	0x1E – 0x1F	16-bit ROM image revision
	Kernel Version[0...1]	0x20 – 0x21	16-bit kernel image revision
	User Version[0...1]	0x22 – 0x23	16-bit user image revision
	Feature Status	0x24	Feature description
	Boot Status	0x25	Boot status description
	Chip ID	0x2B	Product specific number
	Interrupt Status	0x2D	Interrupt status info
	Internal Debug Registers	0x2E-0x31	Debug & Post mortem info

Section	Host Register Name	Address	Main Function
Event Interrupt System	Timestamp Event Request Host Interrupt Timestamp[0..4]	0x15 0x26-0x2A	Host controlled software event interrupt request Timestamp of last host interrupt (can be configured as software event interrupt response via Host Interface Control Register)
General Purpose Registers	Host_Gen_Purpose Proc_Gen_Purpose	0x08 – 0x13 0x32 – 0x3D	General-purpose Read/Write register General-purpose Read only register
Reserved	Reserved Registers Reserved Reserved	0x04, 0x2C 0x3E – 0xFF	Do not use

4.5.2 Host Command Protocol

When it comes to operating and configuring the BHI385, the main communication mechanism between the host and the BHI385 firmware is the Host Command Protocol, which is used to initially upload a firmware to the device or to configure and enable/disable virtual sensors.

The Host Command Protocol is realized by sending command packets to the Host Channel 0 - Command Input (0x00) and reading back the status packets out of the Host Channel 3 - Status and Debug FIFO Output (0x03), if any are available for that specific command.

There are several different commands for different purposes. They are summarized and explained in Table 32 in Section 12 Host Interface Commands. The overall structure of a command packet is always the same: 2-byte command + 2-byte length + (optionally) 4 bytes or more parameters or data. This is shown in the Table 10: Structure of a command packet. The structure of a status packet is shown in Table 11: Structure of a status packet.

The size of a command packet must be a multiple of 4 bytes. If the command length does not end on a 4- byte boundary, the command must be padded out to the 4-byte boundary with zeroes. These padded bytes must be included in the Length field of the command structure.

Table 10: Structure of a command packet

Field Name	Byte Offset	Description
Command ID	0x00-0x01	2-byte command, LSB first
Length	0x02-0x03	Command content's length in bytes, LSB first
Contents	0x04... Length + 0x03	Optional parameters or data

In response to certain commands, a status packet in the following format will be placed in the output channel 3 (Status and Debug FIFO):

Table 11: Structure of a status packet

Field Name	Byte Offset	Description
Status Code	0x00-0x01	2-byte status, LSB first
Length	0x02-0x03	Status content's length in bytes, LSB first
Contents	0x04... Length + 0x03	Optional parameters or data

A list of all possible status packets is given in Table 32 in Section 12 Host Interface Commands. The size of any status packet shall be a multiple of 4 bytes.

5 Device Configuration and Operation

5.1 Overview of the Event-Driven Software Framework and Virtual Sensor Stack

The Event-Driven Software Framework consists of different software layers that abstract typical functions in a smart sensor such as power management, sensor data acquisition, sensor data synchronization, and sensor data processing, and features a complete virtual sensor stack ready to use. The core components of the virtual sensor stack are virtual sensors. They are the main providers of data output of the BHI385 via the FIFOs.

A virtual sensor typically consumes data from one or more virtual or physical sensors and either sends the processed data to the FIFO or another virtual sensor. A physical sensor driver communicates with an external physical sensor connected to the BHI385 (or one of the physical sensors integrated inside the BHI385) to retrieve data and send it to a virtual sensor. There are different types of virtual sensors, depending on how the data is generated and written into the FIFO.

- Continuous: Data frames are written synchronously into the FIFO by the virtual sensor at the configured sample rate.
- On-Change: Data frames are written when there is a change in the value of the virtual sensor. The data frames have a maximum sample frequency as the one configured.
- One-Shot: Only one data frame is written in the FIFO following a trigger to enable the sensor. The fundamental parts of the Event-Driven Software Framework are explained in the following sections.

5.2 Using Virtual Sensors

Virtual sensors are part of the firmware image, and hence there is a correlation between the number of virtual sensors compiled into the firmware image and the size of this firmware, ensuring better control of the RAM memory. The remaining memory space then can be used as a FIFO for data batching.

Each virtual sensor (e.g. gravity sensor) may come in two types defined by two distinct Sensor IDs. These types are wake-up virtual sensor and non-wake-up virtual sensor and their data is placed in one of the two corresponding output FIFOs (i.e. wake-up FIFO and non-wake-up FIFO).

To configure a virtual sensor, the “Configure Sensor” host command needs to be used, see Section 12.2.7.

It takes three parameters to configure the behavior of the virtual sensor:

- Sensor ID
- Sample Rate
- Latency

The **Sensor ID** is a unique identifier that defines a specific virtual sensor. This Sensor ID is defined as an 8- bit unsigned integer value.

A list of all Virtual sensors and their corresponding sensor IDs can be found in Section 14.

The **sample rate** or output data rate (ODR) defines the desired frequency, at which the BHI385 should provide the Virtual sensor’s data. The Sample Rate is defined as a 32-bit float value in Hz.

The Virtual sensor is enabled by setting its sample rate parameter to a value higher than 0 Hz. Once enabled, the Event-Driven Framework will operate the virtual sensor by providing it with data and transferring the processed data to the corresponding FIFO. To disable a virtual sensor, the sample rate must be set to 0 Hz.

If the configured sample rate parameter is supported by the virtual sensor, this value will be selected as the actual sample rate. In the case that the configured sample rate is not supported, the actual sample rate will be selected by the Event-Driven Software Framework and will be in the range of 90%...210% of the requested data rate as explained in Section 6.6.

The **Latency** defines the maximum time that the BHI385 allows a virtual sensor event to remain in the FIFO, before the host is notified via the host interrupt.

The latency parameter is defined as 24-bit unsigned integer value in milliseconds.

With the latency set to 0 ms the host will instantly receive an interrupt after a new sample is stored in the FIFO. This behavior is similar to that of a Data Ready Interrupt.

The latency feature enables the batching of data for a period of time, during which the host may sleep, while the BHI385 continues collecting and processing sensor data, e.g. setting the latency parameter to its maximum value allows to keep the host into sleep mode for ~4.6 h. This “offloading” of the sensor processing or “batching” of sensor data, allows a significant reduction of the system power consumption in real world applications.

Whether a virtual sensor is available in the setup is determined by the presence (or absence) of the physical sensors required to implement these virtual sensors and whether the virtual sensor implementation has been integrated in the firmware configuration at the time of build.

It is possible to check at runtime which virtual sensors are supported by the current setup and firmware by means of the “*Virtual Sensors Present (0x011F)*” parameter, see Section 12.3.2.5.

Additionally, the “Virtual Sensor Information Parameters (0x0301 – 0x0395)” and “Virtual Sensor Configuration Parameters (0x0501 – 0x0595)” parameters give access to the details and capabilities as well as the current configuration of each virtual sensor. Please refer to Section 12.3.4 and Section 12.3.5 for details.

5.3 Using FIFOs and FIFO Events

The concept of FIFOs and Events is fundamental to the proper operation of the BHI385. All virtual sensor data and associated timing information are sent to a FIFO.

The data is structured in a specific package format and referred to as a Virtual Sensor Event. In this context Virtual Sensor Events are not only single occurrences like a double-tap, but also any virtual sensor data like an accelerometer data sample. In addition to Virtual Sensor Events, the FIFOs also contain other kind of data, as described in the following sections.

5.3.1 FIFOs

The BHI385 has three output FIFOs: the Wake-Up FIFO, the Non-Wake-Up FIFO and the Status and Debug FIFO.

The first two FIFOs contain “FIFO Events” which are mainly data from the enabled virtual sensors, together with certain associated FIFO Events such as timestamps.

By default, the Status and Debug FIFO contains responses to host commands. Alternatively, by setting the “Async Status Channel” bit in the Host Interface Control register, the output of additional debug and asynchronous status messages can be enabled like debug text strings.

The size of the Wake-Up and Non-Wake-Up FIFOs can be configured at build time, while the Status and Debug FIFO has a fixed size of 512 bytes.

Before the host can read data from any of the FIFOs, it needs to wait for the BHI385 to request a transfer by raising a host interrupt request.

The behavior of the host interrupt request can be configured by setting the virtual sensors ODR, the virtual sensors latency, the FIFO watermark settings and the “AP suspended” state of the “Host Interface Control” register.

The host can also initiate an immediate transfer without waiting for an interrupt. In this case, the “FIFO Flush” command with the appropriate parameter should be used, see Section 12.2.3 FIFO Flush (0x0009) for details.

A transfer request from the BHI385 to the host is indicated via the HIRQ pin (Host Interrupt Signal) and the Interrupt Status register.

By reading the *Interrupt Status (0x2D)* register, the host can find out which FIFO contains data. Next, the host shall read out all data related to the transfer request from the respective Host Output Channels [1...3]. The amount of data in each FIFO is given by the first two bytes read from the FIFO (16-bit value, LSB first). The data in the FIFO are structured in a specific format that allows for parsing and distinguishing the contained information even when read as a single byte stream (see Section 107 FIFO Data Formats).

5.3.2 FIFO Events

The term “FIFO Event” relates to any data packet that can be read from the Wake-Up and Non-Wake-Up FIFOs. There are several types of FIFO events:

- Virtual Sensor Events
- Timestamp Events
- Meta Events
- Debug Data Events

Any FIFO event starts with a 1-byte FIFO Event ID, followed by zero or more payload bytes. The number of payload bytes depends on the FIFO Event ID and is fixed per FIFO Event ID. In this way, a parser can separate different events from the FIFO byte streams. A complete list of FIFO Event IDs is given in Table 107: Overview of FIFO event IDs.

Virtual Sensor Events are the data output samples from any of the virtual sensors. They contain the sample data as described in Section 14.1 Format of Virtual Sensor Events. The big majority of FIFO Events are Virtual Sensor Events.

Timestamp Events declare a certain point of time, which is then valid for all following events, until a new timestamp event occurs. Different absolute and delta timestamp events exist, which helps to optimize the data load. They can be disabled for use cases which do not require timing information of events. See details in Section 14.2 Retrieving Timestamps of Virtual Sensor Events.

Meta Events are part of the Event-Driven Software Framework and are used by the BHI385 to notify the host about the occurrence of situations that might be of interest to the host but are not regular sensor data and they can be individually enabled or disabled. Examples for Meta Events are the Initialization of the FIFOs, confirmation for a change of ODR or updates of the calibration status of virtual sensors. They are placed in the different FIFOs by the Fuser2 and read by the host like any other FIFO Event. Together with the timestamps, the host can exactly reconstruct the time a FIFO Meta Event occurred. A complete list of all Meta Events are given in Section 14.3 Format of Meta Events. It also describes in which one of the FIFOs the different Meta Events are placed.

Debug Data Events are used to communicate any kind of debug information from the firmware to the host, for example, by using printf() calls in the firmware. See details in Section 14.4: Debug Data.

5.4 Using the Parameter Interface

The Parameter Interface is integrated as a subsystem of the generic command interface, i.e. reading or writing of parameters is handled by sending appropriate commands to the BHI385 following the command protocol described in Section 4.5.2 Host Command Protocol.

The intention of the Parameter Interface is to modify the configuration of the BHI385 during runtime. The parameter space of the BHI385 is very wide and covers several parameter pages to allow a versatile and flexible configuration of the device behavior.

Parameter related commands are grouped into the following sections:

- System Parameters
- BSX Algorithm Parameters
- Virtual Sensor Information Parameters
- Virtual Sensor Configuration Parameters
- Virtual Sensor Control Parameters
- Customer Defined Parameters

A complete list including descriptions of all available parameters is given in Section 12.3 Parameter Interface.

5.5 Current Consumption in Operation

The table below lists example current consumption of the BHI385 device for typical virtual sensors. The BSX fusion library can be configured to run in the default High Performance mode or in a Low Power mode optimized for power-constraint applications, e.g. wearables.

Table 12: Power consumption vs. operating mode

Virtual Sensor	Sensor Mode	ODR [Hz]	BHI385 current [µA typical][1]
Game Rotation Vector (Accelerometer and Gyroscope)	IMU Normal	100	784
	IMU Low power	25	441
Rotation Vector (Accelerometer, Gyroscope and Magnetometer)[2]	IMU Normal	100	840
Geomagnetic Rotation vector (Accelerometer and Magnetometer)[2]	Accel Normal	50	390
Wearable Activity Recognition	Accel Low Power	n.a.	43
No Motion	Accel Low Power	n.a.	32
Any Motion	Accel Low Power	n.a.	32
Wrist Wear wake-up	Accel Low Power	n.a.	38
Step Counter (wearable)	Accel Low Power	n.a.	45
Mutli-Tap detector	Accel Normal	n.a.	305

[1] Current consumption may vary depending on firmware version and sensor related configurations.
[2] Requires an external magnetometer. Power consumption of the magnetometer is not included.

6 Integrated Product Software

6.1 Event-Driven Software Framework and Virtual Sensor Stack

Figure 17 shows the structure of the Event-Driven Software Framework running inside the BHI385.

Using the BHI385's SDK, additional customer specific application software can be added. Specifically, new in-sensor data processing algorithms can be added as new custom virtual sensors. Also, if additional external sensors are connected over one of the secondary interfaces, the corresponding new physical sensor data can be made available to the BHI385 by integrating a custom driver.

6.2 Hardware Abstraction Layer

The hardware abstraction layer provides a low level API, containing support functions to access peripherals such as SPI and I2C master interfaces, Timer and Event subsystem. This is typically only used in a physical driver.

This layer is used by the Event-Driven Software Framework and therefore there is typically no need for the user to interact with it directly within the context of a Virtual Sensor Driver.

6.3 OPENRTOS Multithreading Real-Time Kernel

The integrated OPENRTOS v9.0.0 kernel is the commercial version of the well-established FreeRTOS operating system for embedded devices.

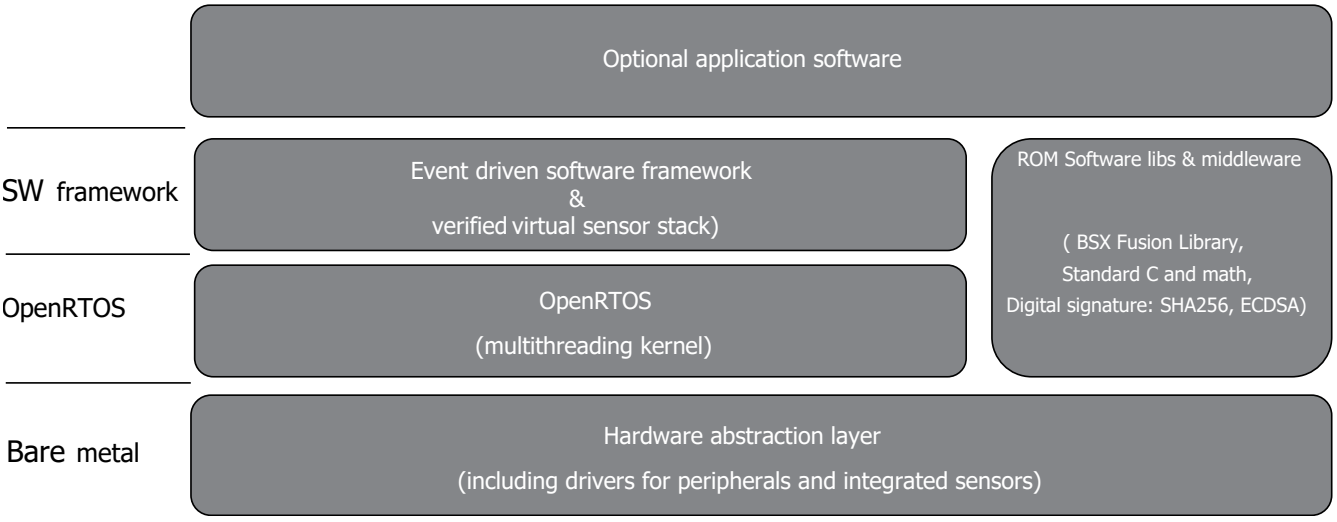
It is used by the Event-Driven Software Framework internally and therefore the user will not interact with the OPENRTOS functionality directly.

6.4 Virtual Sensor Stack

The core of the functionality delivered by the BHI385 as a smart sensor is the Virtual Sensor Stack. The BHI385 provides many algorithms that implement sensor data processing tasks that run inside the BHI385 MCU (Fuser2), providing pre-processed data to the Host processor to improve algorithm reliability and performance as well as improving system power consumption. These algorithms implemented as software modules in Fuser2 firmware are called virtual sensors. The set of all integrated algorithms in the BHI385 is the virtual sensor stack.

The BHI385 supports all virtual sensors as defined in the Android CDD and beyond. Originally based on the Android specification virtual sensors are independent of another. So each virtual sensor has its own sample rate, type, report

Figure 17: Structure of the BHI385 Event-Driven Software Framework



latency, and trigger mode.

Each virtual sensor software module may access a physical sensor via its sensor driver, or it may use the output data from other software modules (e.g. the BSX library) as an input for its data processing.

The virtual sensors generate virtual sensor events, which may be continuous (e.g. samples at a configured data rate) or single events (on-change, one-shot, or special, for example, when a step or significant motion has been detected). These virtual sensor events are represented as data packets of a specific virtual sensor data type and are placed into one of the output FIFOs of BHI385, either *Host Channel 1 - Wake-Up FIFO Output (0x01) and/or Host Channel 2 - Non-Wake-Up FIFO Output (0x02)*.

Each virtual sensor has a fixed ID and can be configured as a wake-up or non-wake-up sensor. For most virtual sensor types, both a wake-up and a non-wake-up ID exists, each can be also configured with an independent sample rate and report latency values. For the definition and details of all virtual sensors supported in the SDK, see Section 14 FIFO Data Types and Format. In addition to the existing Virtual sensors, additional features (algorithms) can be implemented and added as new virtual sensors by creating and uploading a new firmware to the BHI385 using the Software Development Kit (SDK). In this way, the functionality of the BHI385 can be extended while still using the benefits of the available Event-Driven Software Framework. For details and technical support please see the reference 2 Programmer's Manual in Section 20 References or contact our regional offices, distributors and sales representatives.

6.5 Other Available Software Modules

6.5.1 Standard C Library (libc) and Standard Math (libm)

As part of the ROM libraries the most used functions of the standard C library (libc) and standard math library (libm) are included in the ROM image.

6.5.2 Libraries for Digital signature

As part of the ROM libraries a SHA256 library and an ECDSA160 library are included in the ROM image. These SHA and ECDSA algorithms are used by the bootloader for firmware verification. They can also be used within a customized firmware image if needed.

6.6 BSX Sensor Fusion Library

The BSX sensor fusion library can be used to provide corrected and fused sensor data based on the raw sensor data received from the connected sensors. Virtual sensor data can be identical to the raw physical sensor data, offset calibrated data or certain preprocessed data via BSX algorithms like orientation, gesture recognition algorithms, step counter, etc. For a more comprehensive description of the virtual sensors integrated in the BHI385 see Table 107: Overview of FIFO event IDs.

The virtual sensor stack computes the sensor data with a maximum Output Data Rate (ODR) of up to 800 Hz and stores them in the FIFOs. The host CPU can access the FIFO over the host interface.

Based on the max ODR of 800 Hz, additional ODRs are available with a step size of 0.5x down to 1.56 Hz. Down to an ODR of 25 Hz, the sensor fusion software will accordingly reduce the data rate of the fusion computation and data sampled from the required physical sensors. Below an ODR of 25 Hz, the data provided is sub sampled from a 25 Hz signal to avoid loss in signal quality.

If the virtual sensor is set to a supported ODR[1], this will be the actual ODR used for processing and writing data in the appropriate FIFO. In case that the requested ODR is not supported, the actual output data rate will be in the range of 90%...210% of the requested data rate.

If multiple virtual sensors with different output data rates are requested by the host, the internal physical and virtual sensor data rates will be selected by the Event-Driven Software Framework such that all the requested output data rates can be fulfilled.

[1] Specific configurations can be created, supporting e.g. higher ODRs.

7 Software Development Kit (SDK)

A Software Development Kit (SDK) is available for users who want to create customized firmware, e.g. for running their own sensor data processing algorithms with low latency and at ultra-low power consumption on the BHI385.

In addition to the SDK, a C compiler toolchain for the ARC processor is required.

Supported toolchains are:

- Synopsys Metaware C Compiler for ARC
- GNU C Compiler for ARC

For details please see Section 20, Reference 2 and 6.

The SDK can be downloaded from the Bosch Sensortec website from the BHI385 product page. You can find the link in Section 20, Reference 9.

8 Device Initialization and Start-up

8.1 Power-on and Reset

The BHI385 has four possible reset sources as described in the following table:

Table 13: Available reset sources

Reset	Source	Suppressible	Reset is effective on
Power-On Reset	Analog Power Management Unit	No	All Fuser2 blocks
External Reset	External pad RESETN, active low	No	All Fuser2 blocks
Host Reset	Host write access to register Reset Request (0x14)	No	All Fuser2 blocks except 2-wire JTAG debug interface
Core Watchdog Reset	Core watchdog	Yes[1]	All Fuser2 blocks except 2-wire JTAG debug interface

Due to an integrated power-on reset controller with a brown-out detector, it is possible to reset the BHI385 without an external reset circuitry. The External Reset mechanism can be used if the host requires to trigger of a hard reset without relying on the operation of the host interface. For instance, a dedicated GPIO of the host can be connected to the RESETN pad. Optionally, a button with a pull-up circuitry can be used. If not used, the RESETN pad can be connected to VDDIO, however this is not mandatory, e.g. for applications that do not connect the inner pads.

After reset, the bootloader in ROM is executed.

8.2 BHI385 Initialization and Boot modes

Before the BHI385 can operate as a smart sensor, an initialization procedure has to be completed. During this initialization, the appropriate FW image will be loaded onto the BHI385. An external host processor will trigger and control the initialization procedure and will load the FW image to the BHI385 by writing the appropriate file over the primary host interface. The BHI385 requires a firmware image to operate. It is the responsibility of the integrated bootloader in the ROM to make this firmware available to the processor and to verify it.

8.2.1 BHI385 Initialization

An external host processor will trigger and control the initialization procedure, including loading the appropriate FW image to the BHI385. After initialization the boot loader will set the Host Interface Ready bit in the Boot Status register to 1, to indicate that it is ready to receive a firmware upload via the host interface. The sequence below should be followed to load the firmware and start the device:

1. Reset the Fuser2 with a Host Reset as specified in Table 13: Available reset sources.

2. Optional: Write the register *Host Control (0x16)* to select 3-wire SPI mode and/or I2C watchdog mode and time out.

3. Write the register *Host Interrupt Control (0x07)* to configure the type of host interrupt signaling desired (see Section 11.1.8 for details).

4. Optional: Write the *Chip Control (0x05)* register to set the CPU Turbo Disable bit as appropriate (0 = fastest firmware verification, highest power consumption, running at 50MHz; 1 = slower , lower power consumption, running at 20MHz). See Section 11.1.6 for details.

5. Poll the Boot Status register or wait for T_boot_bl_host time (see Table 136 in Section 17.5.1) until the Host Interface Ready bit is set.

6. Optional: Read the registers: *Fuser2 Identifier (0x1C), Fuser2 Revision (0x1D), and ROM Version (0x1E-0x1F)* to identify the revision of Fuser2 and select the proper Firmware image to load.

[1] Either “Core Watchdog disabled” or “JTAG enabled” suppresses the Core Watchdog reset.

7. Load the Firmware image to the BHI385 by issuing the Bootloader Command "*Upload to Program RAM (0x0002)*" to the register Host Channel 0 - Command Input (0x00) using a multiple of 4 bytes during each write transaction. Note: When uploading the firmware using multiple write transactions, only the first one shall include the *Upload to Program RAM (0x0002)* command. Subsequent transactions shall include firmware data only. The initial command includes the total overall length of the firmware image in words (bytes divided by 4), regardless of whether the entire firmware image will be written with one or more write transactions. Each Write to the Command Interface register shall contain one or more 4-byte packets.

8. Poll the register *Boot Status (0x25)* until the Firmware Verify Done bit is set or the Firmware Verify Error bit is set (in which case, the error shall be handled, for example, by raising an exception or retrying). Under typical conditions, the first poll of boot status will already indicate a completion of the verification, since the verification is started in parallel to loading.

9. Issue the Command "*Boot Program RAM (0x0003)*" to *Host Channel 0 - Command Input (0x00)* to start execution of the firmware. In the register Boot Status (0x25), the Host Interface Ready bit will clear at the start of the boot process.

10. Wait for firmware boot time, T_boot_fw_host, (see Table 136 in Section 17.5.1) or poll the Boot Status (0x25) register until the Host Interface Ready bit is set again. The "Initialized" Meta Events will be then inserted in the Wake-Up and Non-Wake-Up FIFOs, and the host interrupt pin will be asserted.

11. Clear the first interrupt by reading the content of the FIFO.

The BHI385 is now ready to receive the majority of host interface commands, e.g. enabling virtual sensors or setting parameters.

For a firmware upload, the following registers are relevant:

Table 14: Relevant registers for host boot mode

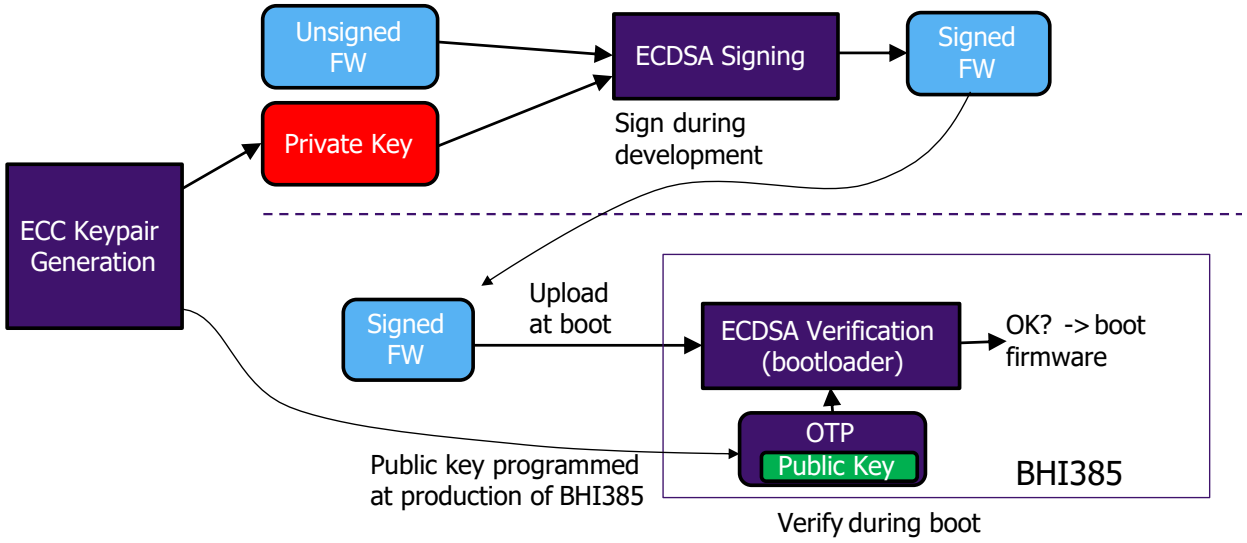
Register Name	Address	Register Value
Command Upload Stream (Host_Channel[0])	Host Channel 0 - Command Input (0x00)	Register to write firmware upload command and other boot commands
Chip Control	Chip Control (0x05)	1 – CPU Turbo Disable
Host Interrupt Control	Host Interrupt Control (0x07)	Configure type of host interrupt signalling desired
Reset Request	Reset Request (0x14)	1 – Reset MCU (Fuser2)
Host Control	Host Control (0x16)	Configure 3-wire SPI or I2C watchdog
Fuser2 Product Identifier	Fuser2 Identifier (0x1C)	0x89
Fuser2 Revision Identifier	Fuser2 Revision (0x1D)	0x02 or 0x03
ROM Version	ROM Version (0x1E-0x1F)	5166
Boot Status	Boot Status (0x25)	Boot status bits to help the host initialize the MCU at the correct times, and learn the results of various operations that may fail, such as image verification.

8.2.2 Secure Boot Mode

The BHI385 uses a secure boot mode concept that verifies the authenticity of a firmware image before it is executed. With this, it is possible to protect the firmware executed on the BHI385 against tampering and hacking. The secure boot concept applies to both host and stand-alone boot modes.

The BHI385 uses the Elliptic Curve Digital Signature Algorithm (ECDSA) for determining the authenticity of a firmware. After compilation, the firmware is signed by using a secret private key. The matching public key is stored in the one-time programmable memory of the BHI385 during production. The bootloader verifies the firmware image with the ECDSA algorithm using the public key stored in the device. The firmware will be executed if the verification is successful.

Figure 18: BHI385 secure boot concept (incl. signing and verification of FW images)



The firmware image consists of two sections, a kernel payload section and the user payload section. The content of the kernel payload section comprises of all algorithms, libraries and configuration items that cannot be modified. With the Software Development Kit (SDK), the kernel payload is provided as a signed binary and is authenticated to run on the BHI385. This concept allows for only authorized firmware images running on a particular device.

The user payload section allows for any kind of custom configuration and extension of the existing functionality. By default, the user payload section of the firmware is signed with a default key provided within the SDK.

If a customer specific key is required, Bosch Sensortec can provide a dedicated SDK for this, which uses a separate key pair (generated and owned by the customer) for signing the user mode section. Please contact your Bosch Sensortec representative for details.

8.3 Device Operation

Once the BHI385 is operational and the firmware has booted, all functionality including the virtual sensor suite is available. The BHI385 indicates its readiness by inserting an “Initialized” meta-event into each of its two FIFOs. It is recommended that the host waits for these events before attempting to query or configure the sensors or other features.

If an incorrect firmware is executed (for example, it is built for a different set of sensors), the FIFO will instead contain one or more Sensor Errors or Error meta-events.

After receiving the “Initialized” meta-events, the host is now free to issue commands, for example:

- Query which virtual sensors are present by reading the parameter “Virtual Sensors Present (0x011F)”
- Read the Sensor Information parameters
- Configure the algorithms by changing the Algorithm parameters
- Configure the sensors to start measuring using the Sensor Configuration parameters

The host may also use the *Meta Event Control (0x0101, 0x0102)* parameter to configure which meta-event appears in the FIFOs, e.g. FIFO Overflow, FIFO Watermark, etc. It can specify whether certain meta-events can cause an immediate host interrupt or are batched to be processed at a later point of time.

Finally, the host may configure the optional FIFO Watermark thresholds using the FIFO Control (0x0103) 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 prevent data in the FIFO from being overwritten. This is especially useful when the Application Processor is asleep.

8.4 Processor Execution Modes

As previously introduced, the ARC EM4 CPU of the BHI385 supports kernel and user execution mode.

- Kernel mode is reserved for low level drivers, the RTOS, the Event-Driven Software Framework, the BSX library, and other key internal functions, physical drivers, as well as interrupt handlers.
- User mode is used for user-provided enhancements (hooks) and virtual drivers. The MPU will block access to memory mapped registers with defined exceptions (e.g. universal timer and GPIO).

Elevation from user mode to kernel mode is implemented by a trap function. It will check that escalation to kernel mode is permitted from the calling function.

8.5 Power States and Run Levels

The various power states and run levels of BHI385 are defined independently for the Fuser2 MCU and the integrated IMU.

During operation they are also handled independently, specifically, the physical driver of the IMU in the Fuser2 firmware controls the power modes of the IMU. As an example, it is possible that the Fuser2 remains in sleep mode, while the IMU remains active, waiting for movement to wake-up the Fuser2.

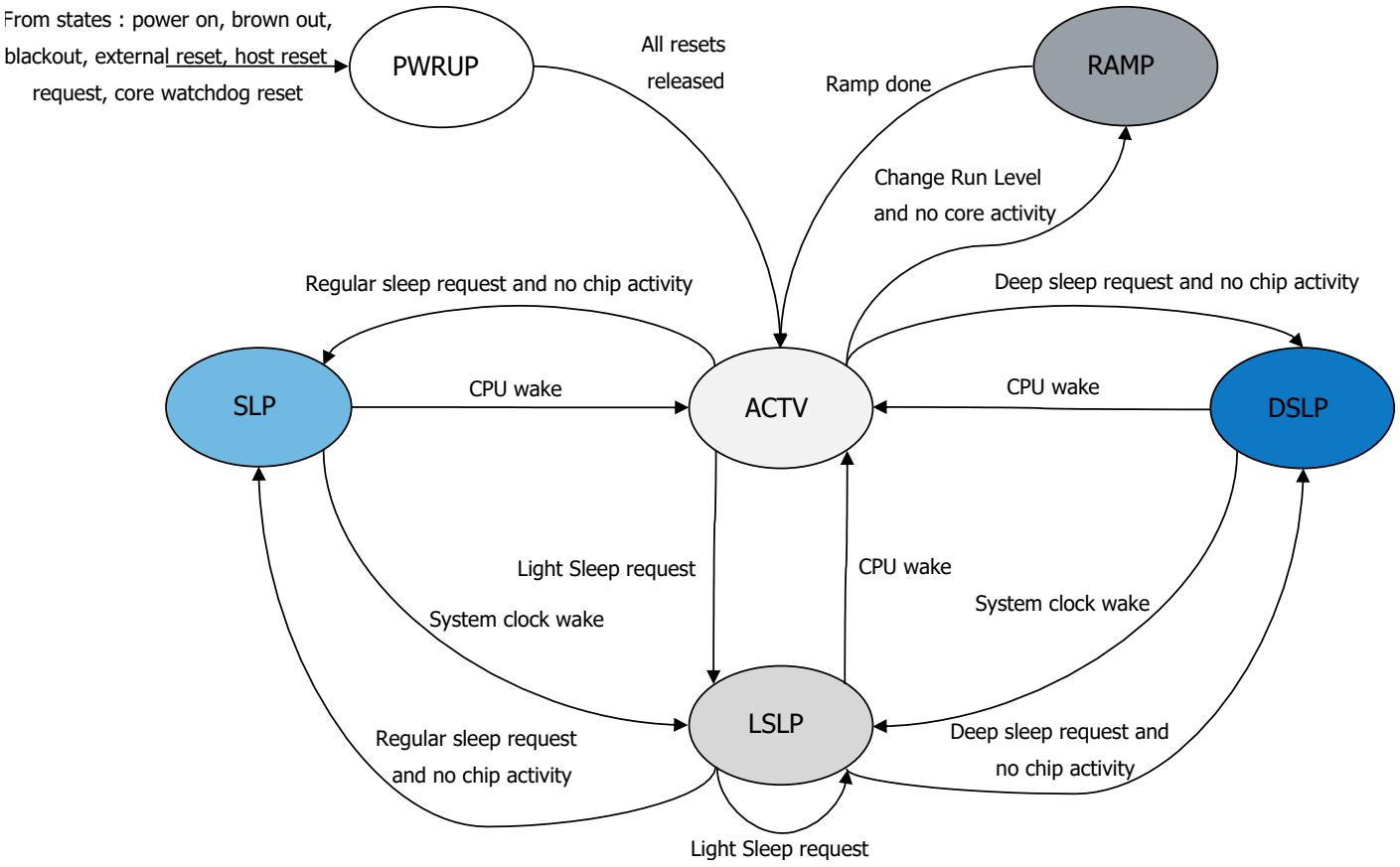
The Fuser2 operates in the following power states: Power-up, Active, Light Sleep, Regular Sleep, and Deep Sleep. In the states Active and Light Sleep, the Fuser2 can run at two clock speeds: Long Run mode (20 MHz core frequency) and Turbo mode (50 MHz core frequency). For the transition between the two run levels, there is a further intermittent Power State called Ramp.

The details of the power states and run levels are described in Table 15 shown below. The power states and run levels are controlled by the Event-Driven Software Framework internally and therefore there is no need for the user to interact with them directly.

Table 15: Available power states in Fuser2 MCU

Power state	Name	Timer Oscillator	System Oscillator at Run Level	Description
PWRUP	Power-Up	Off	Off (powering up)	The analog subsystem powers up with the timer oscillator disabled. All digital modules (CPU, memories, peripherals . . .) are in reset. When the power levels are sound and the system oscillator is stable, the device transitions to ACTV state, provided no other reset is asserted.
ACTV	Active	Unchanged	On	The processor is active. Upon entry from PWRUP or DSLP, the timer oscillator is disabled and activation is under firmware control. The firmware may issue a sleep instruction for transition into any of the sleep modes (LSLP, SLP, DSLP). In the absence of chip activity (e.g. peripherals inactive), a transition to SLP/DSLP is carried out immediately. Otherwise a transition to LSLP is carried out.
LSLP	Light Sleep	Unchanged	On	CPU enters sleep mode. System oscillator is enabled. A valid interrupt wakes the CPU and the device transitions to ACTV. Otherwise: If light sleep request is active, the device remains in this state. If a regular sleep or deep sleep request is active, the device remains in this state until all core activities cease. It then transitions to SLP / DSLP.
SLP	Regular Sleep	Unchanged	Off	The system oscillator is disabled. On a valid interrupt or host interface DMA request, the system oscillator is re-enabled. When the system oscillator is stable, the device transitions to ACTV if a valid interrupt is active. Otherwise, the system transitions to LSLP.
DSLP	Deep Sleep	Off	Off	Same as SLP. Additionally, the timer oscillator is disabled, allowing a lower current consumption.
RAMP	Ramping	Unchanged	Off (transitioning)	This state is entered whenever the run level is switched from Long Run to Turbo or vice versa. The analog subsystem adjusts the regulator voltage and system oscillator to the requested target. While the system oscillator is changing, no system clock is provided to the digital modules. The timer oscillator clock remains unaffected. When voltage and system oscillator are stable, the device transitions back to ACTV.

Figure 19: Transitions of power states



8.6 Debug and Post-Mortem Support

The BHI385 includes several features for debugging and post-mortem support:

- A 2-wire (compact) cJTAG (IEEE1149.7) interface debug port for the embedded ARC EM4 CPU.
- A debug printf() function for textual output through the data FIFOs
- Several dedicated and general-purpose host registers for monitoring the status of the system
- A post-mortem analysis capability, filling dedicated data structures in RAM in case of a watchdog reset or a critical exception, for the purpose of analysis after reset.

The details of these features are described in the Programmers Manual, Reference 2.

The JTAG interface is enabled or disabled during boot, depending on the configuration setting in the firmware. If the JTAG is enabled, it cannot be disabled anymore at runtime (without issuing a reset), to avoid the debug capability being disabled by the firmware accidentally.

The pads of the JTAG interface (JTAG_CLK, JTAG_DIO) can be used as general-purpose IOs in case the debugging is not enabled.

9 Pad and Interface Configuration

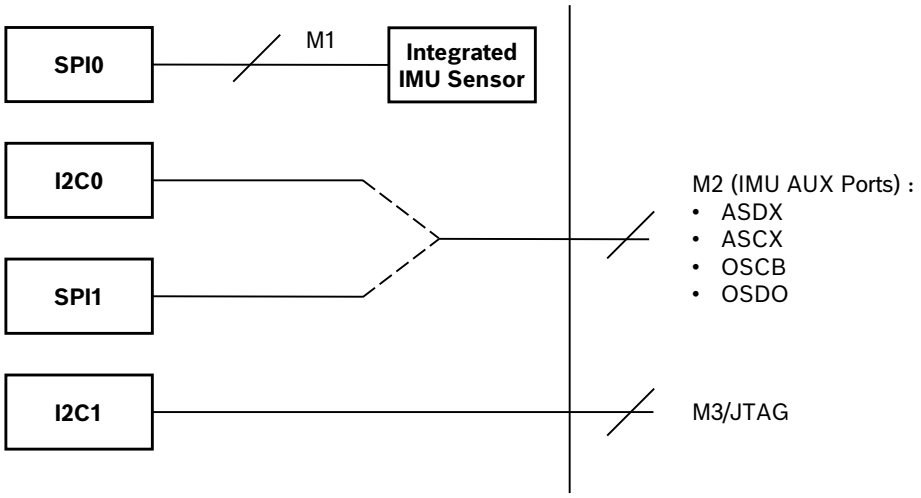
9.1 Configuration of Master Interfaces

The BHI385 has 3 secondary master interface ports, named M1 to M3, see Figure 20. While M1 is dedicated for communication with the integrated IMU sensor, M2 and M3 are available for connection of additional external sensors. Their external connections on the package are partly shared with other functionality, i.e. the functionality of the external connections defined by software configuration.

Inside the BHI385 two SPI (SPI0, SPI1) and two I2C (I2C0, I2C1) master interfaces controllers are implemented, which can be mapped in various ways to these 3 master interface ports.

The possible configurations of the master interface ports M1, M2 and M3 (mapping combinations to the internal interface controllers) are described in the diagram below. M1 is always operated in SPI mode and not available on external pads, but exclusively used to control the integrated IMU. M2 can be configures as SPI or I2C interface. M3 is always operated in I2C mode, The Fuser2 JTAG and M3 interfaces share one pad in common (JTAG_CLK, M3SCL). It can be decided per application that either JTAG or M3 is used, but not both at the same time.

Figure 20: Overview configuration options of master interface ports



The master interface connection routing is configured by a setting in the board configuration file which is part of SDK (see Reference 6).

9.2 General-Purpose Inputs/Outputs

Besides their functional purpose (e.g. host or master interface), almost all pins of the BHI385 can also be configured as General Purpose Inputs/Output (GPIO). The following characteristics of a GPIO pin can be configured:

- Input or output
- Drive strength as output, see Table 130 in Section 17.3: Electrical characteristics for low and high driver strength characteristics
- Tri-state
- Pull-up, pull-down (all GPIOs have pull-up functionality, except the GPIO24, which is a pull-down), see Table 130: Electrical characteristics in Section 17.3 for resistor values Before the firmware is loaded, the state of the GPIOs is according to the column “Reset value” in Table 1: Pin-out and pin connections in Section 3.1 Pad Description, i.e. most pins are configured as inputs with pull-up enabled.

The settings in the board configuration file define:

- The default configuration of all GPIOs after the firmware has started execution
- Which of the GPIOs is used as the SPI chip select or the interrupt line for the attached physical sensor

These settings are active after a firmware has been loaded. If a GPIO is not used as SPI chip select or interrupt input by the Event-Driven Software Framework, it can be controlled by low-level API functions provided by the Hardware Abstraction Layer of Software Development Kit (see Reference 6).

Some GPIOs of the Fuser2 do not have an external connection on the package of the BHI385, due to the available pads of the BHI385. These GPIOs shall be configured as pull-up, to avoid floating input gates which can cause high current consumption.

10 Event and Interrupt Configuration

The BHI385 provides various sources, which can generate Fuser2 interrupts. In this section, the Fuser2 interrupt mechanisms are described.

The interrupt handlers are managed by the Event-Driven Software Framework and therefore there is typically no need for the user to interact with them directly.

However, for cases where the user needs to extend the system, e.g. by connecting an additional external device (e.g. a sensor) or a signal (e.g. a camera shutter interrupt for time synchronization), additional interrupt handlers need to be installed.

The SDK provides a method for this in form of the physical sensor driver, which integrates such extensions into the overall framework in a smooth way.

For details, please see Section 20, Reference 6 or contact our regional offices, distributors and sales representatives. Table 16 provides an overview of available interrupt sources, where interrupt input describes the signal fed into the interrupt controller and interrupt output describes the mapped output of the interrupt controller.

Table 16: MCU Interrupts (Sources and Mapping)

Block	Interrupt Input	Interrupt Output	Map	Event Int	Description
Computational Core	Timer: internal	--	irq16	--	CPU internal Timer0 is triggered
	Watchdog: internal	--	irq18	--	Watchdog timer is triggered
	The DMA interrupts bypass the interrupt controller and connect directly to the processor interrupt inputs.	dma0_done_int	irq17	--	DMA transfer is completed
		dma1_done_int	irq19	--	
		dma2_done_int	irq20	--	
		dma3_done_int	irq21	--	
		dma0_err_int dma1_err_int dma2_err_int dma3_err_int	irq34	-- -- -- --	Error occurs during DMA transfer
Host Interface	hif_comm_irq	hif_comm_int	irq22	no	Host ends a communication thread (I2C mode: Stop bit received; SPI mode: HSCB de-asserted)
	host_ev_irq	host_ev_int	irq30	yes	Host controls software interrupt request generation. It is triggered when host writes “1” to bit 0 in the Timestamp Event Request register
	hif_read_irq[0:15]	hif_read_int	irq35	no	Host read request on one of the following registers: Proc_GenPurpose3_B0 Proc_GenPurpose3_B1 Proc_GenPurpose3_B2 Proc_GenPurpose3_B3 Proc_GenPurpose4_B0 ... Proc_GenPurpose6_B3

Block	Interrupt Input	Interrupt Output	Map	Event Int	Description
	hif_write_irq[0:15]	hif_write_int	irq36	no	Host write request on one of the following registers: Host_GenPurpose0_B0 Host_GenPurpose0_B1 Host_GenPurpose0_B2 Host_GenPurpose0_B3 Host_GenPurpose1_B0 . . . Host_GenPurpose3_B3
	hif_buf0_oflw_irq	hif_buf_err_int	irq32	no	DMA channel 0 buffer overflow
	hif_buf1_uflw_irq			no	DMA channel 1 buffer underflow
	hif_buf2_uflw_irq			no	DMA channel 2 buffer underflow
	hif_buf3_uflw_irq			no	DMA channel 3 buffer underflow
SPI Master Iface 0	spim0_irq	spim0_int	irq23	no	Transaction on master interface is completed
SPI Master Iface 1	spim1_irq	spim1_int	irq24	no	
I2C Master Iface 0	i2cm0_irq	I2cm0_int	irq25	no	
I2C Master Iface 1	i2cm1_irq	i2cm1_int	irq26	no	
Real Time Counter	rtc_irq	rtc_int	irq27	no	RTC counter overflow
Interval Timer	itmr_irq	itmr_int	irq28	no	Interval timer reaches the programmed limit
Universal Timer	utmr_irq	utmr_int	irq29	no	Depending on programmed setup of the universal timer one of the conditions has occurred: ▪ Universal timer reaches programmed limit ▪ Compare function triggers ▪ Input capture event occurs
PAD	gpio_ev_irq0	gpio_ev_int	irq31	yes	Event Channel triggers according to programmed setup
	gpio_ev_irq1			yes	
	gpio_ev_irq2			yes	
	gpio_ev_irq3			yes	
	gpio_ev_irq4			yes	
	gpio_ev_irq5			yes	
	gpio_ev_irq6			yes	
	gpio_ev_irq7			yes	
	gpio_ev_irq8			yes	
	gpio_ev_irq9			yes	
	gpio_ev_irq10			yes	
	gpio_ev_irq11			yes	
SW	sw_irq[0:3]	sw_int[0]	irq37	--	Software interrupt triggered
		sw_int[1]	irq38	--	
		sw_int[2]	irq39	--	
		sw_int[3]	irq40	--	

11 Host Interface Register Map

This is a detailed description of the host interface register map introduced in Section 4.5 Host Data Interface.

All interaction between the host and the BHI385 is through the register map, as summarized below. Each host register can be written from either the host or the Fuser2 side:

- Registers declared as “read-write” can be read and updated by the host.
- Registers declared as “read-only” can be read by the host, but can only be updated by the BHI385 MCU (Fuser2).
- Registers declared as “write-only” can be updated by the host. The register map is valid for both SPI and I2C host interface protocol.
- All “Reserved” registers shall not be written or read. Reserved bits in registers shall be written as 0.

Table 17: Host interface register map

Register Address	Register Name	Description	Host Access
0x00	Host Channel 0 - Command Input	The host can write command packets to this DMA Channel (input)	write-only
0x01	Host Channel 1 - Wake-Up FIFO Output	The host can read the virtual sensor data generated and stored in the wake-up FIFO through this DMA Channel (output)	read-only
0x02	Host Channel 2 - Non-Wake-Up FIFO Output	The host can read the virtual sensor data generated and stored in the Non-wake-up FIFO through this DMA Channel (output)	read-only
0x03	Host Channel 3 - Status and Debug FIFO Output	The host can read Status and Debug packets from FIFO DMA Channel (output)	read-only
0x04	Reserved	Do not use	-
0x05	Chip Control	Control fundamental behavior of Fuser2, like the firmware upload speed or clearing the error and debug registers	read-write
0x06	Host Interface Control	Control various actions regarding the host interface	read-write
0x07	Host Interrupt Control	Control characteristics of host interrupt, mask interrupt sources	read-write
0x08-0x13	WGP1 – WGP3	General-purpose registers for communication from the host to the Fuser2. While the host can read/write these registers, the BHI385 ARC CPU has read access only.	read-write
0x14	Reset Request	Host-controlled reset of Fuser2	read-write
0x15	Timestamp Event Request	The host can trigger an event interrupt (with timestamp response)	read-write
0x16	Host Control	Control the SPI mode and the I2C watchdog behavior of the BHI385	read-write
0x17	Host Status	Check the power status of the chip, the selected host protocol, and DMA channels	read-only
0x18-0x1B	Host Channel CRC (32-bits)	Get CRC of the last used DMA channel	read-only
0x1C	Fuser2 Identifier	This register identifies the Fuser2. It reads 0x89 for	read-only

Register Address	Register Name	Description	Host Access
0x1D	Fuser2 Revision	This register identifies the hardware revision of the Fuser2. For BHI385 it reads 0x02 after reset and before loading a firmware image, and 0x03 after loading the firmware image	read-only
0x1E-0x1F	ROM Version	ROM Image revision	read-only
0x20-0x21	Kernel Version	Firmware Kernel image revision	read-only
0x22-0x23	User Version	Firmware User image revision	read-only
0x24	Feature Status	Get status of various firmware-related features	read-only
0x25	Boot Status	Get status of boot loader	read-only
0x26-0x2A	Host Interrupt Timestamp (40 bits)	Get timestamp of last host interrupt or Timestamp Event Request	read-only
0x2B	Chip ID	Product Identifier	read-only
0x2C	Reserved	do not use	-
0x2D	Interrupt Status	Get current status of interrupt sources	read-only
0x2E	Error Value	Firmware-related error code	read-only
0x2F	Error Aux	Auxiliary information for firmware related error	read-only
0x30	Debug Value	Firmware-related debug value	read-only
0x31	Debug State	Firmware-related debug state	read-only
0x32-0x3D	RGP5 – RGP7	General-purpose registers for communication from the Fuser2 to the host: while the BHI385 ARC processor can read/write these registers, the host has read access only.	read-only
0x3E-0x7F	Reserved	do not use	-

11.1 Register Description

11.1.1 Host Channel 0 - Command Input (0x00)

The host should write one or more command packets with different size to this channel. This register address allows write-only access. The overall structure of a command packet is always the same: 2-byte command + 2-byte length + (optionally) 4 bytes or more parameters or data. This is shown in the Table 10: Structure of a command packet.

The host commands and their responses are described in Section 4.5.2 Host Command Protocol.

A list of available commands is described in Section 12 Host Interface Commands.

11.1.2 Host Channel 1 - Wake-Up FIFO Output (0x01)

When the host receives a host interrupt from BHI385, it should read the first two bytes of this channel to determine the total transfer size, then read the remaining data in one or more transfers until the total number have been read. If there is no data available for the wake-up FIFO, the first two bytes will read 0. If there is data available, this data will be the virtual sensor data generated and stored in the wake-up FIFO since the last wake-up FIFO read access (or wake-up FIFO flush operation). This register address allows read-only access.

11.1.3 Host Channel 2 - Non-Wake-Up FIFO Output (0x02)

When the host receives a host interrupt from the BHI385, it should read the first two bytes of this channel to determine the total transfer size, and then read the remaining data in one or more transfers until the total number is read. If there

is no data available for the non-wake-up FIFO, the first two bytes will read 0. If there is data available, this data will be the virtual sensor data generated and stored in the non-wake-up FIFO since the last non-wake-up FIFO read access (or non-wake-up FIFO flush operation). This register address allows read-only access.

11.1.4 Host Channel 3 - Status and Debug FIFO Output (0x03)

When the host receives a host interrupt from the BHI385, it should read the first two bytes of this channel to determine the total transfer size, and then read the remaining data in one or more transfers until the total number is read. If there is no status or debug data available, the first two bytes will read 0.

The general structure of host commands and their responses are described in Section 4.5.2 Host Command Protocol. A list of available commands is described in Section 12 Host Interface Commands. This Status and Debug FIFO has two different operational modes, Synchronous Mode and Asynchronous Mode. For more details see the description of the bit 7 in register Host Interface Control (0x06) in Section 11.1.7. See also the Sections 13.2.1 Synchronous Mode and 13.2.2 Asynchronous Mode. This register address allows read-only access.

11.1.5 Reserved (0x04)

Do not use this register.

11.1.6 Chip Control (0x05)

This register provides bits that control fundamental behavior of the chip, e.g. the speed during firmware image upload or clearing the error and debug registers. This register address allows read-write access.

Table 18: Chip Control Register (0x05)

Bit	Register Value	Description
0	CPU Turbo Disable	When written with a 1, the Fuser2 is not allowed to run at its maximum speed during firmware image upload and signature verification. By default, if not modified by the host, the bootloader will attempt to run at the maximum CPU speed and the current drawn by the BHI385 may peak up to ~3 mA during firmware upload. After booting the firmware, the run level is defined by the setting compiled into the firmware.
1	Clear Error Regs	When written with a 1, the error and debug registers (address 0x2E-0x31) are cleared. See Sections 11.1.25 and 11.1.26.
2-7		Reserved, write as 0

11.1.7 Host Interface Control (0x06)

This register provides bits that control fundamental behavior of the chip and allows the host to rapidly request specific actions that affect the state of the Fuser2 host interface. This include aborting the transfer of one of the FIFO Channels (0 to 3), informing of AP suspend mode to the BHI385, controlling what is written to the Host Interrupt Timestamp register and controlling the operating mode of the Status and Debug FIFO. This register address allows read-write access.

[1] The host interrupt signal and host interrupt asserted bit in the Interrupt Status register remain asserted and set until all FIFOs with pending transfers have been dealt with – either by aborting them using this register, or by reading their pending data out.
[2] The Abort Transfer bits do not auto-clear. It is up to the host to set these four bits correctly every time it writes this register. It should clear these bits no sooner than 2 ms after asserting them.
[3] Aborting a transfer causes data loss, if the current transaction is < 32 bytes, or if there is only 32 bytes left in the current FIFO block. In those cases those 32 bytes will be lost.

Table 19: Host Interface Control Register (0x06)

Bit	Identifier	Description
0	Abort Transfer on Channel 0	Abort transfer indicates that the host does not intend to complete a transfer with a channel; all pending data in the 32 bytes channel buffer for that channel is cleared, as well as any partial sensor sample that remains, and the interrupt pending bit for that channel in the Interrupt Status register is de-asserted. Since the data is not discarded, the BHI385 will soon request another transfer. Abort Transfer simply stops the transfer of the specified channel for this data transfer session. To discard data entirely, use the FIFO Discard command[1) 2) 3)].
1	Abort Transfer on Channel 1	
2	Abort Transfer on Channel 2	
3	Abort Transfer on Channel 3	
4	AP is Suspended	The AP can inform about its power status to the BHI385 just before going into a suspend (sleep) power mode or just after going to awake (normal) power mode. This affects whether the Fuser2 issue a host interrupt: ▪ When 0 (AP signals that it is awake mode): any wake-up or non-wake-up virtual sensor event may trigger a host interrupt if so configured. ▪ When 1(AP signals that it is in suspend mode): Only wake-up virtual sensor events may trigger a host interrupt and wake up the host.
5	Reserved	Reserved, write as 0
6	Timestamp_Event _Request (via Host Interrupt Timestamp registers)	Controls which timestamp is written to the Host Interrupt Timestamp register: ▪ When 0: Timestamp of the last Host Interrupt generated by the BHI385 ▪ When 1: The timestamp at the time of the last Host Timestamp Request register write access. In this case, the Host Event Request Timestamp will not be output as a status packet. Reset value is 0. Both timestamps are always available via the Timestamps parameter (see Section 12.3.2.4).
7	Async Status Channel	This bit controls the operating mode of the Status and Debug FIFO (Output Channel 3): ▪ When 0: Synchronous mode, see Section 13.2.1. ▪ When 1: Asynchronous mode, see Section 13.2.2.

11.1.8 Host Interrupt Control (0x07)

This register allows the host to configure the driving mode for the host interrupt (interrupt generated by the BHI385 to signal an interrupt event to the host), as well as to specify for which reason it would like to be interrupted. This register provides bits that control fundamental behavior of the chip. This register address allows read-write access.

Upon boot, i.e. before the main firmware initialization is complete, the BHI385 will not assert the host interrupt, unless the reset was caused by the watch dog timer or fatal error; in that case, this register will contain the previous configuration information, so the bootloader will reuse the value of this register to determine how to assert the interrupt.

The bootloader does not issue any interrupt, since at this stage the device does not know the interrupt configuration required by the host. By default, the Host Interrupt Signal HIRQ (Pad 10) is used within the SDK as an interrupt output,

a firmware may use any other GPIO, or no interrupt at all. Hence the interrupt can be initialized only after loading the firmware, and then it is initialized according to the value of this register.

After the host has issued the Boot Program RAM (0x0003) command, the host must poll the Boot Status (0x25) register until the Host Interface Ready bit is asserted (Host is ready). At any time prior to or after this, the host may write the Host Interrupt Control (0x07) register to configure the operation mode of host interrupt signal (active high or low, level or pulse, and push-pull or open drain). Once configured by the host, subsequent host interrupts will be signaled this operation mode. The default value of this register after reset is 0. If the value of 0 is the desired configuration (active high, level, push-pull), the host does not need to write this register. Once the main firmware initialization is complete, it will assert the host interrupt using this mode.

Table 20: Host Interrupt Control Register (0x07)

Bit	Identifier	Description	Value
0	Wake-up FIFO Interrupt Mask	Setting this bit disable host interrupts due to the wake-up FIFO (Host_Channel_1), while clearing this bit (the default) allows it to be asserted whenever conditions warrant	0: Interrupt is active 1: Interrupt is masked
1	Non-Wake-up FIFO Interrupt Mask	Setting this bit disable host interrupts due to the non-wake-up FIFO (Host_Channel_2), while clearing this bit (the default) allows it to be asserted whenever conditions warrant	0: Interrupt is active 1: Interrupt is masked
2	Status Available Interrupt Mask	Disable host interrupts due to available output from synchronous Status Channel	0: Interrupt is active 1: Interrupt is masked
3	Debug Available Interrupt Mask	Disable host interrupts due to available output from the asynchronous Status Channel	0: Interrupt is active 1: Interrupt is masked
4	Fault Occurred Interrupt Mask	Disable host interrupts due to various faults.	0: Interrupt is active 1: Interrupt is masked
5	Active Low Interrupt	Level of the host interrupt:	0: Active high 1: Active low
6	Edge Host Interrupt	Host interrupt to drive a level- or edge sensitive host interrupt. 0 = level (e.g., asserted until FIFO empty), 1 = edge (pulse for 10 µs)	0: Level output 1: Pulse output
7	Open Drain Interrupt	0 = push-pull, 1 = open drain (normally open drain is used with active low)	0: Push-pull 1: Open-drain

11.1.9 WGP1 – WGP3 - General Purpose Host Writeable (0x08-0x13)

These registers are available for customer use in custom drivers or extensions. This range of register addresses allows read-write access. While the host can read/write these registers, the BHI385 ARC CPU has read access only.

11.1.10 Reset Request (0x14)

register allows the host to reset the Fuser2 with a register write access. This register address allows read-write access.

Table 21: Reset Request Register (0x14)

Bit Number	Field	Description	Value
0[1) 2)]	Request Reset	Writing “1” causes a reset. This bit self-clears upon reset	0: No operation 1: Initiate reset
1-7	Reserved, write as 0		

All user configuration settings are overwritten with their default state (setting stored in the NVM) wherever applicable. This reset operation is functional in all operation modes but must not be performed while NVM writing operation is in

progress.

11.1.11 Timestamp Event Request (0x15)

This register is used by the host to trigger a hardware timestamp of the exact time the register write transaction occurs. The host can query the timestamp value using the Parameter Timestamps (0x0105).

Table 22: Timestamp Event Request Register (0x15)

Bit Number	Field	Value
0[1) 2)]	Trigger Timestamp	0: No operation 1: Timestamp interrupt is triggered
1-7	Reserved, write as 0	

If the Trigger Timestamp bit (Bit 0) of the Timestamp Event Request register is not set, register write transactions will not trigger any timestamp. If the Trigger Timestamp bit (Bit 0) of the Timestamp Event Request register is set, the timestamp will be accessible to the Host in two different ways depending on the value of the Timestamp_Event_Request bit in the Host Interface Control (0x06) register:

- If the Timestamp_Event_Request bit in the Host Interface Control (0x06) register is set, the timestamp will instead be written to the Host Interrupt Timestamp (0x26-0x2A Section 11.1.22).
- If the Timestamp_Event_Request bit in the Host Interface Control (0x06) register is NOT set, a Timestamp Event Status Packet will be generated that can be read from the Status Channel. This Packet have the format explained in Table 23: Timestamp Event Status Packet.

Table 23: Timestamp Event Status Packet

Field Name	Byte Offset	Description
Status Code	0x00-0x01	Timestamp Event Status Code = 0x000D
Length	0x02-0x03	Total number of bytes to follow = 8
Contents	0x04-0x08	40 bit timestamp
Reserved	0x09-0x0B	Reserved, write as 0

11.1.12 Host Control (0x16)

This register is used by the host to control the SPI mode and the I2C watchdog behavior of the BHI385. This register address allows read-write access.

[1] The host may change this bit at any time but only in a single-register transaction (i.e. bursting is not allowed).
[2] The host shall wait T_wake (5μs) after issuing this reset request before resuming transactions since the reset may have to wake up the device from sleep.

Table 24: Host Control Register (0x16)

Bit	Identifier	Description	Value
0	SPI Protocol	This bit has meaning only when the host interface is operated in SPI mode. It distinguishes between 4-wire and 3-wire SPI host protocol. [1)]	0: 4-wire SPI 1: 3-wire SPI
1	I2C Watchdog	This bit has meaning only when the host interface is operated in I2C mode. It controls whether the I2C watchdog timer is enabled. [2)]	0: Disable I2C Watchdog 1: Enable I2C Watchdog
2	I2C Timeout	This bit has meaning only when the I2C watchdog is enabled. It set the duration of inactivity that triggers the watchdog. [3)]	0: 1 ms timeout 1: 50 ms timeout
3-7	Reserved, write as 0		

11.1.13 Host Status (0x17)

This register provides a way for the host to determine the chip's power state and the selected host protocol (SPI or I2C; it will already be determined just by virtue of reading this register after reset). This register address allows read-only access.

Table 25: Host Status Register (0x17)

Bit	Identifier	Description	Value
0	Power State	This bit indicates whether the BHI385 is active or sleeping.	0: Active1: Sleeping
1	Host Protocol	This bit indicates which protocol on the host interface was selected at chip start-up.	0: I2C1: SPI
2-3	Reserved		
4	Host Channel 0 Not Ready	These bits are for debugging only. The host normally does not need to check this first. These bits are set whenever firmware is actively loading bytes into the channel just prior to asserting the host IRQ. If the host reads a channel while the Not Ready bit is set, it will read 0s. This indicates that the host should try again (or wait for the Host Interrupt Signal HIRQ to assert, which indicates the data is ready, or wait for the Interrupt Status register to tell if the data is ready, if the host does not utilize the HIRQ signal).	0: Channel is not ready 1: Channel is ready
5	Host Channel 1 Not Ready		
6	Host Channel 2 Not Ready		
7	Host Channel 3 Not Ready		

11.1.14 Host Channel CRC (0x18-0x1B)

This 32-bit register holds the CRC calculated over the bytes transferred via the last channel being used. The bit-field initializes to 0xFFFFFFFF (initial CRC seed) every time a different channel is accessed. HOST_CRC[0] holds the lower byte of the 32-bit CRC and HOST_CRC[3] the upper byte. The CRC is calculated on a byte-by-byte basis. CRC calculation stops on an underflow error condition. The CRC polynomial follows the ISO-3309 / Ethernet / MPEG-2 standard:

[1] The host may change this bit at any time but only in a single-register transaction (i.e. bursting is not allowed).
[2] The host may change this bit at any time but only in a single-register transaction (i.e. bursting is not allowed). Since the watchdog timer uses the timer oscillator as its time base, the function is not available in deep sleep and is available only after firmware has enabled the timer oscillator.
[3] The same restrictions as for the field I2C Watchdog apply also to I2C Timeout.

$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + x^0$

No CRC is calculated when reading from an empty buffer or when the channel is locked.

11.1.15 Fuser2 Identifier (0x1C)

This register identifies the Fuser2. It reads 0x89 for BHI385.

11.1.16 Fuser2 Revision (0x1D)

This register identifies the hardware revision of the Fuser2. For BHI385 it reads 0x02 after reset and before loading a firmware image, and 0x03 after loading the firmware image.

11.1.17 ROM Version (0x1E-0x1F)

This 16-bit register contains the build number corresponding to the ROM firmware image of the Fuser2. It reads 0x142E for BHI385.

11.1.18 Kernel Version (0x20-0x21)

This 16-bit register contains the build number corresponding to the kernel portion of the firmware image. If none is present, this will read back 0.

11.1.19 User Version (0x22-0x23)

This 16-bit register contains the build number corresponding to the user portion of the firmware image. If none is present, this will read back 0. The value is defined during compilation of the firmware. Firmware images supplied by Bosch Sensortec follow the definition as below:

Table 26: User Version encoding

Register	0x23								0x22							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rule	Major			Minor			Build					Fix				
Example	1			1			16					0				

11.1.20 Feature Status (0x24)

Table 27: Feature Status Register (0x24)

Bit	Identifier	Description	Value
0	Reserved		
1	OPENRTOS Framework	Indicates if the current firmware uses OPENRTOS instead of the legacy interrupt-based framework. Currently, this bit always reads as 1. The value is valid after the firmware has initialized	0: OPENRTOS not used 1: OPENRTOS used
2-4	Host Interface ID	This indicates the interface compatibility. The value depends on the release of the Software Development Kit. The value is valid after the firmware has initialized.	4: Android O and beyond
5-7	Algorithm Id	Software Development Kit. The value is valid after the firmware has initialized	0: None 2: BSX

11.1.21 Boot Status (0x25)

Table 28: Boot Status Register (0x25)

Bit	Identifier	Description	Value
0-3	Reserved		
4	Host Interface Ready	After reset, the host should poll this register until the Host Interface Ready bit is set. Once set, the host interface is ready for the host to upload program RAM or do other operations. Until this, the host should not attempt to write commands. The Host Interface Ready bit will clear once an upload has completed and the host has issued a Boot RAM command. Once boot of the uploaded image completes, the Host Interface Ready bit will be set again. Until this, the host should not attempt to write commands during the window when Host Interface Ready is clear.	0: Not ready 1: Ready
5	Firmware Verify Done	Indicates whether the verification of the firmware image loaded via the host interface is done.	0: Verification not done 1: Verification done
6	Firmware Verify Error	Indicates whether the verification of the firmware image loaded via the host interface was successful.	0: Verification passed 1: Verification failed
7	Firmware Idle	Indicates whether the firmware is running or halted.	0: Firmware running 1: Firmware halted

See Section 8 Device Initialization and Start-up for the usage of these bits.

11.1.22 Host Interrupt Timestamp (0x26-0x2A)

This area is written with the 40-bit timestamp of the host IRQ assertion or the Host Event Request, whichever the host has requested.

If the Timestamp_Event_Request bit in the Host Interface Control (0x06) register is not set, the timestamp written here will be the time of the HIRQ (Host Interrupt Signal) assertion. The update occurs immediately before the Interrupt Status register is updated and the HIRQ signal is asserted.

If the Timestamp_Event_Request bit in the Host Interface Control (0x06) register is set, actual Host Interrupt Timestamp will only be accessible via the Timestamps parameter.

11.1.23 Chip ID (0x2B)

This register outputs the value of the Bosch Sensortec product specific identifier. The returned value is 0x7Cfor the BHI385.

11.1.24 Interrupt Status (0x2D)

This provides a way for a host that does not use the Host Interrupt Signal HIRQ to determine when[1)] data is available in one of the three output channels: Wake-Up FIFO, Non-Wake-Up FIFO or the Status and Debug FIFO.

The interrupt mask bits in the Host Interface Control register (one for each of the two FIFOs) will suppress only the Host Interrupt bit and Host IRQ signal. The bits of this register however will remain asserted, so that the host can still learn that there is a pending interrupt by reading this register. If any bit is set, there is a pending interrupt.

These bits are cleared when the host reads the Wake-Up FIFO, Non-Wake-Up FIFO, or Status channels, for just the channel that has been read from. The Host Interrupt Asserted bit and the HOST IRQ signal will remain asserted until all interrupt sources are retrieved by the host.

[1]The time at which the host interrupt was asserted can be queried via the Host Interrupt Timestamp Parameter or via the Host Interrupt Timestamp

Table 29: Interrupt Status Register (0x2D)

Bit	Identifier	Description	Value
0	Host Interrupt Asserted	This bit reflects the state of the host interrupt GPIO pin.	0: Host interrupt not asserted 1: Host interrupt asserted
1-2	Wake-up FIFO Status	The values of these 2 bits show if the data in the Wake-up FIFO fulfils the condition for generating an interrupt. The value will be 1 “Immediate”, if a virtual sensor event has occurred which was configured with no latency; the value is 2 “Latency” if a sensor has generated data, and the latency period has expired; the value will be 3 “Watermark” if the watermark for the respective FIFO was reached.	0: No data in this FIFO at the time this Int Status was generated 1: Immediate (sensor with 0 latency now has data) 2: Latency (latency timed out for sensor with non-zero latency) 3: Watermark (watermark reached in FIFO)
3-4	Non-Wake-up FIFO Status	The values of these 2 bits show if the data in the Non-Wake-Up FIFO fulfills the condition for generating an interrupt. The value will be 1 “Immediate”, if a virtual sensor event has occurred which was configured with no latency; the value is 2 “Latency” if a sensor has generated data, and the latency period has expired; the value will be 3 “Watermark” if the watermark for the respective FIFO was reached.	0: No data in this FIFO at the time this Int Status was generated 1: Immediate (sensor with 0 latency now has data) 2: Latency (latency timed out for sensor with non-zero latency) 3: Watermark (watermark reached in FIFO)
5	Status	This bit is set if there is data in the Status and Debug FIFO as a result of a command, when the Status and Debug FIFO is in Synchronous Mode (Async bit in Host Interface Control (0x06) is clear).	0: No data in the Status and Debug FIFO 1: Data in the Status and Debug FIFO
6	Debug	This bit is set if asynchronous command responses or debug data is currently stored in the Status and Debug FIFO. If the Async bit in Host Interface Control (0x06) is set (Async Mode), then even response packets as a result of a command will set this bit, not the Status bit.	0: No data in the Status and Debug FIFO 1: Data in the Status and Debug FIFO

registers (if enabled); the later follow the Interrupt Status in the register address space, therefore it is possible for the host to read both items in one transaction.

Bit	Identifier	Description	Value
7	Reset or Fault	This bit is set if this interrupt was caused by a fatal error. The Reset bit will be set after reset and cleared when the initialization sequence completes and the host starts the execution of code RAM upload. A reset will normally not generate a hardware interrupt, as the host needs to configure the interrupt type required using the Host Interrupt Control register before it is safe for the Fuser2 to assert it. However, if the reset occurred due to a watchdog reset or fatal error, and memory records the previous Host Interrupt Control value, a hardware interrupt can be generated.	0: No data in the Status and Debug FIFO 1: Data in the Status and Debug FIFO

11.1.25 Error Value (0x2E)

The Error Value register reports an internal error code. Some of this information is also reported in the Status Interface using the Error Meta Event.

Error recovery strategies are described in Section 16.

Table 30: Error Value Register (0x2E)

Error Register Values	Description (* indicates can be issued from bootloader)	Error Category
0x00	*No Error	
0x10	*Firmware Expected Version Mismatch	Fatal
0x11	*Firmware Upload Failed: Bad Header CRC	Fatal
0x12	*Firmware Upload Failed: SHA Hash Mismatch	Fatal
0x13	*Firmware Upload Failed: Bad Image CRC	Fatal
0x14	*Firmware Upload Failed: ECDSA Signature Verification Failed	Fatal
0x15	*Firmware Upload Failed: Bad Public Key CRC	Fatal
0x16	*Firmware Upload Failed: Signed Firmware Required	Fatal
0x17	*Firmware Upload Failed: FW Header Missing	Fatal
0x19	*Unexpected Watchdog Reset	Fatal
0x1A	ROM Version Mismatch	Fatal
0x1B	*Fatal Firmware Error	Fatal
0x1C	Chained Firmware Error: Next Payload Not Found	Fatal
0x1D	Chained Firmware Error: Payload Not Valid	Fatal
0x1E	Chained Firmware Error: Payload Entries Invalid	Fatal
0x1F	*Bootloader Error: OTP CRC Invalid	Fatal
0x20	Firmware Init Failed	Hardware
0x21	Sensor Init Failed: Unexpected Device ID	Hardware
0x22	Sensor Init Failed: No Response from Device	Programming
0x23	Sensor Init Failed: Unknown	Programming
0x24	Sensor Error: No Valid Data	Programming
0x25	Slow Sample Rate	Temporary
0x26	Data Overflow (saturated sensor data)	Fatal

Error Register Values	Description (* indicates can be issued from bootloader)	Error Category
0x27	Stack Overflow	Fatal
0x28	Insufficient Free RAM	Fatal
0x29	Sensor Init Failed: Driver Parsing Error	Fatal
0x2A	Too Many RAM Banks Required	Programming
0x2B	Invalid Event Specified	Programming
0x2C	More than 32 On Change	Programming
0x2D	Firmware Too Large	Fatal
0x2F	Invalid RAM Banks	Fatal
0x30	Math Error	Fatal
0x31	JTAG interface enabled concurrently with M3	Fatal
0x40	Memory Error	Fatal
0x41	SWI3 Error	Fatal
0x42	SWI4 Error	Fatal
0x43	Illegal Instruction Error	Fatal
0x44	*Unhandled Interrupt Error / Exception / Postmortem Available	Fatal
0x45	Invalid Memory Access	Fatal
0x50	Algorithm Error: BSX Init	Programming
0x51	Algorithm Error: BSX Do Step	Programming
0x52	Algorithm Error: Update Sub	Programming
0x53	Algorithm Error: Get Sub	Programming
0x54	Algorithm Error: Get Phys	Programming
0x55	Algorithm Error: Unsupported Phys Rate	Programming
0x56	Algorithm Error: Cannot find BSX Driver	Programming
0x60	Sensor Self-Test Failure	Hardware
0x61	Sensor Self-Test X Axis Failure	Hardware
0x62	Sensor Self-Test Y Axis Failure	Hardware
0x64	Sensor Self-Test Z Axis Failure	Hardware
0x65	FOC Failure	Hardware
0x66	Sensor Busy	Hardware
0x6F	Self-Test or FOC Test Unsupported	Programming
0x72	No Host Interrupt Set	Fatal
0x73	Event ID Passed to Host Interface Has No Known Size	Programming
0x75	Host Download Channel Underflow (Host Read Too Fast)	Temporary
0x76	Host Upload Channel Overflow (Host Wrote Too Fast)	Temporary
0x77	Host Download Channel Empty	Temporary
0x78	DMA Error	Hardware
0x79	Corrupted Input Block Chain	Programming
0x7A	Corrupted Output Block Chain	Programming
0x7B	Buffer Block Manager Error	Programming
0x7C	Input Channel Not Word Aligned	Temporary
0x7D	Too Many Flush Events	Temporary
0x7E	Unknown Host Channel Error	Hardware
0x81	Decimation Too Large	Programming
0x90	Master SPI/I2C Queue Overflow	Fatal
0x91	SPI/I2C Callback Error	Fatal
0xA0	Timer Scheduling Error	Fatal
0xB0	Invalid GPIO for Host IRQ	Fatal
0xB1	Error Sending Initialized Meta Events	Fatal
0xC0	*Command Error	Temporary
0xC1	*Command Too Long	Temporary

Error Register Values	Description (* indicates can be issued from bootloader)	Error Category
0xC2	*Command Buffer Overflow	Temporary
0xD0	User Mode Error: Sys Call Invalid	Fatal
0xD1	User Mode Error: Trap Invalid	Fatal
0xE1	Firmware Upload Failed: Firmware header corrupt	Fatal
0xE2	Sensor Data Injection: Invalid input stream	Programming

11.1.26 Error Aux, Debug Value, Debug State (0x2F-0x31)

The Error Aux, Debug Value, and Debug State registers are for Bosch Sensortec internal troubleshooting.

Table 31: Debug State Register values (0x31)

Debug State Register Values	Description
0x00-0x0F	Bootloader
0x10-0x1F	Firmware initialization stage 1
0x90-0x9F	Firmware initialization stage 2
0x20-0x2F	Outerloop (main routine, non-RTOS)
0x30-0x7F	Reserved
0x80-0x8F	Deinitialization (teardown on fatal failure)
0xA0-0xAF	Fatal failure
0xB0-0xBF	Inside RTOS task (0xBF = idle task = sleeping)

11.1.27 RGP5 – RGP7 - General-Purpose Host Readable (0x32-0x3D)

These general-purpose registers are provided for communication from the Fuser2 to the host. While the BHI385 ARC processor can read/write these registers, the host has read access only. These registers are available for customer use in custom drivers or extensions.

12 Host Interface Commands

The general structure of host commands and their responses are described in Section 4.5.2 Host Command Protocol.

This section describes the details of all available commands and their responses. The following table provides the overview of the existing commands and their availability in the bootloader and the Event-Driven Software Framework.

Table 32: Overview of BHI385 host interface commands

Command ID	Available in		Command	
	Bootloader	Framework	Description	Response Packet on Status Channel[1)]
0x0001	✔		Download Post Mortem	Crash Dump Status Packet
0x0002	✔		Upload to Program RAM	- none -
0x0003	✔		Boot Program RAM	- none -
0x0007		✔	Set Sensor Data Injection Mode	Injected Sensor Configuration Request Status Packet
0x0008		✔	Inject Sensor Data	- none -
0x0009		✔	FIFO Flush	- none -
0x000A		✔	Soft Pass-Through	Soft Pass-Through Results Status Packet
0x000B		✔	Request Sensor Self-Test	Sensor Self-Test Results Status Packet
0x000C		✔	Request Sensor Fast Offset Compensation	Sensor FOC Results Status Packet
0x000D		✔	Configure Sensor	- none -
0x000E		✔	Change Sensor Dynamic Range	- none -
0x000F		✔	Set Change Sensitivity	- none -
0x0010		✔	Debug Test	- none -
0x0011		✔	DUT Continue	DUT Test Status Packet
0x0012		✔	DUT Start Test	DUT Test Status Packet
0x0015		✔	Control FIFO Format	- none -
0x0017	✔		Raise Host Interface Speed	Raise Host Interface Speed Status Packet
0x0100 ... 0x0FFF		✔	Set Parameter (see Section 12.3)	Set Parameter Status Packet
0x1000 ... 0x1FFF		✔	Get Parameter (see Section 12.3)	Get parameter Status Packet
All others	Reserved			

12.1 Bootloader Commands (incl. Bootloader Status Codes)

12.1.1 Download Post Mortem Data (0x0001)

This command requests that the device places all available data relevant to a watchdog or exception-related reset in the Status Interface for the host to read out. There are no additional fields.

[1] Some commands do not return a status packet; however, a garbled or invalid command will receive a command error status packet. Bootloader commands can fail, so it is required that the host checks for and handles errors on any command.

Table 33: Download post mortem data - Command

Field Name	Byte Offset	Description
Command	0x00-0x01	Download Post Mortem Data Command = 0x0001
Length	0x02-0x03	Total number of bytes to follow = 0

A Crash Dump status packet will be returned in the status channel. It contains useful information as possible, such as stack trace, register values, key data structures, etc.

Table 34: Download post mortem data - Response

Field Name	Byte Offset	Description
Status Code	0x00-0x01	Crash Dump Status Code = 0x0003
Length	0x02-0x03	Total number of bytes to follow
Context	0x04-0x83	R0-R26; GP, FP, SP, ILink, R30, Blink
Valid	0x84	0 = no info; 1 = crash occurred
Flags	0x85	1 = Host Interrupt Control valid 2 = Stack info valid 4 = Stack contents included 8 = Do not autoexec following reset
Stack Size	0x86-0x87	Number of bytes in ROM/Kernel stack
Stack Start	0x88-0x8B	RAM address of start of ROM/Kernel stack
ERET	0x8C-0x8F	See Section 20, Reference 3
ERBTA	0x90-0x93	See Section 20, Reference 3
ERSTATUS	0x94-0x97	See Section 20, Reference 3
ECR	0x98-0x9B	See Section 20, Reference 3
EFA	0x9C-0x9F	See Section 20, Reference 3
Diagnostic	0xA0-0xA3	See Table 35
Icause	0xA4-0xA7	See Section 20, Reference 3
Debug Value	0xA8	Host register "Debug Value"
Debug State	0xA9	Host register "Debug State"
Error Reason	0xAA	Host register "Error Value"
Interrupt State	0xAB	Host register "Error Aux"
Host Interrupt Control	0xAC	Host register "Host Interrupt Control"
Reset Reason	0xAD	0 = POR 1 = External 2 = Host 4 = Watchdog Timeout (or Fatal Exception – see ECR register below)
Host Reset Count	0xAE	Internal use
Error Report	0xAF	Exception cause 0x19 = Watchdog reset 0x1B = Fatal error 0x44 = Unhandled interrupt
MPU_ECR	0xB0-0xB3	See Section 20, Reference 3
User SP	0xB4-0xB7	See Section 20, Reference 3
Reserved	0xB8-0xCC	Reserved bytes
Stack CRC	0xC4-0xC7	CRC of bytes 0xCC-Stack Size + 0xCB
CRC	0xC8-0xCB	CRC of bytes 0x04-0x0C7
Stack dump	0xCC... Stack Size + 0xCB	ROM/Kernel stack dump

Table 35: Diagnostic bit field

Bits	Name	Description
0-3	pm_state	This bit-field reflects the current power state of the BHI385 (see Section 5.5). 0x0, 0x1: PWRUP 0x2, 0x3: ACTV 0x4, 0x5: LSLP 0x8: SLP 0x9: DSLP 0x6: RAMP 0x7, 0xA, 0xB: sleep transitions Others: reserved
4	rom_acc_err	When “1”, this bit indicates that a write access was attempted to a ROM location.
5	iccm_acc_err	When “1”, this bit indicates that a reserved location in the ICCM memory region has been accessed.
6	dccm_acc_err	When “1”, this bit indicates that a reserved location in the DCCM memory region has been accessed.
7	Reserved	
08-10	rst_src	This bit-field relays the source of a reset. It is cleared on a POR; otherwise, the encoding is:[0]=1: external reset happened[1]=1: host command reset happened[2]=1: watchdog reset happened
11-16	Reserved	

12.1.2 Upload to Program RAM (0x0002)

The ROM bootloader will configure the input channel and its DMA interface to allow for upload of a firmware image to program RAM. The data stream will include:

Table 36: Upload to program RAM – Command

Field Name	Byte Offset	Description
Command	0x00-0x01	Upload to Program RAM Command = 0x0002
Length	0x02-0x03	Total number of 32-bit words to follow
Firmware Image	0x04... Length + 0x03	Header + executable image

If the firmware image is found to be invalid (bad ECDSA signature or mismatched CRC), a System Error Status Packet will result. Once the image is verified, the host must issue a Boot Program RAM (0x0003) command. Once initialization is complete, a Meta Event: Initialized will be placed in both the Wake-Up and Non-Wake-Up FIFOs.

12.1.3 Boot Program RAM (0x0003)

This command is required to start a firmware image in RAM. However, this command will be ignored if the ECDSA or CRC verification failed.

Table 37: Boot program RAM – Command

Field Name	Byte Offset	Description
Command	0x00-0x01	Boot Program RAM Command = 0x0003
Length	0x02-0x03	Total number of bytes to follow = 0

12.1.4 Raise Host Interface Speed (0x0017)

This command is required to be sent to the host interface before a firmware upload to enable an SPI clock frequency higher than 20 MHz. It will switch BHI385 from the power-saving Long Run mode into Turbo mode. To return to Long Run mode a reset has to be performed. However, when the firmware is loaded, the power mode is defined by the firmware.

Table 38: Raise host interface speed - Command

Field Name	Byte Offset	Description
Command	0x00-0x01	Raise Host Interface Speed Command = 0x0017
Payload[1)]	0x02–0x07	0x02, 0x00, 0x80, 0x00, 0x00, 0x00

Table 39: Raise host interface speed - Response

Field Name	Byte Offset	Description
Status Code	0x00-0x01	0x0F, 0x00
Length	0x02-0x03	0x04, 0x00
Command	0x04-0x05	0x17, 0x00
Status	0x06-0x07	0x00, 0x00

12.2 Main Firmware Commands (incl. Main Firmware Status Codes)

12.2.1 Set Sensor Data Injection Mode (0x0007)

The purpose of the Sensor Data Injection mode is to support debugging and firmware-in-the-loop testing. In this mode, the sensor input data to the algorithms executed in the BHI385 firmware is provided by the host, instead of the physical sensors. A dedicated firmware version is required for this, which replaces the standard physical sensor drivers with data injection drivers. This can be compiled using the SDK.

This command gives the host the ability to enable either real time operation using injected sensor data (which must arrive fast enough to keep the BSX sensor fusion running at the proper rate), or step-by-step operation.

When in either injection mode, the internal requirements by the BSX sensor fusion library regarding the on/off state and required rates for each physical sensor will be passed back to the host through the Status Interface. The host must use this information to provide simulated sensor data at the proper rates (timestamp deltas).

In order for sensor data injection to work, the firmware image must be built with special physical sensor drivers which help with sensor data injection, rather than normal physical sensor drivers.

Table 40: Set sensor data injection mode – Command

Field Name	Byte Offset	Description
Command	0x00-0x01	Set Sensor Data Injection Mode Command= 0x0007
Length	0x02-0x03	Total number of bytes to follow = 4
Injection Mode	0x04	Normal run mode: 0 = normal run mode Injection mode: 1 = real time injection 2 = step-by-step
Reserved	0x05-0x07	Reserved

[1]This parameter has a special protocol and does not match the regular command format. The Raise Host Interface Speed status packet is sent upon successful reception of the command. If an unexpected status packet is available, the command must be resent until the correct response is received.

The Injected Sensor Configuration Request Status Packet will be issued whenever the BSX fusion library requires a change to a simulated physical sensor, such as turning it on or off or changing its (simulated) rate. A requested Sample Rate of 0 means the host should stop injecting sensor data for this simulated physical sensor.

The host must ensure the timestamps for each incoming simulated physical sensor sample are increasing by the required period (1 divided by the rate).

Table 41: Set sensor data injection mode – Response

Field Name	Byte Offset	Description
Status Code	0x00-0x01	Injected Sensor Configuration Request Status Code = 0x0004
Length	0x02-0x03	Total number of bytes to follow = 8
Sample Rate	0x04-0x07	32-bit floating point number, in Hz
Sensor ID	0x08	Physical sensor ID
Reserved	0x09-0x0B	Reserved

12.2.2 Inject Sensor Data (0x0008)

Injected sensor data contents must follow the command outlined below:

Table 42: Inject sensor data – Command

Field Name	Byte Offset	Description
Command	0x00-0x01	Inject Sensor Data Command = 0x0008
Length	0x02-0x03	Total number of bytes to follow, maximum of 124 (pad to multiple of 4)
Data	0x04.. Length + 0x03	One or more sets of: [Sensor ID, Sensor Data, Timestamp]

The format of the injected data follows the output format for virtual sensor events described in Section 14.1. This includes the full and delta timestamps, which allows to optimize the injected data stream by including delta or full timestamps only when necessary. Due to this format it's basically possible to replay previously recorded output data.

Sensor data injection must be enabled first using the Set Sensor Data Injection Mode Command. That allows the host to control whether real-time or step-by-step injection is to be performed.

The firmware must be built with special sensor data injection drivers instead of normal physical sensor drivers. These drivers intercept on/off and rate requests from the BSX fusion library and pass them to the host as Sensor Data Injection Request Status Interface packets. These drivers receive the injected sensor data and pass it to the rest of the system, including BSX.

The upper bounds for injected sensor data rates are determined by host bus utilization, CPU utilization, RAM available for storing the injected data, and number of injected samples per host injection. Injecting more per transaction will be more efficient than injecting single sensor samples per host transaction.

The host must ensure that the timestamps for each incoming simulated physical sensor sample are increasing by the required period (1 divided by the rate).

12.2.3 FIFO Flush (0x0009)

With this command the host can request BHI385 at any time to initiate a data transfer of the content in one or more FIFOs from BHI385 to the host. The BHI385 will assert the Host Interrupt Signal HIRQ, if data is available. Furthermore, the FIFO Flush command can also be used to discard rather than transfer the data content of any of the FIFOs.

Table 43: FIFO flush – Command

Field Name	Byte Offset	Description	
Command	0x00-0x01	FIFO Flush Command = 0x0009	
Length	0x02-0x03	Total number of bytes to follow = 4	
Flush Value	0x04	**Value**	**Operation**
		0xFF	Flushes (sends) all data of wake-up and non-wake-up FIFOs
		0xFE	Clears out (discards, not sent to host) all FIFOs
		0xFD	Flushes (sends) the wake-up FIFO only
		0xFC	Flushes (sends) the non-wake-up FIFO only
		0xFB	Clears out (discards) the wake-up FIFO
		0xFA	Clears out (discards) the non-wake-up FIFO
		0xF9	Clears out (discards) the debug/status FIFO
Reserved	0x05-0x07	Reserved	

FIFO Flush is an optional mechanism. In the standard FIFO read mechanism, the host does not use this register. Instead, the watermark interrupt, sensor latency interrupt, wake-up and/or non-wake-up FIFO interrupts are used, what determine when the host interrupt occurs without the need of a FIFO Flush command. More details about FIFO reading can be found in Chapter: chapter 15 Reading FIFO Data.

The type of FIFO Flush request determines also to which FIFOs the Flush Complete Meta Event will be appended to. The 0xFC (Flush Non-Wake-up) and 0xFD (Flush Wake-up) place the message in the respective FIFO. The 0xFF (Flush All) request will result in a Flush Complete Meta Event in both FIFOs.

If a FIFO Flush command is sent with a discard flush type (0xF9, 0xFA, 0xFB, or 0xFE), then the current FIFO transfer is cancelled and data is lost for the specified FIFO(s).

12.2.4 Soft Pass-Through (0x000A)

This command can be used during normal operation to read and write registers on devices attached to the BHI385 secondary master I2C or SPI busses. It has two selectable operating modes, the Sensor Driver Mode, using firmware built-in physical device drivers, and the Arbitrary Device Mode, allowing access to external devices without the need for a physical device driver by configuring the transfer parameters. The selection between the two depends on the value of the Master Bus field in Table 45 or Table 46 respectively.

Table 44: Soft pass-through – Command

Field Name	Byte Offset	Description
Command	0x00-0x01	Soft Pass-Through Command = 0x000A
Length	0x02-0x03	Total number of bytes to follow, (pad to multiple of 4)[1)]
Mode	0x04-0x05	Select Sensor Driver Mode or Arbitrary Device Mode.Bit fields defined in Table 45 or Table 46 respectively.
Transfer Rate	0x06-0x07	SPI clock rate in kHz[2)]
Cmd Config / Physical Sensor ID	0x08	Arbitrary Device Mode: Bit fields defined in Table 47 (ignored for I2C) Sensor Driver Mode: Physical Sensor ID, see list in Section 12.3.2.6.
Slave Address	0x09	7 bit I2C slave address or GPIO pin for chip select
Num Bytes	0x0A	Number of bytes to read/write (up to 244)
Reg	0x0B	Register address to read from/write to

[1] The host must write commands in multiples of 4 bytes. If the total number of bytes to follow is not a multiple of 4, the host should write additional bytes to make it so, and include them in the Length field. The Num Bytes field indicates the actual number of bytes to read or write.

[2] Transfer Rate specifies the SPI clock rate in kHz (16-bit unsigned int, LSB first). For I2C master busses, this value is ignored and the rate configured by the firmware image is used instead. The available clock speeds are derived from the current system oscillator (depending on Turbo / Long-Run mode) by dividing by the following values: 2, 3, 4, 6, 8, 16, 32. If a value is selected, that does not follow this rule, the firmware will select the highest possible value below the selected rate.

(Write Data)	0x0C...Num Bytes + 0x0B	If a write command, the data to write

Table 45: Sensor driver mode description

Field Name	Bit Offset	Description
Direction	0	0 = read, 1 = write
Transfer Type	1	0 = single burst, 1 = multiple separate single transfers
Delay Control	2	0 = none (fastest), 1 = delay between bytes
Master Bus[1)]	3-4	0 = Sensor Driver Mode is used 1, 2, 3 = Arbitrary Device Mode is used. 1 = M1, 2 = M2, 3 = M3; See Table 46.
Reserved	5-7	Reserved
Delay Value	8-13	0 – 3: Invalid 4+: Multiples of 50 µs (ignored if Delay Control = 0)
Reserved	14-15	Reserved

Table 46: Arbitrary Device Mode description

Field Name	Bit Offset	Description
Direction	0	0 = read, 1 = write
Transfer Type	1	0 = single burst, 1 = multiple separate single transfers
Delay Control	2	0 = none (fastest), 1 = delay between bytes
Master Bus[1)]	3-4	0 = Sensor Driver Mode is used. See Table table 45 for the rest of the Fields 1 = M1, 2 = M2, 3 = M3; Arbitrary Device Mode is used.
SPI Mode	5	0 = 4 wire, 1 = 3 wire (ignored if I2C)
CPOL	6	SPI clock polarity (ignored if I2C)
CPHA	7	SPI clock phase (ignored if I2C)
Delay Value	8-13	0 – 3: Invalid 4+: Multiples of 50 µs (ignored if Delay Control = 0)
CS Level	14	Chip Select level (ignored if I2C): 0 = CS active low 1 = CS active high
LSbit First	15	Least significant bit first (ignored if I2C)

The *Mode* Field in the Soft Pass-Through Command specifies how the transfer should occur. This includes direction: read or write, single burst or multiple single transfers, and with or without a delay between bytes. The delay can be specified using 6 bits in multiples of 50 µs (with 0 meaning no delay and a minimum of 4, meaning 200 µs). SPI Mode specifies 3 or 4 wire SPI, if no device sensor driver has already configured the mode, as well as CPOL, CPHA, CS Level, and LSbit First. Master Bus Numbers 1-3 correspond to the hardware master bus number (M1 – M3), while 0 for using a sensor descriptor of the firmware image.

[1] It is not possible to use a Master Bus that has not been previously enabled and initialized by the firmware image. The value of this field selects between “Sensor Driver Mode” (value 0) and “Arbitrary Device Mode” (values 1-3)

The *Command Config* Field specifies how the firmware should format the command byte for SPI transactions. The position of the read bit indicates which bit in the command byte determines whether the command is a read or a write. This is usually either bit 0 or bit 7. The polarity of the read bit indicates how to interpret the read bit. A value of 0 means that a value of 0 in the read bit position is a read command, and a value of 1 means that a value of 1 in the read bit position is a read command. The address shift indicates how many bits to left shift the register by when including it in the SPI command byte. If the read bit position is 7, the address shift will typically be 0, while if the read bit position is 0, the address shift will typically be 1.

Table 47: CMD config description

Field Name	Bit Offset	Description
Address Shift	0-3	Number of bits to shift register address in command
Polarity of Read Bit	4	0 = active low, 1 = active high
Position of Read Bit	5-7	Bit number in command byte containing the read bit

The *Slave Address* Field in the Soft Pass-Through Command should be the 7-bit I2C slave address if the master bus is I2C, or the GPIO pin for the chip select if the master bus is SPI.

The Reg Field indicates which register on the connected device to read from or write to. When communicating with a SPI device in read mode, this register will be written first in a write transaction, then a second transaction will be performed to read the output data. In I2C mode, it will always be written with a write transaction followed by a RESTART condition and a read transaction. The (Write Data) Field should be provided only if the Mode indicates this is a write transaction. Once the command completes, a Soft Pass-Through status packet is returned in the Status Channel, as described in Table 48.

Table 48: Soft pass through - Response

Field Name	Bit Offset	Description
Status Code	0x00-0x01	Soft Pass-Through Results Status Code = 0x0005
Length	0x02-0x03	Total number of bytes to follow = 9 + Num Bytes (padded to multiple of 4)
Completion Status	0x04	1 = successful I2C transaction; (for SPI always 1) 2 = I2C NACK or I2C error (no equivalent for SPI)
Mode	0x05-0x06	Same as Soft Pass-Through command. Sensor Driver Mode or Arbitrary Device Mode. Bit fields defined in Table table 45 or Table table 46.
Transfer Rate	0x07-0x08	SPI clock rate in kHz (same as requested, always 0 for I2C)
Cmd Config	0x09	Same as Soft Pass-Through command. Arbitrary Device Mode: Bit fields defined in Table table 47 (ignored for I2C) Sensor Driver Mode: Physical Sensor ID, see list in Section section 12.3.2.6.
Slave Address	0x0A	7-bit I2C slave address or GPIO pin for chip select
Num Bytes	0x0B	Number of bytes read/written
Reg	0x0C	Starting register for the read/write
(Read Data)	0x0D.. Num Bytes + 0x0C	If a read command, the data that was read

This returns a status for both I2C and SPI transfers, though SPI cannot detect that a slave device is not responding. It

also returns the mode, transfer rate, cmd config, and slave address, to help the host to distinguish between results for multiple commands requests. If the command was a read command, the read data is also returned. Returned data will be padded to make the status packet a multiple of 4 bytes. The actual number of bytes read will be indicated in the Num Bytes field.

12.2.5 Request Sensor Self-Test (0x000B)

This command requests a specific physical sensor to run its self-test and report the results.

Table 49: Request sensor self-test - Command

Field Name	Byte Offset	Description
Command	0x00-0x01	Request Sensor Self-Test Command = 0x000B
Length	0x02-0x03	Total number of bytes to follow = 4
Sensor ID	0x04	Sensor ID (only physical sensors)
Reserved	0x05-0x07	Reserved

Sensor ID must be a valid BSX physical input sensor ID. To perform a physical sensor self-test, the sensor must be in inactive mode. If the specified physical sensor is currently active because a related virtual sensor is enabled, the command will be rejected by reporting an error in the Error Register (no status packet will be returned), see Section 11.1.25.

The synchronous Sensor Self-Test Results status packet will be returned once the self-test of this sensor is completed:

Table 50: Sensor self-test result - Response

Field Name	Byte Offset	Description
Status Code	0x00-0x01	Sensor Self-Test Result Status Code = 0x0006
Length	0x02-0x03	Total number of bytes to follow = 8
Sensor ID	0x04	Which sensor is reporting results
Test Status	0x05	0: Test Passed 1: X Axis Failed 2: Y Axis Failed 4: Z Axis Failed 7: Multiple Axis Failed or Single Test Failed (if testing of each axis cannot be done) 8: Unsupported: physical sensor driver does not implement self-test 9: No Device: physical sensor ID provided is invalid 10: The physical sensor is busy; Self-test not performed
X Offset	0x06-0x07	Measured 16-bit value of the applied self-test signal. • Accel static offset in mg. Thresholds required to pass for each axis: X: > +16000 Y: < -15000 Z: > +10000 • Not available for Gyro, returned as 0. • Other sensor results are defined by their physical driver.
Y Offset	0x08-0x09	
Z Offset	0x0A-0x0B	

Sensor ID indicates which sensor is reporting test results.

X, Y, and Z Offset values are returned if available from the self-test results for this sensor, otherwise returned as 0.

- For the built-in Accel sensor, the offset values are returned in mg as 16-bit signed values.
- For the built-in Gyro, offset values are not available and returned as 0.
- For other external sensors, the value is defined by the design of the physical driver.

12.2.6 Request Sensor Fast Offset Compensation (0x000C)

This command requests that a specific physical sensor runs fast offset compensation and reports the results.

Table 51: Request sensor fast offset compensation - Command

Field Name	Byte Offset	Description
Command	0x00-0x01	Request Sensor Fast Offset Compensation Command = 0x000C
Length	0x02-0x03	Total number of bytes to follow = 4
Sensor ID	0x04	Sensor ID
Reserved	0x05-0x07	Reserved

Sensor ID must be a valid BSX physical input sensor ID.

The following status packet returns the results of the Sensor FOC Request command.

Table 52: Sensor fast offset compensation - Response

Field Name	Byte Offset	Description
Status Code	0x00-0x01	Sensor Fast Offset Calibration Result Status Code = 0x0007
Length	0x02-0x03	Total number of bytes to follow = 8
Sensor ID	0x04	Which sensor is reporting results
FOC Status	0x05	See values below
X Offset	0x06-0x07	16-bit sign-extended value in LSB of the physical sensor. Values outside of the indicated range results in failure of FOC. • 8 bits (7.8mg/LSB) for Accel Range 0xFF80 – 0x007F (-128 to 127), +- 1 g max. • 10 bits (61mdps/LSB) for Gyro Range 0xFE00 – 0x0x01FF (-512 to 511), +- 31 dps max.
Y Offset	0x08-0x09	
Z Offset	0x0A-0x0B	

Sensor ID indicates which sensor is reporting test results. The FOC status can be ERROR_FOC_FAIL (0x65), ERROR_UNKNOWN (0x23), or SUCCESS (0).

12.2.7 Configure Sensor (0x000D)

This command allows the host to turn on or off a virtual sensor, as well as set its sample rate and latency.

Table 53: Configure sensor - Command

Field Name	Byte Offset	Description
Command	0x00-0x01	Configure Sensor Command = 0x000D
Length	0x02-0x03	Total number of bytes to follow = 8
Sensor ID	0x04	Sensor ID
Sample Rate	0x05-0x08	Floating point sample rate, in Hz
Latency	0x09-0x0B	Number of milliseconds (24 bits)

The sensor ID field can be any of the virtual sensors supported by the currently loaded firmware image – either non-wake-up or wake-up sensor.

The sample rate field is specified as a 32-bit float, allowing for a very wide range of rates. The host can turn off a sensor

by setting the rate to 0.

The 24-bit Latency field is specified as 0 to request no latency or a non-zero number in milliseconds for the allowed delay before reporting this sensor's data to the host. This allows for latencies up to 4.66 hours.

Changes to the Sample Rate field take effect quickly, but not immediately. If the host wishes to know when the rate change is completed, it can enable and wait for the Sample Rate Changed Meta Event. If the host wishes to know when a sensor has been turned on or off, it can enable and monitor the Power Mode Changed Meta Event.

The actual rate is selected to be equal to or greater than the requested rate and less than twice the requested rate:

$$\textit{Requested Rate} \leq \textit{Actual Rate} < \textit{Requested Rate} \times 2$$

The underlying BSX sensor fusion library supports the following rates:

1.5625 Hz, 3.125 Hz, 6.25 Hz, 12.5 Hz, 25 Hz, 50 Hz, 100 Hz, 200 Hz, 400 Hz, 800 Hz

The requested rate will be modified to match the nearest value equal to or greater than the requested rate.

Note: The rates mentioned above are typical rates. Due to the accuracy of the rates of the underlying sensors, the actual rate may vary depending on the tolerance of the sensors.

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 updated. Due to timing, a change may be too late to affect 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 and will result in immediate host transfer request.

12.2.8 Change Sensor Dynamic Range (0x000E)

This command allows the host to select a different dynamic range for a virtual sensor.

Table 54: Change sensor dynamic range - Command

Field Name	Byte Offset	Description
Command	0x00-0x01	Change Sensor Dynamic Range Command = 0x000E
Length	0x02-0x03	Total number of bytes to follow = 4
Sensor ID	0x04	Sensor ID
Dynamic Range	0x05-0x06	Dynamic Range
Reserved	0x07	Reserved

The sensor ID must correspond to a specific physical sensor's virtual sensor (either wake-up or non-wake-up). It must also be for a physical sensor supported by the currently loaded firmware image. Setting the dynamic range is not supported for any virtual sensor being derived from multiple physical sensors, e.g. Rotation Vector.

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 BSX fusion library 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 Physical Sensor Information to determine the actual current dynamic range, see Section 12.3.2.7.

The host should enable and watch for the Dynamic Range Changed Meta Event, so that 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. BHI385 will automatically select the largest requested dynamic range of all virtual sensors that share this particular physical sensor. 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 for selecting the actual dynamic range for the physical sensor.

The dynamic range will determine the scale factor for the sensor data, based on the data type (number of bits and signed/unsigned).

The units of measurement for dynamic range are the units commonly used for sensor ranges:

- Accelerometer: Earth g-s
- Gyroscope: degrees/second
- Magnetometer: µT
- Others: Defined by physical sensor driver.

12.2.9 Control FIFO Format (0x0015)

This command lets the host change the types of timestamps placed in the wake-up and non-wake-up FIFOs.

Table 55: Control FIFO format - Command

Field Name	Byte Offset	Description
Command	0x00-0x01	Control FIFO Format Command = 0x0015
Length	0x02-0x03	Total number of bytes to follow = 4
FIFO Format Flags	0x04	"Format bits: 0 = all bits off (default format) Bit 1 = disable delta timestamps between samples Bit 2 = disable full timestamp in FIFO header
Reserved	0x05-0x07	Reserved

With **FIFO Format Flags = 0x01**, delta timestamps between samples in the same FIFO transfer will be disabled, but the full timestamp in the FIFO header will still be output (see Table 106).

If the report latency of an enabled virtual sensor is set to 0, then usually (if the host is awake and responsive) each sample will be sent to the host in a single FIFO transfer with the full timestamp in the FIFO header, so it might appear to always contain timestamps.

With **FIFO Format Flags = 0x02**, the full timestamp in the FIFO header will be disabled, but there will still be delta timestamps between samples. If you turn both sources of timestamps off with FIFO Format Flags = 0x03, neither the FIFO header full timestamp nor the between sample delta timestamps will be output.

12.3 Parameter Interface

The Parameter Interface is used to allow configuration and query of the state of the system and the sensors. Writing and reading parameters follows the command structure of the Host Interface Commands as described in Section 14. They are distinguished from other kind of Host Interface Commands by the Command ID, which is in the range of 0x0100 to 0x0FFF for writing parameters and 0x1100 to 0x1FFF for reading parameters, see Table 32.

12.3.1 Reading and Writing Parameters

The parameters described in the following sections are either of read-only, write-only or read/write access. In a single transaction, a parameter can only be accessed as whole in either read or write mode. The way of specifying the access mode is uniform for all parameters and defined by the upper-most digit of the 4-digit hex Parameter ID. Setting it to “1” initiates read access, setting it to “0” initiates write access.

When initiating a read access, a packet of the following format has to be written to Input Channel 0:

Table 56: Parameter read format

Field Name	Byte Offset	Description
Command ID	0x00-0x01	Parameter Read = 0x1XXX, where XXX equals the Parameter ID to be read

Length	0x02-0x03	Total number of bytes to follow = 0

The response data is then placed inside Output Channel 3 corresponding to the format specified for each parameter in the sections below and can be read from there. To write a parameter the complete data structure, as defined below for each parameter, has to be written into Input Channel 0. Within the parameter space, various groups of parameters exist:

Table 57: Parameter groups

Parameter ID	Parameter Group
0x0100 – 0x01FF	System Parameters
0x0200 – 0x02FF	BSX Algorithm Parameters
0x0300 – 0x03FF	Virtual Sensor Information Parameters
0x0500 – 0x05FF	Virtual Sensor Configuration Parameters
0x0800 – 0x0CFF	Advanced Feature Parameters
0x0D00 – 0x0DFF	Multi-tap Detector Parameters
0x0E00 – 0x0EFF	Physical Sensor Control Parameters
0x0F00 – 0x0FFF	Activity Parameters
Others	Reserved

12.3.2 System Parameters

These parameters control general system-wide features:

Table 58: System parameter overview

Parameter ID	Name	Parameter Read	Parameter Write
0x0101	Meta Event Control (0x0101, 0x0102)	Non-wake-up FIFO Meta Event Enables	Meta Event Enables
0x0102	Meta Event Control (0x0101, 0x0102)	Wake-up FIFO Meta Event Enables	Meta Event Enables
0x0103	FIFO Control (0x0103)	Wake-up and non-wake-up FIFOs Watermark; FIFOs size	Watermark Configuration
0x0104	Firmware Version (0x0104)	Firmware Version	Not Used
0x0105	Timestamps (0x0105)	Timestamps	Not Used
0x0106	Framework Status	Internal Data; Watchdog	Not Used
0x011F	Virtual Sensors Present (0x011F)	Bitmap of Compiled-in Virtual Sensors	Not Used
0x0120	Physical Sensors Present (0x0120)	Bitmap of Compiled-in Physical Sensors	Not Used
0x0121 - 0x0160	Physical Sensor Information (0x0121 – 0x0160)	Orientation Matrix etc.	Orientation Matrix
Others	Reserved	Not Used	Not Used

12.3.2.1 Meta Event Control (0x0101, 0x0102)

Meta Event Control Parameter 0x0101 is used for enabling and disabling non-wake-up FIFO Meta Events, while Meta Event Control Parameter 0x0102 is used for enabling and disabling wake-up FIFO Meta Events.

The 8 bytes in this writeable parameter is divided into 32 two-bit sections. Each section controls whether the corresponding Meta Event is enabled (so that it will appear in the output FIFO when it occurs), as well as whether the occurrence of this Meta Event will lead to an immediate host interrupt. See Section 14.3 Format of Meta Events for a list of Meta Events. Each specific Meta Event has a default enable state, also specified in that section.

The MSbit in each two-bit section is the event enable, and each LSbit is the event interrupt enable, as shown in the following table:

Table 59: Meta event control

Field Name	Byte Offset	Description								Access
Parameter ID	0x00 - 0x01	Meta Event Control: 0x0101: Non-Wake-up FIFO Meta Event Control 0x0102: Wake-up FIFO Meta Event Control								
Length	0x02 - 0x03	Number of bytes to follow = 8								
Bitmap		Bit 7: Event Enable	Bit 6: Int Enable	Bit 5: Event Enable	Bit 4: Int Enable	Bit 3: Event Enable	Bit 2: Int Enable	Bit 1: Event Enable	Bit 0: Int Enable	
	0x04	Meta Event 4		Meta Event 3		Meta Event 2		Meta Event 1		Read/Write
	0x05	Meta Event 8		Meta Event 7		Meta Event 6		Meta Event 5		Read/Write
	0x06	Meta Event 12		Meta Event 11		Meta Event 10		Meta Event 9		Read/Write
	0x07	Meta Event 16		Meta Event 15		Meta Event 14		Meta Event 13		Read/Write
	0x08	Meta Event 20 [1)]		Meta Event 19		Meta Event 18		Meta Event 17		Read/Write
	0x09	Meta Event 24		Meta Event 23		Meta Event 22		Meta Event 21		Read/Write
	0x0A	Meta Event 28		Meta Event 27		Meta Event 26		Meta Event 25		Read/Write
	0x0B	Meta Event 32		Meta Event 31		Meta Event 30		Meta Event 29		Read/Write

12.3.2.2 FIFO Control (0x0103)

This Parameter provides a mechanism for the host to set a target watermark level, which is the number of bytes the wake-up FIFO buffer and/or the non-wake-up FIFO buffer may 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, since additional data may be written into the FIFO while the host responds on the interrupt signal. It is up to the customer to determine, in their application, based on the maximum host rate and host interrupt response time, to determine a safe maximum watermark level[2)].

The size of each FIFO can be retrieved by reading the same parameter. It is returned in bytes 8-11 for the wake-up FIFO and bytes 16-19 for the non-wake-up FIFO. This size is determined at compile time of the firmware image and can be configured in the board configuration file.

Table 60: FIFO control

Field Name	Byte Offset	Description	Access
Parameter ID	0x00 - 0x01	FIFO Control: 0x0103	-
Length	0x02 - 0x03	Number of bytes to follow = 16	-
Wake-up FIFO Watermark	0x04 - 0x07	Number of bytes in FIFO before interrupt is asserted	Read/Write
Wake-up FIFO Size	0x08 - 0x0B	FIFO Size in bytes	Read only
Non-Wake-Up FIFO Watermark	0x0C - 0x0F	Number of bytes in FIFO before interrupt is asserted	Read/Write

[1] Although the Meta Event 20 “Spacer” is always enabled, it is reported as “disabled” in the meta Event Control parameter.

[2] A non-zero Watermark has no effect if all enabled virtual sensors are configured with a low maximum report latency, and the Host is active (the Host Interface Control register’s AP Suspend bit is 0). In this case, host interrupts are generated based on the report latency requirements of the virtual sensors, and the FIFO level may never reach the watermark level. The Watermark is only useful when all enabled continuous output sensors are configured with non-zero latencies, or the host is suspended, and no wake-up events occur.

Table 60: FIFO control

Field Name	Byte Offset	Description	Access
Non-Wake-Up FIFO Size	0x10 - 0x13	FIFO Size in bytes	Read only
Status FIFO Size	0x14 - 0x17	FIFO Size in bytes	Read only

12.3.2.3 Firmware Version (0x0104)

The firmware version register provides a host-readable version information.

The Custom Version Number can be set by the developer during compilation of the firmware in the board .cfg, see the Programmer's Manual for details.

The Kernel hash 1, Kernel hash 2 and Customer Hash fields provide 6 bytes each for specifying a unique number to identify a build. This could be, for example, the most significant 6 bytes of a Git commit hash. While the Kernel hashes are defined by Bosch Sensortec and are fixed with every SDK release, the User Hash can be defined by the customer.

The Hash values are captured during the build of the firmware. If the SDK is located in a Git tree, the Git hash will be used.

If the SDK is not located in git, the values of the file <SDK_root>/version and <SDK_root>/hash are used.

Table 61: Firmware version

Field Name	Byte Offset	Description	Access
Parameter ID	0x00 - 0x01	Firmware Version: 0x0104	-
Length	0x02 - 0x03	Number of bytes to follow = 20	-
Custom Version Number	0x04 - 0x05		Read only
Kernel hash 1	0x06 - 0x0B		Read only
Kernel hash 2	0x0C - 0x11		Read only
User hash	0x12 - 0x17		Read only

12.3.2.4 Timestamps (0x0105)

In order to provide a mechanism for the host to translate 40-bit sensor data timestamps to host-relative timestamps[1)], this parameter may be read to determine the time at which the last host-interrupt was asserted, as well as the current system time.

Table 62: Timestamps

Field Name	Byte Offset	Description	Access
Parameter ID	0x00 - 0x01	Timestamps: 0x0105	-
Length	0x02 - 0x03	Number of bytes to follow = 16	-
Host Interrupt Timestamp	0x04 - 0x08	Time (in units of 1/64000 seconds) latched in hardware that the last host interrupt was triggered (5 bytes)	Read only
Current Timestamp	0x09 - 0x0D	Time (in units of 1/64000 seconds) for current system time (5 bytes)	Read only

[1] The Host Interrupt Timestamp can be read more efficiently from the Host Interrupt Timestamp registers.

Table 62: Timestamps

Field Name	Byte Offset	Description	Access
Timestamp Event	0x0E - 0x12	Time (in units of 1/64000 seconds) immediately upon receipt and processing of the Get Parameter command for this Parameter Number latched in hardware when host wrote to the Timestamp Event Request register (5 bytes)	Read only
Reserved	0x13	Reserved	Read only

12.3.2.5 Virtual Sensors Present (0x011F)

This Parameter contains a read-only 256-bit bitmap, where a set bit indicates the corresponding virtual sensor is present in the system.

For example, if a virtual accelerometer, magnetometer, and humidity sensor were the only virtual sensors present, the bit map would have bits set for sensor ID 4 (accelerometer), 22 (magnetometer) and 130 (humidity). In this example, the bit map would be:

Byte 0: 0001 0000 (binary; left most bit is bit 7, right most is bit 0)

Byte 1: 0000 0000 (left most is bit 15; right most is bit 8)

Byte 2: 0100 0000

. . .

Byte 16: 0000 0100

. . .

Byte 31: 0000 0000

Table 63: Virtual sensors present

Field Name	Byte Offset	Description	Access
Parameter ID	0x00 - 0x01	Virtual Sensors Present: 0x011F	-
Length	0x02 - 0x03	Number of bytes to follow = 32	-
Virtual Sensors present	0x04 - 0x23	A 256-bit bitmap. In there, each virtual sensor is represented by one bit at the position of its ID, where “1” stands for present, and “0” stands for non-present.	Read only

12.3.2.6 Physical Sensors Present (0x0120)

This Parameter contains a read-only 64-bit bitmap, where a set bit indicates the corresponding physical sensor is present in the system. The sensor IDs in this parameter are BSX Input IDs, which are different from the BSX OUTPUT and WAKEUP IDs listed later. The following IDs are currently assigned:

- Accelerometer = 1
- Gyroscope = 3
- Magnetometer = 5
- Temperature Gyroscope = 7
- Pressure = 11
- Position = 13
- Humidity = 15
- Temperature = 17

- Gas Resistor = 19
- Step Counter = 32
- Step Detector = 33
- Significant Motion = 34
- Any Motion = 35
- External Camera Input = 36
- GPS = 48
- Light = 49
- Proximity = 50
- Activity Recognition = 52
- No motion = 55
- Wrist Gesture Detector = 56
- Wrist Wear Wakeup = 57
- BMP Temperature = 59
- BMP Pressure = 61

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), 5 (magnetometer) and 15 (humidity). In this example, the bit map would be:

Byte 0: 0010 0010 (binary; left most bit is bit 7, right most is bit 0)

Byte 1: 1000 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

Table 64: Physical sensors present

Field Name	Byte Offset	Description	Access
Parameter ID	0x00 - 0x01	Physical Sensors Present: 0x0120	-
Length	0x02 - 0x03	Number of bytes to follow = 8	-
Physical Sensors present	0x04 - 0x0B	A 64-bit bitmap. In there, each physical sensor is represented by one bit at the position of its ID, where “1” stands for present, and “0” stands for non-present.	Read only

12.3.2.7 Physical Sensor Information (0x0121 – 0x015F)

Each parameter in the range of 0x0121 to 0x015F refers to a specific physical sensor ID. The parameter 0x0121 refers to Physical Sensor ID 1, while 0x015F refers to Physical Sensor ID 63. This structure is returned for any Parameter Number read when the corresponding physical sensor is present. If not present, this structure returns all 0s.

Table 65: Physical sensor information

Field Name	Byte Offset	Description	Access
Parameter ID	0x00 - 0x01	Physical Sensor Information: 0x0121 - 0x015F	-
Length	0x02 - 0x03	Number of bytes to follow = 20	-
Physical Sensor ID	0x04	Physical Sensor ID, e.g. 0x01 for parameter 0x0121	Read only
Driver ID	0x05	Unique per driver / vendor / part number	Read only
Driver Version	0x06	Denotes notable change in behavior	Read only
Current Consumption	0x07	Estimated current consumption of the physical sensor. Scale factor is 0.1 mA	Read only
Current Dynamic Range	0x08 - 0x09	Current dynamic range of sensor in SI units	Read only
Flags	0x0A	Bit 0: IRQ enabled 0: disabled 1: enabled Bits 1-4: master interface used: 0: none 1: SPI0 2: I2C0 3: SPI1 4: I2C1 Bits 5-7: power mode 0: Sensor Not Present 1: Power Down 2: Suspend 3: Self-Test 4: Interrupt Motion 5: One Shot 6: Low Power Active 7: Active	Read only
Slave Address	0x0B	If I2C: 7 bit device address (MSbit = 0) If SPI: the GPIO pin used as chip select	Read only
GPIO Assignment	0x0C	GPIO pin used as data ready interrupt input	Read only
Current Rate	0x0D - 0x10	Current sample output rate of sensor in Hz, as 32-bit float value	Read only
Number of Axes	0x11	Number of axes of the sensor (e.g., X/Y/Z = 3)	Read only
Orientation matrix	0x12 - 0x16	Orientation matrix of the sensor, 5 bytes, see the description below	Read only
Reserved	0x17	-	-

The orientation matrix is provided for sensors that support this, such as 3-axis accelerometers, magnetometers, and gyroscopes. It 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:

$$[XYZ] = [X_s\, Y_s\, Z_s] \cdot \begin{bmatrix} C_0 & C_1 & C_2 \\ C_3 & C_4 & C_5 \\ C_6 & C_7 & C_8 \end{bmatrix}$$

The orientation matrix output from this parameter is in the same order as the elements listed in the board configuration file used to generate a firmware image, described in Programmer's Manual, where each matrix element is stored in successive nibbles.

For example, if the board configuration file contains:

#DriverID,Bus,Addr,GPIO,C0,C1,C2,C3,C4,C5,C6,C7,C8,Off0,Off1,Off2,MaxRate

48,spi0,25,2, 1, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0, 0, -1.00000

Then bytes 12-16 of the Physical Sensor Information structure would be:

Byte 0x12: 01 (hexadecimal)

Byte 0x13: 00

Byte 0x14: 0F

Byte 0x15: 00

Byte 0x16: 0F

(Notice: Read from the right to left in Byte)

This Parameter can also be used to update the orientation matrix in use at runtime. When written, the following structure is used:

Table 66: Orientation matrix write format

Field Name	Byte Offset	Description
Parameter ID	0x00 - 0x01	Physical Sensor Information: 0x0121 - 0x0160
Length	0x02 - 0x03	Number of bytes to follow = 8
Orientation Matrix	0x04 – 0x08	See bytes 0x12 to 0x16 in Table 65
Reserved	0x09 – 0x0B	-

For more details regarding axis remapping, please refer to Section 20.3 Sensing Axes and Axis Remapping.

12.3.3 BSX Algorithm Parameters

The BSX Algorithm parameters are used to read and control various aspects of the Sensor Fusion algorithms in the BHI385. The interface directly with the BSX library in the BHI385 firmware, which implements the fusion algorithms.

Parameters 0x0201 – 0x0240 contain calibration states for each physical sensor, matching the BSX input IDs in the range of 0x01 to 0x40. The BSX library currently supports accelerometer, gyroscope, and magnetometer as physical sensors, so the available parameters within this range are 0x0201, 0x0203, and 0x0205. The calibration state contains intermediate calibration state for each sensor data. It should be saved when the accuracy of the corresponding sensor reaches 3 and loaded on system boot to achieve the effect of warm start.

Besides the physical sensor calibration states it is also possible to write a soft iron calibration (SIC) matrix and read BSX version information. In order to obtain an accurate heading estimate, the impact of nearby magnetic disturbances must be mitigated, user should generate SIC matrix according to real product and write to BHI385.

Noted that parameters 0x0205 and 0x027D are only available when you have magnetometer connected.

Table 67: Algorithm parameters

Parameter ID	Usage	Content Format	Access
0x0201	Calibration state for accelerometer	BSX State Exchange Structure	Read/Write
0x0203	Calibration state for gyroscope		Read/Write
0x0205	Calibration state for magnetometer		Read/Write

Table 67: Algorithm parameters

Parameter ID	Usage	Content Format	Access
0x027D	SIC Matrix		Read/Write
0x027E	BSX Version	4-byte BSX Version number	Read only

12.3.3.1 Reading and writing the BSX State Exchange Structure

The calibration state and SIC matrix[1)] are read / written through multiple operations due to the large size of the data. Each operation transfers one data block with the format of BSX state exchange structure.

During the read operation, the host is responsible for assembling the data together according to block number and block data length in BSX state exchange structure. Each block contains at most 64 bytes of data. The completion flag is set in the last block. When this bit is set, it notifies the host that the last block has been read, and the BSX state exchange structure is complete.

During the write operation, the host is responsible for breaking down the previously saved data to blocks in the BSX state exchange structure format with each block containing at most 64 bytes of data. All blocks but the last block should contain exactly 64 bytes of data. The last block contains the remaining data and the transfer length should still be an entire BSX state exchange structure. The completion flag must be set in the last block.

Table 68: BSX state exchange structure

Field Name	Byte Offset	Description	Access
Parameter ID	0x00 - 0x01	Calibration state: 0x0201 - 0x0240 SIC matrix: 0x027D	-
Length	0x02 - 0x03	Number of bytes to follow <= 68	-
Block information	0x04	Bit 0-6: Block number starting at 0 Bit 7: Completion flag	Read/Write
Block length	0x05	Valid data length in the current block	Read/Write
Structure length	0x06 - 0x07	Total data length for the entire data	Read/Write
Block data	0x08 - 0x47 max	Block data	Read/Write

12.3.3.2 BSX Version (0x027E)

The 4-byte BSX Version information can be read with the following parameter:

Table 69: BSX version

Field Name	Byte Offset	Description	Access
Parameter ID	0x00 - 0x01	BSX Version: 0x027E	-
Length	0x02 - 0x03	Number of bytes to follow = 4	-
Major Version	0x04	BSX major version	Read only
Minor Version	0x05	BSX minor version	Read only
Major Bug Fix Version	0x06	BSX major bug fix version	Read only
Minor Bug Fix Version	0x07	BSX minor bug fix version	Read only

12.3.4 Virtual Sensor Information Parameters (0x0301 – 0x03BF)

Each parameter within the range of 0x0301 to 0x03BF is read-only and refers to a specific Virtual Sensor ID as specified in Section FIFO Data Types and Format. For example, the parameter 0x0301 refers to virtual sensor ID 1, while 0x03BF refers to virtual sensor ID 191 (0xBF).

[1] The calibration state and SIC matrix use a BSX internal binary data format and are not intended to be interpreted by the host.

The structure outlined in Table 70 is returned for every parameter and reports essential information about a virtual sensor. If the requested sensor ID is not supported by the current firmware image, then all fields of this structure are reported as zero. All values are of type unsigned integer, LSB first, with their respective length, unless specified differently.

Table 70: Virtual sensor information structure

Field Name	Byte Offset	Description	Access
Field Name	Byte Offset	Description	Access
Parameter ID	0x00 - 0x01	Virtual Sensor Information: 0x0301 - 0x03BF	-
Length	0x02 - 0x03	Number of bytes to follow = 28	-
Sensor ID	0x04	Same as (Parameter Number - 0x0300)	Read only
Driver ID	0x05	Unique per driver / vendor / part number	Read only
Driver Version	0x06	Denotes notable change in behavior	Read only
Power	0x07	Estimated current consumption of the virtual sensor. Scale factor is 0.1 mA	Read only
Max Range	0x08 - 0x09	Maximum range of sensor data in SI units	Read only
Resolution	0x0A - 0x0B	Number of bits of resolution of the underlying sensor	Read only
Max Rate	0x0C - 0x0F	Maximum supported output data rate in Hz, as 32-bit float value	Read only
FIFO Reserved	0x10 - 0x13	Number of samples of this virtual sensor for which FIFO space is reserved. This can be 0 in case of a single shared FIFO	Read only
FIFO Max	0x14 - 0x17	Maximum number of samples of this virtual sensor that can be stored when using the entire FIFO	Read only
Event Size	0x18	Number of bytes in FIFO for this virtual sensor's data packet (including Sensor Type)	Read only
Min Rate	0x19 - 0x1C	Minimum supported output data rate in Hz, as 32-bit float value	Read only
Reserved	0x1D - 0x1F	-	-

The Max Range field will be set to the maximum possible range, that the underlying physical 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 units commonly used for sensor ranges (Earth Gs, degrees/second, µT). 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 by dividing the Max Range or current Dynamic Range by 2 Resolution for unsigned values, or by 2 Resolution-1 for signed values.

12.3.5 Virtual Sensor Configuration Parameters (0x0501 – 0x05BF)

Each parameter in the range of 0x0501 to 0x05BF is read-only and refers to a specific Virtual Sensor ID as specified in Section 14. For example, the parameter 0x0501 refers to virtual sensor ID 1, while 0x05BF refers to virtual sensor ID 191.

The structure outlined in Table 71 is returned for every parameter and reports the current configuration of a virtual sensor. If the requested sensor ID is not supported by the current firmware image, then all fields of this structure are reported as zero. All values are of type unsigned integer, LSB first, with their respective length, unless specified differently.

Table 71: Virtual sensor configuration structure

Field Name	Byte Offset	Description	Access
Parameter ID	0x00 - 0x01	Virtual Sensor Configuration: 0x0501 - 0x05BF	-
Length	0x02 - 0x03	Number of bytes to follow = 12	-

Field Name	Byte Offset	Description	Access
Sample Rate	0x04 - 0x07	Rate in Hz, reads back the actual current sensor rate, as 32-bit float value	Read only
Max Report Latency	0x08 - 0x0B	Is 0 for non-batch mode (no latency), reads back actual current latency in ms	Read only
Reserved	0x0C - 0x0D	-	-
Dynamic Range	0x0E - 0x0F	Range setting for underlying physical sensor in their respective units. See Section 12.2.8	Read only

12.3.6 Advanced Feature Parameters(0x0800 – 0x0CFF)

Advanced features include, but are not limited to, the following functions such as Head Orientation, Self-Learning AI , Swim, Motion AI Studio, SCS(Smart Connected Sensors), BSEC(Bosch Sensortec Environmental Cluster), etc. Accessing and utilizing these advanced features requires using an SDK or FW including these specific algorithms and referencing the corresponding application notes.

12.3.6.1 Self-Learning AI Software Parameters (0x0900 – 0x090F)

These parameters allow the host to control the operation and configuration of the Self-Learning AI Software. For the content format, see the corresponding subsections for each parameter.

Table 72: Self learning AI parameter IDs

Parameter ID	Usage	Access
0x900	Learning and recognition state	Read/Write
0x901	Read learned pattern	Read
0x901	Write pattern for recognition	Write
0x902	Algorithm parameters	Read/Write
0x904	Pattern state operation	Write
0x905	Pattern similarity calculation	Read/Write
0x907	Driver Status	Read
0x908	Reset driver	Write
0x909	Pattern parameters	Read/Write

Learning and recognition state Reading this parameter returns the current state of the learning and recognition algorithms. The reset is always read as 0. Writing 1 to the reset field resets all internal algorithm state and removes any patterns loaded for recognition.

Table 73: Learning and recognition state

Field Name	Byte Offset	Description	Access
Parameter ID	0x00 - 0x01	0x1900 for read, 0x900 for write.	-
Length	0x02 - 0x03	Number of bytes to follow = 4	-
Learning enabled	0x04	0 - disabled. 1 - enabled.	Read/Write
Learning reset	0x05	0 - no operation. 1 - Reset.	Write
Recognition enable	0x06	0 - disabled. 1 - enabled.	Read/Write
Recognition reset	0x07	0 - no operation. 1 - Reset.	Write

Read learned pattern After learning of a new pattern has been signaled by a sensor data event, the learned pattern may be read using this parameter. The returned pattern is wrapped in structure as described in Table 74. When reading a learned pattern, the size and pattern_data fields contain relevant information, and the pattern_data field may later be used for recognition or stored for future use.

Note: The learned pattern will be cleared after reading it or when learning state is reset by writing the "Learning and recognition state" parameter.

Table 74: Format used for patterns

Parameter ID	Usage	Description
Parameter ID	0x00 - 0x01	0x1901 for read, 0x901 for write.
Length	0x02 - 0x03	Number of bytes to follow = length of parameter payload. The parameter payload (Reserved + Size + Pattern ID + Pattern data + zero-padded data) need to be zero-padded at the end and total size is aligned to a multiple of 4 bytes.
Reserved	0x04	Write as 0.
Size	0x05 - 0x06	Size of pattern_data in bytes. 16-bit unsigned integer.
Pattern ID	0x07	ID of pattern to write. Must start at 0 after reset, and be incremented by 1 for each pattern written.
Pattern data	0x08 -	Actual pattern data describing the exercise(maximum pattern blob size: 244 max).

Write pattern for recognition This parameter is used for writing new patterns for the recognition algorithm. The pattern is wrapped in the same kind of structure as when reading patterns, see Table 74. When writing patterns, the first pattern must be written with pattern_id equal to 0, and the pattern_id must then be incremented with one for each pattern written.

Table 75: Self-learning AI algorithm parameters

Algorithm ID	Direction	Description	Parameter Data Format
2	Read / Write	The approximate number of cycles / repetitions for recognition to recognize an activity	IEEE754 Single precision float
7	Read	Maximum pattern blob size in bytes	16-bit unsigned integer
8	Read	Maximum number of patterns.	16-bit unsigned integer
10	Read / Write	Flag if insignificant movements should be ignored in learning	8-bit unsigned integer, value 0 or 1

Algorithm parameters Writing algorithm parameters is only permitted when the algorithm is reset and disabled, i.e. before writing 1 to any of the enable fields in the state parameter.

Table 76: Parameter data

Field Name	Byte Offset	Description
Parameter ID	0x00 - 0x01	0x1902 for read, 0x902 for write.

Length	0x02 - 0x03	Number of bytes to follow = length of parameter payload. The parameter payload (Algorithm ID + Reserved + Size + Data + zero-padded data) need to be zero-padded at the end and total size is aligned to a multiple of 4 bytes.
Algorithm ID	0x04	ID of the algorithm parameter to write.
Reserved	0x05	Write as 0.
Size	0x06	Size of the data as an 8-bit unsigned integer.
Data	0x07 -	Actual data in the format specified by the Parameter Data Format in table Self-learning AI algorithm parameters.

Reading an algorithm parameter is performed by first writing the parameter with a size of 0 and no data, and then reading back the actual parameter data.

Pattern State Operation There are currently three pattern operations that can be performed, which are to enable, disable or switch hand of the given patterns. After a pattern has been written, it needs to be enabled before being used for recognition.

Table 77: Format used for pattern state operations

Field Name	Byte Offset	Description
Parameter ID	0x00 - 0x01	0x904
Length	0x02 - 0x03	Number of bytes to follow = length of parameter payload. The parameter payload (Operation + Count + Pattern IDs + zero-padded data) need to be zero-padded at the end and total size is aligned to a multiple of 4 bytes.
Operation	0x04	Operation to perform as an 8-bit unsigned integer. 0 = Pattern disable 1 = Pattern enable 2 = Switch hand
Count	0x05	Number of pattern indices to perform the operation on as an 8-bit unsigned integer.
Pattern IDs	0x06 -	List of pattern IDs to perform operation on, specified as 8-bit unsigned integers.

Pattern Similarity Calculation This parameter is used for comparing the similarity between two different patterns. This is a process involving 3 stages:

1. Write first pattern, data is a pattern wrapped in the structure defined in Table 78.
2. Write second pattern, data is a pattern wrapped in the structure defined in Table 78.
3. Read similarity, this step may need to be repeated until calculation is completed.

These 3 stages are implemented in the helper function bhy2_klio_similarity_score(). The data returned by the third step is an IEEE754 Single precision float where 1.0 means the pattern are very similar, and negative values means they are very different.

This parameter is also used to perform similarity calculations between multiple patterns already uploaded to the BHI385.

To perform this operation, the following format is used, and the result is read as an array of IEEE754 Single precision floats. This operation is implemented in the helper function bhy2_klio_similarity_score_multiple().

Table 78: Format used to start multiple pattern comparison

Field Name	Byte Offset	Description
Parameter ID	0x00 - 0x01	0x905
Length	0x02 - 0x03	Number of bytes to follow = length of parameter payload. The parameter payload (Zero + Index + Count + Pattern IDs + zero-padded data) need to be zero-padded at the end and total size is aligned to a multiple of 4 bytes.
Zero	0x04 - 0x07	Write as 0, used to distinguish from regular pattern comparison.
Index	0x08	Base pattern to compare with other patterns.
Count	0x09	Number of pattern indices to perform the operation on as an 8-bit unsigned integer.
Indexes	0x0A -	List of pattern IDs to perform operation on, specified as 8-bit unsigned integers.

Table 79: Format used to read the result of multiple pattern comparison

Field Name	Byte Offset	Description
Parameter ID	0x00 - 0x01	0x1905
Length	0x02 - 0x03	Number of bytes to follow = the number of scores multiplied by 4 bytes.
Scores	0x04 -	List of similarity scores, four bytes per score requested.

Driver status The driver status can provide the state of the executed parameter writes. It can be checked after each parameter operation.

Table 80: Driver status format

Field Name	Byte Offset	Description
Parameter ID	0x00 - 0x01	0x1907
Length	0x02 - 0x03	Number of bytes to follow = 4
Status	0x04 - 0x08	Driver status, returned as an unsigned 32-bit integer. 0 = No error 1 = Invalid parameter 2 = Parameter out of range 3 = Invalid pattern operation 4 = Not implemented 5 = Buffer too small for requested operation 6 = Internal error 7 = Undefined error 8 = Previous operation is still progressing

Reset driver Reset Self-Learning AI driver to its startup state.

Table 81: Reset driver format

Field Name	Byte Offset	Description
Parameter ID	0x00 - 0x01	0x0908
Length	0x02 - 0x03	Number of bytes to follow = 4
Reset	0x04	Write as 1
Reserved	0x05-0x07	Write as 0

Pattern parameters Read and write parameters of individual patterns. Writing pattern parameters is only permitted when the algorithm is reset and disabled, i.e. before writing 1 to any of the enable fields in the state parameter. In addition, patterns must have been written for recognition.

Table 82: Pattern parameter

Pattern Parameter ID	Byte Offset	Description	Parameter Data Format	Direction
0	0x00 - 0x01	Exponent scaling factor. Allowed values are between 0.0 and 2.0. A value smaller than 1.0 will make the pattern less likely to be recognized. A value larger than 1.0 will make the pattern more likely to be recognized.	IEEE754 Single precision float	RW

Table 83: Format used for pattern parameter

Field Name	Byte Offset	Description
Parameter ID	0x00 - 0x01	0x1909 for read, 0x909 for write.

Length	0x02 - 0x03	Number of bytes to follow = length of parameter payload. The parameter payload (Pattern ID + Pattern parameter ID + Size + Reserved + Data + zero-padded data) need to be zero-padded at the end and total size is aligned to a multiple of 4 bytes.
Pattern ID	0x04	Pattern ID for which to write parameter, specified as an 8-bit unsigned integer.
Pattern parameter ID	0x05	ID of the pattern parameter to write, specified as an 8-bit unsigned integer.
Size	0x06	Size of the data as an 8-bit unsigned integer.
Reserved	0x07	Write as 0.
Data	0x08 -	Actual data in the format specified by the Pattern Parameter Data Format.

Reading a pattern parameter is performed by first writing the parameter with a size of 0 and no data, and then reading back the actual parameter data.

12.3.7 Multi-tap Detector Parameters (0X0D00 – 0x0DFF)

Table 84: Multi-tap detector parameters

Parameter number	Functionality	Access	Payload length	Payload description	Comment
0X0D01 (1)	Enable taps	R/W	4	Byte 0: bit 0 for detecting single tap, bit 1 for detecting double taps, bit 2 for detecting triple taps Byte 1: Not used Byte 2: Not used Byte 3: Not used	Default value: 0x00000007 (Single tap, double taps and triple taps will be detected)
0X0D02 (2)	Tap detector configuration	R/W	6	See table 85	The structure follows endianness and data structure alignment with Fuser2.

Table 85: Data structure of tap detector configuration

Byte Offset	Bits	Field Name	Description	Default Value
0x00	1:0	axis_sel	Tap axis selection 0 - X 1 - Y 2 - Z	2
	2	wait_for_timeout	Wait for gesture confirmation 0 - disabled 1 - enabled	1
	5:3	max_peaks_for_tap	Maximum number of peaks that can occur when a tap is done	6

Byte Offset	Bits	Field Name	Description	Default Value
	7:6	mode	Filter configuration for various detection modes: 0 - sensitive 1 - normal 2 - robust	1
0x01	7:0	reserved	reserved	0
0x02	7:0	tap_peak_thres[7:0]	Minimum threshold for peak detection, Resolution = 1.953mg, Range = 0 to 2000mg	45
0x03	1:0	tap_peak_thres[9:8]	see above	
	7:2	max_gesture_dur	Maximum time duration from first tap within which matching tap/s should happen to be detected as double/triple tap, Resolution = 40ms, Range = 0 to 2520 ms	16
0x04	3:0	max_dur_between_peaks	Maximum time duration within which matching peaks of tap should occur, Resolution = 5ms, Range = 0 to 75 ms	4
	7:4	tap_shock_settling_dur	Maximum duration to wait for tap shock settling, Resolution = 5ms, Range = 0 to 75 ms	6
0x05	3:0	min_quite_dur_between_taps	Minimum quite time between detection of consecutive taps of double/triple taps, Resolution = 5ms, Range = 0 to 75 ms	8
	7:4	quite_time_after_gesture	Minimum quite time between detection of 2 consecutive selected gesture, Resolution = 40ms, Range = 0 to 600 ms	6

12.3.8 Physical Sensor Control Parameters (0x0E00 – 0x0EFF)

These parameters allow the host to set or get physical sensor control information. The Sensor Control Parameters are in the range of 0x0E00 to 0x0EFF, for which the lower byte is the physical sensor ID.

The meaning and nature of this information is defined by the implementation of the specific physical sensor driver. It can have registered callback functions, which are called when this parameter is written or read. Please see the *Programmer's Manual* (Section 20, Reference 2) for details.

While the format of the data is defined by the design of the physical sensor driver, the only constraint placed on this format by the BHI385 host interface and Event-Driven Software Framework is that the first byte must contain an 8-bit code indicating the type of data to follow, and can be followed by 0 or more bytes of additional code-specific information.

The parameter is formatted as follows:

Table 86: Physical sensor control parameters

Field Name	Byte Offset	Description	Access
Parameter ID	0x00 - 0x01	Calibration state: 0x0E00 - 0x0EFF	-
Length	0x02 - 0x03	Number of bytes to follow	-
Code	0x04	Bit 0-6: Sensor control code Bit 7: Direction: 0 = write (set_sensor_ctl callback function) 1 = read (get_sensor_ctl callback function)	Read/Write
Payload (optional)	0x05 - (0x05 + Length -1)	Data to be sent to the set_sensor_ctl function or to be retrieved from the get_sensor_ctl function	Read/Write

When setting the parameter, the Set Parameter command length must be a multiple of 4 bytes and must be large enough to include the code byte and any additional data bytes.

If the direction bit of the code, bit 7, is 0, then the host is requesting that the Fuser2 pass the code and data to the set_sensor_ctl() function for the physical driver whose ID is indicated as the LSB of the parameter number.

If the direction bit is 1 in the code byte of the Set Parameter command, then the host is requesting that the next Get Parameter from the host for the same sensor ID (as indicated in the LSB of the parameter number) return the data obtained by calling the get_sensor_ctl() function after passing it the code set by the Set Parameter operation.

When getting the parameter, the Get Parameter command length should be 0 as usual. The Get Parameter Output Response's length field will indicate the number of sensor control parameter bytes that follow, including the original code requested earlier by the host in the Set Parameter command.

12.3.8.1 BHI385 Accelerometer

The accelerometer sensor can be configured by the control parameter 0x0E01.

For certain applications, it is often desirable to calibrate the offset once and to store the compensation values permanently. This can be achieved by using manual offset compensation to determine the proper compensation values and then storing these values permanently in the NVM. For manual compensation, these Accel fast offset calibration values can be overwritten through the control code by the user at any time. The compensation offsets are initialized out of the corresponding registers in the NVM during power on or reset.

Each time the device is reset, the compensation values are loaded from the non-volatile memory into the image registers and used for offset compensation.

Table 87: Accelerometer sensor control parameter format

Field Name	Control Code	Direction	Bytes	Data	Description	Default Value
Accel fast offset calibration	0x01	r/w	6	Byte0: x-axis offset LSB Byte1: x-axis offset MSB Byte2: y-axis offset LSB Byte3: y-axis offset MSB Byte4: z-axis offset LSB Byte5: z-axis offset MSB	FOC is a one-shot process that compensates errors of accel and gyro by setting compensation registers to the negated offset error. The values have a range of -128 ... 127 using two's complement notation and sign extension on the upper bits. Values outside of this range are saturated. The offset resolution is 7.8 mg and the offset range is ±1 g. Both are independent of the range setting.	n.a.

Accel low power mode	0x05	r/w	1	0x00-normal mode 0x02-low power mode	This parameter offers a trade-of of power consumption versus sensor noise. Low power mode may introduce aliasing artifacts. Low power mode falls back to normal mode at ODR > 400Hz.	0x00
Axis remapping for internal IMU features	0x07	r/w	6	Byte0: map_x_axis Byte1: map_x_axis_sign Byte2: map_y_axis Byte3: map_y_axis_sign Byte4: map_z_axis Byte5: map_z_axis_sign	The axis remapping function would not influence the IMU outputs(accel data and gyro data). It can change the sign for each axis and switch between each axis. Range for byte0, byte2 and byte4 are {0,1} Range for byte1, byte3 and byte5 are {0,1,2}	Byte0: 0x00 Byte1: 0x00 Byte2: 0x01 Byte3: 0x00 Byte4: 0x02 Byte5: 0x00
Trigger a NVM	0x08	r/w	1	For write: 0x01-Enable to trigger a NVM writing process For read: 0x00-NVM writing is done 0x01-NVM writing is in progress	Once NVM writing process is triggered, the value in backup registers value(offset/crt) will be written into NVM area.	0x00

12.3.8.2 BHI385 Gyroscope

The gyroscope sensor can be configured by the control parameter 0x0E03.

For certain applications, it is often desirable to calibrate the offset once and to store the compensation values permanently. This can be achieved by using manual offset compensation to determine the proper compensation values and then storing these values permanently in the NVM. For manual compensation, these Gyroscope fast offset calibration values can be overwritten through the control code by the user at any time. The compensation offsets are initialized out images of the corresponding registers in the NVM during power on or reset.

Each time the device is reset, the compensation values are loaded from the non-volatile memory into the image registers and used for offset compensation.

Table 88: Accelerometer sensor control parameter format

Field Name	Control Code	Direction	Bytes	Data	Description	Default Value
Gyroscope fast offset calibration	0x01	r/w	6	Byte0: x-axis offset LSB Byte1: x-axis offset MSB Byte2: y-axis offset LSB Byte3: y-axis offset MSB Byte4: z-axis offset LSB Byte5: z-axis offset MSB	FOC is a one-shot process that compensates errors of accel and gyro by setting compensation registers to the negated offset error. The values have a range of -512 ... 511 using two's complement notation and sign extension on the upper bits. Values outside of this range are saturated. The offset resolution is 61 mdps and the offset range is ±31 dps. Both are independent of the range setting.	n.a.
Gyroscope OIS enable	0x02	r/w	1	0x00-disable 0x01-enabled	Optical image stabilization can be enabled with this control code	0x00
Gyroscope fast start up enable	0x03	r/w	1	0x00-disable 0x01-enabled	When enabled, the measurement part of the sensor frontend is powered down, while the drive and digital parts remain operational. This allows a faster transition into normal mode while keeping power consumption significantly lower than in normal mode.	0x00

Field Name	Control Code	Direction	Bytes	Data	Description	Default Value
Gyroscope Component ReTrim	0x04	r/w	4	Byte 0: 0x00: Success (R) 0x01: Start(W) / Ongoing (R) 0x02: Failure (R) Byte 1-3: Read CRT Parameters (R) or combined with 0x01 for Byte 0 (W) to restore parameters.	Start by setting the data byte to 0x01. The first data byte will be read as 0x00 if CRT is successful, otherwise 0x02. The parameter bytes 1-3 are the outputs from the CRT algorithm. They can be re-written by the host after boot.	0x00
Gyroscope performance mode	0x05	r/w	1	0x00-normal mode 0x01-performance mode 0x02-low power mode	This parameter offers a trade-of of power consumption versus sensor noise. Low power mode may introduce aliasing artifacts. Low power mode falls back to normal mode at ODR > 100Hz. The parameter has to be set before enabling the sensor. Performance modes are not re-configured while the sensor is enabled.	0x00
Gyroscope timer auto trim	0x06	r/w	1	0x00-disable 0x01-start	Sync the Fuser2 timer oscillator PLL. When enabled, the frequency of the timer oscillator is referenced by gyroscope sample rate, benefiting from the high stability of the gyroscope MEMS oscillator. This feature is only applicable when gyroscope is enabled.	0x00
Trigger a NVM	0x08	r/w	1	For write: 0x01-Enable to trigger a NVM writing process For read: 0x00-NVM writing is done 0x01-NVM writing is in progress	Once NVM writing process is triggered, the value in backup registers value(offset/crt) will be written into NVM area.	0x00

Field Name	Control Code	Direction	Bytes	Data	Description	Default Value
Manual sensitivity compensa-tion	0x09	r/w	6	Byte0-1: bit[10:0]: gain update value for x-axis. Byte2-3: bit[10:0]: gain update value for y-axis Byte4-5: bit[10:0]: gain update value for z-axis.	Write the ratio of reference system report and gyro report into the gyro gain update register.These ratio values will compensate the sensitivity error.The ratio range is:0.75 ... 1.25.	0x00

12.3.8.3 BHI385 Wrist Wear Wakeup

The Wrist Wear Wakeup sensor can be configured by the control parameter 0x0E39.

Table 89: Wrist wake control

Field Name	Control Code	Direction	Pay load Bytes
Wrist wear wakeup control	0x01	r/w	10

The payload for Wrist Wear Wakeup is formatted as follows:

Table 90: Wrist wear wakeup sensor control parameter format

Field Name	Byte Offset	Description	Default Value
min angle focus	0x00 - 0x01	Cosine of minimum expected attitude change of the device within 1 second time window when moving within focus position. The parameter is scaled by 2048 i.e. 2048 * cos(angle). Range is 1024 to 1774.	0x06EE
min angle nonfocus	0x02 - 0x03	Cosine of minimum expected attitude change of the device within 1 second time window when moving from non-focus to focus position. The parameter is scaled by 2048 i.e. 2048 * cos(angle). Range is 1448 to 1856.	0x05F2
angle landscape right	0x04	Sine of the maximum allowed tilt angle in landscape right direction of the device, when it is in focus position (i.e. the user is able to comfortably look at the dial of wear device). The configuration parameter is scaled by 256 i.e. 256 * sin(angle). Range is 88 to 128.	0x80
angle landscape left	0x05	Sine of the maximum allowed tilt angle in landscape left direction of the device, when it is in focus position (i.e. the user is able to comfortably look at the dial of wear device). The configuration parameter is scaled by 256 i.e. 256 * sin(angle). Range is 88 to 128.	0x80
angle portrait down	0x06	Sine of the maximum allowed backward tilt angle in portrait down direction of the device, when it is in focus position (i.e. the user is able to comfortably look at the dial of wear device). The configuration parameter is scaled by 256 i.e. 256 * sin(angle). Range is 0 to179.	0x16
angle portrait up	0x07	Sine of the maximum allowed forward tilt angle in portrait up direction of the device, when it is in focus position (i.e. the user is able to comfortably look at the dial of wear device). The configuration parameter is scaled by 256 i.e. 256 * sin(angle). Range is 222 to 247.	0xF1

Field Name	Byte Offset	Description	Default Value
min dur moved	0x08	Minimum duration the arm should be moved while performing gesture. Range: 1 to 10, resolution = 20 ms.	0x02
min dur quite	0x09	Minimum duration the arm should be static between two consecutive gestures. Range: 1 to 10, resolution = 20 ms.	0x02

12.3.8.4 BHI385 Any Motion

The Any Motion sensor can be configured by the control parameter 0x0E23.

Table 91: Any motion

Field Name	Control Code	Direction	Pay load Bytes
Any motion control	0x01	r/w	4

The payload for Any Motion is formatted as follows:

Table 92: Payload format for any motion

Field Name	Byte Offset	Description	Default Value
Duration[7:0]	0x00	Defines the number of consecutive data points for which the threshold condition must be respected, for interrupt assertion. It is expressed in 50Hz samples (20ms). Range is 0-163sec.	Default is 5=100ms.
Duration[12:8] and axis selection	0x01	Bit[4:0] - Duration[12:8] Bit[5] - Select_X Bit[6] - Select_Y Bit[7] - Select_Z	Select_X/Y/Z: 1
Threshold[7:0]	0x02	Slope threshold value for any-motion detection. Range is 0 to 1g.	Default value is 0xAA = 83mg.
Threshold[10:8]	0x03	Bit[2:0] - Threshold[10:8] Bit[7:3] - Reserved	-

12.3.8.5 BHI385 No Motion

The No Motion sensor can be configured by the control parameter 0x0E37.

Table 93: No motion

Field Name	Control Code	Direction	Pay load Bytes
No motion control	0x01	r/w	4

The payload for No Motion is formatted as follows:

Table 94: Payload format for no motion

Field Name	Byte Offset	Description	Default Value
Duration[7:0]	0x00	Defines the number of consecutive data points for which the threshold condition must be respected, for interrupt assertion. It is expressed in 50Hz samples (20ms). Range is 0-163sec.	Default is 5=100ms.

Duration[12:8] and axis selection	0x01	Bit[4:0] - Duration[12:8] Bit[5] - Select_X Bit[6] - Select_Y Bit[7] - Select_Z	Select_X/Y/Z: 1
Threshold[7:0]	0x02	Slope threshold value for no-motion detection. Range is 0 to 1g.	Default value is 0x90 = 70mg.
Threshold[10:8]	0x03	Bit[2:0] - Threshold[10:8] Bit[7:3] - Reserved	-

12.3.8.6 BHI385 Wrist Gesture Detector

The Wrist Gesture Detector sensor can be configured by the control parameter 0x0E38.

Table 95: Wrist Gesture Detector

Field Name	Control Code	Direction	Pay load Bytes
Wrist gesture control	0x07	r/w	19

The payload for Wrist Gesture Detector is formatted as follows:

Table 96: Payload for wrist gesture detector

Field Name	Byte Offset	Description	Default Value
Min_flick_peak_y_threshold	0x0 - 0x1	Minimum threshold for flick peak on y-axis. Scaling: 0.4883, Range: 0x3E8 to 0x9C4	0x0640
Min_flick_peak_z_threshold	0x2 - 0x3	Minimum threshold for flick peak on z-axis. Scaling: 0.4883, Range: 0x1F4 to 0x5DC	0x02BC
Gravity_bounds_x_pos	0x4 - 0x5	Maximum expected value of positive gravitational acceleration on x-axis when arm is in focus pose. Scaling: 0.4883, Range: 0x0 to 0x800	0x0784
Gravity_bounds_x_neg	0x6 - 0x7	Maximum expected value of negative gravitational acceleration on x-axis when arm is in focus pose. Scaling: 0.4883, Range: 0x0 to 0xFC00	0xFC00
Gravity_bounds_y_neg	0x8 - 0x9	Maximum expected value of negative gravitational acceleration on y-axis when arm is in focus pose. Scaling: 0.4883, Range: 0x0 to 0xFC3F	0xFC3F
Gravity_bounds_z_neg	0xA - 0xB	Maximum expected value of negative gravitational acceleration on z-axis when arm is in focus pose. Scaling: 0.4883, Range: 0x800 to 0xF912	0xF912
Flick_peak_decay_coeff	0xC - 0xD	Exponential smoothing coefficient for adaptive peak threshold decay. Scaling: 0x1/0x8000, Range: 0x0 to 0x8000	0x7851
Lp_mean_filter_coeff	0xE - 0xF	Exponential smoothing coefficient for acceleration mean estimation. Scaling: 0x1/0x8000, Range: 0x0 to 0x8000	0x7D71
Max_duration_jiggle_peaks	0x10 - 0x11	Maximum duration between 2 peaks of jiggle in samples @50Hz Range: 0xA to 0x19	0x0010
Device_position	0x12	Device in left (0) or right (1) arm. By default, the wearable device is assumed to be in left arm	0x00

12.3.8.7 BHI385 Step Counter

The Step Counter sensor can be configured by the control parameter 0x0E20.

Table 97: Step Counter

Field Name	Control Code	Direction	Pay load Bytes
Step counter control	0x01	r/w	54

The payload for Step Counter is formatted as follows:

Table 98: Payload for step counter

Field Name	Byte Offset	Description	Default Value (Wearable)	Default Value (Hearable)	Minimum Resolution	Range
env_min_dist_up	0x0 - 0x1	minimum distance for top envelope to be away from long-term mean value	301	307	10 bit	[0...0.5] * 2048
env_coef_up	0x2 - 0x3	first order IIR filter coefficient for top side envelope decay towards mean value.	31700	30966	16 bit	
env_min_dist_down	0x4 - 0x5	minimum distance for bottom envelope to be away from long-term mean value	315	133	10 bit	[0...0.5] * 2048
env_coef_down	0x6 - 0x7	first order IIR filter coefficient for bottom side envelope decay towards mean value.	31451	27853	16 bit	
step_buffer_size	0x8 - 0x9	number of step to buffer before adding the steps to the count	4	7	3-4 bit	[4-10]
mean_val_decay	0xA - 0xB	first order IIR filter coefficient to get the mean acceleration value,was designed for 25 Hz data and needs to be scaled for other rates	31551	31130	16 bit	
mean_step_dur	0xC - 0xD	Decay constant for adapting the mean step duration	27853	29491	16 bit	
filter_coeff_b2	0xE - 0xF	filter coefficient for a 2nd order low-pass filter	1219	1374	16 bit	
filter_coeff_b1	0x10 - 0x11	filter coefficient for a 2nd order low-pass filter	2437	2749	16 bit	
filter_coeff_b0	0x12 - 0x13	filter coefficient for a 2nd order low-pass filter	1219	1374	16 bit	
filter_coeff_a2	0x14 - 0x15	filter coefficient for a 2nd order low-pass filter	-6420	0xE896	16 bit	

Field Name	Byte Offset	Description	Default Value (Wearable)	Default Value (Hearable)	Minimum Resolution	Range
filter_coeff_a1	0x16 - 0x17	filter coefficient for a 2nd order low-pass filter	17932	0x41EE	16 bit	
filter_cascade_enabled	0x18 - 0x19	flag indicating if the filter cascade shall be activated	1	1	1 bit	0/1
peak_duration_min_walking	0x1A - 0x1B	minimum duration of a peak for walking	39	13	4 bit	[0.15...0.80] * 50
peak_duration_min_running	0x1C - 0x1D	minimum duration of a peak for running	25	12	4 bit	[0.15...0.50] * 50
step_duration_max	0x1E - 0x1F	maximum duration of a step	150	74	4 bit	[1...3] * 50
step_duration_window	0x20 - 0x21	time window for detecting steps if a step was missed	160	160	4 bit	[0..0.3] * 50 * 16
half_step_enabled	0x22 - 0x23	flag to activate half step feature	1	0	1 bit	0/1
activity_detection_factor	0x24 - 0x25	factor for activity detection	12	12	4 bit	[0...15]
activity_detection_threshold	0x26 - 0x27	activity detection threshold	15600	15600	16 bit	[1000...1500] * 12
step_counter_increment	0x28 - 0x29	scaling factor for step count	256	256	16 bit	[0.8...1.2] * 256
step_duration_pp_enabled	0x2A - 0x2B	flag to activate step duration ratio based post-processing	1	1	1 bit	0/1
stepDurationThres	0x2C - 0x2D	step duration ratio threshold	3	3	3 bit	[1-9]
enableMcrPostProc	0x2E - 0x2F	flag to activate mean crossing rate based post-processing	1	1	1 bit	0/1
mcrThres	0x30 - 0x31	mean crossing rate threshold	14	8	4 bit	[10-20]

12.3.8.8 Barometric Pressure Sensor

The configuration of the barometric pressure sensor depends on the type of sensor. Currently Bosch Sensortec BMP390 and BMP581 are supported and selected via the appropriate firmware image. Depending on the selected sensor, different configuration parameter structures have to be applied.

The BMP390 sensor has a configuration payload of 2 bytes:

Table 99: BMP390 Configuration Parameter Payload

Field Name	Control Code	Direction	Payload Bytes
Pressure sensor control	0x01	r/w	2

The BMP390 parameter payload is defined as follows. Please refer to the datasheet of the BMP390 for details of these configurations.

Table 100: BMP390 Configuration Payload

Byte Offset	Bits	Field Name	Description	Default Value
0x00	2:0	osr_p	Oversampling setting pressure measurement 000b - no oversampling 001b - x2 oversampling 010b - x4 oversampling 011b - x8 oversampling 100b - x16 oversampling 101b - x32 oversampling	100b
	5:3	osr_t	Oversampling setting temperature measurement 000b - no oversampling 001b - x2 oversampling 010b - x4 oversampling 011b - x8 oversampling 100b - x16 oversampling 101b - x32 oversampling	001b
	7:6	reserved_1	reserved	0
0x01	2:0	iir_filter	Filter coefficient for IIR filter 000b - filter coefficient is 0 -> bypass mode 001b - filter coefficient is 1 010b - filter coefficient is 3 011b - filter coefficient is 7 100b - filter coefficient is 15 101b - filter coefficient is 31 110b - filter coefficient is 63 111b - filter coefficient is 127	000b
	7:3	reserved_2	reserved	0

The BMP581 sensor has a configuration payload of 3 bytes:

Table 101: BMP581 Configuration Parameter Payload

Field Name	Control Code	Direction	Payload Bytes
Pressure sensor control	0x01	r/w	3

The BMP581 parameter payload is defined as follows. Please refer to the datasheet of the BMP581 for details of these configurations.

Table 102: BMP581 Configuration Payload

Byte Offset	Bits	Field Name	Description	Default Value
0x00	2:0	osr_p	Oversampling setting pressure measurement 000b - no oversampling 001b - x2 oversampling 010b - x4 oversampling 011b - x8 oversampling 100b - x16 oversampling 101b - x32 oversampling	100b

Byte Offset	Bits	Field Name	Description	Default Value
	5:3	osr_t	Oversampling setting temperature measurement 000b - no oversampling 001b - x2 oversampling 010b - x4 oversampling 011b - x8 oversampling 100b - x16 oversampling 101b - x32 oversampling	001b
	7:6	reserved_1	reserved	0
0x01	2:0	iir_filter_p	Filter coefficient for pressure IIR filter LPF band filter 000b - filter coefficient is 0 -> bypass mode 001b - filter coefficient is 1 010b - filter coefficient is 3 011b - filter coefficient is 7 100b - filter coefficient is 15 101b - filter coefficient is 31 110b - filter coefficient is 63 111b - filter coefficient is 127	001b
	5:3	iir_filter_t	Filter coefficient for temperature IIR filter LPF band filter 000b - filter coefficient is 0 -> bypass mode 001b - filter coefficient is 1 010b - filter coefficient is 3 011b - filter coefficient is 7 100b - filter coefficient is 15 101b - filter coefficient is 31 110b - filter coefficient is 63 111b - filter coefficient is 127	001b
	7:6	reserved_2	reserved	0
0x02	7:0	dsp_config	DSP configuration. Refer to register DSP_CONFIG(0x30) in BMP581 datasheet.	-

12.3.9 Activity Parameters (0x0F00 – 0x0FFF)

The parameters allow the host to control the configurations of the Activity algorithm. In the BHI385 , there are two kinds of Activity Recognition algorithms: Wearable and Hearable. They use different parameter formats. For the specific content format of each parameter, see Table 103 and Table 104.

Table 103: Parameters of Activity Recognition Algorithm - Wearable

Field Name	Byte Offset	Description	Relation	Range	Default Value
postProcessingEnable	0x00 - 0x01	Enable/disable post processing of the activity detected by the classifier. Post processing improves the robustness of outputs and decreases the fluctuations in the output. Type in uint16_t.	Enabling increases the accuracy and la-tency.Disabling decreases the accuracy and latency.	0 to 1 0: Disable 1: Enable	1

Field Name	Byte Offset	Description	Relation	Range	Default Value
minGdiThreshold	0x02 - 0x03	Minimum threshold for the Gini's diversity index (GDI) to accept and add the activity detected by the classifier to the activity buffer. Type in uint16_t.	The greater the value, the lower the accuracy and latency.	0 to 4095	1761
maxGdiThreshold	0x04 - 0x05	Maximum threshold for the Gini's diversity index (GDI) to reject the activity detected by the classifier. Type in uint16_t.	The greater the value, the lower the accuracy.	0 to 4095	2662
activityOutBufferSize	0x06 - 0x07	Buffer size for post processing of the activity detected by the classifier. To prevent noisy recognition of activity, mode of the activity buffer is the output by the algorithm as the predicted activity. Type in uint16_t.	The lower the value, the noisier the activity output, but the lower the latency.	1 to 10	10
minSegModerateConf	0x08 - 0x09	Minimum segments classified with moderate confidence as they belong to a certain activity type to be added to the activity buffer. Type in uint16_t.	The lower the value, the noisier the activity output, but the lower the latency.	1 to 10	10
Reserved	0xA - 0xB	Reserved	Reserved	Reserved	0

Table 104: Parameters of Activity Recognition Algorithm - Hearable

Field Name	Byte Offset	Description	Relation	Range	Default Value
segmentSize	0x00 - 0x01	Static segment size over which predictors are computed for activity classification. Type in uint16_t.	As the segment size increases, the accuracy and latency also increase.	0 to 2 0: 5.12 secs 1: 2.56 secs 2: 1.28 secs	0
postProcessingEnable	0x02 - 0x03	Enable/disable post processing of the activity detected by the classifier. Post processing improves the robustness of outputs and decreases the fluctuations in the output. Type in uint16_t.	Enabling increases the accuracy and latency. Disabling decreases the accuracy and latency.	0 to 1 0: Disable 1: Enable	1

Field Name	Byte Offset	Description	Relation	Range	Default Value
minGdiThreshold	0x04 - 0x05	Minimum threshold for the Gini's diversity index (GDI) to accept and add the activity detected by the classifier to activity buffer. Type in uint16_t.	The greater the value, the lower the accuracy and latency.	0 to 4095	1761
maxGdiThreshold	0x06 - 0x7	Maximum threshold of the Gini's diversity index (GDI) to reject the activity detected by the classifier. Type in uint16_t.	The greater the value, the lower the accuracy.	0 to 4095	2662
activityOutBufferSize	0x8 - 0x9	Buffer size for post processing of the activity detected by the classifier. To prevent noisy recognition of activity, mode of the activity buffer is the output by the algorithm as the predicted activity. Type in uint16_t.	The lower the value, the noisier the activity output, but the lower the latency. When the value is 1, the algorithm has the worst performance and normal activities may not be detected.	1 to 10	10
minSegModerateConf	0xA - 0xB	Minimum segments classified with moderate confidence as they belong to a certain activity type to be added to the activity buffer. Type in uint16_t.	The lower the value, the noisier the activity output, but the lower the latency.	1 to 10	10

12.4 Command Error Response

This command response will be returned by the bootloader and main firmware when there is a problem with the received command.

If Error Value register (see Table 30) is 0xC0 (Command Error) or 0xC1 (Command Too Long), the following registers are also updated:

- Error Aux Register (see Section 11.1.26) = Command error value (see below)
- Debug Value Register (see Section 11.1.26)= Command ID in error (least significant byte)

Table 105: Command error response

Field Name	Byte Offset	Description

[1]The input buffer size is 128 bytes in bootloader and 1024 bytes when the Event-Driven Software Framework is used.

Field Name	Byte Offset	Description	
Status Code	0x00 - 0x01	Command Error = 0x000F	
Length	0x02 - 0x03	4	
Command	0x04 - 0x05	The command ID with the error	
Error	0x06	**Value**	**Command Error**
		0x01	Incorrect Length
		0x02	Too Long (length specified is longer than input buffer[1]; recover by issuing an Abort Transfer on Channel 0)
		0x03	Parameter Write Error (incorrect page or unhandled parameter number or length provided is too short)
		0x04	Parameter Read Error (parameter size too big for output buffer, or invalid page or parameter specified)
		0x05	Invalid Command
		0x06	Invalid Parameter
		0xFF	Command Failed (did not complete successfully)
Reserved	0x07	Unused	

13 FIFO Data Formats

When the host retrieves data from a FIFO by reading that FIFO's output stream, it will receive blocks of sensor data encapsulated by a FIFO Descriptor. This descriptor consists of the number of bytes to follow in this transfer and a Small Delta Timestamp with the delta set to 0, which is required to make the FIFO Descriptor length compatible with BHI385's DMA system.

Following the FIFO Descriptor, there will be one or more FIFO Blocks; the total size of data sent (including the initial small delta timestamp) will equal the FIFO Transfer Length field in the FIFO Descriptor. The maximum size of a single transfer is limited to the 16-bits of Transfer Length.

Each FIFO Block begins with a FIFO Block Header. This contains a Meta Event and a full 40-bit timestamp. The Meta Event will be a "spacer" Meta Event, with the two payload bytes set to a running 16-bit block count (the host can ignore this), unless one or more previous FIFO blocks were discarded due to a FIFO overflow, in which case the Meta Event will be a FIFO Overflow Meta Event. The full timestamp is provided (see section 12.2.9 Control FIFO Format Command[1)]) to guarantee that the host can always know the proper time for a block's sensor data regardless of any preceding FIFO overflows.

When more than one FIFO Blocks are sent in a transfer, all but the last one will be filled out to the full block size, which is 512 bytes including the Block Header, with one or more single byte Filler sensor IDs (0xFF). The host should ignore these filler bytes and continue parsing. The last (or only) FIFO block will often not be completely full of packed sensor data. In this case the packed sensor data will be padded out to the next 32-bit boundary with single byte Padding sensor IDs (0x00).

Table 106: FIFO Data Format

FIFO Descriptor		**Header at Start of each FIFO Block**		**Contents of each FIFO block**		**0 or more additional FIFO Block Headers and FIFO Block Contents**	
FIFO Transfer Length (16 bits)	Small Delta Timestamp (16 bits) – always 0	Meta Event (Spacer or Over-flow)(32 bits)	Full Times-tamp (48 bits)	Packed Sensor Data	Filler Bytes (0xFF)	...	0 – 3 Pad Bytes (0x00)

The format of the packed sensor data is described in Section 14 FIFO Data Types. All multi-byte fields are little-endian.

13.1 Wake-Up and Non-Wake-Up FIFO

These FIFOs always comply with the format described in Section 15 above.

13.2 Status and Debug FIFO

The Status and Debug FIFO (Output Channel 3) has two operating modes: the synchronous mode and the asynchronous mode. The operating mode is controlled by the Async Status Channel bit of the Host Interface Control (0x06) register. Each mode has a corresponding bit in the Interrupt Status register 0x2D (Section 11.1.24):

- Bit 5 indicates a synchronous status packet is available (as result of a command)
- Bit 6 indicates asynchronous data is available (as result of an error, debug output, or command)

Further, each bit can be masked using the Host Interrupt Control register 0x07 (Section 11.1.8):

[1] If the host uses the Control FIFO Format Command to suppress the full timestamp in the header, then only a 4 byte Meta Event will be present at the start of each FIFO block. In this case, the proper full timestamp at which a FIFO overflow condition occurs will not be available. It is recommended that this feature is only be used when FIFO overflows cannot occur.

- Bit 2, if set, masks off (prevents) the synchronous status packet interrupt
- Bit 3, if set, masks off (prevents) the asynchronous debug FIFO data interrupt

When the host is about to transmit a command, it is recommended to clear the Async Status Channel bit of the Host Interface Control register 0x06 (Section 11.1.7), to place the channel in synchronous mode. While in this mode, any asynchronous data will be stored in a memory buffer. Once the command is completed and any associated command response has been received, the host may switch back to asynchronous mode. The host may mask off the “Status Available” Interrupt to not be alerted once the command response is available in synchronous mode by setting the bit 2 in the Host Interrupt Control register 0x07 (Section 11.1.8).

13.2.1 Synchronous Mode

In synchronous mode, the FIFO format described in Table table 106: FIFO Data format will NOT apply to data read from the Status and Debug FIFO. The only data that will appear in the Status and Debug FIFO (Output Channel 3) will be command responses as defined throughout Section 12 Host Interface Commands.

While in this mode, asynchronous status and debug messages will be queued in the FIFO in the background. These data will be transferred to the host once it switches the Status and Debug FIFO to Asynchronous mode. Presence of data will then be signaled via the Host Interrupt pin and Interrupt Status register.

13.2.2 Asynchronous Mode

In Asynchronous mode, the FIFO format described in Table table 106: FIFO Data format will apply to data read from the Status and Debug FIFO. The content of the FIFO is the following:

- Sensor Error and System Error Meta events (see Table table 30: Error Value Register (0x2E))
- Debug events (Post Mortem Data)
- Command responses

The host has to parse the FIFO content to find the response corresponding to a recent command.

14 FIFO Data Types and Format

The FIFO events listed below depend on how different sensors are connected and combined, as well as whether the corresponding sensor firmware has integrated the relevant features. Before interacting with any of them, please refer to Section 12.3.2.5 to verify the specific virtual sensors supported by the firmware.

Table 107 describes all available types of events which can occur in the FIFO and their characteristics.

Table 107: Overview of FIFO event IDs

FIFO Event Type	FIFO Event	ID (Non-wake-up)	ID (Wake-up)	Sensor Payload[1]	Scale Factor	Bytes in FIFO	Reporting mode	Requires external sensor[2]
Virtual Sensor Event	Rotation Vector	34	35	Quaternion+	Defined by format "Quaternion+"	11	Continuous	BMM150, BMM350
	Game Rotation Vector	37	38	Quaternion+	Defined by format "Quaternion+"	11	Continuous	-
	Geomagnetic Rotation Vector	40	41	Quaternion+	Defined by format "Quaternion+"	11	Continuous	BMM150, BMM350
	Orientation	43	44	Euler	Defined by format "Euler"	7	Continuous	BMM150, BMM350
	Accelerometer Passthrough	1	-	3D Vector	Defined by format "Accelerometer"	7	Continuous	-
	Gyroscope Passthrough	10	-	3D Vector	Defined by format "Gyroscope"	7	Continuous	-
	Magnetometer Passthrough	19	-	3D Vector	Defined by format "Magnetometer"	7	Continuous	BMM150, BMM350
	Accelerometer Corrected	4	6	3D Vector	Defined by format "Accelerometer"	7	Continuous	-
	Magnetometer Corrected	22	24	3D Vector	Defined by format "Magnetometer"	7	Continuous	BMM150, BMM350
	Gyroscope Corrected	13	15	3D Vector	Defined by format "Gyroscope"	7	Continuous	-
	Gravity	28	29	3D Vector	Defined by format "Accelerometer"	7	Continuous	-
	Linear Acceleration	31	32	3D Vector	Defined by format "Accelerometer"	7	Continuous	-

FIFO Event Type	FIFO Event	ID (Non-wake-up)	ID (Wake-up)	Sensor Payload[1]	Scale Factor	Bytes in FIFO	Reporting mode	Requires external sensor[2]
	Raw Accelerometer	3	7	3D Vector	Defined by format "Accelerometer"	7	Continuous	-
	Raw Magnetometer	21	25	3D Vector	Defined by format "Magnetometer"	7	Continuous	BMM150, BMM350
	Raw Gyroscope	12	16	3D Vector	Defined by format "Gyroscope"	7	Continuous	-
	Accelerometer Offset	5	-	3D Vector	Defined by format "Accelerometer"	7	Continuous	-
	Magnetometer Offset	23	-	3D Vector	Defined by format "Magnetometer"	7	Continuous	BMM150, BMM350
	Gyroscope Offset	14	-	3D Vector	Defined by format "Gyroscope"	7	Continuous	-
	Humidity	130	134	8-bit unsigned integer	1%RH	2	On-change	BME688
	Step Counter, low power	136	139	32-bit unsigned integer	1 step	5	On-change	-
	Temperature	128	132	16-bit signed integer	°C / 100 (range: -4000 to 8500)	3	On-change	BMP390, BMP581, BME688
	Barometer	129	133	24-bit unsigned integer	1/128 Pa	4	Continuous	BMP390, BMP581, BME688
	Gas	131	135	32-bit unsigned integer	1 Ohms gas sensor resistance	5	Continuous	BME688
	Step Detector, low power	137	140	Event (none)	n.a.	1	Special	-
	Multi-Tap Detector	153	-	Multi-Tap Detector Data	n.a.	3	On-change	-
	Activity recognition for wearables	-	154	Activity Data	n.a.	3	On-change	-
	No Motion, low power	-	159	Event (none)	n.a.	1	One-shot	-
	Any Motion, low power	-	143	Event (none)	n.a.	1	One-shot	-
	Wrist wear wake-up, low power	-	158	Event (none)	n.a.	1	On-change	-

FIFO Event Type	FIFO Event	ID (Non-wake-up)	ID (Wake-up)	Sensor Payload[1]	Scale Factor	Bytes in FIFO	Reporting mode	Requires external sensor[2]
	Wrist gesture detector, low power	-	156	Wrist Gesture Detector Data	n.a.	2	On-change	-
	Self-learning AI Data	112	-	Self-Learning AI data	n.a.	11	On-change	-
	Motion AI Sensor 1	170	-	Motion AI Sensor Data	n.a.	2	On-change	-
	Motion AI Sensor 2	171	-	Motion AI Sensor Data	n.a.	2	On-change	-
	Motion AI Sensor 3	172	-	Motion AI Sensor Data	n.a.	2	On-change	-
	Motion AI Sensor 4	173	-	Motion AI Sensor Data	n.a.	2	On-change	-
Debug Data Event	Debug Data	250	-	Binary or string data (fwrite or printf)	n.a.	18	n.a.	-
Timesta-mp Event	Timestamp Small Delta	251	245	8-bit integer; incremental change from previous	1/64000 seconds	2	n.a.	-
	Timestamp Large Delta	252	246	16-bit integer; incremental change from previous	1/64000 seconds	3	n.a.	-
	Full Timestamp	253	247	40-bit unsigned integer; wraps every 198 days	1/64000 seconds	6	n.a.	-
Meta Event	Meta Events	254	248	Meta Event	n.a.	4	n.a.	-
Filler	Filler [4]	255	255	n.a.	n.a.	1	n.a.	-
Padding	Padding [5]	0	0	n.a.	n.a.	1	n.a.	-

14.1 Format of Virtual Sensor Events

In general, every virtual sensor event in the FIFO data consists of a 1-Byte Sensor ID and a multi-byte payload. The size of the virtual sensor event is fixed per Sensor ID as described by the column “Bytes in FIFO” in the Table 107: Overview

[1] See section below for definition of sensor value formats.
[2] Other physical sensor can be supported by creating a dedicated physical driver for the sensor using the SDK (Software Development Kit). See References for more information.
[3] Dynamic: scaled to current dynamic range of sensor.
[4] Used to space FIFO blocks to even boundary; host should ignore but continue parsing.
[5] Optionally used to mark end of a FIFO read; host should stop parsing here.

of FIFO event IDs shown above. This value includes the Sensor ID byte.

For some of the Sensor IDs, the format and scaling of the payload (e.g. “1 bit unsigned integer” and “1%RW” for Humidity”) is described in the same Table 107: Overview of FIFO event IDs shown above. For Sensor IDs with complex payload the format is described in the following sections.

14.1.1 Format “Accelerometer”

For the five Accelerometer virtual sensors (Accelerometer Passthrough, Accelerometer Corrected, Linear Acceleration, Raw Accelerometer and Accelerometer Offset), the “Accelerometer” format is used:

Table 108: Virtual sensor event format “Accelerometer”

Table 108: Virtual sensor event format “Accelerometer”

Byte Number	**Contents**	**Format**
0	Sensor ID (Rotation Vector, etc.)	8-bit unsigned, see Table : 107: Overview of FIFO event IDs.
1 .. 2	Acceleration on the X-axis in g-s	Signed 16-bit fixed point integer, least significant byte first, S_{4g}: scaled by 2^{-14} S_{8g}: scaled by 2^{-13} S_{16g}: scaled by 2^{-12} S_{32g}[1)]: scaled by 2^{-11}
3 .. 4	Acceleration on the Y-axis in g-s	Signed 16-bit fixed point integer, least significant byte first, S_{4g}: scaled by 2^{-14} S_{8g}: scaled by 2^{-13} S_{16g}: scaled by 2^{-12} S_{32g}[1)]: scaled by 2^{-11}
5 .. 6	Acceleration on the Z-axis in g-s	Signed 16-bit fixed point integer, least significant byte first, S_{4g}: scaled by 2^{-14} S_{8g}: scaled by 2^{-13} S_{16g}: scaled by 2^{-12} S_{32g}[1)]: scaled by 2^{-11}

14.1.2 Format “Gyroscope”

For the four Gyroscope virtual sensors (Gyroscope Passthrough, Gyroscope Corrected, Raw Gyroscope and Gyroscope Offset), the “Gyroscope” format is used:

Table 109: Virtual sensor event format “Gyroscope”

[1] When the sensor is configured for the 32g range, its output may not change with accelerations exceeding the limits in Table 131: Operating conditions accelerometer. The maximum measurable acceleration is typically +-28g.

Table 109: Virtual sensor event format “Gyroscope”

Byte Number	Contents	Format
0	Sensor ID (Rotation Vector, etc.)	8-bit unsigned, see Table : 107: Overview of FIFO event IDs.
1 .. 2	Angular velocity around the X-axis in ° / s	Signed 16-bit fixed point integer, least significant byte first, R_{FS2000}: scaled by $1000/2^{14}$ R_{FS1000}: scaled by $1000/2^{15}$ R_{FS500}: scaled by $1000/2^{16}$ R_{FS250}: scaled by $1000/2^{17}$ R_{FS125}: scaled by $1000/2^{18}$
3 .. 4	Angular velocity around the Y-axis in ° / s	Signed 16-bit fixed point integer, least significant byte first, R_{FS2000}: scaled by $1000/2^{14}$ R_{FS1000}: scaled by $1000/2^{15}$ R_{FS500}: scaled by $1000/2^{16}$ R_{FS250}: scaled by $1000/2^{17}$ R_{FS125}: scaled by $1000/2^{18}$
5 .. 6	Angular velocity around the Z-axis in ° / s	Signed 16-bit fixed point integer, least significant byte first, R_{FS2000}: scaled by $1000/2^{14}$ R_{FS1000}: scaled by $1000/2^{15}$ R_{FS500}: scaled by $1000/2^{16}$ R_{FS250}: scaled by $1000/2^{17}$ R_{FS125}: scaled by $1000/2^{18}$

14.1.3 Format “Magnetometer”

For the four Magnetometer virtual sensors (Magnetometer Passthrough, Magnetometer Corrected, Raw Magnetometer and Magnetometer Offset), the “Magnetometer” format is used:

Table 110: Virtual sensor event format “Magnetometer”

Table 110: Virtual sensor event format “Magnetometer”

Byte Number	Contents	Format
0	Sensor ID (Rotation Vector, etc.)	8-bit unsigned, see Table : 107: Overview of FIFO event IDs.
1 .. 2	Magnetic field strength on the X-axis in µT	Signed 16-bit fixed point integer, least significant byte first BMM150 R_{FS2500}: scaled by $2500/2^{15}$ BMM350 R_{FS2000}: scaled by $2000/2^{15}$
3 .. 4	Magnetic field strength on the Y-axis in µT	Signed 16-bit fixed point integer, least significant byte first BMM150 R_{FS2500}: scaled by $2500/2^{15}$ BMM350 R_{FS2000}: scaled by $2000/2^{15}$
5 .. 6	Magnetic field strength on the Z-axis in µT	Signed 16-bit fixed point integer, least significant byte first BMM150 R_{FS2500}: scaled by $2500/2^{15}$ BMM350 R_{FS2000}: scaled by $2000/2^{15}$

14.1.4 Format “Quaternion+”

For the three rotation vector virtual sensors (Rotation Vector, Game Rotation Vector, and Geomagnetic Rotation Vector), the “Quaternion+” format is used:

Table 111: Virtual sensor event format “Quaternion+”

Table 111: Virtual sensor event format “Quaternion+”

Byte Number	Contents	Format
0	Sensor ID (Rotation Vector, etc.)	8-bit unsigned, see Table : 107: Overview of FIFO event IDs.
1 .. 2	X component of quaternion	Signed 16-bit fixed point integer, least significant byte first, scaled by 2^{-14}
3 .. 4	Y component of quaternion	Signed 16-bit fixed point integer, least significant byte first, scaled by 2^{-14}
5 .. 6	Z component of quaternion	Signed 16-bit fixed point integer, least significant byte first, scaled by 2^{-14}
7 .. 8	W component of quaternion	Signed 16-bit fixed point integer, least significant byte first, scaled by 2^{-14}
9 .. 10	Estimated accuracy in radians	Unsigned 16-bit fixed point integer, least significant byte first, scaled by 2^{-14}

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

14.1.5 Format “Euler”

The Orientation Sensor outputs the device orientation as Euler angles: heading, pitch, and roll.

Table 112: Virtual sensor event format “Euler”

Table 112: Virtual sensor event format “Euler”

Byte Number	Contents	Format
0	Sensor ID (Orientation)	8-bit unsigned, see Table : 107: Overview of FIFO event IDs.
1 ... 2	Heading	Signed 16-bit fixed point integer, least significant byte first, scaled by (360° / 2^{15})
3 ... 4	Pitch	Signed 16-bit fixed point integer, least significant byte first, scaled by (360° / 2^{15})
5 ... 6	Roll	Signed 16-bit fixed point integer, least significant byte first, scaled by (360° / 2^{15})

14.1.6 Format “3D Vector”

For the many 3 axis sensors (e.g. Accelerometer, Magnetometer, Gyroscope, Gravity), the following layout is used. The scale factor of the values depends on the type of the sensor and optionally on the range setting. Please check Table 107for details.

Table 113: Virtual sensor event format “3D Vector”

Byte Number	Contents	Format
0	Sensor ID	8-bit unsigned, see Table 107: Overview of FIFO event IDs.
1 ... 2	X	Signed 16-bit fixed point integer, least significant byte first
3 ... 4	Y	Signed 16-bit fixed point integer, least significant byte first
5 ... 6	Z	Signed 16-bit fixed point integer, least significant byte first

14.1.7 Format “Activity Data”

The activity sensor outputs a sensor whenever there is a change detected in activity. In the payload of the virtual sensor, event bits are provided to indicate start the onset of an activity and the end of it. A bit value of 1 indicates the change of the activity (start or end), while a value of 0 indicates no change.

Table 114: Virtual sensor event format “Activity”

Byte Number	Contents	Format
0	Sensor ID	8-bit unsigned, see Table 107: Overview of FIFO event IDs.
1 ... 2	Activity change bitmap	16-bit field, least significant byte first

Table 115: Bitmap of activities

Bit Number	Contents
0	Still activity ended
1	Walking activity ended
2	Running activity ended
3	Biking activity ended
4	In-vehicle ended
5 ... 7	Reserved
8	Still activity started
9	Walking activity started
10	Running activity started
11	Biking activity started
12	In-vehicle started
13 ... 15	Reserved

14.1.8 Format of “Self-Learning AI Data”

The Self-Learning AI Software outputs sensor events whenever the learning state or recognition state changes, for example, when a new activity has been learned or when an already learned activity has been recognized.

Table 116: Virtual sensor event format “Self-learning AI data”

Byte Number		Contents	Format
0		Sensor ID	8-bit unsigned, see Table 107: Overview of FIFO event IDs.
1	Learning	Reserved	
2		Index	8-bit signed.
3		Progress	8-bit unsigned.
4		Change Reason	8-bit unsigned.
5	Recognition	Reserved	
6		Index	8-bit unsigned.
7 ... 10		Count	IEEE754 single precision float
11 ... 14		Score	IEEE754 single precision float

Field	Description	
Learning index	A value of -1 means no new learning has occurred. If the value is >= 0, then a new pattern has been learned, and reading of this pattern may be performed.	
Learning Progress	While learning a new pattern, this field counts from 0 to 5. When 5 is reached a new pattern will be learned. If learning is interrupted, this progress will return to 0, and change reason will be set to indicate why learning was interrupted.	
Learning Change Reason	0	Learning is progressing.
	1	Learning was interrupted by a non-repetitive activity.
	2	Learning was interrupted because no significant movement was detected.
Recognition Index	The index of the recognized activity. 255 means no activity was recognized.	
Recognition Count	The current repetition count of the recognized activity.	
Recognition Score	The current score of the recognized activity.	

14.1.9 Format of “Motion AI Sensor Data”

Table 117: Virtual sensor event format “Motion AI Sensor Data”

Byte Number	Contents	Format	Range
0	SENSOR ID	8-bit unsigned int	170 ... 173
1	Movement Class: Each value represents the movement class that the sensor recognized.	8-bit unsigned int	1 ... 255

14.1.10 Format of “Multi-Tap Detector Data”

Table 118: Multi-tap FIFO data format

Byte Number	Contents	Format	Range
0	SENSOR ID	8-bit unsigned int	-
1	Taps detected: bit 0 single tap; bit 1 double taps; bit 2 triple taps	8-bit unsigned int	0...7

14.1.11 Format of “Wrist Gesture Detector Data”

Table 119: Wrist gesture detector data format

Byte Number	Contents	Format	Range
0	SENSOR ID	8-bit unsigned int	-
1	Output value: 0: Unknown gesture 3: Wrist shake/jiggle 4: Arm flick in gesture 5: Arm flick out gesture	8-bit unsigned int	-

14.1.12 Format of Scalar Data

Many virtual sensors report a single signed or unsigned value. Currently, sensors with a payload of 1 to 5 bytes exist, corresponding to 8 to 40-bit signed or unsigned integer values. The format of these events is:

Table 120: Virtual sensor event format for scalar data

Byte Number	Contents	Format
0	Sensor ID	8-bit unsigned, see Table 107: Overview of FIFO event IDs.
1 ... N	Scalar data	8 to 40 bits of signed or unsigned integer data, least significant byte first.

14.1.13 Format of Sensors Without Payload

Some virtual sensors generate no payload data, i.e. the virtual sensor event notifies the occurrence of the event. The format of these events consists of the Sensor ID only:

Table 121: Virtual sensor event format with no payload

Byte Number	Contents	Format
0	Sensor ID	8-bit unsigned, see Table 107: Overview of FIFO event IDs.

14.2 Retrieving Timestamps of Virtual Sensor Events

Every virtual sensor event has a timestamp associated to it, indicating at which time the event has occurred. The timestamps are not part of the event payload but transferred as separate events. This allows for multiple virtual sensor events having the same timestamp, transferring the timestamp only once in order to save FIFO space.

As a general rule, if a timestamp is transferred, it is valid for all following virtual sensor events, until the next timestamp is transferred.

The format of a timestamp is a 40-bit unsigned value with a resolution of typical 1/64000 second, starting from 0 when the system has booted. The timestamp wraps around after approximately 198 days of continuous operation.

In order the save further FIFO space, 3 different types of timestamp events have been defined, one (“Full Timestamp”) providing the full 40-bit absolute timestamp, while the other two (“Timestamp Small delta”, “Timestamp Large Delta”) provide an 8-bit or 16-bit increment to the previous timestamp.

See Table 107: Overview of FIFO event IDs for the definition of the timestamps and their format.

14.3 Format of Meta Events

Meta Events indicate asynchronous, low periodicity events. They can be individually enabled or disabled with the Meta Event Control (0x0101, 0x0102) Parameter (Section 12.3.2.1).

Meta Events can also be configured to trigger or cancel a host interrupt whenever the Meta Event occur. In this case the host interrupt would be issued even if there were no pending virtual sensor events in the FIFOs.

The “Initialized” Meta Event will be the first event in the FIFO (following the current timestamp) after initialization. After the host receives this, it is safe to send Host Commands, for example, to configure the device or to enable virtual sensors.

There are two types of events: System Meta Events, and Meta Events related to a specific physical sensor.

System Meta Events will be placed in the output channel 3 (Status and Debug FIFO). These Meta Events are Error and Sensor Error. These Meta Events are enabled by default and wake up the AP by default.

Meta events related to a specific physical sensor (generated as a result of a sensor configuration request from the host) such as the Sample Rate Changed, Power Mode Changed, and Dynamic Range Changed, are always sent to the appropriate FIFO. For example, if the host requests to turn on the wake-up accelerometer, and it was not enabled before this, all three events (if enabled and the state changes) may be generated in the wake-up FIFO.

Table 122: Overview of meta events

	Meta Event Type	**Event-Specific Values**		**Wake-up FIFO**		**Non-wake-up FIFO**		**Status FIFO**	
Name	**Byte 1**	**Byte 2**	**Byte 3**	**Default Enable State**	**Default Int Enable State**	**Default Enable State**	**Default Int Enable State**	**Default Enable State**	**Default Int Enable State**
Not used	0								
Flush Complete	1	Sensor Type from FIFO_ FLUSH register	Not used	Enabled		Enabled			
Sample Rate Changed	2	Sensor Type	Sample Rate	Enabled		Enabled			
Power Mode Changed	3	Sensor Type	Power Mode	Enabled		Enabled			
System Error	4	Error Register	Interrupt State Register					Enabled	Enabled
Algorithm Events	5	Sub Event	Event Payload	Enabled		Enabled			
Sensor Status	6	Sensor Type	Status	Enabled		Enabled			
Reserved	7								
Reserved	8								
Reserved	9								
Reserved	10								
Sensor Error	11	Sensor Type	Error Register					Enabled	Enabled
FIFO Overflow	12	Loss Count LSB	Loss Count MSB	Enabled[1]		Enabled[1]			

[1] These Meta Events cannot be disabled.

	Meta Event Type	Event-Specific Values		Wake-up FIFO		Non-wake-up FIFO		Status FIFO	
Name	**Byte 1**	**Byte 2**	**Byte 3**	**Default Enable State**	**Default Int Enable State**	**Default Enable State**	**Default Int Enable State**	**Default Enable State**	**Default Int Enable State**
Dynamic Range Changed	13	Sensor Type	Not used	Enabled		Enabled			
FIFO Watermark	14	Bytes Remaining LSB	Bytes Remaining MSB	Enabled		Enabled			
Reserved	15								
Initialized	16	RAM Ver LSB	RAM Ver MSB	Enabled	Enabled	Enabled	Enabled		
Transfer Cause	17	Sensor Type	Not used						
Event-Driven Software Framework	18	Sensor Type	Condition			Enabled			
Reset	19	0	Reset Cause	Enabled	Enabled	Enabled	Enabled		
Spacer	20	Block Count LSB	Block Count MSB	Enabled[1]		Enabled[1]			

14.3.1 Meta Event: Flush Complete

This Meta Event will be inserted in the appropriate FIFO(s) after a Flush FIFO request, whether there is any data in the FIFO or not. In line with the Android terminology, flushing in this context means “transfer to host”, and not “discard.”

14.3.2 Meta Event: Sample Rate Changed

This Meta Event occurs when a given virtual 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.

Byte 2 indicates the new sample rate (rounded down) for the sensor, saturated at 255. To determine the exact sample rate, the virtual sensor information should be read if 255 is reported or if a fractional value is needed.

This Meta Event will be placed in the same FIFO as the related virtual sensor reports to, for example, if a virtual sensor reports its events into the wake-up FIFO, this event will also be placed in the wake-up FIFO.

14.3.3 Meta Event: Power Mode Changed

This Meta Event indicates when a given sensor powers up or down; the Sensor ID of the related virtual sensor is passed in byte 2.

This Meta Event will be placed in the same FIFO as the related virtual sensor reports to, for example, if a virtual sensor reports its events into the wake-up FIFO, this event will also be placed in the wake-up FIFO.

[1] These Meta Events cannot be disabled.

14.3.4 Meta Event: System Error

This Meta Event reports when a system error has occurred. It is reported in the status FIFO.

See Table 30 for error codes (Byte 2) and Table 29 for the description of the interrupt status (Byte 3).

14.3.5 Meta Event: Algorithm Events

This Meta Event is reserved for algorithm specific reports. The current BSX Fusion Library does not use this feature. It is reserved for future extensions.

14.3.6 Meta Event: Sensor Status

This Meta Event indicates the data quality of the specified virtual sensor. It is sent whenever the status changes. It will be reported to the same FIFO as the virtual sensor reports to.

The Sensor ID of the related virtual sensor is in Byte 2, and the sensor status is in Byte 3. The following sensor status values are used:

Table 123: Sensor status values

Status Value	Meaning
0	Unreliable
1	Accuracy Low
2	Accuracy Medium
3	Accuracy High

14.3.7 Meta Event: Sensor Error

This Meta Event reports when a sensor error has occurred. It is reported in the status FIFO.

Byte 2 is the Physical Sensor ID, as specified in Section 12.3.2.6. Byte 3 is the error code according to Table 30.

14.3.8 Meta Event: 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, for example when the host 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.

Bytes 2 and 3 report a saturating count of lost bytes. Following this Meta Event there will be a full 40-bit timestamp, to ensure the host will be able to report accurate timestamps after the area of data loss.

The BHI385, when it detects a FIFO overflow, automatically discards a block of FIFO data (maximum size 512 bytes) 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.

The Meta Event will be placed in the appropriate wake-up or non-wake-up FIFO.

14.3.9 Meta Event: Dynamic Range Changed

This Meta 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.

This event is inserted in the FIFOs for all enabled virtual sensors whose physical source is the sensor of which dynamic range was changed. For example, if both the wake-up and non-wake-up accelerometer corrected sensors are enabled, and the accelerometer dynamic range is changed by the host, then a dynamic range changed Meta Event will be issued

to the wake-up FIFO for the wake-up accelerometer corrected sensor as well as to the non-wake-up FIFO for the non-wake-up accelerometer corrected sensor.

14.3.10 Meta Event: FIFO Watermark

This Meta Event occurs when the specified watermark level of a FIFO has been reached. Due to synchronization issues, this Meta Event may be displaced by a few dozen bytes, i.e. some other events may be reported after the watermark has been triggered and before the Meta Event occurs.

The Meta Event will be placed in the appropriate wake-up or non-wake-up FIFO.

14.3.11 Meta Event: Initialized

This is the first Meta Event reported after the firmware has completed initialization. It will be placed in both the wake-up and non-wake-up FIFOs. It indicates the end of the initialization phase and shall be read from both FIFOs before any other operation, e.g. enabling a virtual sensor, is performed. See Section 5 Device Configuration and Operation for details of the initialization procedure.

14.3.12 Meta Event: Transfer Cause

The Meta Event occurs when,

- a host transfer has begun
- the reason is because of an on-change sensor, and
- the Transfer Cause Meta Event is enabled

In this case, the first event in the FIFO will be a Transfer Cause Meta Event. The sensor ID of the sensor causing the transfer will be included in byte 2.

It will be placed in the appropriate wake-up vs. non-wake-up FIFO.

14.3.13 Meta Event: Software Framework

This Meta Event will be issued for various Event-Driven Software Framework-related errors.

The sensor ID for the sensor which is associated with the error is reported in byte 2, and the specific framework error is reported in byte 3.

Software Framework errors are:

- 1 = virtual sensor trigger was delayed (CPU is heavily loaded)
- 2 = virtual sensor trigger was dropped (an entire sample period passed without time to trigger the sensor; CPU overloaded)
- 3 = because the requested rate is so low, the hang detection logic has been disabled for this sensor (otherwise not an error)
- 4 = the parent of the specified virtual sensor is unexpectedly not enabled

14.3.14 Meta Event: Reset

This Meta Event is placed by the boot loader in the wake-up and non-wake-up streams, to increase the chances that the host will notice a watchdog reset during a period where the host is expecting to receive sensor data.

Byte 2 is always 0. Byte 3 indicates the Reset cause:

Table 124: List of reset causes

Byte 3 value	Reset Cause
0	Power-On Reset
1	External Reset (RESETN pin)
2	Reset by host command
4	Watchdog Reset

14.3.15 Meta Event: Spacer

This Meta Event is used to maintain a fixed size for the FIFO block header. If a previous block was discarded due to FIFO overflow, the FIFO block header will instead contain a FIFO Overflow Meta Event. The host should ignore the Spacer Meta Event.

14.4 Debug Data

This event contains binary or string debug data that has been created by using the fwrite() or printf() functions for debugging during the software development phase.

Table 125: Debug data format

Byte Number	Contents	Format
0	Sensor ID	8-bit unsigned, value: 250, see Table 107: Overview of FIFO event IDs.
1	Flags	8-bit unsigned integer
2 .. 17	Data	8-bit unsigned integer

The flags consist of 6 bits of valid length, which indicates the number of bytes in the Data area that are valid, and 1 bit indicating the type (binary or string):

Table 126: Flags in debug data format

Bit Number	Contents
0-5	Valid length
6	Format (1 = binary, 0 = string)
7	Reserved

15 Reading FIFO Data

The FIFO content is read from Output Channel 1-3, which contain the wake-up, non-wake-up, and Status and Debug FIFO streams. The information about the amount of data in each channel is given by the first two bytes of each FIFO stream at the start of a host transfer (see Section 13 FIFO Data Formats).

When the host receives the host interrupt from the BHI385, it can optionally first read the Interrupt Status register to determine which channel has available data (or some other reason for the interrupt). It can then read at least the first 2 bytes of the channels that have data pending to determine how many bytes follow. It should then read all those bytes in one or more consecutive SPI or I2C read operations[1)].

FIFO data is stored in one or more FIFO blocks by BHI385, as needed. The host only needs to start reading the appropriate channel register and continue reading until the entire transfer is complete. The host may break this read into smaller blocks at its convenience, as long as it eventually reads the entire pending amount[2)].

15.1 Host Interrupt Behaviour

During normal operation, the host interrupt signal will be asserted when data is available in any of the FIFOs and it is time to notify the host.

For example, this is the case when:

- An enabled virtual sensor has a zero max report latency (timeout) and has generated a sample
- 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
- One (or more) virtual sensor has a non-zero max report latency, it has samples in the FIFO, the FIFO Watermark is non-zero, and it has been exceeded before the latencies timed out
- A Meta Event has occurred which has its interrupt enable set (by default, only internal firmware errors or sensor hardware errors can generate interrupts)
- The AP is in suspend mode, one or more wake-up sensors are enabled, and one or more wake-up events have occurred (no other event except unexpected reset will generate an interrupt in this state), and the wake-up FIFO interrupt disable bit is clear in the Host Interface Control register
- The AP is in suspend mode, the wake-up FIFO watermark is non-zero, and all enabled sensors have non-zero latency; when the watermark is reached, the AP will be woken up

Upon power-up, the host interrupt signal is only asserted if a watchdog timeout or other fatal error caused an unplanned reset, then using the previous interrupt configuration. However, a host-initiated reset due to power-on reset, reset pad, or write to the reset register will not result in an interrupt being raised.

In level interrupt mode, the host interrupt signal 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[3)].

It may then reassert immediately depending on the notification criteria above.

[1] Reading more data than the Transfer Length indicates is allowed; all bytes read beyond the Transfer Length will be 0. However, this is only true until the end of the current read transaction (until the host de-asserts the chip select pin in SPI mode, or until it issues the I2C STOP condition in I2C mode). A subsequent read transaction could return the beginning of the FIFO Descriptor of the next transfer.

[2] Stopping to read before the total amount of bytes is read, will block any new host interrupt until the remaining data of the FIFO is read, or until a transfer abort is requested (which might cause data loss).

[3] The host interrupt may take up to 150us to de-assert after the host interface has been handled.

Figure 21: Active high level host interrupt example

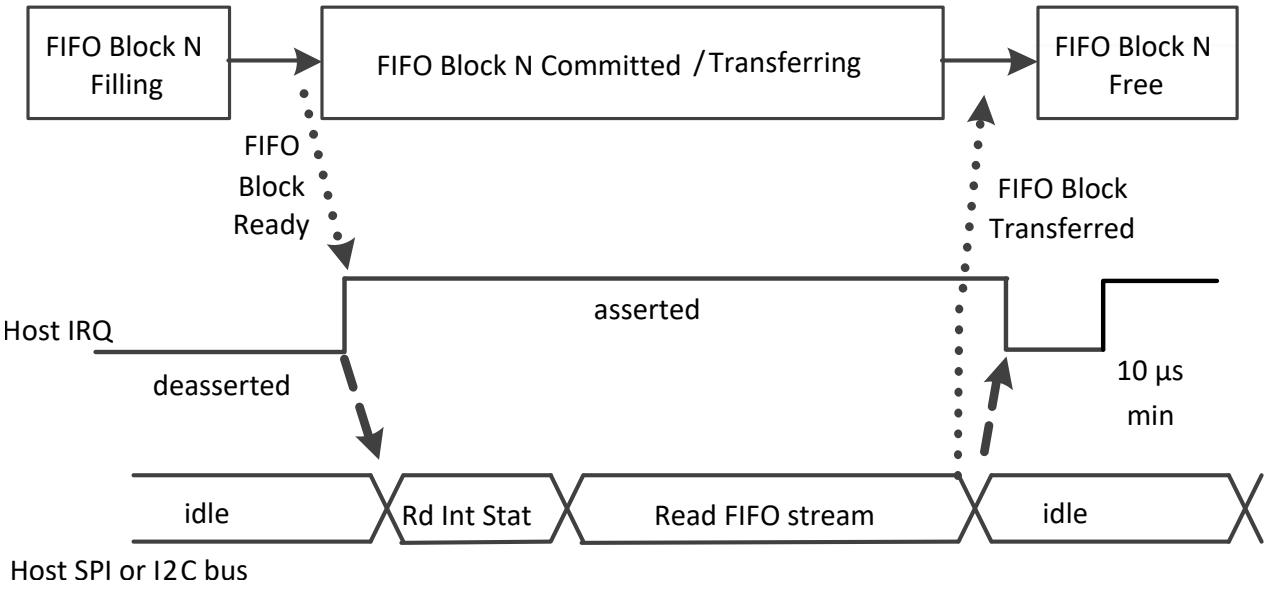
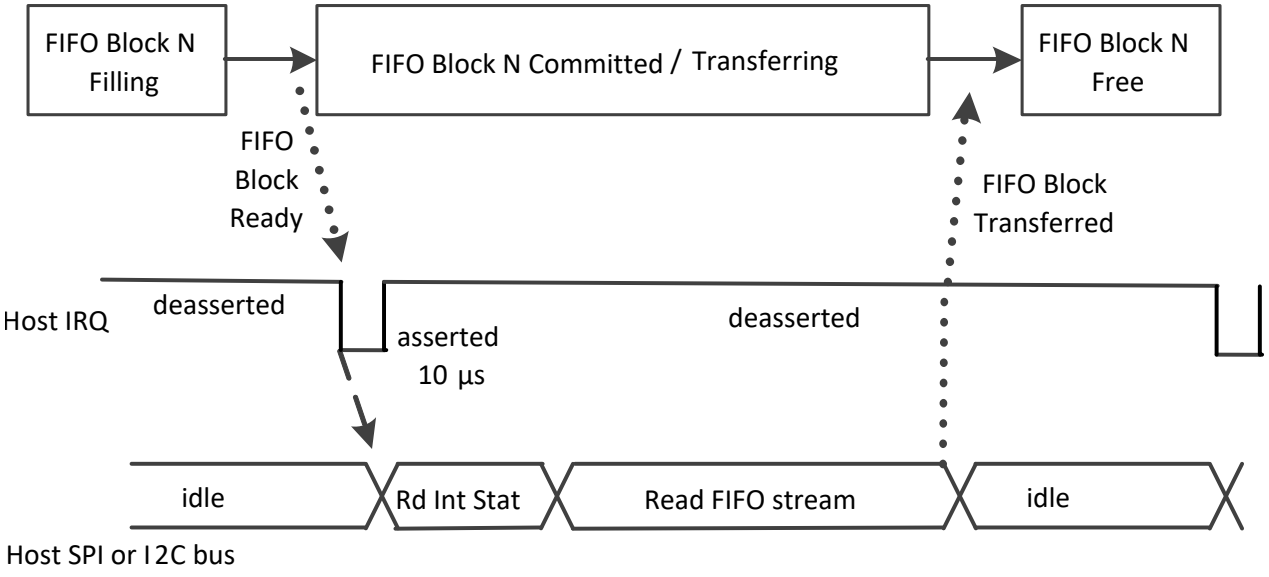


Figure 22: Active low edge triggered host interrupt example



In edge triggered mode, the host interrupt will pulse for 10µs to start the transfer. It will pulse again after the host has emptied the FIFO or aborted the transfer if there is more data to be read.

The host interrupt will always go low for a minimum of 10µs between the time the host receives and empties the FIFO and any subsequent transfer. This is to ensure that either level or edge-triggered interrupts can be used by the host.

The time at which the rising edge of the interrupt occurred will be accessible from the Host Interrupt Timestamp Parameter, see Section 11.1.22.

15.2 FIFO Overflow Handling

In the event that the FIFO overflows, due to too high sample rates and/or slow reading by the host, or simply the host being asleep, BHI385 will discard the oldest data in the FIFO first. The data will be discarded in full FIFO Blocks, following the FIFO format described in Section 13. In the event a host interrupt is pending or a host transfer is actively underway, some amount of the oldest data, namely the currently transferred and the following FIFO block, will still be

transferred to the host and are not affected by the discard. Only in the rare case that besides the currently transferred block, only one other FIFO block is present, then this block will be discarded.

The Event-Driven Software Framework will insert an Overflow Meta Event (if enabled) into the header of the FIFO Block at the point of data loss, followed immediately by the Full Timestamp for the continuation point, followed finally by the new data sampled after the overflow. This way, the host can know that data was lost and has an accurate timestamp for the data that continues after the data loss.

Table 127: Data Chronology after FIFO overflow

Last sample before data loss (oldest sample)	
<gap in data>	**Time** ⇓⇓
Overflow Meta Event	
Full Timestamp	
Most recent samples	
...	

15.3 Application Processor Suspend Mode

When the application processor goes into suspend mode, it should inform BHI385 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 precedes this with a FIFO Flush and complete emptying of the FIFO, so that the host is not immediately woken by mistake.

When the AP wakes up, it should notify BHI385 by clearing the AP Suspend bit again, so that all enabled sensors can (if configured) issue a host interrupt as needed. The BHI385 will detect this action and automatically prepare the output channels and assert the host interrupt, if data is pending. If the AP did not notify BHI385 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 first pending FIFO transfer will contain a smaller Transfer Length than the total available. Once the host has transferred this smaller number of bytes, the next host transfer will cover the remaining amount to be transferred.

15.4 Loss of Sync Recovery

It might occur that the host can't interpret the FIFO data that it is reading anymore (e.g. in case of data corruption on the bus). In this case it may have lost sync to the FIFO structure. To recover from this state, the host should abort the current transfer by writing the appropriate Abort Transfer bit in the Host Interface Control register 0x06 (Section 11.1.7) to 1. This may cause data loss of 32 bytes of data.

Alternatively, the host can also finish the current transfer, discard what it is receiving, and then wait for the next transfer as signaled by a host interrupt.

After that, the next transfer will start again with a whole block of data as specified in Section 13 FIFO Data Formats.

16 Error Detection and Recovery

The BHI385 can, under extraordinary circumstances, reach a fatal error condition during the boot or execution of firmware. Therefore, the host software or driver need to be able to detect and recover from such a condition.

It is recommended for a production system to make the BHI385 driver self-monitor the recovery process. It should back off recovery attempts over longer and longer time intervals, and eventually stop recovery attempts after some reasonable amount of time and log a fatal error. This is to prevent an infinite loop of recovery attempts caused by a continuous fatal hardware error or corrupted firmware file on the host system. The following three scenarios should be addressed in the host:

1. Watchdog / Power on Reset occurred during operation with one of the following symptoms:
- Host IRQ is received with 0 bytes to transfer
- Firmware Idle is set in the Boot Status register
- Kernel version number is now 0
- Error register contains a non-zero error code
- Wake-up and non-wake-up FIFOs contain Reset Meta Events

As a reaction, the following steps should be taken:
1. Log registers 0x04 to and including 0x31
2. Issue Download Post Mortem Data command, receive Crash Dump status packet, store for later analysis
3. Reload firmware image
4. Configure system parameters (FIFO watermark, etc.)
5. Restart virtual sensors

2. Errors detected by BHI385 firmware:
- Error Event in a FIFO (precondition: the error Meta Events are not masked)
- Non-zero state of Error register

Reaction for different error categories should be:
- Fatal errors: issue reset over SPI or I2C (write 0x01 to register 0x9B or assert HW reset pin), then the same as item 1
- Hardware errors: same as previous point
- Programming errors: should never occur in the field; recovery same as previous point
- Temporary errors: It is acceptable to ignore and continue. It is recommended to log the error state for later analysis.

3. Unexpected/Unknown State/Live-lock
- Detection only by software watchdog in the host driver Reaction should be:
- Follow the same recovery as item 2) for Fatal Errors

17 Physical and Electrical Specifications

17.1 Absolute Maximum Ratings

Table 128: Absolute maximum ratings [1]

Parameter	Condition	Min	Max	Unit
Voltage at Supply Pin	VDD Pin	-0.3	4	V
	VDDIO Pin	-0.3	2.75	V
Voltage at any Logic Pin	Non-Supply Pin	-0.3	VDDIO+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
Non-volatile memory (NVM) write-cycles	Using Physical Sensor Control Parameters		14	cycles
Mechanical Shock	Duration 200 µs, half sine		10,000	g
	Duration 0.3 ms, half sine		2,900	g
	Free fall onto hard surfaces		2	m
ESD	HBM		2	kV
	CDM		500	V
	MM		200	V

17.2 Operating Conditions

Table 129: Operating conditions

Parameter	Symbol	Min	Typ	Max	Unit
Supply Voltage IMU Analog Domain	V_{DD}	1.71	1.8	3.6	V
Supply Voltage I/O Domain and Fuser2	V_{DDIO}	1.71	1.8	1.89	V
External regulator decoupling capacitor	C_{reg}	0.1	0.22	0.5	µF
	ESR			0.5	Ω
Operating Temperature	T_A	-40		85	°C
RESETN pulse duration (active low)	t_{RSTN}	100			ns

[1] Stress above these limits may cause damage to the device. Exceeding the specified electrical limits may affect the device reliability or cause malfunction.

17.3 Electrical Characteristics Table

Table 130: Electrical characteristics

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Brown-out detection level rising	$V_{BROUT-R}$	T_A = -40°C to 85°C	1.3		1.6	V
Voltage Input Low Level	V_{IL}				0.3VDDIO	-
Voltage Input High Level	V_{IH}		0.7VDDIO			-
Voltage Output Low Level, auxiliary IMU interface (pads ASDX, ASCX, OCSB, OSDO)	V_{OAL}	VDDIO=1.71V, IOL=3mA			0.2VDDIO	-
Voltage Output High Level, auxiliary IMU interface (pads ASDX, ASCX, OCSB, OSDO)	V_{OAH}	VDDIO=1.71V, IOH=3mA	0.8VDDIO			-
Voltage Output Low Level, all other outputs	V_{OML}	VDDIO=1.71V, IOL=4mA for LDS [3], IOL=6mA for HDS[4]			0.3VDDIO	-
Voltage Output High Level, all other outputs	V_{OMH}	VDDIO=1.71V, IOL=4mA for LDS[3], IOL=6mA for HDS[4]	0.7VDDIO			-
Internal Pull-up resistor (M3SCL pad)	R_{UPCLK}		35	61	107	kΩ
Internal pull-up resistors (other pads)	R_{UP}		71	121	214	kΩ
Internal Pull-down resistor (HSDO pin)	R_{DN}		62	109	208	kΩ
Pad Capacitance, HSCX, HCSB, HSDX, HSDO, HIRQ, RESETN, JTAG_DIO, M3SDA pads	C_{PAD1}				1.8	pF
Pad Capacitance M3SCL pad	C_{PAD2}				3.6	pF
Pad Capacitance ASCX, ASDX, OCSB, OSDO, RESV1, RESV2 pads	C_{PAD3}				6.8	pF
System Oscillator Frequency	f_{SYSLR}	Long run mode, T_A=25°C	18.4	20	21.6	MHz
	f_{SYST}	Turbo Mode, T_A=25°C	46	50	54	MHz
System Oscillator Temperature Drift	DF_{SYS_TEMP}	Drift from 25°C for T_A = -40°C to 85°C	-6		6	%

[1] Start-up time is the time from oscillator enable until the first rising edge of the oscillator. The first cycle will be within 10 percent of the final frequency.
[2] Fuser2 in Deep Sleep: all oscillators are disabled.
[3] LDS: low drive strength configuration of output driver.
[4] HDS: high drive strength configuration of output driver.
[5] Expressed as percentage of the oscillator frequency.
[6] Theses values refer to a processor load of 100 percent. For realistic application use cases please refer to Section 5.5.

Parameter	Symbol	Condition	Min	Typ	Max	Unit
System Oscillator Start-up time[1)]	OSC_{SYS_START}	T_A=25°C			4	µs
Timer Oscillator Frequency	f_{TMR}	T_A=25°C	125	128	131	kHz
Timer Oscillator Temperature Drift	DF_{TMR_TEMP}	Drift from 25°C for T_A = -40°C to 85°C	-5		5	%
Timer Oscillator Start-up time[1)]	OSC_{TMR_START}	T_A=25°C			250	µs
Timer Oscillator trim step size[5)]	d_{TRIM_TMR}	T_A=25°C		0.9		%
Current consumption, total on pins VDD and VDDIO	(I_{DD}+ I_{DDIO})	Gyro and Accel in suspend, Fuser2 in Deep Sleep, 32KBytes RAM retention, T_A= 25°C		7.8		µA
		Gyro and Accel in suspend, Fuser2 in Regular Sleep, 32KBytes RAM retention, T_A = 25°C		8.1		µA
		Accel in Low Power Mode, ODR 25Hz, Gyro in suspend, Fuser2 in deep sleep, T_A=25°C		14		µA
		Accel in Normal Mode, Gyro in suspend, Fuser2 in deep sleep, T_A=25°C		214		µA
		Accel and Gyro in Low Power Mode, ODR 25Hz, Fuser2 in deep sleep, T_A=25°C		424		µA
		Accel and Gyro in Normal Mode, ODR max, Fuser2 in deep sleep, T_A=25°C		689		µA
		Accel and Gyro in Performance Mode, ODR max, Fuser2 in deep sleep, T_A=25°C		974		µA
		Fuser2 executing matrix multiplication in Long Run mode, Gyro and Accel in suspend, T_A= 25°C		840		µA
		Fuser2 executing CoreMark® in Long Run mode[6)], Gyro and Accel in suspend, T_A = 25°C		950		µA
		Fuser2 executing CoreMark® in Turbo mode[6)], Gyro and Accel in suspend, T_A= 25°C		2800		µA

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Fuser2 CPU benchmark		Metaware compiler ccac, compiler options set for maximum performance		3.67		Core Mark / MHz

17.4 Physical Characteristics and Measurement Performance

Table 131: Operating conditions accelerometer

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Acceleration Range	g_{FS4g}			±4		g
	g_{FS8g}			±8		g
	g_{FS16g}			±16		g
	g_{FS32g} [1)]		±24	±28	±32	g
Start-up time	$t_{A,su}$	Suspend to normal mode VDD=1.8 V, T_A=25°C, ODR_{max}		2		ms

Table 132: Output signal accelerometer

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Resolution				16		bit
Sensitivity	S_{4g}	g_{FS4g}, T_A=25°C		8192		LSB/g
	S_{8g}	g_{FS8g}, T_A=25°C		4096		LSB/g
	S_{16g}	g_{FS16g}, T_A=25°C		2048		LSB/g
	S_{32g}	g_{FS32g}, T_A=25°C		1024		LSB/g
Sensitivity Error	$S_{A,err}$	T_A=25°C Nominal VDD supplies, all ranges, soldered, over life time[2)]		±0.4		%
Sensitivity Temperature Drift	TCS_A	g_{FS2g}, Nominal VDD supplies best fit straight line, soldered		±0.005		%/K
Sensitivity change over supply voltage	$S_{A,VDD}$	T_A=25°C, VDD,min ≤ VDD ≤ VDD, max best fit straight line, soldered		0.0001		%/V
Zero-g Offset	$Off_{A,life}$	g_{FS8g}, T_A=25°C, nominal VDD supplies, fast offset compensation off, soldered, over life time[2)], data read from host interface		±25		mg
Zero-g Offset Temperature Drift	TCO_A	g_{FS8g}, Nominal VDD supplies best fit straight line, soldered, data read from host interface		±0.25		mg/K
Nonlinearity	NL_A	Best fit straight line, g_{FS4g} and g_{FS32g} over ±2g input acceleration as full-scale, soldered		±0.5		%FS
Output Noise	$n_{A,nd,8g}$	Normal mode T_A=25°C, nominal VDD, g_{FS8g}, soldered		0.16		mg/√Hz
	$n_{A,nd,32g}$	Normal mode T_A=25°C, nominal VDD, g_{FS32g}, soldered		0.17		mg/√Hz

Parameter	Symbol	Condition	Min	Typ	Max	Unit
	$n_{A,rms,32g}$	Low power mode T_A=25°C, nominal VDD, g_{FS32g}, soldered, ODR = 100Hz and AVG = 8		2		mg_{rms}
Cross Axis Sensitivity	S_A	Relative contribution between any two of the three axes		0.2		%
Alignment Error	E_A	Relative to package outline		±0.5		°
Zero-g offset over PCB strain	$Off_{A,PCB}$	T_A=25°C, nominal V_{DD}, soldered		±0.010		mg/ µstrain
Nominal Output Data rate	$ODR_{A,OIS}$	Read via OIS interface, normal mode	12.5		1600	Hz
	$ODR_{A,HIF}$	Read via host interface, normal mode	12,5		800	Hz
	$ODR_{A,lpm}$	Low power mode[3)]	12,5		400	Hz

Table 133: Operating conditions gyroscope

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Range	R_{FS125}	Selectable via Change Sensor Dynamic Range command, Section 12.2.8		125		°/s
	R_{FS250}			250		°/s
	R_{FS500}			500		°/s
	R_{FS1000}			1,000		°/s
	R_{FS2000}			2,000		°/s
Start-up time	$t_{G,su}$	Suspend to normal mode, repeated, VDD = 1.8 V, T_A=25°C, ODR_{max}		45		ms
	$t_{G,FS}$	Fast start mode, VDD = 1.8 V, T_A=25°C, ODR_{max}		2		ms

Table 134: Output signal gyroscope

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Resolution				16		bit
Sensitivity	R_{FS2000}	T_A=25°C, with default sensitivity correction in Fuser2 firmware applied		16.384		LSB/°/s
	R_{FS1000}			32.768		LSB/°/s
	R_{FS500}			65.536		LSB/°/s
	R_{FS250}			131.072		LSB/°/s
	R_{FS125}			262.144		LSB/°/s
Sensitivity Error	S_{G_err}	T_A=25°C Nominal V_{DD} supplies, soldered, over life time[2)]		±2		%
	$S_{G\_err\_c}$	T_A=25°C Nominal V_{DD} supplies, soldered, over life time[2)] application PCB compensated		±0.4		%

[1]When the sensor is configured for the 32g range, its output may not change with accelerations exceeding the limits in Table 131: Operating conditions accelerometer. This is due to increased non-linearity in the transfer function near the signal chain's saturation point.
[2]Values taken from qualification, according to IPC/JEDEC J-STD-020E.
[3]If the sensor is configured at a higher ODR than the maximum ODR in low power mode, it is operated automatically normal mode.

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Sensitivity change over temperature	TCS_G	R_{FS2000}, Nominal V_{DD} supplies, best fit straight line, soldered		±0.02		%/K
Sensitivity change over supply voltage	$S_{G,VDD}$	T_A=25°C, $V_{DD,min} \leq V_{DD} \leq V_{DD,max}$ best fit straight line, soldered		0.0005		%/V
Nonlinearity	NL_G	Best fit straight line, R_{FS1000}, R_{FS2000}, soldered		0.01		%FS
Zero-rate offset	Off, Ω_x, Ω_y, Ω_z	T_A=25°C, fast offset compensation off, sensor data read from host interface, soldered, over life time[2)]		±0.5		°/s
Zero-rate offset change over temperature	TCO_G	R_{FS2000}, Nominal V_{DD} supplies, best fit straight line, sensor data read from host interface, soldered		±0.015		°/s/K
Output Noise	$n_{G,nD}$	Performance mode T_A=25°C, nominal V_{DD}, range = 250dps, soldered		0.007		°/s/√Hz
		Normal mode T_A=25°C, nominal V_{DD}, range = 250dps, soldered		0.01		°/s/√Hz
Nominal Output Data Rate	$ODR_{G,OIS}$	Read via OIS interface, normal mode	25		6400	Hz
	$ODR_{G,HIF}$	Read via host interface, normal mode	25		800	Hz
	$ODR_{G,lpm}$	Low power mode[1)]	25		100	Hz
Output Data rate accuracy (set of x,y,z rate)	$OAcy_{G,n}$	Normal and performance mode, T_A=25°C, nominal V_{DD}		±1.7		%
	$OAcy_{G,n,T}$	Normal mode, full T_A range, same part, nominal V_{DD}		±0.0048		%/K
Cross Axis Sensitivity	$X_{G,S}$	Sensitivity to stimuli in non-sense-direction		±0.2		%
Zero-g offset over PCB strain	$Off_{G,PCB}$	T_A=25°C, nominal V_{DD}, soldered		±0.3		mdps/ µstrain

Table 135: Output signal of temperature sensor

Parameter	Symbol	Condition	Min	Typ	Max	Units
ADC Resolution				16		bits
Temperature Sensor Measurement Range	T_s		-40		+87	°C
Output at 23 °C				0		LSB
Sensitivity	S_T			512		LSB/K
Temperature offset	O_T	After soldering		±3		K
Temperature sensitivtiy error		After soldering		±2		%

[1] If the sensor is configured at a higher ODR than the maximum ODR in low power mode, it is operated automatically in normal mode

Parameter	Symbol	Condition	Min	Typ	Max	Units
Output Data Rate Temperature Sensor	$ODR_{T,G}$	Normal mode and performance mode, TA=25°C, nominal VDD, Gyroscope on			100	HZ
	ODR_T	All other modes incl. low power mode			0.78	
ODR Accuracy Temperature Sensor	$OAcy_{T,G}$	Normal mode and performance mode, TA=25°C, nominal VDD, Gyroscope on		<1.5		%
	$OAcy_T$	All other modes incl. low power mode		1.5		

17.5 Timing Characteristics

17.5.1 Fuser2 Power-Up and Power-Down Timing Characteristics

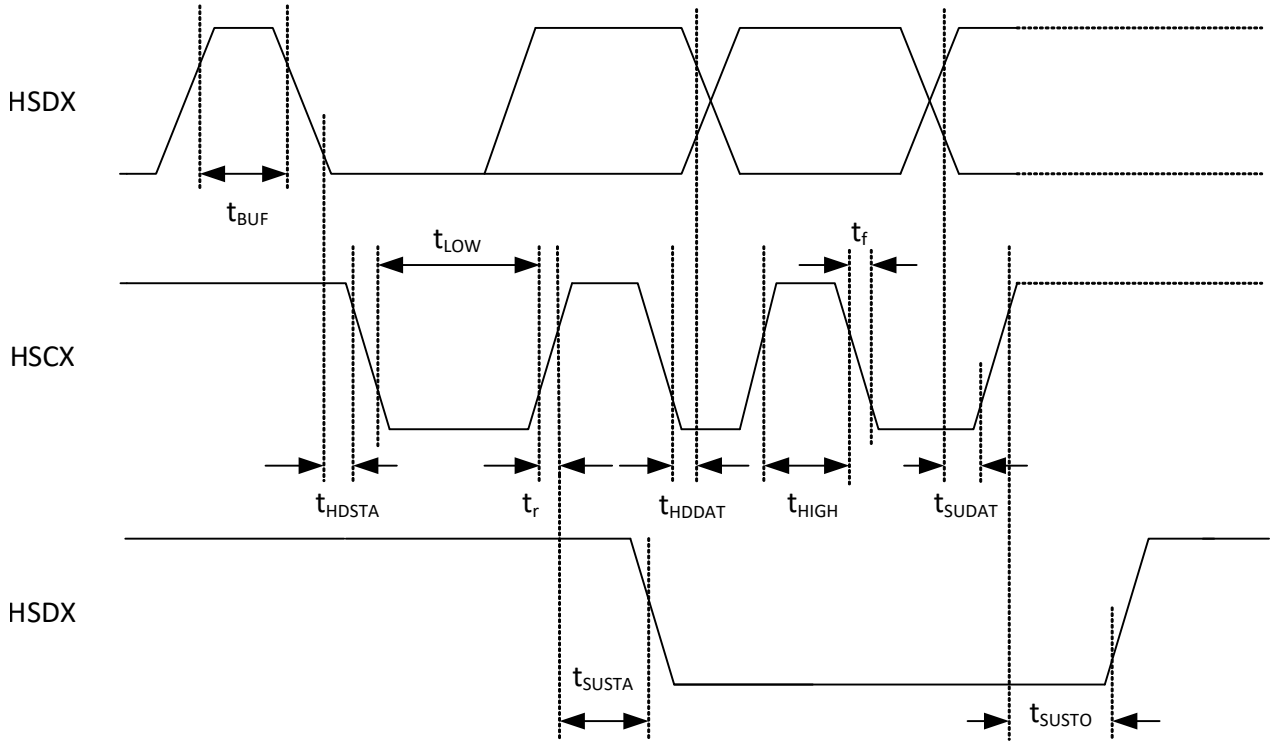
Table 136: Fuser2 power-up and power-down timing characteristics

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Start-up time[1)]	T_start				500	us
Bootloader boot time						
	T_boot_bl_host	Host boot[2)]			1300	us
Firmware load time						
	T_load_fw_host	Host firmware load and verify[3)]		121		ms
Firmware boot time	T_boot_fw_host	Host firmware boot[4)]		81		ms
Run-level switch time to turbo mode	T_switch_turbo	--			3	us
Run-level switch time to long-run mode	T_switch_longrun	--			3	us
Wake-up time from sleep / deep-sleep	T_wake	--			5	us
Time between RAM bank power-on in long-run mode	T_PwrRam_long_run	--			8	CLK[5)]
Time between RAM bank power-on in turbo mode	T_PwrRam_turbo	--			20	CLK[5)]

[1] The start-up time is measured from the point of time when the supply voltage reaches the power-on-reset trip level until the system oscillator is enabled.
[2] The host boot time is measured from the release of the reset until the device is ready to receive a firmware upload ("Host Interface Ready" bit in register "Boot Status" set).
[3] Long run mode, Firmware load and verify time for a 76 kByte firmware file, using host interface in SPI mode at 5 Mbit clock rate.
[4] Long run mode, firmware boot time, measured from sending the command "Boot firmware" until the "Host Interface Ready" bit gets high. Please note that the firmware verification is started already during upload of the firmware, i.e. this time depends on $T_{load_fw_host}$.
[5] CLK is the clock period of the system clock, depending on the mode setting (Long Run vs. Turbo).

17.5.2 Host I2C Interface Timing

Figure 23: Host I2C Interface Timing



Unless otherwise specified, see I2C Specification (see Section 20, Reference 1) for details.

Table 137: Host I2C interface timing

Parameter	Symbol	Condition	Min	Typ	Max	Unit
HSCX frequency	$f_{i2c_s m}$	STM, FM, FM+[1)]		400	1000	kHz
HSCX frequency	$f_{i2c_h s}$	HSM[1)]			3400	kHz
Bus load capacitor	Cb	on HSDX and HSCX			400	pF
HSCX low time	t_{LOW}	HSM / Cb <= 100pF	160			ns
HSDX input setup time[2)]	$t_{SU;DAT}$	STM, FM, FM+	85			ns
		HSM	15			ns
HSDX output hold time	$t_{HD;DAT}$	STM, FM, FM+ / Cb <= 100pF	70			ns
		STM, FM, FM+ / Cb <= 400pF	90			ns
		HSM / Cb <= 100pF	24			ns
		HSM / Cb <= 400pF	27			ns
HSDX output fall time	t_{OF}	Cb <= 100pF	1.5			ns
		Cb <= 400pF	6.5			ns
HSDX-HSCX input delta	$t_{HD;STA}$	STM, FM, FM+	50	70	90	ns
		HSM	6	11	20	ns

[1] STM = standard mode, FM = fast mode, FM+ = fast+ mode and HSM = high speed mode.
[2] Measured with transition times of 16ns from 0 percent VDD on SDA to 0 percent VDD on SCL for SDA rising and 100 percent VDD on SDA to 0 percent VDD on SCL for SDA falling.

17.5.3 Host SPI Interface Timing

Figure 24: Write transaction on 4-wire host SPI

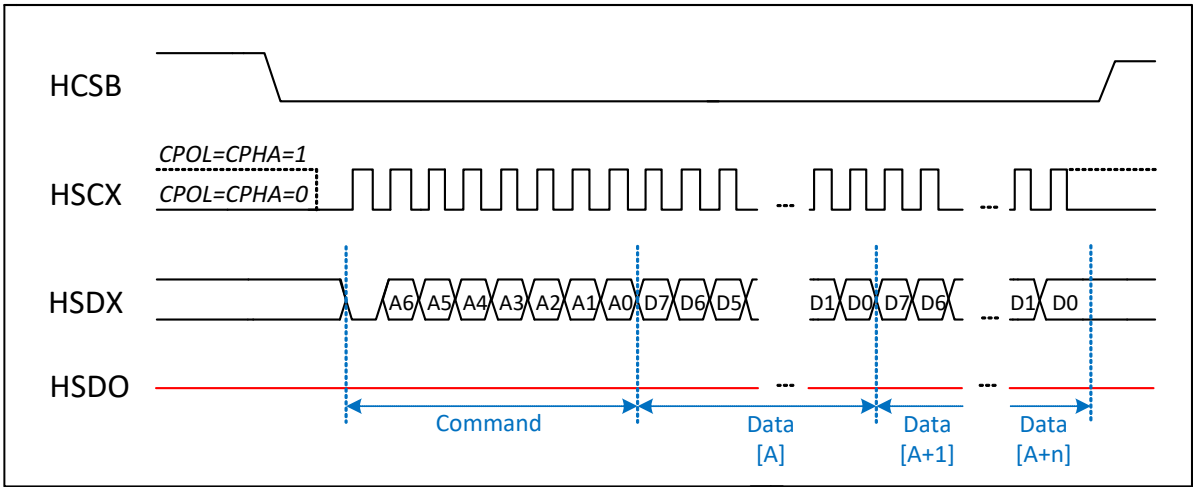


Figure 25: Read transaction on 4-wire host SPI

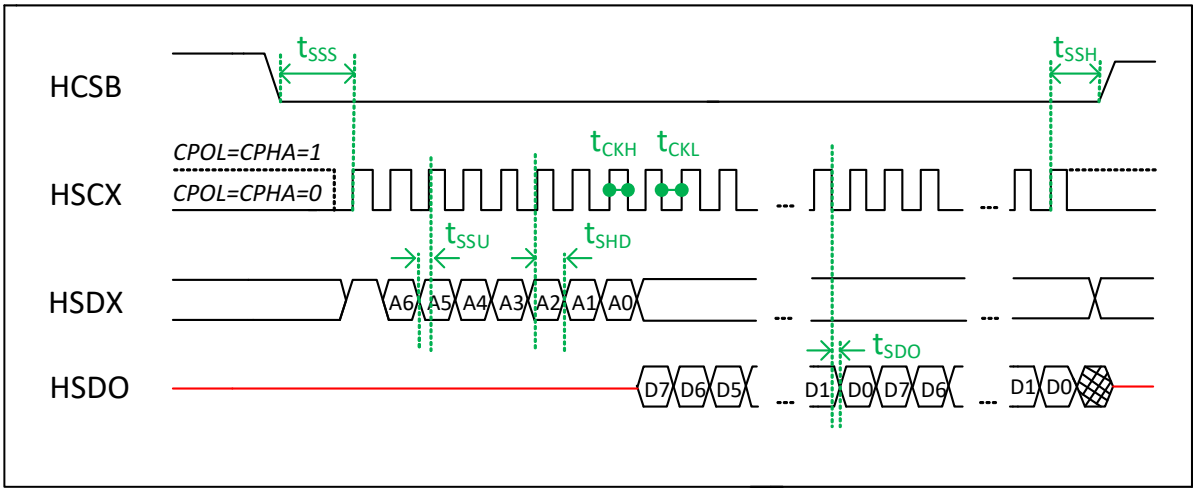


Figure 26: Read transaction on 3-wire host SPI

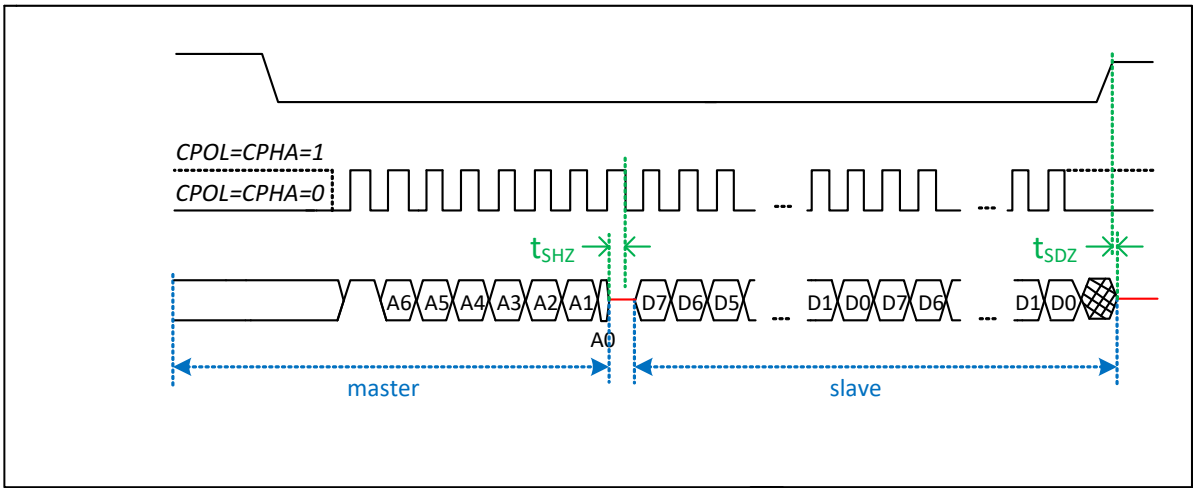


Table 138: Host SPI interface timing parameters in Long-Run (LR) and Turbo mode

Parameter	Symbol	Condition	Min		Max		Unit
			LR	Turbo	LR	Turbo	
SPI clock frequency	f_{SCK}				20	50	MHz
SPI clock high time	t_{CKH}		25	10			ns
SPI clock low time	t_{CKL}		25	10			ns
SPI select setup time	t_{SSS}		50	20			ns
SPI select hold time	t_{SSH}		50	20			ns
HSDX input setup time	t_{SSU}	max trans: 1.5ns	17	6			ns
HSDX input hold time	t_{SHD}	max load: 12pF (for 3-wire SPI)	7	4			ns
HSDX output to Hi-Z time	t_{SHZ}		15	5			ns
HSDO output valid time	t_{SDO}	max load: 12pF			17	9	ns
SPI select to MISO Hi-Z time	t_{SDZ}				20	10	ns

17.5.4 Master Interface I2C Timing

The I2C Master Timing is compliant with I2C specification (Section 20, Reference 1), Standard, Fast and Fast+ Mode.

17.5.5 Master Interface SPI Timing

Figure 27: Master interface SPI clock timing

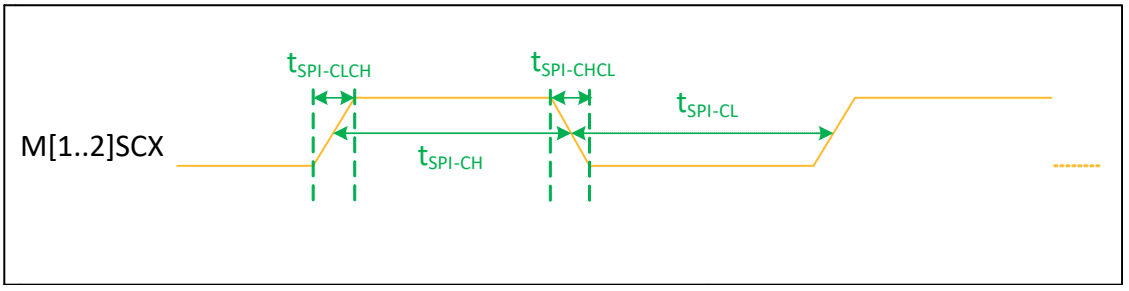
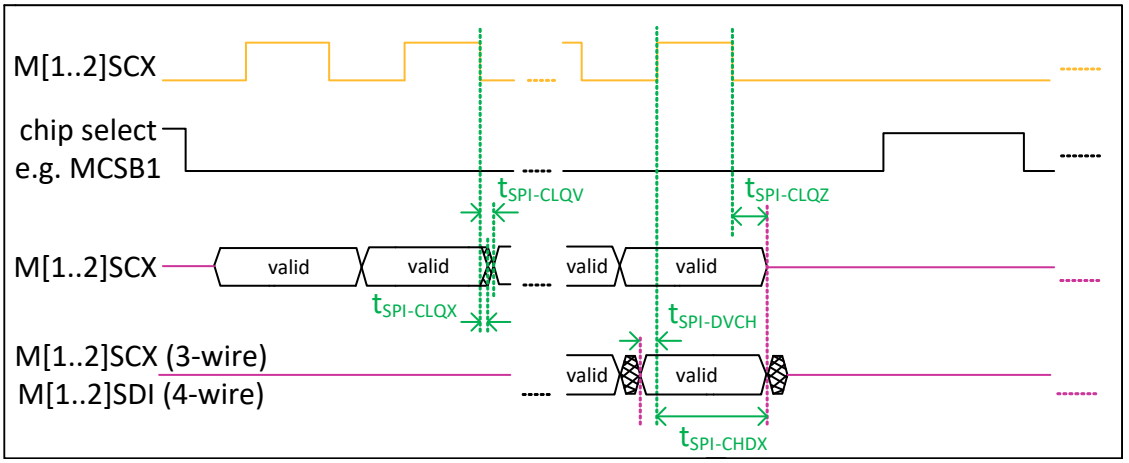


Figure 28: Master interface SPI timing



[1] Timing pertains to both Master Interface 1 and 2. Timing parameters are specified for CPOL/CPHA of 00 and 11; they also hold for CPOL/CPHA of 01 and 10 with corresponding edge adjustment.
[2] Maximum transition time = 1.5ns; load number are for 3-wire SPI.
[3] T = clock period.

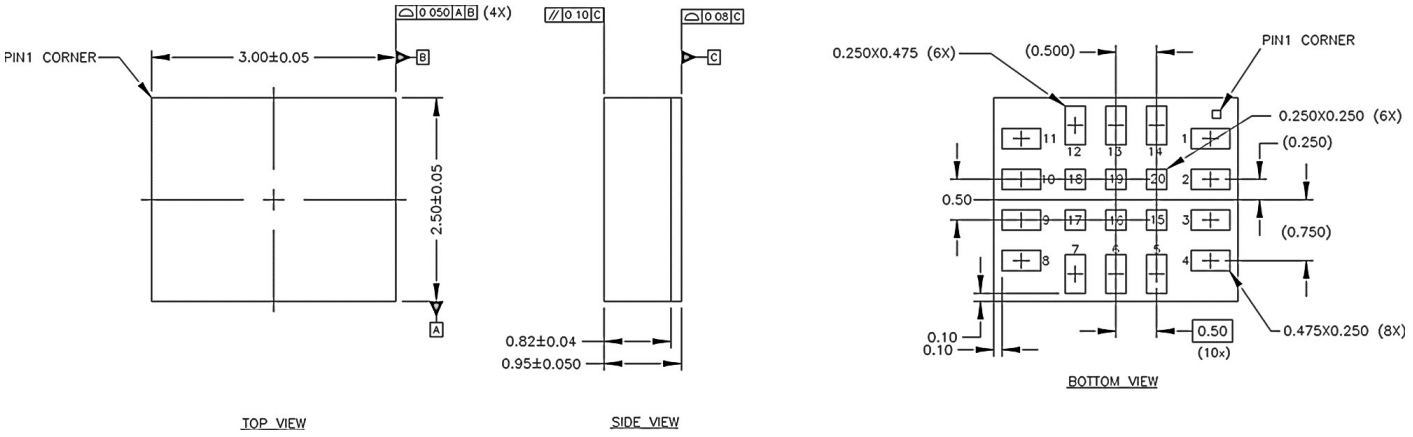
Table 139: Master interface SPI timing parameters in Long-Run (LR) and Turbo Mode

Parameter	**Symbol**	**Condition**	**Min**[1]		**Max**[1]		**Unit**
			LR	Turbo	LR	Turbo	
SPI clock frequency	f_{SPI}	no division	18	45	22	55	MHz
Clock rise time	$t_{SPI-CLCH}$	load: 12pF			2.1	2.1	ns
Clock fall time	$t_{SPI-CHCL}$				2.1	2.1	ns
Clock high time	t_{SPI-CH}	no division	20	8	29	12	ns
Clock low time	t_{SPI-CL}		20	8	29	12	ns
Input data valid to clock rising edge	$t_{SPI-DVCH}$	max trans[2]	4	4			ns
Clock rising edge to input data valid[3]	$t_{SPI-CHDX}$	load: 12pF	50	50			%T
Clock falling edge to output data invalid	$t_{SPI-CLQX}$	load: 12pF	-4	-4			ns
Clock falling edge to output data valid	$t_{SPI-CLQV}$				5	5	ns

18 Mechanical Specifications

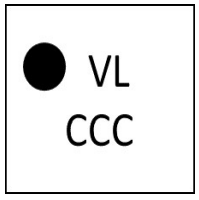
18.1 Outline Dimensions

Figure 29: Outline dimensions



18.2 Device Marking

Table 140: BHI385 device marking

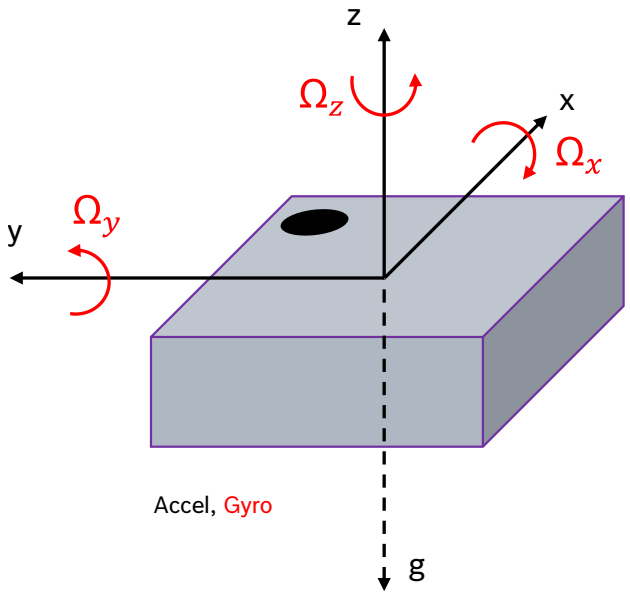
Labelling	Name	Symbol	Remark
	Product identifier	V	One alphanumeric digit fixed to “V” to identify the product
	Internal code	L	One alphanumeric digit, fixed to “L” or “P”, internal use only
	Counter ID	CCC	Tracing identification by three alphanumeric digits
	Pin 1 indicator	•	Pin 1 indicator; fixed; left alignment; font N/A

18.3 Sensing Axes and Axis Remapping

The default axis orientation is shown in Figure 30. This orientation is valid for all sensor outputs (both physical and virtual). For any other sensor connected to the hub, e.g. magnetometer, the physical alignment of the corresponding sensor and its axis should be according to Figure 32 to guarantee correct 9DoF data fusion.

In case that the default axis orientation of BHI385 and/or its sensor extensions does not match with the target device coordinate system, axis remapping can be performed to reassign the sensing axes of BHI385 so that they match with the axes defined by the target device coordinate system.

Figure 30: Sensing axes



Possible placement and remapping options are given in Figure 31.

Figure 31: Placement options with respect to device orientation

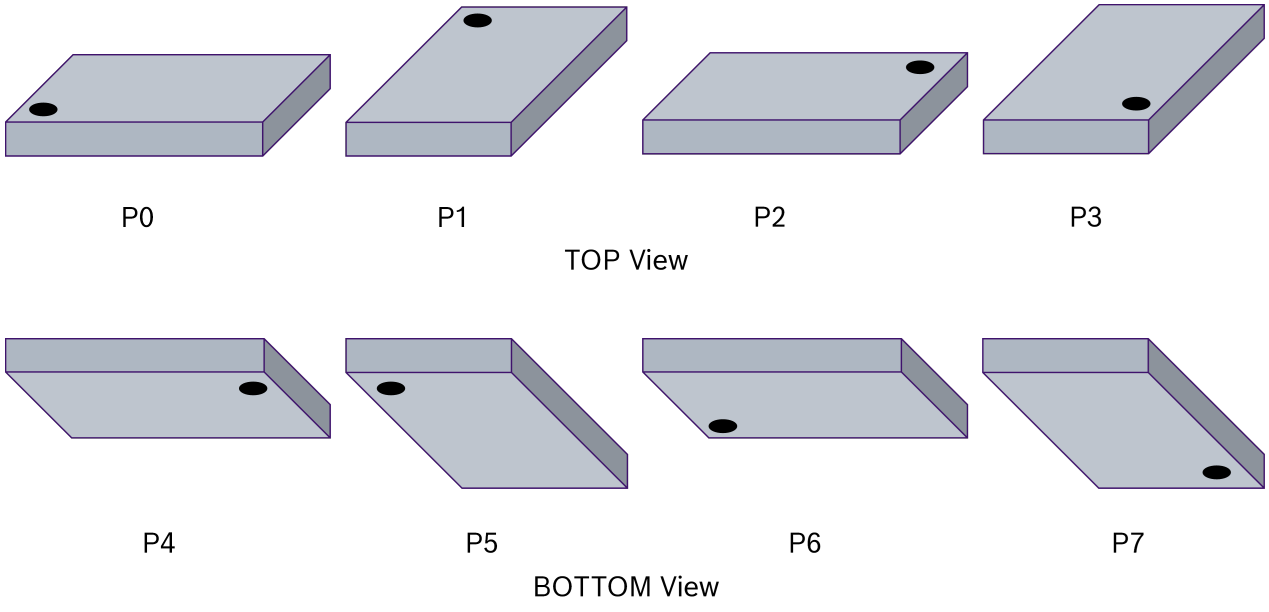
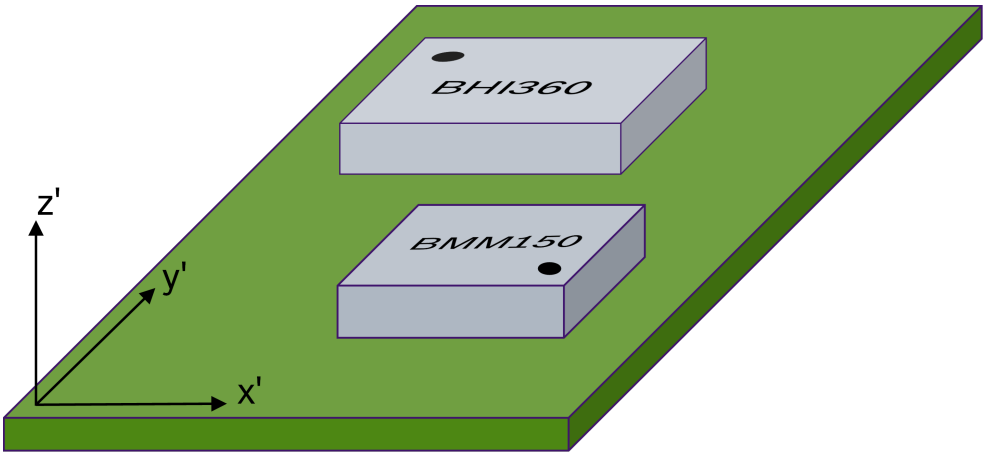


Figure 32: Example component placement



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

The following equation transforms the device axes (xyz) into the target coordinate system ($x'y'z'$). It can also be used to transform the axes of sensors that are attached to the hub:

$$(x'y'z') = (xyz) \cdot \begin{pmatrix} x_x & x_y & x_z \\ y_x & y_y & y_z \\ z_x & z_y & z_z \end{pmatrix}$$

For the example shown in the coordinate system of BHI385 needs to be re-mapped to the target device as in placement option P1 shown in Figure 31. The BMM150 also needs to be re-mapped to the target coordinate system (see BMM150 datasheet for axis orientation). The remapping matrices, corresponding to this example, are given below for both sensors.

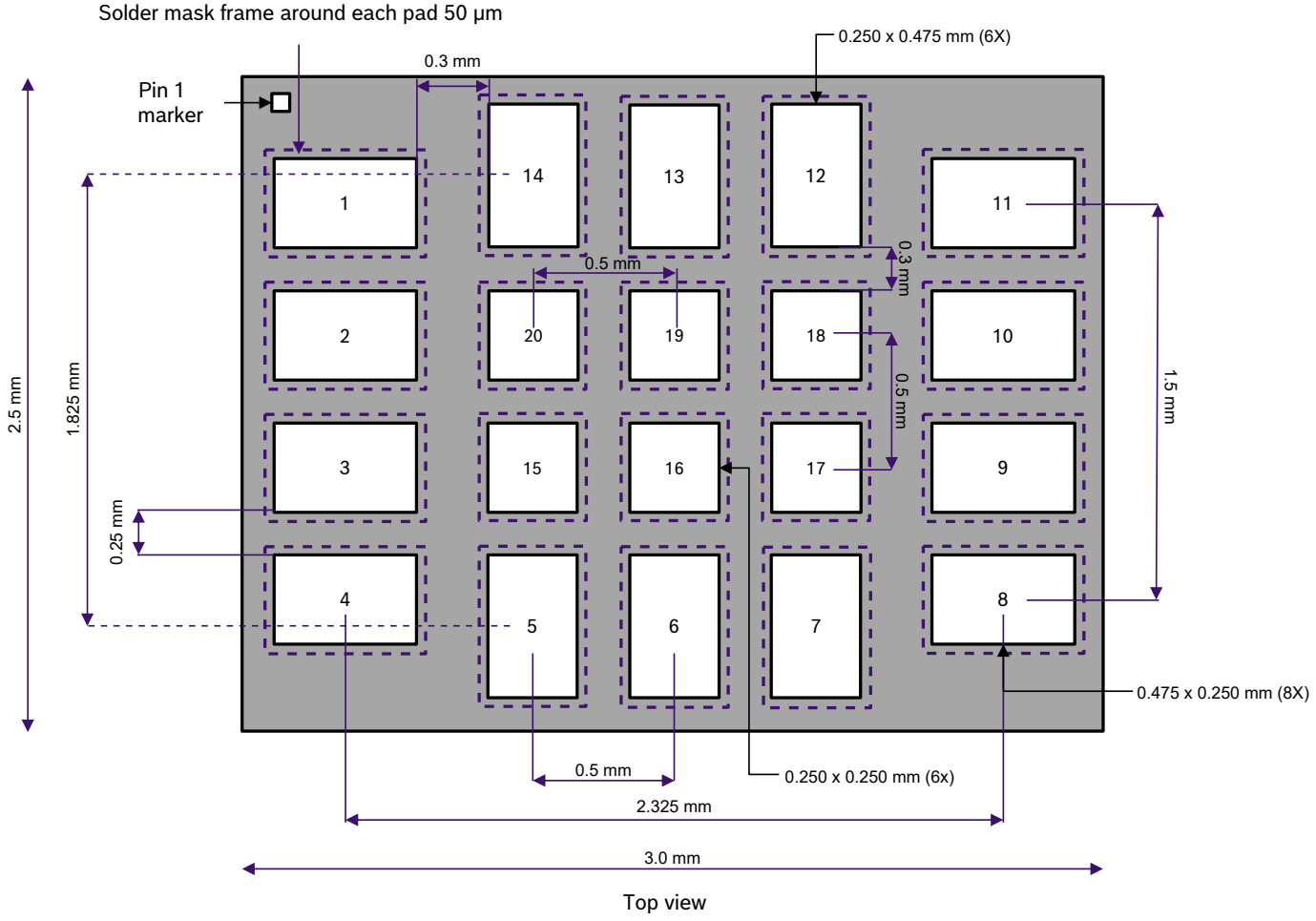
To apply the remapping defined by the matrix $\begin{pmatrix} x_x & x_y & x_z \\ y_x & y_y & y_z \\ z_x & z_y & z_z \end{pmatrix}$ for any triaxial sensor, the matrix elements can be inserted in the specific board configuration file. The values Cal0 (c_0) to Cal8 (c_8) for a sensor correspond to the matrix elements as follows:

$$\begin{pmatrix} x_x & x_y & x_z \\ y_x & y_y & y_z \\ z_x & z_y & z_z \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 \\ c_3 & c_4 & c_5 \\ c_6 & c_7 & c_8 \end{pmatrix}$$

For further information on how to apply this matrix, please refer to Section 12.3.2.7. In case of questions and for technical support, please contact our regional offices, distributors, and sales representatives.

18.4 PCB Footprint Recommendation

Figure 33: Footprint recommendation



19 Handling, Soldering and Environmental Guidelines

19.1 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 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 refer to the corresponding "Handling, soldering and mounting instructions" document or contact our regional offices, distributors and sales representatives.

19.2 Soldering Guidelines

The moisture sensitivity level of the BHI385 sensors corresponds to JEDEC Level 1, see also:

- IPC/JDEC J-STD-020E "Joint Industry Standard: Moisture/Reflow Sensitivity Classification for non-hermetic Solid-State Surface Mount Devices"
- IPC/JDEC J-STD-033D "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.

19.3 Environmental Safety

The BHI385 sensors meet the requirements of the EC restriction of hazardous substances (RoHS) directive, see also:

RoHS - Directive 2011/65/EU and its amendments, including the amendment 2015/863/EU on the restriction of the use of certain hazardous substances in electrical and electronic equipment.

The BHI385 is halogen-free. For more details on the analysis results please contact our regional offices, distributors and sales representatives.

20 References

1. NXP UM10204 I2C-Bus Specifications and User Manual, Rev 6 (April 2014)
2. Programmer's Manual (BST-BHI2xx-BHI3xx-AN002)
3. ARC v2 Programmer's Reference Manual from Synopsys
4. IPC/JEDEC J-STD-020E "Joint Industry Standard: Moisture/Reflow Sensitivity Classification for non-hermetic Solid-State Surface Mount Devices"
5. IPC/JEDEC J-STD-033A "Joint Industry Standard: Handling, Packing, Shipping and Use of Moisture/Reflow Sensitive Surface Mount Devices"
6. BHI3xy SDK Quick Start Guide (BST-BHI3xy-AN000)
7. HSMI for BMI2xy/BMI3xy/BHI3xy IMUs(BST-MIS-HS001)
8. BHI385 Shuttle Board Flyer (BST-BHI385-SF000)
9. BHI385 Product webpage:

21 Legal Disclaimer

i. Engineering samples

Engineering Samples are marked with an asterisk (*), (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.

ii. 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 safety-critical systems. Safety-critical systems are those for which a malfunction is expected to lead to bodily harm, death or severe property damage. In addition, they shall not be used directly or indirectly for military purposes (including but not limited to nuclear, chemical or biological proliferation of weapons or development of missile technology), nuclear power, deep sea or space applications (including but not limited to satellite technology).

Bosch Sensortec products are released on the basis of the legal and normative requirements relevant to the Bosch Sensortec product for use in the following geographical target market: BE, BG, DK, DE, EE, FI, FR, GR, IE, IT, HR, LV, LT, LU, MT, NL, AT, PL, PT, RO, SE, SK, SI, ES, CZ, HU, CY, US, CN, JP, KR, TW. If you need further information or have further requirements, please contact your local sales contact.

The resale and/or use of Bosch Sensortec 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 accepts the responsibility to monitor the market for the purchased products, particularly with regard to product safety, and to inform Bosch Sensortec without delay of all safety-critical incidents.

iii. 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.

22 Document History and Modifications

Rev. No	Chapter	Description of modification/changes	Date
1.0	-	Initial release	August 2025

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

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

Document number: BST-BHI385-DS001-01