BHy2CLI
User Guide

BHy2CLI User Guide

Document revision	2.2
Document release date	28 May 2025
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.

Document number: [TBD] Revision_2.2_052025

Table of Contents

List of Tables

List of Figures

Abbreviations

BHy	***BHy Smart Sensor Hub [BHIxyz]***
CLI	***Command Line Interface***
GPIO	***General Purpose Input Output***
ODR	***Output Data Rate***
PC	***Personal Computer***
RAM	***Random Access Memory***
SoC	***System on Chip***
WRD	***Wearable Reference Design***

1 Overview

The BHy2CLI command-line interface tool is an application based on the Bosch BHy SensorAPI and COINES SDK (Communication with Inertial and Environmental Sensors) application board framework. It can be used to quickly test and evaluate the BHIxyz shuttle board with the application board.

The core functionalities of the BHy2CLI include:

- Programming the BHy customer shuttles
- Accessing registers
- Accessing System/Driver Specific Parameters
- Data Acquisition
- Sensor Configuration
- Accessing System/Application Status Information
- Application Configuration
- Data Injection
- Diagnostics

Figure 1: Overview

This user guide introduces the BHy2CLI commands with examples to demonstrate the use of these commands to operate the BHy Smart Sensor.

1.1 Compatibility

Table 1 describes the compatibility of BHy2CLI with other dependencies.

Item	BHy2CLI	FW	BSX4	BHy SensorAPI	COINES SDK	Supported Boards	Supported Sensors
Version	0.6.0	1.1.18	IR84.3	2.2.0	2.10	APP30 APP31 Nicla Sense	BHIxyz

Table 1: BHy2CLI Compatibility Matrix

***Note**: BHy2CLI is not tested with Nicla Sense board*

2 Setup

Before using the BHy2CLI, it is necessary to understand the system design and set up the embedded environment accordingly.

Note: This Application Note is referenced with COINES SDKv2.10 and BHy SensorAPIv2.2.0.

2.1 COINES SDK Environment

As mentioned earlier, BHy2CLI is based on the COINES SDK Framework. As such, the COINES SDK environment must be set up prior to using the BHy2CLI application, since COINES installation also installs USB drivers which is necessary for MCU testing of BHY2CLI.

1. Install COINES SDK at COINES SDK | Bosch Sensortec (bosch-sensortec.com)
2. For further instructions, refer to Chapter 4, “Installation” in the COINES_SDK_User_Manual.
3. Install USB drivers to execute BHy2CLI application

2.2 Application board setup

Figure 2: HW Setup

The BHy2CLI can be used with an Application Board or the Nicla Sense boards to evaluate the BHy Sensor Hubs (BHIxyz).

The Application Board refers to the Bosch Sensortec Application Board 3.0 | Bosch Sensortec (bosch-sensortec.com) and Application Board 3.1 | Bosch Sensortec (bosch-sensortec.com).

The Nicla Sense board is a compact (wearable form-factor), low-power, and robust development board that enables users to develop smart sensing applications and is suitable for rapid prototyping and deployment. Refer to Arduino Nicla Sense ME | Bosch Sensortec (bosch-sensortec.com) for further details.

Note: BHy2CLI is not tested with the Nicla Sense board.

Table 2 shows the BHy2CLI application can be run in two operation modes.

Mode	Description
MCU	The application runs on the Application Board and interacts directly with the Sensor Hub.
PC	The application runs on a PC and interacts with the Sensor Hub via a base application running on the Application Board.

Table 2: Modes of Operation

Note: *The following instructions for the board setup are referenced for the application board.*

Below is the release package of BHy2CLI, which is received with BHy2CLI v0.6.0, when unzipped, holds BHI3-firmware (BHIxyz firmware), docs (Bhy2CLI User Guide, Change log and Compatibility), firmware (COINES firmware – bootloader, coines_bridge and MTP firmware), MCU (BHy2CLI binaries and MTP tools), PC (BHy2CLI executables) and tools (app_switch and usb-dfu – required to transfer firmware files to External FLASH)

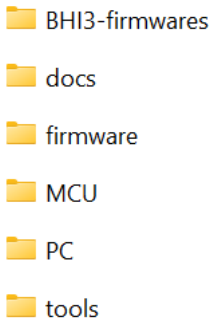


Figure 3: BHy2CLI release package

2.2.1 Running BHy2CLI in MCU mode

In MCU mode, the BHy2CLI application runs in the application board and communicates directly with the BHy Sensor Hub.

Figure 4: MCU Mode

Execute BHyCLI in MCU Mode:

1. From release package (Refer Figure 3 above), open the folder for the Application board connected (***<board>***: app30 or app31)
2. For **app30**,
    a. In <release>/firmware/app30,
        i. Flash, coines_bridge/ update_coines_bridge_flash_fw.bat
        ii. Flash, bootloader_update/ update_bootloader.bat
        iii. Flash, mtp_fw_update/ update_mtp_fw.bat
    b. In <release>/MCU/app30,
        i. Execute app30_format_flash.bat (Formats FLASH)
        ii. Reset board.
        iii. Execute app30_bhi360_mcu_mode.bat (Uploads BHI360 firmwares to External FLASH)
3. For **app31**,
    a. In <release>/firmware/app31,
        i. Flash, coines_bridge/ update_coines_bridge_flash_fw.bat
        ii. Flash, bootloader_update/ update_bootloader.bat
        iii. Flash, mtp_fw_update/ update_mtp_fw.bat
    b. In <release>/MCU/app31,
        i. Execute app31_format_flash.bat (Formats FLASH)
        ii. Reset board.
        iii. Execute app31_bhi360_mcu_mode.bat (Uploads BHI360 firmwares to External FLASH)
4. Reset board.
5. BHyCLI application is ready to communicate (Refer Table below for Platform/Application reference)

Note: When executing **<board>**_format_flash.bat, there is a prompt message to confirm if format can be performed. Entering ‘y’ performs format of External FLASH.

Note: By default, the SPI Interface of BHy2CLI is present. To change to the I2C interface, update to i2c_bhy2cli.bin in <board>_bhixyz_mcu_mode.bat

- The following host applications can be used to communicate with the BHy2CLI application.
- For communication over serial interface –

Platform	*Terminal Application*
Windows	*PuTTY, HTerm etc. Check COMX port in Device Manager*
Linux	*cat command. eg: cat /dev/ttyACM0*
Mac	***screen*** *command, eg: screen /dev/tty.usbmodem9F31*

Table 3: Terminal Applications for various OS Environments

- For communication over BLE you can use the Serial over BLE tool at: *https://wiki.makerdiary.com/web-device-cli/*

Platform	*Terminal Application*

Windows	*Tested with PuTTY, HTerm etc. by checking COMX port in Device Manager*
Linux	*not provided need to be self-tested*
Mac	*not provided need to be self-tested*

Table 4: Tested Terminal Applications for various OS Environments

2.2.2 Running BHy2CLI in PC mode

In PC mode, the BHy2CLI application runs on a PC and communicates to the BHy Sensor Hub through a base application running on the application board.
The base application is a middleware between the PC application (BHy2CLI) and the BHy Sensor Hub Firmware (referenced in section 2.3).

Figure 5: PC Mode

Execute BHyCLI in PC Mode:

1. To prepare the application board (**<board>**: app30 or app31) for first use, connect the Application Board to a PC, and load the base firmware to the evaluation board.
    a. Execute, (Refer Figure 3 above for release package folder structure) <release>/firmware**/<board>**/coines_bridge/update_coines_bridge_flash_fw.bat
2. Reset Board.
3. Move to <release>/PC/bin folder to run BHyCLI application in PC mode.
    a. Executable present in x86/x64 folder based on 32-bit/64-bit compiler used.
    b. I2C and SPI Executables are present for BHYCLI.
    c. Refer Table below for Platform/Usage reference.

Note: As the base application is being loaded onto the external Flash of the application board, this step needs to be done only once during the first-time setup.

The BHy2CLI in PC mode is run as a terminal command, and this depends on the local operating system.

Platform	Usage
Windows	spi_bhy2cli.exe [<options>] [<option arguments>] OR spi_bhy2cli.exe [port] [COMx] [<options>] [<option arguments>]
Linux	./spi_bhy2cli [<options>] [<option arguments>]
MAC	./spi_bhy2cli [<options>] [<option arguments>]

Table 5: PC Mode execution across various OS Environments

Note: When running an application in PC mode, each prompt triggers a re-initialization of the application. As such, when multiple BHy2CLI commands need to be executed in sequence, run all the commands in a single command prompt execution, to avoid re-initializing the BHy2CLI each time, e.g.:

spi_bhy2cli.exe [<cmd_1> <cmd_1_arguments>] [<cmd_2> <cmd_2_arguments>]..[<cmd_n> <cmd_n_arguments>]

2.3 Sensor Hub Setup

The BHy, as programmable Smart Sensor Hubs, contain an embedded microcontroller that must be loaded with an appropriate firmware image to use their Smart Sensor features.

Figure 6: Smart Sensor Architecture

To load the firmware onto the BHy microcontroller, first select the relevant firmware and use the boot commands of BHy2CLI to load the firmware and boot the device. The BHy generic firmware is in the '**BHI3-firmwares**' folder of the **Release package**.

Note: *User can either use the provided example firmware located in the 'BHI3-firmwares' folder, or they can create a firmware of their own and use it. Details regarding creating a custom firmware can be found in the BHy FW SDK User Guide*

See ***Boot Commands*** for more details.

3 BHy2CLI Commands

3.1 Overview

The commands supported by the BHy2CLI can be classified as follows:

- Info
- Boot
- Register Read/Write
- Parameter Read/Write
- Data Acquisition
- Log Generation
- Use Case Specific
- System Parameters
- BSX Algorithm Parameters
- Virtual Sensor Information Parameters
- Virtual Sensor Configuration Parameters
- BSEC Parameters
- Physical Sensor Configuration
- Activity Recognition Parameters
- Data Injection
- Diagnostics
- Utility Commands

The following table lists the commands available in BHy2CLI *[v0.6.0]*.

Feature Class	*Command*	*Description*
Info	***-h OR help***	*List the available commands along with their usage*
	version	*Prints the HW, SW and FW version*
	-i OR info	*Show device information*
	-p OR physeninfo	*Display Physical Sensor Information of a physical sensor*
	schema	*Display the schema of the available sensors*
	chipid	*Get Chip ID of the sensor*
Boot	***-n OR reset***	*Reset sensor hub*
	ram	*Upload firmware to RAM*
	-g OR boot	*Boot from the specified medium -ram*
	-b OR ramb	*Reset, upload specified firmware to RAM and boot from RAM*
	-r OR rd	*Read from register*

Register Read/Write	***-w OR wr***	*Write to register*
Parameter Read/Write	***-s OR rdp***	*Read parameter*
	-t OR wrp	*Write parameter*
Data Acquisition	***-c OR actse***	*Activate/Deactivate the sensor in ascii streaming mode*
	hexse	*Activate/Deactivate the sensor in hex streaming mode*
	logse	*Activate/Deactivate the sensor in logging mode*
	dactse	*Deactivate all the active sensors*
	-a OR addse	*Register the expected payload of a new custom virtual sensor*
	lsactse	*List the active sensors and their respective acquisition modes*
Log Generation	***attlog***	*Attach (and create if required) a log file (write-only), where data can be logged to*
	detlog	*Detach the log file*
	logandstream	*Log and stream data for sensor*
Multi Tap	***mtapen***	*Enable the Multi Tap*
	mtapinfo	*Get the Multi Tap Info*
	mtapsetcnfg	*Set the Multi Tap Configurations*
	mtapgetcnfg	*Get the Multi Tap Configurations*
System Parameters	***syssetphyseninfo***	*Set system param physical sensor information*
	sysgetphysenlist	*Get list of physical sensors*
	sysgetvirsenlist	*Get list of virtual sensors*
	sysgettimestamps	*Get system timestamps*
	sysgetfwversion	*Get system firmware version*
	sysgetfifoctrl	*Get FIFO control*
	syssetwkffctrl	*Set watermark for wake-up FIFO control*
	syssetnwkffctrl	*Set watermark for Non wake-up FIFO control*
	sysgetmectrl	*Get meta event control*
	syssetmectrl	*Set meta event control*
BSX Algorithm Parameters	***setbsxparam***	*Set bsx algorithm calibration states*
	getbsxparam	*Get bsx algorithm calibration states*
	getbsxver	*Get the BSX version*
Virtual Sensor Information Parameters	***virtseinfo***	*Get virtual sensor information*
Virtual Sensor Configuration Parameters	***setvirtsenconf***	*Set virtual sensor configuration*
	getvirtsenconf	*Get virtual sensor configuration*
BSEC Parameters	***bsecsetalstate***	*Set the BSEC Algorithm State*
	bsecgetalstate	*Get the BSEC Algorithm State*
	bsecsettempoff	*Set the BSEC Temperature Offset*
	bsecgettempoff	*Get the BSEC Temperature Offset*
	bsecsetsamrate	*Set the BSEC Sample Rate*
	bsecgetsamrate	*Get the BSEC Sample Rate*
Physical Sensor Control	***accsetfoc***	*Set the Accelerometer Fast Offset Calibration*
	accgetfoc	*Get the Accelerometer Fast Offset Calibration*
	accsetpwm	*Set the Accelerometer Power Mode*
	accgetpwm	*Get the Accelerometer Power Mode*
	accsetar	*Set the Accelerometer Axis Remap*
	accgetar	*Get the Accelerometer Axis Remap*
	acctrignvm	*Trigger NVM for Accelerometer*

	accgetnvm	*Get the Accelerometer NVM Status*
	gyrosetfoc	*Set the Gyroscope Fast Offset Calibration*
	gyrogetfoc	*Get the Gyroscope Fast Offset Calibration*
	gyrosetois	*Set the Gyroscope OIS state*
	gyrogetois	*Get the Gyroscope OIS status*
	gyrosetfs	*Set the Gyroscope Fast Startup*
	gyrogetfs	*Get the Gyroscope Fast Startup status*
	gyrosetcrt	*Start Gyroscope CRT*
	gyrogetcrt	*Get the Gyroscope CRT status*
	gyrosetpwm	*Set the Gyroscope Power Mode*
	gyrogetpwm	*Get the Gyroscope Power Mode*
	gyrosettat	*Set the Gyroscope Timer Auto Trim state*
	gyrogettat	*Get the Gyroscope Timer Auto Trim status*
	gyrotrignvm	*Trigger NVM for Gyroscope*
	gyrogetnvm	*Get the Gyroscope NVM Status*
	magsetpwm	*Set the Magnetometer Power Mode*
	maggetpwm	*Get the Magnetometer Power Mode*
	wwwsetcnfg	*Set the Wrist Wear Wakeup Configuration*
	wwwgetcnfg	*Get the Wrist Wear Wakeup Configuration*
	amsetcnfg	*Set the Any Motion Configuration*
	amgetcnfg	*Get the Any Motion Configuration*
	nmsetcnfg	*Set the No Motion Configuration*
	nmgetcnfg	*Get the No Motion Configuration*
	wgdsetcnfg	*Set the Wrist Gesture Detector Configuration*
	wgdgetcnfg	*Get the Wrist Gesture Detector Configuration*
	baro1setcnfg	*Set the Barometer Pressure Type 1 Configuration*
	baro1getcnfg	*Get the Barometer Pressure Type 1 Configuration*
	baro2setcnfg	*Set the Barometer Pressure Type 2 Configuration*
	baro2getcnfg	*Get the Barometer Pressure Type 2 Configuration*
	scsetcnfg	*Set the Step Counter Configuration*
	scgetcnfg	*Get the Step Counter Configuration*
	phyrangeconf	*Set the range of physical sensor*
	foc	*Enable the fast offset compensation for the sensor*
Activity Recognition Parameters	***sethearactvcnfg***	*Set the Hearable Activity Configuration*
	gethearactvcnfg	*Get the Hearable Activity Configuration*
	setwearactvcnfg	*Set the Wearable Activity Configuration*
	getwearactvcnfg	*Get the Wearable Activity Configuration*
Head Orientation	***hmctrig***	*Trigger Head Misalignment Calibration*
	hmcsetcnfg	*Set the Head Misalignment Configuration*
	hmcgetcnfg	*Get the Head Misalignment Configuration*
	hmcsetdefcnfg	*Set the Default Head Misalignment Configuration*
	hmcver	*Get Head Misalignment Calibrator Version*
	hmcsetcalcorrq	*Set the Head Misalignment Quaternion Calibration Correction*
	hmcgetcalcorrq	*Get the Head Misalignment Quaternion Calibration Correction*
	hmcsetmode	*Set the Head Misalignment Mode and Vector X value*
	hmcgetmode	*Get the Head Misalignment Mode and Vector X value*
	hosetheadcorrq	*Set Initial Heading Correction, only for IMU Head Orientation Quaternion*
	hogetheadcorrq	*Get Initial Heading Correction, only for IMU Head Orientation Quaternion*
	hover	*Get IMU/NDOF Head Orientation Version*
	hosetheadcorre	*Set Initial Heading Correction, only for IMU Head Orientation Euler*
	hogetheadcorre	*Get Initial Heading Correction, only for IMU Head Orientation Euler*

Data Injection	***dmode***	*Set the Data Injection mode*
	dinject	*Compute virtual sensor output from raw IMU data*
Diagnostics	***-m OR postm***	*Get Post Mortem Data and log to a file*
Utility	***-v OR verb***	*Set the verbose level.*
	echo	*Enable/Disable echo*
	heart	*Enable/Disable Heartbeat Message*
	mklog	*Create a log file*
	rm	*Remove a log file*
	ls	*List the files in the external Flash*
	wrfile	*Write to a log file*
	rdfile	*Read from a log file*
	slabel	*Set a string label in the log file*
	cls	*Clear Screen*
	strbuf	*Set the streaming buffer size*

Table 6: BHy2CLI Commands

This chapter explains the usage of the commands with the help of examples to guide users on how to operate the BHy.

Note: *The outputs provided for reference are generated in* ***MCU*** *mode over* ***BLE*** *communication.*

3.2 Info Commands

The Info commands are commands oriented towards giving System and Application specific information.

Feature Class	***Command***	***Description***
Info	***-h OR help***	*List the available commands along with their usage*
	version	*Prints the HW, SW and FW version*
	-i OR info	*Show device information*
	-p OR physeninfo	*Display Physical Sensor Information of a physical sensor*
	schema	*Display the schema of the available sensors*
	chipid	*Get Chip ID of the sensor*

Table 7: Overview of Info Commands

Get Info -	
Action	***Usage***
Get BHy2CLI Application Info	*help*
Get BHy2CLI Application Version Details	*version*
Get System Info	*info*
Get Physical Sensor Information of a physical sensor	*physeninfo <phy_sensor_id>, eg: physeninfo 1*
Get schema of the available sensors	*schema*
Get Chip ID of the sensor	*chipid*

Table 8: Info Commands Usage

3.2.1 Help

```
help
Usage:
bhy2cli [<port>] [<port_name>] [<options>]
<port>: optional input parameter, trigger keyword
<port_name>: optional input parameter, use it when want to support running multiple applications in parallel
Options:
  -h OR help
        = Print this usage message
  version
        = Prints the version
  -v OR verb <verbose level>
        = Set the verbose level. 0 Error, 1 Warning, 2 Infos
  -b OR ramb <firmware path>
        = Reset, upload specified firmware to RAM and boot from RAM
          [equivalent to using "reset ram <firmware> boot r" successively]
  -n OR reset
        = Reset sensor hub
  -a OR addse <sensor id>:<sensor name>:<total output payload in bytes>:
     <output_format_0>:<output_format_1>
        = Register the expected payload of a new custom virtual sensor
         -Valid output_formats: u8: Unsigned 8 Bit, u16: Unsigned 16 Bit, u32:
          Unsigned 32 Bit, s8: Signed 8 Bit, s16: Signed 16 Bit, s32: Signed 32 Bit,
          f: Float, c: Char
         -e.g.: addse 160:"Lean Orientation":2:c:c
         -Note that the corresponding virtual sensor has to be enabled in the same function
          call (trailing actse option), since the registration of the sensor is temporary.
  -g OR boot <medium>
        = Boot from the specified <medium>: "r" for RAM
  -c OR actse <sensor id>:<frequency>[:<latency>][:<downsampling>]
        = Activate sensor <sensor id> at specified sample rate <frequency>,
         -latency <latency>, duration time <time>, sample counts <count>
         -At least <frequency> is a must input parameter
         -<latency> is optional
         -<downsampling> is optional, in case downsampling  = 0, it means there is no stream data is output
         -One or more sensors can be active by passing multiple actse options
         -id: sensor id
         -frequency(Hz): sensor ODR
         -latency(ms): sensor data outputs with a latency
         -downsampling(Hz): sensor downsampling value, it should be smaller than frequency to take effect
         -Eg:
         -actse 3:50::10
  schema
        = Show schema information:
          ID: Name: Event size: Parsing format: Axis names: Scaling
  hexse <sensor id>:<frequency>[:<latency>]
        = Stream sensor <sensor id> at specified sample rate <frequency>, in hex format
```

Figure 7: 'help' Output

3.2.2 Version

```
version
HW info:: Board: 5, HW ID: 11, Shuttle ID: 179, SW ID: 10
SW Version: 0.6.0
Build date: May 20 2025
```

Figure 8: 'version' Output

3.2.3 Info

```
info
Product ID     : 89
Kernel version : 2380
User version   : 9792
ROM version    : 5166
Power state    : sleeping
Host interface : SPI
Feature status : 0x4a
Boot Status : 0x38: No flash installed. Host interface ready. Firmware verification done.
Virtual sensor list.
Sensor ID |                        Sensor Name |  ID | Ver |  Min rate |  Max rate |
----------+------------------------------------+-----+-----+-----------+-----------|
        1 |           Accelerometer passthrough | 205 |   1 |    1.5625 |  400.0000 |
        3 |          Accelerometer uncalibrated | 203 |   1 |    1.5625 |  400.0000 |
        4 |             Accelerometer corrected | 241 |   1 |    1.5625 |  400.0000 |
        5 |               Accelerometer offset | 209 |   1 |    1.0000 |    1.0000 |
        6 |     Accelerometer corrected wake up | 192 |   1 |    1.5625 |  400.0000 |
        7 |  Accelerometer uncalibrated wake up | 204 |   1 |    1.5625 |  400.0000 |
       10 |              Gyroscope passthrough | 207 |   1 |    1.5625 |  400.0000 |
       12 |             Gyroscope uncalibrated | 244 |   1 |    1.5625 |  400.0000 |
       13 |                Gyroscope corrected | 243 |   1 |    1.5625 |  400.0000 |
```

Figure 9: 'info' Output

3.2.4 Physical Sensor Information

```
physeninfo 1
Field Name            hex                   | Value (dec)
-------------------------------------------------------
Physical Sensor ID    01                    | 1
Driver ID             1A                    | 26
Driver Version        01                    | 1
Current Consumption   01                    | 0.100mA
Dynamic Range         0008                  | 8
Flags                 22                    | IRQ status       : Disabled
                                            | Master interface : SPI0
                                            | Power mode       : Power Down
Slave Address         19                    | 25
GPIO Assignment       02                    | 2
Current Rate          00000000              | 0.000Hz
Number of axes        03                    | 3
Orientation Matrix    0100010001            | +1 +0 +0 |
                                            | +0 +1 +0 |
                                            | +0 +0 +1 |
Reserved              00                    | 0
```

Figure 10: 'physeninfo' Output

3.2.5 Schema Information

```
schema
Schema List.
ID: Name: Event size: Parse format: Axis names: Scaling
1: Accelerometer passthrough: 6: s16,s16,s16: x,y,z: 0.000244
3: Accelerometer uncalibrated: 6: s16,s16,s16: x,y,z: 0.000244
4: Accelerometer corrected: 6: s16,s16,s16: x,y,z: 0.000244
5: Accelerometer offset: 6: s16,s16,s16: x,y,z: 0.000244
6: Accelerometer corrected wake up: 6: s16,s16,s16: x,y,z: 0.000244
7: Accelerometer uncalibrated wake up: 6: s16,s16,s16: x,y,z: 0.000244
10: Gyroscope passthrough: 6: s16,s16,s16: x,y,z: 0.061035
12: Gyroscope uncalibrated: 6: s16,s16,s16: x,y,z: 0.061035
13: Gyroscope corrected: 6: s16,s16,s16: x,y,z: 0.061035
14: Gyroscope offset: 6: s16,s16,s16: x,y,z: 0.061035
15: Gyroscope wake up: 6: s16,s16,s16: x,y,z: 0.061035
16: Gyroscope uncalibrated wake up: 6: s16,s16,s16: x,y,z: 0.061035
19: Magnetometer passthrough: 6: s16,s16,s16: x,y,z: 0.076294
21: Magnetometer uncalibrated: 6: s16,s16,s16: x,y,z: 0.076294
22: Magnetometer corrected: 6: s16,s16,s16: x,y,z: 0.076294
23: Magnetometer offset: 6: s16,s16,s16: x,y,z: 0.076294
```

Figure 11: 'schema' Output

3.2.6 Chip ID

```
chipid
CHIP_ID : 0x7a
```

Figure 12: 'chipid' Output

3.3 Boot Commands

The boot commands are used to load and boot the firmware of the BHy sensor hubs.
The firmware can be loaded by the application board into the Fuser2 RAM and then booted**.**

Feature Class	*Command*	*Description*
Boot	***-n OR reset***	*Reset sensor hub*
	-u OR ram	*Upload firmware to RAM*
	-g OR boot	*Boot from the specified medium -ram*
	-b OR ramb	*Reset, upload specified firmware to RAM and boot from RAM*

Table 9: Overview of Boot Commands

To upload firmware to the sensor hub -		
Action	***Location***	***Usage***
Reset the Sensor Hub		*reset*
Upload firmware	*to RAM*	*ram <ram_firmware_path>, eg: ram Bosch_Shuttle3_BHIxyz.fw*
Boot Sensor Hub	*from RAM*	*boot r*
Alternatively, uploading the firmware and booting can be done using a single command	*from RAM*	*ramb <ram_firmware_path>, eg: ramb Bosch_Shuttle3_BHIxyz.fw*

Table 10: Boot Commands Usage

Note: *For* ***MCU*** *mode, the application board addresses the firmware path from the external Flash of the application board. The firmware is pre-loaded to the external Flash of the application board. For* ***PC mode****, the location of the firmware is passed as path.*

```
reset
Reset successful


ram Bosch_Shuttle3_BHI380_BME680.fw
Uploading 155916 bytes of firmware to RAM
Uploading firmware to RAM successful


boot r
Waiting for firmware verification to complete
Boot Status : 0x38: No flash installed. Host interface ready. Firmware verification done.
[D][META EVENT WAKE UP]; T: 0.534265625; Firmware initialized. Firmware version 5991
[D][META EVENT]; T: 0.534265625; Firmware initialized. Firmware version 5991
Booting from RAM successful
```

Figure 13: RAM Boot using 'ram' and 'boot'

```
ramb Bosch_Shuttle3_BHI380_BME680.fw
Reset successful
Uploading 155916 bytes of firmware to RAM
Uploading firmware to RAM successful
Waiting for firmware verification to complete
Boot Status : 0x38: No flash installed. Host interface ready. Firmware verification done.
[D][META EVENT WAKE UP]; T: 0.534250000; Firmware initialized. Firmware version 5991
[D][META EVENT]; T: 0.534250000; Firmware initialized. Firmware version 5991
Booting from RAM successful


boot r
Waiting for firmware verification to complete
Boot Status : 0x38: No flash installed. Host interface ready. Firmware verification done.
Booting from RAM successful
```

Figure 14: RAM Boot using 'ramb'

Note: *Fuser Core in the BHy Sensor Hub acts as the intermediary between the Host Application and the Physical Sensor on the BHy Sensor Hub. The interaction of the Fuser Core is defined by the firmware (BHy Firmware Image), running in it. As such, it is important to make sure that a firmware is loaded onto the Fuser Core. If not, use the above commands to load the firmware.*

3.4 Register Read/Write Commands

The Read/Write commands allow the user to read/write the BHy. This feature can also be extended to read/write various system and application specific parameters.

Feature Class	Command	Description
Register Read/Write	**-r OR rd**	*Read from register*
	-w OR wr	*Write to register*

Table 11: Register Read/Write Commands Overview

To explore the functionality of the register read/write commands -	
Action	***Usage***
Read from a particular register up to n bytes	*rd <reg_addr>:<len>, eg: rd 0x07:4*
Write from a particular register onwards	*wr <reg_addr>=<val1>,<val2>..., eg: wr 0x07=0x0a,0x0b,0x0c,0x0d*
Validate the write operation by reading back the written registers	*rd 0x07:4*

Table 12: Register Read/Write Commands Usage

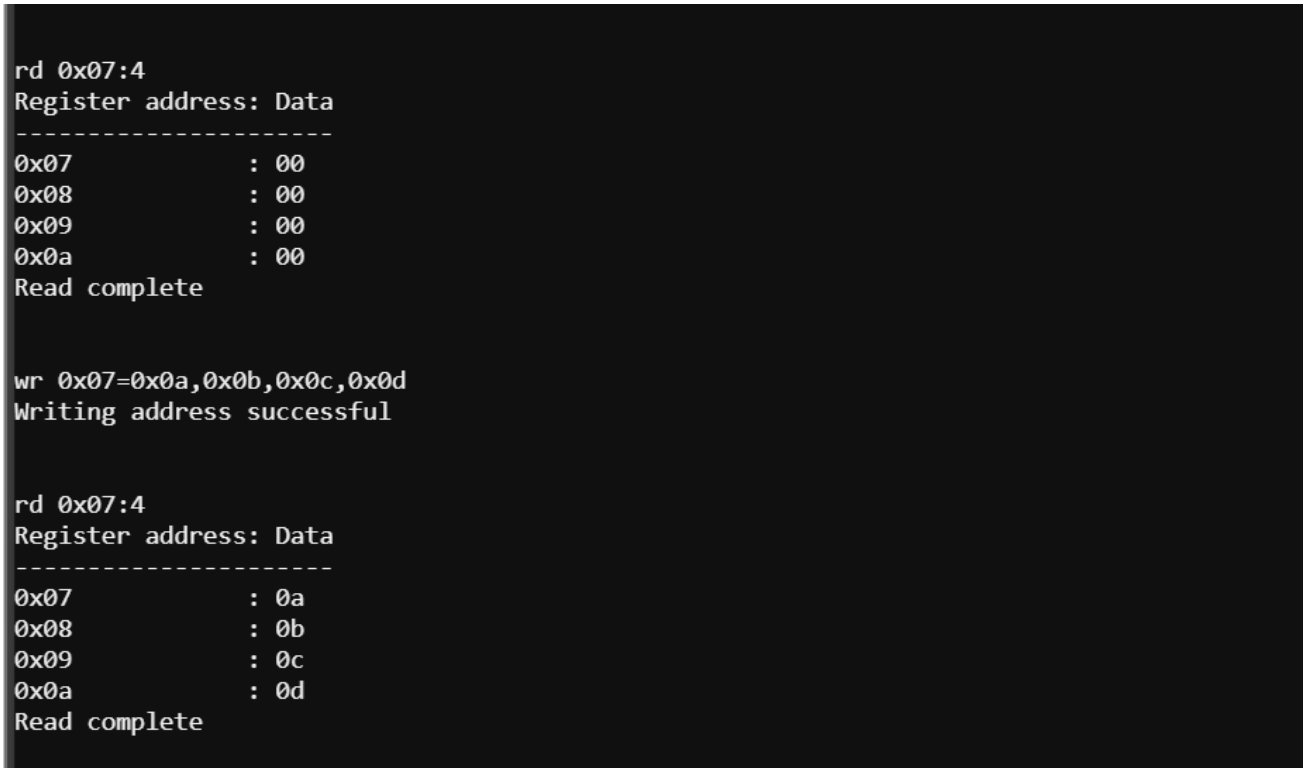


Figure 15: Read and Write from register

3.5 Parameter Read/Write Commands

The parameter interface is used to allow the configuration and query the state of the system and the sensors. The parameters differ from the normal registers because the length of data transfer is pre-determined and access to a particular set of parameters is dependent on the loaded firmware. ***[For more details, refer section 13.3 in*** *BHIxyz Datasheet* ***]***.

Feature Class	Command	Description
Parameter Read/Write	***-s OR rdp***	*Read parameter*
	-t OR wrp	*Write parameter*

Table 13: Parameter Read/Write Commands Overview

To explore the functionality of the parameter read/write commands -	
Action	***Usage***
Read a parameter	*rdp <param_id>, eg: rdp 0x103*
Write to a parameter	*wrp <param_id >=<val1>,<val2>.., eg: wrp 0x103=0x05,0x06*
Validate the write operation by reading back the written parameter	*rdp 0x103*

Table 14: Parameter Read/Write Commands Usage

```
rdp 0x103
Byte hex      dec | Data
-----------------------------------------
0x000000        0 |00 00 00 00 00 48 00 00
0x000008        8 |00 00 00 00 00 48 00 00
0x000010       16 |80 01 00 00
Reading parameter 0x0103 successful


wrp 0x103=0x05,0x06
Writing parameter successful


rdp 0x103
Byte hex      dec | Data
-----------------------------------------
0x000000        0 |05 06 00 00 00 48 00 00
0x000008        8 |00 00 00 00 00 48 00 00
0x000010       16 |80 01 00 00
Reading parameter 0x0103 successful
```

Figure 16: Read and Write from parameter

3.6 Data Acquisition Commands

Data acquisition commands allow the user to configure the ODR, latency of specific virtual sensors as well as to control output data quantity and activation duration. The tool provides provision for streaming and logging the data.

Feature Class	*Command*	*Description*
Data Acquisition	***-c OR actse***	*Activate/Deactivate the sensor in ascii streaming mode*
	hexse	*Activate/Deactivate the sensor in hex streaming mode*
	logse	*Activate/Deactivate the sensor in logging mode*
	dactse	*Deactivate all the active sensors*
	-a OR addse	*Register the expected payload of a new custom virtual sensor*
	lsactse	*List the active sensors and their respective acquisition modes*

Table 15: Data Acquisition Commands Overview

Sensor Acquisition -			
Action	***Configuration***	***Mode***	***Usage***
Reset the Sensor Hub			*reset*
Upload firmware	*to RAM*		*ramb <ram_firmware>, eg: ram Bosch_Shuttle3_BHIxyz.fw*
Enable the sensor acquisition	*single sensor*	*ascii streaming*	*actse <id>:<odr>, eg: actse 4:100*
		hex streaming	*hexse <id>:<odr>, eg: hexse 4:100*
		logging	*logse <id>:<odr>, eg: logse 4:100*
	multiple sensors	*ascii streaming*	*actse <id1>:<odr> actse <id2>:<odr>, eg: actse 4:100 actse 13:50*
		hex streaming	*hexse <id1>:<odr> hexse <id2>:<odr>, eg: hexse 4:100 hexse 13:50*
		logging	*logse <id1>:<odr> logse <id2>:<odr>, eg: logse 4:100 logse 13:50*

	latency	*ascii streaming*	*actse <id>:<odr>:<latency>, eg: actse 4:100:1000*
		hex streaming	*hexse <id>:<odr>:<latency>, eg: hexse 4:100:1000*
		logging	*logse <id>:<odr>:<latency>, eg: logse 4:100:1000*
List all the active sensors	*generic*	*all modes*	*lsactse*
Disable the sensor acquisition	*single sensor*	*ascii streaming*	*actse <id>:0, eg: actse 4:0*
		hex streaming	*hexse <id>:0, eg: hexse 4:0*
		logging	*logse <id>:0, eg: logse 4:0*
	multiple sensors	*ascii streaming*	*actse <id1>:0 actse <id2>:0, eg: actse 4:0 actse 13:0*
		hex streaming	*hexse <id1>:0 hexse <id2>:0, eg: hexse 4:0 hexse 13:0*
		logging	*logse <id1>:0 logse <id2>:0, eg: logse 4:0 logse 13:0*
	single Shot	*all modes*	*dactse*
Define parsing format for a new/custom sensor	*generic*	*all modes*	*addse<id>:"name":<payload (in bytes)>:<paring_format> eg: addse 161:"Altitude":4:s32*

Table 16: Data Acquisition Commands Usage

Note: *The usage of the* ***'logse'*** *command requires some pre-requisites, which are discussed in* ***Log Generation Commands*****.**

3.6.1 Activate sensor

3.6.1.1 Activate one virtual sensor

```
actse 4:100


[D][META EVENT]; T: 1320.715406250; Power mode changed for sensor id 4
[D][META EVENT]; T: 1320.715406250; Sample rate changed for sensor id 4
[D][META EVENT]; T: 1320.715062500; Flush complete for sensor id 4
[D]SID: 4; T: 1320.774625000; x: -0.094971, y: 0.891357, z: -0.475586; acc: 0
[D][META EVENT]; T: 1320.774625000; Accuracy for sensor id 4 changed to 1
[D]SID: 4; T: 1320.784640625; x: -0.095947, y: 0.897949, z: -0.480469; acc: 1
[D]SID: 4; T: 1320.794671875; x: -0.096680, y: 0.895508, z: -0.479980; acc: 1
[D]SID: 4; T: 1320.804687500; x: -0.095703, y: 0.897949, z: -0.478027; acc: 1
[D]SID: 4; T: 1320.814703125; x: -0.096191, y: 0.897949, z: -0.480957; acc: 1
[D]SID: 4; T: 1320.824718750; x: -0.094971, y: 0.897705, z: -0.476807; acc: 1
[D]SID: 4; T: 1320.834734375; x: -0.095703, y: 0.898682, z: -0.479736; acc: 1
[D]SID: 4; T: 1320.844750000; x: -0.096680, y: 0.895752, z: -0.479004; acc: 1
[D]SID: 4; T: 1320.854765625; x: -0.094727, y: 0.896729, z: -0.477783; acc: 1
[D]SID: 4; T: 1320.864796875; x: -0.095703, y: 0.899902, z: -0.478516; acc: 1
[D]SID: 4; T: 1320.874812500; x: -0.097656, y: 0.900635, z: -0.480713; acc: 1
```

Figure 17: Activate one virtual sensor

3.6.1.2 Activate one sensor with downsampling

```
actse 3:50::10
Sensor ID: 3, sample rate: 50.000000 Hz, latency: 0 ms


[D][META EVENT]; T: 356.202609375; Flush complete for sensor id 3
[D][META EVENT]; T: 356.203046875; Power mode changed for sensor id 3
[D][META EVENT]; T: 356.203046875; Sample rate changed for sensor id 3
[D]SID: 3; T: 356.357531250; x: 0.203125, y: 0.547363, z: -0.803711; acc: 1
[D]SID: 3; T: 356.457140625; x: 0.202881, y: 0.546143, z: -0.803711; acc: 1
[D]SID: 3; T: 356.556734375; x: 0.204102, y: 0.546143, z: -0.803223; acc: 1
[D]SID: 3; T: 356.656343750; x: 0.204346, y: 0.546143, z: -0.803223; acc: 1
[D]SID: 3; T: 356.755937500; x: 0.204102, y: 0.546631, z: -0.803711; acc: 1
[D]SID: 3; T: 356.855546875; x: 0.203857, y: 0.545898, z: -0.803467; acc: 1
[D]SID: 3; T: 356.955140625; x: 0.203857, y: 0.545898, z: -0.803711; acc: 1
[D]SID: 3; T: 357.054750000; x: 0.202881, y: 0.547607, z: -0.804199; acc: 1
```

Figure 18: Activate sensor with downsampling

3.6.1.3 Activate multiple sensors

```
actse 4:25 actse 13:25



[D][META EVENT]; T: 10.640328125; Power mode changed for sensor id 4
[D][META EVENT]; T: 10.640328125; Sample rate changed for sensor id 4
[D][META EVENT]; T: 10.639859375; Flush complete for sensor id 4
[D][META EVENT]; T: 10.658671875; Power mode changed for sensor id 13
[D][META EVENT]; T: 10.658671875; Sample rate changed for sensor id 13
[D][META EVENT]; T: 10.657890625; Flush complete for sensor id 13
[D]SID: 4; T: 10.840546875; x: -0.087402, y: 0.917480, z: 0.518066; acc: 0
[D][META EVENT]; T: 10.840546875; Accuracy for sensor id 4 changed to 0
[D]SID: 13; T: 10.840546875; x: 0.305176, y: 0.061035, z: -0.183105; acc: 0
[D][META EVENT]; T: 10.840546875; Accuracy for sensor id 13 changed to 0
[D]SID: 4; T: 10.880734375; x: -0.088379, y: 0.926025, z: 0.523438; acc: 0
[D]SID: 13; T: 10.880734375; x: 0.915527, y: 0.000000, z: -0.244141; acc: 0
[D]SID: 4; T: 10.920937500; x: -0.087891, y: 0.924072, z: 0.522217; acc: 0
[D]SID: 13; T: 10.920937500; x: 2.868652, y: 0.427246, z: -0.305176; acc: 0
```

Figure 19: Activate multiple sensors

3.6.1.4 Activate sensor in HEX mode

```
hexse 1:10


[D][META EVENT]; T: 1083.120734375; Flush complete for sensor id 1
[D][META EVENT]; T: 1083.121265625; Power mode changed for sensor id 1
[D][META EVENT]; T: 1083.121265625; Sample rate changed for sensor id 1
[H]010000043b1ae50d3e5002c8008710
[H]010000043b1fa853085602cc00ae10
[H]010000043b246bd5db5502cb00a510
[H]010000043b292f1ba55702c900a510
[H]010000043b2df29e785302ca00a710
[H]010000043b32b6214b5602c800a510
[H]010000043b3779e1275502ca00a610
[H]010000043c00a299fa5502c600a410
[H]010000043c056659d65402c900a210
[H]010000043c0a29dca95402cc00a510
[H]010000043c0eed9c855802cb00a410
[H]010000043c13b15c615302c900a510
```

Figure 20: Activate sensor in HEX streaming mode

3.6.1.5 Activate virtual sensor with latency

```
actse 4:200:1000


[D][META EVENT]; T: 294.314843750; Power mode changed for sensor id 4
[D][META EVENT]; T: 294.314843750; Sample rate changed for sensor id 4
[D][META EVENT]; T: 294.314437500; Flush complete for sensor id 4
[D]SID: 4; T: 294.350781250; x: -0.050781, y: 0.796387, z: 0.623535; acc: 1
[D][META EVENT]; T: 294.350781250; Accuracy for sensor id 4 changed to 1
[D]SID: 4; T: 294.355796875; x: -0.051025, y: 0.801514, z: 0.628174; acc: 1
[D]SID: 4; T: 294.360796875; x: -0.052246, y: 0.801025, z: 0.627441; acc: 1
[D]SID: 4; T: 294.365812500; x: -0.053223, y: 0.802490, z: 0.626953; acc: 1
[D]SID: 4; T: 294.370812500; x: -0.053223, y: 0.802490, z: 0.626953; acc: 1
```

Figure 21: Activate sensor with latency

The parameter "*latency*" can control the activated virtual sensor to output sensor data with a delay. The default unit of this parameter is milliseconds. The parameter is optional, and the default is 0ms if not specified.

3.6.2 List Active sensors

```
actse 50:1


[D][META EVENT]; T: 12.897625000; Power mode changed for sensor id 50
[D][META EVENT]; T: 12.897625000; Sample rate changed for sensor id 50
[D][META EVENT]; T: 12.897203125; Flush complete for sensor id 50

logse 52:1


[D][META EVENT]; T: 18.051250000; Power mode changed for sensor id 52
[D][META EVENT]; T: 18.051250000; Sample rate changed for sensor id 52
[D][META EVENT]; T: 18.050453125; Flush complete for sensor id 52

lsactse
Active Sensors -
SID :  50        ODR : 1.00      R :    1        Acquisition : Streaming
SID :  52        ODR : 1.00      R :    1        Acquisition : Logging
No File attached for Logging
```

Figure 22: List the active sensors

3.6.3 Deactivate virtual sensor

```
actse 50:1


[D][META EVENT]; T: 980.431968750; Flush complete for sensor id 50
[D]SID: 50; T: 982.658843750;
[D]SID: 50; T: 983.106703125;
[D]SID: 50; T: 983.574937500;
[D]SID: 50; T: 984.470421875;
[D]SID: 50; T: 985.203000000;

actse 50:0


[D][META EVENT]; T: 993.326859375; Flush complete for sensor id 50
```

Figure 23: Deactivate sensor using actse

```
lsactse
Active Sensors -
SID :  50        ODR : 1.00      R :    1        Acquisition : Streaming
SID :  52        ODR : 1.00      R :    1        Acquisition : Logging
No File attached for Logging

dactse
Deactivated all the Sensors
[D][META EVENT]; T: 207.311296875; Power mode changed for sensor id 50
[D][META EVENT]; T: 207.311296875; Sample rate changed for sensor id 50
[D][META EVENT]; T: 207.315750000; Power mode changed for sensor id 52
[D][META EVENT]; T: 207.315750000; Sample rate changed for sensor id 52

lsactse
No Active Sensors
No File attached for Logging
```

Figure 24: Deactivate sensor using dactse

3.6.4 Custom/New Virtual Sensor

For a new virtual sensor or for any virtual sensor that is not supported in the BHy2CLI application, it is still possible to use the virtual sensor in BHy2CLI by using the *'addse'* command.

Feature Class	*Command*	*Description*
Custom/New Virtual Sensor Support	***addse***	*Extend support for a new/custom virtual sensor, which is not yet supported in the BHy2CLI application*

Table 17: Custom/New Virtual Sensor support Extension

Adding support for Custom/New Virtual Sensor -	
Action	***Usage***
Reset the sensor hub	*reset*
Load the custom firmware with support for custom/new virtual sensor	*ramb <custom_firmware> or <flb <custom_firmware>*
Check if support available for custom/virtual by reading the list of virtual sensors	*info*
Extend the support for the Custom/New virtual sensor in BHy	*addse <sensor_id>:"<sensor_name>":<parse_size>:<parse_format>* *eg: addse161:"Test_Sensor ":4:s32*
Enable the custom/new virtual sensor	*actse <custom_sensor_id>, eg: actse 161:50*

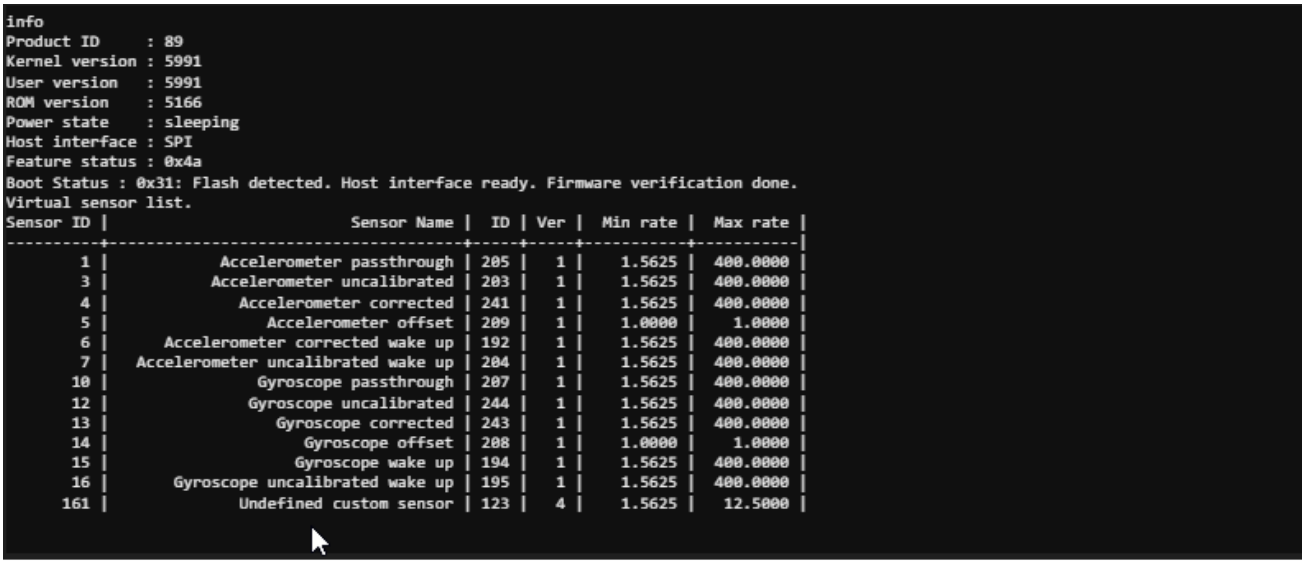
Table 18: Adding support for Custom/New Virtual Sensor

```
addse 161:"test_sensor":4:s32
Adding custom driver payload successful


actse 161:10


[D][META EVENT]; T: 1595.970046875; Flush complete for sensor id 161
[D][META EVENT]; T: 1595.971312500; Power mode changed for sensor id 161
[D][META EVENT]; T: 1595.971312500; Sample rate changed for sensor id 161
[D]161; 1596.051500000; 57050
[D]161; 1596.131437500; 57022
[D]161; 1596.211375000; 57036
[D]161; 1596.291312500; 57029
[D]161; 1596.371250000; 57022
[D]161; 1596.451187500; 57043
[D]161; 1596.531125000; 57036
```

Figure 25: Custom/New Virtual Sensor Detected



```
info
Product ID     : 89
Kernel version : 5991
User version   : 5991
ROM version    : 5166
Power state    : sleeping
Host interface : SPI
Feature status : 0x4a
Boot Status : 0x31: Flash detected. Host interface ready. Firmware verification done.
Virtual sensor list.
Sensor ID |                        Sensor Name |  ID | Ver |  Min rate |  Max rate |
----------+------------------------------------+-----+-----+-----------+-----------|
        1 |           Accelerometer passthrough | 205 |   1 |    1.5625 |  400.0000 |
        3 |         Accelerometer uncalibrated | 203 |   1 |    1.5625 |  400.0000 |
        4 |            Accelerometer corrected | 241 |   1 |    1.5625 |  400.0000 |
        5 |               Accelerometer offset | 209 |   1 |    1.0000 |    1.0000 |
        6 |     Accelerometer corrected wake up | 192 |   1 |    1.5625 |  400.0000 |
        7 |  Accelerometer uncalibrated wake up | 204 |   1 |    1.5625 |  400.0000 |
       10 |              Gyroscope passthrough | 207 |   1 |    1.5625 |  400.0000 |
       12 |            Gyroscope uncalibrated | 244 |   1 |    1.5625 |  400.0000 |
       13 |               Gyroscope corrected | 243 |   1 |    1.5625 |  400.0000 |
       14 |                  Gyroscope offset | 208 |   1 |    1.0000 |    1.0000 |
       15 |                 Gyroscope wake up | 194 |   1 |    1.5625 |  400.0000 |
       16 |      Gyroscope uncalibrated wake up | 195 |   1 |    1.5625 |  400.0000 |
      161 |            Undefined custom sensor | 123 |   4 |    1.5625 |   12.5000 |
```

Figure 26: Custom/New virtual Sensor Acquisition

3.7 Log Generation Commands

The log generation commands are used in conjunction with the 'logse' command to acquire the data in logging mode and store the log in the external Flash.

Feature Class	*Command*	*Description*
Log Generation	***attlog***	*Attach (and create if required) a log file (write-only), where data can be logged to*
	detlog	*Detach the log file*
	logandstream	*Log and stream data for sensor*

Table 19: Log Generation Commands Overview

Sensor in Logging mode -	
Action	***Usage***
Attach a log file for the logging	*attlog <file_name.file_extension>, eg: attlog abc.bin*
Enable the sensor acquisition in logging mode	*logse <id>:<odr>, eg: logse 4:100*
Disable the sensor acquisition	*logse <id>:0, eg: logse 4:0*
Detach log file from logging	*detlog <file_name.file_extension>, eg: detlog abc.bin*
Log and stream data for sensor with downsampling	*logandstream <sensor id>:<frequency>[:<latency>][:<downsampling>] <filename> <start>* *or logandstream <stop>* *eg: logandstream 3:50::10 4:100::20 log.bin start* *logandstream stop*

Table 20: Log Generation Commands Usage

```
attlog abc.bin
File abc.bin was created


logse 4:100


[D][META EVENT]; T: 363.048609375; Flush complete for sensor id 4
[D][META EVENT]; T: 363.049187500; Power mode changed for sensor id 4
[D][META EVENT]; T: 363.049187500; Sample rate changed for sensor id 4
[D][META EVENT]; T: 363.111359375; Accuracy for sensor id 4 changed to 0
[D][META EVENT]; T: 364.246390625; Accuracy for sensor id 4 changed to 1

lsactse
Active Sensors -
SID :   4       ODR : 100.00    R :    8        Acquisition : Logging
Attached Log File : abc.bin


logse 4:0


[D][META EVENT]; T: 393.928468750; Power mode changed for sensor id 4
[D][META EVENT]; T: 393.928468750; Sample rate changed for sensor id 4
[D][META EVENT]; T: 393.928093750; Flush complete for sensor id 4

detlog abc.bin
File abc.bin was detached for logging


lsactse
No Active Sensors
No File attached for Logging
```

Figure 27: Log Generation Output

```
logandstream 4:50::10 log.bin start
Executing log and stream together
Creating log.bin
File log.bin was created
[D]Streaming sensor ID 4 with sample rate is 10.000000Hz
Sensor ID: 4, sample rate: 50.000000 Hz, latency: 0 ms
[D][META EVENT]; T: 36.844421875; Flush complete for sensor id 4
[D][META EVENT]; T: 36.844875000; Power mode changed for sensor id 4
[D][META EVENT]; T: 36.844875000; Sample rate changed for sensor id 4
[D][META EVENT]; T: 36.951203125; Accuracy for sensor id 4 changed to 0
[D]SID: 4; T: 36.951203125; x: 0.007568, y: -0.146729, z: 1.004150; acc: 0
[D]SID: 4; T: 37.050781250; x: 0.009277, y: -0.147217, z: 1.000732; acc: 0
[D]SID: 4; T: 37.150343750; x: 0.010742, y: -0.146484, z: 1.002197; acc: 0
[D]SID: 4; T: 37.249906250; x: 0.011230, y: -0.146240, z: 1.001465; acc: 0
[D]SID: 4; T: 37.349468750; x: 0.011719, y: -0.146729, z: 1.002197; acc: 0
```

Figure 28: 'logandstream' Command Output

3.8 Use Case Specific Commands

The use case specific commands allow the user to read/write and configure the various controls and parameters for various use case applications such as Multi-tap, etc.

Note: *Before executing various Use Case Specific commands, ensure that firmware for the respective Use Case Applications is flashed onto the BHy Sensor Hub.*

3.8.1 Multi Tap

The Multi-Tap sensor detects single, double and triple taps. The following commands configure and enable/ disable the various aspects of the Multi-Tap sensor.

Feature Class	Command	Description
Multi Tap	***mtapen***	*Enable the Multi Tap*
	mtapinfo	*Get the Multi Tap Info*
	mtapsetcnfg	*Set the Multi Tap Configurations*
	mtapgetcnfg	*Get the Multi Tap Configurations,*

Table 21: Multi-Tap Commands Overview

Multi-Tap commands usage -	
Command	***Usage***
mtapen	*mtapen <tap_setting>, eg: mtapen 2*
mtapinfo	*mtapinfo*
mtapsetcnfg	*mtapsetcnfg <single_tap_config> <double_tap_config> <triple_tap_config>* *eg: mtapsetcnfg 0x0080 0x4022 0x6862* ***Note**: The single/double/triple tap configurations passed as arguments to this command are in combined format. Refer above table for specifics. Refer datasheet for bit mapping*
mtapgetcnfg	*mtapgetcnfg*

Table 22: Multi-Tap Commands Usage

Please refer to **Multi-Tap Application Note** for more details on the Multi-Tap application.

```
mtapinfo
Multi Tap Info : TRIPLE_DOUBLE_SINGLE_TAP

mtapen 2
Multi Tap Parameter set to  DOUBLE_TAP

mtapinfo
Multi Tap Info : DOUBLE_TAP

mtapgetcnfg
Single Tap CNFG : 0x0076
                  -<axis_sel> : 2
                  -<wait_for_timeout> : 1
                  -<max_pks_for_tap> : 6
                  -<mode> : 1
Double Tap CNFG : 0x402d
                  -<tap_peak_thrs> : 45
                  -<max_ges_dur> : 16
Triple Tap CNFG : 0x6864
                  -<max_dur_bw_pks> : 4
                  -<tap_shock_settl_dur> : 6
                  -<min_quite_dur_bw_taps> : 8
                  -<quite_time_after_ges> : 6


mtapsetcnfg 0x0072 0x4022 0x6862
Multi Tap Detector Parameter set successfully

mtapgetcnfg
Single Tap CNFG : 0x0072
                  -<axis_sel> : 2
                  -<wait_for_timeout> : 0
                  -<max_pks_for_tap> : 6
                  -<mode> : 1
Double Tap CNFG : 0x4022
                  -<tap_peak_thrs> : 34
                  -<max_ges_dur> : 16
Triple Tap CNFG : 0x6862
                  -<max_dur_bw_pks> : 2
                  -<tap_shock_settl_dur> : 6
                  -<min_quite_dur_bw_taps> : 8
                  -<quite_time_after_ges> : 6
```

Figure 29: Multi-Tap Commands Output

3.8.2 Head Orientation

The head orientation tracking software processes the raw sensor input to estimate head orientation information with respect to the Earth reference system. It combines functional modules and a set of calibration routines to correct the original sensor input, IMU and NDOF fusion algorithms, misalignment estimation between the head and device coordinate system and the rotation transformation between different reference systems.

Feature Class	Command	Description
Head Orientation	***hmctrig***	*Trigger Head Misalignment Calibration*
	hmcsetcnfg	*Set the Head Misalignment Configuration*
	hmcgetcnfg	*Get the Head Misalignment Configuration*
	hmcsetdefcnfg	*Set the Default Head Misalignment Configuration*

	hmcver	*Get Head Misalignment Calibrator Version*
	hmcsetcalcorrq	*Set the Head Misalignment Quaternion Calibration Correction*
	hmcgetcalcorrq	*Get the Head Misalignment Quaternion Calibration Correction*
	hmcsetmode	*Set the Head Misalignment Mode and Vector X value*
	hmcgetmode	*Get the Head Misalignment Mode and Vector X value*
	hosetheadcorrq	*Set Initial Heading Correction, only for IMU Head Orientation Quaternion*
	hogetheadcorrq	*Get Initial Heading Correction, only for IMU Head Orientation Quaternion*
	hover	*Get IMU/NDOF Head Orientation Version*
	hosetheadcorre	*Set Initial Heading Correction, only for IMU Head Orientation Euler*
	hogetheadcorre	*Get Initial Heading Correction, only for IMU Head Orientation Euler*

Table 23: Head Orientation Commands Overview

Head Orientation commands usage -	
Command	***Usage***
hmcver	*hmcver*
hover	*hover*
hmcsetcnfg	*hmcsetcnfg <sp_max_dur> <sp_min_dur> <dp_max_samples> <acc_diff_th>* *eg: hmcsetcnfg 0x06 0x02 0x32 0x00002042*
hmcgetcnfg	*hmcgetcnfg*
hmcsetdefcnfg	*hmcsetdefcnfg*
hmcsetdefcnfg	*hmcsetdefcnfg*
hmctrig	*hmctrig*
hmcsetcalcorrq	*hmcsetcalcorrq <quat_x> <quat_y> <quat_z> <quat_w>* *eg: hmcsetcalcorrq 1.000 1.000 1.000 1.000*
hmcgetcalcorrq	*hmcgetcalcorrq*
hmcsetmode	*hmcsetmode <mode> <x[0]> <x[1]> <x[2]>* *eg. hmcsetmode 0 0.0 0.0 1.0*
hmcgetmode	*hmcgetmode*
hosetheadcorrq	*hosetheadcorrq <enable/disable>* *eg: hosetheadcorrq 1*
hogetheadcorrq	*hogetheadcorrq*
hosetheadcorre	*hosetheadcorre <enable/disable>* *eg: hosetheadcorre 1*
hogetheadcorre	*hogetheadcorre*

Table 24: Head Orientation Commands Usage

Please refer to ***Head Orientation Application Note*** for more details on the Head Orientation application.

3.9 System Parameters Commands

The following commands allow the host to set or get system parameters.

Feature Class	***Command***	***Description***
System Parameters	***syssetphyseninfo***	*Set system param physical sensor information*
	sysgetphysenlist	*Get list of physical sensors*
	sysgetvirsenlist	*Get list of virtual sensors*
	sysgettimestamps	*Get system timestamps*
	sysgetfwversion	*Get system firmware version*
	sysgetfifoctrl	*Get FIFO control*
	syssetwkffctrl	*Set watermark for wake-up FIFO control*
	syssetnwkffctrl	*Set watermark for Non wake-up FIFO control*

	sysgetmectrl	*Get meta event control*
	syssetmectrl	*Set meta event control*

Table 25: System Parameters Commands Overview

System Parameters commands usage -	
Command	***Usage***
syssetphyseninfo	*syssetphyseninfo <sensor_id> <orientation_matrix>* *eg. syssetphyseninfo 1 0,-1,0,1,0,0,0,0,1*
sysgetphysenlist	*sysgetphysenlist*
sysgetvirsenlist	*sysgetvirsenlist*
sysgettimestamps	*sysgettimestamps*
sysgetfwversion	*sysgetfwversion*
sysgetfifoctrl	*sysgetfifoctrl*
syssetwkffctrl	*syssetwkffctrl <watermark value>* *eg. syssetwkffctrl 500*
syssetnwkffctrl	*syssetnwkffctrl <watermark value>* *eg. syssetnwkffctrl 500*
sysgetmectrl	*sysgetmectrl <address>* *eg. sysgetmectrl 0x101*
syssetmectrl	*syssetmectrl <address> <group idx> <value>* *eg. syssetmectrl 0x101 1 1*

Table 26: System Parameters Commands Usage

```
sysgetvirsenlist
Virtual sensor list.
Sensor ID |                              Sensor Name |
----------+------------------------------------------|
        1 |            Accelerometer passthrough
        3 |          Accelerometer uncalibrated
        4 |             Accelerometer corrected
        5 |                Accelerometer offset
        6 |     Accelerometer corrected wake up
        7 |  Accelerometer uncalibrated wake up
       10 |               Gyroscope passthrough
       12 |             Gyroscope uncalibrated
       13 |                Gyroscope corrected
       14 |                   Gyroscope offset
       15 |                 Gyroscope wake up
       16 |      Gyroscope uncalibrated wake up
       28 |                       Gravity vector
       29 |               Gravity vector wake up
       31 |                  Linear acceleration
       32 |         Linear acceleration wake up
       37 |                 Game rotation vector
       38 |        Game rotation vector wake up
       43 |                          Orientation
       44 |                 Orientation wake up
      136 |              Low Power Step counter
      137 |            Low Power Step detector
      143 |        Low Power Any motion wake up
      153 |                 Multi Tap Detector
      154 |  Activity recognition for Wearables
      156 |     Low Power Wrist Gesture wake up
      158 |        Low Power Wrist Wear wake up
      159 |         Low Power No Motion wake up
```

```
sysgettimestamps
Host interrupt timestamp: 383.323375000
Current timestamp: 457.535703125
Timestamp event: 0.000000000


sysgetfwversion

Custom version number: 0
EM Hash: 1755aba11aa9
BST Hash: ac49f211
User Hash: ac49f211


sysgetfifoctrl

Wakeup FIFO Watermark = 0
Wakeup FIFO size =  17920
Non Wakeup FIFO Watermark = 0
Non Wakeup FIFO size = 18432
```

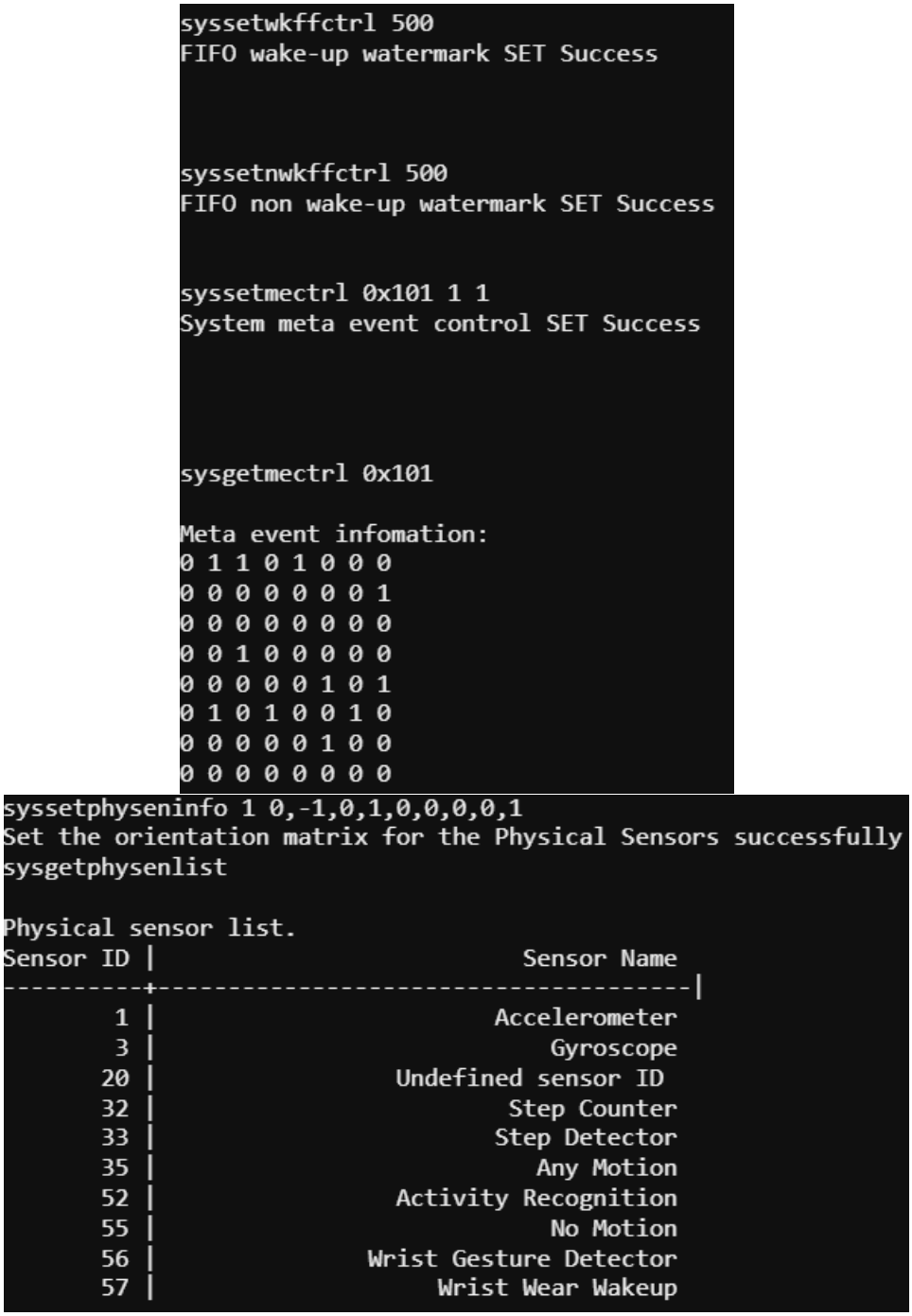


Figure 30: System Parameters Commands Output

3.10 BSX Algorithm Parameters Commands

The following commands allow the host to set or get BSX Algorithm parameters.

Feature Class	*Command*	*Description*
BSX Algorithm Parameters	***setbsxparam***	*Set bsx algorithm calibration states*
	getbsxparam	*Get bsx algorithm calibration states*
	getbsxver	*Get the BSX version*

Table 27: BSX Algorithm Parameters Commands Overview

BSX Algorithm commands usage -

Command	*Usage*
setbsxparam	*setbsxparam <parameter_id> or <parameter_id> <file_name>* *eg. setbsxparam 0x201*
getbsxparam	*getbsxparam <parameter_id> or <parameter_id> <file_name>* *eg. getbsxparam 0x201*
getbsxver	*getbsxver*

Table 28: BSX Algorithm Parameters Commands Usage

```
getbsxparam 0x201


Get Calibration profile of BSX parameter id: 0x0201
Block number: 0
Completion flag: 0, Transfer not complete
Block length: 64
Struct length: 72
----------------------------------------
Block data
Byte Hex      Dec | Data
----------------------------------------
0x000000        0 |03 54 00 04 00 30 00 40
0x000008        8 |04 00 6f 00 00 00 3e 00
0x000010       16 |00 00 0b 30 10 41 00 01
0x000018       24 |00 00 78 30 00 42 00 c2
0x000020       32 |00 03 0b 00 08 00 00 04
0x000028       40 |00 00 06 30 10 41 00 01
0x000030       48 |00 00 78 30 00 42 00 c2
0x000038       56 |00 03 06 00 08 00 00 02

Block number: 1
Completion flag: 1, Transfer complete
Block length: 8
Struct length: 72
----------------------------------------
Block data
Byte Hex      Dec | Data
----------------------------------------
0x000000        0 |00 d1 00 00 20 00 8f 00


getbsxparam 0x201 acc.txt


Get Calibration profile of BSX parameter id: 0x0201

Calibration profile for BSX parameter id 0x0201 is saved to the file acc.txt
```

```
setbsxparam 0x201


Setting Calibration profile of BSX parameter id 0x0201 is completed
```

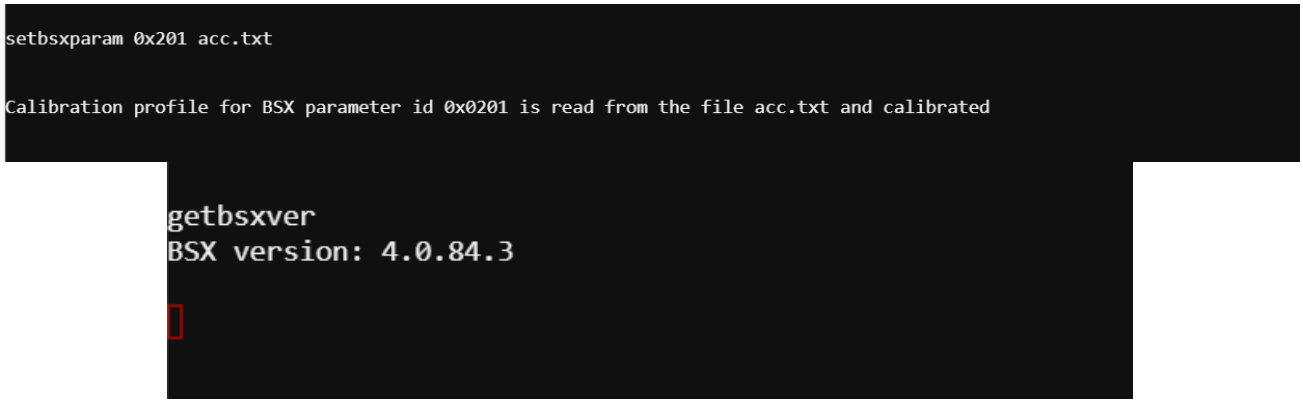


Figure 31: BSX Algorithm Parameters Commands Output

3.11 Virtual Sensor Information Parameters Commands

The following command allows the host to get virtual sensor information parameters.

Feature Class	Command	Description
Virtual Sensor Information Parameters	***virtseinfo***	*Get virtual sensor information*

Table 29: Virtual Sensor Information Parameters Command Overview

Virtual Sensor Information Parameters commands usage -	
Command	***Usage***
virtseinfo	*virtseinfo <sensor_id>*

Table 30: Virtual Sensor Information Parameters Command Usage

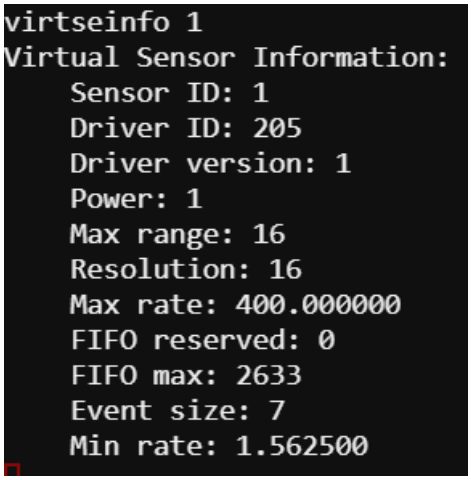


Figure 32: Virtual Sensor Information Command Output

3.12 Virtual Sensor Configuration Parameters Commands

The following commands allow the host to set or get Virtual Sensor Configuration Parameters.

Feature Class	Command	Description
Virtual Sensor Configuration Parameters	***setvirtsenconf***	*Set virtual sensor configuration*
	getvirtsenconf	*Get virtual sensor configuration*

Table 31: Virtual Sensor Configuration Parameters Commands Overview

Virtual Sensor Configuration Parameters commands usage -	
Command	***Usage***
setvirtsenconf	*setvirtsenconf <sensor_id> <sample_rate> <latency>* *eg. setvirtsenconf 3 10 0*
getvirtsenconf	*getvirtsenconf <sensor_id>*

Table 32: Virtual Sensor Configuration Parameters Commands Usage

```
setvirtsenconf 3 10 0
Virtual sensor configuration set successfully
[D][META EVENT]; T: 296.813750000; Power mode changed for sensor id 3
[D][META EVENT]; T: 296.813750000; Sample rate changed for sensor id 3
getvirtsenconf 3
Virtual sensor configuration get successfully
Custom sensor ID=3, sensitivity=0, rate=12.50Hz,latency=0, range=8
```

Figure 33: Virtual Sensor Configuration Parameters Commands

3.13 Physical Sensor Configuration Commands

The following commands allow the host to set or get sensor control information of the physical sensors.

Feature Class	***Command***	***Description***
Physical Sensor Control	***accsetfoc***	*Set the Accelerometer Fast Offset Calibration*
	accgetfoc	*Get the Accelerometer Fast Offset Calibration*
	accsetpwm	*Set the Accelerometer Power Mode*
	accgetpwm	*Get the Accelerometer Power Mode*
	accsetar	*Set the Accelerometer Axis Remap*
	accgetar	*Get the Accelerometer Axis Remap*
	acctrignvm	*Accelerometer Trigger NVM*
	accgetnvm	*Get the Accelerometer NVM Status*
	gyrosetfoc	*Set the Gyroscope Fast Offset Calibration*
	gyrogetfoc	*Get the Gyroscope Fast Offset Calibration*
	gyrosetois	*Set the Gyroscope OIS state*
	gyrogetois	*Get the Gyroscope OIS status*
	gyrosetfs	*Set the Gyroscope Fast Startup*
	gyrogetfs	*Get the Gyroscope Fast Startup status*
	gyrosetcrt	*Start Gyroscope CRT*
	gyrogetcrt	*Get the Gyroscope CRT status*
	gyrosetpwm	*Set the Gyroscope Power Mode*
	gyrogetpwm	*Get the Gyroscope Power Mode*
	gyrosettat	*Set the Gyroscope Timer Auto Trim state*
	gyrogettat	*Get the Gyroscope Timer Auto Trim status*
	gyrotrignvm	*Gyroscope Trigger NVM*
	gyrogetnvm	*Get the Gyroscope NVM Status*
	magsetpwm	*Set the Magnetometer Power Mode*
	maggetpwm	*Get the Magnetometer Power Mode*
	wwwsetcnfg	*Set the Wrist Wear Wakeup Configuration*
	wwwgetcnfg	*Get the Wrist Wear Wakeup Configuration,*
	amsetcnfg	*Set the Any Motion*
	amgetcnfg	*Get the Any Motion Configuration*
	nmsetcnfg	*Set the No Motion Configuration*
	nmgetcnfg	*Get the No Motion Configuration*
	wgdsetcnfg	*Set the Wrist Gesture Detector Configuration*
	wgdgetcnfg	*Get the Wrist Gesture Detector Configuration*
	baro1setcnfg	*Set the Barometer Pressure Type 1 Configuration*

	baro1getcnfg	*Get the Barometer Pressure Type 1 Configuration*
	baro2setcnfg	*Set the Barometer Pressure Type 2 Configuration*
	baro2getcnfg	*Get the Barometer Pressure Type 2 Configuration*
	scsetcnfg	*Set the Step Counter Configuration*
	scgetcnfg	*Get the Step Counter Configuration*
	phyrangeconf	*Set the range of physical sensor*
	foc	*Enable the fast offset compensation for the sensor*

Table 33: Physical Sensor Control Parameter Commands Overview

3.13.1 Accelerometer

The following commands allow the user to configure the accelerometer sensor.

Feature Class	*Command*	*Description*
Accelerometer Sensor Control	***accsetfoc***	*Set the Accelerometer Fast Offset Calibration*
	accgetfoc	*Get the Accelerometer Fast Offset Calibration*
	accsetpwm	*Set the Accelerometer Power Mode*
	accgetpwm	*Get the Accelerometer Power Mode*
	accsetar	*Set the Accelerometer Axis Remap*
	accgetar	*Get the Accelerometer Axis Remap*
	acctrignvm	*Trigger NVM for Accelerometer*
	accgetnvm	*Get the Accelerometer NVM Status*

Table 34: Accelerometer Control Parameter Commands Overview

Parameter	*Description*
Fast Offset Calibration	*FOC is a one-shot process that compensates errors of the sensor by setting the compensation registers to the negated offset error*
Power Mode	*The power mode of the accelerometer can be configured to operate in other power modes in order to facilitate use case appropriate performance. The accelerometer sensor has 2 power modes –* *0 - Normal Mode, The default state of the sensor* *2 - Low Power, Low power mode on the account of trade-of of power consumption versus sensor noise. Low power mode may introduce aliasing artifacts*
Axis remapping for internal imu features	*The axis remapping function is only works for the features in BHIxyz, and it would not influence the IMU outputs (accel data and gyro data).* *It can change the sign for each axis and switch between each axis.*
Triggering a NVM writing	*Once NVM writing process is triggered, the value in backup registers value(offset/crt) will be written into NVM area.*

Table 35: Accelerometer Control Parameters

Accelerometer Parameter commands usage -	
Command	***Usage***
accsetfoc	*accsetfoc <x_offset> <y_offset> <z_offset>, eg: accsetfoc 0x0016 0x00f8 0x07f*
accgetfoc	*accgetfoc*
accsetpwm	*accsetpwm <power_mode>, eg: accsetpwm 2*
accgetpwm	*accgetpwm*
accsetar	*accsetar <x> <x_sign> <y> <y_sign> <z> <z_sign>* *eg: accsetar 1 1 1 1 1 1*
accgetar	*accgetar*
acctrignvm	*acctrignvm*
accgetnvm	*accgetnvm*

Table 36: Accelerometer Control Parameter Commands Usage

```
accsetfoc 0x0072 0x0064 0x007F
Set the Accelerometer Fast Offset Calibration
        -<x_offset> : 114
        -<y_offset> : 100
        -<z_offset> : 127
accgetfoc
Accelerometer Fast Offset Calibration :
        -<x_offset> : 114
        -<y_offset> : 100
        -<z_offset> : 127
accsetpwm 0
Set the Accelerometer Power Mode to NORMAL
accgetpwm
Accelerometer Power Mode : NORMAL
accsetar 1 1 1 1 1 1
Set the Accelerometer axis remapping
        -<x> : 1
        -<x_sign> : 1
        -<y> : 1
        -<y_sign> : 1
        -<z> : 1
        -<z_sign> : 1
accgetar
Accelerometer axis remapping :
        -<x> : 1
        -<x_sign> : 1
        -<y> : 1
        -<y_sign> : 1
        -<z> : 1
        -<z_sign> : 1
acctrignvm
Trigger a NVM writing for accelerometer
accgetnvm
NVM writing status for accelerometer: Done
```

Figure 34: Accelerometer Control Parameter Commands Output

3.13.2 Gyroscope

The following commands allow the user to configure the gyroscope sensor.

Feature Class	*Command*	*Description*
Gyroscope Sensor Control	***gyrosetfoc***	*Set the Gyroscope Fast Offset Calibration*
	gyrogetfoc	*Get the Gyroscope Fast Offset Calibration*
	gyrosetois	*Set the Gyroscope OIS state*
	gyrogetois	*Get the Gyroscope OIS status*
	gyrosetfs	*Set the Gyroscope Fast Startup*
	gyrogetfs	*Get the Gyroscope Fast Startup status*
	gyrosetcrt	*Start Gyroscope CRT*
	gyrogetcrt	*Get the Gyroscope CRT status*
	gyrosetpwm	*Set the Gyroscope Power Mode*
	gyrogetpwm	*Get the Gyroscope Power Mode*
	gyrosettat	*Set the Gyroscope Timer Auto Trim state*
	gyrogettat	*Get the Gyroscope Timer Auto Trim status*
	gyrotrignvm	*Trigger NVM for Gyroscope*

	gyrogetnvm	*Get the Gyroscope NVM Status*

Table 37: Gyroscope Control Parameter Commands Overview

Parameter	***Description***
Fast Offset Calibration	*FOC is a one-shot process that compensates errors of the sensor by setting compensation registers to the negated offset error*
Optical Image Stabilization	*Optical Image Stabilization mode of the gyroscope sensor*
Fast Startup Mode	*Fast Startup Mode, when enabled, powers down the measurement part of the Sensor frontend, while keeping the drive and digital parts operational. This allows a faster transition into normal mode while keeping power consumption significantly lower than in normal mode.*
Component Re-Trim	*Trigger and read back the status of the Component Re-Trimming of the gyroscope sensor.*
Power Mode	*The power mode of the gyroscope can be configured to operate in other power modes in order to facilitate use case appropriate performance. The gyroscope sensor has 3 power modes –* *0 - Normal Mode, The default state of the sensor* *1 - Performance Mode* *2 - Low Power, Low power mode on the account of trade-of of power consumption versus sensor noise. Low power mode may introduce aliasing artifacts*
Timer Auto Trim	*Timer Auto Trim allows to sync the Fuser2 timer oscillator PLL. When enabled, the frequency of the timer oscillator is referenced by gyroscope sample rate, benefitting from the high stability of the gyroscope MEMS oscillator. This feature is only applicable when gyroscope is enabled.*
Triggering a NVM writing	*Once NVM writing process is triggered, the value in backup registers value(offset/crt) will be written into NVM area.*

Table 38: Gyroscope Control Parameters

Gyroscope Parameter commands usage -	
Command	***Usage***
gyrosetfoc	*gyrosetfoc <x_offset> <y_offset> <z_offset>, eg: gyrosetfoc 0x0016 0x00f8 0x0080*
gyrogetfoc	*gyrogetfoc*
gyrosetois	*gyrosetois <enable/disable>, eg: gyrosetois 1*
gyrogetois	*gyrogetois*
gyrosetfs	*gyrosetfs <enable/disable>, eg: gyrosetfs 1*
gyrogetfs	*gyrogetfs*
gyrosetcrt	*gyrosetcrt*
gyrogetcrt	*gyrogetcrt*
gyrosetpwm	*gyrosetpwm <power_mode>, eg: gyrosetpwm 2*
gyrogetpwm	*gyrogetpwm*
gyrosettat	*gyrosettat <enable/disable>, eg: gyrosettat 1*
gyrogettat	*gyrogettat*
gyrotrignvm	*gyrotrignvm*
gyrogetnvm	*gyrogetnvm*

Table 39: Gyroscope Control Parameter Commands Usage

```
gyrosetfoc 0x0016 0x00f8 0x0080
Set the Gyroscope Fast Offset Calibration
        -<x_offset> : 22
        -<y_offset> : 248
        -<z_offset> : 128
gyrogetfoc
Gyroscope Fast Offset Calibration :
        -<x_offset> : 22
        -<y_offset> : 248
        -<z_offset> : 128
gyrosetois 1
Gyroscope OIS Enabled
gyrogetois
Gyroscope OIS Status : Enabled
gyrosetfs 1
Gyroscope Fast Startup Enabled
gyrogetfs
Gyroscope Fast Startup Status : Enabled
gyrosetcrt
Start Gyroscope Component ReTrim (CRT)
gyrogetcrt
Gyroscope CRT Status : Successful
gyrosetpwm 2
Set the Gyroscope Power Mode to LOW POWER
gyrogetpwm
Gyroscope Power Mode : LOW POWER
gyrosettat 1
Gyroscope Timer Auto Trim Started
gyrogettat
Gyroscope Timer Auto Trim Status : Started
gyrotrignvm
Trigger a NVM writing for gyroscope
gyrogetnvm
NVM writing status for gyroscope: Done
```

Figure 35: Gyroscope Control Parameter Commands Output

3.13.3 Magnetometer

The following commands allow the user to configure the Magnetometer sensor.

Feature Class	*Command*	*Description*
Magnetometer Sensor Control	***magsetpwm***	*Set the Magnetometer Power Mode*
	maggetpwm	*Get the Magnetometer Power Mode*

Table 40: Magnetometer Control Parameter Commands Overview

Parameter	*Description*
Power Mode	*The power mode of the magnetometer can be configured to operate in other power modes in order to facilitate use case appropriate performance. The magnetometer sensor has 2 power modes –* *0 - Normal Mode, The default state of the sensor* *2 - Low Power, Low power mode on the account of trade-of of power consumption versus sensor noise. Low power mode may introduce aliasing artifacts*

Table 41: Magnetometer Control Parameters

Magnetometer Parameter commands usage -	
Command	***Usage***

magsetpwm	*magsetpwm <power_mode>* *eg: magsetpwm 0*
maggetpwm	*maggetpwm*

Table 42: Magnetometer Control Parameter Commands Usage

```
magsetpwm 0
Set the Magnetometer Power Mode to NORMAL
maggetpwm
Magnetometer Power Mode : NORMAL
```

Figure 36: Magnetometer Control Parameter Commands Output

3.13.4 Wrist Wear Wakeup

The following commands allow the user to configure the Wrist Wear Wakeup sensor.

Feature Class	*Command*	*Description*
Wrist Wear Wakeup Sensor Control	***wwwsetcnfg***	*Set the Wrist Wear Wakeup Configuration*
	wwwgetcnfg	*Get the Wrist Wear Wakeup Configuration,*

Table 43: Wrist Wear Wakeup Control Parameter Commands Overview

Wrist Wear Wakeup Parameter commands usage -	
Command	***Usage***
wwwsetcnfg	*wwwsetcnfg <maf> <manf> <alr> <all> <apd> <apu> <mdm> <mdq>,* *eg: wwwsetcnfg 1700 1600 120 100 150 225 5 7*
wwwgetcnfg	*wwwgetcnfg*

Table 44: Wrist Wear Wakeup Control Parameter Commands Usage

```
wwwgetcnfg
min_angle_focus : 1774
min_angle_nonfocus : 1522
angle_landscape_right : 128
angle_landscape_left : 128
angle_portrait_down : 22
angle_portrait_up : 241
min_dur_moved : 2
min_dur_quite : 2

wwwsetcnfg 1700 1600 120 100 150 225 5 7
min_angle_focus : 1700
min_angle_nonfocus : 1600
angle_landscape_right : 120
angle_landscape_left : 100
angle_portrait_down : 150
angle_portrait_up : 225
min_dur_moved : 5
min_dur_quite : 7

wwwgetcnfg
min_angle_focus : 1700
min_angle_nonfocus : 1600
angle_landscape_right : 120
angle_landscape_left : 100
angle_portrait_down : 150
angle_portrait_up : 225
min_dur_moved : 5
min_dur_quite : 7
```

Figure 37: Wrist Wear Wakeup Control Parameter Commands Output

3.13.5 AnyMotion/NoMotion

The following commands allow the user to configure the AnyMotion/NoMotion sensor.

Feature Class	*Command*	*Description*

AnyMotion/NoMotion Sensor Control	***amsetcnfg***	*Set the Any Motion Configuration*
	amgetcnfg	*Get the Any Motion Configuration*
	nmsetcnfg	*Set the No Motion Configuration*
	nmgetcnfg	*Get the No Motion Configuration*

Table 45: AnyMotion/NoMotion Control Parameter Commands Overview

The AnyMotion and NoMotion sensors share same set of parameters. These include:

Configuration	***Description***
duration	*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*
axis selection	*Minimum threshold for flick peak on z-axis.* *Scaling: 0.4883, Range: 0x1F4 to 0x5DC*
threshold	*Slope threshold value for any-motion/no-motion detection.* *Range is 0 to 1g.*

Table 46: AnyMotion/NoMotion Control Parameters

AnyMotion/NoMotion Parameter commands usage -	
Command	***Usage***
amsetcnfg	*amsetcnfg <duration> <axis> <threshold>* *eg: amsetcnfg 5 7 170*
amgetcnfg	*amgetcnfg*
nmsetcnfg	*nmsetcnfg <duration> <axis> <threshold>* *eg: nmsetcnfg 1 7 144*
nmgetcnfg	*nmgetcnfg*

Table 47: AnyMotion/NoMotion Control Parameters Usage

```
amgetcnfg
Any Motion Configuration:
         -duration : 16
         -axis_sel : 4
         -threshold : 44

amsetcnfg 5 7 170
Set the Any Motion Configuration
         -duration : 5
         -axis_sel : 7
         -threshold : 170

amgetcnfg
Any Motion Configuration:
         -duration : 5
         -axis_sel : 7
         -threshold : 170

nmgetcnfg
No Motion Configuration:
         -duration : 1
         -axis_sel : 2
         -threshold : 120

nmsetcnfg 1 7 144
Set the No Motion Configuration
         -duration : 1
         -axis_sel : 7
         -threshold : 144
```

Figure 38: AnyMotion/NoMotion Control Parameter Commands Output

3.13.6 Wrist Gesture Detector

The following commands allow the user to configure the Wrist Gesture Detector sensor.

Feature Class	***Command***	***Description***

Wrist Gesture Detector Sensor Control	***wgdsetcnfg***	*Set the Wrist Gesture Detector Configuration*
	wgdgetcnfg	*Get the Wrist Gesture Detector Configuration*

Table 48: Wrist Gesture Detector Control Parameter Commands Overview

Wrist Gesture Detector Parameter commands usage -	
Command	***Usage***
wgdsetcnfg	*wgdsetcnfg <mfpy_th> <mfpz_th> <gx_pos> <gx_neg> <gy_neg> <gz_neg> <fpdc> <lmfc> <mdjp> <dp0>,* *eg: wgdsetcnfg 0x400 0x200 0x600 0xf000 0xa000 0x900 0x4000 0x6000 0xb 1*
wgdgetcnfg	*wgdgetcnfg*

Table 49: Wrist Gesture Detector Control Parameter Commands Usage

```
wgdgetcnfg
min_flick_peak_y_threshold : 0x0640
min_flick_peak_z_threshold : 0x02bc
gravity_bounds_x_pos : 0x0784
gravity_bounds_x_neg : 0xfc00
gravity_bounds_y_neg : 0xfc3f
gravity_bounds_z_neg : 0xf912
flick_peak_decay_coeff : 0x7851
lp_mean_filter_coeff : 0x7d71
max_duration_jiggle_peaks : 0x0010
device_position : 0x00

wgdsetcnfg 0x400 0x200 0x600 0xf000 0xa000 0x900 0x4000 0x6000 0xb 1
Wrist Gesture Detector Parameter set successfully

wgdgetcnfg
min_flick_peak_y_threshold : 0x0400
min_flick_peak_z_threshold : 0x0200
gravity_bounds_x_pos : 0x0600
gravity_bounds_x_neg : 0xf000
gravity_bounds_y_neg : 0xa000
gravity_bounds_z_neg : 0x0900
flick_peak_decay_coeff : 0x4000
lp_mean_filter_coeff : 0x6000
max_duration_jiggle_peaks : 0x000b
device_position : 0x01
```

Figure 39: Wrist Gesture Detector Control Parameter Commands Output

3.13.7 Barometer Pressure Type 1/Type 2

The following commands allow the user to configure the Barometer Pressure type 1/type 2 sensor.

Feature Class	***Command***	***Description***
Barometer Pressure Sensor Control	***baro1setcnfg***	*Set the Barometer Pressure Type 1 Configuration*
	baro1getcnfg	*Get the Barometer Pressure Type 1 Configuration*
	baro2setcnfg	*Set the Barometer Pressure Type 2 Configuration*
	baro2getcnfg	*Get the Barometer Pressure Type 2 Configuration*

Table 50: Barometer Pressure Control Parameter Commands Overview

Barometer Pressure Parameter commands usage -	
Command	***Usage***
baro1setcnfg	*baro1setcnfg <osr_p> <osr_t> <iir_filter>* *eg: baro1setcnfg 1 1 1*
baro1getcnfg	*baro1getcnfg*
baro2setcnfg	*baro2setcnfg <osr_p> <osr_t> <iir_filter_p> <iir_filter_t> <dsp_config>* *eg: baro2setcnfg 1 1 1 1 1*
baro2getcnfg	*baro2getcnfg*

Table 51: Barometer Pressure Control Parameter Commands Usage

```
baro1setcnfg 1 1 1
Set the Barometer pressure type 1 Configuration
        -osr_p : 1
        -osr_t : 1
        -iir_filter : 1
baro1getcnfg
Barometer pressure type 1 Configuration:
        -osr_p : 1
        -osr_t : 1
        -iir_filter : 1
baro2setcnfg 1 1 1 1 1
Set the Barometer pressure type 2 Configuration
        -osr_p : 1
        -osr_t : 1
        -iir_filter_p : 1
        -iir_filter_t : 1
        -dsp_config : 1
baro2getcnfg
Barometer pressure type 2 Configuration:
        -osr_p : 1
        -osr_t : 1
        -iir_filter_p : 1
        -iir_filter_t : 1
        -dsp_config : 1
```

Figure 40: Barometer Pressure Control Parameter Commands Output

3.13.8 Step Counter

The following commands allow the user to configure the Step Counter sensor.

Feature Class	*Command*	*Description*
Step Counter Sensor Control	***scsetcnfg***	*Set the Step Counter Configuration*
	scgetcnfg	*Get the Step Counter Configuration*

Table 52: Step Counter Control Parameter Commands Overview

Step Counter Parameter commands usage -	
Command	***Usage***
scsetcnfg	*scsetcnfg <emdu> <ecu> <emdd> <ecd> <sbs> <mvd> <msd> <fcb2> <fcb1> <fcb0> <fca2> <fca1> <fce> <pdmw> <pdmr> <sdm> <sdw> <hse> <adf> <adt> <sci> <sdpe> <sdt> <empp> <mt> <sc26> <sc27>* *eg: scsetcnfg 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1*
scgetcnfg	*scgetcnfg*

Table 53: Step Counter Control Parameter Commands Usage

```
scsetcnfg 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Set the Step Counter Configuration
        -env_min_dist_up: 1
        -env_coef_up: 1
        -env_min_dist_down: 1
        -env_coef_down: 1
        -step_buffer_size: 1
        -mean_val_decay: 1
        -mean_step_dur: 1
        -filter_coeff_b2: 1
        -filter_coeff_b1: 1
        -filter_coeff_b0: 1
        -filter_coeff_a2: 1
        -filter_coeff_a1: 1
        -filter_cascade_enabled: 1
        -peak_duration_min_walking: 1
        -peak_duration_min_running: 1
        -step_duration_max: 1
        -step_duration_window: 1
        -half_step_enabled: 1
        -activity_detection_factor: 1
        -activity_detection_thres: 1
        -step_counter_increment: 1
        -step_duration_pp_enabled: 1
        -step_dur_thres: 1
        -en_mcr_pp: 1
        -mcr_thres: 1
        -sc_26: 1
        -sc_27: 1
```

Figure 41: Step Counter Control Parameter Commands Output

3.13.9 Physical Range

The following commands allow the user to set the physical range of each sensor.

Feature Class	*Command*	*Description*
Physical Range Configuration	***phyrangeconf***	*Set the range of physical sensor*

Table 54: Physical Range Configuration Commands Overview

Physical Range Configuration commands usage -	
Command	***Usage***
phyrangeconf	*phyrangeconf <sensor_id> <range_value>* *eg: phyrangeconf 1 0x10*

Table 55: Physical Range Configuration Commands Usage

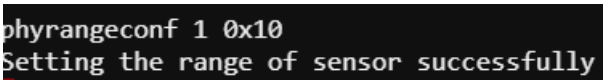


Figure 42: Physical Range Configuration Commands Output

3.13.10 Fast Offset Compensation

The following commands allow the user to enable the fast offset compensation for the sensor.

Feature Class	*Command*	*Description*

Fast Offset Compensation	***foc***	*Enable the fast offset compensation for the sensor*

Table 56: Fast Offset Compensation Commands Overview

Fast Offset Compensation commands usage -	
Command	***Usage***
foc	*foc <sensor id ('1' for accelerometer, '3' for gyroscope)> eg: foc 1*

Table 57: Fast Offset Compensation Commands Usage

```
foc 1
Keep the sensor stable for accel foc
FOC Success
```

Figure 43: Fast Offset Compensation Commands Output

3.14 Activity Recognition Parameters Commands

The following commands allow the host to set or get Activity Recognition parameters.

Feature Class	Command	Description
Activity Recognition Parameters	***sethearactvcnfg***	*Set the Hearable Activity Configuration*
	gethearactvcnfg	*Get the Hearable Activity Configuration*
	setwearactvcnfg	*Set the Wearable Activity Configuration*
	getwearactvcnfg	*Get the Wearable Activity Configuration*

Table 58: Activity Recognition Parameters Commands Overview

Activity Recognition Parameters commands usage -	
Command	***Usage***
sethearactvcnfg	*sethearactvcnfg <ss> <ppe> <mingt> <maxgt> <obs> <msmc> eg. sethearactvcnfg 1 1 1 1 1 1*
gethearactvcnfg	*gethearactvcnfg*
setwearactvcnfg	*setwearactvcnfg <ppe> <mingt> <maxgt> <obs> <msmc> eg. setwearactvcnfg 1 1 1 1 1*
getwearactvcnfg	*getwearactvcnfg*

Table 59: Activity Recognition Parameters Commands Usage

```
sethearactvcnfg 1 1 1 1 1 1
Set hearable activity configuration
    - seg_size: 1
    - post_process_en: 1
    - min_gdi_thre: 1
    - max_gdi_thre: 1
    - out_buff_size: 1
    - min_seg_moder_confg: 1
gethearactvcnfg
Hearable activity configuration:
    - seg_size: 1
    - post_process_en: 1
    - min_gdi_thre: 1
    - max_gdi_thre: 1
    - out_buff_size: 1
    - min_seg_moder_confg: 1
setwearactvcnfg 1 1 1 1 1
Set wearable activity configuration
    - post_process_en: 1
    - min_gdi_thre: 1
    - max_gdi_thre: 1
    - out_buff_size: 1
    - min_seg_moder_confg: 1
getwearactvcnfg
Wearable activity configuration:
    - post_process_en: 1
    - min_gdi_thre: 1
    - max_gdi_thre: 1
    - out_buff_size: 1
    - min_seg_moder_confg: 1
```

Figure 44: Activity Recognition Parameters Commands Output

3.15 Data Injection Commands

The following commands allow the host to set or inject data.

Feature Class	Command	Description
Data Injection	***dmode***	*Set the Data Injection mode*
	dinject	*Compute virtual sensor output from raw IMU data*

Table 60: Data Injection Commands Overview

Data Injection commands usage -	
Command	***Usage***
dmode	*dmode <mode>* *eg. dmode s*
dinject	*dinject <input_file.txt>* *eg. dinject field_log.txt*

Table 61: Data Injection Commands Usage

```
Sensor ID: 129, sample rate: 4.000000 Hz, latency: 0 ms


Opened Log File baro_chg.bin.csv_Injection.txt

File Size : 29624
[D][META EVENT]; T: 0.347015625; Flush complete for sensor id 129
[D][META EVENT]; T: 0.347875000; Power mode changed for sensor id 129
[D][META EVENT]; T: 0.347875000; Sample rate changed for sensor id 129
[D]SID: 129; T: 0.250000000; 90297.000000
[D]SID: 129; T: 0.500000000; 90299.000000
[D]SID: 129; T: 0.750000000; 90300.000000
[D]SID: 129; T: 1.000000000; 90299.000000
[D]SID: 129; T: 1.250000000; 90299.000000
[D]SID: 129; T: 1.500000000; 90299.000000
[D]SID: 129; T: 1.750000000; 90299.000000
[D]SID: 129; T: 2.000000000; 90301.000000
[D]SID: 129; T: 2.250000000; 90300.000000
[D]SID: 129; T: 2.500000000; 90300.000000
[D]SID: 129; T: 2.750000000; 90299.000000
[D]SID: 129; T: 3.000000000; 90299.000000
[D]SID: 129; T: 3.250000000; 90295.000000
[D]SID: 129; T: 3.500000000; 90298.000000
[D]SID: 129; T: 3.750000000; 90296.000000
[D]SID: 129; T: 4.000000000; 90297.000000
[D]SID: 129; T: 4.250000000; 90298.000000
[D]SID: 129; T: 4.500000000; 90298.000000
[D]SID: 129; T: 4.750000000; 90299.000000
[D]SID: 129; T: 5.000000000; 90298.000000
[D]SID: 129; T: 5.250000000; 90297.000000
[D]SID: 129; T: 5.500000000; 90299.000000
[D]SID: 129; T: 5.750000000; 90299.000000
[D]SID: 129; T: 6.000000000; 90301.000000
[D]SID: 129; T: 6.250000000; 90300.000000
[D]SID: 129; T: 6.500000000; 90301.000000
[D]SID: 129; T: 6.750000000; 90299.000000
[D]SID: 129; T: 7.000000000; 90300.000000
[D]SID: 129; T: 7.250000000; 90301.000000
[D]SID: 129; T: 7.500000000; 90299.000000
[D]SID: 129; T: 7.750000000; 90301.000000
[D]SID: 129; T: 8.000000000; 90301.000000
[D]SID: 129; T: 8.250000000; 90301.000000
[D]SID: 129; T: 8.500000000; 90300.000000
[D]SID: 129; T: 8.750000000; 90298.000000
[D]SID: 129; T: 9.000000000; 90298.000000
[D]SID: 129; T: 9.250000000; 90296.000000
[D]SID: 129; T: 9.500000000; 90297.000000
[D]SID: 129; T: 9.750000000; 90299.000000
[D]SID: 129; T: 10.000000000; 90298.000000
[D]SID: 129; T: 10.250000000; 90299.000000
[D]SID: 129; T: 10.500000000; 90298.000000
[D]SID: 129; T: 10.750000000; 90298.000000
[D]SID: 129; T: 11.000000000; 90298.000000
[D]SID: 129; T: 11.250000000; 90297.000000
[D]SID: 129; T: 11.500000000; 90298.000000
[D]SID: 129; T: 11.750000000; 90299.000000
[D]SID: 129; T: 12.000000000; 90298.000000
[D]SID: 129; T: 12.250000000; 90298.000000
[D]SID: 129; T: 12.500000000; 90299.000000
[D]SID: 129; T: 12.750000000; 90299.000000
```

Figure 45: Data Injection Commands Output

3.16 Diagnostics Commands

The following commands allow the host to get diagnostics information.

Feature Class	Command	Description
Diagnostics	***-m OR postm***	*Get Post Mortem Data and log to a file*

Table 62: Diagnostics Command Overview

Diagnostics commands usage -

Command	*Usage*
postm	*postm <pm_log_filename.bin>* *eg. postm pm_log_filename.bin*

Table 63: Diagnostics Command Usage

```
Error Reg Value : 44
POST MORTEM STATUS :
valid           :       0x00000001
flags           :       0x00000007

CONTEXT :
reg_1           :       0x00000000
reg_2           :       0x00000000
reg_3           :       0x00a15238
reg_4           :       0x00000024
reg_5           :       0x00a04b90
reg_6           :       0x00f0000c
reg_7           :       0x00000000
reg_8           :       0x00200000
reg_9           :       0x0000003f
reg_10          :       0x10101010
reg_11          :       0x00002300
reg_12          :       0x00a15638
reg_13          :       0x00000000
reg_14          :       0x00a042b4
reg_15          :       0x00a04e04
reg_16          :       0x00000001
reg_17          :       0x00a111c0
reg_18          :       0x00000001
reg_19          :       0x00000008
reg_20          :       0x00a042c0
reg_21          :       0x21212121
reg_22          :       0x22222222
reg_23          :       0x23232323
reg_24          :       0x24242424
reg_25          :       0x25252525
reg_26          :       0x00a05c1c
gp    [reg_27]  :       0x00a1117c
fp    [reg_28]  :       0x00a11158
sp    [reg_29]  :       0x8000481e
ilink [reg_30]  :       0x30303030
reg_31          :       0x0011ad74
blink [reg_32]  :       0x001029f2

SYSTEM STATUS :
eret            :       0x0010e562
erbta           :       0x0010e4bc
erstatus        :       0x8000461e
ecr             :       0x00020000
efa             :       0x0010e562
icause          :       0x00000000
mpu_ecr         :       0x00000000

DIAGNOSTIC :
diagnostic      :       0x00000002
debug state     :       0x000000b1
debug val       :       0x00000000
error val       :       0x00000000
interrupt       :       0x00000000
err report      :       0x00000044

STACK INFO :
stack start     :       0x00a05c20
stack size      :       0x00001000
```

Figure 46: Diagnostics Command Output

3.17 Utility Commands

The utility commands are application specific commands and allow the users to control and manage the various application specific configurations to have an enhanced user interaction.

Feature Class	Utility	Command	Description
Utility	**Debug**	**-v OR verb**	Set the verbose level.
	Status	**echo**	Enable/Disable echo
		heart	Enable/disable Heartbeat Message
	File	**mklog**	Create a log file
		rm	Remove a log file
		ls	List the files in the external Flash
		wrfile	Write to a log file
		rdfile	Read from a log file
		slabel	Set a string label in the log file
	Stream	**strbuf**	Set the streaming buffer size
	UI	**cls**	Clear Screen

Table 64: Utility Commands Overview

Note: *Except the 'verb' command, all other utility commands are exclusive to MCU mode.*

3.17.1 Debug Utility

The verbose command '-v' or 'verb' sets the verbose level. The verbose level refers to the level of response that the user expects from the application regarding its state of execution. The verbose level is classified in the following table.

verbose	*scope*
0	*Give only error notifications.*
1	*Give notifications regarding errors as well as warnings.*
2	*Give notifications regarding the complete state of the system in terms of errors, warnings, and information about the current state of execution.*

Table 65: Verbose Levels

Debug Utility -	
Command	**Usage**
verb	*verb <verbose_level>, eg: verb 1*

Table 66: Debug Utility Command Usage

```
version
HW info:: Board: 5, HW ID: 11, Shuttle ID: 179, SW ID: 10
SW Version: 0.6.0
Build date: May 20 2025


verb 2
[I]Executing verb 2
Setting verbose to 2


version
[I]Executing version
HW info:: Board: 5, HW ID: 11, Shuttle ID: 179, SW ID: 10
SW Version: 0.6.0
Build date: May 20 2025
```

Figure 47: Debug Utility Commands Output

By default, the verbose level is set to 0 to limit the number of notifications to need-to-know. The verbose level can be configured accordingly based on debugging and application requirements.

3.17.2 Status Utility

The echo command echoes back the input given to the application. The echo command can be used to enable/disable the echo feature.

The ‘heart’ command is used to indicate the system notifications to the user by blinking the LED every time a notification is sent. The heart command can be used to enable/disable the heartbeat feature.

Status Utility -	
Command	**Usage**
echo	*echo <state>, eg: echo on*
heart	*heart <state>, eg: heart off*

Table 67: Status Utility Commands Usage

```
echo off
Setting echo to off


Setting echo to on


heart on
Setting Heartbeat message to on


[H]34807
[H]34857
[H]37307
[H]37357
[H]39807
[H]39857

[H]42307
[H]42357
heart off
Setting Heartbeat message to off
```

Figure 48: Status Utility Commands Output

3.17.3 File Utility

The File utility commands allow the user to handle the various File and Log operations such as create/delete file, read/write file, add annotations etc. The file utility commands are different from the logging commands described in **Log Generation Commands**, in the sense that the log commands are more oriented towards logging the sensor data, whereas the File utility commands are more oriented towards facilitating the user with file management of the external Flash on the application board.

The following table describes those commands:

File Utility -	
Action	**Commands**
Generate the log file	*mklog <file_name.file_extension>, eg: mklog abc.txt*
List the files in the external Flash of the application board	*ls*
Write to the log file	*wrfile <file_name.file_extension> <length_in_bytes>, eg: wrfile abc.txt 150* ***Note 1**: In an event, a file is not present with the passed filename, a new file with same filename is generated.* ***Note 2**: There is a input timeout duration of 10s. After executing the ‘wrfile’ command, if no input is provided within the 10s, the application will assert the ‘wrfile’ callback and return ‘Write Timed Out’ error.* ***Note 3**: If the same command executed second time. The new input data will be appended to the previous content of the file.* ***Note 4**: While uploading the data from a PC using any application, ensure that <size_of_file> is passed as the input to length argument, and not the <size_of_file_on_disk>*
Read the log file	*rdfile <file_name.file_extension, eg: rdfile abc.txt*
Delete the File	*rm <file_name.file_extension>, eg: rm abc.txt*
Annotate the log file	*slabel <label string>, eg: slabel Activity_N* ***Note**: slabel command can be used in association with Log generation commands, when acquiring the data in logging mode to annotate the events.*

Table 68: File Utility Commands Usage

3.17.3.1 File Handling

```
ls
                              README.TXT | 731 B


mklog ab.txt
File ab.txt was created


ls
                              README.TXT | 731 B
                                  ab.txt | 0 B


wrfile abc.txt 10

Waiting for 10 bytes of data
File Transferred : 100.00%
Write Completed


rdfile abc.txt
Read Initiated
abcdef0123
Read Completed

rm ab.txt
File ab.txt was removed


ls
                              README.TXT | 731 B
                                 abc.txt | 10 B
```

Figure 49: File Utility Commands Output

3.17.3.2 Annotation

Annotation is very useful from a data collection perspective. It is used to annotate the various events covered during the scope of data collection.

File Annotation	
Action	**Usage**
Attach a log file for the logging	*attlog <file_name.file_extension>, eg: attlog abc.bin*
Enable the sensor acquisition in logging mode	*logse <id>:<odr>, eg: logse 4:100*
Annotate the log file	*slabel <label string>, eg: slabel Activity_N*
Disable the sensor acquisition	*logse <id>:0, eg: logse 4:0*
Detach log file from logging	*detlog <file_name.file_extension>, eg: detlog abc.bin*

Table 69: File Annotation using 'slabel'

3.17.4 Stream Utility

The 'strbuf' command allows the user to configure the streaming packet length. This is a performance improvement feature. At higher ODRs, due to limited BLE packet size, there is a sample loss due to high frequency of notifications. In such a scenario, to ensure that there is no loss of data, the samples are maintained in a local FIFO and sent packet by packet. The 'strbuf' command is used to configure the length of this local FIFO.

Stream Utility -	
Command	**Usage**
strbuf	*strbuf <buffer_size>, eg: strbuf 100* ***Note:*** *The maximum buffer size is 240*

Table 70: 'strbuf' Command Usage

The utility of *'strbuf'* command is difficult to showcase in the scope of this document. However, the following steps can be followed to understand the utility of *'strbuf'* command.

'strbuf' Utility	
Action	**Usage**
Enable Sensor Acquisition	*actse <sensor_id>:<odr>, eg: actse 1:200*
Disable Sensor Acquistition	*actse <sensor_id>:0, eg: actse 1:0*
Performance Analysis	*Copy the data and compute time difference*
Configure the Stream Buffer Length	*strbuf <buffer_size>, eg: strbuf 100*
Enable Sensor Acquisition	*actse <sensor_id>:<odr>, eg: actse 1:200*
Disable Sensor Acquistition	*actse <sensor_id>:0, eg: actse 1:0*
Performance Analysis	*Copy the data and compute time difference*

Table 71: 'strbuf' Utility

3.17.5 UI Utility

The 'cls' is a UI command and used to clear the screen of the UI.

Debug Utility	
Command	**Usage**
cls	*cls*

Table 72: 'cls' Command Usage

4 BHy2CLI Limitations

4.1 HW Limitations

- There is a limitation on the file name length when transferring files to APP30 External Flash in MTP mode. It allows a maximum of 39 characters for file names.

4.2 Platform Limitations

- Copying of the log files using Windows Explorer can lead to data corruption.
- For Head Orientation sensors, the ODR change is not reflected for subsequent sensor activation with different ODRs.
- Sometimes, firmware files with large file name lengths can't be copied to external Flash. To address this, shorten the file name.
- Due to filesystem constraints, it does not support the simultaneous writing of multiple files to the external memory of application boards.
- In MCU mode, a file will be appended instead of removed and/or created as a new one for all commands (except 'rm' and 'getbsxparam') with handling file.
- HEX streaming only supports ODR with a frequency not bigger than 400Hz.
- Due to algorithm, sometimes, BSX magnetometer parameters are not restored properly when the sensors are enabled with logse/actse command

5 Legal Disclaimer

5.1 Engineering Samples

Engineering Samples are marked with an asterisk (*) Or (e). Samples may vary from the valid technical specifications of the product series contained in the user guide. 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.

5.2 Product Use

Bosch Sensortec products are developed for the consumer goods industry. They are not designed or approved for use in military applications, life-support appliances, safety-critical automotive applications and devices or systems where malfunctions of these products can reasonably be expected to result in personal injury. They may only be used within the parameters of this user guide.

The resale and/or use of products are at the Purchaser’s own risk and the Purchaser’s own responsibility.

The Purchaser shall indemnify Bosch Sensortec from all third party claims arising from any product use not covered by the parameters of this user guide 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 inform Bosch Sensortec without delay of any security relevant incidents.

5.3 Application examples and hints

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

6 Document history and modifications

Rev. No	Chapter	Description of modification/changes	Date
1.0		First Draft	Nov 2023
1.1		Addressed the Review Comments	Jan 2024
2.0		Added new commands for Parameters, Info Added new chapter for limitations	Nov 2024
2.1		Updated compatibility Corrected some typos in commands	Apr 2025
2.2		Updated compatibility Changed name from BHyCLI to BHy2CLI Corrected some typos in commands Removed commands related to Flash Updated limitation related to BSX magnetometer	May 2025

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

www.bosch-sensortec.com

Document number: [TBD]
Revision_2.2_052025