BOSCH
Invented for life

Bosch Sensortec

Software Document
MCU Driver Porting Guide

BHy1 - MCU Driver Porting Guide

Document revision	1.5
Document release date	03 January 2017
Document number	BST-MHS-SD001-04

Technical reference code(s)	0 273 141 230 0 273 141 231	0 273 141 309 0 273 141 310

Notes

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

Index of Contents

1. Prerequisites

Before the porting effort, the customer is expected to have a working sample code for i2c communication. In order to request and streamline support from a Bosch Sensortec FAE, a regional office, distributor or sales representative the customer is expected to provide proof of unsuccessful i2c transactions (i.e. a logic analyzer data-log).

2. Description of the package

The driver is separated in two parts: the API and the actual driver. Some files should not be modified; others should only be modified by power users and others need to be modified by all users.

2.1 API files

2.1.1 BHIfw.h

This file contains the definition of the RAM patch, also known as firmware file or fw file. The default RAM patch supports full 9DoF data fusion using the BMM150 magnetometer attached to the smart-hub. The author recommends to generate and to convert the available RAM patches automatically with a free tool called “fw2header”.
Link: https://github.com/BoschSensortec/BHy1-fw-convert-tool
This file **has to be modified for every RAM patch.**

2.1.2 bhy.h

This file contains the declarations for the BHy API. Refer to these for missing features in the driver.
This file **should not be modified.**

2.1.3 bhy.c

This file contains the definitions of the BHy API.
This file **should not be modified.**

2.1.4 Bhy_support.h

This file contains the declarations of the platform-specific functions that the driver needs.
This file **should not be modified.**

2.1.5 Bhy_support.c

This file contains the definitions of the platform-specific functions that the driver needs.
This file **has to be modified for every platform.**

2.2 Driver files

2.2.1 bhy_uc_driver_config.h

This file allows you to enable/disable the debug and the callback features.
This file **should be modified for every application**

2.2.2 bhy_uc_driver_constants.h

This file contains the sensor ID definitions.
This file **may be modified if creating a new virtual sensor ID.***

2.2.3 bhy_uc_driver_types.h

This file contains all the types definition that are used in the driver. The user should look through this file and use these types.
This file **may be modified if creating a new virtual sensor ID.***

2.2.4 bhy_uc_driver.h

This file contains the driver functions declarations that are called from the main application.
This file **should not be modified.**

2.2.5 bhy_uc_driver.c

This file contains the main code of the driver.
This file **may be modified if creating a new virtual sensor ID.***

Note:
Adding a new virtual sensor ID may impact your ability to merge new driver code into your modified version. It is only recommended for power users.

3. Porting instructions

3.1 Moving to a new platform

When moving to a new platform. The user needs to modify the “bhy_support.c” file and do the following:

- Remove the following lines that are Atmel-specific :

```
#include "twi.h"

/*! instantiates an I2C packet software instance structure which retains
* I2C slave address, data buffer and data length and is used to read/write
* data on the I2C bus */

twi_packet_t bhy_i2c_packet;
```

- Edit the generic bhy_i2c_write function so that it makes an i2c write to the specified address and register on your platform
- Edit the generic bhy_i2c_read function so that it makes an i2c read to the specified address and register on your platform
- In your main code, make sure you implement a function called “mdelay” that creates a delay of roughly 1ms

When compiling the code it will automatically try to detect the fixed-width types for your platform. If this fails, you will get the following compiler warning:

```
#warning the data types defined above which not supported define the data types manually
```

If that happens, then you need to modify the bhy.h file to define the following fixed-width types:
s8, s16, s32, u8, u16, u32

4. Using the driver

4.1 Configuration

The driver has 2 configuration parameters. The user needs to set them in bhy_uc_driver_config.h

4.1.1 BHY_DEBUG

This setting allows the driver to print debug messages. The BHy firmware releases do not output debug messages, so this parameter should only be set to ‘1’ by users in development of their own BHy firmware. Otherwise it should be set to ‘0’

4.1.2 BHY_CALLBACK_MODE

This setting allows the callback feature of the driver. A callback is a software interrupt. If enabled, the installed software callbacks will automatically be called when the FIFO packets are decoded in the parsing process. The callbacks are called in the context of the FIFO parsing function. The only drawback is that is uses a little bit of RAM (~350 bytes on a 32-bit system).

4.1.3 BHY_APPLICATION_BOARD

This setting allows the driver to work successfully on the BST application board limiting the I²C transaction sizes to 51 bytes.

4.1.4 RAM patch modifications

A compiled RAM patch comes in the *.fw format, which is a pure binary file. The user has to create a C constant definition with the binary data included in the firmware file. This data has to be put in BHIfw.h.

4.2 Usage

The driver helps you in 3 phases of the application code: initialization, configuration and data readout.

4.2.1 Initialization

To initialize the driver, the user should make sure that the MCU has configured i2c module and then call the bhy_driver_init function. This will initialize the driver, download the RAM patch to the bhy from the ROM of the MCU, verify the checksum, launch the Fuser core and report an error code. If the status return value is BHY_SUCCESS, then the initialization was succesful.

4.2.3 Configuration

To configure the driver for you application, the user needs to enable the required sensors via the bhy_enable_virtual_sensor function. Then he may want to install either sensor callbacks, timestamp callbacks or meta_event callbacks with the functions bhy_install_sensor_callback, bhy_install_timestamp_callback and bhy_install_meta_event_callback.

Note:
Code that is released to consumers should always monitor the BHY_META_EVENT_TYPE_ERROR and BHY_META_EVENT_TYPE_SENSOR_ERROR meta events for increased robustness.

4.2.3 Data readout

The data readout is done in two phase: reading the FIFO data from the bhy into the MCU memory via the bhy_read_fifo function, and then decoding binary data into useful data structures via the function bhy_parse_next_fifo_packet. The user should get familiar with the decoding algorithm provided in the sample code before modifying it.

5. Legal disclaimer

5.1 Engineering samples

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

5.2 Product use

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

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

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

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

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 modification

Rev. No	Chapter	Description of modification/changes	Date
1.5	Cover sheet + Header Cover sheet	Introduced Document Type Added 2 new tech. ref. codes: - 0.273.141.309 (BHI160B) - 0.273.141.310 (BHA250B)	Jan. 03, 2017
1.4	2.1.1 all	Updated tool “bin2h” to “fw2header” incl. URL Updated format and index of content	Aug. 08, 2016
1.3	2.1.1	Added URL for tool “bin2h”	Jun. 27, 2016
1.2	4.1.3	Added BST_APPLICATION_BOARD setting	Mar. 21, 2015
1.1		Added prerequisites section	Nov. 24, 2015
1.0		Document creation	Aug. 20, 2015

Bosch Sensortec GmbH
Gerhard-Kindler-Strasse 9
72770 Reutlingen / Germany
Contact@bosch-sensortec.com
www.bosch-sensortec.com

Document number: BST-MHS-SD001-04