BHy1 Software Document

MCU Driver Porting Guide

BHy1 - MCU Driver Porting Guide

Document revision	1.7
Document release date	05 Feb 2018
Document number	BST-BHy1-SD001-01
Technical reference code(s)	0 273 141 230 0 273 141 309 0 273 141 231 0 273 141 310
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.

Table of contents

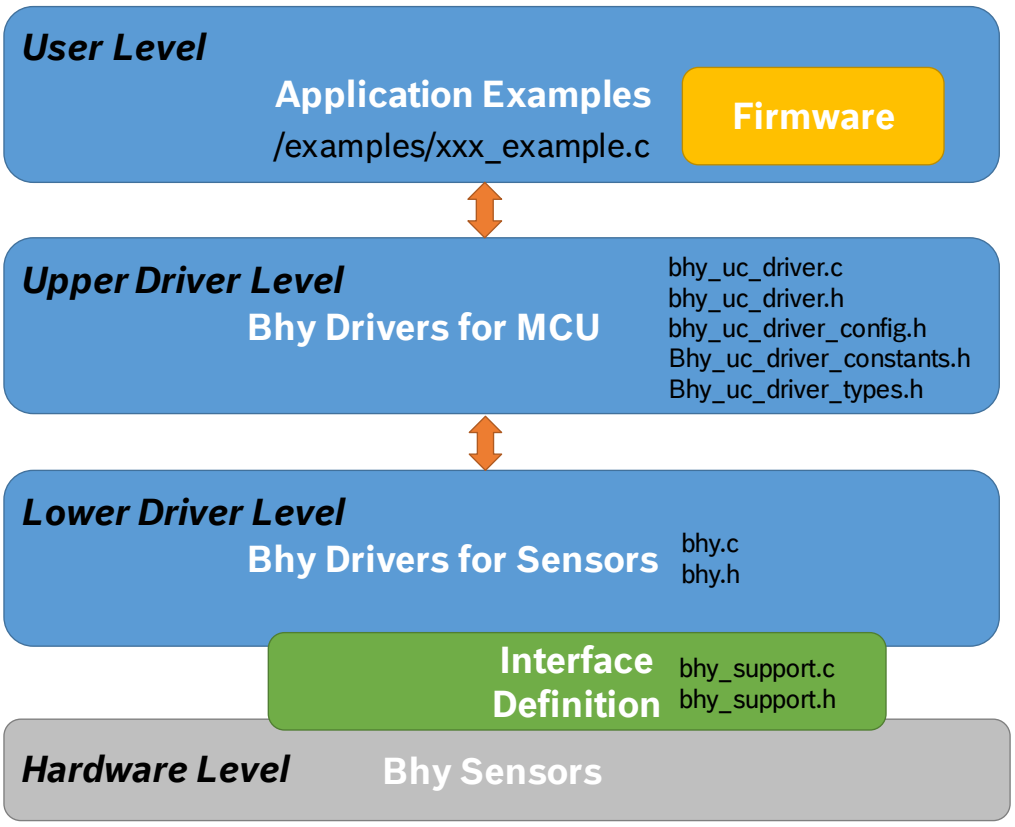
1. Introduction

This guide is intended for users who want to interface with BHY1 sensors using BHY1 sensor driver.

This Guide describes how to port the drivers to your MCU platform. The reference code is available at https://github.com/BoschSensortec/BHy1_driver_and_MCU_solution and it can also be requested from a Bosch Sensortec FAE, regional officer, distributor or sales representative.

The current MCU reference code version is 1.1.0.0.

The chart below illustrates the hierarchy of the package necessary for driver porting. The firmware is also mentioned as ram patch in this AN.



2. Description of the package

The package contains the driver files, examples and firmware as described in the following tables.

- **Driver files:**

Item	Description	Remarks
bhy_uc_driver.c	bhy combination type functions	Modifying this file is not recommended.
bhy.c	bhy lower-level API	Modifying this file is not recommended.
bhy_support.c	Platform-specific functions	It is needed to implement the functions *sensor_i2c_read* and *sensor_i2c_write* based on your own platform.
bhy_uc_driver.h	bhy combination type functions	
bhy.h	bhy lower-level API	
bhy_support.h	Platform-specific functions	
Bhy_uc_driver_config.h	Enable/disable the debug and callback features using MACRO control	This file can be modified if debug is needed.
Bhy_uc_driver_constants.h	Sensor ID definitions	This file can be modified if a new virtual sensor has been added.*
Bhy_uc_driver_types.h	Definitions of data structure types	This file can be modified if a new virtual sensor has been added.*

*Note:
Adding a new virtual sensor ID may affect the ability to merge new driver code into the modified version. It is recommended contacting your local distributor or FAE for support.

- **Examples:**

You can refer to these examples while implementing your own application.

Item	Description
accelerometer_remapping_example.c	Remapping Acc sensor application example
activity_recognition_example.c	AR application example
bmp280_example.c	Pressure sensor example
calib_profile_example.c	Calibration example
custom_sensor_example.c	Customized sensor example case
fifo_watermark_example.c	FIFO watermark setting example
gesture_recognition_example.c	Example for one of the motion sensors
rotation_vector_example.c	Rotation vector example
selftest_example.c	Selftest example

- **Firmware:**

Different firmwares should be used depending on different user requirements.

Item	Description
Bosch_PCB_7183_di03_BMI160-7183_di03.2.1.11696_170103.h	Default BHI160B fw head file Constant array converted from firmware "*.fw" file by a free tool "fw2header". The tool is available at: *https://github.com/BoschSensortec/BHy1-fw-convert-tool*.
Bosch_PCB_7183_di03_BMA2x2_Cus-7183_di03.2.1.11703.h	Default BHA250B stand alone fw header file
Bosch_PCB_7183_di03_BMI160_BMM150-7183_di03.2.1.11696_170103.h	Default BHI160B + BMM150 fw header file
Bosch_PCB_7183_di01_BMI160_BMP280-7183_di01.2.1.10836.h	Default BHI160A + BMP280 fw header file
Bosch_PCB_7183_di01_BMI160-7183_di01.2.1.10836_170103.h	Default BHI160A fw header file

NOTE For other firmwares, please download from our website and then convert it into .h file, or contact local distributors or FAE for support.

3. Porting instructions

The MCU reference code was based on bosch app2.0 board which uses Atmel MCU, user can follow below steps for a new platform.

3.1 Modifying for a new platform

To port the driver to a new platform, it is needed to modify the *bhy_support.c* file and do as follows:

- Remove the following lines that are Atmel-specific:
  #include "FreeRTOS.h"
  #include "task.h"

  *extern int8_t sensor_i2c_write(uint8_t addr, uint8_t reg, uint8_t *p_buf, uint8_t size);*
  *extern int8_t sensor_i2c_read(uint8_t addr, uint8_t reg, uint8_t *p_buf, uint8_t size);*
  *extern void trace_log(const char *fmt, ...);*

- Re-implement the low-level driver based on the new platform. Please align the input parameters and return format of below functions with above function declaration.
  - *sensor_i2c_write()*
  - *sensor_i2c_read()*
  - *bhy_delay_msec()*
  - *bhy_printf()*

When compiling the code, the compiler automatically detects the fixed-width types for the new platform. If that fails, a compiler warning will appear:

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

In this case, the *bhy.h* file should be modified to define the following fixed-width types: s8, s16, s32, u8, u16, u32.

4. Using the driver

4.1 Configuration

The driver has three parameters that should be configured in *bhy_uc_driver_config.h*.

4.1.1 BHY_DEBUG

This parameter is commented by default, which disables the driver to print the debug messages. When it's defined, the feature is eabled.

4.1.2 BHY_CALLBACK_MODE

This parameter is set to 1 by default, which enables the callback feature of the driver.

A callback is a software interrupt. The installed software callbacks are automatically called when the FIFO packets are decoded in the parsing process. The only drawback of this feature is that it consumes a little bit space on the RAM (approximate 350 bytes on a 32-bit system).

When this parameter is set to 0, the callback feature is disabled.

4.1.3 BHY_APPLICATION_BOARD

This parameter is set to 0 by default. When set to 1, it enables the driver to work properly on the Bosch Sensortec application board while limiting the I2C transaction size to 51 bytes.

4.2 RAM patch modifications

A compiled RAM patch is a pure binary file in *.fw format. A C constant array must be created using the binary data included in the firmware file. This data must be put in the *firmware.h* file.

4.3 Example for BHY1 existing virtual sensor

The examples are helpful in driver initialization, configuration, and data readout.

4.3.1 Initialization

Before initializing the driver, make sure that the I2C module in the MCU has been configured so that the I2C read and write function can be called by the *bhy_driver_init* function. The *bhy_driver_init* function initializes the hub device, downloads the RAM patch to the BHy1, verifies the CRC, launches the BHy system, and reports an error code if any. If the status return value is BHY_SUCCESS, the initialization is completed.

After the initialization, an interrupt will be generated to indicate that the hub is now in main execute mode, only then can the hub to be accessed.
Note: *bhy_fw* is the array name that includes the firmware to be used in this project.

Example for initialize:

```
int8_t rslt = BHY_SUCCESS;
rslt = bhy_driver_init(&bhy1_fw);

while (ioport_get_pin_level(BHY_INT));
```

```
while (!ioport_get_pin_level(BHY_INT));
```

4.3.2 Configuration

The required virtual sensors should be enabled and configured via *bhy_enable_virtual_sensor* function. Sensor callbacks, timestamp callbacks, and meta_event callbacks can be enabled via the functions *bhy_install_sensor_callback*, *bhy_install_timestamp_callback*, and *bhy_install_meta_event_callback* respectively.

Note:
To ensure the system stability, it is recommended always monitoring the *BHY_META_EVENT_TYPE_ERROR* and *BHY_META_EVENT_TYPE_SENSOR_ERROR* meta events. For detailed analysis, please contact us.

Example for configure the sensor with rotation vector sensor for below settings:

(Below MACRO was defined in 'bhy_uc_driver_types.h')

```
Enable Sensor ID: VS_TYPE_ROTATION_VECTOR (10)
Wake-up status: VS_WAKEUP (32)
Report latency: 0ms
Flash sensor: VS_FLUSH_NONE (not flush)
Change sensitivity: 0(not change sensitivity)
Dynamic range: 0(use commonly used unit)
```

```
#define ROTATION_VECTOR_SAMPLE_RATE    100
rslt = bhy_enable_virtual_sensor(VS_TYPE_ROTATION_VECTOR, VS_WAKEUP, ROTATION_VECTOR_SAMPLE_RATE, 0,
VS_FLUSH_NONE, 0, 0)
```

Example for install wakeup rotation vector sensor callbacks for parse fifo data:

```
rslt = bhy_install_sensor_callback(VS_TYPE_ROTATION_VECTOR, VS_WAKEUP, sensors_callback_rotation_vector)
```

Note: *sensors_callback_rotation_vector* is a function which will transfer the data from FIFO to desired data format. User should write this function according to the enabled virtual sensor.

4.3.3 Data readout

The data readout is done in two steps:

1. Read the FIFO data from the bhy into the MCU memory via the function *bhy_read_fifo*,
2. Parse the FIFO data into useful data structures via the function *bhy_parse_next_fifo_packet*.

Before modifying the data readout function, you need to familiarize yourself with the FIFO data format provided in the reference code.

Example for data read out and parse:

```
uint8_t                 *fifoptr          = NULL;
uint8_t                 bytes_left_in_fifo = 0;
uint16_t                bytes_remaining    = 0;
uint16_t                bytes_read         = 0;


int8_t read_parse_bhy_sensor_data (void)
```

```
{

    bhy_data_generic_t         fifo_packet;
    bhy_data_type_t            packet_type;
    int8_t   result;
     /* wait until the interrupt fires or there are bytes remaining in the fifo */
    while (!ioport_get_pin_level(BHY_INT) && !bytes_remaining)
            {
            }

    bhy_read_fifo(fifo + bytes_left_in_fifo, FIFO_SIZE - bytes_left_in_fifo, &bytes_read,
    &bytes_remaining);
    bytes_read           += bytes_left_in_fifo;
    fifoptr              = fifo;
    packet_type          = BHY_DATA_TYPE_PADDING;

    do
    {
        /* this function will call callbacks that are registered */
        result = bhy_parse_next_fifo_packet(&fifoptr, &bytes_read, &fifo_packet, &packet_type);

        /* prints all the debug packets */
        if (packet_type == BHY_DATA_TYPE_DEBUG)
        {
        bhy_print_debug_packet(&fifo_packet.data_debug, bhy_printf);
        }
             /* the logic here is that if doing a partial parsing of the fifo, then we should not parse
 the last 18 bytes (max length of a packet) so that we don't try to parse an incomplete packet */
    } while ((result == BHY_SUCCESS) && (bytes_read > (bytes_remaining ? MAX_PACKET_LENGTH : 0)));

    bytes_left_in_fifo = 0;

    if (bytes_remaining)
    {
        /* shifts the remaining bytes to the beginning of the buffer */
        while (bytes_left_in_fifo < bytes_read)
        {
        fifo[bytes_left_in_fifo++] = *(fifoptr++);
        }
    }
}
```

4.4 Example of custom sensor

The examples are helpful in driver initialization, configuration, and data readout for a custom sensor.

4.4.1 Initialization

The initialization sequence is same as existing BHY1 virtual sensors, please refer to 4.3.1.

4.4.2 Configuration for custom sensor

The custom sensor can be enabled and configured by *bhy_enable_virtual_sensor function. Sensor ID, wake-up or non wake-up, sample rate, report latency, flush or not, change sensitivity and dynamic range can be configured by input variable of this function.*

Sensor_callback function should be designed by user to gather custom sensor data structure, and to be installed while initializing, so it can be called when data read out for fifo decode.

bhy_sync_cus_evt_size function should be called during initialize phase to obtain existed virtual custom sensor data length in this ram patch, and this length will be used in fifo parse function.

Example for configure custom sensor for below settings:

(Below MACRO was defined in 'bhy_uc_driver_types.h')

```
Enable Sensor ID: VS_ID_CUS1 (26)
Wake-up status: VS_WAKEUP (32)
Report latency: 0ms
Flash sensor: VS_FLUSH_NONE (not flush)
Change sensitivity: 0(not change sensitivity)
Dynamic range: 0(use commonly used unit)
```

```
#define CUSTOM_SENSOR1_SAMPLE_RATE    5
rslt = bhy_enable_virtual_sensor(VS_ID_CUS1, VS_WAKEUP, CUSTOM_SENSOR1_SAMPLE_RATE, 0, VS_FLUSH_NONE, 0, 0)
```

Example for Custom sensor 1 *sensor callback* function design:

```
void sensors_callback(bhy_data_generic_t * sensor_data, bhy_virtual_sensor_t sensor_id)
{
    uint16_t i = 0;

    switch(sensor_id)
    {
        case VS_ID_CUS1:
        case VS_ID_CUS1_WAKEUP:
            DEBUG("Cus1 id = %d      len = %d ", sensor_id, bhy_get_cus_evt_size(VS_ID_CUS1));
            for(i = 0; i < (bhy_get_cus_evt_size(VS_ID_CUS1) - 1); i++)
            {
                DEBUG("%2x ", sensor_data->data_custom.data[i]);
            }
            DEBUG("\n\r");
            break;
        default:
            DEBUG("unknown id = %d\n\r", sensor_id);
            break;
    }
}
```

Example for install sensor callback for wakeup custom sensor 1:

```
rslt = bhy_install_sensor_callback(VS_ID_CUS1, VS_WAKEUP, sensors_callback);
```

4.4.3 Data readout and decode for custom sensor

The data read out and decode procedure is the same as existing BHY1 virtual sensors, please refer to 4.3.3.

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
1.0		Document creation	Aug. 20, 2015
1.1		Added prerequisites section	Nov. 24, 2015
1.2	4.1.3	Added BST_APPLICATION_BOARD setting	Mar. 21, 2015
1.3	2.1.1	Added URL for tool “bin2h”	Jun. 27, 2016
1.4	2.1.1 All	Updated tool “bin2h” to “fw2header” incl. URL Updated format and index of content	Aug. 08, 2016
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
1.6	All	Update for change to Atmel studio 6.0	Nov. 01, 2017
1.7	4.3 and 4.4	Update examples for initialization, configuration and data read out for existing virtual sensor and custom sensor, the document is aligned with code version 1.1.0.0	Feb. 05, 2018

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

Document number: BST-BHy1-SD001-00