Bosch Sensortec

Software Document
Interfacing reference code from generic driver

BHy1 - Interfacing reference code from generic driver

Technical Reference Codes	0 273 141 230
	0 273 141 231
Document revision	1.2
Release date	September 2016
Document number	BST-MHS-SD002-02
Notes	Data in this document are subject to change without notice.

Index of Contents

1. General Description

This manual is provided as a quick-start guide on how to use the BHA250 and BHI160 generic MCU driver to load the firmware onto the sensor hub, configure virtual sensors and stream sensor data using the sample code provided with the driver.

2. Getting Started

To get started with this guide, you will need the following:

1. Read and understand the BHA250/BHI160 datasheet:
   Doc#: BST-BHA250-DS000-01 / BST-BHA250-DS000-01

2. A BHA250/BHI160 device connected over I²C and interrupt line.

3. A programmable host.

4. Tested and debugged software for I²C communication specific to that host.

5. Read and understand the BHA250/BHI160 MCU Driver Porting Guide
   Doc#: BST-BHy1-SD001-00

For related document self-service visit:
https://www.bosch-sensortec.com/bst/support_tools/downloads/overview_downloads

3. Configuring the driver

All provided sample code require the BHY_CALLBACK_MODE to be enabled.
BST_APPLICATION_BOARD should be set to zero by default, unless your host is a Linux PC and you are using the application board as an I²C <-> USB converter. The BHY_DEBUG should be set to 0, as the release firmware does not output any debug messages.

4. Loading the BHA250/BHI160 RAM patch into host memory

The RAM patches are supplied as a *.fw binary file that needs to be loaded into the host memory in order to be pushed to the BHA250/BHI160 at power-on.

Depending on purpose and/or use case there are different *.fw files. E.g. a Stand-alone version to operate the BHA250/BHI160 with their integrated sensors, only. Or versions to operate the BHA250/BHI160 with external magnetometers, prior attached to the sensor-hub interface. For available RAM patches please refer to

BHA250 “documents & driver tab” under
https://www.bosch-sensortec.com/bst/products/all_products/bha250

BHI160 “documents & driver tab” under
https://www.bosch-sensortec.com/bst/products/all_products/bhi160

and follow the “Firmware” Download Link, select the appropriate *.fw binary file matching your desired configuration and download the file.

Hint: Use the **md5sum** given below the *.fw file to verify your data was not corrupted during the download due to potential network disturbances. For details on “how to” visit:
https://help.ubuntu.com/community/HowToMD5SUM

5. Walkthrough of the rotation_vector_example demo

5.1 Initialization and RAM patch loading

After power-on, the demo code runs some device-specific initialization:

```
  /* Initialize the SAM G55 system */
device_specific_initialization();
```

Then, the driver initializes the I²C read & write function, verifies the cip ID and load the RAM patch located in BHIfw.h into the attached BHA250/BHI160.

```
/* initializes the BHI160 and loads the RAM patch */
bhy_driver_init(_bhi_fw, _bhi_fw_len);


//wait for the interrupt pin to go down then up again
while (ioport_get_pin_level(EXT1_PIN_GPIO_1));
while (!ioport_get_pin_level(EXT1_PIN_GPIO_1));
```

5.2 Configuring sensors and enabling callbacks

The following lines enable the Rotation vector sensor and installs a function callback on the rotation vector data.

```
/* enables the absolute orientation vector and assigns the callback */
bhy_enable_virtual_sensor(VS_TYPE_ROTATION_VECTOR, VS_WAKEUP, 25, 0, VS_FLUSH_NONE, 0, 0);
bhy_install_sensor_callback(VS_TYPE_ROTATION_VECTOR, VS_WAKEUP, sensors_callback);
```

The most important parts are:

5.2.1 bhy_enable_virtual_sensor

This is the main function to call to enable all virtual sensors. It has to be called for every virtual sensor to enable in the system. After this function call, data will automatically get stored in the FIFO once the event happen.

5.2.2 VS_TYPE_ROTATION_VECTOR

This is the virtual sensor type to enable. Note that the driver supports all types that are defined in the datasheet. However, not all virtual sensor types are enabled for a specific hardware / firmware / extension combination. E.g. the BHA250 firmware won't support VS_TYPE_GYROSCOPE as there is no physical gyroscope data source available in the BHA250, the Stand-alone firmware won't support VS_TYPE_ORIENTATION as there is no physical magnetometer data source in Stand-alone operation. Using the BHI160 with a magnetometer attached to the sensor-hub interface and its corresponding RAM patch will support any virtual sensor type from bhy_uc_driver_types.h.

Here is the complete list:

```
typedef enum {
  VS_TYPE_ACCELEROMETER      = VS_ID_ACCELEROMETER,
  VS_TYPE_GEOMAGNETIC_FIELD      = VS_ID_MAGNETOMETER,
  VS_TYPE_ORIENTATION       = VS_ID_ORIENTATION,
  VS_TYPE_GYROSCOPE        = VS_ID_GYROSCOPE,
  VS_TYPE_LIGHT         = VS_ID_LIGHT,
  VS_TYPE_PRESSURE       = VS_ID_BAROMETER,
  VS_TYPE_TEMPERATURE       = VS_ID_TEMPERATURE,
  VS_TYPE_PROXIMITY       = VS_ID_PROXIMITY,
  VS_TYPE_GRAVITY       = VS_ID_GRAVITY,
  VS_TYPE_LINEAR_ACCELERATION  = VS_ID_LINEAR_ACCELERATION,
  VS_TYPE_ROTATION_VECTOR     = VS_ID_ROTATION_VECTOR,
  VS_TYPE_RELATIVE_HUMIDITY      = VS_ID_HUMIDITY,
  VS_TYPE_AMBIENT_TEMPERATURE  = VS_ID_AMBIENT_TEMPERATURE,
  VS_TYPE_MAGNETIC_FIELD_UNCALIBRATED  = VS_ID_UNCALIBRATED_MAGNETOMETER,
  VS_TYPE_GAME_ROTATION_VECTOR   = VS_ID_GAME_ROTATION_VECTOR,
  VS_TYPE_GYROSCOPE_UNCALIBRATED   = VS_ID_UNCALIBRATED_GYROSCOPE,
  VS_TYPE_SIGNIFICANT_MOTION     = VS_ID_SIGNIFICANT_MOTION,
  VS_TYPE_STEP_DETECTOR     = VS_ID_STEP_DETECTOR,
  VS_TYPE_STEP_COUNTER     = VS_ID_STEP_COUNTER,
  VS_TYPE_GEOMAGNETIC_ROTATION_VECTOR = VS_ID_GEOMAGNETIC_ROTATION_VECTOR,
  VS_TYPE_HEART_RATE       = VS_ID_HEART_RATE,
  VS_TYPE_TILT         = VS_ID_TILT_DETECTOR,
  VS_TYPE_WAKEUP       = VS_ID_WAKE_GESTURE,
  VS_TYPE_GLANCE       = VS_ID_GLANCE_GESTURE,
  VS_TYPE_PICKUP       = VS_ID_PICKUP_GESTURE,
  VS_TYPE_ACTIVITY_RECOGNITION   = VS_ID_ACTIVITY
} bhy_virtual_sensor_t;
```

5.2.3 25, 0,

These two parameters are respectively the desired frequency (in Hertz) and the maximum latency (in milliseconds). Note:

1. The requested frequency may not be supported by this sensor type, in which case the closest will be chosen.

2. For the case of virtual sensors that stream data (e.g. accelerometer, gyroscope, rotation vector, etc.), the interrupt rate of the whole system will match the shortest enabled sensor latency.

5.2.4 bhy_install_sensor_callback

This is the function to enable a software interrupt (also known as callback) when the specified virtual sensor event is read out of the FIFO.

5.2.5 sensors_callback

This is the name of the function that will be called with the sensor data when a VS_TYPE_ROTATION_VECTOR gets read out of the FIFO. Note that a single callback function can handle many different sensor IDs, how different sensor IDs can be assigned to different callback functions.

5.2 Configuring sensors and enabling callbacks

The last part of the sample code is the FIFO readout procedure:

```
/* continuously read and parse the fifo */
 while (true) {
   /* wait until the interrupt fires */
   /* unless we already know there are bytes remaining in the fifo */

   while (!ioport_get_pin_level(EXT1_PIN_GPIO_1) && !bytes_remaining) ;

   bhy_read_fifo(array+bytes_left_in_fifo, ARRAYSIZE-bytes_left_in_fifo, &bytes_read, &bytes_remaining);

   bytes_read += bytes_left_in_fifo;

   fifoptr = array;
   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 );
   /* 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 ? 18 : 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)
       array[bytes_left_in_fifo++] = *(fifoptr++);
   }
}
```

It is highly recommended to leave this portion as-is and use the callback method for handling data. The only recommended modification is to replace the active polling while loop by an interrupt-triggered readout.

5.4. Special cases / advanced driver usage

5.4.1 Timestamp usage

In gesture_recognition_example, it keeps track of the BHI160 timestamp. This is done in few steps.

1. Create a 32-bit global variable to store the timestamp.

```
/* system timestamp */
u32 g_system_timestamp = 0;
```

2. Create a timestamp callback function to update this variable on new timestamp packets.

```
void timestamp_callback( bhy_data_scalar_u16_t *new_timestamp ) {
 /* updates the system timestamp  */
 bhy_update_system_timestamp( new_timestamp, &g_system_timestamp);
}
```

3. Install the timestamp callback.

```
bhy_install_timestamp_callback(VS_WAKEUP, timestamp_callback);
```

This variable will then always contain the latest timestamp reported (1 tick = 1/32000 second). When using virtual sensor callbacks, it will be equal to the timestamp of the packet being parsed by design.

5.4.2 One_shot sensors

Some sensors types, such as glance, pickup or significant motion, are defined by the Android open-source project as “one-shot”. Meaning that they get disabled automatically when the event happens. For constant monitoring, the callback should re-enable them.

To learn more on reporting modes and Android sensor types visit:
https://source.android.com/devices/sensors/report-modes.html

6. Legal disclaimer

6.1 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.

6.2 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.

6.3 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.

7. Document history and modification

Rev. No	Chapter	Description of modification/changes	Date
1.1	All	Updated format and index of content	Sep. 19, 2016
1.0		Document creation	Mar. 21, 2016