BOSCH

BHI260AB BHA260AB

Ultra-low power, high performance, programmable Smart Sensor with integrated IMU

BHI260AB-BHA260AB SDK Quick Start Guide

Document revision	2.11	
Document release date	May 18th, 2020	
Document number	BST-BHI260AB-AN000-09	
Technical reference code(s)	0 273 141 367	0 273 141 392
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

List of Figures

List of Tables

Abbreviations

BST	Bosch Sensortec
BSX	Bosch Sensortec Fusion Library
FIFO	First In First Out
GCC	GNU Compiler Collection
RAM	Random Access Memory
SDK	Software Development Kit
USB	Universal Serial Bus
TRNG	True Random Number Generator
RDRAND	Read Random

1 Introduction to the SDK

This document briefly describes the process of developing firmware for the BHI260AB/BHA260AB. It provides instructions to,

- setup the development environment
- build the SDK
- getting started with custom configuration files

For more details about hardware configuration, refer to the BHI260AB/BHA260AB Datasheet.
For more details about developing new drivers, refer to the following manual and user guide:

- BHI260AB-BHA260AB Programmer's Manual
- BHI260AB-BHA260AB Evaluation Setup Guide

The BHI260AB/BHA260AB SDK can be used to develop a custom firmware image. The customization includes,

- modifying the board configuration,
- changing the mapping of pins,
- changing the device orientation,
- memory allocated to the FIFO,
- create custom drivers, which can run algorithms or other tasks
- data injection for processor in the loop verification

The firmware built by the SDK can be downloaded to the BHI260AB/BHA260AB's RAM or the BHI260AB's external flash.

For more details, refer to the BHI260AB-BHA260AB Programmer's Manual.

2 Setup in Windows

This chapter describes how to install the required tools in Windows. Jump to Chapter 3 to install the SDK for Linux. The BHI260AB/BHA260AB SDK supports two toolchains: ARC GNU toolchain and Synopsys Metaware. This guide focuses on how to build the SDK with the ARC GNU toolchain. Since the SDK generates signed firmware images and the signing tool requires the True Random Number Generator feature of the CPU to generate a valid signature, the CPU that is used to build the SDK must support the RDRAND instruction.

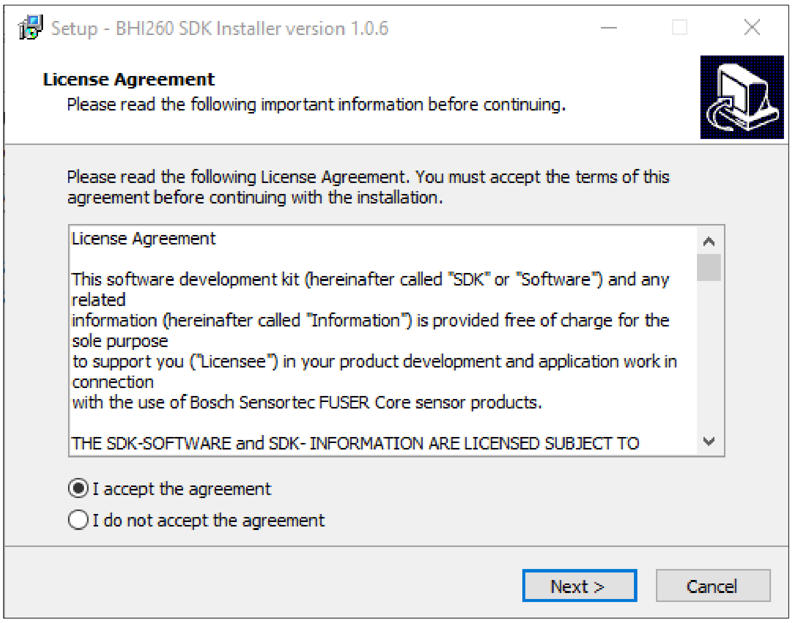
2.1 Installing the compiler and support tools

The GNU Toolchain for ARC Processors can be downloaded from the Synopsys Github Website. Download the file "*arc_gnu_2019.09_ide_win_install.exe*" or newer and run this setup installer executable. This will primarily install the Eclipse IDE and the ARC GNU Compiler.

2.2 Installing the SDK

For Windows system, a SDK installer is provided.

1. Execute the *BHI260_SDK_V1.0.6_Install.exe* or newer, accept the license agreement and click *Next* as shown in Figure 1.



Setup - BHI260 SDK Installer version 1.0.6

License Agreement

Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

License Agreement

This software development kit (hereinafter called "SDK" or "Software") and any related
information (hereinafter called "Information") is provided free of charge for the sole purpose
to support you ("Licensee") in your product development and application work in connection
with the use of Bosch Sensortec FUSER Core sensor products.

THE SDK-SOFTWARE and SDK- INFORMATION ARE LICENSED SUBJECT TO

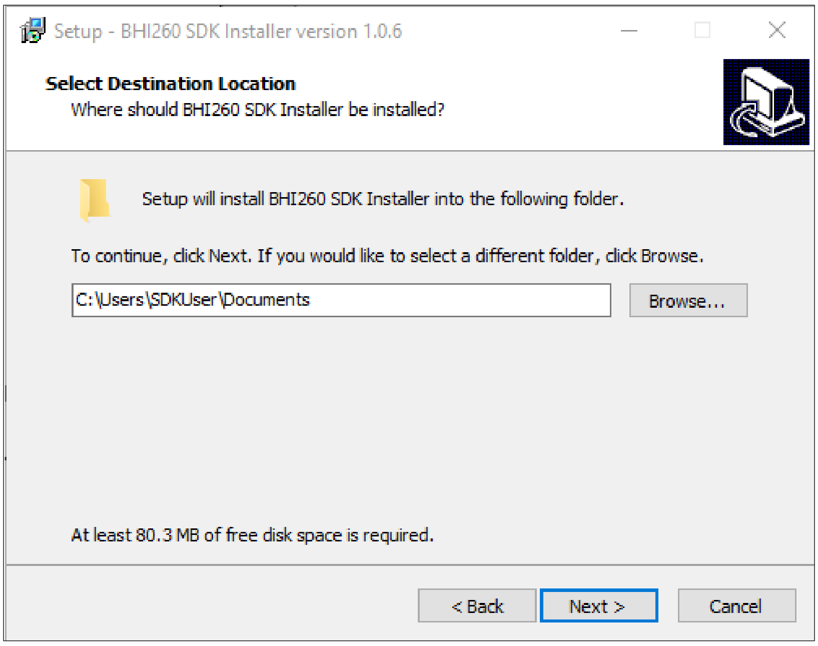
● I accept the agreement

○ I do not accept the agreement

Figure 1: BHI260 or BHA260 SDK installer

2. Figure 2 illustrates changing of the installation directory.

Then in the desired SDK directory, either “*BHI260_SDK_VX.Y.Z*” or “*BHA260_SDK_VX.Y.Z*” is created.



Setup - BHI260 SDK Installer version 1.0.6

Select Destination Location

Where should BHI260 SDK Installer be installed?

Setup will install BHI260 SDK Installer into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

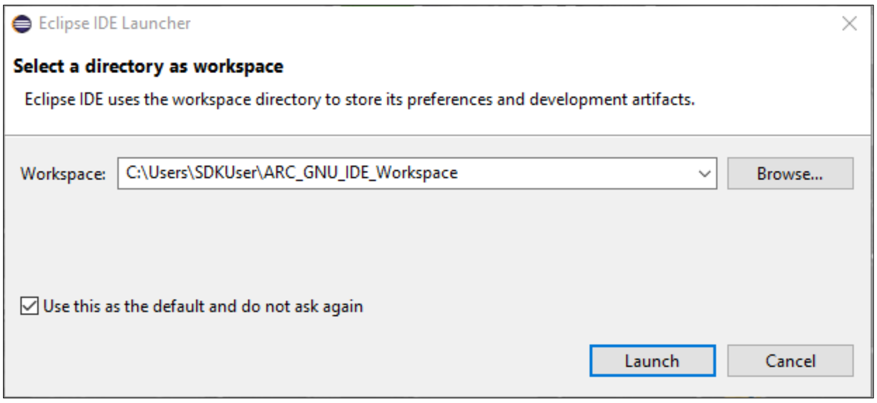
C:\Users\SDKUser\Documents

At least 80.3 MB of free disk space is required.

Figure 2: Installed directory

2.3 Importing the SDK into Eclipse

1. Setup Eclipse
   a. Run the Eclipse IDE which should be a shortcut on the Desktop generically named *“ARC GNU IDE YYYY.MM(-rcN) Eclipse”*. For example *“ARC GNU IDE 2019.09 Eclipse”.*
   b. During the first launch, you will be prompted to select a workspace. This is an empty directory that stores multiple projects. Select the preferred workspace directory.



Eclipse IDE Launcher

Select a directory as workspace

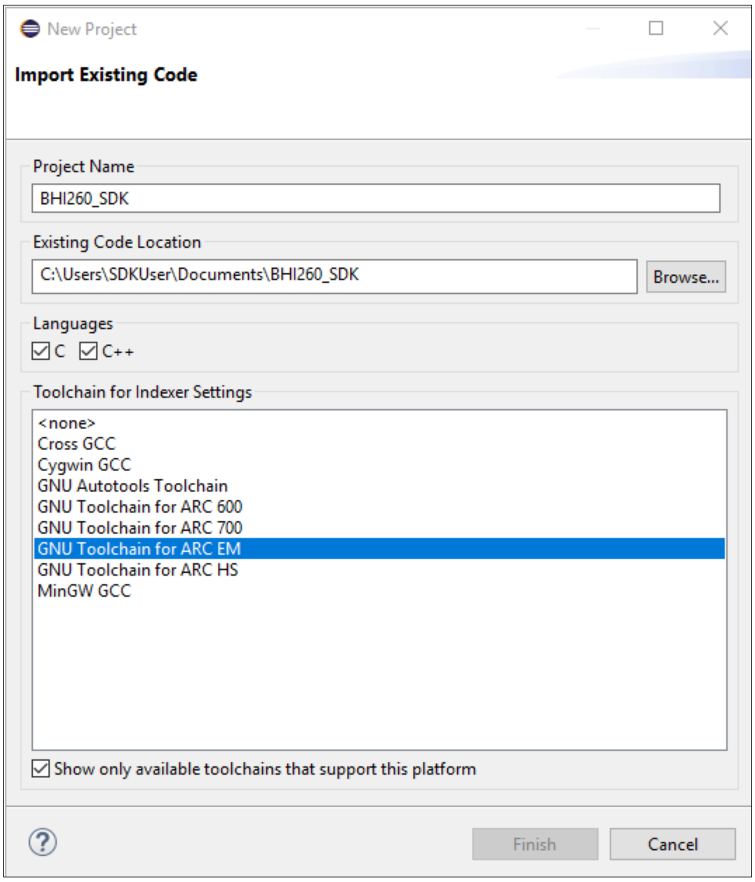
Eclipse IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\SDKUser\ARC_GNU_IDE_Workspace

☑ Use this as the default and do not ask again

Figure 3: Eclipse workspace prompt

2. Import the BHI260/BHA260 SDK as a project
    a. In the Eclipse IDE, go to *File > New > Makefile Project with Existing Code*
    b. In the prompt, select the SDK directory and select a suitable project name like in the image below and select *Finish*.



New Project

Import Existing Code

Project Name: BHI260_SDK

Existing Code Location: C:\Users\SDKUser\Documents\BHI260_SDK

Languages: ☑ C ☑ C++

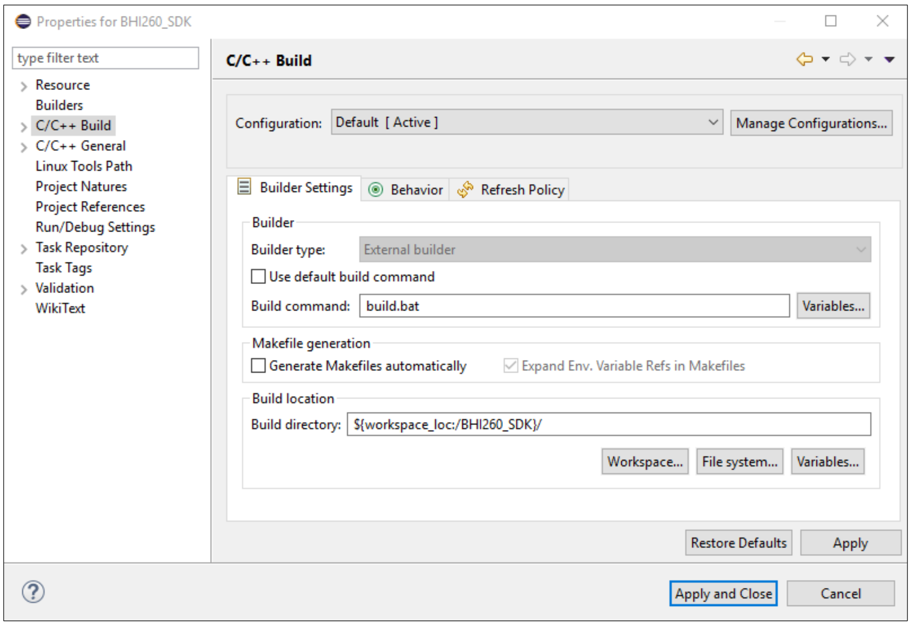
Toolchain for Indexer Settings:

<none>
Cross GCC
Cygwin GCC
GNU Autotools Toolchain
GNU Toolchain for ARC 600
GNU Toolchain for ARC 700
GNU Toolchain for ARC EM
GNU Toolchain for ARC HS
MinGW GCC

☑ Show only available toolchains that support this platform

Figure 4: Importing a directory as an existing Makefile project

    c. If the Welcome tab is open, close it to reveal the *Project Explorer*.
3. Link the project build to the batch script that builds the firmware.
    a. Under Windows, the building of the firmware is managed by a batch script named *build.bat* which can be found in the root of the SDK directory.
    b. Right-click on the *BHI260_SDK or BHA260_SDK* project and select *Properties*.



Configuration: Default [Active]

Builder type: External builder

□ Use default build command

Build command: build.bat

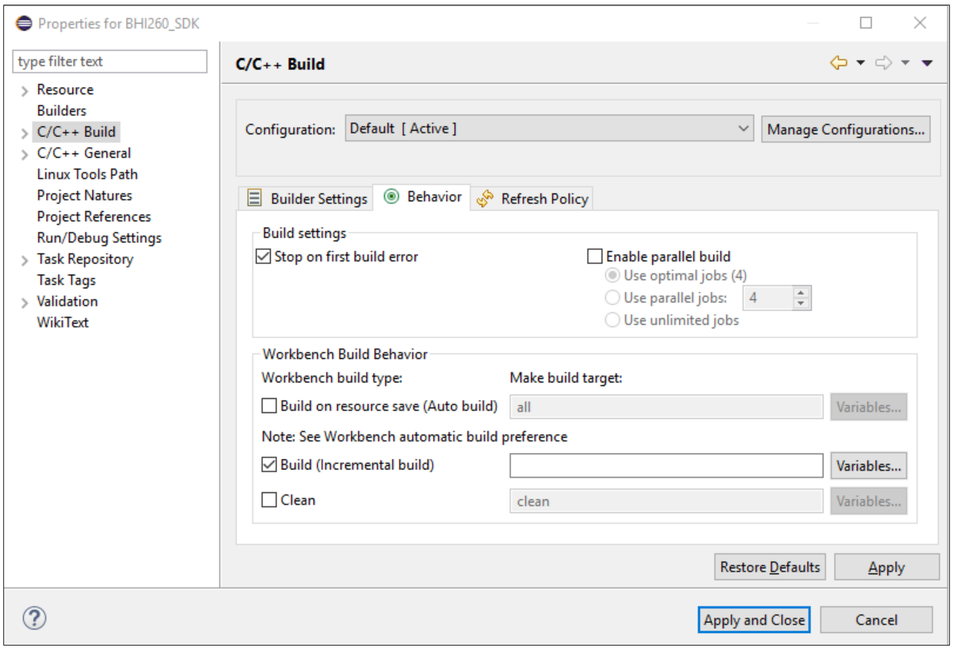
□ Generate Makefiles automatically ☑ Expand Env. Variable Refs in Makefiles

Build directory: ${workspace_loc:/BHI260_SDK}/

Figure 5: Configuring the build trigger

c. Under *C/C++ Build / Builder settings*, deselect *Use default build command* and refer the image for selecting the build trigger. Select *Apply*.

d. Under the *C/C++ Build / Behavior*, deselect Clean and remove the command *all* from the *Build behavior* like in the image below. Select *Apply and Close*.



Configuration: Default [Active]

☑ Stop on first build error

□ Enable parallel build

● Use optimal jobs (4)

○ Use parallel jobs: 4

○ Use unlimited jobs

Workbench build type: Make build target:

□ Build on resource save (Auto build) all

Note: See Workbench automatic build preference

☑ Build (Incremental build)

□ Clean clean

Figure 6: Configuring the build trigger's arguments

e. Click on the build icon .This should run *build.bat* and the progress is visible in the console located at the bottom.

4. Locate the built firmware.

a. The firmware build can be found under *release/gccfw* in the root directory of the SDK. If the firmware is built by Metaware rather than ARC GNU toolchain, it can be found under *release/fw* instead.

Table 1: Pre-build SDK directory structure in Windows

SDK File/Directory	Description
apps	Directory which contains the source code for applications running outside the sensor framework
boards	Configuration files for the supported development boards and sensors
cmake	CMake files used to build the SDK
common	Source code for initialization code and reference header files
docs	SDK documentation
drivers	Source codes of sensor drivers from Bosch Sensortec
drivers_custom	Source codes of additional custom drivers
gdb	Support files for using gdb
kernel	Exported symbols for kernel-mode firmware
libs	Linkable binary image and header files for API libraries
user	Entry code for user-mode firmware, source code for custom user-mode RAM patches
win64	Executable image manipulation utilities, command line interface
build.bat	Shell script used to set up build directory and build the specified target

3 Setup in Linux

3.1 Installing the ARC GNU toolchain and support tools

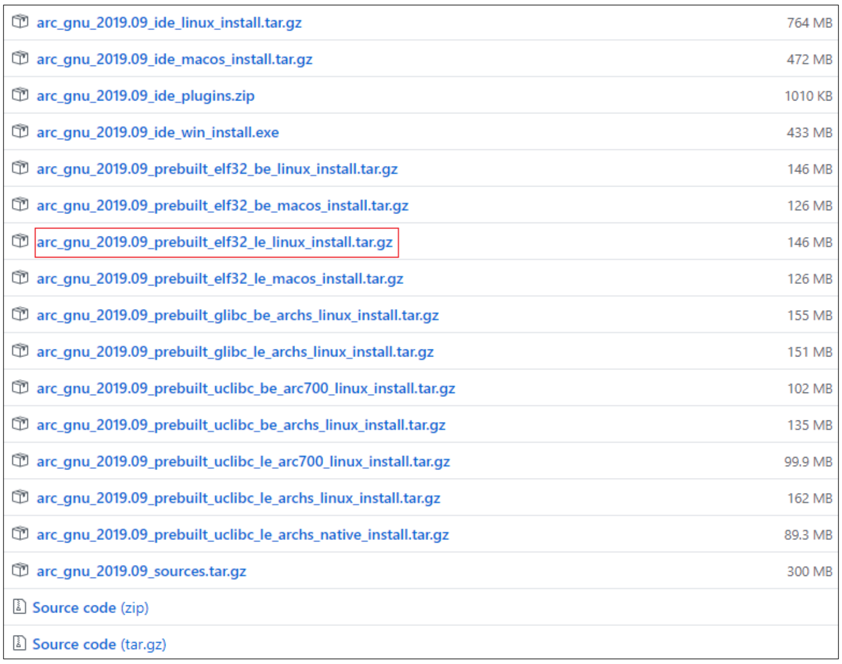
To get started, the following system requirements must be met:

- 64-bit Linux operating system (Ubuntu 14.04 LTS or later)
- At least 1.1 GB of free disk space

Before the SDK can be used, ARC GNU toolchain, CMake, and other necessary dependencies must be installed.

The operations in this guide have been verified on Ubuntu 14.04 LTS and 16.04 LTS.

1. Downloading the ARC GNU toolchain
   The ARC GNU toolchain releases are available on the Synopsys Github Website. A pre-built toolchain that supports elf32 little-endian hosts is required.
   In this example, the 2019.09 release is used. This release can be downloaded from the previous releases download page. The correct installation package to download is “arc_gnu_2019.09_prebuilt_elf32_le_linux_install.tar.gz”.



arc_gnu_2019.09_ide_linux_install.tar.gz	764 MB
arc_gnu_2019.09_ide_macos_install.tar.gz	472 MB
arc_gnu_2019.09_ide_plugins.zip	1010 KB
arc_gnu_2019.09_ide_win_install.exe	433 MB
arc_gnu_2019.09_prebuilt_elf32_be_linux_install.tar.gz	146 MB
arc_gnu_2019.09_prebuilt_elf32_be_macos_install.tar.gz	126 MB
arc_gnu_2019.09_prebuilt_elf32_le_linux_install.tar.gz	146 MB
arc_gnu_2019.09_prebuilt_elf32_le_macos_install.tar.gz	126 MB
arc_gnu_2019.09_prebuilt_glibc_be_archs_linux_install.tar.gz	155 MB
arc_gnu_2019.09_prebuilt_glibc_le_archs_linux_install.tar.gz	151 MB
arc_gnu_2019.09_prebuilt_uclibc_be_arc700_linux_install.tar.gz	102 MB
arc_gnu_2019.09_prebuilt_uclibc_be_archs_linux_install.tar.gz	135 MB
arc_gnu_2019.09_prebuilt_uclibc_le_arc700_linux_install.tar.gz	99.9 MB
arc_gnu_2019.09_prebuilt_uclibc_le_archs_linux_install.tar.gz	162 MB
arc_gnu_2019.09_prebuilt_uclibc_le_archs_native_install.tar.gz	89.3 MB
arc_gnu_2019.09_sources.tar.gz	300 MB
Source code (zip)	
Source code (tar.gz)	

Figure 7: GNU toolchain releases download page

2. Installing the GNU toolchain
    a. Run the following commands to extract the GNU toolchain installation package:

    ```
    $ tar -xvf arc_gnu_2019.09_prebuilt_elf32_le_linux_install.tar.gz
    $ sudo mkdir -p /opt/arc_gcc
    $ sudo mv arc_gnu_2019.09_prebuilt_elf32_le_linux_install /opt/arc_gcc
    ```

    b. Run the following commands to verify the GNU toolchain has been installed successfully:

    ```
    $ cd /opt/arc_gcc/arc_gnu_2019.09_prebuilt_elf32_le_linux_install/bin/
    $ ./arc-elf32-gcc -dumpversion
    ```

    c. Update the PATH variable to include "opt/arc_gcc/arc_gnu_2019.09_prebuilt_elf32_le_linux_install/bin/". This can be done by modifying the shell start-up script as appropriate. For example, edit "/etc/profile" with the following command.

    ```
    $ sudo nano /etc/profile
    ```

    d. Add the path to the file by adding the following line.

```
export PATH=$PATH:/opt/arc_gcc/arc_gnu_2019.09_prebuilt_elf32_le_linux_install/bin
```

3. Installing the CMake and other dependencies
    a. To install the CMake, run the following commands:

    ```
    $ sudo apt-get install cmake
    $ cmake --version
    ```

    b. To install the other dependencies or tools if necessary, run the following commands.

    ```
    $ sudo apt-get install libelf-dev
    $ sudo apt-get install g++
    $ sudo apt-get install lib32stdc++6
    ```

    c. It is highly recommended to install ninja to speed up the build process by parallel building.

    ```
    $ sudo apt-get install ninja-build
    ```

3.2 Installing the SDK into Linux

The SDK is released as an installer "*BHI260_SDK_VX.Y.Z_Install.sh*" or "*BHA260_SDK_VX.Y.Z_Install.sh*".
Take BHI260 SDK V1.0.6 for example, to make the installer executable, run the following command:

```
$./BHI260_SDK_V1.0.6_Install.sh
```

Bosch Sensortec License must be accepted by typing yes in the command line prompt. After, the installer prompts to move to a preferred directory. The default installation directory is "**${HOME}/Bosch_Sensortec_Fuser2_SDK**".
The SDK has the directory structure as shown in Table 2.

Table 2: Pre-build SDK directory structure in Linux

SDK File/Directory	Description
apps	Directory that contains the source code for applications running outside the sensor framework
boards	Configuration files for the supported development boards and sensors
cmake	CMake files used to build the SDK
common	Source code for initialization code and reference header files

docs	SDK documentation
drivers	Source codes of sensor drivers from Bosch Sensortec
drivers_custom	Source codes of additional custom drivers
gdb	Support files for using gdb
kernel	Exported symbols for kernel-mode firmware
libs	Linkable binary image and header files for API libraries
user	Entry code for user-mode firmware, source code for custom user-mode RAM patches
utils	Executable image manipulation utilities, command line interface
build.sh	Shell script used to set up a build directory and build the specified SDK

4 Building the SDK and Loading firmware into the BHI260AB/BHA260AB

In Windows, click on the build icon 🔨 in the Eclipse IDE or executing the build.bat script triggers the build process.
In Linux, run the build script in its root:

```
$ ./build.sh
```

build and *release* directories are created after executing the build script. If both the ARC GNU compiler and the Metaware compiler are available on path, the Metaware compiler is used. To override this behavior and force the use of the ARC GNU compiler, add the option 'USE_GCC' as an argument to the build script.

```
$ ./build.sh USE_GCC
```

```
$ ./build.bat USE_GCC
```

The generated firmware "**.fw*" file is a binary loadable RAM image in the case of the BHA260. In the case of the BHI260, 2 images are generated, one for the RAM and other for the FLASH. With successive build triggers, all previously generated files under the *build* and *release* directories are removed and new firmware files are generated under the *release/gccfw* or *release/fw* folder.

The generated "**.fw*" file can be verified by using the bhy2cli tool. The bhy2cli is a command line tool based on the COINES tool that interfaces with the BHI260AB or BHA260AB through Bosch Sensortec's application board. The tool can be used to load and run, standard and custom firmware images among other features.

For example, running

```
$ bhy2cli -b release\fw\Bosch_SHUTTLE_BHI260_BMM150.fw -c 34:25
```

loads the firmware file `Bosch_SHUTTLE_BHI260_BMM150.fw` for the board configuration `Bosch_SHUTTLE_BHI260_BMM150.cfg` and switches on streaming of the sensor ID 34 at 25Hz to the terminal. Refer the BHI260AB-BHA260AB Evaluation Setup Guide for more information on building the *bhy2cli* tool.

5 Adding a BSX based new custom virtual driver

In order to demonstrate how one can add a custom driver to the SDK, there are two drivers *VirtBSXLeanDeviceOrientation* and *VirtBSXCustomAccelDataSource* that are already included in the SDK as examples but not used in any of the firmware images.

Both *VirtBSXLeanDeviceOrientation* and *VirtBSXCustomAccelDataSource* are in the *drivers_custom* directory of the SDK. *VirtBSXCustomAccelDataSource* receives accelerometer data from the Bosch Sensortec Fusion Library, but does not send it to the host. Instead it triggers *VirtBSXLeanDeviceOrientation* which receives the data, processes it and stores the processed data in the requested FIFO.

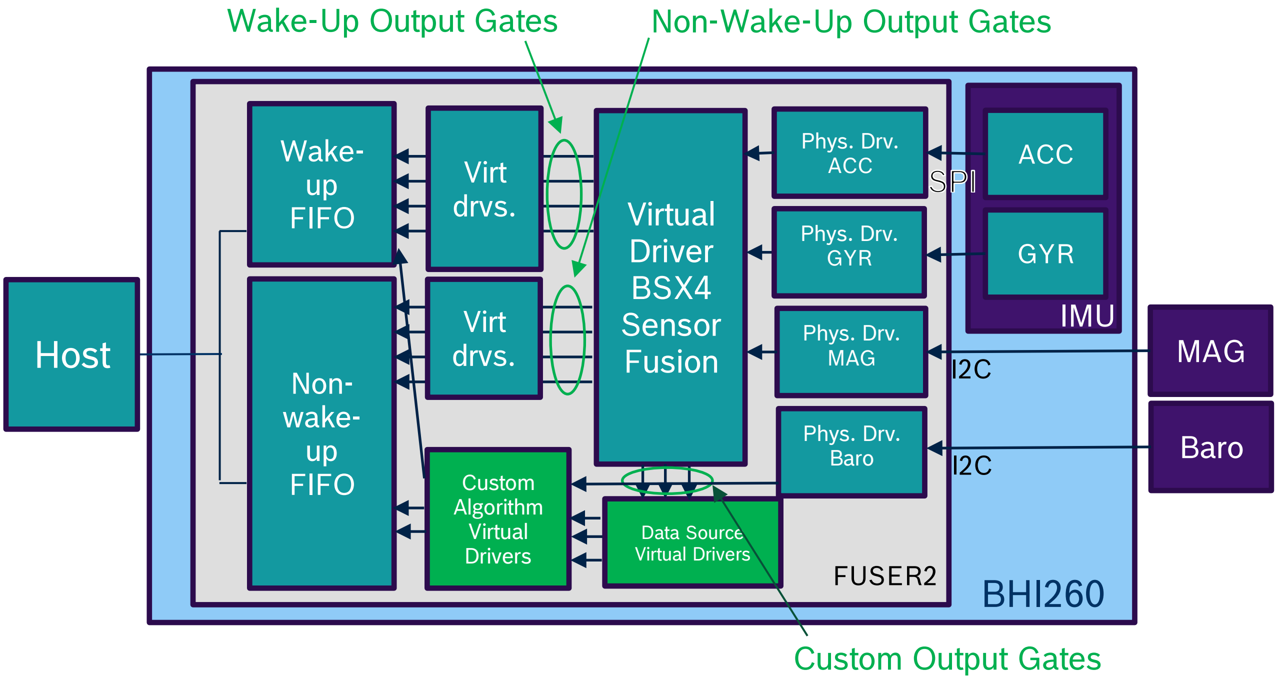


Figure 8: Architecture of physical and virtual drivers

For more information on how to develop a new physical sensor driver or virtual sensor driver in the SDK, refer to the BHI260AB-BHA260AB Programmer’s Manual.

5.1 Driver directory structure

The sensor driver code must be in its own directory in the *drivers* directory of the SDK. The directory name should reflect the device name and driver type, for example *VirtBSXLeanDeviceOrientation*.

Table 3: Driver directory content

File in Driver Directory	Description
CMakeLists.txt	Build description of the driver
VirtBSXLeanDeviceOrientation.c	Source code of the driver
Header file	Header file typically defining register locations and other constants for the driver if needed

5.2 Writing driver code

The dependency between the two virtual sensors is described below.

For a more detailed and complete information on how to program sensor drivers, refer the BHI260AB-BHA260AB Programmer’s Manual.

“hidden =TRUE” means the sensor is not visible to host, and it only provides data source.

```
VIRTUAL_SENSOR_DESCRIPTOR VirtualSensorDescriptor
descriptor_virt_bsx_custom_accel_data_source = {
    .physicalSource = {
        .sensor = {
            .type = {
                .value = BSX_INPUT_ID_ACCELERATION,
                .flags = DRIVER_TYPE_PHYSICAL_FLAG,
            },
        },
    },

    .info = {
        .id = DRIVER_ID,
        .version = DRIVER_REV,
    },

    .type = {
        .value =
SENSOR_TYPE_BSX(BSX_CUSTOM_ID_ACCELERATION_CORRECTED),
        .flags = DRIVER_TYPE_VIRTUAL_FLAG,
        .wakeup_ap = FALSE,
        .hidden = TRUE,
    },
    .expansionData = {
        .f32 = OUTPUT_SCALING_FACTOR,
    },

    .maxRate = 800.0F,
    .minRate = 1.5625F,

    .outputPacketSize = sizeof(output_3axis_t),
    .priority = PRIORITY_2,

    .initialize = NULL,
    .handle_sensor_data =
BSXSupport_trigger_custom_sensors,
};
```

VirtBSXCustomAccelDataSource is the trigger source of *VirtBSXLeanDeviceOrientation*.

```
VIRTUAL_SENSOR_DESCRIPTOR VirtualSensorDescriptor
descriptor_virt_bsx_lean_device_orientation = {
    .triggerSource = {
        .sensor = {
            .type = {
                .value =
SENSOR_TYPE_BSX(BSX_CUSTOM_ID_ACCELERATION_CORRECTED)
,
                .flags = DRIVER_TYPE_VIRTUAL_FLAG,
            },
        },
    },

    .physicalSource = {
        .sensor = {
            .type = {
                .value = BSX_INPUT_ID_ACCELERATION,
                .flags = DRIVER_TYPE_PHYSICAL_FLAG,
            },
        },
    },

    .info = {
        .id = DRIVER_ID,
        .version = DRIVER_REV,
    },

    .type = {
        .value = SENSOR_TYPE_CUSTOMER_VISIBLE_START,
        .flags = DRIVER_TYPE_VIRTUAL_FLAG,
        .wakeup_ap = FALSE,
    },

    .maxRate = 800.0F,
    .minRate = 1.5625F,

    .outputPacketSize = sizeof(output_t),
    .priority = PRIORITY_2,

    .initialize = ldo_initialize,
    .handle_sensor_data = ldo_handle_sensor_data,
    .mode_changed = ldo_on_power_mode_changed,
};
```

The Sensor ID is made visible ID to the host.

Figure 9: Driver descriptor overview

5.3 Selecting a driver ID

To add a new virtual sensor driver, the first step is to select the available driver ID for compilation. Unless the driver to be developed is already included in the SDK, users may choose any unused 8-bit number. There is a python script in the root directory of the SDK. Running it will show the existing driver names and associated driver IDs. Using this script will need an existing installation of Python.

```
$ python find_BHy2_driver_IDs.py
```

In this excerpt, we have selected the driver IDs **131** and **132** in the Driver CMakeLists.txt file (See section 5.4). Each driver has a unique driver ID.

5.4 Driver CMakeLists.txt File

The below mentioned *CMakeLists.txt* file automatically pulls in the sources from each driver. It is used by the build system at link time to associate the driver ID listed with a driver's object file. More driver IDs can be defined in the same way. Usually users do not need to modify it.

Take *drivers_custom/VirtBSXLeanDeviceOrientation/CMakeLists.txt* for example:

```
SET(DRIVER_ID 132)

get_filename_component( DRIVER_KEY ${CMAKE_CURRENT_LIST_DIR} NAME)

project(${DRIVER_KEY} C)

FILE(GLOB SOURCES "*.c")

include_directories(../../libs/BSXSupport/includes/
                    ../../libs/BSX/includes/)

ADD_ARC_DRIVER(${DRIVER_KEY} ${DRIVER_ID} ${SOURCES})
```

Figure 10: Driver CmakeLists.txt

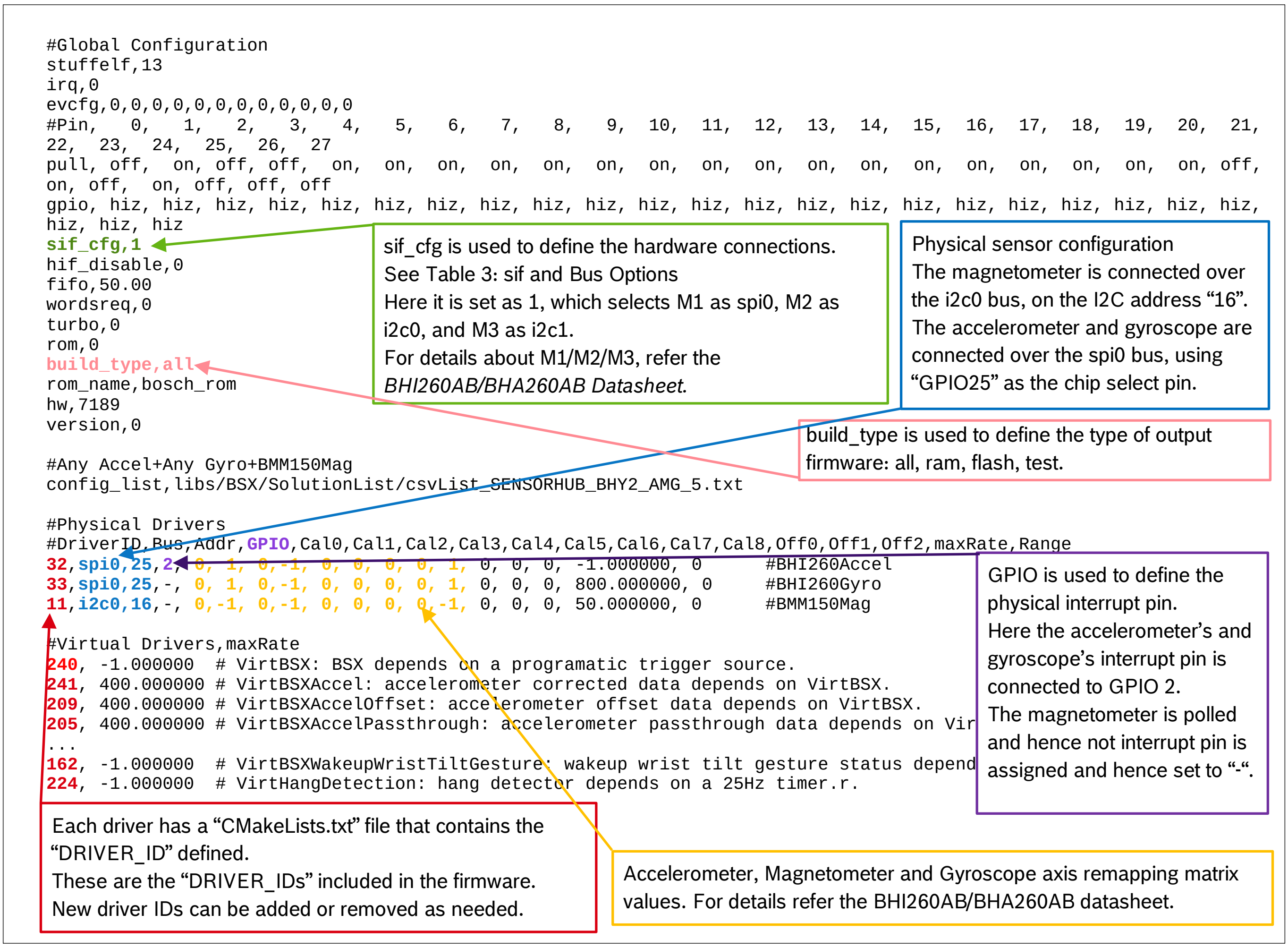
5.5 Brief introduction to the board configuration file

Board configuration files are used to specify the configuration for a firmware build. A board configuration file consists of a global configuration section, a physical driver configuration section and a virtual driver configuration section. Lines can be commented with a hash (#) and are commented until the end of the current line.

- Default configuration of GPIO pins
- Sensor interface configurations (SPI, I2C masters)
- Allocation of FIFO memory
- CPU speed: long run (20MHz) or turbo (50 MHz)
- Building firmware for Host boot or external Flash or both
- Configuration parameters for BSX fusion library
- List of physical drivers to be linked into the firmware file and their characteristics
- List of virtual drivers to be linked Into the firmware file

All board configuration files are in the *boards* directory.

Take *boards/Bosch_SHUTTLE_BHI260_BMM150.cfg* for example:



```
#Global Configuration
stuffelf,13
irq,0
evcfg,0,0,0,0,0,0,0,0,0,0,0,0
#Pin,    0,    1,    2,    3,    4,    5,    6,    7,    8,    9,   10,   11,   12,   13,   14,   15,   16,   17,   18,   19,   20,   21,
22,   23,   24,   25,   26,   27
pull, off,   on, off, off,   on,   on,   on,   on,   on,   on,   on,   on,   on,   on,   on,   on,   on,   on,   on,   on,   on, off,
on, off,   on, off, off, off
gpio, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz, hiz,
hiz, hiz, hiz
sif_cfg,1
hif_disable,0
fifo,50.00
wordsreq,0
turbo,0
rom,0
build_type,all
rom_name,bosch_rom
hw,7189
version,0

#Any Accel+Any Gyro+BMM150Mag
config_list,libs/BSX/SolutionList/csvList_SENSORHUB_BHY2_AMG_5.txt

#Physical Drivers
#DriverID,Bus,Addr,GPIO,Cal0,Cal1,Cal2,Cal3,Cal4,Cal5,Cal6,Cal7,Cal8,Off0,Off1,Off2,maxRate,Range
32,spi0,25,2, 0, 1, 0,-1, 0, 0, 0, 0, 1, 0, 0, 0, -1.000000, 0       #BHI260Accel
33,spi0,25,-, 0, 1, 0,-1, 0, 0, 0, 0, 1, 0, 0, 0, 800.000000, 0      #BHI260Gyro
11,i2c0,16,-, 0,-1, 0,-1, 0, 0, 0, 0,-1, 0, 0, 0, 50.000000, 0       #BMM150Mag

#Virtual Drivers,maxRate
240, -1.000000  # VirtBSX: BSX depends on a programatic trigger source.
241, 400.000000 # VirtBSXAccel: accelerometer corrected data depends on VirtBSX.
209, 400.000000 # VirtBSXAccelOffset: accelerometer offset data depends on VirtBSX.
205, 400.000000 # VirtBSXAccelPassthrough: accelerometer passthrough data depends on Vir
...
162, -1.000000  # VirtBSXWakeupWristTiltGesture: wakeup wrist tilt gesture status depend
224, -1.000000  # VirtHangDetection: hang detector depends on a 25Hz timer.r.
```

sif_cfg is used to define the hardware connections.
See Table 3: sif and Bus Options
Here it is set as 1, which selects M1 as spi0, M2 as i2c0, and M3 as i2c1.
For details about M1/M2/M3, refer the *BHI260AB/BHA260AB Datasheet.*

Physical sensor configuration
The magnetometer is connected over the i2c0 bus, on the I2C address "16".
The accelerometer and gyroscope are connected over the spi0 bus, using "GPIO25" as the chip select pin.

build_type is used to define the type of output firmware: all, ram, flash, test.

GPIO is used to define the physical interrupt pin.
Here the accelerometer's and gyroscope's interrupt pin is connected to GPIO 2.
The magnetometer is polled and hence not interrupt pin is assigned and hence set to "-".

Each driver has a "CMakeLists.txt" file that contains the "DRIVER_ID" defined.
These are the "DRIVER_IDs" included in the firmware.
New driver IDs can be added or removed as needed.

Accelerometer, Magnetometer and Gyroscope axis remapping matrix values. For details refer the BHI260AB/BHA260AB datasheet.

Figure 11: Board configuration file overview

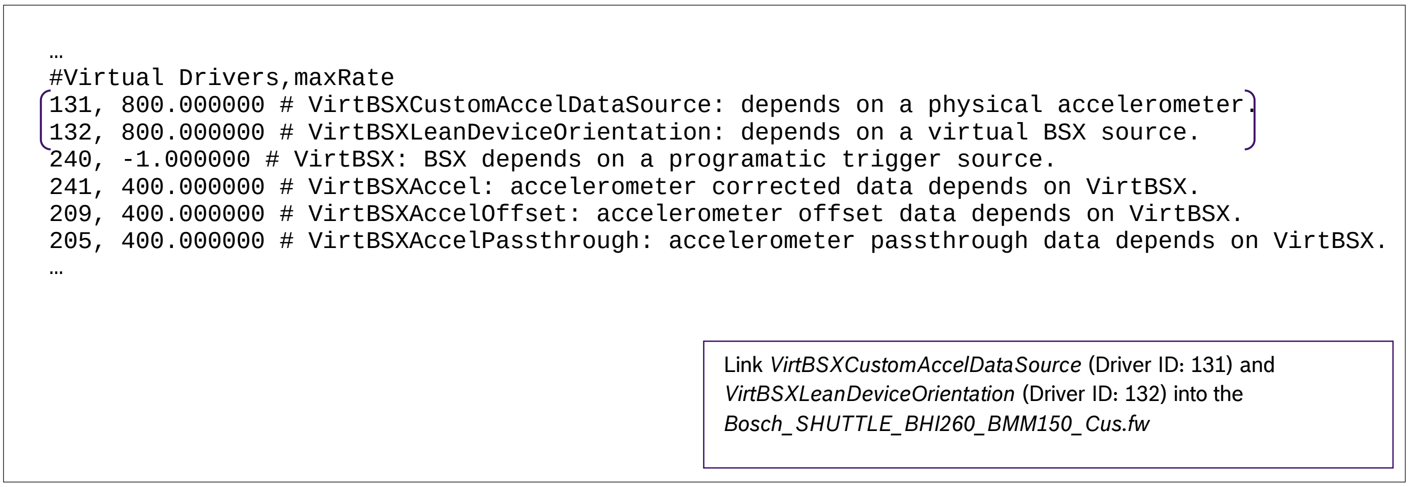
Table 4: sif and bus options

sif	M1	M2	M3
0	SPI0	SPI1	I2C1
1	SPI0	I2C0	I2C1
2	I2C0	SPI0	I2C1

For more details about the sif configuration, refer the BHI260AB/BHA260AB datasheet. For details about the board configuration file, refer the BHI260AB-BHA260AB Programmer's manual.

5.6 Modifying the board configuration file

In order to add the *VirtBSXLeanDeviceOrientation* and *VirtBSXCustomAccelDataSource* virtual sensors into the *Bosch_SHUTTLE_BHI260_BMM150_Cus.fw* one must add a new configuration file *boards/Bosch_SHUTTLE_BHI260_BMM150_Cus.cfg* (take *Bosch_SHUTTLE_BHI260_BMM150.cfg* as the reference) and add the virtual drivers to the virtual sensor list in the respective "*.cfg" file as shown below.



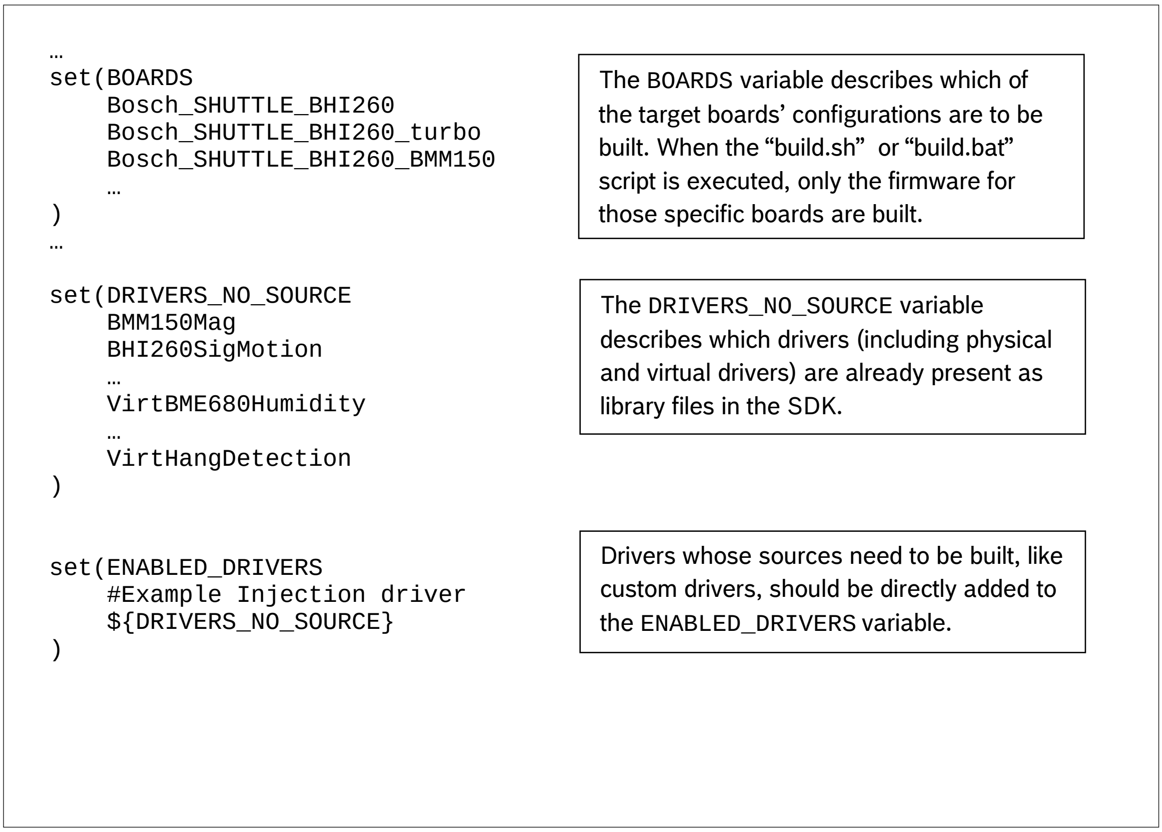
```
…
#Virtual Drivers,maxRate
131, 800.000000 # VirtBSXCustomAccelDataSource: depends on a physical accelerometer.
132, 800.000000 # VirtBSXLeanDeviceOrientation: depends on a virtual BSX source.
240, -1.000000 # VirtBSX: BSX depends on a programatic trigger source.
241, 400.000000 # VirtBSXAccel: accelerometer corrected data depends on VirtBSX.
209, 400.000000 # VirtBSXAccelOffset: accelerometer offset data depends on VirtBSX.
205, 400.000000 # VirtBSXAccelPassthrough: accelerometer passthrough data depends on VirtBSX.
…
```

Link *VirtBSXCustomAccelDataSource* (Driver ID: 131) and *VirtBSXLeanDeviceOrientation* (Driver ID: 132) into the *Bosch_SHUTTLE_BHI260_BMM150_Cus.fw*

Figure 12: Modifying the board configuration file

5.7 Brief introduction to the SDK configuration file

In brief, all SDK generated firmware images include both the pre-built kernel image and user images. This configuration file includes board configuration files, enabled drivers, libraries, etc.
The SDK has one configuration file *common/config.7189_di03_rtos_bhi260.cmake*, which can be edited as needed.



```
…
set(BOARDS
    Bosch_SHUTTLE_BHI260
    Bosch_SHUTTLE_BHI260_turbo
    Bosch_SHUTTLE_BHI260_BMM150
    …
)
…

set(DRIVERS_NO_SOURCE
    BMM150Mag
    BHI260SigMotion
    …
    VirtBME680Humidity
    …
    VirtHangDetection
)


set(ENABLED_DRIVERS
    #Example Injection driver
    ${DRIVERS_NO_SOURCE}
)
```

The BOARDS variable describes which of the target boards' configurations are to be built. When the "build.sh" or "build.bat" script is executed, only the firmware for those specific boards are built.

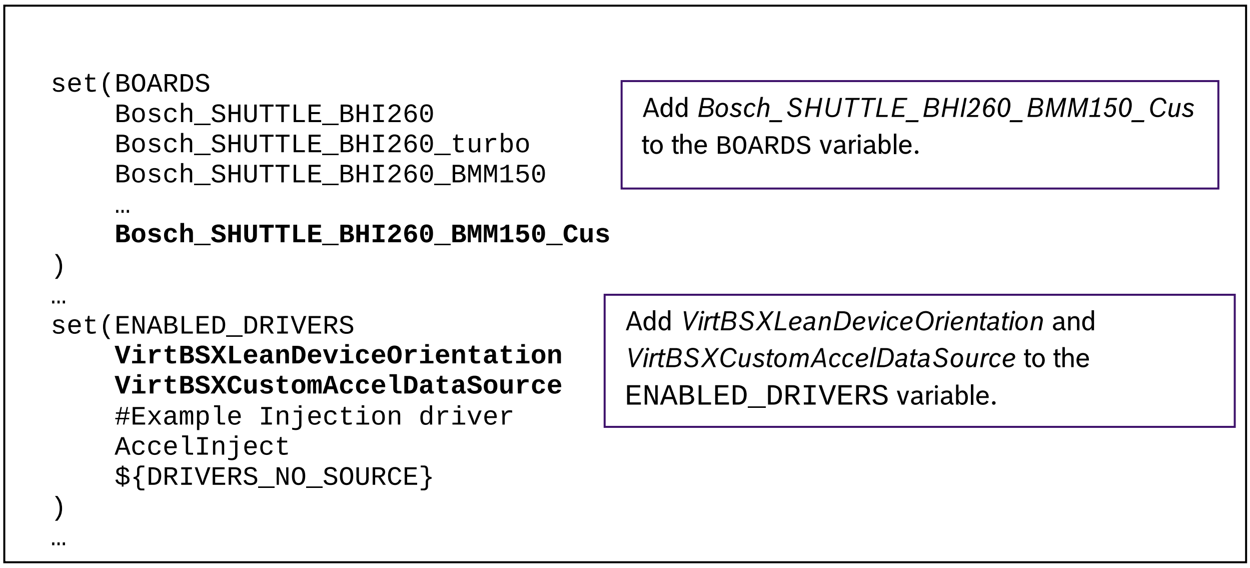
The DRIVERS_NO_SOURCE variable describes which drivers (including physical and virtual drivers) are already present as library files in the SDK.

Drivers whose sources need to be built, like custom drivers, should be directly added to the ENABLED_DRIVERS variable.

Figure 13: Overview of the SDK configuration file

5.8 Modifying the SDK configuration file

In order to build a firmware that contains the reference custom drivers for the target board configuration the *common/config.7189_di03_rtos_bhi260.cmake* needs to be modified as shown below.



```
set(BOARDS
    Bosch_SHUTTLE_BHI260
    Bosch_SHUTTLE_BHI260_turbo
    Bosch_SHUTTLE_BHI260_BMM150
    …
    Bosch_SHUTTLE_BHI260_BMM150_Cus
)
…
set(ENABLED_DRIVERS
    VirtBSXLeanDeviceOrientation
    VirtBSXCustomAccelDataSource
    #Example Injection driver
    AccelInject
    ${DRIVERS_NO_SOURCE}
)
…
```

Add *Bosch_SHUTTLE_BHI260_BMM150_Cus* to the BOARDS variable.

Add *VirtBSXLeanDeviceOrientation* and *VirtBSXCustomAccelDataSource* to the ENABLED_DRIVERS variable.

Figure 14: Modifying the SDK configuration file

5.9 Building the custom firmware

To build only a specific board configuration file, the name of the board can be passed as an argument. For example, in Linux this would look like,

```
$ ./build.sh Bosch_SHUTTLE_BHI260_BMM150_Cus
```

If more than one board needs to be built in a similar way, a semicolon is required between board names like below,

```
$ ./build.sh "Bosch_SHUTTLE_BHI260_BMM150_Cus;Bosch_SHUTTLE_BHI260"
```

The build.bat file for Windows can accept similar arguments.

The custom firmware is now available under *release/gccfw* as *Bosch_SHUTTLE_BHI260_BMM150_Cus.fw*. Like with the reference firmware, you can use the bhy2cli like below to load the firmware and view the lean orientation sensor's output using the generic type handle.

```
$ bhy2cli -a 160:"Lean Orientation":2:c:c –b release\fw\Bosch_SHUTTLE_BHI260_BMM150_Cus.fw –c
160:1
```

5.10 Lean orientation example

With the new custom virtual drivers inside the firmware, the host should also be able to configure the sensor ID and parse sensor events from the FIFO.
The corresponding host side example of the virtual sensor VirtBSXLeanDeviceOrientation called "Lean Orientation" is provided separately. Refer to the BHI260AB-BHA260AB Evaluation Setup Guide on how to verify and evaluate the aforementioned newly integrated virtual sensors.

6 Adding a non-Bosch Sensortec Fusion Library related new custom virtual driver

This chapter takes *VirtAltitude* for example to describe how to add a non-Bosch Sensortec Fusion Library related new custom virtual sensor driver. The steps are same as in Chapter 5, except that some details vary depending on the actual requirements.

VirtAltitude is in the *drivers_custom* directory. It creates custom altitude data and sends it to the respective FIFO.

For more information on how to develop a physical/virtual sensor driver in the SDK, refer to the BHI260AB-BHA260AB Programmer's Manual.

6.1 Driver directory structure

The sensor driver code must be in its own directory in the *drivers_custom* directory of the SDK. The directory name should reflect the device name and driver type, for example *VirtAltitude*.

Table 5: Driver directory content

File in Driver Directory	Description
CMakeLists.txt	Build description of the driver
VirtAltitude.c	Source code of the driver
Header file	Header file typically defining register locations and other constants for the driver if needed

6.2 Writing driver code

Below are code snippets *VirtAltitude* driver in Figure 16, which explains its trigger source and how to exchange reference sea level values with the host through the parameter interface.
For more information on how to program sensor drivers, refer the BHI260AB-BHA260AB Programmer's Manual.

```
…
#define DRIVER_REV        (4u)
…

#define PARAM_PAGE_OPTIONAL_SDK         (8)
#define OPTIONAL_SDK_PARAM_ALTITUDE_SEE_LEVEL   (0x00)

bool optionalSDKPageWriteHandler(UInt8 param, UInt16 length, UInt8 buffer[])
…

bool optionalSDKPageReadHandler(UInt8 param, UInt16 length, UInt8 buffer[], UInt16 *ret_length)
…

static SensorStatus virt_altitude_initialize_sensor(VirtualSensorDescriptor *self)
{
    (void) registerWriteParamHandler(PARAM_PAGE_OPTIONAL_SDK, optionalSDKPageWriteHandler);
    (void) registerReadParamHandler(PARAM_PAGE_OPTIONAL_SDK, optionalSDKPageReadHandler);
    verbose("altitude initialize\n");
    return SensorOK;
}
…

VIRTUAL_SENSOR_DESCRIPTOR VirtualSensorDescriptor virt_altitude_descriptor =
{
    .triggerSource =
    {
        .sensor =
        {
            .type =
            {
                .value = BSX_INPUT_ID_PRESSURE,
                .flags = DRIVER_TYPE_PHYSICAL_FLAG,
            },
        },
    },

   .physicalSource =
   {
        .sensor =
        {
            .type =
            {
                .value = BSX_INPUT_ID_PRESSURE,
                .flags = DRIVER_TYPE_PHYSICAL_FLAG,
            },
        },
    },

   .info =
   {
      .id = DRIVER_ID,
      .version = DRIVER_REV,
   },

   .type =
   {
        .value = SENSOR_TYPE_ALTITUDE,
        .flags = DRIVER_TYPE_VIRTUAL_FLAG,
        .wakeup_ap = FALSE,
        .no_decimation = FALSE,
        .on_change = FALSE,
   },

    .outputPacketSize = sizeof(output_t),
    .priority = PRIORITY_6,    /* Low priority */
    .initialize = virt_altitude_initialize_sensor,
    .handle_sensor_data = virt_altitude_handle_sensor_data,
};
```

The driver's version number is 4.

Use the Parameter page 0x08, index 0x00 to set and get the reference pressure at sea level. It is 4 bytes register to contain an unsigned 32 bit value. Host can set and get this reference pressure configuration in run-time to get accurate altitude estimation.

VirtAltitude's trigger source is physical pressure sensor.

VirtAltitude's sensor ID is a visible ID, which means that it is visible to the HOST.

Figure 15: Driver descriptor overview

6.3 Selecting a driver ID

To add a new virtual sensor driver, the first step is to select the available virtual driver ID for compilation. Unless the driver to be developed is already included in the SDK, users may choose any unused 8-bit number.

In this excerpt, we have selected the driver ID **123** in the CMakeLists.txt file. Each driver has a unique driver ID.

6.4 Driver CMakeLists.txt file

The *CMakeLists.txt* file pulls in all the sources from the root directory of each driver. It is used by the build system while linking to associate the driver ID listed with a driver's object file. Additional driver IDs can be defined in the same way. This generic file typically needs no modification.
Take *drivers/VirtAltitude/CMakeLists.txt* for example:

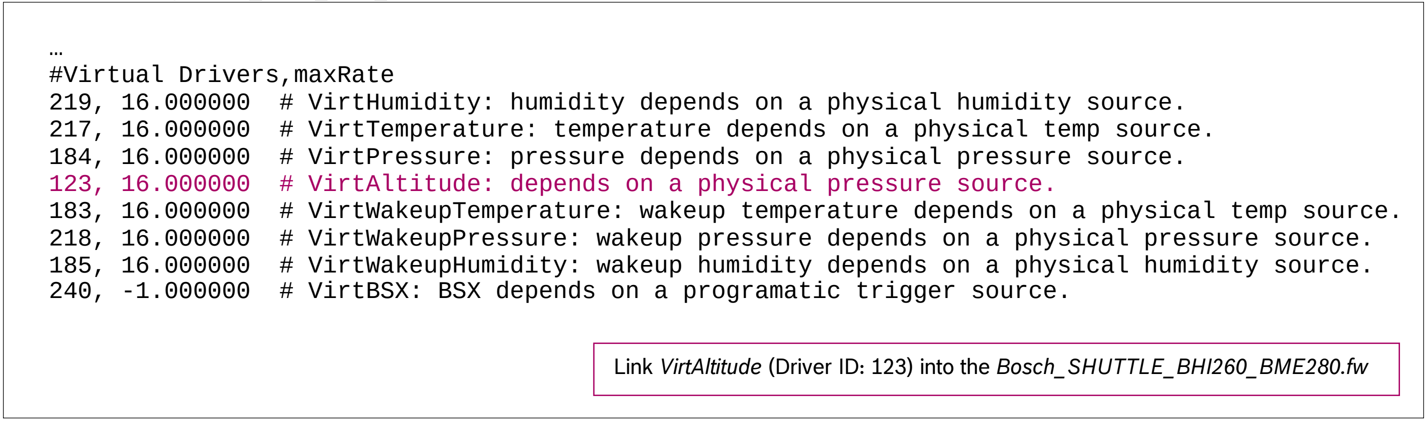
```
SET(DRIVER_ID 123)
get_filename_component( DRIVER_KEY ${CMAKE_CURRENT_LIST_DIR} NAME)

project(${DRIVER_KEY} C)

FILE(GLOB SOURCES "*.c")

ADD_ARC_DRIVER(${DRIVER_KEY} ${DRIVER_ID} ${SOURCES})
```

Figure 16: Driver CmakeLists.txt

6.5 Modifying the board configuration file

The example below describes how to include VirtAltitude in Bosch_SHUTTLE_BHI260_BME280.fw by editing the existing configuration file “$SDK/boards/Bosch_SHUTTLE_BHI260_BME280.cfg” as shown below.

In order to build a firmware that contains the *VirtAltitude* virtual driver for the target board configuration the *common/config.7189_di03_rtos_bhi260.cmake* needs to be modified as shown below.

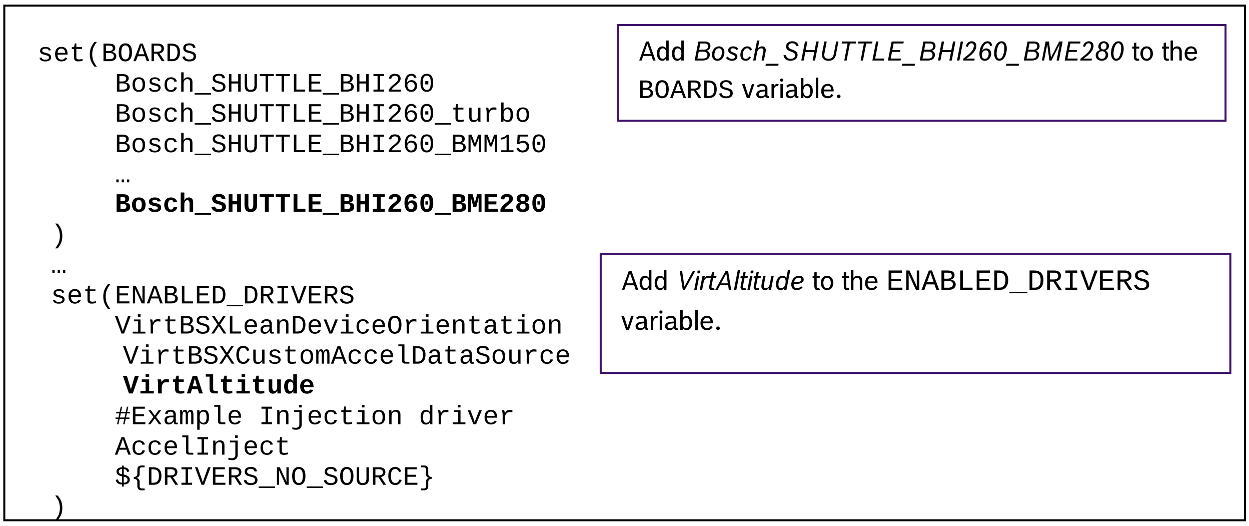


```
…
#Virtual Drivers,maxRate
219, 16.000000  # VirtHumidity: humidity depends on a physical humidity source.
217, 16.000000  # VirtTemperature: temperature depends on a physical temp source.
184, 16.000000  # VirtPressure: pressure depends on a physical pressure source.
123, 16.000000  # VirtAltitude: depends on a physical pressure source.
183, 16.000000  # VirtWakeupTemperature: wakeup temperature depends on a physical temp source.
218, 16.000000  # VirtWakeupPressure: wakeup pressure depends on a physical pressure source.
185, 16.000000  # VirtWakeupHumidity: wakeup humidity depends on a physical humidity source.
240, -1.000000  # VirtBSX: BSX depends on a programatic trigger source.
```

Link *VirtAltitude* (Driver ID: 123) into the *Bosch_SHUTTLE_BHI260_BME280.fw*

Figure 17: Modifying the board configuration file

6.6 Modifying the SDK configuration file



```
set(BOARDS
     Bosch_SHUTTLE_BHI260
     Bosch_SHUTTLE_BHI260_turbo
     Bosch_SHUTTLE_BHI260_BMM150
     …
     Bosch_SHUTTLE_BHI260_BME280
 )
 …
 set(ENABLED_DRIVERS
     VirtBSXLeanDeviceOrientation
      VirtBSXCustomAccelDataSource
      VirtAltitude
     #Example Injection driver
     AccelInject
     ${DRIVERS_NO_SOURCE}
 )
```

Add *Bosch_SHUTTLE_BHI260_BME280* to the BOARDS variable.

Add *VirtAltitude* to the ENABLED_DRIVERS variable.

Figure 18: Modifying the SDK configuration file

6.7 Build the custom firmware

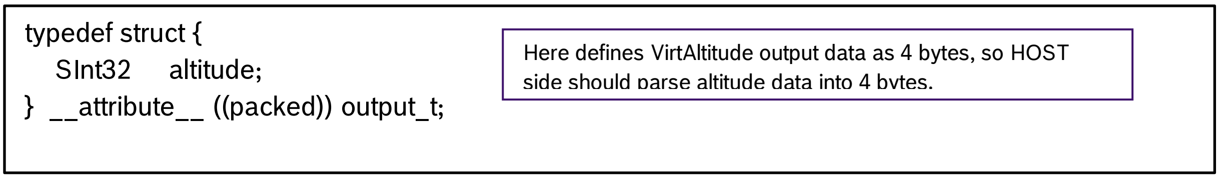
Like in Chapter 2 for Windows and Chapter 3 for Linux system, trigger the respective build.

The custom firmware is now available in *release/gccfw* as *Bosch_SHUTTLE_BHI260_BME280.fw*. Like with the reference firmware, you can run the bhy2cli like below to load the firmware and view the lean orientation sensor’s output using the generic type handle.

```
$ bhy2cli -a 161:"Altitude":4:s32 –b release\fw\Bosch_SHUTTLE_BHI260_BME280.fw –c 161:1
```

6.8 Altitude example

With new virtual sensor drivers in the firmware, a new sensor data parser should also be added to the host. An example of the virtual sensor VirtAltitude is provided separately.
The virtual altitude sensor's output unit is in centimeters.
For information on how to verify and evaluate the BHI260AB/BHA260AB's virtual sensors, refer to the BHI260AB-BHA260AB Evaluation Setup Guide.



```
typedef struct {
    SInt32     altitude;
}  __attribute__ ((packed)) output_t;
```

Here defines VirtAltitude output data as 4 bytes, so HOST side should parse altitude data into 4 bytes.

Figure 19: Altitude output data structure

7 Integrating a library and applying it to the custom sensor driver

This chapter covers how to integrate a library and use it in a driver. The library can be found under *libs/template* and the driver under *VirtIntegrateLibTemplate*. Chapter 5 and 6 cover the development of a custom driver.

7.1 Library directory structure

The library directory and its files must be in the *libs* directory of the SDK. The directory name should indicate the library name, for example, *libs/template*.

Table 6: Library directory content

Source in Library Directory	Description
CMakeLists.txt	Build description of the library
template.sdk.cmake	CMake file of the library
libtemplate.a	Library file
includes	Header files directory of the library

7.2 Implementing a sensor driver that uses the library

Table 7: Driver directory content

Source in Library Directory	Description
CMakeLists.txt	Build description of the driver
VirtIntegrateLibTemplate.c	Driver file

Content of CmakeLists.txt. Note the includes required.

```
SET(DRIVER_ID 111)

get_filename_component( DRIVER_KEY ${CMAKE_CURRENT_LIST_DIR} NAME)

project(${DRIVER_KEY} C)

FILE(GLOB SOURCES "*.c")

include_directories(../../libs/template/includes/)

add_definitions("-DDESCRIPTOR_NAME=virt_${DRIVER_KEY}_desc")

ADD_ARC_DRIVER(${DRIVER_KEY} ${DRIVER_ID} ${SOURCES})
```

Figure 20: CMakeLists.txt

7.3 Modifying the board configuration file

In order to build a firmware that contains library and driver, the board configuration file needs modification.

Using the *boards/Bosch_SHUTTLE_BHI260.cfg* as reference.

```
#Virtual Drivers,maxRate
111, -1.000000  #VirtIntegrateLibTemplate: an example for integrate library
240, -1.000000   # VirtBSX: BSX depends on a programatic trigger source.
241, 400.000000 # VirtBSXAccel: accelerometer corrected data depends on VirtBSX.
209, 400.000000 # VirtBSXAccelOffset: accelerometer offset data depends on VirtBSX.
```

Figure 21: Modifying the board configuration file

7.4 Modifying the SDK configuration file

In order to build a firmware that contains library and driver, the SDK configuration file needs modification.

SDK configuration file *common/config.7189_di03_rtos_bhi260.cmake*

```
.....
set(LIBRARIES
    SensorInterfaceRAM
    HostInterface${HOST_INTERFACE}RAM
    Outerloop
    SensorInterfaceInit
    MetawarePrintf
    MetawareDouble
    Hooks
    BSXSupport
    SensorCalibration
    OscTrim
     DMA
     template
)

.....
# libraries linked to standard board images
set(BOARDS_LIBS
    MetawareDouble
     MetawarePrintf
     template
)
...

set(ENABLED_DRIVERS
     BMM150Mag
     BHI260Accel
     ....
   VirtIntegrateLibTemplate

# Example Injection driver
   AccelInject

  ${DRIVERS_NO_SOURCE}
)
.....
```

Figure 22: Modifying the SDK configuration file

7.5 Build the custom firmware

Like in Chapter 2 for Windows and Chapter 3 for Linux system, trigger the respective build.

8 Glossary

8.1 Virtual Sensor

A Virtual Sensor is a term used to identify the output of one or more algorithms. This output is available in the FIFO and can be identified and referenced by the host of the BHI260AB/BHA260AB using a Sensor ID.

8.2 Driver ID

A Virtual sensor driver is responsible for implementing the interface between the Software Framework and the algorithm, among other tasks. Each driver has a unique ID in the SDK which is referred to as the Driver ID. This Driver ID is used to select which driver can be included into a firmware build.

8.3 Sensor ID

The Sensor ID is a unique identifier for a Virtual sensor. This Sensor ID is defined as an 8 bit unsigned integer value. A list of all Virtual sensors and their corresponding sensor IDs also known as a FIFO event IDs are described in the datasheet in the table Overview of FIFO Event IDs. The Sensor ID is defined as part of the driver's descriptor.

9 Legal disclaimer

1.1 Engineering Samples

Engineering samples are marked with an asterisk (*) or (e) or (E). Engineering samples may vary from the valid technical specifications of the product series contained in this guide. Therefore, they are 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.

1.2 Product Use

Bosch Sensortec products are developed for the consumer goods industry. They may only be used within the parameters of this product guide. They are not fit for use in life-sustaining or safety-critical systems. Safety-critical systems are those for which a malfunction is expected to lead to bodily harm, death or severe property damage. In addition, they shall not be used directly or indirectly for military purposes (including but not limited to nuclear, chemical or biological proliferation of weapons or development of missile technology), nuclear power, deep sea or space applications (including but not limited to satellite technology).

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

The purchaser shall indemnify Bosch Sensortec from all third party claims arising from any product use not covered by the parameters of this product 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 to inform Bosch Sensortec without delay of all safety-critical incidents.

1.3 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.

10 Document history

Rev. No	Chapter	Description of modification/changes	Date
2.8	All	Main release	2020-02-04
2.9	All	Updated references	2020-02-24
2.10	All	Added Glossary	2020-04-24
2.11	All	Updated bhy2cli command formats	2020-05-18

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

www.bosch-sensortec.com

Modifications reserved
Preliminary - specifications subject to change without notice
Document number: BST-BHI260AB-AN000-09
Revision_2.11