Software Documentation

BSX 4.x MCU Solution Porting Guide

Bosch Sensortec

BSX4.x: MCU Solution Porting Guide

Document revision	1.1
Document release date	Jan 19th , 2016
Document number	BST-BSX040-SD002-00
Technical reference code(s)	
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.

BSX 4.x MCU Solution
Porting Guide

General Description

In BSX4.x, there are five solutions defined with the BSX4.x library: MULTI_DOF, IMU, COMMPAS, M4G, and ACCELERATION. Each solution have its matching string, customer only need to change string & library to switch solutions. Customer should ask FAE for the library & string.

This application note introduces a basically example of sensor fusion on MCU. It provides the guide of how to integrate BSX4.x into a framework together with sensors, and also figures out the key points in the fusion of BSX4.x MCU solution, which could be easily ignored.

The following system requirements MUST be respected depending on the solution of BSX4.x Sensor Fusion Library.

Contents

1. A Glance of BSX4.X Integration

The integration of the library into a framework consists of two part, one part is selecting the set of features required for customer, get the corresponding library solution archive and configure strings, the other part is integrating the library solution archive and header files with the framework and build the executable of the bsx4 solution.

1.1 Supported virtual sensor output signals

There are many virtual sensors supported in bsx4 solutions, you could have an overview of all the virtual sensors according to the table below.Two types of signals from virtual sensor outputs are available from bsx4:

- time-continuous and value-continuous signals sampled in equidistant time intervals that are related to a configured physical sample rate such as acceleration sampled at 100Hz, angular rate sampled at 200 Hz or rotation at 50 Hz.

- time-discrete samples occurring at random time instants that are not related to a physical meaningful sample rate, e.g. detected activities or detected gestures.

Bsx4 provides two different kinds of outputs for equidistant sampled signals from virtual sensor outputs: non-wake-up outputs using a very simple re-sampling algorithm and wake-up outputs employing computational more demanding decimation algorithms.

Especially for Android, the non-wake-up virtual sensor outputs shall be preferred for high sample rates compared to wake-up virtual sensor outputs that are dedicated to lower sample rates.

Signal group	Signal name	Non-wake-up	Wake-up	Sampling
motion	raw acceleration (1)	yes	yes	equidistant
motion	equalized acceleration (2)	yes	yes	equidistant
motion	raw angular rate (1)	yes	yes	equidistant
motion	equalized angular rate (2)	yes	yes	equidistant
motion	raw magnetic field (1)	yes	yes	equidistant
motion	equalized magnetic field (2)	yes	yes	equidistant
motion	accelerometer offset	yes	**no**	random
motion	gyroscope offset	yes	**no**	random
motion	magnetometer offset	yes	**no**	random
motion fusion	rotation (NDOF, e-Compass, M4G)	yes	yes	equidistant

motion fusion	geomagnetic rotation	yes	yes	equidistant
motion fusion	game rotation	yes	yes	equidistant
motion fusion	linear acceleration	yes	yes	equidistant
motion fusion	gravity	yes	yes	equidistant
motion fusion	orientation (**deprecated**)	yes	yes	equidistant
motion interpretation	tilt gesture	yes	**no**	random
motion interpretation	glance gesture	yes	**no**	random
motion interpretation	pick-up gesture	yes	**no**	random
motion interpretation	wake gesture (by double-tap)	yes	**no**	random
motion interpretation	flip gesture	yes	**no**	random
motion interpretation	activity recognition	yes	**no**	random
motion interpretation	step detection	yes	**no**	random
motion interpretation	step counter	yes	yes	random (3)

(1) A signal from raw virtual sensor output provides the signal as received from physical sensor where a scaling to the standard or user-specific physical unit is applied.

(2) The signal from an equalized virtual sensor output provides the raw signal without known (estimated) non-linearities e.g. a bias/offset. Equalized signals are also known as compensated or corrected signals.

(3) Especially for Android, the virtual sensor output step counter is provided with a pseudo equidistant sampling.

Note: the orientation sensor providing Euler angles is marked deprecated by Google. The rotation vectors providing quaternions shall be used instead. Android provides mapping functions to convert a rotation vector to an orientation vector containing the euler angles.

In addition to above given output signals, bsx4 internal signal are available by debug outputs.

1.2 Bsx4 virtual sensor ODR supported

The output data rate of a sensor is the speed of the sensor to provide raw data, so the calling rate of the fusion library data processing for this sensor must not be higher than its ODR, generally be equal to its ODR. The bsx4 library ODR supported are 6.25HZ, 12.5HZ, 25HZ, 50HZ, 100HZ, 200 HZ.

2. Application Software Package

2.1 Macro Definition

According to different use cases, there are five Macro definitions in this reference solution as followed. Customer should choose macro both the solution and sensor.

2.1.1 LIBRARY_SOLUTION_MACRO

Solution Macro Control	Solution supported
INCLUDE_SOLUTION_NDOF	9Dof
INCLUDE_SOLUTION_IMU	IMU
INCLUDE_SOLUTION_COMPASS	Compass
INCLUDE_SOLUTION_M4G	M4G
INCLUDE_SOLUTION_ACCELERATION	accelerometer

2.1.2 SENSOR_MACRO

Sensor Macro Control	Sensor supported
INCLUDE_BMI160API	Bmi160 API
INCLUDE_BMM050API	Bmm055 API
INCLUDE_BMA2X2API	Bma2x2 API
INCLUDE_BMG160API	Bmg160 API
INCLUDE_BMI160API_BMM050_2TH_INTERFACE	Bmi160 + Bmm150(2th interface connected to MCU)
INCLUDE_BMM050API_DRICTIVE_INTERFACE	Bma2x2 + Bmm050(directive connected to MCU)

2.1.2 MACRO_TABLE

This table show the MACRO relationship between solution and demo shuttle board. The shuttle board is a kind of simple board which only have sensors, and could be easily connected to MCU board, user could easily switch sensors.

Solution Macro Control	Sensor Shuttle Board supported MACRO
INCLUDE_USECASE_NDOF	SHUTTLE_BOARD_BMI160_BMM050_2TH or SHUTTLE_BOARD_BMI160_BMM050_DIRECT
INCLUDE_USECASE_IMU	SHUTTLE_BOARD_BMI160 or SHUTTLE_BOARD_BMI055
INCLUDE_USECASE_COMPASS	SHUTTLE_BOARD_BMC156
INCLUDE_USECASE_M4G	SHUTTLE_BOARD_BMC156
INCLUDE_USECASE_ACCELERATION	

When one of the use case macro definitions is available in header file whose name is “define.h”, all the related configurations will be set correspondingly.

2.2 Folder Structure

As for the folder structures, all solutions are the same, however the device sensor APIs are optional.

```
|-- algo
|   |-- algo_adapter.c
|   |-- algo_adapter.h
|   |-- algo_bsx
|       |-- Inc
|       |   |--bsx_library.h
|       |   |--ladon_utils_defs.h
|       |   |--ladon_enums.h
|       |   |--bsx_android.h
|       |   |--bsx_constant.h
|       |   |--bsx_datatypes.h
|       |   |--bsx_user_def.h
|       |   |--bsx_activity_bit_identifier.h
|       |   |--bsx_module_identifier.h
|       |   |--bsx_physical_sensor_identifier.h
|       |   |--bsx_property_set_identifier.h
|       |   |--bsx_return_value_identifier.h
|       |   |--bsx_vector_index_identifier.h
|       |   |--bsx_virtual_sensor_identifier.h
|       |-- lib
|           |-- libalgobsx.<ext> (E.G.libalgobsx.a)
|
`-- device
    |
    |   |-------------------------------------------------------------------------------
    |   |-- bma2x2_api
    |   |   |-- bma2x2.c
    |   |   `-- bma2x2.h
    |   |-------------------------------------------------------------------------------
    |   |-- bmi160_api
    |   |   |-- bmi160.c
    |   |   `-- bmi160.h
    |   |-------------------------------------------------------------------------------
    |   |-- bmg160_api
    |   |   |-- bmg160.c
    |   |   `-- bmg160.h
    |   |-------------------------------------------------------------------------------
    |   `-- bmm050_api
    |       |-- bmm050.c
    |       `-- bmm050.h
    |-- define.h
    |-- sensorcontrol.c
    `-- sensorcontrol.h
```

2.1.1 Release Package Description

Component Name	Files	Description	
Algo	algo_adapter.h algo_adapter.c	The adapter interfaces of fusion library which help with integration	
	algo_bsx		
	inc	bsx_library.h bsx_android.h bsx_constant.h bsx_datatypes.h bsx_user_def.h bsx_activity_bit_identifier.h bsx_module_identifier.h bsx_physical_sensor_identifier.h bsx_property_set_identifier.h bsx_return_value_identifier.h bsx_vector_index_identifier.h bsx_virtual_sensor_identifier.h	The public APIs and MACRO of BSX4.x fusion library
	lib	**libalgobsx.<ext> (E.G.libalgobsx.a)**	The SLA/ESLA protected lib of BSX4.x fusion library
Device	sensorcontrol.c sensorcontrol.h	The sensor control functionalities of sensor device	
	define.h	The basic target platfrom related data type or MACRO definition	

2.1.2 Header Files description

Header file	Description
bsx_constant.h	Values used to scale I/O signals
bsx_datatypes.h	Data types used by interface functions
bsx_return_value_identifier.h	Definition of codes return by interfaces functions
bsx_library.h	Declaration of interface functions
bsx_android.h	Mapping from Android sensor identifiers to BSX virtual sensor identifiers
bsx_physical_sensor_identifier.h	Identifiers for physical input sensor signals
bsx_virtual_sensor_identifier.h	Identifiers for virtual output sensor signals
bsx_property_set_identifier.h	Identifiers for sets of configuration values and state values
bsx_activity_bit_identifier.h	Helpers to decode the activity signal
bsx_vector_index_identifier.h	Helpers to decode the signals rotation (quaternion), orientation (Euler angle) and motion in space (x, y, z)
bsx_user_def.h	User definable support, e.g. assertion handler, tracing support

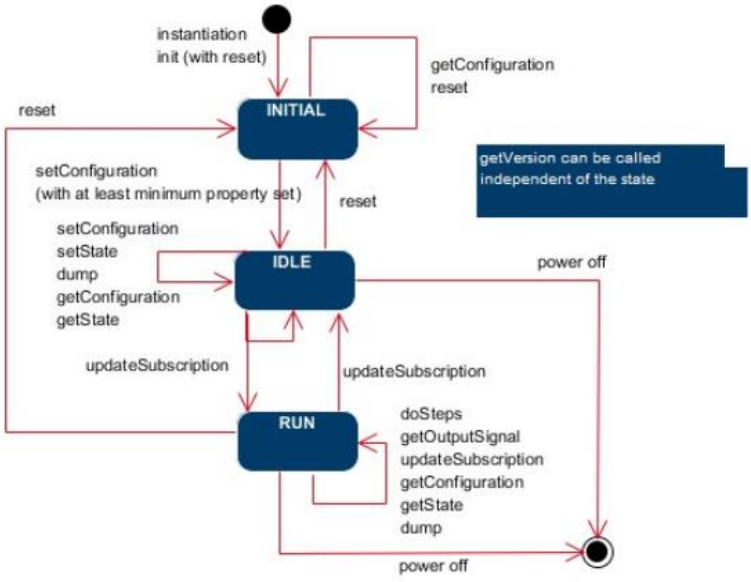
2.3 Software Architecture Overview

2.3.1 BSX4 Operation State

The library owns an operation state machine to have a robust user interface against unexpected use and avoid unpredictable behavior.

The states of the library are idle and run with an additional intermediate state initial between start and idle to guarantee a correctly configured library and ensure the intended quality of virtual sensor output signals.

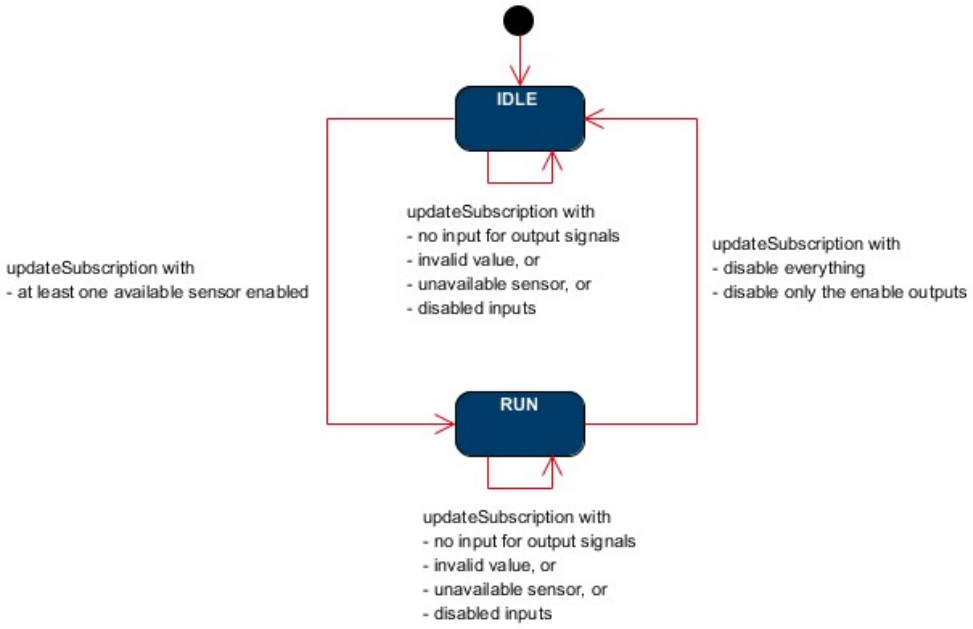
The following UML state diagram depicts the state machine and the callable functions as well as the resulting transitions among each state.



Operation state machine

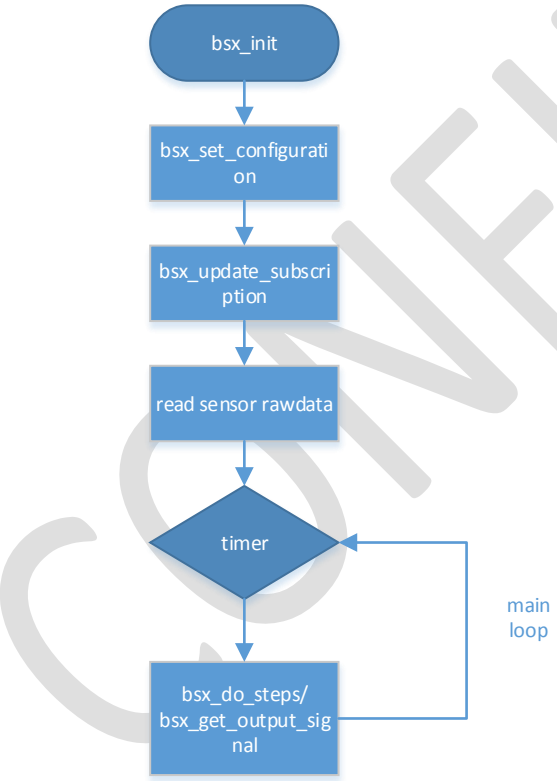
2.3.1 BSX4 Running and Idle State

The transition among the states IDLE and RUN performed by the update of the subscription to library outputs plays an important role and is therefore detailed in the additionally provided UML sub-state diagram.

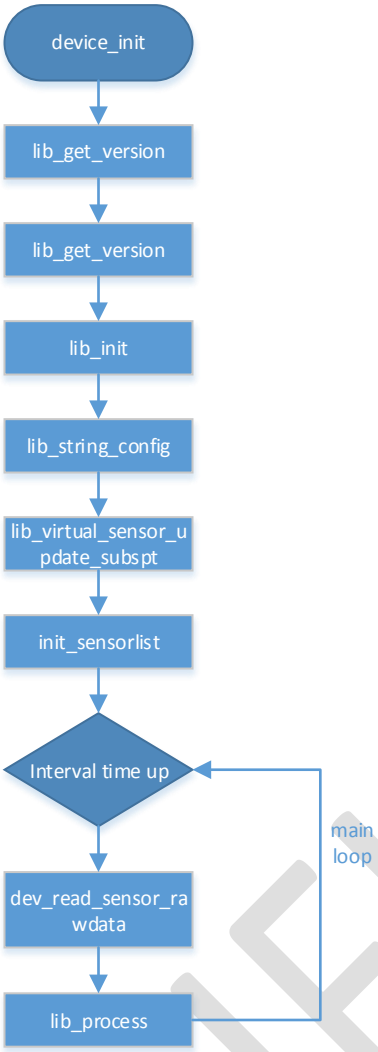


Operation state machine: IDLE-RUN-transition

2.3.3 The Basic Flow of BSX4



2.3.4 BSX4 Porting Flow According to The Basic Flow



Note: the time interval should be set according to the sensor ODR.

2.3.5 Units for input data to bsx4 library

acc, gyro, mag are all 32bit signed integer, timestamp is 64 bit signed integer.

- acc sensor data (x, y, z) : mg
- mag sensor data (x, y, z) : 0.1uT
- gyro sensor data (x, y, z) : 0.0011rps
- timestamps : nanoseconds (64 bits)

2.3.6 Bsx4 library output data

The following table details the maximum available virtual output sensor signals from the library. For each signal, we give data type, dimensionality and the content description with the unit of measurement and whether a status is provided along with the signal. The actual unit provided by the library can be customized according to customer needs.

All virtual sensor name are mapped to virtual sensor ID in headfile bsx_android.h.

Output signal name	data type	dim	content	Unit of measurement	status
raw acceleration	single	3	3-axis	m/S^2	no
equalized acceleration	single	4	3-axis	m/S^2	yes
raw angular rate	single	3	3-axis	rad/s	no
equalized angular rate	single	4	3-axis	rad/s	yes
raw magnetic field	single	3	3-axis	uT	no
equalized magnetic field	single	4	3-axis	uT	yes
accelerometer offset	single	4	3-axis	m/S^2	yes
gyroscope offset	single	4	3-axis	rad/s	yes
magnetometer offset	single	4	3-axis	uT	yes
rotation (NDOF, e-Compass, M4G)	single	5	quaternion		yes
geomagnetic rotation	single	5	quaternion		yes
game rotation	single	5	quaternion		yes
linear acceleration	single	4	3-axis	m/S^2	yes
gravity	single	4	3-axis	m/S^2	yes
orientation (**deprecated**)	single	5	Euler	rad	yes
tilt gesture	uint8	1	boolean		no
glance gesture	uint8	1	boolean		no
pick-up gesture	uint8	1	boolean		no
flip gesture	uint8	1	boolean		no
wake gesture (by double-tap)	uint8	1	boolean		no
activity	uint32	1	bit field		no

recognition					
step detection	uint8	1	boolean		no
step counter	uint64	1	count		no

2.3.7 Library calibration and retrieval

After finished the integration. Customer calibrate the library, then could store the calibration data into the flash, and retrieval the data next time.

Please refer to the document BSX_library_documentation.pdf

1.4.2.4 Retrieval of the BSX configuration
1.4.2.5 Update of the BSX state
1.4.2.6 Retrieval of the library state

3. Key APIs Description of Adapter

This chapter introduces APIs from the key adapter layer which support all the virtual sensors.

3.1 Algorithm Adapter

The algorithm adapter layer shows an example of how to run the main functionalities by utilizing APIs of fusion library and devices hardware setting.

3.1.1 Bsx4 library Initialization

bsx_return_t lib_init(void)	
Description	initialize the bsx4 algorithm application
Parameters	NO
Return Type	Error Code. Zero, successful
Calling Frequency	Called when required(power on, only once)

3.1.2 Bsx4 string configure

bsx_return_t lib_string_config(void)	
Description	In the initialization flow, configure the bsx4 library according to the strings, which provided by FAE according to the customer demo board.
Parameters	void
Return Type	Error Code. Zero, successful

3.1.3 Virtual sensor update subscription in the initialization flow

bsx_return_t lib_virtual_sensor_update_subspt(void)

Description	Subscribe the virtual sensor in the initialization flow, use only once
Parameters	void
Return Type	Error Code. Zero, successful
Usage Guide	1. Call the API when subscribe the virtual sensors in the initialization flow. Note that if you want to change the virtual sensors (add new virtual sensors, delete virtual sensors or change virtual sensor ODR) in the initialization flow, just modify the virtual sensor list in this API. 2. The physical sensor odr should be set according to the bsx_update_subscription,physical_sensor_config output parameter.

3.1.4 Virtual sensor update subscription in running time

static __inline bsx_return_t lib_virtual_sensor_update_config(void)

Description	Update the Subscription of the virtual sensor in running time
Parameters	void
Return Type	Error Code. zero, successful
Usage Guide	Update the virtual sensor configuration, when virtual sensor ODR or list need to be changed in the running time.

3.1.5 Getting raw data from sensors

static __inline bsx_s8_t device_read_sensor_rawdata(sensor_output_rawdate_t *p_sensor_rawdate)

Description	Get the raw data from sensors
Parameters	sensor_output_rawdate_t *p_sensor_rawdate (unit changed)
Return Type	Error Code. zero, successful
Usage Guide	Get the raw data from sensors. read accelerometer raw data, then change it to bsx4 input unit first(mg). read gyroscope raw data, then change it to bsx4 input unit first(0.011rps). read mag raw data, then change it to bsx4 input unit first(0.1uT). Note: the raw data unit of various sensor firms may be different, customer should check the sensor datasheet carefully.

3.1.6 Algorithm Main Process

static __inline bsx_return_t lib_process(sensor_output_rawdate_t *p_sensor_output_rawdate)

Description	The main process of algorithm, input the rawdata into the library
Parameters	The rawdata(unit changed)
Return Type	Error Code, Zero when successful, positive value for information and warnings, otherwise a negative value as error code. the head file bsx_return_value_identifier.h define the error code.
Usage Guide	Call this API to finish the following items: 1. Input the rawdata, then get the bsx4 output data.

	2. The output data flow is that: according to virtual sensor ID, you could get this virtual sensor dimensionality one by one. For instance, the accelerometer is a virtual sensor, input the ACCELEROMETER virtual sensor ID and then the library output the 4 accelerometer dimensionality data one by one. 3. The library timestamp should be the real time stamp (a kind of fusion of sensor time and MCU timer). The artificial timestamp could be only be used in the porting test, which will be replaced by real time stamp later. To guaranty the accuracy of real timestamp, the way may be different in different system. According to android M, the bsx4 accuracy of time stamp reach to nanosecond.

3.2 Device Driver Adapter

These APIs are examples for implementing the functionalities which served for fusion library. They are the adapter demo for different hardware sensors.

It includes Accelerator, Magnetometer and Gyroscope sensors. Platform compatible.

Function name	Description
dev_acc_read_xyzdata()	Get the xyz data from acceleration sensor
dev_acc_get_grange()	Get the acceleration range of sensor
dev_acc_set_grange()	Set the acceleration range of sensor
dev_acc_get_opmode()	Get the operation mode of acceleration sensor
dev_acc_set_opmode()	Set the operation mode of acceleration sensor
dev_acc_get_bandwidth()	Get the bandwidth setting of acceleration sensor
dev_acc_set_bandwidth()	Set the bandwidth setting of acceleration sensor
dev_acc_get_register()	Get the register values of acceleration sensor
dev_acc_set_register()	Set the register values of acceleration sensor
dev_mag_set_opmode()	Set the operation mode of magnetic sensor
dev_mag_get_opmode()	Get the operation mode of magnetic sensor
dev_mag_read_xyzdata()	Get the xyz data from magnetic sensor
dev_gyro_read_xyzdata()	Get the xyz data from gyroscope sensor
dev_gyro_set_opmode()	Set the operation mode of gyroscope sensor
dev_gyro_set_range()	Set the range of gyroscope sensor
dev_gyro_set_bandwidth()	Set the bandwidth setting of gyroscope sensor
delay()	The delay function which is platform dependent. millisecond
dev_init_sensorlist()	Initial the possible hardware sensors and register the bus read, write and delay functionalities.
dev_enable()	Enable or Disable the use case related hardware sensors.(optional)

4. Parameter adjustment

4.1 SOFTWARE IRON CALIBRATION

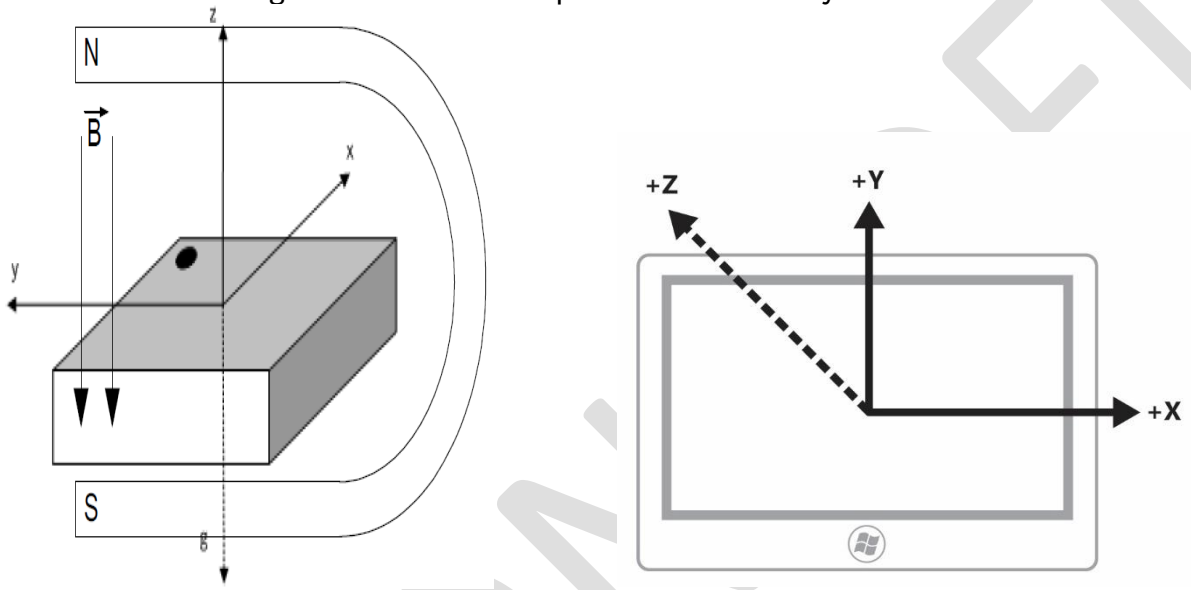
SIC is a kind of calibration which arm to calibrate magnetic sensor in device, customer ask FAE for the device test result and get the Soft Iron Correction Matrix.

4.2 CALIBRATION SET STATE AND GET STATE

Set state and Get state function is a kind of calibration which could get the magnetic field strength. After finished the integration. Customer calibrate the library, then could store the calibration data into the flash, and retrieval the data next time. Customer could refer to the demo source code.

4.2 AXIS REMAPPING

Unify hardware components' axis in the same coordinate according to the placement and target platform definition. The following diagram and corresponding codes show how to remap a sensor's axis to the target's axis. And then input to the bsx4 library.



$X_{target}=-Y_{sensor}$; $Y_{target}=X_{sensor}$; $Z_{target}=Z_{sensor}$;

Axis Remapping

for example, for axis remapping in the figure above,remapping codes are as follows.

```
U8 remap_bmi160_read_acc_xyz (struct bmi160acc_t *acc_p)
{

        bmi160_read_acc_xyz (acc_p);    //get acceleration raw data from sensor
        acc_p->x = acc_p->x *(-1);      //assigning accelerometer x values to x remapping
        acc_p->y = acc_p->x;            //assigning accelerometer x values to y remapping
        acc_p->z = acc_p->z;            //assigning accelerometer x values to z  remapping
        return OK;
```

```
}
```

Note: if the accelerometer, gyroscope, magnetometer are independently layout in sensor design, therefore the remapping will also independently.

5. FAE Tool

FAE & customer could configure the string with FAE tool ConfigurationSelector.exe, for more information, please refer to document ConfigurationSelectorUserManual.

6. Critical Compute Resouce Statistics

GCC options (MDOF,IR28.1)	**ROM size**	**RAM size**
-c -std=c99 -mcpu=cortex-m3 -mthumb -O2 **-fno-common(option2)**	108.35+1.87=110.22KB	1.87+8.27=10.14KB

For compiler arm-none-eabi-gcc.exe:
ROM size= text/code + data
RAM size= data + bss

7. Legal Disclaimer

a) 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.

b) 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.

c) 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.

8. Document History and Modifications

Rev. No	Chapter	Description of modification/changes	Date
1.0	All	Document creation	2016-01-19
1.1	1,2	Update virtual sensor list	2016-03-05