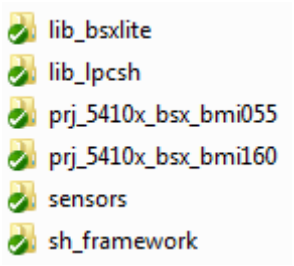
SensorHub Project Structure and Procedures

The SensorHub project implements a framework for reading and processing motion sensors at a low power based on requests from a host over I2C/SPI, it supports integration of a fusion library which can convert the raw motion sensor data into meaning data that can be used by the host. The framework implements the batching requirements suggested by Android plus a logging service for system debugging, the framework also implements a breathing LED feature.
The project is split into two parts:

- Proprietary library for host command handler + logging service and breathing LED driver.
- Open sourced sensorhub framework example using the Bosch BSX Lite Motion library.

The SensorHub project structure is as shown below:



The "*lib_bsxlite*" folder contains the BSX Lite motion library and its header files, the "*src*" folder contains the source code which implements an algorithm wrapper for BSX library. The "*lib_lpcsh*" folder contains the proprietary framework library and the necessary header files. The "*sh_framework*" contains the open sourced sensorhub framework, "*sensors"* contains vendor provided sensor drivers or NXP sensor drivers. The project files are in the *prj_5410x_bsx_bmi055 and prj_5410x_bsx_bmi160* folder with each supported tool chain projects in a separate folder. The BMI055 project has the sensorhubconfig.h configured to use the BMI055 sensor configuration whereas the BMI160 project configures the BMI160 gyro sensor. The tool specific project folders contains the sensor hub workspace and only has a chip project and a sensorhub project. **Note that there is no board specific project**. The board related setup data is in a configuration file, during initialization this data is read and the board configuration is set accordingly. Note that the chip source code should be at the same folder level as the sensorhub.

Build Procedures

The sensorhub project uses configuration files to configure the project for different boards or different features/software configurations. The configuration files are in "sh_framework\inc\config".
All the configuration files should have exactly the same number of configuration parameters, if for any board or configuration, a particular configuration parameter is not valid then just #undef the parameter. If a new parameter is added for a particular configuration file then that parameter should be added to all the configuration files with a #undef for that parameter.
The sensorhubconfig.h is a header file present in "sh_framework\inc\" which chooses different configuration files based on project macro. Currently there are two projects supported PRJ_5410X_BSX_BMI055 and PRJ_5410X_BSX_BMI160. These macros are defined in the project files.
So to add a new project create a new project folder and the define a project macro in the project options and the macro in the sensorhubconfig.h and inside the #ifdef choose the right configuration file.

Configuration Files

The configuration file is a compile time configuration which configures the sensor hub framework to work on a specific board with different sensors and different host interfaces i.e. I2C/SPI. It also configures what features are enabled and how they are configured for example Breathing LED example.

A typical sensor configuration parameters is as shown below for an accelerometer.

```
/* Accelerometer */
#define CFG_HW_SENSOR_ACCEL                     ACCEL_BMA2X2
#undef CFG_HW_SENSOR_ACCEL2

#define CFG_HW_ACCEL_INT_PORT                   0x0
#define CFG_HW_ACCEL_INT_PIN                    18
#define CFG_HW_ACCEL_PINT_SEL                   PININTSELECT0
#define CFG_HW_ACCEL_PINT_CH                    PININTCH0
#define CFG_HW_ACCEL_PINT_IRQn                  PIN_INT0_IRQn
#define CFG_HW_ACCEL_WAKE                       SYSCON_STARTER_PINT0
#define CFG_HW_ACCEL_IRQHandler                 PIN_INT0_IRQHandler
#define CFG_HW_ACCEL_INT2_PORT                  0x0
#define CFG_HW_ACCEL_INT2_PIN                   7
```

The configuration parameter XXX_SENSOR_ACCEL defines the sensor i.e. used for the accelerometer. In the example above it is defined to ACCEL_BMA2X2. In sensors.c source file, the table g_phySensors contains the list of sensors used by the sensorhub framework, currently the framework supports Accel, Gyro, Mag, pressure, proximity and ambient light sensors only. Each sensor is a macro named as G_XXX_CTRL. Each of this macro needs to be defined to a pointer of type PhysicalSensor_t and PhysicalSensor_t will provide all the handles to initialize, activate and read the sensor. Now based on the configuration parameter i.e. XXX_SENSOR_ACCEL the G_ACCEL_CTRL is defined to different pointers in sensors.c as shown below.

```
    #if (CFG_HW_SENSOR_ACCEL == ACCEL_BMA2X2)
extern PhysicalSensor_t g_bma2x2Accel;
        #define G_ACCEL_CTRL    &g_bma2x2Accel
    #elif (CFG_HW_SENSOR_ACCEL == UNUSED)
        #undef CFG_HW_SENSOR_ACCEL
        #define G_ACCEL_CTRL    NULL
    #else
        #warning "CFG_HW_SENSOR_ACCEL not defined or not recognized"
        #define G_ACCEL_CTRL    NULL
    #endif
```

To implement a new sensor driver the driver handles should be packed into the PhysicalSensor_t structure and a new sensor needs to be defined in sensor_defines.h and the G_XXX_CTRL macro needs to defined in the above code in sensors.c.

Now the sensor data interrupt lines are seen as pin interrupts for the microcontroller and the configuration configures the port and pin for the interrupt using the parameters XXX_INT_PORT XXX_INT_PIN. The XXX_PINT_XX configures the pin interrupt channel and the IRQ channel number. The XXX_IRQHandler configures the IRQ handler that will be used for the sensor pin interrupt. Since the pin

interrupts should be able to wake up the microcontroller from power down mode, the XXX_WAKE parameter configures the wakeup bit that needs to be enabled in SysCon.

The Host interface configuration for I2C is as shown below:

```
/** Host interface (I2C slave) defines */
#define CFG_FW_HOSTIF_I2C
#define CFG_FW_HOSTIF_I2C_CLOCK_DIV             2
#define CFG_FW_HOSTIF_I2C_ADDR                  0x68
#define CFG_FW_HOSTIF_I2C_CH                    LPC_I2C2
#define CFG_FW_HOSTIF_I2C_IRQ_CH                I2C2_IRQn
#define CFG_FW_HOSTIF_I2C_IRQ_HDL               I2C2_IRQHandler
#define CFG_FW_HOSTIF_I2C_WAKE                  SYSCON_STARTER_I2C2
#define CFG_FW_HOSTIF_I2C_DMAID                 DMAREQ_I2C2_SLAVE
```

To enable I2C as the host interface XXX_I2C needs to be defined, to use SPI then XXX_SPI needs to be defined. Only one of these definitions should be active for any build. The XXX_I2C_ADDR configures the slave address for sensor hub. The XXX_XXX_CH configures the I2C/SPI channel i.e. used for host communication, XXX_XXX_IRQ_CH configures the IRQ channel number, XXX_XXX_IRQ_HDL configures the IRQ handler for I2C/SPI, XXX_XXX_WAKE configures the bit to enable for low power wakeup, and XXX_XXX_DMAID configures the DMA channel used for I2C. SPI needs two DMA channels to be configured one for TX and the other for RX.

A typical SPI host interface configuration is as shown below:

```
/** Host interface (SPI slave) defines */
#undef CFG_FW_HOSTIF_SPI
#define CFG_FW_HOSTIF_SPI_BITRATE               (1000000)
#define CFG_FW_HOSTIF_SPI_CH                    LPC_SPI0
#define CFG_FW_HOSTIF_SPI_IRQ_CH                SPI0_IRQn
#define CFG_FW_HOSTIF_SPI_IRQ_HDL               SPI0_IRQHandler
#define CFG_FW_HOSTIF_SPI_WAKE                  SYSCON_STARTER_SPI0
#define CFG_FW_HOSTIF_SPI_DMA_TXCH              DMAREQ_SPI0_TX
#define CFG_FW_HOSTIF_SPI_DMA_RXCH              DMAREQ_SPI0_RX
```

The sensors I2C communication is configured as follows:

```
/** Sensor Interface configuration */
#define CFG_FW_SENSOR_I2C_CLKDIV                2
#define CFG_FW_SENSOR_I2C_CLKSPEED              400000
#define CFG_FW_SENSOR_I2C_CH                    LPC_I2C0
#undef  CFG_FW_SENSOR_I2C_IRQ_BASED
#undef CFG_FW_SENSOR_I2C_IRQ_CH
#undef CFG_FW_SENSOR_I2C_IRQ_HDL
#undef  CFG_FW_SENSOR_I2C_DMA_BASED
#undef CFG_FW_SENSOR_I2C_DMAID
#undef CFG_FW_SENSOR_I2C_DMA_BYTELIM
#define CFG_FW_SENSOR_BUS_BUFFER                32
```

The sensor I2C communication is by default configured to work in polled mode. The configuration parameters configure the clock divider, the clock speed and I2C channel number.

Other than this the configuration file configures the pin muxing for the board in the pinmuxing table which configures the function and pull up/pull down resistors for each pin. Different features can be enabled/disabled in the configuration file. For example, to disable breathing LED feature just change #define CFG_FEAT_BREATHING_LED to #undef CFG_FEAT_BREATHING_LED.