SN10-2x

User's Guide

Product Line	
Document Name	
Version	v 0.2.5
Date	
DocID	
Status	

Approved by	Reviewed by	Issued by
		Joe Hou

Revision History

Revision	Released Date	Comments/Remark	Author
Draft	2016/11/24	Initial release	
V 0.1	2016/12/06	Added section 3 " Use of the pins…", references to TI documents and web sites and some minor changes	
V0.1.1	2016/12/07	Corrected some inconsistencies between the document and the SDK	
V 0.1.2	2016/12/08	Corrected AT command codes for changing RCZ. Removed AT command for getting device version	
V 0.1.3	2017/01/20	Added high level functions sendData() and sendBitStatus()	
V 0.1.4	2017/03/02	Changed default interval for location reporting to 1 minute. Customers may change it in the source code.	
V 0.1.5	2017/03/03	Removed command code 12, 14 and 15.	
V 0.2.0	2017/04/26	Updated for 2640 R2	
V 0.2.1	2017/06/13	Added a step to check the zone and the key type to the build process.	
V 0.2.2	2017/07/19	Updated the description of Appendix B Added Appendix C for hardware setup.	
V 0.2.3	2017/08/08	Added screen shots for the SDK setup procedure	
V 0.2.4	2017/08/18	Updated PIN assignment and PIN name	
V 0.2.5	2017/09/28	Separated AT command interpreter as a stand alone application from the low power tracker application to free up one additional GPIO (IO_EXP_P5) for the application.	

TRANSLATE, REVERSE ENGINEER, CREATE DERIVATIVE WORKS; DISASSEMBLE OR DECOMPILE THE PRODUCT OR OTHERWISE USE THE PRODUCT EXCEPT AS SPECIFICALLY AUTHORIZED. THE RECIPIENT UNDERTAKES FOR AN UNLIMITED PERIOD OF TIME TO OBSERVE CONFIDENTIALLITY REGARDING ANY INFORMATION AND DATA PROVIDED TO THEM IN THE CONTEXT OF THE DELIVERY OF THE PRODUCT. THIS GENERAL NOTE
SHALL BE GOVERNED AND CONSTRUED ACCORDING TO TAIWAN LAW.

TABLE OF CONTENT

1. Introduction

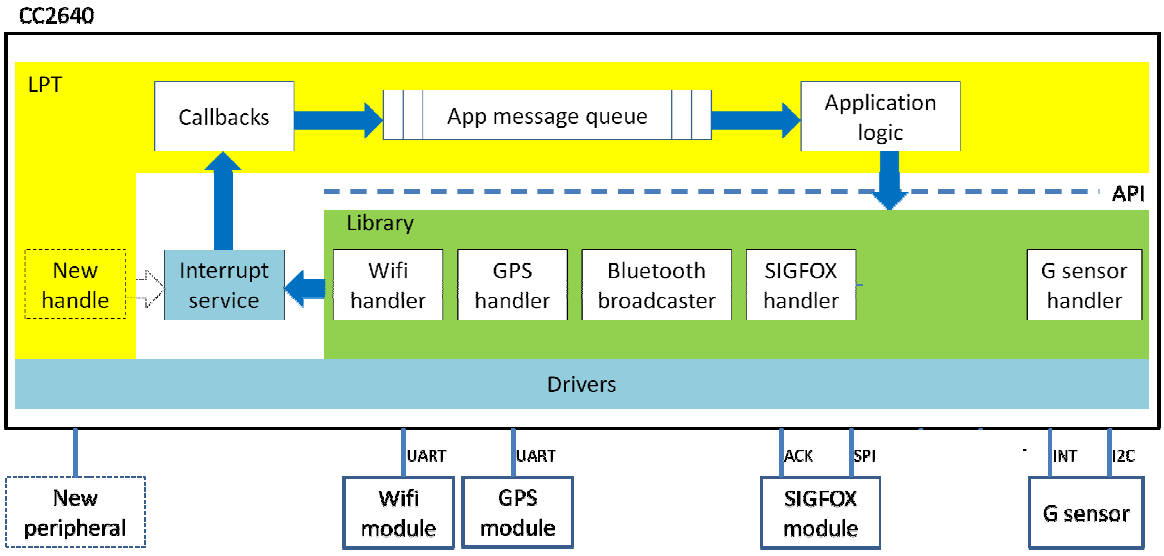
SN10 is a low power combo-module with wifi module, GPS module, BLE module and SIGFOX module and a acceleration sensor built in. The SDK consists of the following components:

- TI's Code Composer Studio
- TI's real-time operating system
- TI's driver library and BLE stack
- Innocomm's library for SIGOFX module interface
- Low power tracker application with full source codes

Section 2 gives an overview of the library and the sample application. Section 3 describes the SDK installation procedure, and section 4 describes the APIs in detail. Please visit the following websites for documents of the Code Composer Studio IDE and TIRTOS.

http://www.ti.com/tool/ccstudio
http://www.ti.com/tool/ti-rtos

2. Library and sample application overview



The library consists of the following components. Each exposes a set of APIs for the applications' access. Source codes are available for all but the SIGFOX handler.

- Wifi handler interfaces to the wifi module via UART to scan the surrounding for Wifi APs.
- GPS handler interfaces to the GPS module via UART to get the fix of the GPS location.
- SIGFOX handler interfaces to the SIGFOX module via SPI to configure the module, sends and receives payload from the network, and to perform test mode procedures.
- G sensor handler interfaces to the G sensor via I2C to configure the sensor and register call back for G sensor interrupt events.
- Bluetooth broadcaster configures the content and the interval of the beacon and broadcasts

beacons periodically.

The low power tracker sample application will scan surrounding wifi APs and fix GPS location every 60 minute or upon G sensor interrupt,. It will also broadcast BLE beacons every 100 ms. The application consists of the following components:

- A collection of the call back functions that convert timer and sensor interrupts and BT protocol events to application events,
- An application message queue, and
- The application logic that initiates the module , and processes the application events and calls the module APIs to broadcast the beacon and to collect the location data and send it to the SIGFOX network.

The users may extend the functionality of the application by adding new peripherals or sensors to the device and adding new events, new call backs and new handling functions for them.

The following high level application flow demonstrates how the application and the library interact to serve an interrupt event .

1. At start up, the application calls the library initiation function to initiate the module and pass the call back functions for hourly timer interrupt, G sensor interrupt, AT interpreter 5 ms timer interrupt and BT protocol event to the library.
2. The library initializes the wifi, GPS, SIGFOX, G sensor, BT broadcast and the drivers, and registers the call back functions to the interrupt service of the OS. The application is now ready to receive and process events.
3. When an interrupt(e.g. G sensor interrupt) happens, the interrupt service calls the corresponding call back function(e.g. G sensor call back).
4. The call back function converts the interrupt event to the application event and en-queue it to the application message queue for processing.
5. The interrupt handling function(e.g. G sensor interrupt handling function) de-queues the event and calls the library to process the event (e.g. scan the wifi AP, fix the GPS location and send the result to the SIGFOX network).
6. Repeat 3 ~ 5 when new interrupts (e.g. hourly timer interrupts, G sensor interrupts) happen.

3. Pin assignments

In addition to the existing IO pins of the MCU, an I2C based IO expander is added to the module to add additional 8 pins (IO_EXP_P0 ~ IO_EX_P7) to the module. The pins used by the library are listed in the following table.

function	SW Pin name	Pin define
IO-expander (I2C)	DIO_9	SDA
	DIO_10	SCL
	DIO_8	INT
G-sensor (I2C)	DIO_9	SDA
	DIO_10	SCL
	IO_EXP_P6	EINT1
SIGFOX (SPI)	DIO_7	AK
	IO_EXP_P3	RESET
	IO_EXP_P4	CS
	DIO_12	SDI
	DIO_11	SDO
	DIO_13	SCK
GPS (Board_UART0)	IO_EXP_P2	RESET
	DIO_2	UART RX
WIFI (Board_UART1)	IO_EXP_P1	LDO_EN
	DIO_1	UART TX
	DIO_0	UART RX
Trace (SW UART)	DIO_3	LOG_TX
	DIO_4	LOG_RX
DC enable	IO_EXP_P0	DC_EN

Pins free for the application are listed in the following table.

Pin name	SW pin name	Pin define
G_SDA2	DIO_9	I2C SDA
G_SCL2	DIO_10	I2C SCL
JTAG_TDI	DIO_6	JTAG TDI or UART TX (Board_UART2) or digital GPIO
JTAG_TDO	DIO_5	JTAG TDO or UART RX (Board_UART2) or digital GPIO
GPIO2	DIO_14	Analog
GPIO3	IO_EXP_P5	Digital
GPIO4	IO_EXP_P7	Digital

If your application requires more IO's, GPIO5(DIO_4) and (GPIO6(DIO_3) can be made available by disabling the trace function. However, since the AT command interpreter application also uses these two pins, it might not work properly depending on how these pins are used in your application. If it doesn't work, you'll have to write a simple application to send the command sequences to the Sigfox module required for certification test of your device. Please contact fae@innocomm.com for support.

4. Setting up the SDK

You'll see the following files and folder after unzipping the SDK package, for compatibility please do not replace these files with newer or older versions available on the web:

- The CCS installation file < ccs_setup_7.1.0.00016.exe >
- The BLE stack and TI RTOS installation file < simplelink_cc2640r2_sdk_1_30_00_25.exe >
- SN10_CONFIG2_SDK folder that contains the SN10 library , the sample project and the source code for the sample application.

Please follow the following steps to complete the installation. For details of the installation procedures please refer to section 2.6.3 of the CC2640 Developer's Guide (http://www.ti.com/lit/ug/swru393d/swru393d.pdf)

Install the CCS IDE

Execute ccs_setup_7.1.0.00016.exe and follow the flow to install the CCS IDE.

1. Do not change the default installation location (c:\ti)
2. In Processor Support, select "SimpleLink CC13xx and CC26xx Wireless MCUs".
3. In Select Debug Probes, select "TI XDS Debug Probe Support".

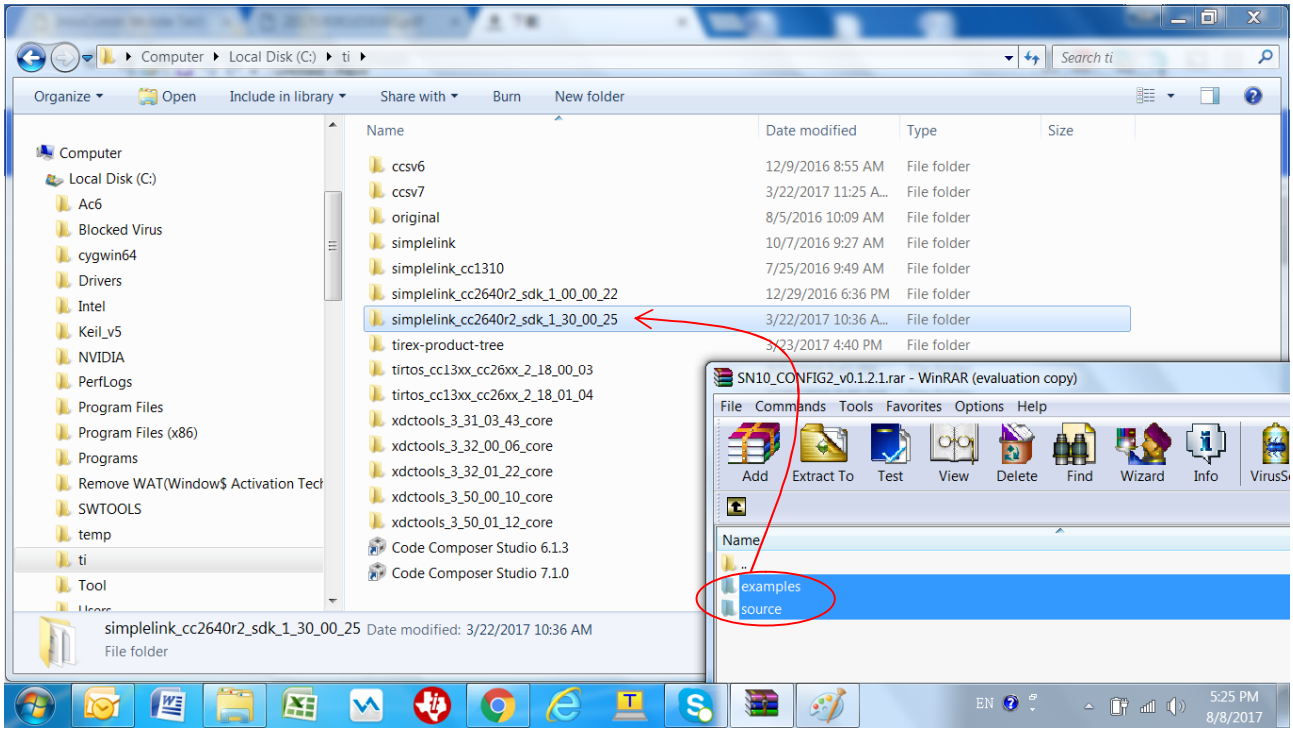
Install the BLE stack and TI RTOS

Execute simplelink_cc2640r2_sdk_1_30_00_25.exe and follow the flow to install the BLE stack and the TI RTOS.

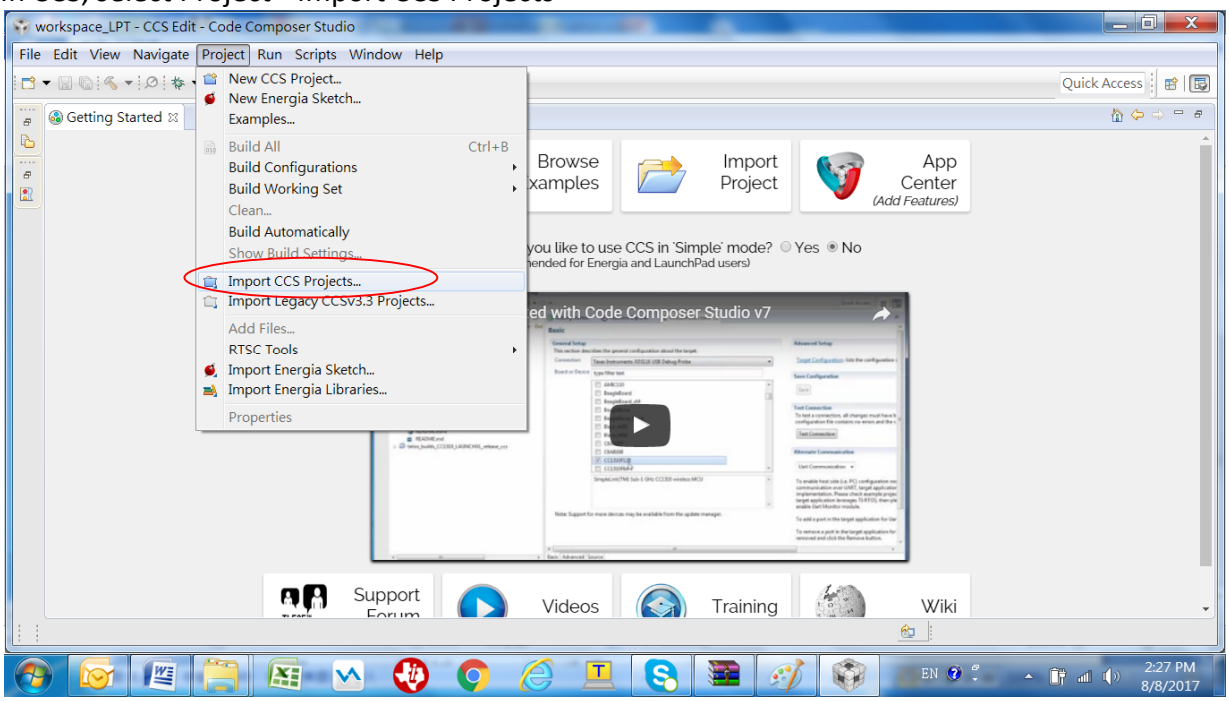
- Do not change the default installation location of the stack (c:\ti\ simplelink_cc2640r2_sdk_1_30_00_25) and the RTOS (c:\ti).

Install innocomm's library and sample application

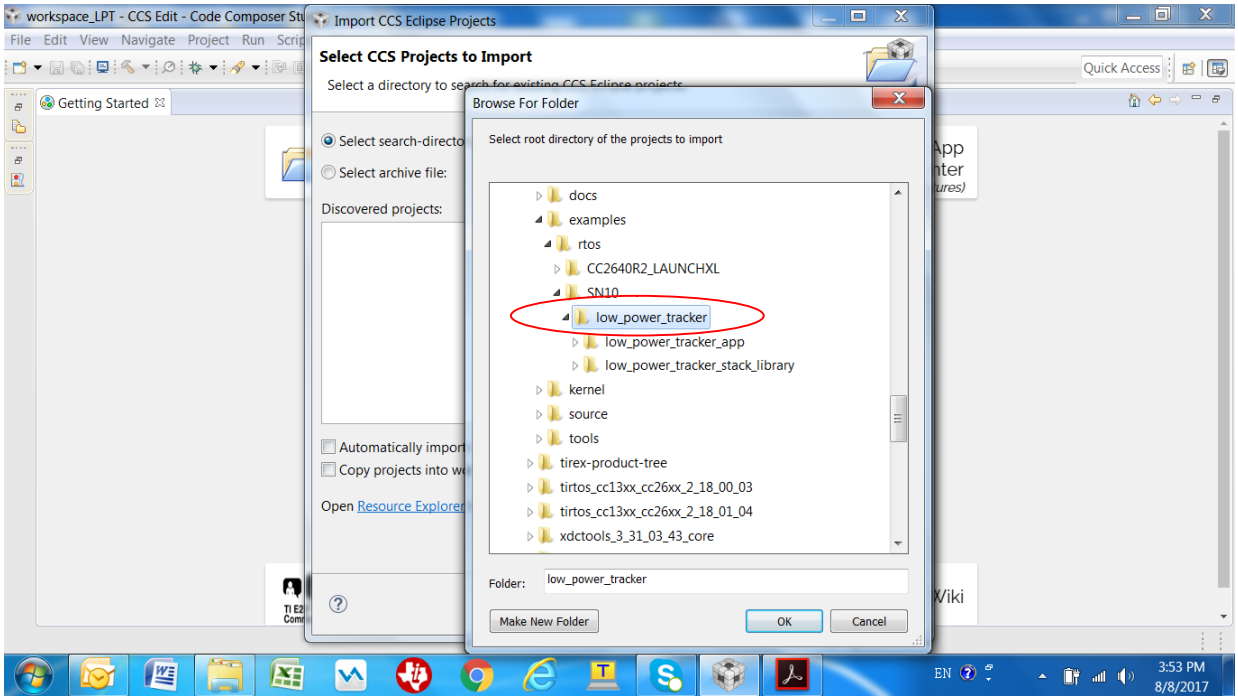
1. Open the SN10_CONFIG2_SDK folder
2. Copy the examples folder and the source folder to C:\ti\ simplelink_cc2640r2_sdk_1_30_00_25\.
   Note: There are examples and source folders under the simplelink_cc2640r2_sdk_1_30_00_25 folder. This step will copy contents in the examples folder and the source folder of the SDK in to the examples folder and the source folder of the simplelink_cc2640r2_sdk_1_30_00_25 folder respectively. Please overwrite the existing files when prompted.



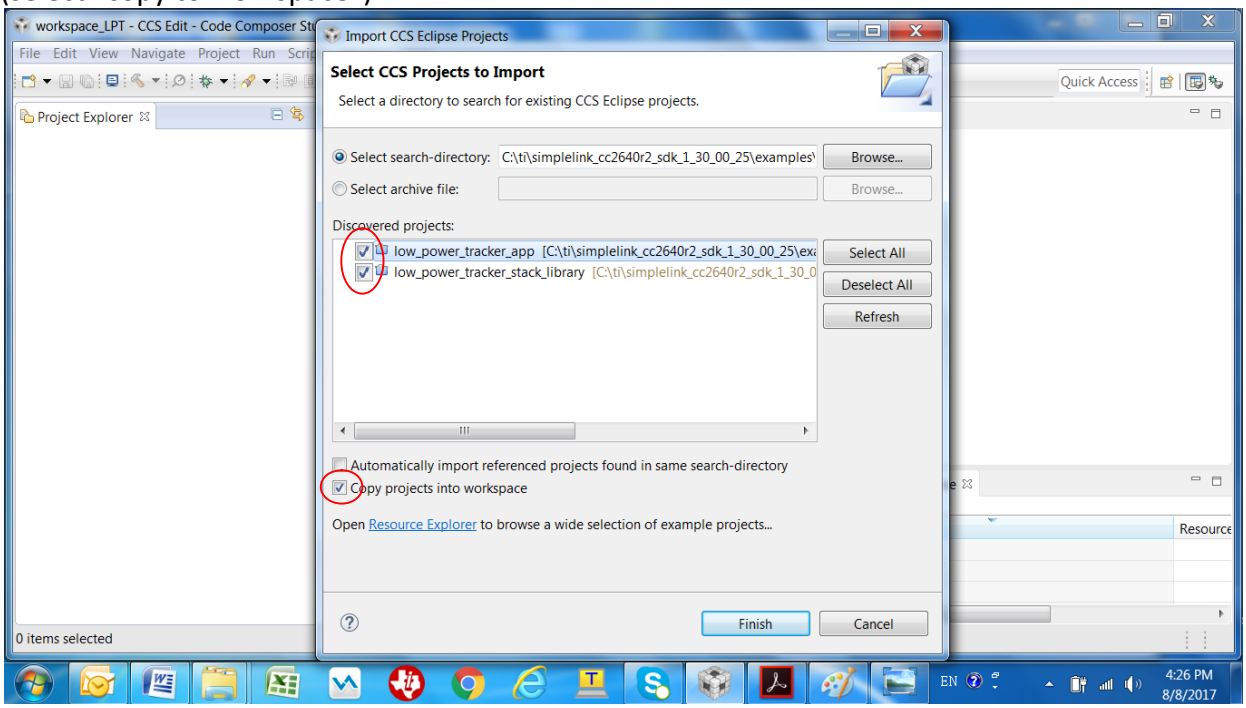
3. In CCS, select Project->Import CCS Projects



4. Click Browse… button and navigate to C:\ti\simplelink_cc2640r2_sdk_1_30_00_25\examples\rtos\SN10\low_power_tracker



5. Click the OK button and select the two discovered projects: low_power_tracker_app and low_power_tracker_stack_library and click Finish button to import.
(select “copy to workspace”)

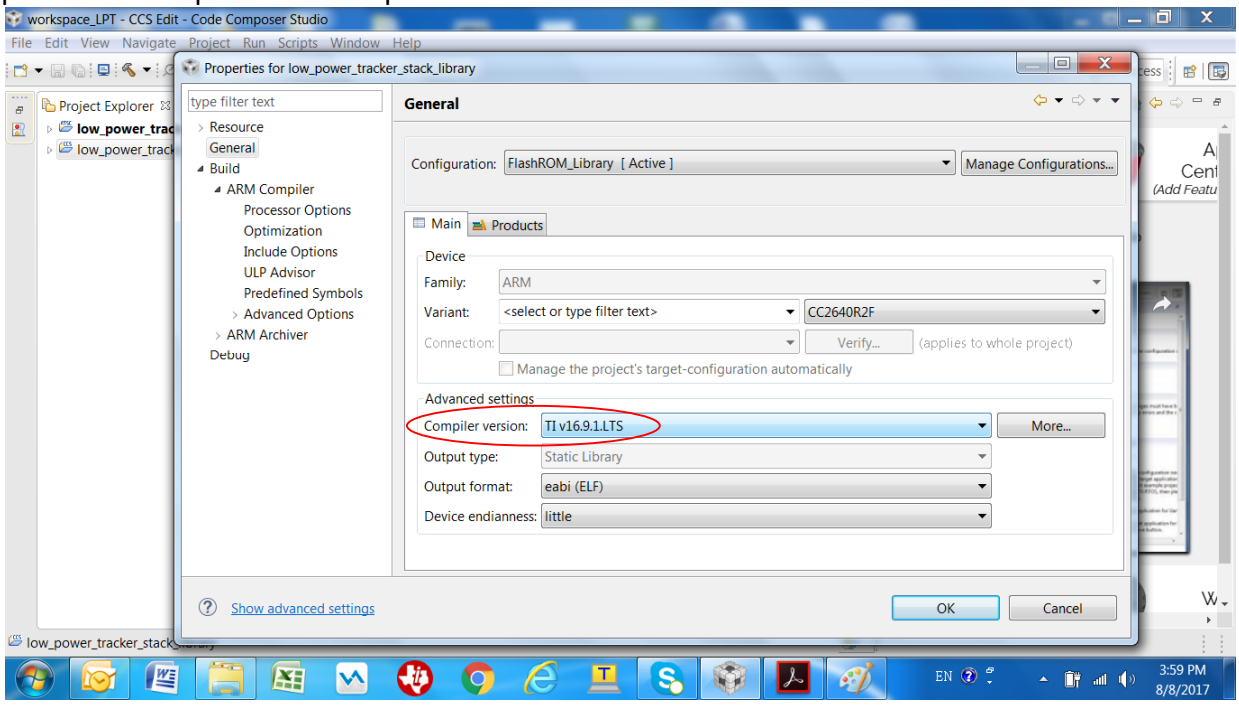


Build and run the application

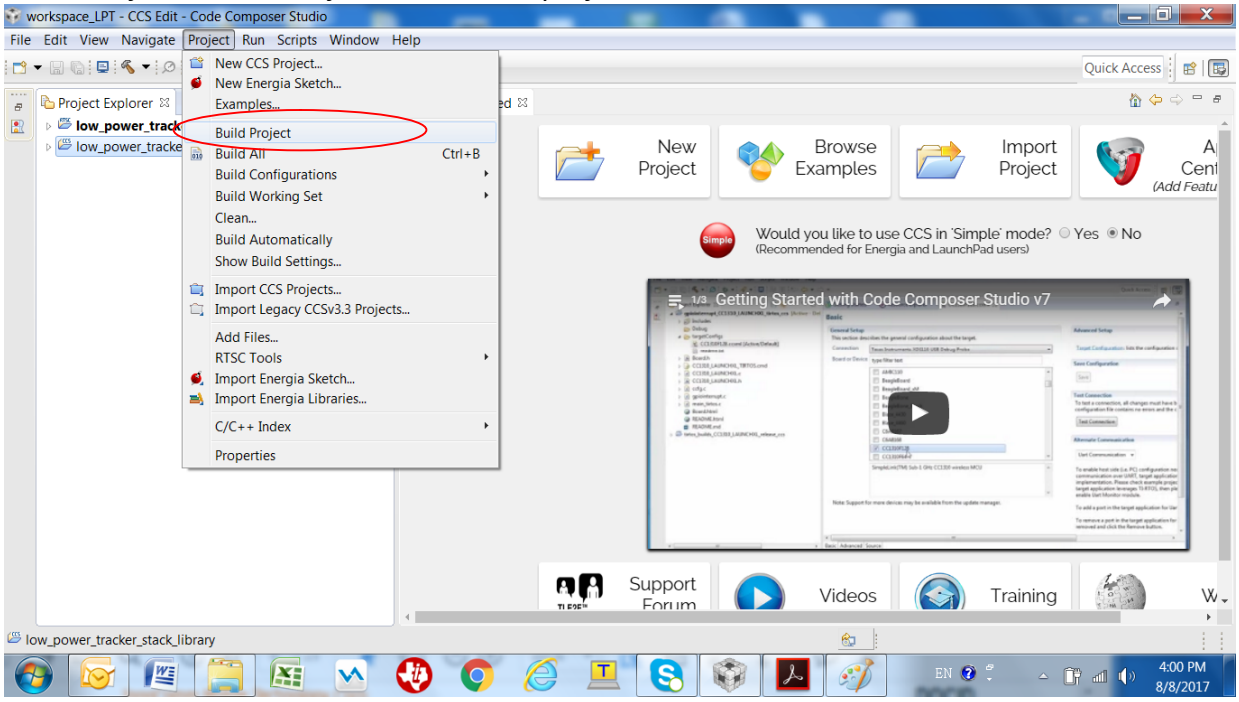
You may now build the two projects, the low_power_tracker_app and the low_power_tracker_stack_library, you just imported.

1. Go to the Project Explorer of the CCS and select the low_power_tracker_stack_library project
2. Check the compiler version of the CCS at Project->Properties->General->Advanced setting-

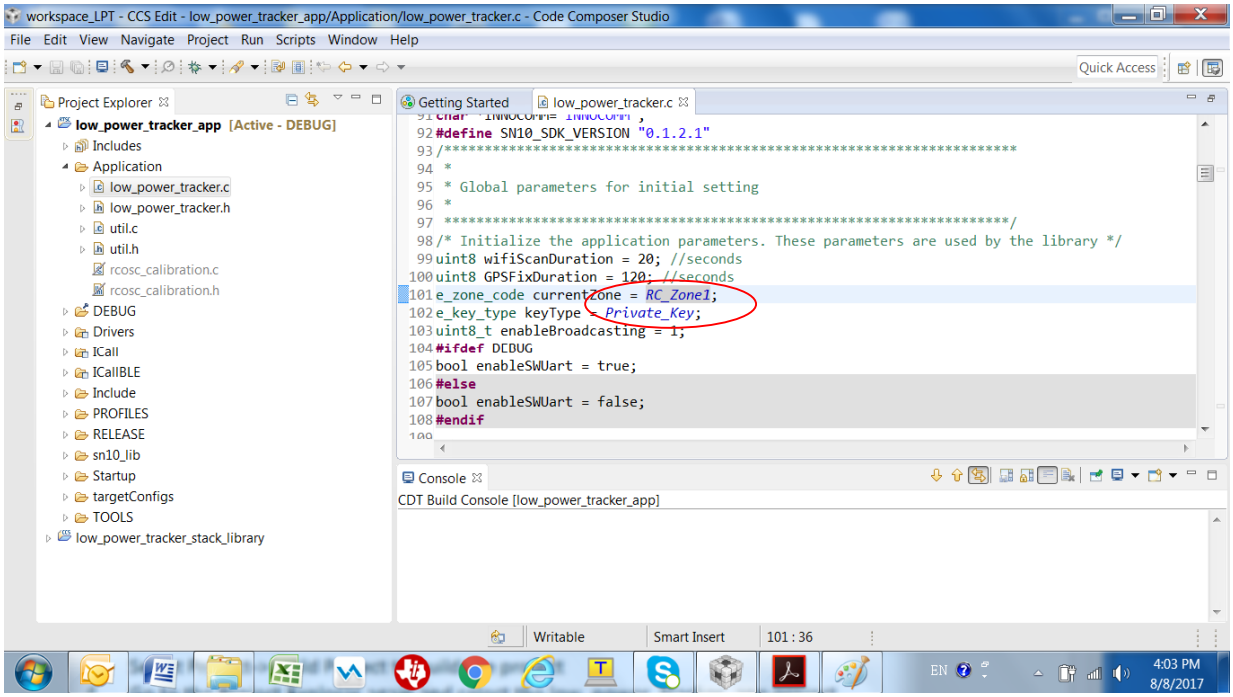
>Compiler version. If the version is not v16.9.1 follow "Update the TI ARM Compiler" procedure to update the compile



3. Select Project-> Build Project to build the project

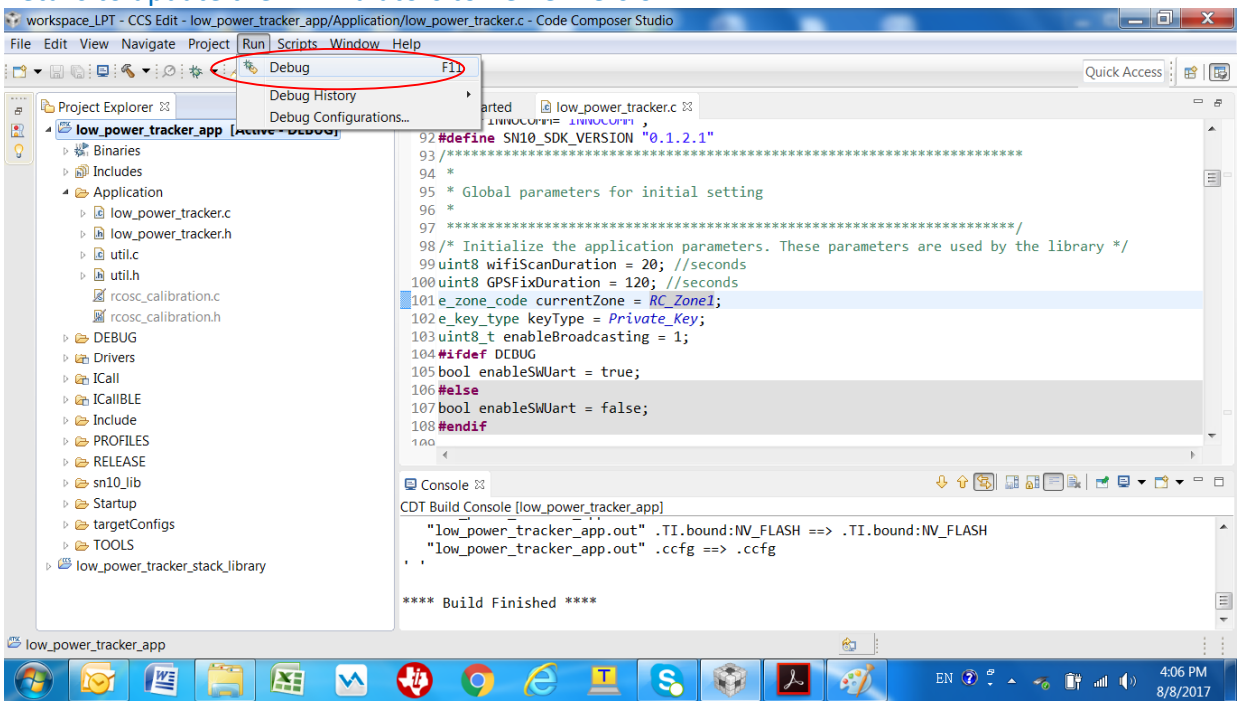


4. Go to the Project Explorer again and select the low_power_tracker_app project
5. Open low_power_tracker.c under Application folder and check the currentZone and the keyType parameters. The default is zone 1 and private key. Change the currentZone parameter to the zone your device will be working in, and change the keyType parameter to public if it is going to be connected to the network emulator SNEK.

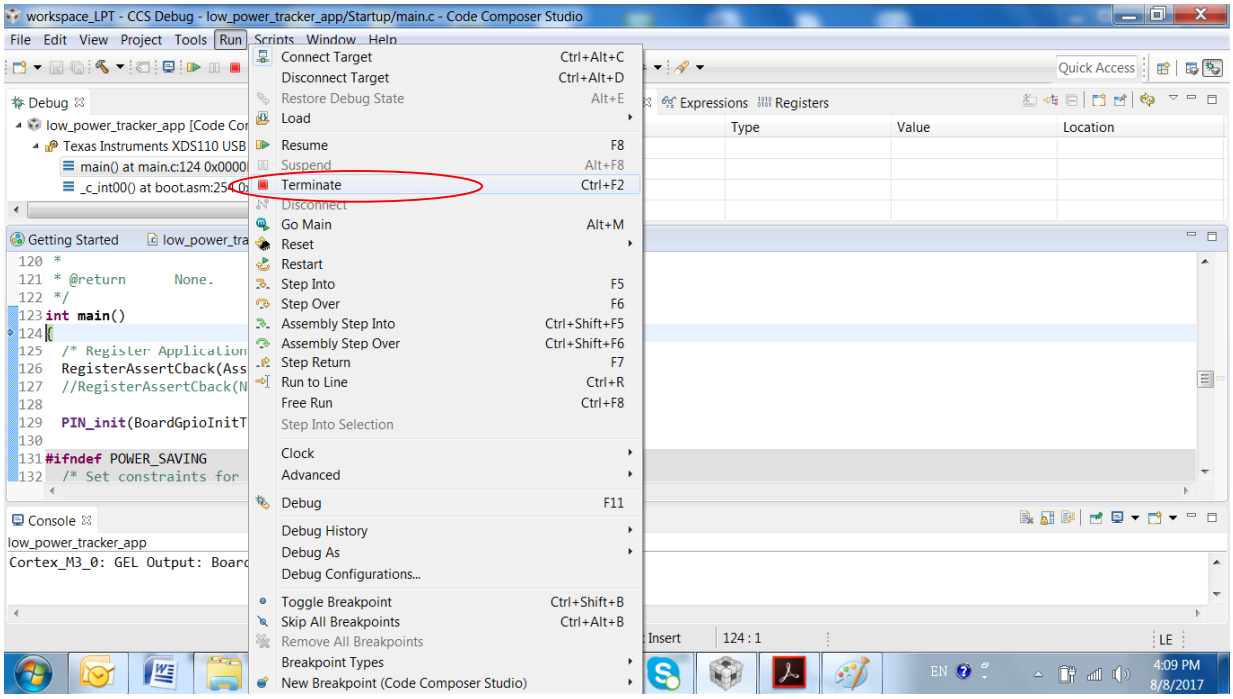


6. Select Project->Build Project to build the project
7. Select Run-> Debug to flash the binary to the module

   Note: if you experienced problem flashing the binary to the target, select Help-> Installation Details to update the TI Emulators to newer version.



8. Select Run->Terminate to terminate the debug session.



9. Reset the EVB to start the low power tracker application

Update the TI ARM Compiler

1. In CCS select Help->Install New Software
2. In “Work with”, select Code Generation Tools Updates (windows or Linux of your choice) and uncheck “Show only the latest versions of available software”
3. Select TI Compiler Updates -> ARM compiler Tools 16.9.1 and follow the flow to install

5. Enumerations, Structures and Unions

This section gives the details of the data structures and the APIs of the library.

typedef enum
{
***SIGFOX_OK* = 0,**
SIGFOX_NOK
} e_sfx_ret;
Used in all sigfox related APIs.

typedef enum
{
***RC_Zone1* = 0,**
***RC_Zone2*,**
***RC_Zone3*,**
RC_Zone4
} e_zone_code;
Used in changeRCZ() function to indicate the RC zone to change to.

typedef enum
{
***Private_Key* = 0,**
Public_Key
} e_key_type;
Used in switchKey() function to select the private key or the public key to be used.

typedef enum
{
***Uplink_Frequency* = 0,**
Downlink_Frequency
} e_frequency_type;
Used in setFrequency() and getFrequency() function to select the frequency type.

typedef enum
{
***Switch_Off_Continuous_Wave* = 0,**
Switch_On_Continuous_Wave
} e_continuousWave_state;
Used in continuousWave() function to switch on or switch off the continuous wave.

typedef enum
{
***SFX_TEST_MODE_TX_BPSK* = 0,**
***SFX_TEST_MODE_TX_PROTOCOL* = 1,**

```
    FX_TEST_MODE_RX_PROTOCOL = 2,
    FX_TEST_MODE_RX_GFSK     = 3,
    SFX_TEST_MODE_RX_SENSI   = 4,
    SFX_TEST_MODE_TX_SYNTH   = 5,
} sfx_test_mode_t;
```

Used in sendTestMode() function to select the test mode.

```
typedef struct
{
    uint8 devId[BSSID_LEN];
    int16 rssi;
} wifiAPRec_t;
```

Used in scanWifiAPs() function to hold the information of the returned Wifi APs.

```
typedef struct
{
    float latitude;
    float longitude;
} gpsRec_t;
```

Used in fixGPSLocation() function to hold the latitude and longitude of the fix.

```
typedef union
{
     int8 s;
     uint8 us;
     int16   i;
     uint16 ui;
     int32 l;
     uint32 ul;
     float f;
     char *p;
} data_t;
```

Used in trace() function to indicate the type of data to be print out.

6. Functions

6.1 Module

void initiateModule(void (*IOexpanderCB)(), void (*GAPEvtCB)())
Function to initiate the module.
Parameters:
[in] void (*IOexpanderCB)()
Pointer to the call back function for the IOexpander interrupts
[in] void(*GAPEvtCB)()
Pointer to the call back function for the GAP events
Return:
None

6.2 SIGFOX

e_sfx_ret sendData(uint8 *payload, uint8 payloadLen, uint8 *rcvBuf, uint8 *pLen)
Function to send data to the SIGFOX network. This function will wake up the SIGFOX module, switch to the designated RCZ, send the payload and at the end, send the SIGFOX module to sleep.
Parameters:
[in] payload
The payload to be sent to the network.
[in] payloadLen, up to 12.
Size of the payload in bytes.
[out] rcvBuf
The feedback from the network. Should pass NULL if not expecting feedback.
[out] pLen
Size of the feedback in bytes.
Return:
SIGFOX_OK or SIGFOX_NOK

e_sfx_ret sendBitStatus(uint8 bitValue, uint8 *rcvBuf, uint8 *pLen)
Function to send bit status to the SIGFOX network. This function will wake up the SIGFOX module, switch to the designated RCZ, send the bit status and at the end, send the SIGFOX module to sleep.
Parameters:
[in] bitValue
The status, should be 0 or 1.
[out] rcvBuf
The feedback from the network. Should pass NULL if not expecting feedback.
[out] pLen
Size of the feedback in bytes
Return:

SIGFOX_OK or SIGFOX_NOK

(The aforementioned two APIs should be sufficient for most of the applications. The following APIs are only needed only if finer control of the SIGFOX module is required)

e_sfx_ret sendWakeup()
Function to wakeup the SIGFOX module.
Parameters:
None
Return:
SIGFOX_OK or SIGFOX_NOK

e_sfx_ret sendToSleep()
Function to send the SIGFOX module to sleep.
Parameters:
None
Return:
SIGFOX_OK or SIGFOX_NOK.

e_sfx_ret sendPayload(uint8 *payload, uint8 payloadLen, uint8 *rcvBuf, uint8 *pLen)
Function to send payload to the network and optionally receive feedback from it.
Parameters:
[in] payload
The payload to be sent to the network.
[in] payloadLen, up to 12.
Size of the payload in bytes.
[out] rcvBuf
The feedback from the network. Should pass NULL if not expecting feedback.
[out] pLen
Size of the feedback in bytes.
Return:
SIGFOX_OK or SIGFOX_NOK.

e_sfx_ret sendBit(uint8 bitValue, uint8 *rcvBuf, uint8 *pLen)
Function to send a bit status to the network and optionally receive feedback from it.
Parameters:
[in] bitValue
The status, should be 0 or 1.
[out] rcvBuf
The feedback from the network. Should pass NULL if not expecting feedback.
[out] pLen
Size of the feedback in bytes
Return:
SIGFOX_OK or SIGFOX_NOK.

e_sfx_ret sendOutOfBand()
Function to have SIGFOX module send out of band message.
Parameters:
None.
Return:
SIGFOX_OK or SIGFOX_NOK.

e_sfx_ret getInfo(uint8 *deviceId, uint8 *pac, char *libVersion)
Function to get the information of the SIGFOX module.
Parameters:
[out] deviceId
The 4 bytes SIGFOX ID of the SIGFOX module
[out] pac
The 8 bytes PAC of the SIGFOX module
[out] libVersion
The library version, 11 bytes.
Return:
SIGFOX_OK or SIGFOX_NOK.

e_sfx_ret setFrequency (e_frequecny_type fType, uint32 freq)
Function to set the frequency of the SIGFOX module
Parameters:
[in] fType
Uplink_Frequency or Downlink_Frequency.
[in] freq
The frequency to set to.
Return:
SIGFOX_OK or SIGFOX_NOK.

e_sfx_ret getFrequency (e_frequecny_type fType, uint32* freq)
Function to get the frequency of the SIGFOX module
Parameters:
[in] fType
Uplink_Frequency or DownLInk_Frequency
[in] freq
The returned frequency.
Return:
SIGFOX_OK or SIGFOX_NOK.

e_sfx_ret continuousWave (e_continuousWave_state state)
Function to have the SIGFOX module send continuous wave
Parameters:
[in] state
Switch_On_Continuous_Wave or Switch_Off_Continuous_Wave
[in] freq

The returned frequency.
Return:
SIGFOX_OK or SIGFOX_NOK.

e_sfx_ret getDeviceVersion (char *hwVersion, char *devVersion)
Function to get the hardware version and the device version of the SIGFOX module
Parameters:
[out] hwVersion
8 bytes hardware version
[out] devVersion
7 bytes device version.
Return:
SIGFOX_OK or SIGFOX_NOK.

e_sfx_ret triggerWatchdog ()
Function to reset the SIGFOX module
Parameters:
None
Return:
SIGFOX_OK or SIGFOX_NOK

e_sfx_ret sendTestMode(sfx_test_mode_t txTestMode, uint8 txTestConfig)
Function to have the SIGFOX module send signals for RF testing.
Parameters:
[in] txTestMode
[in] txTestConfig
Return:
SIGFOX_OK or SIGFOX_NOK.

e_sfx_ret changeRCZ (e_zone_code zone)
Function to switch to the selected zone.
Parameters:
[in] zone
RC_Zone1, ETSI Europe (863~870MHz)
RC_Zone2, FCC US (902~928MHz)
RC_Zone3, ARIB Japan, Korea (915~930MHz)
RC_Zone4, FCC Latin America, Australia, New Zealand (902~915MHz)
Return:
SIGFOX_OK or SIGFOX_NOK.

e_sfx_ret switchKey (e_key_type keyType)
Function to switch to selected key type.
Parameters:
[in] keyType
Public_Key, use public key

Private_Key, use private key
Return:
SIGFOX_OK or SIGFOX_NOK.

e_sfx_ret setFCCMacroChannel(char *configString, uint8 defaultChannel)
Function to configure the enabled channels for FCC
Parameters:
[in] configString
Configuration (disabled or enabled) of the Macro channels.
[in] defaultChannel
The default SIGFOX macro channel.
Return:
SIGFOX_OK or SIGFOX_NOK.

void toZone(e_zone_code zone)
Function to set the zone code of the module.
Parameters:
[in] zone
Zone to set to
Return:
None

void useKeyType (e_key_type type)
Function to set the module to use private key or public key when sending data to the network.
Parameters:
[in] type
Private key or public key
Return:
None

uint8 getErrCode()
Function to the error code of a sigfox API.
Parameters:
None
Return:
uint8 errCode

6.3 WIFI & GPS

uint8 scanWifiAPs(wifiAPRec_t *APList)
Function to scan the surrounding for the top two Wifi APs of strongest signal.
Parameters:
[out] APList
Records of the Wifi APs found.

Return:
Number of the APs found: 0, 1, or 2.

bool fixGPSLocation (gpsRec_t *GPSInfo)
Function to get the fix of the GPS location.
Parameters:
[out] GPSInfo
Latitude and longitude of the fix.
Return:
true if a fix is attained, false otherwise.

6.4 Bluetooth

void BLEBroadcaster_processStackMsg(ICall_Hdr *pMsg)
Function to process the BLE stack message.
Parameters:
[in] pMsg
Return:
None.

void BLEBroadcaster_processStateChangeEvt(gaprole_States_t newState)
Function to process the state change events from GAP.
Parameters:
[in] newState
Return:
None.

void setBroadcastInterval(uint16 interval)
Function to set the interval of the beacon broadcast.
Parameters:
[in] interval
Time interval in ms.
Return:
None.

void setBroadcastData(uint8* data)
Function to set the beacon broadcast data.
Parameters:
[in] data
Broadcast data.
Return:
None.

void getBroadcastData(uint8* data)

Function to get the beacon broadcast data.
Parameters:
[out] data
Broadcast data.
Return:
None.

void enableBroadcast()
Function to enable beacon broadcast
Parameters:
None
Return:
None.

void disableBroadcast()
Function to disable beacon broadcast
Parameters:
None
Return:
None.

6.5 G-Sensor

void setGsensor_Slope_Threshold(Slope_Threshold_Level Slope_Level)
Function to set the G sensor slope threshold.
Parameters:
[in] Slope_Threshold_Level Slope_Level
G sensor slope level.
Return:
None

void setGsensor_HighG_Threshold(HigG_Threshold_Level HighG_Level)
Function to set the G sensor slope threshold.
Parameters:
[in] HighG_Threshold_Level HighG_Level
G sensor high G level.
Return:
None

6.6 Utilities

void trace (char *str, data_t d)
Function to print a trace line.
Parameters:

[in] str
The format string.
%ds for int8, %d for int16, %u for uint16, %l for int32, %ul for uint32, %s for char*
[in] d
Data to be print out
Return:
None.
(Note: this function only takes one parameter, as follows:
Int16 rssi = -109;
trace ("RSSI = %d\r\n", (data_t)rssi);
)

6.7 IO expander

Void IOEXP_setPinConfig(uint8 pin, bool enable_int)
Function to configure input pin.
Parameters:
[in] pin
Pin ID, can be either IO_EXP_P4, IO_EXP_P5 or IO_EXP_P7
[in] enable_int, true or false.

Uint8 IOEXP_getInputValue(uint8 pin)
Function to get the input value of the input pin.
Parameters:
[in] pin
Pin ID, can be either IO_EXP_P4, IO_EXP_P5 or IO_EXP_P7
Return:
0 or 1.

void IOEXP_setOutputValue(uint8 pin, unit8 value)
Function to set the value of the output pin.
Parameters:
[in] pin
Pin ID, can be either IO_EXP_P4, IO_EXP_P5 or IO_EXP_P7
[in] value, 0 or 1

Appendix A

Error codes of the SIGFOX functions.

Error Code Name	Code	Error Description
SFX_ERR_NONE	0x00	No error
SFX_ERR_OPEN_MALLOC	0x10	Error on MANUF_API_malloc or buffer pointer NULL
SFX_ERR_OPEN_ID_PTR	0x11	ID pointer NULL
SFX_ERR_OPEN_GET_SEQ	0x12	Error on MANUF_API_get_nv_mem w/ SFX_NVMEM_SEQ_CPT
SFX_ERR_OPEN_GET_PN	0x13	Error on MANUF_API_get_nv_mem w/ SFX_NVMEM_PN
SFX_ERR_OPEN_STATE	0x14	State is not idle, library should be closed before
SFX_ERR_OPEN_GET_FCC	0x15	Error on MANUF_API_get_nv_mem w/ SFX_NVMEM_FCC
SFX_ERR_CLOSE_FREE	0x20	Error on MANUF_API_free
SFX_ERR_CLOSE_RF_STOP	0x21	Error on MANUF_API_rf_stop
SFX_ERR_SEND_FRAME_DATA_LENGTH	0x30	Customer data length > 12 Bytes
SFX_ERR_SEND_FRAME_STATE	0x31	State != READY, must close and reopen library
SFX_ERR_SEND_FRAME_RESPONSE_PTR	0x32	Response data pointer NULL in case of downlink
SFX_ERR_SEND_FRAME_BUILD_UPLINK	0x33	Build uplink frame failed
SFX_ERR_SEND_FRAME_SEND_UPLINK	0x34	Send uplink frame failed
SFX_ERR_SEND_FRAME_RECEIVE	0x35	Receive downlink frame failed or timeout
SFX_ERR_SEND_FRAME_DELAY_OOB_ACK	0x36	Error on MANUF_API_delay w/ SFX_DLY_OOB_ACK (Downl
SFX_ERR_SEND_FRAME_BUILD_OOB_ACK	0x37	Build out of band frame failed (Downlink)
SFX_ERR_SEND_FRAME_SEND_OOB_ACK	0x38	Send out of band frame failed (Downlink)
SFX_ERR_SEND_FRAME_DATA_PTR	0x39	Customer data pointer NULL
SFX_ERR_SEND_FRAME_CARRIER_SENSE_CONFI	0x3A	Carrier Sense configuration need to be initialized
SFX_ERR_SEND_FRAME_CARRIER_SENSE_TIMEO	0x3B	Wait for clear channel has returned time out
SFX_ERR_SEND_FRAME_WAIT_TIMEOUT	0x3E	Wait frame has returned time out
SFX_ERR_SEND_FRAME_INVALID_FCC_CHAN	0x3F	FCC invalid channel, must call SIGFOX_API_reset
SFX_ERR_SEND_BIT_STATE	0x41	State != READY, must close and reopen library
SFX_ERR_SEND_BIT_RESPONSE_PTR	0x42	Response data pointer NULL in case of downlink
SFX_ERR_SEND_BIT_BUILD_UPLINK	0x43	Build uplink frame failed
SFX_ERR_SEND_BIT_SEND_UPLINK	0x44	Send uplink frame failed
SFX_ERR_SEND_BIT_RECEIVE	0x45	Receive downlink frame failed or timeout
SFX_ERR_SEND_BIT_DELAY_OOB_ACK	0x46	Error on MANUF_API_delay w/ SFX_DLY_OOB_ACK (Downl
SFX_ERR_SEND_BIT_BUILD_OOB_ACK	0x47	Build out of band frame failed (Downlink)
SFX_ERR_SEND_BIT_SEND_OOB_ACK	0x48	Send out of band frame failed (Downlink)
SFX_ERR_SEND_BIT_DATA_PTR	0x49	Customer data pointer NULL
SFX_ERR_SEND_BIT_WAIT_TIMEOUT	0x4E	Wait frame has returned time out
SFX_ERR_SEND_BIT_INVALID_FCC_CHAN	0x4F	FCC invalid channel, must call SIGFOX_API_reset
SFX_ERR_SEND_OOB_STATE	0x51	State != READY, must close and reopen library
SFX_ERR_SEND_OOB_BUILD_UPLINK	0x53	Build uplink frame failed
SFX_ERR_SEND_OOB_SEND_UPLINK	0x54	Send uplink frame failed

SFX_ERR_SEND_OOB_INVALID_FCC_CHAN	0x5F	Send out of band frame failed (Downlink)
SFX_ERR_SET_STD_CONFIG_SIGFOX_CHAN	0x90	Default SIGFOX channel out of range
SFX_ERR_SET_STD_CONFIG_SET	0x91	Unable to set configuration
SFX_ERR_TEST_MODE_0_RF_INIT	0xA0	Error on MANUF_API_rf_init
SFX_ERR_TEST_MODE_0_CHANGE_FREQ	0xA1	Error on MANUF_API_change_frequency
SFX_ERR_TEST_MODE_0_RF_SEND	0xA2	Error on MANUF_API_rf_send
SFX_ERR_TEST_MODE_0_DELAY	0xA3	Error on MANUF_API_delay
SFX_ERR_TEST_MODE_0_RF_STOP	0xA4	Error on MANUF_API_rf_stop
SFX_ERR_TEST_MODE_STATE	0xB1	State != READY, must close and reopen library
SFX_ERR_TEST_MODE_2_REPORT_TEST	0xC0	Error on MANUF_API_report_test_result
SFX_ERR_TEST_MODE_3_RF_INIT	0xD0	Error on MANUF_API_rf_init
SFX_ERR_TEST_MODE_3_CHANGE_FREQ	0xD1	Error on MANUF_API_change_frequency
SFX_ERR_TEST_MODE_3_TIMER_START	0xD2	Error on MANUF_API_timer_start
SFX_ERR_TEST_MODE_3_REPORT_TEST	0xD3	Error on MANUF_API_report_test_result
SFX_ERR_TEST_MODE_3_TIMER_STOP	0xD4	Error on MANUF_API_timer_stop
SFX_ERR_TEST_MODE_3_RF_STOP	0xD5	Error on MANUF_API_rf_stop
SFX_ERR_TEST_MODE_4_BUILD_UPLINK	0xE0	Build uplink frame failed
SFX_ERR_TEST_MODE_4_SEND_UPLINK	0xE1	Send uplink frame failed
SFX_ERR_TEST_MODE_4_REPORT_TEST	0xE2	Error on MANUF_API_report_test_result
SFX_ERR_TEST_MODE_4_GET_RSSI	0xE3	Error on MANUF_API_get_rssi
SFX_ERR_TEST_MODE_4_DELAY	0xE4	Error on MANUF_API_delay
SFX_ERR_TEST_MODE_5_RF_INIT	0xF0	Error on MANUF_API_rf_init
SFX_ERR_TEST_MODE_5_CHANGE_FREQ	0xF1	Error on MANUF_API_change_frequency
SFX_ERR_TEST_MODE_5_BUILD_UPLINK	0xF2	Build uplink frame failed
SFX_ERR_TEST_MODE_5_SEND_UPLINK	0xF3	Send uplink frame failed
SFX_ERR_TEST_MODE_5_RF_STOP	0xF4	Error on MANUF_API_rf_stop

Appendix B

Hardware setup for MCU software download and update.

1) Connect the USB from the PC to the download port.
2) Short the JTAG pins.
3) Initiate download from CCS
4) Press the reset button after the download to start the application.

Hardware setup for trace.

1) Connect the AT command/trace port to the USB port of the PC
2) Start the terminal emulator (e.g. putty or tera term) and enter the following settings: baud rate:9600, data:8 bit, stop: 1 bit, parity: none, flow control:none.

Federal Communication Commission Interference Statement

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

FCC Caution: Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate this equipment.

This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

Radiation Exposure Statement:

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.

This device is intended only for OEM integrators under the following conditions:

1) The antenna must be installed such that 20 cm is maintained between the antenna and users, and
2) The transmitter module may not be co-located with any other transmitter or antenna.
3) Module approval valid only when the module is installed in the tested host or compatible series of host which have similar RF exposure characteristic with equal or larger antenna separation distance.

As long as **3** conditions above are met, further transmitter test will not be required. However, the OEM integrator is still responsible for testing their end-product for any additional compliance requirements required with this module installed

IMPORTANT NOTE: In the event that these conditions can not be met (for example certain laptop configurations or co-location with another transmitter), then the FCC authorization is no longer considered valid and the FCC ID can not be used on the final product. In these circumstances, the OEM integrator will be responsible for re-evaluating the end product (including the transmitter) and obtaining a separate FCC authorization.

End Product Labeling

This transmitter module is authorized only for use in device where the antenna may be installed such that 20 cm may be maintained between the antenna and users. The final end product must be labeled in a visible area with the following:
"Contains FCC ID:YAISN10-22". The grantee's FCC ID can be used only when all FCC compliance requirements are met.

Manual Information To the End User

The OEM integrator has to be aware not to provide information to the end user regarding how to install or remove this RF module in the user's manual of the end product which integrates this module.
The end user manual shall include all required regulatory information/warning as show in this manual.