A module/product solution provider

WS2118/WS2119 MODULE/EVB

FW DEVELOPMENT GUIDE

Version: 3.5.3

CHANGE HISTORY

Version	Date	Description	Author
WS211x_FW_Guide_3.5.1	2019.3.18	Updated descriptions for Jorjin SDK v.6.0.1	Joshua Guo
WS211x_FW_Guide_3.5.2	2021.12.29	Added notation for Sigfox transmissions	Jack Tseng
WS211x_FW_Guide_3.5.3	2022.1.22	Updated descriptions for Jorjin SDK v6.0.7	Jack Tseng

CATALOG

1 INTRODUCE

This document is for Sigfox and BLE dual mode module FW development guide, which is suitable for WS2118 and WS2119 EVB and Module. The WS2118 is designed for Sigfox RC1 and RC3 region (RC5 and RC6 regions can also be supported but are not certified). WS2119 is for Sigfox RC2 and RC4 region.

1.1 DEVELOPMENT SDK FROM STMICRO

2 items need to download from STMicro.

1. STSW-BNRG-S2LP evaluation software package based on BlueNRG-1 and S2-LP

http://www.st.com/content/st_com/en/products/embedded-software/wireless-connectivity-software/stsw-bnrg-s2lp.html

2. BlueNRG-1 ST-LINK utility for BlueNRG-1, BlueNRG-2 MCU

http://www.st.com/content/st_com/en/products/embedded-software/wireless-connectivity-software/stsw-bnrg1stlink.html

NOTE1: Default BlueNRG-1_ST-LINK_CLI.exe in NRG ST-LINK installation directory has issue which is section erase.

Please unzip **BlueNRG-1_ST-LINK_CLI.7z** and copy it inside the folder C:\Program Files (x86)\STMicroelectronics\BlueNRG-1_2 ST-Link Utility V 2.0.0\ST-LINK_Utility to replace the original file.

NOTE2: DON'T ERASE Page 78 and Page 79 of flash by STLINK CLI or GUI

More BlueNRG-1 SW package for from ST:

http://www.st.com/en/wireless-connectivity/bluenrg-1.html

More information from Sigfox:

https://resources.sigfox.com/

1.2 DEBUG TOOL - STLINK

Buy it from ST or contact ST distributor.

http://www.st.com/en/development-tools/st-link-v2.html

1.3 SIGFOX SIMULATOR - SDR DONGLE

You must use antenna with correct frequency domain for your development sigfox RC zone.

Sigfox RC zones have different frequency as below:

Zone 1: Europe, Oman, South Africa

Tx Frequency: 868.13MHz

Rx Frequency: 869.525MHz

Zone 2: USA, Mexico, Brazil

Tx Frequency: 902.2MHz

Rx Frequency: 905.2MHz

Zone 3: Japan

Tx Frequency: 923.2MHz

Rx Frequency: 922.2MHz

Zone 4: Australia, New Zealand, Singapore, Taiwan, Hong Kong, Colombia, Argentina

Tx Frequency: 920.8MHz

Rx Frequency: 922.3MHz

Zone 5: South Korea

Tx Frequency: 923.25MHz

Rx Frequency: 922.25MHz

Zone 6: India

Tx Frequency: 865.2MHz

Rx Frequency: 866.3MHz

More information

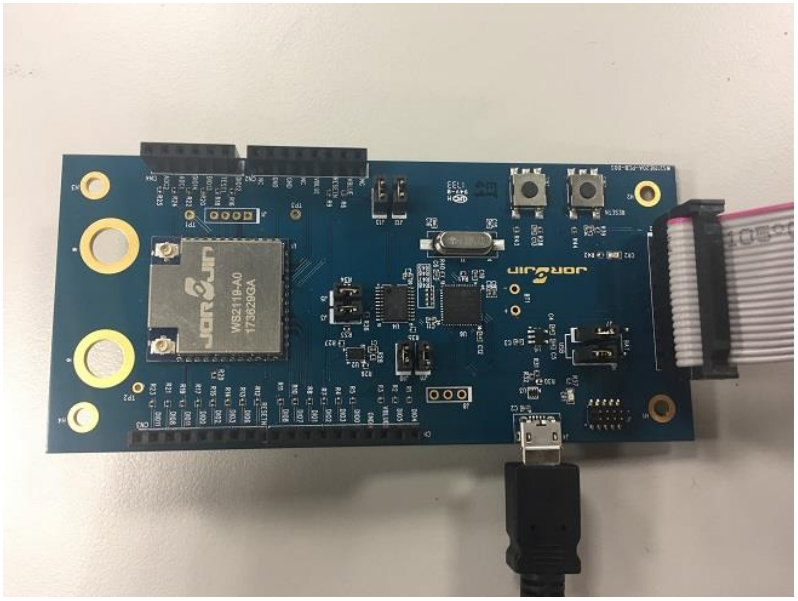
https://resources.sigfox.com/document/sigfox-sdr-dongle

https://storage.sbg1.cloud.ovh.net/v1/AUTH_669d7dfced0b44518cb186841d7cbd75/staging_docs/att19630513-1709-SIGFOX-DATASHEET-SDR_dongle.pdf

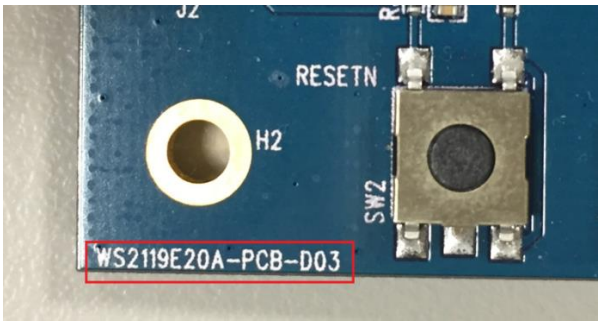
Where to buy SDR Dongle:

https://www.digikey.tw/product-detail/zh/sigfox/SDR-DONGLE/1895-1000-ND/7930762

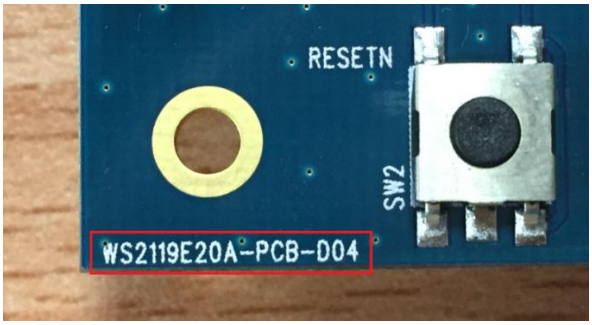
1.4 JORJIN SIGFOX AND BLE DUAL MODE EVB



There are 2 versions for EVB in current status. One is **WS211920A-PCB-D03** version which has limitation about STLINK debugger can't support runtime debug. Because EEPROM CS Pin has conflict with STLINK Pin.



The other one is **WS211920A-PCB-D04 which is fixed issue on D03 version EVB**

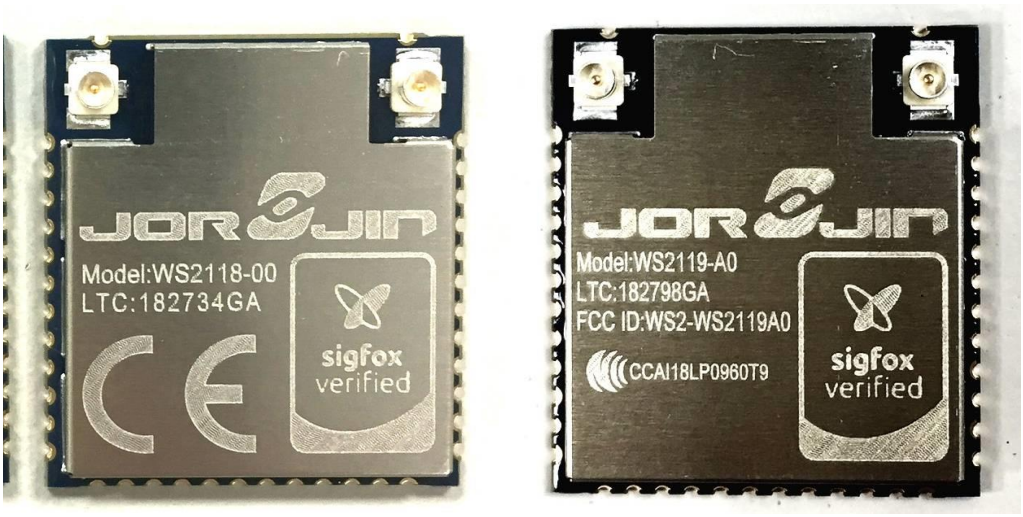


For the WS2118 and WS2119 EVB user guide you can download from Jorjin website or contact with Jorjin Sales.

1.5 JORJIN DUAL MODE MODULE

OLD version module

NEW version



1.6 DEVELOPMENT TOOL

- IAR embedded workbench 8.32+ (*Note: 8.50.9 is required since SDK v6.0.7)
- Jorjin SDK version 6.0.0+
- Standard AT command FW developed by Jorjin

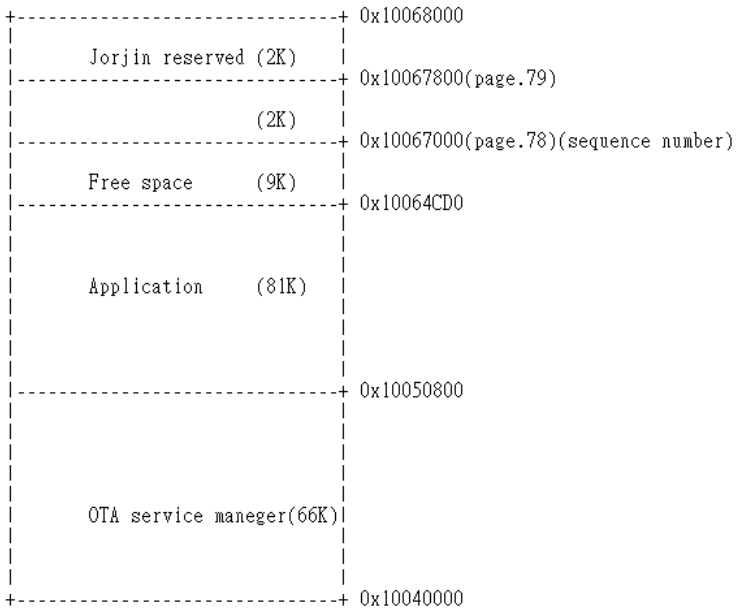
2 FLASH MAPPING IN BLUENRG-1

2.1 FLASH ADDRESS MAPPING

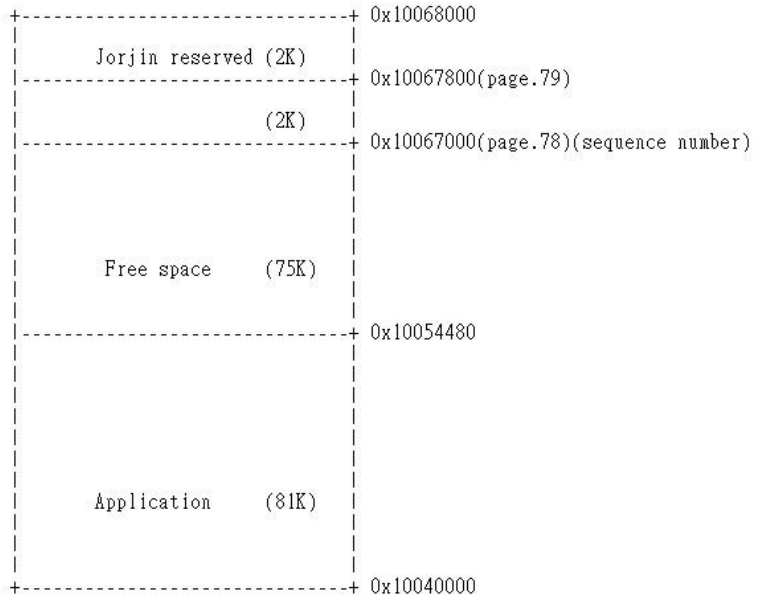
Note: DON'T modify page 78 and 79, there have module information and calibration parameter and SIGFOX sequence number.

Based on OTA service manager!

2.1.1 Enable OTA service manager



2.1.2 Disable OTA service manager



2.2 FLASH CONTROL

1. Reading Flash memory

   To read one single word of the flash, just read it as if RAM memory: read the desired flash address and get read data on the bus.

2. Erasing Flash

   The Flash controller allows erasing one page or the full main Flash.

3. Write function

   The Flash Controller allows writing one word (4 bytes), up to 4 words or the full main Flash memory (with a single fixed word).

4. Basic Flash operations:

   Erase a page

   Write a page word by word

   Verify write operation word by word

You have to erase flash (page) first then program flash. Otherwise you can't program data successfully.

2.2.1 Flash functions

Erase page (BlueNRG1_flash.c)

- *Void FLASH_ErasePage(uint16_t PagNumbere;)*

Erase all flash (BlueNRG1_flash.c)

- *void FLASH_EraseAllFlash(void);*

Read flash (uint32) (BlueNRG1_flash.c)

- *uint32_t FLASH_ReadWord(uint32_t Address);*

Read flash (uint8) (BlueNRG1_flash.c)

- *uint8_t FLASH_ReadByte(uint32_t Address);*

Program flash (BlueNRG1_flash.c)

- *void FLASH_ProgramWord(uint32_t Address, uint32_t Data);*

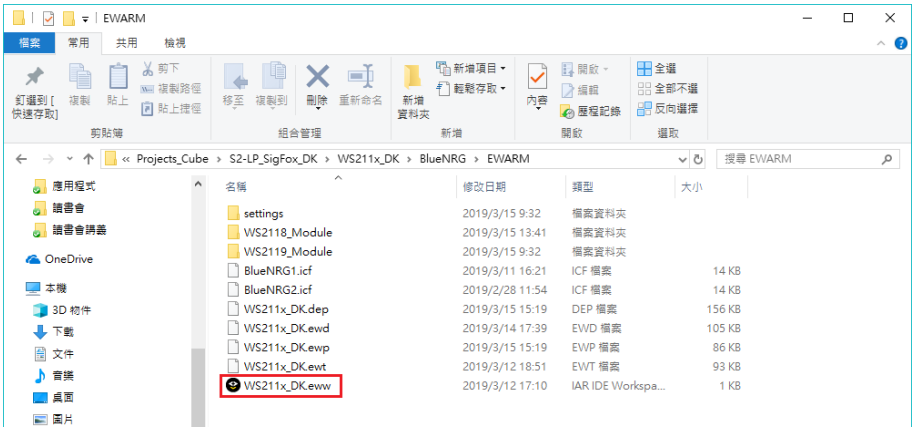
3 JORJIN SDK

3.1 SDK INFORMATION

Jorjin SDK is customized based on ST official released, which include latest sigfox library and PIN configuration base on WS2118 and WS2119 PIN definition.

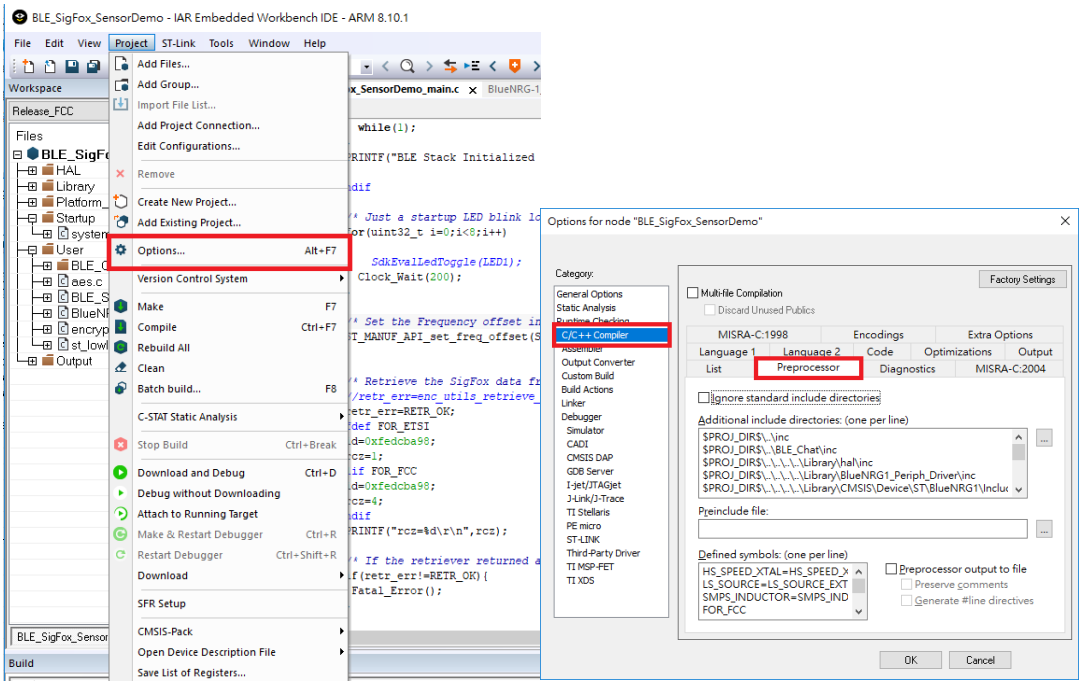
3.2 IAR EMBEDDED WORKBENCH IDE

3.2.1 Open project file



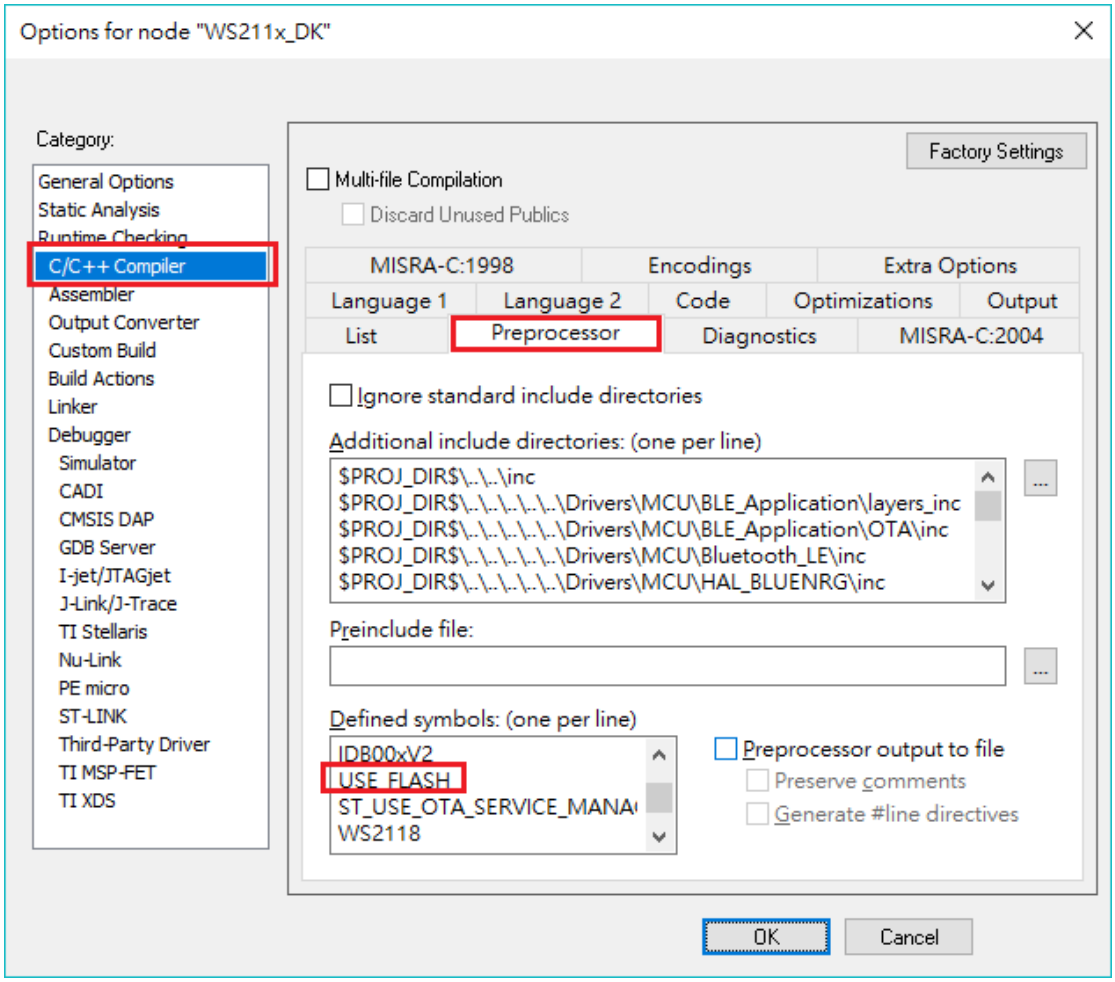
3.2.2 Defined symbols

Project→option→C/C++ compiler→preprocessor



1. Memory type

There have two types of memories, internal FLASH and external EEPROM. To use FLASH for data storing, the “USE_FLASH” symbol must be defined in the “Defined symbols” in IAR’s preprocessor settings (as shown in the following picture). If the “USE_FLASH” symbol is not defined, EEPROM will be used during the initialization.



The “HARDCODE” symbol controls the active ID, PAC, key and rcz at “ST_Sigfox_Init.c”. If it’s not defined, the Sigfox information stored in the memory will be used for the Sigfox operations.

If “HARDCODE” is defined, the hardcoded Sigfox information in “ST_Sigfox_Init.c” will be used, and the original information in the memory will also be overwritten by the hardcoded ones.

main.c | nvm_api.c | ST_Init.c | chat.c | chat.h | ST_Sigfox_Init.c | mcu_api_bluenrg1.c | jorjin_sigfox_retriever.h

```
ST_SFX_ERR ST_Sigfox_Init(NVM_BoardDataType *sfxConfig, uint8_t openAfterInit)
{
  ST_SFX_ERR ret_err = ST_SFX_ERR_NONE;
#ifdef HARDCODE
  uint32_t id = 0xFEDCBA98;
  uint8_t pac[8] = {0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88};
  uint8_t key[16] = {00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0xAA,0xBB,0xCC,0xDD,0xEE,0xFF};
#if S2LP_FEM_PRESENT == S2LP_FEM_NO
  uint8_t rcz = 1;
#else
  uint8_t rcz = 4;
#endif
#endif

  /* Configure XTAL frequency and offset for the RF Library */
  ST_RF_API_set_xtal_freq(S2LPManagementGetXtalFrequency());

  /* Macro that defines and initializes the nvmconfig structure */
  INIT_NVM_CONFIG(nvmConfig);

  /* Sigfox Credentials Management */
#ifdef USE_FLASH
  nvmConfig.nvmType = NVM_TYPE_FLASH;
  nvmConfig.sfxDataAddress = (uint32_t)FLASH_USER_START_ADDR;   /* Set here the address for 'NVM sigfox da
#endif

  /* Configure the NVM_API */
  SetNVMInitial(&nvmConfig);

#ifdef HARDCODE
  sfxConfig->id = id;
  sfxConfig->rcz = rcz;
  memcpy(sfxConfig->pac, pac, 8);

  enc_utils_set_rcz(sfxConfig->rcz);
  enc_utils_set_id(sfxConfig->id);
  enc_utils_set_key(key);
#ifdef USE_FLASH
  ret_err = (ST_SFX_ERR)ST_RF_API_set_xtal_freq(S2LPManagementGetXtalFrequency()+sfxConfig->freqOffset);

  if(!ret_err)
    ret_err = (ST_SFX_ERR)ST_RF_API_set_rssi_offset(sfxConfig->rssiOffset); /* Override RSSI offset */
#endif  //USE_FLASH
```

Use in main.c, ST_Sigfox_Init.c, nvm_api.c and S2LP_IDB00xV2_AUTO.h

main.c | ST_Sigfox_Init.c | ST_Init.c | chat.c | chat.h

Appli_Exti_CB(uint16_t)

```
#ifdef USE_FLASH
#include <string.h>
#endif
```

```
#ifdef USE_FLASH
  nvmConfig.nvmType = NVM_TYPE_FLASH;
  nvmConfig.sfxDataAddress = (uint32_t)FLASH_USER_START_ADDR;   /* Set here the address for 'NVM sigfox data' management */
#endif
```

```
#ifdef USE_FLASH
  ret_err = (ST_SFX_ERR)ST_RF_API_set_xtal_freq(S2LPManagementGetXtalFrequency()+sfxConfig->freqOffset);

  if(!ret_err)
    ret_err = (ST_SFX_ERR)ST_RF_API_set_rssi_offset(sfxConfig->rssiOffset); /* Override RSSI offset */
#endif  //USE_FLASH
```

```
#ifdef USE_FLASH
  /* Retrieve Sigfox info from FLASH */
  if((enc_utils_retrieve_data(&sfxConfig->id, sfxConfig->pac, &sfxConfig->rcz))==RETR_ERR)
  {
    if((enc_utils_check_key())==RETR_OK)
    {
      enc_utils_store_module_data();
      PRINTF("Update sigfox information\r\n");
      SdkDelayMs(100);
      NVIC_SystemReset();
    }
    else
    {
      enc_utils_set_public_key(1);
      PRINTF("Sigfox key is wrong, set public key\r\n");
    }
  }

  if(enc_check_version() == RETR_ERR)
  {
    updated_version();
    PRINTF("Sigfox sequence number returned to zero\r\n");
    SdkDelayMs(100);
    NVIC_SystemReset();
  }
  else
  {
    ret_err = (ST_SFX_ERR)ST_RF_API_set_xtal_freq(S2LPManagementGetXtalFrequency()+sfxConfig->freqOffset);

    if(!ret_err)
      ret_err = (ST_SFX_ERR)ST_RF_API_set_rssi_offset(sfxConfig->rssiOffset); /* Override RSSI offset */

    if(ret_err) /* An error occured reading freq, RSSI or LBT offsets */
        ret_err = ST_SFX_ERR_OFFSET;
  }
#else
```

main.c | nvm_api.c | ST_Init.c | chat.c | chat.h | jorjin_sigfox_retriever.h | ST_Sigfox_Init.c | mcu_api_bluenrg1.c

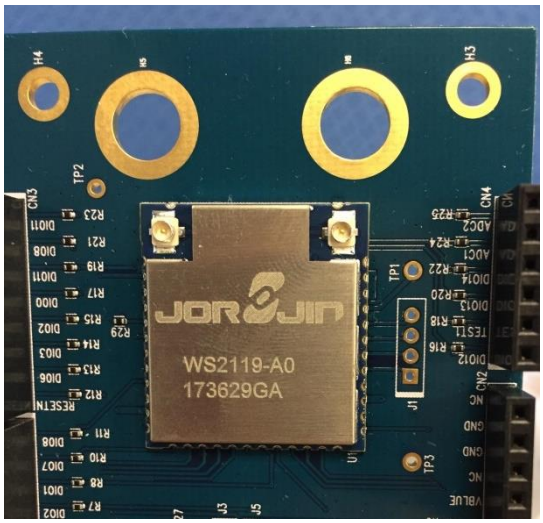
_setBlockState(uint32_t, uint64_t)

```
#ifdef USE_FLASH
#define _nvmReadOperation(addr, nbytes, buff) FlashRead(addr, nbytes, buff)
#define _nvmWriteOperation(addr, nbytes, buff, mode) FlashWrite(addr, nbytes, buff, mode)
#else
#include "S2LP_AUX_EEPROM.h"

#define _nvmReadOperation(addr, nbytes, buff) EepromRead(addr, nbytes, buff)
#define _nvmWriteOperation(addr, nbytes, buff, mode) EepromWrite(addr, nbytes, buff)
#endif
```

main.c | S2LP_IDB00xV2_AUTO.h | nvm_api.c | ST_Init.c | chat.c | chat.h | ST_Sigfox_

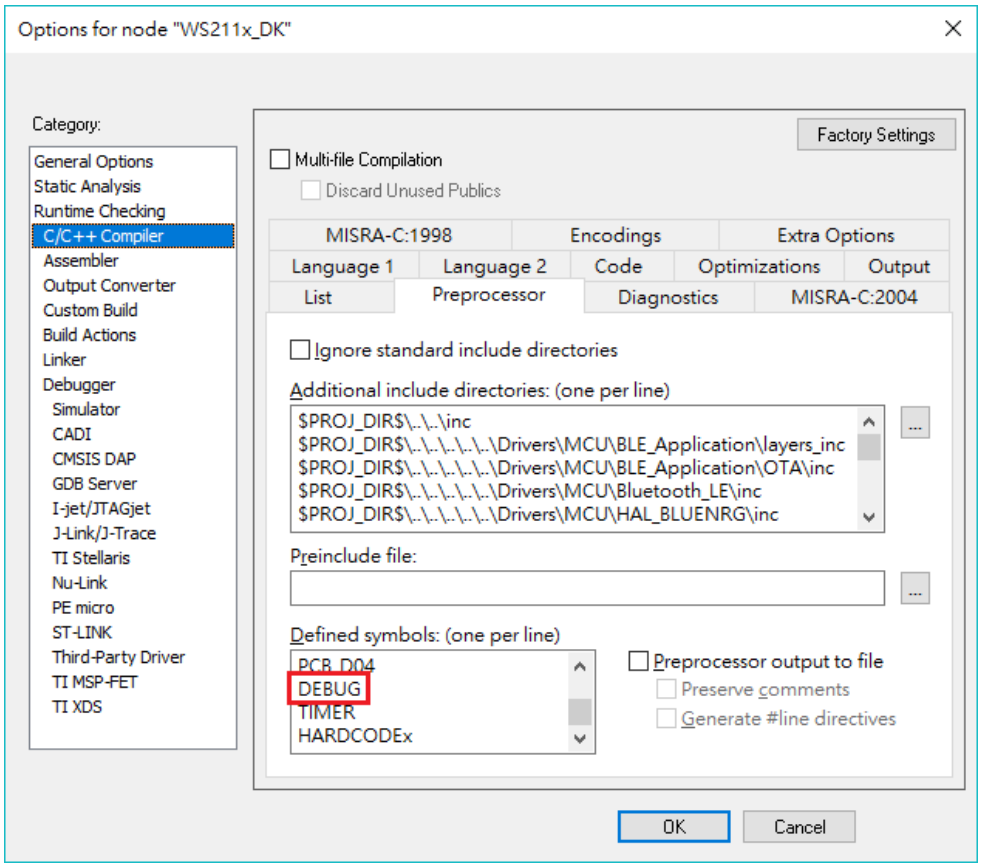
```
 * @brief Definitions for EEPROM
 */
#ifdef USE_FLASH
#define EEPROM_PRESENT                          EEPROM_NO
#else
#define EEPROM_PRESENT                          EEPROM_YES
#endif

/**
```

Notice: JORJIN WS2118 module is for zone 1, 3, 5, 6 and JORJIN WS2119 module is for zone 2 and 4.

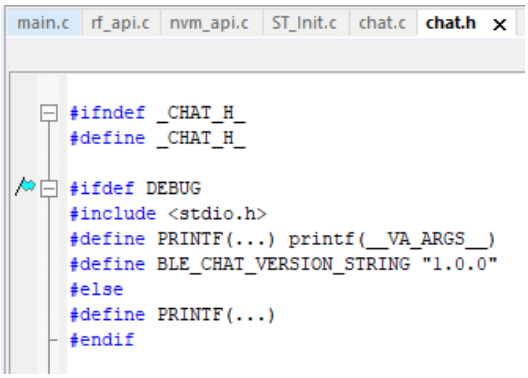


2. Debug mode

Set “DEBUG” to open debug mode, will show PRINTF(“”). Don't define symbols “DEBUG” to disable debug mode. If use printf(“”) debug message always show context(PRINTF ≠ printf).



It is used in chat.c, rf_api.c and mcu_api _bluewnrg1.c.



main.c | rf_api.c | nvm_api.c | ST_Init.c | chat.c | chat.h | ST_Sigfox_Init.c | mcu_api_bluenrg1.c

```
#ifdef DEBUG
//void ST_dbg_CB(const char *vectcStr,...);
//#define PRINTF(...)     { ST_dbg_CB(__VA_ARGS__);}
//#elif DEBUG_BLE
#include "SDK_EVAL_Com.h"
#define PRINTF(...) printf(__VA_ARGS__)
#else
#define PRINTF(...)
#endif
```

main.c | nvm_api.c | ST_Init.c | chat.c | chat.h | ST_Sigfox_Init.c | mcu_api_bluenrg1.c | jorjin_sigfox_retriever.h

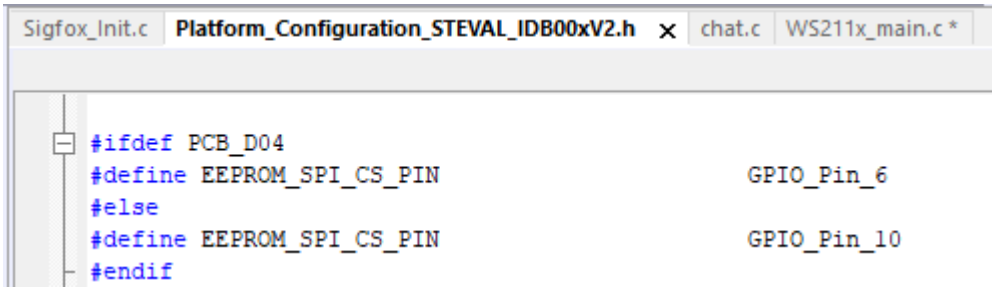
```
#ifdef DEBUG
//      #include <stdio.h>
//      void ST_dbg_CB(const char *vectcStr,...);
//      #define PRINTF(...)     { ST_dbg_CB(__VA_ARGS__);}
//#elif DEBUG_BLE
        #include <stdio.h>
        #include "SDK_EVAL_Com.h"
        #define PRINTF(...) printf(__VA_ARGS__)
#else
        #define PRINTF(...)
#endif
```

3. GPIO pin mapping

PCB-D03 and PCB-D04 have defined different eeprom cs pin.

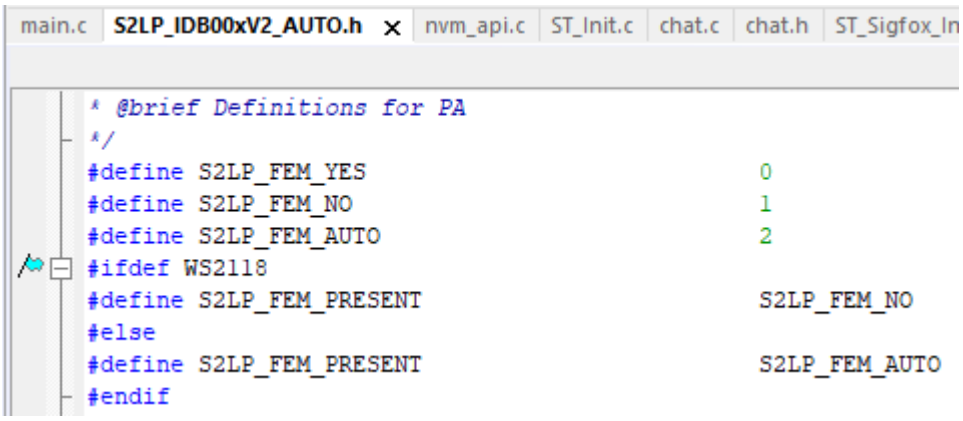
It is used in S2LP_IDB00xV2_AUTO.h.

- If use PCB-D04, defined PCB-D04 version.



4. Definitions for PA

Defined WS2118 or WS2119 to switch PA configuration.



5. OTA mode

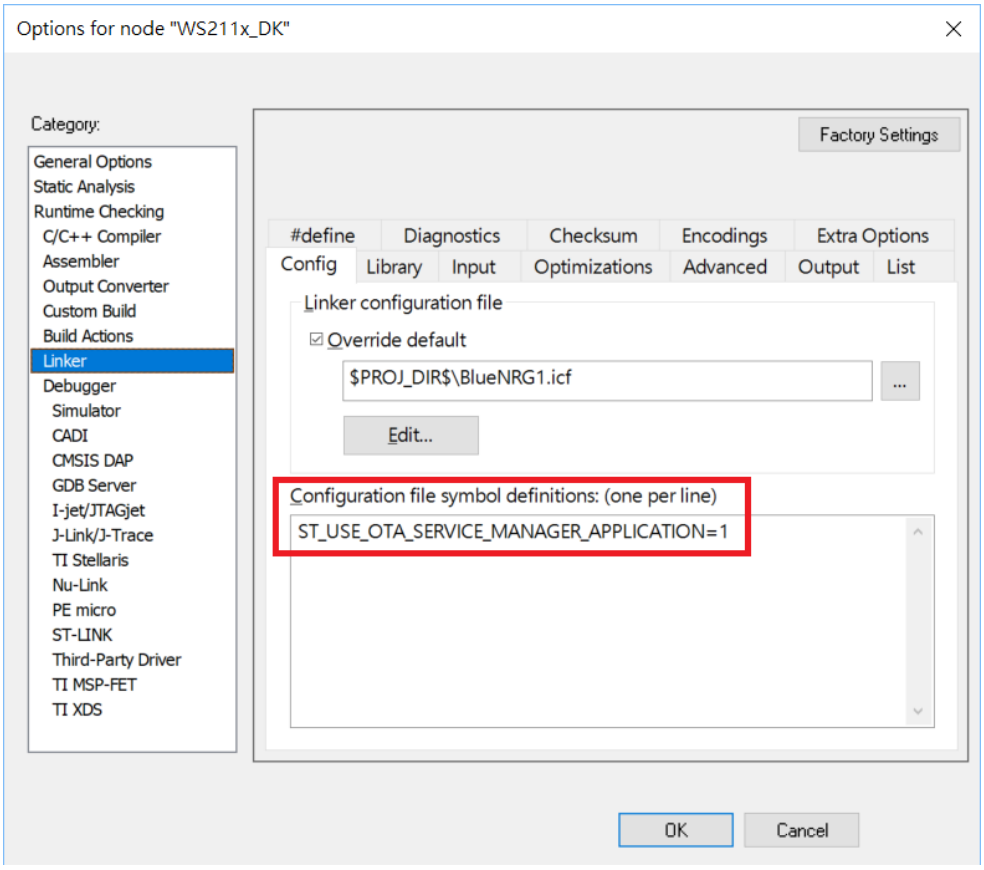
i. Enable OTA service manager

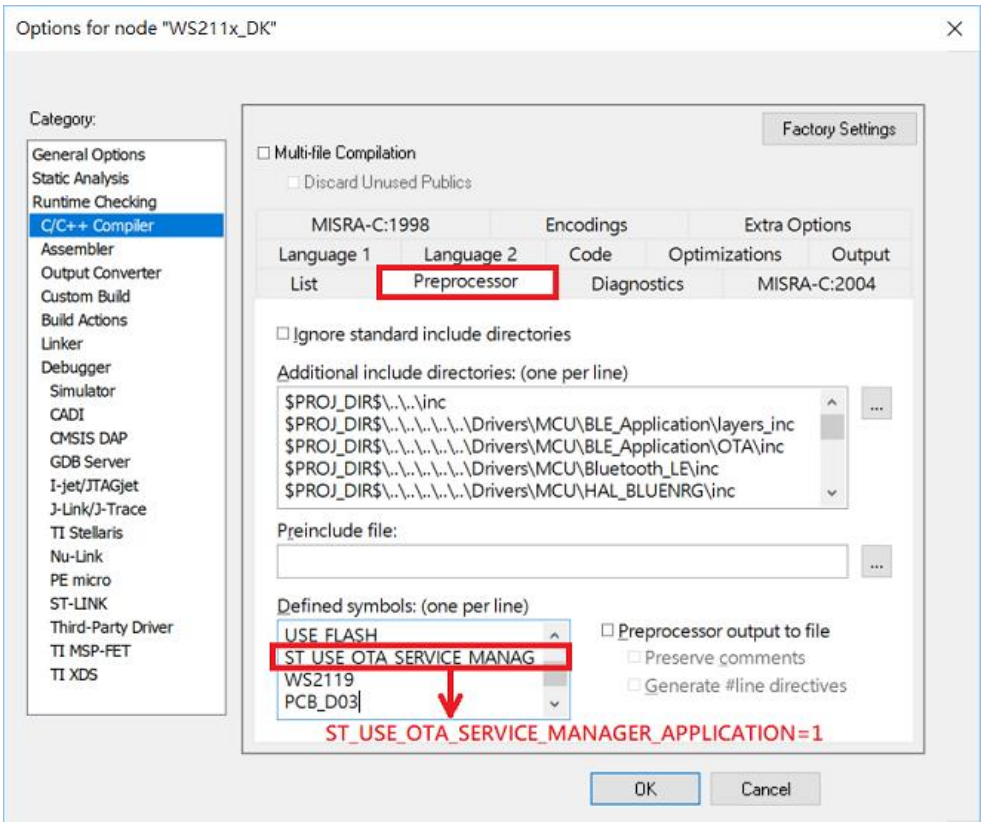
ST_USE_OTA_SERVICE_MANAGER_APPLICATION=1 at

Linker→ config → configuration file symbol definitions and

C/C++ compiler→ preprocessor

to enable OTA service manager will change Application start address to 0x10050800.





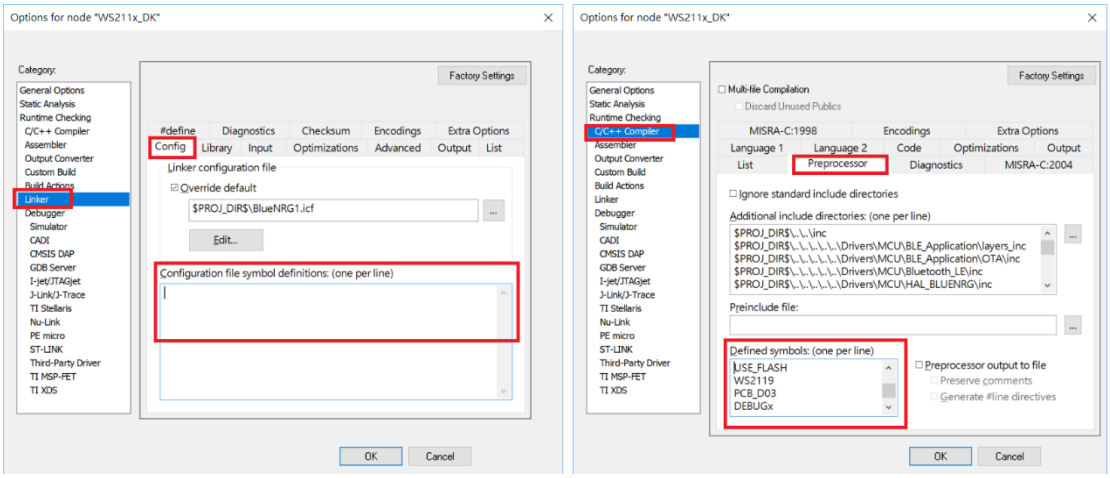
ii. Disable OTA service manager

Remove ST_USE_OTA_SERVICE_MANAGER_APPLICATION=1 at

Linker→ config → configuration file symbol definitions and

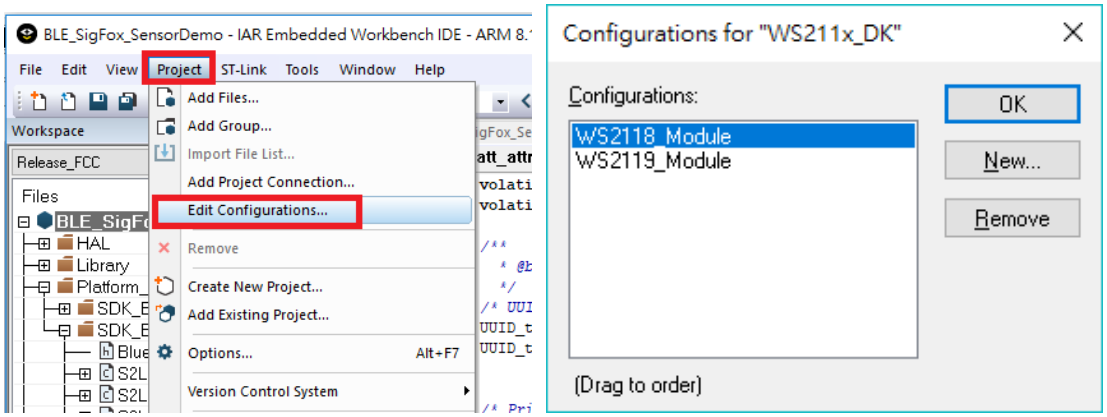
C/C++ compiler→ preprocessor

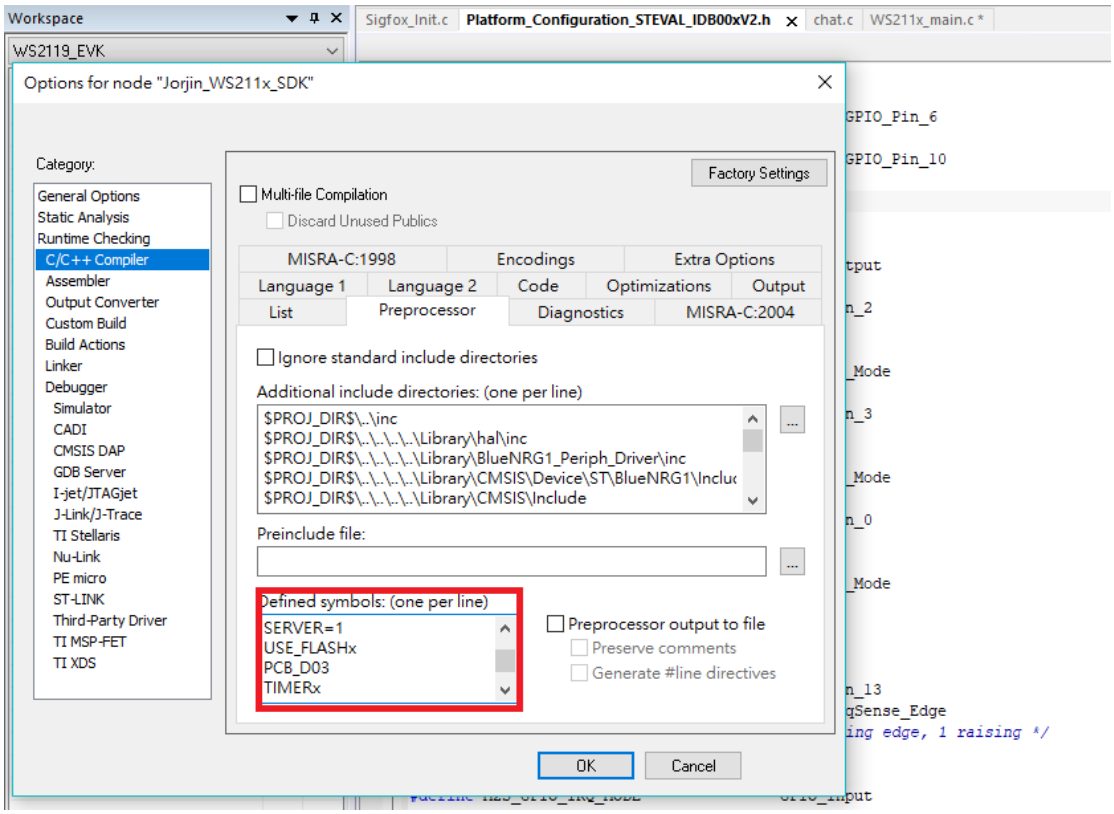
to disable OTA service manager will change Application start address to 0x10040000.

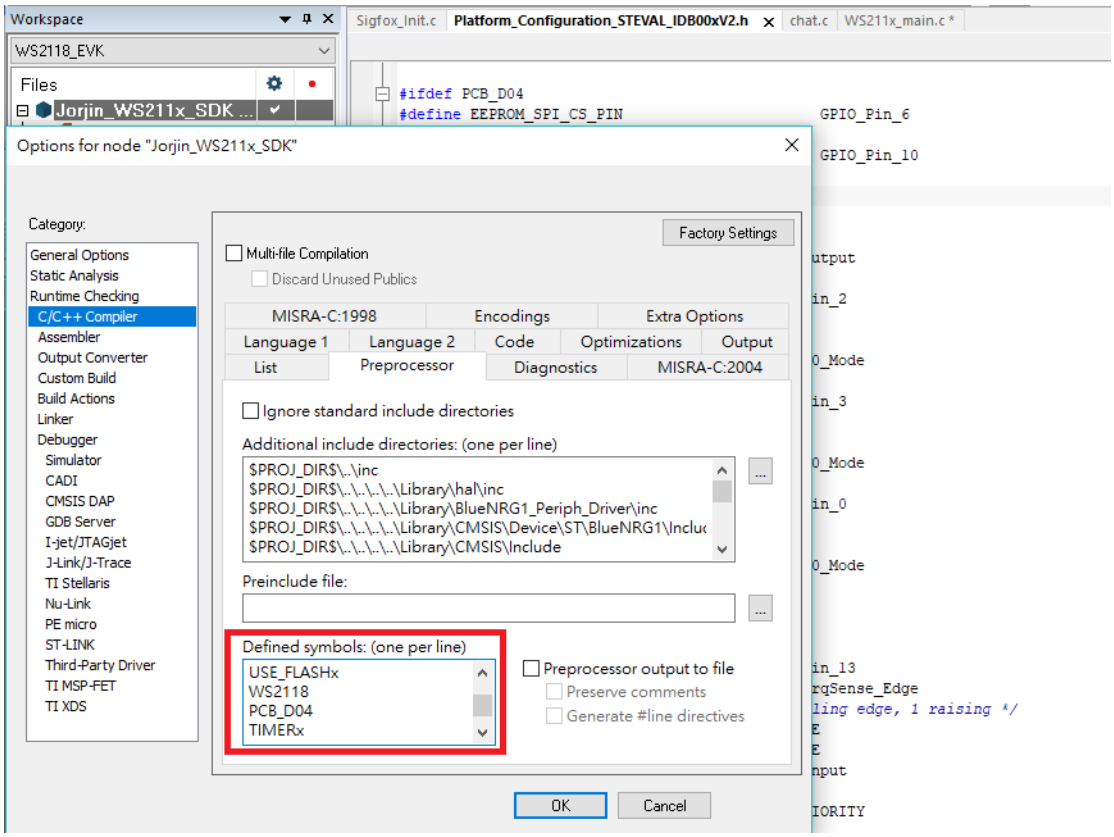


3.2.3 Switch your module/EVB type

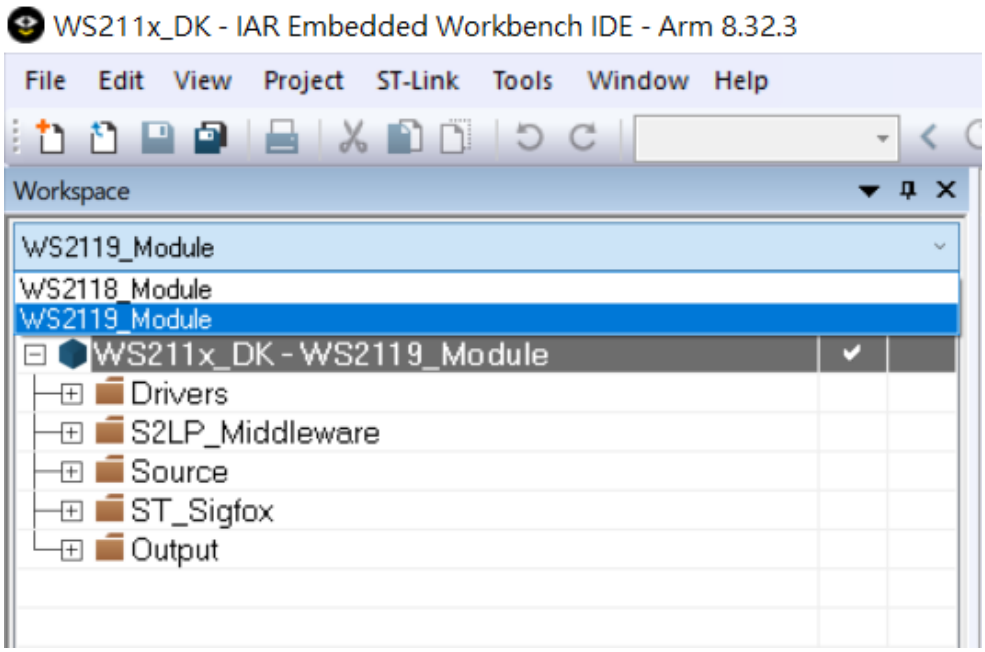
Select edit configurations to switch type. Each configuration has different defined symbol.







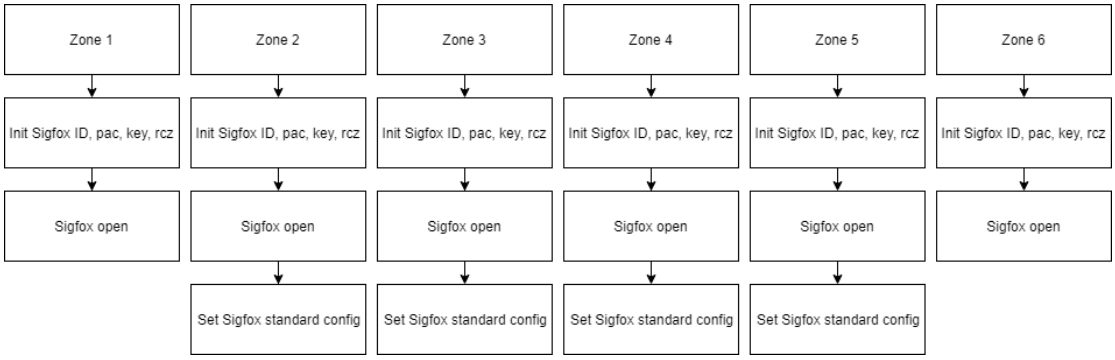
Quick switch configuration



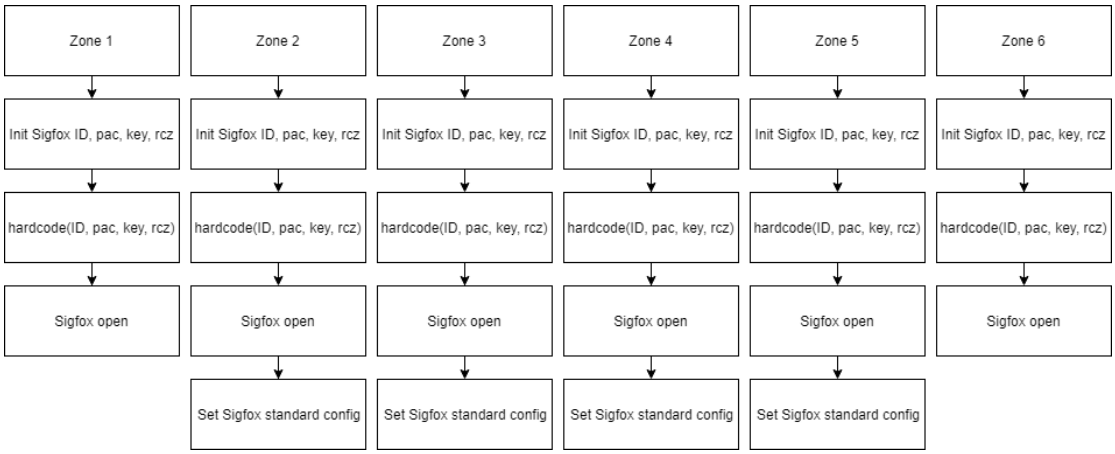
3.2.4 Sigfox/BLE functions

- Sigfox functions.

Initialize flow (SIGFOX information from Flash/EEPROM)



Initialize flow (HARDCODE)



Transmit the data to connected device via Sigfox. (12 bytes limit) (main.c) Must initialize Sigfox

- *SIGFOX_API_send_frame(sfx_u8 *customer_data,*

  sfx_u8 customer_data_length,

  *sfx_u8 *customer_response,*

  sfx_u8 tx_repeat,

  sfx_bool initiate_downlink_flag);

Set Sigfox private key (main.c) (Must register ID, key from Sigfox)

- *static uint8_t key[16]=*

  { 0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0xAA,0xBB,0xCC,0xDD,0xEE,0xFF};

- *enc_utils_set_key(key);*

Set Sigfox reduction power (main.c)

- *ST_RF_API_reduce_output_power(int16_t reduction);*

Set Sigfox private id (main.c)

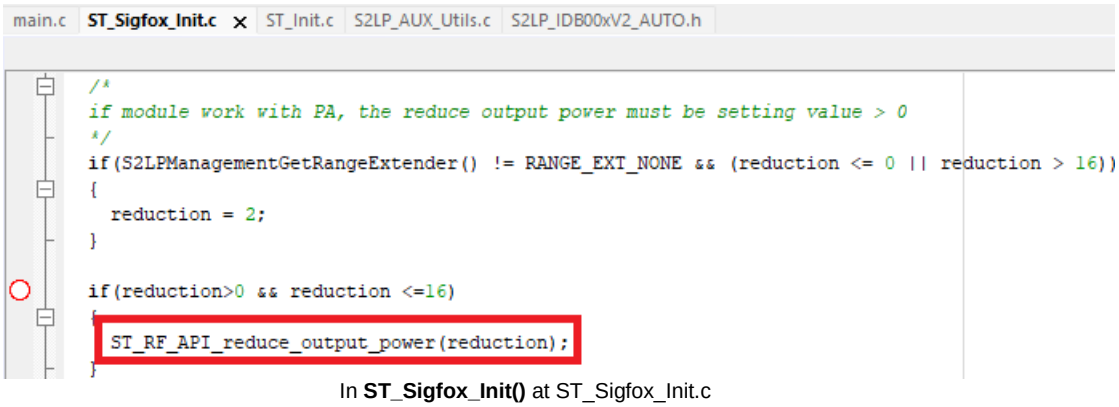
- *static uint32_t id=0xfedcba98;*
- *enc_utils_set_id(id);*

Switch the private (0)/public (1) key. (main.c)

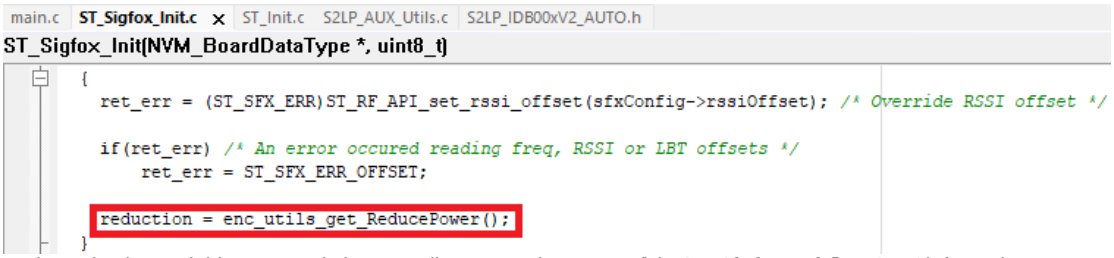
- use_public_key = 0;

NOTE: In **WS211x SDK 6.0.6 or earlier**, the following operations are needed for Sigfox compliant transmissions.

1. In **ST_Sigfox_Init.c**, adjust the reduction parameter passed to the **ST_RF_API_reduce_output_power()** function to limit the Sigfox Tx power within the spec at the selected RC.



In **ST_Sigfox_Init()** at ST_Sigfox_Init.c

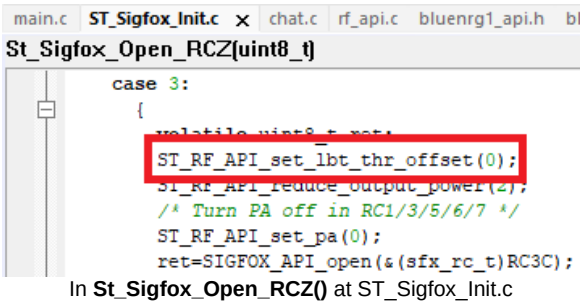


The reduction variable was read via enc_utils_get_ReducePower() in **ST_Sigfox_Init()** at ST_Sigfox_Init.c

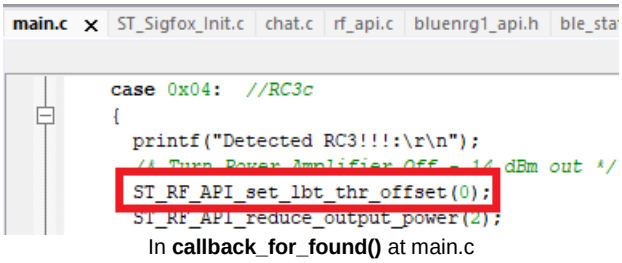
Suggested reduction values for RC1/RC3 are listed as follows:

TX (Uplink)	RC1	RC2	RC3	RC4
	868.13MHz	902.2MHz	923.2MHz	920.8MHz
Reduction	4	40	2	42

2. In **main.c** & **ST_Sigfox_Init.c**, set the offset parameters passed to the **ST_RF_API_set_lbt_thr_offset()** function to 0.

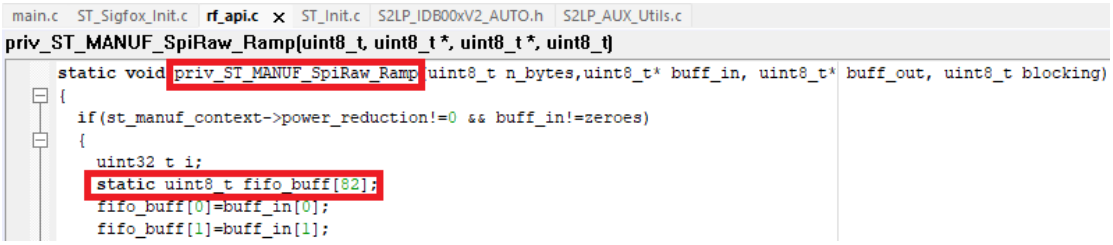


In **St_Sigfox_Open_RCZ()** at ST_Sigfox_Init.c



In **callback_for_found()** at main.c

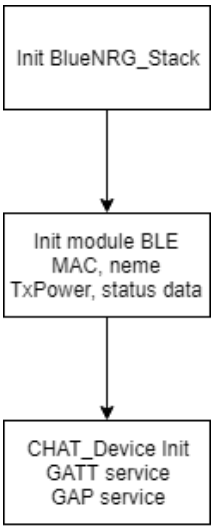
3. In **rf_api.c**, change the local variable **fifo_buff []** in the **priv_ST_MANUF_SpiRaw_Ramp()** function to a static one to avoid system RAM glitch.

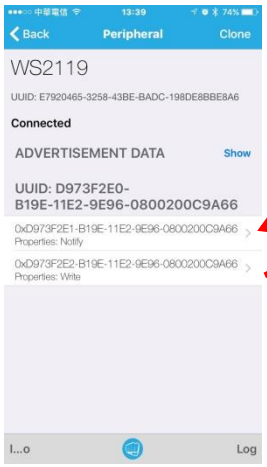


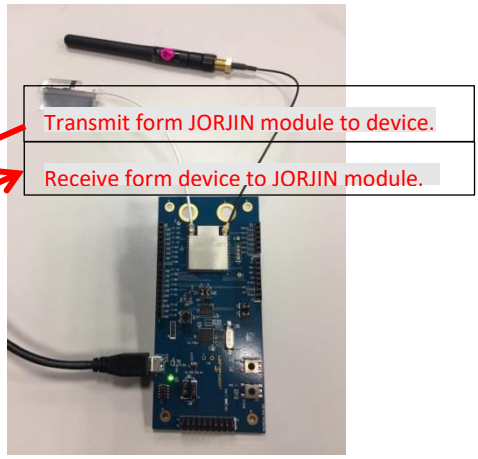
- BLE functions.

Initialize flow

(Information from Flash/EEPROM)







BLE MAC (chat.c)

- *static uint8_t bdmacaddr [6]={0x00, 0x19, 0x94, 0xFF, 0xFF, 0xFF};*

BLE name (8 bytes limit) (chat.c)

- *static uint8_t newname [8]={'J', 'O', 'R', 'J', 'I', 'N', 0x00,0x00};*

BLE local_ name (chat.c)

- *uint8_t local_name[] =*
  {AD_TYPE_COMPLETE_LOCAL_NAME,'J','o','r','j','i','n','_','W','S','2','1','1','x'};

(Example in 3.2.6.2)

BLE Tx power (chat.c)

- *aci_hal_set_tx_power_level(1, 4);*

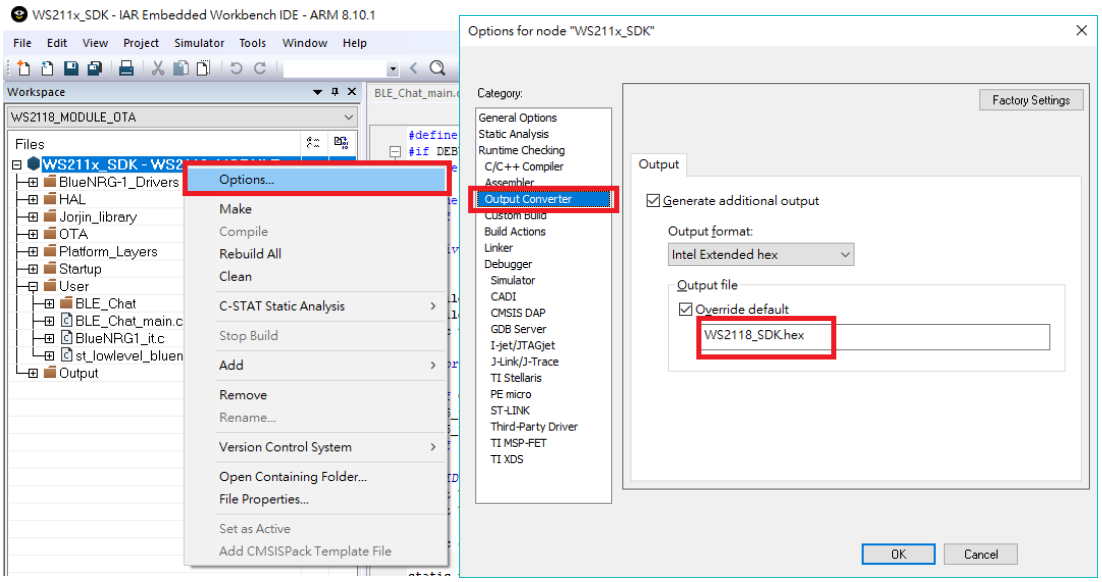
(Example in 3.2.6.3)

Transmit and receive uint8 data to connected device via BLE by UART. (chat.c)

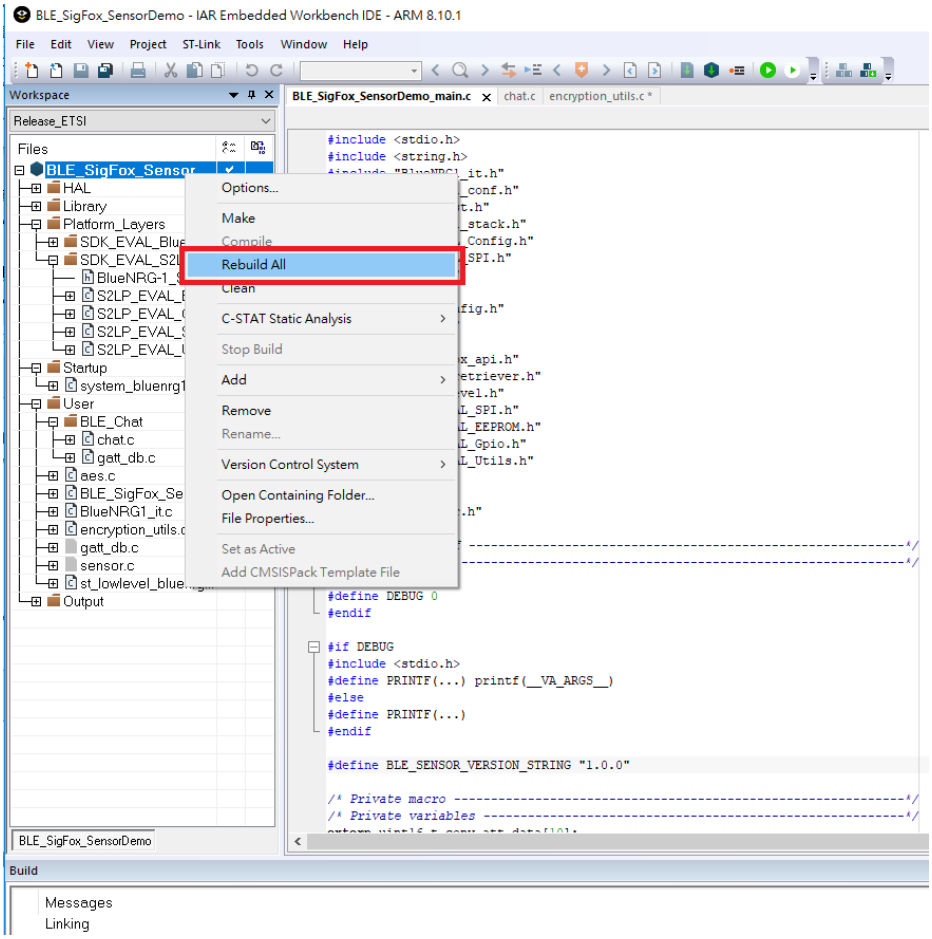
- *Process_InputData(uint8_t* data_buffer, uint16_t Nb_bytes)* (Example in 3.2.6.1)

3.2.5 Build code and file location

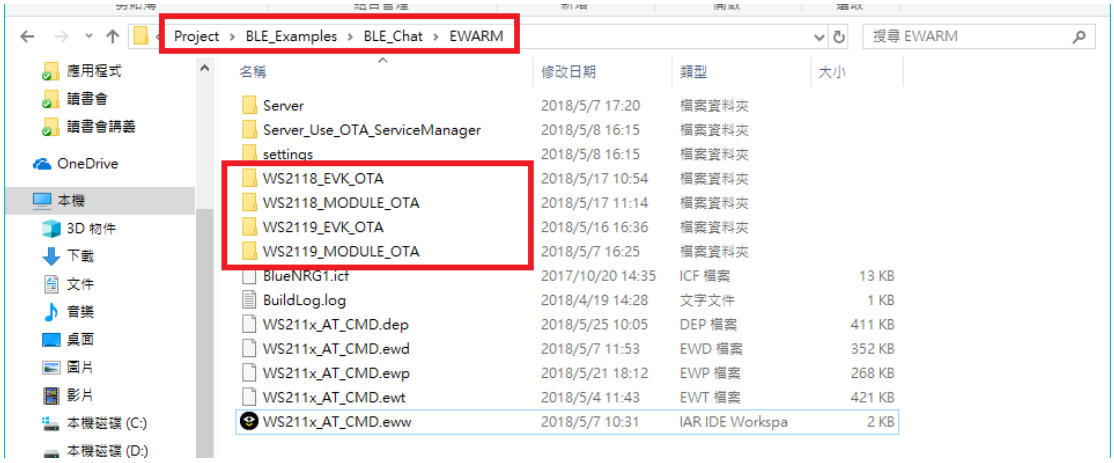
1. Change output file name. “Option→ Output Converter→Output file”



2. Build code



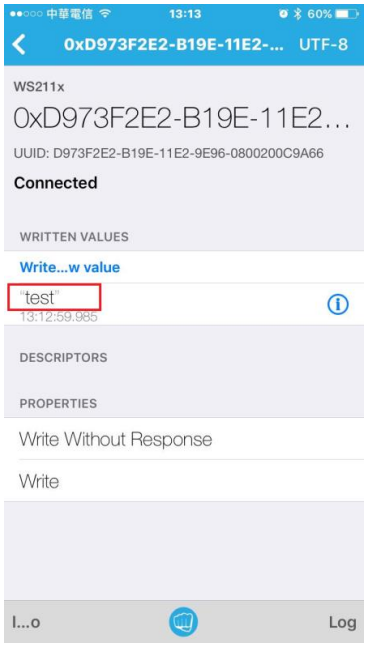
3. File location

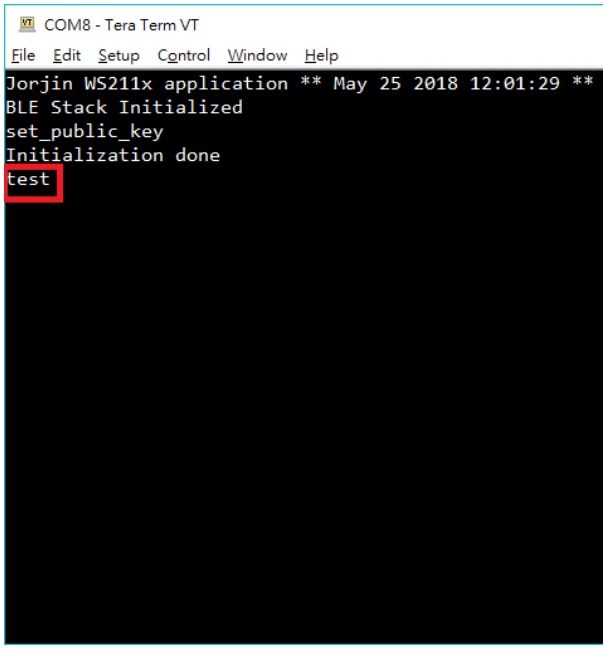


3.2.6 Function Testing

1. Written value via BLE to module.

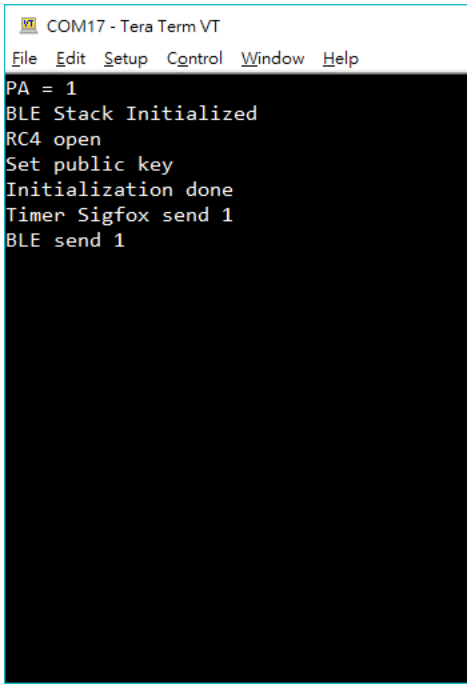
Application → module.

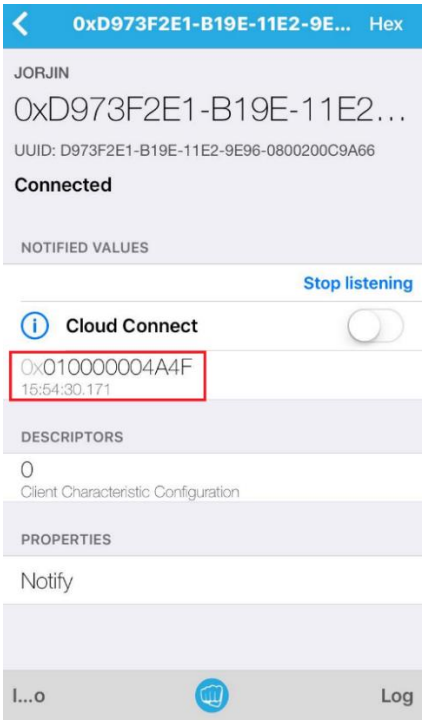




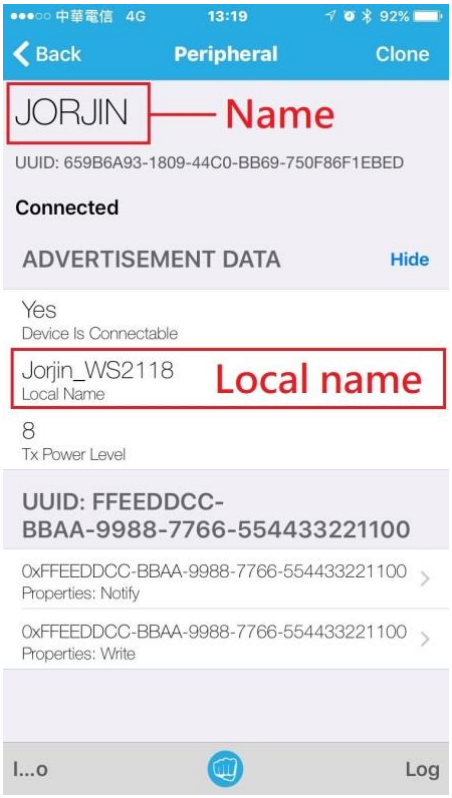
Module → application.

Push button or enable timer (Implement in while loop)





2. BLE name and local name



NOTE: The local name could only be changed by modifying the **local_name** variable in **Make_Connection()** at **chat.c** in the SDK. But, in the new AT command FW, it can also be changed through the **AT+NAME** command.

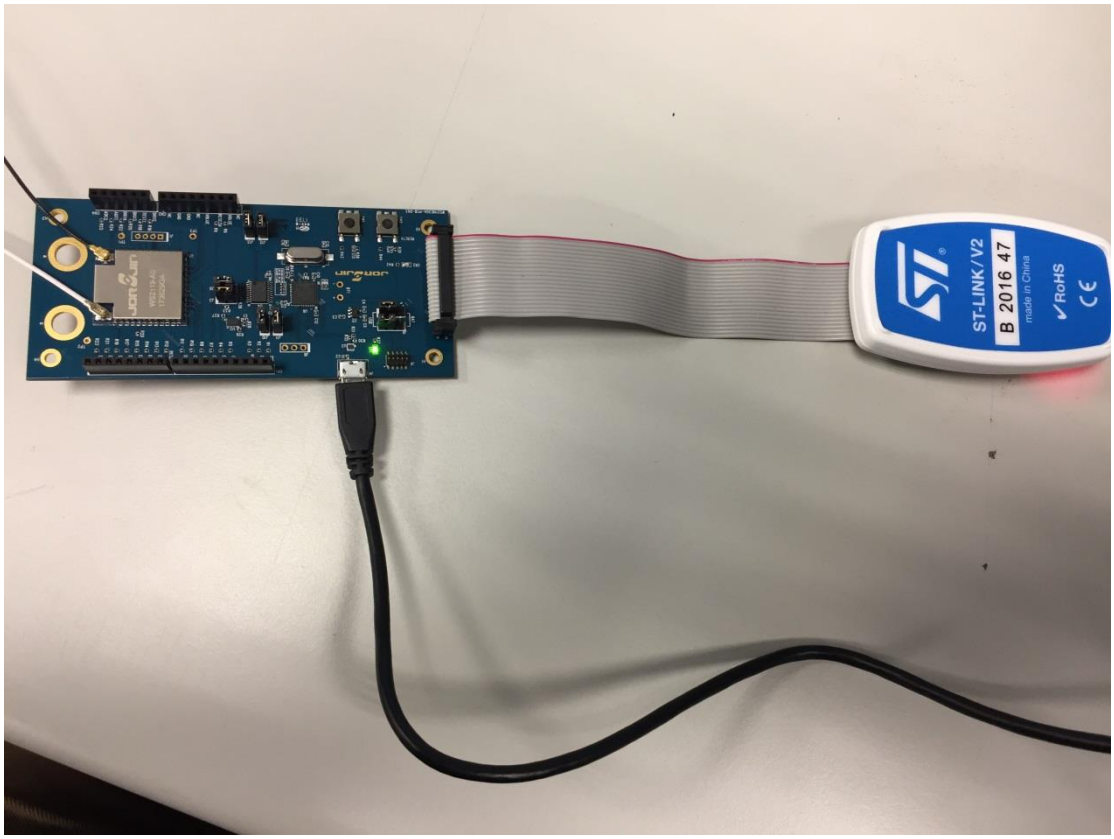
3. BLE Tx power parameter (bluewnrg1_api.h)

```
  * @param En_High_Power Enable High Power mode
  * Values:
  - 0x00: Standard Power
  - 0x01: High Power
  * @param PA_Level Power amplifier output level. Output power is indicative and it depends on the PCB layout and associated components
  * Values:
  - 0x00: -18 dBm (Standard Power), -14 dBm (High Power)
  - 0x01: -15 dBm (Standard Power), -11 dBm (High Power)
  - 0x02: -12 dBm (Standard Power), -8 dBm (High Power)
  - 0x03: -9 dBm (Standard Power), -5 dBm (High Power)
  - 0x04: -6 dBm (Standard Power), -2 dBm (High Power)
  - 0x05: -2 dBm (Standard Power), 2 dBm (High Power)
  - 0x06: 0 dBm (Standard Power), 4 dBm (High Power)
  - 0x07: 5 dBm (Standard Power), 8 dBm (High Power)
  * @retval Value indicating success or error code.
*/
tBleStatus aci_hal_set_tx_power_level(uint8_t En_High_Power,
                                      uint8_t PA_Level);
```

4 FW OTA

4.1 ST-LINK

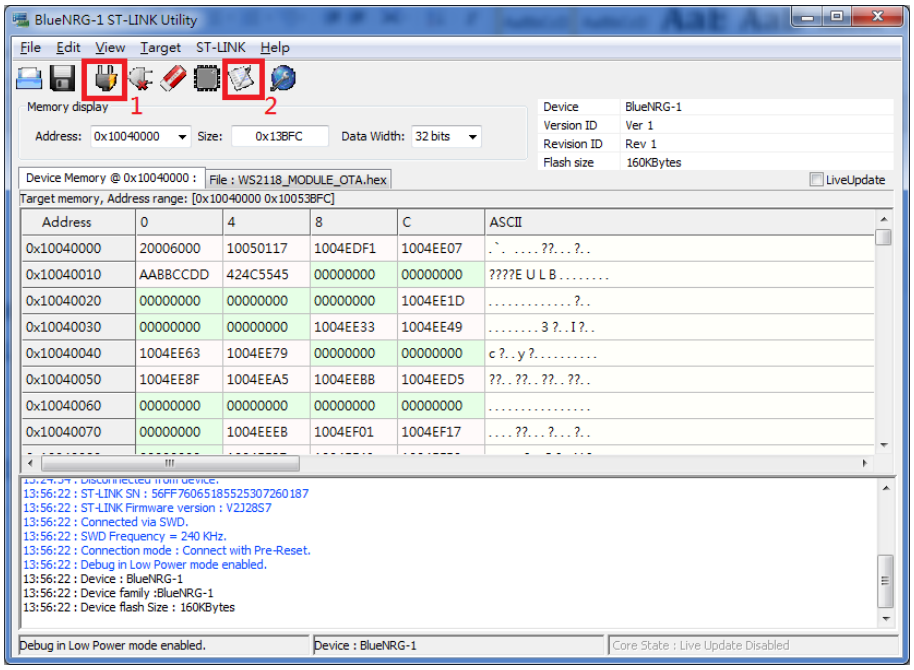
WS211x setup, connect power and ST-link then open BlueNRG-1 ST-LINK Utility.



(EVB demo, module has to connect correct pin yourself)

1. Connect to the target

2. Program verify (OTA_ServiceManager_1.0.0)

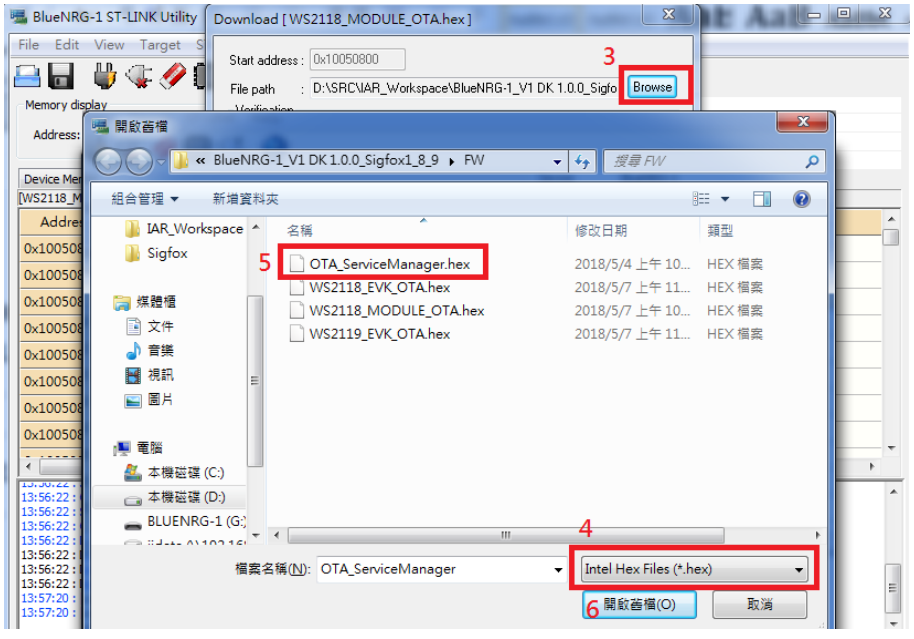


3. Browse and select file

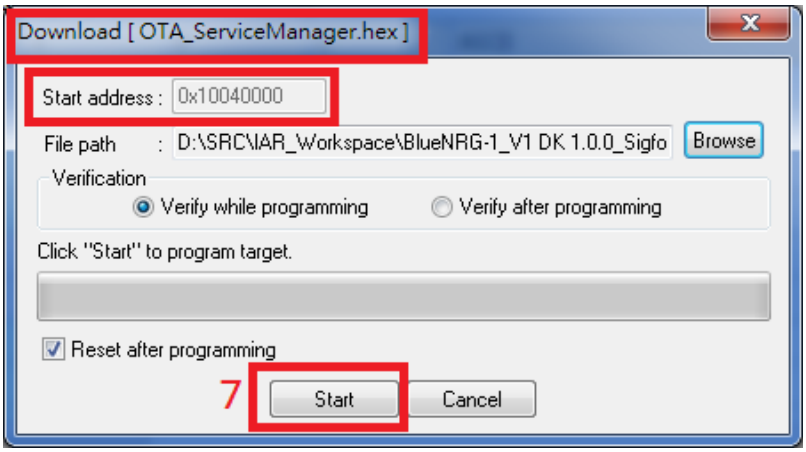
4. Setting filter "*.hex" files

5. Select “OTA_ServiceManager_1.0.0.hex”

6. Open file.



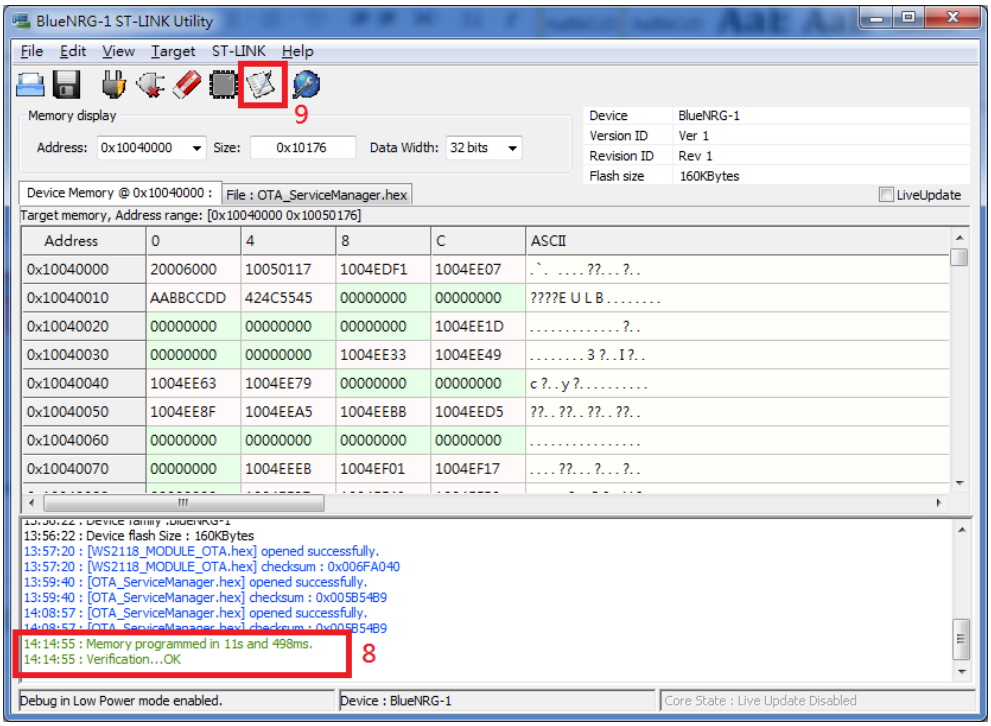
7. Check file、start address and start programming



8. Check success message

9. Program verify (WS211x firmware)

In OTA_ServiceManager FW also can use cell phone to program FW.

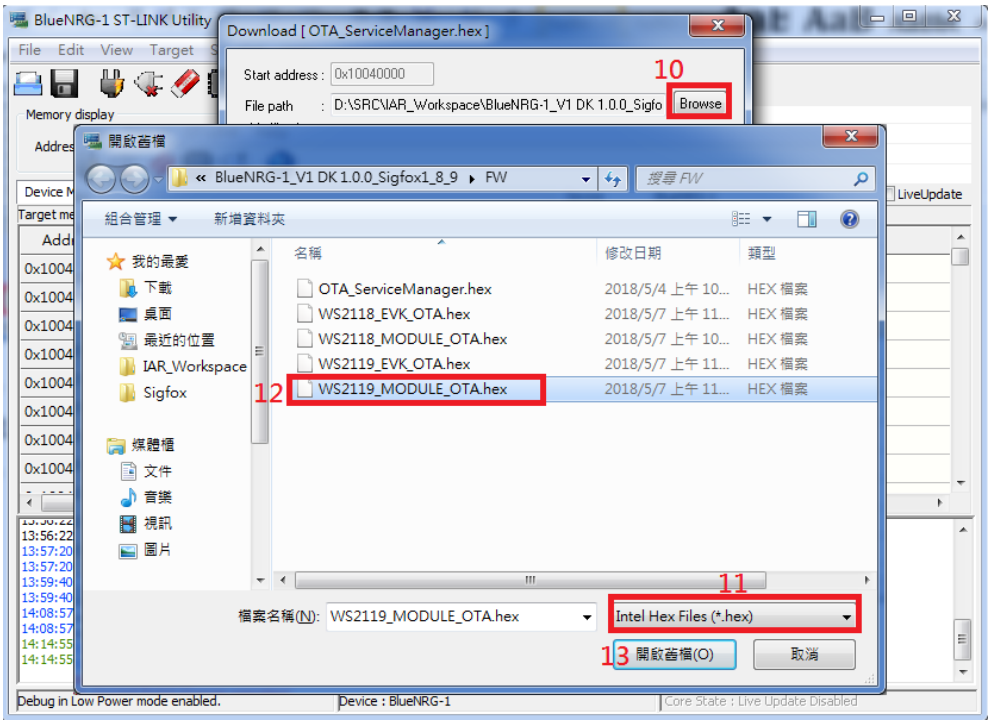


10. Browse and select file

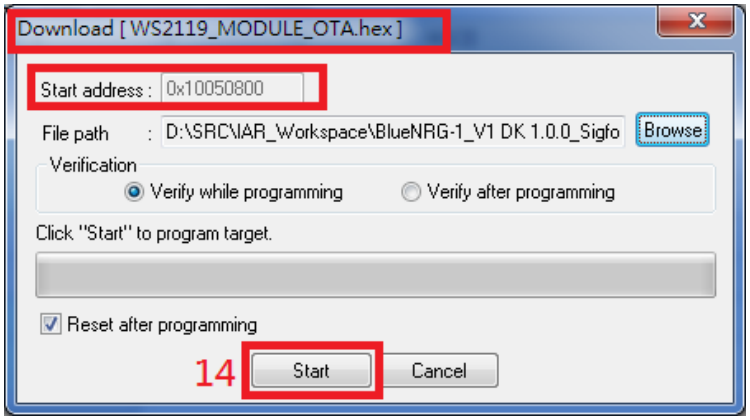
11. Setting filter "*.hex" files

12. Select “(WS211x firmware).hex” **(Must be *.hex file)**

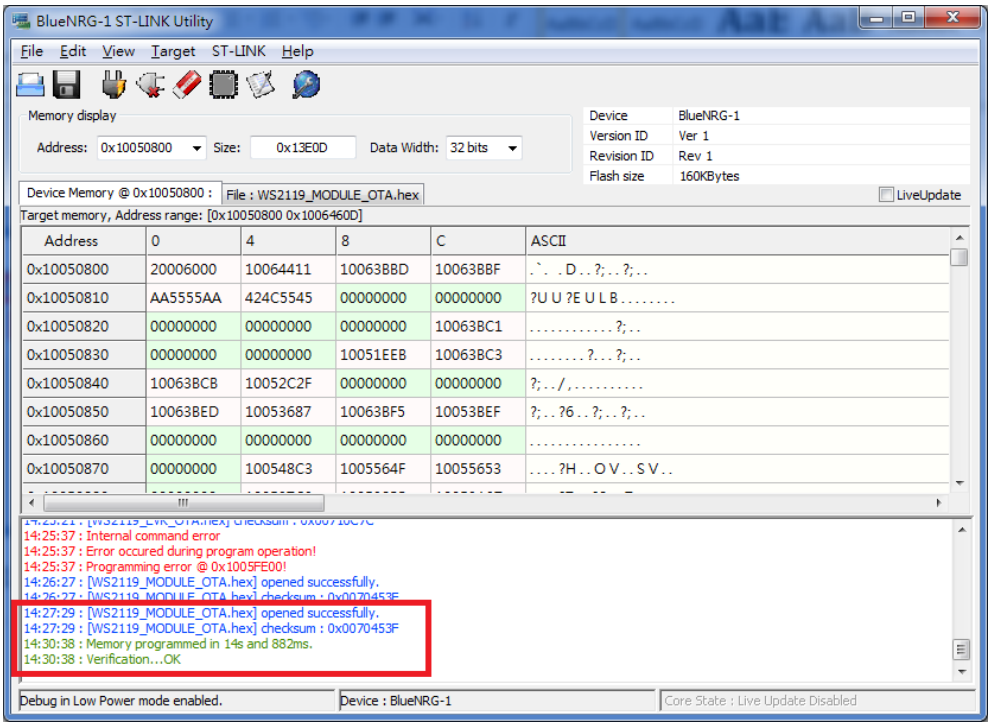
13. Open file.



14. Check file、start address and start programming

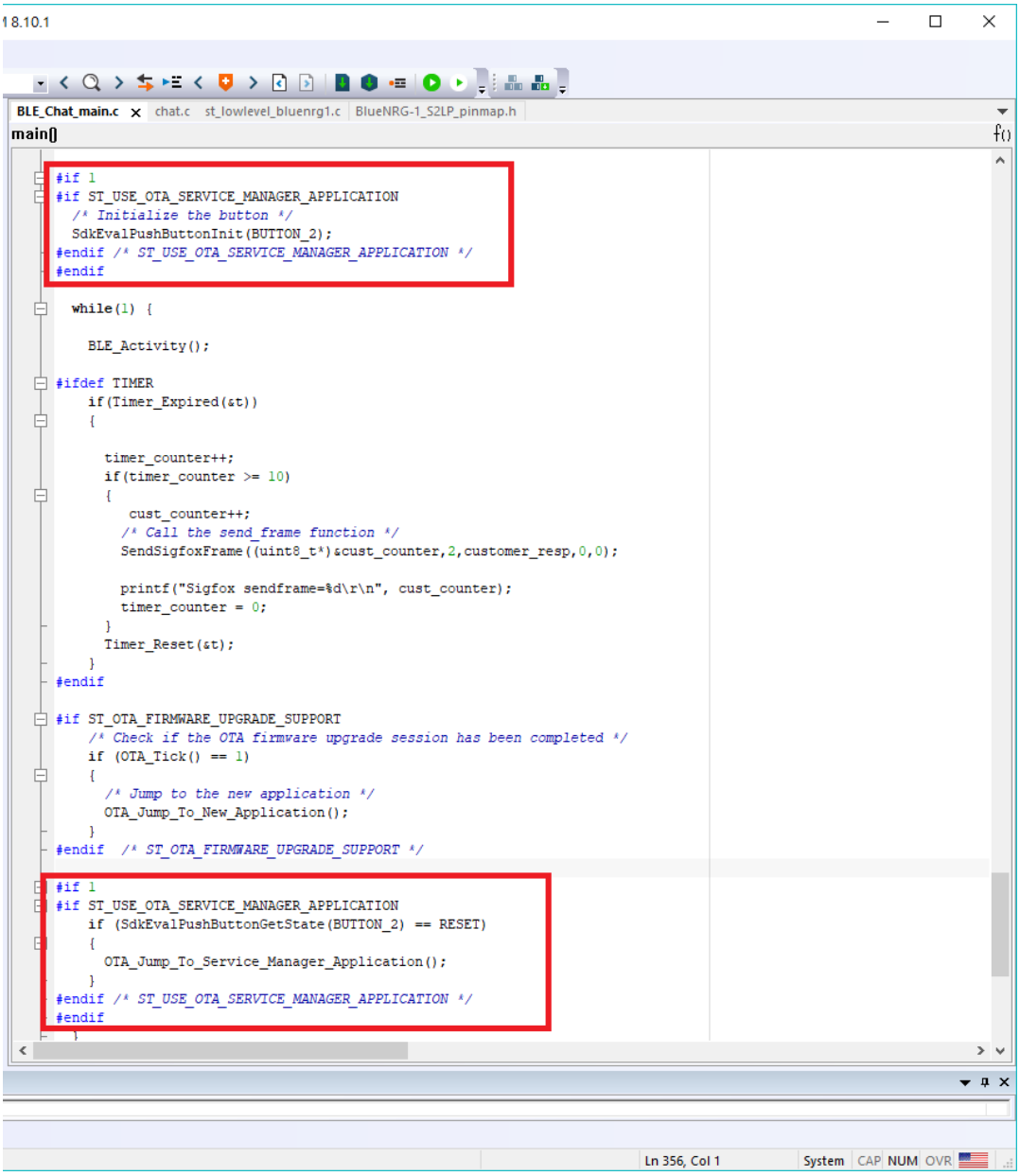


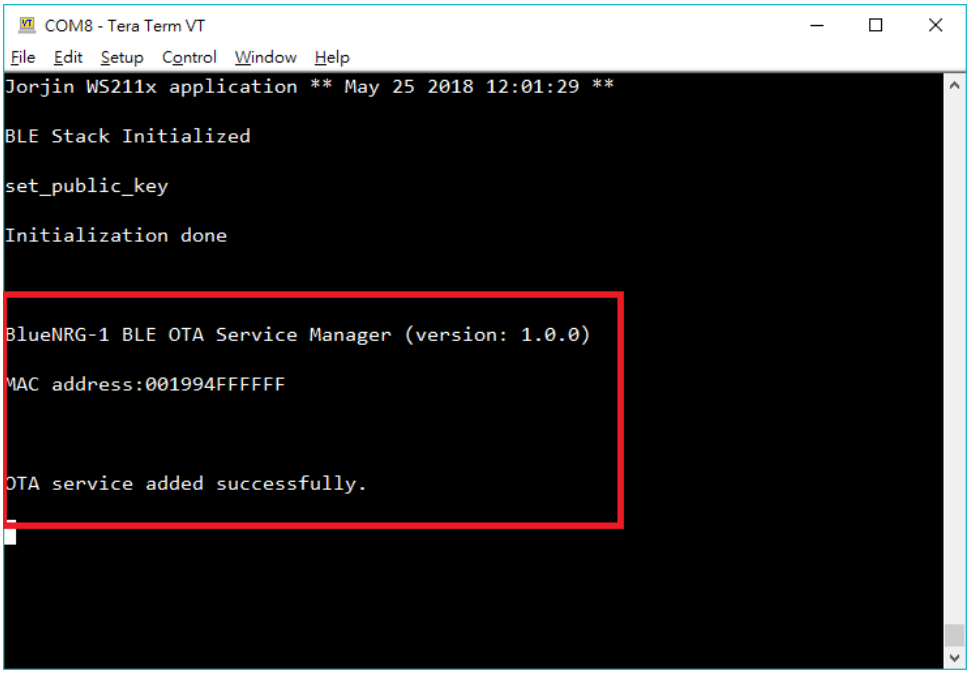
15. Check success message



4.2 CELL PHONE

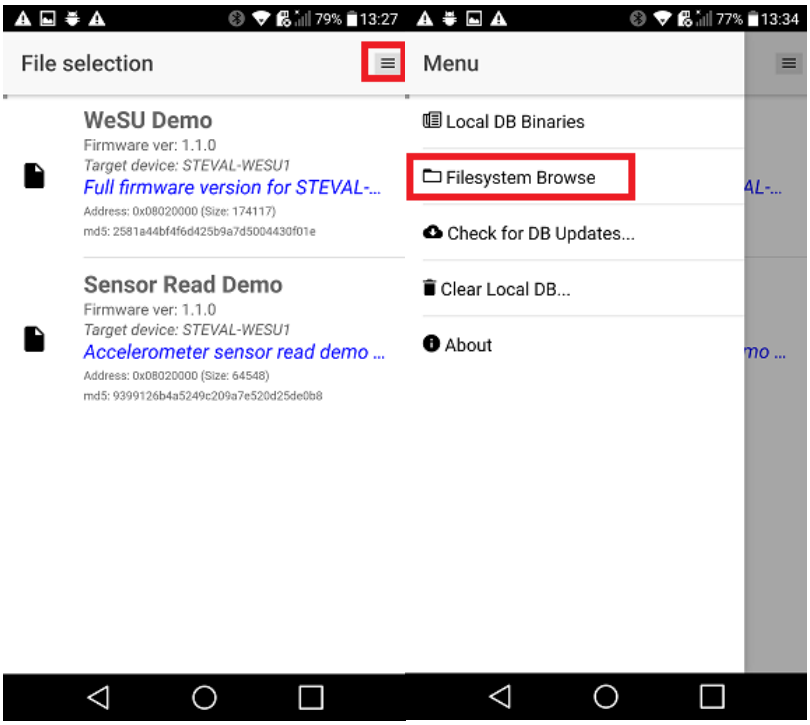
1. Press SW1 to enter OTA mode. Please ensure SW1 is setting "OTA_Jump_To_Service_Manager_Application".



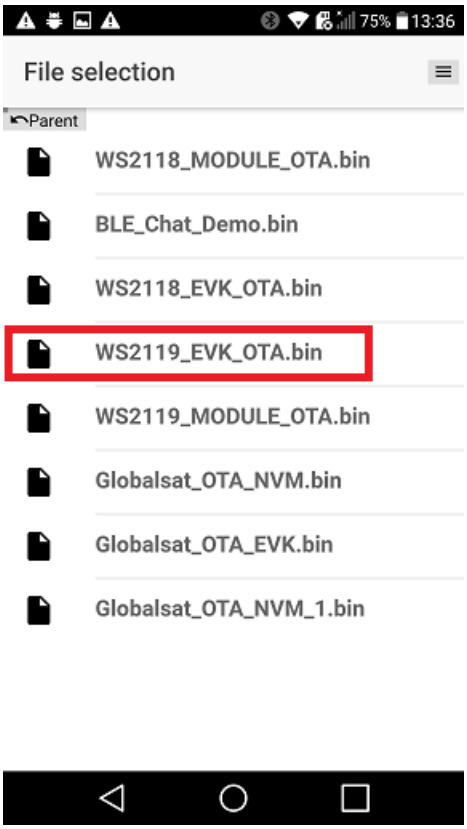


2. Open cell-phone APP “ST BlueDFU”

3. Press “Menu” and select “Filesystem Browse”

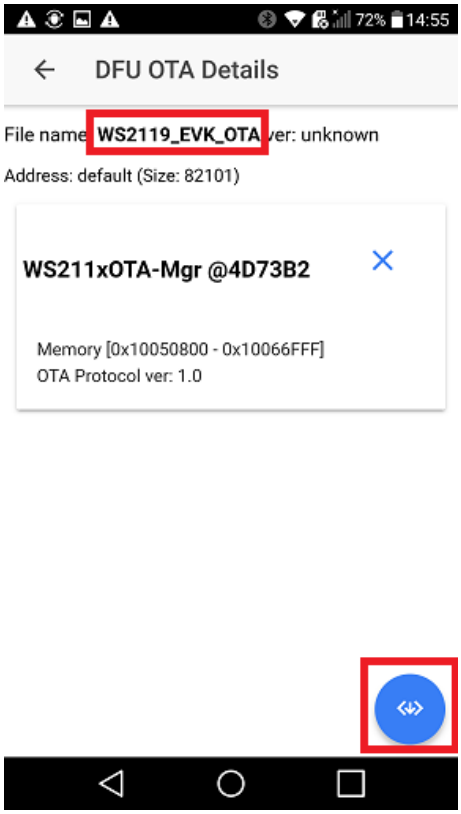


4. Select “(WS211x firmware).bin” **(Note: Must be *.bin file.)**

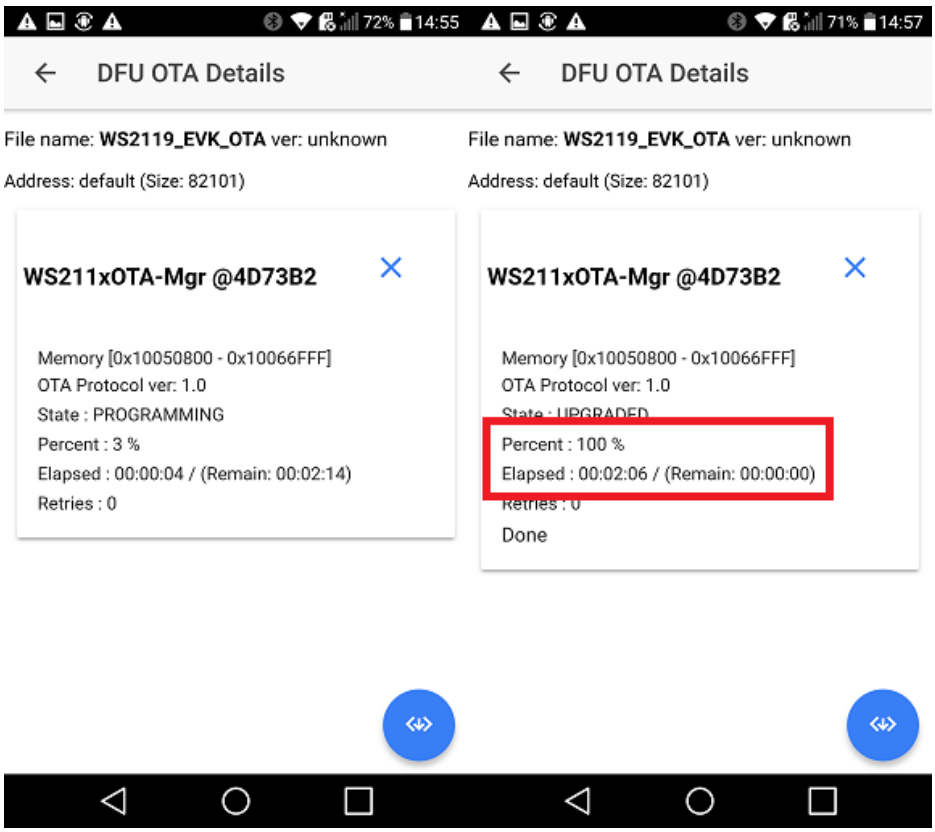


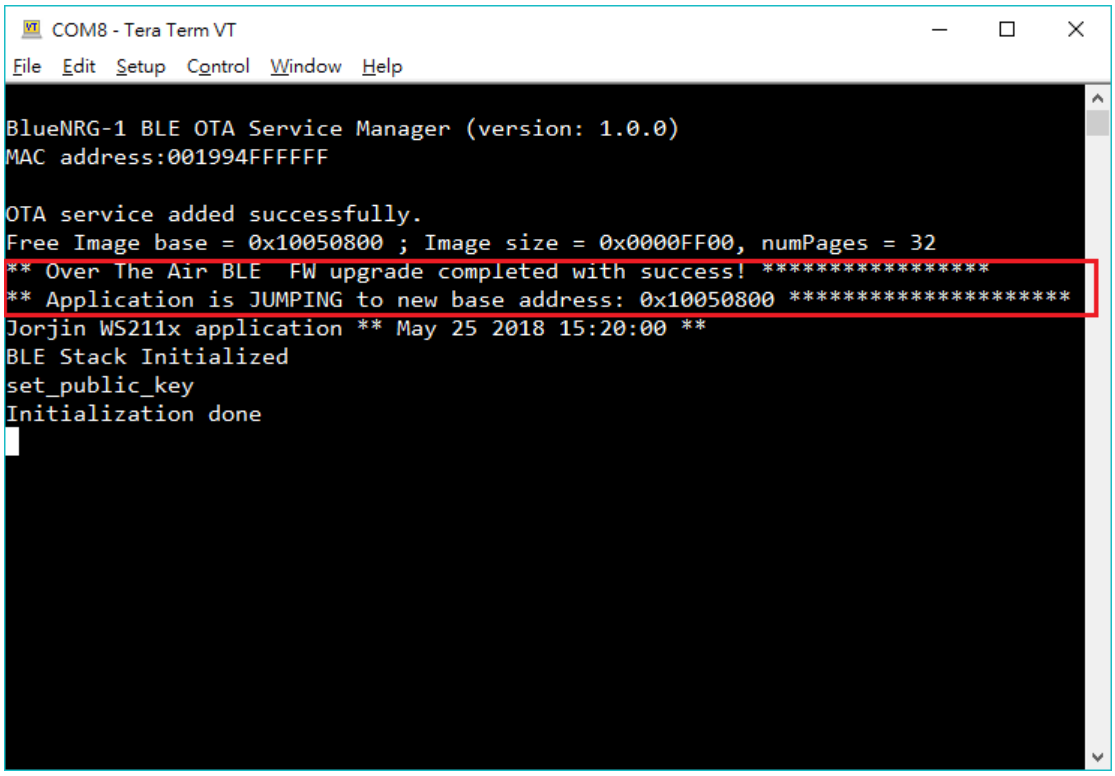
5. Connect WS211x device and press “Play” icon

6. Check file name and press “Download” icon



7. Wait finish.





4.3 RETURN OTA SERVICE MANAGER

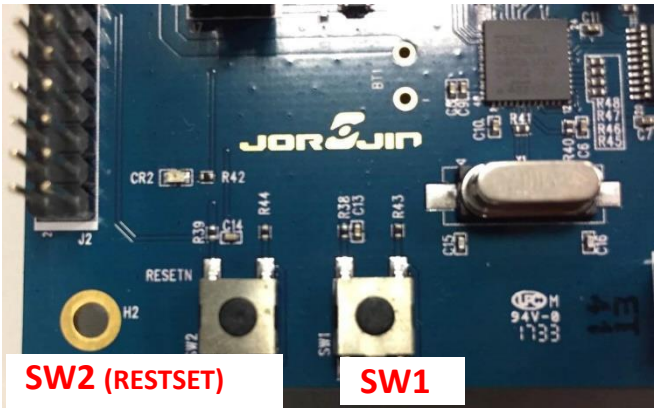
If use EVB and define “TIMER” to press SW1 to return OTA service manager.

Sample code as following.

main.c | S2LP_IDB00xV2_AUTO.h | nvm_api.c | ST_Init.c | chat.c | chat.h | ST_Sigfox_Init.c | rf_api.c | mcu_api_bluenrg1.c | jorjin_sigfo

Appli_Exti_CB(uint16_t)

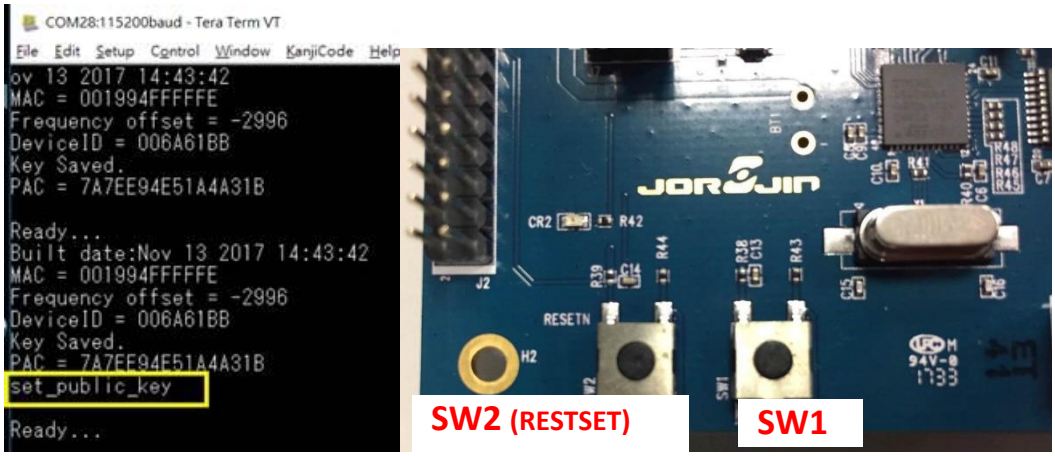
```
#ifdef TIMER
    if(Timer_Expired(&t))
    {
      timer_counter++;
      if(timer_counter >= 30)
      {
        LedBlink(LED2, 5);

        /* If the interrupt is raised, prepare the buffer to send with a 4-bytes counter */
        cust_counter++;

        for(uint8_t i=0;i<4;i++)
          customer_data[i]=(uint8_t)(cust_counter>>((3-i)*8));

        /* Call the send_frame function */
        SIGFOX_API_send_frame(customer_data,4,customer_resp,2,0);

        LedBlink(LED2, 6);
        timer_counter = 0;
      }
      Timer_Reset(&t);
    }

    if(but_pressed)
    {
#if ST_USE_OTA_SERVICE_MANAGER_APPLICATION
      printf("Enter OTA\r\n");
      Clock_Wait(100);
      OTA_Jump_To_Service_Manager_Application();
#endif /* ST_USE_OTA_SERVICE_MANAGER_APPLICATION */
    }
#else
```



5 HOW TO EVALUATE WITH SIGFOX FUNCIOTN

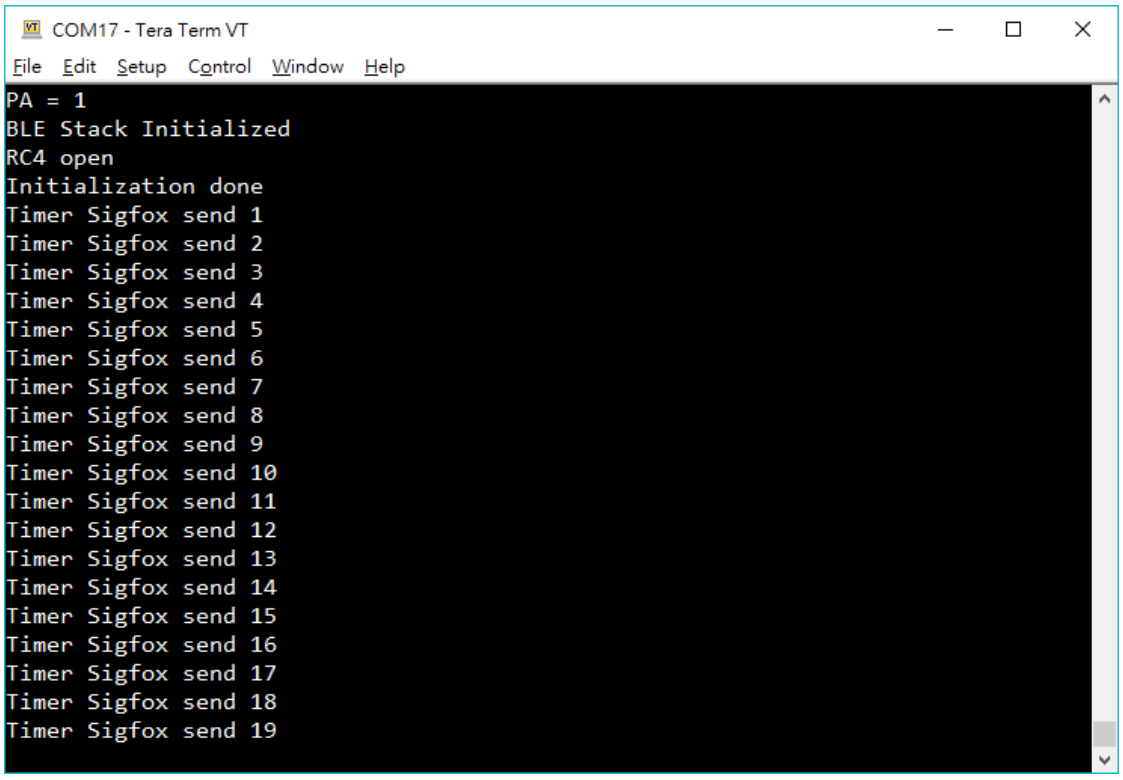
5.1 USE PUBLIC KEY (USE SDR DONGLE)

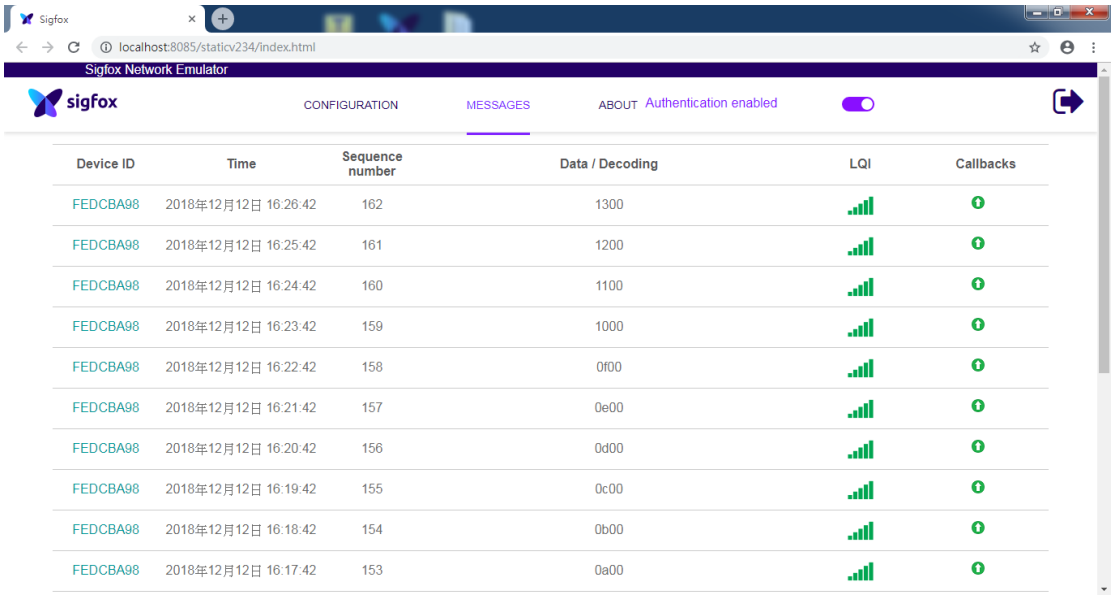
1. Hold SW1
2. Pressed and released SW2
3. Released SW1
4. Check the log as below



5. Defined TIMER to send frame to sigfox cloud. If loop to control send interval.

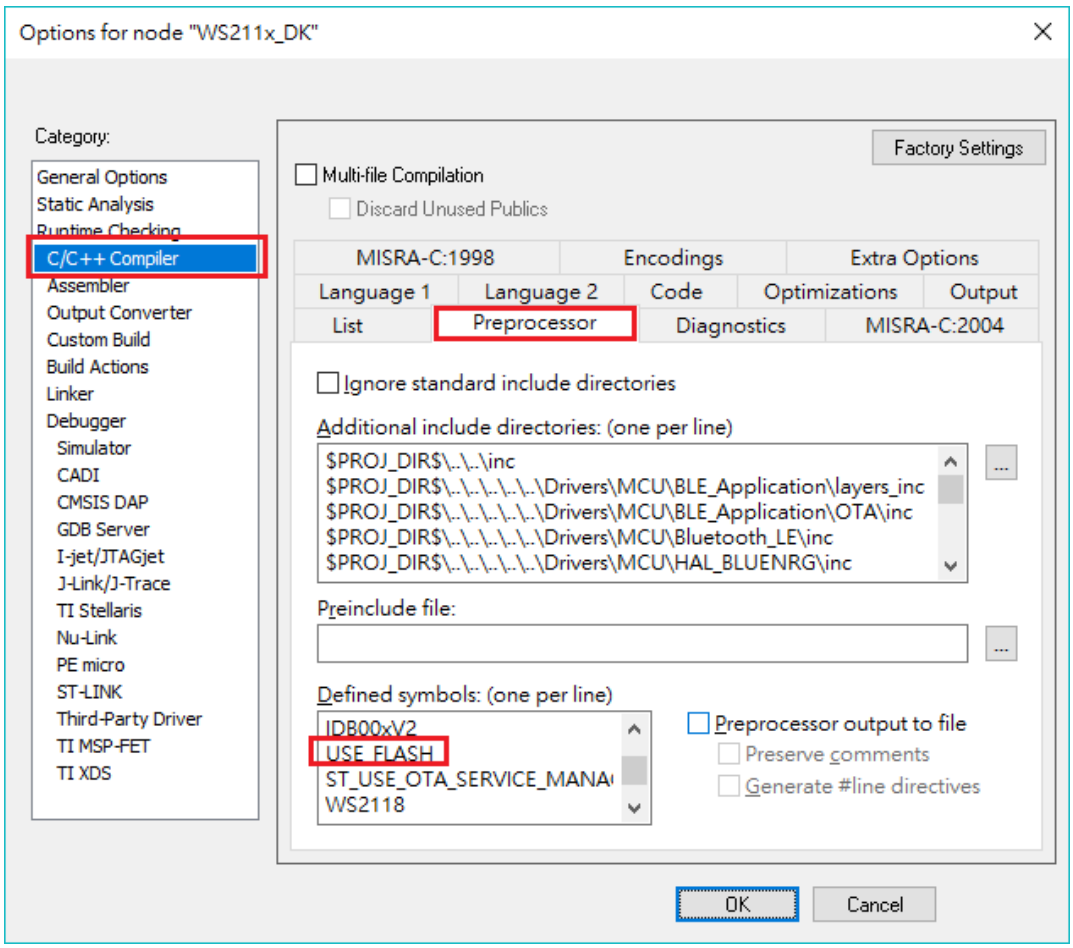
```
#ifdef TIMER
    if(Timer_Expired(&t))
    {
      timer_counter++;
      if(timer_counter >= 30)
      {
        LedBlink(LED2, 5);

        /* If the interrupt is raised, prepare the buffer to send with a 4-bytes counter */
        cust_counter++;

        for(uint8_t i=0;i<4;i++)
          customer_data[i]=(uint8_t)(cust_counter>>((3-i)*8));

        /* Call the send_frame function */
        SIGFOX_API_send_frame(customer_data,4,customer_resp,2,0);

        LedBlink(LED2, 6);
        timer_counter = 0;
      }
      Timer_Reset(&t);
    }

    if(but_pressed)
    {
#if ST_USE_OTA_SERVICE_MANAGER_APPLICATION
      printf("Enter OTA\r\n");
      Clock_Wait(100);
      OTA_Jump_To_Service_Manager_Application();
#endif /* ST_USE_OTA_SERVICE_MANAGER_APPLICATION */
    }
#else
```

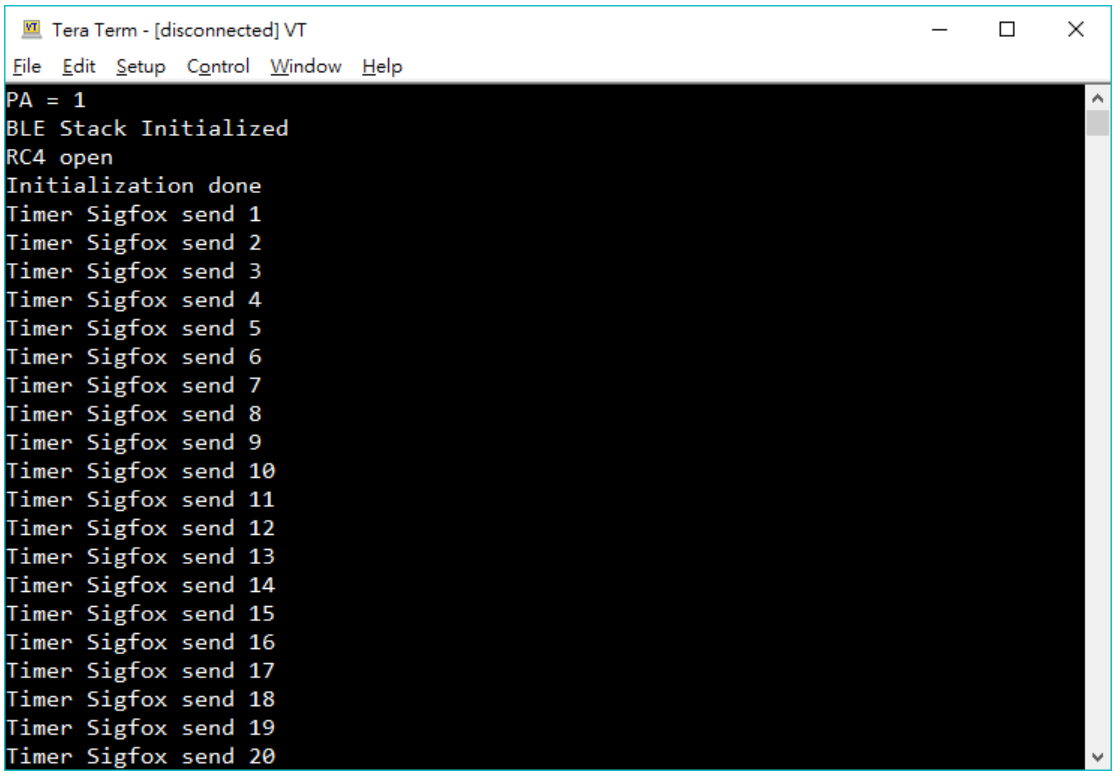


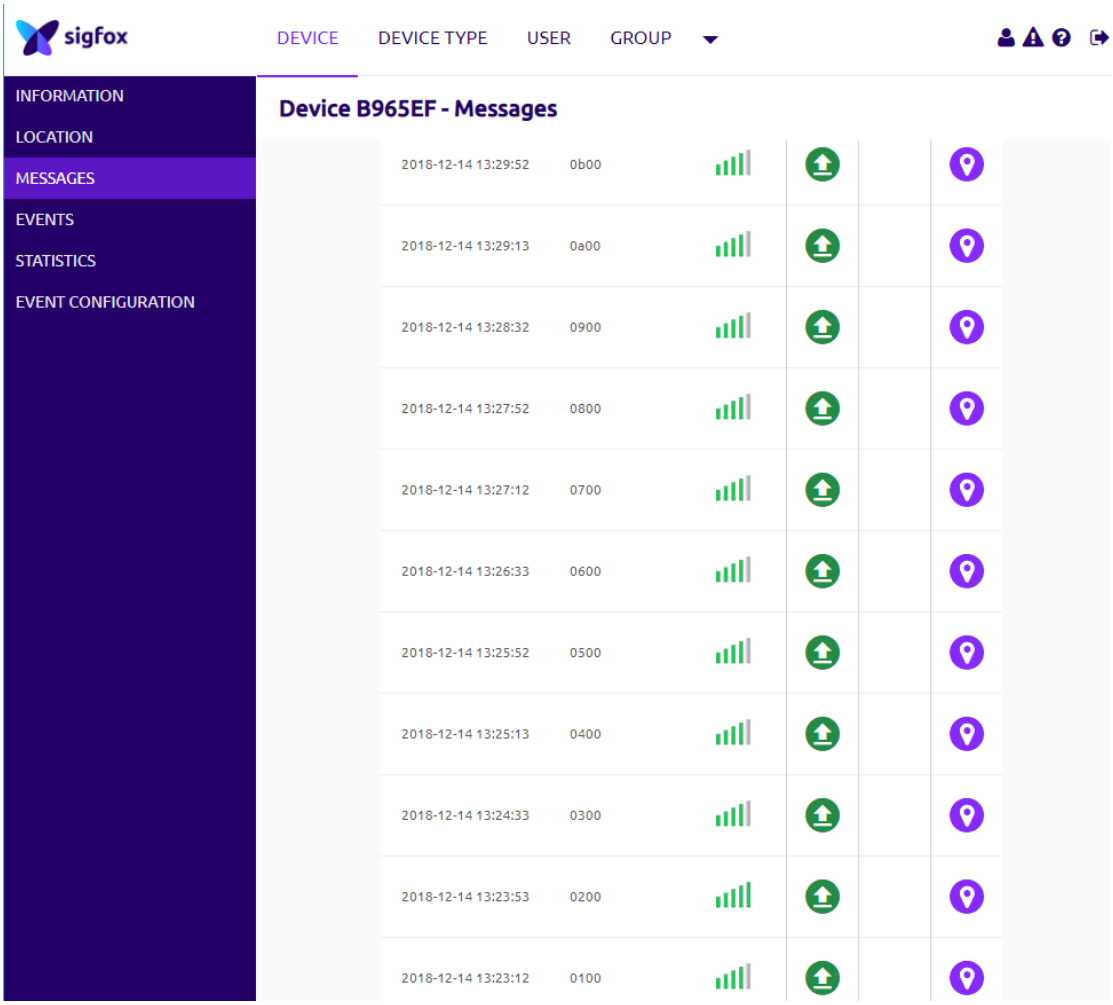


5.2 USE PRIVATE KEY

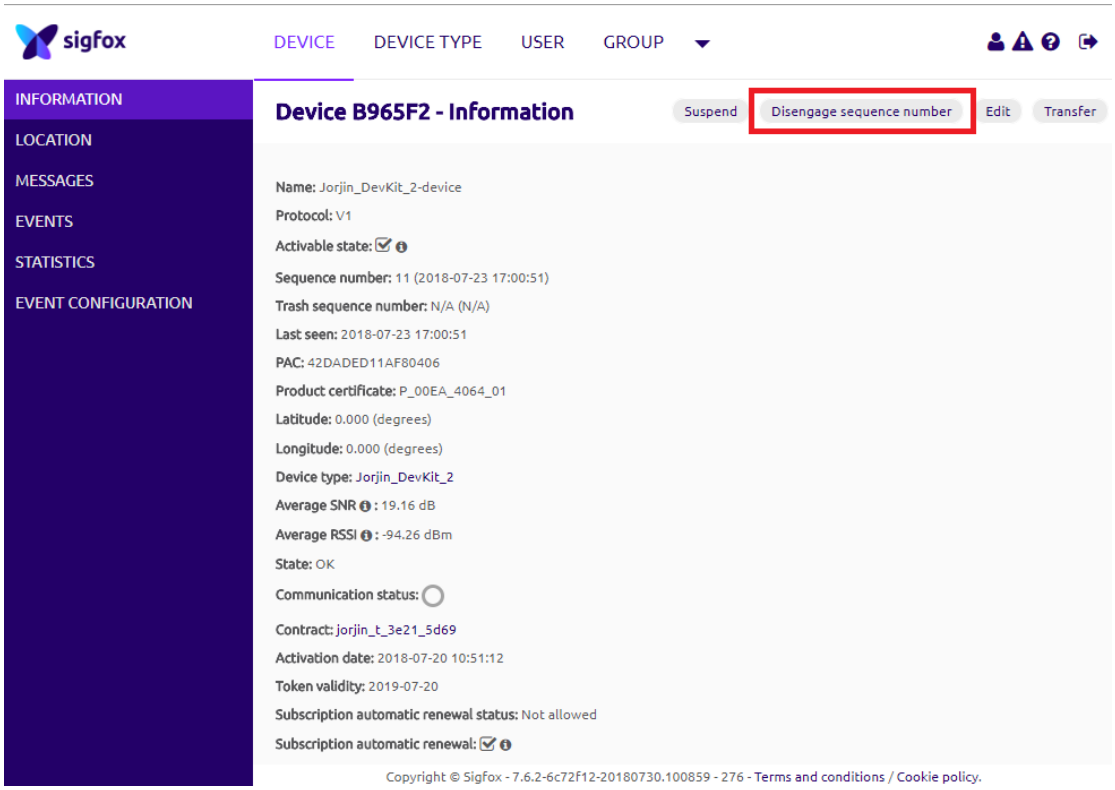
Both EEPROM/flash have stored SIGFOX information (id, pac and key etc.), depend on module or EVB you use to define symbol to read SIGFOX information (id, pac and key etc.) which Jorjin provide to simulate sigfox function.







If show information below, please click " Disengage sequence number" on Sigfox backend. Then the sequence number will match.



5.3 READ SIGFOX ID AND PAC

This section is based on SDK 3.32+ version. If you are using old SDK. Pls upgrade.

Don't define “HARDCODE” to read module data SIGFOX information, UART will show ID and PAC.