Uplynx Software Development Kit

Users Manual

Elite Semiconductor Memory Technology Inc.

Important Notice

Trademarks

Contact Information

Elite Semiconductor Memory Technology Inc.

No.23, Industry E Rd. IV Science-Based Industrial Park, Hsinchu 300, Taiwan, R.O.C.

TEL: +886-3-5781970

FAX: +886-3-5644432

Revision Control

Ver.	Date	Description of Change	Approved
1.0	23/8/2018	Document First version	Charles Lee

Table of Contents

1 *Uplynx Software Development Kit Overview*

The Uplynx XS8001 and XS8001-T carry a 32-bit Andes Core MCU with 128kB flash and 24kB SRAM for user applications to run including the Sigfox API. To allow maximum flexibility and computation power for a user's application, this document describes the basic flow to install the development environment, software source code and libraries, and how to run through the whole bring up procedure.

This document is based upon SDK version code: ESMT UPLYNX SOC V1.23

ESMT recommends that customers check for the latest SDK version code with the ESMT FAE Team.

The complete software development environment consists of the following components:

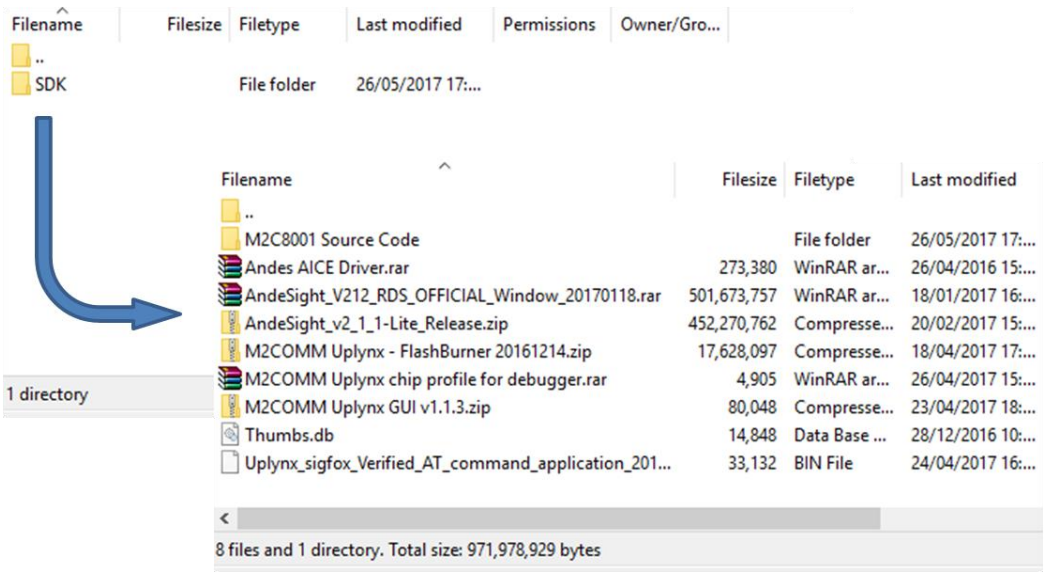
Tool	Description
Development Platform	AndeSight_v212_RDS (V2.1.2 RDS Version) or AndeSight_v2_1_1-Lite_Release.zip (evaluation version with code size limitation)
Debug Tool/Flash programmer	AICE-MCU
Burner Tool	ESMT Flash Burner software
Serial Number	Serial Number(s) to activate AndeSight RDS v2.1.2
Evaluation board	BSE8001-0x evaluation board

2 *Software Download*

To download the components, users need to access the ESMT's FTP server:

Please contact ESMT's FAE team to arrange access.

The folder structure in the FTP server is as shown below:



Filename Name	Description
XS8001 Source Code	ESMT UPLYNX SOC V1.23.rar
Andes AICE Driver.rar	Windows Driver for Andes AICE-MCU
AndeSight_V212_RDS_OFFICIAL_ Window_20170118.rar	AndeSight V2.1.2 Installation package
AndeSight_v2_1_1-Lite_Release.zip	AndeSight v2.1.1 evaluation version access via http://www.knect.me/#d
ESMT Uplynx – FlashBurner 20161214.zip	ESMT Flash Programmer Application

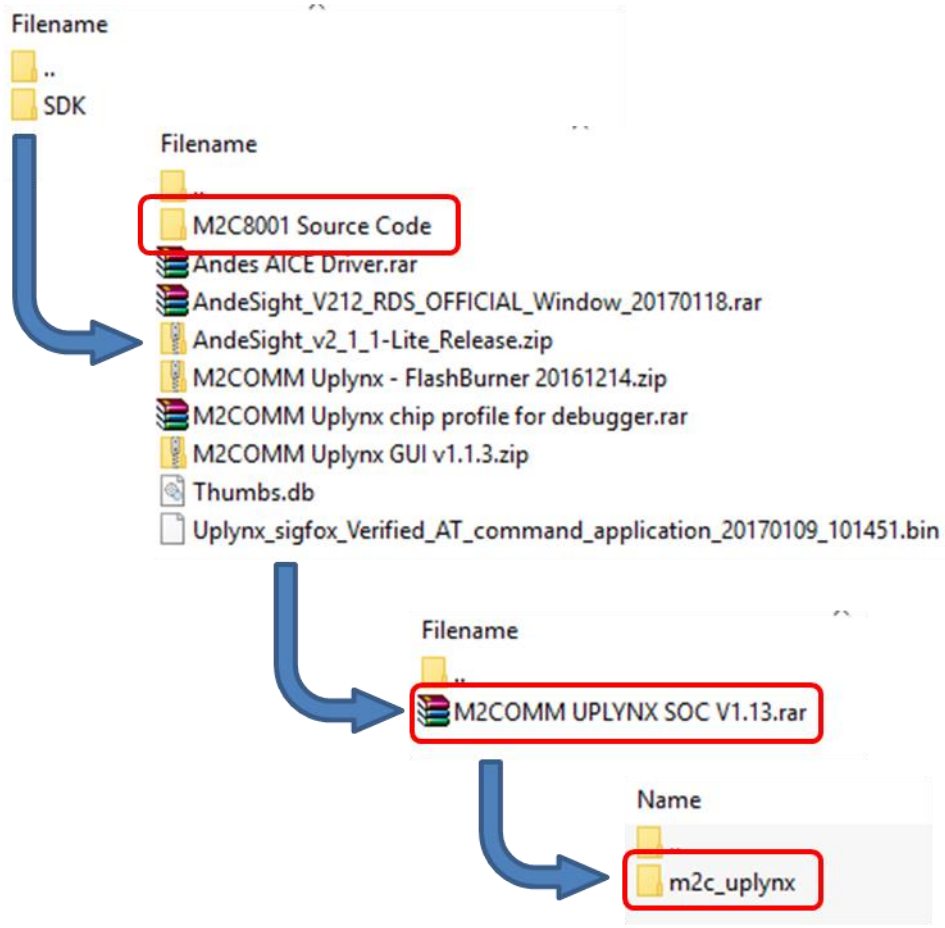
Note: To utilise the AndeShape™ ICE debug environment it is essential that the user installs the FULL version of AndeSight™ RDS

Download the required AndeSight development tool and follow the instructions in the AndeSight_v2.1_RDS_Installation_Guide_UM111_V1.2.pdf or AndeSight_v2.1_Lite_Installation_Guide_UM126_V1.1.pdf that is bundled with the tool to install.

AndeSight RDS requires a serial number and license file; the user should receive a serial number to activate the tools at time of purchase. The serial number will be used to apply for a license file which is MAC address specific.

3 *Uplynx Source Code*

The Uplynx source code is stored in the XS8001 Source Code folder as detailed below;



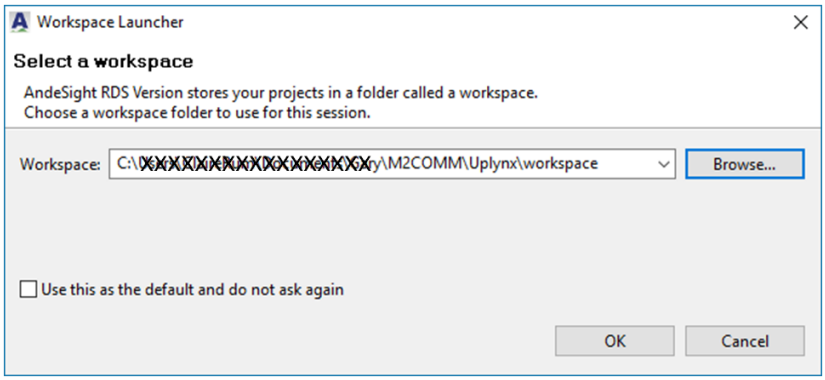
i. Locate and download the “ESMT UPLYNX SOC V1.23.rar” file from the ESMT ftp site as detailed in Section 2 above.

ii. Extract the “m2c_uplynx” folder and save to your local drive

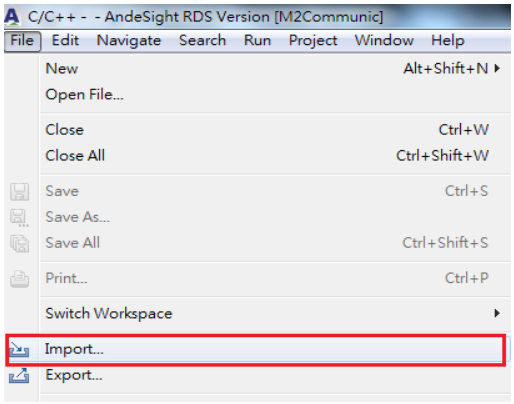
4 *Setting Up the Software Development Environment*

4.1 *Creating a Project Template with AndeSight RDS*

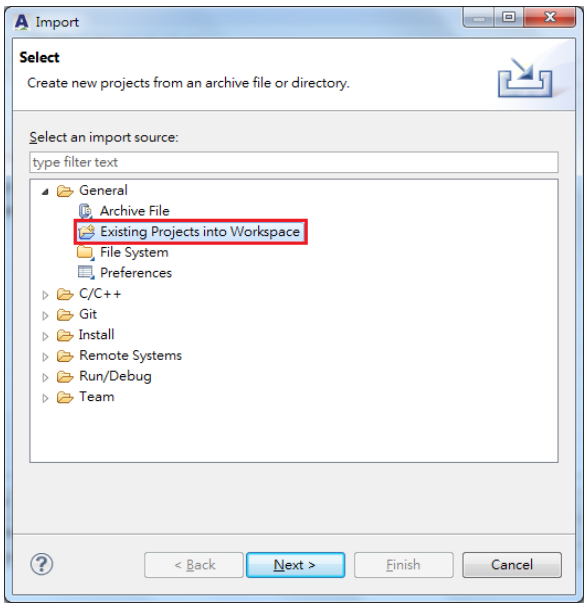
Launch AndeSight™ RDS to compile source code.



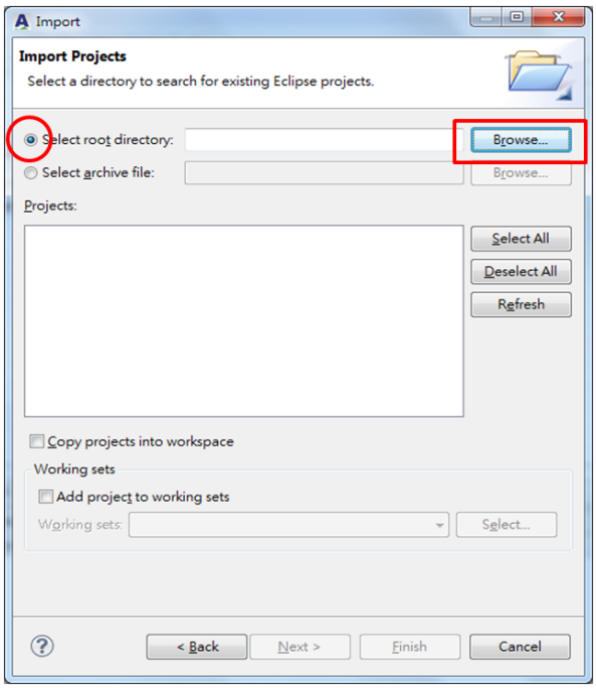
Create a workspace folder on your local drive



Select “File->Import” on the main menu of AndeSight™

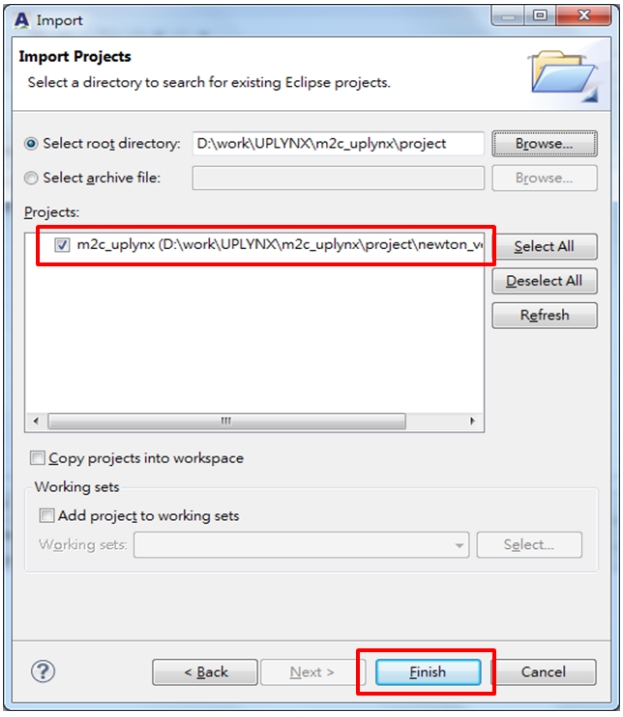


Choose “General->Existing Projects into Workspace” and click “Next”

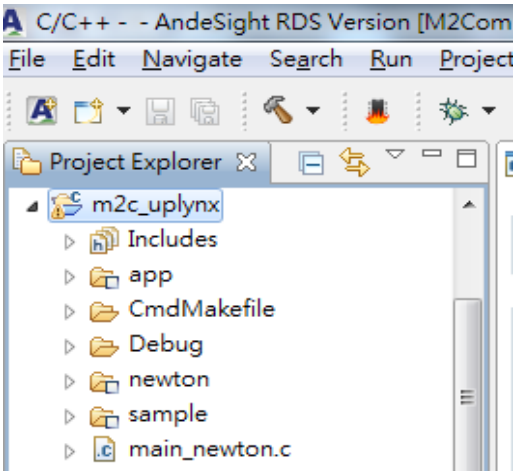


Check the “Select root directory” radio button to import a project from your file system.

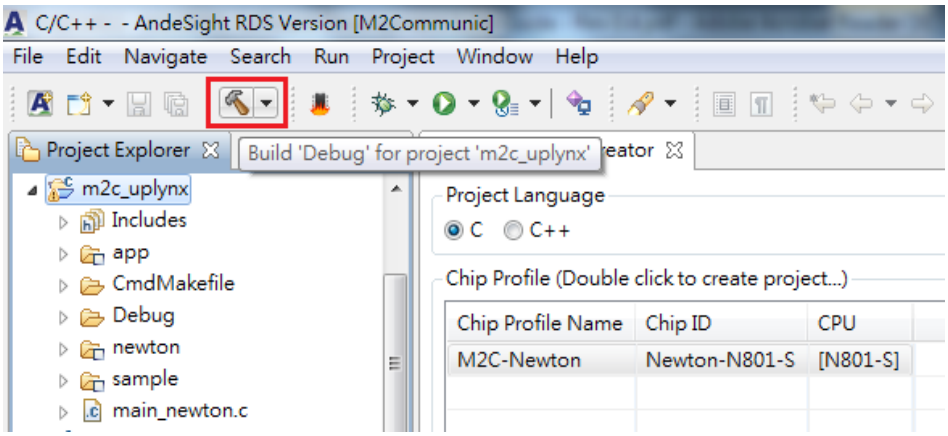
Click the “Browse” button to select the desired project to import.



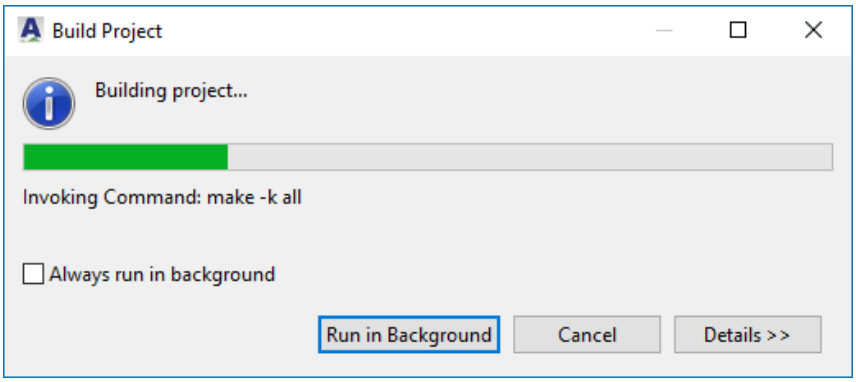
When you have located your project, select it and then click Finish to import the file into AndeSight.



Using the AndeSight Project Explorer locate your project



Use the debug icon to build your Uplynx project; the image file will output to "C:\{Root Directory}\project\newton_verification\Debug\output"

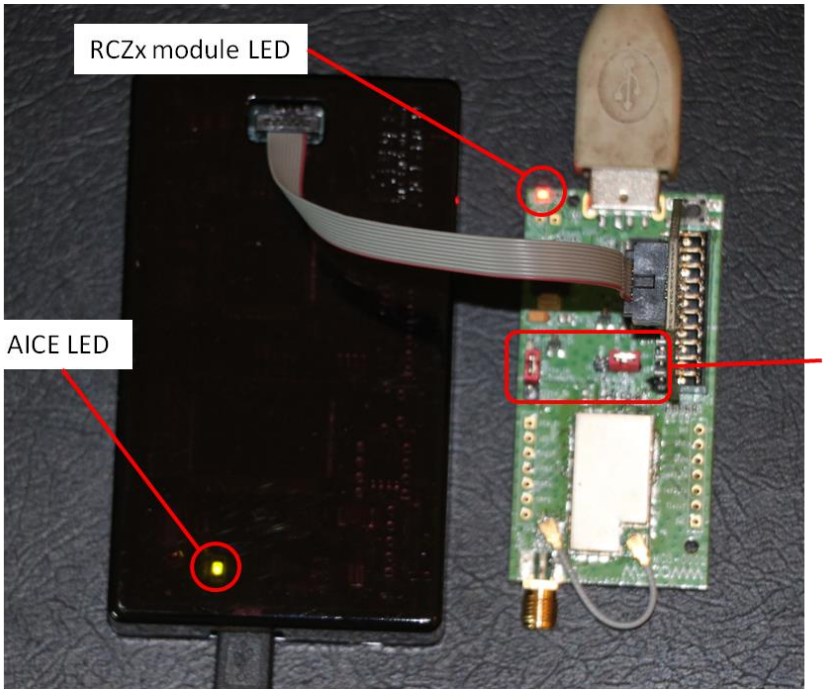


4.2 *Uplynx RCZx and AICE Set Up*

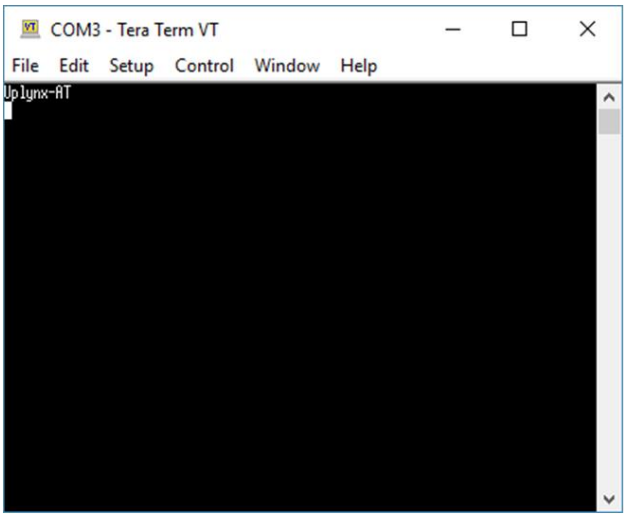
Connect Uplynx RCZx module (or Uplynx RCZx evaluation board) to the AndeShape™ In-Circuit Emulator (AICE).

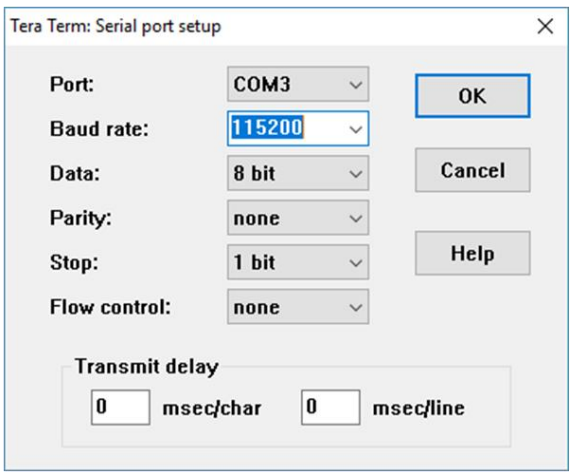
Connect Uplynx RCZx module (or Uplynx RCZx evaluation board) and AICE to your computer via USB cables.

Ensure that there is a single LED alight on the Uplynx RCZx module (or Uplynx RCZx evaluation board) and AICE, ensure that jumper connectors are installed; see picture below for details.



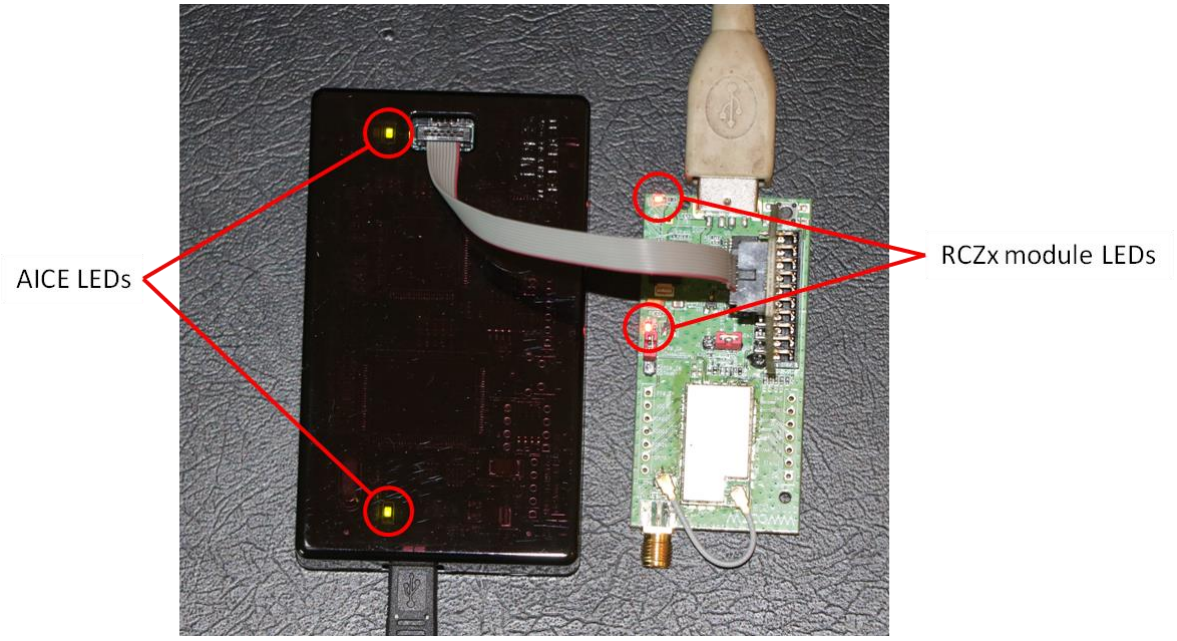
Open the Tera Term VT Terminal Emulator





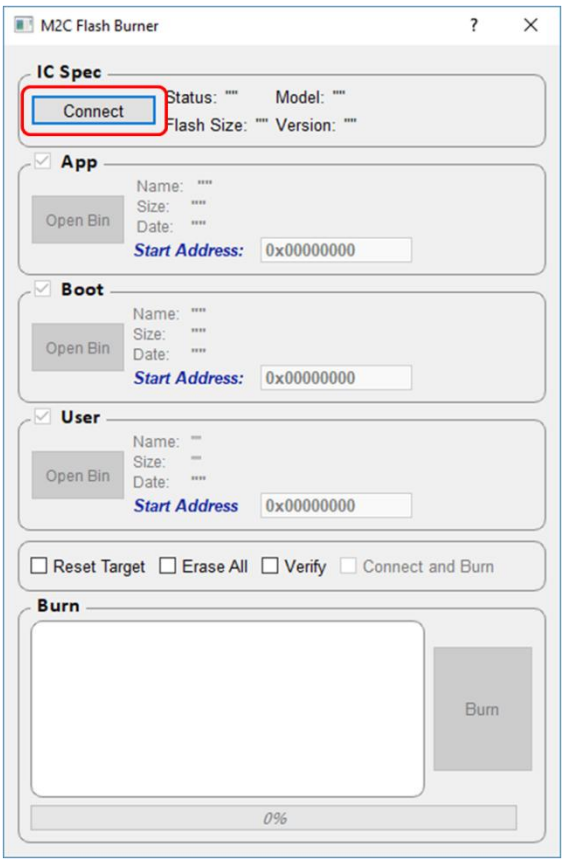
Select “Serial” in the New connection window and then click on “OK” (Ensure that the Baud rate is set 11520), the Tera Term window will display “Uplynx-AT” to signify that the port is active;

the Uplynx RCZx module (or Uplynx RCZx evaluation board) and AICE should both now have two LEDs lit up.

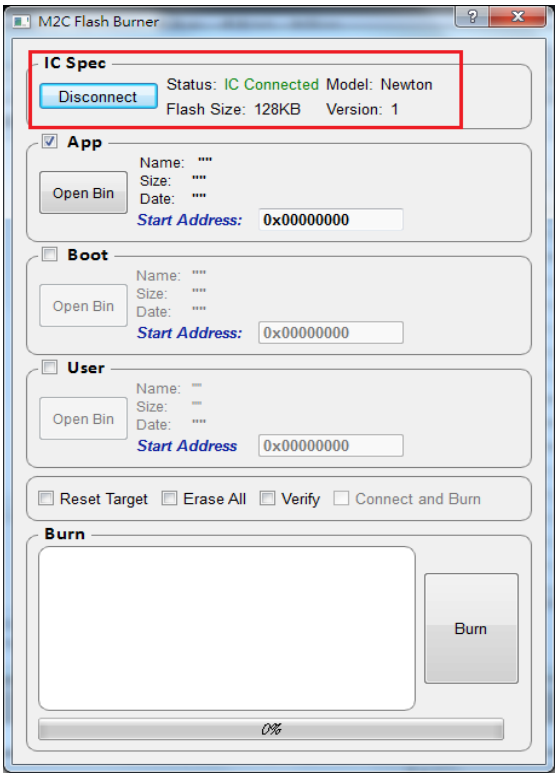


4.3 *Check for Firmware Upgrades*

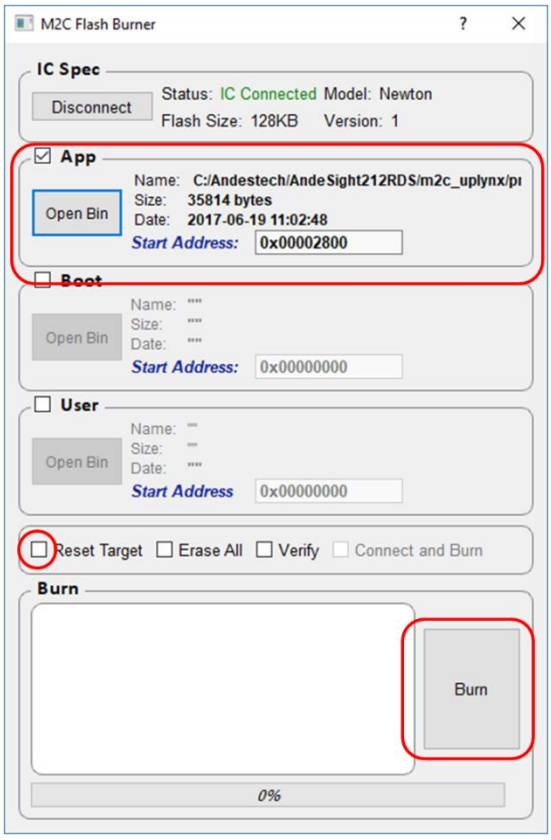
Navigate to “C:\{Root Directory}\M2C_Flash_Burner_20161214” and launch “aether.exe”



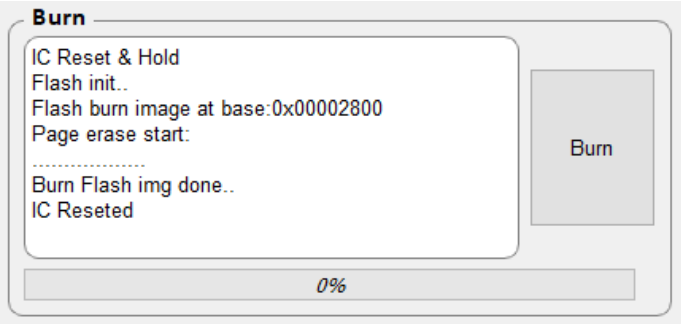
Select “Connect” and check that the “Status” displays “IC Connected”



Load the binary file from
"C:\{Root Directory}\m2c_uplynx\project\newton_verification\Debug\output" and select the "Reset Target" check box.

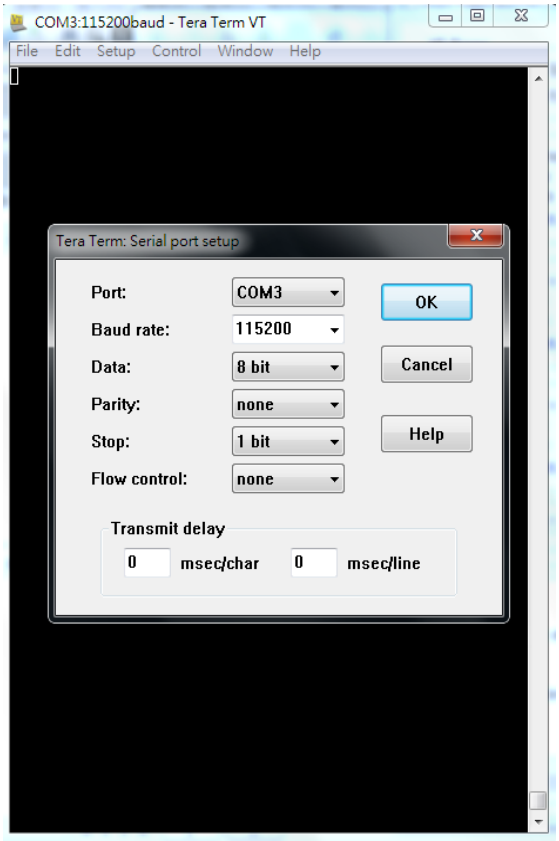


Click the “Burn” button and wait; you will be notified when flash has been upgraded

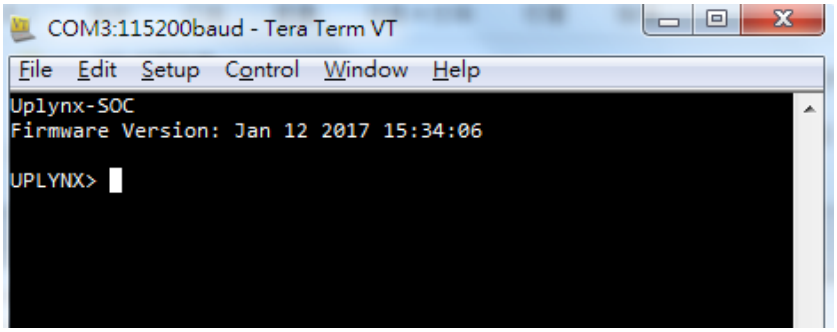


4.4 *Setting the Debug Port*

Launch “Tera Term” terminal emulator, Serial port setup



Check that details closely match those shown above and click OK to accept



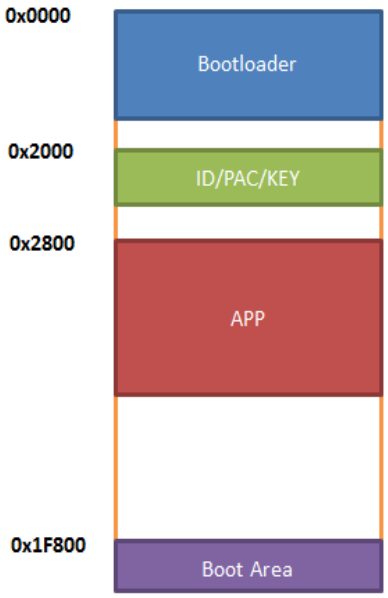
Terminal Emulator (Tera Term) Commands

Command	Description	Usage
Read	Read registry value from an indicated memory location	read [mem]
Write	Write registry value from an indicated memory location	write [mem] [value]
Read Sigfox ID		sid
Information	Inquire values of configuration data	info [item] where item values are; 0 – Tx power 1 – Frequency 2 – RCZ 3 – XO offset 4 – Check public key 5 – Configure word 6 – FM Ver. 7 – Get ID 8 – Get PAC 9 – Get operating condition
Set Configuration		config [item] where item values are; 0 – Set power (0 – 22dBm) 1 – Set frequency (e.g. 868130000) 2 – Set RCZ (1/2/3/4) 3 – Set XO offset 4 – 0/1 Disable/Enable public key
Word	Sets macro channel value for RCZ2/RCZ3/RCZ4	word [w1] [w2] [w3] [w4]
Print	Displays a copy of all the configuration settings	gcd
Save	Saves a copy of the configuration settings	save
Reset	Resets MCU	rst
Enter continuous wave mode		rfcw [freq]
Default	Restores default settings for RCZ1/RCZ2/RCZ3/RCZ4	def [rcz] where rcz = 1, 2, 3, 4
Dump memory	Dumps continuous 128 bytes memory content from an indicated memory address	dump [mem]
Demo	Runs a demo procedure using the SIGFOX API to send packets with/without SNEK according to RCZ setting	demo [rcz] [snek] where rcz = 1, 2, 3, 4 where snek = 0/1 (0:Disables SNEK, 1:Enables SNEK)

Run I2C	Processes the I2C functions using SI7120 device to measure relative humidity and temperature. Display data in m% and m°C	7120
Run sleep	Runs the system in sleep mode which causes the CPU to sleep for a while and wake up from wait-timer signal.	slp [sec] where sec = sleep period (in seconds)

5 *Flash Memory Mapping*

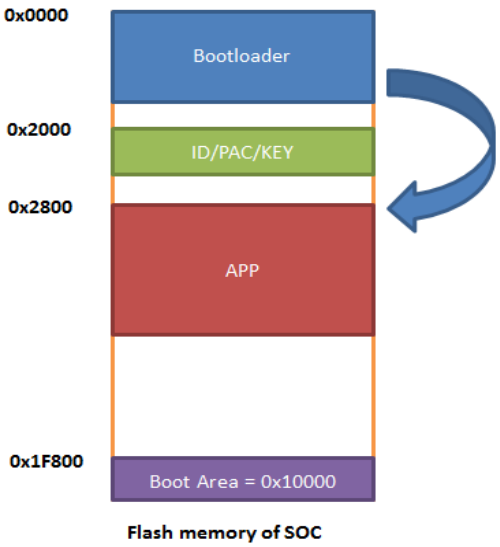
The flash memory is divided into four blocks



Flash memory of SOC

5.1 *Boot*

- The bootloader image is around 7kB, reverse 8KB offset.
- Includes the XMODEM procedure to receive image data.
- When booting, the bootloader will read the update flag from the flag section to check if it should enter update mode or normal mode.



Flash memory of SOC

- Flag:
  - 1: update mode
  - 0: normal mode

5.2 *APP*

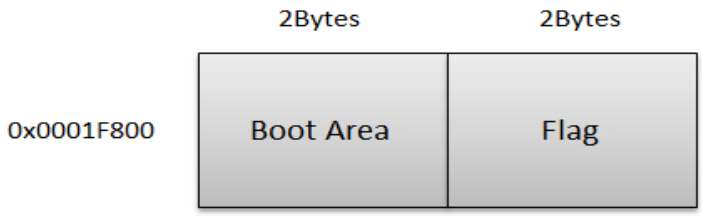
- When the bootloader reads that the update flag is 0, it will jump to the address of the APP to run the Sigfox procedure
- Application starts at an offset of 10KB, so the base address should be set to 0x2800

5.3 *ID*

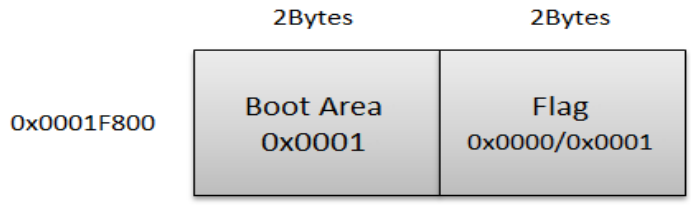
- Stores the ID/PAC information to this area.
- Allocates a page size of flash memory.
- Base address: 0x2000 (use 4th page)
  - 0x2000: ID (16 bytes)
  - 0x2010: PAC (16 bytes)
  - 0x2020:Key (16 bytes)

5.4 *Flag*

- Stores the flag data to flash memory.
- Flag == 0: normal mode
- Flag == 1: update mode
  - Base address: 0x0001F800 (first 2 bytes)
- Flag structure:



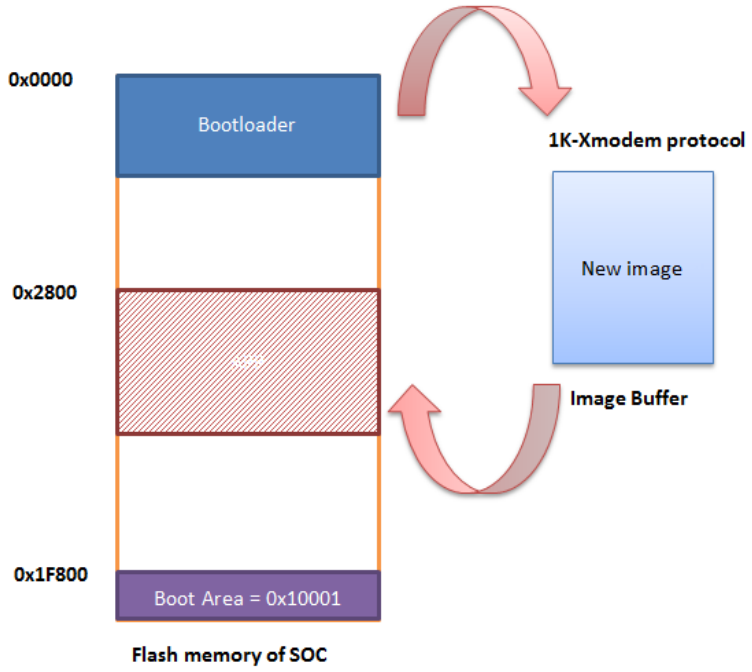
- Boot Area:
  - 0x0000: for module
  - 0x0001: for SOC
- Flag:
  - Update flag of SOC
  - 0x0000: normal mode
  - 0x0001: update mode
- SOC boot flow:
- Read content of base address 0x1F800



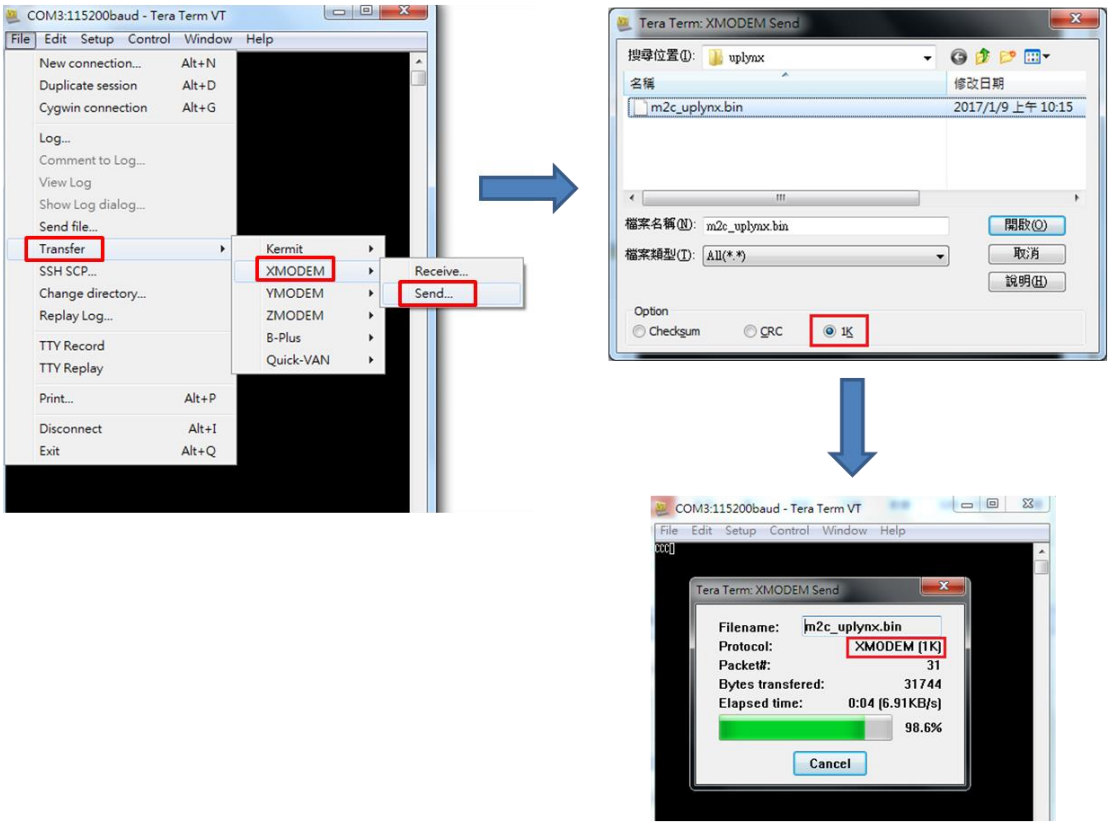

```
Val = *(u32 *)(0x0001F800);
Boot_area = (val >> 16) & 0xFFFF;
If(boot_area == 1)
  xflag = val & 0xFFFF;
```

  - xflag == 1: update mode
  - xflag == 0: normal mode

6 *SOC Firmware Upgrade*

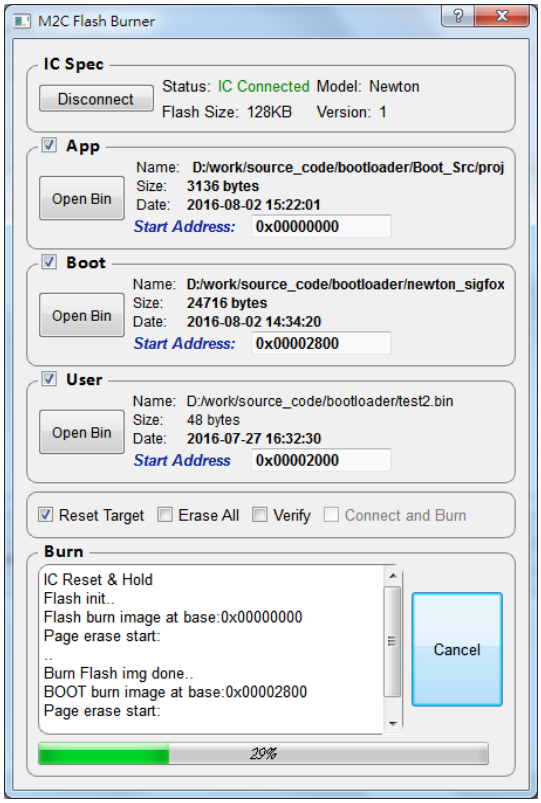


- Use the command “fwum 0” to enter update mode
  - UART baud rate should be set to 115200 to receive image data
  - After reboot, a char ‘C’ will print on the terminal. Choose the file to transfer over 1K-XMODEM
- The bootloader checks flag and enters XMODEM procedure to receive image data
- After upgrading the firmware, the flag will be cleared



7 *Default Settings and Configuration Data*

7.1 *Default Setting*



If the flash is empty, use ICE to write the bootloader and APP image.

Also write the configuration data to the flash, i.e. flags, ID and PAC.

When booting, use the “def” command to set default settings

7.2 *Configuration Data*

The configuration data requires 84bytes of storage in flash memory. Declaration into the “mamuf_api.h”

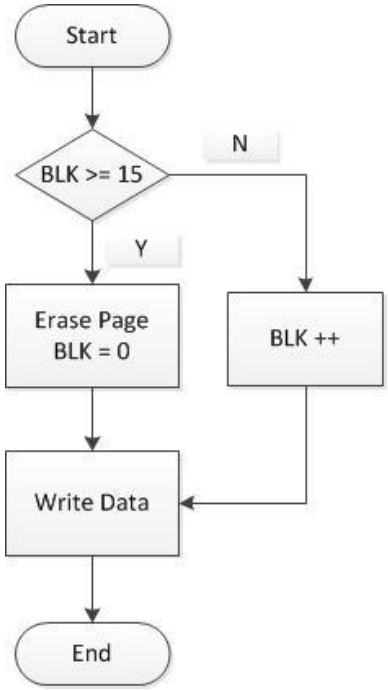
```
typedef struct
{
    sfx_u32 magic;
    sfx_u32 nv_mem[NV_MEM_LENGTH];
    sfx_u32 config_words[3];
    sfx_u32 xoOffset,tempOffset;
    sfx_u32 tempCP,tempCPOffset;
    sfx_u32 sclOffset, sclFreq;
    sfx_u32 RxFrequency;
    sfx_u32 defaultFccChannel;
    sfx_s32 TxPower;
    sfx_u32 standard;
    sfx_u32 flash_page;
    sfx_u32 flash_erase_counter;
    sfx_u32 flash_block;
    sfx_u32 crc;
} SfxConfig_t;
```

These 84bytes of data must be stored in flash memory. An Uplynx project will read/write the configuration data to allow it to proceed.

Flash pages are approx 2K bytes in size, this limitation results in having to divide the data into 16 blocks per page. The last block is reserved. One block size is 128 bytes.

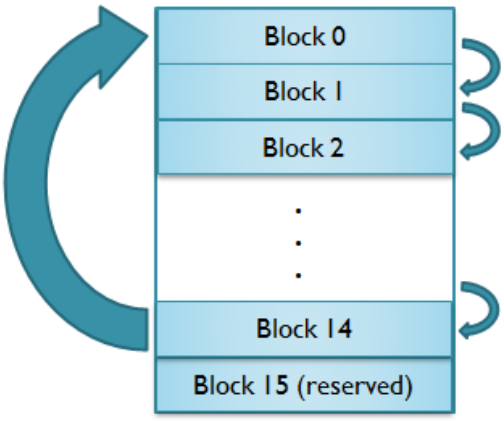
How to read/write these 16 blocks which are firmware controlled.

Block	Block Base Address	Block Size
Block 0	0x0001E000 – 0x0001E07F	128 Bytes
Block 1	0x0001E080 – 0x0001E0FF	128 Bytes
Block 2	0x0001E100 – 0x0001E17F	128 Bytes
Block 3	0x0001E180 – 0x0001E1FF	128 Bytes
-	-	-
-	-	-
-	-	-
Block 13	0x0001E680 – 0x0001E6FF	128 Bytes
Block 14	0x0001E700 – 0x0001E77F	128 Bytes
Block 15	0x0001E780 – 0x0001E7FF	128 Bytes

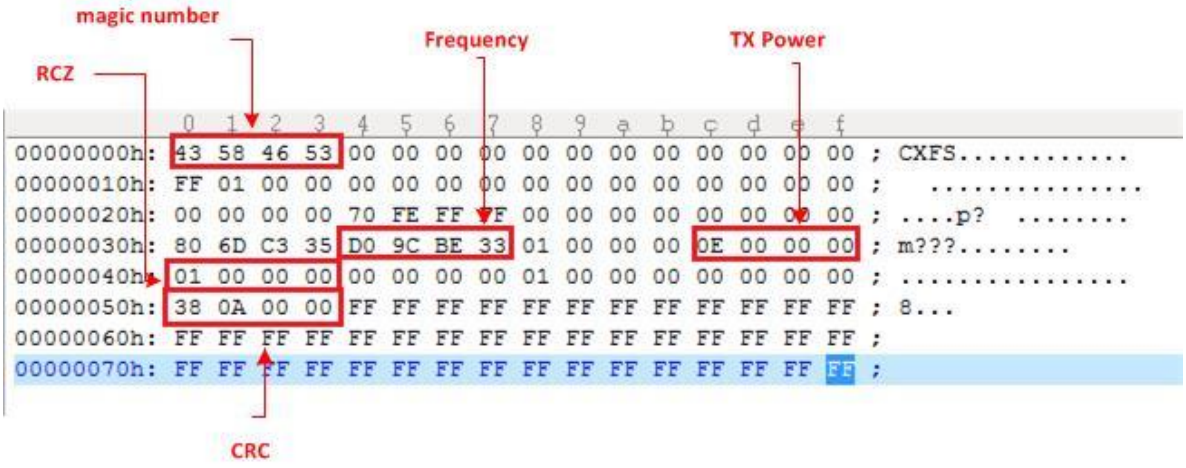


Flash control block diagram

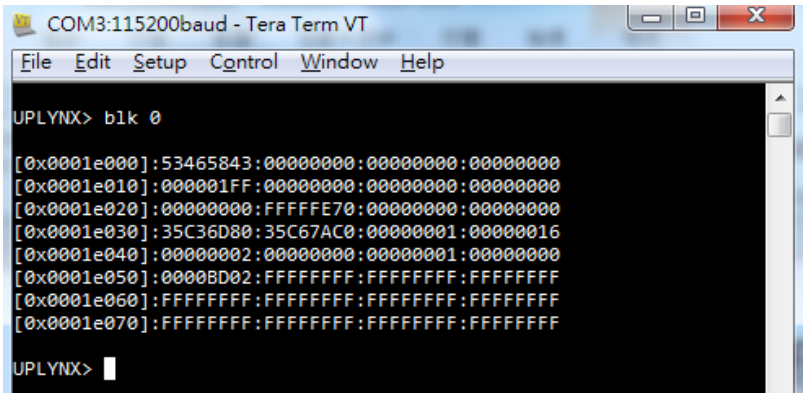
Block sequence



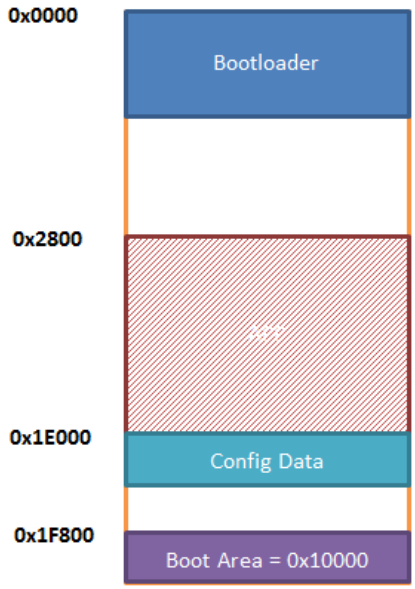
RCZ1 Configuration data example



To review the flash memory configuration data, use the “blk” command, e.g. blk [blk_num] where blk_num = 0 -15. The “blk” command can also be used as an aid for de-bugging.



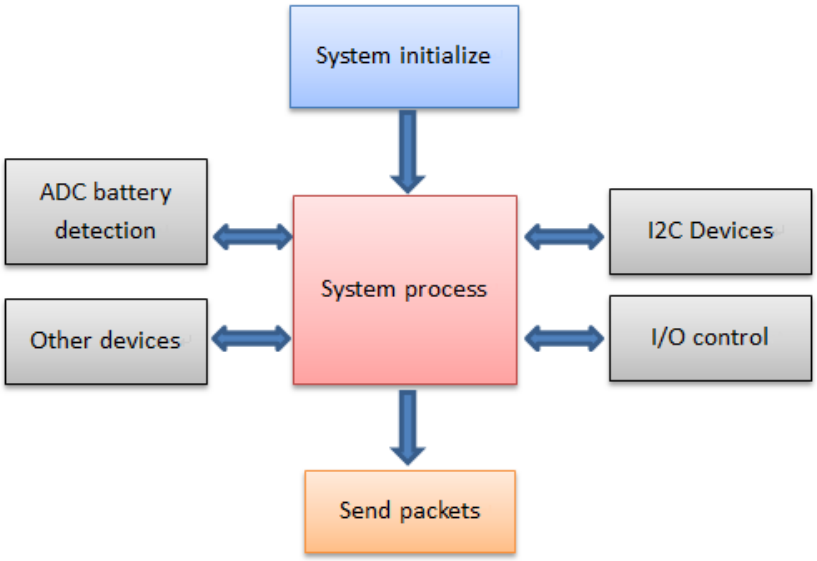
The maximum code size of an APP is approx 110KB but because the configuration data is stored at base address 0x1E000 of flash memory, hence maximum codes size = 01E000 – 0x2800



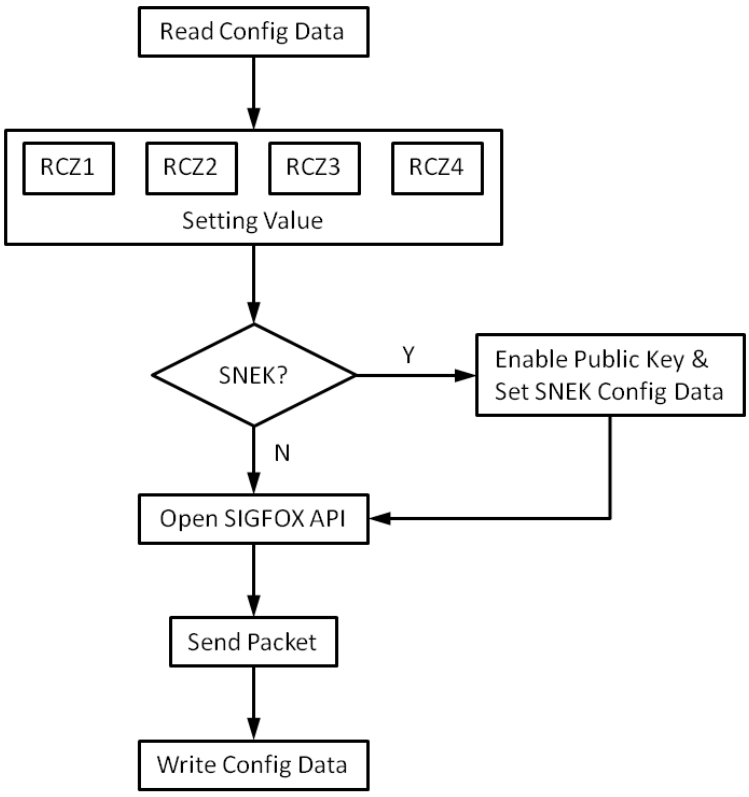
Flash memory of SOC

8 *System Flow*

8.1 *Functional Overview*



System flow



SIGFOX API Flow

```
35  int main()
36  {
37      System_init();
38
39      /* init debug port */
40      UART_DATA_t tUartData={UART_DRV_BAUDRATE_115200,    \
41          UART_DRV_DATABITS_8,UART_DRV_STOPBITS_1,UART_DRV_PARITY_NONE};
42      UART_Init(UART0,&tUartData);
43
44      sio_printf("Uplynx-SOC\r\n");
45      sio_printf("%s\r\n",version);
46      SfxInit();
47
48      SYS_SetAPBClockSrc(APB_CLK_SRC_XO);
49      SYS_SetCpupllFreq(CPUPLL_FREQ_60M);
50      SYS_SetAPBClockSrc(APB_CLK_SRC_CPUPLL);
51      UART_EnableInt(UART0, UART_IER_RDA);
52      while (1){
53          CPU_STANDBY();
54          CMD_Process();
55      }
56
57      return 0;
58  }
```

Initial code of chip

Put your Initial cod here.
e.g. I/O

Debug port

Load config data

Put your demo code here

Command line

8.2 *Sigfox Library API*

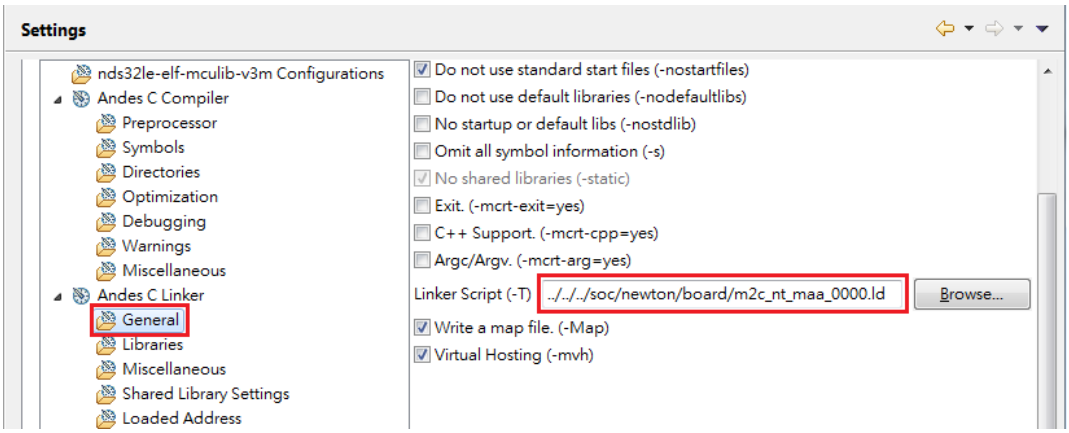
The following table lists the available functions for the Sigfox library API

Function Name	Parameters	Description	Comments
SIGFOX_API_open	sfx_rcz_t*rcz	Pointer for RCZ	Initializes the library
	sfx_u8*id_ptr	Pointer for 32-bit ID	
SIGFOX_API_close			Closes the library
SIGFOX_API_send_frame	sfx_u8*customer_data	Data to transmit	Sends a standard SIGFOX frame with customer payload. Customer payloads cannot exceed 12 Bytes
	sfx_u8*customer_data_length	Data length in Bytes	
	sfx_u8*customer_response	Returned 8 Bytes data in case of downlink	
	sfx_u8 tx_repeat	Number of repetitions	
	sfx_bool initiate_downlink_flag	Downlink flag	
SIGFOX_API_send_bit	sfx_bool bit_value	Bit state (SFX_TRUE or SFX_FALSE)	Sends a standard SIGFOX frame with null customer payload
	sfx_u8*customer_response	Returned 8 Bytes data in case of downlink	
	sfx_u8 tx_repeat	Number of repetitions	
	sfx_bool initiate_downlink_flag	Downlink flag	
SIGFOX_API_reset		This function has no effect in either DC or LBT	This function is used for FH and will reset the macro channel to default_sigfox_channel configured with SIGFOX_API_set_std_config

Function Name	Parameters	Description	Comments
SIGFOX_API_set_std_config	sfx_u32 config_words[3]	Dependant on standard	Function is used to configure specific variables for standards. Call this function after SIGFOX_API_open() for FH and LBT FH (Frequency Hopping) : • Configuration words to enable/disable the 192KHz macro channels authorized for transmissions • Macro channel separation is 300 kHz • At least 9 macro channels must be enabled to ensure the minimum of 50 FCC channels (9 x 6 = 54) • The configured default_sigfox_channel must be enabled in configuration word

9 *Booting*

Uplynx can be booted without the use of the bootloader, building an Uplynx project will set the base address to 0 whilst compiling, however, the user is required to identify a block of flash memory in which to store the ID/PAC/KEY.



10 *Sample Code*

10.1 *GPIO*

- Each of the GPIO pins is multi-functional
- Pin configuration:
  - Program to be GPIO mode
    - GPIO_SetPinMux(u8 pin)
  - Set GPIO direction (Input mode/Output mode)
    - GPIO_SetDir(u8 pin, u8 INorOut)
  - If Output mode, set gpio level to be Hi/Lo
    - GPIO_SetOutput(u8 pin, u8 hi_lo)
  - GPIO INT Enable:
    - GPIO_EnableInt(u8 pin,u8 TrigType,u8 Mode)

10.2 *Sleep Mode*

Search for the smpl_pm.c file (located in \m2c_uplynx\sample\)

- The power management unit (PMU) switches between power mode and wake up by interrupt source
- In sleep mode, the majority of the digital functions are powered off and clock-gated
  - PMU_EnterSleep(); //sleep mode
- Wakeup source:
  - Before the system goes into sleep mode, one or more wakeup sources should be enabled.
  - Wake up by FIQ interrupt sources (GPIO/TIMER/RTC)
- Sample code uses the wait timer to wake up the CPU

10.3 *Uplynx Demo*

- Search for the smpl_uplynx.c file (located in \m2c_uplynx\sample\)
- RCZ1/RCZ2/RCZ3/RCZ4 APIs function codes for configuring UPLYNX with SIGFOX APIs;
  - void smplRCZ1(u8 snek_mode)
  - void smplRCZ2(u8 snek_mode)
  - void smplRCZ3(u8 snek_mode)
  - void smplRCZ4(u8 snek_mode)
- The basic concept of operating UPLYNX is
  - Set power/frequency/RCZ
  - Set config word data (if RCZ=2, RCZ=3 or RCZ=4)
  - Set snek mode to enable/disable public key
  - Open SIGFOX Libraries to use SIGFOX APIs
  - To transmit a packet by calling send_frame or send_bit APIs
- SIGFOX APIs
  - Please refer to sigfox_api.h for more details
- Listed below are the SIGFOX APIs provided by SIGFOX along with a detailed description of how to use the APIs
- The latest version of the SIGFOX Library is V.1.8.9
  - SIGFOX_API_get_version(sfx_u8** version, sfx_u8* size)
- The most frequently used SIGFOX APIs are
  - sfx_error_t SIGFOX_API_open(sfx_rcz_t *rcz, sfx_u8 *id_ptr)
  - sfx_error_t 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
  - sfx_error_t SIGFOX_API_send_bit(sfx_bool bit_value)
  - sfx_u8 *customer_response
  - sfx_u8 tx_repeat
  - sfx_bool initiate_downlink_flag
  - sfx_error_t SIGFOX_API_reset(void)
  - sfx_error_t SIGFOX_API_set_std_config(sfx_u32 config_words[3]
  - sfx_u16 default_sigfox_channel
- Before using the SIGFOX APIs, user must call for the SIGFOX_API_open() function to enable the SIGFOX Libraries and to avoid an “SFX_ERR” message.

10.4 *ADC*

- 2 APIs are provided for detecting the voltage and temperature.
- Voltage detection
  - u32 adcBatteryDetect(void); //unit is mV
- Temperature detection
  - u32 adcTemperatureDetect(void); //unit is °C

10.5 *I2C*

- The I2C configuration functions are
  - void I2C_Init(u8 mode,u8 div)
  - u8 I2C_Read(u8 addr,u8 len,u8 * data)
  - u8 I2C_Write(u8 addr,u8 len,const u8 * data)
- The sample code uses the on-board SI7120 device

10.6 *Others*

- PWM Configuration
  - smpl_pwm.c
- RTC Configuration
  - smpl_rtc.c
- Timer Configuration
  - smpl_timer.c/smpl_wdt.c
- SPI Interface Configuration
  - smpl_spi.c