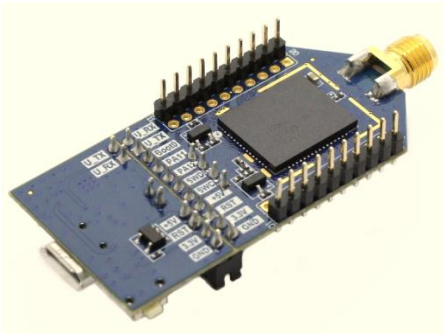
S76S/S78S

SDK Manual



Document Name	**S76S/S78S SDK Manual**
Version	**C**
Doc No	
Date	**May 3, 2017**

Document History

Date	Revised Contents	Revised by	Version
Oct 13 ,2016	Initial version.	Leo Tseng	A
Mar 29, 2017	LoRaWAN™ migration infomation added.	Leo Tseng	B
May 3, 2017	FTP information removed	Leo Tseng	C

Index

1. Overview

The S76S/S78S is designed & manufactured in a smallest form factor - SiP (System in Package). It integrates with Semtech SX1276/SX1278 and a 32-bit ultra-low power Coretex®-M0+ MCU (STM32L073x), supporting global 868 MHz or 915 MHz ISM-Bands/433 MHz ISM-Bands. Capable of 2-way communication and reach over 16 km (10 miles) distance in our field test.

S76S/S78S integrates ARM Coretex®-M0+ MCU with LoRa™ modulation. S76S/S78S has two main parts, one is STM32L073X and the other is SX1276/SX1278. These two powerful combinations can achieve middle range communication and transfer data into several standard interface.

SX1276 transceivers feature the LoRa™ long range modem that provides ultra-long rang spread spectrum communication, it offers bandwidth options ranging from 7.8 KHz to 500KHz with spreading factors ranging from 6 to 12, and covering all available frequency bands (SX1276: 862-1020MHz; SX1278: 137-525MHz). And SX1276 can achieve a sensitivity of over -138 dBm. The high sensitivity combined with the integrated +20 dBm power amplifier yields industry leading link budget making it optimal for any low data rate application requiring range or robustness.

STM32L073X is a high performance but low power MCU running at 32 MHz maximum that provides multiple interfaces like UART, GPIO, ADC, SPI, I2C, JTAG interfaces. It allows customer has lots of flexibility to develop their own products.

In this document, we have introduced and detailed description about flashing FW, using developing environment, flashing code and SDK flow chart.

For costumer fast developing their own product, we provide an evaluation board named as EK-S76SXB/EK-S78SXB which is carried S76S/S78S module and it can easily understand AcSiP reference design toward HW & SW points of view.

In chapter2, it describes the all connections between MCU & SX1276 and the external pins around EK-S76SXB/EK-S78SXB; In chapter3, it describes how to upgrade firmware and the prerequisite by using the different debugger/programmer; In chapter4, it tells user how to establish development environment when user want to modify or trace AcSiP SDK source code. In chapter5, it demonstrates the S76S/S78S SDK features and detailed function blocks; Finally, in chapter6, it introduces LoRaWAN™ protocol and LoRaWAN™ stack implementation, even tells user how to migrate a LoRaWAN™ open source code into AcSiP SDK provided drivers.

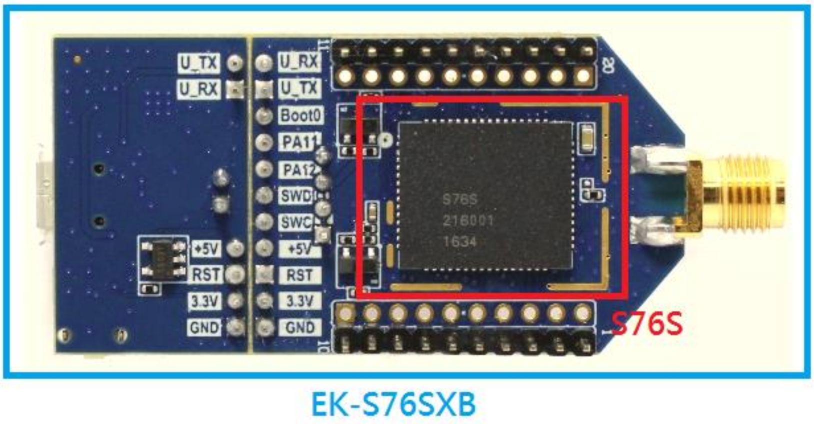


Figure 1.1 (Take S76S as example)

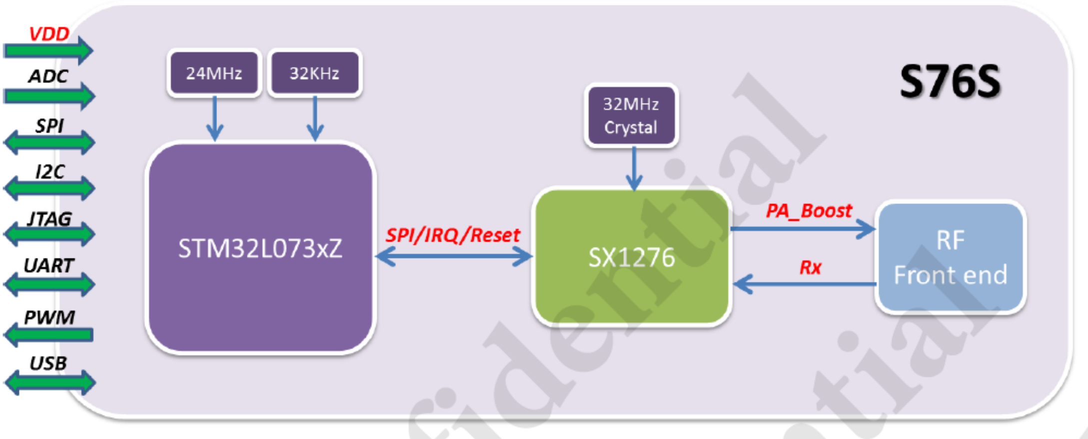


Figure 1.2 S76S Block Diagram

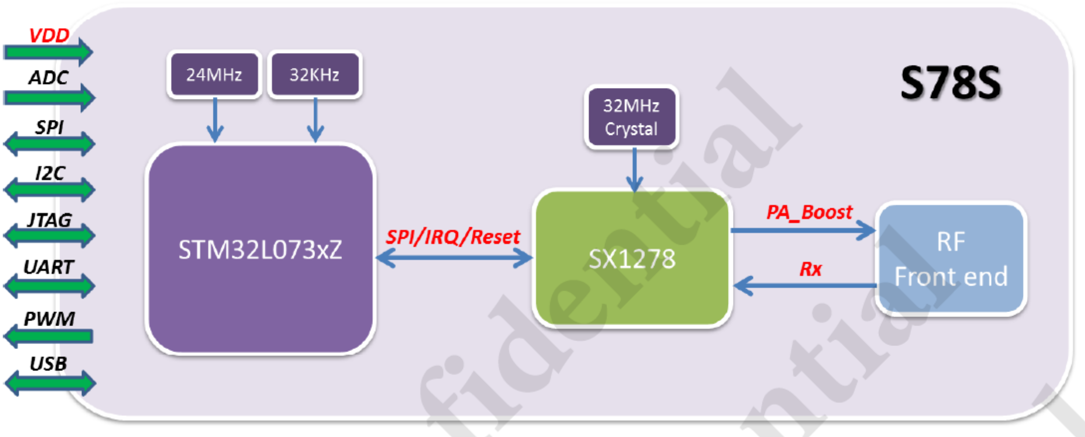


Figure 1.3 S78S Block Diagram

2. Hardware Interface

2.1 Connections between MCU & SX1276

STM32L073X MCU and SX1276 connect with SPI interface and one Reset pin. SPI runs in full-duplex mode and uses 4-pin (SCK, MOSI, MISO and NSS) to communicate. Normally, user doesn't need to configure SPI on this developing board, MCU and SX1276 can already operate well on AcSiP's default configuration.

As the figure 2.2 tells, STM32L073X MCU and SX1276 are packaged as SiP (System in Package) and named as S76S/S78S which is located in the center of the EK-S76SXB/EK-S78SXB board (the center part of figure 2.2).

2.2 Pins definition

2.2.1 XB board Peripheral List

The supporting peripheral interface & important endpoint on XB board area listed below:

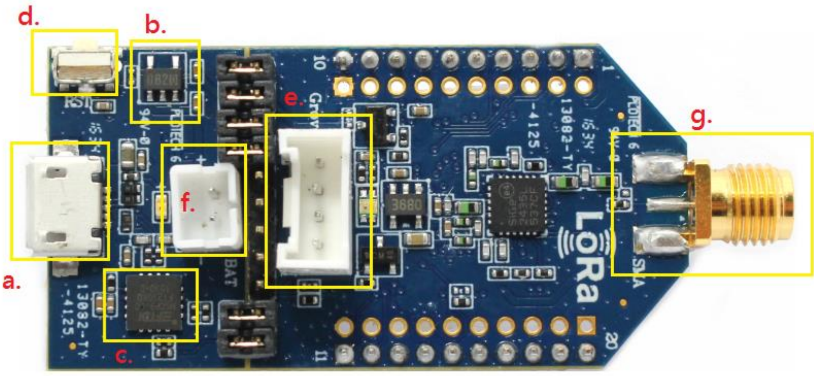


Figure 2.1

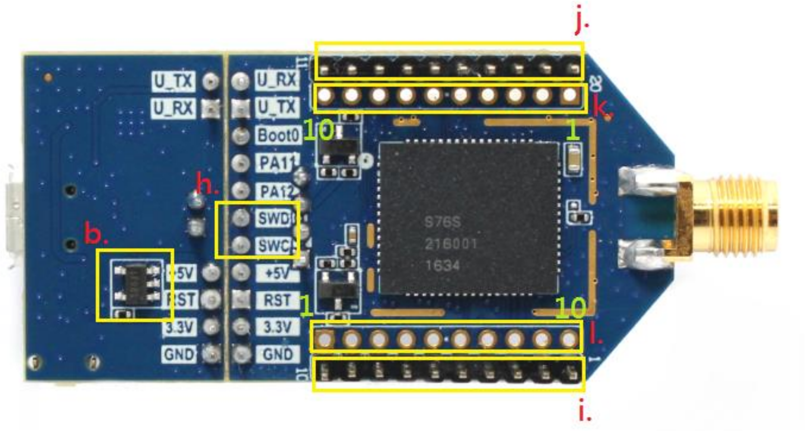


Figure 2.2

a. Mini USB port

⇨ Only used for providing 5V power supply, no any USB protocol functions.

b. 5v-To-3.3V regulator (LDO) & Li-on battery charger IC

c. USB to UART IC

d. Reset button

⇨ Connected to MCU HW reset pin

⇨ Active low (Pull low to trigger reset behavior).

e. I2C1 pins

⇨ Supporting 100KHz, 400KHz and 1MHz.

f. BAT pins

⇨ 2-pin connector for a chargeable battery.

g. SMA-J Antenna Connector

⇨ RF signal input

h. JTAG pins

⇨ 20-pin JTAG connector for ST-Link flashing & debugging FW.

i. XB External Pins (Pin1 to Pin10)

j. XB External Pins (Pin11 to Pin20)

k. XB Additional Breakout (Pin1 to Pin10)

l. XB Additional Breakout (Pin1 to Pin10)

⇨ **Pins definition reference is located at 2.2.2.**

UART1 TX(PA9), RX(PA10) pins

UART4 TX(PA0), RX(PA1) pins

UART2 TX(PA2), RX(PA3) pins

UART5 TX(PC12), RX(PD2) pins

⇨ Three standard serial port pins (UART2/4/5) for debugging message input or output or data pins.

⇨ UART1 TX RX pins converts to mini USB D+/D- by USB to UART IC.

SPI1 pins

⇨ SCL, MISO, MOSI, NSS

⇨ Up to 42 Mbps, it can either configure as Master or Slave mode.

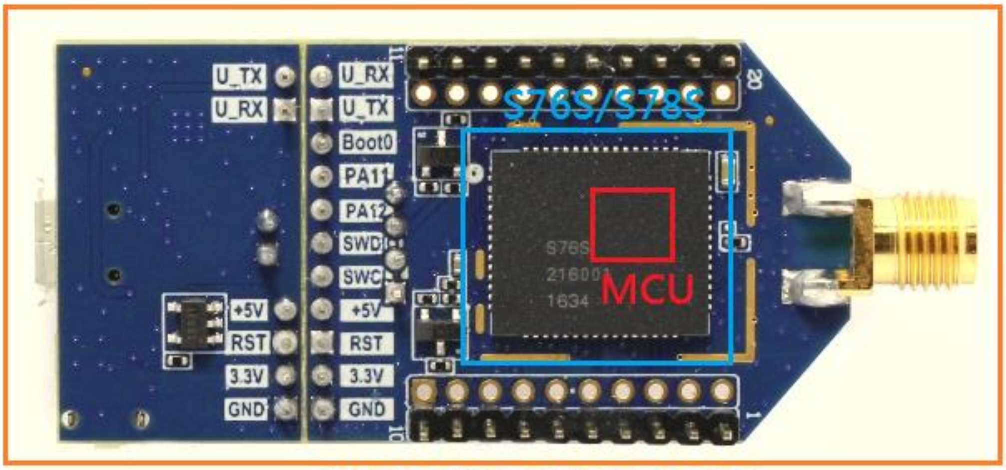
ADC pins

⇨ 12-bit ADC in.

VCC & GND pins

⇨ Test pins for measuring purpose.

2.2.2 Connections between MCU & XB board



EK-S76SXB/EK-S78SXB

EK-S76SXB/EK-S78SXB Board	S76S/S78S	STM32L073X (MCU)
None	pin1	No connection
GND	pin2	GND
GND	pin3	GND
PC0 for ADC/I2C3 SCL	pin4	PC0
PC1 for ADC/I2C3 SDA	pin5	PC1
PC2 for ADC/SPI2 MISO	pin6	PC2
PC3 for ADC/SPI2 MOSI	pin7	PC3
None	pin8	No connection
None	pin9	No connection
None	pin10	No connection
None	pin11	No connection
RST	pin12	MCU_RESET
PA0 for ADC/USART4 TX	pin13	PA0
GND	pin14	GND
GND	pin15	GND
PA2 for ADC/USART2 TX	pin16	PA2_TXD_A
PA3 for ADC/USART2 RX	pin17	PA3_RXD_A
PA4 for SPI1 NSS/USART2 CK	pin18	PA4_SPI1_NSS
PA5 for SPI1 SCK/ADC	pin19	PA5_SPI1_SCK
PA6 for SPI1 MISO/ADC	pin20	PA6_SPI1_MISO
PA7 for SPI1 MOSI/ADC	pin21	PA7_SPI1_MOSI
PC4 for ADC	pin22	PC4
PC5 for ADC	pin23	PC5

PB0 for ADC	pin24	PB0_IO_INT1
PB1 for ADC/I2C2 SDA	pin25	PB1_IO_INT2
PC6	pin26	PC6
PC7	pin27	PC7
PC8	pin28	PC8
PC9 for DAC/I2C3 SDA	pin29	PC9
RXTX Switch	pin30	RXTX/RFMOD
GND	pin31	GND
GND	pin32	GND
RF_ANT	pin33	No connection
GND	pin34	GND
GND	pin35	GND
PA1 for RF RXTX Switch	pin36	PA1_RF_FEM_CPS
GND	pin37	GND
None	pin38	No connection
GND	pin39	GND
None	pin40	No connection
GND	pin41	GND
None	pin42	No connection
VDD_3.3V	pin43	VDD33
VDD_3.3V	pin44	VDD33
PA8 for USART1_CK/I2C3 SCL	pin45	PA8_USART1_CK
PA10 for USART1 RX/I2C1 SDA	pin46	PA10_USART1_RX
PA9 for USART1 TX/I2C1 SCL	pin47	PA9_USART1_TX
PA11 for USART1 CTS/SPI1 MISO/USB DM	pin48	PA11_USART1_CTS
PA12 for USART1 RTS/SPI1 MOSI	pin49	PA12_USART1_RTS
PA13 for SWDIO	pin50	PA13_SWDIO
PA14 for SWCLK	pin51	PA14_SWCLK
PC10 for USART4 TX	pin52	PC10
PC11 for USART4 RX/ADC	pin53	PC11
PC12 for USART5 TX/USART4 CK	pin54	PC12
PD2 for USART5 RX	pin55	PD2
PB5 for USART5 RTS/SPI1 MOSI	pin56	PB5
PB6 for USART1 TX/ I2C1 SCL	pin57	PB6_SCL
PB7 for I2C1 SDA/USART1 RX/USART4 CTS	pin58	PB7_SDA
BOOT0	pin59	BOOT0
PB8 for I2C1 SCL	pin60	PB8_IO_LED_FCT

GND	pin61	GND
GND	pin62	GND

2.3 Usable Interface of EK-S76/78SXB board

2.3.1 RF Antenna

A general RF Antenna connector is compatible with SMA male specification (BNC female to SMA female).

2.3.2 I2C

One I²C bus interfaces can operate in master and slave modes. They can support the standard mode (up to 100 kHz), fast mode (up to 400 kHz) and fast plus mode (up to 1MHz). and it support DMA mode to reduce CPU overload.

2.3.3 VDD33

A 3.3V pin that is provided by a 5V-to-3V regulator (LDO), and it provides the power supply to MCU and SX1276.

2.3.4 SWD

A standard SWD (Serial Wire Debug) port is used for STM32 platform debugging & flashing. User needs to connect this two SWDIO & SWCLK pins with ST-LINK/V2 unit (or ULINK). After that, user can use STM32 ST-LINK Utility or Keil IDE to flash & debug firmware.

The detailed reference about ST-LINK/V2 is located in this below link

http://www.st.com/content/st_com/en/products/development-tools/hardware-development-tools/development-tool-hardware-for-mcus/debug-hardware-for-mcus/debug-hardware-for-stm32-mcus/st-link-v2.html

2.3.5 USART/UART

The devices embed three universal synchronous/asynchronous receiver transmitters (USART1, USART2, USART4 and USART5).

The universal synchronous asynchronous receiver transmitter (USART) offers a flexible means of Full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The USART offers a very wide range of baud rates using a programmable baud rate generator. It supports synchronous one-way communication and Half-duplex Single-wire communication, as well as multiprocessor communications.

High speed data communication is possible by using the DMA (direct memory access) for multi-buffer configuration.

2.3.6 ADC

One 12-bit analog-to-digital converter is embedded and performing conversions in the single, continuous, scan and discontinuous mode. In scan mode, automatic conversion is performed on a selected group of analog inputs. The analog watchdog feature allows the application to detect if the input voltage goes outside the user-defined higher or lower thresholds

The detailed reference about ADC is located in this below link:

http://www.st.com/content/ccc/resource/technical/document/reference_manual/2f/b9/c6/34/28/29/42/d2/DM00095744.pdf/files/DM00095744.pdf/jcr:content/translations/en.DM00095744.pdf Chapter 14 Analog-to-digital converter (ADC)

2.3.7 SPI

The SPI/I²S interface can be used to communicate with external devices using the SPI protocol or the I2S audio protocol. SPI or I2S mode is selectable by software. SPI mode is selected by default after a device reset. The serial peripheral interface (SPI) protocol supports half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multi-master configuration

The detailed reference about SPI is located in this below link:

http://www.st.com/content/ccc/resource/technical/document/reference_manual/2f/b9/c6/34/28/29/42/d2/DM00095744.pdf/files/DM00095744.pdf/jcr:content/translations/en.DM00095744.pdf Chapter 31 Serial peripheral interface/ inter-IC sound (SPI/I2S)

2.3.8 NRST

NRST can trigger an external reset to MCU system reset when V_{NRST} is low. A system reset sets all registers to their reset values except for the RTC, RTC backup registers and control/status registers.

3. Upgrade Firmware

This chapter tells user how to download firmware into EK-S76/78SXB EK board.

3.1 Prerequisite for ST-LINK/V2

3.1.1 Hardware Preparation

HW component - See Figure 3.1

- USB Port x 2 on PC/NB site
- ST-Link/V2 unit x1
- EK-S76SXB/EK-S78SXB board x1
- Mini USB cable x 1 and Micro USB cable x1

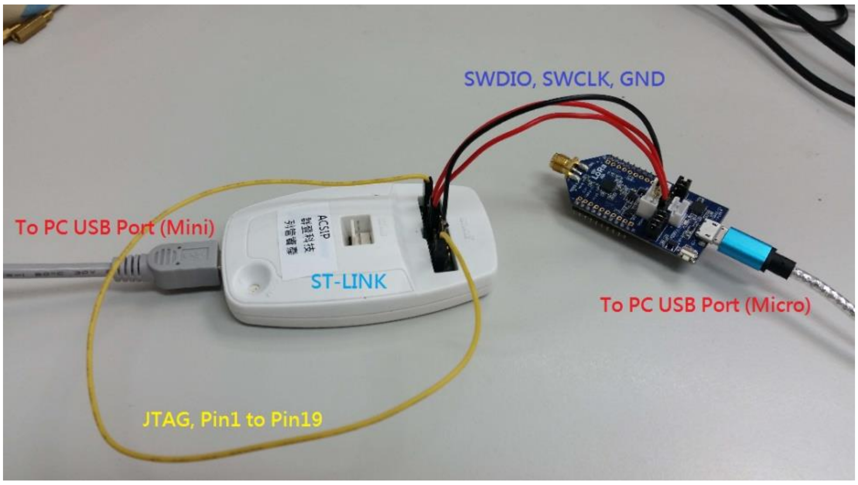


Figure 3.1

Connect HW component as Figure 3.2.

- Connect S EK-S76SXB/EK-S78SXB (Abbr. XB) board to ST-LINK.
  - ST Link Pin7 <==> **SWDIO** on XB board
  - ST Link Pin9 <==> **SWCLK** on XB board
  - ST Link Pin12 <==> **GND** on XB board
  - ST Link Pin1 <==> ST Link Pin19

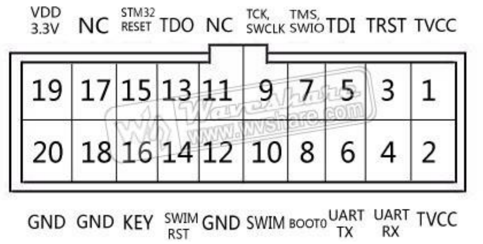


Figure 3.2 Figure 3.3

- ➢ Connect ST-LINK to PC with USB standard A to mini-B cable.
- ➢ Connect XB board to PC/NB USB port with USB standard A to micro-B cable.

3.1.2 Software preparation

SW component – See Figure 3.4

- ➢ STM32 ST-Link Utility

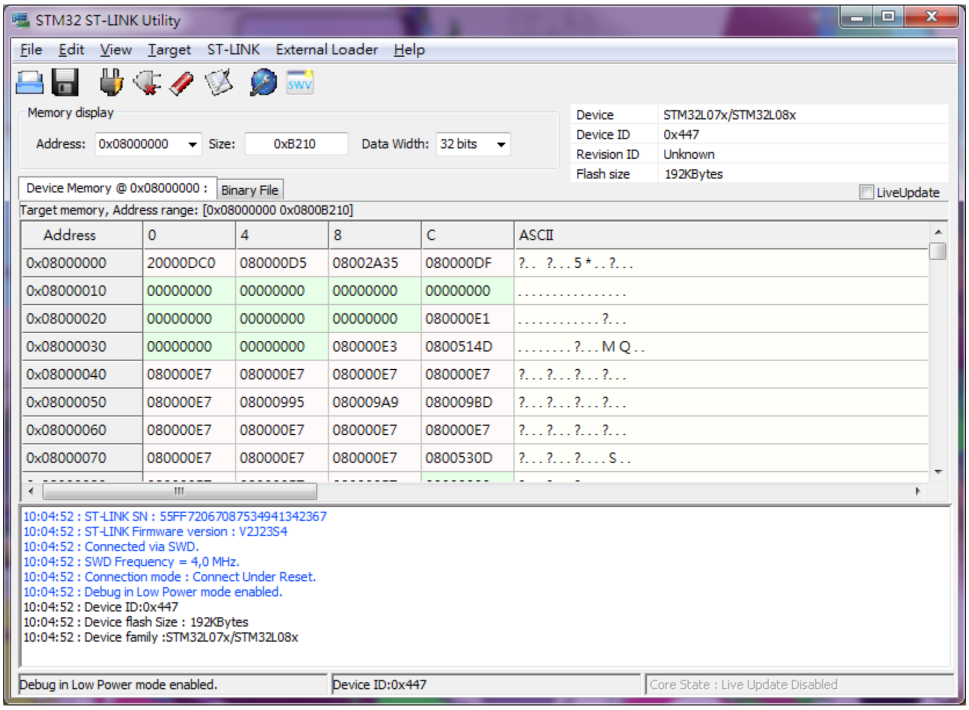


Figure 3.4

3.1.2.1 STM32 ST-LINK/V2 USB Driver Installation

Install ST-LINK/V2 USB Driver for PC/NB.

- Connect to www.st.com.
- In the search tab, part number, look for ST-LINK/V2.
- Download ST-Link/V2 USB driver based on your PC/NB OS and then install it.

3.1.2.2 STM32 ST-LINK Utility Installation

Install STM32 ST-LINK Utility

- Connect to www.st.com.
- In the search tab, part number, look for ST-LINK/V2.
- Click “ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32” link.
- In DEVELOPMENT TOOL SOFTWARE page, to download “STM32 ST-LINK utility”.
- Extract the contents of the downloaded zip file.
- To execute setup.exe and follow the on-screen prompts to install.

3.2 Connection & Disconnection via ST-LINK/V2

3.2.1 Connect ST-LINK

- After installing ST32 LINK/V2 USB driver and STM32 ST-LINK Utility, to execute STM32 ST-LINK Utility.
- If HW & SW installation completes, connection would be establish after click “connect” button.

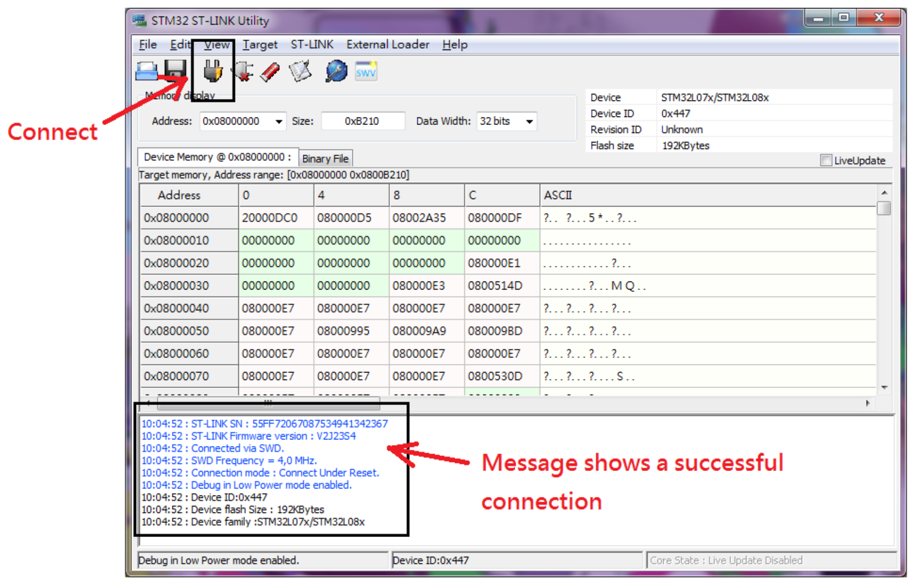


Figure 3.5

3.2.2 Disconnect ST-LINK

click “disconnect” button after program is done

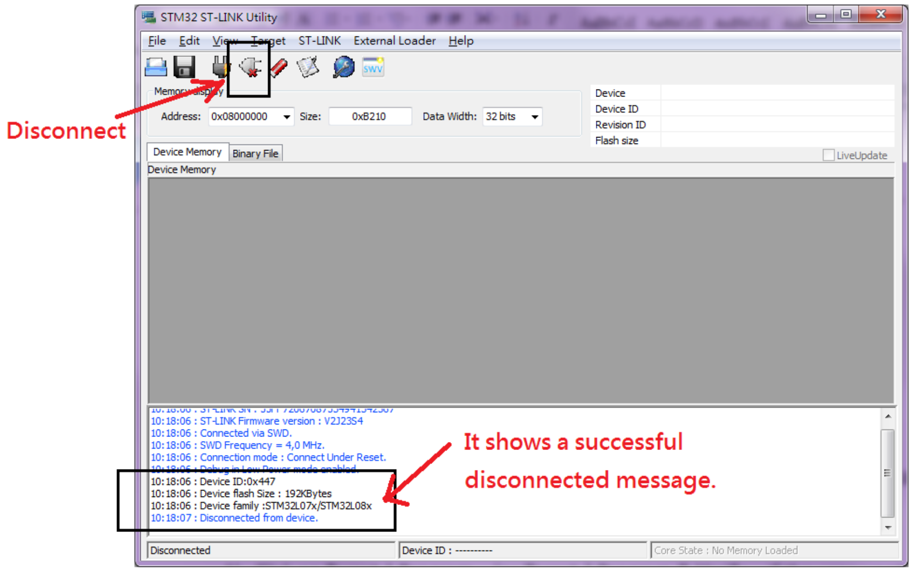


Figure 3.6

3.3 Download Firmware via ST-LINK/V2

The STM32 ST-LINK utility can download Hex image file into Flash. To do this, follow these steps:

1) Click on **Target | Program...** (or **Target | Program & Verify...** if the user wants to verify the written data) to open the Open file dialog box, as shown in Figure 3.7. If a binary file is already opened, go to step 3.
2) Select a hex file and click on the Open button.

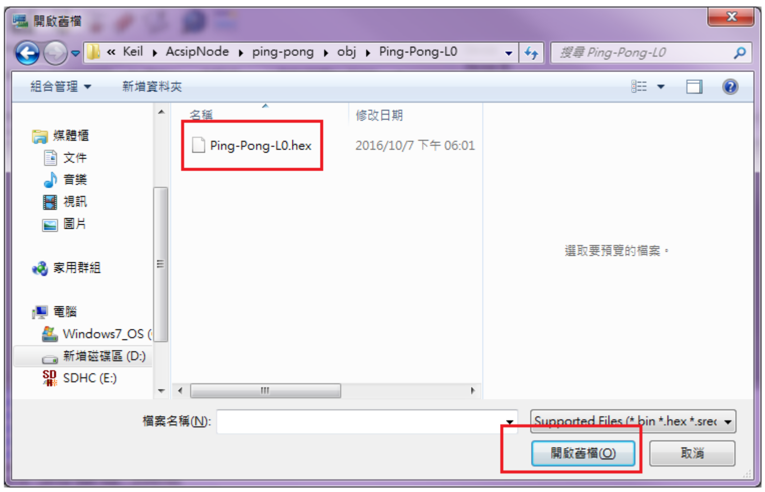


Figure 3.7

3) Specify the address from which to start programming as shown in Figure 3.8, it may be a Flash or RAM address. (Optional)

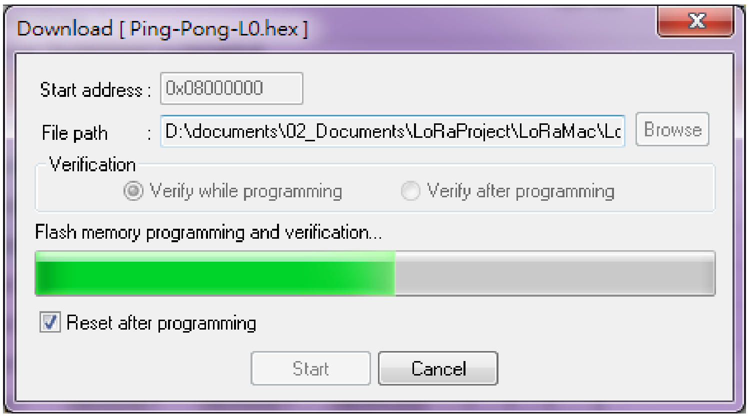


Figure 3.8

4) Choose a verification method by selecting one of the two radio buttons:
   a) **Verify while programming**: fast on-chip verification method which compares the program buffer content (portion of file) with the Flash memory content.
   b) **Verify after programming** (recommended): slow but reliable verification method which reads all the programmed memory zone after the program operation ends and compares it with the file content.
5) At last, click on the Start button to start programming:
   a) If Target | Program & Verify... is selected in the first step, a check is done during the programming operation.
   b) If "**Reset after programming**" box is checked, an MCU reset will be launched.

3.4 Prerequisite for ULINK2

3.4.1 Hardware Preparation

HW component - See below components:

- USB Port x 2 on PC/NB site
- ULINK2 unit x1
- EK-S76SXB/EK-S78SXB board x1
- Mini USB cable x 1 and Micro USB cable x1

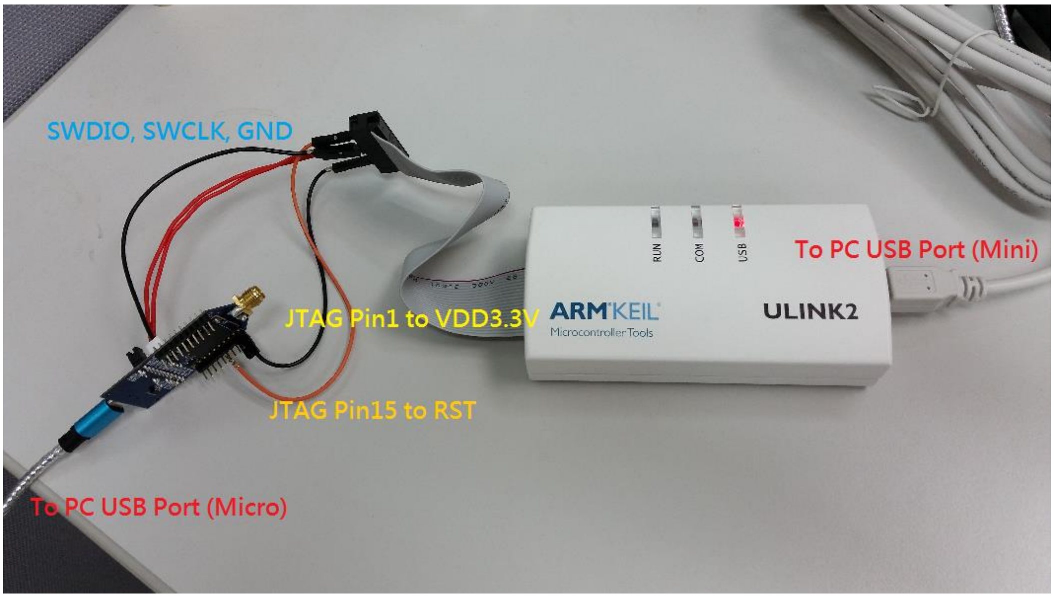


Figure 3.9

Connect HW component as Figure 3.9.

- Connect EK-S76SXB/EK-S78SXB board to ULINK2.
  - ULINK2 JTAG SWDIO Pin7 <==> **SWDIO** on XB board (See Figure 3.12)
  - ULINK2 JTAG SWCLK Pin9 <==> **SWCLK** on XB board
  - ULINK2 JTAG RESET Pin15 <==> **RST** on XB board
  - ULINK2 JTAG GND Pin12 <==> **GND** on XB board
  - ULINK2 JTAG VCC Pin1 <==> **3.3V** on XB board

Figure 3.10

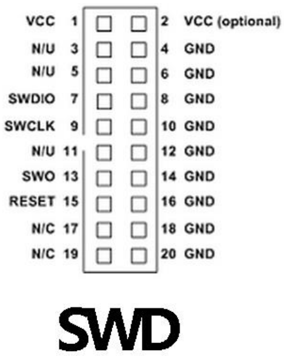


Figure 3.11

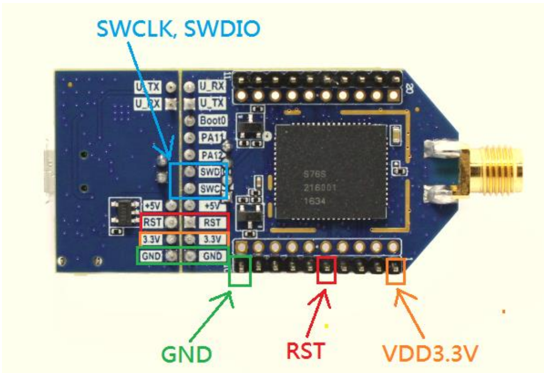


Figure 3.12

- Connect ULINK2 to PC with USB standard A to mini-B cable. (See Figure 3.9)
- Connect XB board to PC/NB USB port with USB standard A to micro-B cable. (See Figure 3.9)

3.4.2 Software Preparation

Please follow chapter 4.1 to complete KEIL installation first.

3.5 Connection & Disconnection via ULINK2

3.5.1 Setup debugger as ULINK2

After finishing KEIL installation, open an AcSiP's SDK project named "**Ping-Pong-L0.uvprojx**" in this path "\Keil\AcsipNode\ping-pong\" by launching uVision5 and looks similar to the picture below (see Figure 3.13).

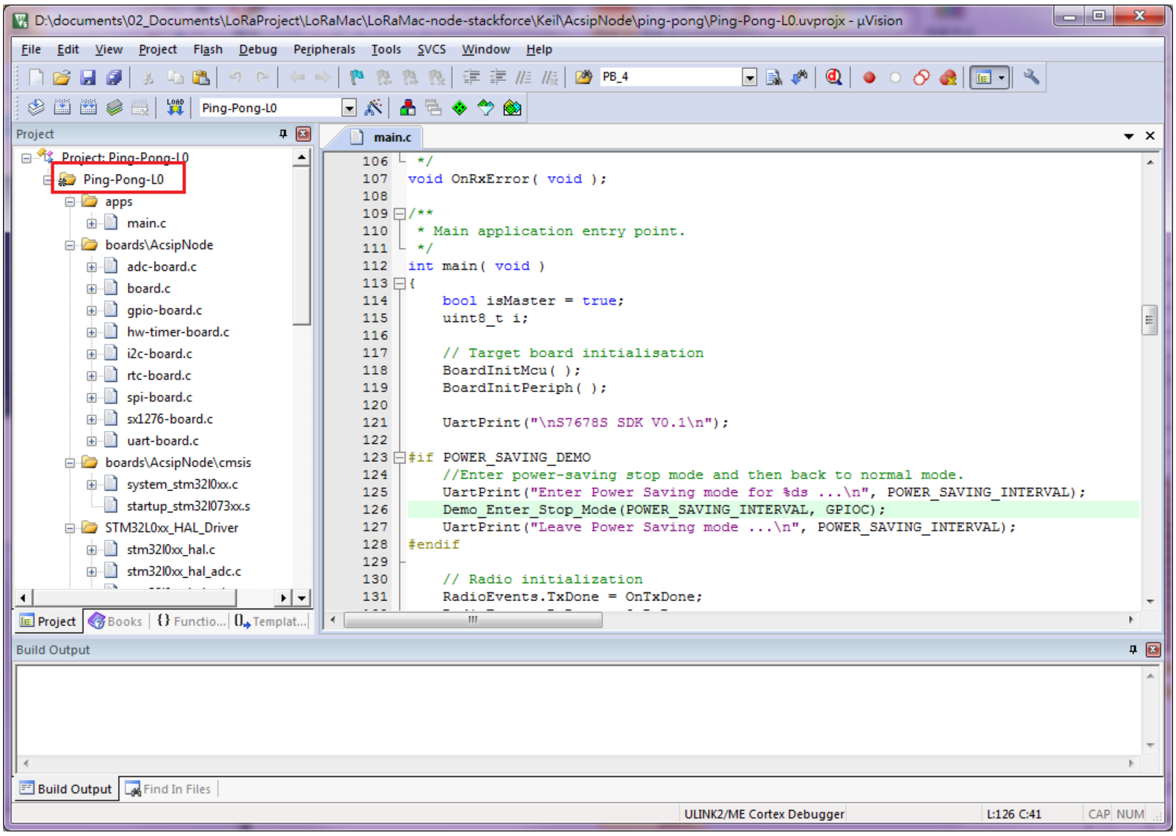


Figure 3.13

Over the position "**Ping-Pong-L0**" and press right button of you mouse. Click "**Options for Target**" on the toolbar and select the **Debug** tab. Verify that the correct debug adapter of the XB board – **ULINK2/ME Cortex Debugger**, is selected as Figure 3.14.

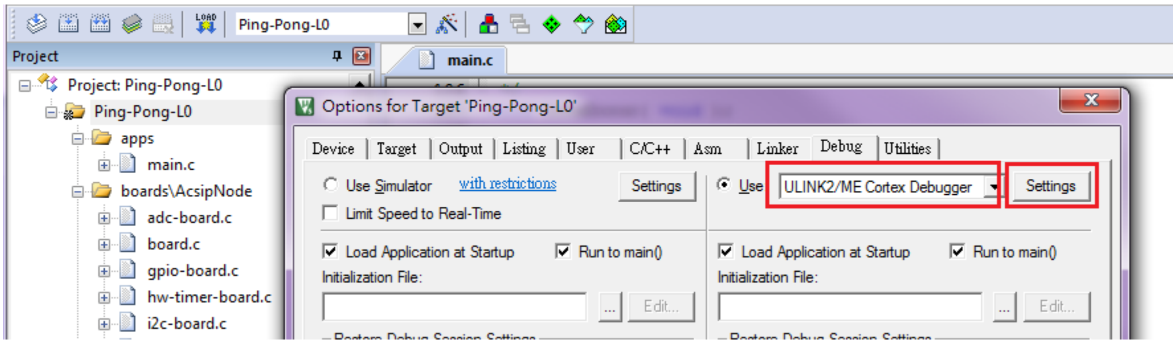


Figure 3.14

After selecting ST-Link Debugger, click "**Setting**" button and then will see the "Debug" tab, if ULINK2 "Serial No." is seen and SWDIO in SW Device is shown, the connection is fine (See Figure 3.15).

Go to "**Flash Download**" tab. In "**Programming Algorithm**", please add a "**STM32L0 192kB Flash**" in the dialog just like the below Figure 3.16.

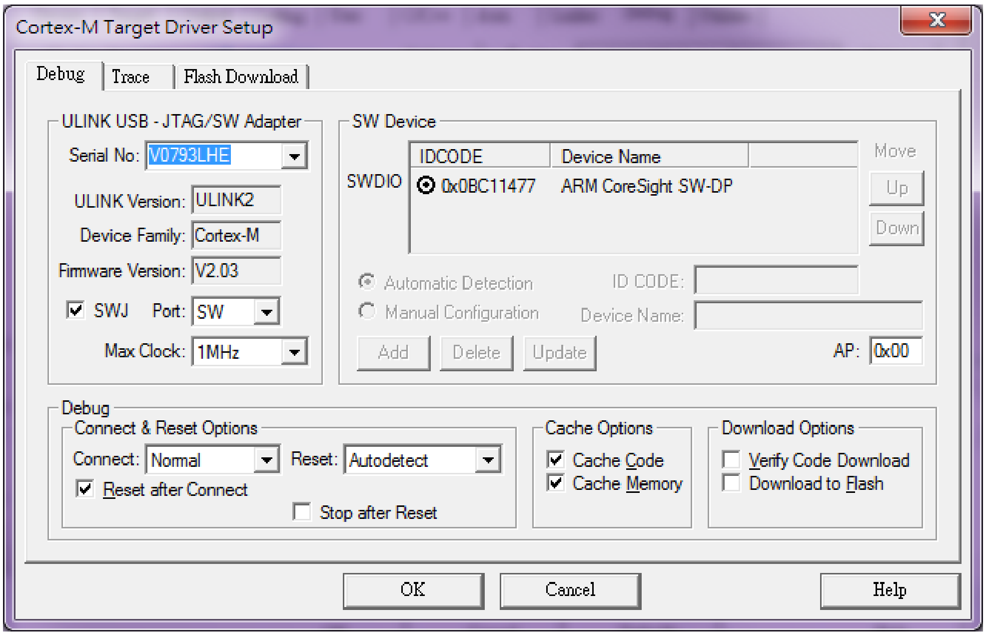


Figure 3.15

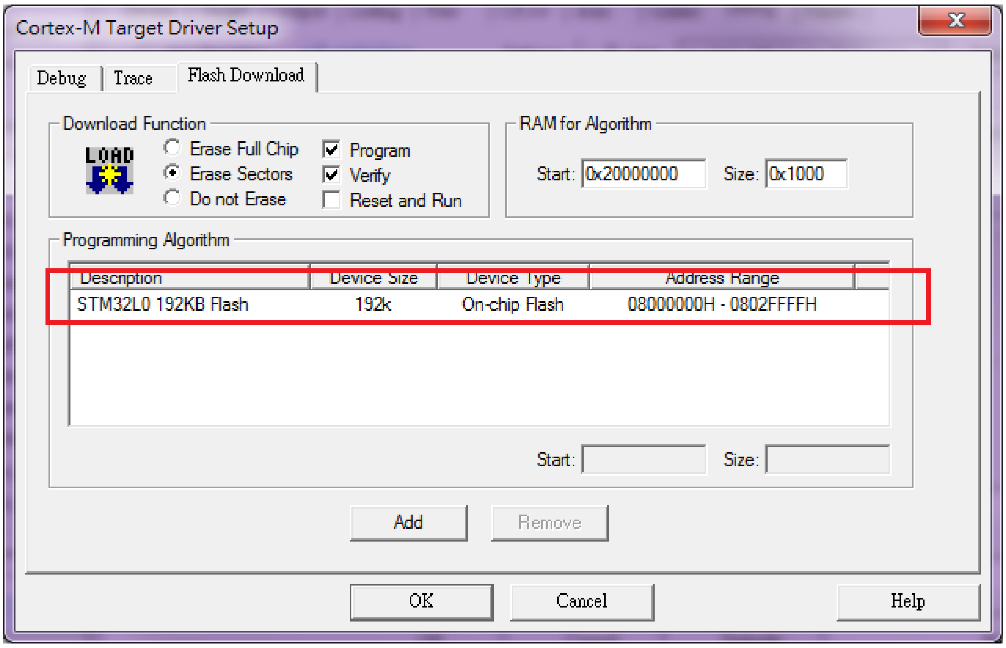


Figure 3.16

4. Developer Environment

We can use Keil MDK (Microcontroller Development Kit) to develop (coding, flashing and debugging) our own code which is based on SW SDK provide by AcSiP.

MDK Core

MDK Core includes all the components that you need to create, build, and debug an embedded application for Cortex-M processor based microcontroller devices. The Pack Installer manages Software Packs that can be added any time to MDK Core. This makes new device support and middleware updates independent from the toolchain.

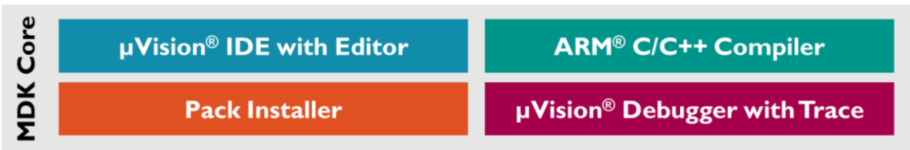


Figure 4.1

Software Packs

Software Packs contain device support, CMSIS libraries, middleware, board support, code templates, and example projects.

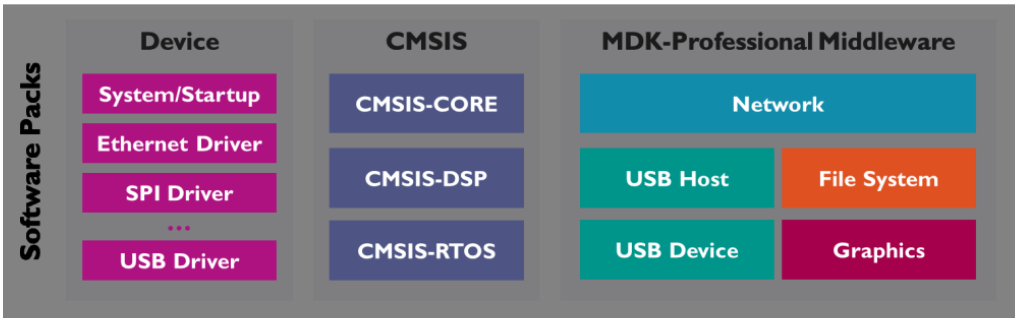


Figure 4.2

CMSIS is a software framework for embedded applications that run on Cortex-M based microcontrollers. It provides consistent software interfaces and hardware abstraction layers (between Cortex-M processor and Suppliers like STM32 or Cypress) that simplify software reuse.

4.1 Installation

MDK is a powerful, easy to learn and use development system. MDK Version 5 consists of the MDK Core plus device-specific Software Packs, which can be downloaded and installed based on the requirements of our application.

Software and Hardware Requirements

MDK has the following minimum hardware and software requirements:

- A PC running Microsoft Windows (32-bit or 64-bit) operating system
- 4 GB RAM and 8 GB hard-disk space
- 1280 x 800 or higher screen resolution; a mouse or other pointing device

Install MDK Core

Download MDK-ARM v5 from www.keil.com/download - Product Downloads and run the installer. Follow the instructions to install the MDK Core on your local computer. The installation also adds the Software Packs for ARM CMSIS and MDK-Professional Middleware.

After the MDK Core installation is complete, the Pack Installer is started automatically, which allows you to add supplementary Software Packs. As a minimum, you need to install a Software Pack that supports your target microcontroller device.

Install MDK legacy

MDK Version 5 is capable of using MDK Version 4 projects after installation of the Legacy Support from www.keil.com/mdk5/legacy, please download “**legacy support for Cortex-M devices Version 5.17**”

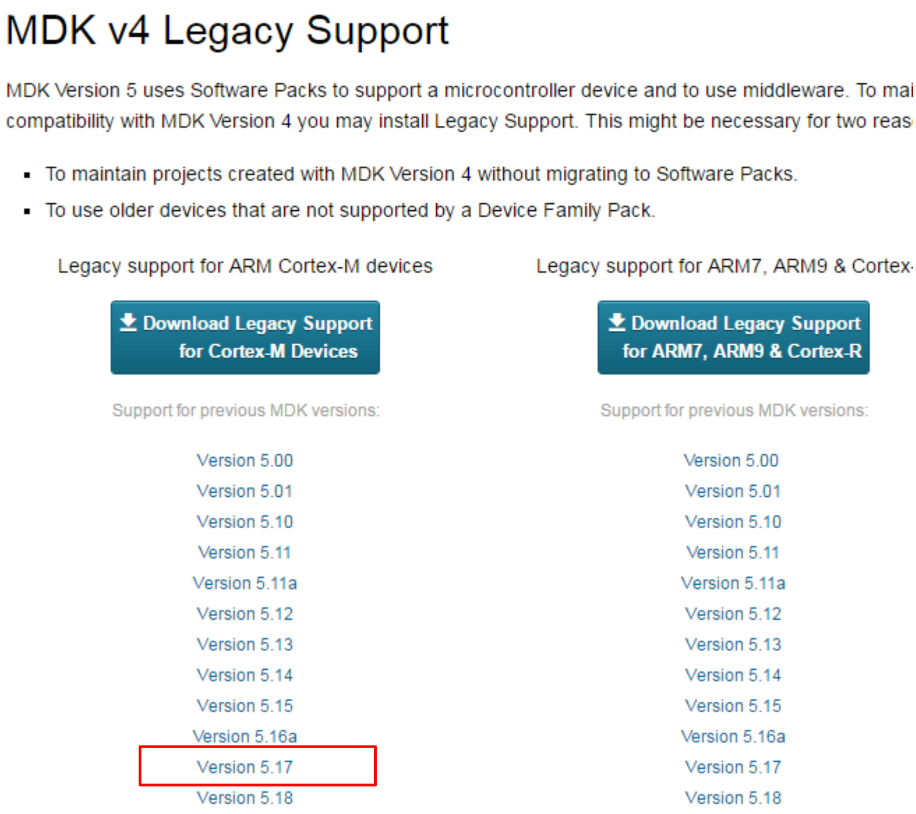


Figure 4.3

Install Software Packs

The Pack Installer is a utility for managing Software Packs on the local computer.

The Pack Installer runs automatically during the MDK installation but also can be run from µVision using the menu item Project – Manage – Pack Installer. To get access to devices and example projects you should install the Software Pack related to your target device or evaluation board.

EK-S76SXB/EK-S78SXB board is using this type, STM32L073RZ. We can update the related packs from “**Device\ All Devices\ STMicroelectronics\ STM32L0 Series\ STM32L073\ STM32L073RZ\ STM32L073RZHx**” as Figure 4.4. Please press the buttons to let them update.

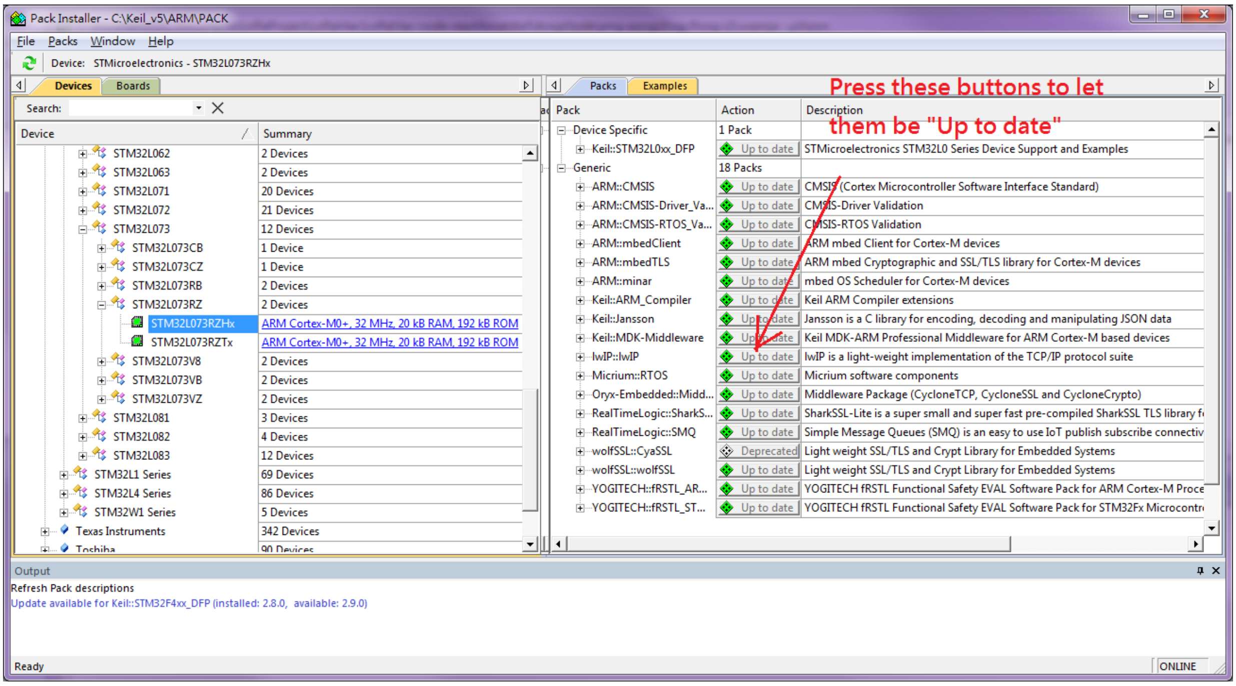


Figure 4.4

4.2 Set-up (Only for ST-LINK/V2)

Open AcSiP's SDK project in this path "**\Keil\AcsipNode\ping-pong\Ping-Pong-L0.uvprojx**" (Explain how to download in following chapter) by launching uVision5 and looks similar to the picture below (Figure 4.5).

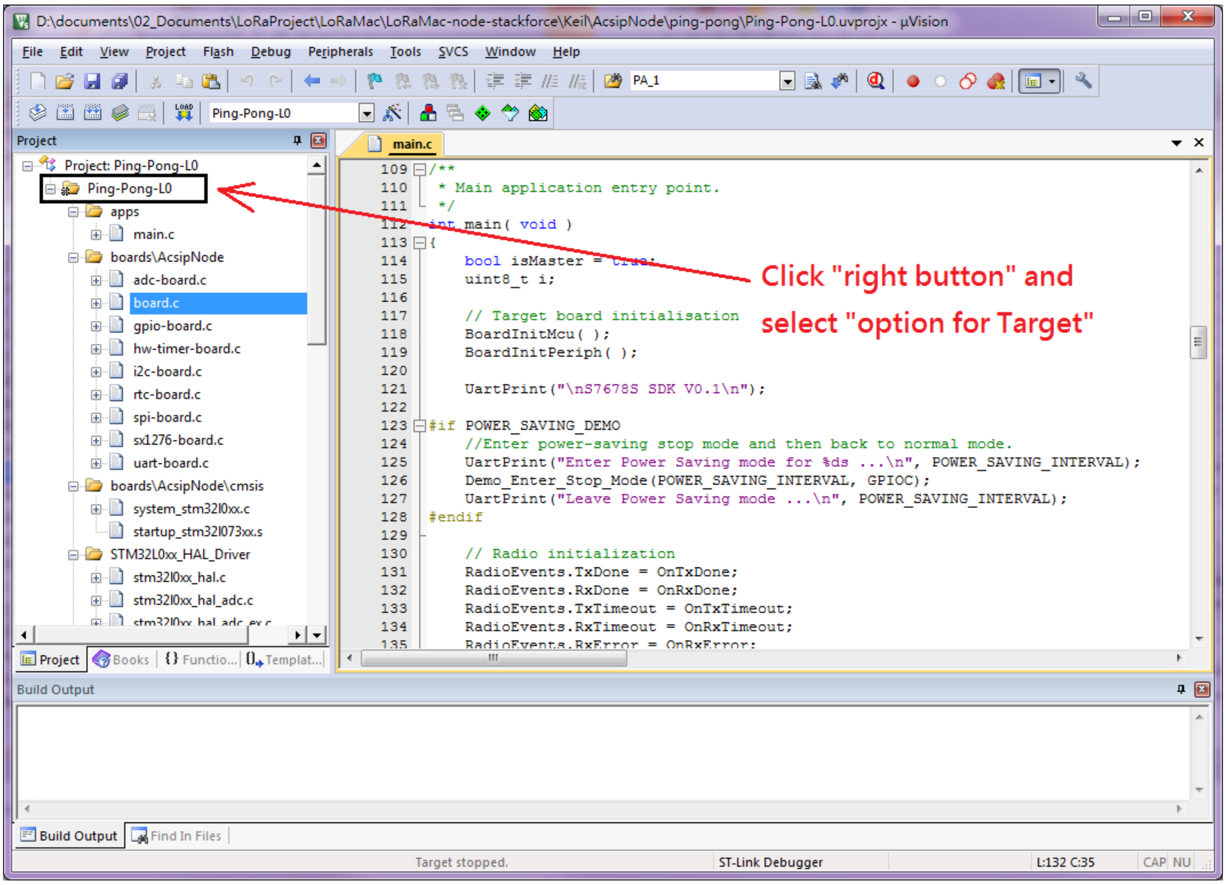


Figure 4.5

Over the position “**Ping-Pong-L0**” and press right button of you mouse. Click “**Options for Target”** on the toolbar and select the **Debug** tab. Verify that the correct debug adapter of the XB board – **ST-Link Debugger**, is selected as Figure 4.6.

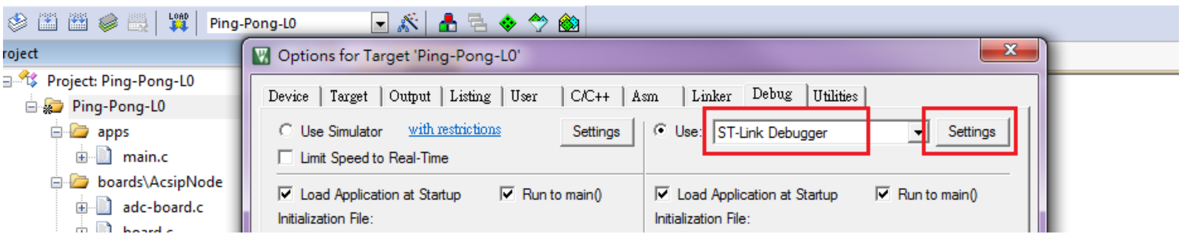


Figure 4.6

After selecting ST-Link Debugger, click “Setting” button and go to “Flash Download” tab. In “Programming Algorithm”, please add a “**STM32L0 192kB Flash**” in the dialog just like the below Figure 4.7.

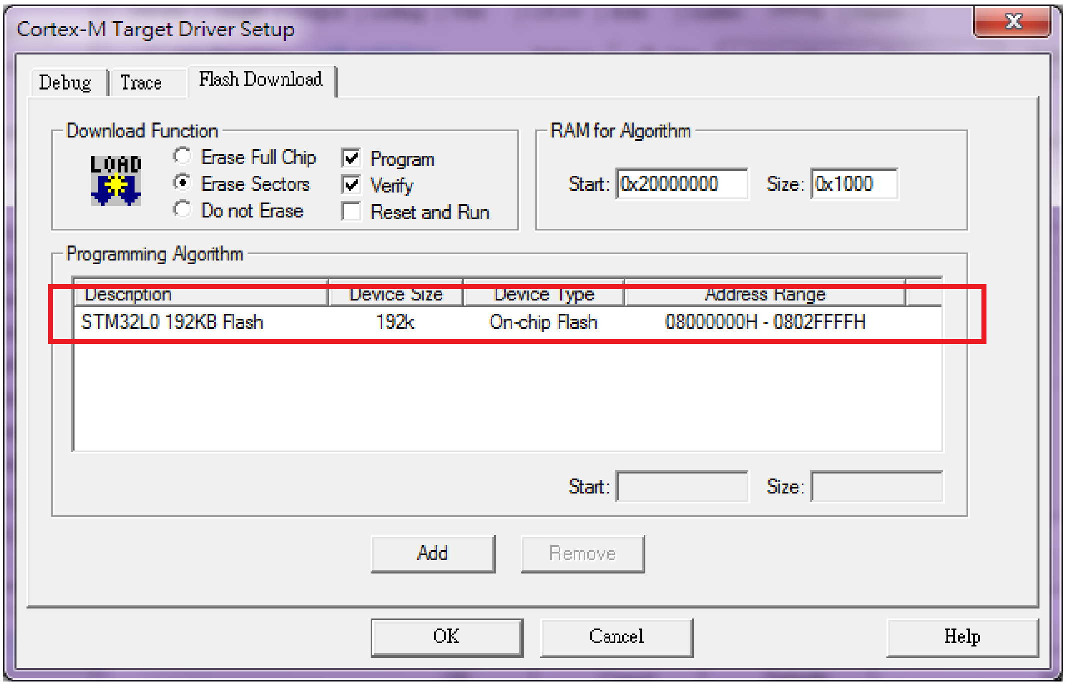
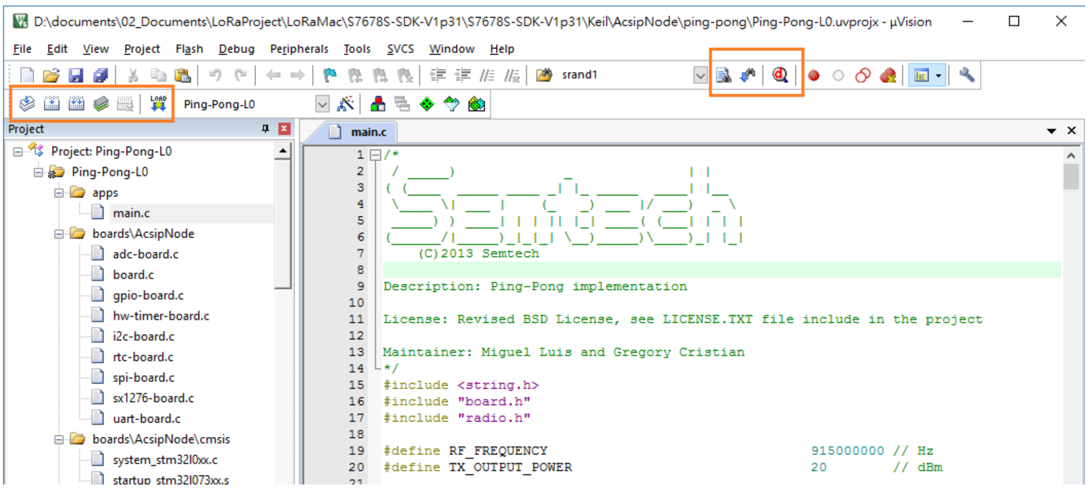


Figure 4.7

4.3 Edit Source and Build

When uVision5 start and loads with the correct setting mentioned above, we can load AcSiP SDK project and do some action of what you want.



1)  Build the application, which compiles and links the related source files.

The **Build Output** window shows information about the download progress as Figure 4.8.

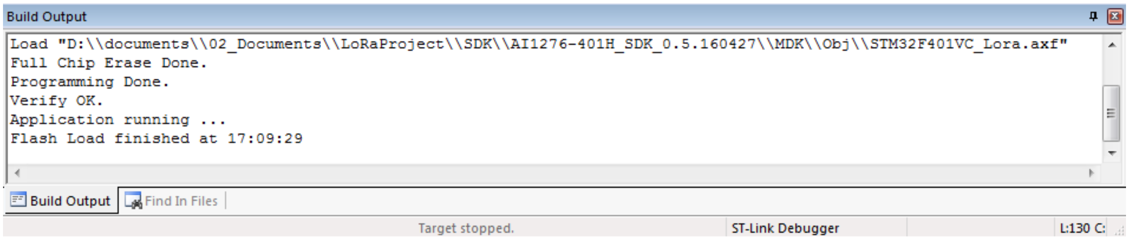


Figure 4.8

2) LOAD Download the application, typically to on-chip Flash ROM of a device.

3) Run the application on the target hardware using ST-Link debugger.

4) Click Run on the debug toolbar to start executing the SDK.

The detailed information of this chapter can be checked from the below link:
http://www2.keil.com/docs/default-source/default-document-library/mdk5-getting-started.pdf?sfvrsn=2[NC,L]

5. SDK Features

AcSiP provides S76S/S78S SDK (Software Development Kit) for customers to customize their own featured product. We have come out some basic functions like **UART debug Logger**, **Ping-Pong Demo** and **Power Saving** for showing the capability of what AcSiP has.

5.1 Features

5.1.1 UART Debug Logger

SDK is occupying UART1 port to receive debug message, and UART1 data is converted into USB interface by UART-To-USB IC on EK-S76/78SXB board. After connecting EK-S76S/S78S with PC/NB by micro USB port, we can start to receive UART debug logger messages through micro USB port directly.

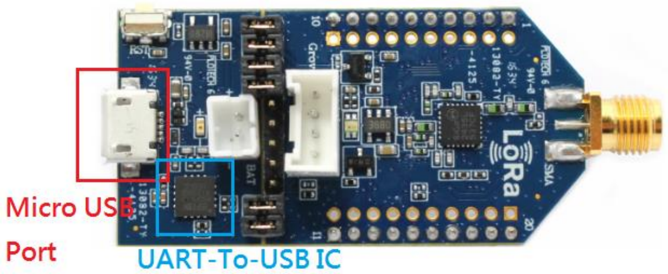


Figure 5.1

The related UART setting, please follow these below settings:

Baud rate: **115200**

Data bits: **8**

Stop bits: **1**

Parity: **none**

Flow Control: **none**

Forward: **none**

5.1.2 Ping-Pong Demonstration

Ping-Pong can demonstrate two LoRa devices communication behavior and measure the signal strength varying with distance. User can use XB board USB interface to get UART1 log data, to know the current RSSI and SNR value when executing Ping-Pong.

a. While a S76S/S78S XB board boots up, it will enter ping-pong master mode by default. Therefore, one of two devices would send "Ping" signal by LoRa first and try to let another XB board to receive (Figure 5.2).

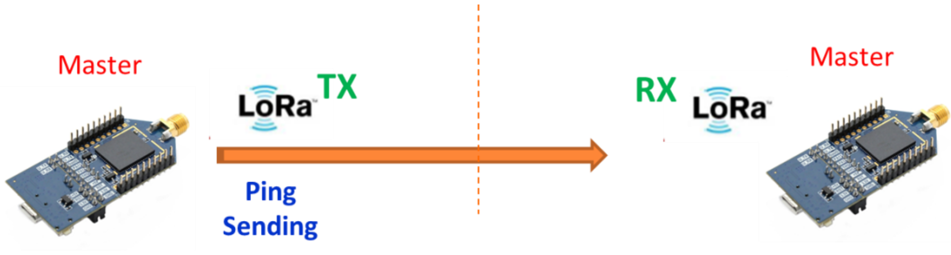


Figure 5.2

b. When the other one gets the “Ping” signal, it would transfer itself to Slave mode from Master mode, and then transfer “Pong” signal from the slave point of view (Figure 5.3).

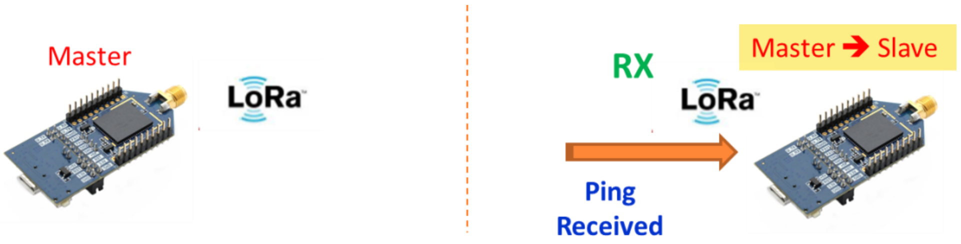


Figure 5.3

c. While Master and Slave roles of two devices confirmed, master mode device could send “Ping” signal and waits to receive “Pong” signal; slave mode device could send “Pong” signal after it receives “Ping” signal as Figure 5.4.

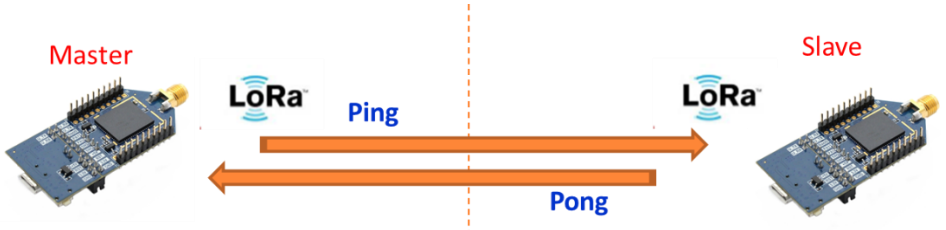


Figure 5.4

Eventually, the two S76S/S78S XB board devices can communicate each other and shows RSSI and SNR information on PC/NB terminal by USB interface (transferred from UART). (See Figure 5.5)

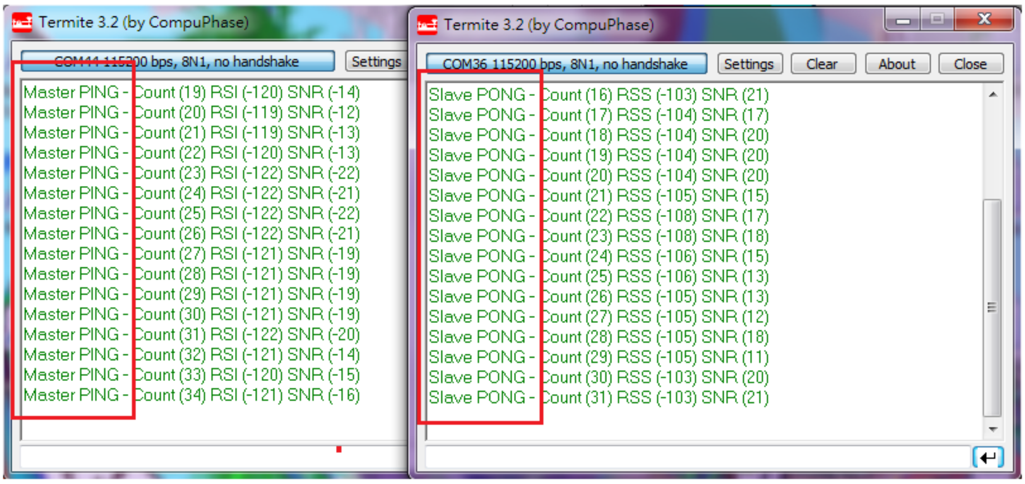


Figure 5.5

5.1.3 Power-Saving

User can be well aware that the importance of when a LoRa device can save its power is curial for a mature product. So the SDK also provide a demonstration of power-saving feature.

Before entering the main loop, ping-pong mode, in main.c c source file, users can select whether the device they're using to enter power-saving stop mode or not by configure this MARCO value, POWER_SAVING_DEMO in board.h header file.

```
#if POWER_SAVING_DEMO
    //Enter power-saving stop mode and then back to normal mode.
    UartPrint("Enter Power Saving mode for %ds ...\n", POWER_SAVING_INTERVAL);
    Demo_Enter_Stop_Mode(POWER_SAVING_INTERVAL, GPIOC);
    UartPrint("Leave Power Saving mode ...\n", POWER_SAVING_INTERVAL);
#endif
```

Figure 5.6

Once this function, Demo_Enter_Stop_Mode(), is called, S76S/S78S would enter power saving from normal mode, so SX1276/78 would enter standby mode to avoid unnecessary energy lost; MCU STM32L073 would enters stop-mode as well, MCU close its system clock and peripheral clock but RTC clock still remains, to achieve the minimum consumption current and waiting the pre-set wake-up time set. The pre-set time interval of power saving mode is given by this parameter, POWER_SAVING_INTERVAL, therefore, user can modify this MARCO value as well, to change the default 6 seconds to other value.

<u>Changeable Parameters in SDK for Power Saving</u>:

POWER_SAVING_DEMO: 1 →Enable Power-Saving Mode

POWER_SAVING_DEMO: 0 →Disable Power-Saving Modes

POWER_SAVING_INTERVAL: 6 (Seconds)

When S76S/S78S entered power saving stop-mode, the total power consumption of these two main components, **STM32L073** & **SX1276/78**, is around **2.7uA**. (The UART-To-USB IC and 5V-To-3V LDO IC are not considered into.)

5.2 Flow Chart

5.2.1 Flow Chart of Main C Source Code

The below flow chart describes when S76S/S78S boots up, it goes through several feature and loops in ping pong mode.

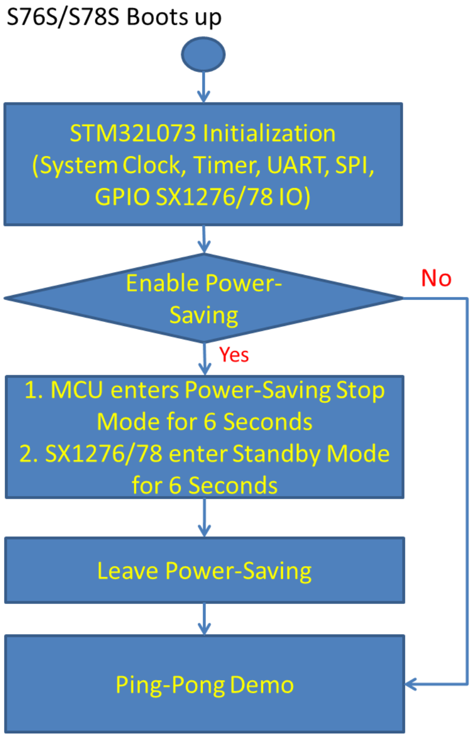


Figure 5.7

Figure 5.8 describes all function blocks inside S76S/S78S SDK. It simply divides as three main featured layers: Application layer, S76S/S78S SDK and S76S/S78S HW.

The mentioned LoRa Ping-Pong behavior is located in Application layer in “main.c”. Users can start to customize their own behavior on this layer like some commands that can interact with external interface (like UART), or start to receive/send sensor data for collection or calculation.

S76S/S78S SDK can be divided as three block: AcSiPNode, STM32L0 HAL driver and SX1276 driver.

AcSiPNode is in charge of all MCU abstractive peripheral behavior like UART, I2C, GPIO, RTC, Timer, ADC, LoRa RF and other I/O control. Hence, when Application layer wants to use UART to send/receive logs to/from PC or sensor board, Application can call the related public functions in *-board.c inside AcSiPNode. Or AcSiPNode can directly operate LoRa RF interface by calling SX1276 driver functions.

As mentioned above, S76S/S78S is using STM32L073 MCU, so ST provides the corresponding HAL (Hardware Abstractive Layer) driver for AcSiPNode board files calling. The physical peripheral implementation is located in this STM32L0 HAL driver block. AcSiPNode need to be adjusted and modified when MCU is changed to other MCU types.

Semtech provides SX1276/SX1278 LoRa modulation IC for S76S and S78S, but the all registers of SX1276 and SX1278 are identical, so S76S/S78S driver can use the same SX driver called sx1276.c. SX1276 driver provide a series of LoRa & FSK related functions, so the basic & advanced LoRa or FSK modulation feature can be implemented in sx1276.c. The detailed register and SX1276/8 specification is listed in this link:
http://www.semtech.com/images/datasheet/sx1276_77_78_79.pdf .

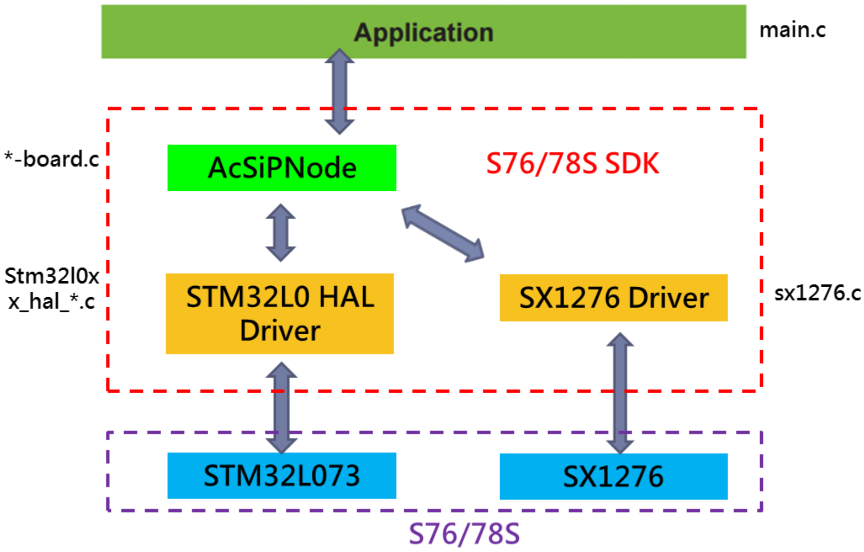


Figure 5.8

6. LoRaWAN™ Deployment

S76S/S78S contains STM32 & SX1276 HW environment for RF transmission by LoRA modulation, providing physical layer for creating long range communication link, and S76S/S78S also has capability to construct LoRaWAN™ protocol for improving battery of life, better security and effective management.

User can utilize AcSiP S76S/S78S SDK MCU & SX1276 ready LoRa® drivers, to develop & customize their own commands and LoRaWAN™ protocol by different regional radiation rules.

6.1 What's LoRaWAN™

LoRaWAN™ defines the communication protocol and system architecture for the network while the LoRa® physical layer enables the long-range communication link. The protocol and network architecture have the most influence in determining the battery lifetime of a node, the network capacity, the quality of service, the security, and the variety of applications served by the network.

The detailed introduction & related resource of LoRaWAN™ can be found by these links:

a. https://www.lora-alliance.org/What-Is-LoRa/LoRaWAN-White-Papers

b. https://www.lora-alliance.org/portals/0/documents/whitepapers/LoRaWAN101.pdf.

Please read them first for reviewing some basic LoRa® & LoRaWAN™ foundation.

6.2 LoRaWAN™ Open Source

For reducing LoRaWAN™protocol (stack) deployment time, there're some existing open source resource that can be referenced and constructed on S76S LoRa®driver. By the basic requirement, the open source LoRaWAN™ stack needs to provide EU868 or US915 bands on Class A and Class C endpoint implementation which is compatible with LoRaWAN™ 1.0.1 specification at least.

By AcSiP's local testing, the open source code, LoRaMac-node, can pass EU868 LoRa-Alliance compliance test. Therefore, combined with AcSiP open source LoRa drivers, user can develop their specialized LoRaWAN™ protocol on S76S/S78S which is compatible the above requirement.

Please take a reference from the below links:

(LoRaMac-node Open Source Link: https://github.com/Lora-net/LoRaMac-node)

(LoRaMac-node stack API documentation: http://stackforce.github.io/LoRaMac-doc/)

6.3 LoRaMac-node works with S76S/S78S SDK LoRa Driver

From the above open source link, checkout the LoRaMac-node source code, user can start to check “src” folder & its subdir folders that contains all open source code; In “Keil” folder, it contains all related uVision (Keil) workable project files.

From the top of either S76S/78S SDK project or LoRaMac-node project, it always starts from Application layer, so user can execute customized peripheral behavior like LoRa ping-pong or simple sensor read/write data in main.c (inside src/app folders).

LoRaMac-node project provides LoRaWAN™ stack implementation, which is included in LoRaMac.c (inside src/mac folder). It implements all LoRaWAN™ & bands behavior like Class A, C or different bands (EU868 or US915).

By the description in chapter 5.3, user can understand the architecture of S76S/S78S SDK. So user can start from S76S/S78S Ping-Pong project and start to migrate (insert) LoRaMac layer into S76/S78 SDK, to replace ping-pong behavior eventually.

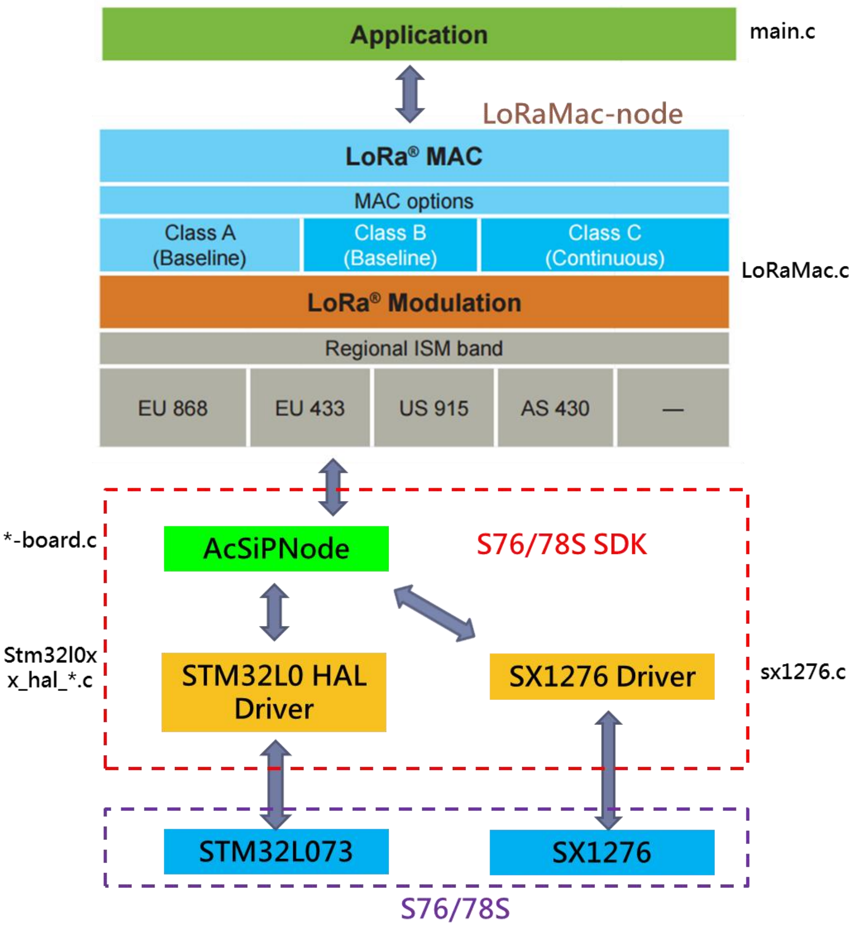


Figure 6.1