



SCM1612

Wi-Fi 6 and BLE 5 Low-Power SoC

SDK Getting Started Guide

Revision 1.3
Date 2024-05-27

Contact Information

Senscomm Semiconductor (www.senscomm.com)
Room 303, International Building, West 2 Suzhou Avenue,
SIP, Suzhou, China
For sales or technical support, please send email to
info@senscomm.com

Disclaimer and Notice

This document is provided on an “as-is” basis only. Senscomm reserves the right to make corrections, improvements and other changes to it or any specification contained herein without further notice.

All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

All third party’s information in this document is provided as is with NO warranties to its authenticity and accuracy.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners and are hereby acknowledged.

© 2024 Senscomm Semiconductor Co.,Ltd. All Rights Reserved.

Senscomm Confidential

Version History

Version	Date	Description
1.3	2024-05-27	Updated UART information.
1.2	2023-09-09	Enhance the EVB guide description.
1.1	2023-08-15	Format Modification
1.0	2023-08-04	Version 1.0 Release
0.1	2023-07-11	Initial Draft

Table of Contents

Version History.....	3
1 Introduction.....	5
2 Setup Development Environment	6
2.1 Linux packages to install	6
2.2 Toolchain	6
2.3 Python and Package	7
2.4 AICE driver for Windows	7
2.5 Terminal Application.....	7
3 Building Firmware.....	8
3.1 Directory Structure	8
3.2 Build	8
3.3 Modifying Configuration	9
3.4 Secure Boot.....	10
3.5 Flash Encryption.....	10
4 Building Host Driver	11
4.1 Build	11
4.1.1 Configuration File Selection and Setup	11
4.1.2 Compiling the sncmfmac.ko Driver	11
4.1.3 Compiling Applications	12
5 Downloading Image.....	13
5.1 Running the bootloader on the RAM.....	16
5.2 Flashing the bootloader for XIP.....	16
5.3 Flashing the main firmware	17
5.4 Running firmware automatically	17
5.5 Updating firmware only	18
6 Debugging.....	19
6.1 JTAG Debugging with OpenOCD	19
6.1.1 General Description.....	19
6.1.2 Set-up Debugging Environment	20
6.1.3 Remote debugging	21

1 Introduction

This document explains the basics: how to setup the development environment, build the SDK and host drivers (if needed), and how to download and run the firmware.

The SDK file provided by Senscomm contains the following directories.

- Doc
- Software
- Toolchain

2 Setup Development Environment

To build scm1612 device firmware, a 64-bit Linux PC is recommended. It is recommended that Ubuntu 20.04 or later is used. In the following sections, the procedure assumes that the user has an Ubuntu.

2.1 Linux packages to install

Install the following packages for the basic build.

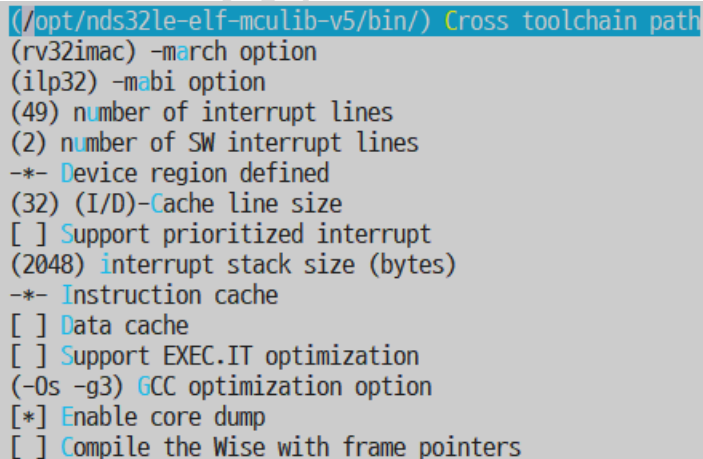
```
$ sudo apt install build-essential libncurses-dev  
$ sudo apt-get install libevent-dev libnl-3-dev libnl-genl-3-dev
```

2.2 Toolchain

Extract the toolchain to the '/opt/' directory.

```
$ sudo tar xvf nds32le-elf-mculib-v5.tar.gz -C /opt/
```

By default, the SDK assumes the toolchain path to be '/opt/nds32le-elf-mculib-v5/bin/'. If the toolchain is extracted elsewhere, then the path must be changed accordingly using 'make menuconfig' before continuing to build.



```
(/opt/nds32le-elf-mculib-v5/bin/) Cross toolchain path  
(rv32imac) -march option  
(ilp32) -mabi option  
(49) number of interrupt lines  
(2) number of SW interrupt lines  
*- Device region defined  
(32) (I/D)-Cache line size  
[ ] Support prioritized interrupt  
(2048) interrupt stack size (bytes)  
*- Instruction cache  
[ ] Data cache  
[ ] Support EXEC.IT optimization  
(-Os -g3) GCC optimization option  
[*] Enable core dump  
[ ] Compile the Wise with frame pointers
```

2.3 Python and Package

Some of the build process includes running python scripts. Install the following python packages.

```
$ sudo apt install python3-pip
$ pip install pycryptodome
$ pip install imgtool
```

Since the packages are often installed `/home/{user}/.local/bin`, the path must be added to the PATH variable.

Edit `~/.bashrc` file, and the following line.

```
export PATH=$PATH:~/.local/bin
```

Another method would be to create a directory named 'bin' under the directory, and the Linux distribution will automatically add '`~/bin`' and '`~/.local/bin`' to the PATH.

2.4 AICE driver for Windows

For debugging with JTAG, AICE adapter is required.

- Install AICE driver

2.5 Terminal Application

A serial terminal application is needed to

- download the firmware via UART
- view the console logs

3 Building Firmware

3.1 Directory Structure

The top directory of the SDK contains the following sub-directories.

Directory	Description
api	Senscomm WiFi APIs
app	Sample applications
configs	Default configurations to be used
hal	Hardware abstraction layer
include	Header files to be included
kernel	Kernel with FreeRTOS and FreeBSD
lib	Useful library modules
prebuilt	Libraries built with pre-defined configuration
scripts	Build related scripts

SCM1612 has a built-in ROM inside the SoC, and the ROM has some of the commonly used software components such C libraries, OS, WLAN driver, network libraries, and BLE controller stack. These software components have header files only. Application can call the functions in the headers as if they are provided as normal software libraries.

3.2 Build

Any files present from the previous build process can be cleared by 'make distclean' command. It is safer to clean the tree before changing any of the configurations.

From the configs directories, user can find a proper configuration as a starting point, and use it to build that specific configuration.

To build a bootloader to be executed on the RAM,

```
$ make distclean
$ make scm1612s_bl_ram_defconfig
$ make
```


This will generate 'wise.scmboot.ram.bin'

To build a bootloader to be flashed,

```
$ make distclean
$ make scm1612s_bl_defconfig
$ make
```

This will generate 'wise.scmboot.bin'

To build a standalone firmware

```
$ make distclean
$ make scm1612s_defconfig
$ make
```

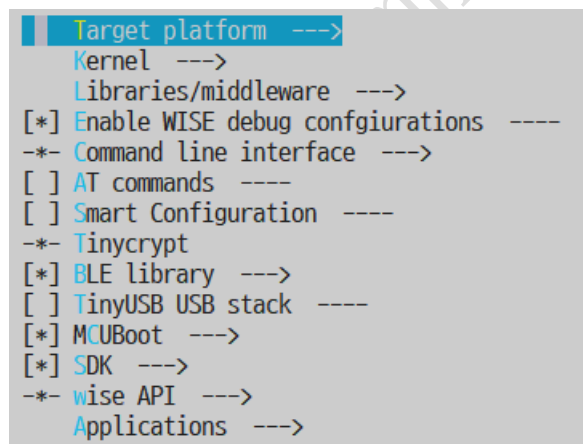
This will generate 'wise.mcuboot.bin'

When the build process is complete, wise.xxxx.bin file is generated. The binary file name can be slightly different depending on the configuration used.

3.3 Modifying Configuration

The build system utilizes the traditional Kconfig and Kbuild. If any of the configuration options should be changed, run 'make menuconfig' and navigate to modify the options as required.

```
$ make menuconfig
```



```
Target platform --->
  Kernel --->
  Libraries/middleware --->
[*] Enable WISE debug configurations ----
-* Command line interface --->
[ ] AT commands ----
[ ] Smart Configuration ----
-* Tinycrypt
[*] BLE library --->
[ ] TinyUSB USB stack ----
[*] MCUBoot --->
[*] SDK --->
-* wise API --->
  Applications --->
```

3.4 Secure Boot

Secure Boot is supported to enable booting the authenticated firmware only. For more information, contact [Senscomm](#) Technical Support.

3.5 Flash Encryption

Flash Encryption is supported to protect the firmware. The firmware will be unusable even if it is flashed to another SCM1612. eFuse must be written with the corresponding security credentials. For more information, contact [Senscomm](#).

Senscomm Confidential

4 Building Host Driver

This section is relevant only to the users wishing to have SCM1612 as a WiFi interface to a host platform via SDIO or USB,

If SCM1612 is connected to the host platform, the corresponding kernel driver must be installed and run on the host.

4.1 Build

4.1.1 Configuration File Selection and Setup

```
cd xiaohu-ax/  
cp configs/cfg_XXX.mk cfg.mk
```

Given the support for multiple platforms and configuration modes, such as SDIO OOB mode, SDIO INT mode, USB mode, etc., you need to select a configuration file based on your platform and requirements for compilation. The system provides default configuration files stored in the configs/ directory:

config	Platform/SDIO Mode
cfg_sdio_normal_fullhan.mk	Fullhan, 4 bit mode interrupt
cfg_sdio_normal_goke.mk	Goke, 4 bit mode interrupt
cfg_sdio_normal.mk	X86/X64, 4 bit mode interrupt
cfg_sdio_polling.mk	X86/X64, Polling mode
cfg_sdio_oob_int_goke.mk	Goke, OOB interrupt mode
cfg_usb.mk	X86/X64

4.1.2 Compiling the sncmfmac.ko Driver

While ARCH and CROSS_COMPILE are already configured in the file, the 'KDIR' parameter needs to be explicitly specified based on your environment. KDIR=/home/apache/page/xxx specifies the location where the kernel resides.

```
make KDIR=/home/apache/page/linux-4.9
```

If 'KDIR' is not specified, the system will use the default kernel KDIR, and the compiled driver will be suitable for running on a Linux PC.

```
make
```

4.1.3 Compiling Applications

```
make KDIR=/home/apache/page/linux-4.9 apps
```

Currently, there are two types of applications:

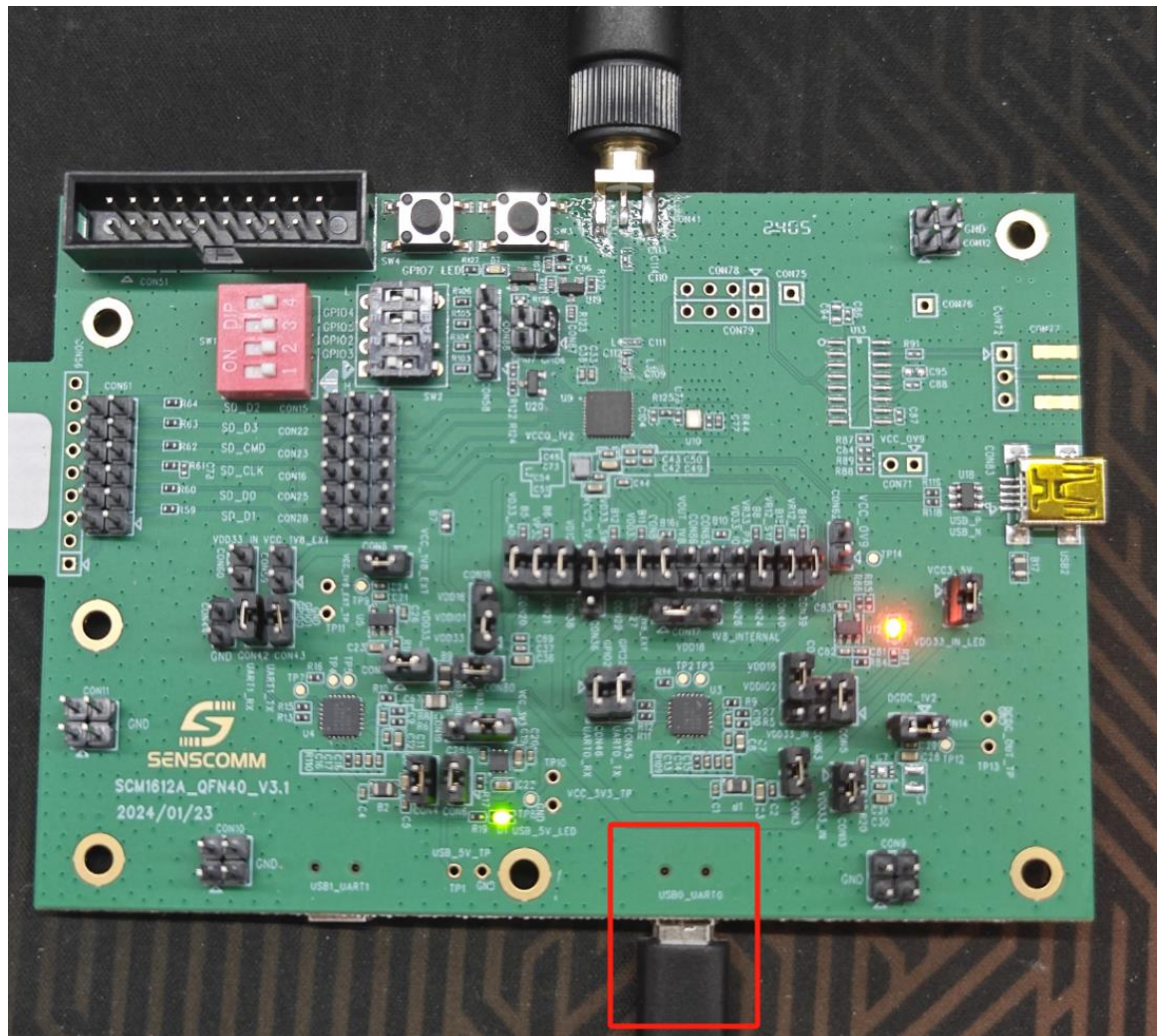
- 'sncm_cmd': Configures and controls Wi-Fi through host commands.
- 'sncm_chn' (also known as ScmChannel): Wi-Fi configuration and control are completed on the SCM1612 side, obtaining Wi-Fi connection information through ScmChannel.
 - 'sample_link' is mainly used to synchronize network node information such as Mac address, IP address, etc., on the SCM1612 side.
 - 'sample_cli': Mainly used to send custom information from the client.

5 Downloading Image

The boot mode of SCM1612 is determined by the GPIO settings.

Boot Mode1 (GPIO3)	Boot Mode0 (GPIO2)	Mode
1	1	FLASH (boot from flash)
0	1	UART (download firmware- from UART)
1	0	USB (download firmware from USB)
0	0	SDIO (download firmware from SDIO)

- Please note that on the EVB board of SCM2010_QFN40_V2.0, there is a labeling error on SW2, where the labels for GPIO2 and GPIO3 have been swapped.
- On a Windows computer, user must use a tool that supports the Ymodem transfer protocol, and we recommend using Tera Term.
- Locate the UART driver for Windows within the release file, CP210x_VCP_Windows.
- Need to connect USB0_UART0 to the computer.

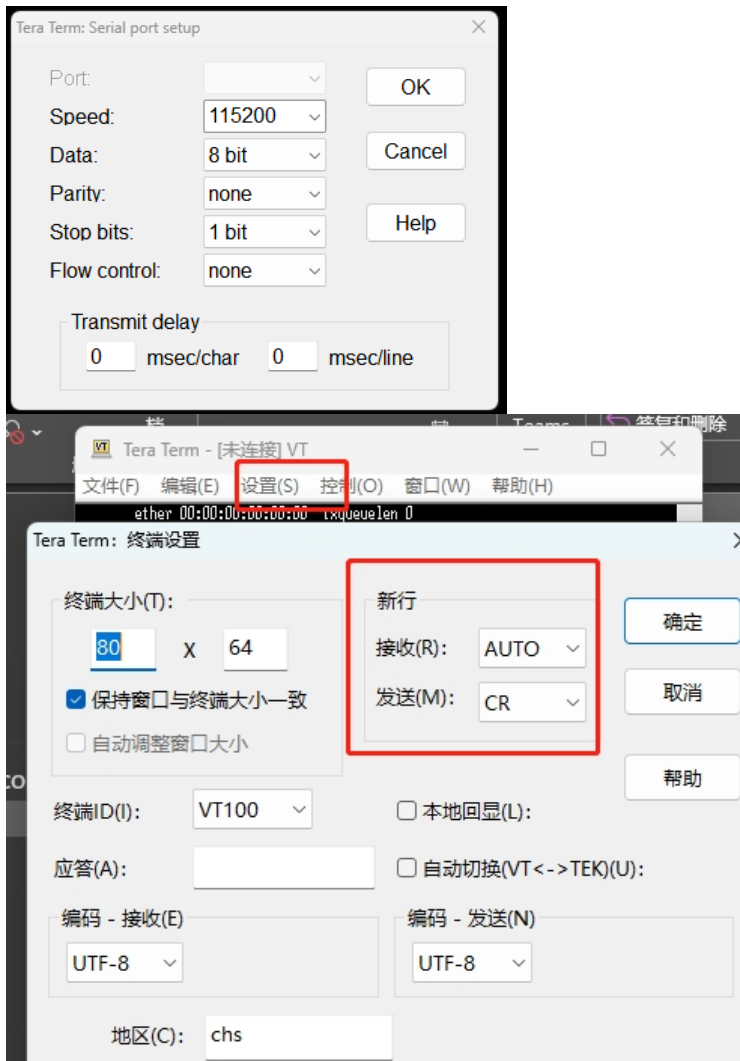


For the UART Boot Mode, SCM1612 will wait for the user to download a RAM executable firmware.

For the XIP Boot Mode, SCM1612 will boot from the flash memory upon power up.

For USB/SDIO download modes, they are typically used in Host mode, where executable programs are loaded into RAM via USB/SDIO and then executed.

There are two UART interfaces in SCM1612. For the UART Boot Mode, the serial terminal should be set as below.



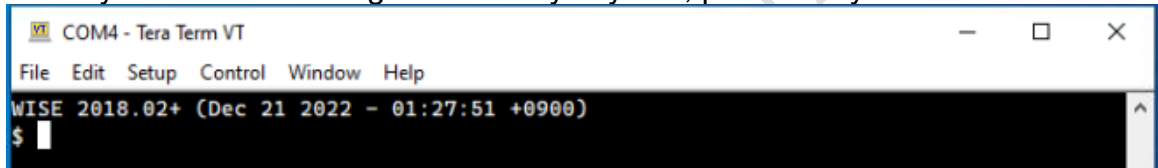
If the XIP bootloader is not flashed yet, the general procedure to flash SCM1612 are as follows.

1. Set SW2 to UART boot mode, download and run the RAM bootloader
2. Flash bootloader for XIP
3. Flash firmware for XIP
4. Set SW2 to XIP boot mode, reset and run automatically

Once the XIP bootloader is flashed (as in step 2. above), users can continue development with step 3 only.

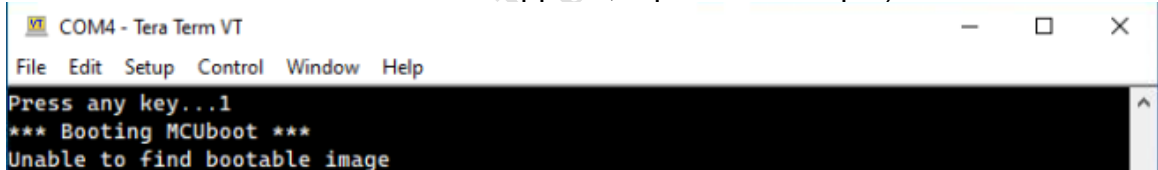
5.1 Running the bootloader on the RAM

1. Set UART boot mode and reset the board.
2. UART0 Teraterm will show “C” indicating that the board is waiting for the firmware to be transferred.
3. Send “wise.scmboot.ram.bin”
 - A. From the Teraterm menu, [File] -> [Transfer] -> [Ymodem] -> [Send] -> Select “wise.scmboot.ram.bin” from the directory where it is
 - B. Wait for the firmware to be transferred completely.
4. When firmware is transferred successfully, it will be executed.
5. UART0 Teraterm is used afterwards once the bootloader is executed. Interrupt the bootloader execution to enter the bootloader shell. When you see the message “Press any key...3”, press a key.



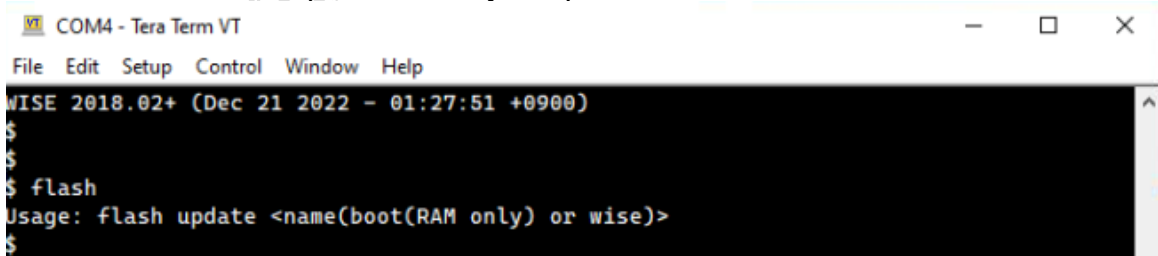
```
COM4 - Tera Term VT
File Edit Setup Control Window Help
WISE 2018.02+ (Dec 21 2022 - 01:27:51 +0900)
$
```

6. If there is no key input from the user, then the bootloader will try to execute the main firmware. If this happens, repeat from step 2) above



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
Press any key...1
*** Booting MCUBoot ***
Unable to find bootable image
```

7. When the bootloader is interrupted by pressing a key, the Wise shell is available for further operations. Try “help” or “flash” commands.

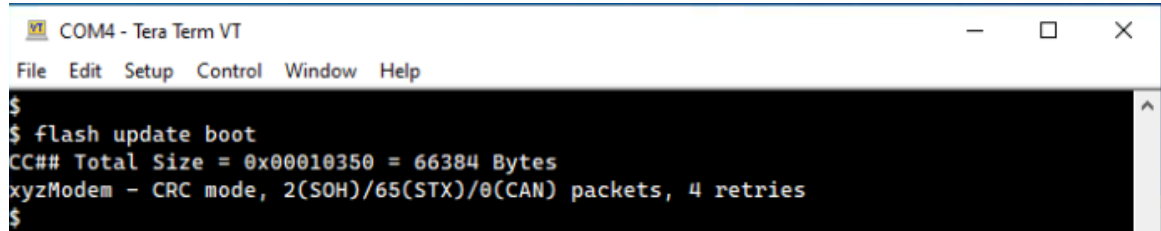


```
COM4 - Tera Term VT
File Edit Setup Control Window Help
WISE 2018.02+ (Dec 21 2022 - 01:27:51 +0900)
$
$
$
$ flash
Usage: flash update <name(boot(RAM only) or wise)>
```

5.2 Flashing the bootloader for XIP

1. Enter the following command
\$ flash update boot

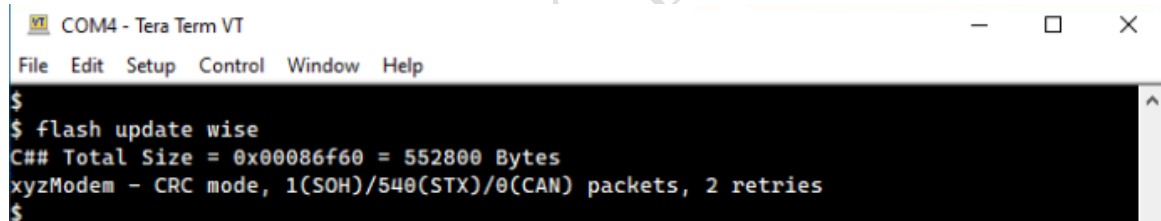
2. From the Teraterm menu, [File] -> [Transfer] -> [Ymodem] -> [Send] -> Select "wise.scmboot.bin"



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
$
$ flash update boot
CC## Total Size = 0x00010350 = 66384 Bytes
xyzModem - CRC mode, 2(SOH)/65(STX)/0(CAN) packets, 4 retries
$
```

5.3 Flashing the main firmware

1. Enter the following command
`$ flash update wise`
2. From the Teraterm menu, [File] -> [Transfer] -> [Ymodem] -> [Send] -> Select "wise.mcuboot.bin"



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
$
$ flash update wise
C## Total Size = 0x00086f60 = 552800 Bytes
xyzModem - CRC mode, 1(SOH)/540(STX)/0(CAN) packets, 2 retries
$
```

5.4 Running firmware automatically

When all the firmware files are flashed, switch to XIP boot mode, and reset the board.

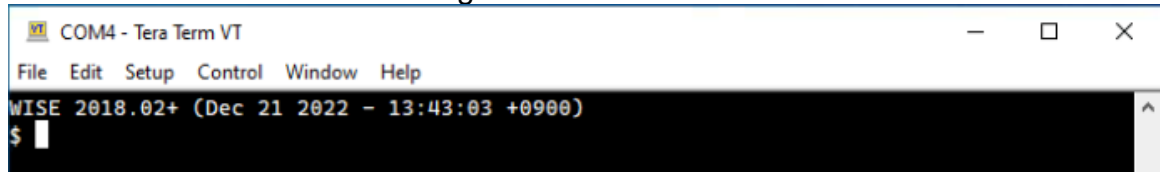
After successful boot,

- 1) UART0 shows bootloader log first



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
Press any key...1
*** Booting MCUboot ***
Booting from 0x80040000Hello world!
```

2) UART0 shows main firmware log next



5.5 Updating firmware only

Once the XIP bootloader is flashed, firmware can be updated again using the XIP bootloader. The XIP bootloader provides the same features as the RAM Bootloader.

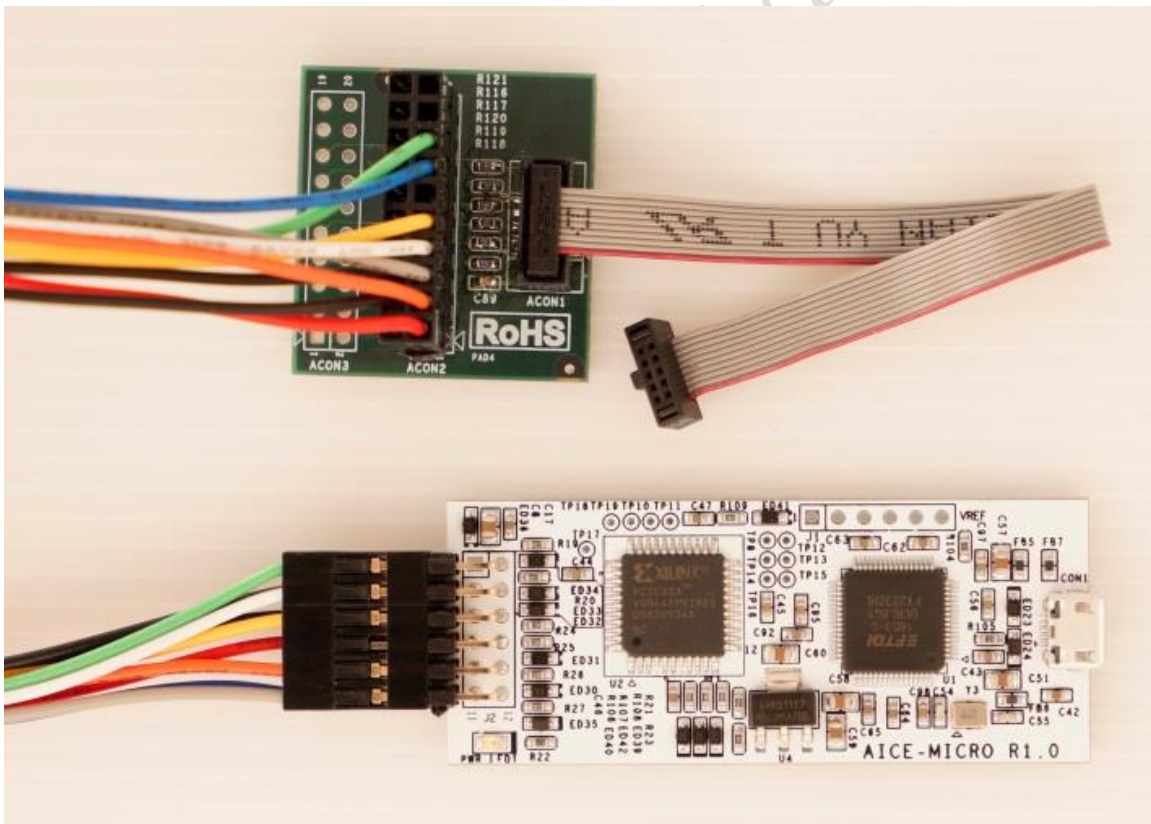
Interrupt the XIP bootloader and follow the same procedures in [section 5.3](#).

6 Debugging

6.1 JTAG Debugging with OpenOCD

6.1.1 General Description

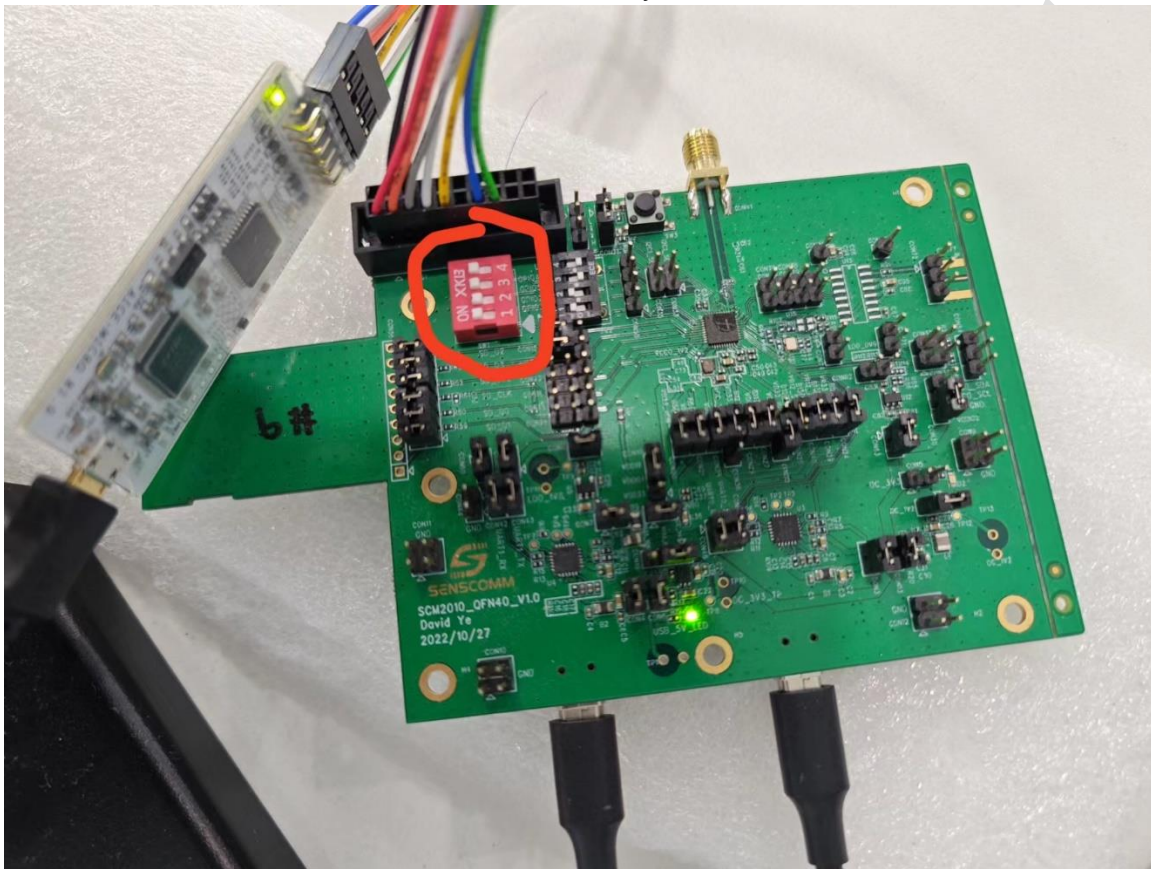
AndeShape AICE-MICRO is a FT2232H based JTAG debugging device, which works with AndeSight™ Development Suite and AndesCore V5 Families through JTAG interface supported by OpenOCD.



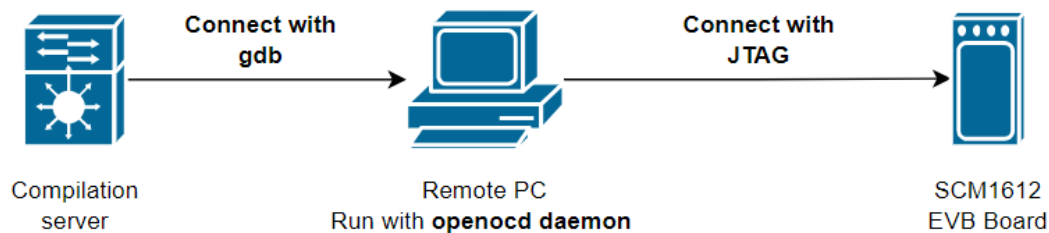
6.1.2 Set-up Debugging Environment

Connecting the JTAG interface to the 1612 EVB board. Please highlight the red-circled area in the image.

During the reboot process, ensure that SW1 switches 1-4 are all set to the down position to complete the boot operation. Once the system has restarted, pull up switches 1-4 to activate the JTAG functionality.



The network topology is as shown in the diagram below.



6.1.3 Remote debugging

- a) Run openocd.exe daemon on remote PC.

```
C:\Work\openocd>openocd.exe
Open On-Chip Debugger 0.10.0+dev-ge990efd64-dirty (2022-06-13-15:31)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
JTAG frequency 10.000 MHz
The core #0 listens on 1234.
The core #1 listens on 1235.
ICEman is ready to use.
|
```

On the compilation server, use `nds32le-elf-mculib-v5` to install `riscv32-elf-gdb`

With the following command, remote debugging can be conducted:

```
/opt/nds32le-elf-mculib-v5/bin/riscv32-elf-gdb
target remote 10.12.7.102:1234
```

Note: The IP address 10.12.7.102 is designated for the host connecting to the JTAG, and 1234 is the target port.

```
GNU gdb (2022-02-07_riscv32-elf-0278d8cc40b) 8.2.50.20190322-git
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-pc-linux-gnu --target=riscv32-elf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
[info] Loading .Andesgdbinit.
[info] .Andesgdbinit loaded.
(gdb) target remote 10.12.7.102:1234
Remote debugging using 10.12.7.102:1234
warning: No executable has been specified and target does not support
determining executable automatically. Try using the "file" command.
0x00000056 in ?? ()
tap0_target_0(gdb) █
```