GigaDevice Semiconductor Inc.

GD32W51x Rapid Development Guide

Application Note

AN079

Table of Contents

List of Figures

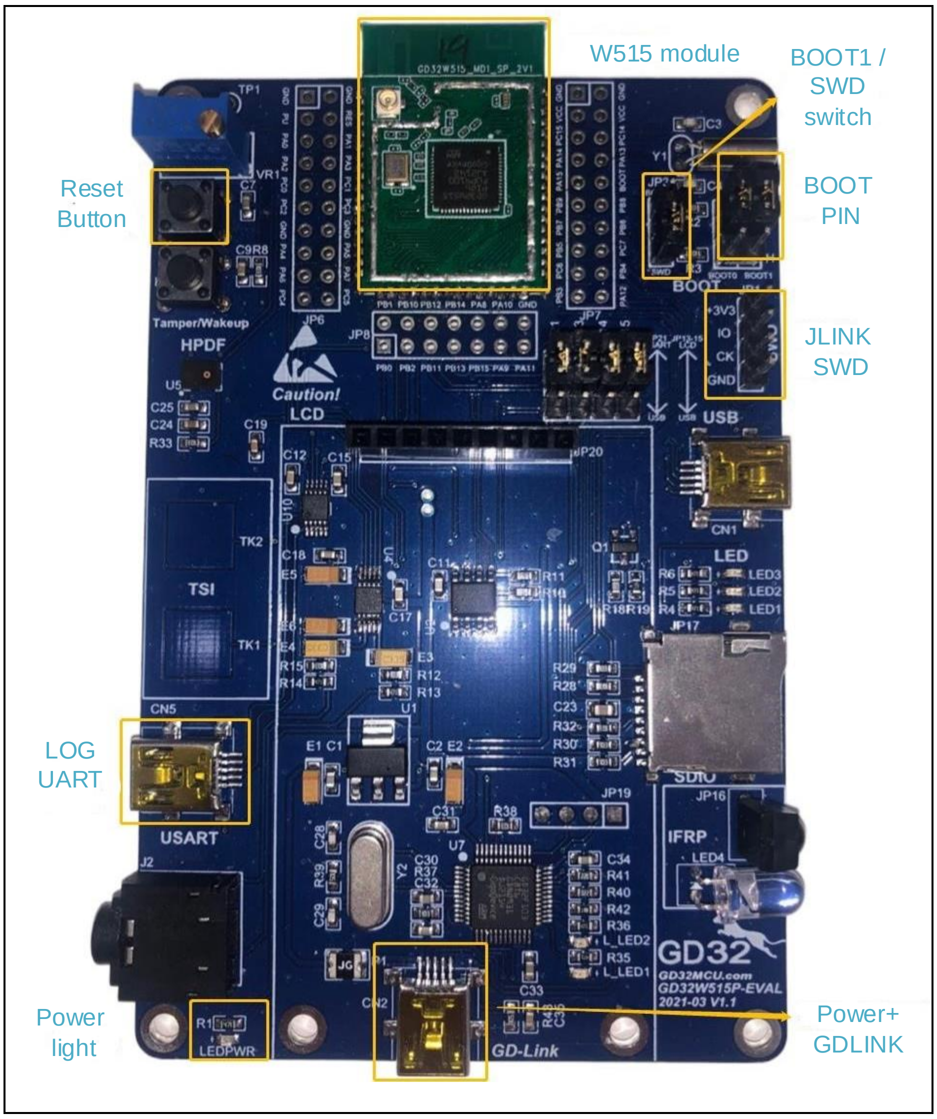
List of Tables

1. Introduction of development board

1.1. Picture of real development board

1.1.1. The EVAL development board

Figure 1-1. The picture of the EVAL development board



The EVAL development board is composed of a baseboard and a module. The module is equipped with GD32W51x WiFi chip and the baseboard provides many peripheral test ports, such as SDIO, I2S, IFRP, TSI and so on. In the figure above, the GD32W51x chip integrates FLASH through SIP (System In a Package). FLASH can also choose the way of external, module size unchanged.

For developers, the main focus may be on the following sections of the development board, which are indicated in the picture above.

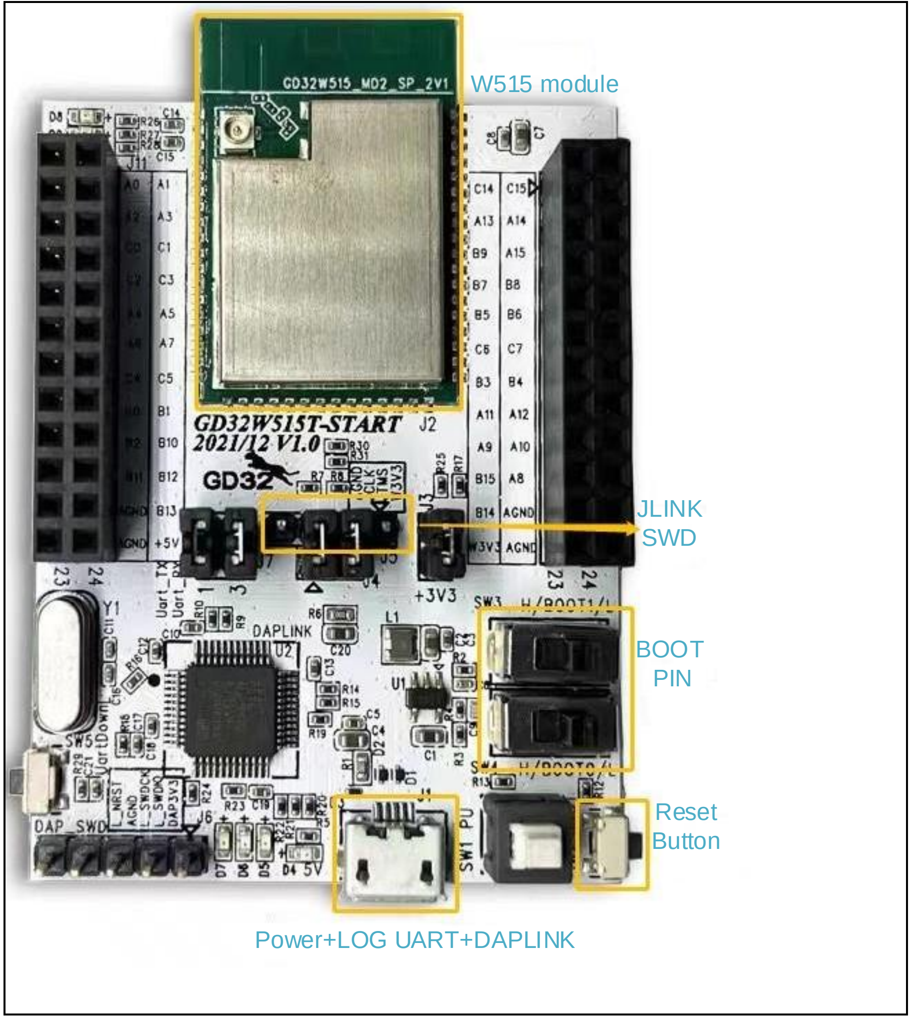
- Boot mode (Boot PIN)
- Power supply port (Power supply)
- Querying log (LOG UART)
- Debugger interface (GDLINK or JLINK)
- Reboot (Reset Button)

Note that the JLINK SWD CK PIN is multiplexed with the BOOT1 PIN. To switch the boot mode, switch the “BOOT1 / SWD Switch” jumper cap to the upper BOOT1 end. If Jlink debugging is required, switch the jumper cap to the SWD end below.

1.1.2. The START development board

The START development board is composed of a baseboard and a module equipped with GD32W51x WiFi chip. There are four types of GD32W51x WiFi chip, either internal SIP FLASH or external FLASH, and either 36 PIN package or 56 PIN package.

Figure 1-2. The picture of the START development board



For developers, the main focus may be on the following sections of the development board, which are indicated in the picture above.

- Boot mode (Boot PIN)
- Power supply port (Power supply)
- Querying log (LOG UART)
- Debugger interface (DAPLINK or JLINK)
- Reboot (Reset Button)

1.2. Boot mode

The GD32W51x chip can be booted from ROM, FLASH or SRAM

The level selection of the BOOT0 and BOOT1 pins of the development board determines the boot mode, As shown in ***Table 1-1. Boot mode***. For more information about the boot mode, see the “GD32W51x_User_Manual”.

Table 1-1. Boot mode

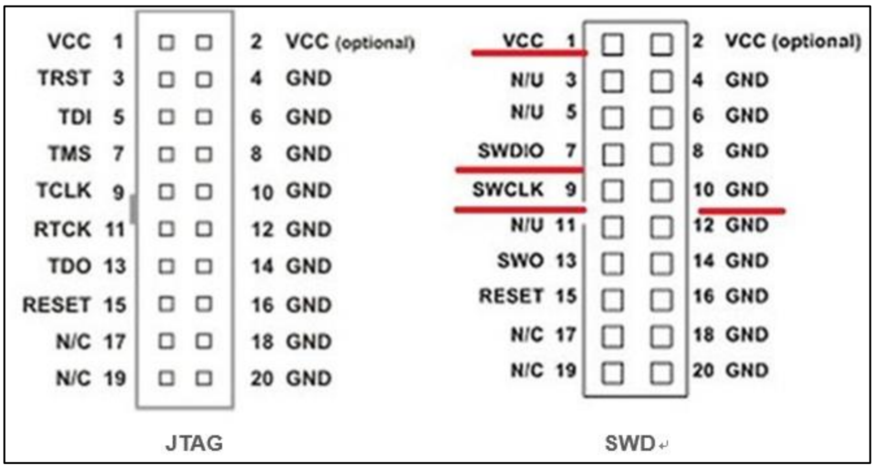
	BOOT1 pin is high	**BOOT1 pin is low**
BOOT0 pin is high	SRAM	ROM (Legacy Bootloader)
BOOT0 pin is low	FLASH	

1.3. Debugger interface

For EVAL development board, GD-LINK or JLINK can be used as the debugging interface. The development board comes with the GD-LINK debugger, which can be used by simply connecting the Mini USB power port. The JLINK SWD mode is supported, leading out four pins +3V3, IO, CK and GND respectively, which correspond to JLINK emulator pins VCC, SWDIO, SWCLK and GND, as shown in ***Figure 1-3. Pins of J-LINK***.

For START development board, DAPLINK or JLINK can be used as the debugging interface. The development board comes with the DAPLINK debugger (GD32F303), which can be used by simply connecting the power port Micro USB. The DAP chip is also integrated with UART, so only one USB cable is needed to complete power supply, debugging and log viewing. JLINK SWD leads out four pins, which are AGND、JCLK、JTWS and W3V3. Connect the four pins with GND, SWCLK, SWDIO and VCC (As shown in ***Figure 1-3. Pins of J-LINK***) corresponding to the J-LINK emulator by using Dupont wires, and then Burning and debugging code can be done via J-Link SWD mode. Connect pins JCLK and JTWS with the lower two pins respectively through jumper caps to download and debug code through DAPLINK. ***Figure 1-2. The picture of the START development board*** shows the debugging through DAPLINK.

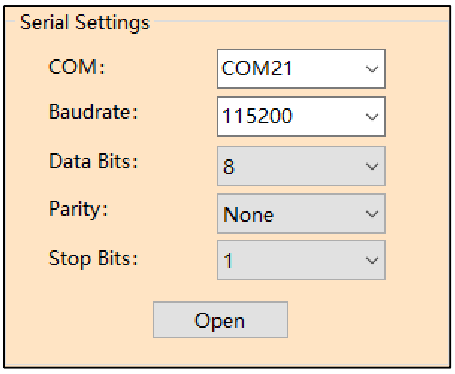
Figure 1-3. Pins of J-LINK



1.4. View log

Use the MiniUSB cable to connect the EVAL development board or the MicroUSB cable to connect the START development board. Then start the serial port tool on the PC and configure parameters according to the ***Figure 1-4. The serial port configuration*** and connect the development board. After the connection is successful, logs can be printed through the serial port.

Figure 1-4. The serial port configuration



Serial Settings

COM:	COM21
Baudrate:	115200
Data Bits:	8
Parity:	None
Stop Bits:	1

2. Build development environment

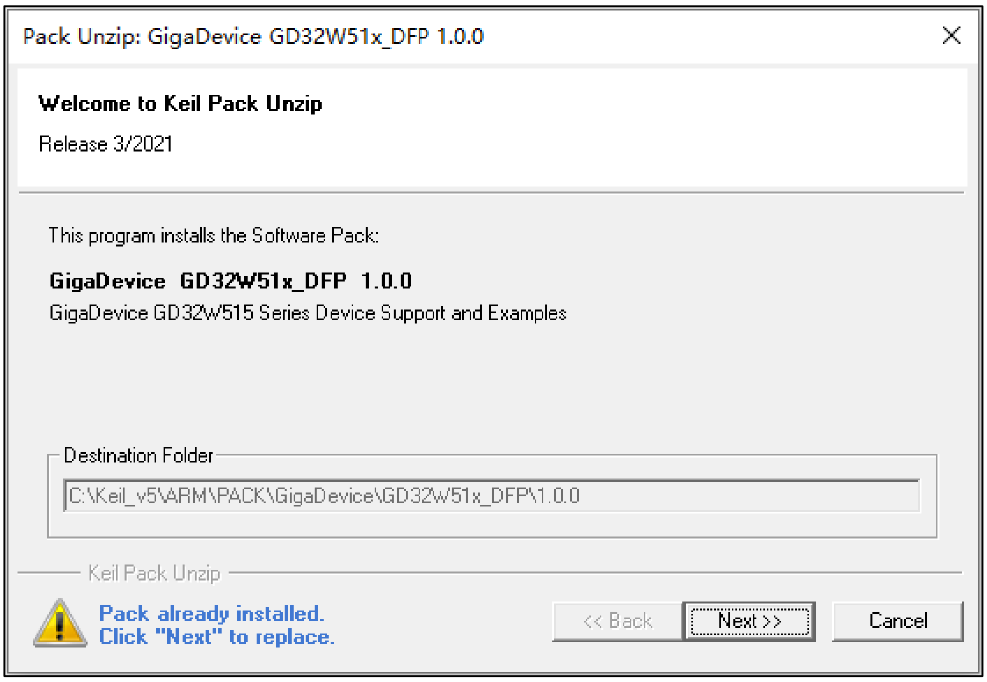
Before compiling and burning the firmware, the development environment needs to be set up.

At present, there are three commonly used development tools, namely KEIL, IAR and GCC. The SDK for GD32W51x already supports all three development tools. Each of the three tools has its own strengths and can be selected according to specific requirements.

2.1. Keil MDK5 installation

- Download
  - Official website: ***https://www2.keil.com/mdk5***, Please select MDK525 and later.
  - If forward compatibility with MDK4 is required, please download MDKCM5xx.EXE from the following website: ***https://www2.keil.com/mdk5/legacy***.
- Installation
  - The installation option can be selected as default, KEIL MDK5 which is installed under the "C: / Keil_v5" directory.
  - GD32W51x PACK
  - The PACK file is located in the GD32W51x_Addon / KEIL folder in the SDK, that is, GigaDevice.GD32W51x_DFP_1.x.x.pack.
  - Double-click the PACK file to run it, click the button "Next", as shown in ***Figure 2-1. Install the KEIL GD32W51x PACK***. After installation, the PACK file is installed under "C: / Keil_v5 / ARM / PACK / GigaDevice".

Figure 2-1. Install the KEIL GD32W51x PACK



Pack Unzip: GigaDevice GD32W51x_DFP 1.0.0

Welcome to Keil Pack Unzip

Release 3/2021

This program installs the Software Pack:

GigaDevice GD32W51x_DFP 1.0.0

GigaDevice GD32W515 Series Device Support and Examples

Destination Folder: C:\Keil_v5\ARM\PACK\GigaDevice\GD32W51x_DFP\1.0.0

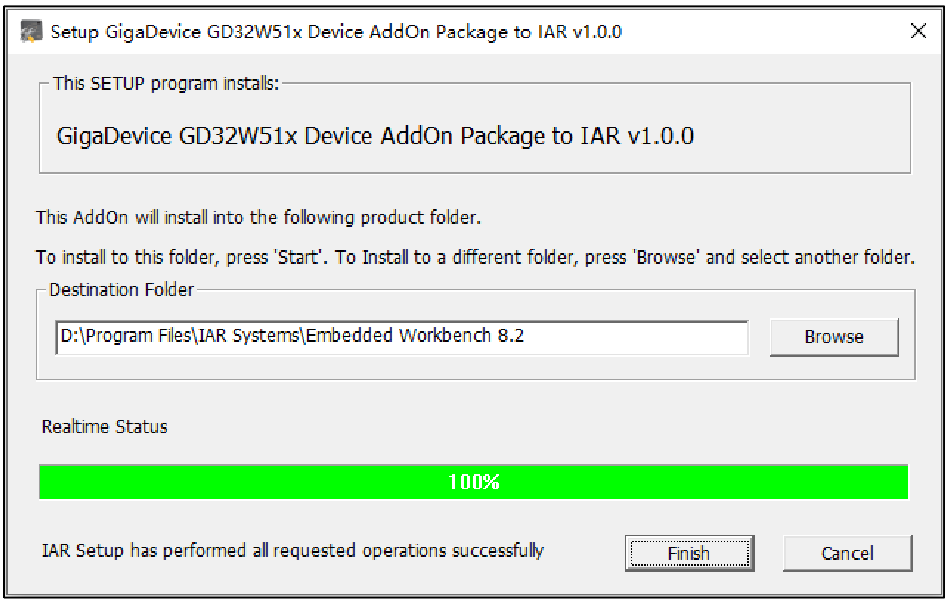
Keil Pack Unzip

Pack already installed.
Click "Next" to replace.

2.2. IAR Installation

- Download
  - Official website: https://www.iar.com/products/architectures/arm.
  - Select version 8.32 or later to better support ARM CM33.
- Installation
  - Default installation is optional. IAR will be installed under “C: / Program Files (x86) / IAR Systems / Embedded Workbench 8.2”.
- GD32W51x Addon
  - Addon is located in the GD32W51x_Addon / IAR folder in the SDK, that is, IAR_GD32W51x_ADDON_1.x.x.exe.
  - Right-click "Run as Administrator", select the path to the IAR installation path, and click "start" to complete the installation, as shown in ***Figure 2-2. Install the IAR GD32W51x Addon***.

Figure 2-2. Install the IAR GD32W51x Addon



Setup GigaDevice GD32W51x Device AddOn Package to IAR v1.0.0

This SETUP program installs:

GigaDevice GD32W51x Device AddOn Package to IAR v1.0.0

This AddOn will install into the following product folder.

To install to this folder, press 'Start'. To Install to a different folder, press 'Browse' and select another folder.

Destination Folder

D:\Program Files\IAR Systems\Embedded Workbench 8.2

Realtime Status

100%

IAR Setup has performed all requested operations successfully

2.3. CMAKE+GCC installation

For the GCC environment, CMAKE and MAKE are selected, and use GCC Toolchain to compile.

- CMAKE installation
  - Download website: https://cmake.org/download/.
  - Select version 3.15 or later. cmake-3.20.3-windows-x86_64 is recommended.
- MAKE installation
  - Download website: http://ftp.gnu.org/gnu/make/.
  - make-3.81 is recommended.

- Toolchain installation
  - Download website: ***https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads***
  - Select gcc-arm-none-eabi-7-2018-q2-update-win32 or later, gcc-arm-none-eabi-9-2020-q2-update-win32 is recommended.
- Add to the system path
  - After the installation is successful by using the default path, add the following path to the path of the system environment variables. If the installation path is changed, add the corresponding installation path to the system path.

    a) C :/ Program Files (x86) / GnuWin32 / bin

    b) C :/ Program Files (x86) / GNU Tools Arm Embedded / 9 2020-q2-update / bin

    c) C :/ Program Files / CMake / bin

If JLINK is used as the debugger, the JLINK driver needs to be installed, refer to ***JLINK driver installation***. If GD-LINK or DAPLINK is used, OpenOCD is used for downloading and debugging. OpenOCD does not need to be installed. The execution file is located in GD32W51x_Addon / openOCD / bin / openocd.exe.

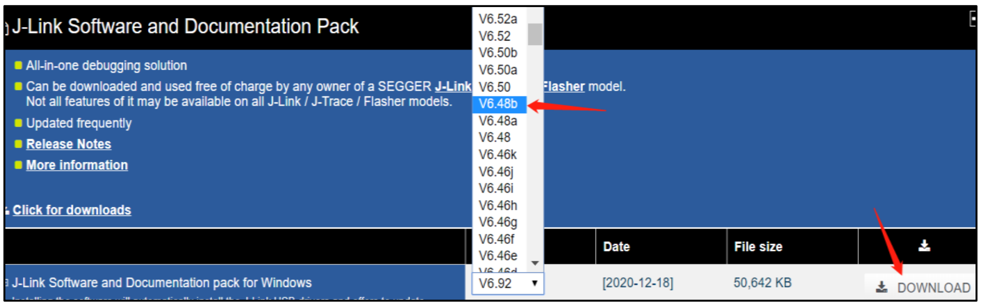
2.3.1. JLINK driver installation

Developers can choose to use JLINK for development debugging, if not, skip this section

If the JLINK driver is not installed or its version is earlier, perform the following steps to download and install the JLINK driver.

- Download JLINK driver
  - Enter the official website: ***https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack***.
  - Download JLink_Windows_V648b driver, as shown in ***Figure 2-3. JLINK driver download***, Select Windows V6.48b and click "DOWNLOAD". On the page that pops up, select accept the protocol and download it directly.

Figure 2-3. JLINK driver download


J-Link Software and Documentation Pack

- All-in-one debugging solution
- Can be downloaded and used free of charge by any owner of a SEGGER J-Link ... Flasher model. Not all features of it may be available on all J-Link / J-Trace / Flasher models.
- Updated frequently
- Release Notes
- More information

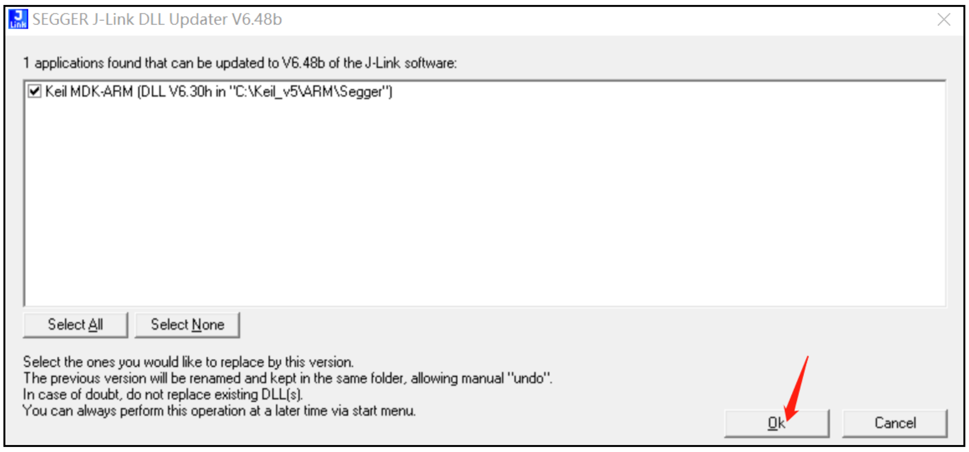
Click for downloads

		Date	File size	
J-Link Software and Documentation pack for Windows	V6.92	[2020-12-18]	50,642 KB	DOWNLOAD

- Install J-LINK driver
  - Double-click JLink_Windows_V648b.exe to install the Jlink driver. Select the "Ok" option only on the screen shown in ***Figure 2-4. JLINK driver installation*** and the default option for the rest. Make sure JLink.exe is installed in C :/ Program Files

(x86) / SEGGER / JLink / JLink.exe.

Figure 2-4. JLINK driver installation


SEGGER J-Link DLL Updater V6.48b

1 applications found that can be updated to V6.48b of the J-Link software:

☑ Keil MDK-ARM (DLL V6.30h in "C:\Keil_v5\ARM\Segger")

Select the ones you would like to replace by this version.
The previous version will be renamed and kept in the same folder, allowing manual "undo".
In case of doubt, do not replace existing DLL(s).
You can always perform this operation at a later time via start menu.

- JLINK flash burning configuration
  - To use JLINK to burn flash in the GCC development environment, go to the GD32W51x_RELEASE / scripts / directory and run the “setup_jlink.bat” file by using rights of administrators. The FLASH burning algorithm file and JLinkDevices.xml will be replaced in the JLink installation directory “C :/ Program Files (x86) / SEGGER / JLink /”.

3. Development instructions

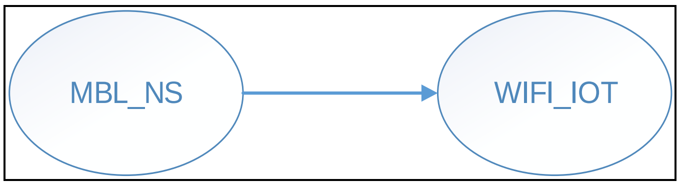
Before the development, first understand the members of the SDK execution program group, how to correctly configure the SDK, and the final firmware generation location and composition.

SDK compilation and debugging can be done according to different development tools, such as KEIL, IAR or GCC, please select the corresponding chapter to understand.

3.1. SDK execution program group

The SDK finally generates two executables: MBL_NS (Main Bootloader) and WIFI_IOT (Some configuration files are called NSPE to be consistent with the name in the Trust Zone enabled mode). They all end up being burned to FLASH. After powering on, the program will start from the Reset_Handler of MBL_NS and then jump to the WIFI_IOT main program to run, as shown in ***Figure 3-1. Boot process***.

Figure 3-1. Boot process

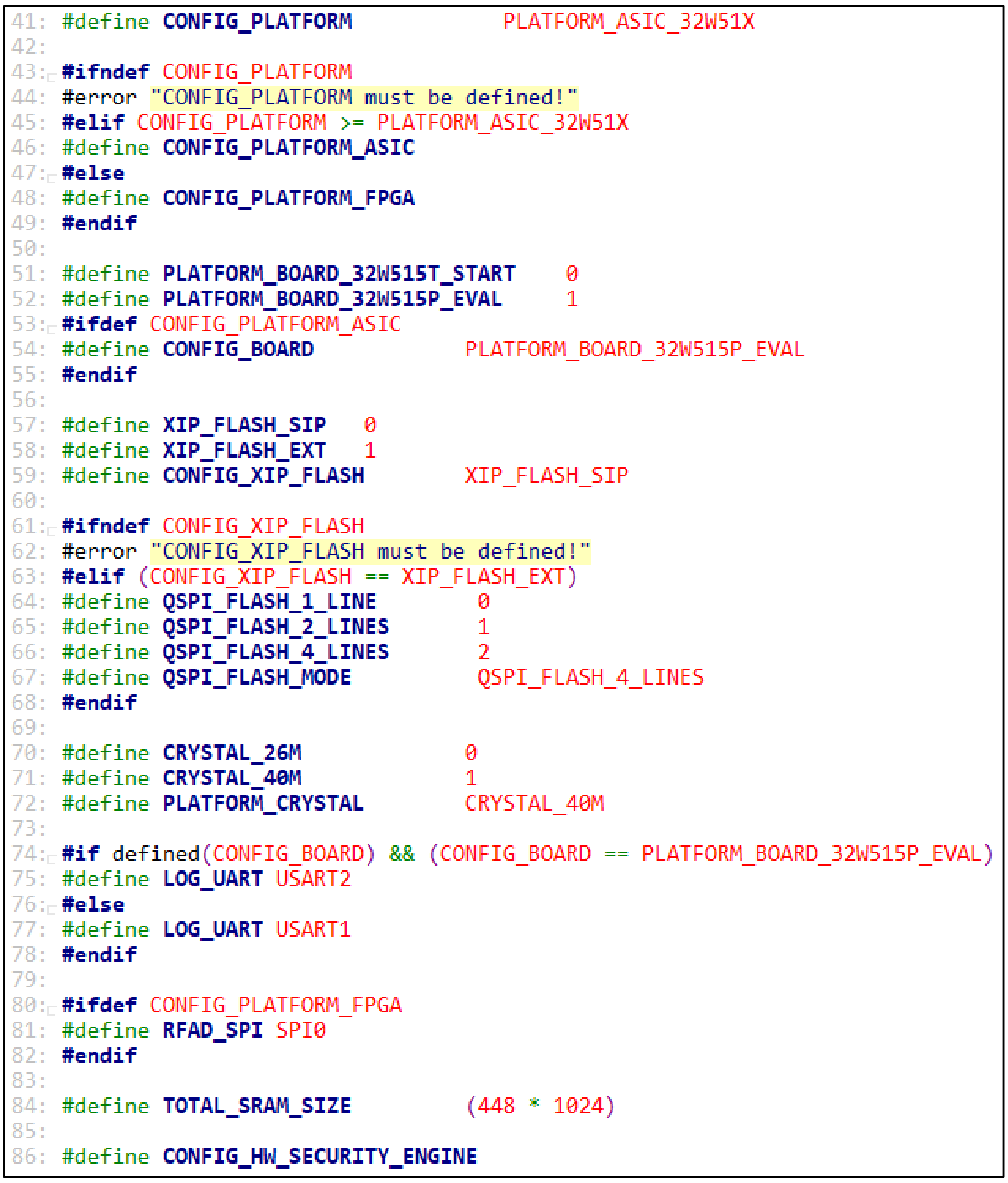


3.2. SDK configuration

3.2.1. Platform configuration

The configuration file is GD32W51x_RELEASE / config / platform_def.h, the main content is shown in ***Figure 3-2. Platform configuration***.

Figure 3-2. Platform configuration



```
#define CONFIG_PLATFORM             PLATFORM_ASIC_32W51X

#ifndef CONFIG_PLATFORM
#error "CONFIG_PLATFORM must be defined!"
#elif CONFIG_PLATFORM >= PLATFORM_ASIC_32W51X
#define CONFIG_PLATFORM_ASIC
#else
#define CONFIG_PLATFORM_FPGA
#endif

#define PLATFORM_BOARD_32W515T_START    0
#define PLATFORM_BOARD_32W515P_EVAL     1
#ifdef CONFIG_PLATFORM_ASIC
#define CONFIG_BOARD            PLATFORM_BOARD_32W515P_EVAL
#endif

#define XIP_FLASH_SIP   0
#define XIP_FLASH_EXT   1
#define CONFIG_XIP_FLASH        XIP_FLASH_SIP

#ifndef CONFIG_XIP_FLASH
#error "CONFIG_XIP_FLASH must be defined!"
#elif (CONFIG_XIP_FLASH == XIP_FLASH_EXT)
#define QSPI_FLASH_1_LINE        0
#define QSPI_FLASH_2_LINES       1
#define QSPI_FLASH_4_LINES       2
#define QSPI_FLASH_MODE          QSPI_FLASH_4_LINES
#endif

#define CRYSTAL_26M             0
#define CRYSTAL_40M             1
#define PLATFORM_CRYSTAL        CRYSTAL_40M

#if defined(CONFIG_BOARD) && (CONFIG_BOARD == PLATFORM_BOARD_32W515P_EVAL)
#define LOG_UART USART2
#else
#define LOG_UART USART1
#endif

#ifdef CONFIG_PLATFORM_FPGA
#define RFAD_SPI SPI0
#endif

#define TOTAL_SRAM_SIZE         (448 * 1024)

#define CONFIG_HW_SECURITY_ENGINE
```

- For the EVAL development board, please select:
  - #define CONFIG_BOARD   PLATFORM_BOARD_32W515P_EVAL
- For the START development board, please select:
  - #define CONFIG_BOARD   PLATFORM_BOARD_32W515T_START

- For the built-in SIP FLASH, please select:
  - #define CONFIG_XIP_FLASH XIP_FLASH_SIP
- For the external EXT FLASH, please select:
  - #define CONFIG_XIP_FLASH XIP_FLASH_EXT
- For crystal is 40 MHz, please select:
  - #define PLATFORM_CRYSTAL CRYSTAL_40M
- For crystal is 26 MHz, please select:
  - #define PLATFORM_CRYSTAL CRYSTAL_26M

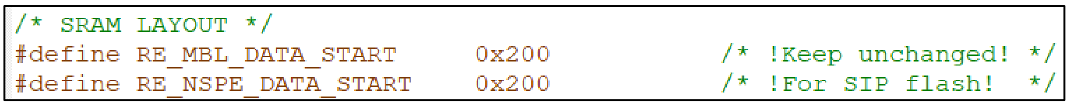
3.2.2. SRAM layout

The configuration file is GD32W51x_RELEASE / config / config_gdm32_ntz.h. Modify the following macro definition values to plan the SRAM space occupied by the executable program segments MBL and NSPE. These values are offset addresses, and the base address is defined at the beginning of the file.

NSPE refers to the WIFI_IOT execution program segment, and Non-Secure Process Environment (NSPE) is the name of the WIFI_IOT execution program segment after the Trust Zone is enabled. As opposed to Secure Process Environment (PROT as seen in the configuration file), WIFI_IOT runs in a non-secure environment.

The line marked "! Keep unchanged!" cannot be modified, otherwise it will affect the running of the code MbedTLS in the ROM.

Figure 3-3. SRAM layout



```
/* SRAM LAYOUT */
#define RE_MBL_DATA_START     0x200          /* !Keep unchanged! */
#define RE_NSPE_DATA_START    0x200          /* !For SIP flash!  */
```

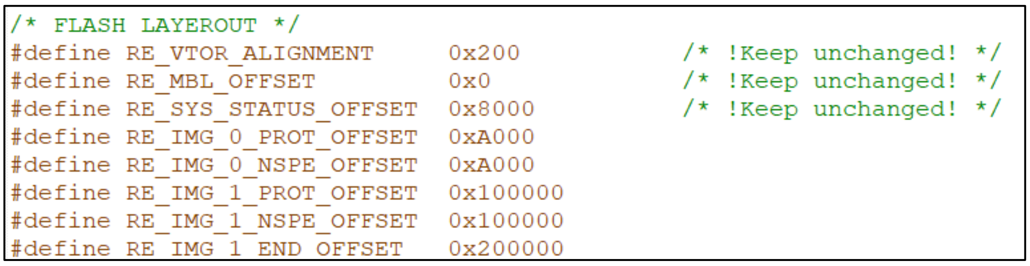
The SRAM space plan inside each executable segment can be viewed in Scatter File on the Linker page of the Project option. The macro definitions in the file can be found in “xxx_region.h”, for example, “nspe_region.h”.

3.2.3. FLASH layout

The configuration file is GD32W51x_RELEASE / config / config_gdm32_ntz.h. Modify the following macro definition values to plan the FLASH space occupied by the executable program segments MBL and NSPE.These values are offset addresses, and the base address is defined at the beginning of the file. NSPE is explained in ***SRAM layout***.

The line marked "! Keep unchanged!" cannot be modified, otherwise it will affect the running of the project.

Figure 3-4. FLASH layout


```
/* FLASH LAYEROUT */
#define RE_VTOR_ALIGNMENT      0x200          /* !Keep unchanged! */
#define RE_MBL_OFFSET          0x0            /* !Keep unchanged! */
#define RE_SYS_STATUS_OFFSET   0x8000         /* !Keep unchanged! */
#define RE_IMG_0_PROT_OFFSET   0xA000
#define RE_IMG_0_NSPE_OFFSET   0xA000
#define RE_IMG_1_PROT_OFFSET   0x100000
#define RE_IMG_1_NSPE_OFFSET   0x100000
#define RE_IMG_1_END_OFFSET    0x200000
```

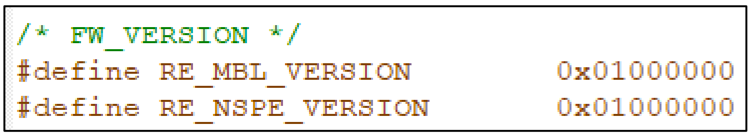
The FLASH space plan inside each executable segment can be viewed in Scatter File on the Linker page of the Project option. The macro definitions in the file can be found in “xxx_region.h”, for example, “nspe_region.h”.

3.2.4. Firmware version number

The configuration file is GD32W51x_RELEASE / config / config_gdm32_ntz.h. Modify the following macro definition values to specify the version number. However, the only version number that will affect future user upgrades is RE_NSPE_VERSION .

The MBL can only be upgraded locally, and the NSPE can be upgraded online. The SDK version is the same as RE_NSPE_VERSION, and the version number is displayed in the startup log through UART, for example, “SDK version: V1.0.0”.

Figure 3-5 Firmware version


```
/* FW_VERSION */
#define RE_MBL_VERSION          0x01000000
#define RE_NSPE_VERSION         0x01000000
```

3.2.5. APP configuration

The configuration file is “GD32W51x_RELEASE / NSPE/WIFI_IOT / app / app_cfg.h”. Some applications can be opened or not, for example: ATCMD, SSL example, Aliyun, FATFS and so on.

3.3. Firmware generation and download

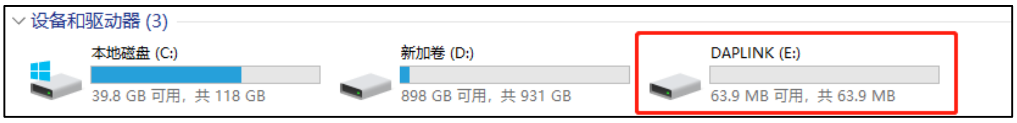
The BIN and HEX files automatically generated by the compiled script are placed in the “GD32W51x_RELEASE / scripts / images /” directory. Image-*.bin is used for production or upgrade.

The “image-all.bin” file contains executable program segments MBL_NS and WIFI_IOT, which can be used for production and burned to blank FLASH.

The “image-ota.bin” file contains only WIFI_IOT (NSPE), which can be used to upgrade.

For download, in addition to the IDE tool download, for the START development board, can also be downloaded by the copy and paste method. When the development board is plugged into the computer via USB cable, the DAPLINK disk as shown in ***Figure 3-6. List of devices and drivers***. Directly copy the" image-all.bin" file into the DAPLINK disk to complete the FLASH burning of GD32W51x chip. HEX file downloads are also supported.

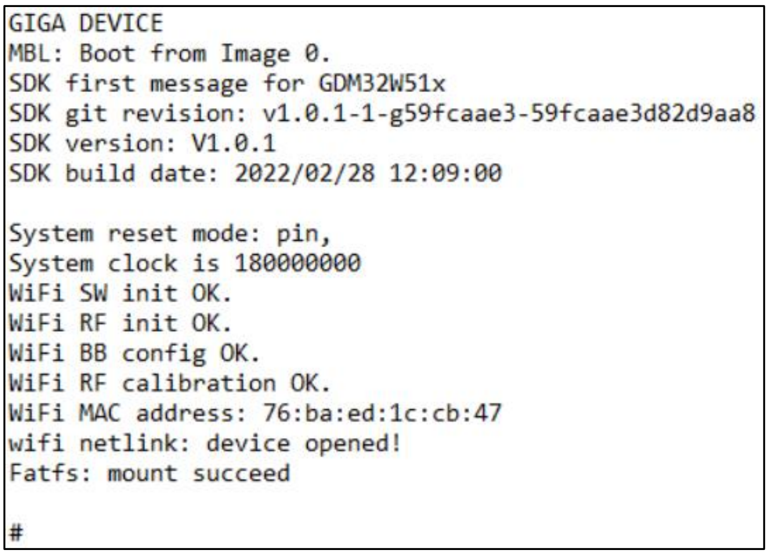
Figure 3-6. List of devices and drivers



3.4. Example of correct log

After the firmware group (MBL_NS+WIFI_IOT) is downloaded successfully, open the serial port tool and press the "Reset" button on the development board. The startup information is shown in ***Figure 3-7. Project boot Information***. If an exception occurs, refer to ***Q&A*** to try to resolve it.

Figure 3-7. Project boot Information


```
GIGA DEVICE
MBL: Boot from Image 0.
SDK first message for GDM32W51x
SDK git revision: v1.0.1-1-g59fcaae3-59fcaae3d82d9aa8
SDK version: V1.0.1
SDK build date: 2022/02/28 12:09:00

System reset mode: pin,
System clock is 180000000
WiFi SW init OK.
WiFi RF init OK.
WiFi BB config OK.
WiFi RF calibration OK.
WiFi MAC address: 76:ba:ed:1c:cb:47
wifi netlink: device opened!
Fatfs: mount succeed

#
```

4. KEIL project

This chapter describes how to compile and debug the SDK in KEIL.

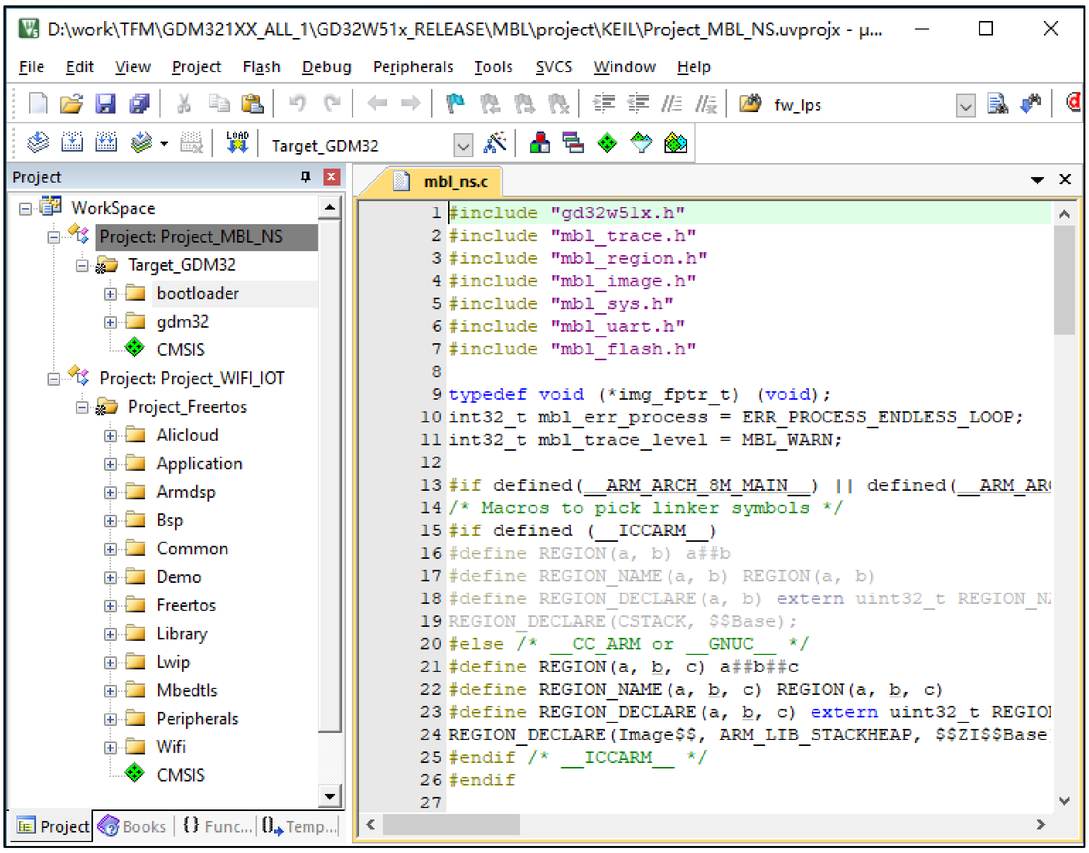
The project group consists of two projects: MBL_NS and WIFI_IOT. WIFI_IOT contains the WiFi protocol stack, peripheral drivers, applications, etc. MBL_NS is mainly responsible for selecting the correct firmware to run from two WIFI_IOT firmware (one for the current firmware and one for the upgraded firmware).

4.1. Project group

To enable KEIL to identify GD32W51x, install “GD32W51x_DFP_1.x.x.pack” in the “gigadevice.GD32w51x_addon /KEIL” directory before opening the project.

Find “MultiProject_NS.uvmpw” from the root directory of GD32W51x_RELEASE and double-click KEIL to open the workspace, as shown in ***Figure 4-1. KEIL project Group interface***.

Figure 4-1. KEIL project Group interface



```
#include "gd32w51x.h"
#include "mbl_trace.h"
#include "mbl_region.h"
#include "mbl_image.h"
#include "mbl_sys.h"
#include "mbl_uart.h"
#include "mbl_flash.h"

typedef void (*img_fptr_t) (void);
int32_t mbl_err_process = ERR_PROCESS_ENDLESS_LOOP;
int32_t mbl_trace_level = MBL_WARN;

#if defined(__ARM_ARCH_8M_MAIN__) || defined(__ARM_AR
/* Macros to pick linker symbols */
#if defined (__ICCARM__)
#define REGION(a, b) a##b
#define REGION_NAME(a, b) REGION(a, b)
#define REGION_DECLARE(a, b) extern uint32_t REGION_N
REGION_DECLARE(CSTACK, $$Base);
#else /* __CC_ARM or __GNUC__ */
#define REGION(a, b, c) a##b##c
#define REGION_NAME(a, b, c) REGION(a, b, c)
#define REGION_DECLARE(a, b, c) extern uint32_t REGIO
REGION_DECLARE(Image$$, ARM_LIB_STACKHEAP, $$ZI$$Base
#endif /* __ICCARM__ */
#endif

```

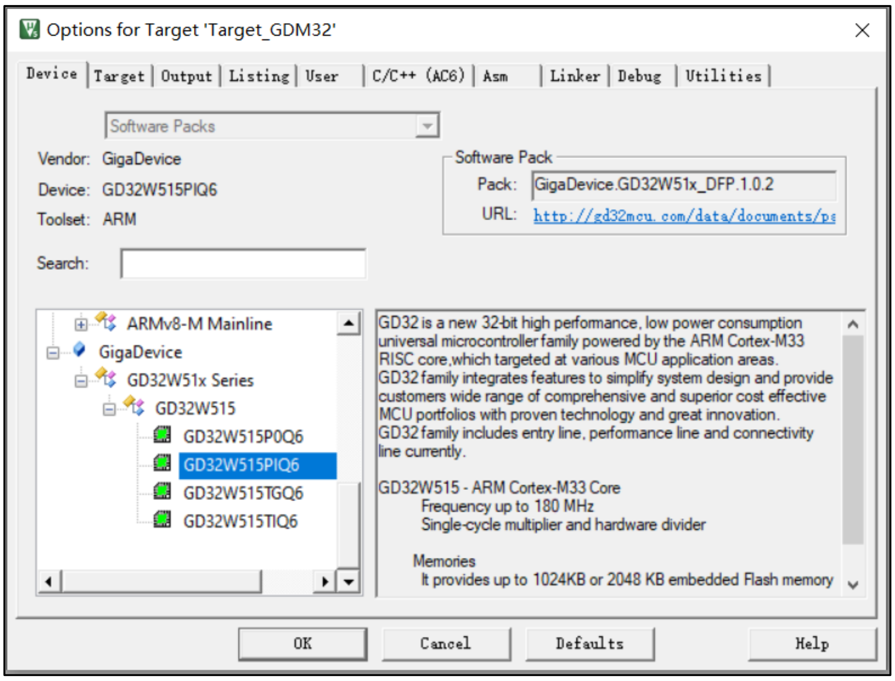
4.2. Project configuration

4.2.1. Device selection

The GD32W515 chip is available in a variety of types and can be seen on the KEIL Options->Device page once the PACK is installed. As shown in ***Figure 4-2. KEIL "Device" options page***. The default type is GD32W515PIQ6, 2M SIP FLASH, and 448K SRAM.

If the chip is of another type, select it based on the actual type.

Figure 4-2. KEIL "Device" options page



Vendor: GigaDevice
Device: GD32W515PIQ6
Toolset: ARM
Search:

Software Pack
Pack: GigaDevice.GD32W51x_DFP.1.0.2
URL: http://gd32mcu.com/data/documents/ps

ARMv8-M Mainline
GigaDevice
GD32W51x Series
GD32W515
GD32W515P0Q6
GD32W515PIQ6
GD32W515TGQ6
GD32W515TIQ6

GD32 is a new 32-bit high performance, low power consumption universal microcontroller family powered by the ARM Cortex-M33 RISC core,which targeted at various MCU application areas.
GD32 family integrates features to simplify system design and provide customers wide range of comprehensive and superior cost effective MCU portfolios with proven technology and great innovation.
GD32 family includes entry line, performance line and connectivity line currently.

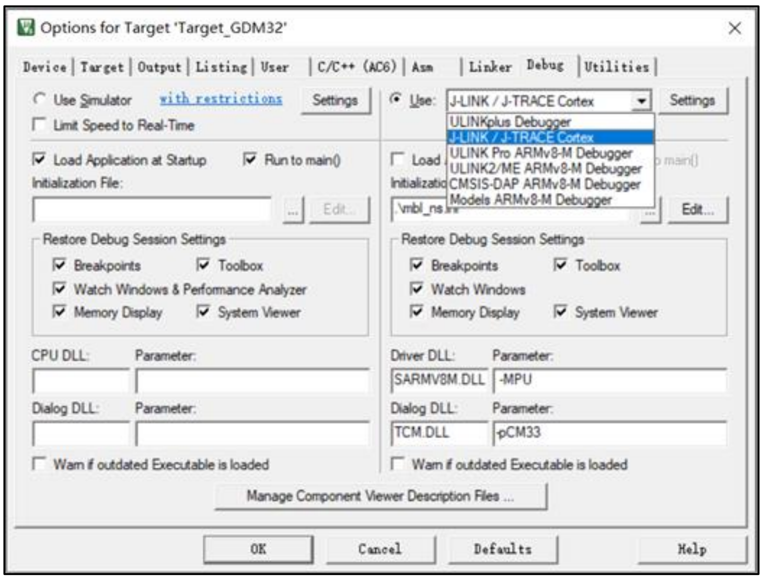
GD32W515 - ARM Cortex-M33 Core
Frequency up to 180 MHz
Single-cycle multiplier and hardware divider

Memories
It provides up to 1024KB or 2048 KB embedded Flash memory

4.2.2. Debugger configuration

For the EVAL development board, select on-board GD-LINK and for the START development board, select on-board DAPLINK. The Debugger on the Debug page is "CMSIS-DAP ARMv8-M Debugger". For JLINK, select "J-LINK/J-TRACE Cortex", as shown in ***Figure 4-3. KEIL "Debug" options page***.

Figure 4-3. KEIL “Debug” options page


Options for Target 'Target_GDM32'

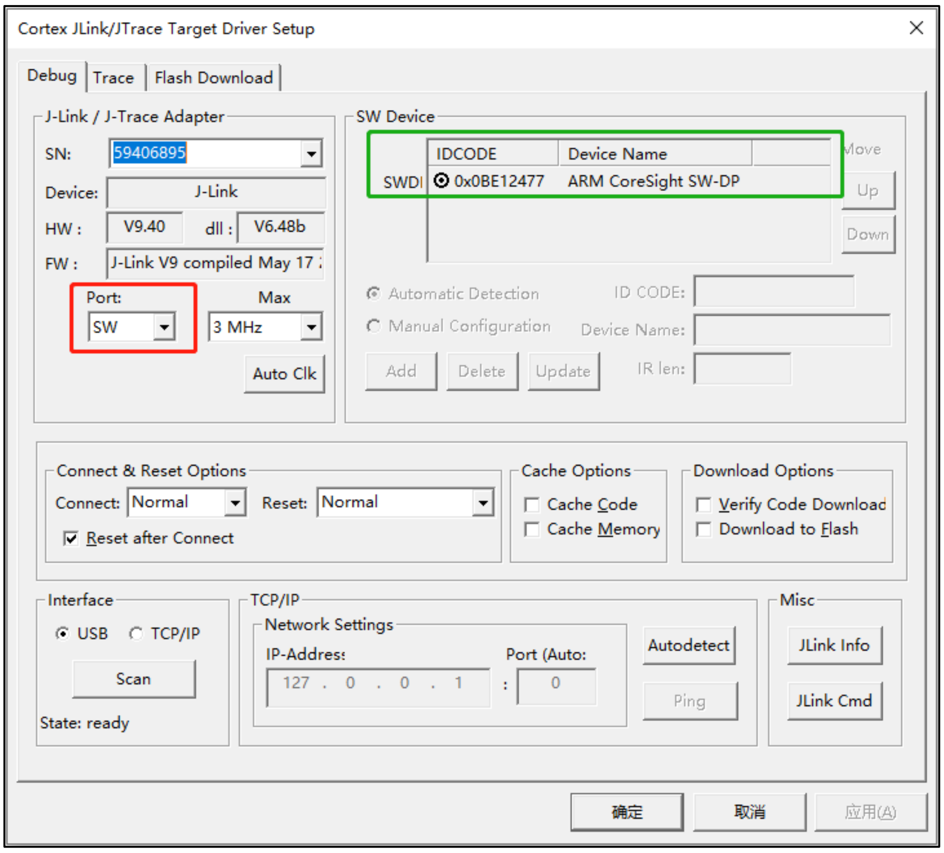
Device | Target | Output | Listing | User | C/C++ (AC6) | Asm | Linker | Debug | Utilities

- ○ Use Simulator with restrictions Settings
- ☐ Limit Speed to Real-Time
- ☑ Load Application at Startup ☑ Run to main()
- Initialization File:
- Restore Debug Session Settings: ☑ Breakpoints ☑ Toolbox ☑ Watch Windows & Performance Analyzer ☑ Memory Display ☑ System Viewer
- CPU DLL: Parameter:
- Dialog DLL: Parameter:
- ☐ Warn if outdated Executable is loaded

- ● Use: J-LINK / J-TRACE Cortex Settings
  - ULINKplus Debugger
  - J-LINK / J-TRACE Cortex
  - ULINK Pro ARMv8-M Debugger
  - ULINK2/ME ARMv8-M Debugger
  - CMSIS-DAP ARMv8-M Debugger
  - Models ARMv8-M Debugger
- ☐ Load main()
- Initializatio: .\mbl_ns.ini Edit...
- Restore Debug Session Settings: ☑ Breakpoints ☑ Toolbox ☑ Watch Windows ☑ Memory Display ☑ System Viewer
- Driver DLL: SARMV8M.DLL Parameter: -MPU
- Dialog DLL: TCM.DLL Parameter: -pCM33
- ☐ Warn if outdated Executable is loaded

In the KEIL project, relevant Debugger settings have been configured by default. Note that the JLINK port must be selected as SW. Use JLINK Debugger, connect GD32W515 chip, the page as shown in ***Figure 4-4. KEIL “JLINK Settings Debug” options page*** will appear.

Figure 4-4. KEIL “JLINK Settings Debug” options page


Cortex JLink/JTrace Target Driver Setup

Debug | Trace | Flash Download

J-Link / J-Trace Adapter
- SN: 59406895
- Device: J-Link
- HW: V9.40 dll: V6.48b
- FW: J-Link V9 compiled May 17
- Port: SW Max: 3 MHz

SW Device

	IDCODE	Device Name
SWDI	0x0BE12477	ARM CoreSight SW-DP

- ● Automatic Detection ID CODE:
- ○ Manual Configuration Device Name:
- IR len:

Connect & Reset Options
- Connect: Normal Reset: Normal
- ☑ Reset after Connect

Cache Options: ☐ Cache Code ☐ Cache Memory

Download Options: ☐ Verify Code Download ☐ Download to Flash

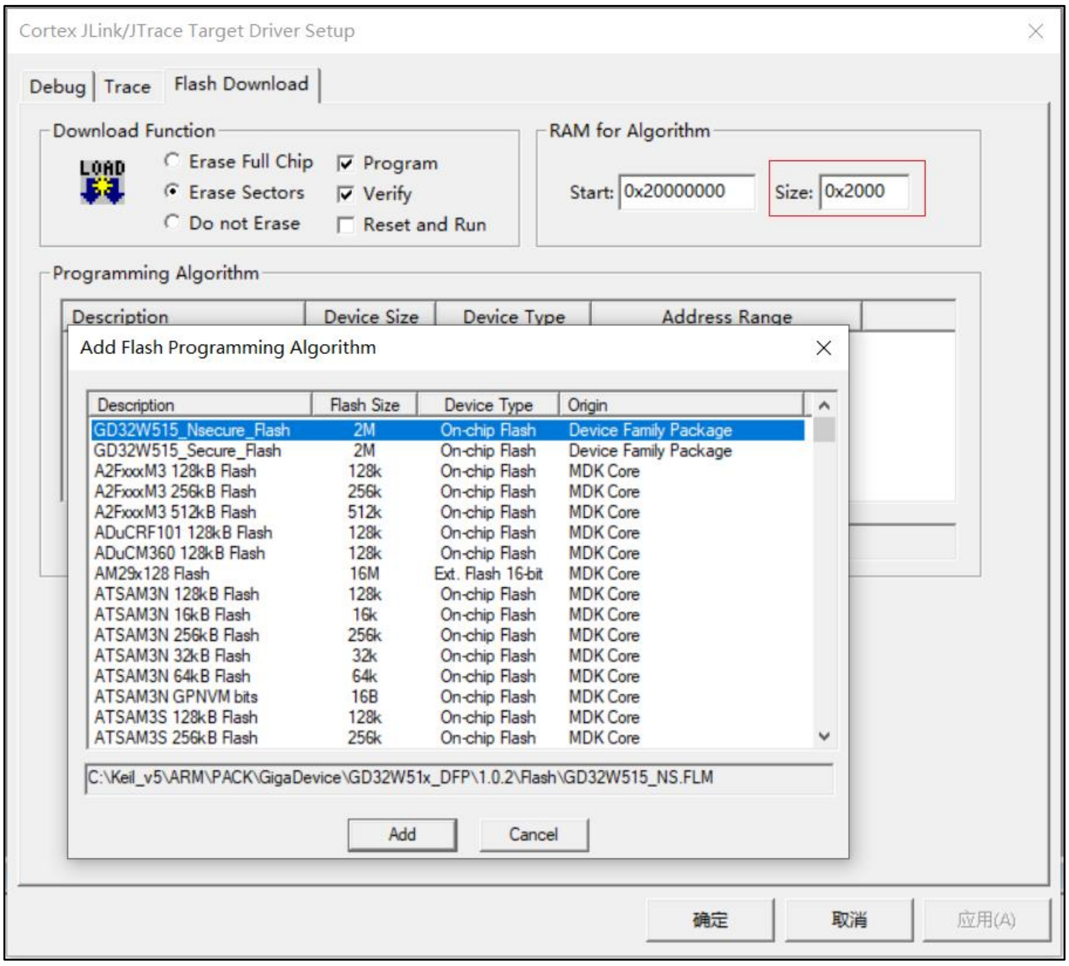
Interface: ● USB ○ TCP/IP State: ready

TCP/IP — Network Settings: IP-Address: 127 . 0 . 0 . 1 Port (Auto: 0

If the Device is changed, the Debugger settings need to be reconfigured. After selecting the correct Debugger, the FLASH download algorithm needs to be selected, as shown in ***Figure***

4-5. KEIL "Flash Download" options page. Note that the "Size" in "RAM for Algorithm" needs to be changed to 0x2000.

Figure 4-5. KEIL "Flash Download" options page



Cortex JLink/JTrace Target Driver Setup

Debug | Trace | Flash Download

Download Function
- Erase Full Chip
- [✓ Erase Sectors]
- Do not Erase
- [✓] Program
- [✓] Verify
- [ ] Reset and Run

RAM for Algorithm
Start: 0x20000000 Size: 0x2000

Programming Algorithm

Description	Device Size	Device Type	Address Range

Add Flash Programming Algorithm

Description	Flash Size	Device Type	Origin
GD32W515_Nsecure_Flash	2M	On-chip Flash	Device Family Package
GD32W515_Secure_Flash	2M	On-chip Flash	Device Family Package
A2FxxxM3 128kB Flash	128k	On-chip Flash	MDK Core
A2FxxxM3 256kB Flash	256k	On-chip Flash	MDK Core
A2FxxxM3 512kB Flash	512k	On-chip Flash	MDK Core
ADuCRF101 128kB Flash	128k	On-chip Flash	MDK Core
ADuCM360 128kB Flash	128k	On-chip Flash	MDK Core
AM29x128 Flash	16M	Ext. Flash 16-bit	MDK Core
ATSAM3N 128kB Flash	128k	On-chip Flash	MDK Core
ATSAM3N 16kB Flash	16k	On-chip Flash	MDK Core
ATSAM3N 256kB Flash	256k	On-chip Flash	MDK Core
ATSAM3N 32kB Flash	32k	On-chip Flash	MDK Core
ATSAM3N 64kB Flash	64k	On-chip Flash	MDK Core
ATSAM3N GPNVM bits	16B	On-chip Flash	MDK Core
ATSAM3S 128kB Flash	128k	On-chip Flash	MDK Core
ATSAM3S 256kB Flash	256k	On-chip Flash	MDK Core

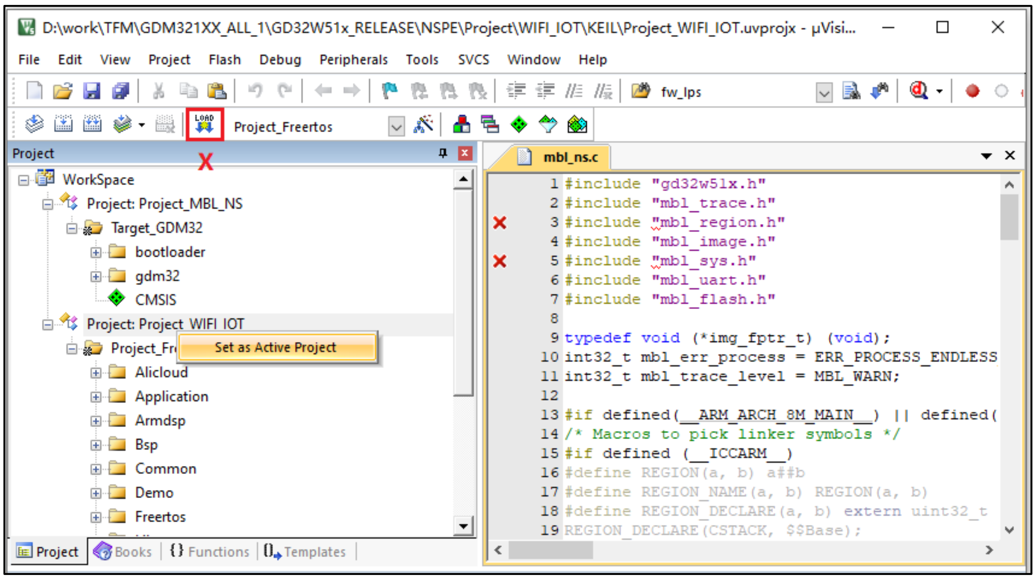
C:\Keil_v5\ARM\PACK\GigaDevice\GD32W51x_DFP\1.0.2\Flash\GD32W515_NS.FLM

4.3. Compilation

- Compilation

  If a project needs to be compiled, verify that the project is currently active project. If not, select the project to be compiled, right-click it, and click "Set as Active Project", as shown in *Figure 4-6. Current project switch*. Then click the icon to compile the project.

Figure 4-6. Current project switch

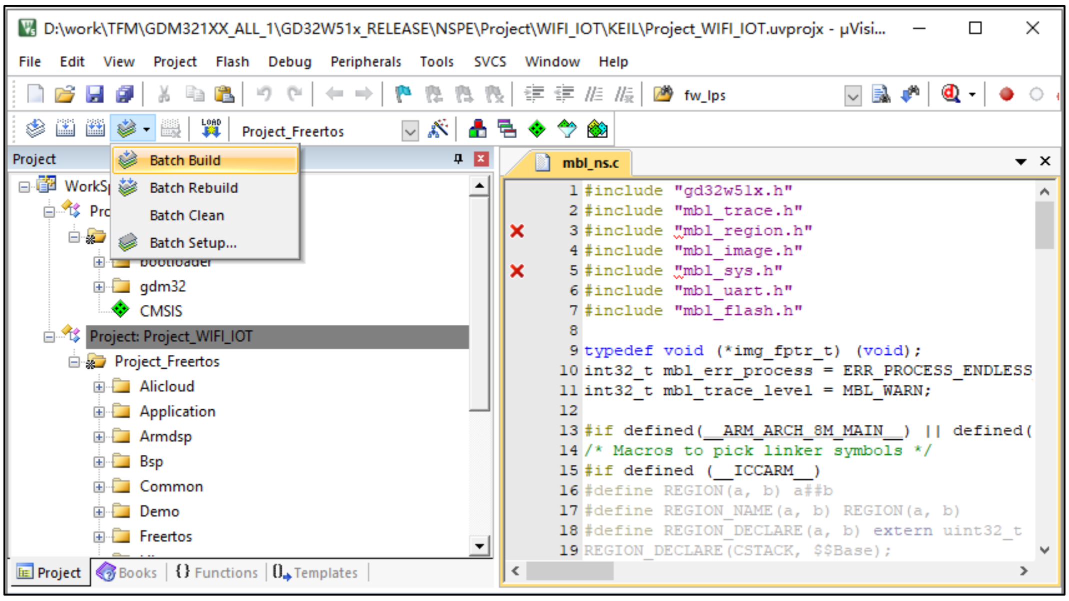


```
#include "gd32w51x.h"
#include "mbl_trace.h"
#include "mbl_region.h"
#include "mbl_image.h"
#include "mbl_sys.h"
#include "mbl_uart.h"
#include "mbl_flash.h"

typedef void (*img_fptr_t) (void);
int32_t mbl_err_process = ERR_PROCESS_ENDLESS
int32_t mbl_trace_level = MBL_WARN;

#if defined(__ARM_ARCH_8M_MAIN__) || defined(
/* Macros to pick linker symbols */
#if defined (__ICCARM__)
#define REGION(a, b) a##b
#define REGION_NAME(a, b) REGION(a, b)
#define REGION_DECLARE(a, b) extern uint32_t
REGION_DECLARE(CSTACK, $$Base);
```

- Batch compilation

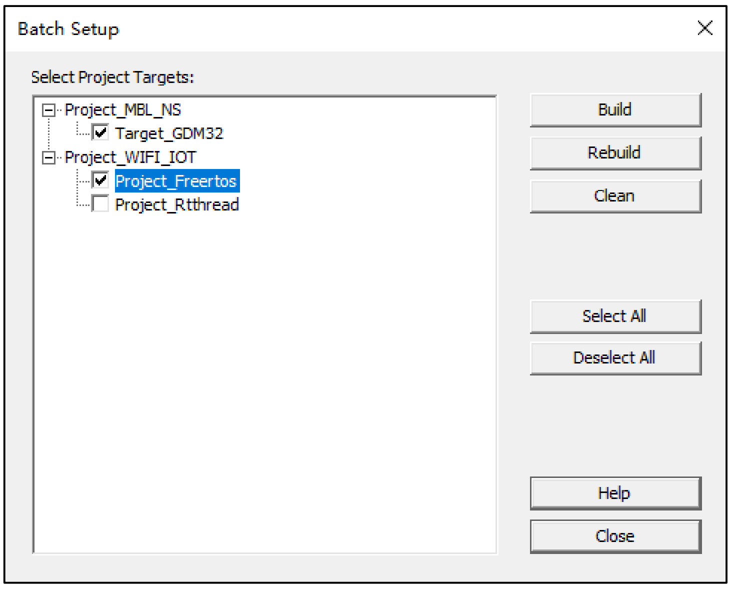
Batch compilation is an option, as shown in ***Figure 4-7. Batch compilation***. After clicking “Batch Build”, Project_MBL_NS and Project_WIFI_IOT are compiled in turn. Click on the "Batch Setup..." ,custom batch compilation is available, as shown in ***Figure 4-8. Batch compilation settings***. The OS for Project_WIFI_IOT can be FreeRTOS or RT-Thread.

Figure 4-7. Batch compilation



```
#include "gd32w51x.h"
#include "mbl_trace.h"
#include "mbl_region.h"
#include "mbl_image.h"
#include "mbl_sys.h"
#include "mbl_uart.h"
#include "mbl_flash.h"

typedef void (*img_fptr_t) (void);
int32_t mbl_err_process = ERR_PROCESS_ENDLESS
int32_t mbl_trace_level = MBL_WARN;

#if defined(__ARM_ARCH_8M_MAIN__) || defined(
/* Macros to pick linker symbols */
#if defined (__ICCARM__)
#define REGION(a, b) a##b
#define REGION_NAME(a, b) REGION(a, b)
#define REGION_DECLARE(a, b) extern uint32_t
REGION_DECLARE(CSTACK, $$Base);
```

Figure 4-8. Batch compilation settings



Project_MBL_NS
- [✓] Target_GDM32

Project_WIFI_IOT
- [✓] Project_Freertos
- [ ] Project_Rtthread

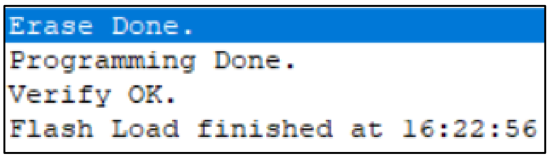
- User Command

  The "User Command" is defined on each Project and automatically runs xxx_afterbuild.bat after compiling and linking successfully. This script mainly extracts from the output AXF file to generate the BIN format firmware. If it is nspe_afterbuild.bat, it will do one more thing, which is to concatenate mbl_ns.bin and nspe.bin to get image_all.bin.

4.4. Download

Click the icon to download. Check whether the download is successful, as shown in ***Figure 4-9. KEIL download result***.

Figure 4-9. KEIL download result



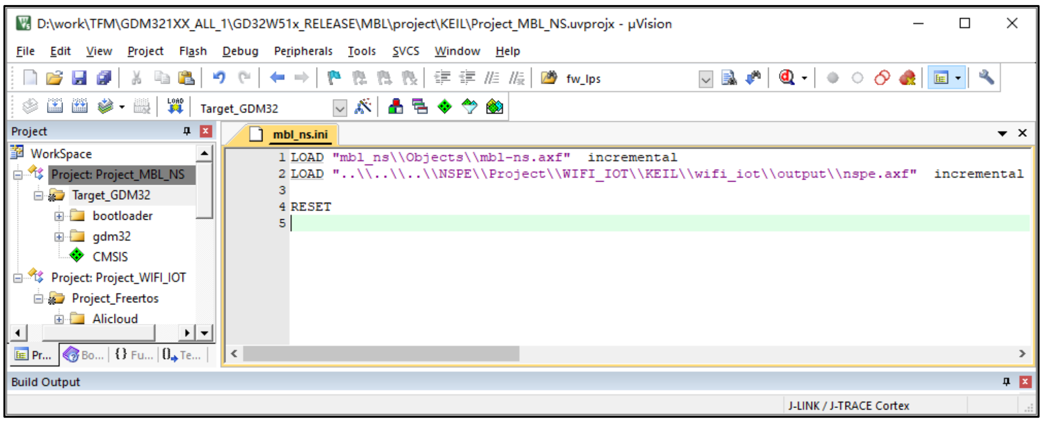
```
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 16:22:56
```

4.5. Debug

- Initialization File

  Before debugging, the "Initialization File (xxx.ini file) " needs to be edited. For cross-project debugging, the AXF files of multiple projects need to be loaded in, see GD32W51x_RELEASE / MBL / project / KEIL / mbl_ns.ini, as shown in ***Figure 4-10. Initialization file for debugging***.

Figure 4-10. Initialization file for debugging

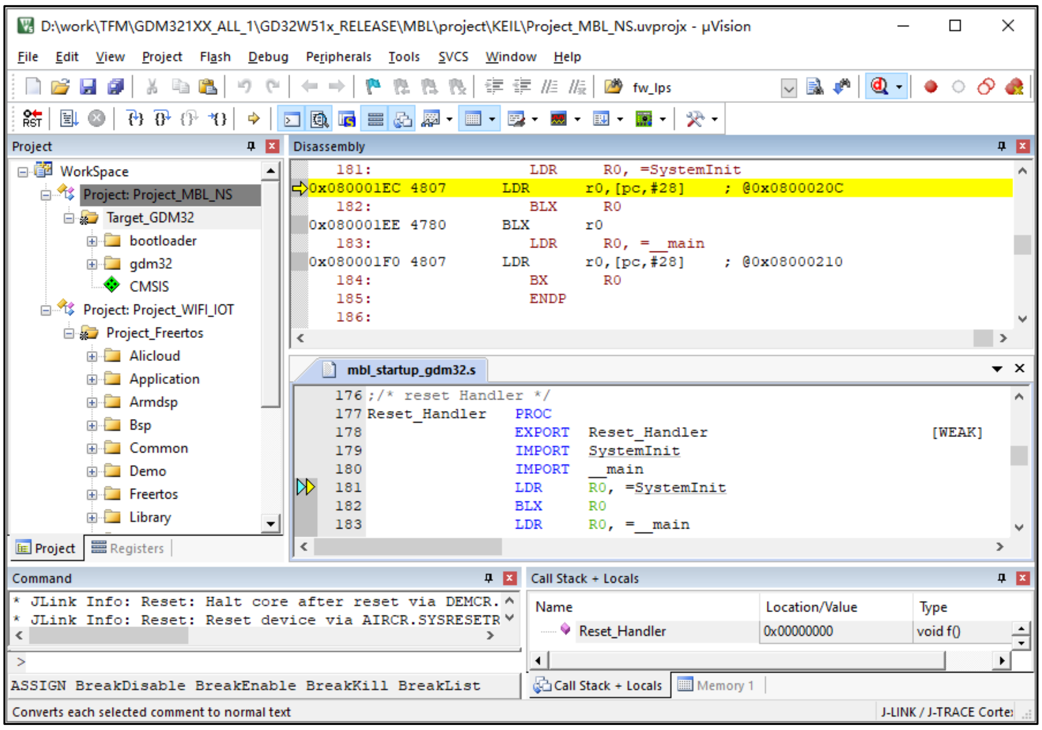

D:\work\TFM\GDM321XX_ALL_1\GD32W51x_RELEASE\MBL\project\KEIL\Project_MBL_NS.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
fw_lps
Target_GDM32
Project
WorkSpace
Project: Project_MBL_NS
Target_GDM32
bootloader
gdm32
CMSIS
Project: Project_WIFI_IOT
Project_Freertos
Alicloud
mbl_ns.ini
Build Output
J-LINK / J-TRACE Cortex

```
LOAD "mbl_ns\\Objects\\mbl-ns.axf"  incremental
LOAD "..\\..\\..\\NSPE\\Project\\WIFI_IOT\\KEIL\\wifi_iot\\output\\nspe.axf"  incremental

RESET
```

- Start debugging

  After the above ready, click on the icon ,Or click "Dubug" and then "Start / Stop Debug Session" to start single-step debugging, as shown in ***Figure 4-11. KEIL debugging interface***.

Figure 4-11. KEIL debugging interface



```
    181:                  LDR      R0, =SystemInit
0x080001EC 4807      LDR      r0,[pc,#28]     ; @0x0800020C
    182:                  BLX      R0
0x080001EE 4780      BLX      r0
    183:                  LDR      R0, =__main
0x080001F0 4807      LDR      r0,[pc,#28]     ; @0x08000210
    184:                  BX       R0
    185:                  ENDP
    186:
```

```
;/* reset Handler */
Reset_Handler    PROC
                 EXPORT  Reset_Handler                  [WEAK]
                 IMPORT  SystemInit
                 IMPORT  __main
                 LDR     R0, =SystemInit
                 BLX     R0
                 LDR     R0, =__main
```

```
* JLink Info: Reset: Halt core after reset via DEMCR.
* JLink Info: Reset: Reset device via AIRCR.SYSRESETR
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Location/Value	Type
Reset_Handler	0x00000000	void f()

5. IAR project

This chapter describes how to compile and debug the SDK in IAR.

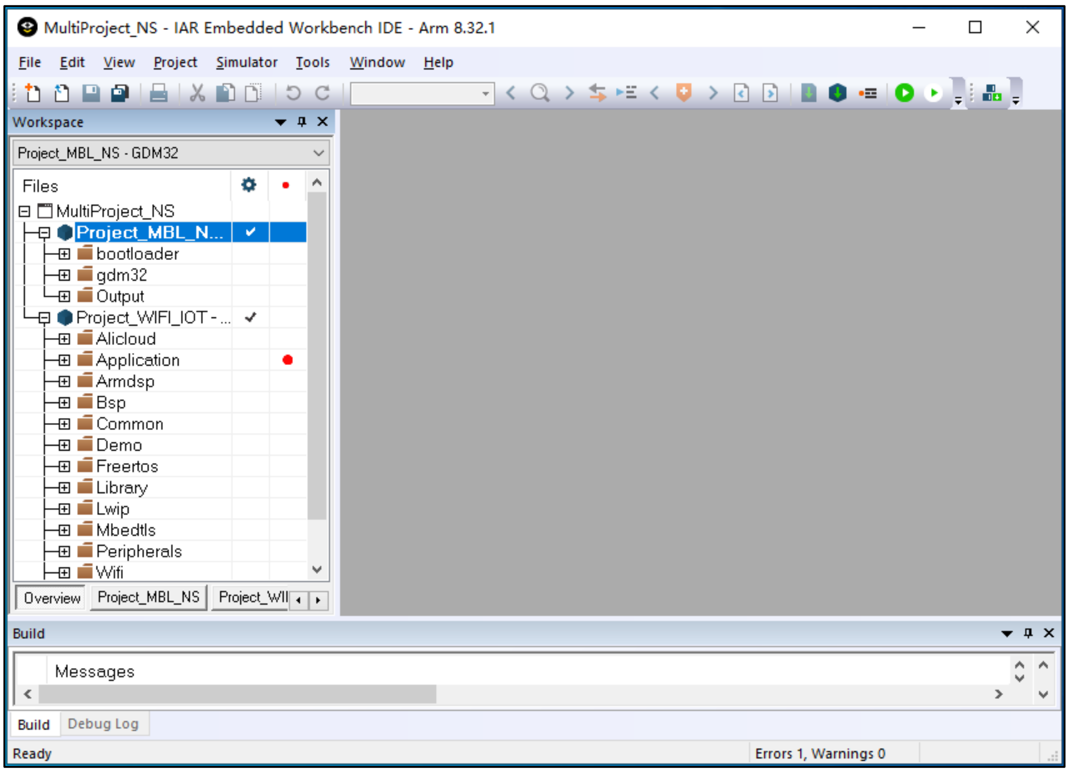
The project group consists of two projects: MBL_NS and WIFI_IOT. WIFI_IOT contains the WiFi protocol stack, peripheral drivers, applications, etc. MBL_NS is mainly responsible for selecting the correct firmware to run from two WIFI_IOT firmware (one for the current firmware and one for the upgraded firmware).

5.1. Project group

Find MultiProject_NS.eww from the GD32W51x_RELEASE root directory and double-click it to open the IAR workspace, as shown in ***Figure 5-1. IAR project group***.

Please confirm that IAR_GD32W51x_ADDON_1.x.x.exe has been installed.

Figure 5-1. IAR project group



5.2. Project configuration

IAR Project configuration, mainly reflected in "Options..." dialog box, which is the IAR option mentioned below. This includes Device selection, output file selection, compile options, link options, download and debug options, and more.

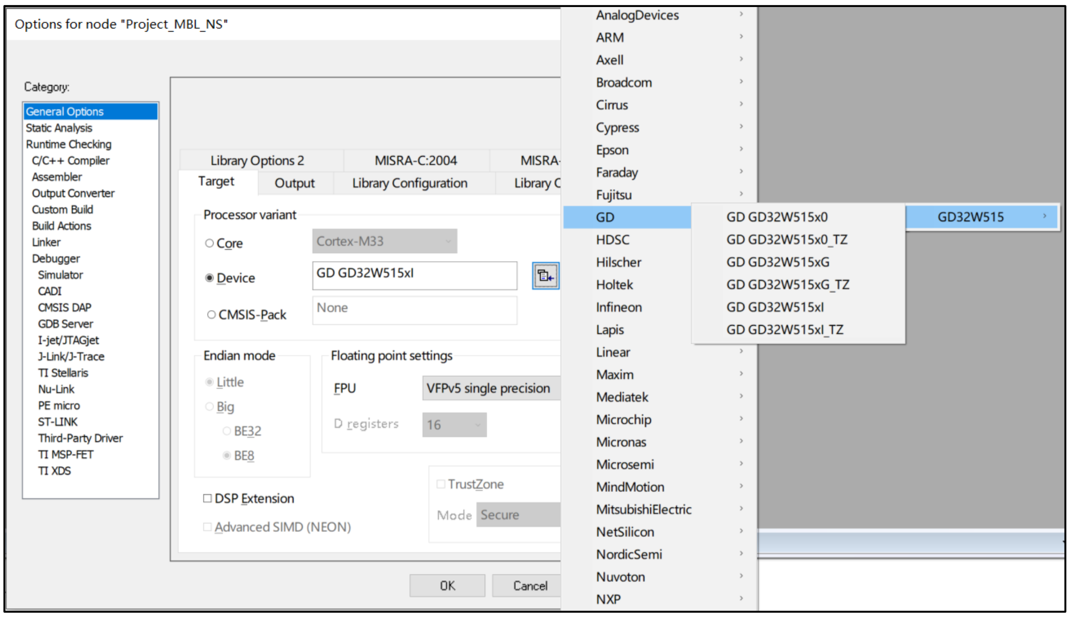
The main things that may need to be adjusted are Device selection and debugger configuration.

5.2.1. Device selection

The GD32W515 chip is available in a variety of types and can be seen on the "IAR Options -> General Options -> Target" page once the Addon is installed, as shown in ***Figure 5-2. IAR "Target" options page***. The default type is GD32W515xI, 2M SIP FLASH, and 448K SRAM.

If the chip is of another type, select it based on the actual type.

Figure 5-2. IAR "Target" options page


Options for node "Project_MBL_NS"

Processor variant
○ Core: Cortex-M33
● Device: GD GD32W515xI
○ CMSIS-Pack: None

Endian mode
● Little
○ Big
○ BE32
● BE8

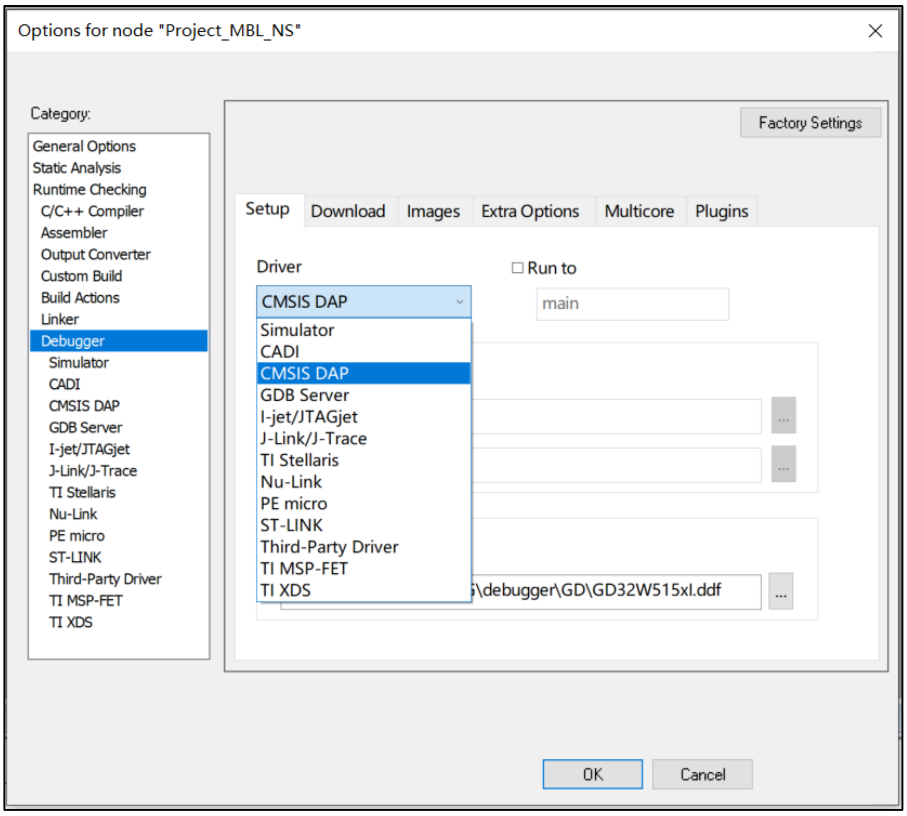
Floating point settings
FPU: VFPv5 single precision
D registers: 16

☐ DSP Extension
☐ Advanced SIMD (NEON)
☐ TrustZone
Mode: Secure

5.2.2. Debugger configuration

For the EVAL development board, select on-board GD-LINK and for the START development board, select on-board DAPLINK. The Driver on the Debugger Setup page is "CMSIS-DAP". For JLINK, select "J-LINK/J-TRACE Cortex". As shown in ***Figure 5-3. IAR "Debugger" options page***.

Figure 5-3. IAR “Debugger” options page

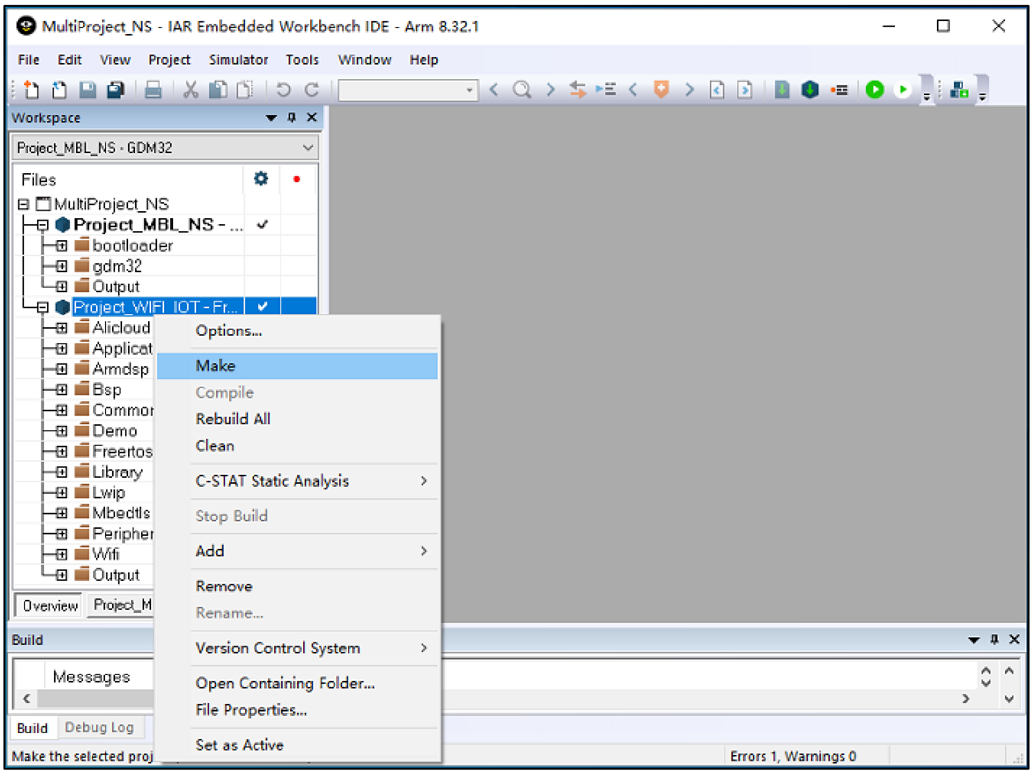


5.3. Compilation

- Compilation

  To compile, just right-click on the corresponding Project and select "Make". To recompile all, select “Rebuild All”, as shown in ***Figure 5-4. IAR compilation***.

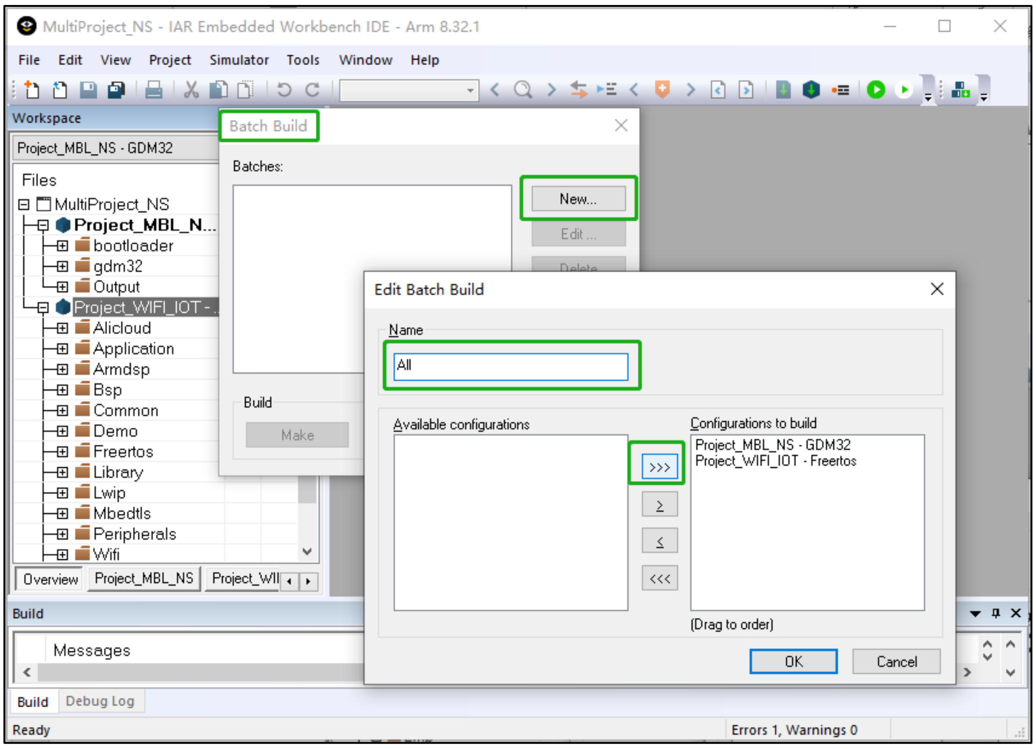
Figure 5-4. IAR compilation



- Batch compilation

Batch compilation needs to be set up first, click "Project" on the menu bar and select "Batch build..." from the drop-down menu and the “Batch Build” dialog box will pop up, as shown in ***Figure 5-5. IAR batch compilation settings***. After clicking "New", the "Edit Batch Build" dialog box will pop up. Select all projects to the right box, fill in "Name" and click "OK". Go back to the “Batch Build” dialog box, select “All” and click “Make” to compile Project_MBL and Project_WIFI_IOT successively.

Figure 5-5. IAR batch compilation settings



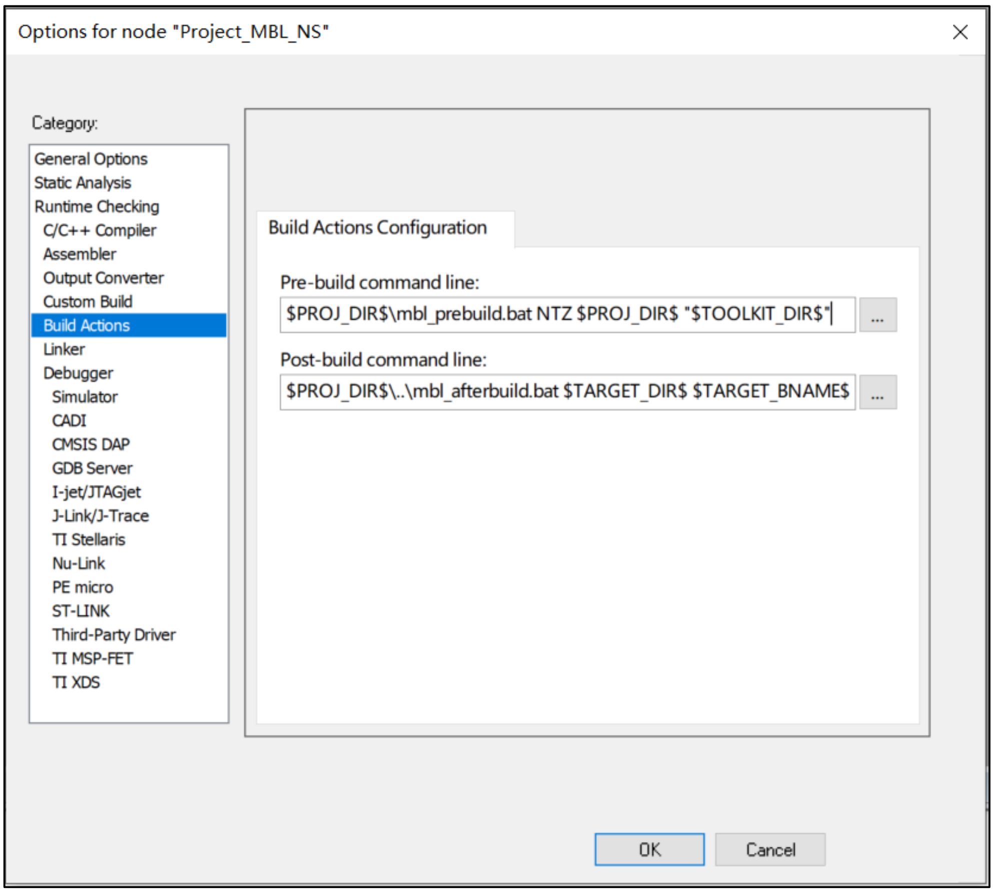
Name: All

Configurations to build:
Project_MBL_NS - GDM32
Project_WIFI_IOT - Freertos

- Build Actions

Each Project user can customize "Build Actions", as shown in ***Figure 5-6. IAR Build Actions***. Currently, the "Pre-build command line" and "Post-build command line" are defined on each project, which are executed before and after compilation, respectively. Xxx_prebuild.bat" is used to complete ICF file precompilation and generate compilation timestamps. "Xxx_afterbuild.bat" is mainly extracted from the output AXF file to generate the BIN format firmware. If it is nspe_afterbuild.bat, it will do one more thing, which is to concatenate mbl_ns.bin and nspe.bin to get image_all.bin.

Figure 5-6. IAR Build Actions



Build Actions Configuration

Pre-build command line:

```
$PROJ_DIR$\mbl_prebuild.bat NTZ $PROJ_DIR$ "$TOOLKIT_DIR$"
```

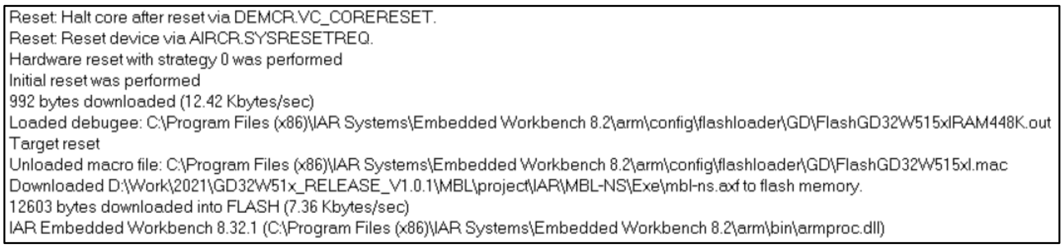
Post-build command line:

```
$PROJ_DIR$\..\mbl_afterbuild.bat $TARGET_DIR$ $TARGET_BNAME$
```

5.4. Download

Click "Project" and then "Download active application" to download. The correct burning Log displayed in the IAR Debug Log window is shown in ***Figure 5-7. IAR download result***.

Figure 5-7. IAR download result



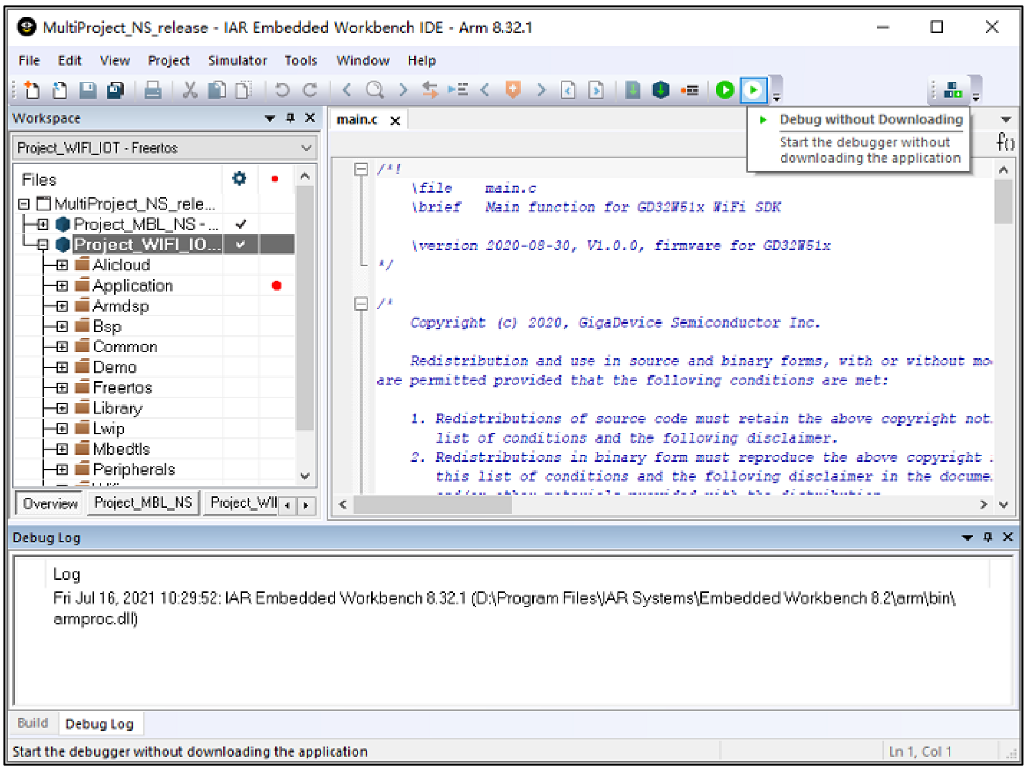
```
Reset: Halt core after reset via DEMCR.VC_CORERESET.
Reset: Reset device via AIRCR.SYSRESETREQ.
Hardware reset with strategy 0 was performed
Initial reset was performed
992 bytes downloaded (12.42 Kbytes/sec)
Loaded debugee: C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.2\arm\config\flashloader\GD\FlashGD32W515xIRAM448K.out
Target reset
Unloaded macro file: C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.2\arm\config\flashloader\GD\FlashGD32W515xI.mac
Downloaded D:\Work\2021\GD32W51x_RELEASE_V1.0.1\MBL\project\IAR\MBL-NS\Exe\mbl-ns.axf to flash memory.
12603 bytes downloaded into FLASH (7.36 Kbytes/sec)
IAR Embedded Workbench 8.32.1 (C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.2\arm\bin\armproc.dll)
```

5.5. Debug

■ Debug

Click “Debug without Downloading” to start single-step debugging, as shown in ***Figure 5-8. IAR debugging button*** and ***Figure 5-9. IAR debugging interface***.

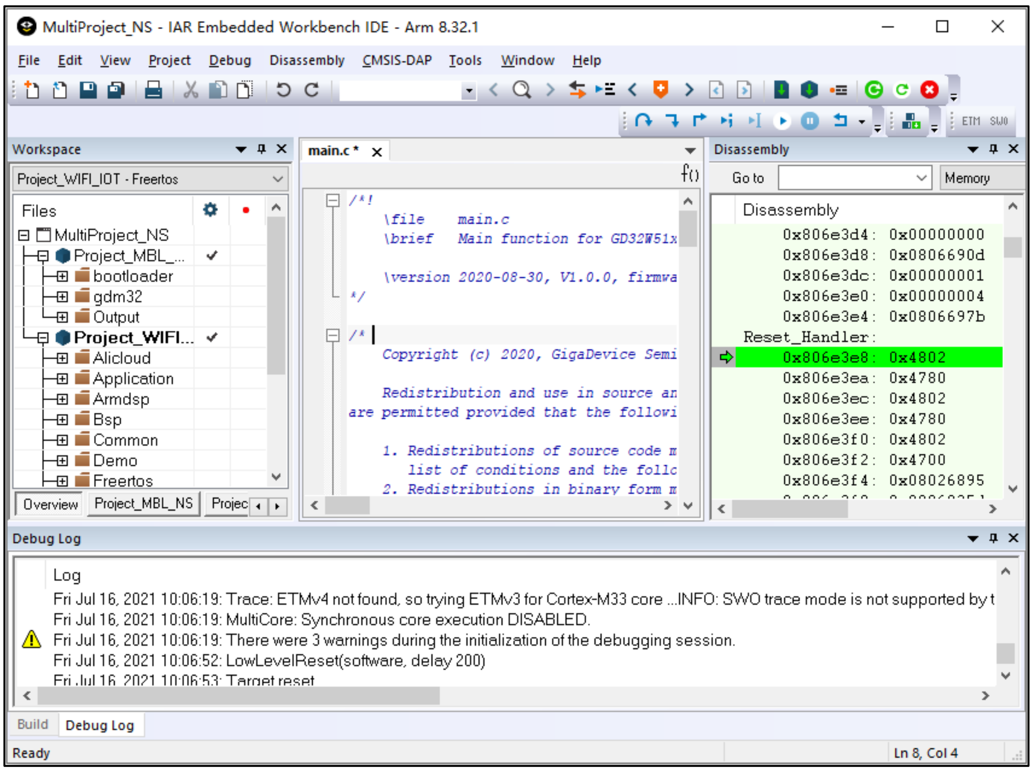
Figure 5-8. IAR debugging button


MultiProject_NS_release - IAR Embedded Workbench IDE - Arm 8.32.1
File Edit View Project Simulator Tools Window Help
Workspace
Project_WIFI_IOT - Freertos
Files
MultiProject_NS_rele...
Project_MBL_NS - ...
Project_WIFI_IO...
Alicloud
Application
Armdsp
Bsp
Common
Demo
Freertos
Library
Lwip
Mbedtls
Peripherals
Overview
Project_MBL_NS
Project_WI
main.c
Debug without Downloading
Start the debugger without downloading the application
Debug Log
Build
Debug Log
Start the debugger without downloading the application
Ln 1, Col 1

```
/*!
    \file    main.c
    \brief   Main function for GD32W51x WiFi SDK

    \version 2020-08-30, V1.0.0, firmware for GD32W51x
*/

/*
    Copyright (c) 2020, GigaDevice Semiconductor Inc.

    Redistribution and use in source and binary forms, with or without mo
are permitted provided that the following conditions are met:

    1. Redistributions of source code must retain the above copyright not
       list of conditions and the following disclaimer.
    2. Redistributions in binary form must reproduce the above copyright
       this list of conditions and the following disclaimer in the docume
```

```
Log
Fri Jul 16, 2021 10:29:52: IAR Embedded Workbench 8.32.1 (D:\Program Files\IAR Systems\Embedded Workbench 8.2\arm\bin\
armproc.dll)
```

Figure 5-9. IAR debugging interface



```
/*!
    \file    main.c
    \brief   Main function for GD32W51x

    \version 2020-08-30, V1.0.0, firmwa
*/

/*
    Copyright (c) 2020, GigaDevice Semi

    Redistribution and use in source an
are permitted provided that the followi

    1. Redistributions of source code m
       list of conditions and the follo
    2. Redistributions in binary form m
```

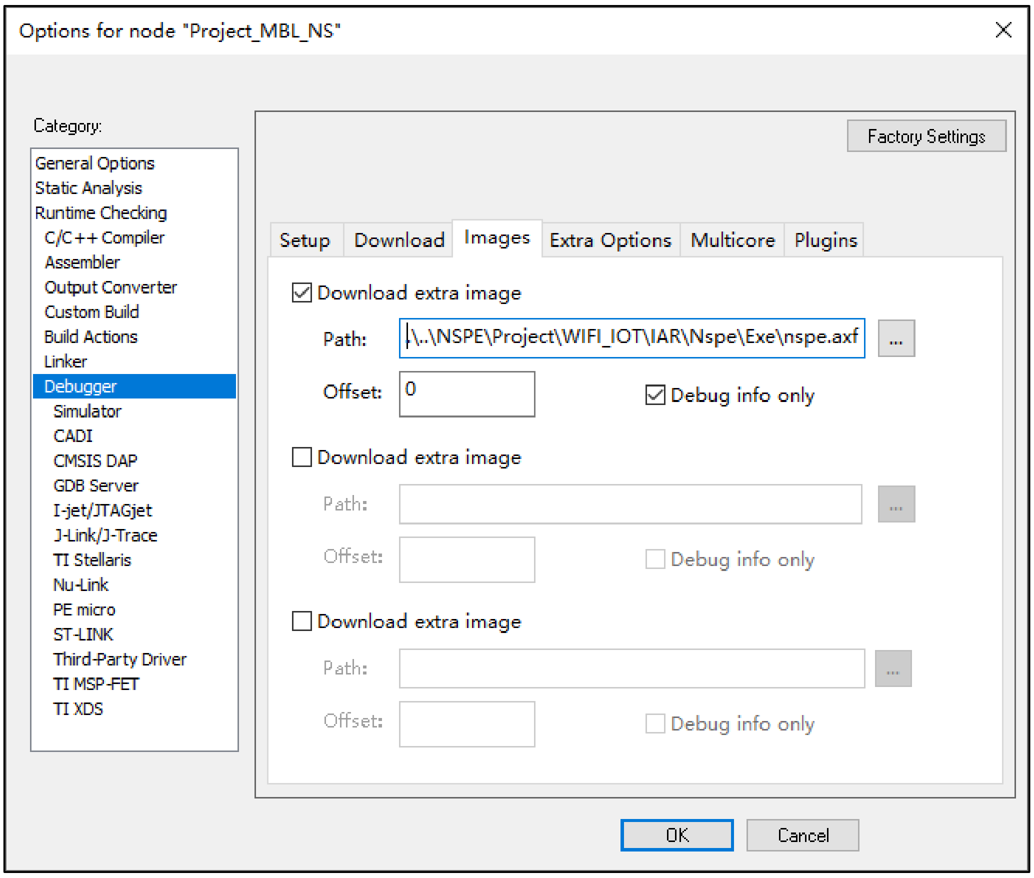
```
Disassembly
    0x806e3d4: 0x00000000
    0x806e3d8: 0x0806690d
    0x806e3dc: 0x00000001
    0x806e3e0: 0x00000004
    0x806e3e4: 0x0806697b
Reset_Handler:
    0x806e3e8: 0x4802
    0x806e3ea: 0x4780
    0x806e3ec: 0x4802
    0x806e3ee: 0x4780
    0x806e3f0: 0x4802
    0x806e3f2: 0x4700
    0x806e3f4: 0x08026895
```

```
Log
Fri Jul 16, 2021 10:06:19: Trace: ETMv4 not found, so trying ETMv3 for Cortex-M33 core ...INFO: SWO trace mode is not supported by t
Fri Jul 16, 2021 10:06:19: MultiCore: Synchronous core execution DISABLED.
Fri Jul 16, 2021 10:06:19: There were 3 warnings during the initialization of the debugging session.
Fri Jul 16, 2021 10:06:52: LowLevelReset(software, delay 200)
Fri Jul 16, 2021 10:06:53: Target reset
```

■ Debugging of multiple images

For cross-project debugging, on the “Debugger”->"Images" page, fill in the "Path" box with the output file (*.axf or *.out) path of the next containing debugging information, and fill in 0 for "Offset", and select "Debug info only", as shown in ***Figure 5-10. IAR multi-image debugging***.

Figure 5-10. IAR multi-image debugging



Options for node "Project_MBL_NS"

Category:

- General Options
- Static Analysis
- Runtime Checking
- C/C++ Compiler
- Assembler
- Output Converter
- Custom Build
- Build Actions
- Linker
- Debugger
- Simulator
- CADI
- CMSIS DAP
- GDB Server
- I-jet/JTAGjet
- J-Link/J-Trace
- TI Stellaris
- Nu-Link
- PE micro
- ST-LINK
- Third-Party Driver
- TI MSP-FET
- TI XDS

Factory Settings

Setup | Download | Images | Extra Options | Multicore | Plugins

[✓] Download extra image
Path: \..\NSPE\Project\WIFI_IOT\IAR\Nspe\Exe\nspe.axf
Offset: 0
[✓] Debug info only

[ ] Download extra image
Path:
Offset:
[ ] Debug info only

[ ] Download extra image
Path:
Offset:
[ ] Debug info only

OK | Cancel

6. GCC project

This chapter describes how to compile and debug the SDK in GCC.

Currently, only JLINK debugging is supported, and the CMSIS-DAP debugging function will be added later.

6.1. Project group

Using cmake + gnu make tools to build the SDK, there are two main targets: WIFI_IOT and MBL_NS. WIFI_IOT contains the WiFi protocol stack, peripheral drivers, applications, etc. MBL_NS is mainly responsible for selecting the correct firmware to run from two WIFI_IOT firmware (one for the current firmware and one for the upgraded firmware).

6.2. Compilation

Start the Windows command window, do not use PowerShell. Go to “GD32W51x_RELEASE” directory and run the following command:

- Creates the cmake build directory.
  - mkdir cmake_build
- Enter the cmake build directory.
  - cd cmake_build
- Run the cmake command to generate the Makefile.
  - cmake -G "Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE: PATH =. / scripts / cmake / toolchain.cmake
- Run the make command to compile.
  - make -j

The preceding steps have been written into gcc_build_ns.bat. Double-click it to complete the first compilation.

- If only nspe needs to compile, go to the cmake_build directory and run the following command.
  - make nspe -j
- If only mbl needs to compile, go to the cmake_build directory and run the following command.
  - make mbl-ns -j
  - Download

Start the Windows command window, do not use PowerShell. Go to “GD32W51x_RELEASE” directory and run the following command:

- gcc_download_ns.bat JLINK

Scripts / images / image-all.bin will be downloaded automatically using JLINK.

6.3. Debug

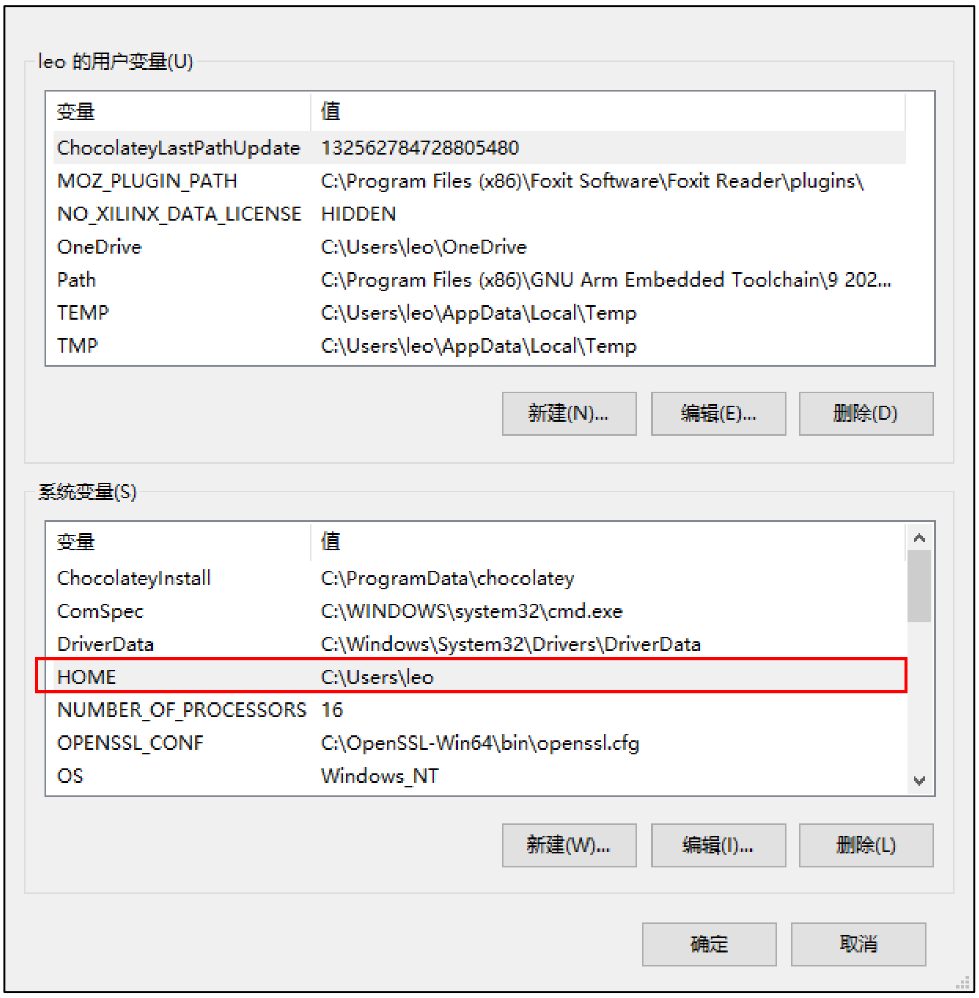
6.3.1. Start the GDB server

The debugging tool is GDB + JLink. Ensure that the Jlink connection is correct.

Conditions preparation:

- Create an environment variable HOME, the recommended value is the user directory, such as C :/ Users / leo, see ***Figure 6-1. Create the environment variable HOME*** for details.

Figure 6-1. Create the environment variable HOME



leo 的用户变量(U)

变量	值
ChocolateyLastPathUpdate	132562784728805480
MOZ_PLUGIN_PATH	C:\Program Files (x86)\Foxit Software\Foxit Reader\plugins\
NO_XILINX_DATA_LICENSE	HIDDEN
OneDrive	C:\Users\leo\OneDrive
Path	C:\Program Files (x86)\GNU Arm Embedded Toolchain\9 202...
TEMP	C:\Users\leo\AppData\Local\Temp
TMP	C:\Users\leo\AppData\Local\Temp

系统变量(S)

变量	值
ChocolateyInstall	C:\ProgramData\chocolatey
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
HOME	C:\Users\leo
NUMBER_OF_PROCESSORS	16
OPENSSL_CONF	C:\OpenSSL-Win64\bin\openssl.cfg
OS	Windows_NT

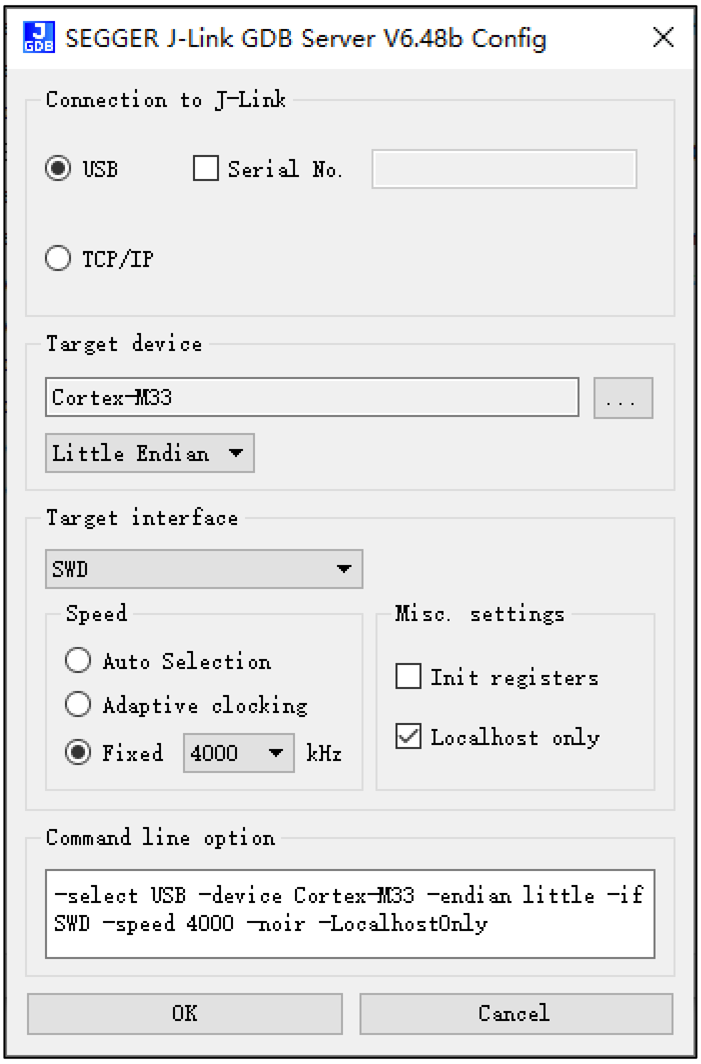
- .In the directory pointed to by the environment variable HOME, create the file named ".gdbinit" with the following contents: set auto-load safe-path /.

The steps to start debugging are as follows:

- Start the JLink Server and double-click JLinkGDBServer.exe in the installation directory of the JLink driver. See Figure 6-2. Start the JLINK GDB Server for the configuration. Click “OK” after the configuration is complete.

– C :/ Program Files (x86) / SEGGER / JLink / JLinkGDBServer.exe.

Figure 6-2. Start the JLINK GDB Server


SEGGER J-Link GDB Server V6.48b Config

Connection to J-Link
[✓ USB] Serial No.
TCP/IP

Target device
Cortex-M33
Little Endian

Target interface
SWD

Speed
Auto Selection
Adaptive clocking
[✓ Fixed] 4000 kHz

Misc. settings
Init registers
[✓ Localhost only]

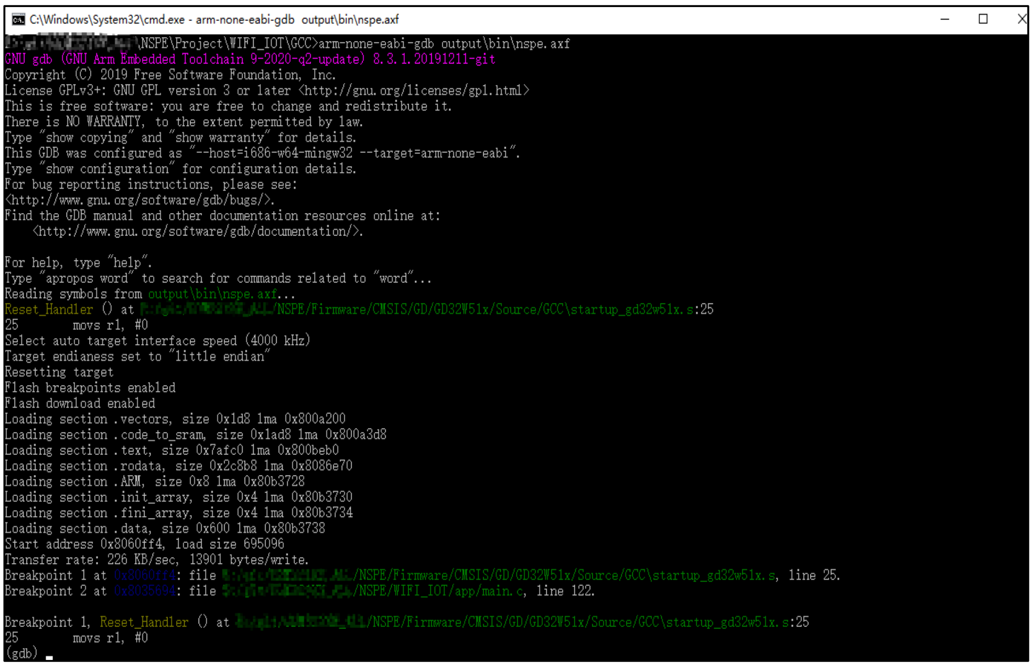
Command line option
```
-select USB -device Cortex-M33 -endian little -if
SWD -speed 4000 -noir -LocalhostOnly
```
OK Cancel

6.3.2. Start the GDB Client for debugging

After starting GDB server, the GDB client is ready for debugging. In addition, start the windows command window, go to the directory “GD32W51x_RELEASE / NSPE / Project / WIFI_IOT / GCC”, enter the command gdb, as shown in Figure 6-3. GDB debug window:

■ arm-none-eabi-gdb output / bin / nspe.axf.
■ Press Enter when prompted until the program stops at Reset_Handler.

Figure 6-3. GDB debug window

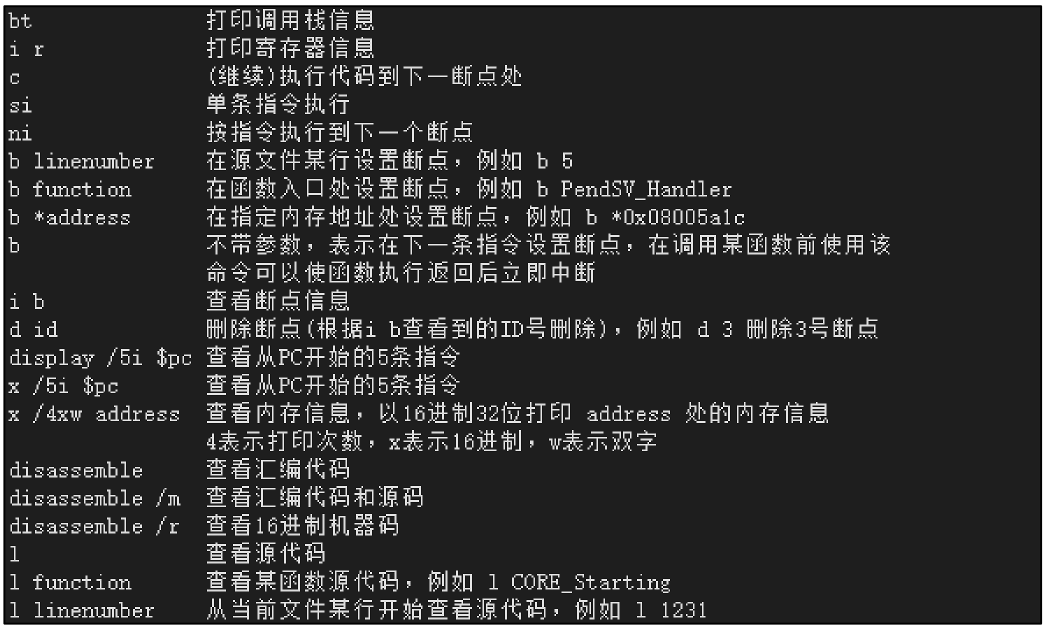


C:\Windows\System32\cmd.exe - arm-none-eabi-gdb output\bin\nspe.axf

```
[redacted]\NSPE\Project\WIFI_IOT\GCC>arm-none-eabi-gdb output\bin\nspe.axf
GNU gdb (GNU Arm Embedded Toolchain 9-2020-q2-update) 8.3.1.20191211-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=i686-w64-mingw32 --target=arm-none-eabi".
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"...
Reading symbols from output\bin\nspe.axf...
Reset_Handler () at [redacted]/NSPE/Firmware/CMSIS/GD/GD32W51x/Source/GCC\startup_gd32w51x.s:25
25          movs r1, #0
Select auto target interface speed (4000 kHz)
Target endianess set to "little endian"
Resetting target
Flash breakpoints enabled
Flash download enabled
Loading section .vectors, size 0x1d8 lma 0x800a200
Loading section .code_to_sram, size 0x1ad8 lma 0x800a3d8
Loading section .text, size 0x7afc0 lma 0x800beb0
Loading section .rodata, size 0x2c8b8 lma 0x8086e70
Loading section .ARM, size 0x8 lma 0x80b3728
Loading section .init_array, size 0x4 lma 0x80b3730
Loading section .fini_array, size 0x4 lma 0x80b3734
Loading section .data, size 0x600 lma 0x80b3738
Start address 0x8060ff4, load size 695096
Transfer rate: 226 KB/sec, 13901 bytes/write.
Breakpoint 1 at 0x8060ff4: file [redacted]/NSPE/Firmware/CMSIS/GD/GD32W51x/Source/GCC\startup_gd32w51x.s, line 25.
Breakpoint 2 at 0x8035694: file [redacted]/NSPE/WIFI_IOT/app/main.c, line 122.

Breakpoint 1, Reset_Handler () at [redacted]/NSPE/Firmware/CMSIS/GD/GD32W51x/Source/GCC\startup_gd32w51x.s:25
25          movs r1, #0
(gdb) _
```

At this point, program debugging can be done by using the gdb command. The following are common command references, as shown in ***Figure 6-4. Common commands***. For details, refer to the official document: ***https://www.gnu.org/software/gdb/documentation/***.

Figure 6-4. Common commands



bt	打印调用栈信息
i r	打印寄存器信息
c	(继续)执行代码到下一断点处
si	单条指令执行
ni	按指令执行到下一个断点
b linenumber	在源文件某行设置断点，例如 b 5
b function	在函数入口处设置断点，例如 b PendSV_Handler
b *address	在指定内存地址处设置断点，例如 b *0x08005a1c
b	不带参数，表示在下一条指令设置断点，在调用某函数前使用该命令可以使函数执行返回后立即中断
i b	查看断点信息
d id	删除断点(根据i b查看到的ID号删除)，例如 d 3 删除3号断点
display /5i $pc	查看从PC开始的5条指令
x /5i $pc	查看从PC开始的5条指令
x /4xw address	查看内存信息，以16进制32位打印 address 处的内存信息 4表示打印次数，x表示16进制，w表示双字
disassemble	查看汇编代码
disassemble /m	查看汇编代码和源码
disassemble /r	查看16进制机器码
l	查看源代码
l function	查看某函数源代码，例如 l CORE_Starting
l linenumber	从当前文件某行开始查看源代码，例如 l 1231

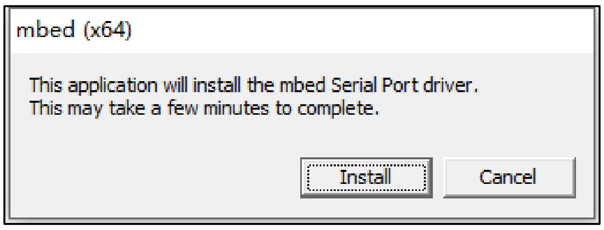
7. Q&A

7.1. DAPLINK disk identification

When the START development board is used, if the serial port cannot be identified as "mbedSerialPort (COMx)" and the operating system is Windows 7 or later, install the Mbed serial port driver.

- Driver download
  - https://os.mbed.com/media/downloads/drivers/mbedWinSerial_16466.exe.
- Installation
  - Double-click "mbedWinSerial_16466.exe",and then click " Install " ,as shown in Figure 7-1. Mbed serial port driver installation. After the installation is successful, click "Finish".

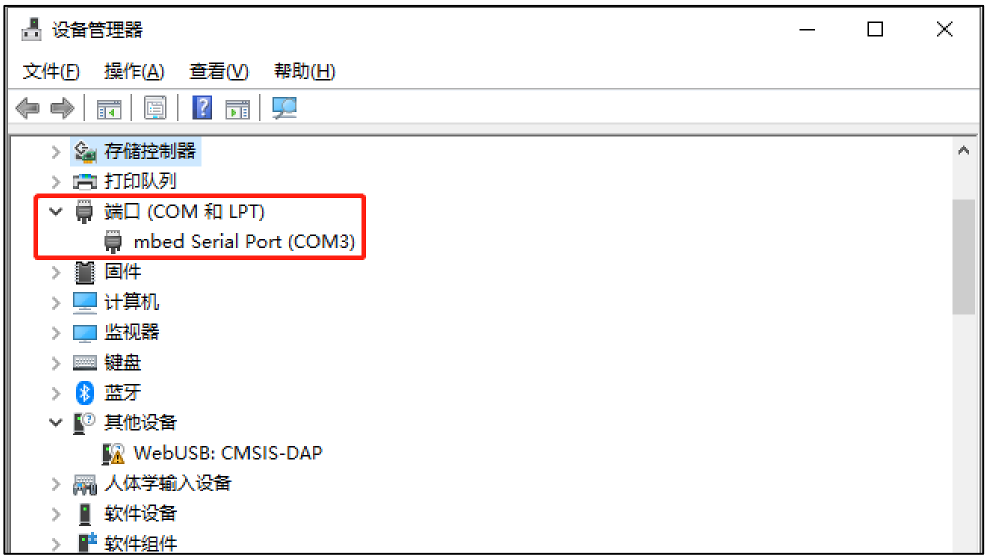
Figure 7-1. Mbed serial port driver installation


mbed (x64)

This application will install the mbed Serial Port driver.
This may take a few minutes to complete.

- Check
  - After the installation is successful, "mbed Serial Port (COMx)" is displayed on the device manager, as shown in Figure 7-2. List of device manager serial ports. The "DAPLINK" disk can be seen in the list of devices and drivers, as shown in Figure 3-6. List of devices and drivers.

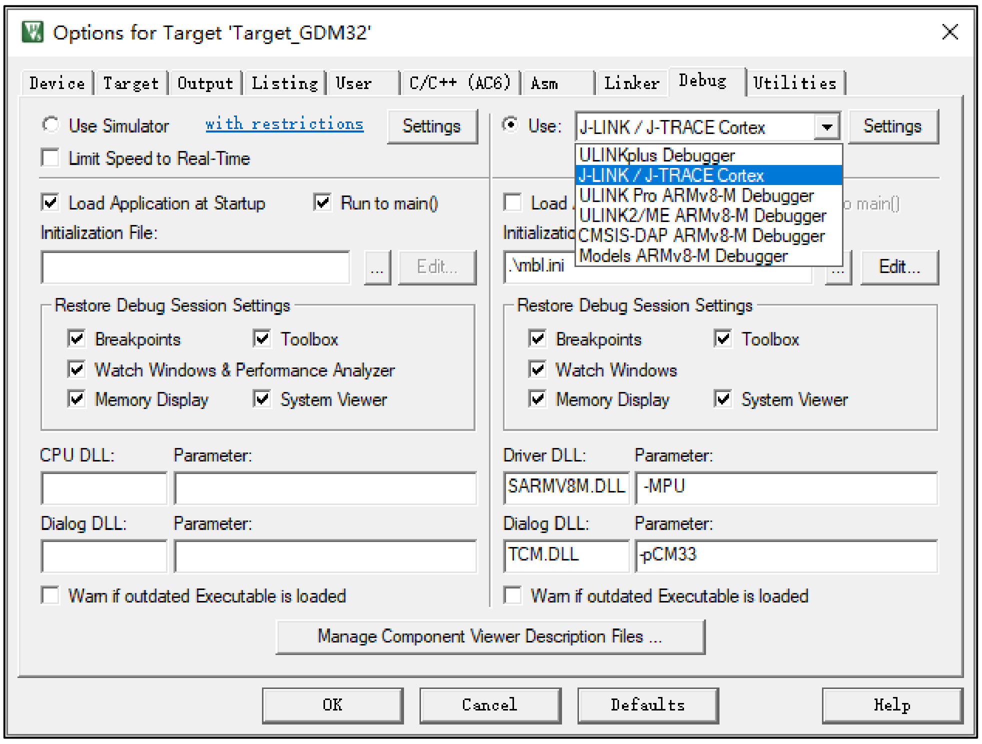
Figure 7-2. List of device manager serial ports



7.2. CM33 debugging by KEIL+JLINK

By default, MDK5.25 does not support debugging ARM CM33 with JLINK. When ARM CM33 needs to use JLINK debugging, if the option to select JLINK debugger cannot be found in the debugger selection column on the Debug page of KEIL tool engineering options, as shown in ***Figure 7-3. KEIL DEBUG options***, the "C :/ Keil_v5 / TOOLS.INI" file needs to be modified.

Figure 7-3. KEIL DEBUG options



Options for Target 'Target_GDM32'

Device | Target | Output | Listing | User | C/C++ (AC6) | Asm | Linker | Debug | Utilities

- ( ) Use Simulator — with restrictions — Settings
- [ ] Limit Speed to Real-Time
- [✓] Load Application at Startup — [✓] Run to main()
- Initialization File:
- Restore Debug Session Settings: [✓] Breakpoints [✓] Toolbox [✓] Watch Windows & Performance Analyzer [✓] Memory Display [✓] System Viewer
- CPU DLL: — Parameter:
- Dialog DLL: — Parameter:
- [ ] Warn if outdated Executable is loaded

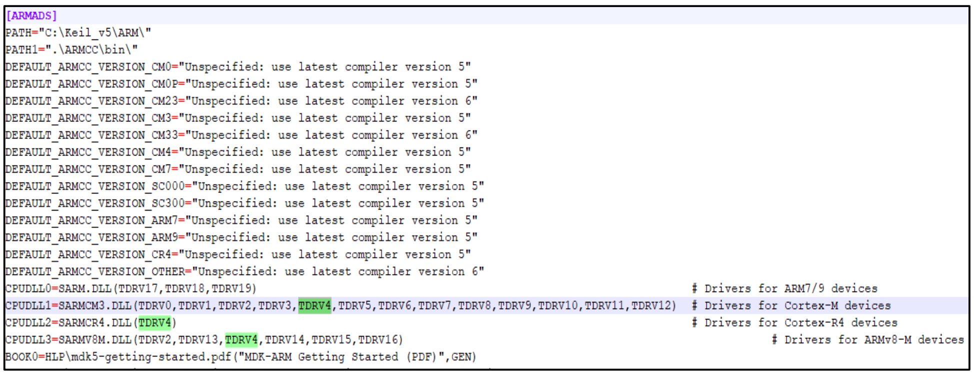
- (●) Use: J-LINK / J-TRACE Cortex — Settings
  - ULINKplus Debugger
  - J-LINK / J-TRACE Cortex
  - ULINK Pro ARMv8-M Debugger
  - ULINK2/ME ARMv8-M Debugger
  - CMSIS-DAP ARMv8-M Debugger
  - Models ARMv8-M Debugger
- [ ] Load — main()
- Initialization File: .\mbl.ini — Edit...
- Restore Debug Session Settings: [✓] Breakpoints [✓] Toolbox [✓] Watch Windows [✓] Memory Display [✓] System Viewer
- Driver DLL: SARMV8M.DLL — Parameter: -MPU
- Dialog DLL: TCM.DLL — Parameter: -pCM33
- [ ] Warn if outdated Executable is loaded

Manage Component Viewer Description Files ...

OK | Cancel | Defaults | Help

Add TDRVx corresponding to J-LINK in line CPUDLL3 of [ARMADS] in TOOLS.INI file to increase support for JLINK. Using J-LINK debugging development board needs to add "TDRV4", according to ***Figure 7-4. Add tools supported by KEIL***, and add "TDRV4" to the corresponding position. After the modification, re-open the KEIL tool and select "J-LINK/J-TRACE Cortex" to debug ARM CM33.

Figure 7-4. Add tools supported by KEIL



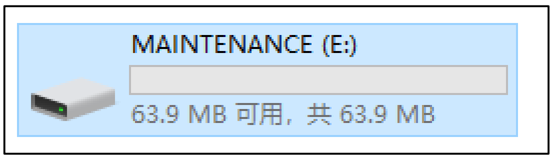
```
[ARMADS]
PATH="C:\Keil_v5\ARM\"
PATH1=".\ARMCC\bin\"
DEFAULT_ARMCC_VERSION_CM0="Unspecified: use latest compiler version 5"
DEFAULT_ARMCC_VERSION_CM0P="Unspecified: use latest compiler version 5"
DEFAULT_ARMCC_VERSION_CM23="Unspecified: use latest compiler version 6"
DEFAULT_ARMCC_VERSION_CM3="Unspecified: use latest compiler version 5"
DEFAULT_ARMCC_VERSION_CM33="Unspecified: use latest compiler version 6"
DEFAULT_ARMCC_VERSION_CM4="Unspecified: use latest compiler version 5"
DEFAULT_ARMCC_VERSION_CM7="Unspecified: use latest compiler version 5"
DEFAULT_ARMCC_VERSION_SC000="Unspecified: use latest compiler version 5"
DEFAULT_ARMCC_VERSION_SC300="Unspecified: use latest compiler version 5"
DEFAULT_ARMCC_VERSION_ARM7="Unspecified: use latest compiler version 5"
DEFAULT_ARMCC_VERSION_ARM9="Unspecified: use latest compiler version 5"
DEFAULT_ARMCC_VERSION_CR4="Unspecified: use latest compiler version 5"
DEFAULT_ARMCC_VERSION_OTHER="Unspecified: use latest compiler version 6"
CPUDLL0=SARM.DLL(TDRV17,TDRV18,TDRV19)                                                  # Drivers for ARM7/9 devices
CPUDLL1=SARMCM3.DLL(TDRV0,TDRV1,TDRV2,TDRV3,TDRV4,TDRV5,TDRV6,TDRV7,TDRV8,TDRV9,TDRV10,TDRV11,TDRV12)  # Drivers for Cortex-M devices
CPUDLL2=SARMCR4.DLL(TDRV4)                                                              # Drivers for Cortex-R4 devices
CPUDLL3=SARMV8M.DLL(TDRV2,TDRV13,TDRV4,TDRV14,TDRV15,TDRV16)                                      # Drivers for ARMv8-M devices
BOOK0=HLP\mdk5-getting-started.pdf("MDK-ARM Getting Started (PDF)",GEN)
```

7.3. DAP chip firmware upgrade

If the DAP chip on the START development board needs firmware upgrade, keep the “Reset” button on the development board pressed, and then use the USB cable to power on, the computer will detect the MAINTENANCE disk at this time, as shown in ***Figure 7-5. MAINTENANCE disk***.

Copy gd32w51x_if_crc_qspi.bin (for chip GD32W51x EXT FLASH) or gd32w51x_if_crc _fmc.bin (for chip GD32W51x SIP FLASH) to the MAINTENANCE disk. Upon success, the new DAP firmware will run and the DAPLINK disk will appear.

Figure 7-5. MAINTENANCE disk


MAINTENANCE (E:)
63.9 MB 可用，共 63.9 MB

7.4. No image error

Print ERR: No image to boot (ret = -5).

Cause: The previous booting WIFI_IOT failed and MBL recorded that the IMAGE was running abnormally. If the other IMAGE was not burned or a booting exception occurred, it would print this message. That is to say, MBL considers that there is no legal IMAGE to jump to, and the boot fails.

Solution: MBL can be burned again, after burning the IMAGE state will be cleared.

7.5. Code runs in SRAM

If programs need to run faster to achieve higher performance, consider moving them to the SRAM.

- Use KEIL

  Open GD32W51x_RELEASE / NSPE / Project / WIFI_IOT / KEIL / Project.sct for editing, add the required files or functions to the Execution Region of RW_IRAM2, such as:

  - port.o (+RO)

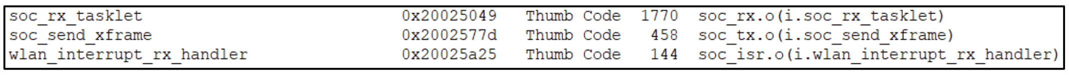
    Put the entire port.c file into the SRAM to run.

  - tasks.o (.text.xTaskIncrementTick)

    Put the xTaskIncrementTick () function in tasks.c into the SRAM to run.

  If the function is not added successfully, check the “GD32W51x_RELEASE / nspe / Project / WIFI_IOT / KEIL / wifi_iot / Freertos / List / nspe.map” and find the correct symbol name according to the function name, as shown in ***Figure 7-6. NSPE MAP file***.

Figure 7-6. NSPE MAP file



```
soc_rx_tasklet                          0x20025049   Thumb Code  1770  soc_rx.o(i.soc_rx_tasklet)
soc_send_xframe                         0x2002577d   Thumb Code   458  soc_tx.o(i.soc_send_xframe)
wlan_interrupt_rx_handler               0x20025a25   Thumb Code   144  soc_isr.o(i.wlan_interrupt_rx_handler)
```

- Use IAR

  If some functions need to be run in SRAM, add SECTION_RAM_CODE to the end of the function, such as:

  - BaseType_t xTaskIncrementTick (void) SECTION_RAM_CODE

- Use GCC

  Open "D32W51x_RELEASE / NSPE / Project / WIFI_IOT / GCC / nspe_gdm32_ns.ld" and find the line ".code_to_sRAM: ". The brace contains code that runs in SRAM. If new content needs to be added, it can be added at the end. Refer to existing documents in the format, such as:

  - KEEP (*port.o* (.text* .rodata*))

  Put the entire port.c file into the SRAM to run.

  - KEEP (*tasks.o* (.text.xTaskIncrementTick))

  Put the xTaskIncrementTick () function in tasks.c into the SRAM to run.

8. Revision history

Table 8-1. Revision history

Revision	Description	Date
1.0	Initial Release	Mar.25, 2022