GigaDevice Semiconductor Inc.

GD32VW553 Quick Development Guide

Application Note
AN154

Revision 1.0

(July 2023)

Table of Contents

List of Figures

List of Tables

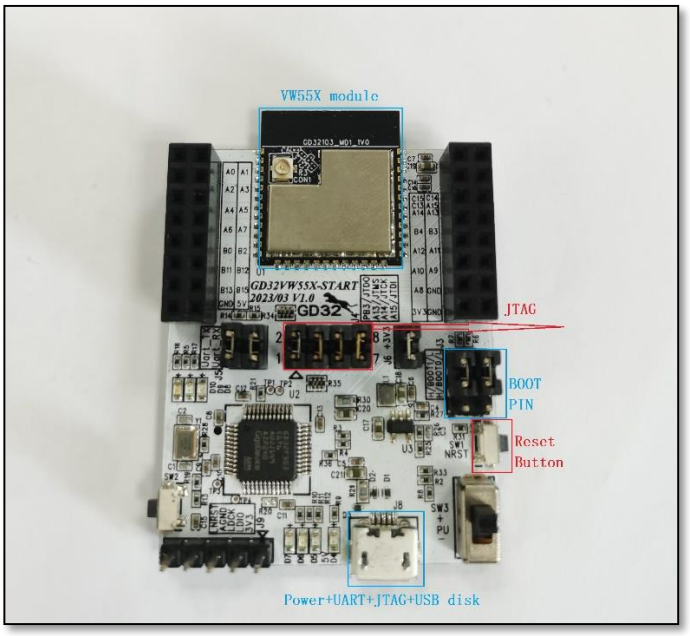
1. Introduction to development board

1.1. Picture of real development board

1.1.1. The START development board

The START development board consists of a baseboard and a module equipped with the GD32VW55x Wi-Fi+BLE chip.

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



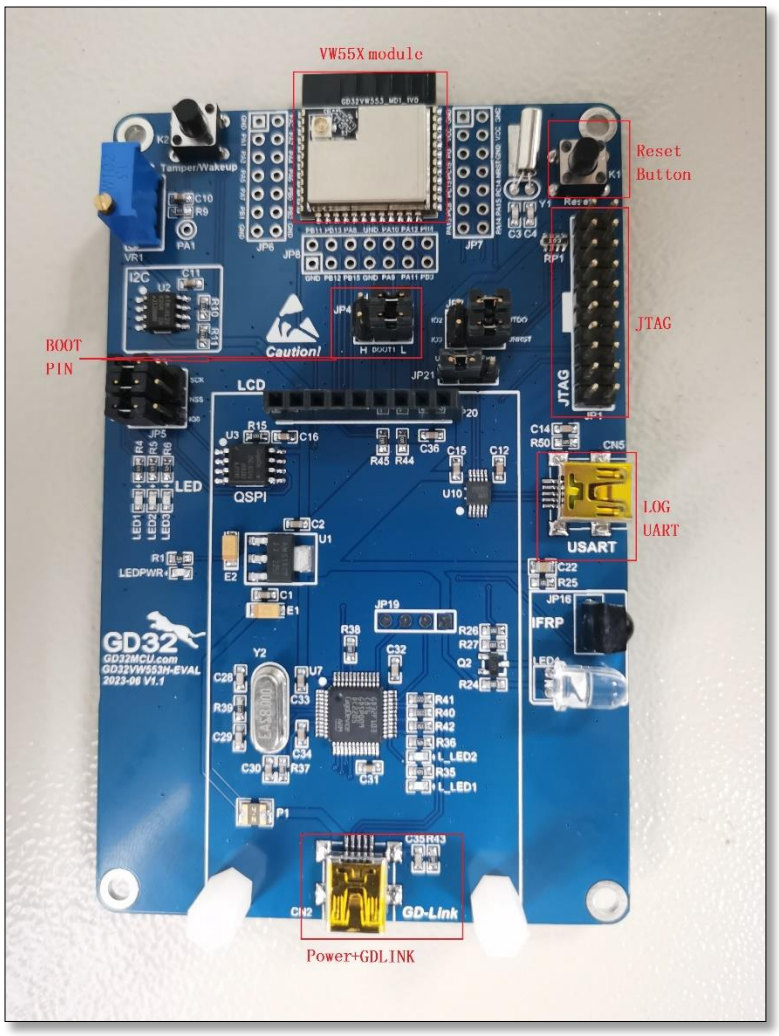
Developers mainly focus on the following parts of the development board, which have been marked in the figure.

- Boot mode (Boot PIN);
- Power supply port (power supply);
- View log (UART);
- Debugger interface (DAPLINK, JLINK, or GDLINK);
- Reboot (Reset Button).

1.1.2. The EVAL development board

The EVAL development board consists of a baseboard and a module equipped with the GD32VW55x Wi-Fi+BLE chip. The baseboard provides many peripheral test ports, such as I2C, IFRP, ADC and so on.

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



Developers mainly focus on the following parts of the development board, which have been marked in the figure.

- Boot mode (Boot PIN);
- Power supply port (power supply);
- View log (UART);
- Debugger interface (JLINK, or GDLINK);
- Reboot (Reset Button).

For the START development board and the EVAL development board, the SDK configuration is different and different macros need to be selected to enable them. As shown below, the SDK selects the START development board configuration as the default. The configuration file is GD32VW55x_RELEASE/config/platform_def.h.

Figure 1-3. Development Board Type Configuration

```
// board type
#define PLATFORM_BOARD_32VW55X_START    0
#define PLATFORM_BOARD_32VW55X_EVAL     1
#ifdef CONFIG_PLATFORM_ASIC
#define CONFIG_BOARD                    PLATFORM_BOARD_32VW55X_START
#endif
```

1.2. Boot mode

GD32VW55x can boot from ROM, FLASH, or SRAM.

The level selection of the two pins BOOT0 and BOOT1 in the BOOT SWD box of the development board determines the boot mode. See ***Table 1-1. Boot mode***. For more instructions on the boot mode, please refer to the document "GD32VW55x_User_Manual".

Table 1-1. Boot mode

EFBOOTLK	BOOT0	BOOT1	EFSB	Boot address	Boot area
0	0	-	0	0x08000000	SIP Flash
0	0	-	1	0x0BF46000	secure boot
0	1	0	-	0x0BF40000	Bootloader/ROM
0	1	1	-	0x20000000	SRAM
1	0	-	0	0x08000000	SIP Flash
1	0	-	1	0x0BF46000	Secure boot
1	1	-	-	0x0BF40000	Bootloader/ROM

1.3. Debugger interface

For START development board, it comes with a DAPLINK(GD32F303) debugger that can be used with OpenOCD. Due to the limitations of the chip capacity, debugging and download speeds are slow. It is recommended to use an external debugger (GDLINK or JLINK) at the JTAG interface of the board for debugging and download. The DAP chip also integrates the UART function, so only one USB cable is required to supply power, debug, and view the log. Connect the pins JCLK and JTWS to the lower two pins through jumper caps, and then download and debug the code through DAPLINK. ***Figure 1-1. The picture of the START development board*** shows how to debug through DAPLINK.

For EVAL development board, GDLINK or JLINK debugger can be used for debugging and download. DAPLINK is not supported.

1.4. Download interface

For START development board, in addition to the firmware download through the debugger mentioned in the previous section, if debugging is not required and only the firmware needs to be downloaded, it can also be downloaded by dragging it into the USB disk. Connect the development board to the computer through a USB cable, and the DAPLINK disk as shown in *Figure 1-4. List of devices and drivers* is displayed. Copy the image-all.bin file(see subsequent chapters) directly into the DAPLINK disk to complete FLASH burning of the GD32VW55x chip.

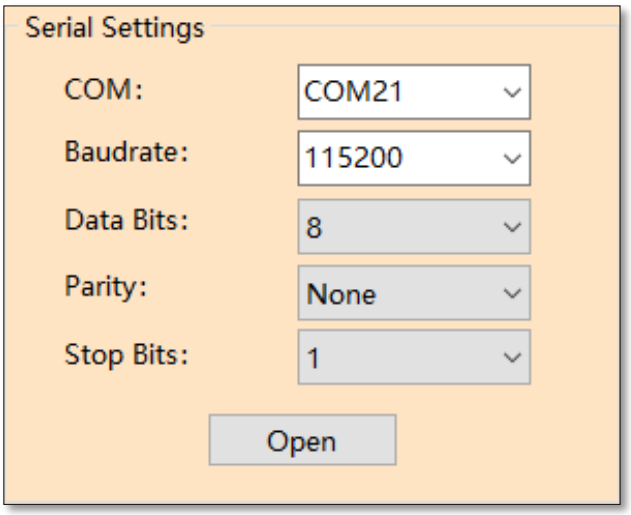
Figure 1-4. List of devices and drivers

For EVAL development board, GDLINK or JLINK debugger can be used for download. Dragging into the USB disk is not supported.

1.5. Viewing log

Connect a MicroUSB cable to the START development board, use a serial port tool on the PC, and configure it according to the parameters in *Figure 1-5. Configuration of serial port* and connect to the board. After that, use the serial port to output logs.

Figure 1-5. Configuration of serial port



2. Building development environment

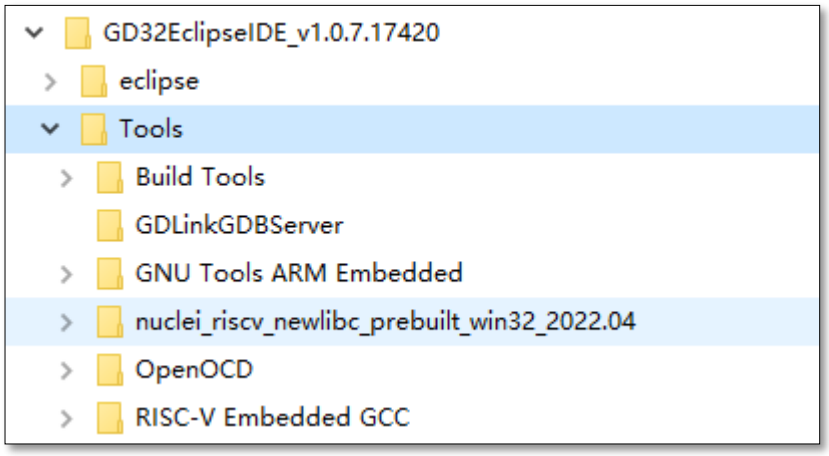
Build a development environment before compiling and burning the firmware.

The development tool currently used is GD32Eclipse IDE.

2.1. Installation of GD32Eclipse IDE

- Unzip GD32EclipseIDE_v1.0.7.17420.7z.
- Check the toolkit, whose content is as shown in ***Figure 2-1. GD32Eclise IDE toolkit***.

Figure 2-1. GD32Eclise IDE toolkit



- To install JDK, double-click eclipse/jdk/jdk-8u152-windows-x64.exe in the GD32EclipseIDE directory.
- To start the IDE, double-click eclipse/GD32EclipseIDE.exe in the GD32EclipseIDE directory.

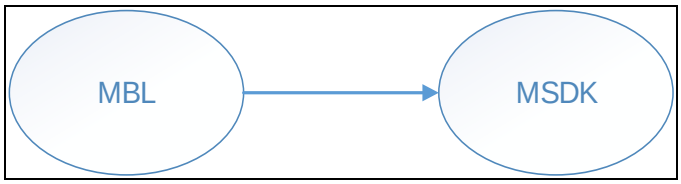
3. What developers must know

Before getting started with development, first understand the members of the SDK execution program group, how to correctly configure the SDK.

3.1. SDK execution program group

SDK will finally generate two main execution programs: MBL (Main Bootloader) and MSDK (Main SDK), which will eventually be burned to FLASH to run. After power-on, the programs will boot from Reset_Handler of MBL, and then jump to the MSDK main program to run, as shown in ***Figure 3-1. Boot process***.

Figure 3-1. Boot process



3.2. SDK configuration

3.2.1. Configuration of wireless module

The configuration file is GD32VW55x_RELEASE/config/platform_def.h, whose main content is as shown in ***Figure 3-2. Configuration of wireless module***.

Figure 3-2. Configuration of wireless module

```
#define CFG_WLAN_SUPPORT
#define CFG_BLE_SUPPORT
#if defined(CFG_WLAN_SUPPORT) && defined(CFG_BLE_SUPPORT)
  #define CFG_COEX
#endif
```

- In the case of BLE/WIFI combo mode, please enable:
  - #define CFG_WLAN_SUPPORT
  - #define CFG_BLE_SUPPORT
- In the case of BLE only, please only enable:
  - #define CFG_BLE_SUPPORT
- In the case of WIFI only, please only enable:
  - #define CFG_WLAN_SUPPORT
- To disable the wireless module, please disable all

3.2.2. SRAM layout

The configuration file is GD32VW55x_RELEASE\config\config_gdm32.h. Modify the following macro definition values to plan the SRAM space occupied by the executable program segments MBL and IMG. These values are offset addresses, and the base address is defined at the beginning of the file.

The line marked "!Keep unchanged!" cannot be modified; otherwise, the operation of the MbedTLS code in the ROM will be affected.

Figure 3-3. SRAM layout

```
/* SRAM LAYOUT */
#define RE_MBL_DATA_START       0x300           /* !Keep unchanged! */
#define RE_IMG_DATA_START       0x200           /* !Keep unchanged! */
```

For the planning of SRAM space in each executable program segment, refer to the *.ld file under the corresponding project, such as MBL\project\eclipse\mbl.ld and MSDK\plf\riscv\env_Eclipse\gd32vw55x.ld files.

3.2.3. FLASH layout

The configuration file is GD32VW55x_RELEASE\config\config_gdm32.h. Modify the following macro definition values to plan the FLASH space occupied by the executable program segments MBL and MSDK. These values are offset addresses, and the base address is defined at the beginning of the file.

The line marked "!Keep unchanged!" cannot be modified; otherwise, the operation of the project will be affected.

Figure 3-4. FLASH layout

```
/* FLASH LAYEROUT */
#define RE_VTOR_ALIGNMENT       0x200           /* !Keep unchanged! */
#define RE_SYS_SET_OFFSET       0x0             /* !Keep unchanged! */
#define RE_MBL_OFFSET           0               /* !Keep unchanged! */
#define RE_SYS_STATUS_OFFSET    0x8000          /* !Keep unchanged! */
#define RE_IMG_0_OFFSET         0xA000
#define RE_IMG_1_OFFSET         0x200000
#define RE_END_OFFSET           0x400000
```

For the planning of FLASH space in each executable program segment, refer to the *.ld file under the corresponding project, such as MBL\project\eclipse\mbl.ld and MSDK\plf\riscv\env_Eclipse\gd32vw55x.ld files.

3.2.4. Firmware version No.

The configuration file is GD32VW55x_RELEASE\config\config_gdm32.h. Modify the following

macro definition values to specify the version No. However, only RE_IMG_VERSION affects future user upgrades.

MBL only supports local upgrade, while IMG supports online upgrade. The version No. released by the SDK is consistent with RE_IMG_VERSION.

Figure 3-5. Firmware version No.

```
/* FW_VERSION */
#define RE_MBL_VERSION          0x00000600
#define RE_IMG_VERSION          0x00000600
```

3.2.5. APP configuration

The configuration file is GD32VW55x_RELEASE\MSDK\app\app_cfg.h. Choose whether to enable some applications, such as ATCMD, Alibaba Cloud, and MQTT.

3.2.6. Configuration Selection

The SDK supports two configurations, one is msdk and other is msdk_ffd. By default, msdk is used. Compared to msdk, msdk_ffd consumes more resources and supports more functions, which is achieved by linking different libs. msdk links libwpas and libble, msdk_ffd links libwpa_supplicant and libble_max. In addition, msdk_ffd enables macro CONFIG_WPA_SUPPLICANT by default, and this macro is built into the msdk_ffd configuration.

Usually, msdk can meet the functional requirements. If there is an additional need such as WFA certification, msdk_ffd can used.

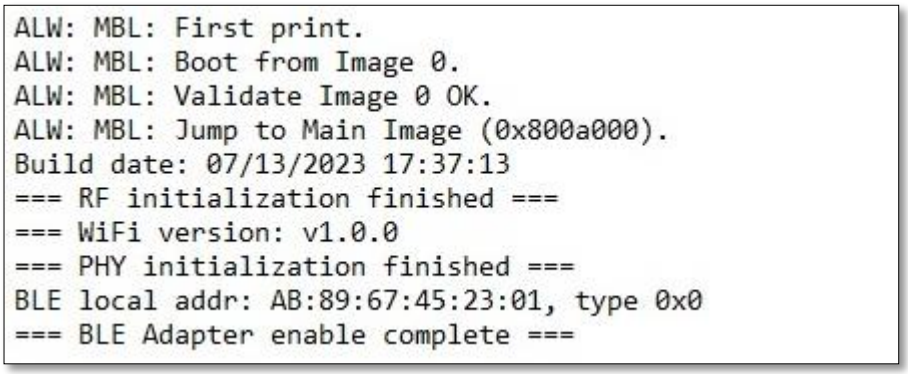
For details on how to make configuration selection for actual use, see the Compiling MSDK Projects section in subsection ***Compilation***.

BLE supports two forms, ble(corresponding to libble) and ble_max(corresponding to libble_max, which supports more functions, refer to the feature macro definition located in the relevant header files in blesw/src/export config and config_max). The lib and header files are used together, with the default msdk using ble. The main difference is that the linked lib and the included header files are different, and the project configuration can be selected flexibly.

3.3. Correct log example

After the firmware group (MBL+MSDK) is successfully downloaded, open the serial port tool, and press the Reset button on the development board. The startup information is shown in ***Figure 3-6. Project boot information***. If an exception occurs, please check ***FAQ*** for help.

Figure 3-6. Project boot information



4. GD32Eclipse IDE project

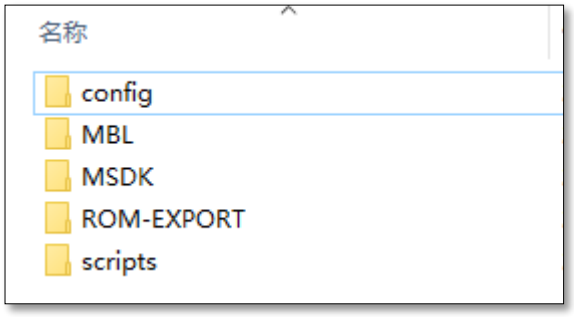
This chapter introduces how to compile and debug the SDK under GD32Eclipse IDE.

The project group consists of two projects: MBL/MSDK. MSDK includes Wi-Fi protocol stack, BLE protocol stack, peripheral drivers, applications, etc. MBL is mainly responsible for selecting the correct firmware to run from two MSDK firmware (one is the current firmware and the other is the upgraded one) to run.

4.1. Opening the project group

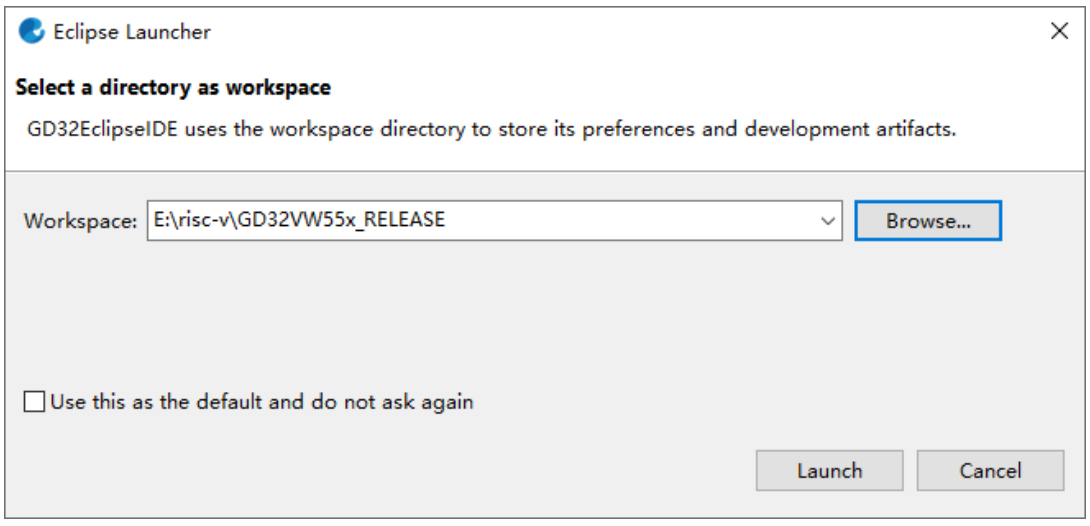
- Check the SDK directory GD32VW55x_RELEASE, as shown in ***Figure 4-1. SDK directory***.

Figure 4-1. SDK directory



- To start the IDE, double-click eclipse/GD32EclipseIDE.exe in the GD32EclipseIDE directory, then select the SDK directory GD32VW55x_RELEASE as the workspace, and click the Launch button, as shown in ***Figure 4-2. Starting GD32EclipseIDE***.

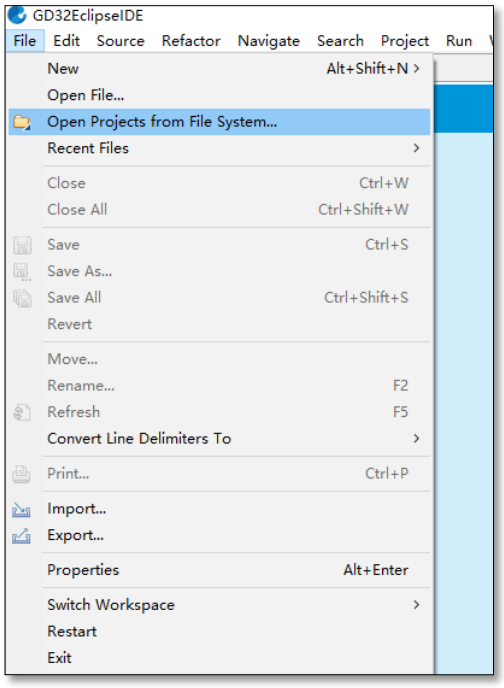
Figure 4-2. Starting GD32EclipseIDE



- Import the MBL project

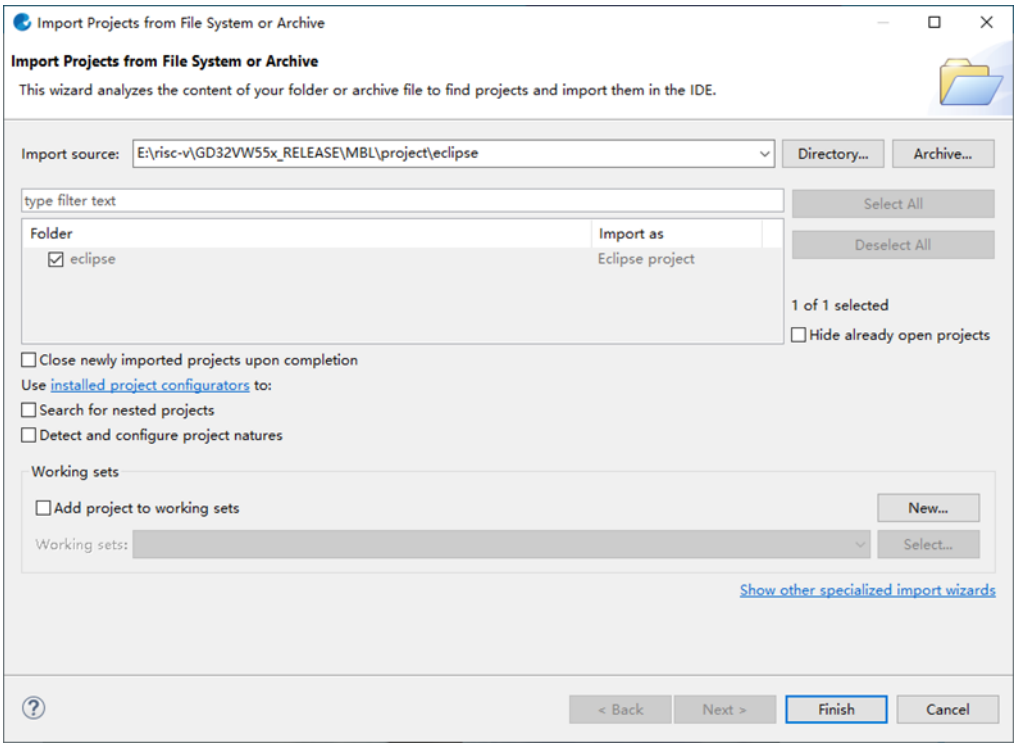
In the File menu, click Open Projects from file System, as shown in ***Figure 4-3. Open Projects from file System***.

Figure 4-3. Open Projects from file System



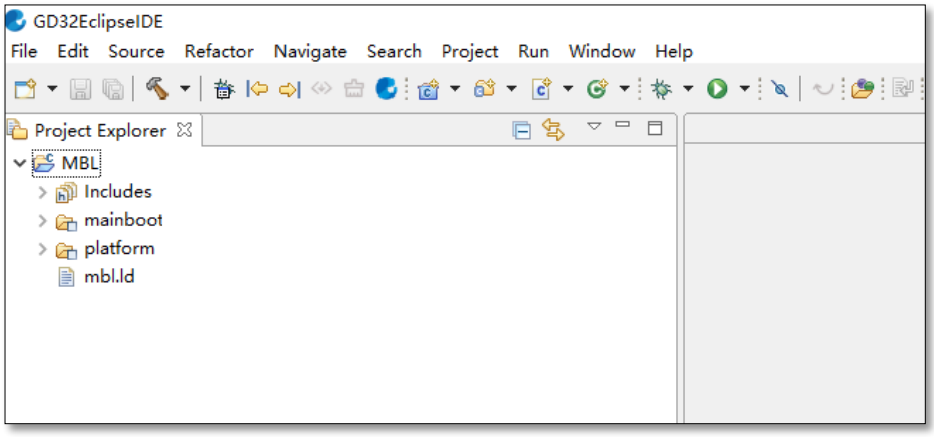
Select the project path GD32VW55x_RELEASE\MBL\project\eclipse, as shown in ***Figure 4-4. Selecting MBL project path***, and click Finish.

Figure 4-4. Selecting MBL project path



Close the welcome interface, and the MBL project is shown in ***Figure 4-5. MBL project interface***.

Figure 4-5. MBL project interface

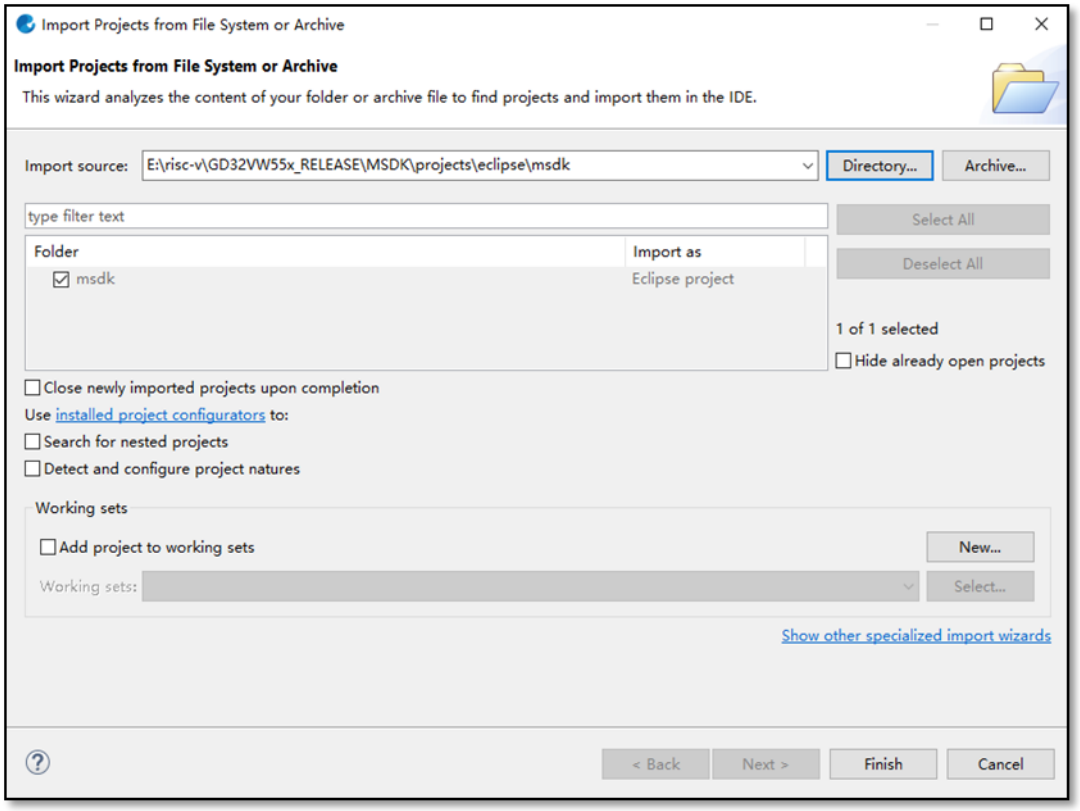


- Import the MSDK project

In the File menu, click Open Projects from file System, as shown in ***Figure 4-3. Open Projects from file System***.

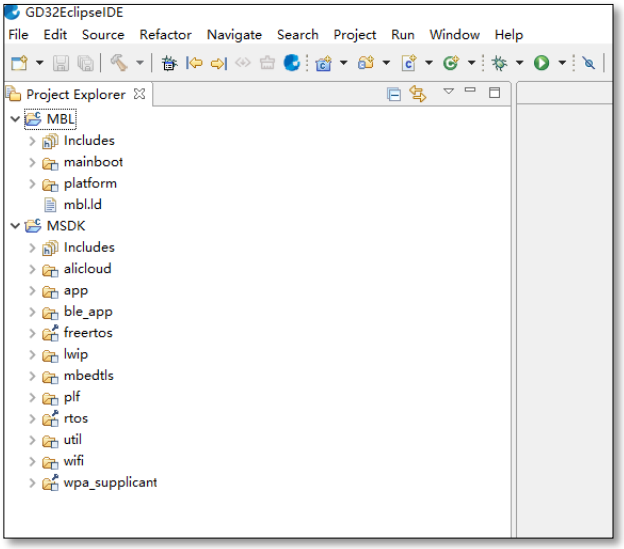
Select the project path GD32VW55x_RELEASE\MSDK\projects\eclipse\msdk, as shown in ***Figure 4-6. Selecting MSDK project path***, and click Finish.

Figure 4-6. Selecting MSDK project path



View the MSDK and MBL project interfaces, as shown in ***Figure 4-7. MSDK and MBL project interfaces***.

Figure 4-7. MSDK and MBL project interfaces

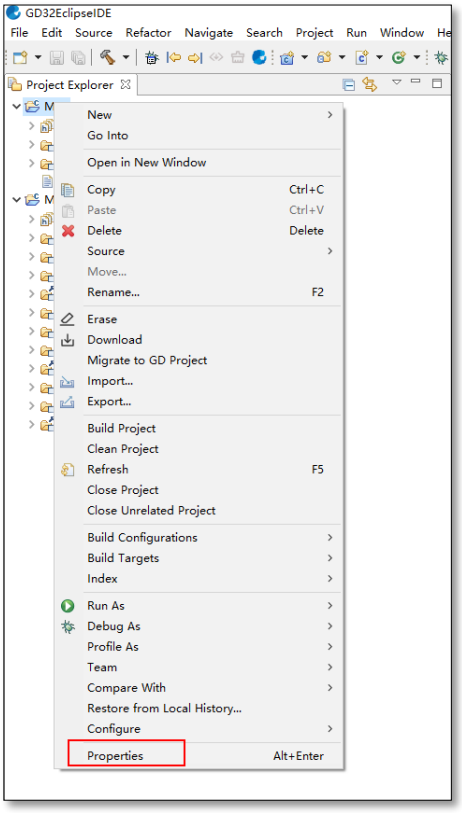


4.2. Compilation

- Check the configuration of the project compilation tool

Right-click the project, and click Properties, as shown in ***Figure 4-8. Opening Properties of the project***.

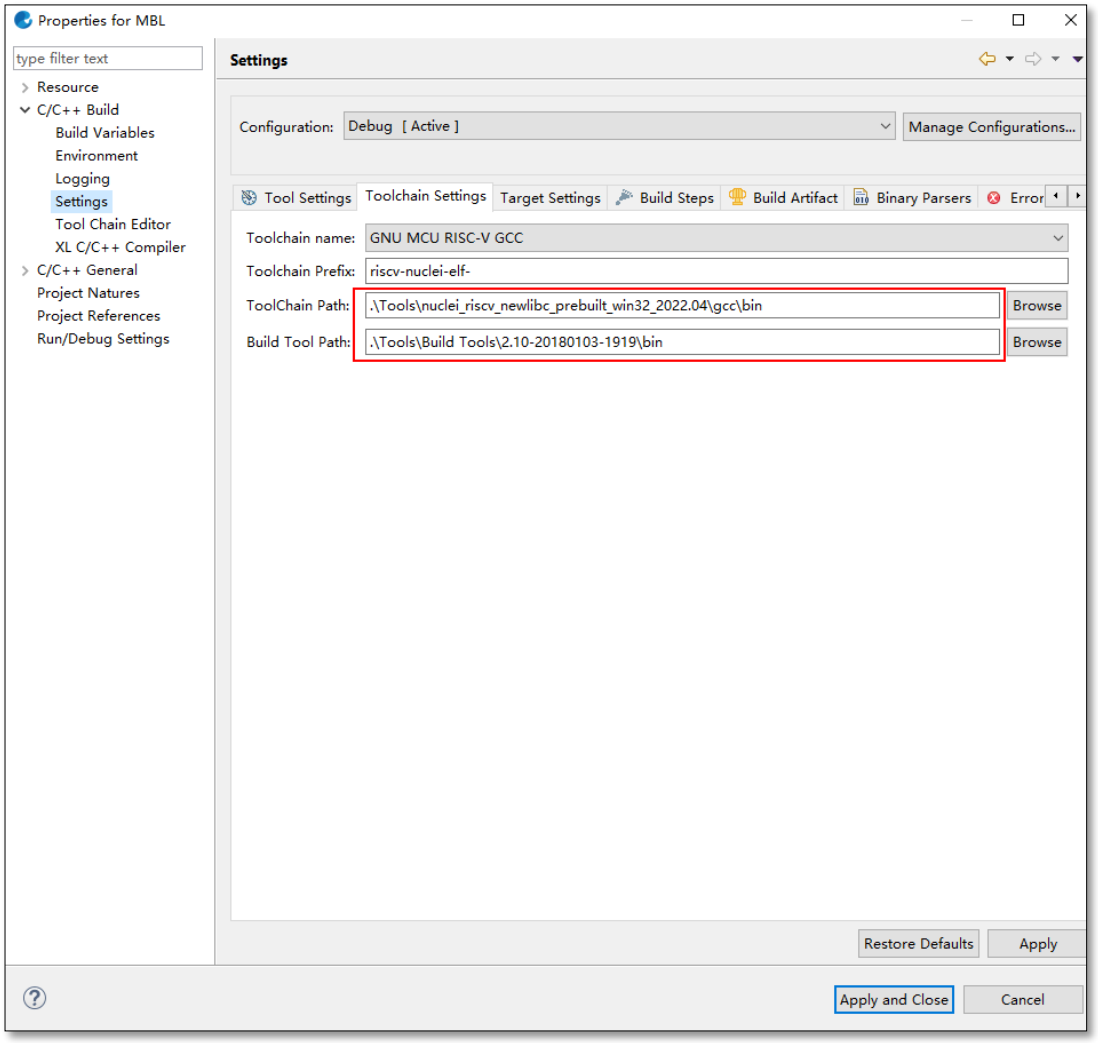
Figure 4-8. Opening Properties of the project



Select C/C++ Build→Settings, and click the Toolchain Settings tab, as shown in ***Figure 4-9. Toolchain Settings***.

The default path is the tools in the IDE directory and does not need to be changed. Click Apply and Close.

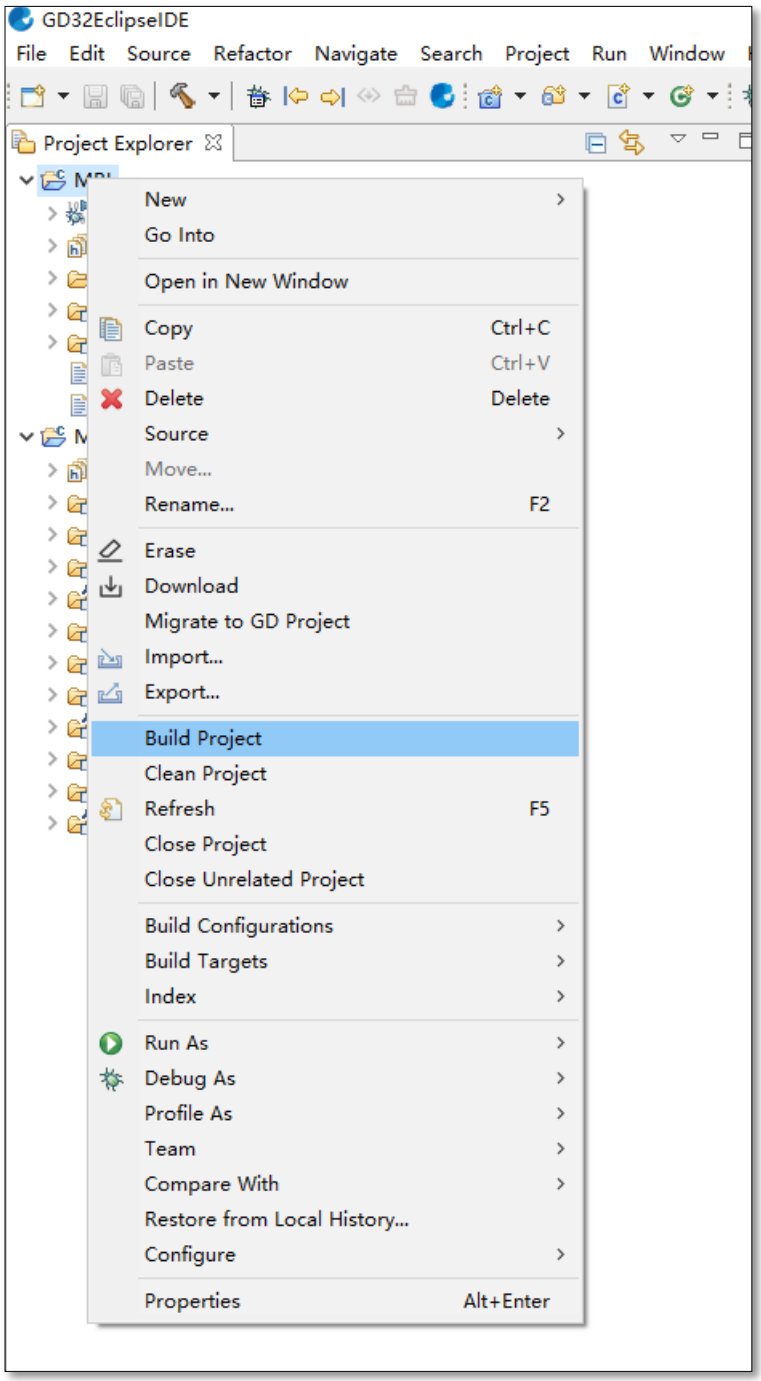
Figure 4-9. Toolchain Settings



- Compile the MBL project

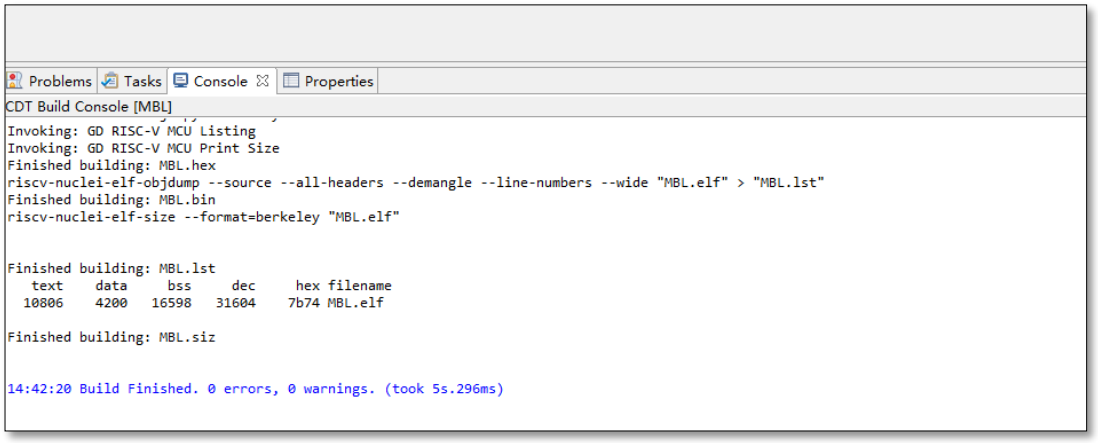
Right-click the project, and click Build Project, as shown in ***Figure 4-10. Compiling the MBL project***.

Figure 4-10. Compiling the MBL project



The compilation result is as shown in ***Figure 4-11. MBL compilation result***.

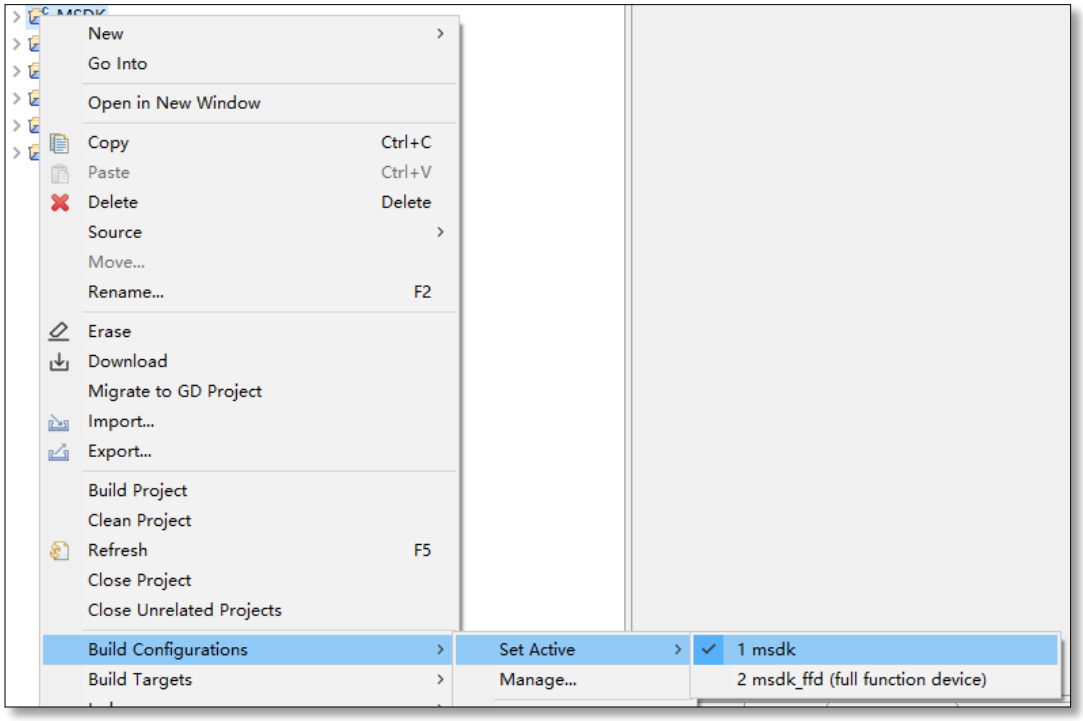
Figure 4-11. MBL compilation result



- Compile the MSDK project

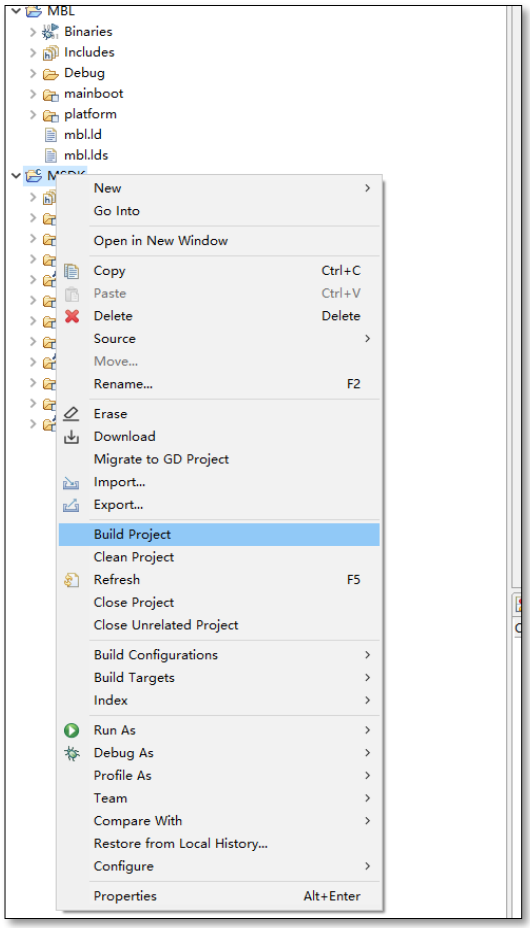
Right-click the project, and click Build Configurations—>Set Active—><target project> in order. as shown in ***Figure 4-12. target project selection***, the default target project is msdk.

Figure 4-12. target project selection



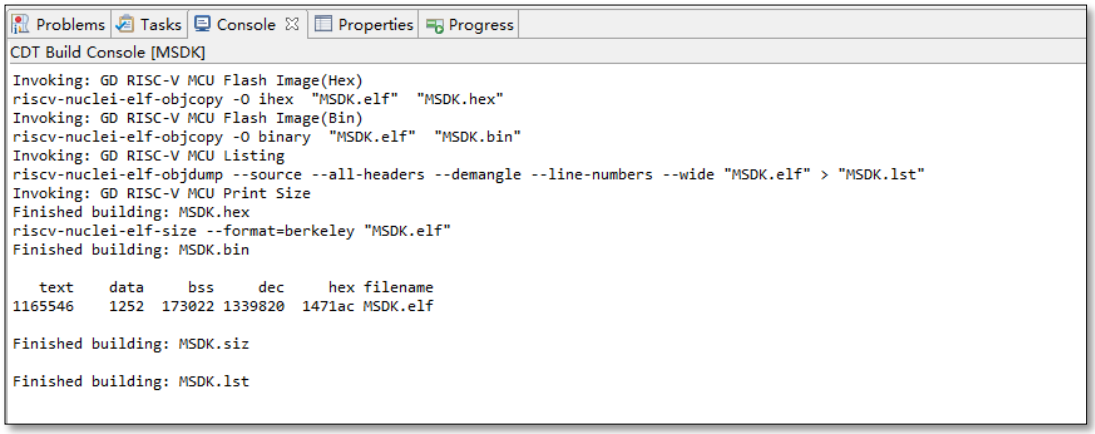
Right-click the project again, and click Build Project, as shown in ***Figure 4-13. Compiling the MSDK project***.

Figure 4-13. Compiling the MSDK project



The compilation result is as shown in ***Figure 4-14. MSDK compilation result***.

Figure 4-14. MSDK compilation result



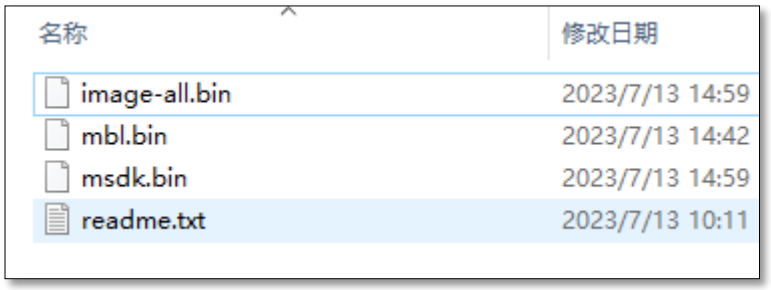
- Images generated by SDK

After MSDK compilation is completed, images are output in the GD32VW55x_RELEASE\scripts\images path, as shown in ***Figure 4-15. Images output***.

Image-all.bin contains executable program segments MBL and MSDK. The firmware can be

used for production and burned into blank FLASH.

Figure 4-15. Images output



4.3. Download

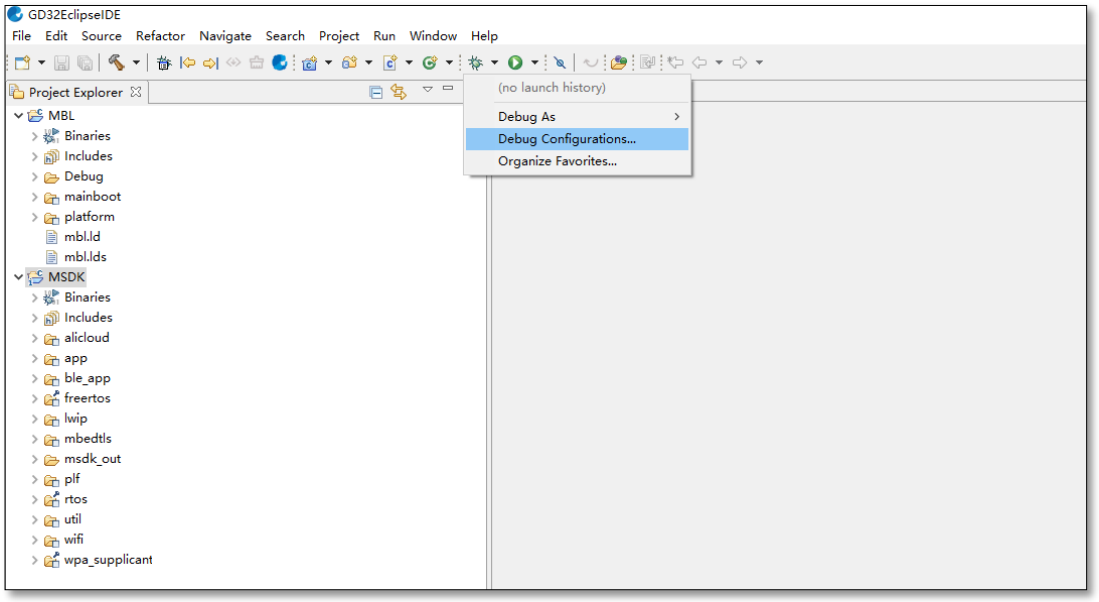
Refer to ***1.4 Download interface***, drag and drop image-all.bin to burn.

4.4. Debugging

- Debugging configuration

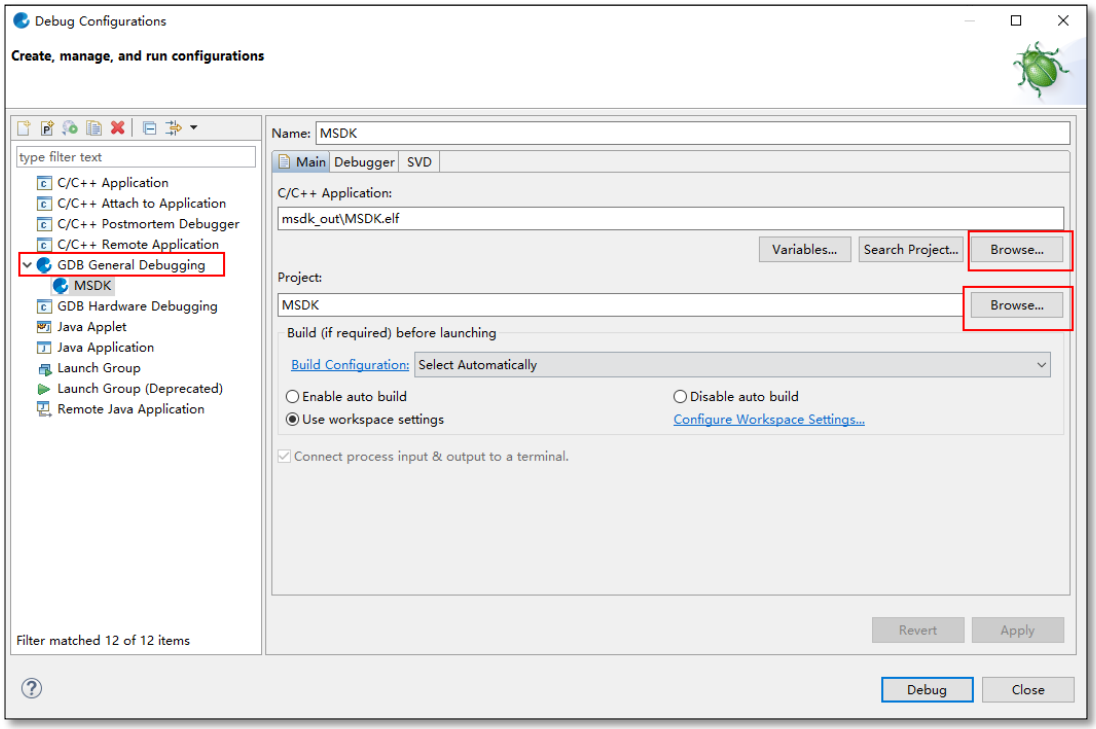
First select the MSDK project, and then click Debug Configuration in the debug drop-down list, as shown in ***Figure 4-16. Opening the Debug Configuration option***.

Figure 4-16. Opening the Debug Configuration option



Double-click GDB General Debugging, and enter MSDK in Name. Click Browse next to C/C++ Application to select msdk_out\MSDK.elf, and click Browse next to Project to select MSDK, as shown in ***Figure 4-17. MSDK debug configuration***, and then click Close.

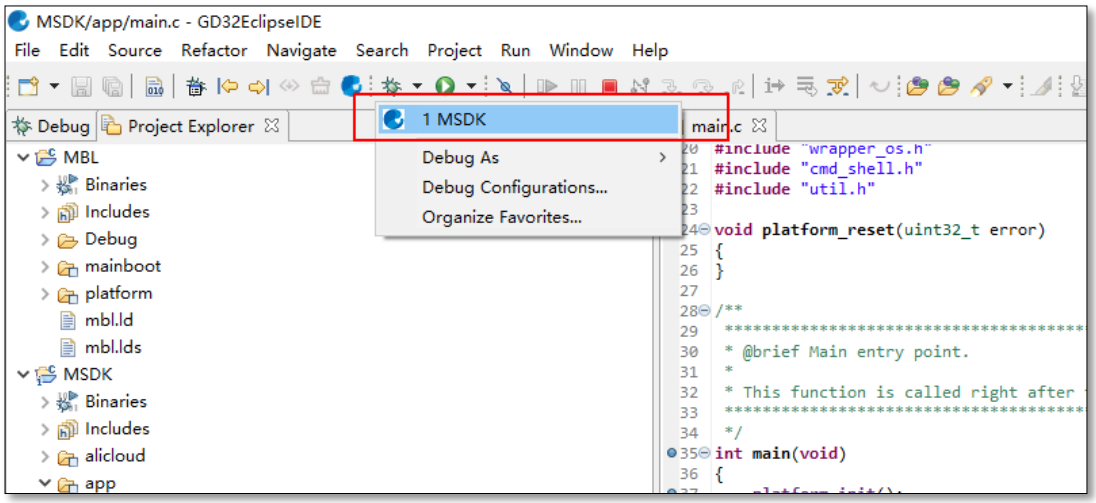
Figure 4-17. MSDK debug configuration



- Start debugging

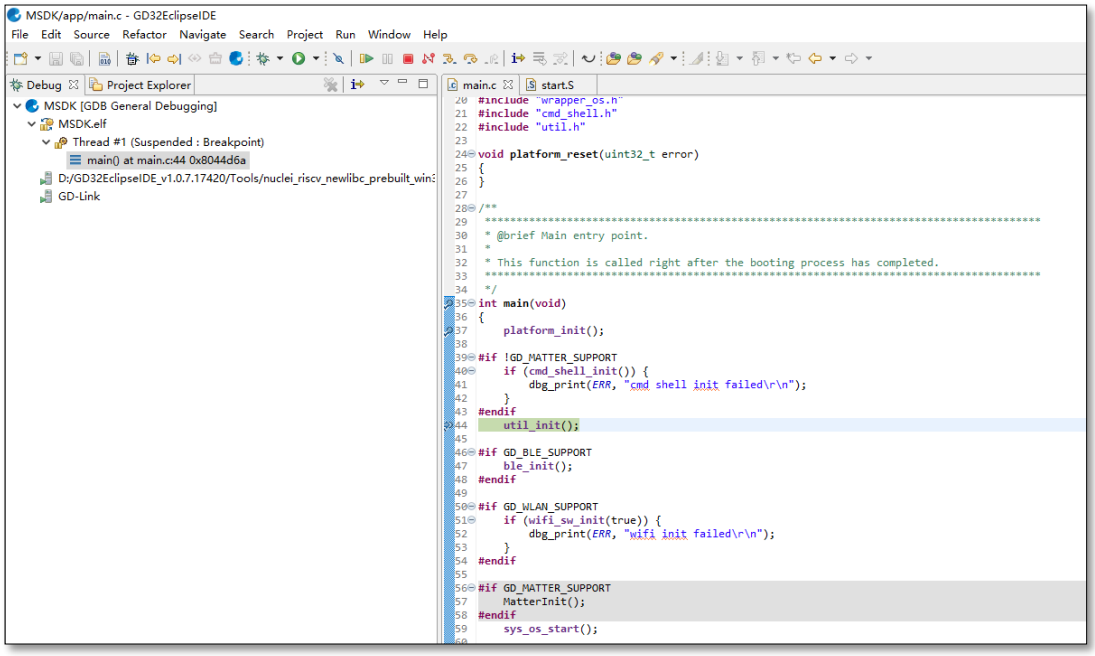
Click the target MSDK in the debug drop-down list to start debugging.

Figure 4-18. Starting MSDK debug



After the debug session is established, click Debug > Thread under the Browse tab to start debugging, as shown in ***Figure 4-19. MSDK debug interface***.

Figure 4-19. MSDK debug interface



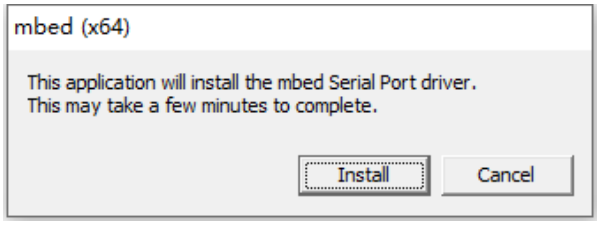
5. FAQ

5.1. DAPLINK disk recognition

When the START development board is used, if the serial port in the Device Manager cannot be recognized as "mbedSerialPort (COMx)", such as Windows 7 and below, the Mbed serial port driver needs to be installed.

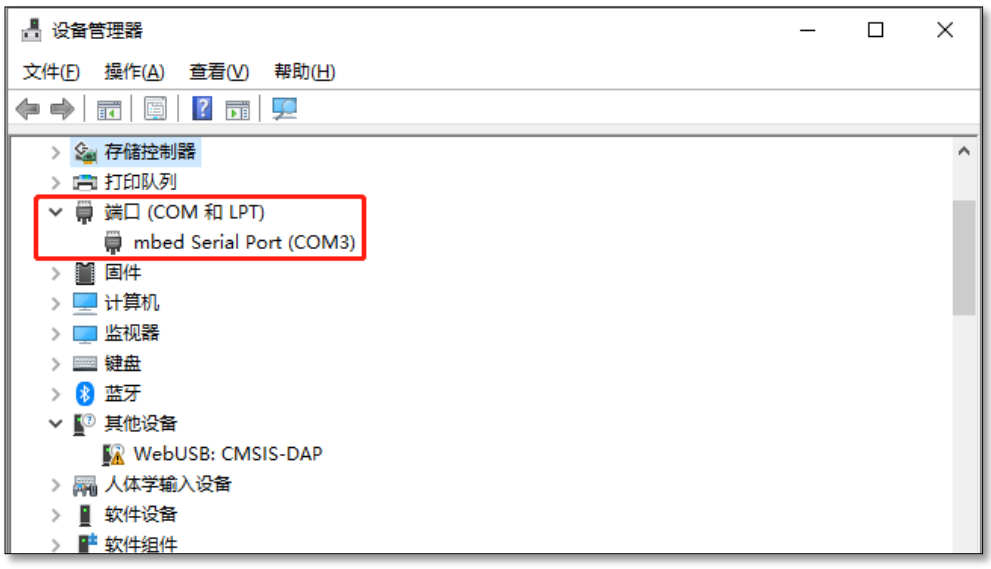
- Download the driver
  - *https://os.mbed.com/media/downloads/drivers/mbedWinSerial_16466.exe*
- Install the driver
  - Double-click to run mbedWinSerial_16466.exe, and click Install, as shown in *Figure 5-1. Installation of Mbed serial port driver*. After successful installation, click Finish.

Figure 5-1. Installation of Mbed serial port driver



- Check
  - After successful installation, the serial port "mbed Serial Port (COMx)" is displayed in the Device Manager, as shown in *Figure 5-2. List of serial ports in Device Manager*, and the list of devices and drivers shows the DAPLINK disk, as shown in *Figure 1-4. List of devices and drivers*.

Figure 5-2. List of serial ports in Device Manager



5.2. No image error

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

Reason: An error occurs during the previous boot of WIFI_IOT, and the MBL records operation exception of the IMAGE. If another IMAGE is not burned or also has a boot exception, this message will be printed. In other words, the MBL believes that there is no valid IMAGE to jump to, and the boot fails.

Solution: Burn the MBL again. After that, the IMAGE status will be cleared.

5.3. Code running in SRAM

To run programs faster to achieve higher performance, move them to the SRAM.

Open GD32VW55x_RELEASE\MSDK\plf\riscv\env_Eclipse\gd32vw55x.ld, and find the line ".code_to_sram:". The code in the braces runs in the SRAM. To add new content, add it at the end of the code. Refer to existing files for the format, for exemple:

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

It is to put the entire port.c file in the SRAM and run it. For example:

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

It is to put the xTaskIncrementTick () function in tasks.c in the SRAM and run it.

6. Revision history

Table 6-1. Revision history

Revision No.	Description	Date
1.0	Initial release	Dec.5, 2023

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries （the "Company"). This document, including any product of the Company described in this document （the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto （if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which have been expressly identified in the applicable agreement, the Products are designed, developed, and / or manufactured for ordinary business, industrial, personal, and / or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems （including resuscitation equipment and surgical implants）, pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage （"Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as its suppliers and / or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as its suppliers and / or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.