

GigaDevice Semiconductor Inc.

GD32VW553 Quick Development Guide

Application Note

AN154

Revision 1.3

(Apr 2025)

Table of Contents

Table of Contents.....	2
List of Figures	4
List of Tables	6
1. Introduction to development board	7
1.1. Picture of real development board	7
1.1.1. The START development board	7
1.1.2. The EVAL development board	8
1.2. Boot mode	9
1.3. Debugger interface.....	9
1.4. Download interface	10
1.5. Viewing log.....	10
2. Building development environment.....	11
2.1. Installation of GD32 Embedded Builder	11
2.2. Installation of SEGGER Embedded Studio IDE	11
2.3. Toolchain download.....	11
3. What developers must know.....	13
3.1. SDK execution program group	13
3.2. SDK configuration.....	13
3.2.1. Configuration of wireless module	13
3.2.2. SRAM layout	14
3.2.3. FLASH layout	14
3.2.4. Firmware version No.....	14
3.2.5. APP configuration	15
3.2.6. Configuration Selection.....	15
3.3. Correct log example.....	16
4. GD32 Embedded Builder IDE project.....	17
4.1. Opening the project group.....	17
4.2. Compilation	20
4.3. Download firmware	23
4.4. Debugging.....	23
5. SEGGER Embedded Studio IDE project.....	26

5.1.	Open projects.....	26
5.2.	Compilation	27
5.3.	Download firmware	29
5.4.	Debugging.....	30
6.	FAQ.....	32
6.1.	No image error	32
6.2.	Code running in SRAM	32
7.	Revision history	33

List of Figures

Figure 1-1. The picture of the START development board	7
Figure 1-2. The picture of the EVAL development board.....	8
Figure 1-3. Development Board Type Configuration.....	9
Figure 1-4. List of devices and drivers	10
Figure 1-5. Configuration of serial port.....	10
Figure 2-1 The Directory Structure of GD32 Embedded Builder	11
Figure 2-2 Download Toolchain	12
Figure 2-3 Nuclei Toolchain Catalog Structure	12
Figure 3-1. Boot process	13
Figure 3-2. Configuration of wireless module	13
Figure 3-3. SRAM layout.....	14
Figure 3-4. FLASH layout.....	14
Figure 3-5. Firmware version No.....	15
Figure 3-6. Project boot information	16
Figure 4-1. SDK directory.....	17
Figure 4-2. Starting GD32 Embedded Builder IDE.....	17
Figure 4-3. Open Projects from file System	18
Figure 4-4. Selecting MBL project path	18
Figure 4-5. MBL project interface.....	19
Figure 4-6. Selecting MSDK project path.....	19
Figure 4-7. MSDK and MBL project interfaces	20
Figure 4-8. Properties of the project	20
Figure 4-9. Compiling the MBL project.....	21
Figure 4-10. MBL compilation result	21
Figure 4-11. target configuration selection.....	22
Figure 4-12. MSDK compilation result.....	22
Figure 4-13. Images output	23
Figure 4-14. Opening the Debug Configuration option	23
Figure 4-15. MSDK debug configuration	24
Figure 4-16. MSDK Debugging Configuration Interface with openocd	25
Figure 4-17. MSDK debug interface	25
Figure 5-1. MBL SES Project Project Interface	26
Figure 5-2. MSDK SES Project Interface	27
Figure 5-3. nuclei toolchain content.....	27
Figure 5-4. Compiling the MBL project.....	28
Figure 5-5. MBL compilation result.....	28
Figure 5-6. Compile MSDK project.....	28
Figure 5-7 MSDK Project Configuration Options.....	29
Figure 5-8 MSDK compilation result	29

Figure 5-9. Images output.....	29
Fogure 5-10 SES IDE image download.....	30
Figure 5-11. MSDK SES Project Configuration Interface.....	31
Figure 5-12. SES IDE Debug Interface.....	31

List of Tables

Table 1-1. Boot mode.....	9
Table 7-1. Revision history.....	33

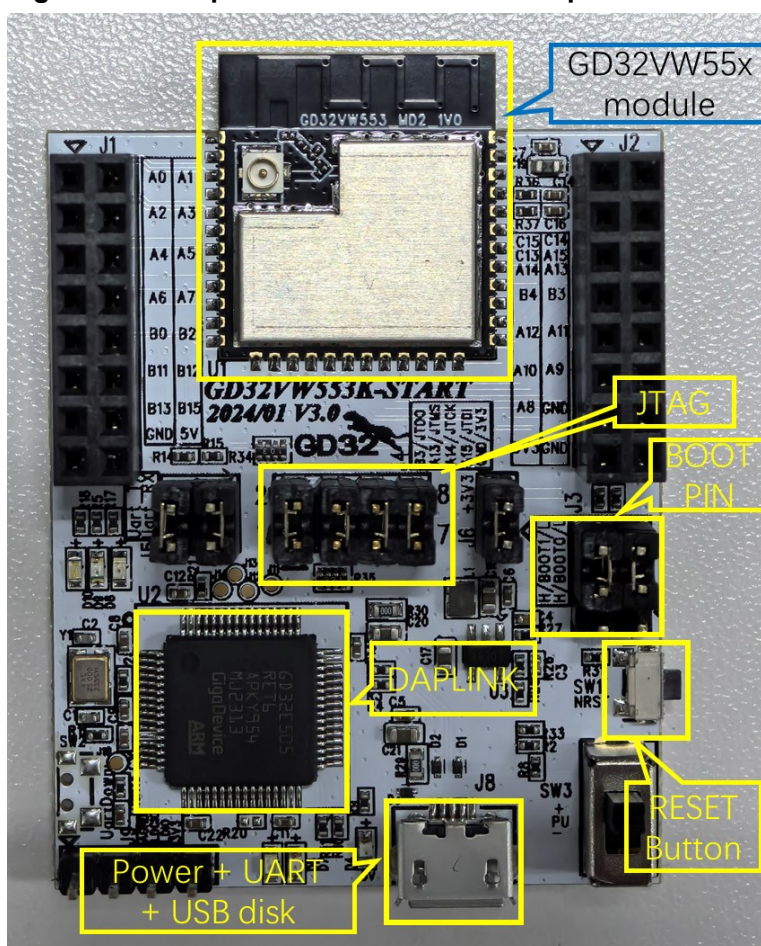
1. Introduction to development board

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



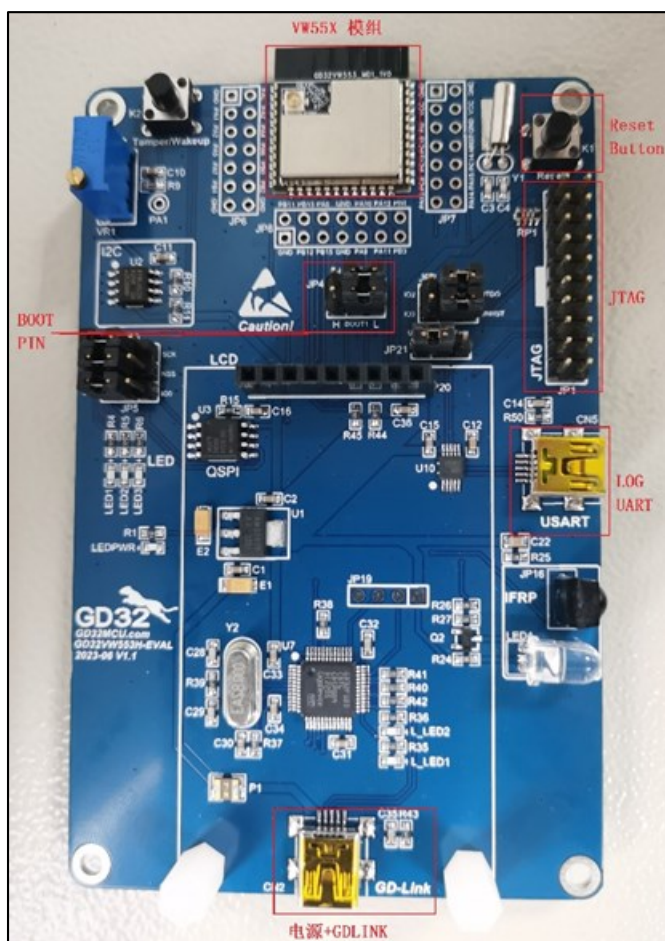
Mainly focus on the following parts of the development board, which have been marked in the [Figure 1-1. The picture of the START development board](#).

- 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 lead out 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 1-2. The picture of the EVAL development board](#).

- 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 in [Figure 1-3. Development Board Type Configuration](#), 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
#define PLATFORM_BOARD_32VW55X_F527 2
#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

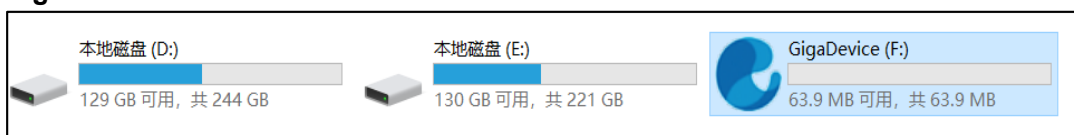
For START development board, it comes with a DAPLINK(GD32F505) debugger that can be used with OpenOCD. Can also use an external debugger (GD-Link or J-Link) at the JTAG interface of the board for debugging and download. The DAP chip also integrates the UART function, so only one USB cable is required to supply power, debug, and view the log. Connect the pins JCLK, JTWS, JTDO and JTDI to the lower four 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, GD-Link or J-Link 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 GigaDevice 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 GigaDevice disk to complete FLASH downloading of the GD32VW55x chip.

Figure 1-4. List of devices and drivers

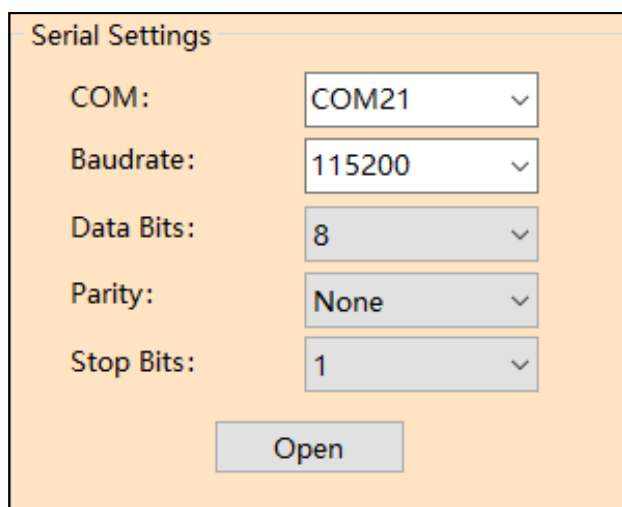


For EVAL development board, GDLINK or JLINK debugger can be used for download. Dragging into the USB disk is not supported.

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

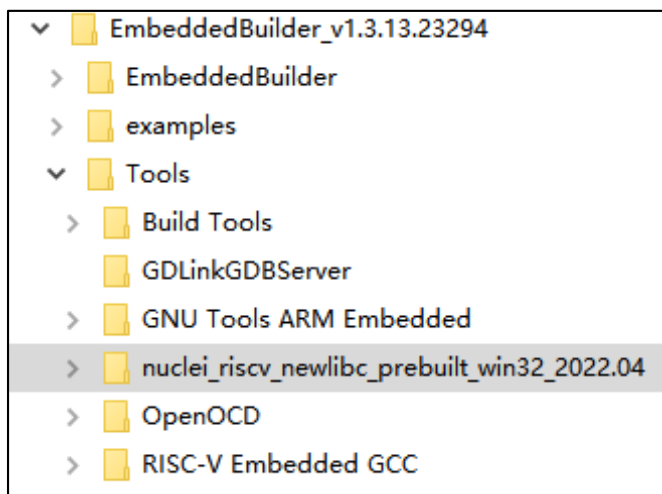
Build a development environment before compiling and downloading the firmware.

The development tool currently used is GD32 Embedded Builder and SEGGER Embedded Studio IDE.

2.1. Installation of GD32 Embedded Builder

- The GD32 Embedded Builder can select GD32VW5 at website: <https://gd32mcu.com/cn/download> to download. The uncompressed downloaded files is as [Figure 2-1 The Directory Structure of GD32 Embedded Builder](#) shows. It is also recommended to unzip the “nuclei_riscv_newlibc_prebuilt_win32_2022.04” download in subsection [2.3Toolchain download](#) and place it in the Tools directory as well.

Figure 2-1 The Directory Structure of GD32 Embedded Builder



2.2. Installation of SEGGER Embedded Studio IDE

Please visit the website: <https://wiki.segger.com/GD32V> for how to get the SEGGER Embedded Studio IDE and License Activation Key

2.3. Toolchain download

- download

Official website: <https://nucleisys.com/download.php>

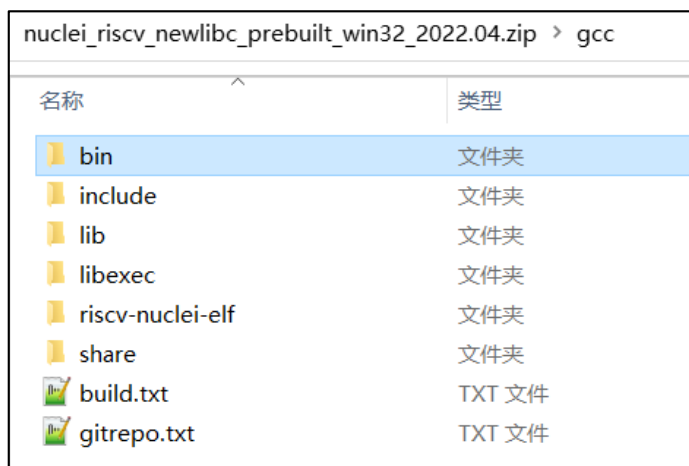
As shown in [Figure 2-2 Download Toolchain](#), ToolChain selects version 2022.04.

The directory structure of the downloaded Nuclei RISC-V Toolchain after unpacking is shown in [Figure 2-3 Nuclei Toolchain Catalog Structure](#)

Figure 2-2 Download Toolchain



Figure 2-3 Nuclei Toolchain Catalog Structure



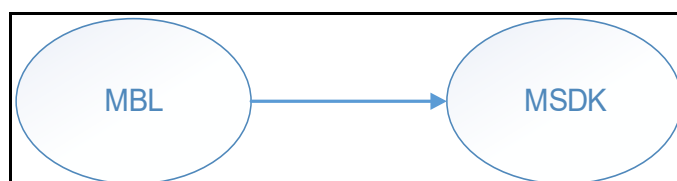
3. What developers must know

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 downloaded 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 (as [Figure 3-3. SRAM layout](#) shows) 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\p\frisc\env\gd32w55x.ld.

3.2.3. FLASH layout

The configuration file is GD32VW55x_RELEASE\config\config_gdm32.h. Modify the following macro definition (as [Figure 3-4. FLASH layout](#) shows) 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 LAYOUT */
#define RE_VTOR_ALIGNMENT ..... 0x200 ..... /* !!Keep unchanged! */
#define RE_SYS_SET_OFFSET ..... 0x0 ..... /* !!Keep unchanged! */
#define RE_MBL_OFFSET ..... 0x0 ..... /* 0x0: Boot from MBL, 0x1000: Boot from ROM */
#define RE_SYS_STATUS_OFFSET ..... 0x8000 ..... /* !!Keep unchanged! */
#define RE_IMG_0_OFFSET ..... 0xA000 ..... /* !!Keep unchanged! */
#define RE_IMG_1_OFFSET ..... 0x1E000
#define RE_IMG_1_END ..... 0x3CB000 ..... /* reserved 192KB for user data */
#define RE_NVDS_DATA_OFFSET ..... 0x3FB000 ..... /* reserved 20KB for nvds data */
#define RE_END_OFFSET ..... 0x400000 ..... /* equal to flash total size */
```

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\p\frisc\env\gd32w55x.ld.

3.2.4. Firmware version No.

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

macro definition values showed in [Figure 3-5. Firmware version No.](#) to specify the version No. In addition, the macro RE_IMG_VERSION is used in Securt Boot to determine the firmware version.

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 .....0x01000002
#define RE_IMG_VERSION .....0x01000002
```

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, MQTT, COAP and so on.

3.2.6. Configuration Selection

- msdk
FreeRTOS + Libwpas.a + libble.a + MbedTLS 3.6.2
- msdk_ffd
FreeRTOS + wpa_supplicant.a + libble_max.a + MbedTLS 3.6.2
- msdk_mbedtls_2.17.0
FreeRTOS + Libwpas.a + libble.a + ROM MbedTLS 2.17.0
- msdk_rttthread
RTThread + Libwpas.a + libble.a + MbedTLS 3.6.2
- msdk_threadx
Threadx + Libwpas.a + libble.a + MbedTLS 3.6.2

As described above, MSDK supports multiple configurations: msdk (default), msdk_ffd, msdk_threadx, msdk_mbedtls_2.17.0, and msdk_rttthread.

Both libwpas.a and wpa_supplicant are Wi-Fi-related libraries, where the latter can be used for WFA certification.

Both libble and libble_max are BLE libraries, with libble_max supporting more features. By default, msdk links to libble, but the configuration can be modified to link to libble_max.

Compared to MbedTLS 3.6.2, the ROM version of MbedTLS 2.17.0 is hardcoded in ROM, saving Flash resources.

For details on how to make configuration selection for actual use, see the Compiling MSDK Projects section in subsection [4.2Compilation](#).

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 [6FAQ](#) for help.

Figure 3-6. Project boot information

```
#ALW: MBL: First print.
ALW: MBL: Boot from Image 0.
ALW: MBL: Validate Image 0 OK.
ALW: MBL: Jump to Main Image (0x0800a000).
Chip: GD32VW55x
=== RF initialization finished ===
SDK Version: v1.0.3-f0e77956a91696b2
Build date: 2025/04/07 14:43:40
=== WiFi calibration done ===
=== PHY initialization finished ===
BLE local addr: 76:BA:ED:23:00:D5, type 0x0
=== BLE Adapter enable complete ===
```


4. GD32 Embedded Builder IDE project

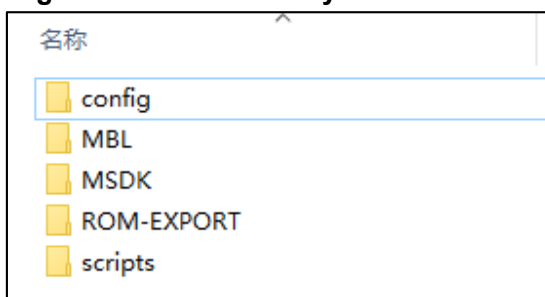
This chapter introduces how to compile and debug the SDK under GD32 Embedded Builder IDE.

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

4.1. Opening the project group

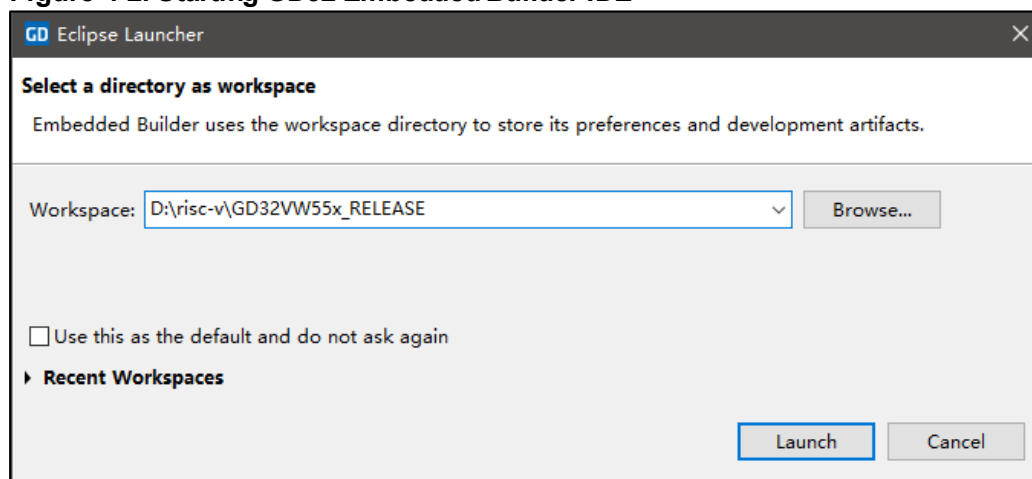
- Check the SDK directory GD32VW55x_RELEASE, as shown in [Figure 4-1. SDK directory](#).

Figure 4-1. SDK directory



- To start the IDE, double-click Embedded Builder.exe in the Embedded Builder directory, and select the SDK directory GD32VW55x_RELEASE as the workspace, and then click the launch button, as shown in [Figure 4-2. Starting GD32 Embedded Builder IDE](#).

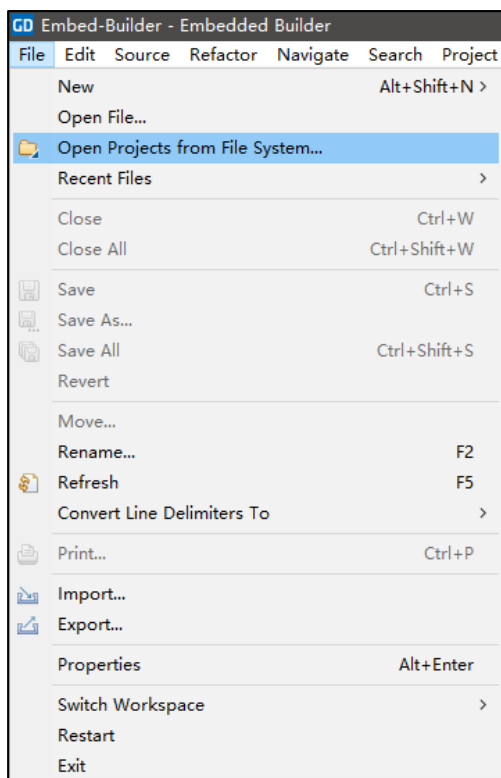
Figure 4-2. Starting GD32 Embedded Builder IDE



- Import the MBL project

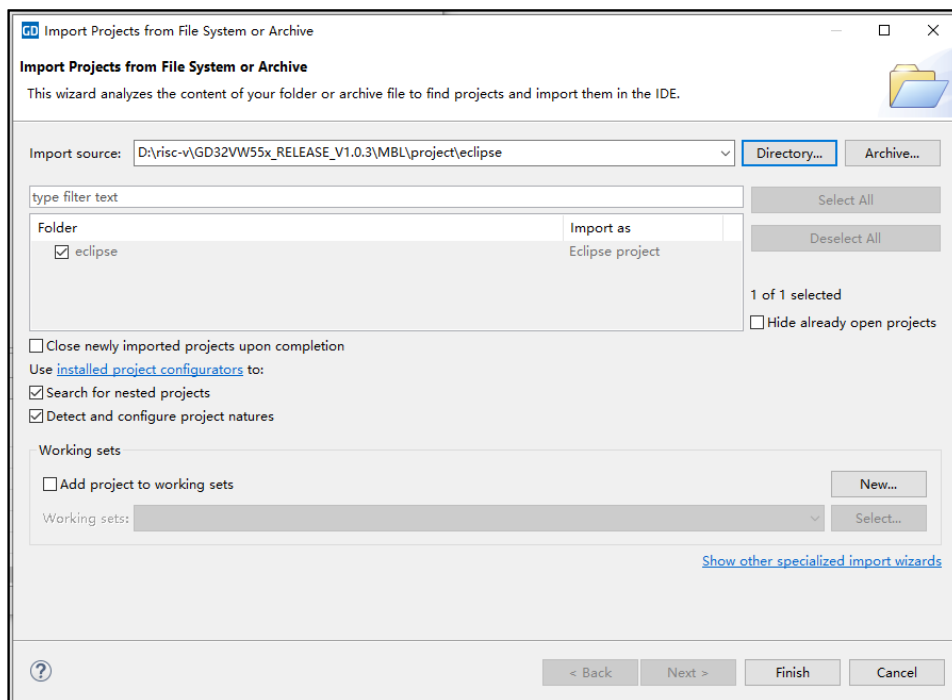
In the File menu, click Open Projects from file System, as shown in [Figure 4-3. Open Projects from file System](#).

Figure 4-3. Open Projects from file System



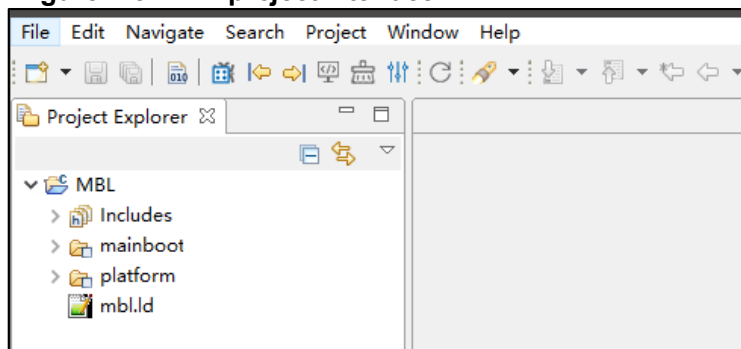
Select the project path `GD32VW55x_RELEASE\MBL\project\eclipse`, as shown in [Figure 4-4. Selecting MBL project path](#), and click Finish.

Figure 4-4. Selecting MBL project path



Close the welcome interface, and the MBL project is shown as [Figure 4-5. MBL project interface](#) shows.

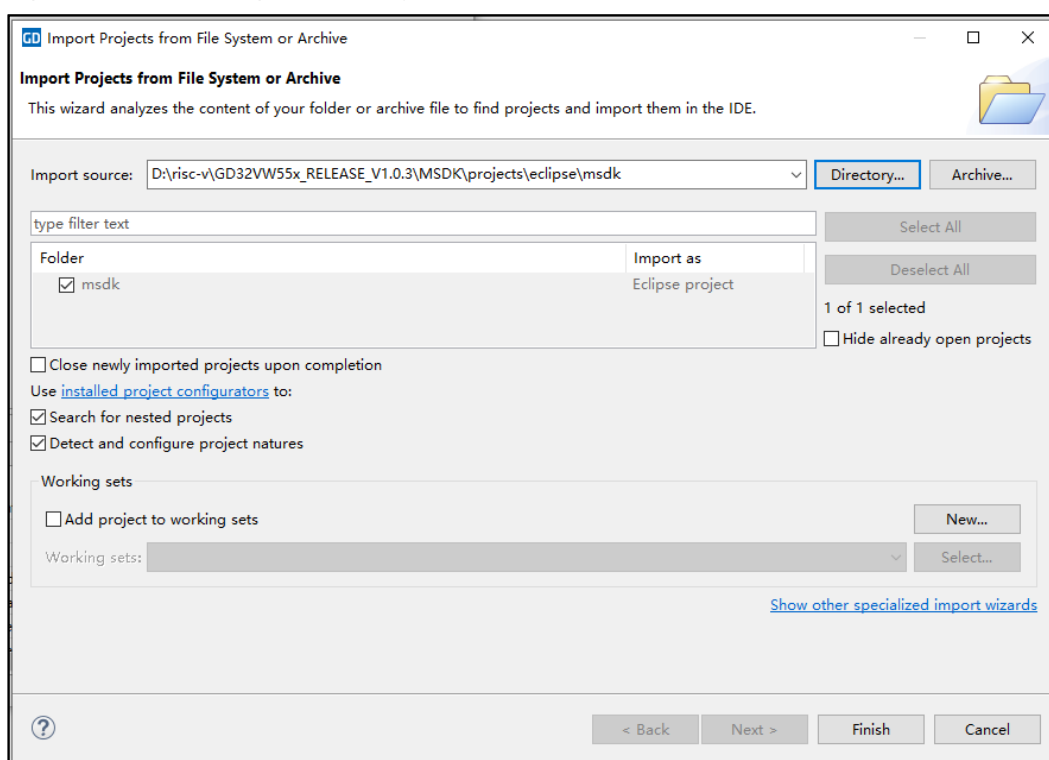
Figure 4-5. MBL project interface



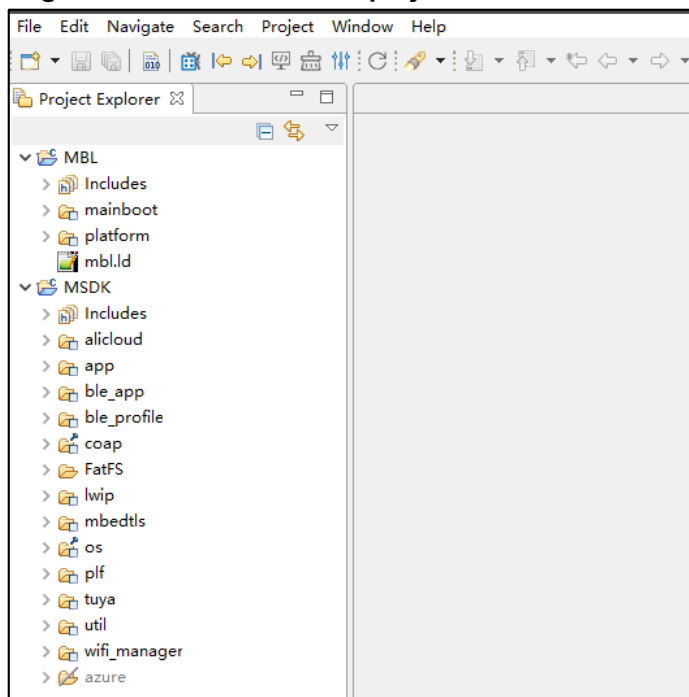
■ Import the MSDK project

In the File menu, click 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



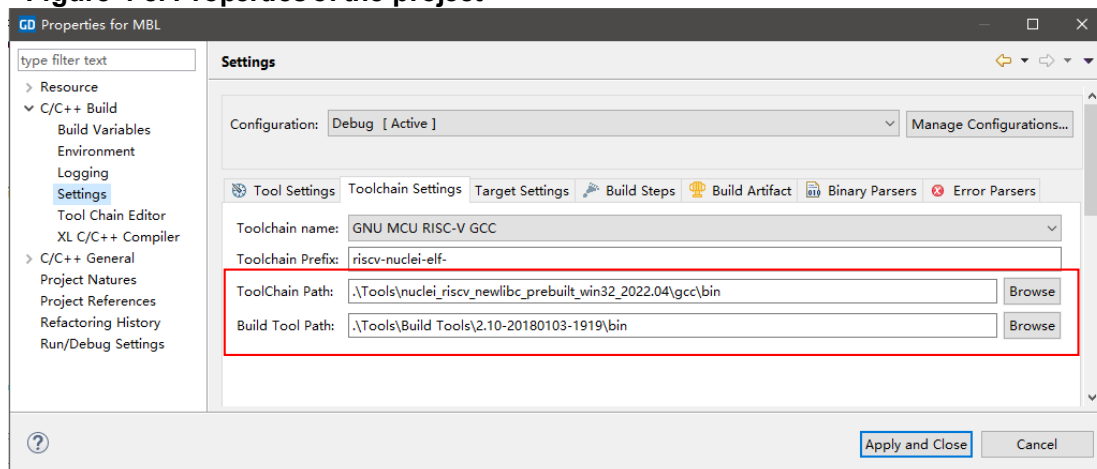
View the MSDK and MBL project interfaces, as shown in [Figure 4-7. MSDK and MBL project interfaces](#).

Figure 4-7. MSDK and MBL project interfaces

4.2. Compilation

- Check the configuration of the project compilation tool

Right-click on the project, click on properties, select C/C++ Build -> Settings in order, and on the tab click on toolchain settings., as shown in [Figure 4-8. Properties of the project](#).

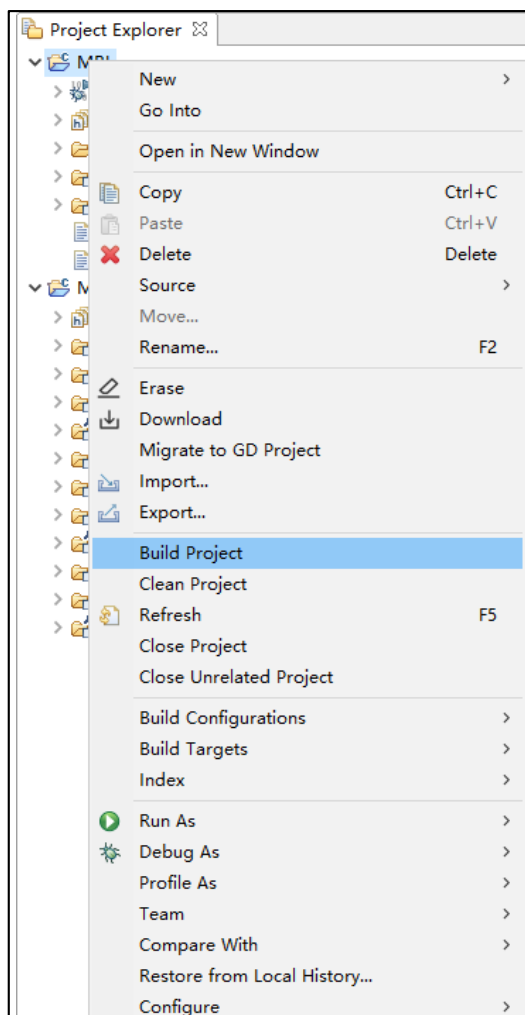
Figure 4-8. Properties of the project

Fill in the paths of ToolChain and Build Tool downloaded by [2.3Toolchain download](#). Click Apply and Close.

- Compile the MBL project

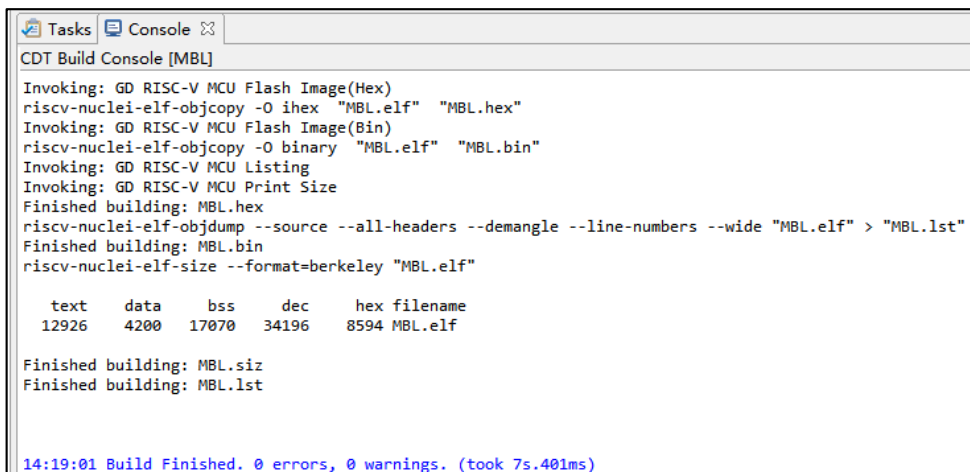
Right-click the project, and click Build Project, as shown in [Figure 4-9. Compiling the MBL project](#).

Figure 4-9. Compiling the MBL project



The compilation result is as shown in [Figure 4-10. MBL compilation result](#).

Figure 4-10. MBL compilation result

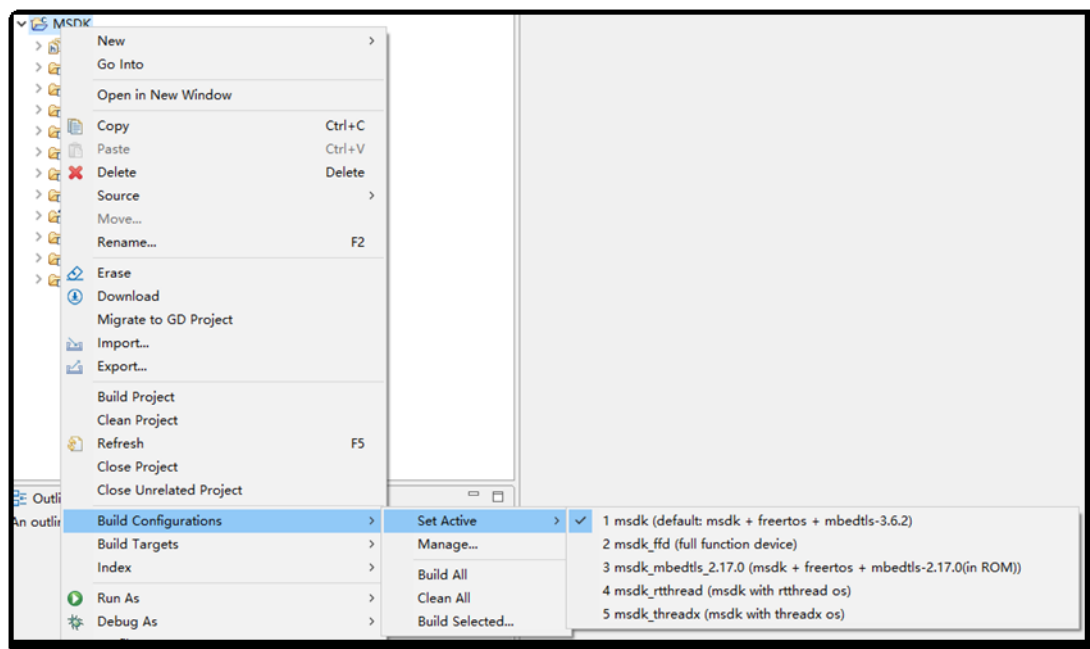


After the compilation is complete, the script MBL\project\mbl_afterbuild.bat will be automatically called to generate mbl.bin and copied to the directory \scripts\images.

■ Compile the MSDK project

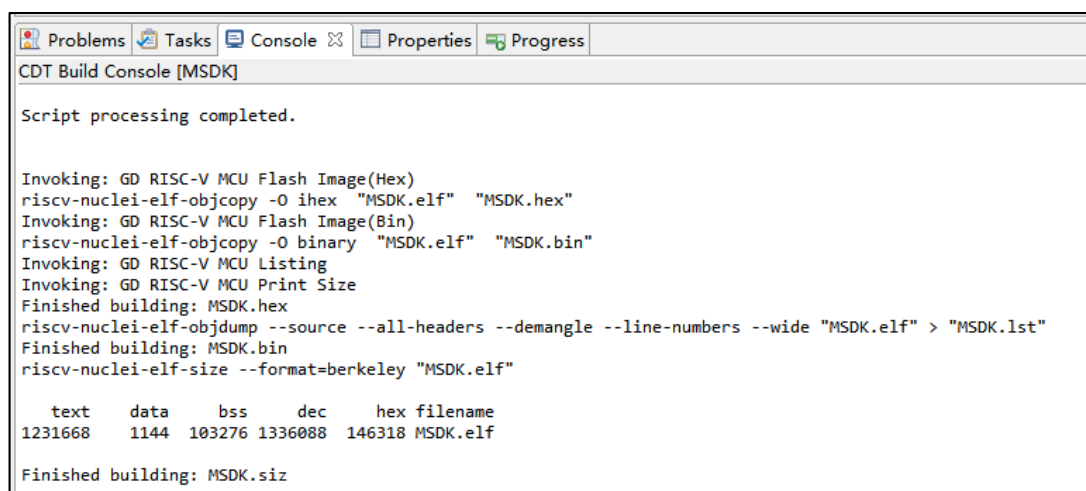
Right-click the project, and click Build Configurations—>Set Active—><target configuration> in order. as shown in [Figure 4-11. target configuration selection](#), the default target project is msdk.

Figure 4-11. target configuration selection



Right-click the project again, and click Build Project, The compilation result is as shown in [Figure 4-12. MSDK compilation result](#).

Figure 4-12. MSDK compilation result



■ Images generated by SDK

After MSDK is compiled, it will call MSDK\projects\image_afterbuild.bat to generate image-ota.bin and image-all.bin, and copy the generated bin files to \scripts\images, as shown in [Figure 4-13. Images output](#).

image-ota.bin is the bin file generated by MSDK project, which can be used for OTA upgrade. image-all.bin is the combination of MBL(mbl.bin) and MSDK(image-ota.bin), the firmware can be used for production, download into FLASH and run.

Figure 4-13. Images output

名称	修改日期	类型	大小
 image-all.bin	2024/7/11 14:26	BIN 文件	788 KB
 image-ota.bin	2024/7/11 14:26	BIN 文件	748 KB
 mbl.bin	2024/7/11 14:19	BIN 文件	17 KB

4.3. Download firmware

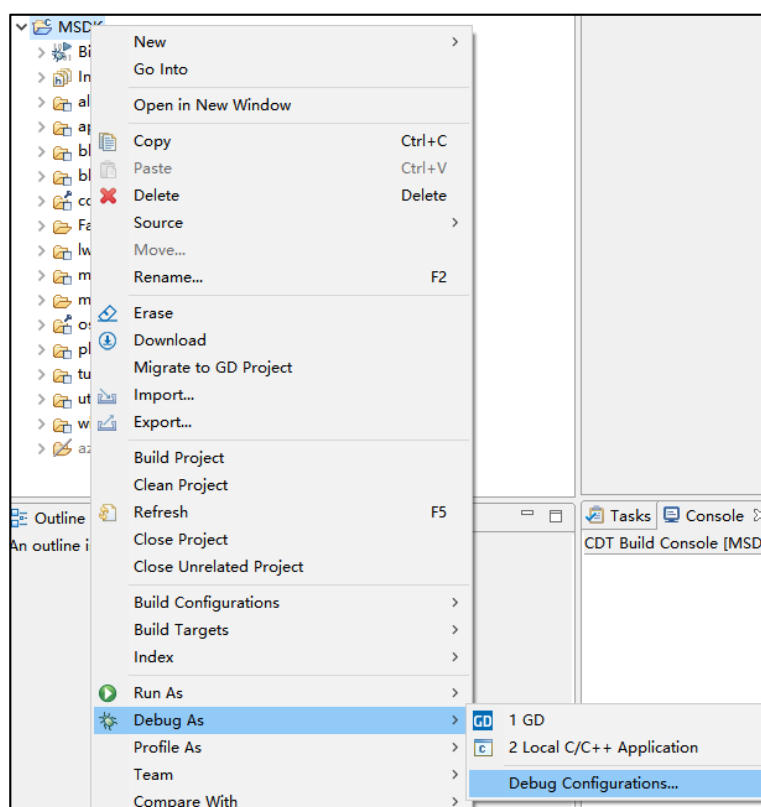
Refer to [1.4 Download interface](#), copy GD32VW55x_RELEASE\scripts\images\image-all.bin to the GigaDevice disc to download bin to flash. And the firmware can also be downloaded via DAPLINK and J-LINK.

4.4. Debugging

■ Debugging configuration

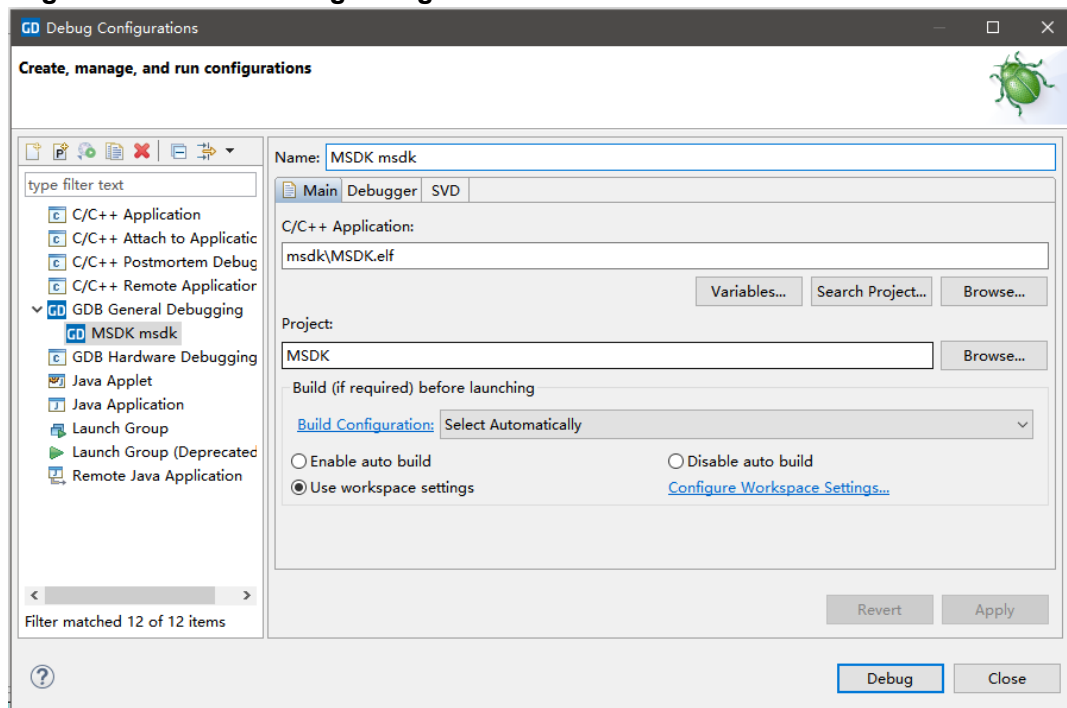
Right-click on the MSDK project and click Debug As->Debug Configurations, as shown in [Figure 4-14. Opening the Debug Configuration option](#).

Figure 4-14. Opening the Debug Configuration option



Double-click GDB General Debugging to open the interface shown in [Figure 4-15. MSDK debug configuration](#), the c/c++ application has automatically selected msdk\MSDK.elf, you can browse to select different configurations corresponding to elf file, for example, msdk_ffd generated elf The file is located at GD32VW55x_RELEASE\MSDK\projects\eclipse\msdk\msdk_ffd\MSDK.elf.

Figure 4-15. MSDK debug configuration



■ Start debugging

Start board integrated GD-Link, it is recommended to use openocd for debugging, openocd is in the path: EmbeddedBuilder_v1.3.13.23294\Tools\OpenOCD, and the GD-Link script for GD32VW55x integrated internally. Switch the GDB Server to openOCD as shown in [Figure 4-16. MSDK Debugging Configuration Interface with openocd](#), and specify the config options as shown in the figure, click Debug, and wait for the completion of image downloading to debug. The debugging interface is shown in [Figure 4-17. MSDK debug interface](#).

Figure 4-16. MSDK Debugging Configuration Interface with openocd

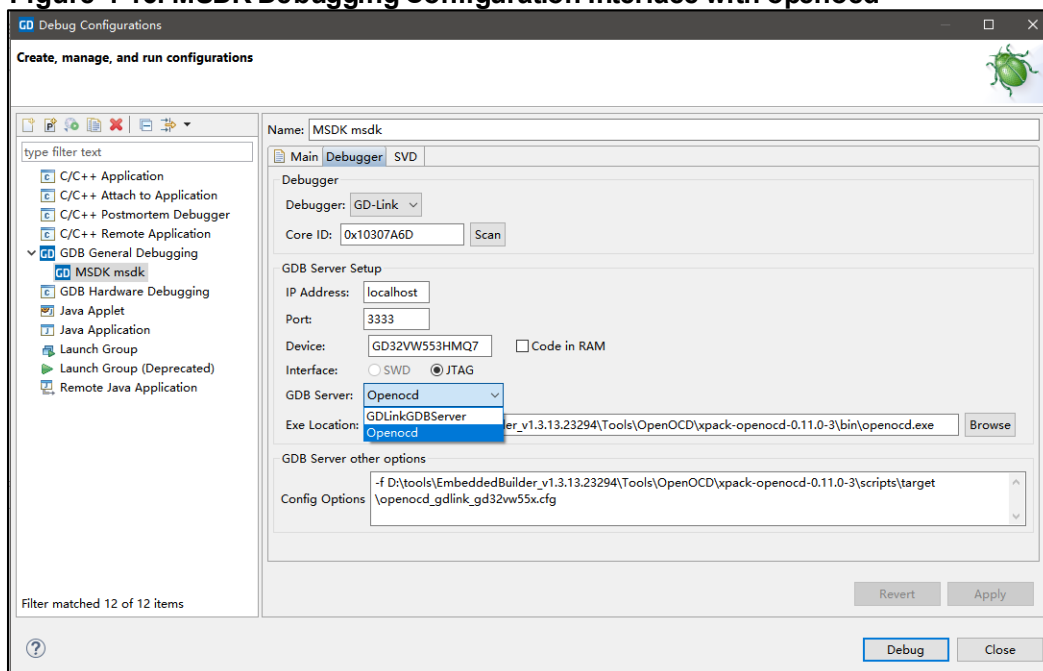
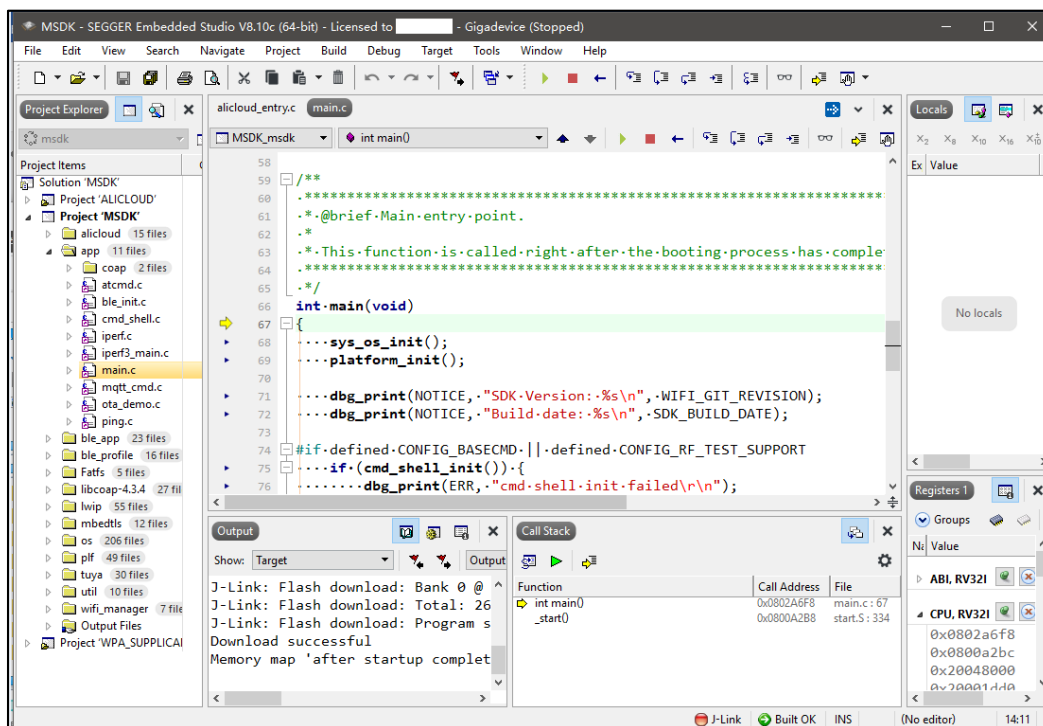


Figure 4-17. MSDK debug interface



5. SEGGER Embedded Studio IDE project

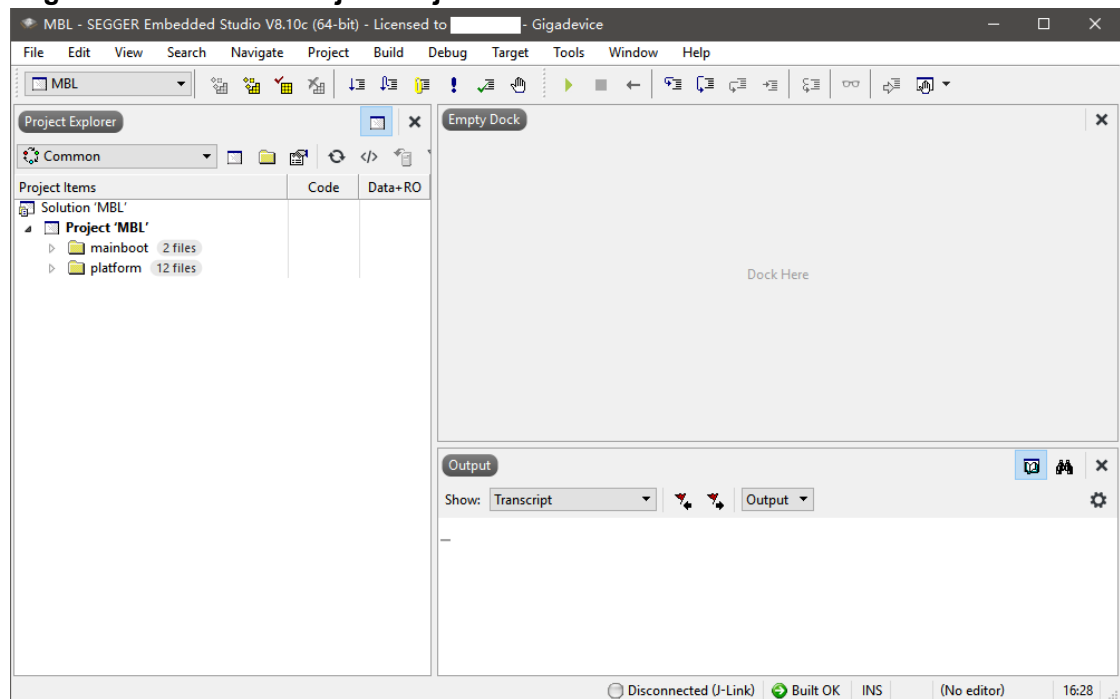
This chapter introduces how to compile and debug the SDK under SEGGER Embedded Studio IDE.

5.1. Open projects

■ Open MBL project

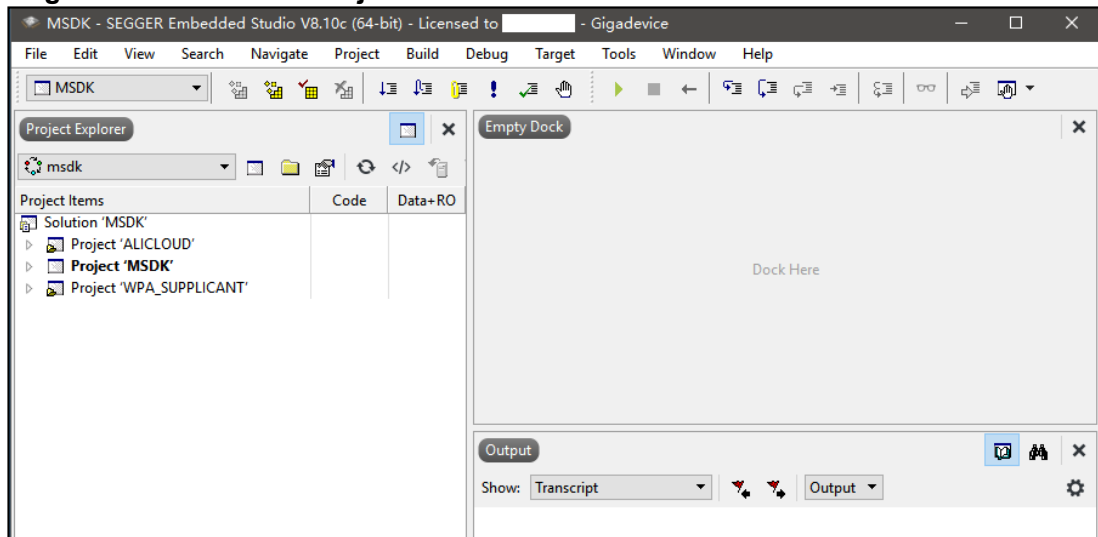
Open the directory: GD32VW55x_RELEASE\MBL\project\segger, double click MBL.emProject to open the MBL SES project. The opened project is shown in [Figure 5-1. MBL SES Project Project Interface](#).

Figure 5-1. MBL SES Project Project Interface



■ Open MSDK project

Open the directory: GD32VW55x_RELEASE\MSDK\projects\segger, double-click on the MSDK.emProject to open the MSDK project, open the project as [Figure 5-2. MSDK SES Project Interface](#) shown.

Figure 5-2. MSDK SES Project Interface

5.2. Compilation

■ Nuclei Toolchain configuration

Please put the directory of Toolchain downloaded in subsection [2.3Toolchain download](#) nuclei_riscv_newlibc_prebuilt_win32_2022.04\gcc\bin into the path of environment variable of windows.

■ SES build tool configuration

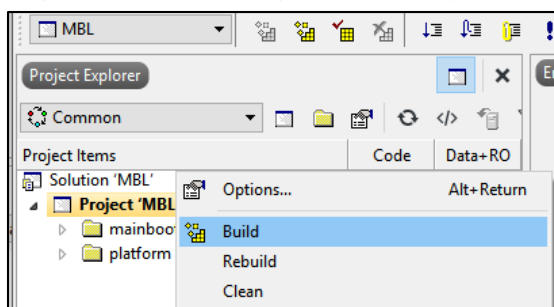
SES compiles the GD32VW55x project using the riscv32-none-elf toolchain by default. In order to better support the extended instruction set of riscv, it needs to be compiled using the nuclei toolchain: riscv-nuclei-elf. The compilation tool can be obtained by contacting sales or FAE. The details of the toolchain are shown in [Figure 5-3. nuclei toolchain content](#). Where the Segger_IDE is the SES IDE installation directory.

Figure 5-3. nuclei toolchain content

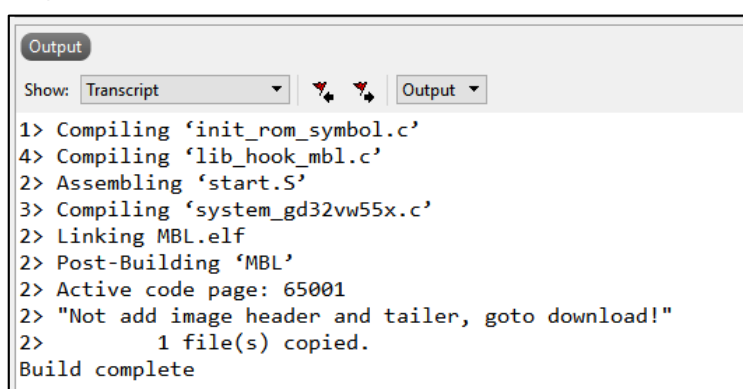
Segger_IDE > gcc > riscv-nuclei-elf		
名称	修改日期	类型
bin	2024/3/18 17:59	文件夹
include	2024/3/19 9:40	文件夹
lib	2024/7/12 15:21	文件夹

■ Compile the MBL project

Right-click the project and click build to guild MBL, as shown in [Figure 5-4. Compiling the MBL project](#); or click Build->Build MBL in the menu bar.

Figure 5-4. Compiling the MBL project

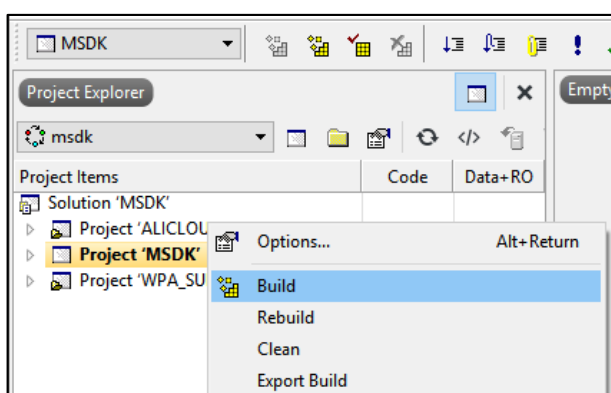
The compilation result is as shown in [Figure 5-5. MBL compilation result](#).

Figure 5-5. MBL compilation result

After the compilation is complete, the script MBL\project\mbl_afterbuild.bat will be automatically called to generate mbl.bin and copied to the directory \scripts\images.

- Compile the MSDK project

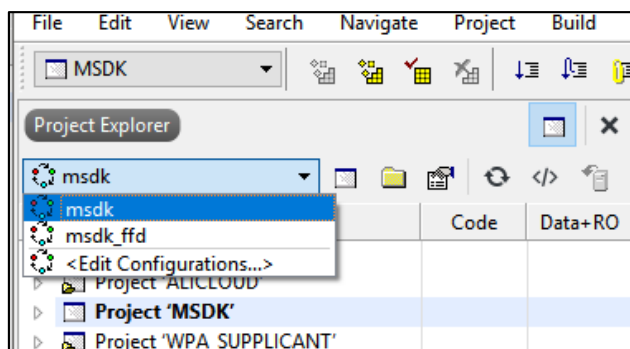
Right-click Project 'MSDK' and click Build, as shown in [Figure 5-6. Compile MSDK project](#)

Figure 5-6. Compile MSDK project

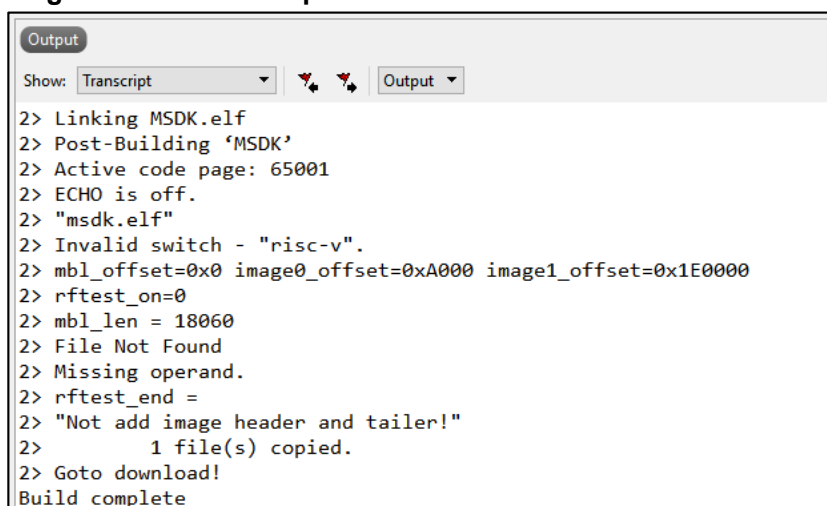
- Configuration selection of MSDK

MSDK configuration switch as shown in [Figure 5-7 MSDK Project Configuration Options](#).

MSDK SES project only supports msdk and msdk_ffd; if you need to use the configuration of msdk_threadx, msdk_ffd_threadx and msdk_azure please use the GD32 EmbeddedBuilder IDE project or wait for subsequent updates.

Figure 5-7 MSDK Project Configuration Options

After selecting the corresponding configuration, right-click the project and click Build, the compilation result is shown in [Figure 5-8 MSDK compilation result](#).

Figure 5-8 MSDK compilation result

■ Image generated by SDK

After MSDK is compiled, it will call MSDK\projects\image_afterbuild.bat to generate image-ota.bin and image-all.bin, and copy the generated bin files to \scripts\images, as shown in [Figure 5-9. Images output](#).

image-ota.bin is the bin file generated by MSDK project, which can be used for OTA upgrade. image-all.bin is the combination of MBL(mbl.bin) and MSDK(image-ota.bin), the firmware can be used for production, download into flash and run.

Figure 5-9. Images output

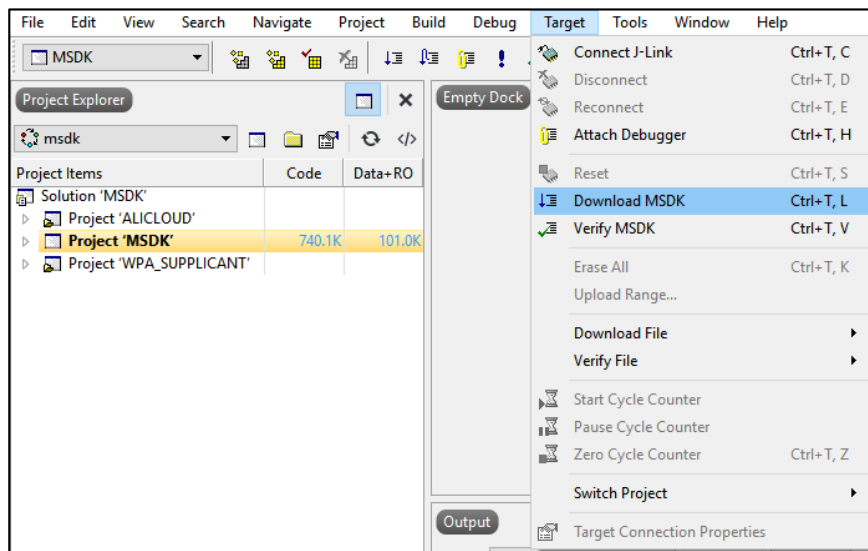
名称	修改日期	类型	大小
image-all.bin	2024/7/11 14:26	BIN 文件	788 KB
image-ota.bin	2024/7/11 14:26	BIN 文件	748 KB
mbl.bin	2024/7/11 14:19	BIN 文件	17 KB

5.3. Download firmware

Refer to [1.4 Download interface](#), copy GD32VW55x_RELEASE\scripts\images\image-

all.bin to the GigaDevice disc to download it. Or download it by clicking Target->Download MSDK in the menu bar, as shown in [Figure 5-10 SES IDE image download](#).

Figure 5-10 SES IDE image download



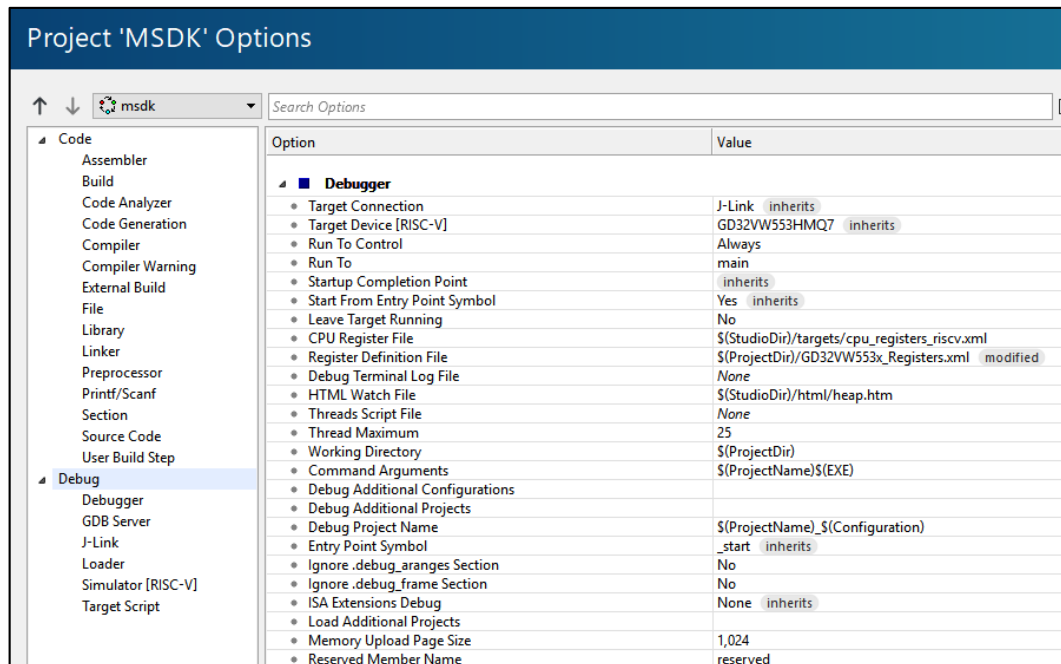
5.4. Debugging

■ Debugging configuration

SES IDE recommends using J-link to debug, and J-link driver version at least V7.92o, this version of J-link driver support GD32VW55x chip.

The project has been configured with Debug information by default, if you need to change it, right-click on the MSDK project, click Options to open the configuration interface, you can modify the Debugger and J-Link under the Debug option, as shown in [Figure 5-11. MSDK SES Project Configuration Interface](#).

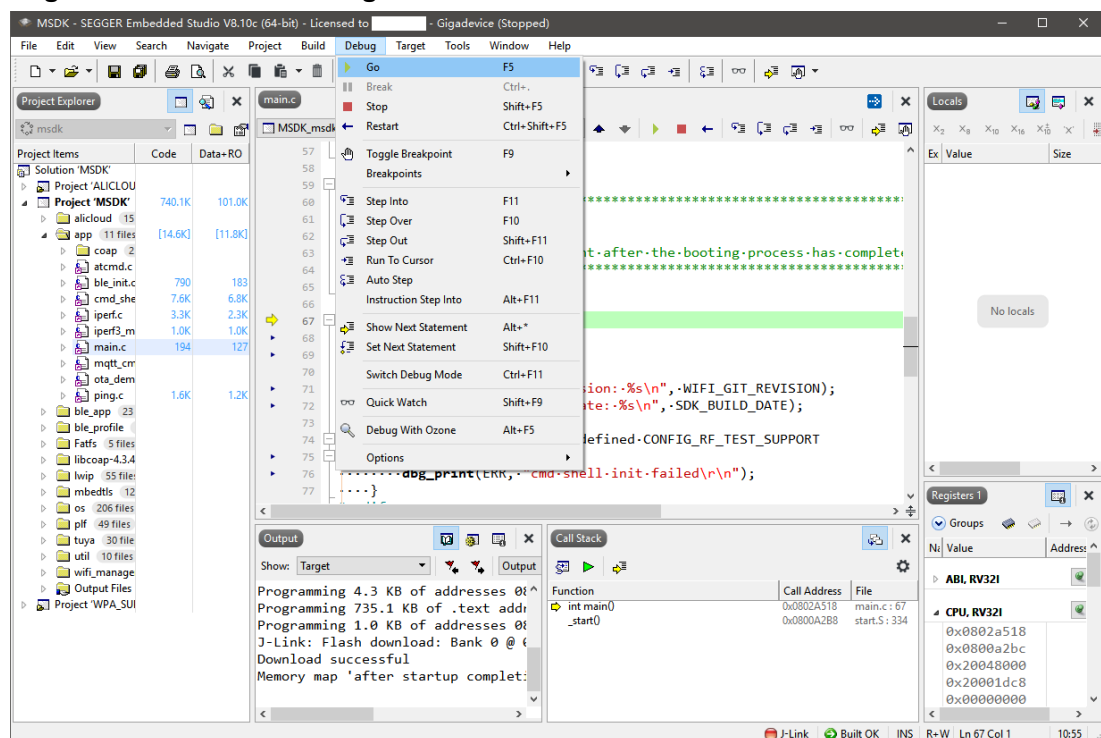
Figure 5-11. MSDK SES Project Configuration Interface



■ Start Debugging

Click Debug->GO in the menu bar to debug, click and wait for the image downloading to complete and enter the interface shown in [Figure 5-12. SES IDE Debug Interface](#).

Figure 5-12. SES IDE Debug Interface



6. FAQ

6.1. 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 downloaded 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: Download the MBL again. After that, the IMAGE status will be cleared.

6.2. Code running in SRAM

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

Open GD32VW55x_RELEASE\MSDK\plf\riscv\env\gd32w55x.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 example:

```
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.

7. Revision history

Table 7-1. Revision history

Revision No.	Description	Date
1.0	Initial release	Nov.24.2023
1.1	Chapter 2 revision	Jan.26.2024
1.2	SES IDE project added, GD32 Eclipse IDE updated to GD32 Embedded Builder	July.17. 2024
1.3	Update some diagrams to be consistent with SDK1.0.3	Apr.8.2025

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 according to the laws of the People's Republic of China and other applicable laws. The Company reserves all rights under such laws and no Intellectual Property Rights are transferred (either wholly or partially) or licensed by the Company (either expressly or impliedly) herein. 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.

To the maximum extent permitted by applicable law, the Company makes no representations or warranties of any kind, express or implied, with regard to the merchantability and the fitness for a particular purpose of the Product, nor does the Company assume any liability arising out of the application or use of any Product. Any information provided in this document is provided only for reference purposes. It is the sole responsibility of the user of this document to determine whether the Product is suitable and fit for its applications and products planned, and properly design, program, and test the functionality and safety of its applications and products planned using the Product. The Product is designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and the Product is not designed or intended for use in (i) safety critical applications such as weapons systems, nuclear facilities, atomic energy controller, combustion controller, aeronautic or aerospace applications, traffic signal instruments, pollution control or hazardous substance management; (ii) life-support systems, other medical equipment or systems (including life support equipment and surgical implants); (iii) automotive applications or environments, including but not limited to applications for active and passive safety of automobiles (regardless of front market or aftermarket), for example, EPS, braking, ADAS (camera/fusion), EMS, TCU, BMS, BSG, TPMS, Airbag, Suspension, DMS, ICMS, Domain, ESC, DCDC, e-clutch, advanced-lighting, etc.. Automobile herein means a vehicle propelled by a self-contained motor, engine or the like, such as, without limitation, cars, trucks, motorcycles, electric cars, and other transportation devices; and/or (iv) other uses where the failure of the device or the Product can reasonably be expected to result in personal injury, death, or severe property or environmental damage (collectively "Unintended Uses"). Customers shall take any and all actions to ensure the Product meets the applicable laws and regulations. The Company is not liable for, in whole or in part, and customers shall hereby 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 Product. Customers shall indemnify and hold the Company, and its officers, employees, subsidiaries, affiliates 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 Product.

Information in this document is provided solely in connection with the Product. The Company reserves the right to make changes, corrections, modifications or improvements to this document and the Product described herein at any time without notice. The Company shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. Information in this document supersedes and replaces information previously supplied in any prior versions of this document.