

**GigaDevice Semiconductor Inc.**

**GD32VW553 Quick Development Guide**

**Application Note**

**AN154**

Revision 1.3b

(Feb 2026)

# Table of Contents

<b>Table of Contents .....</b>	<b>2</b>
<b>List of Figures .....</b>	<b>4</b>
<b>List of Tables .....</b>	<b>6</b>
<b>1. Introduction to development board .....</b>	<b>7</b>
1.1. Picture of real development board .....	7
1.1.1. The START development board .....	7
1.1.2. The EVAL development board .....	9
1.2. Boot mode.....	11
1.3. Debugger interface.....	11
1.4. Download interface.....	12
1.5. Viewing log.....	12
<b>2. Building development environment.....</b>	<b>13</b>
2.1. Installation of GD32 Embedded Builder .....	13
2.2. Installation of SEGGER Embedded Studio IDE .....	13
<b>3. What developers must know .....</b>	<b>14</b>
3.1. SDK execution program group.....	14
3.2. SDK configuration .....	14
3.2.1. Configuration of wireless module.....	14
3.2.2. SRAM layout .....	15
3.2.3. FLASH layout.....	15
3.2.4. Firmware version No.....	15
3.2.5. APP configuration .....	16
3.2.6. Configuration Selection.....	17
3.3. Correct log example .....	18
<b>4. GD32 Embedded Builder IDE project.....</b>	<b>19</b>
4.1. Opening the project group.....	19
4.2. Compilation.....	22
4.3. Download firmware.....	25
4.3.1. USB Drive Copy .....	25
4.3.2. Use afterbuild.bat for downloading .....	25
4.3.3. Using J-Flash Lite for downloading .....	26
4.4. Debugging.....	27
4.4.1. Debugging configuration .....	27

---

4.4.2.	Debugging using GDLink.....	28
4.4.3.	Debugging using Jlink.....	29
<b>5.</b>	<b>SEGGER Embedded Studio IDE project .....</b>	<b>30</b>
5.1.	Open projects .....	30
5.2.	Compilation.....	31
5.3.	Download firmware.....	33
5.4.	Debugging.....	34
<b>6.</b>	<b>FAQ .....</b>	<b>36</b>
6.1.	No image error .....	36
6.2.	Code running in SRAM.....	36
6.3.	Select different project configurations during debugging .....	36
6.4.	JLink Driver Replacement.....	37
<b>7.</b>	<b>Revision history.....</b>	<b>39</b>

## List of Figures

Figure 1-1. The picture of the START development board .....	7
Figure 1-2. GD32VW553-MINI-I Module(Left)and GD32VW553-MINI-E Module(Right) .....	8
Figure 1-3. The picture of the EVAL development board .....	10
Figure 1-4. Development Board Type Configuration .....	11
Figure 1-5. List of devices and drivers .....	12
Figure 1-6. Configuration of serial port .....	12
Figure 2-1 The Directory Structure of GD32 Embedded Builder .....	13
Figure 3-1. Boot process .....	14
Figure 3-2. Configuration of wireless module .....	14
Figure 3-3. SRAM layout .....	15
Figure 3-4. FLASH layout .....	15
Figure 3-5. Firmware version No. ....	16
Figure 3-6 BLE library selection .....	16
Figure 3-7. Project boot information .....	18
Figure 4-1. SDK directory .....	19
Figure 4-2. Starting GD32 Embedded Builder IDE .....	19
Figure 4-3. Open Projects from file System .....	20
Figure 4-4. Selecting MBL project path .....	20
Figure 4-5. MBL project interface .....	21
Figure 4-6. Selecting MSDK project path .....	21
Figure 4-7. MSDK and MBL project interfaces .....	22
Figure 4-8. Properties of the project .....	22
Figure 4-9. Compiling the MBL project .....	23
Figure 4-10. MBL compilation result .....	23
Figure 4-11. target configuration selection .....	24
Figure 4-12. MSDK compilation result .....	24

Figure 4-13. Images output.....	25
Figure 4-14 Configure image automatic downloading .....	25
Figure 4-15 JFlashLite Configuration .....	26
Figure 4-16 J-Flash Programming Interface .....	26
Figure 4-17. Opening the Debug Configuration option .....	27
Figure 4-18. MSDK debug configuration .....	28
Figure 4-19. MSDK Debugging Configuration Interface with openocd .....	28
Figure 4-20. MSDK debug interface .....	29
Figure 5-1. MBL SES Project Project Interface .....	30
Figure 5-2. MSDK SES Project Interface .....	31
Figure 5-3. nuclei toolchain content .....	31
Figure 5-4. Compiling the MBL project.....	32
Figure 5-5. MBL compilation result.....	32
Figure 5-6. Compile MSDK project.....	32
Figure 5-7 MSDK Project Configuration Options .....	33
Figure 5-8 MSDK compilation result.....	33
Figure 5-9. Images output.....	33
Fogure 5-10 SES IDE image download.....	34
Figure 5-11. MSDK SES Project Configuration Interface .....	35
Figure 5-12. SES IDE Debug Interface .....	35
Figure 6-1 Select Project Configuration for Debugging .....	37
Figure 6-2 Zadig Options Selection .....	37
Figure 6-3 Replace JLink Driver.....	38

## List of Tables

Table 1-1. The START development board connectors and switch functions .....	8
Table 1-2. The START development board main interfaces description.....	9
Table 1-3. Boot mode.....	11
Table 7-1. Revision history .....	39

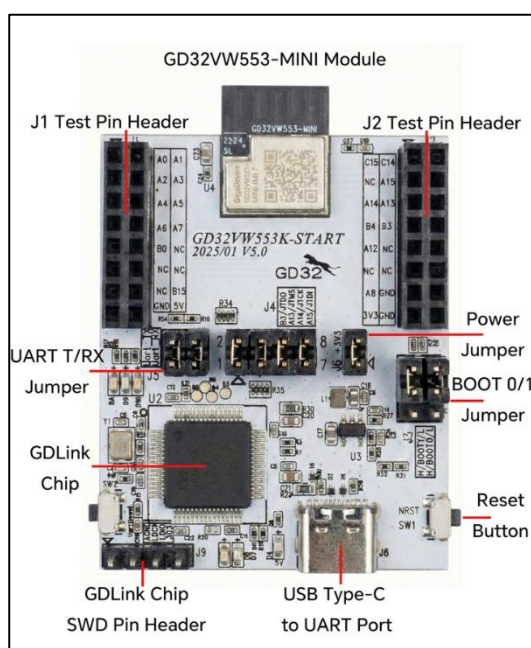
## 1. Introduction to development board

### 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**

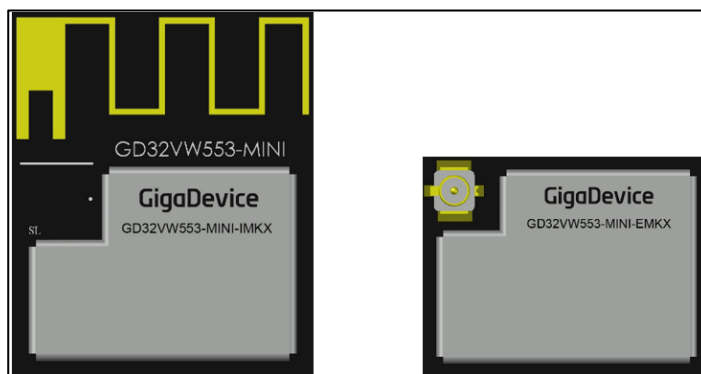


Mainly focus on the following parts of the development board, which have been marked in the [Figure 1-1. The picture of the START development board](#).

- Boot mode (Boot PIN);
- Power supply port (USB Type-C);
- View log (UART);
- Debugger interface (JLink, or GDLink);
- Reboot (Reset Button).

The START V5.0 development board supports the GD32VW553-MINI-I and GD32VW553-MINI-E wireless modules. GD32VW553-MINI-I module and GD32VW553-MINI-E module are shown in [Figure 1-2. GD32VW553-MINI-I Module\(Left\)and GD32VW553-MINI-E Module\(Right\)](#).

**Figure 1-2. GD32VW553-MINI-I Module(Left)and GD32VW553-MINI-E Module(Right)**



The connectors and switch functions of the START development board are shown in [Table 1-1. The START development board connectors and switch functions.](#)

**Table 1-1. The START development board connectors and switch functions**

Interface	Description
J1	Interface to GPIO pins PA0~PA7 / PB0 / PB15 of the main chip, as well as +5V interface and GND interface.
J2	Interface to main chip PA8 / PA12~PA15 / PB3 / PB4 / PC14 / PC15 GPIO pins, as well as module +3.3V power supply test interface, GND interface.
J3	Connect to the main chip PC8 (BOOT0) and PB1 (BOOT1) pins, Boot mode selection needs to be configured accordingly. Default BOOT0 / 1 use shorting caps pull-down, that is, the chip default boot from Sip flash.
J4	Interface to the JTAG pins of PB3(JTDO) / PA13(JTMS) / PA14(JTCK) / PA15(JTDI) of the main chip, and interface to the JTAG pins of L_TDO / L_TMS / L_TCK / L_TDI of the GDLINK chip. By default, the above pins of the main chip and the GDLINK chip are connected by shorting caps, and the firmware can be burned into the main chip by the GDLINK chip.
J5	The interface to the UART TX / RX pins of the main chip PA6 (UART2_TX) and PA7 (UART2_RX), and the interface to the L_UART_RX, L_UART_TX and UART RX / TX pins of the GDLINK chip. By default, the above pins of the main chip and the GDLINK chip are connected via shorting caps, so that serial communication can be done between the GDLINK chip and the main chip.
J6	GD32VW553 module 3.3V power supply connection port, the default use of short-circuit cap connection. For power consumption test, external 3.3V power supply can be directly connected to J6.2.
J8	USB-C interface, default serial communication and +5V power supply interface.
J9	Interface to GDLINK chip L_SWKDIO / L_SWKCK and other SWD pins, as well as GDLINK +3.3V power supply test interface and GND interface.
SW1	Connect the module to the NRST pin with GND via a 1K ohm resistor pull-up to 3.3 V. Press and release this switch to Reset the main chip.
SW2	Reserved button



The main interfaces description of the START development board are described in [Table 1-2. The START development board main interfaces description](#).

For more information about the START development board, you can refer to the GD32 official website's [GD32VW553K-START Demo Suites](#) page and download the attached content.

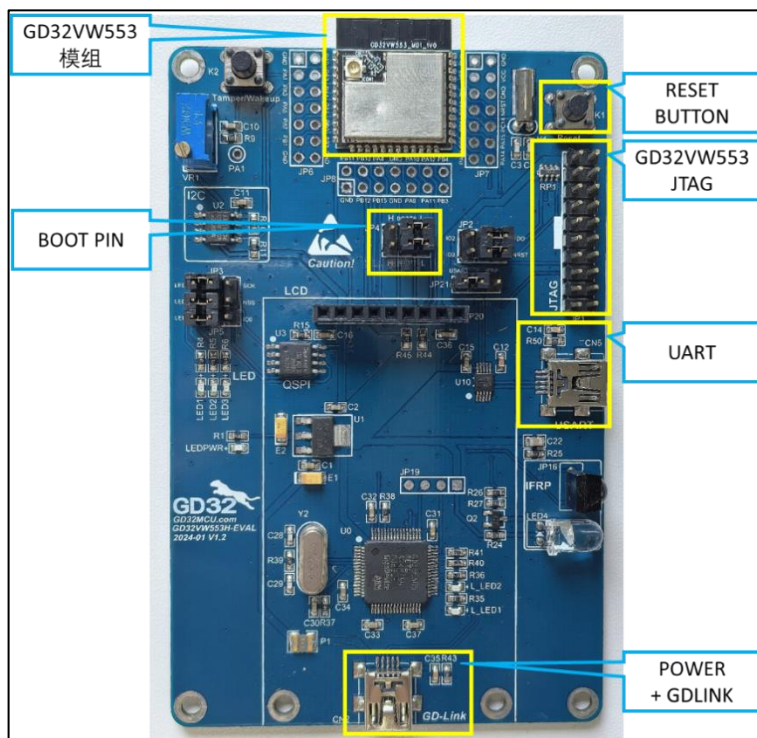
**Table 1-2. The START development board main interfaces description**

Interface	Description
PA0	IO port that can be configured by the user.
PA1	IO port that can be configured by the user.
PA2	IO port that can be configured by the user.
PA3	IO port that can be configured by the user.
PA4	IO port that can be configured by the user.
PA5	IO port that can be configured by the user.
PA6 / UART2_TX	IO port that can be configured by the user; UART TX.
PA7 / UART2_RX	IO port that can be configured by the user; UART RX.
PB0	IO port that can be configured by the user.
PB1 / BOOT1	IO port that can be configured by the user; Boot mode selection.
GND	Reference ground
PB15	IO port that can be configured by the user.
PA8	IO port that can be configured by the user.
PA12	IO port that can be configured by the user.
PB3 / JTDO	IO port that can be configured by the user; JTDO pin.
PB4 / JNTRST	IO port that can be configured by the user; JNTRST pin.
PA13 / JTMS	IO port that can be configured by the user; JTMS pin.
PA14 / JTCK	IO port that can be configured by the user; JTCK pin.
PC8 / BOOT0	IO port that can be configured by the user; Boot mode select.
PA15 / JTDI	IO port that can be configured by the user; JTDI pin.
PC14	IO port that can be configured by the user.
PC15	IO port that can be configured by the user.
NRST	Module reset pin, connect to 3.3V power supply to enable the module.
3V3	3.3V power supply pin
GND	Reference ground

### 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 1-3. 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 1-3. 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 more information about the EVAL development board, you can refer to the GD32 official website's [GD32VW553 Demo Suites](#) page and download the attached content.

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 1-4. 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 1-4. 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 1-3. Boot mode](#). For more instructions on the boot mode, please refer to the document "GD32VW55x\_User\_Manual".

**Table 1-3. 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 GDLink(GD32E505) debugger that can be used with OpenOCD. Can also use an external debugger (GDLink or JLink) at the JTAG interface of the board for debugging and download. The GD32E505 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 middle 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.

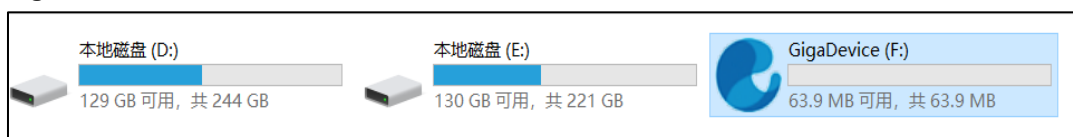
For EVAL development board, GDLink or JLink debugger can be used for debugging and download.

It should also be noted that the GD32VW55x supports cJTAG and JTAG but does not support the SWD debugging interface.

## 1.4. Download interface

For the START development board, in addition to using the GDLink debugger or JLink debugger mentioned in the previous section for firmware downloading, if debugging functionality is not required and only firmware downloading is needed, the firmware can also be downloaded using a USB drive copy method. Connect the development board to a computer via a USB cable, as shown in [Figure 1-5. List of devices and drivers](#), under the devices and drives list as the GigaDevice drive. Copy the "image-all.bin" file (refer to subsequent sections) into the GigaDevice drive to complete the FLASH programming of the GD32VW55x chip.

**Figure 1-5. List of devices and drivers**

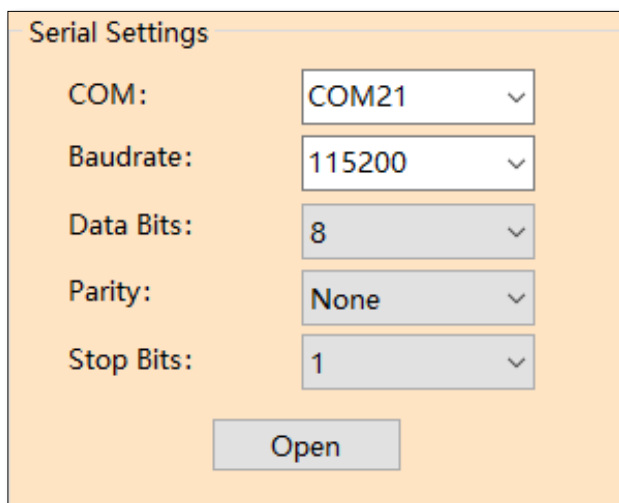


For EVAL development board, GDLink or JLink debugger can be used for download. Dragging into the USB disk is not supported.

## 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 1-6. Configuration of serial port](#) and connect to the board. After that, use the serial port to output logs.

**Figure 1-6. Configuration of serial port**



## 2. Building development environment

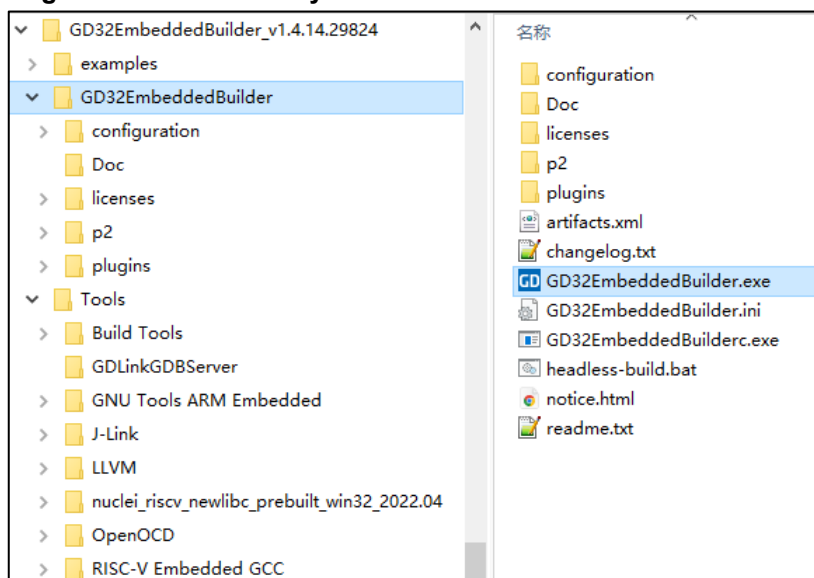
Build a development environment before compiling and downloading the firmware.

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. The build tool, tool chain, openocd, jlink, and other related tools have all been placed in the Tools directory.

**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.

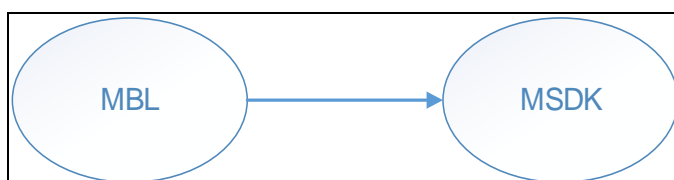
## 3. What developers must know

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\plf\riscv\env\gd32vw55x.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-LAYEROUT-*/
#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.....0x1E0000
#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\plf\riscv\env\gd32vw55x.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 .....0x01000003
#define RE_IMG_VERSION .....0x01000003
```

### 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.

By modifying the macro CONFIG\_BLE\_LIB in app\_cfg.h, the BLE library can be switched. When CONFIG\_BLE\_LIB is set to BLE\_LIB\_MIN (as shown in [Figure 3-6 BLE library selection](#)), the project compilation will use libble.a, and the header file will include ble\_config\_min.h. When CONFIG\_BLE\_LIB is set to BLE\_LIB\_MAX, the project compilation will use libble\_max.a, and the header file will include ble\_config\_max.h.

**Figure 3-6 BLE library selection**

```
#define BLE_LIB_MIN .....0 .....//only peripheral and server
#define BLE_LIB_MAX .....1 .....//add central and client usage

#define CONFIG_BLE_LIB .....BLE_LIB_MIN
```

The features supported by libble.a are as follows:

1. Supports peripheral
2. Supports a single connection link
3. Supports server
4. Supports host and controller
5. Supports EATT
6. Supports WeChat applet WiFi provisioning

Based on libble.a, libble\_max.a additionally supports the following features:

1. Supports central
2. Supports four connection links
3. Supports client



4. Supports periodic advertising
5. Supports PHY updates
6. Supports power control
7. Supports BLE ping
8. Supports secure connection

### 3.2.6. Configuration Selection

The main project-MSDK, supports multiple configurations, with `msdk` selected by default. Additional options include `msdk_ffd`, `msdk_mbedtls_2.17.0`, `msdk_rtthread`, and `msdk_threadx`.

The main difference between `msdk_ffd` and `msdk` lies in the WiFi connection management library included in the project. The `msdk` includes `libwpas`, which is more streamlined and consumes fewer memory resources. The `msdk_ffd` includes `wpa_supplicant`, which is more comprehensive and general-purpose but has a larger codebase and consumes more memory resources. Additionally, `msdk_ffd` includes `libble_max.a` by default, enabling more BLE features. Of course, `msdk` can switch between `libble.a` and `libble_max.a` by modifying the configuration.

The main difference between `msdk_mbedtls_2.17.0` and `msdk` lies in the version of the MbedTLS library included in the project. The `msdk` includes MbedTLS 3.6.5, which runs in FLASH. The `msdk_mbedtls_2.17.0` includes MbedTLS 2.17.0, with most of its content running in ROM. If strict security requirements are needed, it is recommended to choose `msdk`. If FLASH space is limited, it is recommended to choose `msdk_mbedtls_2.17.0`.

The main difference between `msdk_rtthread` and `msdk` lies in the RTOS used in the project. The `msdk` uses FreeRTOS. The `msdk_rtthread` uses RT-Thread.

The main difference between `msdk_threadx` and `msdk` also lies in the RTOS used in the project. The `msdk` uses FreeRTOS. The `msdk_threadx` uses ThreadX.

- `msdk`  
FreeRTOS + `Libwpas.a` + `libble.a` + MbedTLS 3.6.5
- `msdk_ffd`  
FreeRTOS + `wpa_supplicant.a` + `libble_max.a` + MbedTLS 3.6.5
- `msdk_mbedtls_2.17.0`  
FreeRTOS + `Libwpas.a` + `libble.a` + ROM MbedTLS 2.17.0
- `msdk_rtthread`  
RTThread + `Libwpas.a` + `libble.a` + MbedTLS 3.6.5

- msdk\_threadx

Threadx + Libwpas.a + libble.a + MbesTLS 3.6.5

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 3-7. Project boot information](#). If an exception occurs, please check [6FAQ](#) for help.

**Figure 3-7. 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).
=== RF initialization finished ===
SDK Version: v1.0.3a-86d78d058d779fad
Build date: 2025/05/14 16:29:11
=== WiFi calibration done ===
=== PHY initialization finished ===
BLE local addr: AB:89:67:45:23:01, type 0x0
=== BLE Adapter enable complete ===
```

## 4. GD32 Embedded Builder IDE project

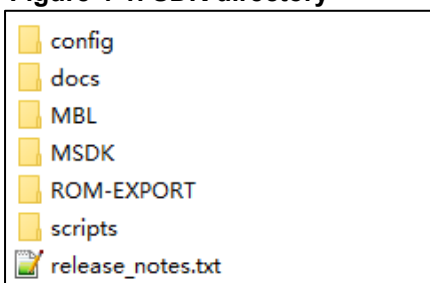
This chapter introduces how to compile and debug the SDK under Embedded Builder IDE.

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

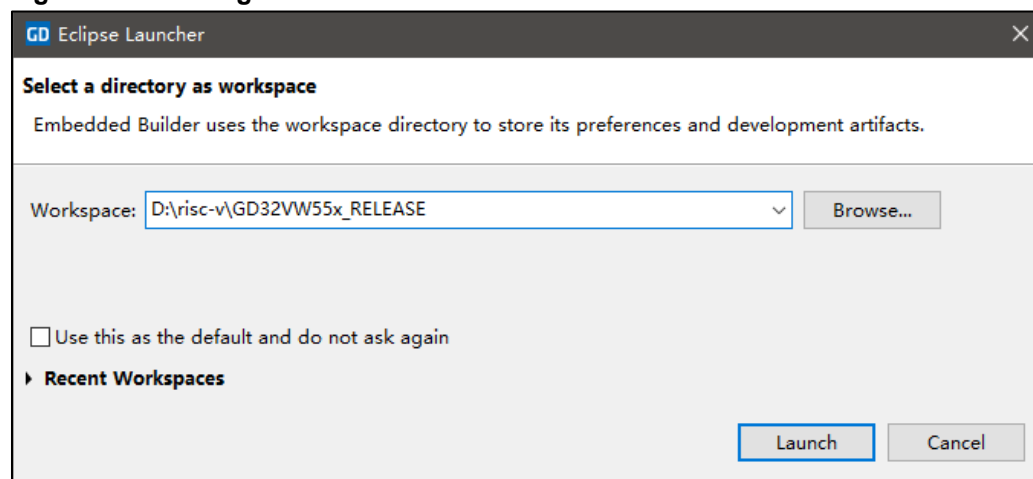
Check the SDK directory GD32VW55x\_RELEASE, as shown in [Figure 4-1. SDK directory](#).

**Figure 4-1. SDK directory**



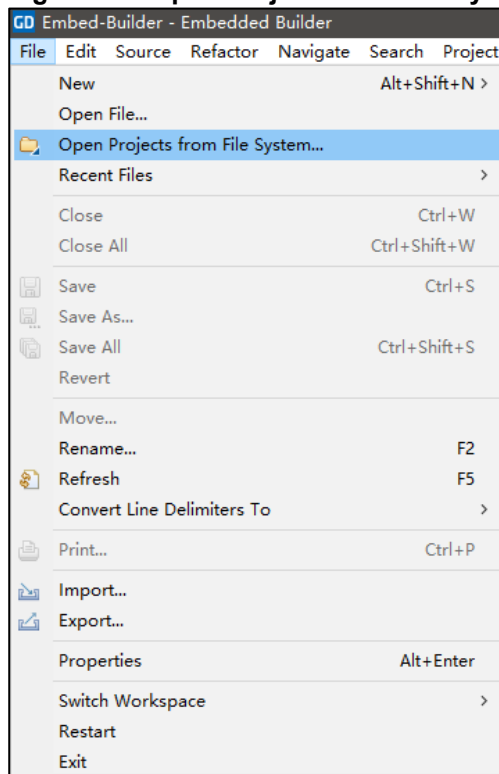
To start the IDE, double-click Embedded Builder.exe in the Embedded Builder directory, and select the SDK directory GD32VW55x\_RELEASE as the workspace, and then click the launch button, as shown in [Figure 4-2. Starting GD32 Embedded Builder IDE](#).

**Figure 4-2. Starting GD32 Embedded Builder IDE**

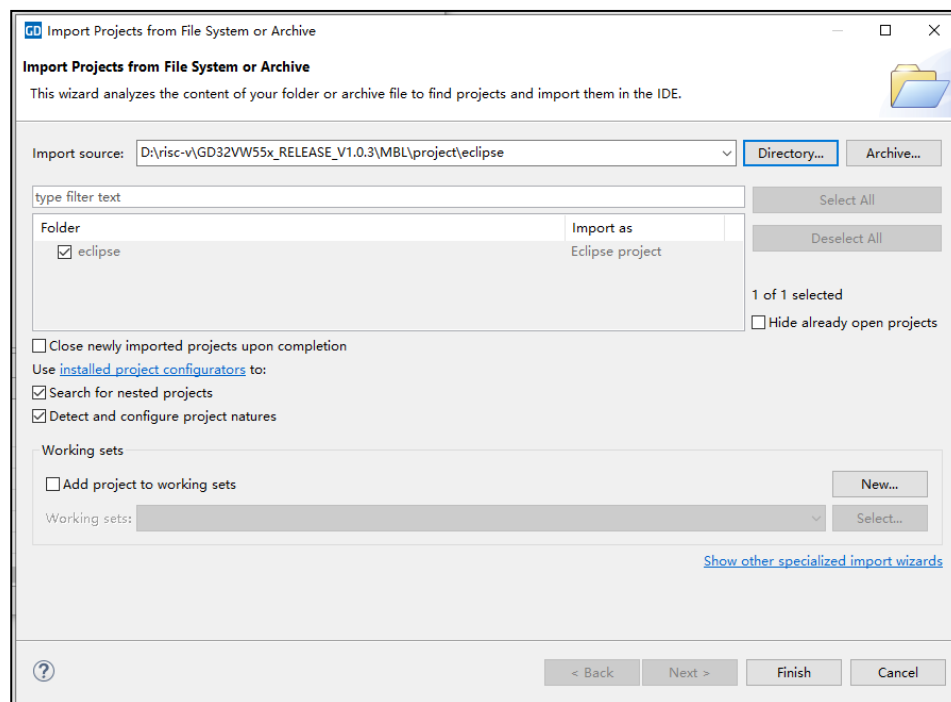


- Import the MBL project

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

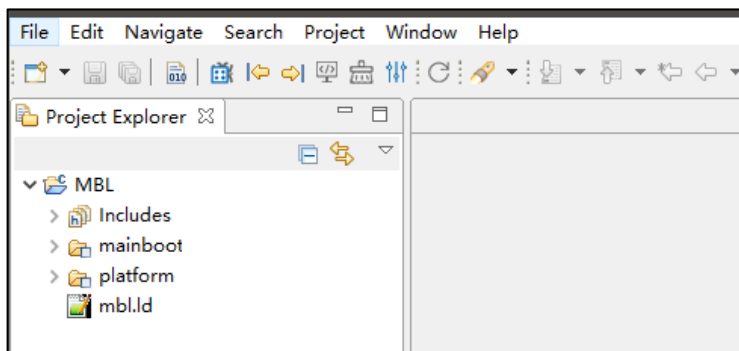
**Figure 4-3. Open Projects from file System**

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.

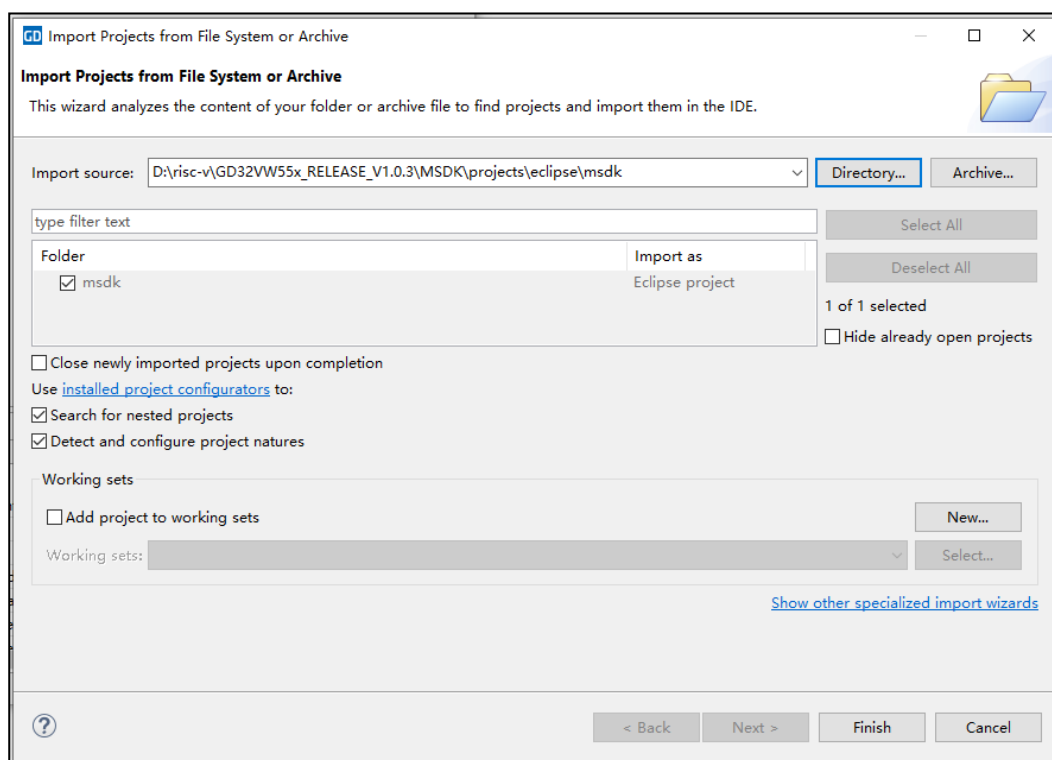
**Figure 4-5. MBL project interface**



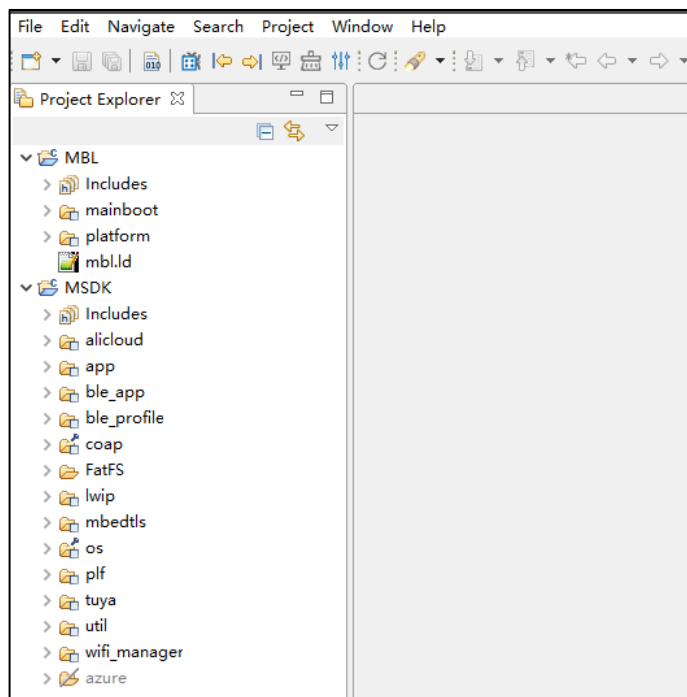
### ■ Import the MSDK project

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**



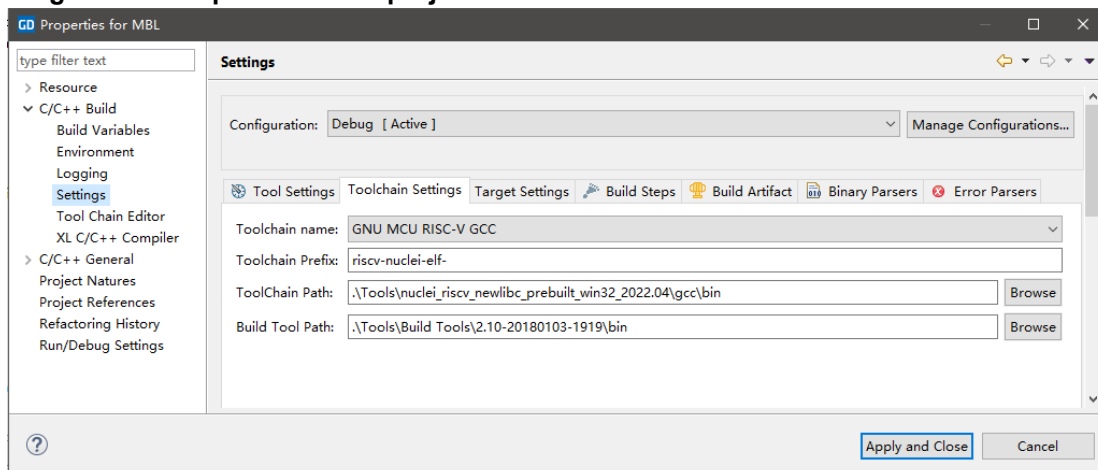
View the MSDK and MBL project interfaces, as shown in [Figure 4-7. MSDK and MBL project interfaces](#).

**Figure 4-7. MSDK and MBL project interfaces**

## 4.2. Compilation

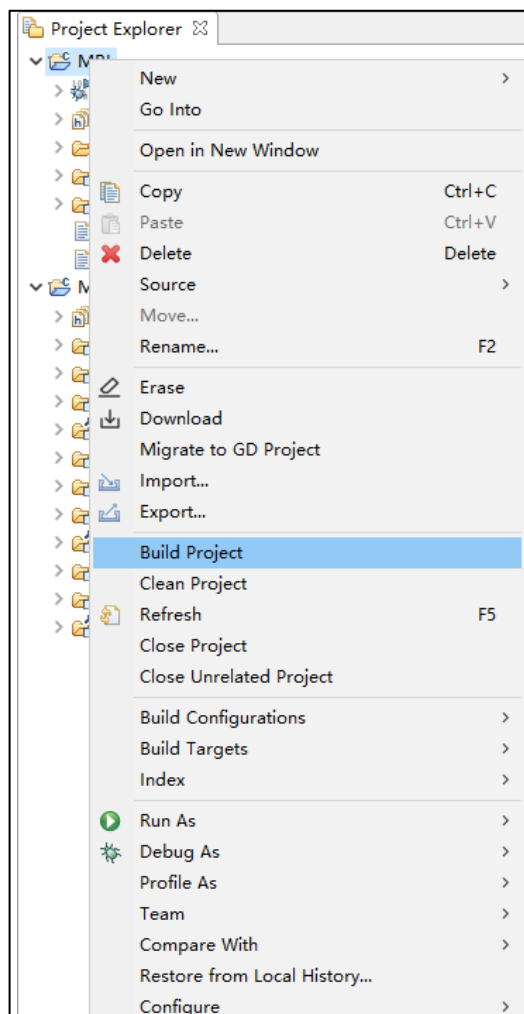
- Check the configuration of the project compilation tool

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](#).

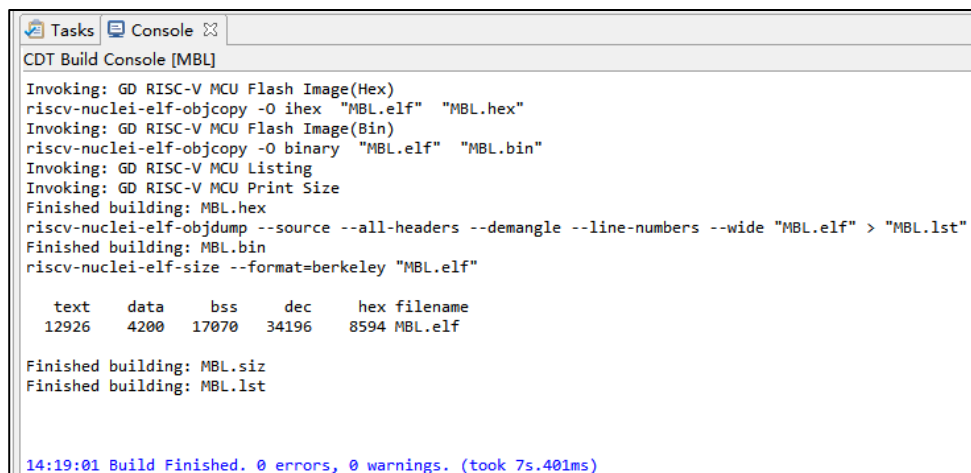
**Figure 4-8. Properties of the project**

- 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](#).

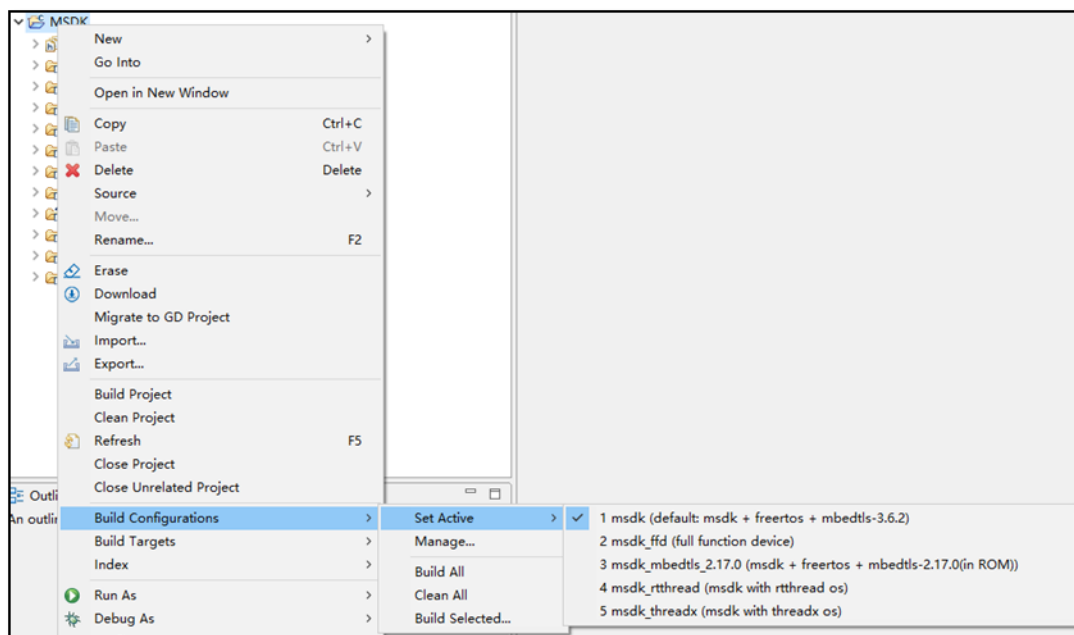
**Figure 4-10. MBL compilation result**

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

### ■ Compile the MSDK project

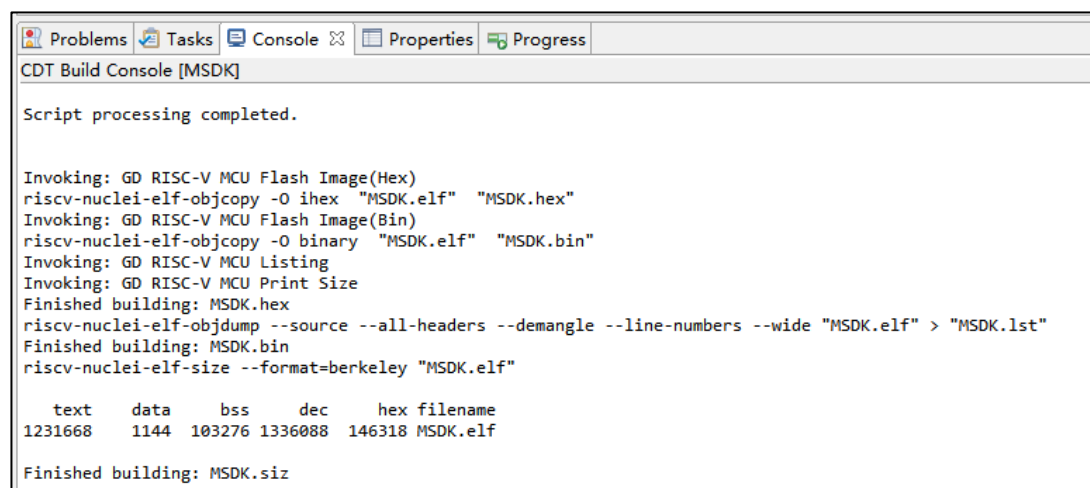
Right-click the project, and click Build Configurations—>Set Active—><target configuration> in order. as shown in [Figure 4-11. target configuration selection](#), the default target project is msdk.

**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

### 4.3.1. USB Drive Copy

As shown in [1.4Download interface](#), copying the image-all.bin file from GD32VW55x\_RELEASE\scripts\images to the GigaDevice drive. This functionality is only supported on the START development board when using the onboard GDLink connection.

### 4.3.2. Use afterbuild.bat for downloading

The project supports automatically downloading the image after compilation by invoking a script through afterbuild. The script file is MSDK\projects\image\_afterbuild.bat.

At the end of image\_afterbuild.bat, there is a section of code configured for automatic image downloading, as shown in [Figure 4-14 Configure image automatic downloading](#).

**Figure 4-14 Configure image automatic downloading**

```
:download
set OPENOCD="%OPENOCD_PATH%\openocd.exe"
::set LINKCFG="..\openocd_gdlink.cfg"
set LINKCFG="..\openocd_jlink.cfg"
@echo on
::%OPENOCD% -f %LINKCFG% -c "program %DOWNLOAD_BIN% 0x080A000 verify reset exit;"
:end
```

This segment of code utilizes OpenOCD with Jlink/GDLink to perform downloading. By configuring OpenOCD to use different .cfg files, users can choose whether to download via Jlink or GDLink.

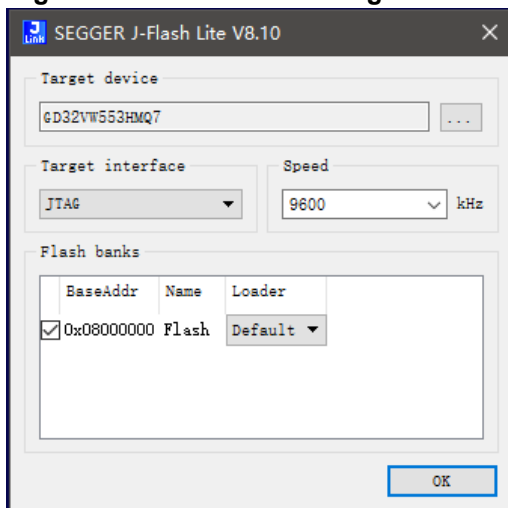
When using GDLink to connect the computer and development board, set LINKCFG to openocd\_gdlink.cfg. For connections via Jlink, set LINKCFG to openocd\_jlink.cfg.

After compilation, Afterbuild will invoke this script to execute the configured commands and complete the corresponding image download (please uncomment the lines within the red box from the figure; this feature is disabled by default).

### 4.3.3. Using J-Flash Lite for downloading

When using Jlink for debugging and firmware downloading, you can find JFlashLite.exe in the directory: GD32EmbeddedBuilder\_v1.4.14.29824\Tools\J-Link. Double-click to open JFlashLite and configure it as shown in [Figure 4-15 JFlashLite Configuration](#).

**Figure 4-15 JFlashLite Configuration**



Set the Target device to GD32VW553xxxx, choose the Target interface as JTAG (cJTAG can also be selected, but it is slower), and set the Speed to 9600 kHz. Then click OK.

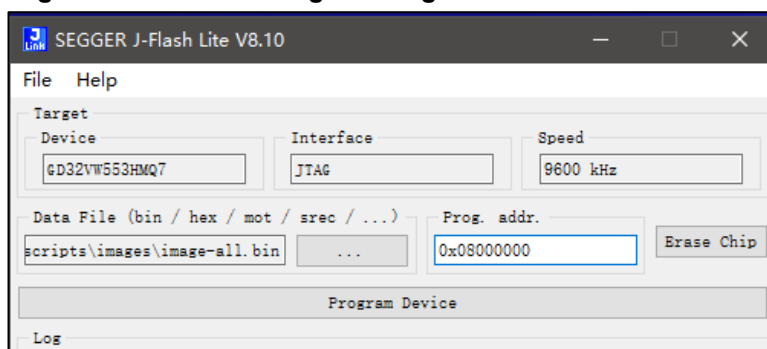
In the opened interface, as shown in [Figure 4-16 J-Flash Programming Interface](#), select the Data File as the compiled image-all.bin or image-ota.bin (stored in the directory GD32VW55x\_RELEASE\_V1.0.xx\scripts\images after compilation).

When selecting image-all.bin, set the Prog. Addr. on the right to 0x08000000.

When selecting image-ota.bin, set the Prog. Addr. on the right to 0x0800A000.

Once the settings are complete, click "Program Device" and wait for the progress bar to finish, which indicates the completion of the programming process.

**Figure 4-16 J-Flash Programming Interface**



## 4.4. Debugging

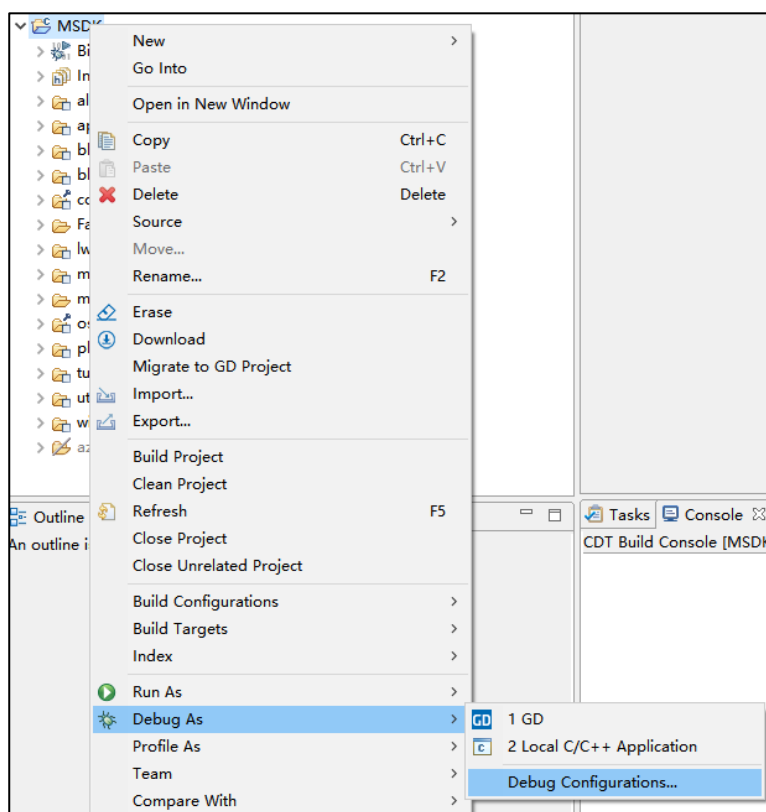
Currently, both the START development board and the EVAL development board feature onboard GDLink, and an external Jlink can also be used for debugging.

The debugging process is described below, and the default project configuration is msdk. If you need to switch to another project configuration, please refer to section [6.3Select different project configurations during debugging](#) for selecting different project configurations for debugging.

### 4.4.1. Debugging configuration

Right-click on the MSDK project and click Debug As->Debug Configurations, as shown in [Figure 4-17. Opening the Debug Configuration option](#).

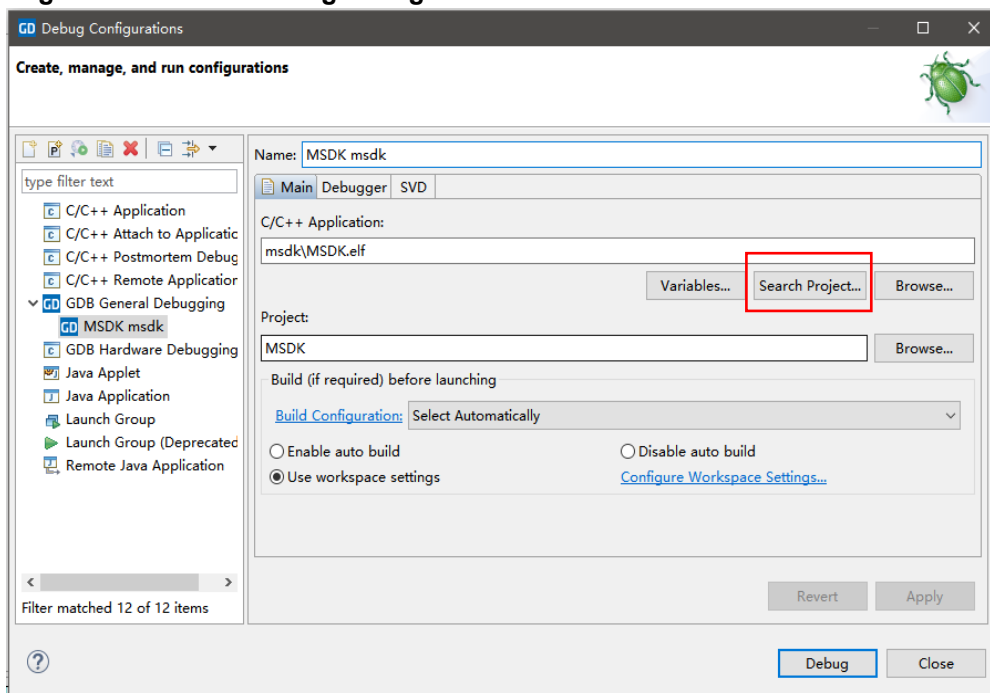
Figure 4-17. Opening the Debug Configuration option



Double-click "GDB General Debugging" on the left, and a Debug configuration will automatically be created, as shown in [Figure 4-18. MSDK debug configuration](#). Here, the c/c++ application field has automatically selected msdk\MSDK.elf, which points to the ELF file generated by the configuration to be debugged.

You can check or uncheck "Enable/Disable auto build" to choose whether or not to compile the project before debugging.

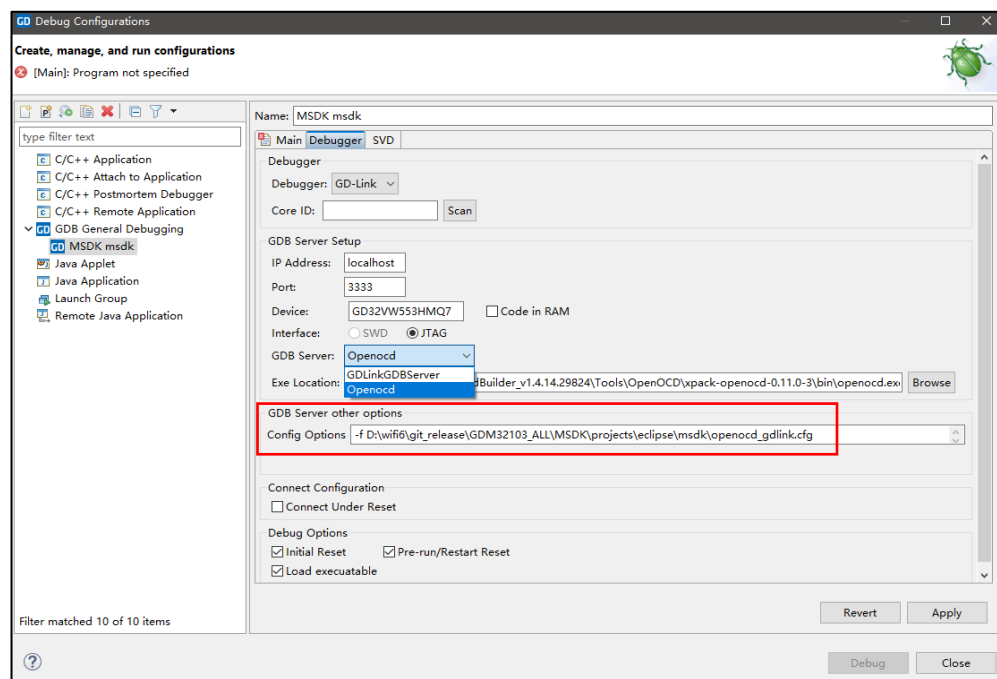
Figure 4-18. MSDK debug configuration



#### 4.4.2. Debugging using GDLink

As shown in [Figure 4-19. MSDK Debugging Configuration Interface with openocd](#), switch the GDB Server to OpenOCD in the Debugger interface. Additionally, specify the Config Options within the red box as shown in the figure. Afterward, click "Debug" to start debugging. The debugging interface is displayed in [Figure 4-20. MSDK debug interface](#).

Figure 4-19. MSDK Debugging Configuration Interface with openocd



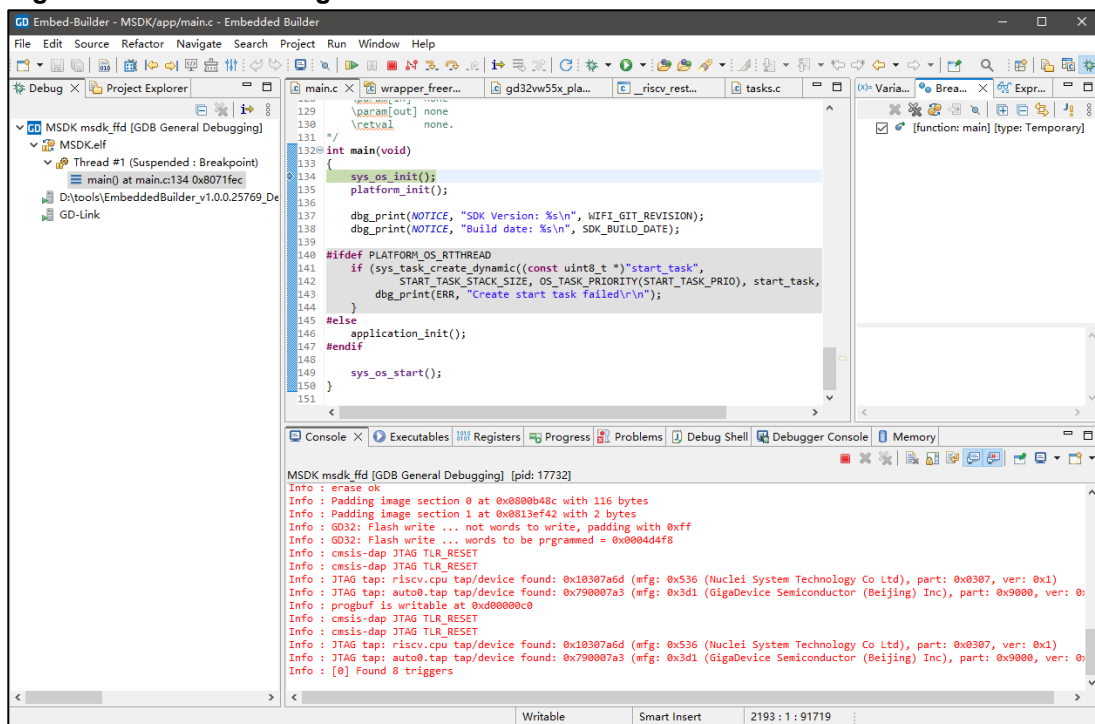
In [Figure 4-19. MSDK Debugging Configuration Interface with openocd](#), selecting "Initial

Reset" and "Pre-run/Restart Reset" in the Debug Options will reset the chip at the start of debugging. Choosing "Load executable" will flash the firmware once before debugging begins.

### 4.4.3. Debugging using Jlink

The GD32VW553 supports JTAG and cJTAG debugging. First, connect the pins of the JLink debugger to the GD32VW553 JTAG pins. Next, replace the cfg file within the red box in the [Figure 4-19. MSDK Debugging Configuration Interface with openocd](#) with "openocd\_jlink.cfg" (this file is located in the directory: MSDK\projects\eclipse\msdk). Then, click "Debug" to enter the debugging process. If driver issues arise during JLink debugging, please refer to [6.4 JLink Driver Replacement](#).

**Figure 4-20. MSDK debug interface**



## 5. SEGGER Embedded Studio IDE project

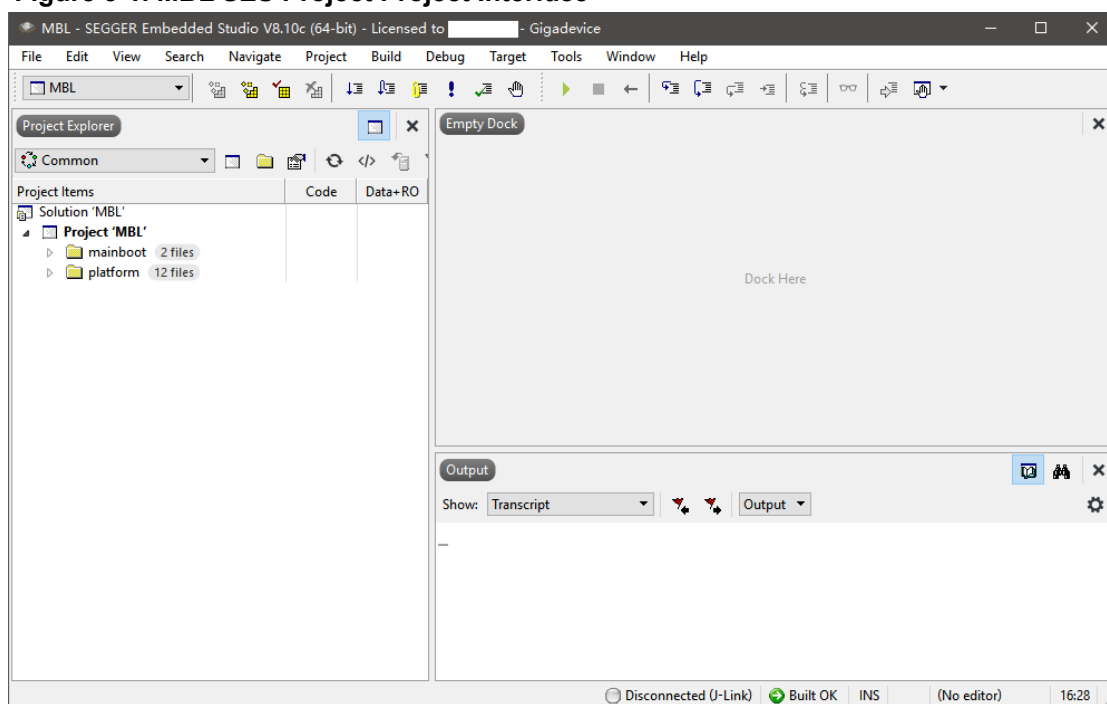
This chapter introduces how to compile and debug the SDK under SEGGER Embedded Studio IDE.

### 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](#).

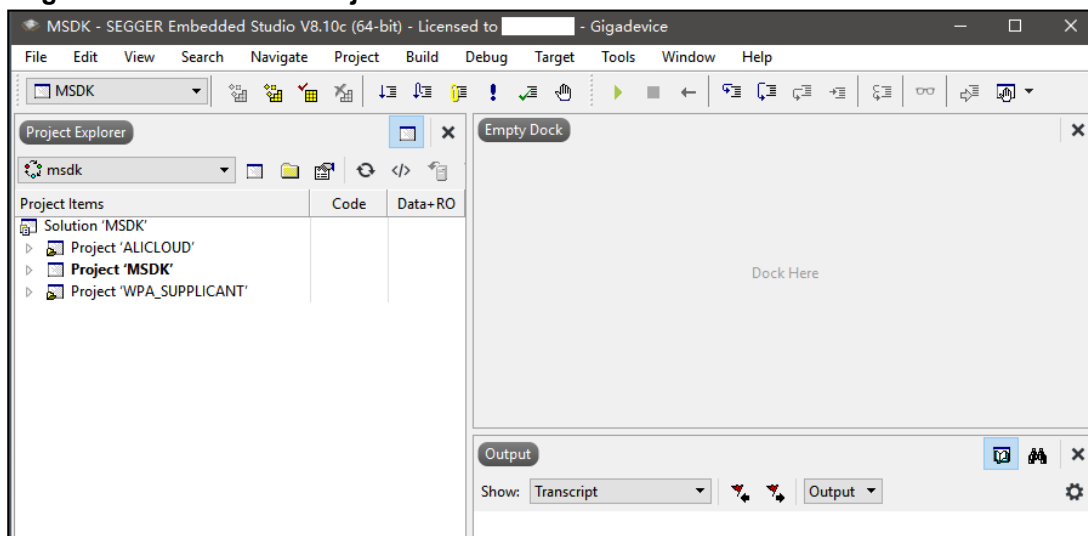
**Figure 5-1. MBL SES Project Project Interface**



#### ■ 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

### ■ SES build tool configuration

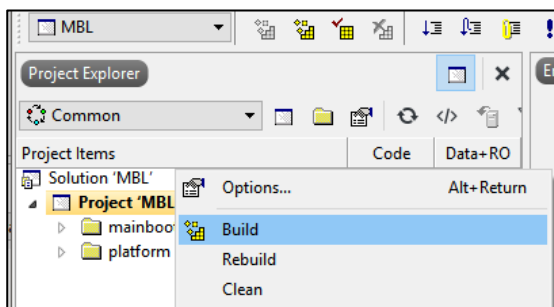
SES compiles the GD32VW55x project using the riscv32-none-elf toolchain by default. In order to better support the extended instruction set of riscv, it needs to be compiled using the nuclei toolchain: riscv-nuclei-elf. The compilation tool can be obtained by contacting sales or FAE. The details of the 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

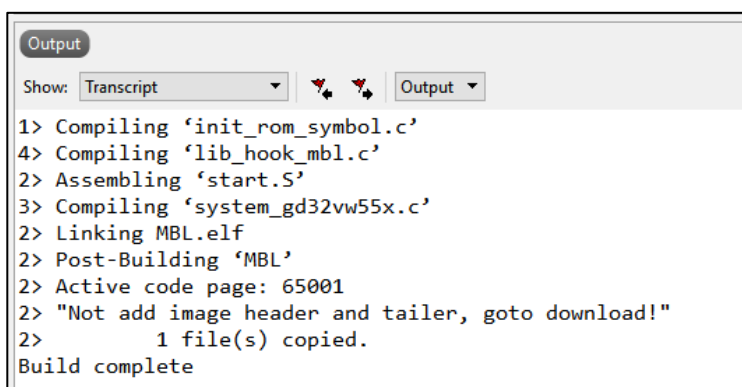
Segger_IDE > gcc > riscv-nuclei-elf		
名称	修改日期	类型
bin	2024/3/18 17:59	文件夹
include	2024/3/19 9:40	文件夹
lib	2024/7/12 15:21	文件夹

### ■ Compile the MBL project

Right-click the project and click build to build 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**

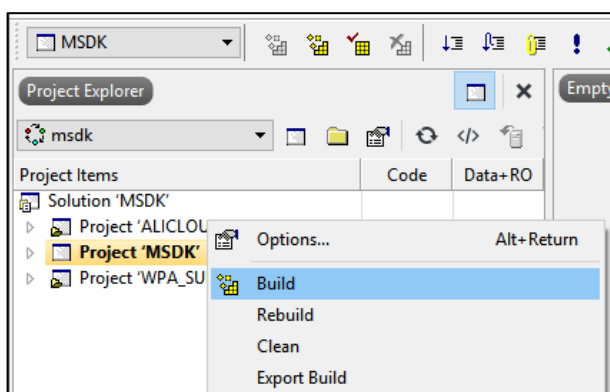
The compilation result is as shown in [Figure 5-5. MBL compilation result](#).

**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

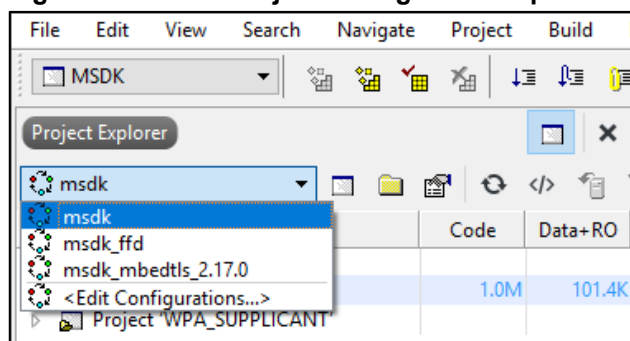
Right-click Project 'MSDK' and click Build, as shown in [Figure 5-6. Compile MSDK project](#).

**Figure 5-6. Compile MSDK project**

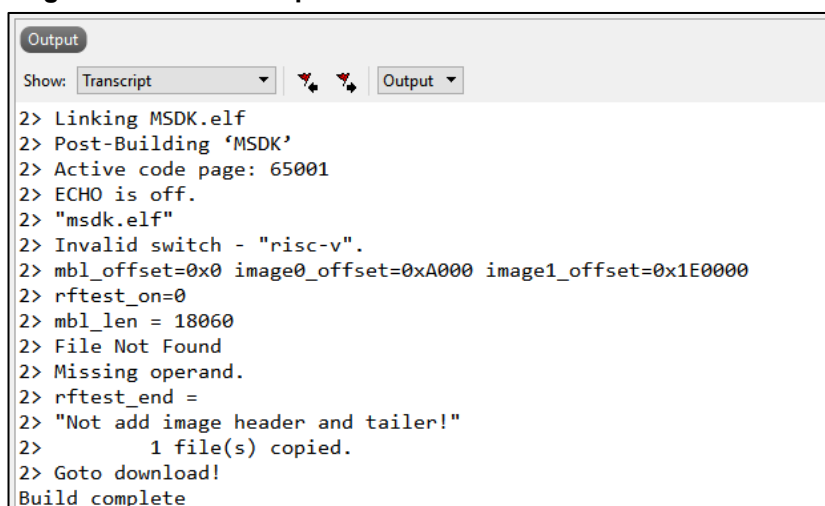
#### ■ Configuration selection of MSDK

MSDK configuration switch as shown in [Figure 5-7 MSDK Project Configuration Options](#). MSDK SES project only supports msdk, msdk\_ffd and msdk\_mbedtls\_2.17.0; if you need to use the configuration of msdk\_threadx, msdk\_ffd\_threadx 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**

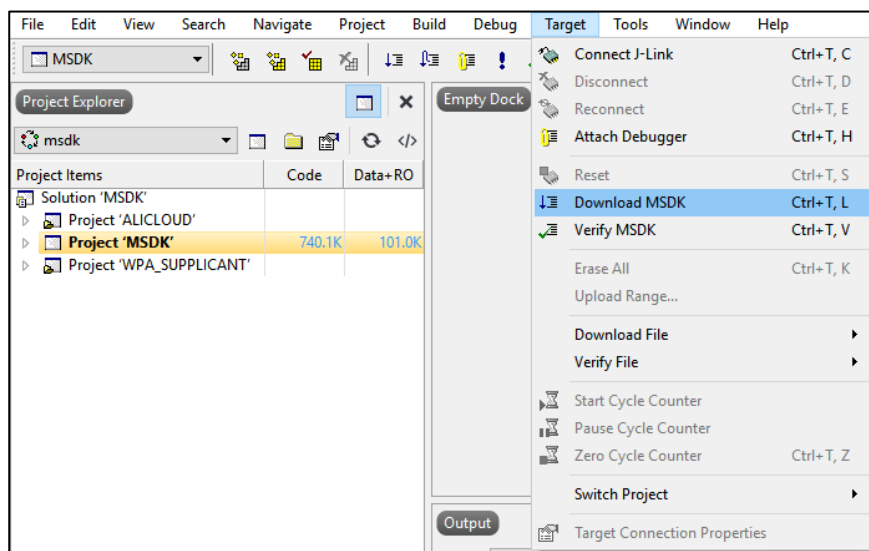
名称	修改日期	类型	大小
image-all.bin	2024/7/11 14:26	BIN 文件	788 KB
image-ota.bin	2024/7/11 14:26	BIN 文件	748 KB
mbl.bin	2024/7/11 14:19	BIN 文件	17 KB

## 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**



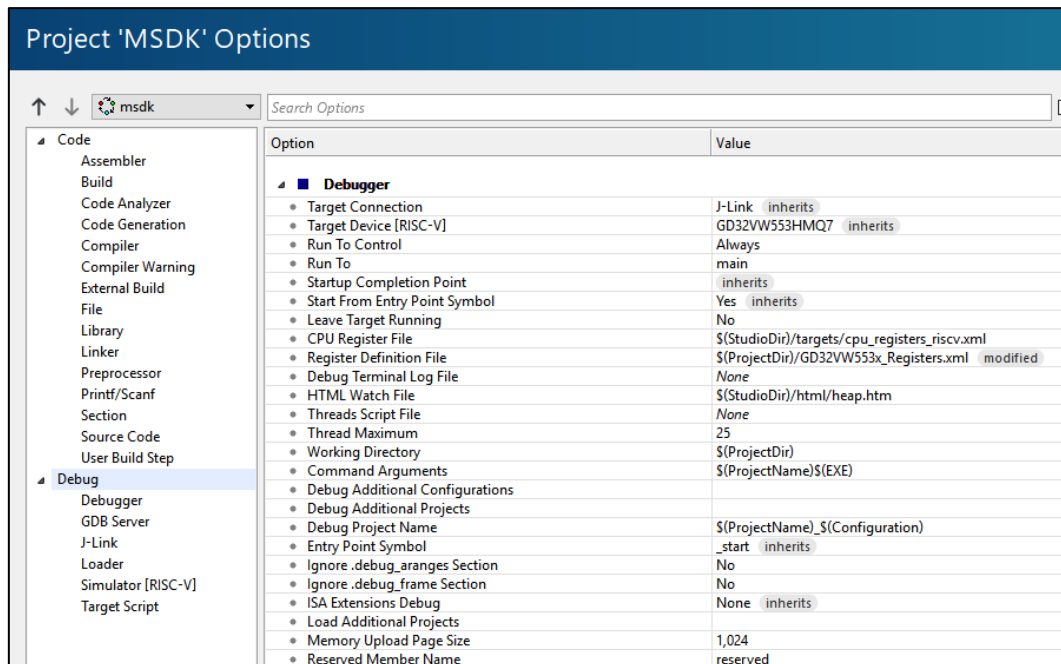
## 5.4. Debugging

### ■ 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 JLink under the Debug option, as shown in [Figure 5-11. MSDK SES Project Configuration Interface](#).

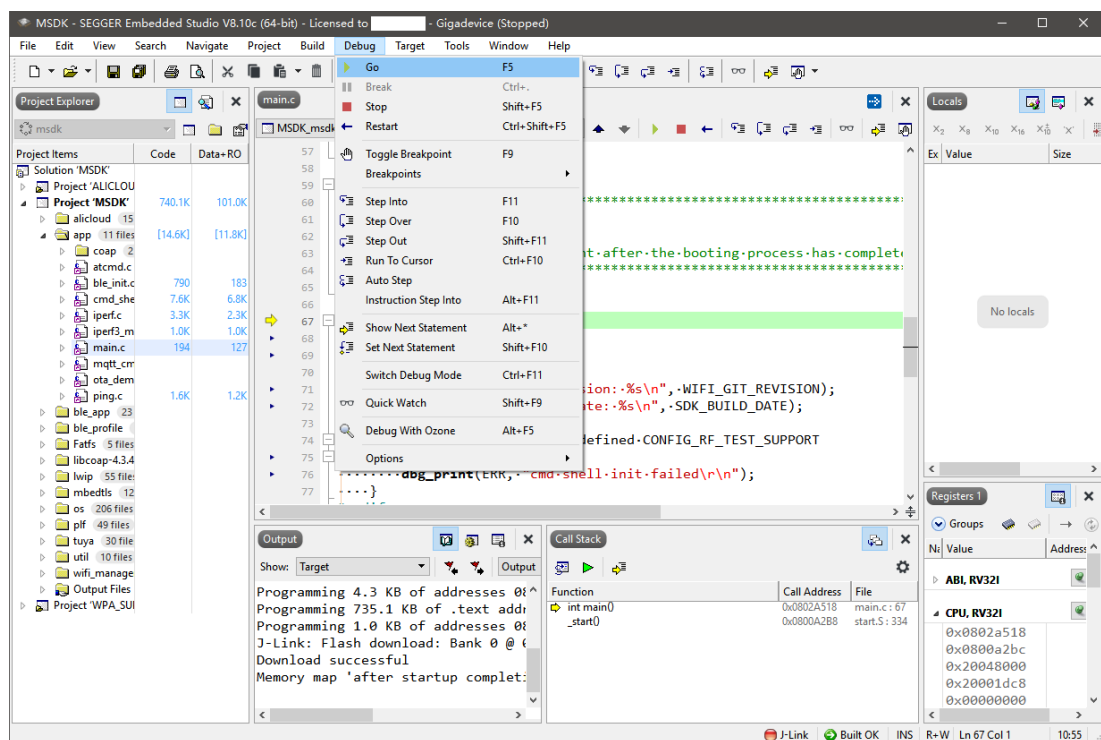
**Figure 5-11. MSDK SES Project Configuration Interface**



### ■ 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\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 example:

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

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

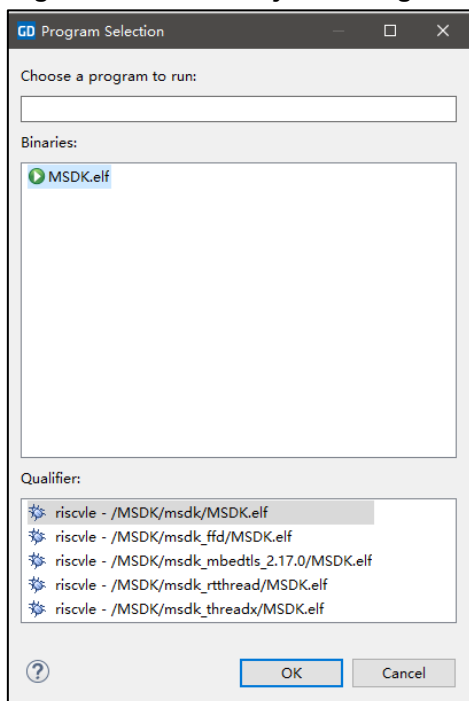
```
KEEP (*tasks.o* (.text.xTaskIncrementTick))
```

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

### 6.3. Select different project configurations during debugging

The MSDK project supports multiple configurations (refer to [3.2.6 Configuration Selection](#)). During debugging, you need to select the appropriate configuration.

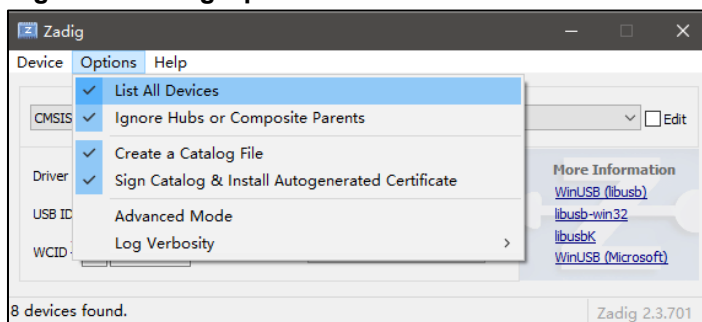
The specific operation is as follows: Click the "Search Project" button within the red box shown in [Figure 4-18. MSDK debug configuration](#). This opens the interface shown in [Figure 6-1 Select Project Configuration for Debugging](#). Double-click the configuration you want to debug within the Qualifier box below (note that you need to compile the corresponding configuration first for the option to appear).

**Figure 6-1 Select Project Configuration for Debugging**

## 6.4. JLink Driver Replacement

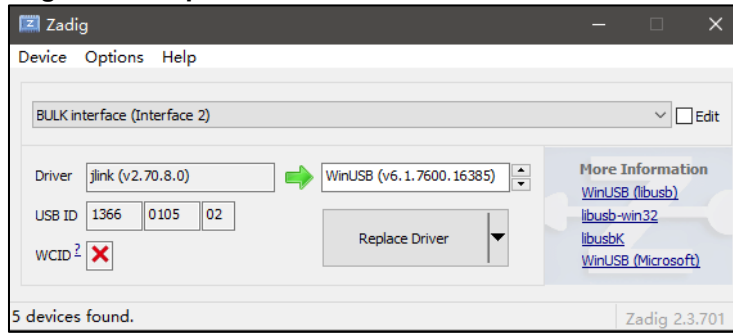
When using OpenOCD+JLink debugging in an Embedded Builder project, if encountering the issue "Error: No J-Link device found," JLink drivers need to be replaced. The steps are as follows:

1. Use administrator privileges to open the `zadig.exe` file (official website: <https://zadig.akeo.ie>). Click "Options" and check "List All Devices," as shown in [Figure 6-2 Zadig Options Selection](#).

**Figure 6-2 Zadig Options Selection**

2. Select "JLink devices" from the dropdown menu, as shown in [Figure 6-3 Replace JLink Driver](#), where the BULK interface is displayed. Click "Replace Driver" to replace the JLink driver with WinUSB.

**Figure 6-3 Replace JLink Driver**



3. After the replacement is complete, unplug and replug the JLink device. Then use JLink for debugging, and there will be no driver issues.

## 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
1.3a	Update the development board images and add firmware downloads	May.15.2025
1.3b	Update the development board images and add description related to the START development board	Feb.26.2026

## 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.