# GigaDevice Semiconductor Inc.

# GD32VW553 Quick Development Guide

# Application Note AN154

Revision 1.3a

(May 2025)



# **Table of Contents**

Table of	of Contents	2
List of	Figures	4
List of	Tables	6
1. Ir	ntroduction to development board	7
1.1. 1.1. 1.1.	Picture of real development board 1. The START development board 2. The EVAL development board	<b>7</b> 7 7
1.2.	Boot mode	9
1.3.	Debugger interface	9
1.4.	Download interface1	0
1.5.	Viewing log1	0
2. B	Building development environment1	1
2.1.	Installation of GD32 Embedded Builder1	1
2.2.	Installation of SEGGER Embedded Studio IDE1	1
3. V	Vhat developers must know1	2
3.1.	SDK execution program group12	2
<b>3.2.</b> 3.2. 3.2. 3.2. 3.2. 3.2. 3.2.	SDK configuration.11. Configuration of wireless module.12. SRAM layout.13. FLASH layout.14. Firmware version No.15. APP configuration16. Configuration Selection1	<b>2</b> 3 3 4 5
3.3.	Correct log example1	6
4. G	D32 Embedded Builder IDE project1	7
4.1.	Opening the project group1	7
4.2.	Compilation2	D
<b>4.3.</b> 4.3. 4.3. 4.3.	Download firmware       2         1. USB Drive Copy.       2         2. Use afterbuild.bat for downloading.       2         3. Using J-Flash Lite for downloading.       2         Debugging       2	<b>3</b> 3 3 4 5
		-



5.	SEGGER Embedded Studio IDE project	
5.1	. Open projects	28
5.2	. Compilation	29
5.3	. Download firmware	31
5.4	. Debugging	32
6.	FAQ	34
6.1	No image error	34
6.2	. Code running in SRAM	34
6.3	. How to select different project configurations during debugging	34
7.	Revision history	37



# **List of Figures**

Figure 1-1. The picture of the START development board7	,
Figure 1-2. The picture of the EVAL development board	;
Figure 1-3. Development Board Type Configuration9	)
Figure 1-4. List of devices and drivers10	)
Figure 1-5. Configuration of serial port10	)
Figure 2-1 The Directory Structure of GD32 Embedded Builder11	
Figure 3-1. Boot process12	2
Figure 3-2. Configuration of wireless module12	?
Figure 3-3. SRAM layout13	;
Figure 3-4. FLASH la yout13	;
Figure 3-5. Firmware version No14	ŀ
Figure 3-6 BLE library selection14	ŀ
Figure 3-7. Project boot information16	;
Figure 4-1. SDK directory17	,
Figure 4-2. Starting GD32 Embedded Builder IDE17	,
Figure 4-3. Open Projects from file System18	;
Figure 4-4. Selecting MBL project path	;
Figure 4-5. MBL project interface	)
Figure 4-6. Selecting MSDK project path19	)
Figure 4-7. MSDK and MBL project interfaces	)
Figure 4-8. Properties of the project	)
Figure 4-9. Compiling the MBL project21	ļ
Figure 4-10. MBL compilation result21	
Figure 4-11. target configuration selection	2
Figure 4-12. MSDK compilation result22	?



## AN154 GD32VW553 Quick Development Guide

Figure 4-13. Images output	23
Figure 4-14 Configure image automatic downloading	23
Figure 4-15 JFlashLite Configuration	24
Figure 4-16 J-Flash Programming Interface	24
Figure 4-17. Opening the Debug Configuration option	25
Figure 4-18. MSDK debug configuration	26
Figure 4-19. MSDK Debugging Configuration Interface with openocd	26
Figure 4-20. MSDK debug interface	27
Figure 5-1. MBL SES Project Project Interface	28
Figure 5-2. MSDK SES Project Interface	29
Figure 5-3. nuclei toolchain content	29
Figure 5-4. Compiling the MBL project	30
Figure 5-5. MBL compilation result	30
Figure 5-6. Compile MSDK project	30
Figure 5-7 MSDK Project Configuration Options	31
Figure 5-8 MSDK compilation result	31
Figure 5-9. Images output	31
Fogure 5-10 SES IDE image download	32
Figure 5-11. MSDK SES Project Configuration Interface	33
Figure 5-12. SES IDE Debug Interface	33
Figure 6-1 Select Project Configuration for Debugging	35



# **List of Tables**

Table 1-1. Bootmode	9
Table 7-1. Revision history	37



## 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);
- View log (UART);
- Debugger interface (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.



## AN154 GD32VW553 Quick Development Guide



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_EVAL1
#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 <u>Table 1-1. Boot mode</u>. For more instructions on the boot mode, please refer to the document "GD32VW55x\_User\_Manual".

EFBOOTL K	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

#### Table 1-1. Boot mode

## 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 1-4. 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 1-4. 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.

Serial Settings					
COM:	COM21	~			
Baudrate:	115200	~			
Data Bits:	8	$\sim$			
Parity:	None	~			
Stop Bits:	1	~			
Open					

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: <u>https://gd32mcu.com/cn/download</u> to download. The uncompress downloaded files is as <u>Figure 2-1 The Directory Structure of GD32 Embedded Builder</u> 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: <u>https://wiki.segger.com/GD32V</u> 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

#define RE MBL DATA START 0x300 #define RE\_IMG\_DATA\_START 0x200

For the planning of SRAM space in each executable program segment, refer to the .ld file under the corresponding project, such MBL\project\eclipse\mbl.ld and as MSDK\plf\riscv\env\qd32wv55x.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	I-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	<b>0x8000</b> · · · · · /* · !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 MBL\project\eclipse\mbl.ld project, such as and MSDK\plf\riscv\env\gd32w55x.ld.

#### 3.2.4. Firmware version No.

The configuration file is GD32VW55x\_RELEASE\config\config\_gdm32.h. Modify the following 13



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.



#### 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 peripharal and server
#define BLE_LIB_MAX 1 //add central and client usage
#define CONFIG_BLE_LIB
```

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 3.6.2, 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 + MbesTLS 3.6.2

• msdk\_ffd

FreeRTOS + wpa\_supplicant.a + libble\_max.a + MbesTLS 3.6.2

msdk\_mbedtls\_2.17.0

FreeRTOS + Libwpas.a + libble.a + ROM MbesTLS 2.17.0

msdk\_rtthread

RTThread + Libwpas.a + libble.a + MbesTLS 3.6.2



• msdk\_threadx

Threadx + Libwpas.a + libble.a + MbesTLS 3.6.2

For details on how to make configuration selection for actual use, see the Compiling MSDK Projects section in subsection <u>4.2Compilation</u>.

## 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 <u>6FA Q</u>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 config docs MBL MSDK ROM-EXPORT scripts release\_notes.txt

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

GD Eclipse Launcher	×				
Select a directory as workspace					
Workspace: D:\risc-v\GD32VW55x_RELEASE	✓ Browse				
Use this as the default and do not ask again					
Recent Workspaces					
	Launch Cancel				

#### 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*.



гıg	ure	4-3. U	pen Pro	Djects fr	OM II	<u>e Sys</u>
GD E	mbed-	Builder -	Embedded	Builder		
File	Edit	Source	Refactor	Navigate	Search	Project
	New				Alt+Sh	ift+N >
	Open	File				
	Open	Projects	from File S	/stem		
	Recen	t Files				>
	Close				Ct	rl+W
	Close	All			Ctrl+Sh	ft+W
e	Save				C	trl+S
	Save A	4s				
R	Save A	All			Ctrl+Sł	nift+S
·	Rever	t				
	Move					
	Renan	ne				F2
5	Refres	sh				F5
	Conve	ert Line De	elimiters To	)		>
8	Print				C	trl+P
è	Impor	rt				
4	Expor	t				
	Prope	rties			Alt+	Enter
	Switch	Worksp	ace			>
	Resta	rt				
	Exit					

Figure 4-3. Open Projects from file System

Select the project path GD32VW55x\_RELEASE\MBL\project\eclipse, as shown in *Figure* <u>4-4. Selecting MBL project path</u>, and click Finish.

Figure 4-4. Selecting MBL project path

GD Import Projec	ts from File System or Archive		_		×	
Import Projects f This wizard analy	Import Projects from File System or Archive This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.					
Import source:	D:\risc-v\GD32VW55x_RELEASE_V1.0.3\MBL\project\eclipse	~	Directory	Archive		
type filter text			Sele	ct All		
Folder	Import as		Decel	oct All		
🗹 eclipse	Eclipse project		Deser	ect All		
			1 of 1 selected			
			Hide already	open pro	jects	
Close newly in	nported projects upon completion		,,			
Use installed pro	<u>pject configurators</u> to:					
Search for ne	sted projects					
Detect and co	intigure project natures					
Working sets						
Add projec	t to working sets			New		
Working sets:			$\sim$	Select		
		Show	other specialized	import wi	zards	
?	< Back	Next >	Finish	Canc	el	

Close the welcome interface, and the MBL project is shown as <u>Figure 4-5. MBL project</u> <u>interface</u> shows.



Figure 4-5. MBL project interface

File Edit Navigate Search Project W	/indow Help
i 🗗 ▼ 🔒 💿 📄 💼 🕪 ⇔ 罕 📩 N	t C Ø ▼ 🖢 ▼ 🖗 ▼ 🏷 ↔ ▼
Project Explorer 🛛 🗖 🗖	
□	
∽ 😤 MBL	
> 🔊 Includes	
> 🔁 mainboot	
> 🔁 platform	
🛃 mbl.ld	

■ 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* <u>MSDK project path</u>, and click Finish.

Figure 4-6. Selecting MSDK project path

GD Import Project	ts from File System or Archive		— 🗆 X
Import Projects f This wizard analy	rom File System or Archive zes the content of your folder or archive file to find projects and imp	ort them in the IDE.	
Import source:	D:\risc-v\GD32VW55x_RELEASE_V1.0.3\MSDK\projects\eclipse\msdk		V Directory Archive
type filter text			Select All
Folder		Import as Eclipse project	Deselect All
<b>V</b> mout		Lenpoo project	1 of 1 selected
☐ Close newly ir Use <u>installed pro</u> ☑ Search for new ☑ Detect and co	nported projects upon completion <u>oject configurators</u> to: sted projects nfigure project natures		☐ Hide already open projects
Working sets	t to working sets		New V Select
		Sho	ow other specialized import wizards
?		< Back Next >	Finish Cancel

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

-	
File Edit Navigate Search Project W	indow Help
	C   <b>♦</b> •   b • 17 • 17 • 17 • 17 •
မြံ Project Explorer 🛛 📃 🗖	
₽	
✓ 👺 MBL	
> 🔊 Includes	
> 🔁 mainboot	
> 🔁 platform	
🎽 mbl.ld	
✓	
> 🔊 Includes	
> 🔁 alicloud	
> 🔁 app	
> 🔁 ble_app	
> 🔁 ble_profile	
> 🚰 coap	
> 📂 FatFS	
> 🔁 lwip	
> 🔁 mbedtls	
> 🚰 os	
> 🔁 plf	
> 🔁 tuya	
> 🔁 util	
> 🔁 wifi_manager	
> 📂 azure	

## 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

GD Properties for MBL - D	×
type filter text Settings	) <b></b>
<ul> <li>&gt; Resource</li> <li>&gt; C/C++ Build</li> <li>Build Variables</li> <li>Environment</li> <li>Logging</li> <li>Settings</li> <li>Tool Chain Editor</li> <li>XL C/C++ Compiler</li> <li>&gt; C/C++ General</li> <li>Project Natures</li> <li>Project References</li> <li>Refactoring History</li> <li>Run/Debug Settings</li> </ul>	∩5 ∧ 
Apply and Close         Can	:el

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

× 125 №	ISDK				
> 🖻 > 🖌		New > Go Into			
> 🔄		Open in New Window			
> @ > @		Copy Ctrl+C			
> 6	Ē	Paste Ctrl+V			
> 6	×	Delete Delete			
> 🔓		Source >			
> 🖌		Move			
> @		Rename F2			
	Q	Erase			
* <	۲	Download			
		Migrate to GD Project			
	2	Import			
	4	Export			
		Build Project			
		Clean Project			
	8	Refresh F5			
		Close Project			
E Outli		Close Unrelated Project			
An outlin		Build Configurations	Set Active >	~	1 msdk (default: msdk + freertos + mbedtls-3.6.2)
		Build Targets >	Manage		2 msdk_ffd (full function device)
		Index >	Build All		3 msdk_mbedtls_2.17.0 (msdk + freertos + mbedtls-2.17.0(in ROM))
	0	Run As >	Clean All		4 msdk_rtthread (msdk with rtthread os)
	*	Debug As >	Build Selected		5 msdk_threadx (msdk with threadx os)

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

🖹 Problems 🧟 Tasks 📮 Console 🛛 🔲 Properties 🔫 Progress
CDT Build Console [MSDK]
Script processing completed.
<pre>Invoking: GD RISC-V MCU Flash Image(Hex) riscv-nuclei-elf-objcopy -0 ihex "MSDK.elf" "MSDK.hex" Invoking: GD RISC-V MCU Flash Image(Bin) riscv-nuclei-elf-objcopy -0 binary "MSDK.elf" "MSDK.bin" Invoking: GD RISC-V MCU Listing Invoking: GD RISC-V MCU Print Size Finished building: MSDK.hex riscv-nuclei-elf-objdumpsourceall-headersdemangleline-numberswide "MSDK.elf" &gt; "MSDK.lst" Finished building: MSDK.bin riscv-nuclei-elf-sizeformat=berkeley "MSDK.elf"</pre>
text data bss dec hex filename 1231668 1144 103276 1336088 146318 MSDK.elf
Finished building: MSDK.siz

Images generated by SDK

After MSDK is compiled, it will call MSDK\projects\ image\_afterbuild.bat to generate imageota.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
rigure	4-13.	iiiiayeə	υμιμαι

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



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

SEGGER J-Flash Lite V8.10	×
Target device GD32VW553HMQ7	
Target interface Speed	∨ kHz
Flash banks	
BaseAddr Name Loader	
⊻OxO8000000 Flash Default ▼	
	OK

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.

🔜 SEGGER J-Flash Lite V	3.10				×
File Help					
Target Device CD32VW553HMQ7 Data File (bin / hex / m scripts\images\image-all.	Interface JTAG Not / srec /) bin	Prog. add	Speed 9600 kHz dr. 0	Erase	e Chip
	Program De	vice			
Log					

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 <u>6.3Select different</u> 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*.

	_					
<ul> <li>✓ MSCC*</li> <li>&gt; 2000 Bit</li> <li>&gt; 2000 Bit</li></ul>		New Go Into Open in New Window Copy Paste Delete Source Move Rename Erase Download Migrate to GD Project Import Export Build Project Clean Project Refresh Close Project	> Ctrl+C Ctrl+V Delete > F2			Tasks ♀ Console ☎ CDT Build Console [MSDK
- Outline	<u>ଛ</u> ୀ	Refresh	F5	F		🔊 Tasks 🗉 Console 🕅
	~	Close Project		⊢		CDT Build Console (MSDk
An outline i		Close Unrelated Project				
		Ruild Configurations	、 、			
		Build Targets	· · · · · · · · · · · · · · · · · · ·			
		Index	\$			
	_					
	0	Run As	>	<u> </u>		
	*	Debug As	>	GD	1 GD	
		Profile As	>	C	2 Local C	/C++ Application
		Team	>		Debug Co	onfigurations
		Compare With	>	-		

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

GD Debug Configurations			– 🗆 X
Create, manage, and run configur	itions		Ť
Image: Second	Name: MSDK msdk Main Debugger SVD C/C++ Application: msdk\MSDK.elf Project: MSDK Build (if required) before launching <u>Build Configuration</u> : Select Automatica O Enable auto build	Variables Search Project Ily O Disable auto build Configure Workspace Settings	Browse
< >> Filter matched 12 of 12 items		Revert	Apply
?		Debug	Close

## 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

GD Debug Configurations	– 🗆 X
Create, manage, and run configurations () [Main]: Program not specified	
Image: Solution of the second seco	Name:       MsDK msdk            Main Debugger SVD        Debugger SVD          Debugger CD-Link ✓        Core ID:
Filter matched 10 of 10 items	Revert Apply
?	Debug Close

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 <u>6.4 JLink Driver Replacement</u>.

Figure 4-20. MSDK debug interface

GD Embed-Builder - MSDK/app/main.c - Embedded Builder — 🗆 🗙							
File Edit Source Refactor Navigate Search Project Run Window Help							
📑 🕶 🖃 👒 🛤 🛤 🕪 🤤 📾 🚻 🗇 🖓	> i 🗳 i 🔌 i 🕨 💷 👪 🕉 Le	i> 🗟 🗷   C   🎋 🗸	• 🔾 • 😂 😂 🛷 •	• 刘 🖄 • 🖗	• * 🗘 🗘 🔶 •	• 🗢 • 📑 🛃 🔍 📑	🏘 💀 👌
🎋 Debug 🗙 🍋 Project Explorer 🗧 🗖	💽 main.c 🗙 🛍 wrapper_freer	🖻 gd32vw55x_pla	_riscv_rest	🚺 tasks.c	🗖 🗖 🔍 (x)= Varia	a 🌯 Brea 🗙 🙀 Expr.	0
MSDK msdk ffd [GDB General Debugging]     WSDKelf     MSDK-elf     MSDK-elf     MSDK-elf     maing at mainc:1340x8071fee     D:\tools\EmbeddedBuilder_v1.0.0.25769_De     GD-Link	Josephilist         Josephilist           129         Vyggraff out Jone           130         Vcttval           131         r/           132         vgraff out Jone           133         r           134         sys_os_init();           135         platform_init();           136         dbg_print(NOTICE, "SO           139         mifdef PLATFORM_OS_RTHREE           141         if (sys_task_create_d)           143         dbg_print(TARS, "G)	<pre></pre>	I_GIT_REVISION); BUILD_DATE); ")"start_task", DRITY(START_TASK_PR Gd/vhu");	NIO), start_tas	* 3	🤹 🏶 🐨 🔌   🖪 🤤 🦉	<mark>≱   J≗</mark> 8 nporary]
	144 } 145 melson_init(); 146 application_init(); 147 mendif 148 sys_os_start(); 159 } 151 <				• • •		~
	Console × 🚺 Executables	egisters 🖷 Progress 🚦	Problems 🗓 Debug	Shell 🙀 Debugg	ger Console 📋 N	Memory	- 0
	MSDK msdk_ffd [GDB General Debugg	ging] [pid: 17732]			<b>= x</b> %	B. 🔝 🖻 💭 🛃 🖬	] • 📑 •
	Into: erase ok Info: Padding inage section 0 a Info: Padding inage section 1 a Info: G032: Flash write not Info: G032: Flash write not Info: cnsis-dap JTAG TLR_RESET Info: cnsis-dap JTAG TLR_RESET Info: cnsis-dap JTAG TLR_RESET Info: jTAG tap: sato0.cop tap/ Info: cnsis-dap JTAG TLR_RESET Info: cnsis-dap JTAG TLR_RESET Info: cnsis-dap JTAG TLR_RESET Info: jTAG tap: sato0.tap tap/ Info: jFlaG tap. sato0.tap tap/ Info: jFlaG tap/ Inf	<pre>xt 0x0800b48c with 11 t 0x0812ef42 with 21 words to write, pad ds to be prgrammed = levice found: 0x10307. levice found: 0x79000 d000000 levice found: 0x10307. levice found: 0x10307.</pre>	5 bytes sytes ing with 0xff 0x9004d4f8 a6d (mfg: 0x536 (Nu a7a3 (mfg: 0x3d1 (Gi a6d (mfg: 0x536 (Nu 7a3 (mfg: 0x3d1 (Gi	iclei System Tec gaDevice Semico iclei System Tec gaDevice Semico	:hnology Co Ltd onductor (Beiji :hnology Co Ltd onductor (Beiji	l), part: 0x0307, ver: 0 ng) Inc), part: 0x9000, l), part: 0x0307, ver: 0 ng) Inc), part: 0x9000,	<pre>x1) ver: 0; x1) ver: 0; v</pre>
< >>	<						>
		Writable	Smart Insert	2193 : 1 : 91	719		



## 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*.

· .gu. • • · · · = •=• · · •j••• · · •	
MBL - SEGGER Embedded Studio V8.10c (64-bit) - Licensed	to - Gigadevice - X
File Edit View Search Navigate Project Build	Debug Target Tools Window Help
🗆 MBL 🗾 👻 🖓 🎽 🗐 🕅	
Project Explorer	Empty Dock X
💱 Common 🔹 🗖 🗋 😭 🚸 🌾 👕	•
Project Items Code Data+RO	
Solution 'MBL'	
Imainboot 2 files	
platform 12 files	Dock Here
	Output 🔯 🏘 🗙
	Show: Transcript 🔹 🍾 🐧 Output 💌
	🔵 Disconnected (J-Link) 📀 Built OK 🛛 INS 🔹 (No editor) 16:28 📑

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 <u>Figure 5-2. MSDK SES</u> <u>Project Interface</u> shown.



# AN154 GD32VW553 Quick Development Guide

#### Figure 5-2. MSDK SES Project Interface

		1.4	-	o! 1						_	~
MSDK - SEGGER Embedded Studio V	/8.10c (64-bit) - Licen	sed to		- Gigadev	ice				—		×
File Edit View Search Navigate	Project Build	Debug	Target	Tools	Window	/ Help					
🗖 MSDK 🔻 📲 🐐		₽!	⊿ 🖑		←	দি⊒ [⊒ি	Ţ⊒ →⊒  ξ	1 00	₫	•	
Project Explorer		Emp	ty Dock								×
🖏 msdk 🔻 🔟 🧰	😭 🗘 🗘 👘										
Project Items	Code Data+RO										
Solution 'MSDK'  Solution 'MSDK'  Figure 'ALICLOUD'  Figure 'MSDK'  Figure 'MSAK'  Figure 'WPA_SUPPLICANT'						Dock H	lere				
		Outp	out				-			10 M	×
		Show	r: Iranscr	ipt	•	74 74	Output 🔻				-12

## 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-nucl	ei-elf	
名称	修改日期	类型
bin include	2024/3/18 17:59 2024/3/19 9:40	文件夹 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

MBL		- 😨 😼 Y	in %a   ↓	1 <b>1</b> 1 <b>(</b> 1 1
Project Explorer				
🕄 Common		-	🗗 😯	
Project Items			Code	Data+RO
Solution 'MBL'	r SP	Options		Alt+Return
Project 'MBL		optionsin		
🔋 🔋 📄 mainboo	<b>*</b>	Build		
D Datform		Rebuild		
		Clean		

The compilation result is as shown in *Figure 5-5. MBL compilation result*.

Figure 5-5. MBL compilation result

Output
Show: Transcript 🔹 🍾 🐪 Output 👻
1> Compiling 'init_rom_symbol.c'
<pre>4&gt; Compiling 'lib_hook_mbl.c'</pre>
<pre>2&gt; Assembling 'start.S'</pre>
<pre>3&gt; Compiling 'system_gd32vw55x.c'</pre>
2> Linking MBL.elf
<pre>2&gt; Post-Building 'MBL'</pre>
<pre>2&gt; Active code page: 65001</pre>
2> "Not add image header and tailer, goto download!"
<pre>2&gt; 1 file(s) copied.</pre>
Build complete

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

MSDK	•	°a 🐮 🕯	<u> </u>	ta ta Da	1
Project Explorer					Empty
🛟 msdk		• 🗈 🗀	<b>0</b>		
Project Items			Code	Data+RO	
Solution 'MSDK'					
<ul> <li>Project 'ALICLOU</li> <li>Project 'MSDK'</li> </ul>	P	Options		Alt+Re	turn
Project 'WPA_SU	<b>%</b>	Build			
		Rebuild			
		Clean			
		Export Build			

■ 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

File	Edit	View	Search	Na	vigate	Proje	ct	Build	[
	MSDK		•	°⊒ <b>*</b>	i 1	<b>1</b>	ţ⊒	₽≣	)je
Proje	ct Explor	er							×
្ទី៣	nsdk		-			<b>8</b> 0	÷ :</td <td></td> <td>1</td>		1
l i i	nsdk acdk ffd					Code	D	)ata+F	0
l 🖑 n	nsdk_mb	edtls_2.1	7.0						
<u>،</u> د	Edit Con	figuratio	ns>			1.0	M	101	.4K
	] Projec	t 'WPA_S	UPPLICA	NT'					

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 imageota.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 *Fogure 5-10 SES IDE image download*.

File Edit View Search Navigate Project Bui	ild Debug	Targe	t Tools Window	Help
🗖 MSDK 👻 🐜 🖌 🌾 🛵 🚛 🕼	i (iz 🕴 🦯	۵ 🔇	Connect J-Link	Ctrl+T, C
	~ ~ ~	i چک	Disconnect	Ctrl+T, D
Project Explorer X	Empty Dock	% F	Reconnect	Ctrl+T, E
🛟 msdk 🔹 🖸 🔂 🌗	I	<u>)</u>	Attach Debugger	Ctrl+T, H
Project Items Code Data+RO		<b>b</b> I	Reset	Ctrl+T, S
Solution 'MSDK'	4	<b>↓</b> ∃ [	Download MSDK	Ctrl+T, L
Project 'ALICLOUD'		<u>ک</u> ا	/erify MSDK	Ctrl+T, V
Project MSDK 740.1K 101.0K		_	All	Chill T. K
			Indiana di Dana an	Ctri+1, K
			Jpioad Kange	
		[	Download File	•
		١	/erify File	•
		3	Start Cycle Counter	
	,	3	ause Cycle Counter	
		37	ero Cycle Counter	Ctrl+T 7
		_		
		5	Switch Project	+
	Output	<b>9</b> 1	Farget Connection Proper	ies

#### Fogure 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 <u>Figure 5-11. MSDK SES</u> <u>Project Configuration Interface</u>.



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

Project 'MSDK' Options						
↑ ↓ 🕄 msdk	▼ Search Options					
<ul> <li>Code</li> <li>Assembler</li> <li>Build</li> </ul>	Option	Value				
Code Analyzer Code Generation Compiler Compiler Warning External Build File Library Linker Preprocessor Printf/Scanf Section Source Code User Build Step	Target Connection     Target Connection     Target Device [RISC-V]     Run To Control     Run To     Startup Completion Point     Start From Entry Point Symbol     Leave Target Running     CPU Register File     Register Definition File     Debug Terminal Log File     HTML Watch File     Threads Script File     Thread Maximum     Working Directory     Command Argumentr	J-Link inherits GD32VW553HMQ7 inherits Always main inherits Yes inherits No \$(StudioDir)/targets/cpu_registers_riscv.xml \$(ProjectDir)/GD32VW553x_Registers.xml modified None \$(StudioDir)/html/heap.htm None 25 \$(ProjectDir) \$(ProjectDir) \$(ProjectDir) \$(ProjectDir) \$(ProjectDir) \$(ProjectDir) \$(ProjectDir)				
Debug     Debugger     GDB Server     J-Link     Loader     Simulator [RISC-V]     Target Script	Command Arguments     Debug Additional Configurations     Debug Additional Configurations     Debug Project Name     Entry Point Symbol     Ignore .debug_aranges Section     Ignore .debug_frame Section     IsA Stensions Debug     Load Additional Projects     Memory Upload Page Size     Reserved Member Name	S(ProjectName)S(EXE) S(ProjectName)S(Configuration)start inherits No No No No 1,024 reserved				

#### 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\risc\\env\gd32w55x.Id, 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.

## 6.3. Select different project configurations during debugging

The MSDK project supports multiple configurations (refer to <u>3.2.6Configuration Selection</u>). 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* <u>Select Project Configuration for Debugging</u>. 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 C	Configuration for	Debugging
-----------------------------	-------------------	-----------

D Program Selection	<
Choose a program to run:	
Binaries:	
MSDK.elf	ור
Qualifier:	_
🅸 riscyle - /MSDK/msdk/MSDK.elf	ור
riscvle - /MSDK/msdk_ffd/MSDK.elf	
riscvle - /MSDK/msdk_mbedtls_2.17.0/MSDK.elf	
riscvle - /MSDK/msdk_rtthread/MSDK.elf	
riscvle - /MSDK/msdk_threadx/MSDK.elf	
OK Cancel	

## 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:

 Use administrator privileges to open the zadig.exe file (official website: <u>https://zadig.akeo.ie</u>). Click "Options" and check "ListAll Devices," as shown in *Figure* <u>6-2 Zadig Options Selection</u>.

Figure 6-2 Zadig Options Selection



 Select "JLink devices" from the dropdown menu, as shown in <u>Figure 6-3 Replace JLink</u> <u>Driver</u>, where the BULK interface is displayed. Click "Replace Driver" to replace the JLink driver with WinUSB.



#### Figure 6-3 Replace JLink Driver

Zadig			×
Device Options Help			
BULK interface (Interface 2)		~	Edit
Driver jlink (v2.70.8.0)	More Ir <u>WinUSB</u>	iformatio (libusb)	on
WCID <sup>2</sup> X Replace Driver	libusbK WinUSB	(Microsoft	Ð
5 devices found.	Z	adig 2.3	3.701

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 firmw are dow nloads	May.15.2025



#### **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 selfcontained 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 w hatsoever 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.

© 2025 GigaDevice Semiconductor Inc. - All rights reserved