GigaDevice Semiconductor Inc.

GD32W51x TrustZone Development Guide

Application Note AN103



# **Table of Contents**

Tabl	e of (	Contents	2
List	of Fi	gures	3
List	of Ta	bles	4
1.	Intr	oduction	5
2.	Tru	stZone introduction	6
3.	Sof	tware development	8
3.1	. D	eveloping TrustZone in Keil	8
3	8.1.1.	Secure Project	
3	3.1.2.	Non-secure Project	
3	3.1.3.	Compile Project	13
3	8.1.4.	Download Project	14
3.2	2. D	eveloping TrustZone in IAR	14
3	3.2.1.	Secure Project	15
3	3.2.2.	Non-secure Project	17
3	3.2.3.	Compile Project	
3	3.2.4.	Download Project	20
4.	Coc	de interpretation	
4.1	. S	ecure Project	
4.2	2. N	on-secure project	24
5.	Rev	vision history	26



# **List of Figures**

Figure 2-1. Example of memory map security attribution vs SAU configuration regions	6
Figure 2-2. Enable TrustZone using the GD-Link Programmer	7
Figure 3-1. Keil project file structure	8
Figure 3-2. Select secure project	8
Figure 3-3. Select Secure mode	9
Figure 3-4. Select scattered load file and NSC output library	9
Figure 3-5. Debugger setup in secure project	10
Figure 3-6. Download algorithm and function in secure project	11
Figure 3-7. Select non-secure project	11
Figure 3-8. Select non-secure mode	11
Figure 3-9. Select scattered load file and import NSC library	12
Figure 3-10. Debugger setup in non-secure project	13
Figure 3-11. Download algorithm and function in non-secure project	13
Figure 3-12. Compile Project	14
Figure 3-13. Boot option	14
Figure 3-14. Download project	14
Figure 3-15. IAR project file structure	15
Figure 3-16. Select secure project	15
Figure 3-17. Select Secure mode	16
Figure 3-18. Select scattered load file and NSC output library	16
Figure 3-19. Debugger setup in secure project	17
Figure 3-20. Select non-secure project	18
Figure 3-21. Select non-secure mode	18
Figure 3-22. Select scattered load file and import NSC library	19
Figure 3-23. Debugger setup in non-secure project	20
Figure 3-24. Compile project	20
Figure 3-25. Boot option	21
Figure 3-26. Download project	21
Figure 4-1. SAU configuration	22



# **List of Tables**

Table 3-1. Project_S.sct code	10
Table 3-2. Project_NS.sct code	12
Table 3-3. gd32w51x_flash_s.icf code	16
Table 3-4. gd32w51x_flash_ns.icf code	19
Table 4-1. main code in secure project	22
Table 4-2. FMC configuration	23
Table 4-3. SRAM configuration	23
Table 4-4. Call non-secure function in secure code	24
Table 4-5. Call secure function in non-secure code	25
Table 5-1. Revision history	26



# 1. Introduction

This article describes developing the TrustZone program on the GD32W51x series. The TrustZone security attribute is the main ARMv8-M extension, which provides hardware security separation and protection. The sample code is also divided into two parts, security and non-security. The development process and key points for Keil and IAR are described.

In this article, the GD32W515P-EVAL development board was used, the chip model was GD32W515PIQ6, and the Cortex-M33 kernel with TrustZone was used, with max frequency up to 180MHz. The TrustZone security is activated by the TZEN option bit in the EFUSE\_TZCTL register or the TZEN option bit in the option byte. The sample code is provided in 'GD32W515P\_EVAL\_Demo\_Suites / 23\_TrustZone' and is provided by the software package in <u>GD32MCU.COM</u>.

Applicable product: GD32W51x series



# 2. TrustZone introduction

The M33 kernel with TrustZone has both secure and non-secure states, which use a separate set of kernel registers and Systick. Memory is classified into secure (S), non-secure callable (NSC), and non-secure (NS) domain. A secure domain can access a non-secure domain, and a non-secure can only access a non-secure callable(in secure code), providing hardware protection for access. When TrustZone is enabled, SAU (security attribution unit) and IDAU (implementation defined attribution unit) are used together to set the security attribute of the memory address. The final security level is the attribute of higher security level defined in SAU and IDAU (S>NSC>NS). See <u>Figure 2-1. Example of memory map security attribution vs SAU configuration regions</u>.

IDAU provides a hardware partition of non-secure (NS) and non-secure Callable (NSC) security attributes, which cannot be changed by software, and memory mapped partitions refer to Section 1.4 Memory Mapping in the GD32W51x\_User manual.

SAU can be software-configured with up to 8 security attribute zones, which can be made secure (S), non-secure (NS), or non-secure callable (NSC).



# Figure 2-1. Example of memory map security attribution vs SAU configuration regions

GD32W51x uses TrustZone protection controller union (TZPCU) to manage the security attributes of peripherals, SRAM, and flash memory. TZPCU provides unauthorized access control, working with the kernel to achieve full features of TrustZone.

Ensure that the TZEN bit in the EFUSE\_TZCTL register or the TZEN bit in the option byte is enabled before using TrustZone or TZPCU. Using GD-Link Programmer to enable TZEN bit in option bytes, as shown in *Figure 2-2. Enable TrustZone using the GD-Link Programmer*. The GD-Link Programmer can be obtained from <u>GD32MCU.COM</u>.



# Figure 2-2. Enable TrustZone using the GD-Link Programmer

📸 GD	-Link Programmer 4.6.1	0.13769		
File(F)	Target(T) GD-Link R	efresh Edit(E) Vie	w(V) Help(H)	
GD-Lin	k Option Bytes Configur	ation		×
	OntionBytes 0y40022	040		<b>^</b>
E	EMC_OBR	0x000080AA		
	TZEN		Trust zone enable bit	
- 11	SRAM1_RST		SRAM1 reset enable bit	
	SPC	0xAA	Flash security protection v	value
Ξ	FMC_OBUSER	0xFFFFFFFF		
	USER	0xFFFFFFFF	Option byte USER value	
Ξ	FMC_SECMCFG0	0x003F0000		
	SECM0_EPAGE	0x3F	End page of secure mark	area O
	SECM0_SPAGE	0x0	Start page of secure mark	area 0
Ξ	FMC_DMP0	0x00000000		
	DMP0EN		DMP area 0 enable	
	DMP0_EPAGE	0x0	End page of DMP area 0	~
Tips:	please refer to the chip Use	r Manual to modify the	option bytes.	
			Reset OK	Cancel



# 3. Software development

This article uses two development environments, MDK-ARM v5.28.0.0 and IAR v8.50.9. Please ensure that the corresponding software and device package are correctly installed. The software can be obtained from the official website of **KEIL** and **IAR**. The device package (GD32W51x\_AddOn) can be obtained from <u>GD32MCU.COM</u>.

The sample project has secure and non-secure projects that use different Flash and SRAM. The secure project uses the first 256KB in the FMC (starting address 0x0C000000, size 0x40000), and the memory uses SRAM0 blocks (starting address 0x30000000, size 0x10000). The non-secure project uses the last 1792KB in the FMC (starting address 0x08040000, size 0x1C0000) and the memory uses SRAM1 block (starting address 0x20010000, size 0x10000).

# 3.1. Developing TrustZone in Keil

Multiple project can be used to develop two projects at the same time, as shown in *Figure* <u>3-1. Keil project file structure</u>, project path:

GD32W51x\_Demo\_Suites\GD32W515P\_EVAL\_Demo\_Suites\Projects\23\_Trustzone\MDK-ARM\GD32W515P\_EVAL\_TRUSTZONE.uvmpw



# Figure 3-1. Keil project file structure

# 3.1.1. Secure Project

1. Select secure project as current project, as shown in Figure 3-2. Select secure project.

Figure 3-2. Select secure project

📴 WorkSpace	l l	407
Project: GD32W515P_EV	SECURE	498
🖻 ᇶ GD32W515P_EVAL_S	Set as Active Project	9
🗉 🧰 Application		410



2. In Project/Options for Target/Target, select Code Generation/Software Model as Secure Mode, as shown in *Figure 3-3. Select Secure mode*.

#### Figure 3-3. Select Secure mode

Options for Target 'GD32W515P_EVAL_SEC	CURE' ×
Device Target Output Listing User   C/C	C++ (AC6)   Asm   Linker   Debug   Utilities
GigaDevice GD32W515PIQ6	Code Generation
Xtal (MHz): 12.0	ARM Compiler: Use default compiler version 6
Operating system: None	Software Model: Secure Mode
System Viewer File:	🔽 Use MicroLIB 🔲 Big Endian
GD32W515Px.svd	Floating Point Hardware: Single Precision
Use Custom File	
Read/Only Memory Areas	Read/Write Memory Areas
default off-chip Start Size Sta	tartup default off-chip Start Size NoInit
ROM1:	C RAM1:
ROM2:	C 🗆 RAM2:
ROM3:	C 🗆 RAM3:
on-chip	on-chip
□ IROM1: 0x8040000 0x1C0000	C □ IRAM1: 0x20010000 0x10000 □
IROM2: 0xC000000 0x40000	
OK	Cancel Defaults Help

3. Secure code and non-secure code use different Flash and SRAM, and use a scattered load file to assign the Flash and SRAM for secure code. In Project/Options for Target/Linker, select not to assign addresses using the Target interface, select scatter load file, and set the output library for NSC functions. See *Figure 3-4. Select scattered load file and NSC output library*.

The output command is --import\_cmse\_lib\_out=.\secure\_nsclib.o. The compiler will compile the code at compile time with \_\_attribute((cmse\_nonsecure\_entry)) identifying code into.\secure\_nsclib.o. Non-secure code can access function of secure\_nsclib.o.

#### Figure 3-4. Select scattered load file and NSC output library

Options for Target 'GD32W515P_EVAL_SECURE'	$\times$				
Device       Target       Output       Listing       User       C/C++ (AC6)       Asm       Linker       Debug       Utilities         Use Memory Layout from Target Dialog       X/O Base:					
ScatterSecure_Code\Project_S.sct Edit					
Misc controls					
Linker -cpu=Cortex-M33 *.o controllibrary_type=microlib -strict -scatter "\\Secure_Code\Project_S.sct"					
OK Cancel Defaults Help					

4. The secure project uses Flash address 0x0C000000 with size 0x40000 and SRAM address 0x30000000 with size 0x10000. NSC function is assigned to address 0x0C03E000 and size is 0x00002000. Project\_S.sct code is shown in <u>Table 3-1. Project\_S.sct code</u>.



Table 3-1. Project\_S.sct code

```
LR_IROM1 0x0C000000 0x00040000 {
                                       ; load region size_region
  ER_IROM1 0x0C000000 0x0003E000 { ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
   .ANY (+XO)
  }
  RW_IRAM2 0x3000000 0x00010000 { ; RW data
   .ANY (+RW +ZI)
 }
}
LR_IROM2 0x0C03E000 0x00002000 {
  ER_IROM2 0x0C03E000 0x00002000 { ; load address = execution address
   *(Veneer$$CMSE)
                                      ; check with partition.h
 }
```

5. Use GD-Link for debugging and downloading. Connect GD-Link interface of Eval-board directly to PC-USB. Select CMSIS-DAP ARMv8-M Debugger under Project/Options for Target/Debug. Identify the debugger ID in Settings, use the SW interface and set the maximum clock to 1MHz. See *Figure 3-5. Debugger setup in secure project*.



Options for Target 'GD32W515P_EVAL_SECURE'	$\times$			
Device   Target   Output   Listing   User   C/C++ (AC6)   Asm   Linker Debug   Utilities				
C Use Simulator <u>with restrictions</u> Settings ( Use: CMSIS-DAP ARMv8-M Debugg ▼ Settings				
CMSIS-DAP ARMv8-M Target Driver Setup	×			
Debug Trace   Flash Download				
CMSIS-DAP - JTAG/SW Adapter       SW Device       Move         Serial No:       IDCODE       Device Name       Up         Firmware Version:       2.0.0       IDCODE       Device Name       Up         IV       SWJ Port:       SW       IDCODE       Device Name       Up         Max Clock:       111114       IDCODE       IDCODE:       IDCODE				
Max Clock: 1MHz  Add Delete Update AP: Connect & Reset Options Connect: Nomal  Reset: Autodetect  Connect Nomal  Reset: Autodetect  Connect & Delay Accesses Son after Peared Download to Play				
OK Cancel Help				

6. Under Flash Download TAB, select Erase Sectors and uncheck Reset and Run. Add programming algorithm GD32W515\_Secure\_Flash. GD32W515\_Secure\_Flash is provided by the installed device package. See *Figure 3-6. Download algorithm and function in secure project*.



#### Figure 3-6. Download algorithm and function in secure project

WINIOAD FUNCTION C Erase Full Chip C Erase Sectors C Do not Erase	<ul> <li>✓ Program</li> <li>✓ Verify</li> <li>✓ Reset and F</li> </ul>	RAM for A Start:	Algorithm
ogramming Algorithm			
Description	Device Size	Device Type	Address Range
GD32W515_Secure_Flash	2M	On-chip Flash	0C000000H - 0C1FFFFFH

## 3.1.2. Non-secure Project

1. Select non-secure project as current project, as shown in *Figure 3-7. Select non-secure project*.

#### Figure 3-7. Select non-secure project

∋ 🚰 Wo	orkSpace		
÷. 🔧	Project: GD32W515	5P_EVAL_SECU	
÷.~\$\$	Project: GD32W51	SD EVAL NON	
:	👦 GD32W515P_E	Set as Active Project	
	📥 🧰 🔥 martinester		

2. In Project/Options for Target/Target TAB, select Code Generation/Software Model as Nonsecure Mode, as shown in *Figure 3-8. Select non-secure mode*.

#### Figure 3-8. Select non-secure mode

Options for Target 'GD32W515P_EVAL_NON_SEC	CURE' X
Device Target Output   Listing   User   C/C++ (A	C6)   Asm   Linker   Debug   Utilities
GigaDevice GD32W515PIQ6 Xtal (MHz): 12.0	Code Generation ARM Compiler: Use default compiler version 6
Operating system: None	Software Model: Non-Secure Mode
System Viewer File:	▼ Use MicroLIB
GD32W515Px.svd	Floating Point Hardware: Single Precision
Use Custom File	
Read/Only Memory Areas	Read/Write Memory Areas
default off-chip Start Size Startup	default off-chip Start Size NoInit
□ ROM1: ○	□ RAM1: □
□ ROM2: □ C	□ RAM2: □
□ ROM3: □ C	□ RAM3: □
on-chip	on-chip
IROM1: 0x8000000 0x200000 €	IRAM1: 0x20000000 0x70000 □
□ IROM2: 0xC000000 0x200000 C	□ IRAM2: 0x30000000 0x70000 □
OK Car	ncel Defaults Help

3. Secure code and non-secure code use different Flash and SRAM, and use a distributed load file to assign the Flash and SRAM for secure code. In Project/Options for Target/Linker,



select not to assign addresses using the Target interface, select the scattered load file, and use the NSC function library exported by the Security Project. See <u>Figure 3-9. Select</u> <u>scattered load file and import NSC library</u>.

Library import command ../Secure\_Project/secure\_nsclib.o, non-secure project can directly call the \_\_attribute((cmse\_nonsecure\_entry)) identification code in the security project.

Figure 3-9. Select scattered load file and import NSC library

Options for Target 'GD32W515P_EVAL_NON_SECURE'	×			
Device   Target   Output   Listing   User   C/C++ (AC6)   Asm Linker   Debug   Utilities				
□ Use Memory Layout from Target Dialog       X/O Base:         □ Make RW Sections Position Independent       R/O Base:         □ Make RO Sections Position Independent       R/O Base:         □ Don't Search Standard Libraries       0x20000000         ☑ Report 'might fail' Conditions as Errors       disable Warnings:				
ScatterSecure_Code\Project_NS.sct	]			
Misc/Secure_Project/secure_nsclib.o	]			
Linker -cpu=Cortex-M33 *.o controlIbrary_type=microlib -strict -scatter "\\NSecure_Code\Project_NS.sct"				
OK Cancel Defaults Kelp				

4. Non-secure code uses Flash address 0x08040000 with size 0x001C0000 and SRAM address 0x20010000 with size 0x10000. Project\_NS.sct code is shown in <u>Table 3-2</u> <u>Project NS.sct code</u>.

Table 3-2. Project\_NS.sct code

```
LR_IROM1 0x08040000 0x001C0000 { ; load region size_region
ER_IROM1 0x08040000 0x001C0000 { ; load address = execution address
 *.o (RESET, +First)
 *(InRoot$$Sections)
.ANY (+RO)
.ANY (+RO)
}
RW_IRAM1 0x20010000 0x00010000 { ; RW data
.ANY (+RW +ZI)
}
```

5. Use GD-Link for debugging and downloading. Connect GD-Link interface of Eval-board directly to PC-USB. Select CMSIS-DAP ARMv8-M Debugger under Project/Options for Target/Debug. Identify the debugger ID in Settings, use the SW interface and set the maximum clock to 1MHz. See *Figure 3-10. Debugger setup in non-secure project*.



#### Figure 3-10. Debugger setup in non-secure project

Options for Target 'GD32W515P_EVAL_SECURE'	×
Device   Target   Output   Listing   User   C/C++ (AC6)   Asm   Linker Debug   Utilities	1
C Use Simulator <u>with restrictions</u> Settings Settings CMSIS-DAP ARMv8-M Debugg Settings	attings
CMSIS-DAP ARMv8-M Target Driver Setup	×
Debug   Trace   Flash Download	
CMSIS-DAP - JTAG/SW Adapter       SW Device         Serial No:       IDCODE         Firmware Version:       20.0         V       SWJ Port:         SWJ       Port:         Max Clock:       IMHz    SW Divice Device Name SWDIO O 0 0x0BE12477 ARM CoreSight SW-DP	Move Up Down
Debug       Connect & Reset Options       Download Options         Connect:       Nomal       ▼       Reset:       Autodetect       ▼         ✓       Cache Code       ✓       Cache Code       ✓       Download Options         ✓       Reset after Connect       ✓       Cache Memory       ✓       Download to Ra         ✓       Log Debug Accesses       ✓       Stop after Reset       ✓       Ownload to Ra	nload sh
OK Cancel	Help

6. Under Flash Download TAB, select Erase Sectors and uncheck Reset and Run. Add programming algorithm GD32W515\_Nsecure\_Flash. GD32W515\_Nsecure\_Flash is provided by the installed device package. See *Figure 3-11. Download algorithm and function in non-secure project*.

Figure 3-11. Download algorithm and function in non-secure project

CMSIS-DAP ARMv8-M Target D	Driver Setup	×
Debug   Trace Flash Downloa	a	
Download Function     C Erase Full Chip     C Erase Sectors     C Do not Erase	✓         Program           ✓         Verify           □         Reset and Run	
Programming Algorithm		
Description	Device Size Device Type Address Range	
GD32W515_Nsecure_Flash	2M On-chip Flash 08000000H - 081FFFFFH	
	Start:  0x08000000 Size:  0x00200000	
	Add Remove	
	OK Cancel Help	

## 3.1.3. Compile Project

Firstly compile secure project and then non-secure project. Non-secure project need secure\_nsclib.o library generated in secure project. Compile the secure and insecure projects in order.

Batch setup can also compile two projects in order, as shown in *Figure 3-12. Compile Project*. Select two projects and click Build. The order is determined by project directory sequence, which can be modified by project/Manage/Multi-Project Workspace.



#### Figure 3-12. Compile Project

🎯 🔹 🔜 🕌 GD32W51	Batch Setup	×
👹 Batch Build	Select Project Targets:	
Batch Rebuild	GD32W515P_EVAL_SECURE	Build
Batch Setup	GD32W515P_EVAL_SECURE	Rebuild
	GD32W515P_EVAL_NON_SECURE	Clean

## 3.1.4. Download Project

Set up Eval-board correctly before downloading code. JP4 (BOOT0/BOOT1) connects to L MCU will boot from the secure code. JP3 connects to SWD. See <u>Figure 3-13. Boot option</u>. JP21 connects to USART for printf. Connect Eval-board GD-Link and USART to PC, and ensure that software drivers are correctly installed.

#### Figure 3-13. Boot option



Firstly download non-secure code and then secure code. MCU always boots from secure code and jumps to non-secure code. See *Figure 3-14. Download project*.

#### Figure 3-14. Download project



Reset and Run are not checked in the Flash Download option. Press manually Reset button to restart MCU. Two messages will be printed and two LED lights flash respectively.

# 3.2. Developing TrustZone in IAR

Two projects can be developed at the same time using multiple project methods. The file structure is shown in *Figure 3-15. IAR project file structure*, project path:

GD32W51x\_Demo\_Suites\GD32W515P\_EVAL\_Demo\_Suites\Projects\23\_Trustzone\EWA RM\GD32W515P\_EVAL\_TRUSTZONE.eww



### Figure 3-15. IAR project file structure



## 3.2.1. Secure Project

1. Select secure project as active project, as shown in Figure 3-16. Select secure project.



## Figure 3-16. Select secure project

2. In Project / Options / General Options / Target, select TrustZone / Mode as Secure, as shown in *Figure 3-17. Select Secure mode*.



#### Figure 3-17. Select Secure mode

Options for node "GD32W	515P_EVAL_SECURE*			×
Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler	Library Options 2	2 MISRA It Library Co	A-C:2004	MISRA-C:1998 Library Options 1
Custom Build	Processor variant			
Linker Debugger	⊖ Core	Cortex-M33	$\sim$	
Simulator CADI	Device	GD GD32W515x	I_TZ	"⊡+
CMSIS DAP GDB Server	O CMSIS-Pack	None		
J-Link/J-Trace TI Stellaris	Endian mode	Floating point	settings	
Nu-Link PE micro		FPU	None	$\checkmark$
ST-LINK Third-Party Driver TI MSP-FFT	O Big O BE32	D registers	- ~	
TI XDS	BE8			
	DSP Extension	) (NEON)	✓ TrustZo Mode Se	cure V
		[	OK	Cancel

3. Secure code and non-secure code use different Flash and SRAM, and use a scattered load file to assign the Flash and SRAM for secure code. In Project / Options / Linker / Config, select scatter load file and set the output library for NSC functions. See <u>Figure 3-18. Select</u> scattered load file and NSC output library.

The output command is --import\_cmse\_lib\_out=.\secure\_nsclib.o. The compiler will compile the code at compile time with \_\_attribute((cmse\_nonsecure\_entry)) identifying code into.\secure\_nsclib.o. Non-secure code can access functions of secure\_nsclib.o.

Options for node "GD32W515P_EVAL_SECURE" X	Options for node "GD32W515P_EVAL_SECURE" X
Calegory: Factor Dotors Stats: Analyse Runtime Checking CC+++ Config Assemble: Output Converter Cale Debugge: Smutator CADI CHSIS DAP GOB Server Jatti Jatti Jatti TI Stideris Nu-Link Pe mico ST-LDNK TH Steleris TI Stideris Nu-Link Pe mico ST-LDNK The diffus Diagnostics Checksum Encodings Extra Options Config Library Input Optimizations Advanced Output List Checksum Encodings Extra Options (Inker configuration file SPRO)_DIR\$\_\_\Secure_Code\gd32w51x_flash_s.icf I Stideris Nu-Link Pe mico ST-LDNK The diffus Driver TI XDS DK Cencel	Category: General Options Static Andryis Ruther Checking C(C++ Compler Acasmbier Output Commetre Cutom Build Build Actions Mathematical Comfigure Library Industry Input Optimizations Advanced Output List Output filename: Project.Comfigure Smultator CADI

Figure 3-18. Select scattered load file and NSC output library

4. The secure project uses Flash address 0x0C000000 with size 0x40000 and SRAM address 0x30000000 with size 0x10000. NSC function is assigned to address 0x0C03E000 and size is 0x00002000. gd32w51x\_flash\_s.icf code is shown in <u>Table 3-3. gd32w51x\_flash\_s.icf</u> code.



/\*-Specials-\*/



define symbolICFEDIT_intvec_start = 0x0C000000;
/*-Memory Regions-*/
define symbolICFEDIT_region_ROM_start = 0x0C000000;
define symbolICFEDIT_region_ROM_end = 0x0C03FFFF;
define symbolICFEDIT_region_RAM_start = 0x30000000;
define symbolICFEDIT_region_RAM_end = 0x3000FFFF;
define symbolregion_ROM_NSC_start = 0x0C03E000;
define symbolregion_ROM_NSC_end = 0x0C03FFFF;
define region ROM_region = mem:[fromICFEDIT_region_ROM_start to
ICFEDIT_region_ROM_end];
define region ROM_NSC_region = mem:[fromregion_ROM_NSC_start to
region_ROM_NSC_end];
define region RAM_region = mem:[fromICFEDIT_region_RAM_start to
ICFEDIT_region_RA M_end];
place in ROM_region { readonly };
place in ROM_NSC_region { section Veneer\$\$CMSE };
place in RAM_region { readw rite, block HEAP, block CSTACK };

5. Use GD-Link for debugging and downloading. Select CMSIS-DAP under Project / Options / Debugger. See *Figure 3-19. Debugger setup in secure project*.

Figure 3-19. Debugger setup in secure project
---

	Factory Settings
General Options Static Analysis Runtime Checking C/C++ Compiler Assembler	Setup Download Images Multicore Extra Options Plugins
Custom Build	Driver 🗹 Run to
Build Actions	CMSIS DAP y main
Debugger Smulator CADI OMSIS DAP GDB Server I jet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver	Setup macros Use macro file(s) C\Program Files (x86)\IAR Systems\Embedded Workbench Device description file Cverride default STOOLKIT_DIR\$\CONFIG\debugger\GD\GD32W515xLddf
TI MSP-FET TI XDS	

# 3.2.2. Non-secure Project

1. Select non-secure project as active project, as shown in *Figure 3-20. Select non-secure project*.



#### Figure 3-20. Select non-secure project

	NON SECURE-G	21
	Options	
He CMSIS He CMSIS He CDoc He GD32W51x_EVAL	<b>Make</b> Compile	
HE Startun	Rebuild All	
L = Standp L = Output	Clean	
	C-STAT Static Analysis	>
	Stop Build	
	Add	>
	Remove	
	Rename	
	Version Control System	>
	Open Containing Folder	
	File Properties	
	Set as Active	

2. In Project / Options / General Options / Target, select TrustZone / Mode as Non-secure, as shown in *Figure 3-21. Select non-secure mode*.

Category: General Options State Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Cutput Converter Cutput Converter Build Actions Linker Deb unger	Library Options Target Outp Processor varian O Core	2 M ut Librar t Cortex-M33	IISRA-C:2004 y Configuration	MISRA-C:1998 Library Options 1
Simulator CADI CMSIS DAP GDB Server I-jet J-Link/J-Trace TI Stellaris	Device     OCMSIS-Pack     Endian mode	GD GD32W None	515xI_TZ	
Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET TI XDS	<ul> <li>Little</li> <li>Big</li> <li>BE32</li> <li>BE8</li> </ul>	FPU D registe	None	~
	DSP Extension	ID (NEON)	Mode No	on-secure V

#### Figure 3-21. Select non-secure mode

3. Secure code and non-secure code use different Flash and SRAM, and use a scattered load file to assign the Flash and SRAM for secure code. In Project / Options / Linker / Config, select scatter load file and use the NSC function library exported by the Security Project. See <u>Figure</u> 3-22. Select scattered load file and import NSC library.

Library import command \$PROJ\_DIR\$\...\Secure\_Project\GD32W51x\Exe\secure\_nsclib.o, non-secure project can directly call the \_\_attribute((cmse\_nonsecure\_entry)) identification code in the security project.



## Figure 3-22. Select scattered load file and import NSC library

4. Non-secure code uses Flash address 0x08040000 with size 0x001C0000 and SRAM address 0x20010000 with size 0x10000. Project\_NS.sct code is shown in <u>Table 3-4.</u> <u>gd32w51x flash\_ns.icf code</u>.

Table 3-4. gd32w51x\_flash\_ns.icf code

/*-Specials-*/
define symbolICFEDIT_intvec_start = 0x08040000;
/*-Memory Regions-*/
define symbolICFEDIT_region_ROM_start = 0x08040000;
define symbolICFEDIT_region_ROM_end = 0x081FFFFF;
define symbolICFEDIT_region_RAM_start = 0x20010000;
define symbolICFEDIT_region_RAM_end = 0x2001FFFF;
define region ROM_region = mem:[fromICFEDIT_region_ROM_start to
ICFEDIT_region_ROM_end];
define region RAM_region = mem:[fromICFEDIT_region_RAM_start to
ICFEDIT_region_RA M_end];
place in ROM_region { readonly };
place in RAM_region { readw rite, block HEAP, block CSTACK};

5. Use GD-Link for debugging and downloading. Select CMSIS-DAP under Project / Options / Debugger. See *Figure 3-23. Debugger setup in non-secure project*.



### Figure 3-23. Debugger setup in non-secure project

Lategory:	Factory Settings
Seneral Options	
itatic Analysis	
Runtime Checking	
C/C++ Compiler	Setup Download Images Multicore Extra Options Plugins
Assembler	
Output Converter	Driver
Custom Build	Main to
Build Actions	CMSIS DAP v main
Linker	
Debugger	Setup macros
CADI	
CADI CMETE DAD	Use macro file(s)
CDB Server	C/\ Brogrom Files (v86)\\AR Systems\Embedded Workbarch
T-iet	e. (Frogram Fries (xoo) (FRIE Systems (Embedded Workberten
A Link (A Trans	
I-LIDK/I-IFACE	
TI Stellaris	
TI Stellaris Nu-Link	
TI Stellaris Nu-Link PE micro	Device description file
TI Stellaris Nu-Link PE micro ST-LINK	Device description file
TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver	Device description file ☑ Override default
TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET	Device description file
J-LINK/J-Frace TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET TI XDS	Pevice description file     Override default     \$TOOLKIT_DIR\$\CONFIG\debugger\GD\GD32W515xl.ddf
J-Link/J-Hade TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET TI XDS	Device description file Override default \$TOOLKIT_DIR\$\CONFIG\debugger\GD\GD32W515x1.ddf
Junicy-Inace Ti Stellaris Nu-Link PEmicro ST-LINK Third-Party Driver TI MSP-FET TI XDS	Device description file ☑ Override default \$TOOLKIT_DIR\$\CONFIG\debugger\GD\GD32W515x1.ddf
J-Link Ti Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET TI XDS	Device description file ⊘ Override default \$TOOLKIT_DIR\$\CONFIG\debugger\GD\GD32W515xLddf
J-Link/J-Irace Ti Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET TI MSP	Device description file       Override default       \$TOOLKIT_DIR\$\CONFIG\debugger\GD\GD32W515xl.ddf
J-LINK/J-Irace TI Stellaria Nu-Link PEmicro ST-LINK Third-Party Driver TI NSP-FET TI NSP- TI NSP	Device description file     Override default     STOOLKIT_DIR\$\CONFIG\debugger\GD\GD32W515xLddf
J-UNK/J-Irace TS Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET TI MSP-FET TI MSP	Device description file     Override default     STOOLKIT_DIR\$\CONFIG\debugger\GD\GD32W515xl.ddf
J-LINK/J-Irace TS Stellaris Nu-Link PE micro ST-LINK Third-Party Driver Third-Party Driver TI MSP-FET TI MSP-FET TI MSP	The series of t
J-umk/J-irace TS-tellans Nu-Link PE-micro ST-LINK Third-Party Driver TI MSP-FET TI MSP-FET TI XDS	Device description file Override default \$TOOLKIT_DIR\$\CONFIG\debugger\GD\GD32W515x1.ddf

## 3.2.3. Compile Project

Firstly compile secure project and then non-secure project. Non-secure project need secure\_nsclib.o library generated in secure project. Compile the secure and insecure projects in order.

Batch setup can also compile two projects in order, as shown in *Figure 3-24. Compile project*. The order is determined by the order of addition.

Figure 3-24. Compile project

		Edit Batch Build	×
Batch Build	×	Name	
Batches: GD32w515_T2	New		
	Edit t	Available configurations Configurati	SECURE · GD32\ NON_SECURE · I
	Close Cancel	× · · · · · · · · · · · · · · · · · · ·	
Build Make Clean	Rebuild All	(Drag to order)	
		ОК	Cancel

## 3.2.4. Download Project

Set up Eval-board correctly before downloading code. JP4 (BOOT0/BOOT1) connects to L MCU will boot from the secure code. JP3 connects to SWD. See <u>Figure 3-25. Boot option</u>. JP21 connects to USART for printf. Connect Eval-board GD-Link and USART to PC, and ensure that software drivers are correctly installed.



#### Figure 3-25. Boot option



Firstly download non-secure code and then secure code. MCU always boots from secure code and jumps to non-secure code. See *Figure 3-26. Download project*.

#### Figure 3-26. Download project



Reset and Run are not checked in the Flash Download option. Press manually Reset button to restart MCU. Two messages will be printed and two LED lights flash respectively.



# 4. Code interpretation

# 4.1. Secure Project

If Code Generation/Software Model is set to Secure Mode in the Project/Options for Target/Target TAB, \_\_ARM\_FEATURE\_CMSE is defined as 3. After power-on and reset, MCU will execute system\_gd32w51x.c/SystemInit()/sau\_region\_config() function to configure the SAU based on the partition\_gd32w51x.h content. The partition\_gd32w51x.h file can be configured using the Configuration Wizard interface and generate Text Editor code. As shown in *Figure 4-1. SAU configuration*. SAU is enabled and four regions are configured. Region0 configures NSC addres, corresponding to Project\_S.sct. Region1 configures non-secure code address. Region2 configures peripheral address to be non-secure. Peripheral can be accessed in both secure or non-secure code.

## Figure 4-1. SAU configuration

partition_gd32w51x.h				
Expand All Collapse All Help Show Grid				
Option	Value			
Configure Security Attribution Unit (SAU)				
Enable SAU	<b>V</b>			
When SAU is disabled, the atrributes of all memory	Non-Secure			
-Configure Security Attribution Unit (SAU) Address Regions Attributes				
😑 Configure SAU Region 0	▼			
Start Address	0x0C03 E000			
End Address	0x0C03 FFFF			
Region is	Secure, Non-Secure Callable			
😑 Configure SAU Region 1	<b>v</b>			
Start Address	0x0804 0000			
End Address	0x081F FFFF			
Region is	Non-Secure			
😑 Configure SAU Region 2	<b>v</b>			
Start Address	0x2001 0000			
End Address	0x2001 FFFF			
Region is	Non-Secure			
😑 Configure SAU Region 3	✓			
Start Address	0x4000 0000			
End Address	0x4FFF FFFF			
Region is	Non-Secure			
E-Configure SAU Region 4				
i∎Configure SAU Region 5				
E-Configure SAU Region 7				
Configure behaviour of Floating Point Unit				
Text Editor Configuration Wizard				

In main function of secure project, secure interrupt is enabled, Systick configuration, FMC configuration, TZBMPC configuration, LED configuration, USART configuration, and finally jump to non-secure code. See <u>Table 4-1. main code in secure project</u>.

Table 4-1. main code in secure project

```
int main(void)
{
    /* enable SecureFault handler */
    SCB->SHCSR |= SCB_SHCSR_SECUREFAULTENA_Msk;
```



/\* configure systick \*/ systick\_config(); /\* configure mark secure pages \*/ if(SECM\_SPAGE != (FMC\_SECMCFG0 & 0x3FF) || SECM\_EPAGE != ((FMC\_SECMCFG0 >> 16) & 0x3FF)){ fmc\_secmark\_config(); } if (OU == (FMC\_OBR & FMC\_OBR\_TZEN)){ /\* enable trustzone \*/ fmc\_trustzone\_enable(); } /\* configure TZBMPC \*/ tzbmpc\_config(); led\_config(); com\_config(); /\* setup and jump to non-secure \*/ nonsecure\_init(); w hile(1){ }

Function fmc\_secmark\_config() configures pages 0-63 of FMC. First 256KB is used for secure code.

## Table 4-2. FMC configuration

```
void fmc_secmark_config(void)
{
    fmc_unlock();
    ob_unlock();
    /* configure mark secure pages */
    ob_secmark_config(SECM_SPAGE, SECM_EPAGE, SECM_INDEX0);
    ob_start();
    w hile(0U != (FMC_SECSTAT & (FMC_SECSTAT_SECBUSY))){
    }
    ob_reload();
    ob_lock();
    fmc_lock();
```

Function tzbmpc\_config() configures SRAM1 as non-secure. This area is used for non-secure code.

#### Table 4-3. SRAM configuration

void tzbmpc\_config(void)



```
uint16_t block_number = 0U;
/* enable TZPCU_clock */
rcu_periph_clock_enable(RCU_TZPCU);
/* SRAM1 is used to nonsecure code, so all blocks of SRAM1 should set to nonsecure */
for(block_number = 0U; block_number <= TZBMPC1_BLOCK_NUMBER; block_number++){
    tzpcu_tzbmpc_block_secure_access_mode_config(TZBMPC1, block_number,
BLOCK_SECURE_ACCESS_MODE_NSEC);
  }
}
```

Function led\_config() configures LED1 and LED2 ports as non-secure. LED1 and LED2 will ce controlled by non-secure code. Function com\_config() configures USART2 (serial port) as secure. USART2 will be controlled by secure code.

entry\_cb\_func\_register() and non\_secure\_print() are non-secure callable functions marked with \_\_attribute((cmse\_nonsecure\_entry). Non-secure code can call these functions directly to pass non-secure function addresses and information to print.

If secure code needs to call non-secure function, use cmse\_nsfptr\_create(func\_addr) to register the function address and \_\_attribute((cmse\_nonsecure\_call)) to convert the function to non-secure. Finally use the function. In this demo, non-secure code first calls the entry\_cb\_func\_register() to get address of toggle\_led1, and then calls nonsecure\_func() to toggle LED1. The code is shown in <u>Table 4-4. Call non-secure function in secure code</u>.

## Table 4-4. Call non-secure function in secure code

```
#define CMSE_NS_ENTRY __attribute((cmse_nonsecure_entry))
#define CMSE_NS_CALL __attribute((cmse_nonsecure_call))
typedef void CMSE_NS_CALL (*ns_fptr)(void);
ns_fptr nonsecure_func = (ns_fptr)NULL;
......
CMSE_NS_ENTRY void entry_cb_func_register(void *callback)
{
    if(callback != NULL){
    nonsecure_func = (ns_fptr)cmse_nsfptr_create(callback);
    }
}
```

# 4.2. Non-secure project

NSC functions can be used in non-secure code after imported secure\_nsclib.o library. In main() function, entry\_cb\_func\_register() passes address of toggle\_led1 to security code. Then configure Systick of non-secure domain, initialize LED port. The while loop periodically flips LED2 and calls the NSC function non\_secure\_print to print the information.



	Table 4-5. Call secure function in non-secure code					
ĺ	extern void entry_cb_func_register(void *callback);					
	extern void non_secure_print(const char * str);					
	int main(void)					
	{					
	entry_cb_func_register((void *)toggle_led1);					
	/* configure systick */					
	systick_config();					
	gd_eval_led_init(LED1);					
	gd_eval_led_init(LED2);					
	w hile(1){					
	/* toggle LED2 */					
	gd_eval_led_toggle(LED2);					
	non_secure_print("non-secure code toggle LED2.\r\n");					
	delay_1ms(1000);					
	}					
	}					



# 5. Revision history

# Table 5-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Mar.3 2023



## **Important Notice**

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

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

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

© 2023 GigaDevice - All rights reserved