GigaDevice Semiconductor Inc.

Guideline for migrating the IEC60730 ClassB library onto GD32F3x0 series

Application Note AN171

Revision 1.0

(Jul. 2024)



Table of Contents

Table	of Contents	. 2
List o	of Figures	.3
List o	of Tables	.4
1.	Introduction	. 5
2.	Migration of the IEC60730 class B certification library	.6
2.1.	Migration of certification library project in IAR environment	. 6
2.2.	Migration of certification library project in Keil environment	12
2.3.	Migration of certification library project in eclipse environment	17
3.	Test results in three different IDEs	24
4.	Revision history	25

List of Figures

Figure 2-1. Modify the RAM boundary in the gd32f3x0_test.h	6
Figure 2-2. Add the ClassB library file to the IAR project	7
Figure 2-3. Modify the .s startup file	8
Figure 2-4. Modify the scattered loading file	8
Figure 2-5. Device Configuration	9
Figure 2-6. Modify the precompiled macro 1	10
Figure 2-7. Modify the boundary values in the project settings 1	10
Figure 2-8. Modify the checksum configuration in the project properties1	11
Figure 2-9. Add configurations to the Extra Options tab1	11
Figure 2-10. Modify the RAM and Flash boundaries in the gd32f3x0_test.h file 1	12
Figure 2-11.Add the ClassB library file to the Keil project1	12
Figure 2-12. Editing the scatter loading file 1	15
Figure 2-13. Device Configuration 1	16
Figure 2-14. Modify the precompiled macro 1	16
Figure 2-15.Add a batch file 1	17
Figure 2-16. Add a .ini configuration file 1	17
Figure 2-17. Modify the RAM boundary in the gd32f3x0_test.h file 1	18
Figure 2-18. Add the ClassB library file to the Eclipse project1	19
Figure 2-19. Modify the startup file 2	
Figure 2-20. Scatter loading file location 2	20
Figure 2-21. Modify the Eclipse project to load files in a ld file 2	21
Figure 2-22. Device name configuration 2	21
Figure 2-23. Modify the assembly compiler's precompiled macros 2	22
Figure 2-24.Modify the post-build command 2	22
Figure 2-25. Modify the configuration of the executable file 2	23
Figure 3-1.Test Results 2	24



List of Tables

able 4-1. Revision history 25



1. Introduction

The GD32 MCUs provide IEC60730 Class B certified library support, offering project templates for each GD32 MCU series. When users conduct IEC60730 self-testing certifications for different chips within the same series, they can adapt the target chip by porting the template program. This application note comprehensively outlines the considerations during the migration process for the GD32F3x0 series in various Integrated Development Environments (IDEs) such as Keil, IAR, and Eclipse, guiding users in porting the IEC60730 Class B certification library.



2. Migration of the IEC60730 class B certification library

The GD32 MCU offers IEC60730 Class B certification library support for development environments including IAR, Keil, and Eclipse. There are variations in the template project migration among these three environments.

This application note will elaborate on the differences and provide guidance on project migration specifically for the GD32F350RB in each of these development tools.

2.1. Migration of certification library project in IAR environment

1. Modify the RAM boundary in the macro __IAR_SYSTEMS_ICC__ of the gd32f3x0_test.h file according to the datasheet of the chip, as shown in *Figure 2-1. Modify the RAM boundary in the gd32f3x0_test.h*. Modify the RAM boundary in the gd32f3x0_test.h as indicated.

Figure 2-1. Modify the RAM boundary in the gd32f3x0_test.h

#ifdefIAR_SYSTEMS_ICC					
<pre>extern uint32 tICFEDIT_region_ROM_start_; extern uint32 tICFEDIT_region_RAM_start_; extern uint32 tICFEDIT_region_RAM_start_; extern uint32 tICFEDIT_region_RAM_end_; extern uint32 tICFEDIT_region_IECTEST_FARAM_start_; extern uint32 tICFEDIT_region_IECTEST_FARAM_end_;</pre>					
<pre>#define FLASH_START #define FLASH_SIZE #define FLASH_SIZE_WORDS #define FLASH_END</pre>	<pre>(unsigned int *)&_ICFEDIT_region_ROM_start</pre>				
<pre>#define FLASH_BLOCK_SIZE #define FLASH_BLOCKNUM #define FLASH_BLOCKNUM_WORDS</pre>	((uint32_t)512uL) (uint32_t)((FLASH_SIZE) / FLASH_BLOCK_SIZE) ((uint32_t)(FLASH_SIZE_WORDS / FLASH_BLOCK_SIZE))				
<pre>#define RAM_START #define RAM_END</pre>	(uint32_t *)0x2000000 (uint32_t *) <mark>0x20003F40</mark>				
	((uint32_t *)(6ICFEDIT_region_IECTEST_PARAM_start)) ((uint32_t *)(6ICFEDIT_region_IECTEST_PARAM_end))				
extern voidiar_program_start(void); extern void Reset_Handler(void); #define DefaultSystemStartUp() Reset_Handler()					
<pre>void test_fail_reset(void);</pre>					
-#endif /*IAR_SYSTEMS_ICC */					

2. Check the .s startup file in the Startup folder of the project directory to see if it matches the current chip type. At the same time, add the ClassB library file to the project.

Files ۰ 🖻 🌒 Project - Debug J 🕀 🖬 Application ----EE 🛋 CMSIS – 🗉 🛋 Doc ⊢⊞ 蔰 GD32F3x0_EVAL –⊕ 蔰 GD32F3x0_Peripherals -🗗 💻 IEC60730_ClassB -⊞ 🗟 qd32f3x0_test_clock.c —⊞ 🖻 qd32f3x0_test_cpu_prerun_IAR.s —⊞ 🗟 qd32f3x0_test_cpu_run_IAR.s —⊞ 🗟 gd32f3x0_test_flash.c —⊞ 🗟 qd32f3x0_test_fullram_run_IAR.s —⊞ 🗟 gd32f3x0_test_param.c —⊞ 🗟 qd32f3x0_test_prerun.c —⊞ 💿 qd32f3x0_test_ram.c --⊞ 💿 qd32f3x0_test_run.c 🖵 🗎 readme.txt -🕀 💼 Starup -🕀 🛋 Output

Figure 2-2. Add the ClassB library file to the IAR project

If the startup file is compatible with the current chip model, no modification is needed; if it is not, you need to select a .s startup file that is compatible with the current chip model in the firmware library folder GD32F3x0_Firmware_Library\CMSIS\GD\GD32F3x0\Source\ARM, and modify it as shown in *Figure 2-3. Modify the .s startup file* to enable the chip to perform self-checking after power-on.

Figure 2-3. Modify the .s startup file



3. Modify the project's scatter loading file (in this case, ..\GD32F3x0_IEC_Test\GD32350R_EVAL_Demo_Suites\Projects\IEC_Test\EWARM\I EC_TEST_BOOT_FLASH.icf) in the IAR environment. As shown in *Figure 2-4. Modify the scattered loading file*, adjust the sections for Flash and RAM sizes to match the current chip's specifications according to the datasheet.

Figure 2-4. Modify the scattered loading file

```
/*###ICF### Section handled by ICF editor, don't touch! ****/
    /*-Editor annotation file-*/
2
    /* IcfEditorFile="$TOOLKIT_DIR$\config\ide\IcfEditor\cortex_v1_0.xml" */
3
4
    /*-Specials-*/
5
    define symbol __ICFEDIT_intvec_start_ = 0x08000000;
    /*-Symbols-*/
6
7 define symbol __ICFEDIT_region_ROM_start_
8 define symbol __ICFEDIT_region_ROM_end__
                                                                      = 0 \times 08000000;
                                                                      = 0x0801FFFF;
    define symbol __ICFEDIT_region_RAM_start
                                                                     = 0x20000B0;
9
                                                                     = 0x20003FFF;
10
    define symbol __ICFEDIT_region_RAM_end_
    define symbol _____ICFEDIT_region_IECTEST_PARAM_start __ 0x20000040;
define symbol _____ICFEDIT_region_IECTEST_PARAM_end ____ = 0x200000B0;
11
    define symbol ____
12
```

4. Update the configuration for the 'Target' within 'Options for Target' in the IAR project settings, selecting the current chip model, as depicted in *Figure 2-5. Device Configuration*.

Category: General Options Static Analysis Runtime Checking C/C++ Compiler Library Options 2 MISRA-C:2004 MISRA-C:1998 Assembler Target Library Configuration Library Options 1 Output Output Converter Custom Build Processor variant **Build Actions** Linker Cortex-M4 ⊖ Core Debugger Simulator GD GD32F350xB "∎⊷ Device CADI CMSIS DAP None O CMSIS-Pack GDB Server I-jet/JTAGjet J-Link/J-Trace Endian mode Floating point settings TI Stellaris Nu-Link Little FPU VFPv4 single precision \sim PE micro OBig ST-LINK D registers Third-Party Driver 16 O BE32 TI MSP-FET TI XDS BE8 TrustZone DSP Extension Mode Secure Advanced SIMD (NEON)

Figure 2-5. Device Configuration

5. In the Options for the 'Project' node -> C/C++ Compiler -> Preprocessor, add the necessary preprocessor macros for the current project, mainly modifying the macros that match the type of the current chip, as shown in *Figure 2-6. Modify the precompiled macro*.

Figure 2-6. Modify the precompiled macro

Category:						Fac	tory Settings
General Options	🔄 Multi-file Compila	ition					
Static Analysis	🗌 Discard Unu	used Publics					
Runtime Checking							
C/C++ Compiler	MISRA-C:1	1998	En	codings		Extra O	ptions
Assembler Output Converter	Language 1	Language 2		Code	Optim	izations	Output
Custom Build	List	Preprocessor		Diagno	ostics	MISR	A-C:2004
Build Actions				-			
Linker	☐ Ignore star	ndard include di	recto	ories			
Debugger							
Simulator	Additional inc	lude directories:	(on	ne per line)		
CADI	\$PROJ DIR\$\						^
CMSIS DAP GDB Server	\$PROJ DIR\$	\\\GD32F3x	0 Fi	irmware L	ibrary\G	D32F3x0 s	t
I-jet/JTAGjet	\$PROJ DIR\$	\\\GD32F3x	0 Fi	irmware L	ibrary\C	MSIS\GD\	e i
J-Link/J-Trace	_	\\\GD32F3x	_	_	-		
TI Stellaris		\\\GD32F3x	_	_			<u> </u>
Nu-Link			-	-		-	
PE micro	Preinclude						
ST-LINK							
Third-Party Driver							
TI MSP-FET	Defined symb	ols: (one per line	e)	_			
TI XDS	USE_STDPER	PH DRIVER	\sim	Pre	eprocess	or output	to file
	GD32F350	-			Preserve	e commen	ts
	GD32F3x0 BC	DOT FLASH				e #line dir	-
	-	-			Generat	e #line dir	ecuves

6. Click on Options for the 'Project' node -> Linker -> Configuration -> Edit -> Memory Regions, and modify the boundary values of ROM (Flash) and RAM according to the current chip's datasheet, as shown in *Figure 2-7. Modify the boundary values in the project settings*.

Figure 2-7. Modify the boundary values in the project settings

Vector Table	Memory Regions	Stack/Heap Sizes	
ROM RAM	Start: 0x08000000 0x20000080		

7. In the Options for the 'Project' node -> Linker -> Checksum, change the End address to the size of the Flash, as shown in *Figure 2-8. Modify the checksum configuration in the project properties*, and add "--keep __checksum" in the Extra Options tab, as shown in *Figure 2-9. Add configurations to the Extra Options tab*.

Figure	2-8.	Modify	the	checksum	configuration	in	the	project	propertie	S
--------	------	--------	-----	----------	---------------	----	-----	---------	-----------	---

Config L	library	Inpu	ıt Optimiza	ations	Α	dvanced	Output	List
#define	Diagr	nostio	s Checks	ım	n Encodings		Extra C	ptions
⊡ Fill unu	used cod	le me	emory					
Fill pa	attern:		0xFF					
Start	Start address:		0x8000000	Ē	ind a	address:	0x801FF	FF
🗹 Ge	nerate c	heck	sum					
С	hecksum	n size	: 4 bytes 🕓	<u>۸</u>	lign	ment:	4	
A	lgorithm		CRC polyno	omia N	/	0x4C11D	B7	
	Res	ult in	full size			Initial valu	e	
С	Complement: As is			`	~	0xFFFFFFF	F	
Bit order: MS			MSB first	`	~	Use as	input	
	Reverse byte order within word							
Checksum unit size: 32-bit \checkmark								

Figure 2-9. Add configurations to the Extra Options tab

Config	Library	Input	Optimizations	Advanced	Output	List		
#define	e Diag	nostics	Checksum	Checksum Encodings				
	✓Use command line options							
<u>C</u> om	mand line	options:	(one per					
ke	ep _chec	ksum				\sim		
						× .		



2.2. Migration of certification library project in Keil environment

1. Modify the RAM and Flash boundaries in the gd32f3x0_test.h file according to the datasheet of the current chip model, as shown in *Figure 2-10. Modify the RAM and Flash boundaries in the gd32f3x0_test.h file*.

Figure 2-10. Modify the RAM and Flash boundaries in the gd32f3x0_test.h file

#if defined (ARMCC_VERSION) && (ARMCC_VERSION >= 6010050)				
<pre>#define FLASH_START #define FLASH_SIZE #define FLASH_SIZE_WORDS #define FLASH_END</pre>	((uint32_t *)0x08000000) ((uint32_t) <u>0x00020000 - 4)</u> (uint32_t)(((uint32 t)FLASH_END-(uint32_t)FLASH_START)/4) ((uint32_t *)0x08020000)				
#define FLASH_BLOCKNUM	((uint32_t)512uL) (uint32_t)(((uint32_t)FLASH_END-(uint32_t)FLASH_START) / FLASH_BLOCK_SIZE) (uint32_t)((FLASH_SIZE_WORDS) / FLASH_BLOCK_SIZE)				
<pre>#define IEC_TEST_PARAM_START #define IEC_TEST_PARAM_END</pre>	((uint32_t *)0x20000040) ((uint32_t *)0x200000B0)				
<pre>#define RAM_START #define RAM_END</pre>	(uint32_t *)0x20000000 (uint32_t *)0x20003F40				
<pre>extern void Reset_Handler(void); #define DefaultSystemStartUp() Reset_Handler() void test_fail_reset(void);</pre>					
= #endif /* (ARMCC_VERSION) && (ARMCC_VERSION >= 6010050) */				

2. Check the .s startup file in the Startup folder of the project directory to see if it matches the current chip model, and at the same time, add the ClassB library file to the project.



Figure 2-11.Add the ClassB library file to the Keil project

If the startup file is applicable to the current chip model, no modification is needed; otherwise,

you need to select the .s startup file suitable for the current chip model in the firmware library $folder \ GD32F3x0_Firmware_Library\CMSIS\GD\GD32F3x0\Source\ARM, \ and \ make \ the$ following modifications as indicated in red in the code.

			-
Stack_Size	EQU	0x00000400	
Stack_Mem initial_sp	AREA SPACE	STACK, NOINIT, REAE Stack_Size	WRITE, ALIGN=3
; <h> Heap Con ; <o> Heap ; </o></h>	-	tes) <0x0-0xFFFFFFFF:{	}>
Heap_Size	EQU	0x00000400	
	AREA	HEAP, NOINIT, READV	VRITE, ALIGN=3
heap_base Heap_Mem heap_limit	SPACI	E Heap_Size	
IMPOF	RT test_pr	erun	
	PRESER THUMB	RVE8	
;	AREA EXPORT	ector Mapped to at Addre RESET, DATA, READC Vectors Vectors_End Vectors_Size	
Vectors	DCD DCD DCD DCD	initial_sp test_prerun NMI_Handler USBFS_IRQHandler	; Top of Stack ; Reset Handler> test_prerun ; NMI Handler ; 83:USBFS
Vectors_End	DCD		, 63.6661 3
		HECKSUM, DATA, REAE Check_Sum	ONLY, ALIGN=2

	ALIGN	
Check_Sum	D	CD 0xEEF15A05
Vectors_Size	EQU	Vectors_EndVectors
	AREA	.text , CODE, READONLY
;/* reset Handler	*/	
Reset_Handler	PROC	
	•••••	
	•••••	
	•••••	

3. Modify the code in the scatter loading file IEC_TEST_BOOT_FLASH.sct, as shown in the following table, the parts marked in red need to be modified (modify the Flash boundary according to the current chip's datasheet; change the name of the .o file corresponding to the .s startup file).

· ************************************
. ************************************
; *** Scatter-Loading Description File generated by uVision ***
· ************************************
LR_IROM1 0x08000000 0x0001FFF8{
ER_IROM1 0x08000000
*.o (RESET, +First)
*(InRoot\$\$Sections)
.ANY (+RO)
}
; RW data
RW_IRAM1 0x200000B0 0x0002000
{
.ANY (+RW +ZI)
}

Scatter loading files can be modified by clicking the Edit button in "Options for Target --> Linker --> Scatter File", as shown in *Figure 2-12. Editing the scatter loading file*: Editing the scatter loading file.

Figure 2-12. Editing the scatter loading file

Device Target Output I	isting User C/C++	Asn Linker	Debug Utilities	
Use Memory Layout from Make RW Sections Po Make RO Sections Po Don't Search Standard Report 'might fail' Cond	osition Independent sition Independent d Libraries	X/O Base: R/O Base: R/W Base disable Warnings:	0x08000000 0x20000000	
Misc controls Linker -cpu Cortex-M4	OOT_FLASH.sct ifp *.o icrolib -strict -scatter ".\IEC	_TEST_BOOT_FLAS	H.sct"	Edt
	ок	Cancel D	efaults	Help

4. Modify the "Options for Target" under "Device" to select the current chip model, as shown in *Figure 2-13. Device Configuration*.

Figure 2-13. Device Configuration

Device Target Output Listing User	C/C++ (AC6) Asm Linker Debug Utilities
Software Packs Vendor: GigaDevice Device: GD32F350RB Toolset: ARM Search:	Software Pack Pack: GigaDevice.GD32F3x0_DFP.3.0.4 URL: <u>https://gd32mcu.com/data/documents/r</u>
GD32F350R6 GD32F350R8 GD32F350R8 GD32F4xx Series GD32F7xx Series GD32H7xx Series GD32H7xx Series GD32L23x Series GD32L23x Series GD32W51x Series	The GD32F3x0 device belongs to the value line of GD32 MCU family. It is a new 32-bit general-purpose microcontroller based on the Arm Cortex-M4 RISC core with best cost-performance ratio in terms of enhanced processing capacity, reduced power consumption and peripheral set. The Cortex-M4 core features implement a full set of DSP instructions to address digital signal control markets that demand an efficient, easy-to-use blend of control and signal processing capabilities. GD32F350 - ARM Cortex-M4 Core Frequency up to 108 MHz Flash access zero wait state Single-cycle multiplier and hardware divider

5. Add the necessary preprocessor macros for the current project that are consistent with the current chip in Options for Target -> C/C++ -> Preprocessor Symbols, as shown in <u>Figure</u> <u>2-14. Modify the precompiled macro</u>:

Figure 2-14. Modify the precompiled macro

Device Target Output Listing User C/C++ (AC6) Asm Linker Debug Utilities
Preprocessor Symbols Define: USE_STDPERIPH_DRIVER,GD32F3X0,GD32F350 GD32F3x0_BOOT_FLASH Undefine:
Language / Code Generation
Execute-only Code Warnings: AC5-like Warnings 🗸 Language C: c99
Optimization: -00 🔽 🗖 Tum Warnings into Errors Language C++: c++11 💌
Link-Time Optimization
Split Load and Store Multiple Read-Only Position Independent use RTTI
I One ELF Section per Function □ Read-Write Position Independent □ No Auto Includes
Include Paths Misc Controls
Compiler control string

6. When it is necessary to automatically calculate the CRC using batch processing, the

gen_crc.bat file needs to be added, as shown in Figure 2-15.Add a batch file.

Figure 2-15.Add a batch file

Device Target Output Listing User C/C++ Asm Linker Debug Utilities						
		1				
Command Items	User Command		Stop on Exi	S		
Before Compile C/C++ File						
		2	Not Specified			
Run #2		2	Not Specified			
Before Build/Rebuild						
Run #1		2	Not Specified			
Run #2		2	Not Specified			
ia After Build/Rebuild						
Run #1	gen_crc.bat	2	•			
Run #2		2	Not Specified			
🔲 <u>R</u> un 'After-Build' Conditionally						
✓ Beep When Complete	Start Debugging					

7. Finally, in the Utilities tab, select the CRC_LOAD.ini, as shown in the *Figure 2-16. Add a .ini configuration file*.

Figure 2-16. Add a .ini configuration file

Device Target Output Listing User C/C++	Asm Linker Debug Utilities
Configure Flash Menu Command	
Use Target Driver for Flash Programming	Use Debug Driver
Use Debug Driver	Settings IV Update Target before Debugging
Init File: \\CRC_LOAD.ini	Edit
O Use External Tool for Flash Programming	
Command:	
Arguments:	
🗖 Run Independent	
Configure Image File Processing (FCARM):	
Output File:	Add Output File to Group:
	Application
Image Files Root Folder:	Generate Listing

2.3. Migration of certification library project in eclipse environment

The migration steps of the authentication library project are similar in various environments,

but the compilation chains differ in each development environment, leading to different settings for the boundaries of RAM and Flash. The porting steps in the Eclipse environment are as follows:

1. Modify the RAM boundary in the macro ___GNU__ of the gd32f3x0_test.h file according to the datasheet to ensure that the entire space of the chip's Flash and RAM is detected, as shown in *Figure 2-17. Modify the RAM boundary in the gd32f3x0_test.h file*.

Figure 2-17. Modify the RAM boundary in the gd32f3x0_test.h file



2. Check if the .S startup file in the project is suitable for the current chip, and <u>Figure 2-18.</u> Add the ClassB library file to the Eclipse project.



Figure 2-18. Add the ClassB library file to the Eclipse project

If the startup file is applicable to the current chip model, no modification is needed; if it does not match, a .s startup file that is compatible with the current chip model needs to be selected again, and modifications should be made as shown in *Figure 2-19. Modify the startup file*.

Figure 2-19. Modify the startup file



3. Modify the scatter-loading file in the scatter-loading file directory, as shown in <u>Figure 2-21.</u> <u>Modify the Eclipse project to load files in a Id file</u> is the modified part, and adjust the size of Flash and RAM according to the current datasheet.

Figure 2-20. Scat	tter loading file location
-------------------	----------------------------



Figure 2-21. Modify the Eclipse project to load files in a ld file

```
2 ENTRY(Reset_Handler)
З
4 /* end of Stack */
5_estack = 0x20004000;
6
7 /* memory map */
8 MEMORY
9 {
                   : ORIGIN = 0x08000000, LENGTH = 128K
10
    FLASH (rx)
  iec_test (wxa!ri) : ORIGIN = 0x20000000, LENGTH = 0xB0
11
               : ORIGIN = 0x200000B0, LENGTH = 0x3F50 /*8K*/
12
   RAM (xrw)
   flash_end (rxai!w) : ORIGIN = 0x0801FFC0, LENGTH = 0x40
13
14 }
```

4. Modify the "Device name" configuration in "Debug Configurations->Debugger", select the model of the current chip, as shown in *Figure 2-22. Device name configuration*:

Nan	lame: gd32f350r_iec_test Debug										
	📄 Main 🎋 Debugger 🜘 Startup 🤤 Source 🔲 Common 🔀 SVD Path										
J	J-Link GDB Server Setup										
E	Start the J-Link 🛛	GDB server le	ocally				Connect to running t	arget			
E	Executable path:	\${jlink_path}	}/\${jlink_gdbse	erver}					E	Browse	Variables
	Actual executable:	D:/Program	Files/SEGGER	k/JLink/JLin	kGDBServer	rCL.ex	e				
		(to change it	t use the <u>glob</u>	al or <u>work</u>	space prefe	rence	s pages or the <u>proje</u>	<u>ct</u> prope	erties pa	ige)	
	Device name:	GD32F350R	В						<u>Su</u>	upported of	device names
E	Endianness:	Little	⊖ Big								
(Connection:	● USB	OIP				(USB serial or IP nan	ne/addr	ress)		
	Interface:	SWD	⊖ JTAG								
1	Initial speed:	Auto	OAdaptive	Fixed	1000	kHz					
(GDB port:	2331									
9	SWO port:	2332					Verify dow	nloads/	🗹 Initia	alize regis	sters on start
1	Telnet port:	2333					✓ Local host	only	Siler	nt	
l	Log file:										Browse
(Other options:	-singlerun -	strict -timeout	t 0 -nogui							

Figure 2-22. Device name configuration

Add the necessary precompiled macros for the current project in the engineering properties "C/C++ Build -> Settings -> Tool Settings -> Cross ARM GNU Assembler -> Preprocessor" and "C/C++ Build -> Settings -> Tool Settings -> Cross ARM GNU C Compiler -> Preprocessor", mainly modifying the precompiled macros that conform to the current chip type (as shown in *Figure 2-23. Modify the assembly compiler's precompiled macros*).

Figure 2-23. Modify the assembly compiler's precompiled macros

🛞 Tool Settings 🛞 Toolchains 📕 D	evices 🎤 Build Steps 譻 Build Artifact 🗟 Binary Parse	rs 🔕 Error P 🔸 🕨
 Target Processor Optimization Warnings Debugging S GNU ARM Cross Assembler Preprocessor Includes Warnings Miscellaneous S GNU ARM Cross C Compiler Preprocessor Includes Optimization Warnings Miscellaneous S GNU ARM Cross C Linker 	Do not search system directories (-nostdinc) Preprocess only (-E) Defined symbols (-D) USE_STDPERIPH_DRIVER GD32F350 GD32F3x0_BOOT_FLASH	1 k k k k k k k k k k k k k k k k k k k
👺 General 🍅 Libraries 🌺 Miscellaneous	Undefined symbols (-U)	a 월 전 신
 S GNU ARM Cross Print Size General 		

6. Modify the command in "C++ Build -> Settings -> Build Steps -> Post-build steps -> Command" as shown in *Figure 2-24.Modify the post-build command* to the ELF file of the current project name.

Figure 2-24.Modif	y the post-build command

✓ Resource	
Linked Resources	
Resource Filters	Configuration: Debug [Active]
Builders	
✓ C/C++ Build	
Build Variables	🥸 Tool Settings 🚯 Toolchains 🔳 Devices 🎤 Build Steps 🥊 Build Artifact 🗟 Binary Parsers 😣 Error P 📢
Environment	Pre-build steps
Logging	Command:
Settings	
Tool Chain Editor	×
> C/C++ General	Description:
Linux Tools Path	
✓ MCU	
ARM Toolchains Path	Post-build steps
Build Tools Path	Command:
OpenOCD Path	{cross prefix}{cross objcopy}{cross suffix} -O ihex "gd32f350r iec test.elf" "Project.hex";\gen crc.bat;
pyOCD Path	
QEMU Path	Description:
SEGGER J-Link Path	~ ·
Project References	
Run/Debug Settings	
> Task Repository	
Task Tags	
> Validation	
WikiText	

7. Modify the configuration of the executable file in "Debug Configurations->Startup", and select the Project.hex file generated by the current workspace compilation, as shown in *Figure 2-25. Modify the configuration of the executable file*.

Figure 2-25. Modify the configuration of the executable fi	file
--	------

Name: gd32f350r_iec_test Debug		
📄 Main 🕸 Debugger 🅟 Startup 🧐 Source 🗔 Common 🔀 SVD Path		
		~
Load Symbols and Executable		
✓Load symbols		
OUse project binary: gd32f350r_iec_test.elf		
Use file: \${workspace_loc:\gd32f350r_iec_test\Debug\Project.hex}	Workspace	File System
Symbols offset (hex):		
∠ Load executable		
Use project binary: gd32f350r_iec_test.elf		
⊖Use file:	Workspace	File System
Executable offset (hex):		
Runtime Options		
RAM application (reload after each reset/restart)		
Run/Restart Commands		
☑ Pre-run/Restart reset Type: (always executed at Restart)		
		^
		~
Set program counter at (hex):		
Set breakpoint at:		
☑ Continue		

3. Test results in three different IDEs

Follow the steps in Chapter 2 to modify the code and configure the project, compile and download the project in various environments, and run it. The test results are shown in *Figure* <u>3-1.Test Results</u>.

Figure 3-1.Test Results

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CPU Test(PreRun) Success!
\ldots Power reset or software reset, next step —> FWDGT reset test \ldots
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CPV Test(PreRun) Success!
FWDGT reset FWDGT reset test OK, next step> WWDGT reset test
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CPV Test(PreRun) Success!
FWDGT reset WWDGT reset
WWDGT reset test OK, WDGT test completed
VFull RAM Test Success!
FLASH CRC32 Test(PreRun) Success! €
Clock Frequency Test Success!
Program counter test(PreRun) Success!y
********************************* Main program starts **************************
FLASH CRC(Run-Time) Test running! Next Address> 0x080001fc
FLASH CRC(Run-Time) Test running! Next Address ->> 0x080003f8
FLASH CRC(Run-Time) Test running! Next Address> 0x080005f4
FLASH CRC(Run-Time) Test running! Next Address -> 0x080007f0

4. Revision history

Table 4-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Jul.9, 2024

Important Notice

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

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which 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 instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as 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.

© 2024 GigaDevice - All rights reserved