GigaDevice Semiconductor Inc.

Guideline for migrating the IEC60730 ClassB library onto GD32F4xx series

Application Note AN156

Revision 1.0

(Jul. 2024)

Table of Contents

Tabl	e of Contents	2
List	of Figures	3
List	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	2. Migration of certification library project in Keil environment	12
2.3	8. Migration of certification library project in eclipse environment	18
3.	Test results in three different IDEs	26
4.	Revision history	27

List of Figures

Figure 2-1. Modify the RAM boundary in the gd32f4xx_test.h	. 6
Figure 2-2. Startup file check	. 7
Figure 2-3. Add the ClassB library file to the IAR project	. 7
Figure 2-4. Modify the .s startup file	. 8
Figure 2-5. Modify the scattered loading file	. 8
Figure 2-6. Device Configuration	. 9
Figure 2-7. Modify the precompiled macro	10
Figure 2-8. Modify the boundary values in the project settings	10
Figure 2-9. Modify the checksum configuration in the project properties	11
Figure 2-10. Add configurations to the Extra Options tab	11
Figure 2-11. Modify the RAM and Flash boundaries in the gd32f4xx_test.h file	12
Figure 2-12. Check the startup file	
Figure 2-13.Add the ClassB library file to the Keil project	13
Figure 2-14. Editing the scatter loading file	16
Figure 2-15. Device Configuration	16
Figure 2-16. Modify the precompiled macro	17
Figure 2-17.Add a batch file	17
Figure 2-18. Add a .ini configuration file	18
Figure 2-19. Modify the RAM boundary in the gd32f4xx_test.h file	19
Figure 2-20. Startup file check	19
Figure 2-21. Add the ClassB library file to the Eclipse project	20
Figure 2-22. Modify the startup file	21
Figure 2-23. Scatter loading file location	21
Figure 2-24. Modify the Eclipse project to load files in a ld file	
Figure 2-25. Device name configuration	22
Figure 2-26. Modify the assembly compiler's precompiled macros	23
Figure 2-27.Modify the post-build command	24
Figure 2-28. Modify the configuration of the executable file	25
Figure 3-1.Test Results	26



List of Tables



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 GD32F4xx 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 GD32F470IK 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 gd32f4xx_test.h file according to the datasheet of the chip, as shown in *Figure 2-1. Modify the RAM boundary in the gd32f4xx_test.h*. Modify the RAM boundary in the gd32f4xx_test.h as indicated.

Figure 2-1. Modify the RAM boundary in the gd32f4xx_test.h

DifferIAR_SYSTEMS_ICC
<pre>extern.uint32_tICFEDIT_region_ROM_start; extern.uint32_tICFEDIT_region_ROM_start; extern.uint32_tICFEDIT_region_RAM_start; extern.uint32_tICFEDIT_region_RAM_start; extern.uint32_tICFEDIT_region_IECTEST_PARAM_start; extern.uint32_tICFEDIT_region_IECTEST_PARAM_end;</pre>
<pre>#define -FLASH_START(unsigned -char .*) &ICFEDIT_region_ROM_start(unsigned -int) &ICFEDIT_region_ROM_end(unsigned -int) &ICFEDIT_region_ROM_start+.1./*.FLASH_SIZE .in .byte .*/ #define -FLASH_SIZERORD(unsigned -int) &ICFEDIT_region_ROM_end(unsigned -int) &</pre>
<pre>#define -FLASH_BLOCK_SIZE(uint32_t) 512uL) #define -FLASH_BLOCKNUMuint32_t) ((FLASH_SIZE) // -FLASH_BLOCK_SIZE) #define -FLASH_BLOCKNUM_WORDS ((uint32_t) (FLASH_SIZE_WORDS ./ -FLASH_BLOCK_SIZE))</pre>
<pre>#define.RAM_START</pre>
<pre>#define TCMRAM_START(uint32_t.*)0x10000000 #define TCMRAM_END(uint32_t.*)0x1000FFC0</pre>
<pre>#define.IEC_TEST_PARAM_START(uint32_t.*)(6_ICFEDIT_region_IECTEST_PARAM_start_)) #define.IEC_TEST_PARAM_END(uint32_t.*)(6_ICFEDIT_region_IECTEST_PARAM_end_))</pre>
<pre>extern.voidiar_program_start(void); extern.void.Reset_Handler(void); #define.DefaultSystemStartDp().Reset Handler()</pre>
<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, as shown in *Figure 2-2. Startup file check*. At the same time, add the ClassB library file to the project.

Figure 2-2. Startup file check







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 GD32F4xx_Firmware_Library\CMSIS\GD\GD32F4xx\Source\ARM, and modify it as shown in *Figure 2-4. Modify the .s startup file* to enable the chip to perform self-checking after power-on.

AN156 Guideline for migrating the IEC60730 ClassB library onto GD32F4xx series

Figure 2-4. Modify the .s startup file

	MODULE	?cstartup	
	;; Forw		
	SECTION	CSTACK: DATA: NOROOT (3)	
	SECTION	.intvec:CODE:NOROOT(2)	
	EXTERN	test_prerun	
	EXTERN	iar_program_start	
	EXTERN	SystemInit	
	PUBLIC	vector_table	
	DATA		
vecto	r_table		
	DCD	sfe (CSTACK)	; top of stack
	DCD	test_prerun	; Reset Handler> test_prerun
	DCD	MAT N dlaw	; Vector Number 2.NMI Handler
	DCD	NMI_Handler HardFault Handler	; Vector Number 2,NM1 Handler ; Vector Number 3,Hard Fault Handler
	DCD		
	DCD	MemManage_Handler	; Vector Number 4, MPU Fault Handler
	100000	BusFault_Handler	; Vector Number 5,Bus Fault Handler
	DCD	UsageFault_Handler	; Vector Number 6,Usage Fault Handler : Reserved
		0	; Reserved
	DCD		; Reserved
		0	
	DCD	•	; Reserved
		SVC_Handler	; Vector Number 11, SVCall Handler
	DCD	DebugMon_Handler	; Vector Number 12, Debug Monitor Handler
	DCD	0	; Reserved
	DCD	PendSV_Handler	; Vector Number 14, PendSV Handler
	DCD	SysTick_Handler	; Vector Number 15, SysTick Handler

3. Modify the project's scatter loading file (in this case, ..\GD32F4xx_IEC_Test\GD32470I_EVAL_Demo_Suites\Projects\IEC_Test\EWARM\I EC_TEST_BOOT_FLASH.icf) in the IAR environment. As shown in *Figure 2-5. 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 2-5. Modify the scattered loading file

1	/*###ICF### Section handled by ICF editor, don't touch! ****/
2	/*-Editor annotation file-*/
3	/* IcfEditorFile="\$TOOLKIT_DIR\$\config\ide\IcfEditor\cortex_v1_0.xml" */
4	/*-Specials-*/
5	<pre>define symbolICFEDIT_intvec_start = 0x08000000;</pre>
6	/*-Symbols-*/
7	<pre>define symbolICFEDIT_region_ROM_start =0x08000000;</pre>
8	<pre>define symbolICFEDIT_region_ROM_end = 0x082FFFFF;</pre>
9	<pre>define symbolICFEDIT_region_RAM_start = 0x200000B0;</pre>
10	<pre>define symbolICFEDIT_region_RAM_end = 0x2002FFFF;</pre>
11	<pre>define symbolICFEDIT_region_IECTEST_PARAM_start_ = 0x20000040;</pre>
12	define symbol ICFEDIT region IECTEST PARAM end = 0x200000B0;

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 2-6. Device Configuration*.

Figure 2-6. Device Configuration

Category: General Options Static Analysis Runtime Checking					
C/C++ Compiler Assembler	· · · · · ·	Options 2			MISRA-C:1998
Output Converter	Target	Output	Library Co	nfiguration	Library Options 1
Custom Build Build Actions	Processo	r variant —			
Linker Debugger	⊖ Core	Co	rtex-M4	\sim	
Simulator CADI	Device	GD	GD32F470xK		Ĩ⊒+
CMSIS DAP GDB Server I-jet/JTAGjet		S-Pack No	ne		
J-Link/J-Trace TI Stellaris	Endian m	ode - F	loating point	settings	
Nu-Link PE micro	LittleBig		FPU	VFPv4 single	e precision 🗸
ST-LINK					
Third-Party Driver TI MSP-FET	⊖ BE:	32	D registers	16 ~	
TI XDS	BE8	в			
	☑ DSP E	xtension ced SIMD (N	EON)	☐ TrustZo Mode Se	

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 2-7. Modify the precompiled macro*.

Figure 2-7. Modify the precompiled macro

Category: Factor						
tatic Analysis		used Publics				
Runtime Checking						
C/C++ Compiler	MISRA-C:	:1998	Encodings		Extra Options	
output Converter	Language 1	Language 2	ige 2 Code		Optimizations	
ustom Build	List	Preprocessor	Diagn	ostics	MISR	A-C:2004
ebugger Simulator CADI CMSIS DAP GDB Server I-jet/JTAGjet J-Link/J-Trace TI Stellaris Nu-Link PE micro	\$PROJ_DIR\$ \$PROJ_DIR\$ \$PROJ_DIR\$ \$PROJ_DIR\$	clude directories: \ \\\\GD32F4x \\\\GD32F4x \\\\GD32F4x \\\\GD32F4x	x_Firmware_I x_Firmware_I x_Firmware_I	Library\GD Library\CN Library\CN	ISIS\GD\ ISIS	
ST-LINK Third-Party Driver TI MSP-FET TI XDS		ools: (one per line RIPH_DRIVER		eprocesso Preserve		

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 2-8. Modify the boundary values in the project settings*.

Figure 2-8. Modify the boundary values in the project settings

Vector Table	Memory Regions Stack/Heap Sizes	
ROM RAM	Start: End: 0x08000000 0x082FFFFF 0x200000B0 0x2002FFFF	

7. In the Options for the 'Project' node -> Linker -> Checksum, change the End address to the size of the Flash, as shown in *Figure 2-9. Modify the checksum configuration in the project properties*, and add "--keep __checksum" in the Extra Options tab, as shown in *Figure 2-10. Add configurations to the Extra Options tab*.

Guideline for migrating the IEC60730 ClassB library onto GD32F4xx series

-									
Config	Library	Inpu			A	dvanced	Output	List	
#define	e Diag	nostic	s Checksur	m	En	codings	Extra C	Options	
✓ Fill unused code memory									
Fill	pattern:		0xFF	xFF					
Start address:			0x8000000	E	nd a	address:	0x82FFFFF		
\checkmark	Generate o	hecks	um						
	4 bytes \sim	A	lign	ment:	4				
	Algorithm	1	CRC polynor	nie 🔻	/	0x4C11DB7			
	Res	ult in	full size		Г	Initial valu	e		
	Complement: As is				/	0xFFFFFF	F		
Bit order: MSB first				`	/	Use as	input		
Reverse byte order within word									
Checksum unit size: 32-bit \checkmark									

Figure 2-9. Modify the checksum configuration in the project properties

Figure 2-10. Add configurations to the Extra Options tab

Config	Library	Input	Optimizations	Advanced	Output	List	
#define	e Diagnostics Checksum Encodings		Extra Options				
	✓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 gd32f4xx_test.h file according to the datasheet of the current chip model, as shown in *Figure 2-11. Modify the RAM and Flash boundaries in the gd32f4xx_test.h file*.

Figure 2-11. Modify the RAM and Flash boundaries in the gd32f4xx_test.h file

<pre>#define FLASH_START ((uint32_t.*)0x08000000) #define FLASH_SIZE ((uint32_t)0x00300000-4) #define FLASH_SIZE WORDS ((uint32_t)((uint32_t)FLASH_END-(uint32_t)FLASH_START)/4) #define FLASH_END ((uint32_t.*)0x08300000)</pre>
<pre>#define FLASH_BLOCK_SIZE</pre>
<pre>#define IEC_TEST_PARAM_START ((uint32_t .*) 0x20000040) #define IEC_TEST_PARAM_END ((uint32_t .*) 0x200000B0)</pre>
<pre>#define RAM_START</pre>
<pre>#define TCMRAM_START</pre>

2. Check the .s startup file in the Startup folder of the project directory to see if it matches the current chip model, as shown in *Figure 2-12. Check the startup file*, and at the same time, add the ClassB library file to the project.

Figure 2-12. Check the startup file



GigaDevice Guideline for migrating the IEC60730 ClassB library onto GD32F4xx series



Figure 2-13.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 GD32F4xx_Firmware_Library\CMSIS\GD\GD32F4xx\Source\ARM, and make the following modifications as indicated in red in the code.

5		
Stack_Size	EQU	0x00000400
	AREA	STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem	SPACE	Stack_Size
initial_sp		
; <h> Heap Con ; <o> Heap ; </o></h>	-	tes) <0x0-0xFFFFFFFF:8>
Heap_Size	EQU	0x0000400
	AREA	HEAP, NOINIT, READWRITE, ALIGN=3
heap_base		
Heap_Mem	SPAC	E Heap_Size
heap_limit		

IMPOR	T test_pre	erun	
	PRESER THUMB	VE8	
;	AREA EXPORT EXPORT	ector Mapped to at Addres RESET, DATA, READON Vectors Vectors_End Vectors_Size	
Vectors			; Reset Handler> test_prerun
	DCD	USBFS_IRQHandler	; 83:USBFS
Vectors_End		IECKSUM, DATA, READ(Check_Sum	ONLY, ALIGN=2
Check_Sum		CD 0xEEF15A05	
Vectors_Size	EQU	Vectors_EndVect	ors
	AREA	.text , CODE, READON	LY
;/* 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).

 GigaDevice Guideline for migrating the IEC60730 ClassB library onto GD32F4xx series

```
; *** Scatter-Loading Description File generated by uVision ***
LR_IROM1 0x08000000 0x002FFF8 {
   ER_IROM1 0x08000000 0x002FFF8 {
       *.o (RESET, +First)
       *(InRoot$$Sections)
       .ANY (+RO)
   }
                .....
                .....
                .....
   ; RW data
   RW_IRAM1 0x20000B0 0x002E000
   {
       .ANY (+RW +ZI)
   }
   ; stack overflow test
   STACK_IRAM2 0x2002E0B0 UNINIT 0x00001F50
   {
       gd32f4xx_test_param.o (STACK_OV_TEST, .bss.STACK_OV_TEST)
       startup_gd32f450_470.o (STACK, +Last)
   }
}
LR_IROM2 0x082FFFFC 0x0000004 {
   ER_IROM2 0x082FFFFC 0x0000004
   {
       *.o (CHECKSUM, +Last)
   }
}
```

Scatter loading files can be modified by clicking the Edit button in "Options for Target --> Linker --> Scatter File", as shown in *Figure 2-14. Editing the scatter loading file*: Editing the scatter loading file.

Figure 2-14. Editing the scatter loading file

Device Target Output I	isting User C	/C++ Asn Li	inker Debug	Utilities	
Use Memory Layout from Make RW Sections P Make RO Sections Po Don't Search Standar	osition Independent osition Independent d Libraries	R/0	Base: 0x08000 Base: 0x08000 /Base 0x20000 mings: 0		
Misc controls Linker control -cpu Cortex-M4 -library_type=m	OOT_FLASH.sct	".VIEC_TEST_BOOT	_FLASH.sct"		Edt
string	ОК	Cancel	Defaults		¥ Help

4. Modify the "Options for Target" under "Device" to select the current chip model, as shown in *Figure 2-15. Device Configuration*.

Figure 2-15. Device Configuration

Device Target Output Listing Vser	C/C++ Asm Linker Debug Utilities	
Software Packs	_	
Vendor: GigaDevice	Software Pack	
Device: GD32F470IK	Pack: GigaDevice.GD32F4xx_DFP.3.0.2	
Toolset: ARM	URL: <u>http://gd32mcu.com/data/documents/ps</u>	
Search:		
GD32F470IK	GD32 is a new 32-bit high performance, low power consumption universal microcontroller family powered by the ARM Cortex-M4 RISC	^
GD32F470VE GD32F470VG	core which targeted at various MCU application areas. GD32 family integrates features to simplify system design and provide	
GD32F470VI	customers wide range of comprehensive and superior cost effective MCU portfolios with proven technology and great innovation.	
GD32F470VK	GD32 family includes entry line, performance line and connectivity	
GD32F470ZE	line currently.	
GD32F470ZG	GD32F470 - ARM Cortex-M4 Core	
GD32F470ZI	Frequency up to 240 MHz Flash access zero wait state	
GD32F470ZK	Single-cycle multiplier and hardware divider	
	Memories	>

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 *Figure*

2-16. Modify the precompiled macro:

Figure 2-16. Modify the precompiled macro

Device Target Output Listing U	ser C/C++ Asm Linker Debug Vti	lities
Preprocessor Symbols		
Define: GD32F470,VECT_TAB_FL	ASH	
U <u>n</u> define:		
Language / Code Generation		
Execute-only Code	Strict ANSIC Warning	gs: All Warnings 💌
Optimization: Level 0 (-00)	Enum <u>C</u> ontainer always int	🗖 Thum <u>b</u> Mode
Optimize <u>f</u> or Time	Plain Char is Signed	No Auto Includes
Split Load and Store Multiple	Read-Only Position Independent	C99 Mode
✓ One ELF Section per Function	Read-Write Position Independent	GNU extensions
Include\\GD32F4xx_Firmwa	are_Library\CMSIS\GD\GD32F4xx\Include;\\\	GD32F4xx_Firmware
Misc Controls		
	AICROLIB -g -00apcs=interworksplit_sections -I	^
string	are_Library/CMSIS/GD/GD32F4xx/Include -I	¥

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 2-17.Add a batch file*.

Figure 2-17.Add a batch file

Device Target Output Listin	g User C/C++ Asm Linker Debug Ut	ilit	ies	
Command Items	User Command		Stop on Exi	S
Before Compile C/C++ File		-21	Not Specified	
Run #2			Not Specified	
⊡ Before Build/Rebuild				
Run #1		2	Not Specified	
Run #2		2	Not Specified	
🖮 After Build/Rebuild		_		
Run #1	gen_crc.bat	2	Not Specified	
— Run #2		2	Not Specified	
🔲 Run 'After-Build' Conditionally				
✓ Beep When Complete	Start Debugging			

7. Finally, in the Utilities tab, select the CRC_LOAD.ini, as shown in the Figure 2-18. Add

a .ini configuration file.

Figure 2-18. Add a .ini configuration file

Device Target Output Listing Vser C/C++	Asm Linker Debug Vtilities
Configure Flash Menu Command	
Use Target Driver for Flash Programming	Vse Debug Driver
Use Debug Driver	Settings Update Target before Debugging
Init File: \CRC_LOAD.ini	Edit
C 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 gd32f4xx_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 2-19. Modify the RAM boundary in the gd32f4xx_test.h file*.

Figure 2-19. Modify the RAM boundary in the gd32f4xx_test.h file

```
#if ·! defined · (__ARMCC_VERSION) ·&& · (defined · (__GNUC_
 #define.FLASH_START.....((uint32_t.*)0x08000000)
#define ·FLASH_BLOCK_SIZE · · · · · · ((uint32_t)512uL)
#define ·FLASH_BLOCKNUM · · · · · · · · (uint32_t) ( ( (uint32_t) FLASH_END- (uint32_t) FLASH_START) · / ·FLASH_BLOCK_SIZE)
#define ·FLASH_BLOCKNUM_WORDS · · · (uint32_t) ( (FLASH_SIZE_WORDS) · / ·FLASH_BLOCK_SIZE)
 #define ·IEC_TEST_PARAM_START · · · · ((uint32_t ·*))0x20000040)
 #define ·IEC_TEST_PARAM_END ····· ((uint32_t ·*) 0x200000B0)
 #define .RAM START ..... (uint32 t.*) 0x2000000
#define RAM_DATAAREA_END......(uint32_t.*)(0x2000B00.-.0x40)
#define RAM_STACK_START.....(uint32_t.*)0x2001E000
 #define RAM END ..... (uint 32 t .*) 0x2002FFC0
 #define ·TCMRAM_START · · · · · · · · (uint32_t ·*) 0x1000000
 #define ·DefaultSystemStartUp()
 void test_fail_reset(void);
 #endif ·/* ·(!defined ·(__ARMCC_VERSION)) ·&& ·(defined ·(__GNUC_
                                                               )) •*/
```

2. Check if the .S startup file in the project is suitable for the current chip, as shown in the *Figure 2-20. Startup file check*, and *Figure 2-21. Add the ClassB library file to the Eclipse project*.

Figure 2-20. Startup file check



GigaDevice Guideline for migrating the IEC60730 ClassB library onto GD32F4xx series

Figure 2-21. 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 2-22. Modify the startup file*.

Guideline for migrating the IEC60730 ClassB library onto GD32F4xx series

Figure 2-22. Modify the startup file

```
18
   bl test_prerun /* SystemInit */
49
50
51/* Call SystemInit function */
52
  bl SystemInit
53///* Call static constructors */
54// bl __libc_init_array
55/*Call the main function */
56 bl main
57
   bx lr
58.size Reset_Handler, .-Reset_Handler
60
61
52
      .section .text.Default_Handler, "ax", %progbits
63Default_Handler:
64 Infinite Loop:
65
   b Infinite_Loop
66
    .size Default Handler, .-Default Handler
67
68
69
     .section .isr_vector,"a",%progbits
     .global __gVectors
73
    gVectors:
74
5
                        .word _estack
                                                                 /* Top of Stack */
```

3. Modify the scatter-loading file in the scatter-loading file directory, as shown in <u>Figure 2-24.</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.





GigoDevice Guideline for migrating the IEC60730 ClassB library onto GD32F4xx series

Figure 2-24. Modify the Eclipse project to load files in a ld file

```
1
2 ENTRY(Reset_Handler)
З
4 /* end of Stack */
5_estack = 0x20030000;
6
7 /* memory map */
8 MEMORY
9 {
    FLASH (rx) : ORIGIN = 0x08000000, LENGTH = 3072K
                                                            /*3072K*/
10
   iec_test (wxa!ri) : ORIGIN = 0x20000000, LENGTH = 0xB0
11
   RAM (xrw) : ORIGIN = 0x200000B0, LENGTH = 0x2FF50 /*256K*/
12
   flash_end (rxai!w) : ORIGIN = 0x082FFFC0, 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 2-25. Device name configuration*:

Figure 2-25. Device name configuration

J-Link GDB Server	Setup							
Start the J-Link	GDB server l	ocally				Connect to running target		
Executable path:	C:\Program	C:\Program Files (x86)\SEGGER\/Link\/LinkGDBServerCL.exe						
Actual executable:	C:\Program	C:\Program Files (x86)\SEGGER\JLink\JLinkGDBServerCLexe						
	(to change it	t use the <u>glob</u> a	<u>al or worksp</u>	ace preference	es pages or the <u>project</u> pro	operties page)		
Device name:	GD32F470I	ĸ						Suppo
Endianness:	Little	⊖ Big						
Connection:	USB	OIP			(USB serial or IP name/ad	dress)		
Interface:	SWD	⊖ JTAG						
Initial speed:	Auto	○ Adaptive	Fixed	1000 kHz				
GDB port:	2331							
SWO port:	2332						✓ Verify downloads	🗹 Initialize
Telnet port:	2333						✓ Local host only	Silent
Log file:								
Other options:	-singlerun -	strict -timeout	t 0 -nogui					

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 2-26. Modify the assembly compiler's precompiled macros*).



Figure 2-26. Modify the assembly compiler's precompiled macros

6. Modify the command in "C++ Build -> Settings -> Build Steps -> Post-build steps -> Command" as shown in *Figure 2-27.Modify the post-build command* to the ELF file of the current project name.

Properties for gd32f470i_iec_test Х type filter text ← → ⇒ <</p> Settings > Resource Builders Configuration: Debug [Active] Manage Configurations... ✓ C/C++ Build **Build Variables** Environment 🛞 Tool Settings 🛞 Toolchains 📕 Devices 📋 Container Settings 🎤 Build Steps 🙅 Build Artifact 扇 Binary Parse 💶 Logging Settings Pre-build steps Tool Chain Editor Command: > C/C++ General \sim Linux Tools Path > MCU Description: Project Natures \sim Project References Run/Debug Settings Post-build steps Task Repository Command: Task Tags {cross_prefix}{cross_objcopy}{cross_suffix} -O ihex "gd32f470i_iec_test.elf" "Project.hex";..\\gen_crc.bat; > Validation Description: WikiText \sim Restore Defaults Apply ? Apply and Close Cancel

Figure 2-27.Modify the post-build command

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 2-28. Modify the configuration of the executable file*.

Figure	2-28	Modify	the	confid	nuration	of th	ne er	xecutab	le file
Iguie	2-20.	wouny	uie	Conny	guration		10 0	recutab	

Name: gd32f470i_iec_test	Debug		
📄 Main 🕸 Debugger 🞙	Startup 🛛 🦆 Source 🔲 Common 🔀 SVD Path		
Enable SWO CPU fr			
Load Symbols and Exec	utable		
✓ Load symbols			
⊖ Use project binary:			
Ose file:	{workspace_loc:\gd32f470i_iec_test\Debug\Project.hex}	Workspace	File System
Symbols offset (hex):			
✓ Load executable			
⊖ Use project binary:			
● Use file:		Workspace	File System
Executable offset (hex)			
Runtime Options			
	oad after each reset/restart)		
-Run/Restart Commands			
Pre-run/Restart rese	t Type: (always executed at Restart)		
			~
			~
Set program counter	r at (hex):		
Set breakpoint at:	test_prerun		

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