GigaDevice Semiconductor Inc.

Guideline for migrating the IEC60730 ClassB library onto GD32F4xx series

Application Note

AN156

Revision 1.0

(Jul. 2024)

Table of Contents

List of Figures

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.

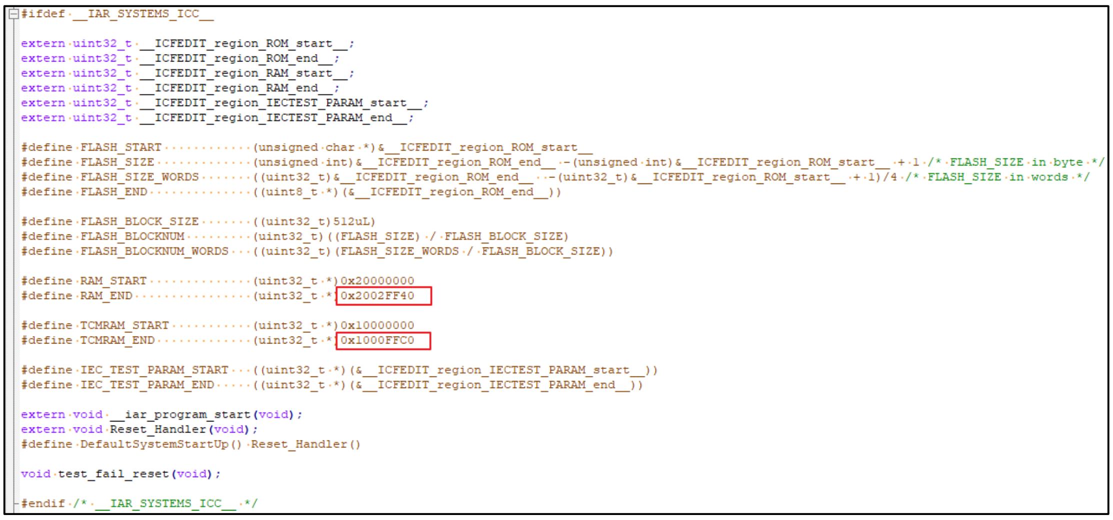
2. Migration of the IEC60730 class B certification library

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



```
#ifdef __IAR_SYSTEMS_ICC__

extern uint32_t __ICFEDIT_region_ROM_start__;
extern uint32_t __ICFEDIT_region_ROM_end__;
extern uint32_t __ICFEDIT_region_RAM_start__;
extern uint32_t __ICFEDIT_region_RAM_end__;
extern uint32_t __ICFEDIT_region_IECTEST_PARAM_start__;
extern uint32_t __ICFEDIT_region_IECTEST_PARAM_end__;

#define FLASH_START             (unsigned char *)&__ICFEDIT_region_ROM_start__
#define FLASH_SIZE              (unsigned int)&__ICFEDIT_region_ROM_end__ -(unsigned int)&__ICFEDIT_region_ROM_start__ + 1 /* FLASH_SIZE in byte */
#define FLASH_SIZE_WORDS        ((uint32_t)&__ICFEDIT_region_ROM_end__  -(uint32_t)&__ICFEDIT_region_ROM_start__ + 1)/4 /* FLASH_SIZE in words */
#define FLASH_END               ((uint8_t *)(&__ICFEDIT_region_ROM_end__))

#define FLASH_BLOCK_SIZE        ((uint32_t)512uL)
#define FLASH_BLOCKNUM          (uint32_t)((FLASH_SIZE) / FLASH_BLOCK_SIZE)
#define FLASH_BLOCKNUM_WORDS    ((uint32_t)(FLASH_SIZE_WORDS / FLASH_BLOCK_SIZE))

#define RAM_START               (uint32_t *)0x20000000
#define RAM_END                 (uint32_t *)0x2002FF40

#define TCMRAM_START            (uint32_t *)0x10000000
#define TCMRAM_END              (uint32_t *)0x1000FFC0

#define IEC_TEST_PARAM_START    ((uint32_t *)(&__ICFEDIT_region_IECTEST_PARAM_start__))
#define IEC_TEST_PARAM_END      ((uint32_t *)(&__ICFEDIT_region_IECTEST_PARAM_end__))

extern void __iar_program_start(void);
extern void Reset_Handler(void);
#define DefaultSystemStartUp() Reset_Handler()

void test_fail_reset(void);

#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

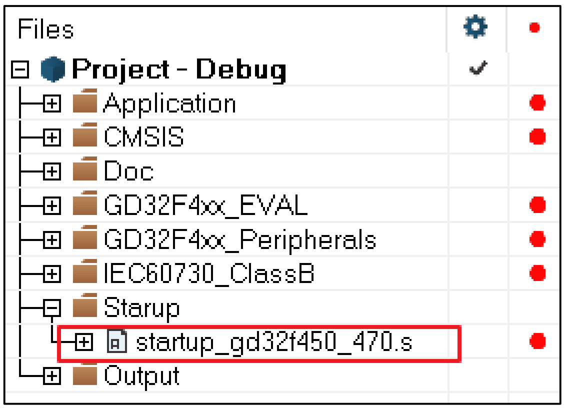
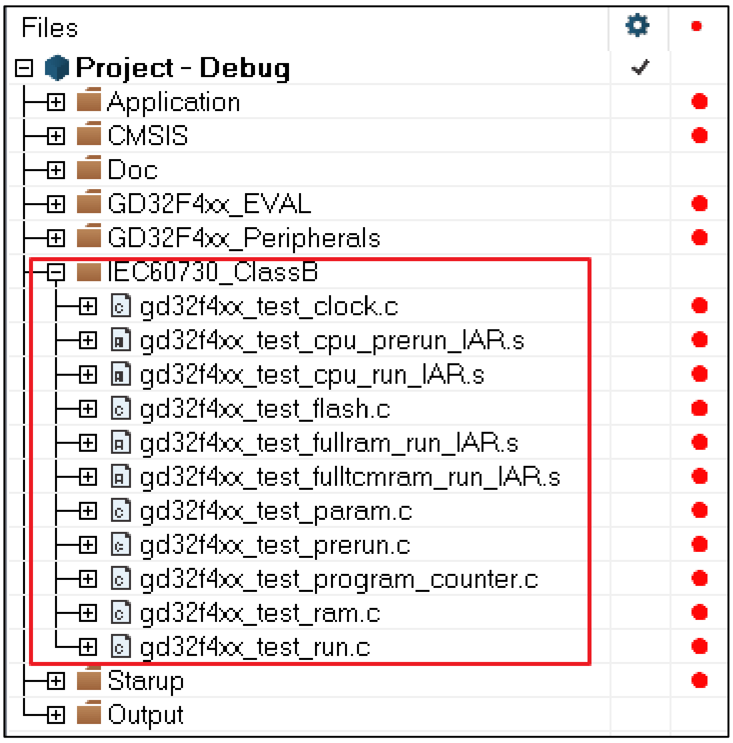
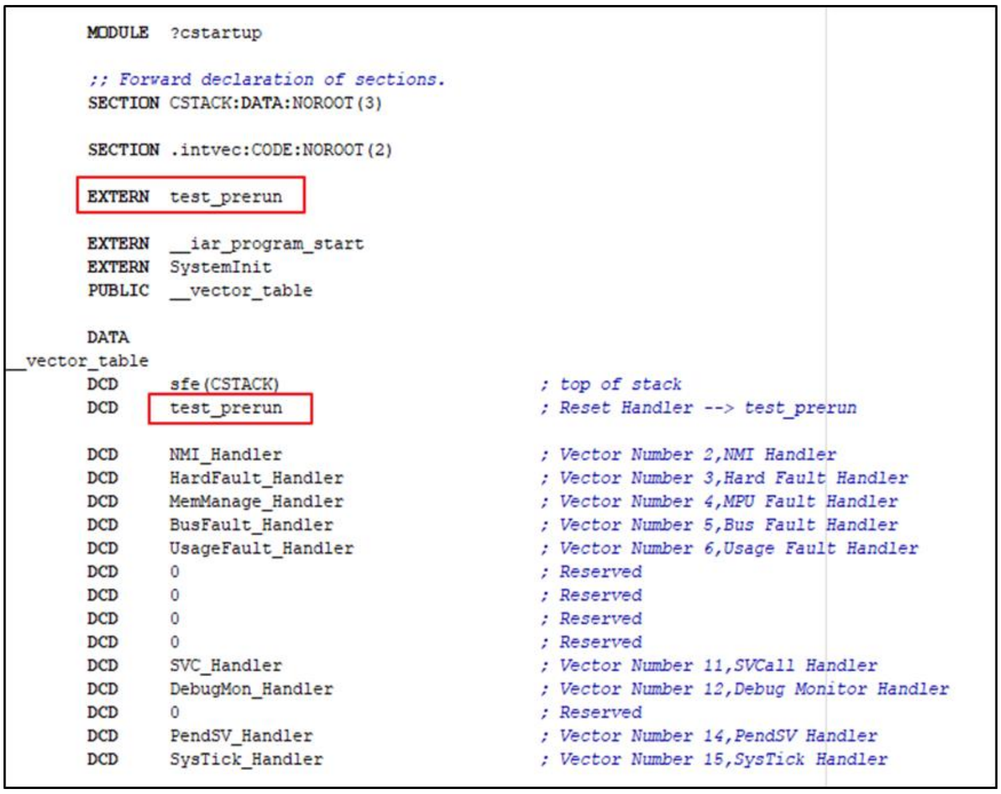

Files
Project - Debug
Application
CMSIS
Doc
GD32F4xx_EVAL
GD32F4xx_Peripherals
IEC60730_ClassB
Starup
startup_gd32f450_470.s
Output

Figure 2-3. Add the ClassB library file to the IAR project


Files
Project - Debug
Application
CMSIS
Doc
GD32F4xx_EVAL
GD32F4xx_Peripherals
IEC60730_ClassB
gd32f4xx_test_clock.c
gd32f4xx_test_cpu_prerun_IAR.s
gd32f4xx_test_cpu_run_IAR.s
gd32f4xx_test_flash.c
gd32f4xx_test_fullram_run_IAR.s
gd32f4xx_test_fulltcmram_run_IAR.s
gd32f4xx_test_param.c
gd32f4xx_test_prerun.c
gd32f4xx_test_program_counter.c
gd32f4xx_test_ram.c
gd32f4xx_test_run.c
Starup
Output

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.

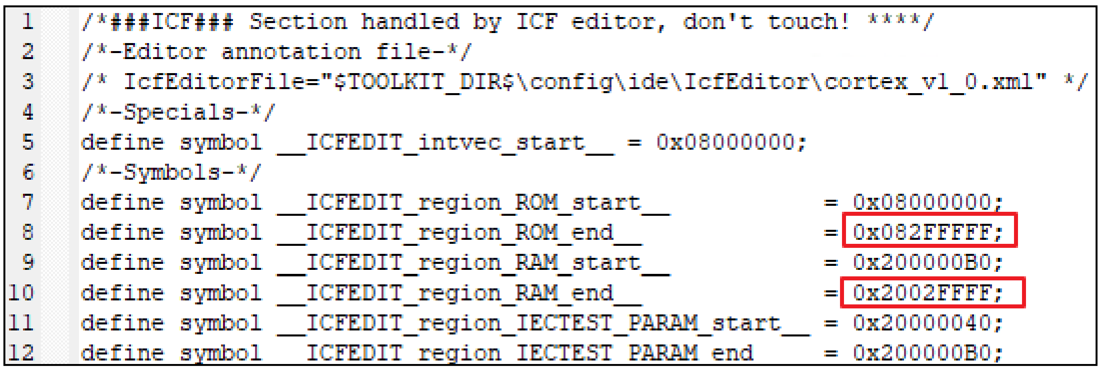
Figure 2-4. Modify the .s startup file



```
        MODULE  ?cstartup

        ;; Forward declaration of sections.
        SECTION CSTACK:DATA:NOROOT(3)

        SECTION .intvec:CODE:NOROOT(2)

        EXTERN  test_prerun

        EXTERN  __iar_program_start
        EXTERN  SystemInit
        PUBLIC  __vector_table

        DATA
__vector_table
        DCD     sfe(CSTACK)                     ; top of stack
        DCD     test_prerun                     ; Reset Handler --> test_prerun

        DCD     NMI_Handler                     ; Vector Number 2,NMI Handler
        DCD     HardFault_Handler               ; Vector Number 3,Hard Fault Handler
        DCD     MemManage_Handler               ; Vector Number 4,MPU Fault Handler
        DCD     BusFault_Handler                ; Vector Number 5,Bus Fault Handler
        DCD     UsageFault_Handler              ; Vector Number 6,Usage Fault Handler
        DCD     0                               ; Reserved
        DCD     0                               ; Reserved
        DCD     0                               ; Reserved
        DCD     0                               ; Reserved
        DCD     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\IEC_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.

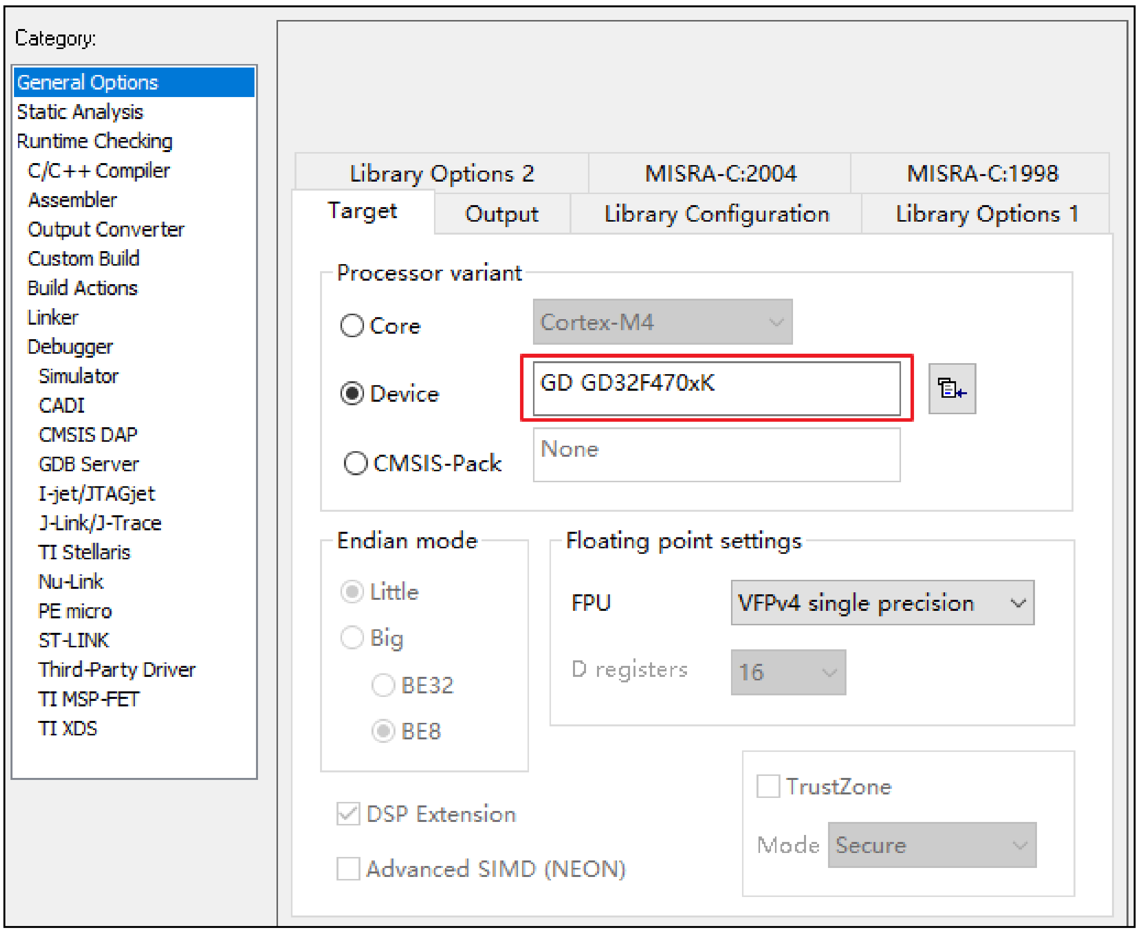
Figure 2-5. Modify the scattered loading file



```
/*###ICF### Section handled by ICF editor, don't touch! ****/
/*-Editor annotation file-*/
/* IcfEditorFile="$TOOLKIT_DIR$\config\ide\IcfEditor\cortex_v1_0.xml" */
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x08000000;
/*-Symbols-*/
define symbol __ICFEDIT_region_ROM_start__             = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__               = 0x082FFFFF;
define symbol __ICFEDIT_region_RAM_start__             = 0x200000B0;
define symbol __ICFEDIT_region_RAM_end__               = 0x2002FFFF;
define symbol __ICFEDIT_region_IECTEST_PARAM_start__   = 0x20000040;
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

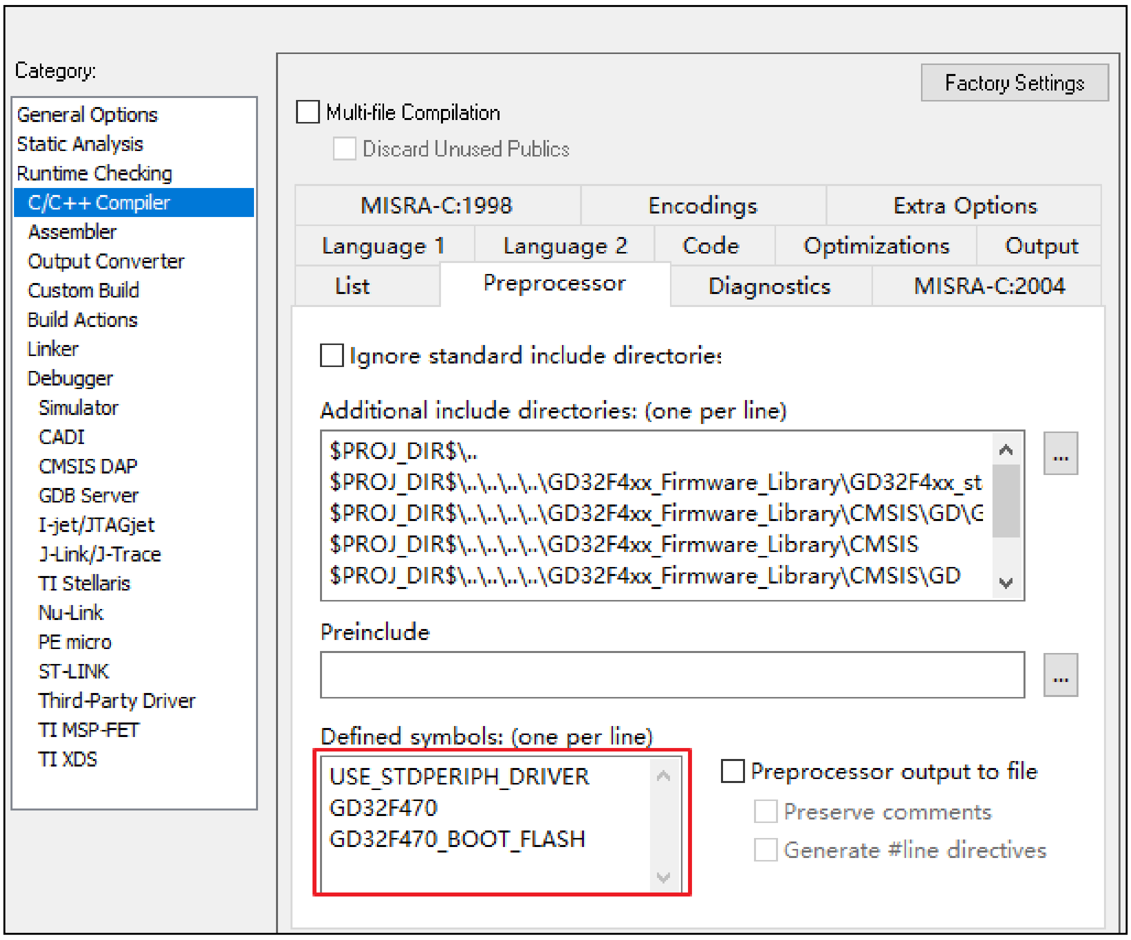


Processor variant: [✓ Device] GD GD32F470xK

FPU: VFPv4 single precision

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



Additional include directories: (one per line)

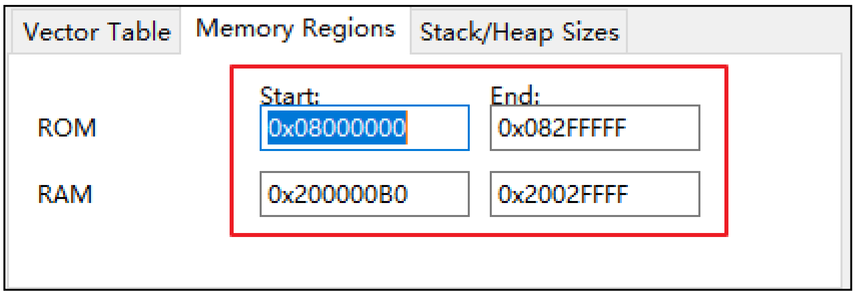
```
$PROJ_DIR$\..
$PROJ_DIR$\..\..\..\..\GD32F4xx_Firmware_Library\GD32F4xx_st
$PROJ_DIR$\..\..\..\..\GD32F4xx_Firmware_Library\CMSIS\GD\G
$PROJ_DIR$\..\..\..\..\GD32F4xx_Firmware_Library\CMSIS
$PROJ_DIR$\..\..\..\..\GD32F4xx_Firmware_Library\CMSIS\GD
```

Defined symbols: (one per line)

```
USE_STDPERIPH_DRIVER
GD32F470
GD32F470_BOOT_FLASH
```

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***.

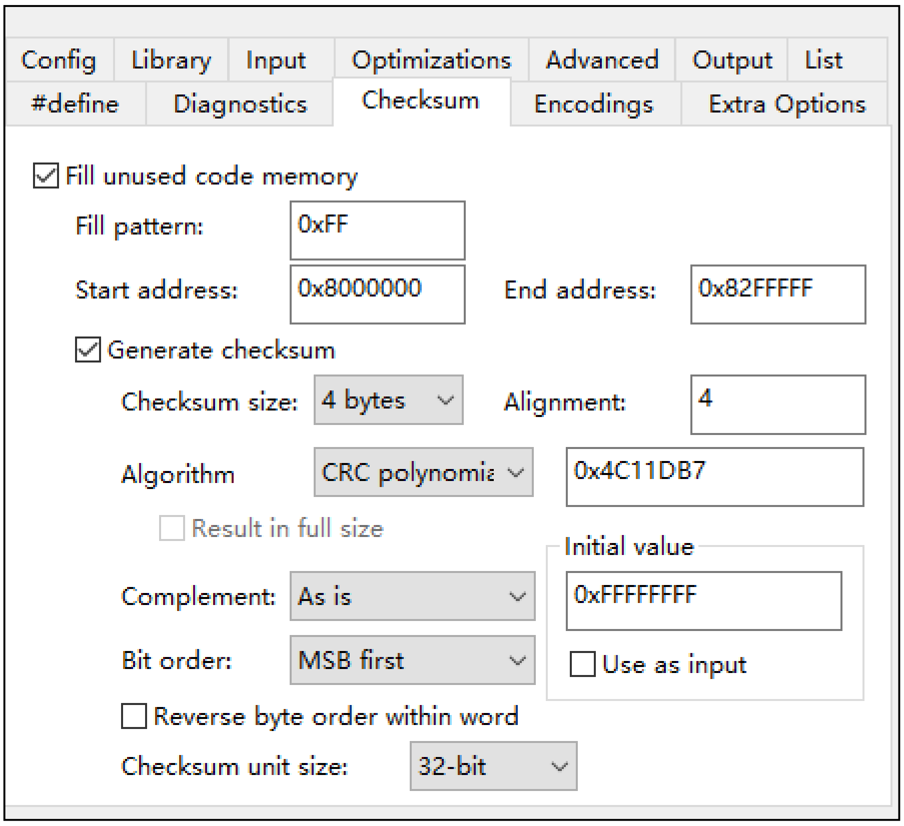
Figure 2-8. Modify the boundary values in the project settings



	Start:	End:
ROM	0x08000000	0x082FFFFF
RAM	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***.

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



☑ Fill unused code memory

Fill pattern: 0xFF

Start address: 0x8000000 End address: 0x82FFFFF

☑ Generate checksum

Checksum size: 4 bytes Alignment: 4

Algorithm CRC polynomiา 0x4C11DB7

☐ Result in full size

Initial value 0xFFFFFFFF

Complement: As is

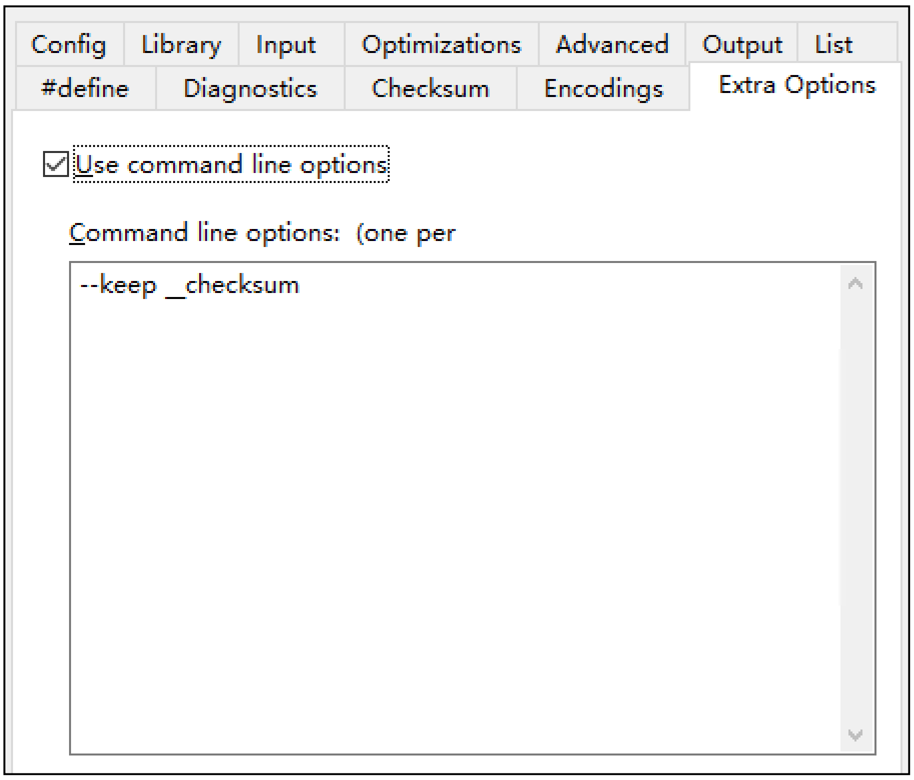
Bit order: MSB first

☐ Use as input

☐ Reverse byte order within word

Checksum unit size: 32-bit

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



☑ Use command line options

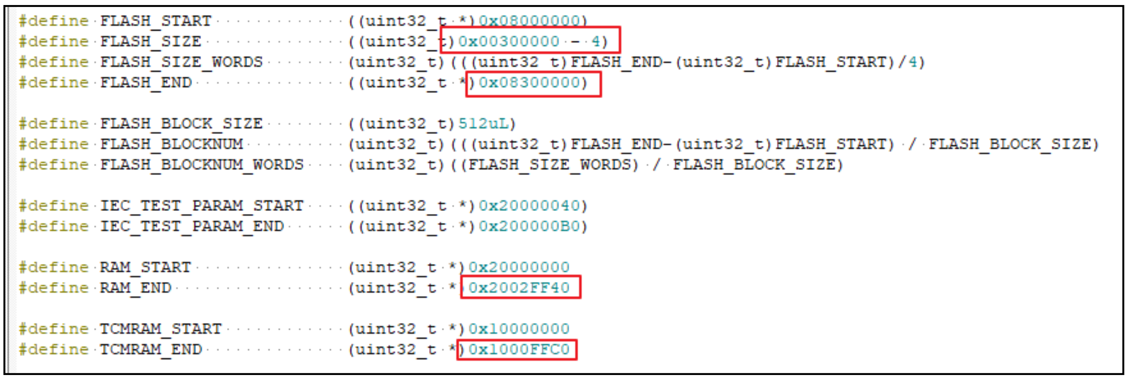
Command line options: (one per

```
--keep __checksum
```

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



```
#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)

#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 *)0x20000000
#define RAM_END                 (uint32_t *)0x2002FF40

#define TCMRAM_START            (uint32_t *)0x10000000
#define TCMRAM_END              (uint32_t *)0x1000FFC0
```

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

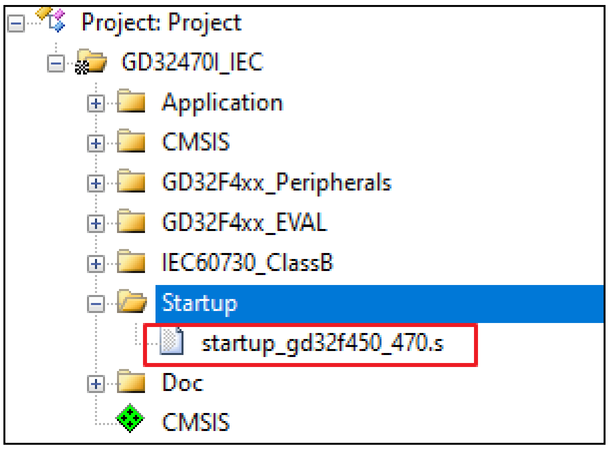
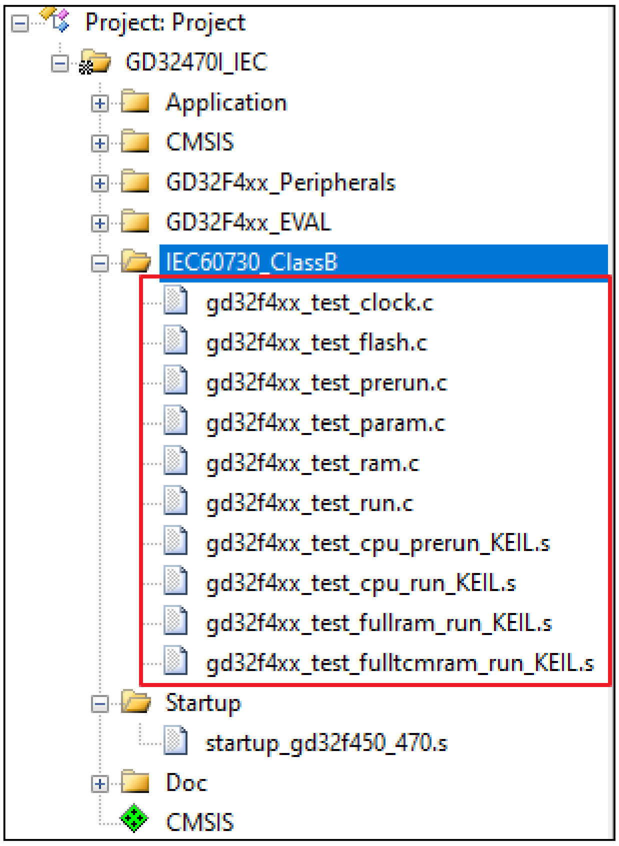


Figure 2-13.Add the ClassB library file to the Keil project



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.

```
Stack_Size      EQU     0x00000400

                AREA    STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem       SPACE   Stack_Size
__initial_sp


; <h> Heap Configuration
;   <o>  Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Heap_Size       EQU     0x00000400

                AREA    HEAP, NOINIT, READWRITE, ALIGN=3
__heap_base
Heap_Mem        SPACE   Heap_Size
__heap_limit
```

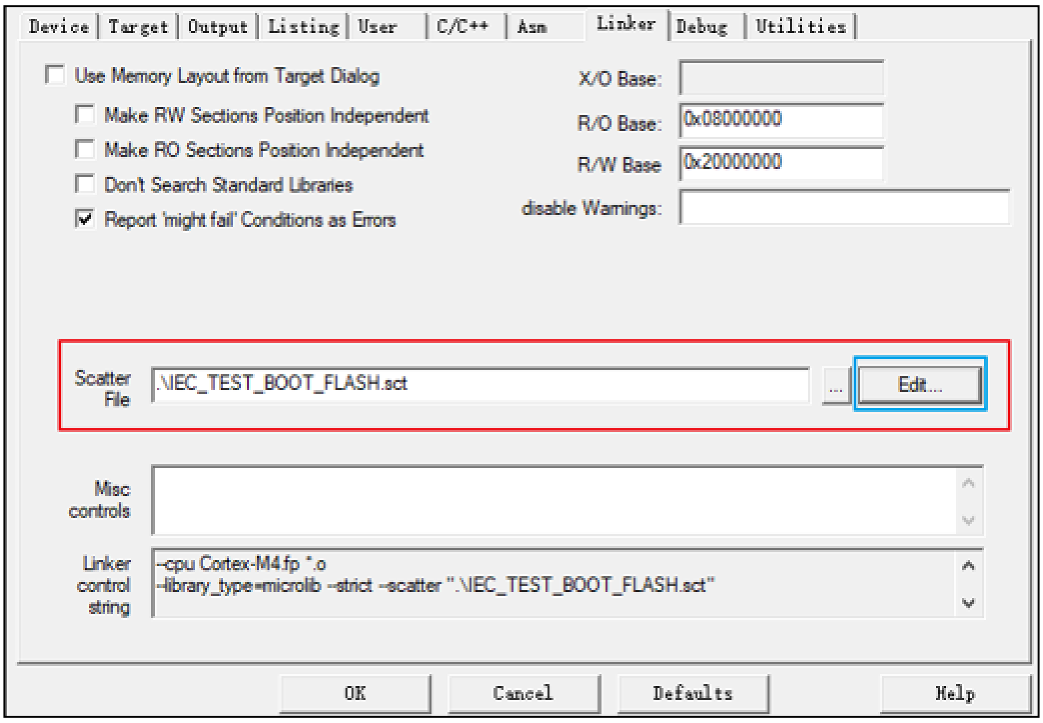
```
        IMPORT   test_prerun

                PRESERVE8
                THUMB

;               /* reset Vector Mapped to at Address 0 */
                AREA     RESET, DATA, READONLY
                EXPORT   __Vectors
                EXPORT   __Vectors_End
                EXPORT   __Vectors_Size

__Vectors       DCD      __initial_sp              ; Top of Stack
                DCD      test_prerun               ; Reset Handler --> test_prerun
                DCD      NMI_Handler               ; NMI Handler
                  ......
                  ......
                  ......
                DCD      USBFS_IRQHandler          ; 83:USBFS

__Vectors_End

                AREA CHECKSUM, DATA, READONLY, ALIGN=2
                EXPORT __Check_Sum
                ALIGN
__Check_Sum           DCD 0xEEF15A05

__Vectors_Size  EQU      __Vectors_End - __Vectors

                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 0x002FFFF8 {
    ER_IROM1 0x08000000   0x002FFFF8 {
        *.o (RESET, +First)
        *(InRoot$$Sections)
        .ANY (+RO)
    }

                  ......
                  ......
                  ......

    ; RW data
    RW_IRAM1 0x200000B0   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



☐ Use Memory Layout from Target Dialog
☐ Make RW Sections Position Independent
☐ Make RO Sections Position Independent
☐ Don't Search Standard Libraries
☑ Report 'might fail' Conditions as Errors

X/O Base:
R/O Base: 0x08000000
R/W Base 0x20000000
disable Warnings:

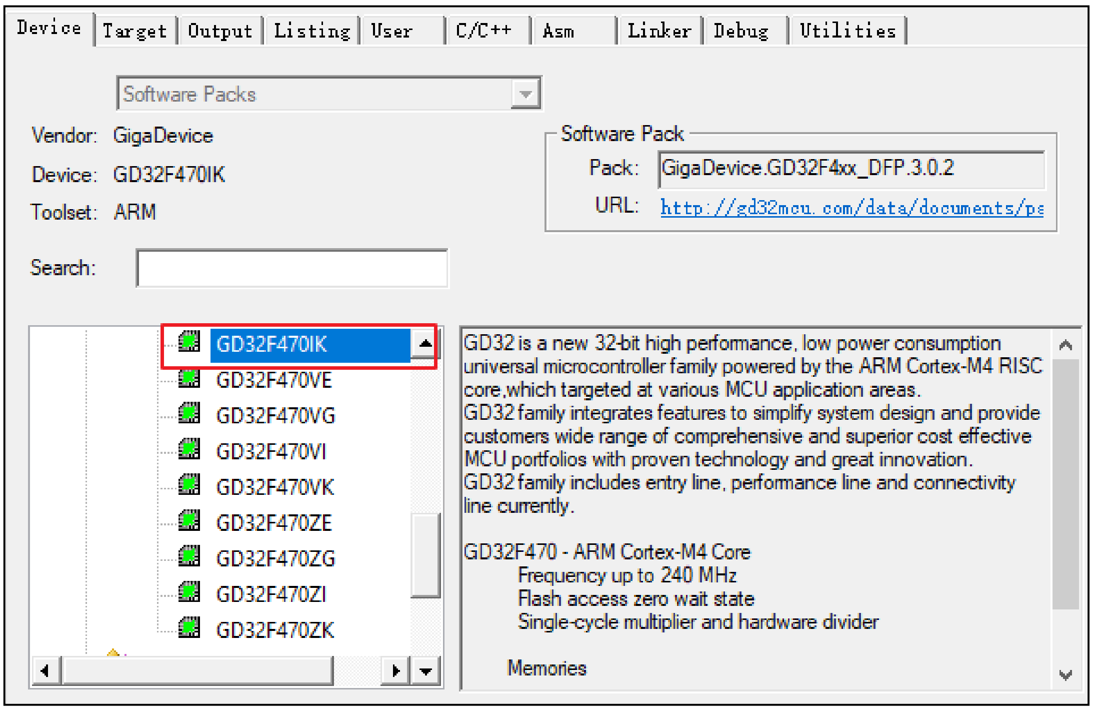
Scatter File: .\IEC_TEST_BOOT_FLASH.sct

Misc controls:

Linker control string:
```
-cpu Cortex-M4.fp *.o
--library_type=microlib --strict --scatter ".\IEC_TEST_BOOT_FLASH.sct"
```

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



Vendor: GigaDevice
Device: GD32F470IK
Toolset: ARM

Software Pack
Pack: GigaDevice.GD32F4xx_DFP.3.0.2
URL: http://gd32mcu.com/data/documents/pa

GD32F470IK
GD32F470VE
GD32F470VG
GD32F470VI
GD32F470VK
GD32F470ZE
GD32F470ZG
GD32F470ZI
GD32F470ZK

GD32 is a new 32-bit high performance, low power consumption universal microcontroller family powered by the ARM Cortex-M4 RISC core,which targeted at various MCU application areas.
GD32 family integrates features to simplify system design and provide customers wide range of comprehensive and superior cost effective MCU portfolios with proven technology and great innovation.
GD32 family includes entry line, performance line and connectivity line currently.

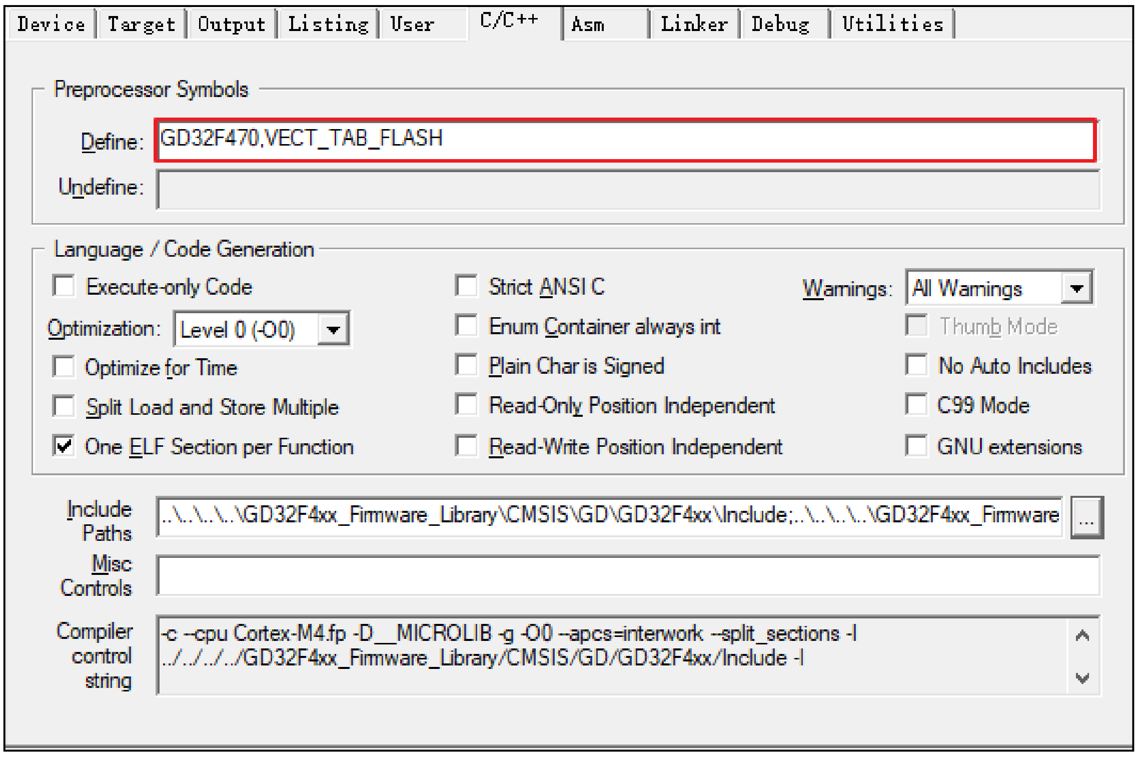
GD32F470 - ARM Cortex-M4 Core
Frequency up to 240 MHz
Flash access zero wait state
Single-cycle multiplier and hardware divider

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



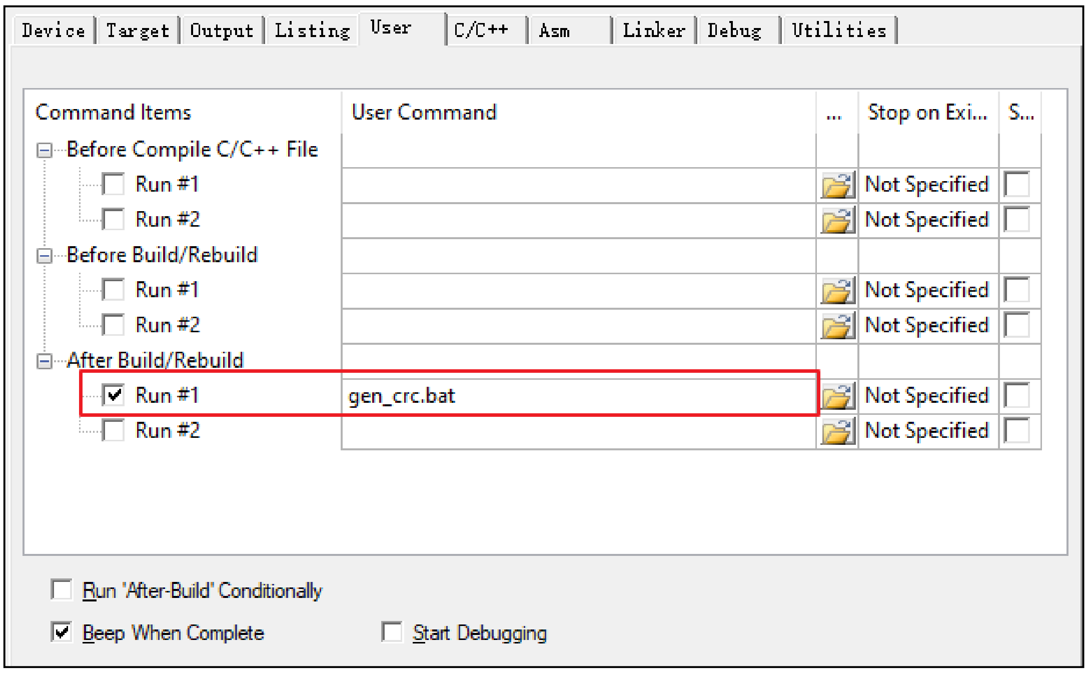
Preprocessor Symbols
Define: GD32F470,VECT_TAB_FLASH
Undefine:

Language / Code Generation
- [ ] Execute-only Code
- Optimization: Level 0 (-O0)
- [ ] Optimize for Time
- [ ] Split Load and Store Multiple
- [x] One ELF Section per Function
- [ ] Strict ANSI C
- [ ] Enum Container always int
- [ ] Plain Char is Signed
- [ ] Read-Only Position Independent
- [ ] Read-Write Position Independent
- Warnings: All Warnings
- [ ] Thumb Mode
- [ ] No Auto Includes
- [ ] C99 Mode
- [ ] GNU extensions

Include Paths: ..\..\..\..\GD32F4xx_Firmware_Library\CMSIS\GD\GD32F4xx\Include;..\..\..\..\GD32F4xx_Firmware
Misc Controls:
Compiler control string: -c --cpu Cortex-M4.fp -D__MICROLIB -g -O0 --apcs=interwork --split_sections -I ../../../../GD32F4xx_Firmware_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



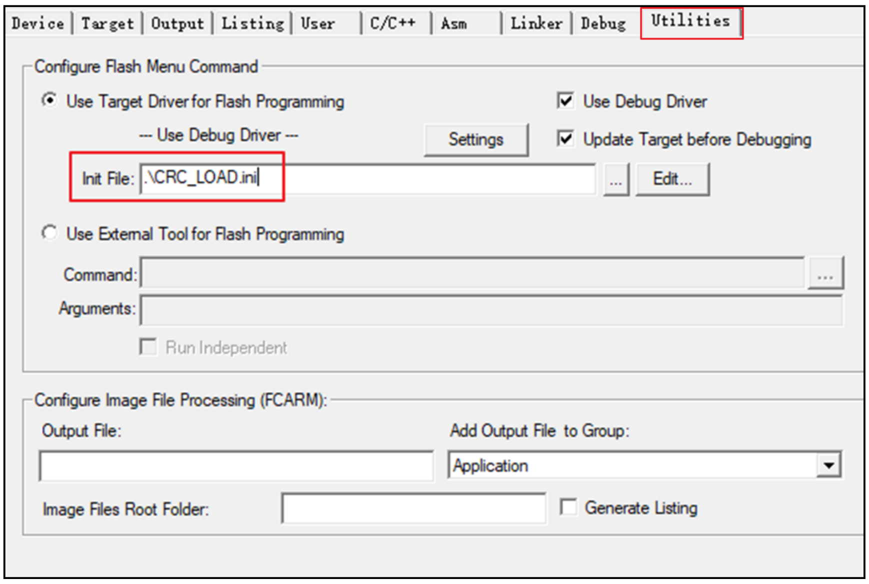
Command Items	User Command	...	Stop on Exi...	S...
Before Compile C/C++ File				
[ ] Run #1			Not Specified	[ ]
[ ] Run #2			Not Specified	[ ]
Before Build/Rebuild				
[ ] Run #1			Not Specified	[ ]
[ ] Run #2			Not Specified	[ ]
After Build/Rebuild				
[x] Run #1	gen_crc.bat		Not Specified	[ ]
[ ] Run #2			Not Specified	[ ]

- [ ] Run 'After-Build' Conditionally
- [x] 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



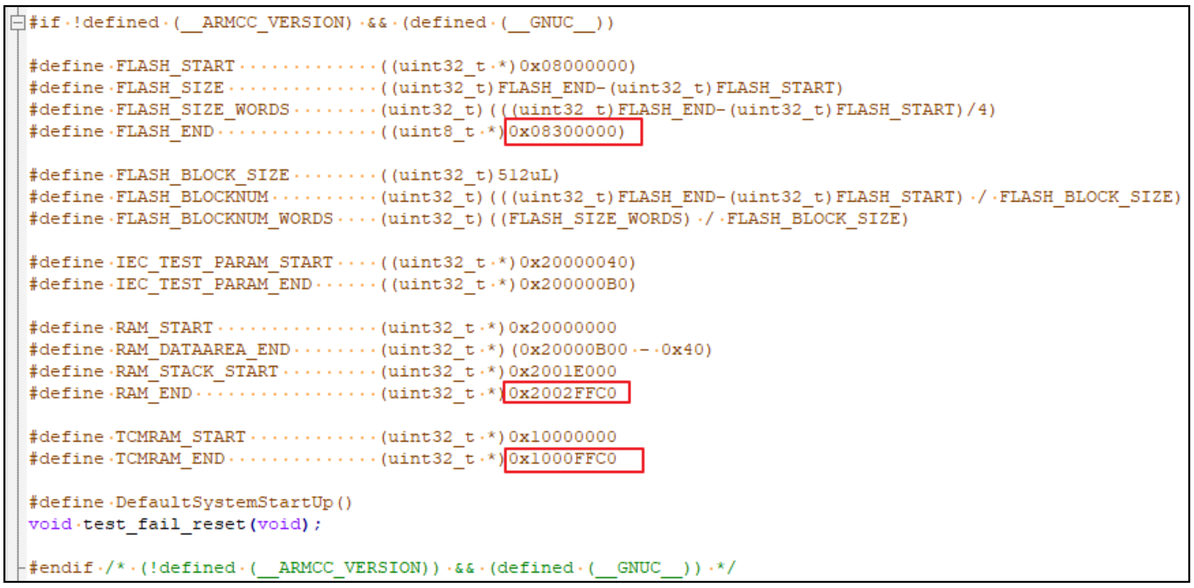
Configure Flash Menu Command
(●) Use Target Driver for Flash Programming
[✓] Use Debug Driver
--- Use Debug Driver ---
[✓] Update Target before Debugging
Init File: .\CRC_LOAD.ini
( ) Use External Tool for Flash Programming
Command:
Arguments:
[ ] Run Independent
Configure Image File Processing (FCARM):
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***.

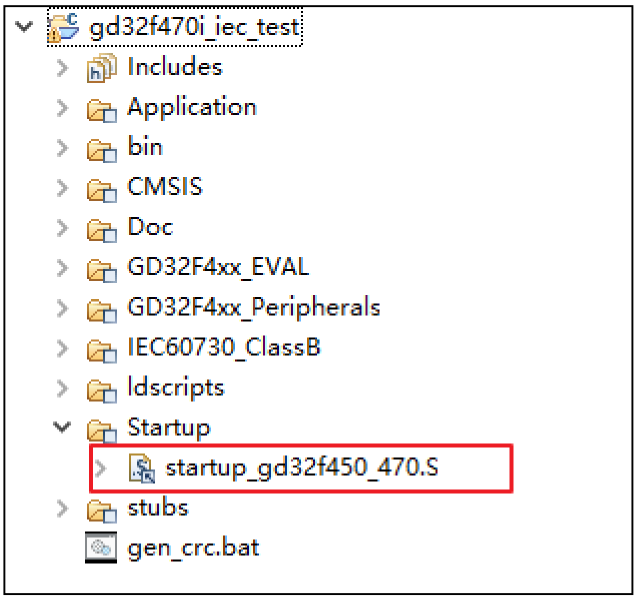
Figure 2-19. Modify the RAM boundary in the gd32f4xx_test.h file



```
#if !defined (__ARMCC_VERSION) && (defined (__GNUC__))

#define FLASH_START              ((uint32_t *)0x08000000)
#define FLASH_SIZE               ((uint32_t)FLASH_END-(uint32_t)FLASH_START)
#define FLASH_SIZE_WORDS         (uint32_t)(((uint32_t)FLASH_END-(uint32_t)FLASH_START)/4)
#define FLASH_END                ((uint8_t *)0x08300000)

#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 *)0x20000000
#define RAM_DATAAREA_END         (uint32_t *)(0x20000B00 - 0x40)
#define RAM_STACK_START          (uint32_t *)0x2001E000
#define RAM_END                  (uint32_t *)0x2002FFC0

#define TCMRAM_START             (uint32_t *)0x10000000
#define TCMRAM_END               (uint32_t *)0x1000FFC0

#define DefaultSystemStartUp()
void test_fail_reset(void);

#endif /* (!defined (__ARMCC_VERSION)) && (defined (__GNUC__)) */
```

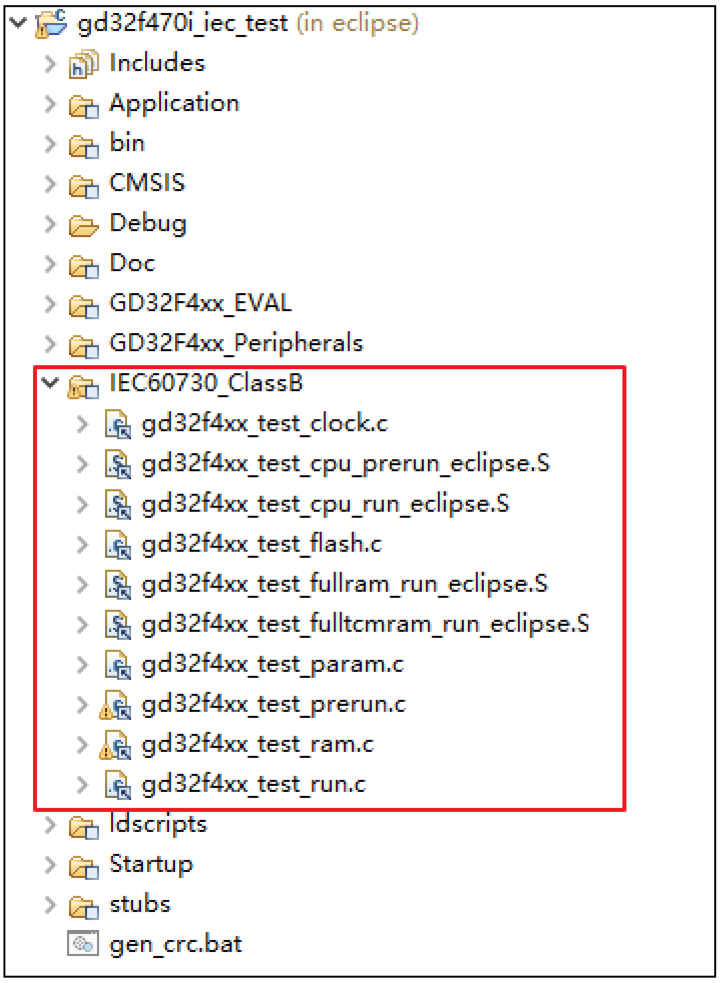
2. Check if the .S startup file in the project is suitable for the current chip, as shown in the ***Figure 2-20. Startup file check***, and ***Figure 2-21. Add the ClassB library file to the Eclipse project***.

Figure 2-20. Startup file check



```
gd32f470i_iec_test
  Includes
  Application
  bin
  CMSIS
  Doc
  GD32F4xx_EVAL
  GD32F4xx_Peripherals
  IEC60730_ClassB
  ldscripts
  Startup
    startup_gd32f450_470.S
  stubs
  gen_crc.bat
```

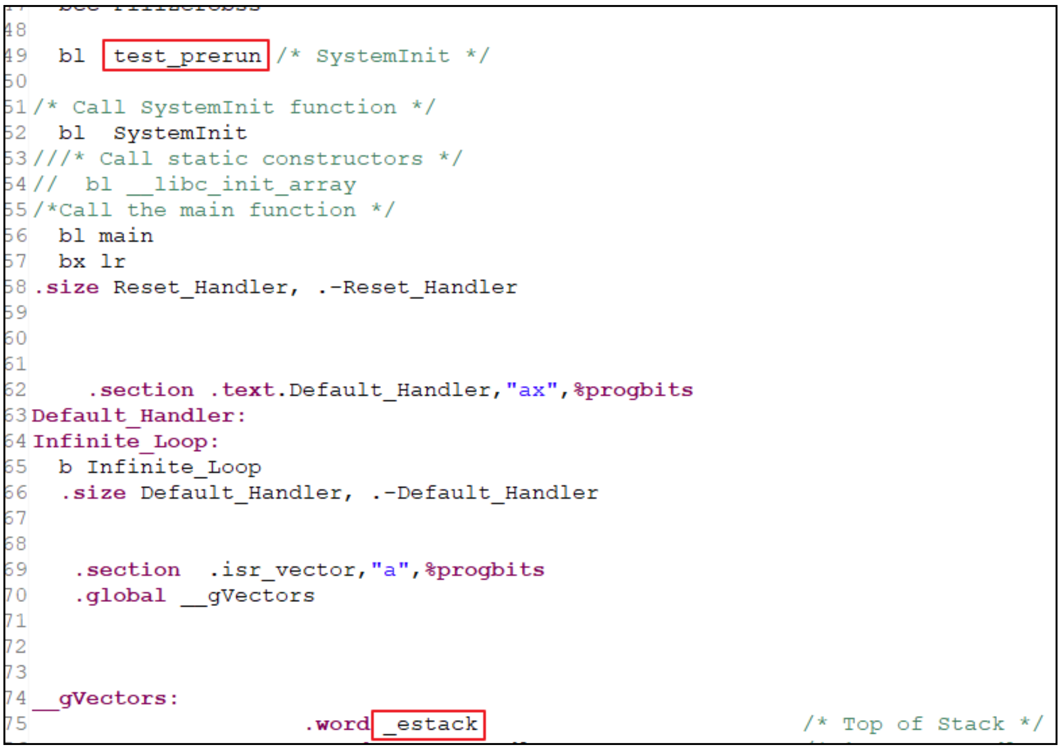
Figure 2-21. Add the ClassB library file to the Eclipse project



```
gd32f470i_iec_test (in eclipse)
  Includes
  Application
  bin
  CMSIS
  Debug
  Doc
  GD32F4xx_EVAL
  GD32F4xx_Peripherals
  IEC60730_ClassB
    gd32f4xx_test_clock.c
    gd32f4xx_test_cpu_prerun_eclipse.S
    gd32f4xx_test_cpu_run_eclipse.S
    gd32f4xx_test_flash.c
    gd32f4xx_test_fullram_run_eclipse.S
    gd32f4xx_test_fulltcmram_run_eclipse.S
    gd32f4xx_test_param.c
    gd32f4xx_test_prerun.c
    gd32f4xx_test_ram.c
    gd32f4xx_test_run.c
  ldscripts
  Startup
  stubs
  gen_crc.bat
```

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***.

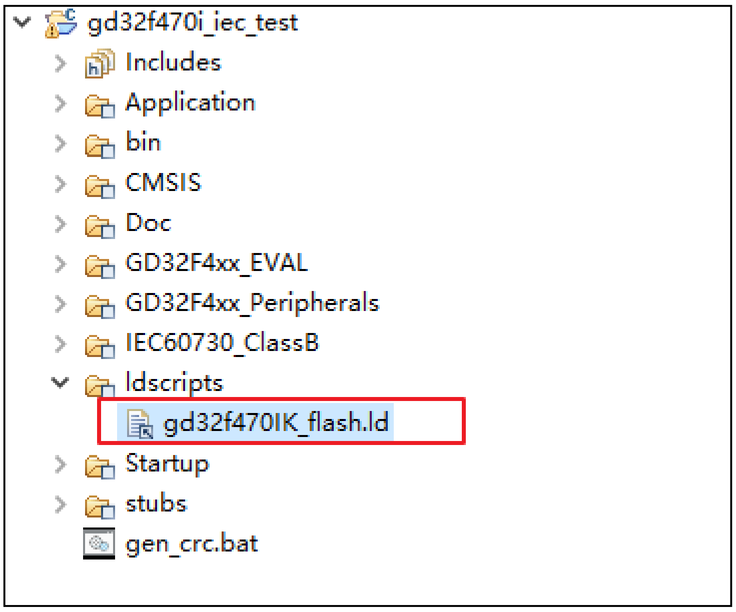
Figure 2-22. Modify the startup file



```
  bl  test_prerun /* SystemInit */

/* Call SystemInit function */
  bl  SystemInit
///* Call static constructors */
//  bl __libc_init_array
/*Call the main function */
  bl main
  bx lr
.size Reset_Handler, .-Reset_Handler



    .section .text.Default_Handler,"ax",%progbits
Default_Handler:
Infinite_Loop:
  b Infinite_Loop
  .size Default_Handler, .-Default_Handler


   .section  .isr_vector,"a",%progbits
   .global __gVectors



__gVectors:
                    .word _estack                     /* Top of Stack */
```

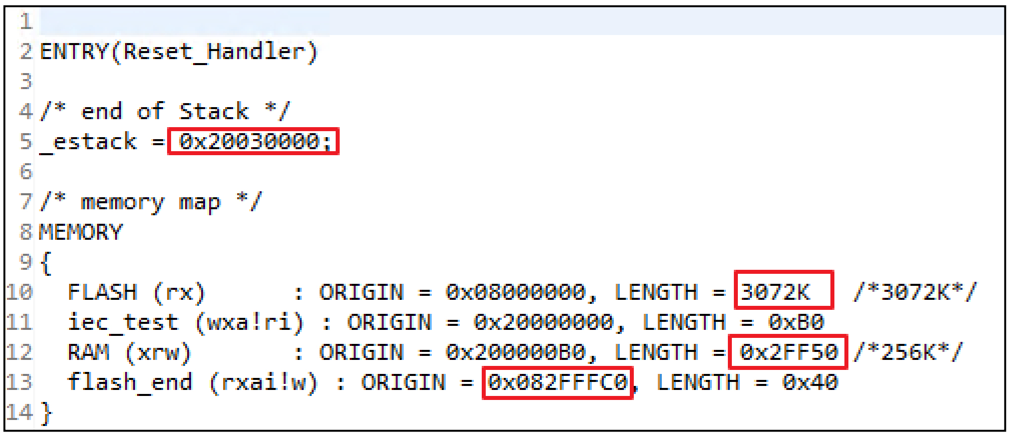
3. Modify the scatter-loading file in the scatter-loading file directory, as shown in ***Figure 2-24. Modify the Eclipse project to load files in a ld file*** is the modified part, and adjust the size of Flash and RAM according to the current datasheet.

Figure 2-23. Scatter loading file location



```
gd32f470i_iec_test
  Includes
  Application
  bin
  CMSIS
  Doc
  GD32F4xx_EVAL
  GD32F4xx_Peripherals
  IEC60730_ClassB
  ldscripts
    gd32f470IK_flash.ld
  Startup
  stubs
  gen_crc.bat
```

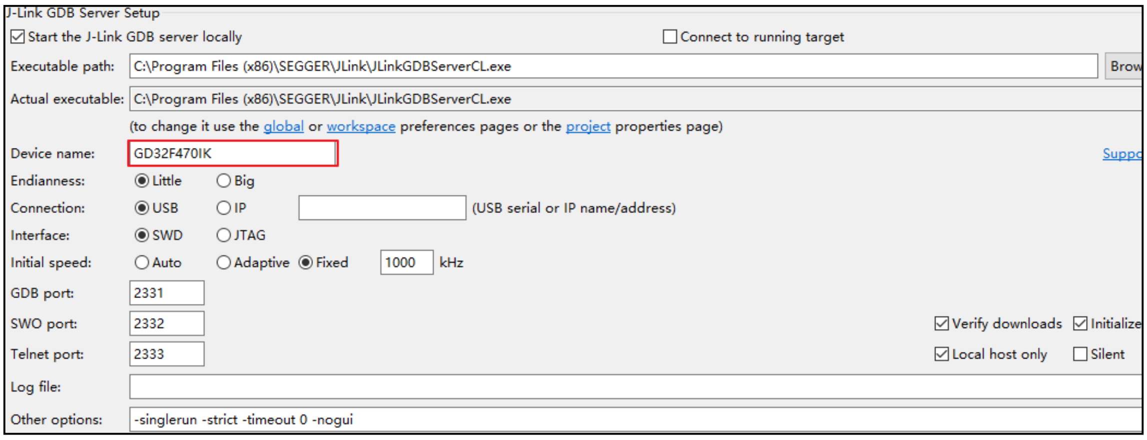
Figure 2-24. Modify the Eclipse project to load files in a ld file



```

ENTRY(Reset_Handler)

/* end of Stack */
_estack = 0x20030000;

/* memory map */
MEMORY
{
  FLASH (rx)       : ORIGIN = 0x08000000, LENGTH = 3072K   /*3072K*/
  iec_test (wxa!ri) : ORIGIN = 0x20000000, LENGTH = 0xB0
  RAM (xrw)        : ORIGIN = 0x200000B0, LENGTH = 0x2FF50 /*256K*/
  flash_end (rxai!w) : ORIGIN = 0x082FFFC0, LENGTH = 0x40
}
```

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

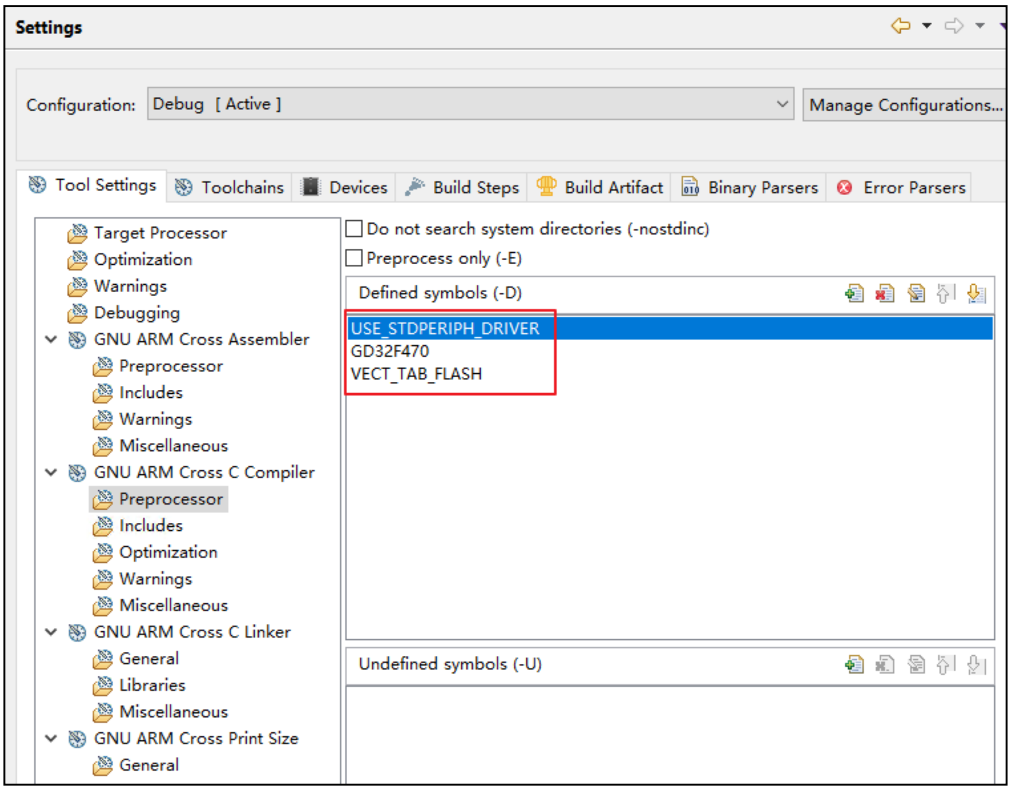
[✓] Start the J-Link GDB server locally ☐ Connect to running target

Executable path:	C:\Program Files (x86)\SEGGER\JLink\JLinkGDBServerCL.exe
Actual executable:	C:\Program Files (x86)\SEGGER\JLink\JLinkGDBServerCL.exe
	(to change it use the global or workspace preferences pages or the project properties page)
Device name:	GD32F470IK
Endianness:	[✓ Little] Big
Connection:	[✓ USB] IP (USB serial or IP name/address)
Interface:	[✓ SWD] JTAG
Initial speed:	Auto Adaptive [✓ Fixed] 1000 kHz
GDB port:	2331
SWO port:	2332
Telnet port:	2333
Log file:	
Other options:	-singlerun -strict -timeout 0 -nogui

[✓] Verify downloads [✓] Initialize
[✓] Local host only ☐ Silent

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


Settings

Configuration: Debug [Active]

Manage Configurations...

- Target Processor
- Optimization
- Warnings
- Debugging
- GNU ARM Cross Assembler
  - Preprocessor
  - Includes
  - Warnings
  - Miscellaneous
- GNU ARM Cross C Compiler
  - Preprocessor
  - Includes
  - Optimization
  - Warnings
  - Miscellaneous
- GNU ARM Cross C Linker
  - General
  - Libraries
  - Miscellaneous
- GNU ARM Cross Print Size
  - General

☐ Do not search system directories (-nostdinc)

☐ Preprocess only (-E)

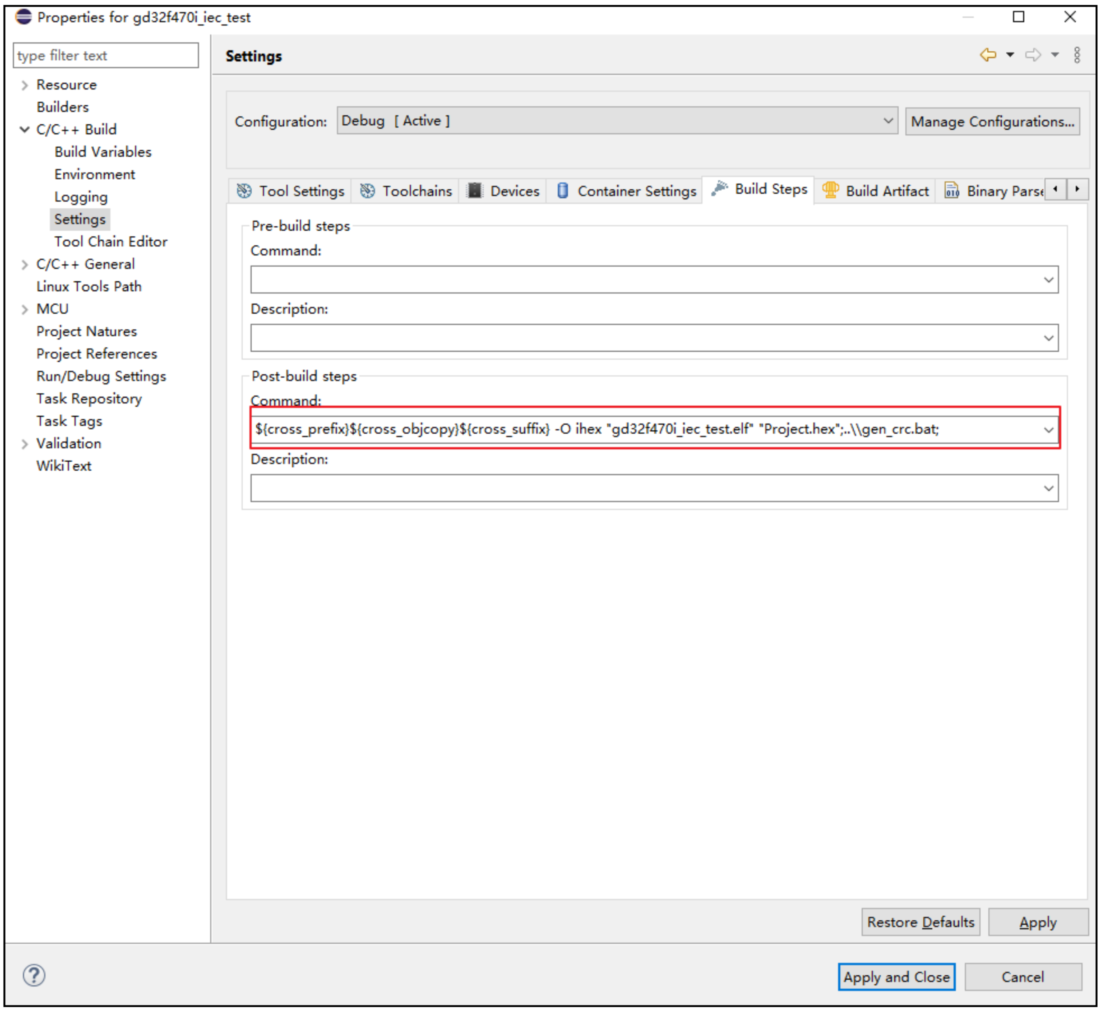
Defined symbols (-D)

```
USE_STDPERIPH_DRIVER
GD32F470
VECT_TAB_FLASH
```

Undefined symbols (-U)

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.

Figure 2-27.Modify the post-build command



Configuration: Debug [Active]

Pre-build steps
Command:
Description:

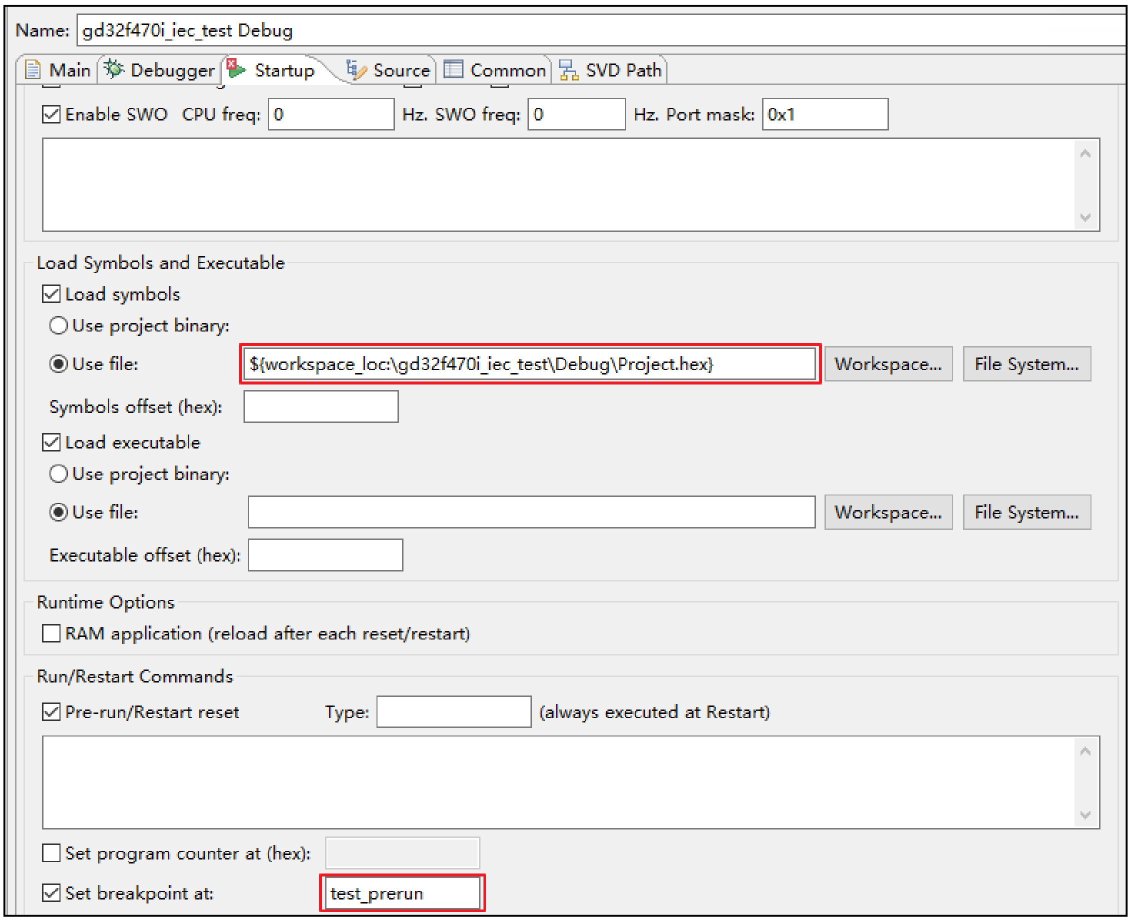
Post-build steps
Command:

```
${cross_prefix}${cross_objcopy}${cross_suffix} -O ihex "gd32f470i_iec_test.elf" "Project.hex";..\\gen_crc.bat;
```

Description:

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 configuration of the executable file


Name: gd32f470i_iec_test Debug

☑ Enable SWO CPU freq: 0 Hz. SWO freq: 0 Hz. Port mask: 0x1

Load Symbols and Executable

☑ Load symbols

○ Use project binary:

● Use file: ${workspace_loc:\gd32f470i_iec_test\Debug\Project.hex}

Symbols offset (hex):

☑ Load executable

○ Use project binary:

● Use file:

Executable offset (hex):

Runtime Options

☐ RAM application (reload after each reset/restart)

Run/Restart Commands

☑ Pre-run/Restart reset Type: (always executed at Restart)

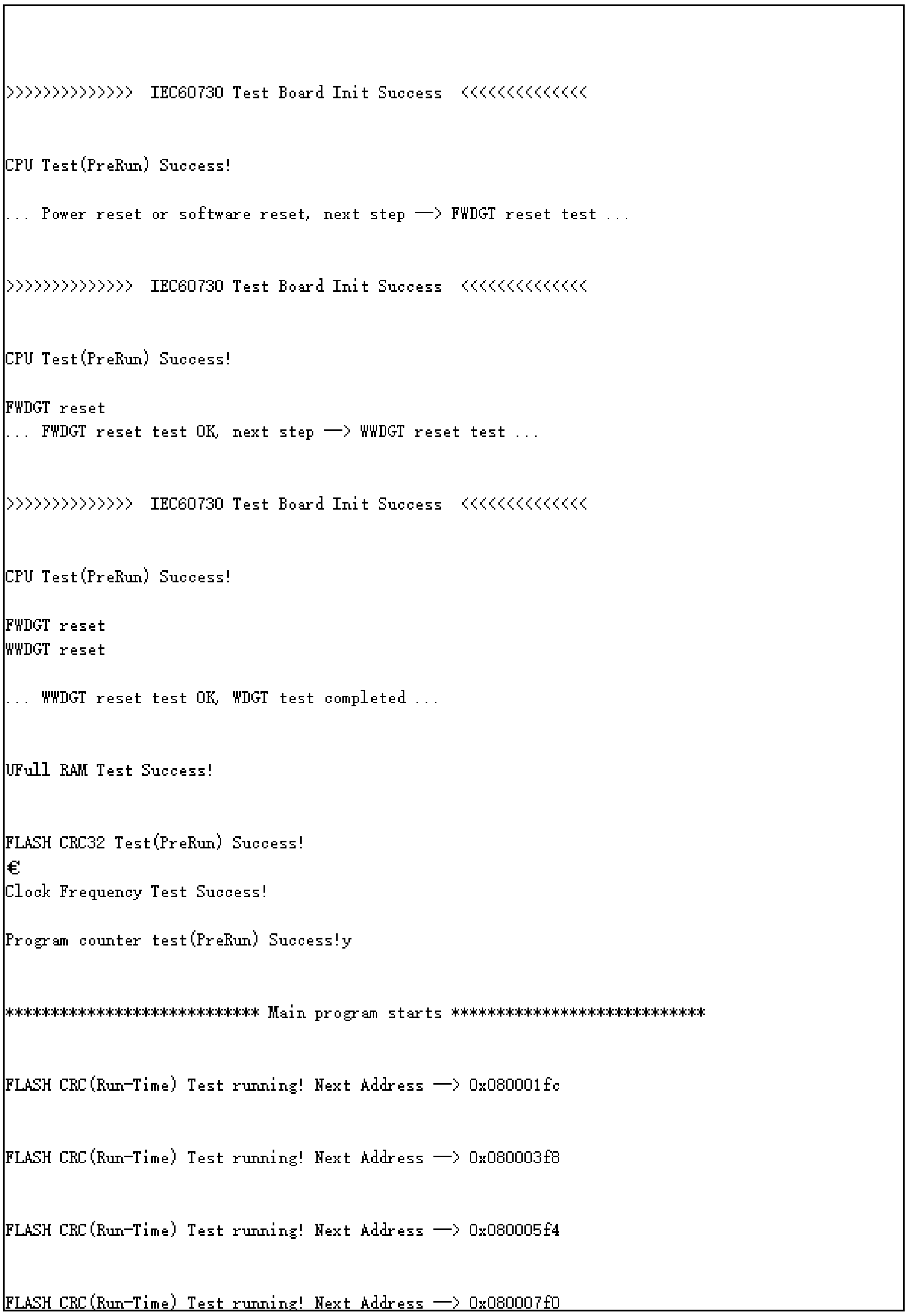
☐ Set program counter at (hex):

☑ Set breakpoint at: 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 3-1.Test Results***.

Figure 3-1.Test Results



```
>>>>>>>>>>>>>>  IEC60730 Test Board Init Success  <<<<<<<<<<<<<<


CPU Test(PreRun) Success!

... Power reset or software reset, next step --> FWDGT reset test ...


>>>>>>>>>>>>>>  IEC60730 Test Board Init Success  <<<<<<<<<<<<<<


CPU Test(PreRun) Success!

FWDGT reset
... FWDGT reset test OK, next step --> WWDGT reset test ...


>>>>>>>>>>>>>>  IEC60730 Test Board Init Success  <<<<<<<<<<<<<<


CPU Test(PreRun) Success!

FWDGT reset
WWDGT reset

... WWDGT reset test OK, WDGT test completed ...


UFull 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