GigaDevice Semiconductor Inc.

Guideline for migrating the IEC60730 ClassB library onto GD32F30x series

Application Note

AN136

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 GD32F30x 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

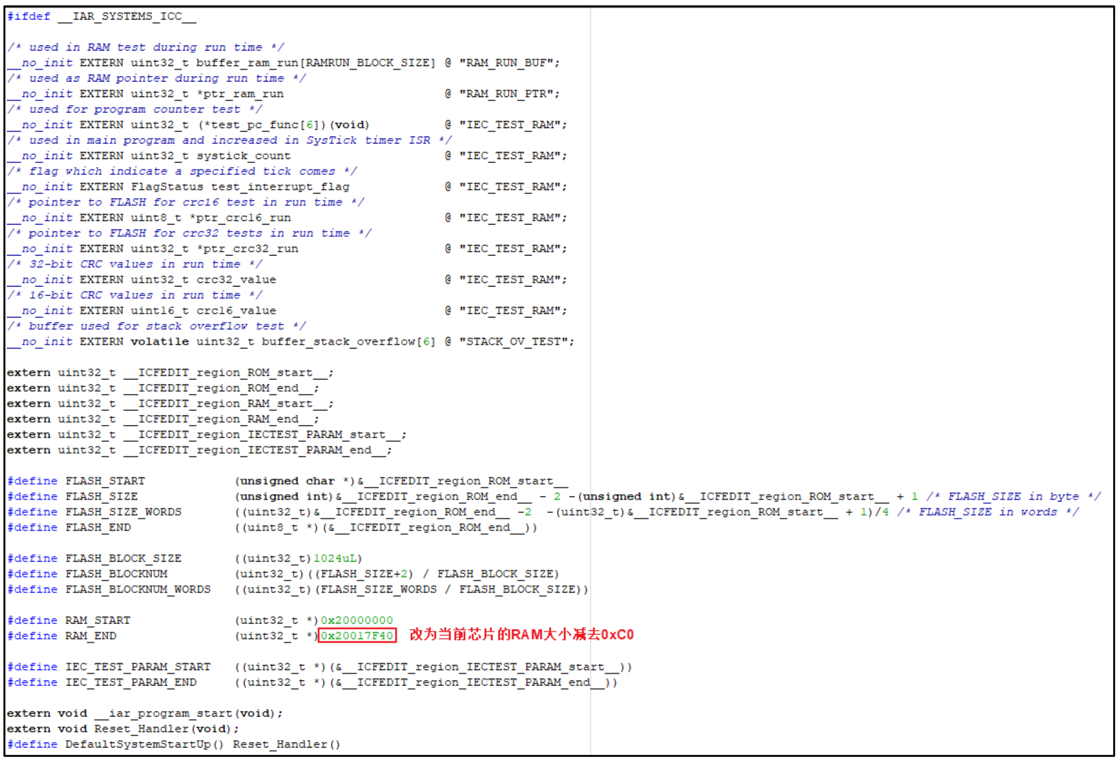
The GD32 MCU offers IEC60730 Class B certification library support for development environments including IAR, Keil, and Eclipse. There are variations in the template project migration among these three environments.

This application note will elaborate on the differences and provide guidance on project migration specifically for the GD32F305RC in each of these development tools. And it is also applicable to other M3/M4 core chips.

2.1. Migration of certification library project in IAR environment

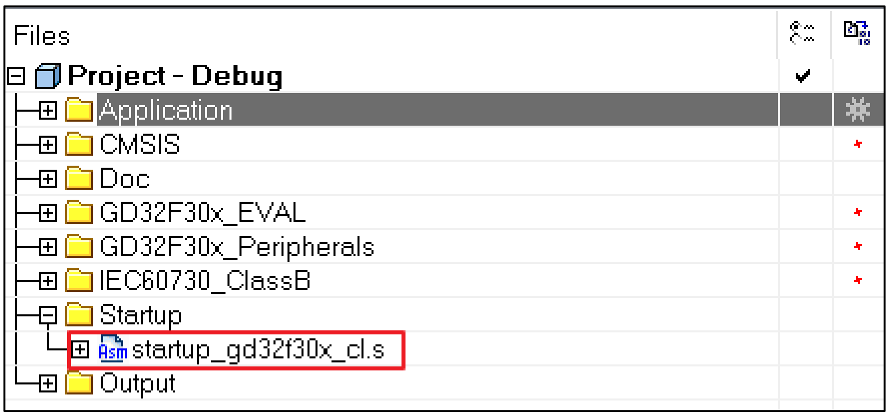
1. Modify the RAM boundary in the macro __IAR_SYSTEMS_ICC__ of the gd32f30x_test.h file according to the datasheet of the chip, as shown in ***Figure 2-1. Modify the RAM boundary in the gd32f30x_test.h***. Modify the RAM boundary in the gd32f30x_test.h as indicated.

Figure 2-1. Modify the RAM boundary in the gd32f30x_test.h


```
#ifdef __IAR_SYSTEMS_ICC__

/* used in RAM test during run time */
__no_init EXTERN uint32_t buffer_ram_run[RAMRUN_BLOCK_SIZE] @ "RAM_RUN_BUF";
/* used as RAM pointer during run time */
__no_init EXTERN uint32_t *ptr_ram_run                      @ "RAM_RUN_PTR";
/* used for program counter test */
__no_init EXTERN uint32_t (*test_pc_func[6])(void)          @ "IEC_TEST_RAM";
/* used in main program and increased in SysTick timer ISR */
__no_init EXTERN uint32_t systick_count                     @ "IEC_TEST_RAM";
/* flag which indicate a specified tick comes */
__no_init EXTERN FlagStatus test_interrupt_flag             @ "IEC_TEST_RAM";
/* pointer to FLASH for crc16 test in run time */
__no_init EXTERN uint8_t *ptr_crc16_run                     @ "IEC_TEST_RAM";
/* pointer to FLASH for crc32 tests in run time */
__no_init EXTERN uint32_t *ptr_crc32_run                    @ "IEC_TEST_RAM";
/* 32-bit CRC values in run time */
__no_init EXTERN uint32_t crc32_value                       @ "IEC_TEST_RAM";
/* 16-bit CRC values in run time */
__no_init EXTERN uint16_t crc16_value                       @ "IEC_TEST_RAM";
/* buffer used for stack overflow test */
__no_init EXTERN volatile uint32_t buffer_stack_overflow[6] @ "STACK_OV_TEST";

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__ - 2 -(unsigned int)&__ICFEDIT_region_ROM_start__ + 1 /* FLASH_SIZE in byte */
#define FLASH_SIZE_WORDS        ((uint32_t)&__ICFEDIT_region_ROM_end__ -2  -(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)1024uL)
#define FLASH_BLOCKNUM          (uint32_t)((FLASH_SIZE+2) / 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 *)0x20017F40  改为当前芯片的RAM大小减去0xC0

#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()
```

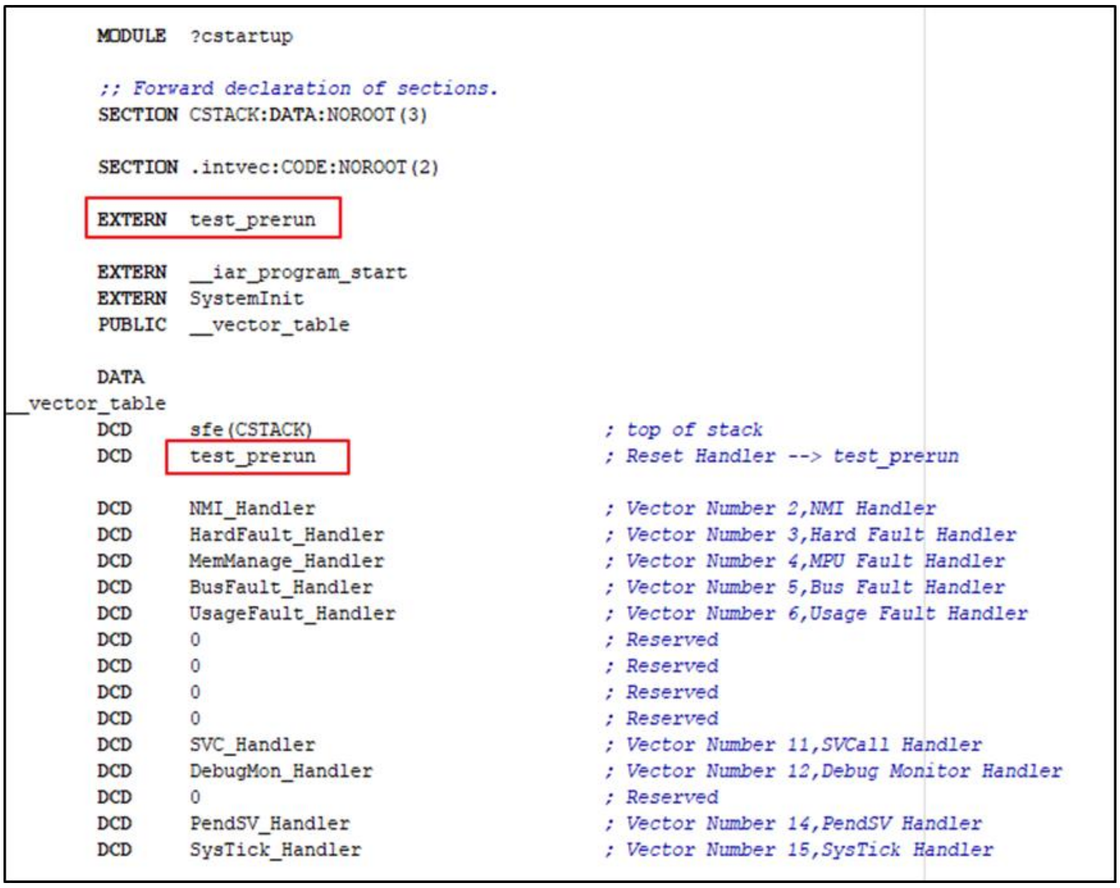
2. Check the .s startup file in the Startup folder of the project directory to see if it matches the current chip type, as shown in ***Figure 2-2. Startup file check***. At the same time, add the ClassB library file to the project.

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 GD32F30x_Firmware_Library\CMSIS\GD\GD32F30x\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

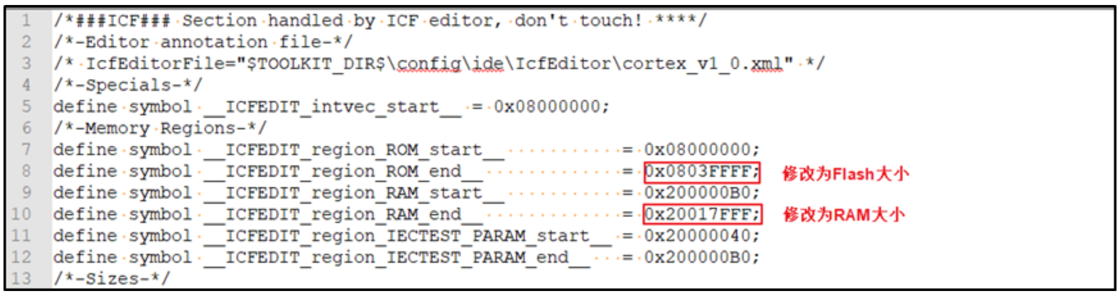


```
        MODULE  ?cstartup

        ;; 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, ..\GD32305_IEC_Test\Projects\IEC_Test\EWARM\IEC_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.

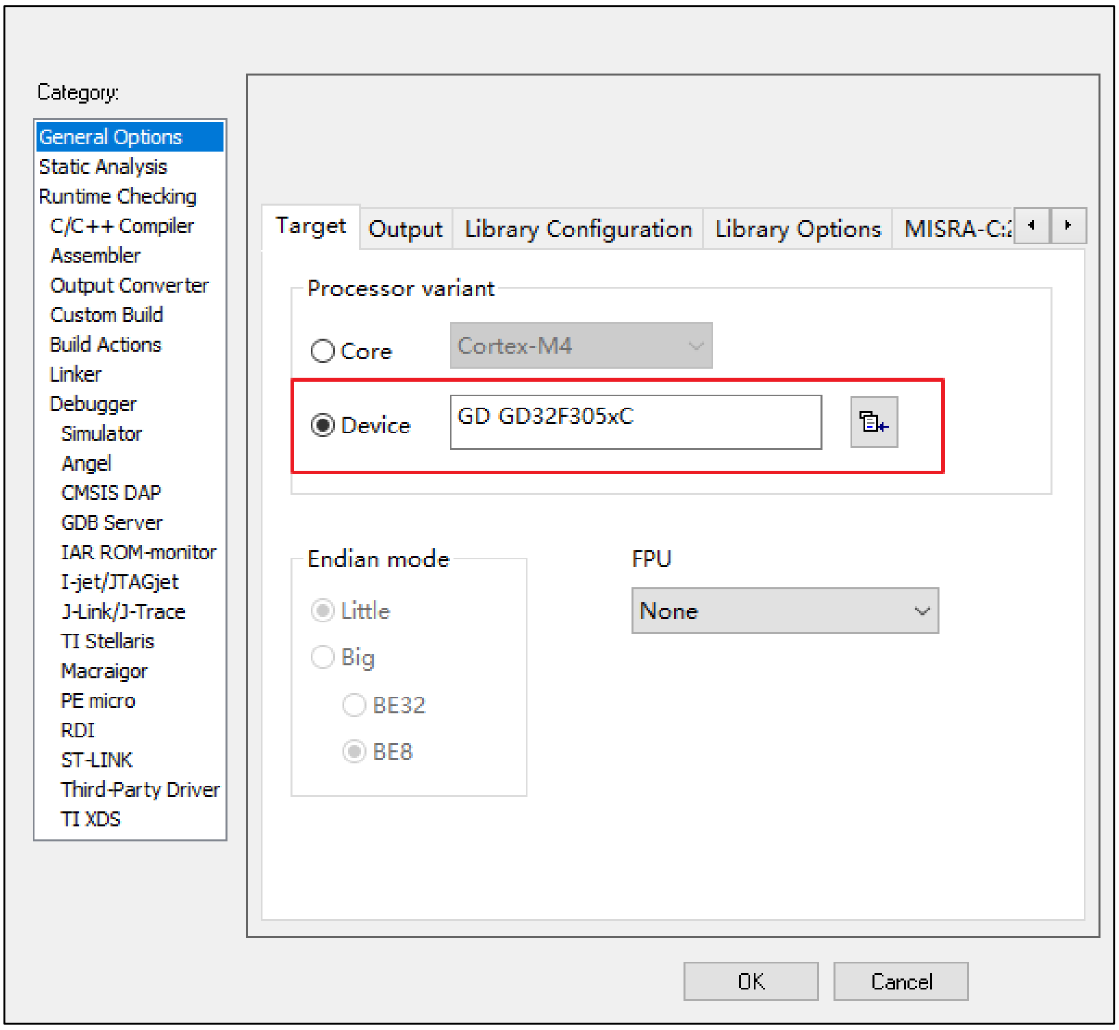
Figure 2-4. Modify the scattered loading file



```
/*###ICF### Section handled by ICF editor, don't touch! ****/
/*-Editor annotation file-*/
/* IcfEditorFile="$TOOLKIT_DIR$\config\ide\IcfEditor\cortex_v1_0.xml" */
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x08000000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__            = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__              = 0x0803FFFF;
define symbol __ICFEDIT_region_RAM_start__            = 0x200000B0;
define symbol __ICFEDIT_region_RAM_end__              = 0x20017FFF;
define symbol __ICFEDIT_region_IECTEST_PARAM_start__  = 0x20000040;
define symbol __ICFEDIT_region_IECTEST_PARAM_end__    = 0x200000B0;
/*-Sizes-*/
```

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***.

Figure 2-5. Device configuration



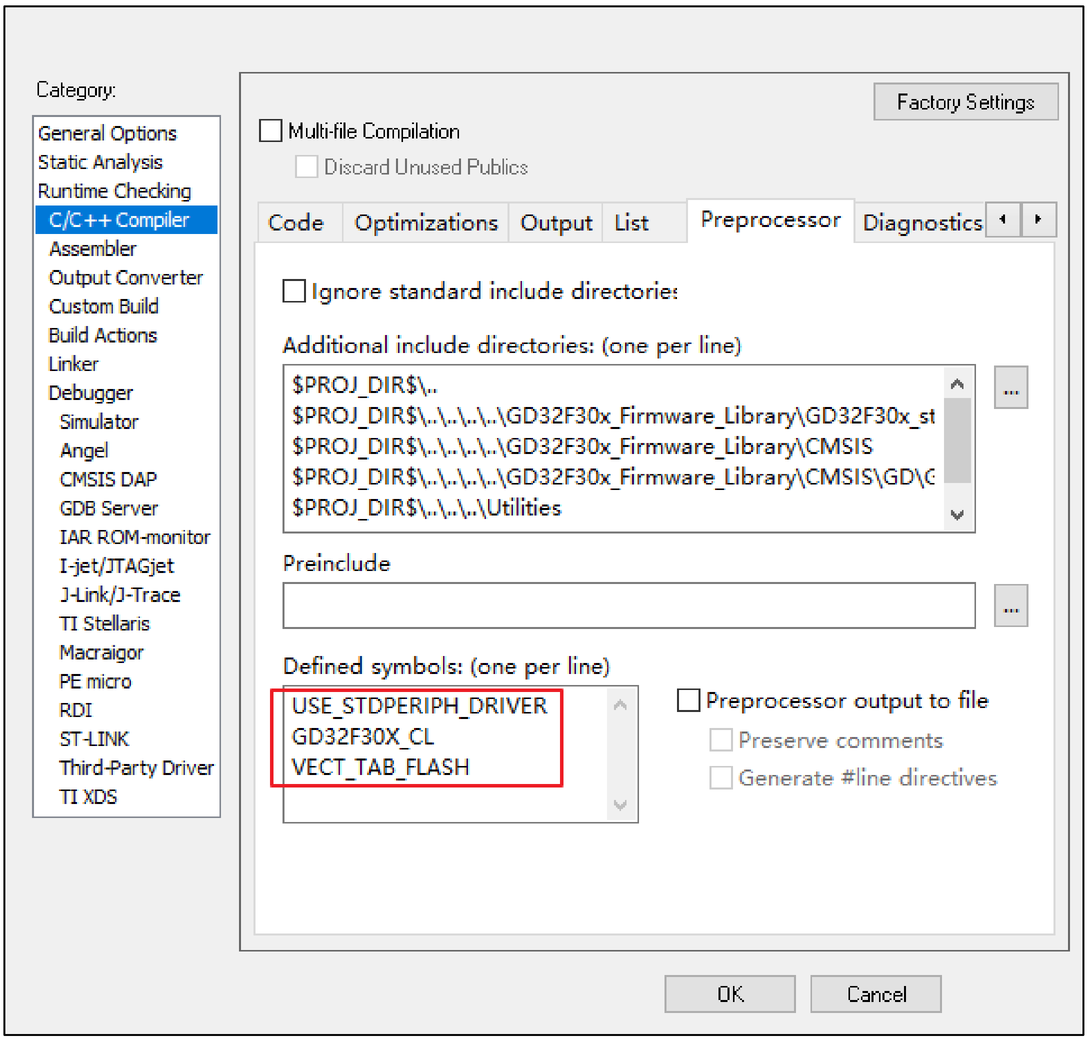
Processor variant
Core: Cortex-M4
[✓ Device]: GD GD32F305xC

Endian mode: [✓ Little] Big (BE32, BE8)

FPU: None

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

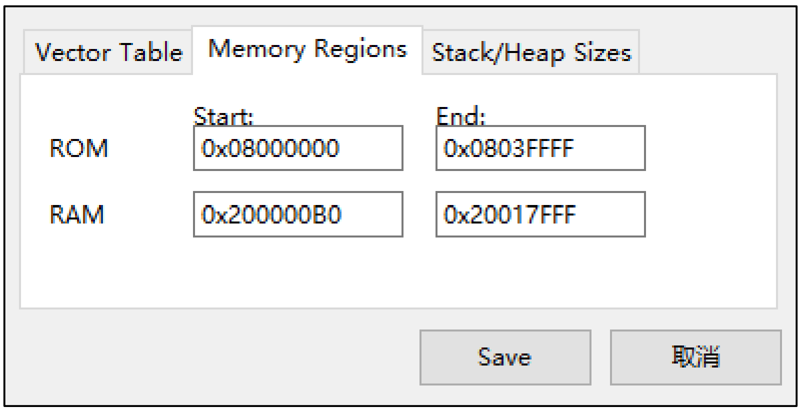


```
$PROJ_DIR$\..
$PROJ_DIR$\..\..\..\..\GD32F30x_Firmware_Library\GD32F30x_st
$PROJ_DIR$\..\..\..\..\GD32F30x_Firmware_Library\CMSIS
$PROJ_DIR$\..\..\..\..\GD32F30x_Firmware_Library\CMSIS\GD\G
$PROJ_DIR$\..\..\..\Utilities
```

```
USE_STDPERIPH_DRIVER
GD32F30X_CL
VECT_TAB_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 2-7. Modify the boundary values in the project settings***.

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

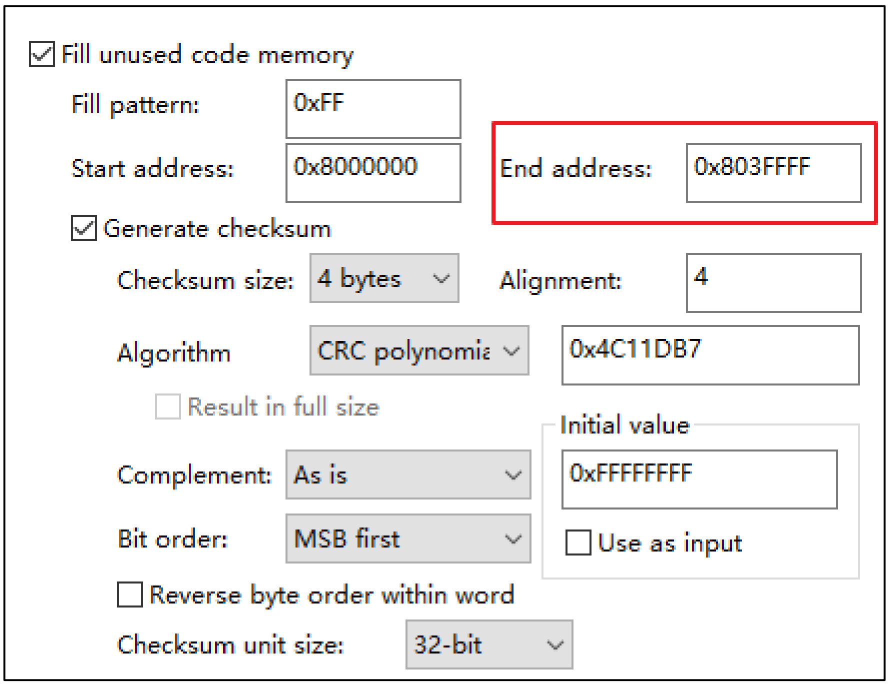


	Start:	End:
ROM	0x08000000	0x0803FFFF
RAM	0x200000B0	0x20017FFF

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.

Figure 2-8. Modify the checksum configuration in the project properties

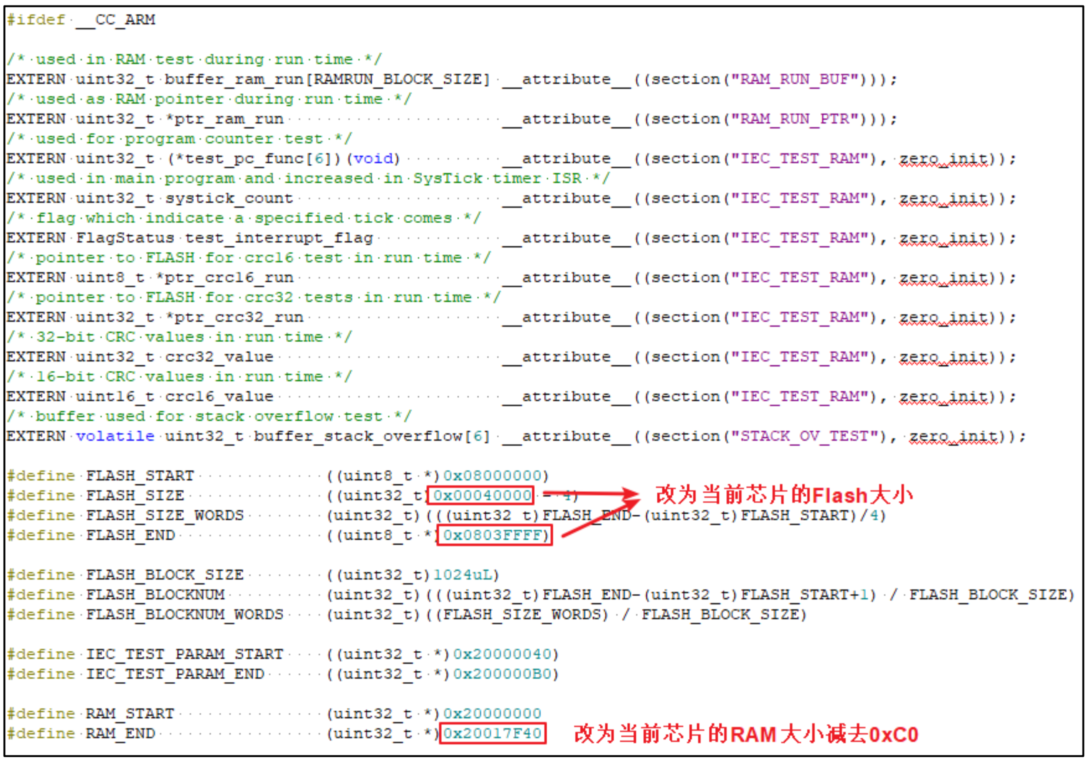


☑ Fill unused code memory			
Fill pattern:	0xFF		
Start address:	0x8000000	End address:	0x803FFFF
☑ Generate checksum			
Checksum size:	4 bytes	Alignment:	4
Algorithm	CRC polynomia		0x4C11DB7
☐ Result in full size		Initial value	0xFFFFFFFF
Complement:	As is		
Bit order:	MSB first	☐ Use as input	
☐ Reverse byte order within word			
Checksum unit size:	32-bit		

2.2. Migration of certification library project in Keil environment

1. Modify the RAM and Flash boundaries in the gd32f30x_test.h file according to the datasheet of the current chip model, as shown in ***Figure 2-9. Modify the RAM and Flash boundaries in the gd32f30x_test.h file***.

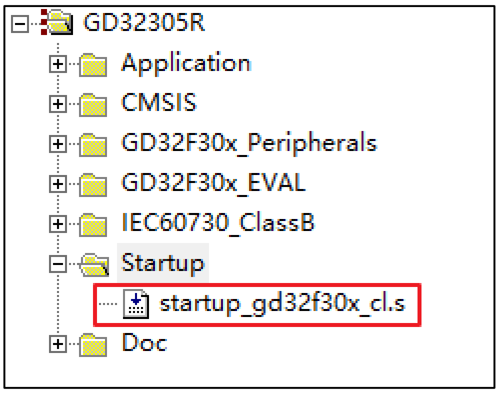
Figure 2-9. Modify the RAM and Flash boundaries in the gd32f30x_test.h file



```
#ifdef __CC_ARM

/* used in RAM test during run time */
EXTERN uint32_t buffer_ram_run[RAMRUN_BLOCK_SIZE]  __attribute__((section("RAM_RUN_BUF")));
/* used as RAM pointer during run time */
EXTERN uint32_t *ptr_ram_run                       __attribute__((section("RAM_RUN_PTR")));
/* used for program counter test */
EXTERN uint32_t (*test_pc_func[6])(void)           __attribute__((section("IEC_TEST_RAM"), zero_init));
/* used in main program and increased in SysTick timer ISR */
EXTERN uint32_t systick_count                      __attribute__((section("IEC_TEST_RAM"), zero_init));
/* flag which indicate a specified tick comes */
EXTERN FlagStatus test_interrupt_flag              __attribute__((section("IEC_TEST_RAM"), zero_init));
/* pointer to FLASH for crc16 test in run time */
EXTERN uint8_t *ptr_crc16_run                      __attribute__((section("IEC_TEST_RAM"), zero_init));
/* pointer to FLASH for crc32 tests in run time */
EXTERN uint32_t *ptr_crc32_run                     __attribute__((section("IEC_TEST_RAM"), zero_init));
/* 32-bit CRC values in run time */
EXTERN uint32_t crc32_value                        __attribute__((section("IEC_TEST_RAM"), zero_init));
/* 16-bit CRC values in run time */
EXTERN uint16_t crc16_value                        __attribute__((section("IEC_TEST_RAM"), zero_init));
/* buffer used for stack overflow test */
EXTERN volatile uint32_t buffer_stack_overflow[6]  __attribute__((section("STACK_OV_TEST"), zero_init));

#define FLASH_START              ((uint8_t *)0x08000000)
#define FLASH_SIZE               ((uint32_t)0x00040000 - 4)
#define FLASH_SIZE_WORDS         (uint32_t)(((uint32_t)FLASH_END-(uint32_t)FLASH_START)/4)
#define FLASH_END                ((uint8_t *)0x0803FFFF)

#define FLASH_BLOCK_SIZE         ((uint32_t)1024uL)
#define FLASH_BLOCKNUM           (uint32_t)(((uint32_t)FLASH_END-(uint32_t)FLASH_START+1) / 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 *)0x20017F40
```

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 2-10. Check the startup file***, and at the same time, add the ClassB library file to the project.

Figure 2-10. Check the startup file



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 GD32F30x_Firmware_Library\CMSIS\GD\GD32F30x\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
                  ......
                  ......
                  ......
```

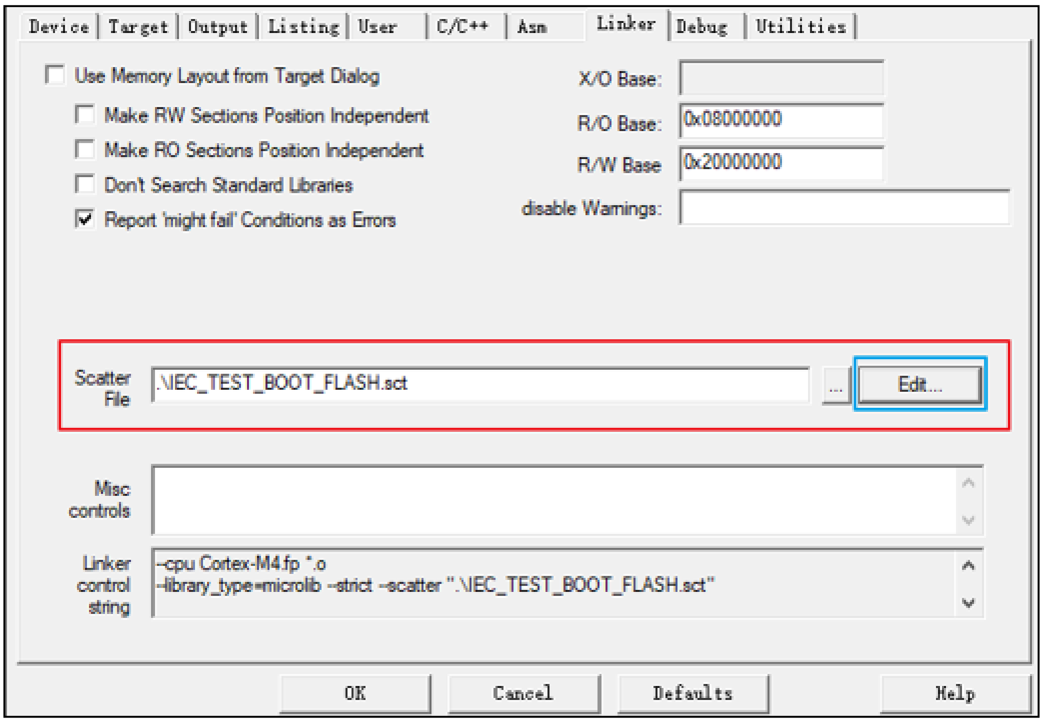
3. Modify the code in the scatter loading file IEC_TEST_BOOT_FLASH.sct, as shown in the following table, the parts marked in red need to be modified (modify the Flash boundary according to the current chip's datasheet; change the name of the .o file corresponding to the .s startup file).

```
    ; *************************************************************

    ; *************************************************************

    ; *** Scatter-Loading Description File generated by uVision ***

    ; *************************************************************


LR_IROM1 0x08000000 0x0003FFFF{
    ER_IROM1 0x08000000 0x0003FFFF {
        *.o (RESET, +First)
        *(InRoot$$Sections)
        .ANY (+RO)
    }

    ; RAM test during run time
    RAM_BUF 0x20000004
    {
        gd32f30x_test_prerun.o (RAM_RUN_BUF)
    }

    ; RAM pointer during run time
    RAM_PTR 0x20000030
    {
        gd32f30x_test_prerun.o (RAM_RUN_PTR)
    }

    ; variables of IEC test
    IEC_TEST_VAR 0x20000040 UNINIT 0x0000070
    {
        gd32f30x_test_prerun.o (IEC_TEST_RAM)
    }

    ; RW data
    RW_IRAM1 0x200000B0 UNINIT 0x00005000
    {
```

```
                .ANY (+RW +ZI)
        }

        ; stack overflow test
        STACK_IRAM2 0x200050B0 UNINIT 0x00006F40
        {
                gd32f30x_test_prerun.o (STACK_OV_TEST)
                startup_gd32f30x_cl.o (STACK, +Last)
        }
}

LR_IROM2 0x0803FFFC 0x0000004 {
        ER_IROM2 0x0803FFFC 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 2-11. Editing the scatter loading file***: Editing the scatter loading file.

Figure 2-11. 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:

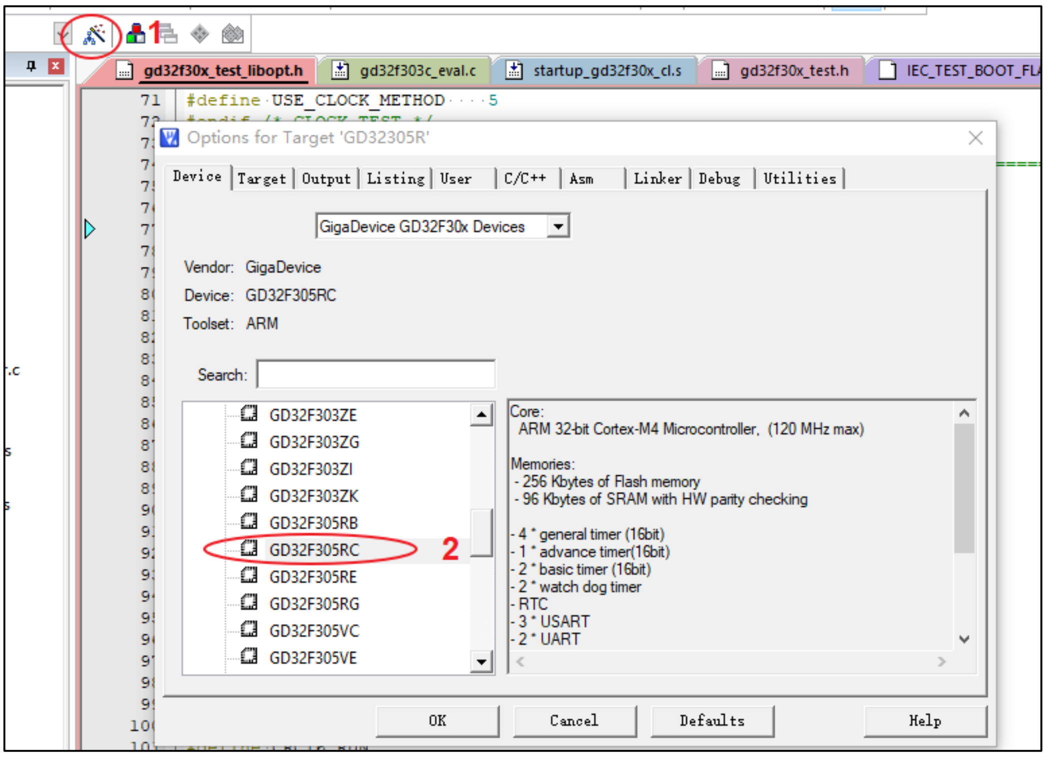
Scatter File: .\IEC_TEST_BOOT_FLASH.sct

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 2-12. Device Configuration***.

Figure 2-12. Device Configuration



```
#define USE_CLOCK_METHOD    5
```

Options for Target 'GD32305R'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

GigaDevice GD32F30x Devices

Vendor: GigaDevice
Device: GD32F305RC
Toolset: ARM

Search:

GD32F303ZE
GD32F303ZG
GD32F303ZI
GD32F303ZK
GD32F305RB
GD32F305RC
GD32F305RE
GD32F305RG
GD32F305VC
GD32F305VE

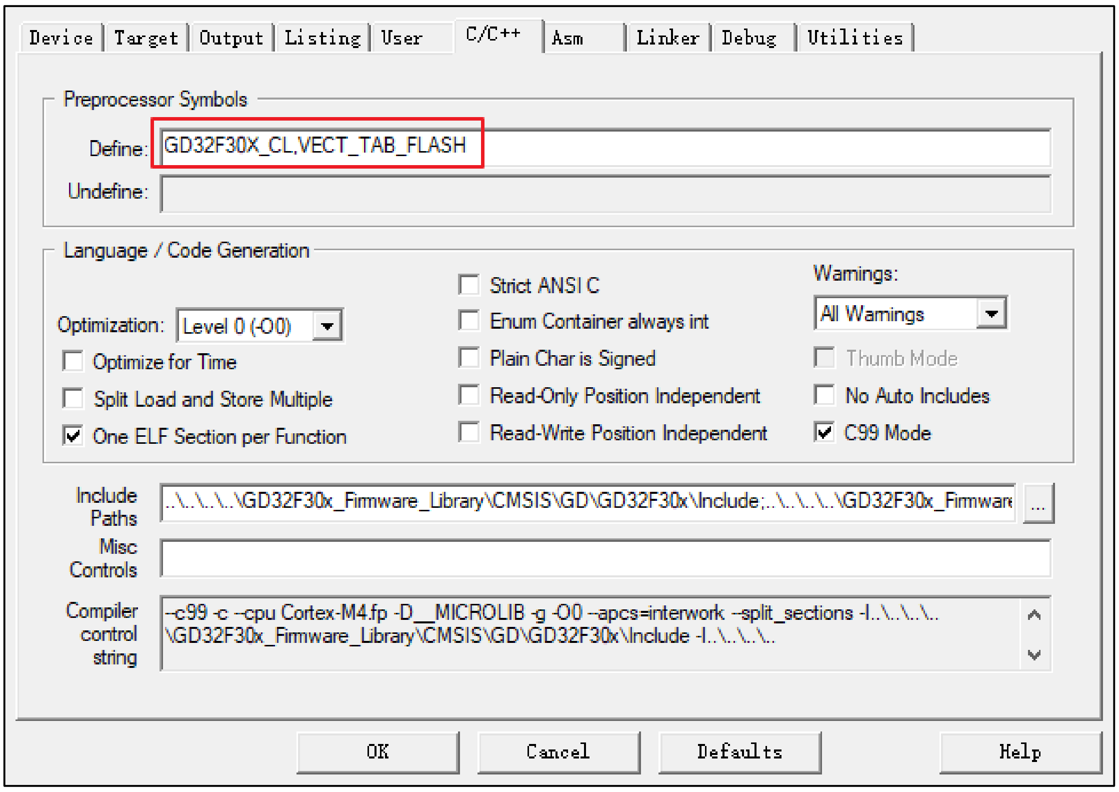
Core:
ARM 32-bit Cortex-M4 Microcontroller, (120 MHz max)

Memories:
- 256 Kbytes of Flash memory
- 96 Kbytes of SRAM with HW parity checking

- 4 * general timer (16bit)
- 1 * advance timer(16bit)
- 2 * basic timer (16bit)
- 2 * watch dog timer
- RTC
- 3 * USART
- 2 * UART

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 2-13. Modify the precompiled macro***:

Figure 2-13. Modify the precompiled macro


Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities
OK
Cancel
Defaults
Help

Preprocessor Symbols

Define: GD32F30X_CL,VECT_TAB_FLASH

Undefine:

Language / Code Generation

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
- [x] C99 Mode

Include Paths: ..\..\..\..\GD32F30x_Firmware_Library\CMSIS\GD\GD32F30x\Include;..\..\..\..\GD32F30x_Firmware

Misc Controls:

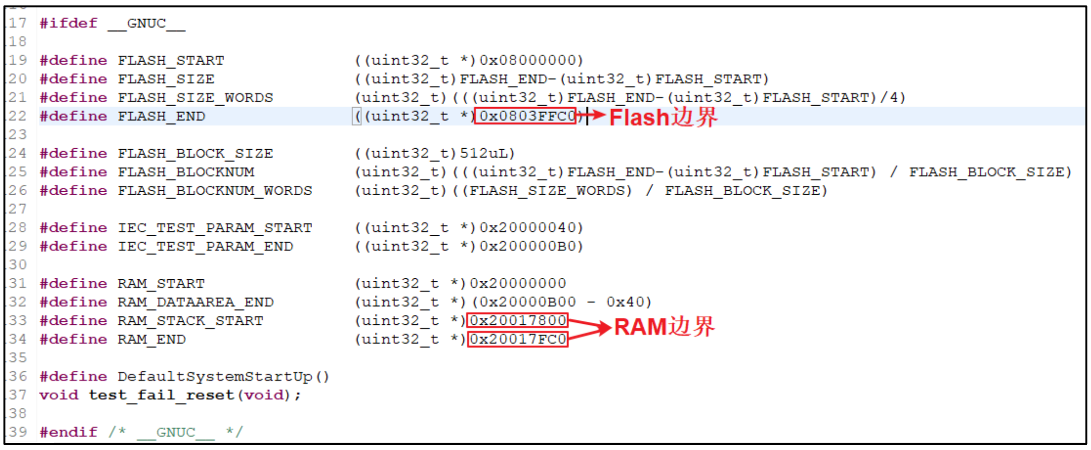
Compiler control string: -c99 -c --cpu Cortex-M4.fp -D__MICROLIB -g -O0 --apcs=interwork --split_sections -I..\..\..\..
\GD32F30x_Firmware_Library\CMSIS\GD\GD32F30x\Include -I..\..\..\..

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 gd32f30x_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 2-14. Modify the RAM boundary in the gd32f30x_test.h file***..

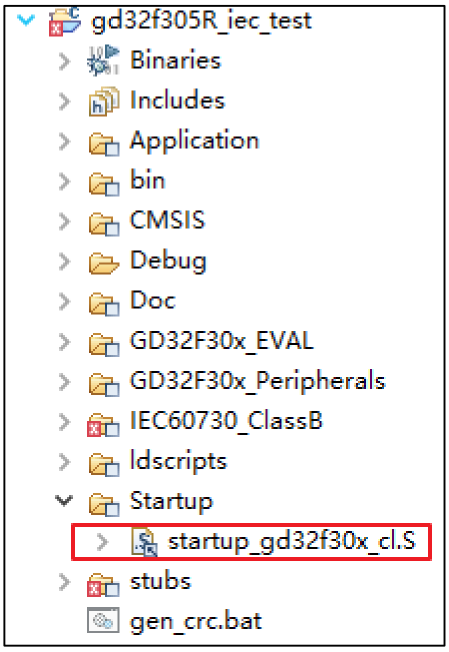
Figure 2-14. Modify the RAM boundary in the gd32f30x_test.h file.



```
#ifdef __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                ((uint32_t *)0x0803FFC0)

#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 *)0x20017800
#define RAM_END                  (uint32_t *)0x20017FC0

#define DefaultSystemStartUp()
void test_fail_reset(void);

#endif /* __GNUC__ */
```

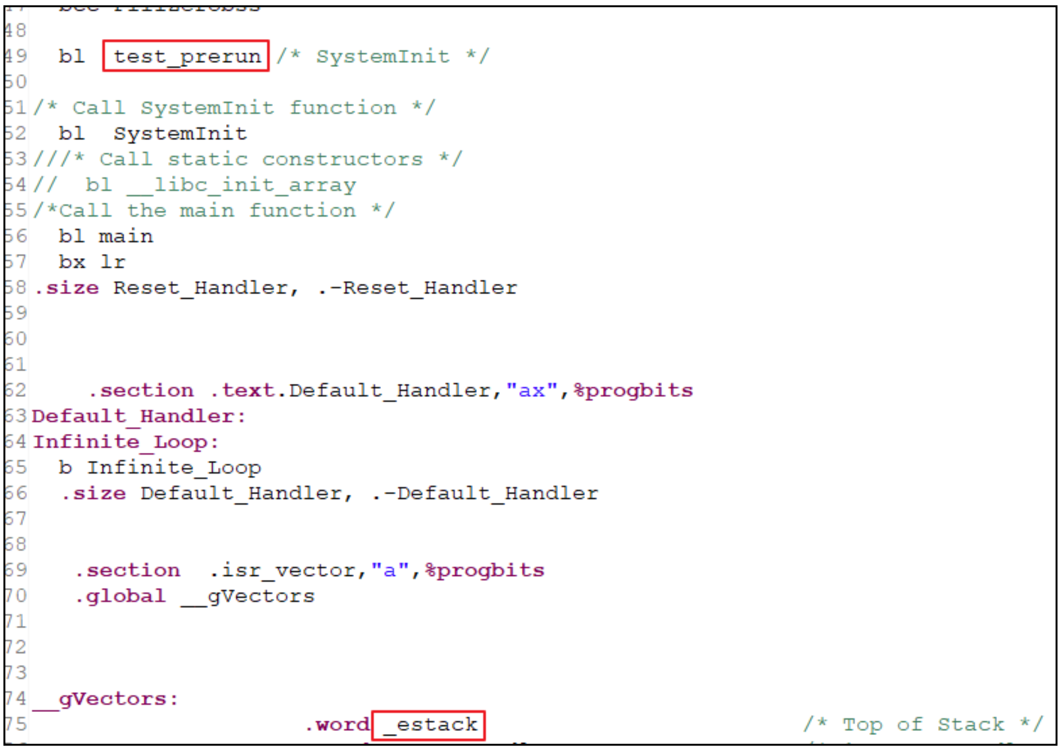
2. Check if the .S startup file in the project is suitable for the current chip, as shown in the ***Figure 2-15. Startup file check***, and *错误!未找到引用源。*.

Figure 2-15. Startup file check



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 2-16. Modify the startup file***.

Figure 2-16. 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 2-18. 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 2-17. Scatter loading file location

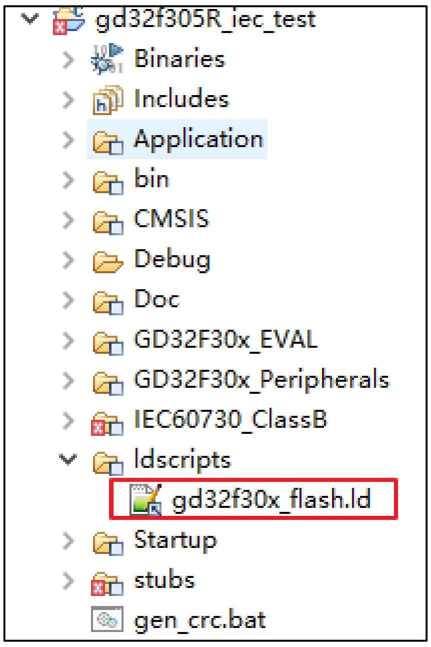
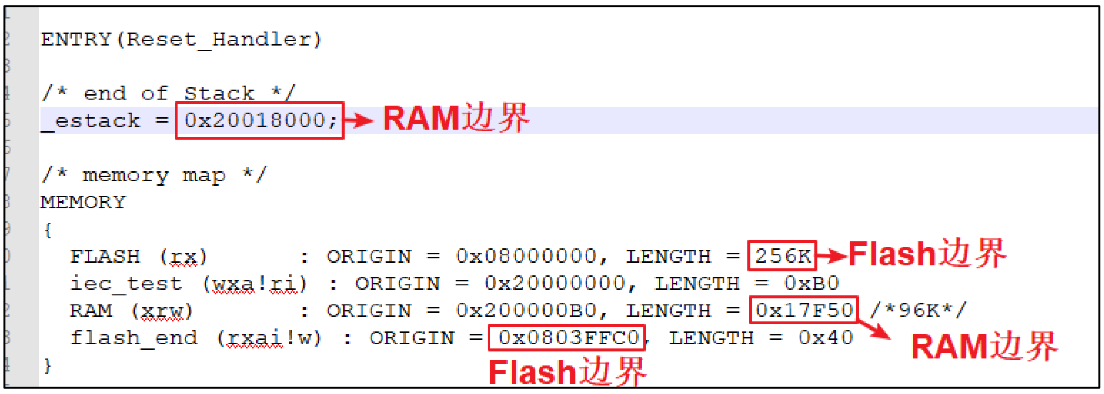


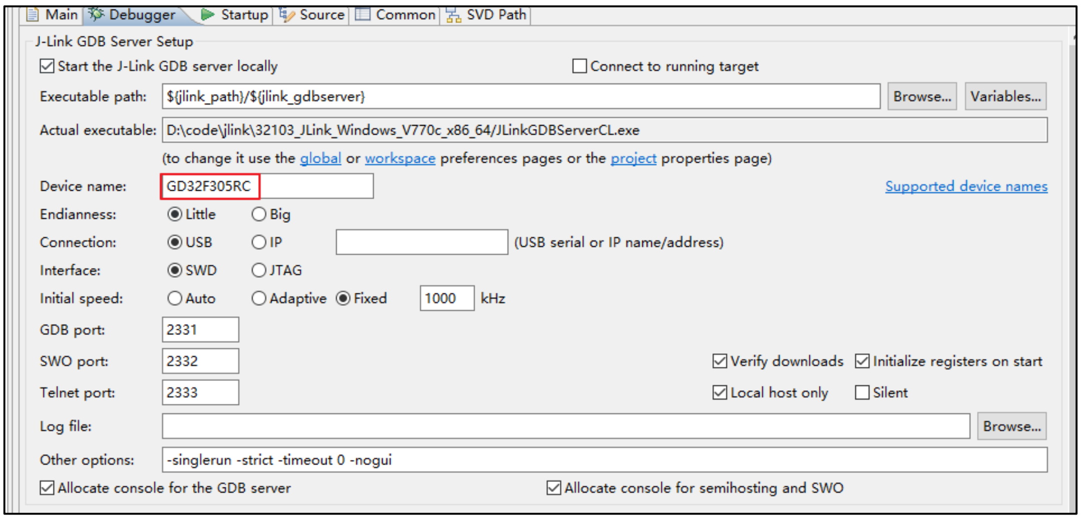
Figure 2-18. Modify the Eclipse project to load files in a ld file



```
ENTRY(Reset_Handler)

/* end of Stack */
_estack = 0x20018000;

/* memory map */
MEMORY
{
  FLASH (rx)       : ORIGIN = 0x08000000, LENGTH = 256K
  iec_test (wxa!ri) : ORIGIN = 0x20000000, LENGTH = 0xB0
  RAM (xrw)        : ORIGIN = 0x200000B0, LENGTH = 0x17F50 /*96K*/
  flash_end (rxai!w) : ORIGIN = 0x0803FFC0, LENGTH = 0x40
}
```

4. Modify the "Device name" configuration in "Debug Configurations->Debugger", select the model of the current chip, as shown in *Figure 2-19. Device name configuration*:

Figure 2-19. Device name configuration

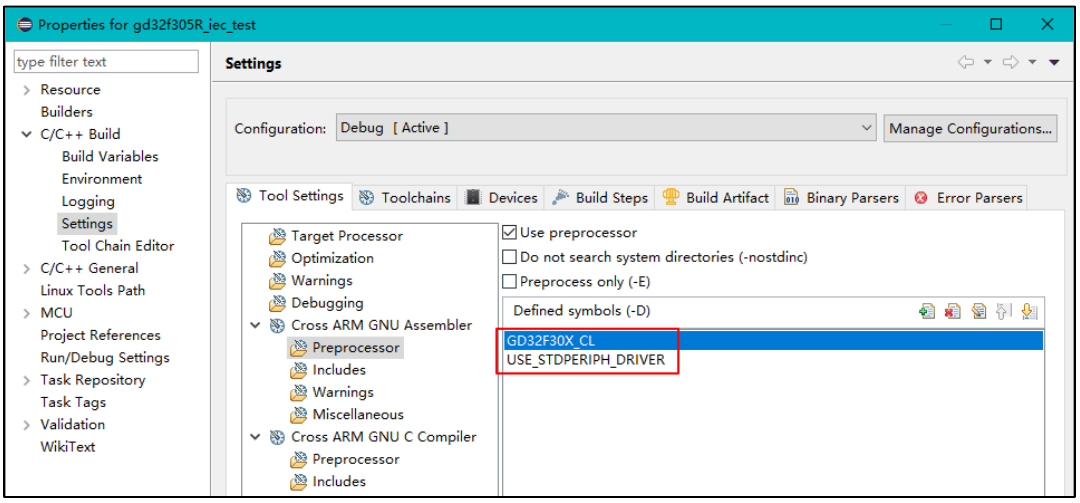


J-Link GDB Server Setup

- ☑ Start the J-Link GDB server locally
- ☐ Connect to running target
- Executable path: ${jlink_path}/${jlink_gdbserver}
- Actual executable: D:\code\jlink\32103_JLink_Windows_V770c_x86_64/JLinkGDBServerCL.exe
- (to change it use the global or workspace preferences pages or the project properties page)
- Device name: GD32F305RC
- 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
- ☑ Verify downloads
- ☑ Initialize registers on start
- ☑ Local host only
- ☐ Silent
- Log file:
- Other options: -singlerun -strict -timeout 0 -nogui
- ☑ Allocate console for the GDB server
- ☑ Allocate console for semihosting and SWO

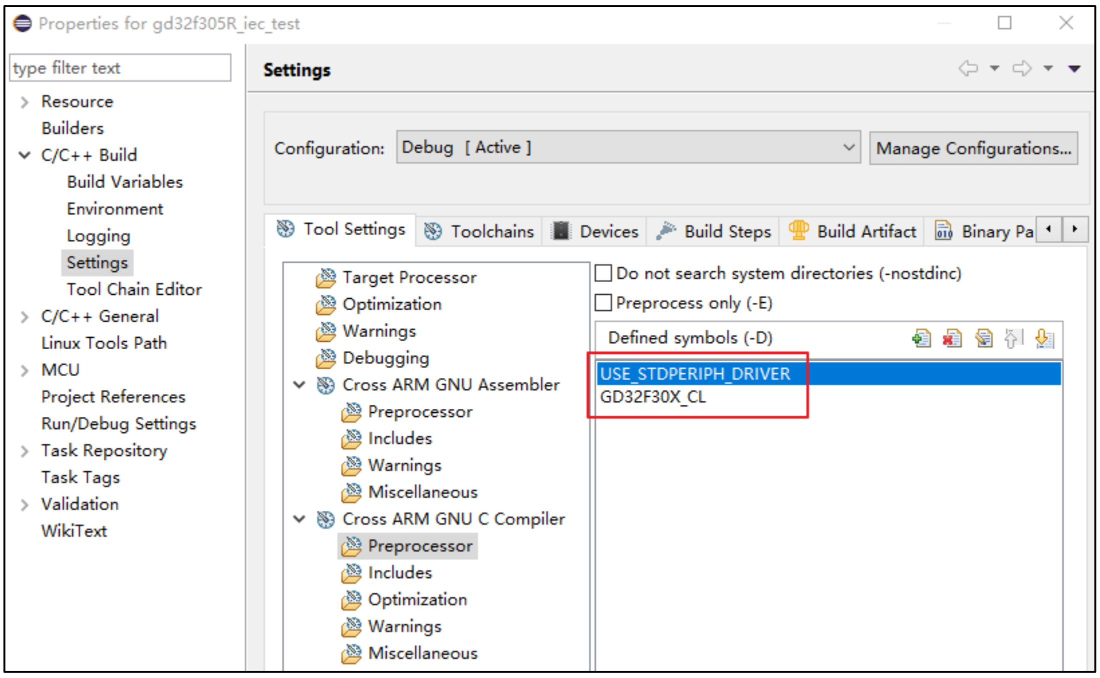
Add the necessary precompiled macros for the current project in the engineering properties "C/C++ Build -> Settings -> Tool Settings -> Cross ARM GNU Assembler -> Preprocessor" and "C/C++ Build -> Settings -> Tool Settings -> Cross ARM GNU C Compiler -> Preprocessor", mainly modifying the precompiled macros that conform to the current chip type (as shown in ***Figure 2-20. Modify the assembly compiler's precompiled macros*** and ***Figure 2-21.Modify the C language compiler's pre-processor macros***).

Figure 2-20. Modify the assembly compiler's precompiled macros



- Configuration: Debug [Active]
- ☑ Use preprocessor
- ☐ Do not search system directories (-nostdinc)
- ☐ Preprocess only (-E)
- Defined symbols (-D)
  - GD32F30X_CL
  - USE_STDPERIPH_DRIVER

Figure 2-21.Modify the C language compiler's pre-processor macros



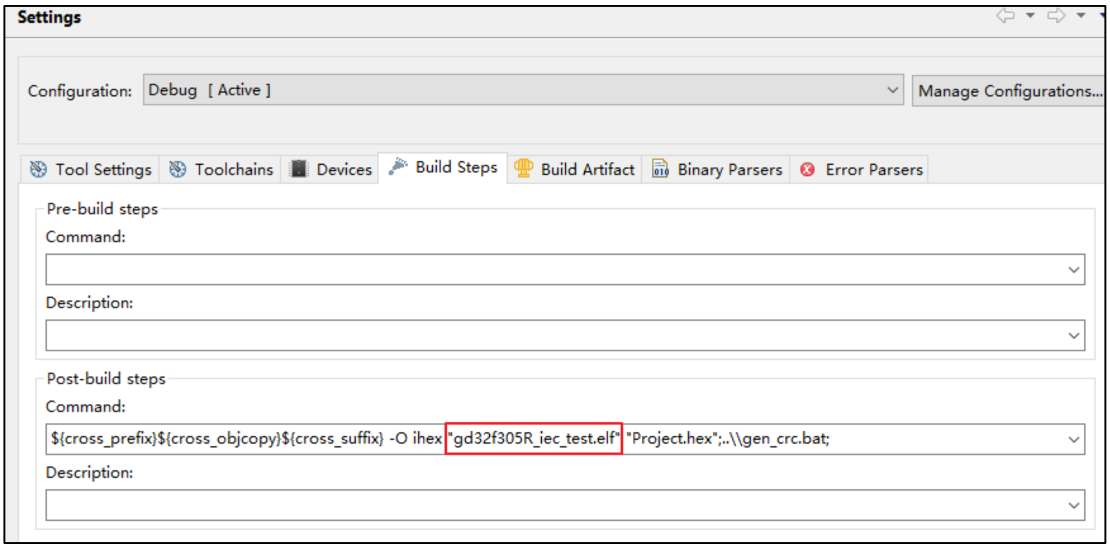
Configuration: Debug [Active]

☐ Do not search system directories (-nostdinc)
☐ Preprocess only (-E)

Defined symbols (-D)
USE_STDPERIPH_DRIVER
GD32F30X_CL

6. Modify the command in "C++ Build -> Settings -> Build Steps -> Post-build steps -> Command" as shown in ***Figure 2-22.Modify the post-build command*** to the ELF file of the current project name.

Figure 2-22.Modify the post-build command



Configuration: Debug [Active]

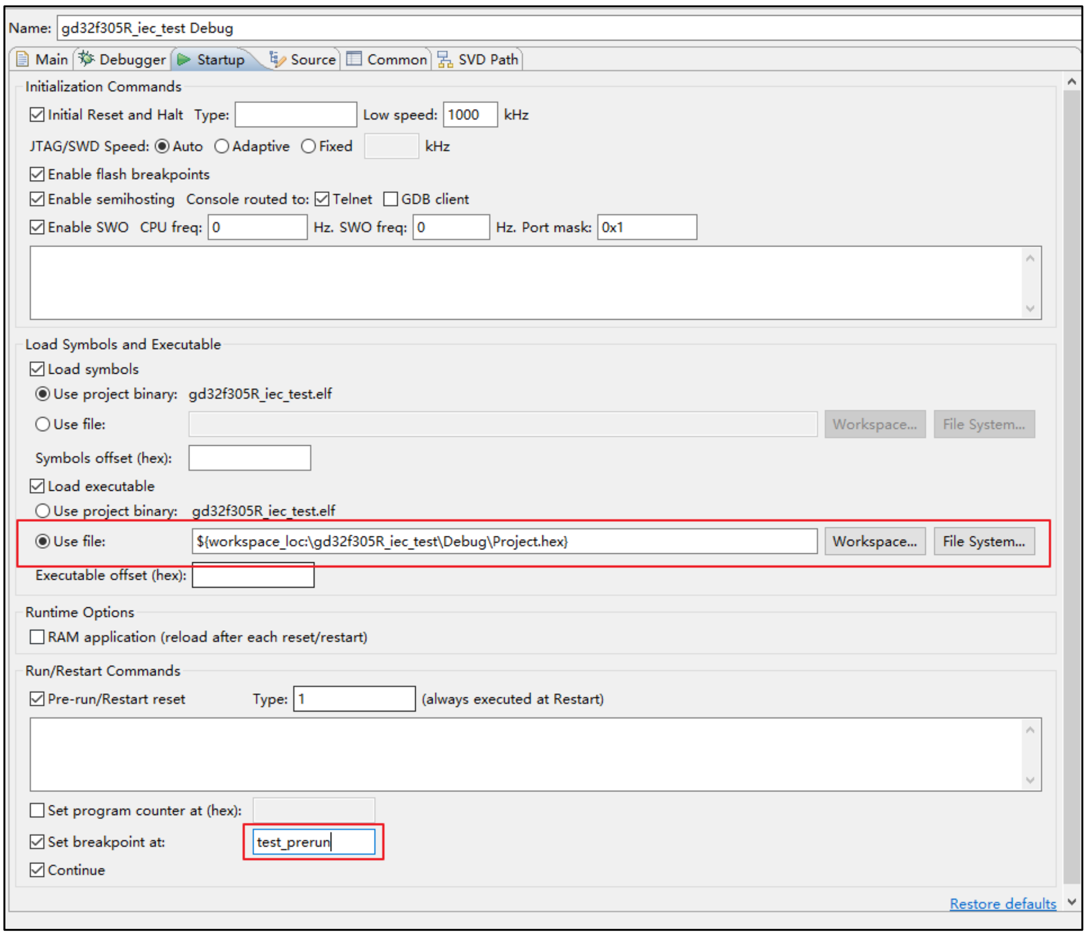
Pre-build steps
Command:

Description:

Post-build steps
Command:
```
${cross_prefix}${cross_objcopy}${cross_suffix} -O ihex "gd32f305R_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 2-23. Modify the configuration of the executable file***.

Figure 2-23. Modify the configuration of the executable file



Name: gd32f305R_iec_test Debug

Initialization Commands

[✓] Initial Reset and Halt Type: Low speed: 1000 kHz

JTAG/SWD Speed: [✓ Auto] Adaptive Fixed kHz

[✓] Enable flash breakpoints

[✓] Enable semihosting Console routed to: [✓] Telnet [ ] GDB client

[✓] Enable SWO CPU freq: 0 Hz. SWO freq: 0 Hz. Port mask: 0x1

Load Symbols and Executable

[✓] Load symbols

[✓ Use project binary:] gd32f305R_iec_test.elf

Use file:

Symbols offset (hex):

[✓] Load executable

Use project binary: gd32f305R_iec_test.elf

[✓ Use file:] ${workspace_loc:\gd32f305R_iec_test\Debug\Project.hex}

Executable offset (hex):

Runtime Options

[ ] RAM application (reload after each reset/restart)

Run/Restart Commands

[✓] Pre-run/Restart reset Type: 1 (always executed at Restart)

[ ] Set program counter at (hex):

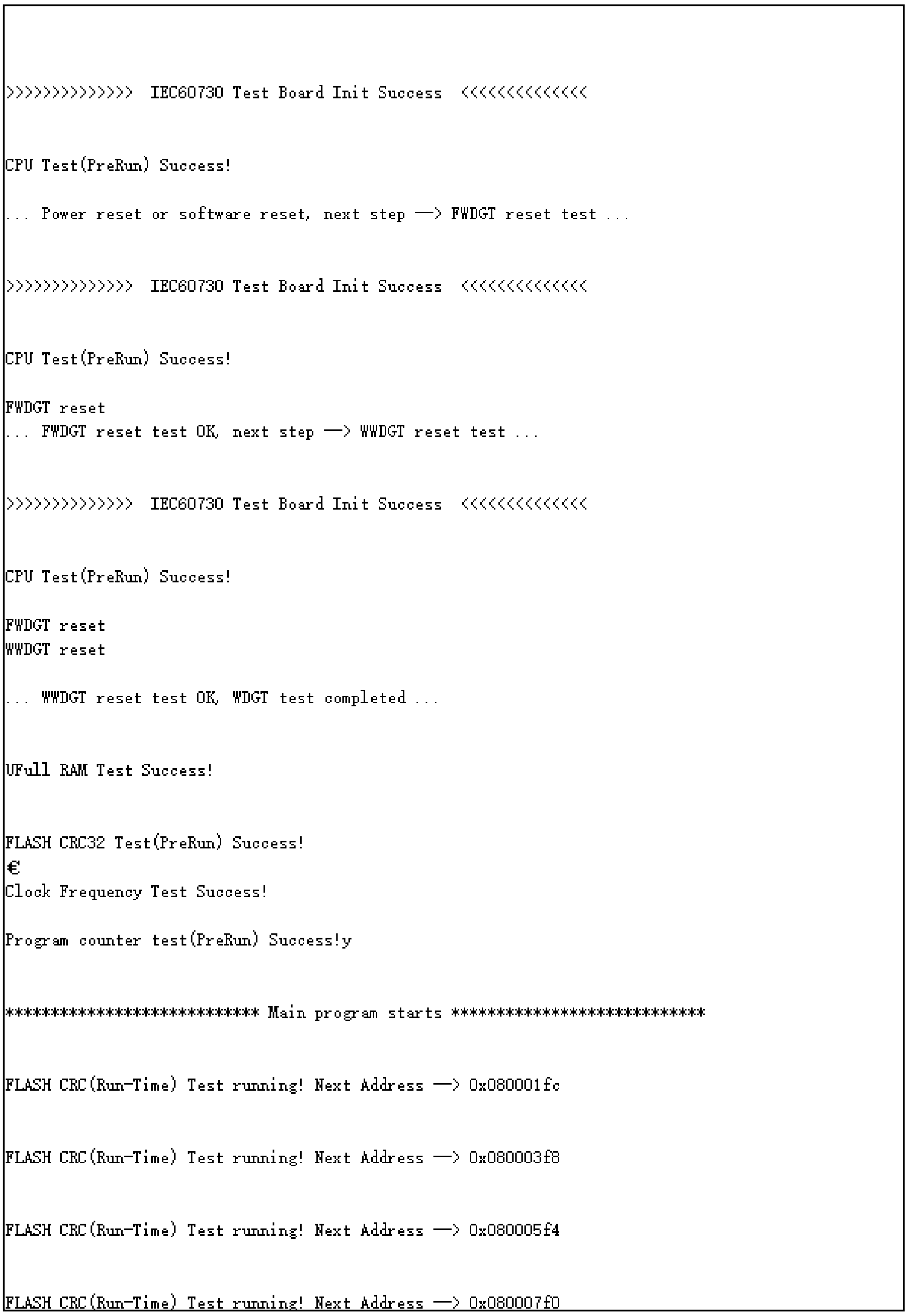
[✓] Set breakpoint at: test_prerun

[✓] 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 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

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