GigaDevice Semiconductor Inc.

Guideline for migrating the IEC60730 ClassB library onto GD32F3x0 series

Application Note
AN171

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 GD32F3x0 series in various Integrated Development Environments (IDEs) such as Keil, IAR, and Eclipse, guiding users in porting the IEC60730 Class B certification library.

2. Migration of the IEC60730 class B certification library

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

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

2.1. Migration of certification library project in IAR environment

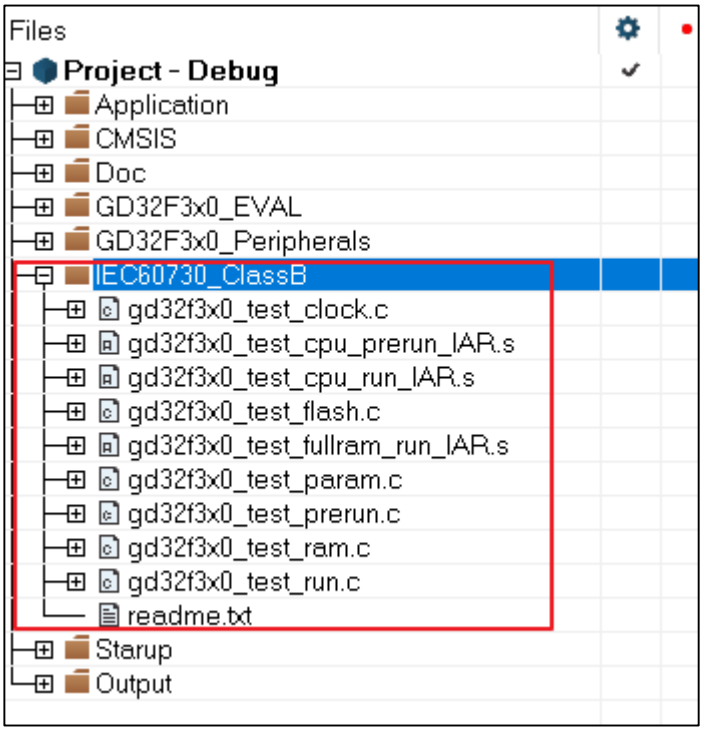
1. Modify the RAM boundary in the macro __IAR_SYSTEMS_ICC__ of the gd32f3x0_test.h file according to the datasheet of the chip, as shown in ***Figure 2-1. Modify the RAM boundary in the gd32f3x0_test.h***. Modify the RAM boundary in the gd32f3x0_test.h as indicated.

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

```
#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 int *)&__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                 ((uint32_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 *)0x20003F40

#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. At the same time, add the ClassB library file to the project.

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



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

Figure 2-3. Modify the .s startup file

```
        MODULE  ?cstartup

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

        SECTION .intvec:CODE:NOROOT(3)

        EXTERN  test_prerun

        EXTERN  __iar_program_start
        EXTERN  SystemInit
        PUBLIC  __vector_table

        DATA
__vector_table
        DCD     sfe(CSTACK)                     ; top of stack
        DCD     test_prerun                     ; Vector Number 1,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
```

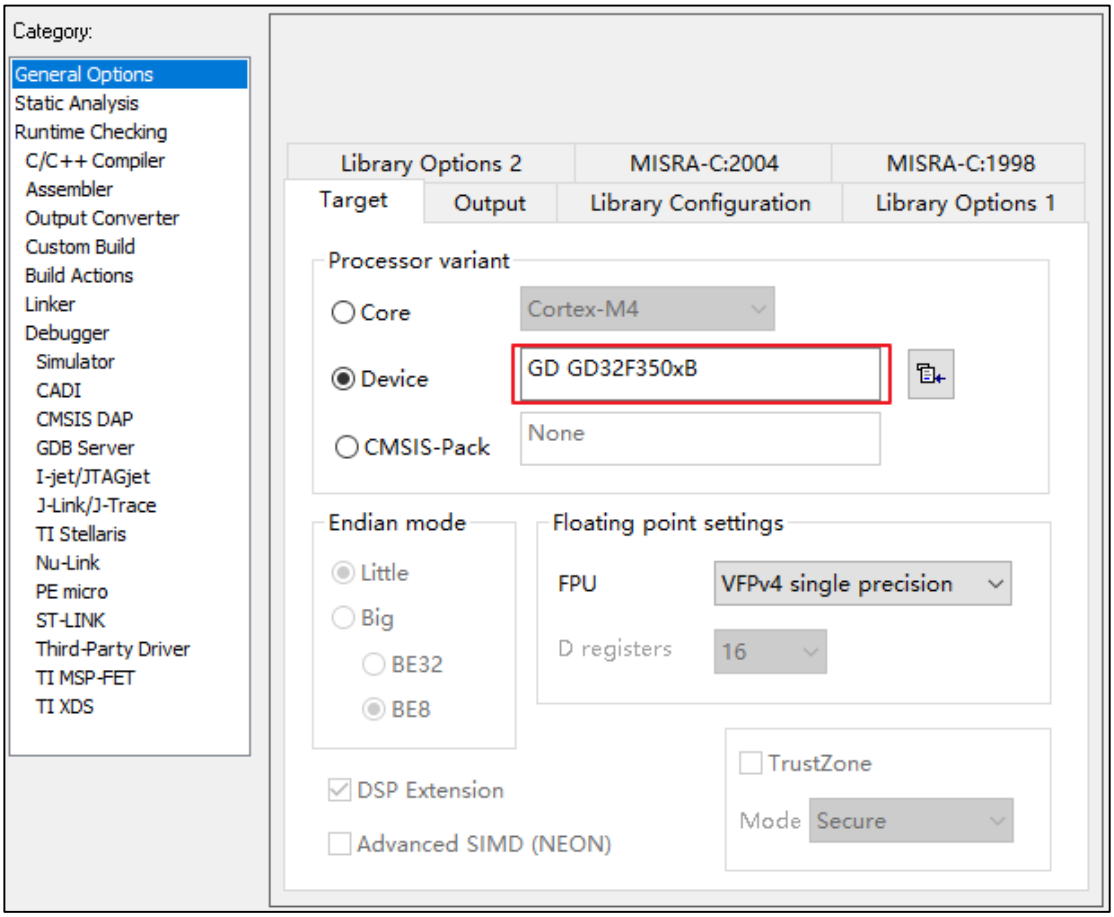
3. Modify the project's scatter loading file (in this case, ..\GD32F3x0_IEC_Test\GD32350R_EVAL_Demo_Suites\Projects\IEC_Test\EWARM\IEC_TEST_BOOT_FLASH.icf) in the IAR environment. As shown in ***Figure 2-4. Modify the scattered loading file***, adjust the sections for Flash and RAM sizes to match the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

Figure 2-4. Modify the scattered loading file

```
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  define symbol __ICFEDIT_intvec_start__ = 0x08000000;
6  /*-Symbols-*/
7  define symbol __ICFEDIT_region_ROM_start__           = 0x08000000;
8  define symbol __ICFEDIT_region_ROM_end__             = 0x0801FFFF;
9  define symbol __ICFEDIT_region_RAM_start__           = 0x200000B0;
10 define symbol __ICFEDIT_region_RAM_end__             = 0x20003FFF;
11 define symbol __ICFEDIT_region_IECTEST_PARAM_start__ = 0x20000040;
12 define symbol __ICFEDIT_region_IECTEST_PARAM_end__   = 0x200000B0;
```

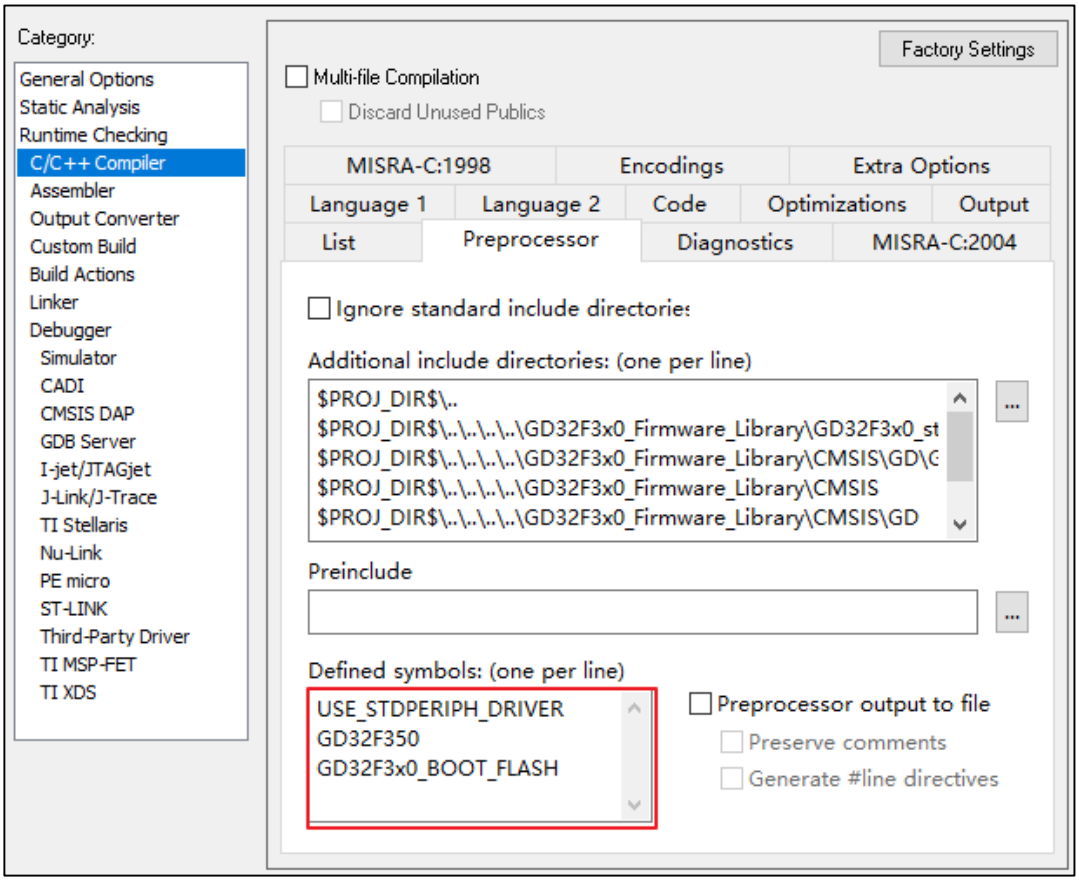
4. Update the configuration for the 'Target' within 'Options for Target' in the IAR project settings, selecting the current chip model, as depicted in ***Figure 2-5. Device Configuration***.

Figure 2-5. Device Configuration



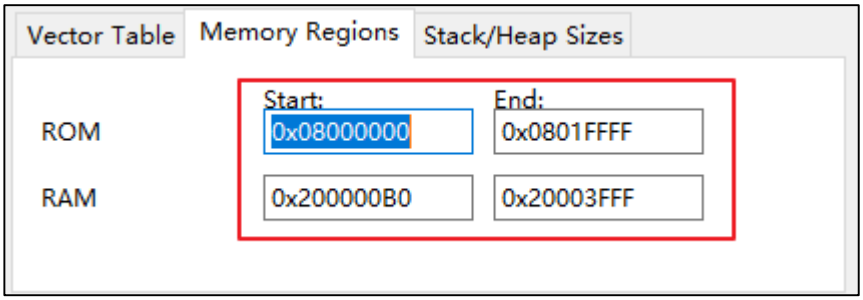
5. In the Options for the 'Project' node -> C/C++ Compiler -> Preprocessor, add the necessary preprocessor macros for the current project, mainly modifying the macros that match the type of the current chip, as shown in ***Figure 2-6. Modify the precompiled macro***.

Figure 2-6. Modify the precompiled macro



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



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

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

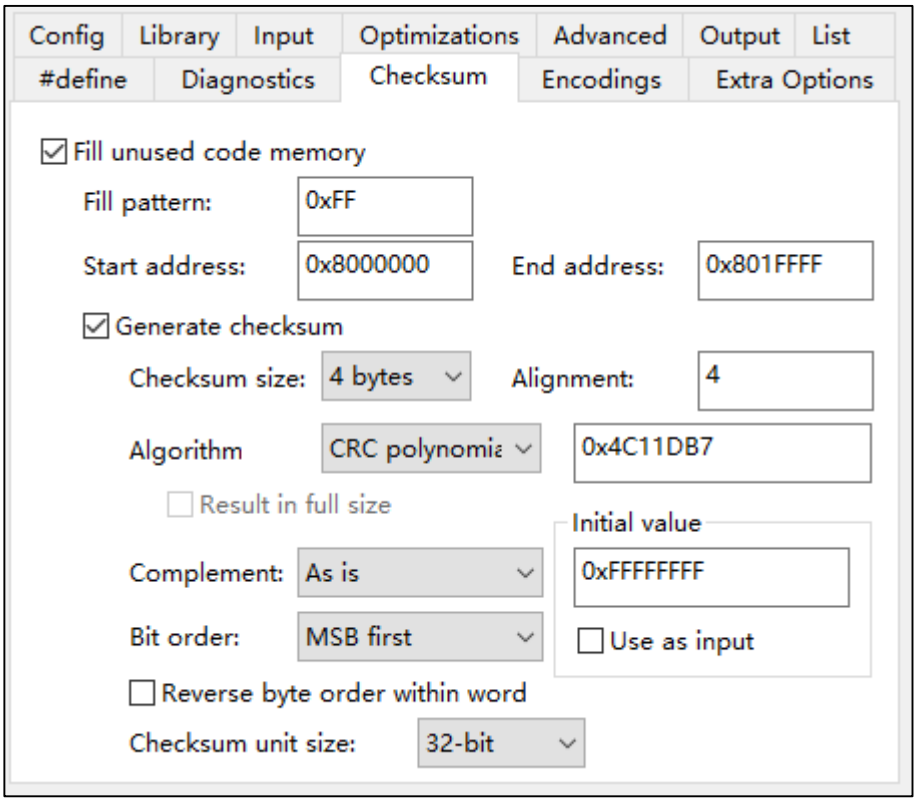
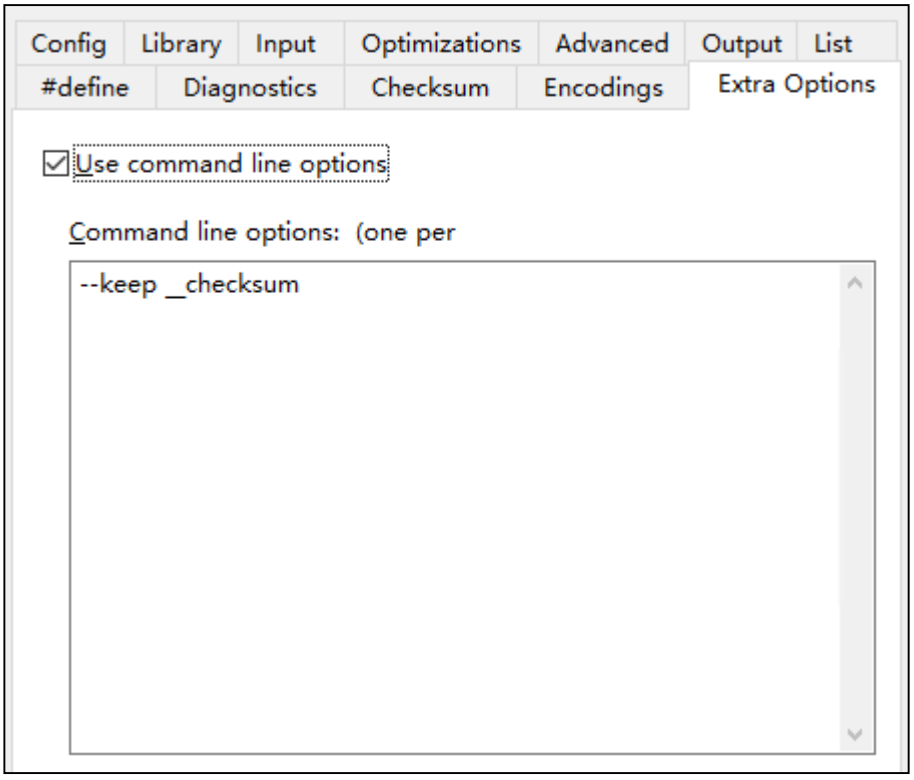


Figure 2-9. Add configurations to the Extra Options tab



2.2. Migration of certification library project in Keil environment

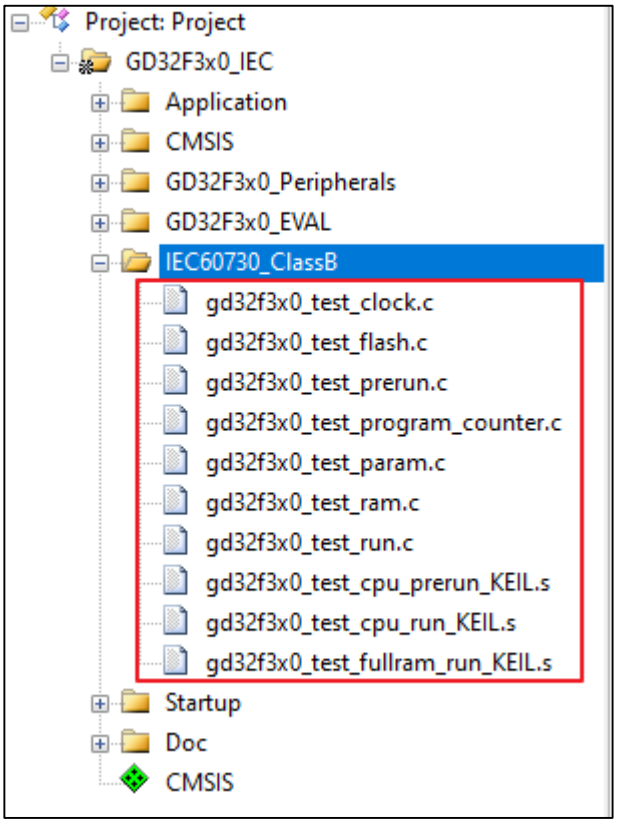
1. Modify the RAM and Flash boundaries in the gd32f3x0_test.h file according to the datasheet of the current chip model, as shown in ***Figure 2-10. Modify the RAM and Flash boundaries in the gd32f3x0_test.h file***.

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

```
#if defined (__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)

#define FLASH_START                ((uint32_t *)0x08000000)
#define FLASH_SIZE                 ((uint32_t 0x00020000 - 4)
#define FLASH_SIZE_WORDS           (uint32_t)(((uint32_t)FLASH_END-(uint32_t)FLASH_START)/4)
#define FLASH_END                  ((uint32_t *)0x08020000)

#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 *0x20003F40

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

#endif /* (__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050) */
```

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

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



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

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

```
Stack_Size      EQU     0x00000400

                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
                ......
                ......
                ......
```

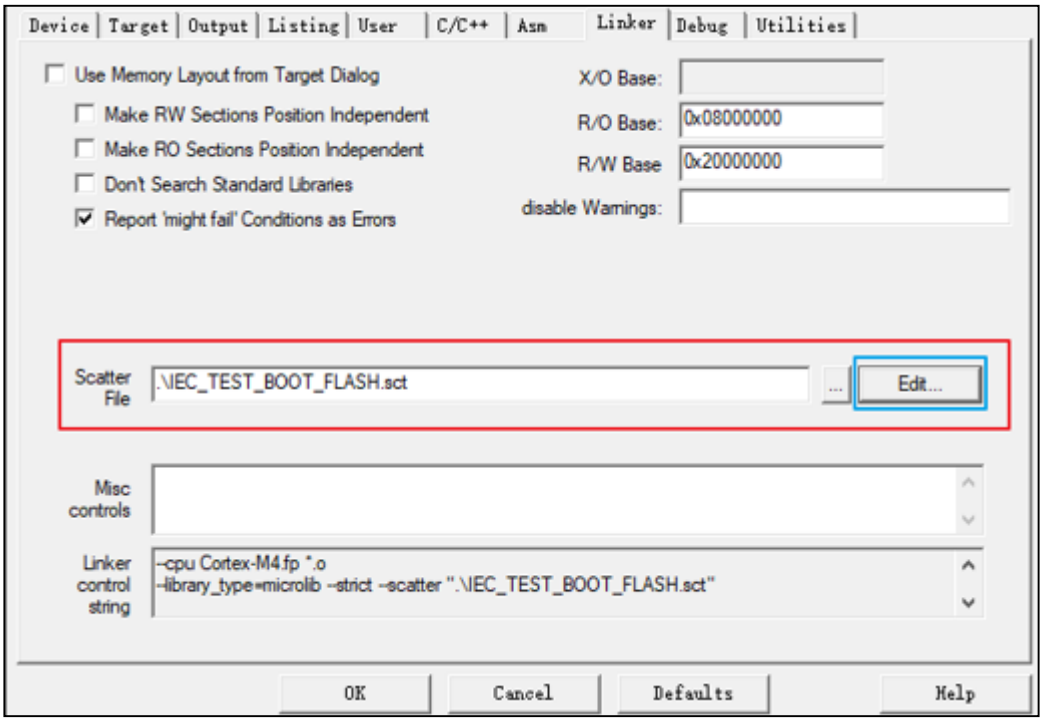
3. Modify the code in the scatter loading file IEC_TEST_BOOT_FLASH.sct, as shown in the following table, the parts marked in red need to be modified (modify the Flash boundary according to the current chip's datasheet; change the name of the .o file corresponding to the .s startup file).

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



LR_IROM1 0x08000000 0x0001FFF8{
    ER_IROM1 0x08000000   0x0001FFF8{
        *.o (RESET, +First)
        *(InRoot$$Sections)
        .ANY (+RO)
    }

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


    ; RW data
    RW_IRAM1 0x200000B0   0x0002000
    {
        .ANY (+RW +ZI)
    }
```

```
    ; stack overflow test
    STACK_IRAM2 0x200020B0 UNINIT 0x00001F40
    {
        gd32f3x0_test_param.o (STACK_OV_TEST, .bss.STACK_OV_TEST)
        startup_gd32f3x0.o (STACK, +Last)
    }
}

LR_IROM2 0x0801FFFC 0x0000004 {
    ER_IROM2 0x0801FFFC 0x0000004
    {
        *.o (CHECKSUM, +Last)
    }
}
```

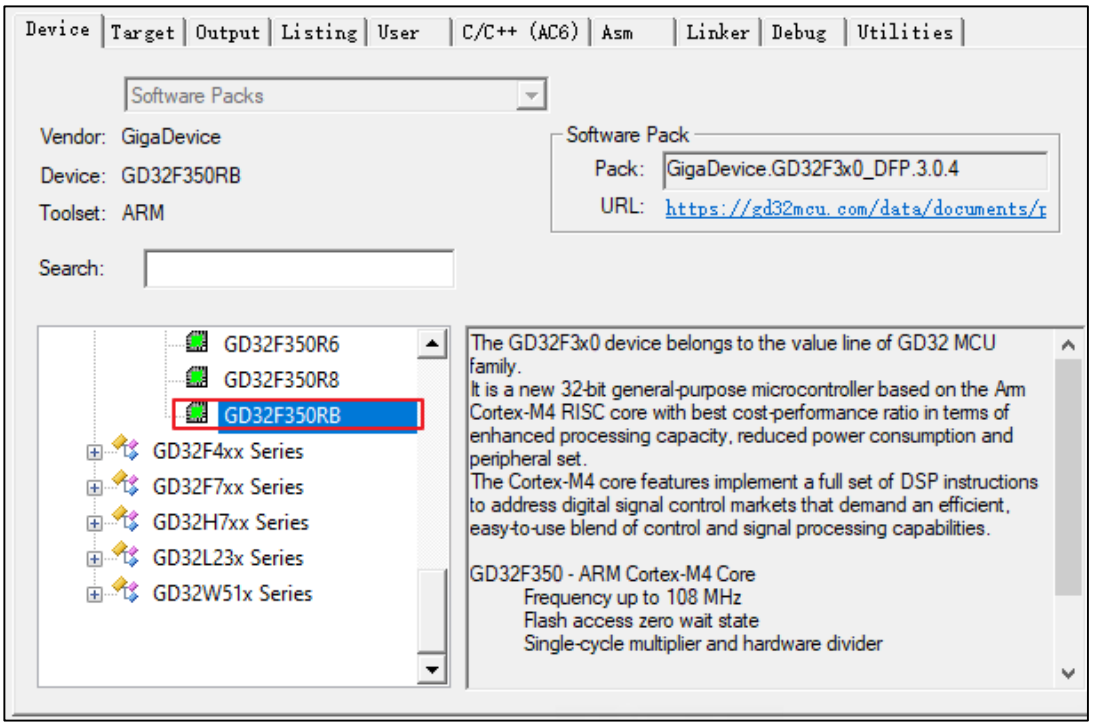
Scatter loading files can be modified by clicking the Edit button in "Options for Target --> Linker --> Scatter File", as shown in ***Figure 2-12. Editing the scatter loading file***: Editing the scatter loading file.

Figure 2-12. Editing the scatter loading file



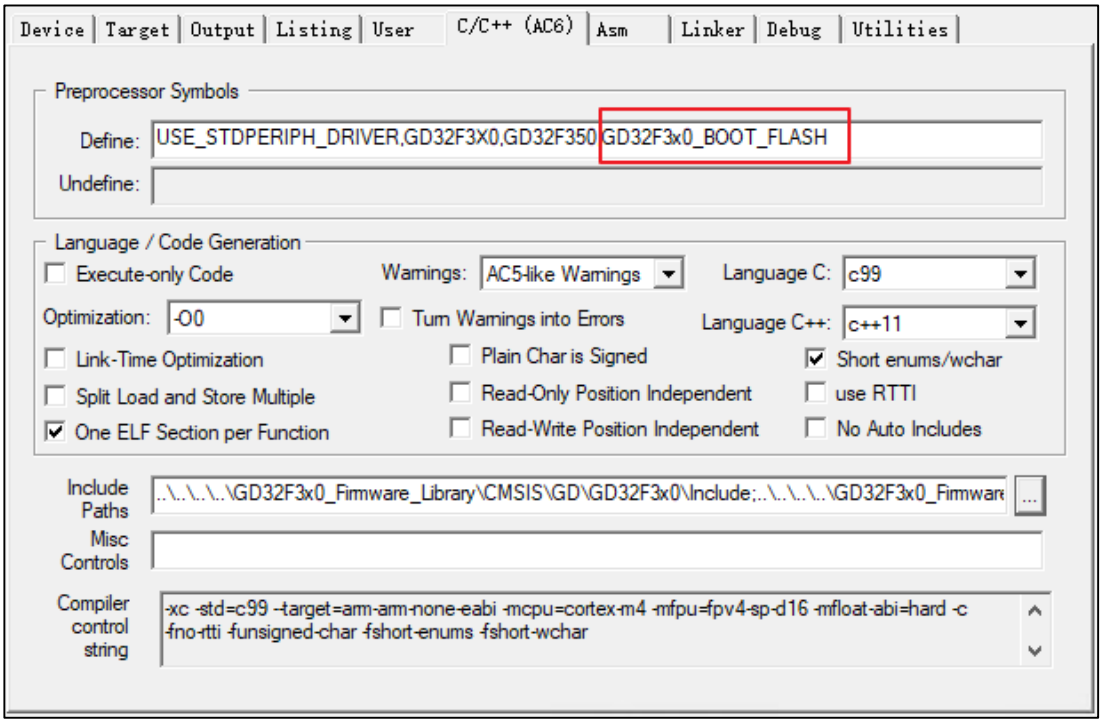
4. Modify the "Options for Target" under "Device" to select the current chip model, as shown in ***Figure 2-13. Device Configuration***.

Figure 2-13. Device Configuration



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 2-14. Modify the precompiled macro***:

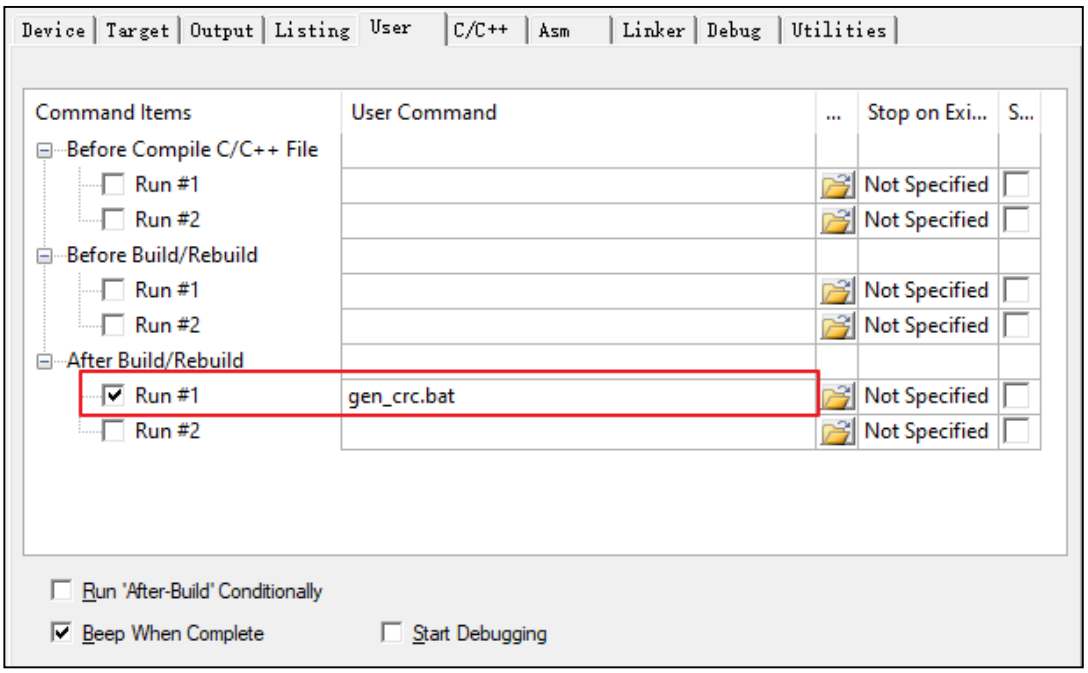
Figure 2-14. Modify the precompiled macro



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

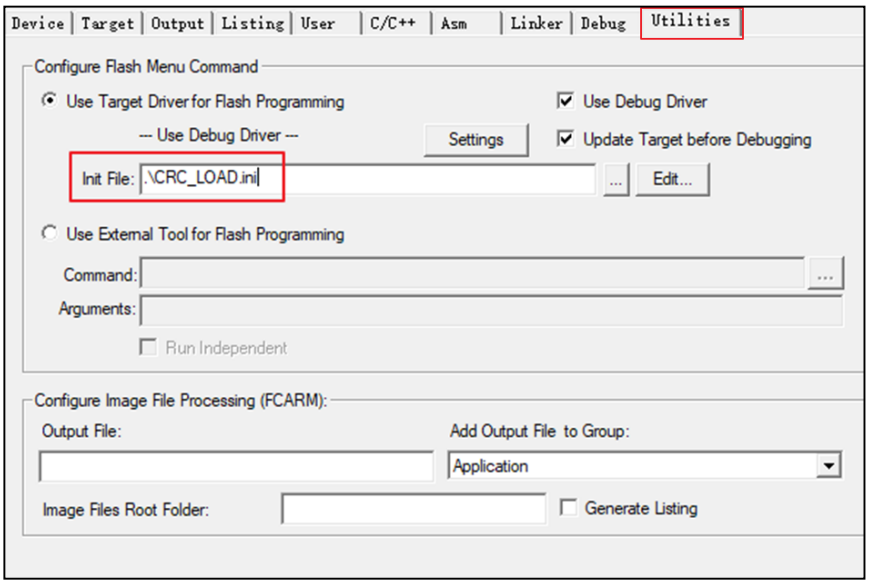
gen_crc.bat file needs to be added, as shown in ***Figure 2-15.Add a batch file***.

Figure 2-15.Add a batch file



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

Figure 2-16. Add a .ini configuration file



2.3. Migration of certification library project in eclipse environment

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

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

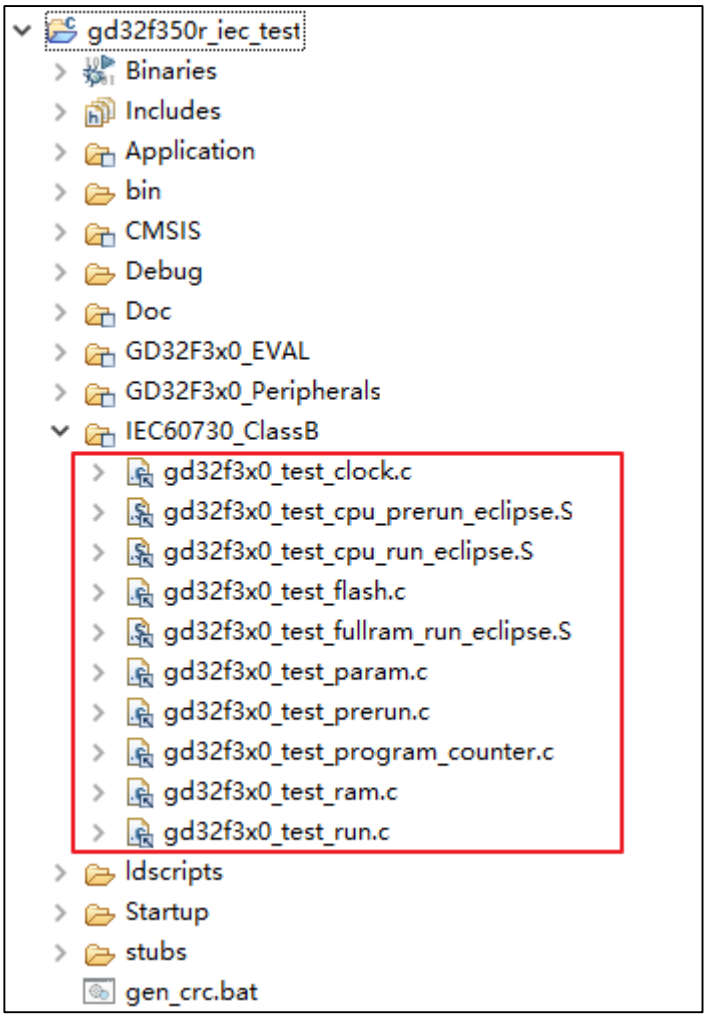
1. Modify the RAM boundary in the macro __GNU__ of the gd32f3x0_test.h file according to the datasheet to ensure that the entire space of the chip's Flash and RAM is detected, as shown in ***Figure 2-17. Modify the RAM boundary in the gd32f3x0_test.h file***.

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

```
#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                  ((uint32_t *)0x0801FFC0)

#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 *)0x20003800
#define RAM_END                    (uint32_t *)0x20003FC0

#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, and ***Figure 2-18. Add the ClassB library file to the Eclipse project***.

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



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

Figure 2-19. Modify the startup file

```
48
49  bl  test_prerun /* SystemInit */
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
59
60
61
62    .section .text.Default_Handler,"ax",%progbits
63Default_Handler:
64Infinite_Loop:
65  b Infinite_Loop
66  .size Default_Handler, .-Default_Handler
67
68
69   .section  .isr_vector,"a",%progbits
70   .global __gVectors
71
72
73
74__gVectors:
75                    .word _estack                    /* Top of Stack */
```

3. Modify the scatter-loading file in the scatter-loading file directory, as shown in ***Figure 2-21. 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 2-20. Scatter loading file location

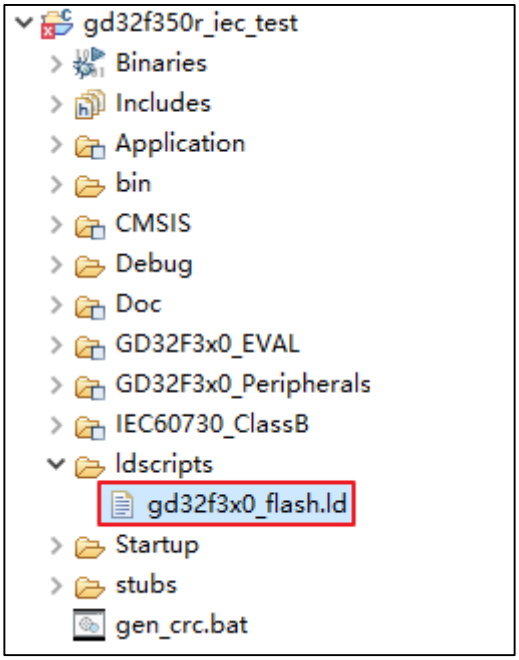
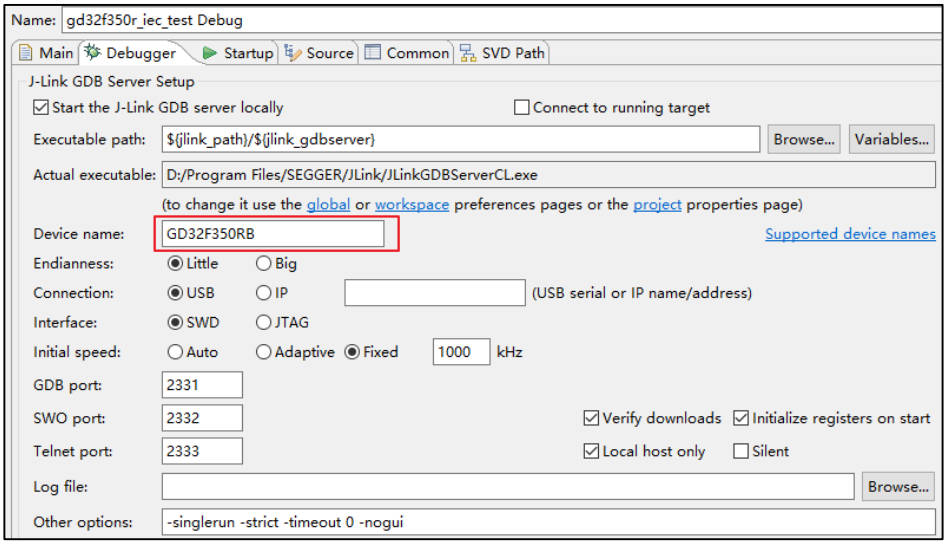


Figure 2-21. Modify the Eclipse project to load files in a ld file

```
1
2 ENTRY(Reset_Handler)
3
4 /* end of Stack */
5 _estack = 0x20004000;
6
7 /* memory map */
8 MEMORY
9 {
10   FLASH (rx)      : ORIGIN = 0x08000000, LENGTH = 128K
11   iec_test (wxa!ri) : ORIGIN = 0x20000000, LENGTH = 0xB0
12   RAM (xrw)       : ORIGIN = 0x200000B0, LENGTH = 0x3F50 /*8K*/
13   flash_end (rxai!w) : ORIGIN = 0x0801FFC0, LENGTH = 0x40
14 }
```

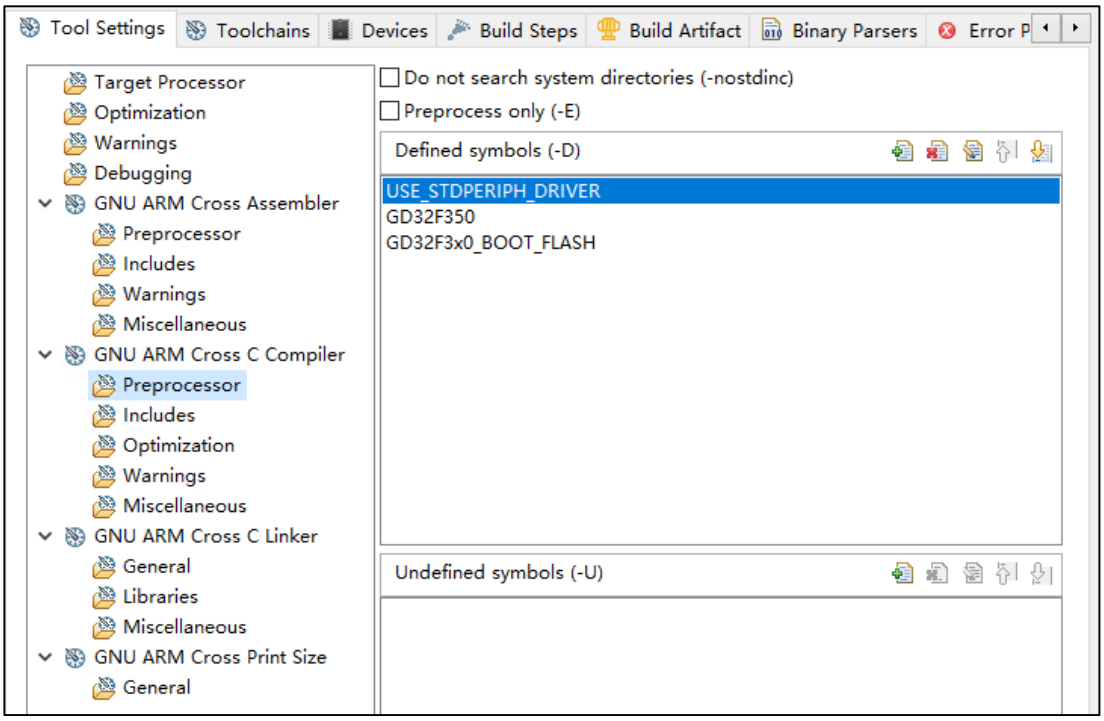
4. Modify the "Device name" configuration in "Debug Configurations->Debugger", select the model of the current chip, as shown in ***Figure 2-22. Device name configuration***:

Figure 2-22. Device name configuration



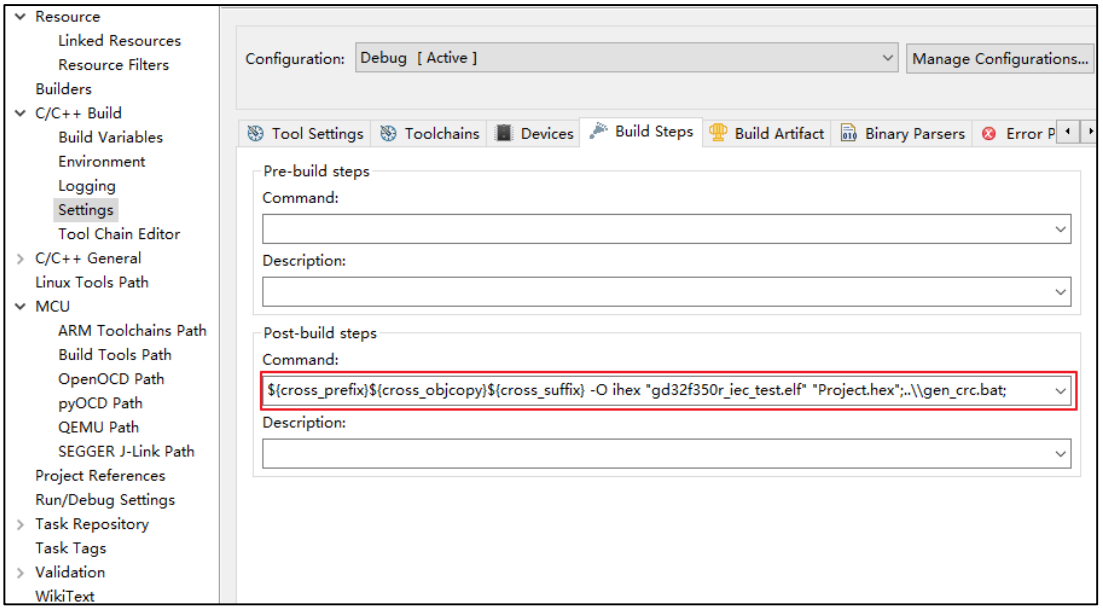
Add the necessary precompiled macros for the current project in the engineering properties "C/C++ Build -> Settings -> Tool Settings -> Cross ARM GNU Assembler -> Preprocessor" and "C/C++ Build -> Settings -> Tool Settings -> Cross ARM GNU C Compiler -> Preprocessor", mainly modifying the precompiled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

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



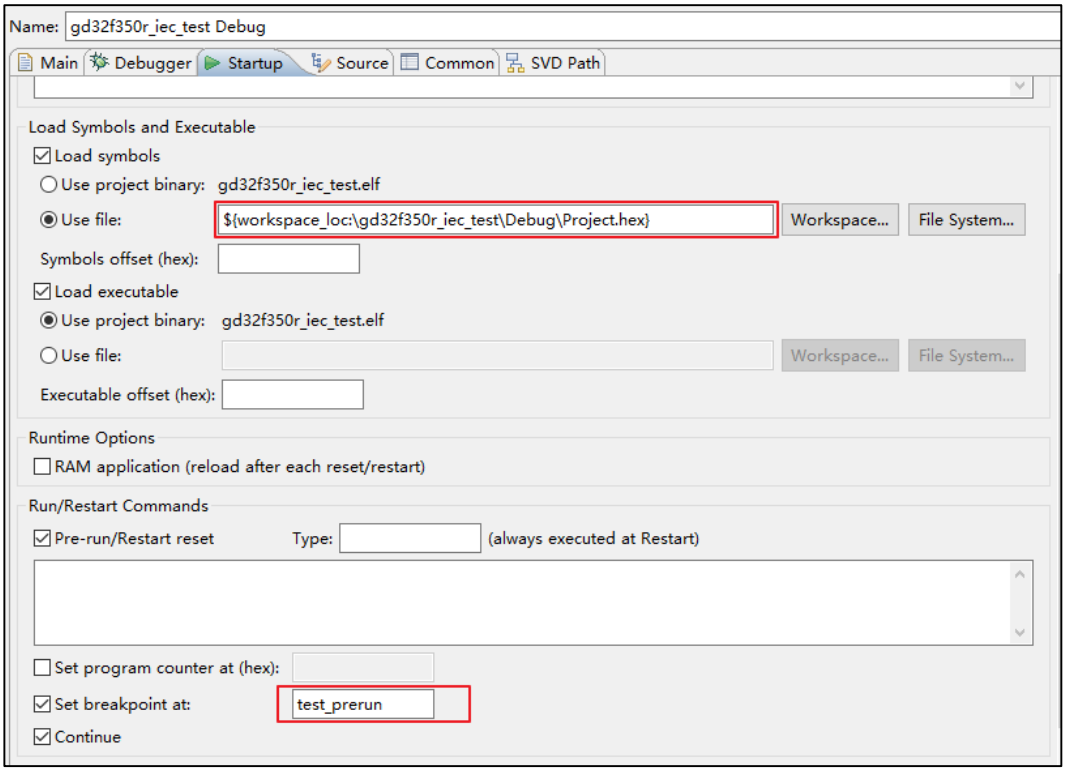
6. Modify the command in "C++ Build -> Settings -> Build Steps -> Post-build steps -> Command" as shown in ***Figure 2-24.Modify the post-build command*** to the ELF file of the current project name.

Figure 2-24.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 2-25. Modify the configuration of the executable file***.

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



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.