GigaDevice Semiconductor Inc.

基于 GD32F4xx 系列的 IEC60730 ClassB 库移植指南

应用笔记

AN156

1.0 版本

（2023 年 8 月）

目录

图索引

表索引

1. 简介

GD32 MCU提供IEC60730 ClassB认证库支持，对于每个系列的GD32 MCU，都会提供常用系列的工程模板，用户在对同一系列不同型号的芯片进行IEC60730自检认证时，可通过移植模板程序来适配目标芯片。本应用笔记将详细介绍GD32F4xx系列芯片在不同IDE（Keil/IAR/eclipse）下，移植过程中的注意事项，帮助客户移植IEC60730 ClassB认证库。

2. IEC60730 ClassB 认证库移植

GD32 MCU提供IAR、Keil、eclipse环境下的IEC60730 ClassB认证库支持，三种开发环境下模板工程的移植存在差异。下面将从三种开发环境展开说明。

本应用笔记基于GD32F470IK进行工程移植说明。

2.1. IAR 环境认证库工程移植

1、根据芯片数据手册修改gd32f4xx_test.h文件中编译宏__IAR_SYSTEMS_ICC__下的RAM边界，如*图2-1. 修改gd32f4xx_test.h中RAM边界*所示。

图 2-1. 修改 gd32f4xx_test.h 中 RAM 边界

```
#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、检查工程目录下Starup文件夹下的.s启动文件，是否符合当前芯片类型，如*图2-2. 启动文件检查*所示，同时添加ClassB库文件到工程中。

图 2-2. 启动文件检查

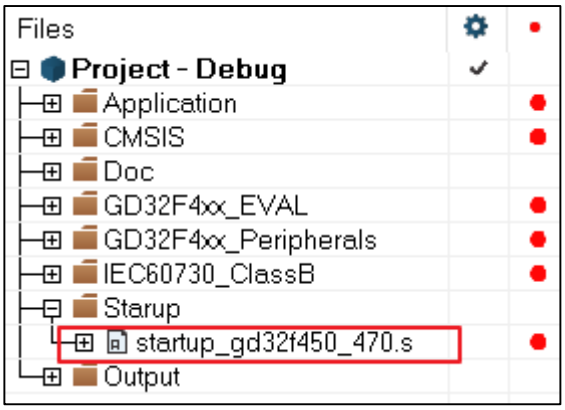
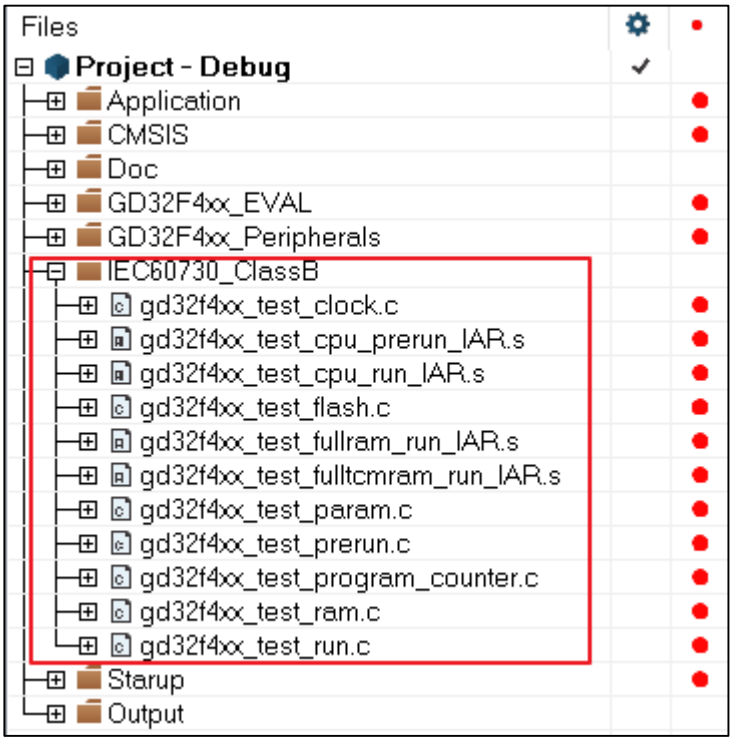


图 2-3. 添加 ClassB 库文件到 IAR 工程



若启动文件符合当前芯片型号，则无需修改；若不符合，则需要在固件库文件夹GD32F4xx_Firmware_Library\CMSIS\GD\GD32F4xx\Source\ARM下，重新选择符合当前芯片类型的.s启动文件，并如

图2-4. 修改.s**启动文件**所示进行修改，使芯片上电之后先进行自检测。

图 2-4. 修改.s 启动文件

```
        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
```

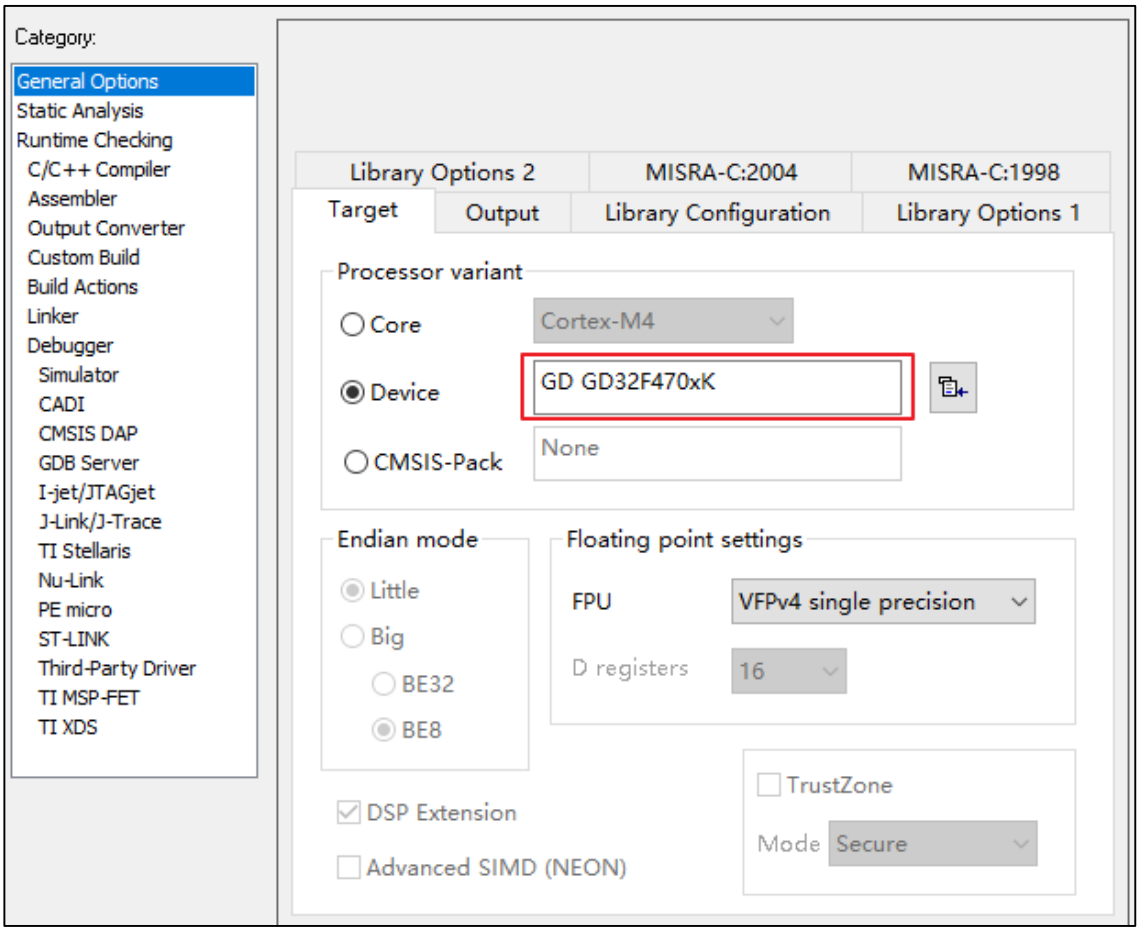
3 、 修 改 工 程 所 在 文 件 夹 （ 在 本 例 中 为：..\GD32F4xx_IEC_Test\GD32470I_EVAL_Demo_Suites\Projects\IEC_Test\EWARM）下的IEC_TEST_BOOT_FLASH.icf文件，如*图2-5. 修改分散加载文件*所示为修改的部分，根据芯片数据手册修改Flash和RAM的大小适配当前芯片。

图 2-5. 修改分散加载文件

```
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__             = 0x082FFFFF;
9   define symbol __ICFEDIT_region_RAM_start__           = 0x200000B0;
10  define symbol __ICFEDIT_region_RAM_end__             = 0x2002FFFF;
11  define symbol __ICFEDIT_region_IECTEST_PARAM_start__ = 0x20000040;
12  define symbol __ICFEDIT_region_IECTEST_PARAM_end     = 0x200000B0;
```

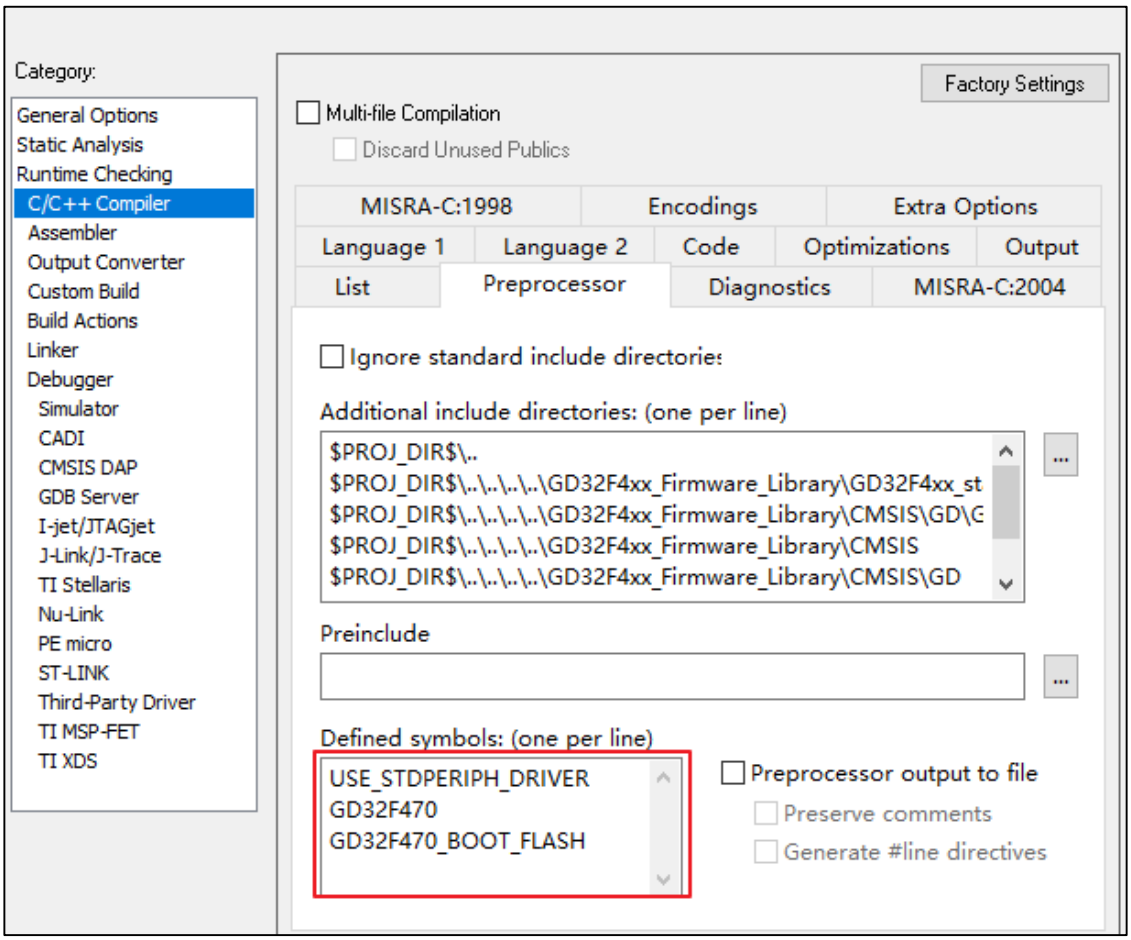
4、修改Options for node ‘Project’中Target的配置，选择当前芯片的型号，如*图2-6. Device配置*所示：

图 2-6. Device 配置



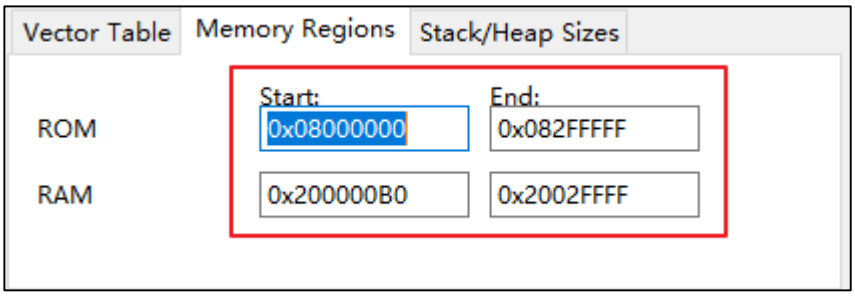
5、在Options for node ‘Project’->C/C++ Complier->Preprocessor中添加当前工程需要的预编译宏，主要修改符合当前芯片类型的预编译宏，如*图2-7. 修改预编译宏*所示：

图 2-7. 修改预编译宏



6、点击Options for node ‘Project’->Linker->config->Edit->Memory Regions，根据当前芯片的数据手册修改ROM (Flash)和RAM的边界值，如*图2-8. 修改工程设置中的边界值*所示。

图 2-8. 修改工程设置中的边界值



7、在Options for node ‘Project’->Linker->Checksum中，修改End address为Flash大小，如*图2-9. 修改工程属性中的Checksum配置*所示，在Extra Options选项卡中添加“--keep __checksum”。

图 2-9. 修改工程属性中的 Checksum 配置

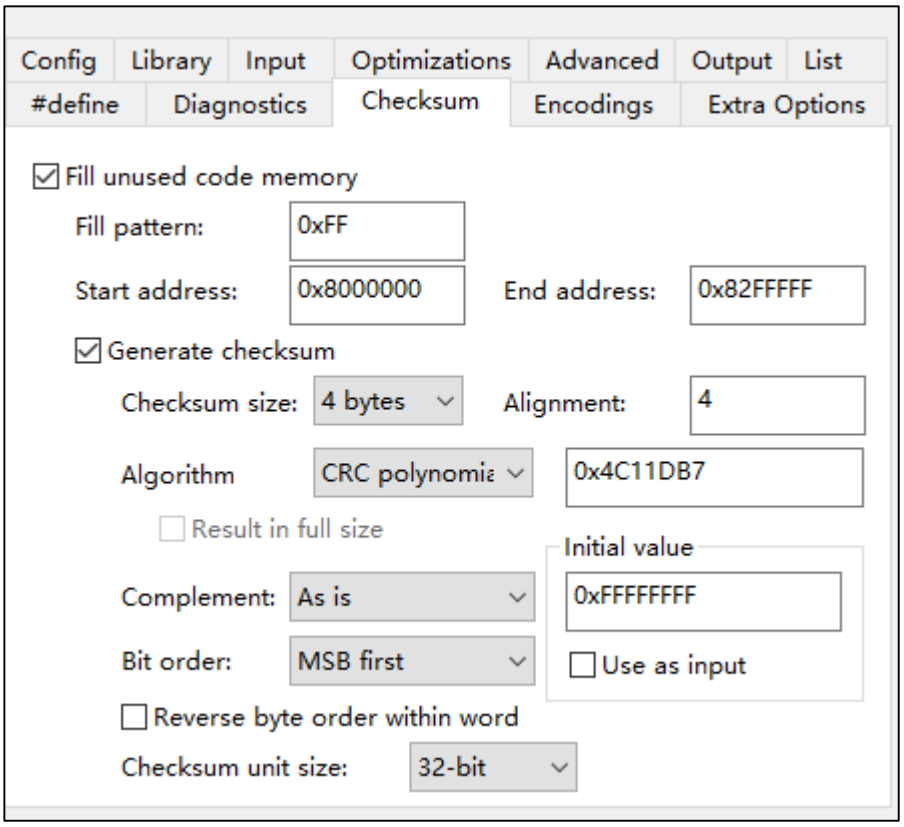
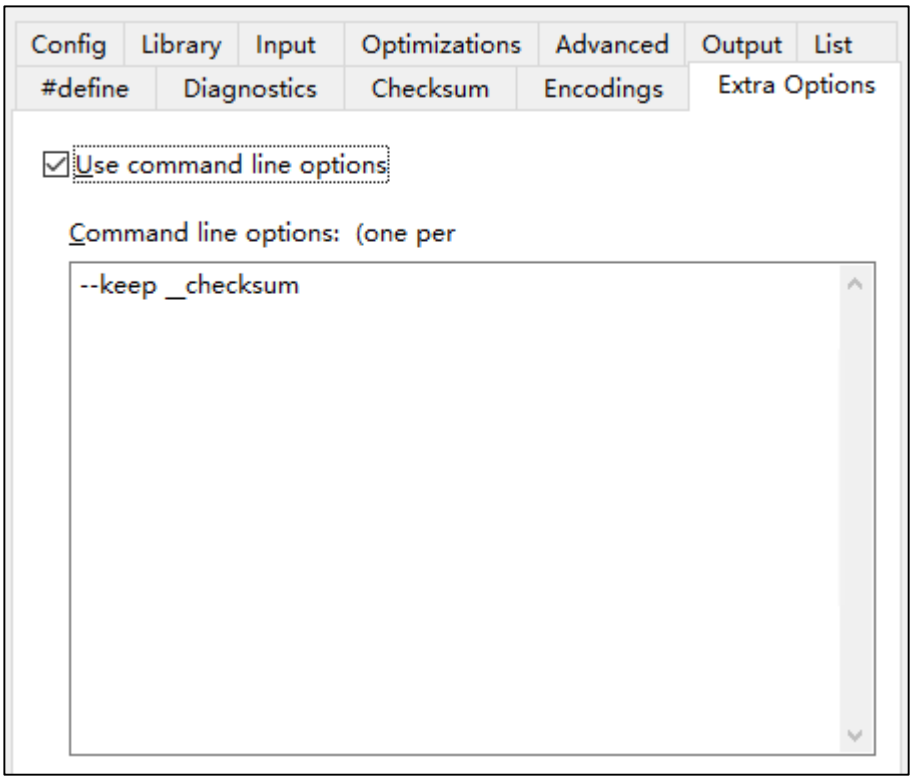


图 2-10. 在 Extra Options 选项卡中添加配置



2.2. Keil 环境认证库工程移植

1、根据当前芯片型号的数据手册修改gd32f4xx_test.h文件中RAM和Flash边界，如*图2-11. 修改gd32f4xx_test.h文件中RAM和Flash边界*所示。

图 2-11. 修改 gd32f4xx_test.h 文件中 RAM 和 Flash 边界

```
#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、检查工程目录下Starup文件夹下的.s启动文件，是否符合当前芯片类型，如*图2-12. .s启动文件检查*所示，同时添加ClassB库文件到工程中。

图 2-12. .s 启动文件检查

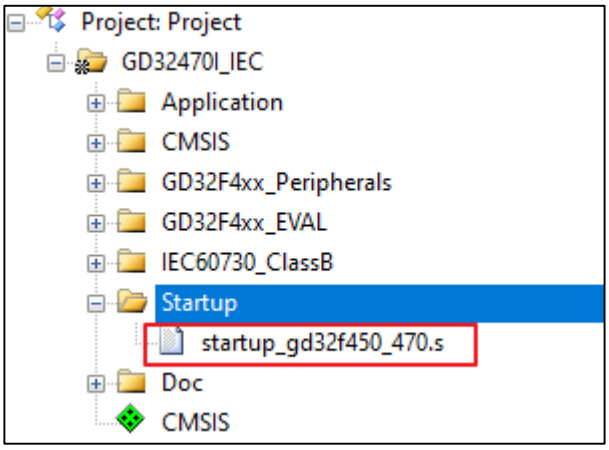
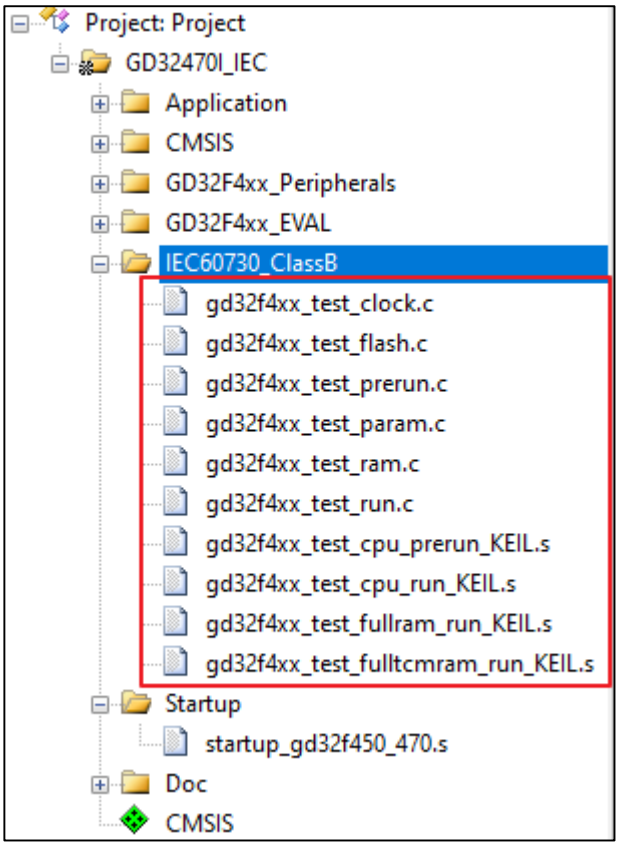


图 2-13. 添加 ClassB 库文件到 Keil 工程



若启动文件适用当前芯片型号，则无需修改；否则需要在固件库文件夹GD32F4xx_Firmware_Library\CMSIS\GD\GD32F4xx\Source\ARM下，重新选择适用当前芯片型号的.s启动文件，并做如下代码中红色标注所示修改：

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

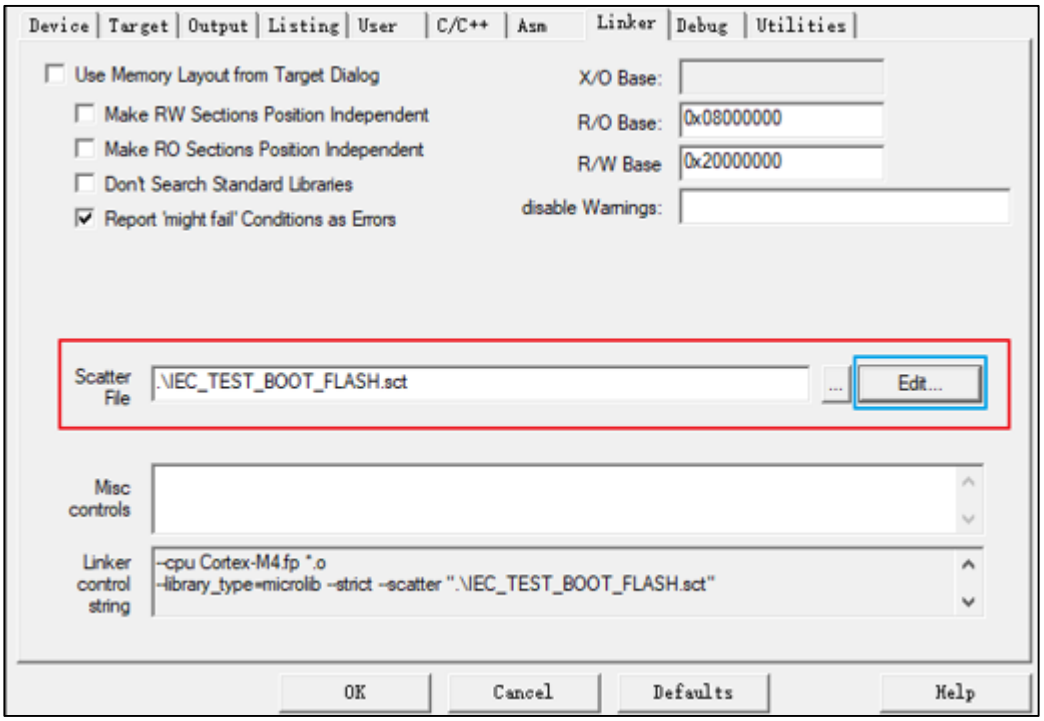
3、修改分散加载文件IEC_TEST_BOOT_FLASH.sct中的代码，如下表红色标注为需要修改的部分（根据当前芯片的数据手册修改Flash边界；修改与.s启动文件相对应.o文件的名称）：

```
; *************************************************************
; *************************************************************
; *** 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)
    }
}
```

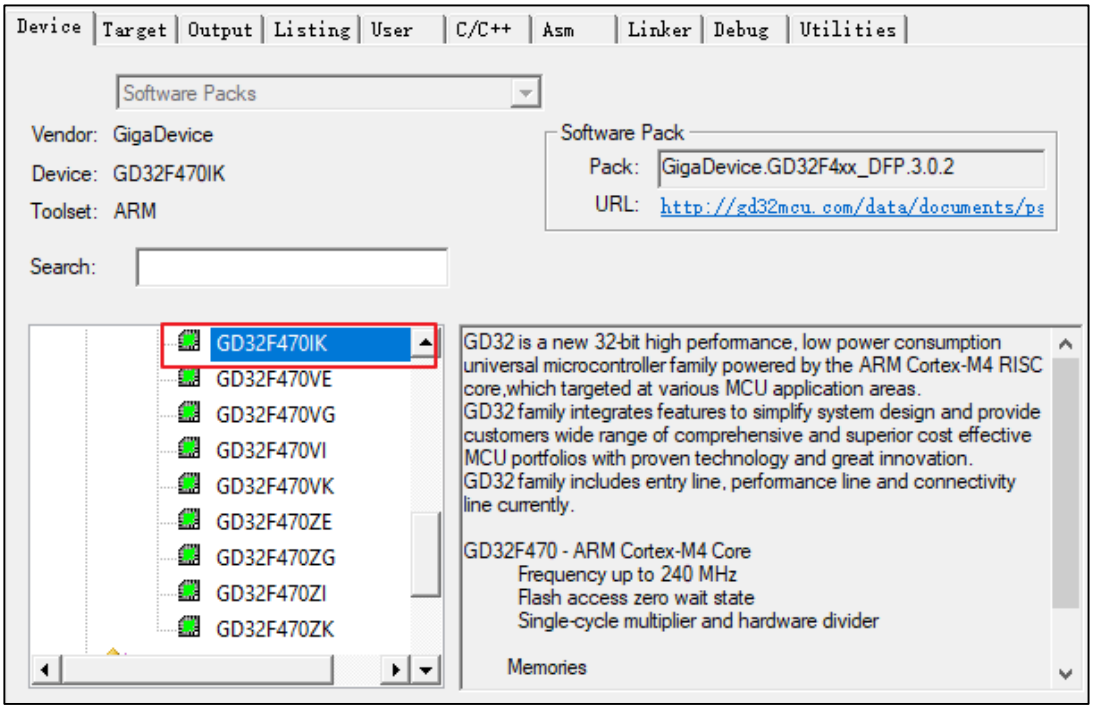
分散加载文件可在“Options for Target -->Linker-->Scatter File”中点击Edit按钮即可修改，如*图 2-14. 编辑分散加载文件*所示：

图 2-14. 编辑分散加载文件



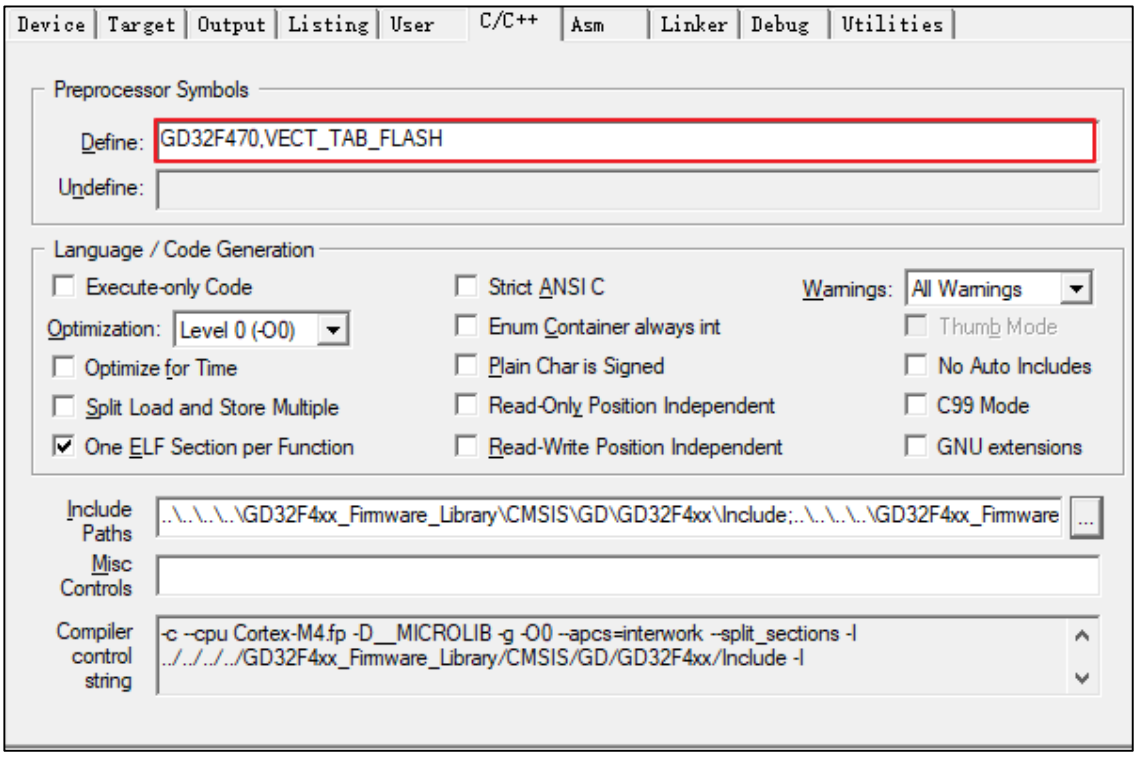
4、修改“Options for Target中DEVICE”的配置，选择当前芯片的型号，如*图2-15. Device配置*所示：

图 2-15. Device 配置



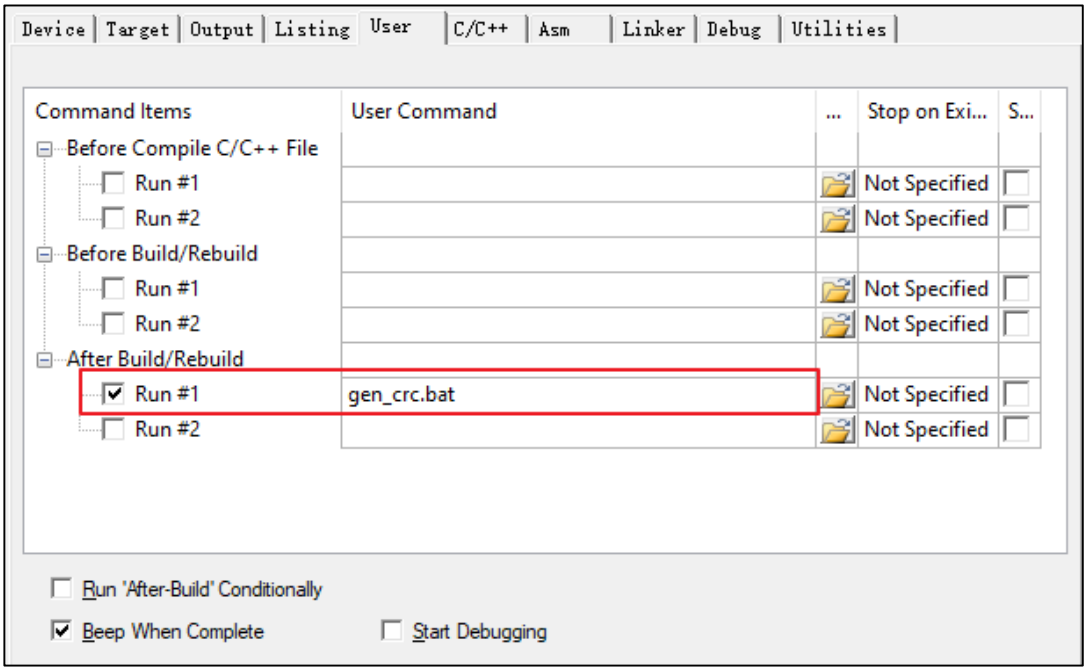
5、在Options for Target->C/C++->Preprocessor Symbols中添加当前工程需要的预编译宏与当前芯片一致，如*图2-16. 修改预编译宏*所示：

图 2-16. 修改预编译宏



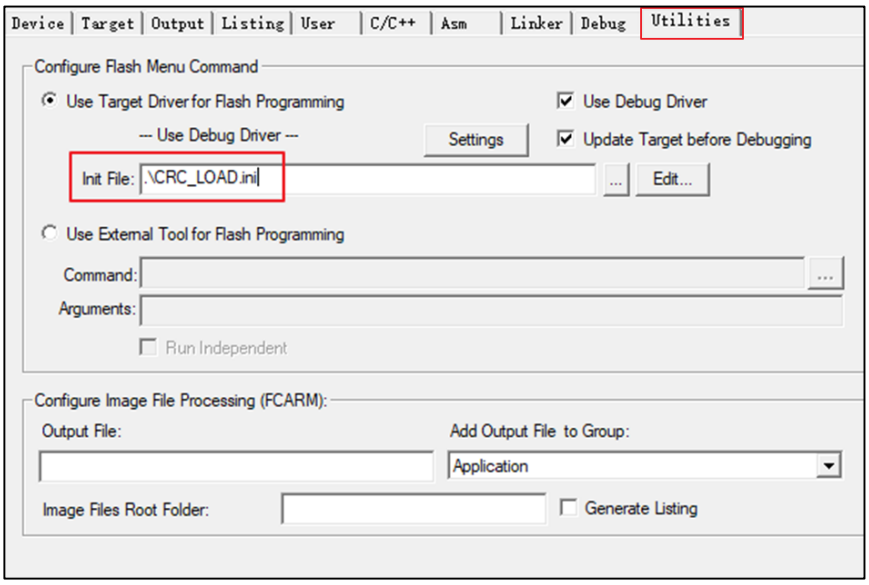
6、当需要使用批处理自动计算CRC时，需要添加gen_crc.bat文件，具体如*图2-17. 添加批处理文件*。

图 2-17. 添加批处理文件



7、最后，在Utilities选项卡中选择CRC_LOAD.ini，具体如*图2-18. 添加.ini配置文件*所示。

图 2-18. 添加.ini 配置文件



2.3. eclipse 环境认证库工程移植

认证库工程在各个环境下的移植步骤类似，但是各开发环境下的编译链有所区别，导致 RAM 和 Flash 边界设置有所不同，eclipse 环境下的移植步骤如下：

1、根据芯片数据手册修改gd32f4xx_test.h文件中编译宏__GNU__下的RAM边界，保证芯片的Flash和RAM全部空间得到检测，如*图2-19. 修改gd32f4xx_test.h中RAM边界*所示。

图 2-19. 修改 gd32f4xx_test.h 中 RAM 边界

```
#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、检查工程中的.S 启动文件是否适用于当前芯片，如*图2-20. 启动文件检查*所示，同时添加 ClassB 库文件到工程中。

图 2-20. 启动文件检查

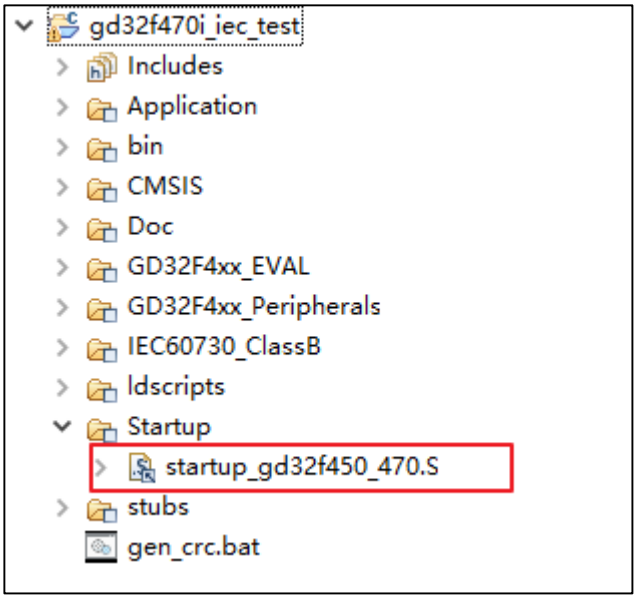
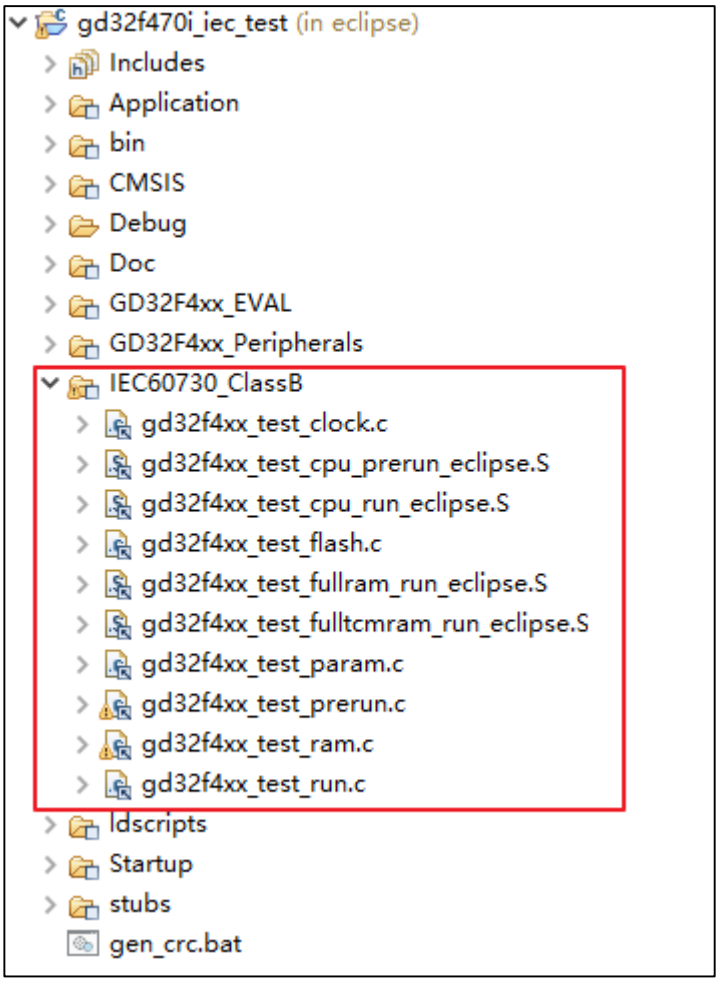


图 2-21. 添加 ClassB 库文件到 eclipse 工程



若启动文件适用当前芯片型号，则无需修改；若不符合，则需重新选择符合当前芯片类型的.s

启动文件，并如*图2-22. 修改启动文件*所示进行修改，使芯片在运行之前进行自检测。

图 2-22. 修改启动文件

```
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、修改如*图2-23. 分散加载文件位置*目录下的分散加载文件，如*图2-24. 修改eclipse工程分散加载文件*所示为修改的部分，根据芯片数据手册修改Flash和RAM的大小适配当前芯片。

图 2-23. 分散加载文件位置

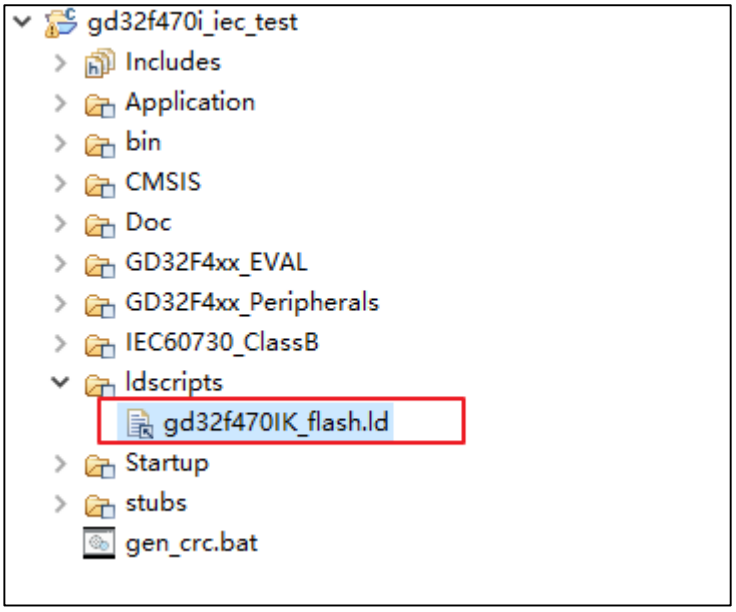
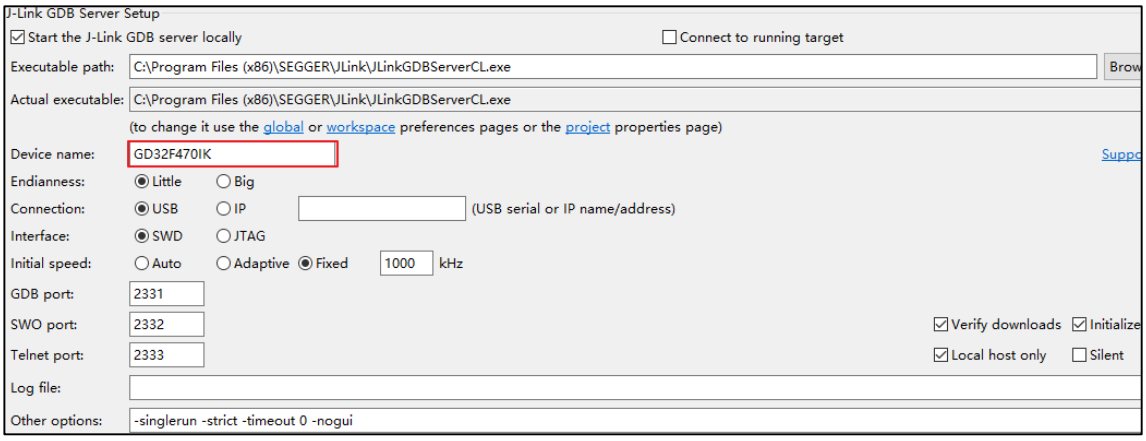


图 2-24. 修改 eclipse 工程分散加载文件

```
1
2 ENTRY(Reset_Handler)
3
4 /* end of Stack */
5 _estack = 0x20030000;
6
7 /* memory map */
8 MEMORY
9 {
10   FLASH (rx)      : ORIGIN = 0x08000000, LENGTH = 3072K   /*3072K*/
11   iec_test (wxa!ri) : ORIGIN = 0x20000000, LENGTH = 0xB0
12   RAM (xrw)       : ORIGIN = 0x200000B0, LENGTH = 0x2FF50 /*256K*/
13   flash_end (rxai!w) : ORIGIN = 0x082FFFC0, LENGTH = 0x40
14 }
```

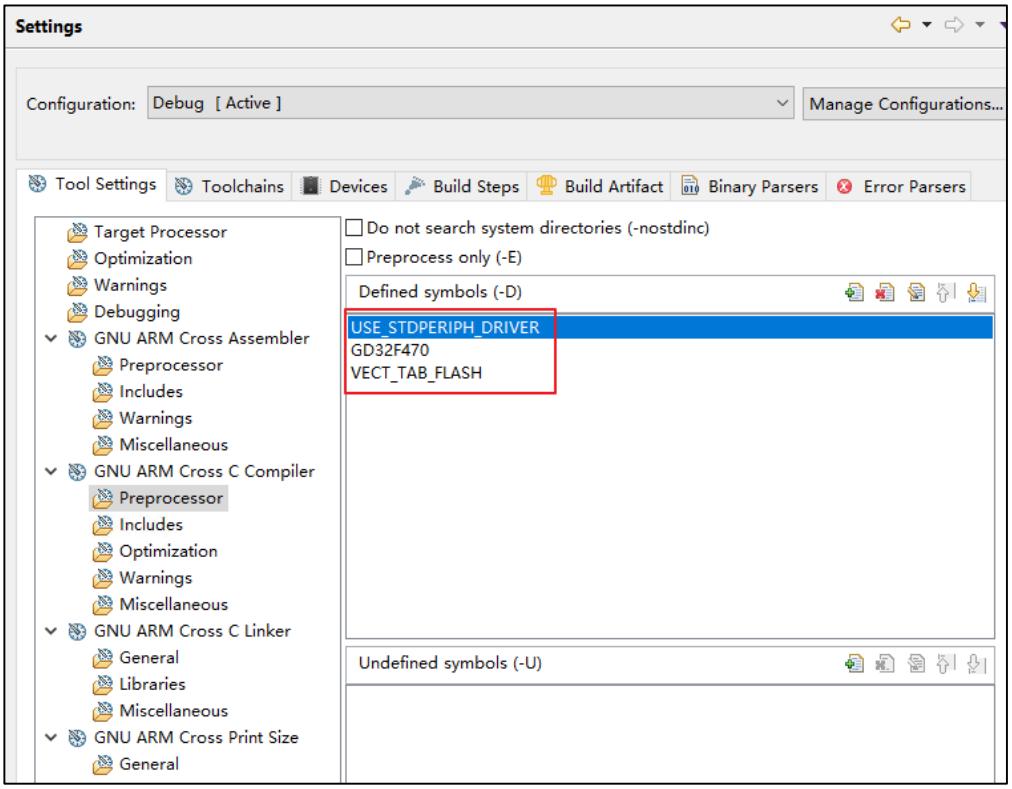
4、修改“Debug Configurations->Debugger”中Device name的配置，选择当前芯片的型号，如*图2-25. Device name配置*所示：

图 2-25. Device name 配置



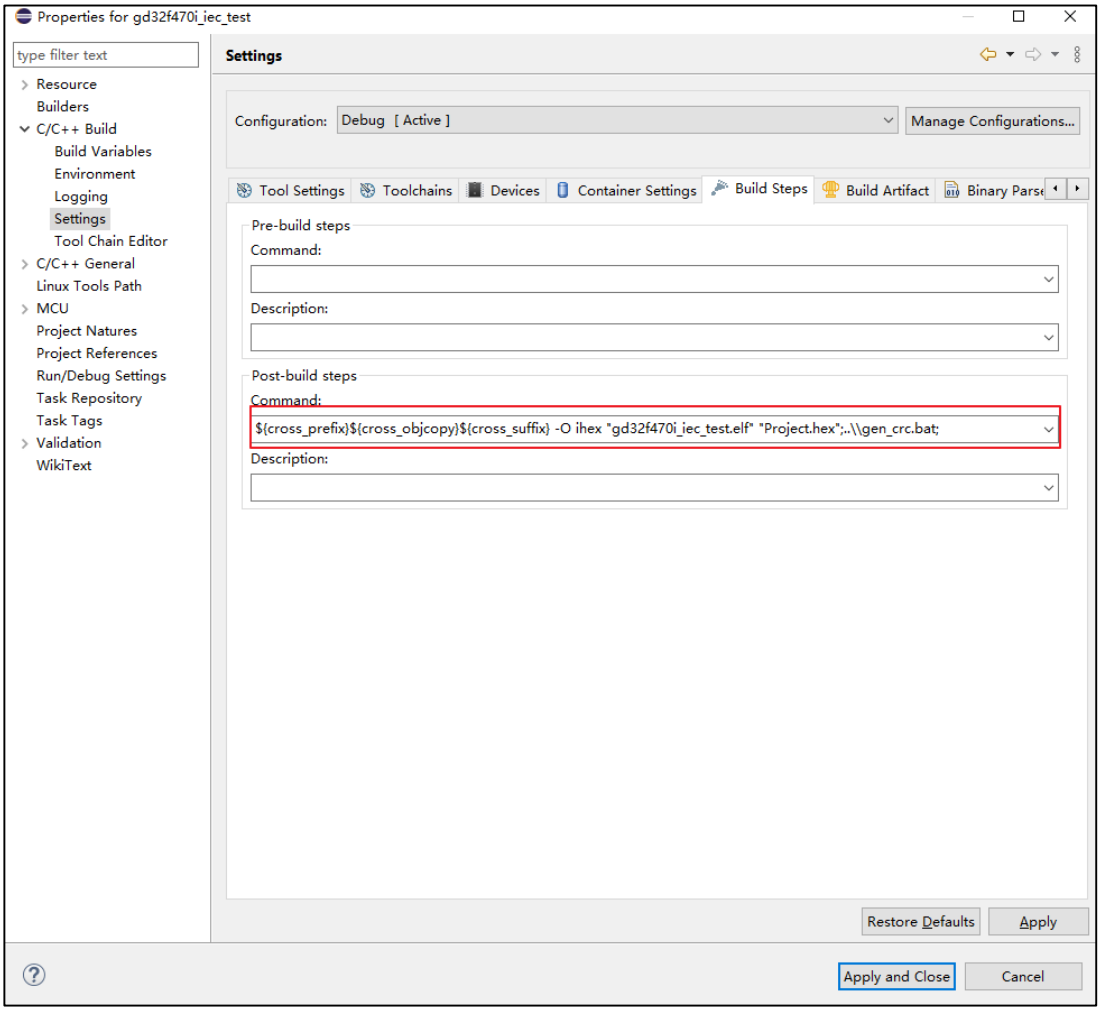
5、在工程属性“C/C++ Build->Settings->Tool Settings->Cross ARM GNU Assembler->Preprocessor”和“C/C++ Build->Settings->Tool Settings->Cross ARM GNU C Complier->Preprocessor”中添加当前工程需要的预编译宏，主要修改符合当前芯片类型的预编译宏（如*图2-26. 修改汇编编译器预编译宏*所示。

图 2-26. 修改汇编编译器预编译宏



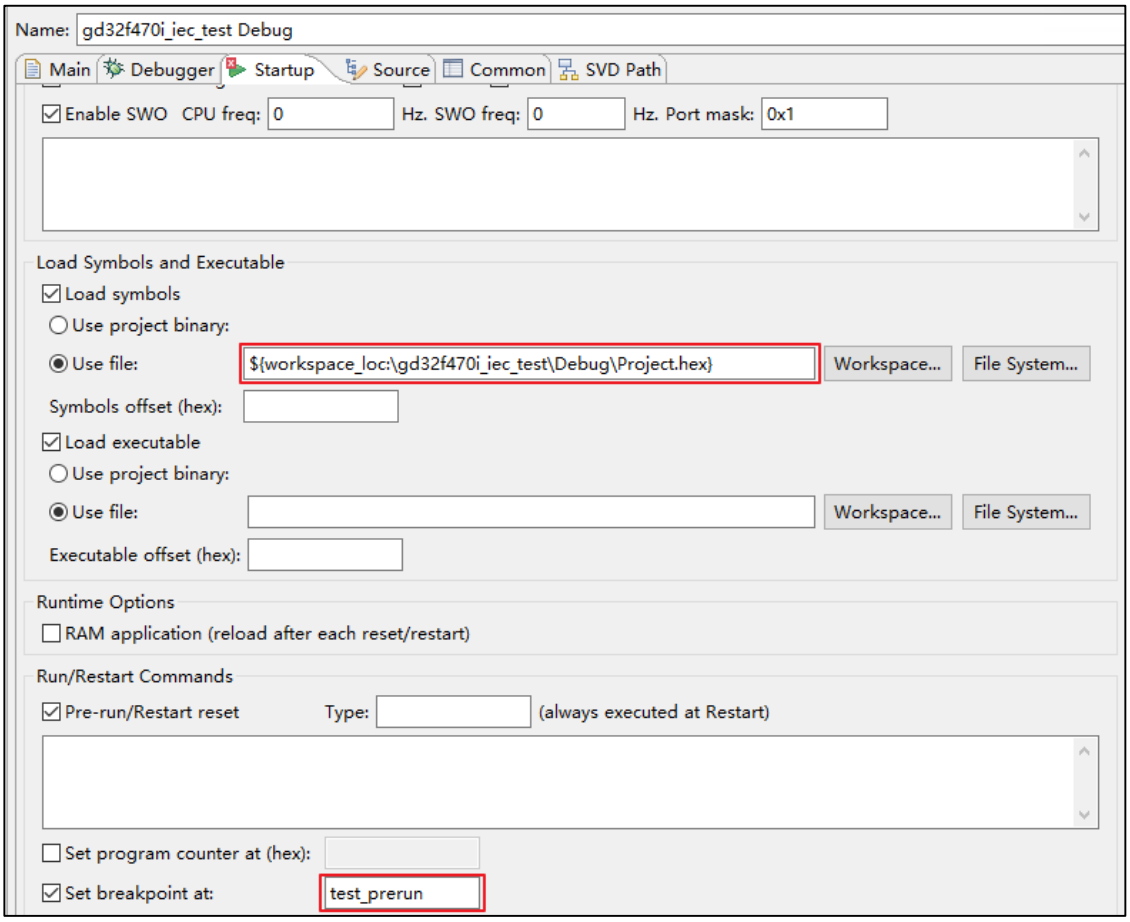
6、修改“C++ Build->Settings->Build Steps->Post-build steps->Command”中的命令，如*图2-27. 修改Post-build指令*所示更换为当前工程名称的elf文件。

图 2-27. 修改 Post-build 指令



7、修改“Debug Configurations->Startup”中的可执行文件的配置，选择当前工作空间中编译生成的Project.hex文件，如*图2-28. 修改可执行文件配置*所示：

图 2-28. 修改可执行文件配置



3. 三种环境下的测试结果

按照第二章的步骤修改代码和配置工程，在各个环境下编译下载工程并运行，测试结果如*图 3-1. 测试结果*所示。

图 3-1. 测试结果

```
>>>>>>>>>>>>>>>  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. 版本历史

表 4-1. 版本历史

版本号	说明	日期
1.0	首次发布	2023 年 08 月 01 日

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.