GigaDevice Semiconductor Inc.

适用于 GD32 Arm Cortex-M 处理器的 IAR 分散加载说明

应用笔记

AN221

1.2 版本

（2026 年 2 月）

目录

表索引

图索引

1. IAR 中分散加载简介

在 IAR 默认配置生成的工程中，IAR 会根据我们在 General option 中所选择的芯片型号，得到芯片 FLASH 和 SRAM 大小等信息，选择相应的*.icf 的分散加载文件(Linker Control File，scatter loading)，链接器根据该文件的配置分配各个节区地址，生成分散加载代码，因此我们可以通过修改该文件来实现指定代码节区在不同位置的存储。

本应用笔记基于 GD32H737_757 系列，采用 GD32H759i-EVAL_V1.1 开发板，IAR 版本为 8.50.9，分别介绍如何实现以下功能：

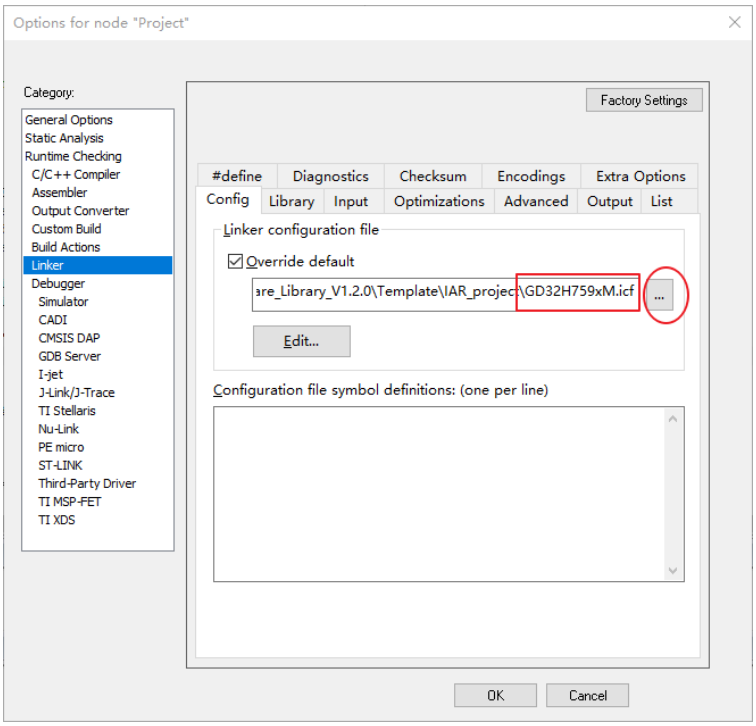
- 实现全局变量加载到指定位置；
- 实现函数加载到指定位置；
- 实现数组加载到指定位置；
- 实现.c 文件加载到指定位置；
- 实现上述功能加载到 SDRAM 指定位置。

2. 分散加载在 IAR 中的实现

2.1. 使用手动编写的 icf 文件

本工程直接使用手动编写的 icf 文件，在 IAR 的“Project->Option->Linker->Config->Linker configuration file”选项勾选 override default，勾选后点击“...”按钮，选择工程目录“GD32H7xx_ScatterLoading_v1.0.0\Project\IAR_project\GD32H759xM.icf”，相关配置如*图 2-1. 使用手动编写的 icf 文件*所示：

图 2-1. 使用手动编写的 icf 文件



打开 GD32H759xM.icf 进行编辑，文件打开代码如*表 2-1. GD32H759xM.icf 代码*：

表 2-1. GD32H759xM.icf 代码

```
/*###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_ROM1_start__ = 0x08000000;
define symbol __ICFEDIT_region_ROM1_end__   = 0x0801FFFF;
define symbol __ICFEDIT_region_ROM2_start__ = 0x08020000;
define symbol __ICFEDIT_region_ROM2_end__   = 0x0802FFFF;
define symbol __ICFEDIT_region_ROM3_start__ = 0x08030000;
```

```
define symbol __ICFEDIT_region_ROM3_end__    = 0x0803FFFF;
define symbol __ICFEDIT_region_ROM4_start__ = 0x08040000;
define symbol __ICFEDIT_region_ROM4_end__    = 0x0804FFFF;
define symbol __ICFEDIT_region_ROM5_start__ = 0x08050000;
define symbol __ICFEDIT_region_ROM5_end__    = 0x0805FFFF;

define symbol __ICFEDIT_region_RAM1_start__ = 0x24000000;
define symbol __ICFEDIT_region_RAM1_end__    = 0x2400FFFF;
define symbol __ICFEDIT_region_RAM2_start__ = 0x24010000;
define symbol __ICFEDIT_region_RAM2_end__    = 0x2401FFFF;
define symbol __ICFEDIT_region_RAM3_start__ = 0x24020000;
define symbol __ICFEDIT_region_RAM3_end__    = 0x2402FFFF;
define symbol __ICFEDIT_region_RAM4_start__ = 0x24030000;
define symbol __ICFEDIT_region_RAM4_end__    = 0x2403FFFF;
define symbol __ICFEDIT_region_RAM5_start__ = 0x24040000;
define symbol __ICFEDIT_region_RAM5_end__    = 0x2404FFFF;
define symbol __ICFEDIT_region_RAM6_start__ = 0x24050000;
define symbol __ICFEDIT_region_RAM6_end__    = 0x2405FFFF;

define symbol __ICFEDIT_region_SDRAM1_start__ = 0xC0000000;
define symbol __ICFEDIT_region_SDRAM1_end__    = 0xC0000FFF;
define symbol __ICFEDIT_region_SDRAM2_start__ = 0xC0001000;
define symbol __ICFEDIT_region_SDRAM2_end__    = 0xC0001FFF;
define symbol __ICFEDIT_region_SDRAM3_start__ = 0xC0002000;
define symbol __ICFEDIT_region_SDRAM3_end__    = 0xC0002FFF;
define symbol __ICFEDIT_region_SDRAM4_start__ = 0xC0003000;
define symbol __ICFEDIT_region_SDRAM4_end__    = 0xC0003FFF;

/*-Sizes-*/
define symbol __ICFEDIT_size_cstack__ = 0x1000;
define symbol __ICFEDIT_size_heap__    = 0x1000;

define memory mem with size = 4G;
define region ROM1_region = mem:[from __ICFEDIT_region_ROM1_start__ to
__ICFEDIT_region_ROM1_end__];
define region ROM2_region = mem:[from __ICFEDIT_region_ROM2_start__ to
__ICFEDIT_region_ROM2_end__];
define region ROM3_region = mem:[from __ICFEDIT_region_ROM3_start__ to
__ICFEDIT_region_ROM3_end__];
define region ROM4_region = mem:[from __ICFEDIT_region_ROM4_start__ to
__ICFEDIT_region_ROM4_end__];
define region ROM5_region = mem:[from __ICFEDIT_region_ROM5_start__ to
__ICFEDIT_region_ROM5_end__];
```

```
define region RAM1_region = mem:[from __ICFEDIT_region_RAM1_start__ to
__ICFEDIT_region_RAM1_end__];
define region RAM2_region = mem:[from __ICFEDIT_region_RAM2_start__ to
__ICFEDIT_region_RAM2_end__];
define region RAM3_region = mem:[from __ICFEDIT_region_RAM3_start__ to
__ICFEDIT_region_RAM3_end__];
define region RAM4_region = mem:[from __ICFEDIT_region_RAM4_start__ to
__ICFEDIT_region_RAM4_end__];
define region RAM5_region = mem:[from __ICFEDIT_region_RAM5_start__ to
__ICFEDIT_region_RAM5_end__];
define region RAM6_region = mem:[from __ICFEDIT_region_RAM6_start__ to
__ICFEDIT_region_RAM6_end__];

define region SDRAM1_region = mem:[from __ICFEDIT_region_SDRAM1_start__ to
__ICFEDIT_region_SDRAM1_end__];
define region SDRAM2_region = mem:[from __ICFEDIT_region_SDRAM2_start__ to
__ICFEDIT_region_SDRAM2_end__];
define region SDRAM3_region = mem:[from __ICFEDIT_region_SDRAM3_start__ to
__ICFEDIT_region_SDRAM3_end__];
define region SDRAM4_region = mem:[from __ICFEDIT_region_SDRAM4_start__ to
__ICFEDIT_region_SDRAM4_end__];

define block CSTACK with alignment = 8, size = __ICFEDIT_size_cstack__ { };
define block HEAP with alignment = 8, size = __ICFEDIT_size_heap__ { };

initialize by copy { readwrite, section SRAM_FUCN, section SDRAM_FUNC};
initialize manually {object hw_config.o};
initialize manually {object test.o};
define block FILE {object hw_config.o};
define block FILE_init {readonly object hw_config.o};
define block SDRAM_FILE {object test.o};
define block SDRAM_FILE_init {readonly object test.o};
do not initialize { section .noinit };

place at address mem:__ICFEDIT_intvec_start__ { readonly section .intvec };

place in ROM1_region { readonly };
place in ROM2_region { section ROM_FUNC };
place in ROM3_region { readonly object gd32h7xx_it.o};
place in ROM5_region { block FILE_init,block SDRAM_FILE_init};
place in RAM1_region { readwrite,
                       block CSTACK, block HEAP };
```

```
place in RAM2_region    { section .bss.RAM_Array};
place in RAM3_region    { section RAM_VARIABLE};
place in RAM4_region    { readwrite section RAM_Array};
place in RAM5_region    { section SRAM_FUCN};
place in RAM6_region    { block FILE};

place in SDRAM1_region    { section SDRAM_FUNC};
place in SDRAM2_region    { section SDRAM_ARRAY};
place in SDRAM3_region    { block SDRAM_FILE};
place in SDRAM4_region    { section SDRAM_VAR};
```

红色部分为实现分散加载主要添加的部分，下面进行详细分析。

2.2. 将全局变量加载到指定位置

本例程中在 main.c 文件中定义全局变量 uint32_t testValue_RAM。通过定义 section RAM_VARIABLE,在 GD32H759xM.icf 文件中添加如下代码,如*表 2-2. GD32H759xM.icf 中将全局变量加载到指定位置代码*所示:

表 2-2. GD32H759xM.icf 中将全局变量加载到指定位置代码

```
define symbol __ICFEDIT_region_RAM3_start__ = 0x24020000;
define symbol __ICFEDIT_region_RAM3_end__   = 0x2402FFFF;
define region RAM3_region = mem:[from __ICFEDIT_region_RAM3_start__ to __ICFEDIT_region_RAM3_end__];
place in RAM3_region    { section RAM_VARIABLE};
```

在 main.c 中定义全局变量 uint32_t testValue_RAM，代码如*表 2-3. Main.c 中将全局变量加载到指定位置代码*所示：

表 2-3. Main.c 中将全局变量加载到指定位置代码

```
/* load the variable testValue_RAM to RAM_VARIABLE section */
int testValue_RAM @"RAM_VARIABLE" = 0xCC;
```

通过 printf 函数打印变量地址，结果如*表 2-4. 将全局变量加载到指定位置打印结果*所示：

表 2-4. 将全局变量加载到指定位置打印结果

```
testValue_RAM address is 0x24020000, value is 0xcc
```

2.3. 将函数加载到指定位置

2.3.1. 将函数加载到 FLASH 指定位置

在 GD32H759xM.icf 文件中加入如下代码，代码如*表 2-5. GD32H759xM.icf 中将函数加载到 FLASH 指定位置代码*所示：

表 2-5. GD32H759xM.icf 中将函数加载到 FLASH 指定位置代码

```
define symbol __ICFEDIT_region_ROM2_start__ = 0x08020000;
define symbol __ICFEDIT_region_ROM2_end__   = 0x0802FFFF;
define region ROM2_region = mem:[from __ICFEDIT_region_ROM2_start__ to __ICFEDIT_region_ROM2_end__];
place in ROM2_region { section ROM_FUNC};
```

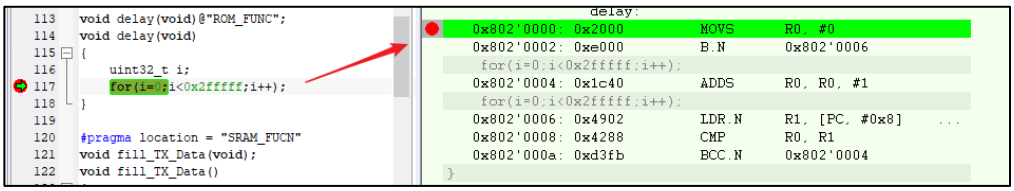
上述代码将通过定义 region，将 section ROM_FUNC 放在 ROM2_region 定义的地址空间，在 main.c 文件中通过加入“@”将 delay 函数 section ROM_FUNC，代码如*表 2-6. Main.c 中将函数加载到指定位置代码*所示：

表 2-6. Main.c 中将函数加载到指定位置代码

```
/*
    \brief        delay program
    \param[in]    none
    \param[out]   none
    \retval       none
*/
void delay(void)@"ROM_FUNC";
void delay(void)
{
    uint32_t I;
    for(i=0;i<0x2fffff;i++);
}
```

程序调试结果如*图 2-2. 函数加载到 FLASH 指定位置程序调试结果*所示：

图 2-2. 函数加载到 FLASH 指定位置程序调试结果



2.3.2. 将函数加载到 SRAM 指定位置

在 GD32H759xM.icf 文件中加入如下代码，代码如*表 2-7. GD32H759xM.icf 中将函数加载到 SRAM 指定位置代码*所示：

表 2-7. GD32H759xM.icf 中将函数加载到 SRAM 指定位置代码

```
define symbol __ICFEDIT_region_RAM5_start__ = 0x24040000;
define symbol __ICFEDIT_region_RAM5_end__ = 0x2404FFFF;
define symbol __ICFEDIT_region_ROM2_end__ = 0x0802FFFF;
define region RAM5_region = mem:[from __ICFEDIT_region_RAM5_start__ to __ICFEDIT_region_RAM5_end__];
initialize by copy { readwrite, section SRAM_FUCN};
```

```
place in RAM5_region    { section SRAM_FUCN};
```

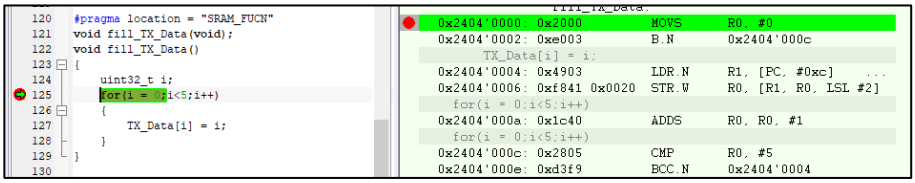
上述代码将通过定义 region，将 section SRAM_FUCN 放在 RAM5_region 定义的地址空间，在 main.c 文件中通过加入“#pragma location =”将 fill_TX_Data 函数加载到 section SRAM_FUCN，代码如*表 2-8. Main.c 中将函数加载到 SRAM 指定位置代码*所示：

表 2-8. Main.c 中将函数加载到 SRAM 指定位置代码

```
#pragma location = “SRAM_FUCN”
void fill_TX_Data(void);
void fill_TX_Data()
{
    uint32_t I;
    for(I = 0;i<5;i++)
    {
        TX_Data[i] = I;
    }
}
```

程序调试结果如*图 2-3. 函数加载到 SRAM 指定位置程序调试结果*所示：

图 2-3. 函数加载到 SRAM 指定位置程序调试结果



2.4. 将数组加载到指定位置

2.4.1. 将全局数组加载到 SRAM 指定位置

通过定义 section RAM_Array,在 GD32H759xM.icf 文件中添加如下代码，代码如*表 2-9. GD32H759xM.icf 中将全局数组加载到 SRAM 指定位置代码*所示：

表 2-9. GD32H759xM.icf 中将全局数组加载到 SRAM 指定位置代码

```
initialize by copy    { readwrite, section SRAM_FUCN, section SDRAM_FUNC};
define symbol __ICFEDIT_region_RAM4_start__ = 0x24030000;
define symbol __ICFEDIT_region_RAM4_end__    = 0x2403FFFF;
define region RAM4_region = mem:[from __ICFEDIT_region_RAM4_start__ to __ICFEDIT_region_RAM4_end__];
place in RAM4_region    { section RAM_Array};
```

在 main.c 中定义数组 test_sram [5]，代码如*表 2-10. Main.c 中将全局数组加载到 SRAM 指定位置*所示：

表 2-10. Main.c 中将全局数组加载到 SRAM 指定位置

```
#pragma location = “RAM_Array”
uint32_t test_sram[5] = {1,2,3,4,5};
```

通过 printf 函数打印变量地址，结果如*表 2-11. 将全局数组加载到 SRAM 指定位置打印结果*所示：

表 2-11. 将全局数组加载到 SRAM 指定位置打印结果

```
test_sram address is 0x24030000
```

2.4.2. 将常量数组加载到 FLASH 指定位置

通过添加“@”操作符直接将数组加载到指定位置，代码如*表 2-12. Constdata 常量数组加载 FLASH 到指定位置代码*所示：

表 2-12. Constdata 常量数组加载 FLASH 到指定位置代码

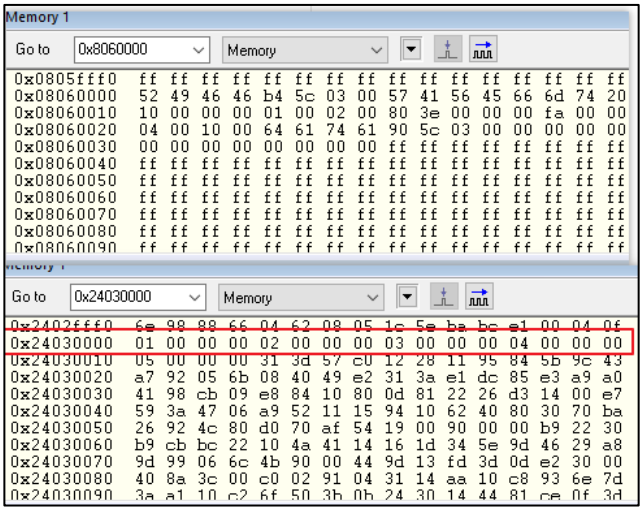
```
const char constdata[]@0x08060000 ={
    0x52,0x49,0x46,0x46,0xB4,0x5C,0x03,0x00,
    0x57,0x41,0x56,0x45,0x66,0x6D,0x74,0x20,
    0x10,0x00,0x00,0x00,0x01,0x00,0x02,0x00,
    0x80,0x3E,0x00,0x00,0x00,0xFA,0x00,0x00,
    0x04,0x00,0x10,0x00,0x64,0x61,0x74,0x61,
    0x90,0x5C,0x03,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
     };
```

通过 printf 函数打印数组地址，结果如*表 2-13. 将常量数组加载到 FLASH 指定位置打印结果*所示：

表 2-13. 将常量数组加载到 FLASH 指定位置打印结果

```
constdata address is 0x8060000
```

图 2-4. 数组加载到指定位置程序调试结果



2.5. 将.c 文件代码段加载到指定位置

2.5.1. 将.c 文件代码段加载到 FLASH 指定位置

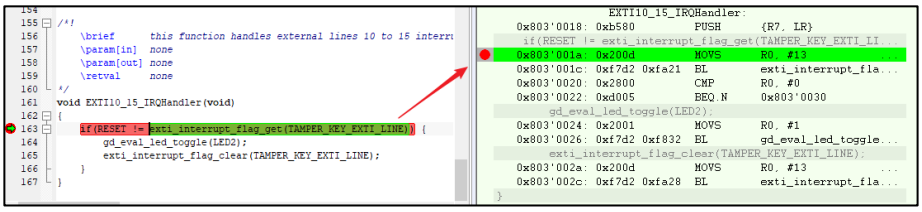
在 GD32H759xM.icf 文件中代码如*表 2-14. GD32H759xM.icf 中.c 文件代码段加载到 FLASH 指定位置代码*所示：

表 2-14. GD32H759xM.icf 中.c 文件代码段加载到 FLASH 指定位置代码

```
define symbol __ICFEDIT_region_ROM3_start__ = 0x08030000;
define symbol __ICFEDIT_region_ROM3_end__ = 0x0803FFFF;
define region ROM3_region = mem:[from __ICFEDIT_region_ROM3_start__ to __ICFEDIT_region_ROM3_end__];
place in ROM3_region { readonly object gd32h7xx_it.o};
```

通过将 gd32h7xx_it.o 文件 ro 段的加载到地址 0x08030000 处，程序调试结果如*图 2-5.*所示：

图 2-5.将.c 文件代码段加载到 FLASH 指定位置



2.5.2. 将.c 文件代码段加载到 SRAM 指定位置

在 GD32H759xM.icf 文件中代码如*表 2-15. GD32H759xM.icf 中.c 文件代码段加载到 SRAM 指定位置代码*所示：

表 2-15. GD32H759xM.icf 中.c 文件代码段加载到 SRAM 指定位置代码

```
define symbol __ICFEDIT_region_ROM5_start__ = 0x08050000;
define symbol __ICFEDIT_region_ROM5_end__ = 0x0805FFFF;
define symbol __ICFEDIT_region_RAM6_start__ = 0x24050000;
define symbol __ICFEDIT_region_RAM6_end__ = 0x2405FFFF;
define region ROM5_region = mem:[from __ICFEDIT_region_ROM5_start__ to __ICFEDIT_region_ROM5_end__];
define region RAM6_region = mem:[from __ICFEDIT_region_RAM6_start__ to __ICFEDIT_region_RAM6_end__];
initialize manually {object hw_config.o};
define block FILE {object hw_config.o};
define block FILE_init {readonly object hw_config.o};
place in ROM5_region { block FILE_init };
place in RAM6_region { block FILE};
```

通过将 hw_config.o 文件 ro 段由定义在 FLASH 上的 ROM5_region，通过手动搬运的方式，加载到地址 RAM6_region 处，在 main.c 中添加搬运代码函数 file_init，在调用 hw_config.c 文件

中的函数前，执行 file_init 函数，完成.c 文件 ro 段加载到 SRAM，如*表 2-16. Main.c 添加搬运代码*所示。程序调试结果如*图 2-6.将.c 文件代码段加载到 SRAM 指定位置*所示：

表 2-16. Main.c 添加搬运代码

```
#pragma section = “FILE_init”
#pragma section = “FILE”
void flie_init(void)
{
    /* Data copy */
    char * from = __section_begin(“FILE_init”);
    char * to = __section_begin(“FILE”);
    memcpy(to, from, __section_size(“FILE”));
}
```

图 2-6.将.c 文件代码段加载到 SRAM 指定位置

3. SDRAM 分散加载实现

3.1. SDRAM 分散加载的实现

在 startup_gd32h7xx.s 中加入下图中红色标注代码，如*图 3-1. SDRAM 分散加载实现中 startup_gd32h7xx.s 添加代码*所示：

图 3-1. SDRAM 分散加载实现中 startup_gd32h7xx.s 添加代码

```
 35
 36          EXTERN  __iar_program_start
 37          EXTERN  SystemInit
 38          EXTERN  DoInit
 39          PUBLIC  __vector_table
 40
286  Reset_Handler
287
288                  LDR     R0, = 0x24000000
289                  ADD     R1, R0, #0x8000
290                  LDR     R2, =0x0
291  MEM_INIT        STRD    R2, R2, [ R0 ] , #8
292                  CMP     R0, R1
293                  BNE     MEM_INIT
294
295                  LDR     R0, =SystemInit
296                  BLX     R0
297                  LDR     R0, =DoInit
298                  BLX     R0
299                  LDR     R0, =__iar_program_start
300                  BX      R0
301          PUBWEAK NMI_Handler
```

其中 DoIint 函数定义在 main.c 中，该函数主要实现 EXMC 初始化和 MPU 的相关配置，以及完成在 SDRAM 上的函数或.c 文件的拷贝，代码如*表 3-1. DoInit 函数代码*所示：

表 3-1. DoInit 函数代码

```
#pragma section = “SDRAM_FILE_init”
#pragma section = “SDRAM_FILE”
/*!
    \brief        initialize the sdram
    \param[in]    none
    \param[out] none
    \retval       none
*/
void DoInit(void)
{
    /* configure the clock of EXMC */
    rcu_exmc_config();

    /* configure the MPU */
    mpu_config();
    /* configure the EXMC access mode */
    exmc_synchronous_dynamic_ram_init(EXMC_SDRAM_DEVICE0);
    __IO int I,j;
```

```
    for(i=0;i<500;i++){
        for(j=0;j<5000;j++);
    }
    /* Data copy */
    char * from = __section_begin("SDRAM_FILE_init");
    char * to = __section_begin("SDRAM_FILE");
    memcpy(to, from, __section_size("SDRAM_FILE"));
}
```

注意：在 M7 内核的默认配置中，部分地址处于禁止执行指令的地址段，因此若当代码加载到该段上，在执行时会发生错误。GD32H7xx 的 EXMC 中 SDRAM 的地址分配为 0xC00000000-0xDFFFFFFF 位于该地址段。通过配置 MPU (Memory Protect Unit) 寄存器，让 0xC0000000 地址段可执行指令。MPU 配置代码如*表 3-2. MPU 配置代码*所示。

表 3-2. MPU 配置代码

```
/*!
    \brief      configure the MPU
    \param[in]  none
    \param[out] none
    \retval     none
*/
void mpu_config(void)
{
    mpu_region_init_struct mpu_init_struct;
    mpu_region_struct_para_init(&mpu_init_struct);

    /* disable the MPU */
    ARM_MPU_Disable();
    ARM_MPU_SetRegion(0, 0);

    /* configure the MPU attributes for SDRAM */
    mpu_init_struct.region_base_address = SDRAM_DEVICE0_ADDR;
    mpu_init_struct.region_size         = MPU_REGION_SIZE_32MB;
    mpu_init_struct.access_permission   = MPU_AP_FULL_ACCESS;
    mpu_init_struct.access_bufferable   = MPU_ACCESS_NON_BUFFERABLE;
    mpu_init_struct.access_cacheable    = MPU_ACCESS_CACHEABLE;
    mpu_init_struct.access_shareable    = MPU_ACCESS_NON_SHAREABLE;
    mpu_init_struct.region_number       = MPU_REGION_NUMBER0;
    mpu_init_struct.subregion_disable   = MPU_SUBREGION_ENABLE;
    mpu_init_struct.instruction_exec    = MPU_INSTRUCTION_EXEC_PERMIT;
    mpu_init_struct.tex_type            = MPU_TEX_TYPE0;
    mpu_region_config(&mpu_init_struct);
    mpu_region_enable();
    /* enable the MPU */
```

```
    ARM_MPU_Enable(MPU_MODE_PRIV_DEFAULT);
}
```

在 GD32H759xM.icf 文件中加入如下代码，代码如*表 3-3. GD32H759xM.icf 中 SDRAM 分散加载代码*所示：

表 3-3. GD32H759xM.icf 中 SDRAM 分散加载代码

```
define symbol __ICFEDIT_region_SDRAM1_start__ = 0xC0000000;
define symbol __ICFEDIT_region_SDRAM1_end__   = 0xC0000FFF;
define symbol __ICFEDIT_region_SDRAM2_start__ = 0xC0001000;
define symbol __ICFEDIT_region_SDRAM2_end__   = 0xC0001FFF;
define symbol __ICFEDIT_region_SDRAM3_start__ = 0xC0002000;
define symbol __ICFEDIT_region_SDRAM3_end__   = 0xC0002FFF;
define symbol __ICFEDIT_region_SDRAM4_start__ = 0xC0003000;
define symbol __ICFEDIT_region_SDRAM4_end__   = 0xC0003FFF;
define region SDRAM1_region = mem:[from __ICFEDIT_region_SDRAM1_start__ to __ICFEDIT_region_SDRAM1_end__];
define region SDRAM2_region = mem:[from __ICFEDIT_region_SDRAM2_start__ to __ICFEDIT_region_SDRAM2_end__];
define region SDRAM3_region = mem:[from __ICFEDIT_region_SDRAM3_start__ to __ICFEDIT_region_SDRAM3_end__];
define region SDRAM4_region = mem:[from __ICFEDIT_region_SDRAM4_start__ to __ICFEDIT_region_SDRAM4_end__];
initialize by copy  { readwrite, section SRAM_FUCN, section SDRAM_FUNC};
initialize manually {object test.o };
define block SDRAM_FILE   {object test.o};
define block SDRAM_FILE_init {readonly object test.o};
place in ROM5_region    { block FILE_init, block SDRAM_FILE_init};
place in SDRAM1_region   { section SDRAM_FUNC};
place in SDRAM2_region   { section SDRAM_ARRAY};
place in SDRAM3_region   { block SDRAM_FILE};
place in SDRAM4_region   { section SDRAM_VAR};
```

上述代码将 SDRAM_ARRAY 段加载到 SDRAM2_region，采用手动拷贝的方式，将 test.o 文件加载到 SDRAM3_region。

在 main.c 中定义全局变量 uint32_t testValue_SDRAM，未初始化的全局数组 uint32_t test_sdram[5],函数 testFuncInSDRAM，以及加入文件 test.c，主要代码如*表 3-4. 将变量、数组、函数和文件分散加载到 SDRAM 指定位置代码*所示：

表 3-4. 将变量、数组、函数和文件分散加载到 SDRAM 指定位置代码

```
uint32_t testValue_SDRAM@"SDRAM_VAR" = 5;
uint32_t test_sdram[5]@"SDRAM_ARRAY";
void testFuncInSDRAM(void)@"SDRAM_FUNC";


void testFuncInSDRAM()
```

```
{
    uint32_t I;
    for(I = 0; I < 0xffff; i++){
    }
}
```

test.c

```
#include “gd32h7xx.h”
#include “test.h”
#include “gd32h759i_eval.h”
#include <stdio.h>

void test_in_sdram()
{
  gd_eval_led_toggle(LED1);
}
```

程序运行和调试结果如*表 3-5. 将变量和数组加载到 SDRAM 指定位置打印结果*和*图 3-2. 将函数和.c 文件加载到 SDRAM 指定位置程序调试结果*所示：

表 3-5. 将变量和数组加载到 SDRAM 指定位置打印结果

```
testValue_SDRAM address is 0xc0003000, value is 0x5
test_sdram address is 0xc0001000
```

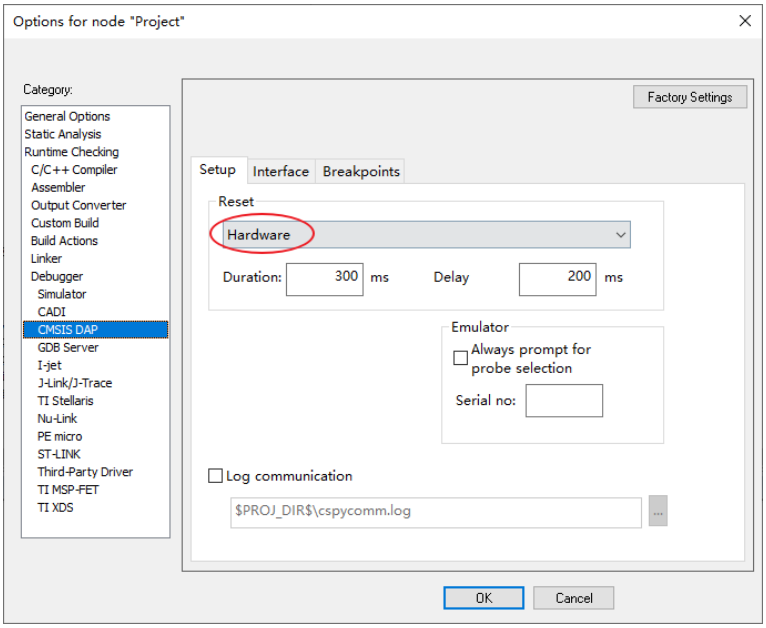
图 3-2. 将函数和.c 文件加载到 SDRAM 指定位置程序调试结果

注意：在使用 CMSIS-DAP 进入调试时，使用 Reset 配置选项选择 Hardware，否则会进入调

试失败。如

*图 3-3. CMSIS-DAP 复位选项配置*所示。

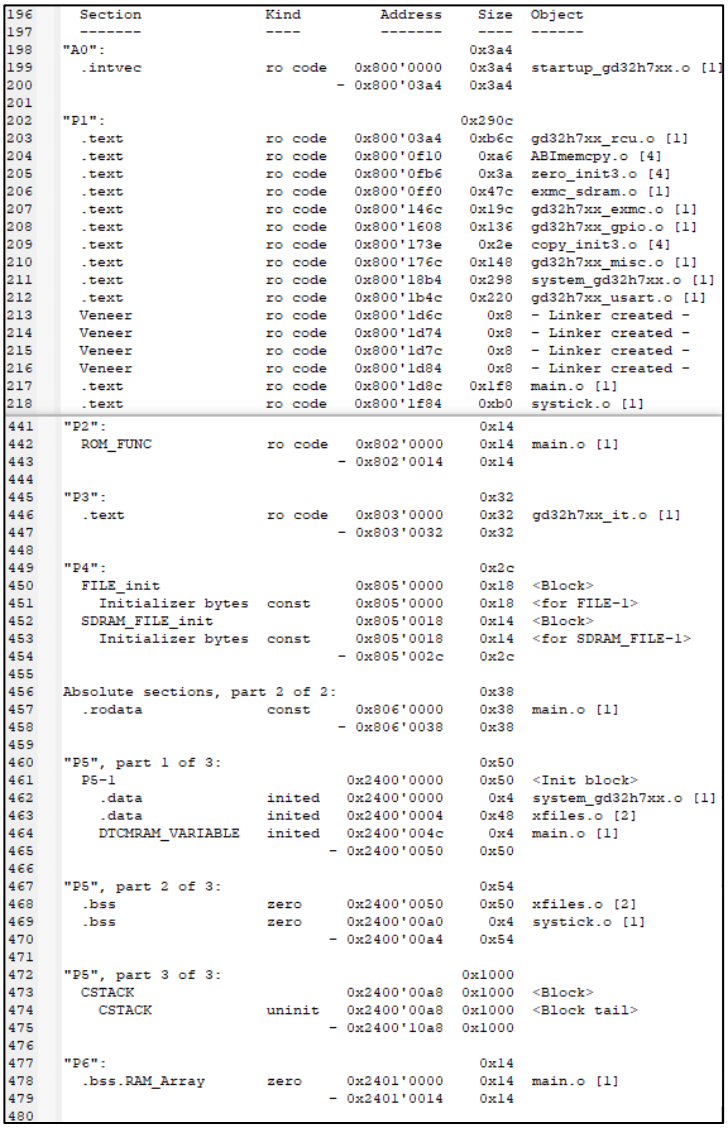
图 3-3. CMSIS-DAP 复位选项配置



4. 结果

查看“GD32H7XX_ScatterLoading_v1.0.0\Project\IAR_project\GD32H7xx\List\Project.map”结果如*图 4-1. 分散加载工程编译 Project.map 文件*所示：

图 4-1. 分散加载工程编译 Project.map 文件



```
196   Section              Kind         Address    Size  Object
197   -------              ----         -------    ----  ------
198 "A0":                                          0x3a4
199   .intvec              ro code   0x800'0000   0x3a4  startup_gd32h7xx.o [1]
200                                - 0x800'03a4   0x3a4
201
202 "P1":                                          0x290c
203   .text                ro code   0x800'03a4   0xb6c  gd32h7xx_rcu.o [1]
204   .text                ro code   0x800'0f10    0xa6  ABImemcpy.o [4]
205   .text                ro code   0x800'0fb6    0x3a  zero_init3.o [4]
206   .text                ro code   0x800'0ff0   0x47c  exmc_sdram.o [1]
207   .text                ro code   0x800'146c   0x19c  gd32h7xx_exmc.o [1]
208   .text                ro code   0x800'1608   0x136  gd32h7xx_gpio.o [1]
209   .text                ro code   0x800'173e    0x2e  copy_init3.o [4]
210   .text                ro code   0x800'176c   0x148  gd32h7xx_misc.o [1]
211   .text                ro code   0x800'18b4   0x298  system_gd32h7xx.o [1]
212   .text                ro code   0x800'1b4c   0x220  gd32h7xx_usart.o [1]
213   Veneer               ro code   0x800'1d6c     0x8  - Linker created -
214   Veneer               ro code   0x800'1d74     0x8  - Linker created -
215   Veneer               ro code   0x800'1d7c     0x8  - Linker created -
216   Veneer               ro code   0x800'1d84     0x8  - Linker created -
217   .text                ro code   0x800'1d8c   0x1f8  main.o [1]
218   .text                ro code   0x800'1f84   0xb0  systick.o [1]
441 "P2":                                          0x14
442   ROM_FUNC             ro code   0x802'0000    0x14  main.o [1]
443                                - 0x802'0014    0x14
444
445 "P3":                                          0x32
446   .text                ro code   0x803'0000    0x32  gd32h7xx_it.o [1]
447                                - 0x803'0032    0x32
448
449 "P4":                                          0x2c
450   FILE_init                      0x805'0000    0x18  <Block>
451     Initializer bytes  const     0x805'0000    0x18  <for FILE-1>
452   SDRAM_FILE_init                0x805'0018    0x14  <Block>
453     Initializer bytes  const     0x805'0018    0x14  <for SDRAM_FILE-1>
454                                - 0x805'002c    0x2c
455
456 Absolute sections, part 2 of 2:                 0x38
457   .rodata              const     0x806'0000    0x38  main.o [1]
458                                - 0x806'0038    0x38
459
460 "P5", part 1 of 3:                              0x50
461   P5-1                           0x2400'0000    0x50  <Init block>
462     .data              inited    0x2400'0000     0x4  system_gd32h7xx.o [1]
463     .data              inited    0x2400'0004    0x48  xfiles.o [2]
464     DTCMRAM_VARIABLE   inited    0x2400'004c     0x4  main.o [1]
465                                - 0x2400'0050    0x50
466
467 "P5", part 2 of 3:                              0x54
468   .bss                 zero      0x2400'0050    0x50  xfiles.o [2]
469   .bss                 zero      0x2400'00a0     0x4  systick.o [1]
470                                - 0x2400'00a4    0x54
471
472 "P5", part 3 of 3:                            0x1000
473   CSTACK                         0x2400'00a8  0x1000  <Block>
474     CSTACK             uninit    0x2400'00a8  0x1000  <Block tail>
475                                - 0x2400'10a8  0x1000
476
477 "P6":                                          0x14
478   .bss.RAM_Array       zero      0x2401'0000    0x14  main.o [1]
479                                - 0x2401'0014    0x14
480
```

```
481  "P7":                                      0x4
482    P7-1                       0x2402'0000    0x4  <Init block>
483      RAM_VARIABLE     inited   0x2402'0000    0x4  main.o [1]
484                             - 0x2402'0004    0x4
485
486  "P8":                                      0x14
487    P8-1                       0x2403'0000   0x14  <Init block>
488      RAM_Array        inited   0x2403'0000   0x14  main.o [1]
489                             - 0x2403'0014   0x14
490
491  "P9":                                      0x18
492    P9-1                       0x2404'0000   0x18  <Init block>
493      SRAM_FUCN        inited   0x2404'0000   0x18  main.o [1]
494                             - 0x2404'0018   0x18
495
496  "P10":                                     0x18
497    FILE                       0x2405'0000   0x18  <Block>
498      FILE-1                   0x2405'0000   0x16  <Init block>
499        Veneer         inited   0x2405'0000    0x8  - Linker created -
500        .text          inited   0x2405'0008    0xe  hw_config.o [1]
501                             - 0x2405'0018   0x18
502
503  "P11":                                     0x10
504    P11-1                      0xc000'0000   0x10  <Init block>
505      SDRAM_FUNC       inited   0xc000'0000   0x10  main.o [1]
506                             - 0xc000'0010   0x10
507
508  "P12":                                     0x14
509    SDRAM_ARRAY        zero     0xc000'1000   0x14  main.o [1]
510                             - 0xc000'1014   0x14
511
512  "P13":                                     0x14
513    SDRAM_FILE                 0xc000'2000   0x14  <Block>
514      SDRAM_FILE-1             0xc000'2000   0x12  <Init block>
515        Veneer         inited   0xc000'2000    0x8  - Linker created -
516        .text          inited   0xc000'2008    0xa  test.o [1]
517                             - 0xc000'2014   0x14
518
519  "P14":                                      0x4
520    P14-1                      0xc000'3000    0x4  <Init block>
521      SDRAM_VAR        inited   0xc000'3000    0x4  main.o [1]
522                             - 0xc000'3004    0x4
```

从 map 文件可以看出各段的加载地址和执行地址，符合指定的分散加载区域。

5. 历史版本

表 5-1. 历史版本

版本号.	描述	日期
1.0	首次发布	2024 年 07 月 20 日
1.1	添加表 3 2. MPU 配置代码	2025 年 05 月 30 日
1.2	更新 AN 名称为“适用于 GD32 Arm Cortex-M 处理器的 IAR 分散加载说明”。	2026 年 02 月 03 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 according to the laws of the People’s Republic of China and other applicable laws. The Company reserves all rights under such laws and no Intellectual Property Rights are transferred (either wholly or partially) or licensed by the Company (either expressly or impliedly) herein. 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.

To the maximum extent permitted by applicable law, the Company makes no representations or warranties of any kind, express or implied, with regard to the merchantability and the fitness for a particular purpose of the Product, nor does the Company assume any liability arising out of the application or use of any Product. Any information provided in this document is provided only for reference purposes. It is the sole responsibility of the user of this document to determine whether the Product is suitable and fit for its applications and products planned, and properly design, program, and test the functionality and safety of its applications and products planned using the Product. The Product is designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and the Product is not designed or intended for use in (i) safety critical applications such as weapons systems, nuclear facilities, atomic energy controller, combustion controller, aeronautic or aerospace applications, traffic signal instruments, pollution control or hazardous substance management; (ii) life-support systems, other medical equipment or systems (including life support equipment and surgical implants); (iii) automotive applications or environments, including but not limited to applications for active and passive safety of automobiles (regardless of front market or aftermarket), for example, EPS, braking, ADAS (camera/fusion), EMS, TCU, BMS, BSG, TPMS, Airbag, Suspension, DMS, ICMS, Domain, ESC, DCDC, e-clutch, advanced-lighting, etc.. Automobile herein means a vehicle propelled by a self-contained motor, engine or the like, such as, without limitation, cars, trucks, motorcycles, electric cars, and other transportation devices; and/or (iv) other uses where the failure of the device or the Product can reasonably be expected to result in personal injury, death, or severe property or environmental damage (collectively "Unintended Uses"). Customers shall take any and all actions to ensure the Product meets the applicable laws and regulations. The Company is not liable for, in whole or in part, and customers shall hereby release the Company as well as its suppliers and/or distributors from, any claim, damage, or other liability arising from or related to all Unintended Uses of the Product. Customers shall indemnify and hold the Company, and its officers, employees, subsidiaries, affiliates as well as its 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 Product.

Information in this document is provided solely in connection with the Product. The Company reserves the right to make changes, corrections, modifications or improvements to this document and the Product described herein at any time without notice. The Company shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. Information in this document supersedes and replaces information previously supplied in any prior versions of this document.