# GigaDevice Semiconductor Inc.

# Arm® Cortex®-M3/4 32-bit MCU

应用笔记

**AN024**

# 目录

# 表索引

# 图索引

# 1. KEIL 中分散加载简介

在 KEIL 默认配置生成的工程中，MDK 会根据我们在 option 选项中所选择的芯片型号，得到芯片的 FLASH 和 RAM 大小等信息，并且会自动生成一个以工程名命名后缀为*.sct 的分散加载文件(Linker Control File，scatter loading)，链接器根据所生成分散加载文件的配置来决定各个节区在存储器上的分配地址。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 GD32F4xx 系列，采用 GD32F450i-EVAL 开发板，keil 版本为 4.74.0.22，编译器版本为 V5.03.0.76，分别介绍如何实现以下功能：

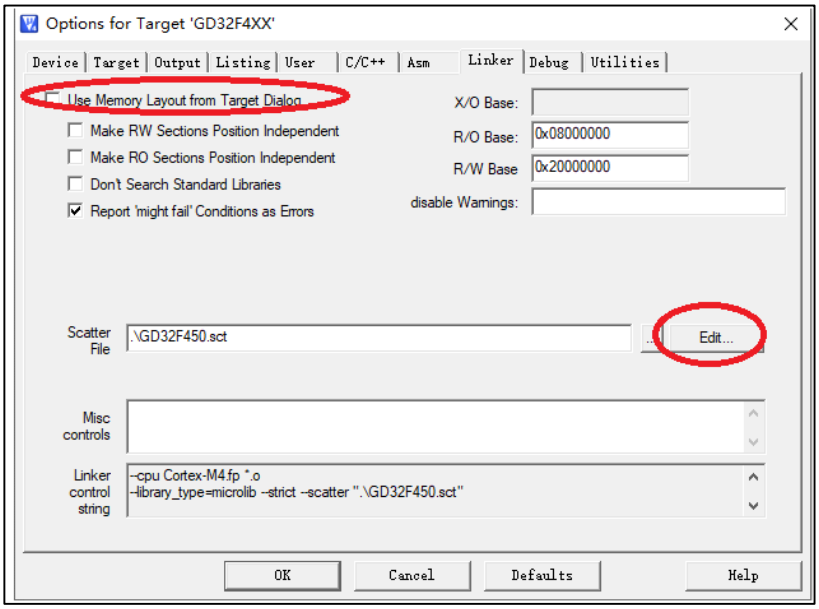
■实现全局变量加载到指定位置
■实现函数加载到指定位置
■实现数组加载到指定位置
■实现.c 文件加载到指定位置
■实现上述功能加载到 SDRAM 指定位置

# 2. 分散加载在 KEIL 中的实现

## 2.1. 使用手动编写的 sct 文件

本工程直接使用手动编写的 sct 文件，在 MDK 的“Options for Target->Linker->Use Memory Layout from Target Dialog”选项取消勾选，取消勾选后可直接点击“Edit”按钮编辑工程的 sct 文件，相关配置如*图 2-1. 使用手动编写的 sct 文件*所示。

**图 2-1. 使用手动编写的 sct 文件**




同时也可到工程目录“GD32F4xx_ScatterLoading_v1.0.0\Project\Keil_project\MDK-ARM\GD32F450.sct”下打开编辑，文件打开代码如*表 2-1. GD32F450.sct 代码*所示：

**表 2-1. GD32F450.sct 代码**

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

LR_IROM1 0x08000000 0x0001ffff  {      ; load region size_region
  ER_IROM1 0x08000000 0x0001ffff  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
  }
  RW_IRAM1 0x20000000 0x00001000  {  ; RW data
   .ANY (+RW +ZI)
  }
  ER_ISDRAM_FUNC 0xc0000000 0x00001000  {
        *(SDRAM_FUNC)
  }
```

```
  ER_ISDRAM_ARRAY 0xc0001000 0x00001000   {
        *(SDRAM_ARRAY)
  }
  ER_ISDRAM_OBJ 0xc0002000 0x00001000   {
        test.o (+RO)
  }
}

/*** Array scatter loading ***/
LR_IROM2 0x08020000 0x0001ffff {
  RW_IRAM_Array 0x20001000 0x00000020 {
   main.o(RAM_Array)
  }
}

/**** File scatter loading ***/
LR_IROM3 0x08040000 0x0001ffff {
  ER_IROM_Object 0x08040000 0x0001ffff {
    gd32f4xx_it.o (+RO)
  }
  RW_IRAM_Object 0x20001100 0x00000100 {
    hw_config.o (+RO)
  }
}
/****   Function scatter loading   ***/
LR_IROM4 0x08060000 0x0001ffff {
  ER_IROM_FUNC 0x08060000 0x0001ffff {
    main.o(ROM_FUNC)
  }
  ER_IRAM_FUNC 0x20001200 0x00000100 {
    main.o(SRAM_FUCN)
  }
}

/****   Variable scatter loading   ***/
LR_IROM5 0x08080000 0x0001ffff {
  ER_IROM_VARIABLE 0x08080000 0x0001ffff {
    main.o(ROM_VARIABLE)
  }
}

LR_IROM6 0x080a0000 0x0025ffff {
  ER_IROM4 0x080a0000 0x0025ffff {
```

```
    .ANY (+RO)
    .ANY (+XO)
  }
}
```

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

方式一：在 GD32F450.sct 文件中加入如下代码，代码如*表 2-2. GD32F450.sct 中将全局变量加载到指定位置代码*所示。

**表 2-2. GD32F450.sct 中将全局变量加载到指定位置代码**

```
/****   Variable scatter loading   ***/
LR_IROM5 0x08080000 0x0001ffff {
  ER_IROM_VARIABLE 0x08080000 0x0001ffff {
    main.o(ROM_VARIABLE)
  }
}
```

上述代码将指定 main.o 模块中的 ROM_VARIABLE 段加载到 0x08080000 起始位置，在 main.c 文件中定义全局变量如*表 2-3. main.c 中将全局变量加载到指定位置代码 1* 所示：

**表 2-3. main.c 中将全局变量加载到指定位置代码 1**

```
/* load the variable testValue_ROM to flash address 0x08080000 */
uint32_t testValue_ROM __attribute__((section("ROM_VARIABLE")))=5;
```

方式二：在全局变量后加入__attribute__((at(xxx)))，本例程在 main.c 中定义变量 uint32_t testValue_RAM，代码如*表 2-4. main.c 中将全局变量加载到指定位置代码 2* 所示：

**表 2-4. main.c 中将全局变量加载到指定位置代码 2**

```
/* load the variable testValue_RAM to ram address 0x20003000 */
uint32_t testValue_RAM __attribute__((at(0x20003000)))=6;
```

通过 printf 函数打印变量地址，结果如*表 2-5. 将全局变量加载到指定位置打印结果*所示：

**表 2-5. 将全局变量加载到指定位置打印结果**

```
variable testValue_ROM address is 0x8080000
variable testValue_RAM address is 0x20003000
```

## 2.3. 将函数加载到指定位置

在 GD32F450.sct 文件中加入如下代码，代码如*表 2-6. GD32F450.sct 中将函数加载到指定位置代码*所示：

**表 2-6. GD32F450.sct 中将函数加载到指定位置代码**

```
/**** Function scatter loading ***/
LR_IROM4 0x08060000 0x0001ffff {
```

```
  ER_IROM_FUNC 0x08060000 0x0001ffff {
    main.o(ROM_FUNC)
  }
  ER_IRAM_FUNC 0x20001200 0x00000100 {
    main.o(SRAM_FUCN)
  }
}
```

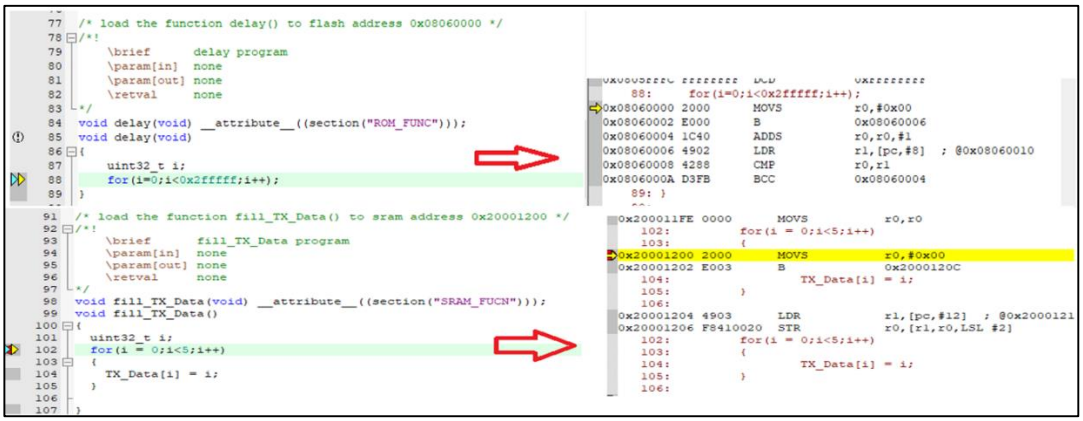
上述代码将指定 main.o 模块中的 ROM_FUNC 段和 SRAM_FUCN 段分别加载到 0x08060000 起始位置和 0x20001200 起始位置。在 main.c 文件中将 delay 函数和 fill_TX_Data 函数分别分配到 ROM_FUNC 和 SRAM_FUCN，代码如*表 2-7. main.c 中将函数加载到指定位置代码*所示：

**表 2-7. main.c 中将函数加载到指定位置代码**

```
/* load the function delay() to flash address 0x08060000*/
/*
    \brief        delay program
    \param[in]    none
    \param[out] none
    \retval       none
*/
void delay(void) __attribute__((section("ROM_FUNC")));
void delay(void)
{
    uint32_t i;
    for(i=0;i<0x2fffff;i++);
}
/* load the function fill_TX_Data() to sram address 0x20001200 */
/*
    \brief        fill_TX_Data program
    \param[in]    none
    \param[out] none
    \retval       none
*/
void fill_TX_Data(void) __attribute__((section("SRAM_FUCN")));
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 2-2. 函数加载到指定位置程序调试结果*所示：

图 2-2. 函数加载到指定位置程序调试结果



## 2.4. 将数组加载到指定位置

方式一：在 GD32F450.sct 文件中加入如下代码，代码如*表 2-8. GD32F450.sct 中将函数加载到指定位置代码*所示：

表 2-8. GD32F450.sct 中将函数加载到指定位置代码

```
/*** Array scatter loading    ***/
LR_IROM2 0x08020000 0x0001ffff {
  RW_IRAM_Array 0x20001000 0x00000020 {
    main.o(RAM_Array)
  }
}
```

上述代码将 main.o 模块中 RAM_Array 段加载到 0x20001000 起始位置，在 main.c 中定义数组 TX_Data[],代码如*表 2-9. main.c 中将数组加载到指定位置代码 1* 所示：

表 2-9. main.c 中将数组加载到指定位置代码 1

```
/* load the array TX_Data[5] to sram address 0x20001000 */
uint32_t TX_Data[5] __attribute__((section(".bss.RAM_Array")))={0};
```

方式二：在数组后加入__attribute__((at(xxx)))，本例程在 main.c 中定义数组 test，在 const-data.c 文件中定义 const char constdata[]代码如*表 2-10. data.c 中将数组加载到指定位置代码*所示，在 main.c 中定义 test_sram[],代码如*表 2-11. main.c 中将数组加载到指定位置代码 2* 所示：

表 2-10. data.c 中将数组加载到指定位置代码

```
/* Load const array constdata to address 0x08001000 */
const char constdata[] __attribute__((at(0x08001000))) ={
    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,
…
}
```

**表 2-11. main.c 中将数组加载到指定位置代码 2**

```
/* load the array test_sram[5] to sram address 0x20007000*/
uint32_t test_sram[5] __attribute__((at(0x20007000)))={1,2,3,4,5};
```

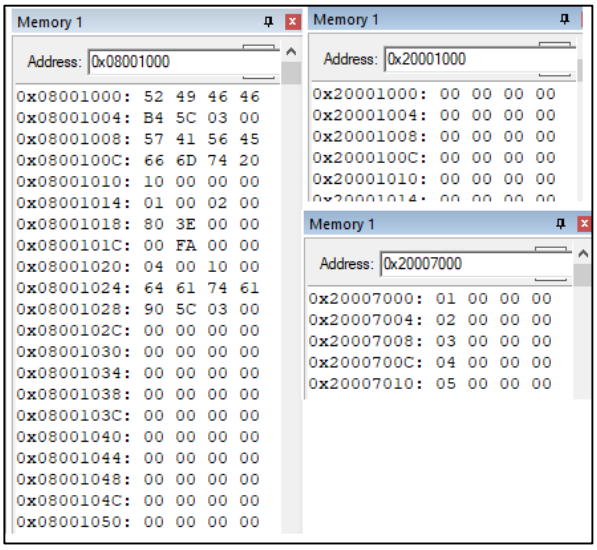
通过 printf 函数打印数组地址，结果如*表 2-12. 将数组加载到指定位置打印结果*所示：

**表 2-12. 将数组加载到指定位置打印结果**

```
constdata address is 0x8001000
TX_Data address is 0x20001000
test_sram address is 0x20007000
```

程序调试结果如*图 2-3. 数组加载到指定位置程序调试结果*所示：

**图 2-3. 数组加载到指定位置程序调试结果**




## 2.5. 将.c 文件加载到指定位置

在 GD32F450.sct 文件中加入如下代码，代码如*表 2-13. GD32F450.sct 中将文件加载到指定位置代码*所示：

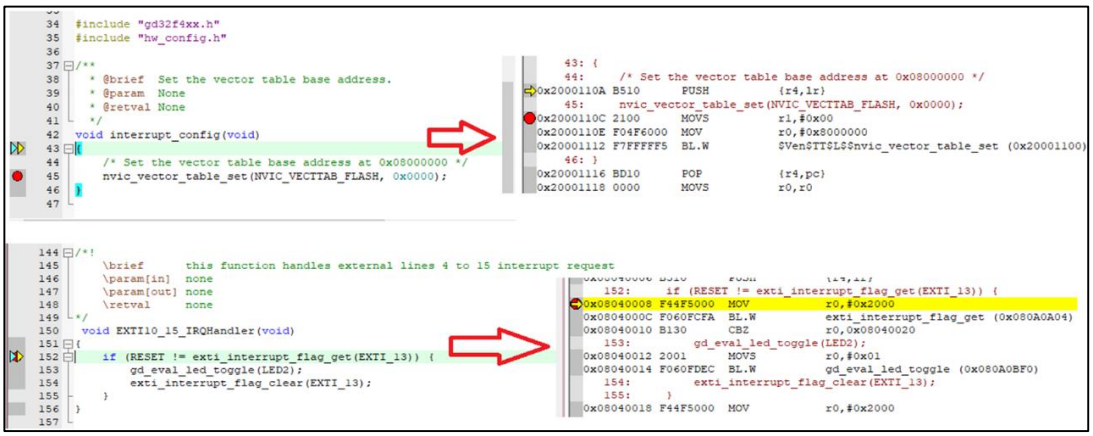
**表 2-13. GD32F450.sct 中将文件加载到指定位置代码**

```
/****   File scatter loading   ***/
LR_IROM3 0x08040000 0x0001ffff {
  ER_IROM_Object   0x08040000 0x0001ffff {
    gd32f4xx_it.o (+RO)
  }
  RW_IRAM_Object 0x20001100 0x00000100 {
```

```
        hw_config.o (+RO)
    }
}
```

上述代码将指定 gd32e230_it.o 文件加载到 0x08040000 起始位置，将 hw_config.o 文件加载到 0x20001100 起始位置，程序调试结果如下：

**图 2-4. 将.c 文件加载到指定位置程序调试结果**



# 3. SDRAM 分散加载实现

## 3.1. 实现 SDRAM 的分散加载的基本原理

在 M4 内核中，我们可以通过系统总线对位于 0x2000 0000 以上的地址进行访问的并对数据和指令进行读取，但是在内核的默认配置中，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。GD32F450 的 EXMC 中 SDRAM 的地址分配为 0xC00000000-0xDFFFFFF 位于该地址段。

针对以上问题，为了在 SDRAM 中实现分散加载，具有两种解决方法：

1. 通过配置 MPU (Memory Protect Unit) 寄存器，让 0xC0000000 地址段可执行指令（本例程将采用这种实现方式）。

2. 采用内存映射的方法（通过配置 SYSCFG 寄存器将 SDRAM 的地址段映射到可执行区）。

## 3.2. SDRAM 分散加载的实现

在 GD32F450.sct 文件中加入如下红色字体代码，代码如*表 3-1. GD32F450.sct 中 SDRAM 分散加载实现代码*所示：

**表 3-1. GD32F450.sct 中 SDRAM 分散加载实现代码**

```
LR_IROM1 0x08000000 0x0001ffff {          ; load region size_region
  ER_IROM1 0x08000000 0x0001ffff {        ; load address = execution address
    *.o (RESET, +First)
    *(InRoot$$Sections)
  }
  RW_IRAM1 0x20000000 0x00001000 {   ; RW data
   .ANY (+RW +ZI)
  }
  ER_ISDRAM_FUNC 0xc0000000 0x00001000 {
    *(SDRAM_FUNC)
  }
  ER_ISDRAM_ARRAY 0xc0001000 0x00001000 {
    *(SDRAM_ARRAY)
  }
  ER_ISDRAM_OBJ 0xc0002000 0x00001000 {
    test.o (+RO)
  }
}
```

上述代码将 SDRAM_FUNC 段、SDRAM_ARRAY 段和 test.o 文件将分别加载到 0xc0001000、0xc0000000 和 0xc0002000 起始地址。

在 startup_gd32f450.s 中加入如下代码，代码如*图 3-1. 在 startup_gd32f450.s 中加入代码*所

示：

**图 3-1. 在 startup_gd32f450.s 中加入代码**

```
;/* reset Handler */
Reset_Handler    PROC
                 EXPORT  Reset_Handler                    [WEAK]
                 IMPORT  SystemInit
                 IMPORT  DoInit
                 IMPORT  __main
                 LDR     R0, =SystemInit
                 BLX     R0
                 LDR     R0, =DoInit
                 BLX     R0
                 LDR     R0, =__main
                 BX      R0
                 ENDP
```

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，函数代码如*表 3-2. DoInit 函数实现代码*所示：

**表 3-2. DoInit 函数实现代码**

```
/*!
    \brief        initialize the sdram, setup the MPU
    \param[in]    none
    \param[out] none
    \retval       none
*/
void DoInit(void)
{
    /* sdram peripheral initialize */
    exmc_synchronous_dynamic_ram_init(EXMC_SDRAM_DEVICE0);
    /* Configures the MPU regions */
    mpu_setup();
}
```

在 main.c 中定义变量 uint32_t testValue_SDRAM，数组 int test_sdram[5], 函数 testFuncInSDRAM，以及加入文件 test.c，主要代码如*表 3-3. 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 3-3. 将变量、数组、函数和文件分散加载到 SDRAM 指定位置代码**

```
/* load the variable testValue_RAM to sdram address 0xC0003000 */
uint32_t testValue_SDRAM __attribute__((at(0xC0003000)));
/* load the array test_sdram[5] to sdram address 0xc0001000 */
uint32_t test_sdram[5] __attribute__((section("SDRAM_ARRAY")))={0};
/* load the function testFuncInSDRAM to sdram address 0xc0000000 */
void testFuncInSDRAM(void) __attribute__((section("SDRAM_FUNC")));
/* test.c */
void test_in_sdram()
{
    gd_eval_led_on(LED3);
```

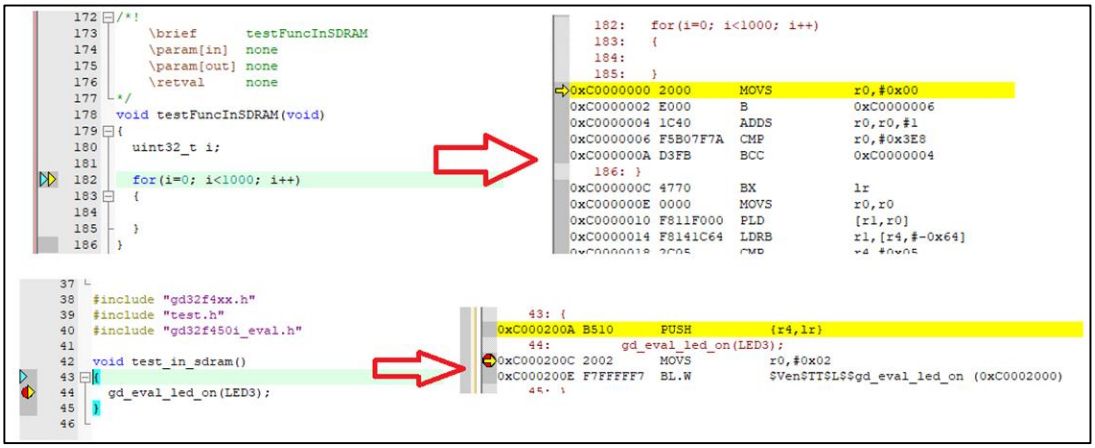
```
}
```

程序运行和调试结果如*表 3-4. 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 3-4. 将变量和数组加载到 SDRAM 指定位置打印结果**

```
variable testValue_SDRAM address is 0xc0003000
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**



# 4. 结果

查看“GD32F4XX_ScatterLoading_v1.0.0\Project\Keil\MDK-ARM\Listings\Project.map”文件，打开如*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**

```
Memory Map of the image
  Image Entry point : 0x08000lad
  Load Region LR_IROM1 (Base: 0x08000000, Size: 0x00009590, Max: 0x0001ffff, ABSOLUTE)
    Execution Region ER_ISDRAM_FUNC (Base: 0xc0000000, Size: 0x00000010, Max: 0x00001000, ABSOLUTE)
    Base Addr    Size         Type   Attr      Idx    E Section Name        Object
    0xc0000000   0x0000000e   Code   RO          343    SDRAM_FUNC          main.o
    Execution Region ER_ISDRAM_ARRAY (Base: 0xc0001000, Size: 0x00000014, Max: 0x00001000, ABSOLUTE)
    Base Addr    Size         Type   Attr      Idx    E Section Name        Object
    0xc0001000   0x00000014   Data   RW          354    SDRAM_ARRAY         main.o
    Execution Region ER_ISDRAM_OBJ (Base: 0xc0002000, Size: 0x00000014, Max: 0x00001000, ABSOLUTE)
    Base Addr    Size         Type   Attr      Idx    E Section Name        Object
    0xc0002000   0x0000000a   Ven    RO         6342    Veneer$$Code        anon$$obj.o
    0xc000200a   0x0000000a   Code   RO          444    i.test_in_sdram     test.o

  Load Region LR_IROM2 (Base: 0x08020000, Size: 0x00000014, Max: 0x0001ffff, ABSOLUTE)
    Execution Region RW_IRAM_Array (Base: 0x20001000, Size: 0x00000014, Max: 0x00000020, ABSOLUTE)
    Base Addr    Size         Type   Attr      Idx    E Section Name        Object
    0x20001000   0x00000014   Data   RW          352    .bss.RAM_Array      main.o

  Load Region LR_IROM3 (Base: 0x08040000, Size: 0x00000050, Max: 0x0001ffff, ABSOLUTE)
    Execution Region ER_IROM_Object (Base: 0x08040000, Size: 0x00000038, Max: 0x0001ffff, ABSOLUTE)
    Base Addr    Size         Type   Attr      Idx    E Section Name        Object
    0x08040000   0x00000004   Code   RO          262    i.BusFault_Handler  gd32f4xx_it.o
    0x08040004   0x00000002   Code   RO          263    i.DebugMon_Handler  gd32f4xx_it.o
    0x08040006   0x0000001c   Code   RO          264    i.EXTI10_15_IRQHandler  gd32f4xx_it.o
    0x08040022   0x00000004   Code   RO          265    i.HardFault_Handler  gd32f4xx_it.o
    0x08040026   0x00000004   Code   RO          266    i.MemManage_Handler  gd32f4xx_it.o
    0x0804002a   0x00000002   Code   RO          267    i.NMI_Handler       gd32f4xx_it.o
    0x0804002c   0x00000002   Code   RO          268    i.PendSV_Handler    gd32f4xx_it.o
    0x0804002e   0x00000002   Code   RO          269    i.SVC_Handler       gd32f4xx_it.o
    0x08040030   0x00000002   Code   RO          270    i.SysTick_Handler   gd32f4xx_it.o
    0x08040032   0x00000004   Code   RO          271    i.UsageFault_Handler  gd32f4xx_it.o
    Execution Region RW_IRAM_Object (Base: 0x20001100, Size: 0x00000018, Max: 0x00000100, ABSOLUTE)
    Base Addr    Size         Type   Attr      Idx    E Section Name        Object
    0x20001100   0x0000000a   Ven    RO         6343    Veneer$$Code        anon$$obj.o
    0x2000110a   0x0000000e   Code   RO          424    i.interrupt_config  hw_config.o

  Load Region LR_IROM4 (Base: 0x08060000, Size: 0x0000002c, Max: 0x0001ffff, ABSOLUTE)
    Execution Region ER_IROM_FUNC (Base: 0x08060000, Size: 0x00000014, Max: 0x0001ffff, ABSOLUTE)
    Base Addr    Size         Type   Attr      Idx    E Section Name        Object
    0x08060000   0x00000014   Code   RO          342    ROM_FUNC            main.o
    Execution Region ER_IRAM_FUNC (Base: 0x20001200, Size: 0x00000018, Max: 0x00000100, ABSOLUTE)
    Base Addr    Size         Type   Attr      Idx    E Section Name        Object
    0x20001200   0x00000018   Code   RO          344    SRAM_FUCN           main.o
  Load Region LR_IROM5 (Base: 0x08080000, Size: 0x00000004, Max: 0x0001ffff, ABSOLUTE)
    Execution Region ER_IROM_VARIABLE (Base: 0x08080000, Size: 0x00000004, Max: 0x0001ffff, ABSOLUTE)
    Base Addr    Size         Type   Attr      Idx    E Section Name        Object
    0x08080000   0x00000004   Data   RW          353    ROM_VARIABLE        main.o
  Load Region LR$$.ARM.__AT_0x20002000 (Base: 0x20002000, Size: 0x00000000, Max: 0x00000020, ABSOLUTE)
    Execution Region ER$$.ARM.__AT_0x20002000 (Base: 0x20002000, Size: 0x00000020, Max: 0x00000020, ABSOLUTE, UNINIT)
    Base Addr    Size         Type   Attr      Idx    E Section Name        Object
   0x20002000   0x00000020   Zero   RW            5    .ARM.__AT_0x20002000  gd32f4xx_mpu.o
  Load Region LR$$.ARM.__AT_0x20003000 (Base: 0x20003000, Size: 0x00000004, Max: 0x00000004, ABSOLUTE)
    Execution Region ER$$.ARM.__AT_0x20003000 (Base: 0x20003000, Size: 0x00000004, Max: 0x00000004, ABSOLUTE, UNINIT)
    Base Addr    Size         Type   Attr      Idx    E Section Name        Object
    0x20003000   0x00000004   Data   RW          349    .ARM.__AT_0x20003000  main.o
  Load Region LR$$.ARM.__AT_0x20007000 (Base: 0x20007000, Size: 0x00000014, Max: 0x00000014, ABSOLUTE)
    Execution Region ER$$.ARM.__AT_0x20007000 (Base: 0x20007000, Size: 0x00000014, Max: 0x00000014, ABSOLUTE, UNINIT)
    Base Addr    Size         Type   Attr      Idx    E Section Name        Object
    0x20007000   0x00000014   Data   RW          350    .ARM.__AT_0x20007000  main.o
  Load Region LR$$.ARM.__AT_0xC0003000 (Base: 0xc0003000, Size: 0x00000004, Max: 0x00000004, ABSOLUTE)
    Execution Region ER$$.ARM.__AT_0xC0003000 (Base: 0xc0003000, Size: 0x00000004, Max: 0x00000004, ABSOLUTE, UNINIT)
    Base Addr    Size         Type   Attr      Idx    E Section Name        Object
    0xc0003000   0x00000004   Data   RW          351    .ARM.__AT_0xC0003000  main.o
```

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 | 2021 年 04 月 30 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.