

GigaDevice Semiconductor Inc.

Arm[®] Cortex[®]-Mx 32-bit MCU

应用笔记

AN040

目录

目录.....	2
图索引.....	3
表索引.....	4
1. 简介.....	5
2. IAP 程序.....	6
2.1. 程序结构.....	6
2.1.1. Bootloader.....	6
2.1.2. APP.....	7
2.2. 工程配置.....	7
2.2.1. Bootloader.....	8
2.2.2. APP.....	10
2.3. 代码解读.....	11
3. 版本历史.....	13

图索引

图 2-1. Bootloader 工程配置.....	8
图 2-2. Bootloader 工程 map 文件.....	9
图 2-3. Bootloader 工程中擦写 APP Flash 指令.....	10
图 2-4. Bootloader 工程中 APP 程序地址宏.....	10
图 2-4. APP 工程配置.....	11

表索引

表 2-1. Bootloader 代码	6
表 2-2. APP 代码	7
表 3-1. 版本历史	13

1. 简介

IAP（在线应用编程）程序通过提前写入用于升级代码的 **bootloader**，可以完成 MCU APP 功能的升级，增强了代码的灵活性，在完成 APP 代码升级之后，程序需要从 **Bootloader** 代码跳转到 APP 代码运行，本应用笔记基于 GD32F10x 系列，介绍如何实现程序从 **Bootloader** 代码跳转到 APP 代码。

2. IAP 程序

IAP 程序通常由两个部分组成: Bootloader 和 APP。Bootloader 和 APP 分别为两个工程程序, 存放在 Flash 的 Main Flash 区, 即 0x08000000 开始的区域。

2.1. 程序结构

2.1.1. Bootloader

Bootloader 代码结构如下表所示。

表 2-1. Bootloader 代码

```
/*!
 *brief      main function
 *param[in]  none
 *param[out] none
 *retval     none
 */
int main(void)
{
    /* init modules ... */
    .....
    /* if no need to update APP */
    if(.....){
        /* Check if valid stack address (RAM address) then jump to user application */
        if (0x20000000 == ((__IO uint32_t*)USER_FLASH_BANK0_FIRST_PAGE_ADDRESS) &
0x2FFE0000)){
            /* disable all interrupts */
            nvic_irq_disable(EXTI0_IRQn);
            ...
            /* Jump to user application */
            JumpAddress = *((__IO uint32_t*) (USER_FLASH_BANK0_FIRST_PAGE_ADDRESS
+ 4);

            Jump_To_Application = (pFunction) JumpAddress;
            /* Initialize user application's Stack Pointer */
            __set_MSP((__IO uint32_t*) USER_FLASH_BANK0_FIRST_PAGE_ADDRESS);
            Jump_To_Application();
        } else {
            /* LED2 ON to indicate bad software (when not valid stack address) */
            gd_eval_led_on(LED2);
            /* do nothing */
            while(1){
```

```

    }
}

/* Bootloader codes for update APP areas */
} else {
    /* Bootloader realizing codes */
    /* including commands of operating flash */
    .....
    while (1){
        /* Bootloader realizing codes */
    }
}
}
}

```

2.1.2. APP

APP 代码结构如下表所示。

表 2-2. APP 代码

```

/*!
 *brief      main function
 *param[in]  none
 *param[out] none
 *retval     none
 */
int main(void)
{
    /* set the NVIC vector table base address to APP code area */
    nvic_vector_table_set(NVIC_VECTTAB_FLASH, APP_OFFSET);
    /* enable global interrupt, the same as __set_PRIMASK(0) */
    __enable_irq();
    /* init modules ... */
    .....

    while (1){
        /* APP realizing codes */
    }
}
}

```

2.2. 工程配置

要完成 APP 的升级，要事先将编写好的 Bootloader 代码下载到 MCU 0x08000000 地址开始的 Flash 中。并且要保证 APP 代码区域不与 Bootloader 代码区域有重叠。以 GD32F107VC

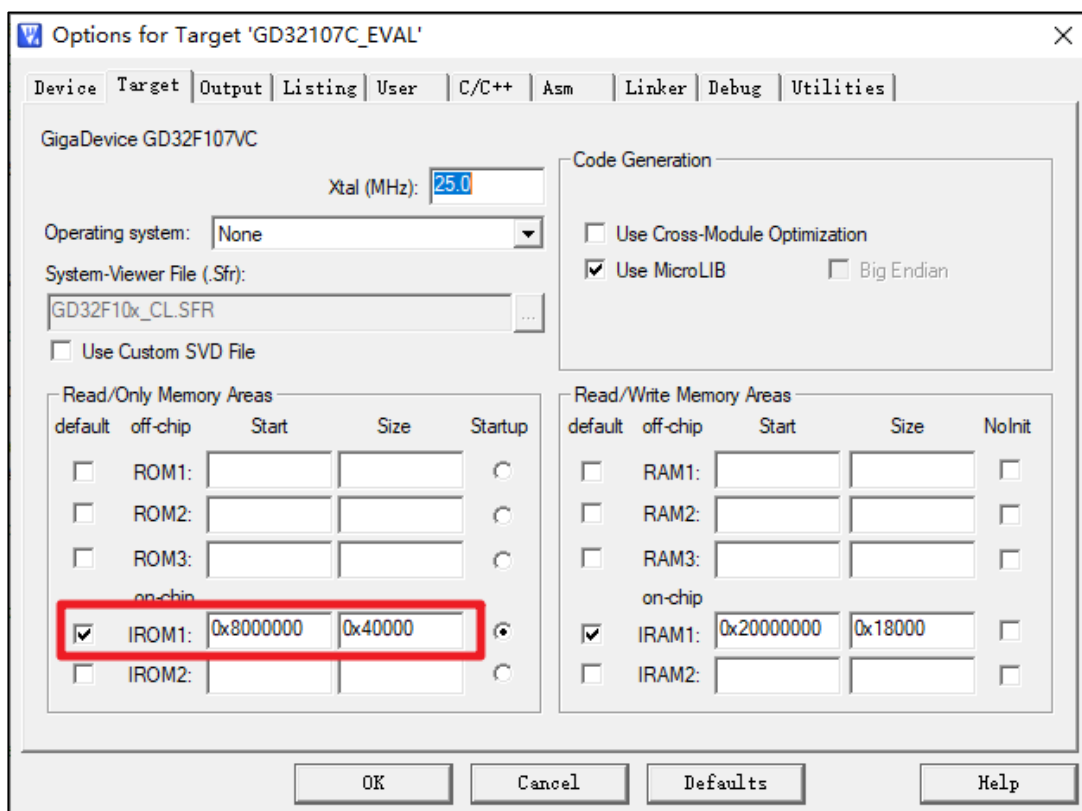
为例，按以下操作进行。

2.2.1. Bootloader

为了确保 APP 代码区域与 Bootloader 代码区域不重叠，Bootloader 工程配置需要按以下步骤进行：

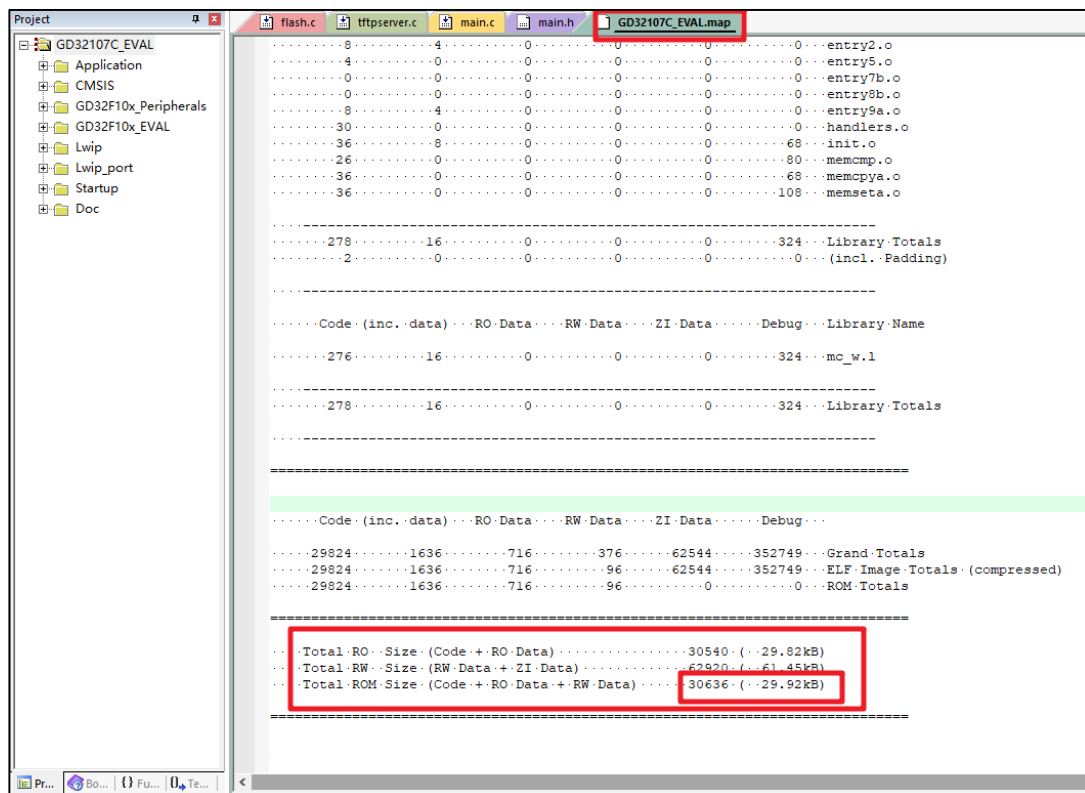
1. 首先查询数据手册，flash 大小为 256KB，工程配置如下图所示，确认是从 0x08000000 开始。

图 2-1. Bootloader 工程配置



2. 查看 Bootloader 工程编译生成的 map 文件，确认代码大小，29.92KB 也就是 0x77AE 字节大小。

图 2-2. Bootloader 工程 map 文件



3. 修改 Bootloader 代码中的擦写 APP Flash 区域相关指令，主要修改要擦写的 Flash 区域的起始地址：USER_FLASH_BANK0_FIRST_PAGE_ADDRESS 宏所对应的地址，改为 0x08010000，表示有 0x100000 字节用于 Bootloader 代码存储，大于 Bootloader 代码的 0x77AE 字节大小。

图 2-3. Bootloader 工程中擦写 APP Flash 指令

```

209 static int IAP_tftp_process_write(struct udp_pcb *upcb, const ip_addr_t *to, int to_port)
210 {
211     tftp_connection_args *args = NULL;
212     /* This function is called from a callback,
213     ** therefore interrupts are disabled,
214     ** therefore we can use regular malloc... */
215     args = mem_malloc(sizeof *args);
216     if (!args) {
217         IAP_tftp_cleanup_wr(upcb, args);
218         return 0;
219     }
220
221     args->op = TFTP_WRQ;
222     args->to_ip.addr = to->addr;
223     args->to_port = to_port;
224     /* the block # used as a positive response to a WRQ is always 0!!! (see RFC1350)... */
225     args->block = 0;
226     args->tot_bytes = 0;
227
228     /* set callback for receives on this UDP PCB (Protocol Control Block) */
229     udp_recv(upcb, IAP_wrq_recv_callback, args);
230
231     total_count = 0;
232
233     /* init flash */
234     FLASH_If_Init();
235
236     /* erase user flash area */
237     FLASH_If_Erase(USER_FLASH_BANK0_FIRST_PAGE_ADDRESS);
238
239     Flash_Write_Address = USER_FLASH_BANK0_FIRST_PAGE_ADDRESS;
240     /* initiate the write transaction by sending the first ack */
241     IAP_tftp_send_ack_packet(upcb, to, to_port, args->block);
242     return 0;
243 }
244

```

图 2-4. Bootloader 工程中 APP 程序地址宏

```

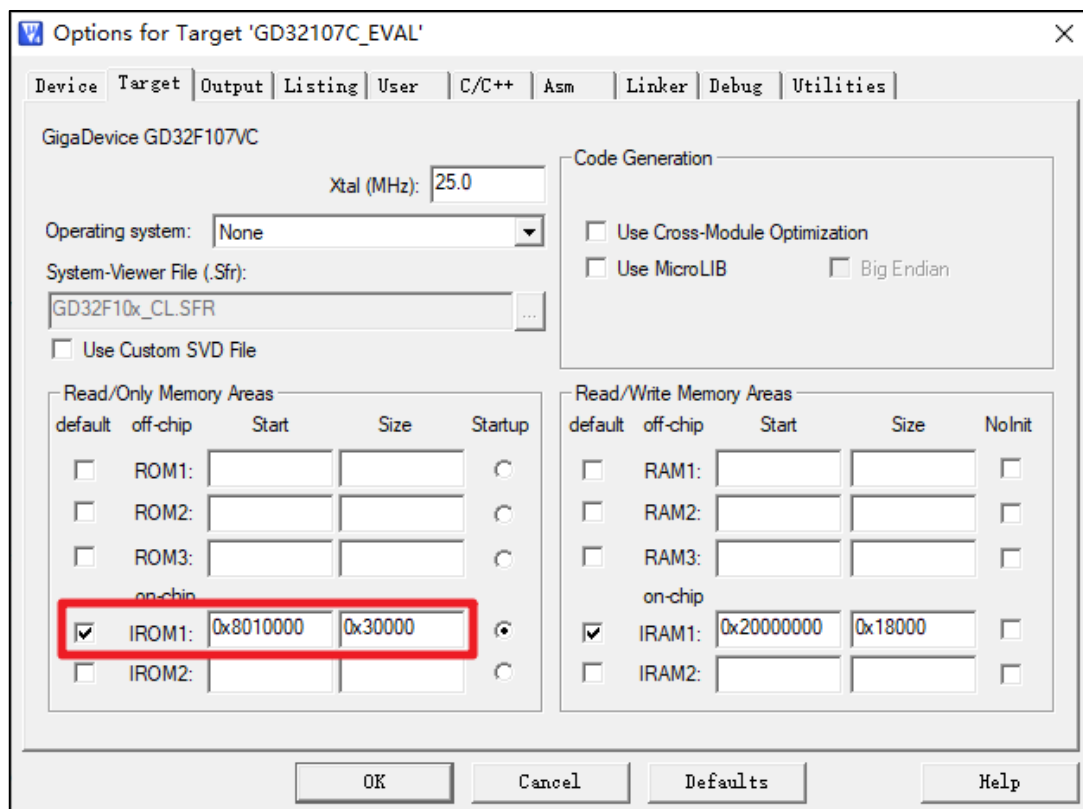
flash.c  tftpserver.c  main.c  main.h  GD32107C_EVAL.map
11  /*
12
13  #ifndef __MAIN_H
14  #define __MAIN_H
15
16  #include "gd32f10x.h"
17  #include "stdint.h"
18  #include "gd32f10x_enet_eval.h"
19
20  /*#define USE_DHCP...*/ enable DHCP, if disabled static address is used */
21
22  /*#define USE_ENET_INTERRUPT
23  /*#define TIMEOUT_CHECK_USE_LWIP
24
25  #define USER_FLASH_BANK0_FIRST_PAGE_ADDRESS 0x08010000 //user flash address start from
26  #define USER_FLASH_BANK0_LAST_PAGE_ADDRESS 0x0807f800 //user flash address start from
27
28  #define USER_FLASH_BANK1_LAST_PAGE_ADDRESS 0x082ff000 //user flash address start from
29  #define USER_FLASH_END_ADDRESS 0x082fffff
30

```

2.2.2. APP

APP 工程配置如下所示。工程 code 起始地址设置为 0x08010000，与 Bootloader 代码中要擦写的地址相同。

图 2-5. APP 工程配置



2.3. 代码解读

Bootloader 代码以及 APP 代码中比较特别的一段代码是 Bootloader 跳转到 APP 的相关代码，在 2.1.1 小节中给出的是适用于 Arm Cortex-M 内核的跳转指令，具体每行按下面来理解。

- ```
if (0x20000000 == ((*(__IO uint32_t*)USER_FLASH_BANK0_FIRST_PAGE_ADDRESS) & 0x2FFE0000))
```

这里 `USER_FLASH_BANK0_FIRST_PAGE_ADDRESS` 宏存储的是 APP 程序起始地址，而 APP 程序起始地址存储的是栈顶指针（查看启动文件向量表的前一个地址），如果下载了 APP 程序的话，则 APP 程序起始地址处必然是写入了栈顶指针，所以可以通过查看栈顶指针值是否位于 SRAM 地址范围来判断是否已经下载了 APP 程序。SRAM 地址范围可以通过查看对应型号 MCU 的 datasheet 得知，例如本例是 96K，即 0x18000 字节，应当查看 SP 是否位于 0x20000000~0x20017FFF，可以检查 SP 的 bit 17-31，即与 0x2FFE0000 相与后判断值，但这个比较并不准确，可以直接采用准确的范围比较方式。如果判断结果为已经下载了 APP 程序，则进行后续的跳转动作。

- ```
nvic_irq_disable(EXTI0_IRQn);
```

在跳转到 APP 程序之前需要关闭所有中断，这么做是为了避免 APP 程序运行出错或卡死。一个原因因为在运行 `Reset_Handler` 函数的 `__main` 时会初始化 APP 应用的 RAM 区数据，如果由于未关闭其他中断而来了一个中断，这个中断此时还是 Bootloader 程序的中断，可能恰好改变了 RAM 区的数据，那么在 APP 程序运行时就会出问题。另一个原因因为在跳到 APP 程序后，由于我们跳转过程中只会对系统时钟进行重新配置，而不会影响到其他模块的寄存器，因此其他已配置的寄存器信息

将保持 Bootloader 时的配置，如果并未初始化所有模块，而在跳转之前又没有关闭所有中断，那么在 Bootloader 程序中运行的一些模块可能在 APP 程序中依然在运行，并自动触发中断，而 APP 程序中如果没有对相应中断服务函数的清标志处理，则 APP 程序可能会陷入中断死循环而无法正常运行，因此需要关闭所有中断。

- `JumpAddress = *(__IO uint32_t*) (USER_FLASH_BANK0_FIRST_PAGE_ADDRESS + 4);`

`Jump_To_Application = (pFunction) JumpAddress;`

`USER_FLASH_BANK0_FIRST_PAGE_ADDRESS + 4` 地址处存储的是 `Reset_Handler` 向量，该向量为 `Reset_Handler` 处理函数的入口地址，由于已经将 `pFunction` 自定义为 `void` 类型的函数指针，所以下一句将 `Jump_To_Application` 指针指向 `Reset_Handler` 函数的入口地址。

- `__set_MSP(*(__IO uint32_t*) USER_FLASH_BANK0_FIRST_PAGE_ADDRESS);`

执行 APP 程序的第一条指令，即将主堆栈指针设置为 APP 程序起始地址 `USER_FLASH_BANK0_FIRST_PAGE_ADDRESS`，需要在真正运行 APP 程序之前就准备好 MSP，因为可能 `Reset_Handler` 的第一条指令还没来得及执行，就发生了 NMI 或者其他 `fault`，此时就需要 MSP 来提供堆栈。

- `Jump_To_Application();`

执行 `Jump_To_Application` 指针指向的函数，即 `Reset_Handler` 函数，在 `Reset_Handler` 函数中执行完 `__main` 函数后，将自动跳转到 `main()` 函数，也就是 APP 主程序部分。

3. 版本历史

表 3-1. 版本历史

版本号.	说明	日期
1.0	首次发布	2021 年 11 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 its 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 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 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.