GigaDevice Semiconductor Inc.

Arm® Cortex®-M3/M4/M23/M33 32-bit MCU

应用笔记

AN035

目录

图索引

表索引

1. 简介

GD32F10x系列微控制器提供了三种引导源，可以通过BOOT0和BOOT1引脚来进行选择，详细说明见*表1-1. 引导模式*。BOOT0和BOOT1引脚的电平状态会在复位后的第四个CK_SYS（系统时钟）的上升沿进行锁存。用户可自行选择所需要的引导源，通过设置上电复位和系统复位后的BOOT0和BOOT1的引脚电平。一旦这两个引脚电平被采样，它们可以被释放并用于其他用途。

表 1-1. 引导模式

引导源选择	启动模式选择引脚	
	BOOT1	BOOT0
主FLASH存储器	x	0
引导装载程序	0	1
片上SRAM	1	1

上电序列或系统复位后，Arm® Cortex®-M3/M4/M23/M33处理器先从0x0000 0000地址获取栈顶值，再从0x0000 0004地址获得引导代码的基地址，然后从引导代码的基地址开始执行程序。

根据所选择的引导源，主FLASH存储器（开始于0x0800 0000的原始存储空间）或系统存储器（开始于0x1FFF F000的原始存储空间）被映射到引导存储空间（起始于0x0000 0000）。片上SRAM存储空间的起始地址是0x2000 0000，当它被选择为引导源时，在应用初始化代码中，你必须使用NVIC异常表和偏移寄存器来将向量表重定向到SRAM中。

嵌入式的Bootloader存放在系统存储空间，用于对FLASH存储器进行重新编程。

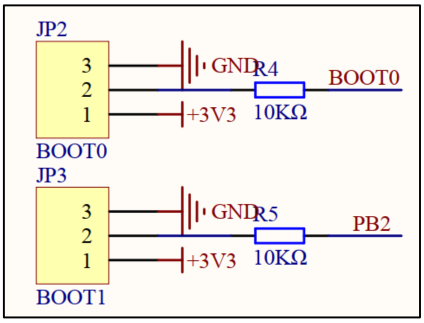
本文介绍了如何在 Keil 中实现 MCU 从 SRAM 中启动程序的方法。

2. SRAM 中启动程序

2.1. 硬件配置

选择从 SRAM 启动程序，BOOT0 和 BOOT1 都必须配置为高电平。如*表 1-1. 引导模式*所示。在设计电路时，通常采用一个跳线帽来切换 BOOT 引脚的高低电平，如*图 2-1. BOOT 引脚原理图*所示。

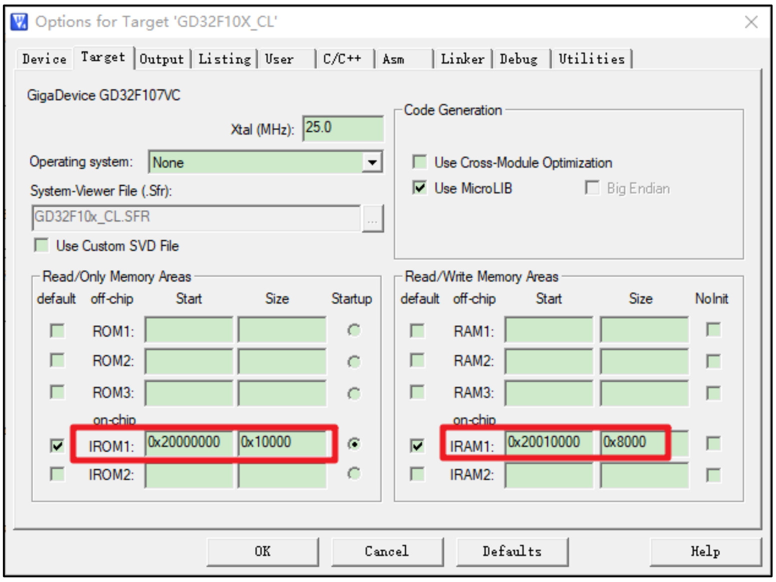
图 2-1. BOOT 引脚原理图



2.2. Keil 配置步骤

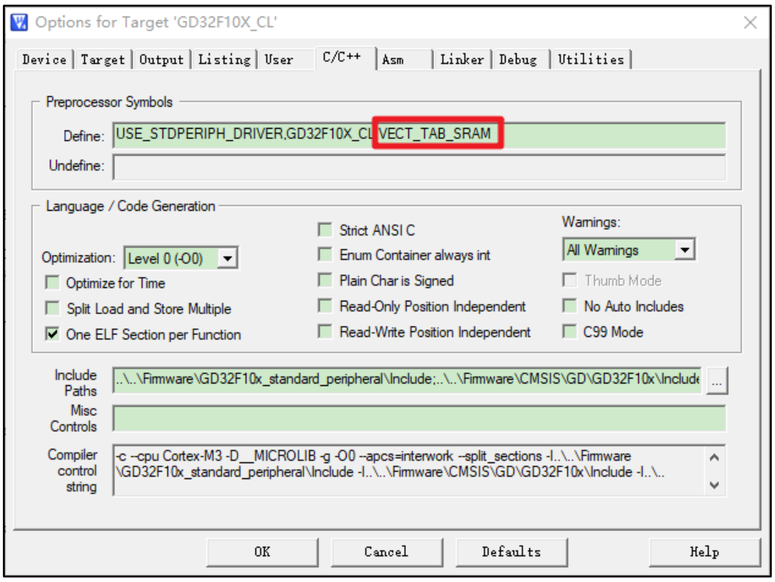
1. 在“Option for Target -> Target”中将 IROM1，IRAM1 修改为 SRAM 地址。如*图 2-2. IROM1 和 IRAM1 地址配置*所示。

图 2-2. IROM1 和 IRAM1 地址配置



2. 使用 NVIC 异常表和偏移寄存器来将向量表重定向到 SRAM 中。在“Option for Target -> c/c++ -> Define”中添加全局宏“VECT_TAB_SRAM”，如*图 2-3. 添加全局宏“VECT_TAB_SRAM”*所示。

图 2-3. 添加全局宏“VECT_TAB_SRAM”



在 SystemInit() 函数中添加宏“VECT_TAB_SRAM”相关代码。如[表 2-1. 宏“VECT_TAB_SRAM”相关代码添加](#)所示。

表 2-1. 宏“VECT_TAB_SRAM”相关代码添加

```
/*!
    \brief      setup the microcontroller system, initialize the system
    \param[in]  none
    \param[out] none
    \retval     none
*/
void SystemInit(void)
{
    /* reset the RCU clock configuration to the default reset state */
    /* enable IRC8M */
    RCU_CTL |= RCU_CTL_IRC8MEN;
    /* reset SCS, AHBPSC, APB1PSC, APB2PSC, ADCPSC, CKOUT0SEL bits */
    RCU_CFG0 &= ~(RCU_CFG0_SCS | RCU_CFG0_AHBPSC | RCU_CFG0_APB1PSC | RCU_CFG0_APB2PSC |
                  RCU_CFG0_ADCPSC | RCU_CFG0_ADCPSC_2 | RCU_CFG0_CKOUT0SEL);
    /* reset HXTALEN, CKMEN, PLLEN bits */
    RCU_CTL &= ~(RCU_CTL_HXTALEN | RCU_CTL_CKMEN | RCU_CTL_PLLEN);
    /* Reset HXTALBPS bit */
    RCU_CTL &= ~(RCU_CTL_HXTALBPS);
    /* reset PLLSEL, PREDV0_LSB, PLLMF, USBFSPSC bits */
#ifdef GD32F10X_CL
    RCU_CFG0 &= ~(RCU_CFG0_PLLSEL | RCU_CFG0_PREDV0_LSB | RCU_CFG0_PLLMF |
                  RCU_CFG0_USBFSPSC | RCU_CFG0_PLLMF_4);
```

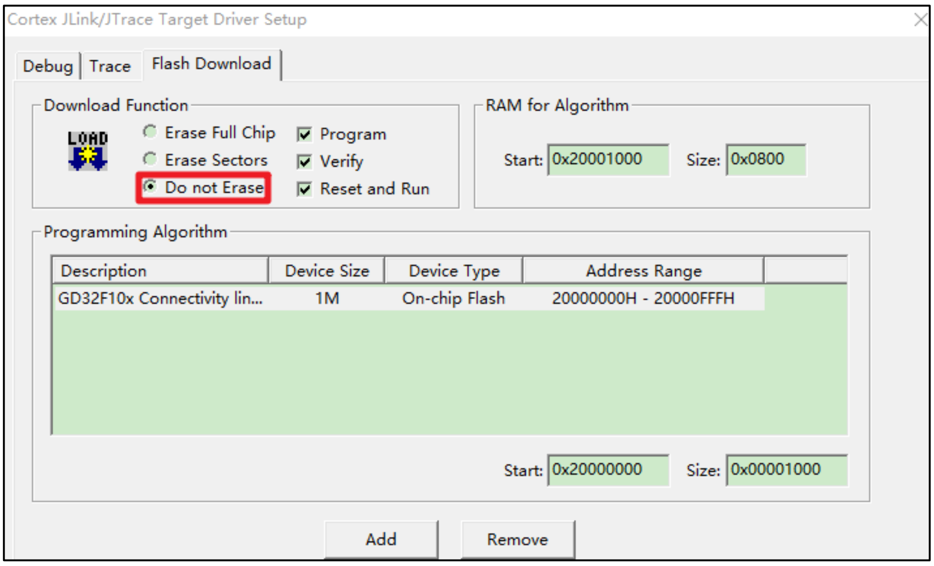
```
    RCU_CFG1 = 0x00000000U;
#else
    RCU_CFG0 &= ~(RCU_CFG0_PLLSEL | RCU_CFG0_PREDV0 | RCU_CFG0_PLLMF |
                  RCU_CFG0_USBDPSC | RCU_CFG0_PLLMF_4);
#endif /* GD32F10X_CL */

#if (defined(GD32F10X_MD) || defined(GD32F10X_HD) || defined(GD32F10X_XD))
    /* reset HXTALEN, CKMEN and PLLEN bits */
    RCU_CTL &= ~(RCU_CTL_PLLEN | RCU_CTL_CKMEN | RCU_CTL_HXTALEN);
    /* disable all interrupts */
    RCU_INT = 0x009F0000U;
#elif defined(GD32F10X_CL)
    /* Reset HXTALEN, CKMEN, PLLEN, PLL1EN and PLL2EN bits */
    RCU_CTL &= ~(RCU_CTL_PLLEN | RCU_CTL_PLL1EN | RCU_CTL_PLL2EN |
RCU_CTL_CKMEN | RCU_CTL_HXTALEN);
    /* disable all interrupts */
    RCU_INT = 0x00FF0000U;
#endif

    /* Configure the System clock source, PLL Multiplier, AHB/APBx prescalers and Flash settings */
    system_clock_config();
#ifdef VECT_TAB_SRAM
    nvic_vector_table_set(NVIC_VECTTAB_RAM,VECT_TAB_OFFSET);
#else
    nvic_vector_table_set(NVIC_VECTTAB_FLASH,VECT_TAB_OFFSET);
#endif
}
```

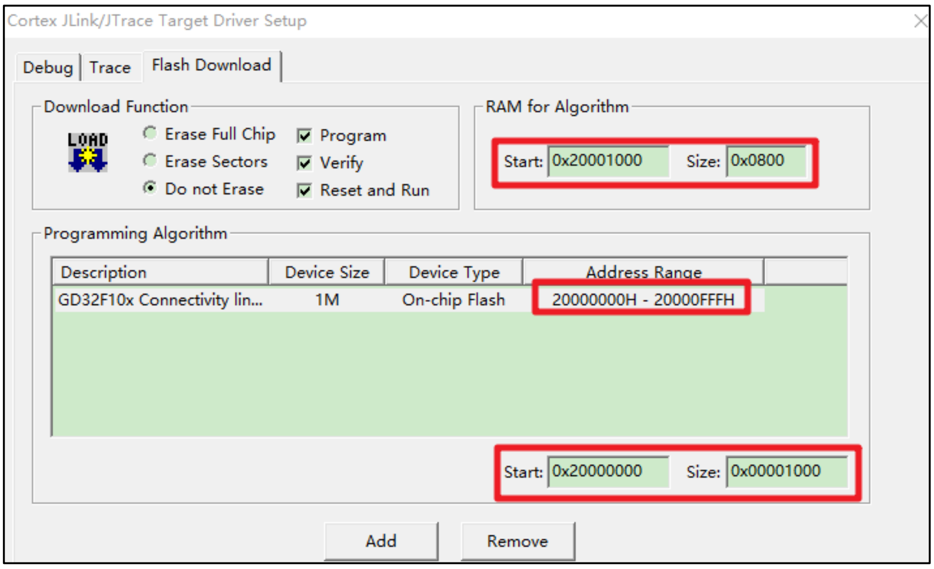
3. 在“Option for Target -> Debug -> Setting -> Flash Download”中配置擦除方式为“Do not Erase”，如*图 2-4. 擦除方式选择*所示。

图 2-4. 擦除方式选择



4. 在“Option for Target -> Debug -> Setting -> Flash Download”中将算法地址修改为 SRAM 地址。如*图 2-5. 算法地址修改*所示。

图 2-5. 算法地址修改



5. 在 startup 文件（如 startup_gd32f10x_cl.s）中 Reset_Handler 之前，采用 SPACE 申请一片空内存，如*图 2-6. SPACE 申请空内存*所示。使得 Reset_Handler 定位到 0x200001E0 地址处，如*图 2-7. Reset_Handler 地址重定位*所示。

图 2-6. SPACE 申请空内存

```
Skip_Mem                SPACE    0x7C
                        ;DCD       0xF1E0F85F
__Vectors_End

__Vectors_Size          EQU    __Vectors_End - __Vectors

                        AREA    |.text|, CODE, READONLY

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

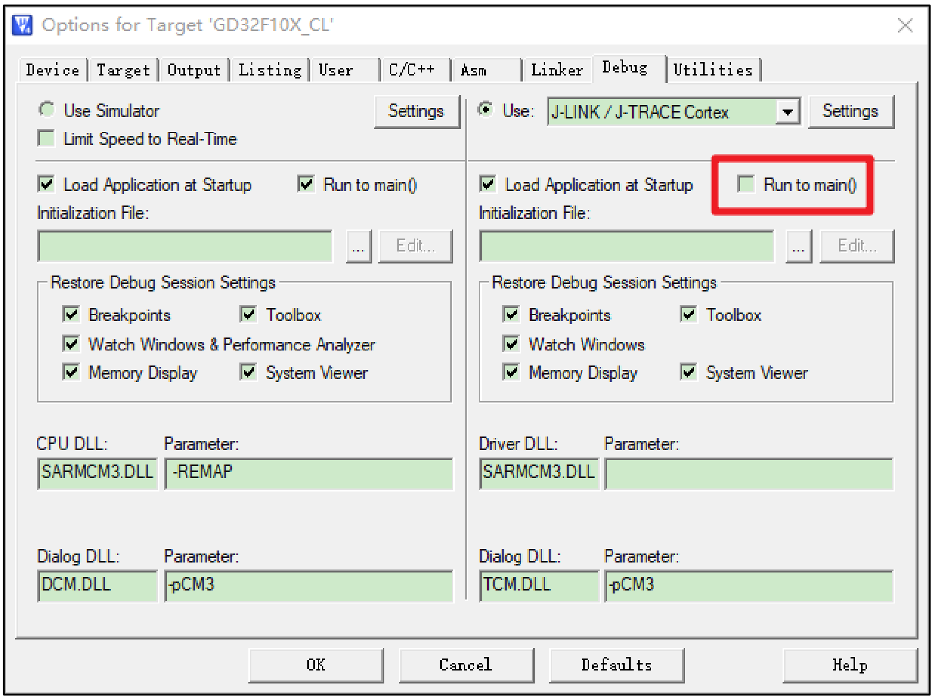
图 2-7. Reset_Handler 地址重定位

```
__rt_final_cpp                          0x200001dd   Thumb  Code     0
__rt_final_exit                         0x200001dd   Thumb  Code     0
Reset_Handler                           0x200001e1   Thumb  Code     8
ADC0_1_IRQHandler                       0x200001fb   Thumb  Code     0
CAN0_EWMC_IRQHandler                    0x200001fb   Thumb  Code     0
CAN0_RX0_IRQHandler                     0x200001fb   Thumb  Code     0
CAN0_RX1_IRQHandler                     0x200001fb   Thumb  Code     0
CAN0_TX_IRQHandler                      0x200001fb   Thumb  Code     0
```

3. Debug 模式下演示

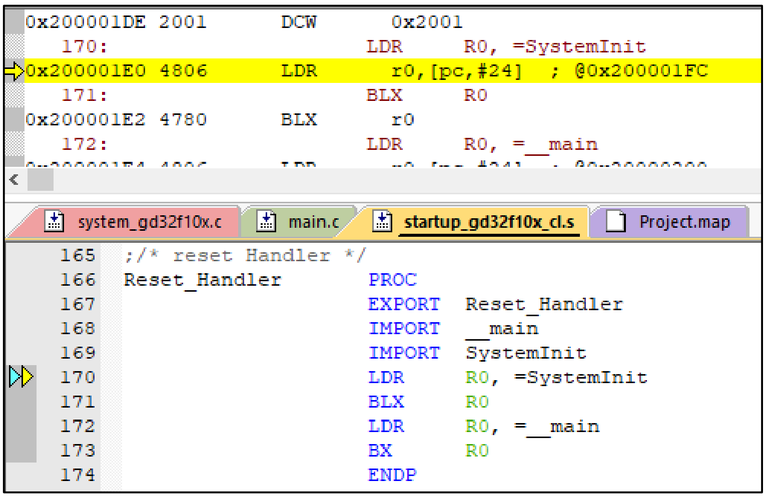
在"Option for Target -> Debug"中去掉"Run to main()"，如图 3-1. 去掉"Run to main()"所示。

图 3-1. 去掉"Run to main()"



进入 debug 模式，可以看到程序从 0x200001E0 地址处开始运行。

图 3-2. Debug 程序



至此，程序从 SRAM 启动成功，只要不断电，复位后程序可运行。

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	2021 年 11 月 01 日