作者：粤嵌.温子祺

1 编写BootLoader程序

1.1 代码搬运

```
/**
 * @bieaf 进行程序的覆盖
 * @detail 1.擦除目的地址
 * 2.源地址的代码拷贝到目的地址
 * 3.擦除源地址
 *
 *@param dest_addr 目的地址
 *@param src_addr 源地址
 *@param word_size 字大小
  *@return none
  */
 void move_code( uint32_t dest_addr, uint32_t src_addr,uint32_t
word_size)
 {
  uint32_t temp[256];
  uint32_t i;

  /*1.擦除目的地址*/
  printf("> start erase application 1 sector......\r\n");
```

```
19
20   sector_erase(APPLICATION_1_SECTOR);
21
22   printf("> erase application 1 success......\r\n");
23
24   /*2.开始拷贝*/
25   printf("> start copy......\r\n");
26
27   for(i = 0; i <word_size/1024; i++)
28   {
29   flash_read((src_addr + i*1024), temp, 256);
30   flash_program((dest_addr + i*1024), temp, 256);
31   }
32
33   printf("> copy finish......\r\n");
34
35   /*3.擦除源地址*/
36   printf("> start erase application 2 sector......\r\n");
37
38   sector_erase(APPLICATION_2_SECTOR);
39
40   printf("> erase application 2 success......\r\n");
41 }
```

1.2 程序跳转

1.程序跳转前，需要重新设置MSP（main stack pointer）。

```
1 /* 采用汇编设置栈的值 */
2 __asm void MSR_MSP (uint32_t ulAddr)
3 {
4  //设置主栈指针
5  MSR MSP, r0
6  BX r14
7 }
```

2.程序跳转需要使用函数指针，预先提前定义。

```
1 typedef void (*jump_func)(void);
```

3.执行程序覆盖。

```
/**
 * @bieaf 进行程序的覆盖
 * @detail 1.擦除目的地址
 * 2.源地址的代码拷贝到目的地址
 * 3.擦除源地址
 *
 * @param none
 * @return none
 */
void iap_execute_app (uint32_t app_addr)
{
 jump_func jump_to_app;

 printf("* ( __IO uint32_t * ) app_addr =%08X ,app_addr=%08X\r\n",* ( __IO uint32_t * ) app_addr,app_addr );

 //检查栈顶地址是否合法
 //该栈顶从应用程序处获取
 //栈顶=IRAM起始地址+RW-data大小+ZI-data大小
 if ( ( ( * ( __IO uint32_t * ) app_addr ) & 0x2FFE0000 ) == 0x200006B0 )
 {
 printf("stack is legal\r\n");

 //用户代码区第二个字为程序开始地址(复位地址)
 jump_to_app = (jump_func) * ( __IO uint32_t *)(app_addr + 4);

 //初始化APP堆栈指针(用户代码区的第一个字用于存放栈顶地址)
 MSR_MSP( * ( __IO uint32_t * ) app_addr );

 //跳转到APP
 jump_to_app();
 }
 printf("stack is illegal\r\n");
}
```

栈顶

栈顶=IRAM起始地址+RW-data大小+ZI-data大小，当程序成功编译后，输出显示以下信息：

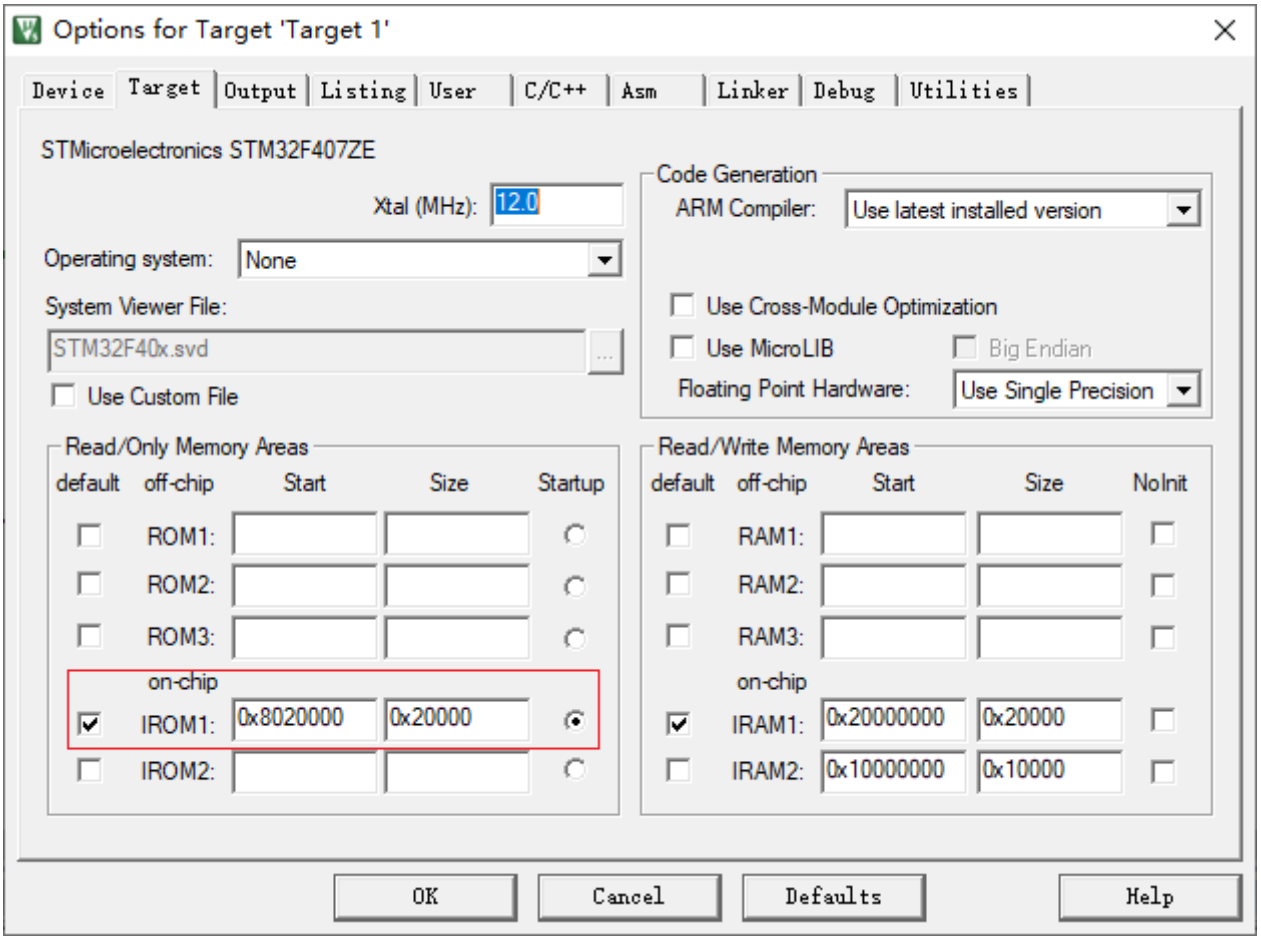
```
Program Size: Code=2636 RO-data=424 RW-data=52 ZI-data=1660
```

若IRAM起始地址为0x20000000，则栈顶地址=0x20000000+424+52=0x200006B0。

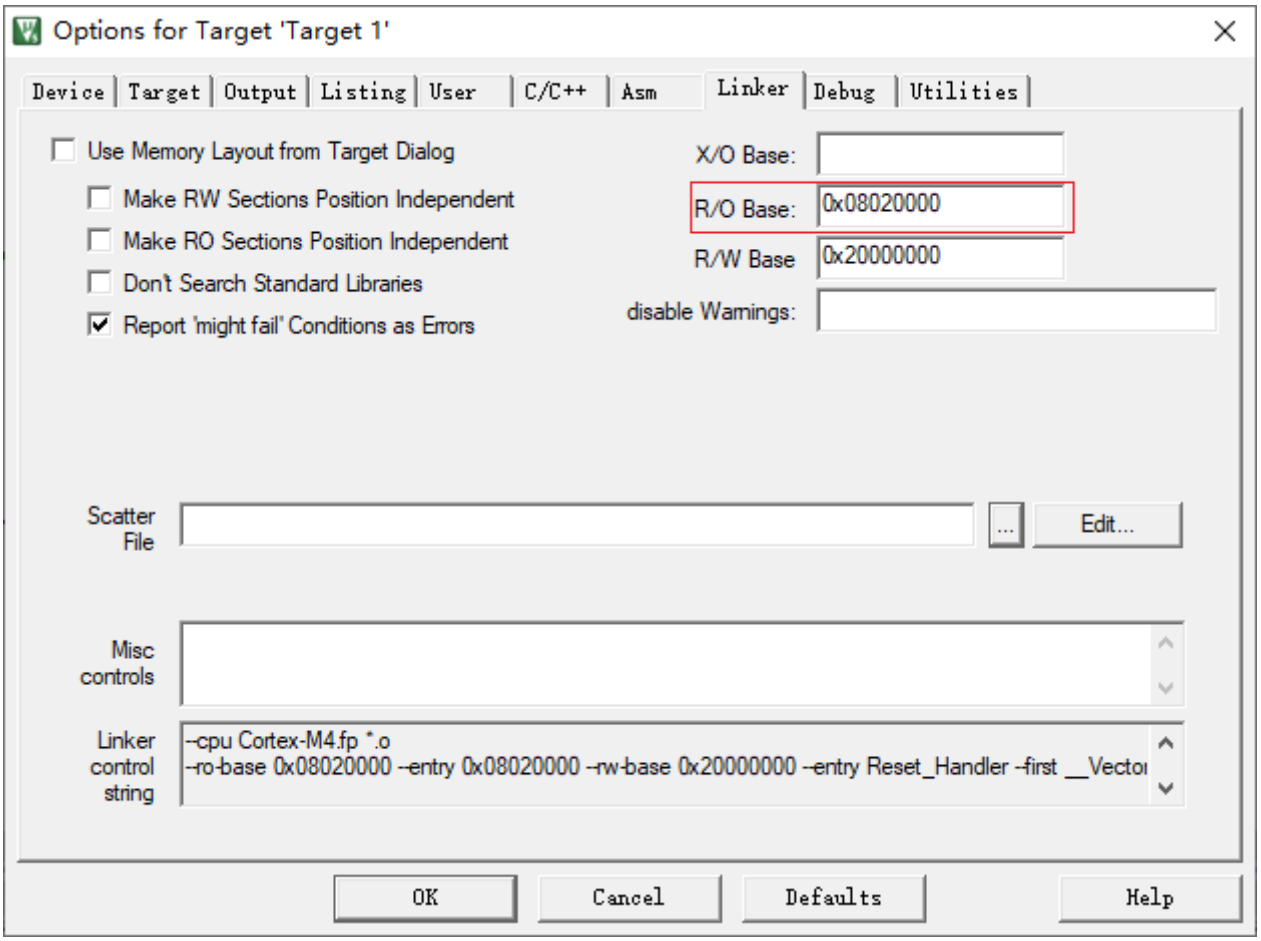
2 应用程序

2.1 工程配置

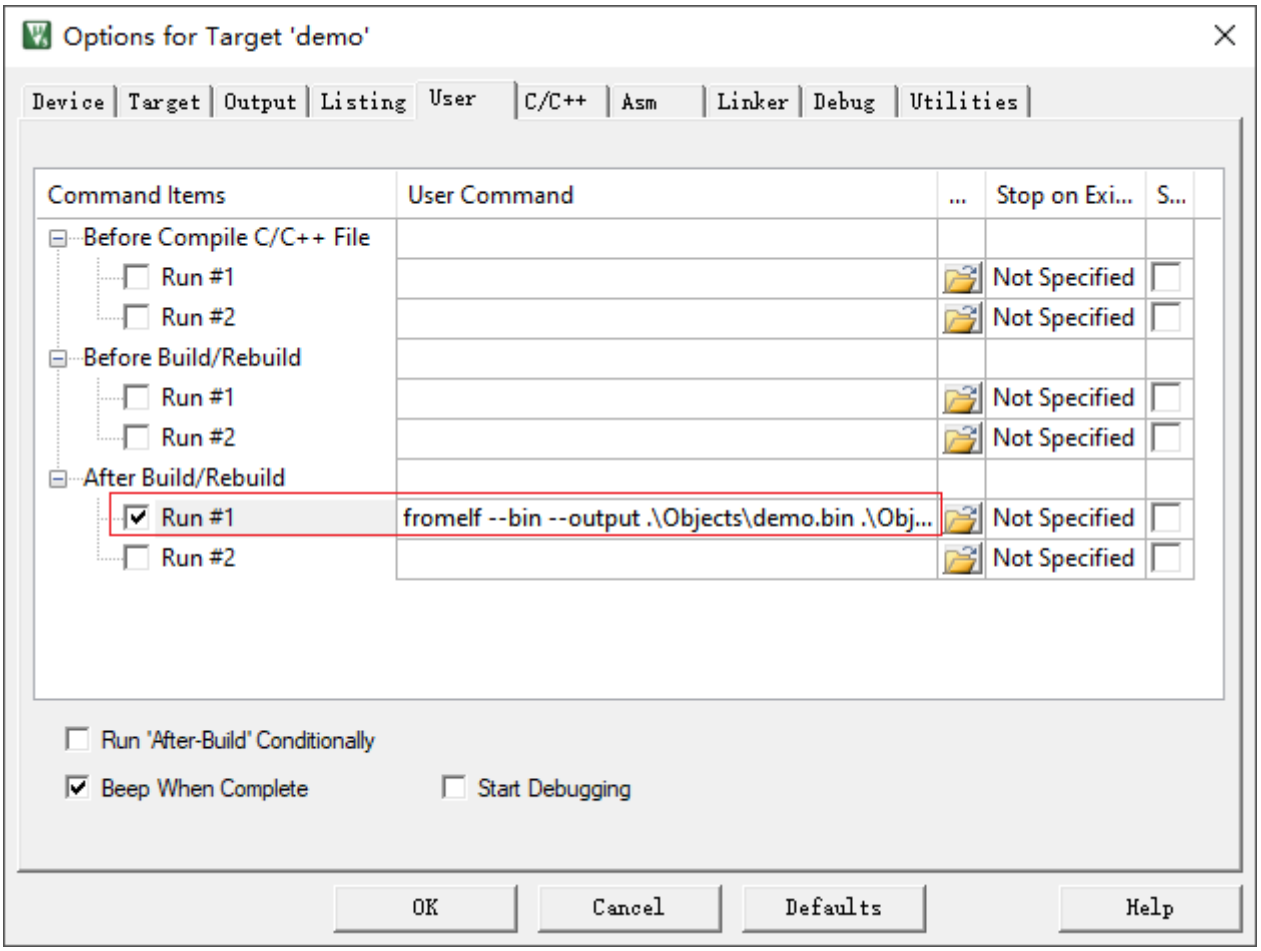
1.在【Target】标签页，重新配置IROM的起始地址和大小，目前该应用代码存储在扇区5，则起始地址为0x08020000，大小为128KB，即0x20000，如下图。



2.在【Linker】标签页，R/O Base设置为扇区5的起始地址0x08020000。



3.在【User】标签页中，在“After Build/Rebuild”中勾选口Run #1，并增加“fromelf --bin --output .\Objects\demo.bin .\Objects*.axf”，生成bin文件，可用于ymodem下载。



2.2 ymodem下载代码

```
void ymodem_download(void)
{
 uint16_t crc = 0;

static
 uint8_t data_state = 0;

 if(ymodem_get_state()==TO_START)
 {
  ymodem_send_cmd(CCC);

  delay_ms(1000);
  }

  /* 串口1接收完一个数据包 */
  if(g_usart1_rx_end)
  {

  /* 清空接收完成标志位、接收计数值 */
```

```
20  g_usart1_rx_end=0;
21  g_usart1_rx_cnt=0;
22
23  switch(g_usart1_rx_buf[0])
24  {
25  case SOH://数据包开始
26  {
27  crc = 0;
28
29  /* 计算crc16 */
30  crc = crc16((uint8_t *)&g_usart1_rx_buf[3], 128, crc);
31
32  if(crc != (g_usart1_rx_buf[131]<<8|g_usart1_rx_buf[132]))
33  return;
34
35  if((ymodem_get_state()==TO_START)&&(g_usart1_rx_buf[1] == 0x00)&&(g_usa
rt1_rx_buf[2] == (uint8_t)(~g_usart1_rx_buf[1])))// 开始
36  {
37
38  ymodem_set_state(TO_RECEIVE_DATA);
39
40  /* 若ymodem_send_cmd执行在sector_erase之前，则导致串口数据丢包，因为擦除会关
闭所有中断 */
41  /* 擦除应用程序2的扇区 */
42  sector_erase(APPLICATION_2_SECTOR);
43
44  data_state = 0x01;
45  ymodem_send_cmd(ACK);
46  ymodem_send_cmd(CCC);
47
48
49  }
50  else if((ymodem_get_state()==TO_RECEIVE_END)&&(g_usart1_rx_buf[1] == 0x
00)&&(g_usart1_rx_buf[2] == (uint8_t)(~g_usart1_rx_buf[1])))// 结束
51  {
52  update_set_down();
53  ymodem_set_state(TO_START);
54  ymodem_send_cmd(ACK);
55
56  /* 嘀一声示，表示下载完成 */
57  beep_on();delay_ms(80);beep_off();
58
```

```
59   /* 复位 */
60   NVIC_SystemReset();
61   }
62   else if((ymodem_get_state()==TO_RECEIVE_DATA)&&(g_usart1_rx_buf[1] == d
ata_state)&&(g_usart1_rx_buf[2] == (uint8_t)(~g_usart1_rx_buf[1])))// 接收数
据
63   {
64
65   /* 烧录程序 */
66   flash_program((APPLICATION_2_ADDR + (data_state-1) * 128), (uint32_t *)
(&g_usart1_rx_buf[3]), 32);
67   data_state++;
68
69   ymodem_send_cmd(ACK);
70   }
71   }break;
72
73   case EOT://数据包传输结束
74   {
75   if(ymodem_get_state()==TO_RECEIVE_DATA)
76   {
77
78   ymodem_set_state(TO_RECEIVE_EOT2);
79   ymodem_send_cmd(NACK);
80   }
81   else if(ymodem_get_state()==TO_RECEIVE_EOT2)
82   {
83
84
85   ymodem_set_state(TO_RECEIVE_END);
86   ymodem_send_cmd(ACK);
87   ymodem_send_cmd(CCC);
88   }
89
90   }break;
91
92   default:break;
93   }
94
95   }
96 }
```