stm32\_keil链接地址修改方法

作者：滑国青

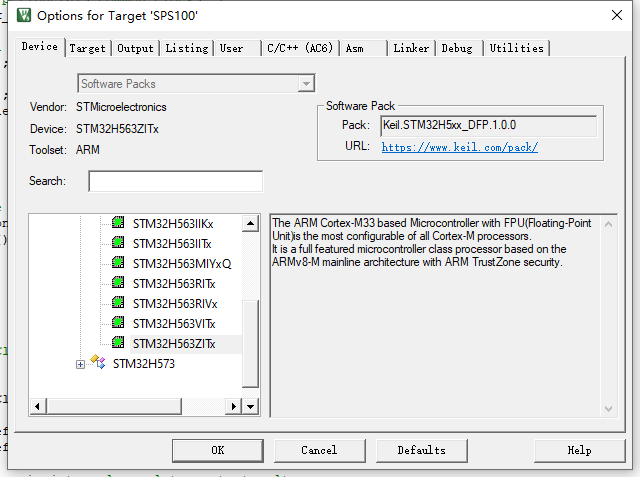
历史：2023.08.12

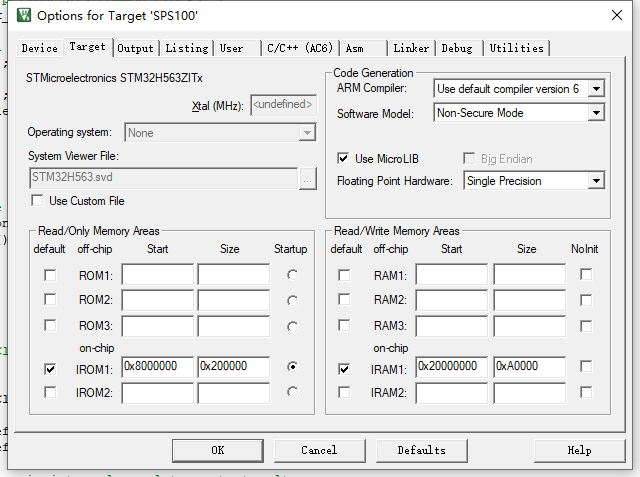
# 前言

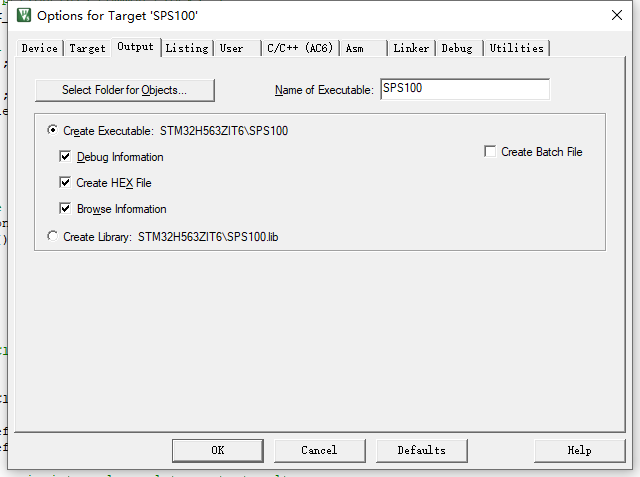
# Sps100链接地址A面

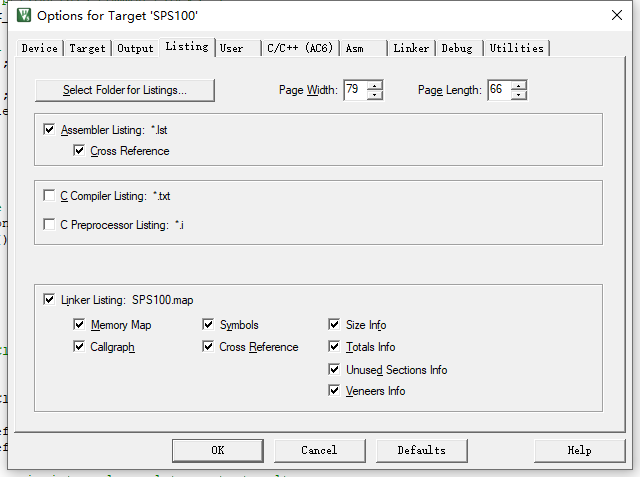
MDK-ARM\STM32H563ZIT6\SPS100.sct

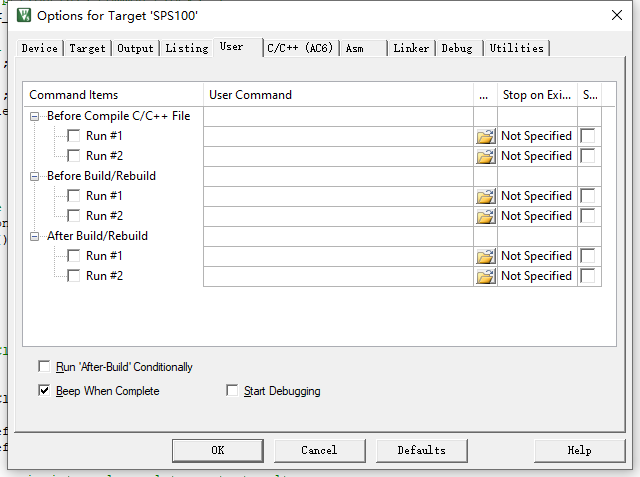


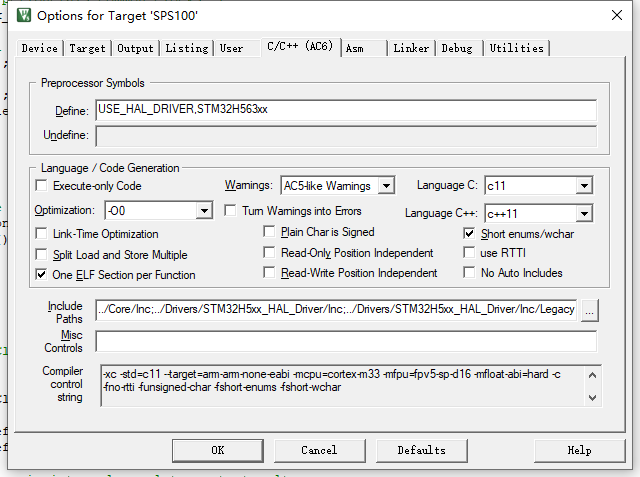


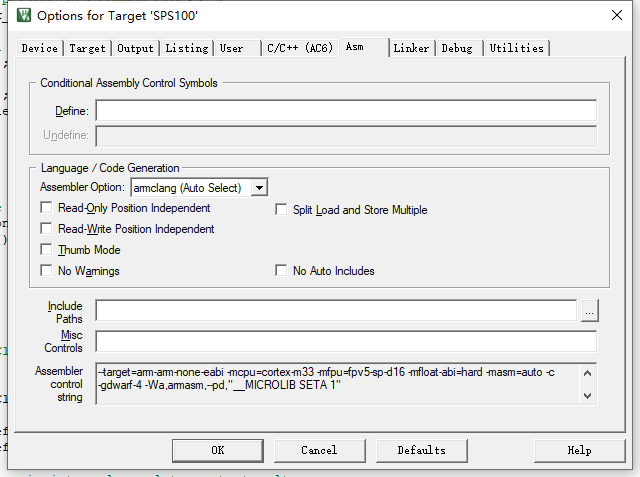


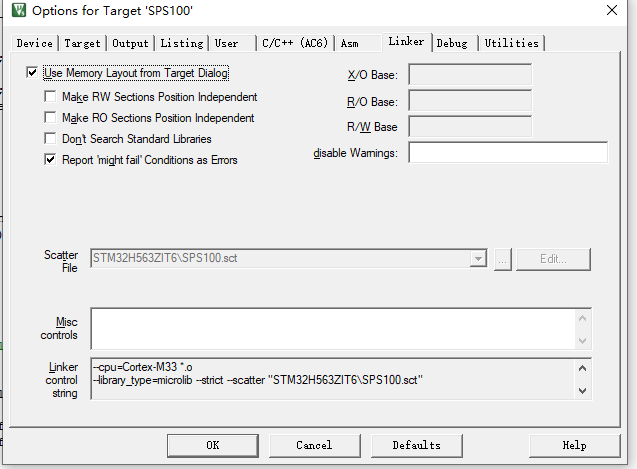


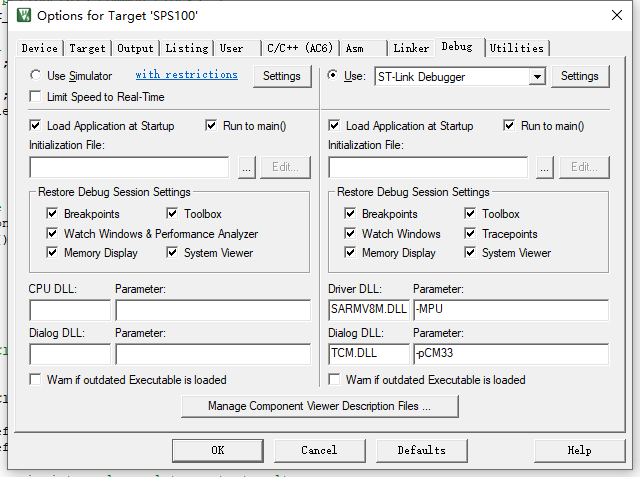


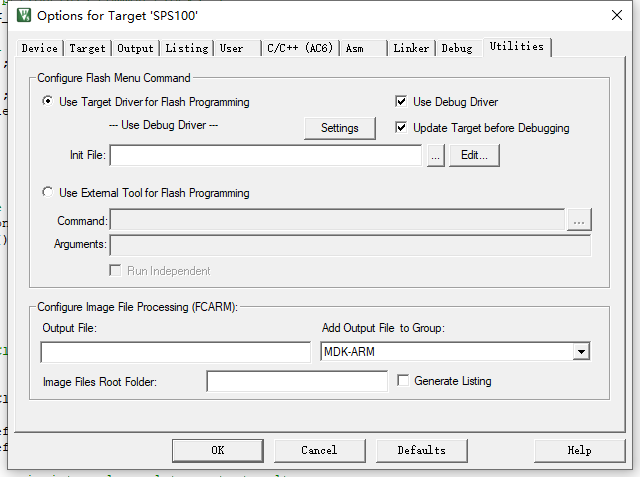












# Sps100链接地址B面

# 关键问题

## 如何修改keil中链接地址

在Keil中修改链接地址，您可以按照以下步骤进行操作：

1. 打开Keil软件，并打开您的项目。
2. 在项目窗口中，右键单击您的项目，并选择“Options for Target”选项。
3. 在弹出的对话框中，选择“Target”选项卡。
4. 在“Code Generation”部分，您可以看到“Use Memory Layout from Target Dialog”选项。确保此选项已勾选。
5. 点击“Manage”按钮，打开“Scatter File”对话框。
6. 在“Scatter File”对话框中，您可以看到当前使用的链接描述文件。您可以选择编辑当前的描述文件，或者创建一个新的描述文件。
7. 如果您选择编辑当前的描述文件，可以点击“Edit”按钮，然后在弹出的编辑器中修改链接地址。
8. 如果您选择创建一个新的描述文件，可以点击“Create”按钮，然后按照您的需求进行描述文件的编辑。
9. 在描述文件中，您可以找到和修改链接地址的相关部分。通常，链接地址会在描述文件的“LR\_IROM1”或类似的部分进行定义和修改。
10. 修改完链接地址后，保存描述文件并关闭编辑器。
11. 返回到“Options for Target”对话框，点击“OK”按钮保存更改。
12. 重新编译和链接您的项目，新的链接地址将会生效。

请注意，修改链接地址可能会对程序的运行产生影响，因此请谨慎操作，并确保您了解所做更改的后果。



## Stm32h5xx读写flash

### chatGPT的回答

STM32H563ZITx flash的参数

以下是STM32H563ZITx微控制器的Flash参数：

* Flash存储器大小：2MB
* 扇区（Sector）大小：128KB
* 页（Page）大小：2KB
* Flash编程字长：64位（8字节）
* Flash编程速度：最高可达400KB/s
* Flash擦除时间：根据擦除的扇区或页的数量和大小而变化，一般在几毫秒到几十毫秒之间
* Flash寿命：通常为10万次擦写/擦除周期

这些参数可以用于计算Flash编程或擦除操作的时间，以及确定要擦除或编程的扇区或页的地址范围。在进行Flash编程或擦除操作时，需要根据这些参数来选择合适的方法和参数。

stm32h5xx flash读写 编程

在STM32H5xx系列微控制器上进行Flash读写编程，可以使用HAL库或者直接操作寄存器的方法。

使用HAL库的方法如下：

1. 初始化Flash模块：

c复制代码

FLASH\_EraseInitTypeDef eraseInitStruct;

eraseInitStruct.TypeErase = FLASH\_TYPEERASE\_PAGES;

eraseInitStruct.PageAddress = FLASH\_BASE;

eraseInitStruct.NbPages = (FLASH\_BANK\_SIZE / FLASH\_PAGE\_SIZE);

uint32\_t pageError = 0;

HAL\_FLASH\_Unlock();

HAL\_FLASHEx\_Erase(&eraseInitStruct, &pageError);

HAL\_FLASH\_Lock();

1. 写入数据到Flash：

c复制代码

uint32\_t data = 0x12345678;

HAL\_FLASH\_Unlock();

HAL\_FLASH\_Program(FLASH\_TYPEPROGRAM\_WORD, FLASH\_BASE, data);

HAL\_FLASH\_Lock();

1. 从Flash读取数据：

c复制代码

uint32\_t data = \*(uint32\_t\*)FLASH\_BASE;

使用直接操作寄存器的方法如下：

1. 初始化Flash模块：

c复制代码

FLASH->KEYR = FLASH\_KEY1;

FLASH->KEYR = FLASH\_KEY2;

FLASH->CR |= FLASH\_CR\_PER;

FLASH->AR = FLASH\_BASE;

FLASH->CR |= FLASH\_CR\_STRT;

while (FLASH->SR & FLASH\_SR\_BSY);

FLASH->CR &= ~FLASH\_CR\_PER;

1. 写入数据到Flash：

c复制代码

uint32\_t data = 0x12345678;

FLASH->CR |= FLASH\_CR\_PG;

\*(volatile uint32\_t\*)FLASH\_BASE = data;

while (FLASH->SR & FLASH\_SR\_BSY);

FLASH->CR &= ~FLASH\_CR\_PG;

1. 从Flash读取数据：

c复制代码

uint32\_t data = \*(volatile uint32\_t\*)FLASH\_BASE;

需要注意的是，Flash的写入操作会擦除整个Flash页，因此在写入之前需要先擦除对应的Flash页。另外，Flash的读写操作需要先解锁Flash模块，操作完成后再锁定Flash模块。

### 测试可用的结果

滑国青 8-14 11:24:34  
int flash\_erase(uint32\_t flash\_addr,int sector\_nums)

{

int sector\_idx = (flash\_addr&0xfffff)/FLASH\_SECTOR\_SIZE;

int bank\_id = (flash\_addr < 0x8100000)?FLASH\_BANK\_1:FLASH\_BANK\_2;

FLASH\_EraseInitTypeDef eraseInitStruct = {0};

eraseInitStruct.TypeErase = FLASH\_TYPEERASE\_SECTORS;

eraseInitStruct.Banks = bank\_id;

eraseInitStruct.Sector = sector\_idx;

eraseInitStruct.NbSectors = 1;

uint32\_t pageError = 0;

printf("erasing:bankid:%d,sector:%d,num:%d\n",bank\_id,sector\_idx,sector\_nums);

HAL\_FLASH\_Unlock();

HAL\_FLASHEx\_Erase(&eraseInitStruct, &pageError);

HAL\_FLASH\_Lock();

return 0;

}

/\*

TypeProgram: 8bytes: FLASH\_TYPEPROGRAM\_QUADWORD;

\*/

int flash\_write(uint32\_t TypeProgram, uint32\_t FlashAddress, uint32\_t DataAddress, uint32\_t len)

{

int i,k;

int n;

uint32\_t \*p = (uint32\_t\*)DataAddress;

//TODO,now only support quadword ,16bytes. So should align to 16 bytes.

if(TypeProgram != FLASH\_TYPEPROGRAM\_QUADWORD || DataAddress & 0xf || FlashAddress & 0xf ){

return -1;

}

n = (len+0xf)/0x10; //every time write 16 bytes.

HAL\_FLASH\_Unlock();

for(i = 0; i < n; i++){

HAL\_FLASH\_Program(TypeProgram, FlashAddress+i\*16, (uint32\_t)DataAddress+i\*16);

}

HAL\_FLASH\_Lock();

return 0;

}  
  
 uint8\_t data0[16] = {0x12,0x34,0x56,0x78,0x12,0x34,0x56,0x78,0x12,0x34,0x56,0x78,0x12,0x34,0x56,0x78};

uint8\_t data1[16] = {0xab,0xcd,0xef,0x78,0x12,0x34,0x56,0x78,0x12,0x34,0x56,0x78,0x12,0x34,0x56,0x78};

uint8\_t data2[16] = {0xef,0xab,0xcd,0x78,0x12,0x34,0x56,0x78,0x12,0x34,0x56,0x78,0x12,0x34,0x56,0x78};

//uint16\_t data0 = 0x1234;

//uint16\_t data1 = 0xabcd;

uint32\_t flashaddr0 = 0x81f8000;

uint32\_t flashaddr1 = 0x81fc000;

uint32\_t flashaddr2 = 0x81fe000;

flash\_erase(flashaddr0,1);

flash\_erase(flashaddr1,1);

flash\_erase(flashaddr2,1);

printf("\*\*\*%s:%d\n",\_\_FILE\_\_,\_\_LINE\_\_);

flash\_write(FLASH\_TYPEPROGRAM\_QUADWORD,flashaddr0,(uint32\_t)data0,sizeof(data0));

flash\_write(FLASH\_TYPEPROGRAM\_QUADWORD,flashaddr1,(uint32\_t)data1,sizeof(data1));

flash\_write(FLASH\_TYPEPROGRAM\_QUADWORD,flashaddr2,(uint32\_t)data2,sizeof(data2));  
  
滑国青 8-14 11:25:21  
你可以试一下，我试没有问题  
  
滑国青 8-14 11:26:26  
每次写16字节。2M分成2个bank,即bank1,bank2;  sector的编号是bank内的编号；sector大小是128KB.

没有问题，在写的时候，注意按16字节边界对齐作缓冲。

用cubeprogramer工具读出flash数据看时，注意选8-bit看，就能看到数据与写入的一致，如果用32-bit看时，工具会转成小序int32数值，让人以为数据大小序不对。

# 结束