Ⅰ、写在前面

本文主要讲述的内容：基于 Keil 开发工具下，STM32 内部 RAM 在线调试配置方法，以及每一项配置的详细说明。如需要了解更多相关的文章，可以到我博客，或微信公众号查看。

让程序运行在 RAM 中调试代码有两优点：1.速度快；2.减少对芯片 FLASH 读写次数，增加芯片寿命。

本文牵涉的知识比较多，如果弄明白所有细节问题，对自己这方面的技能是一种很大的提升。

本文基于 ST 公司 Cortex-M 内核的 STM32 来讲述其配置方法，其实也适用于其他公司（如：TI、NXP 等）的 Cortex-M 芯片，原理都是一样的。

关于 Keil（MDK-ARM）介绍、下载、安装与注册：
http://blog.csdn.net/ybhuangfugui/article/details/51501781

作者：strongerHuang

关于本文的更多详情请往下看。

Ⅱ、本文要点

1.主要内容

由于本文牵涉的内容比较多，我会按章节来讲述各项内容，大体分为：

- 实现 STM32 内部 RAM 调试的配置方法
- 每条配置的详细说明
- 网上配置方法说明及存在的不足

2.工程代码下载

为了方便大家学习，我将**配置前**（一般常用）工程和**配置后**工程分别打包上传至百度网盘供大家下载参考学习。配置前和配置后工程实现的功能都是一样的。本文以 STM32F1 系列芯片为例（其他芯片类似）。

配置前工程代码 STM32F10x_Demo：
http://pan.baidu.com/s/1gfx8J6b

配置后工程代码 STM32F10x_Demo（RAM 调试）：
http://pan.baidu.com/s/1cDXYQM

注意：由于许多网盘近年来受到影响都相继停止服务或关闭了，如果网盘链接失效，可以微信公众号查看更新链接，或微信联系作者。

3.代码功能描述

上面提供下载的代码实现的功能是一样的，具体如下两点：

- 间隔 500ms LED 亮灭变化一次，串口打印数据“Demo..”一次。
- 串口中断接收数据，会将收到数据通过串口发送出去。

第一点是为了有一个状态显示，知道程序在运行。
第二点在本文中的作用也很大，就是使用了**中断功能**。由于 RAM 调试会牵涉到**向量表**，中断功能就会使用到向量表，如果没有配置正确，这里就不会响应中断，或者出错。

4.验证配置成功方法

本文提供的“STM32F10x_Demo”是断电后重新上电会继续运行代码；而“STM32F10x_Demo（RAM 调试）”是断点之后程序丢掉了，也就是不能运行了【请更加 LED 及串口打印现象来判断】。

注意：使用 RAM 调试之前请将 FLASH 里面的数据擦除掉，否则使用 RAM 调试断电再上电，程序会从 FLASH 运行，会认为程序依然在运行，从而影响判断。

Ⅲ、RAM 调试配置方法

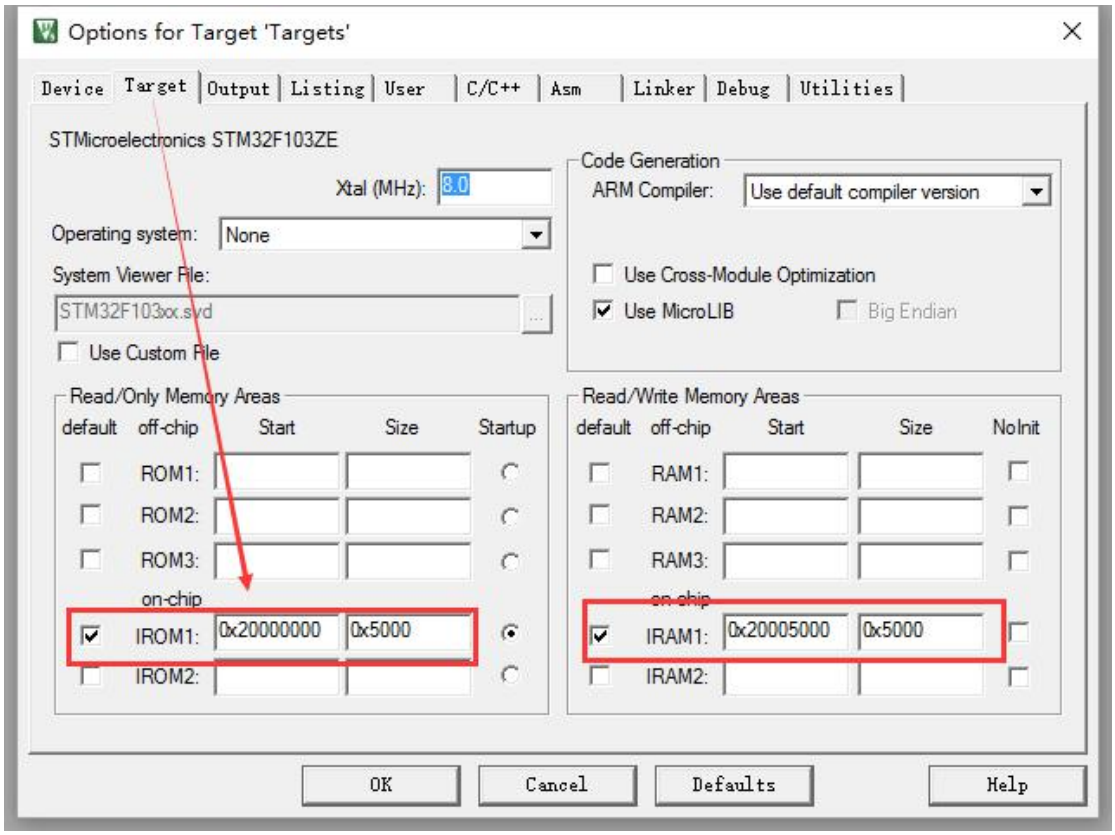
本节主要讲述配置方法的过程，为什么这么配置，以及配置的原理将会在下一章节讲述。

1.修改内存地址

打开目标配置：Project -> Options for Target -> Target 或“工程目标配置”点击快捷按钮。

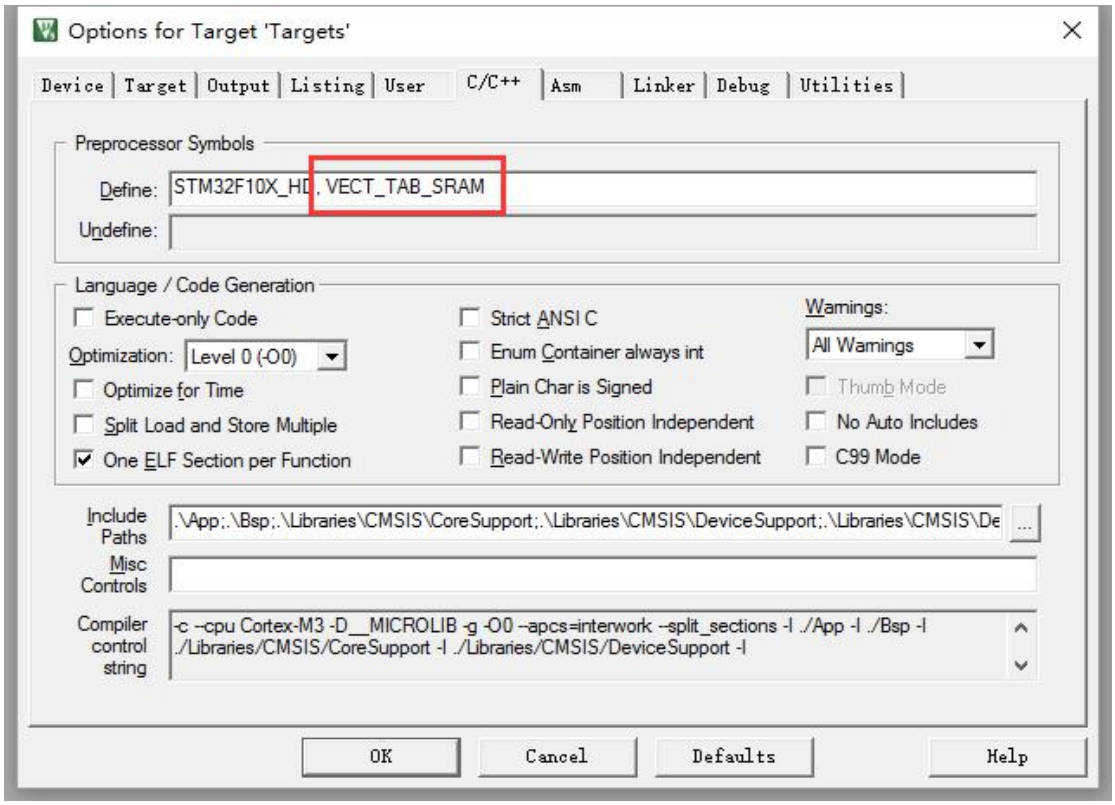
将 ROM 和 RAM 地址映射到如下图地址。 我们使用 STM32F103ZE 芯片，该芯片的 RAM 大小为 0x10000 即 64KB，我们这里平分 RAM，即各自的大小为 0x5000。

注意：配置的地址范围不能超过芯片实际的大小。



2.配置向量表

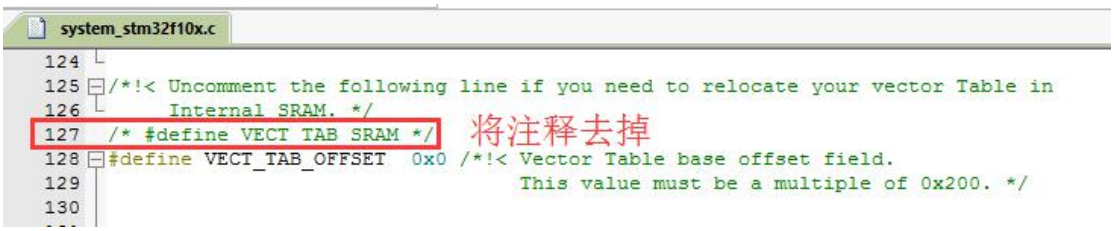
同上，打开目标配置：Project -> Options for Target -> C/C++，使用宏定义**VECT_TAB_SRAM**。如下图：



这里的宏定义是为了让向量表指向 RAM（我们默认是指向 ROM），**重要的一个目的就是让中断向量表指向 RAM**，上面“代码功能”中断的意义就是为了验证向量表的正确性。

注意：这里的宏定义是在工具链中配置的，多个宏定义之间需要有**“逗号”隔离**开来。

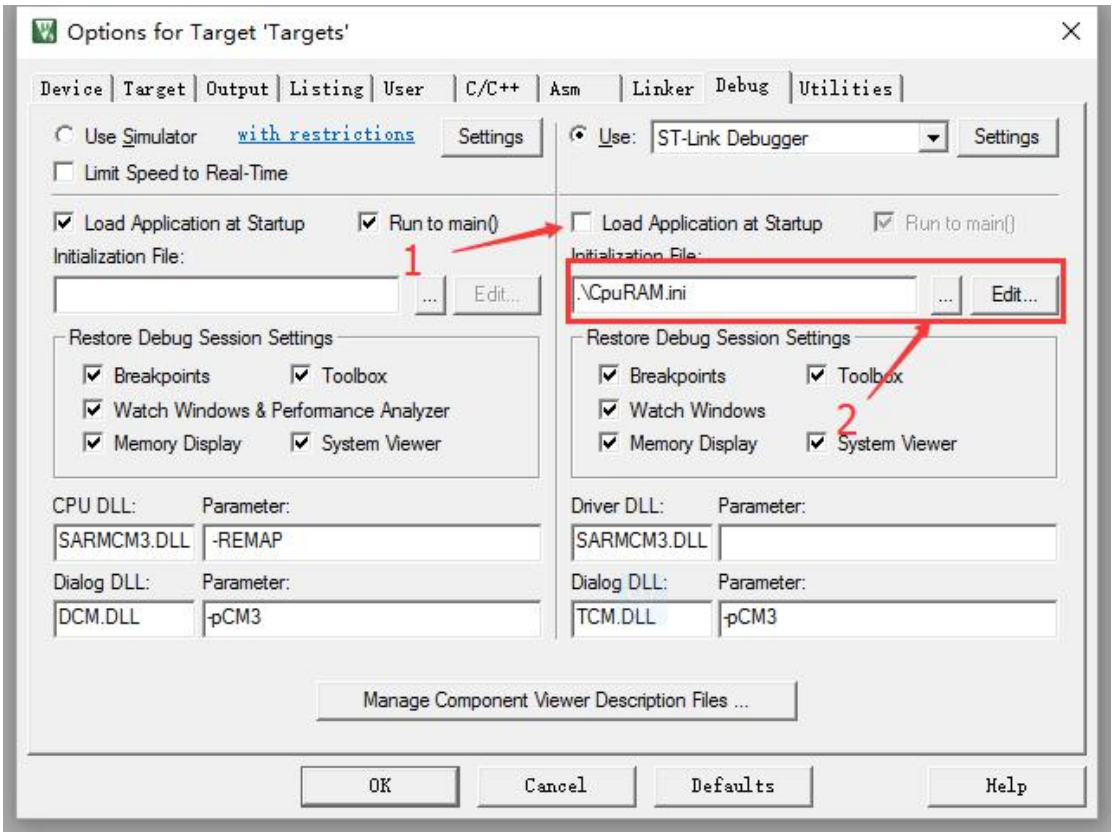
其实这里的宏定义配置也可以在源代码中实现，打开 system_stm32f10x.c 文件下第 127 行的“VECT_TAB_SRAM”宏定义，如下图：



【个人建议：调试和非调试代码最好一致，也就是源代码不变】

3.调试配置

同上：Project -> Options for Target -> Debug， 这里是关于调试的配置。
第一步：去掉“Load Application at Startup”前面的勾选项
第二步：导入 RAM 初始化文件。



RAM 初始化文件里面内容如下：

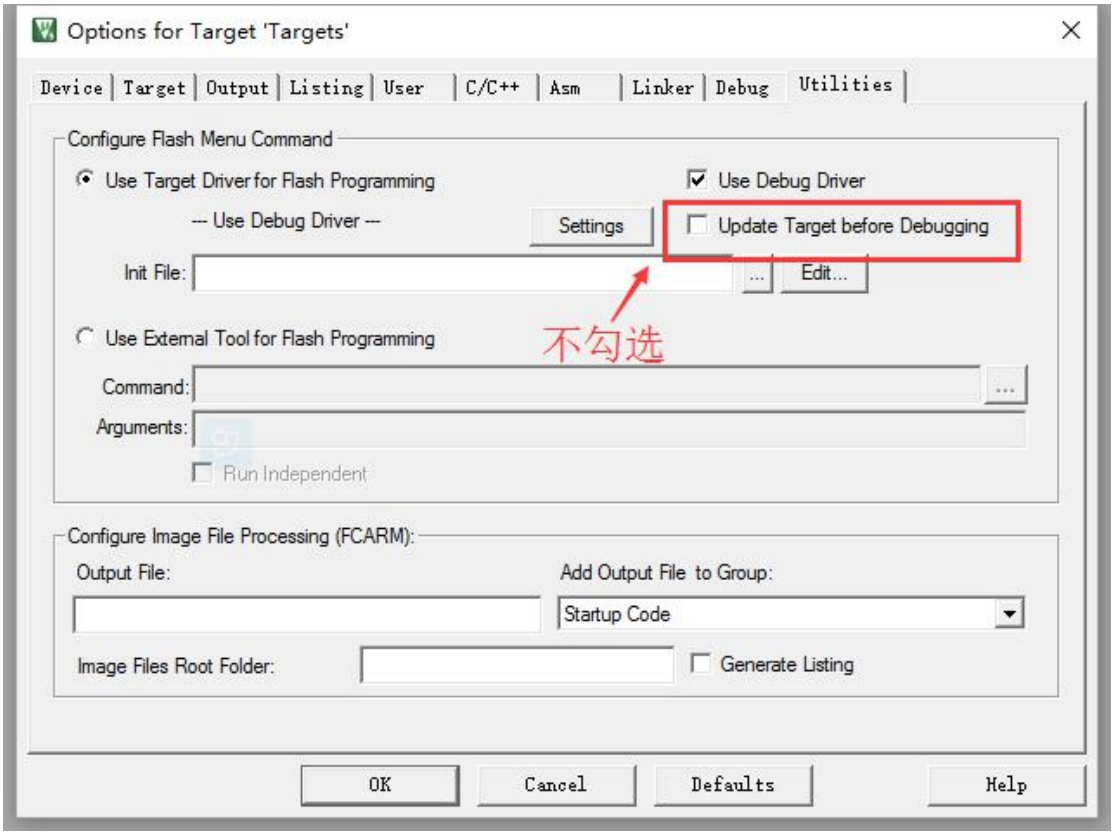
```
FUNC void Setup (void) {
   SP = _RDWORD(0x20000000);
   PC = _RDWORD(0x20000004);
   _WDWORD(0xE000ED08, 0x20000000);
}

LOAD Objects\ExecutableFile.axf INCREMENTAL
Setup();
g, main
```

每一条语句具体意思请见源代码注释，这里提示的是 **Objects\ExecutableFile.axf** 也就是**输出路径**和**输出文件名**，它的路径与文件名与你工程配置需对应。

4.调试不更新目标程序

同上：Project -> Options for Target -> Utilities，不勾选“更新”。意思就是在线调试时，不更新芯片（下载 FLASH）程序。



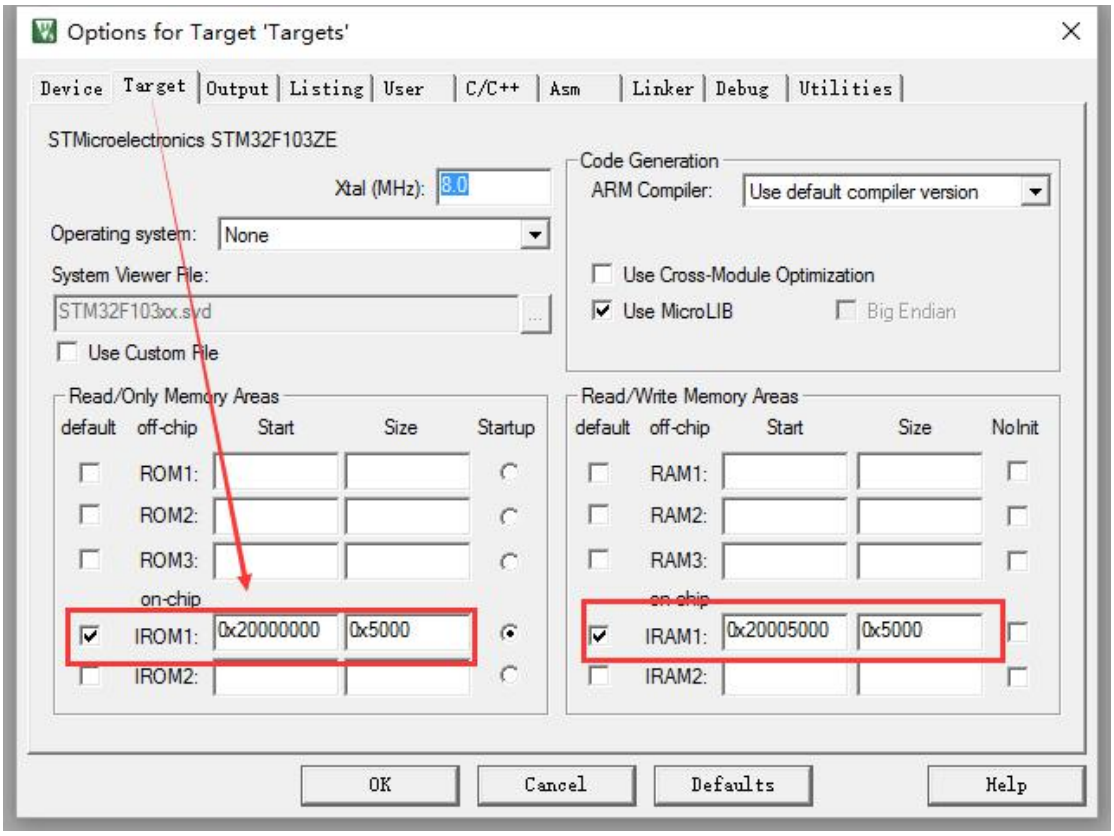
至此，STM32 内部 RAM 在线调试配置方法就完成了，连接开发板就可以使用 RAM 在线调试代码了。

网上相关的问题还有其他无关的配置，我会在下面单独说明一下为什么不用配置那些。

Ⅳ、配置说明

上面配置过程已经知道了，这一节讲述一下为什么这样配置，以及这么配置的意思。

1.修改内存地址说明

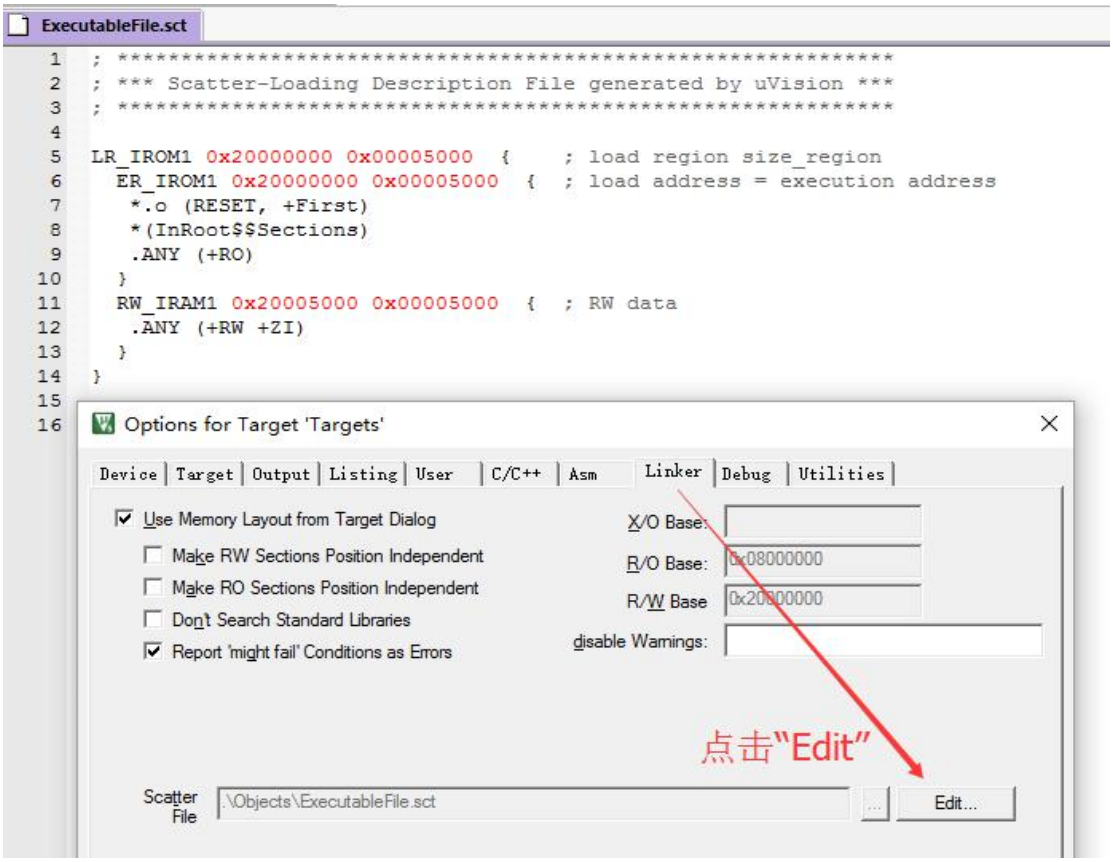


内存地址为什么 ROM 设置为 0x20000000，RAM 设置为 0x20005000。原因在于芯片的 RAM 其实地址就是 0x20000000（没有猜错的话 Cortex-M 那芯片 RAM 起止地址都是 0x20000000）。

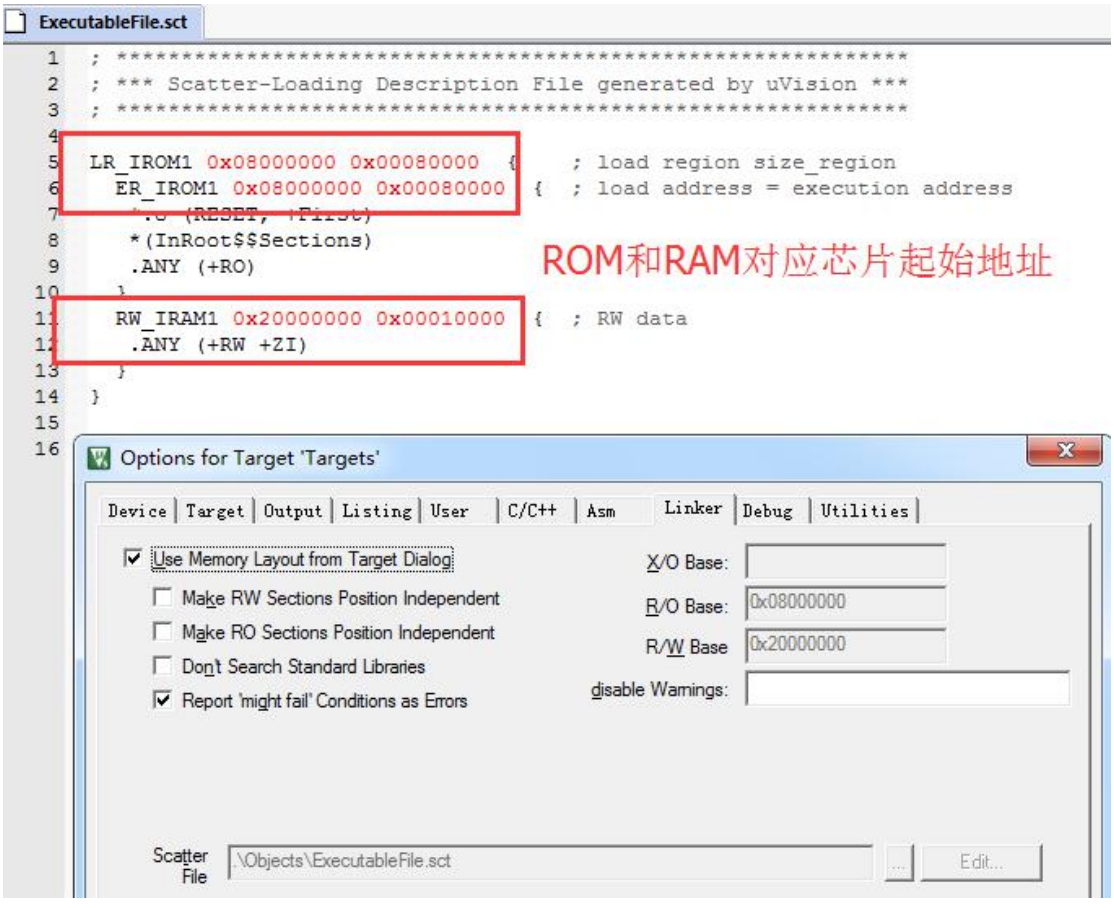
至于大小嘛，就是看芯片型号了，我们这里平分大小，也可以不用平分大小。

这里分配的地址会直接影响输出的文件“ExecutableFile.sct”,也就是我们链接的时候需要使用到的“ExecutableFile.sct”文件。

查看“ExecutableFile.sct”文件的方法：Project -> Options for Target -> Linker，如下图。【需要编译之后才能输出“ExecutableFile.sct”文件，即编译后才能查看】

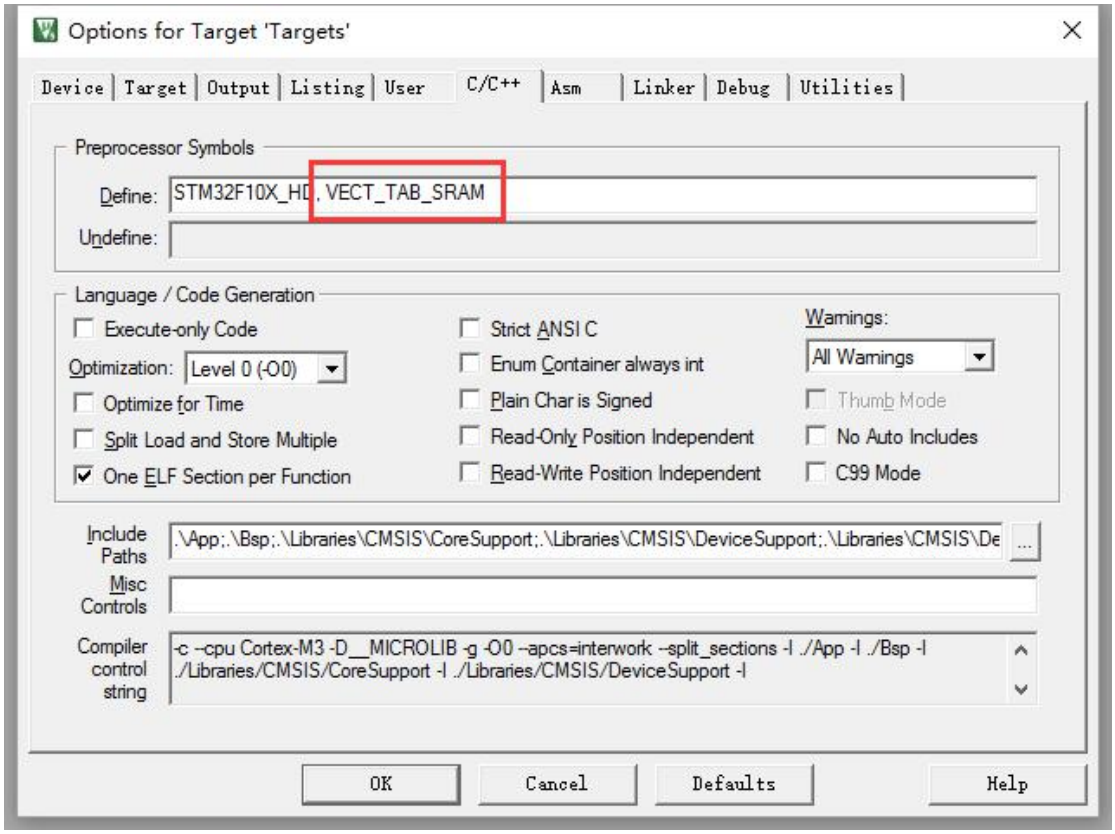


对比没有配置 RAM 调试（也就是没有修改地址）的工程如下图：



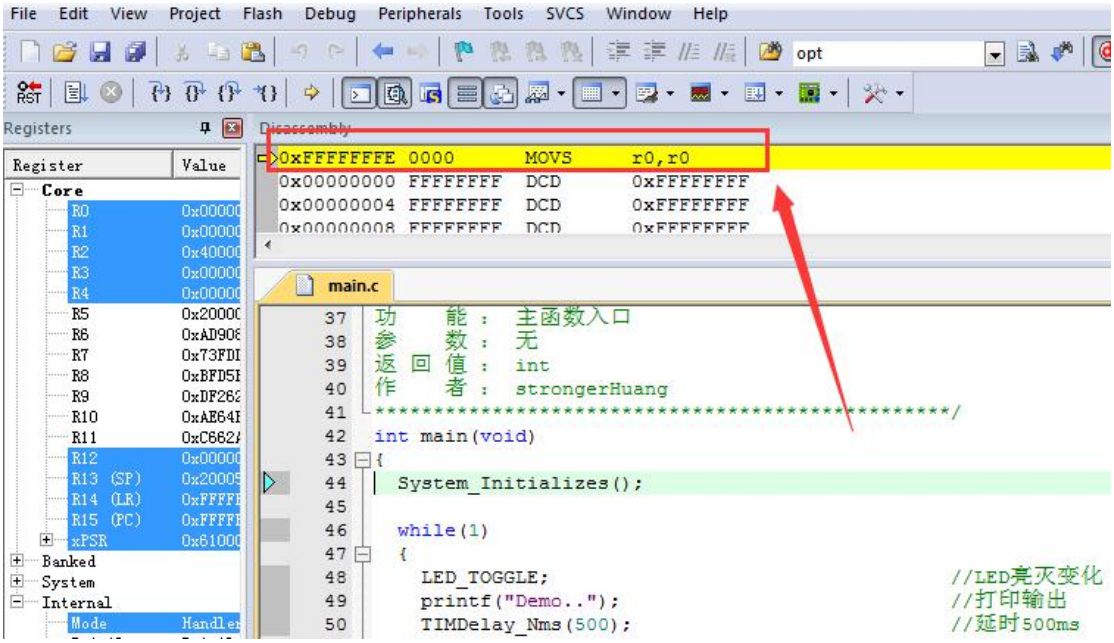
网上配置教程说要修改 Linker 下的地址，其实是**多余的**，详情请见下一章节。

2.配置向量表说明

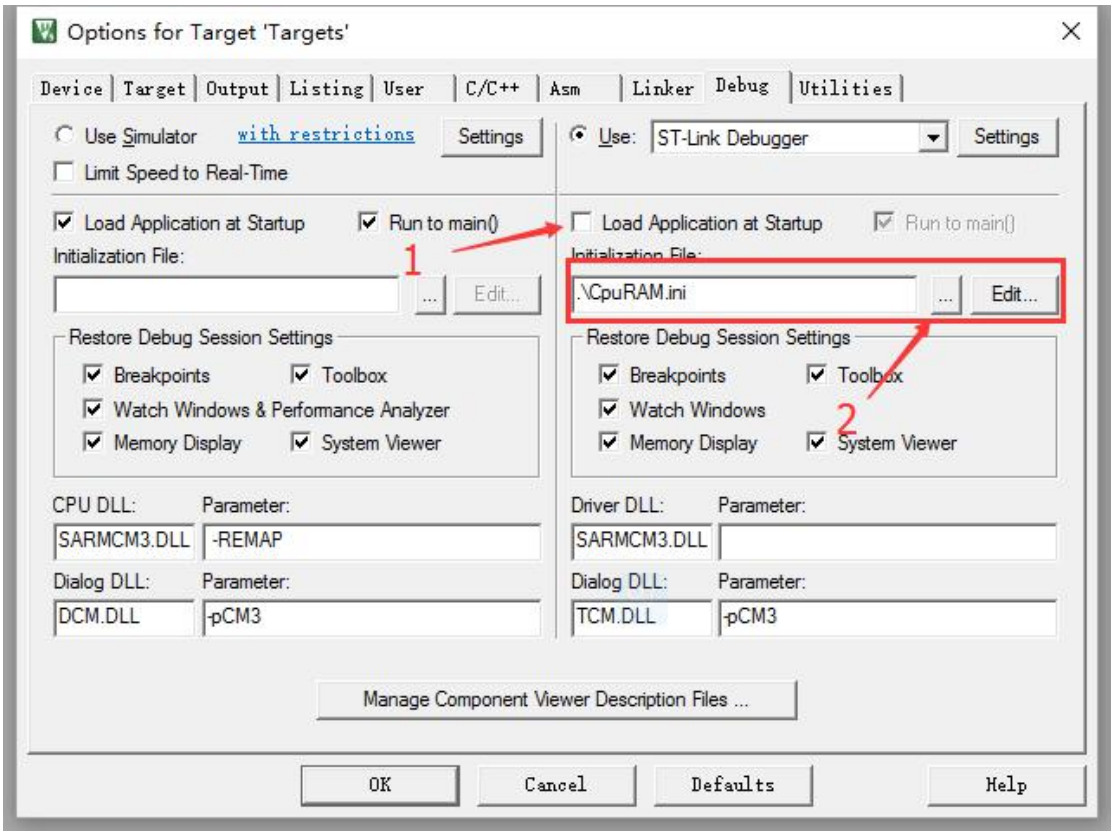


爱思考（或者会寻到问题）的朋友可能会发现，我**不宏定义** VECT_TAB_SRAM 这个参数，程序照常可以运行（LED 变化、串口打印数据）。

其实这里的配置主要是针对“向量表”，比如中断向量表。如果当我们**不宏定义** VECT_TAB_SRAM 这个参数，测试串口中断的时候，程序就会跑死，（暂停）程序会指向一个非法的地址，如下图：



3.调试配置说明

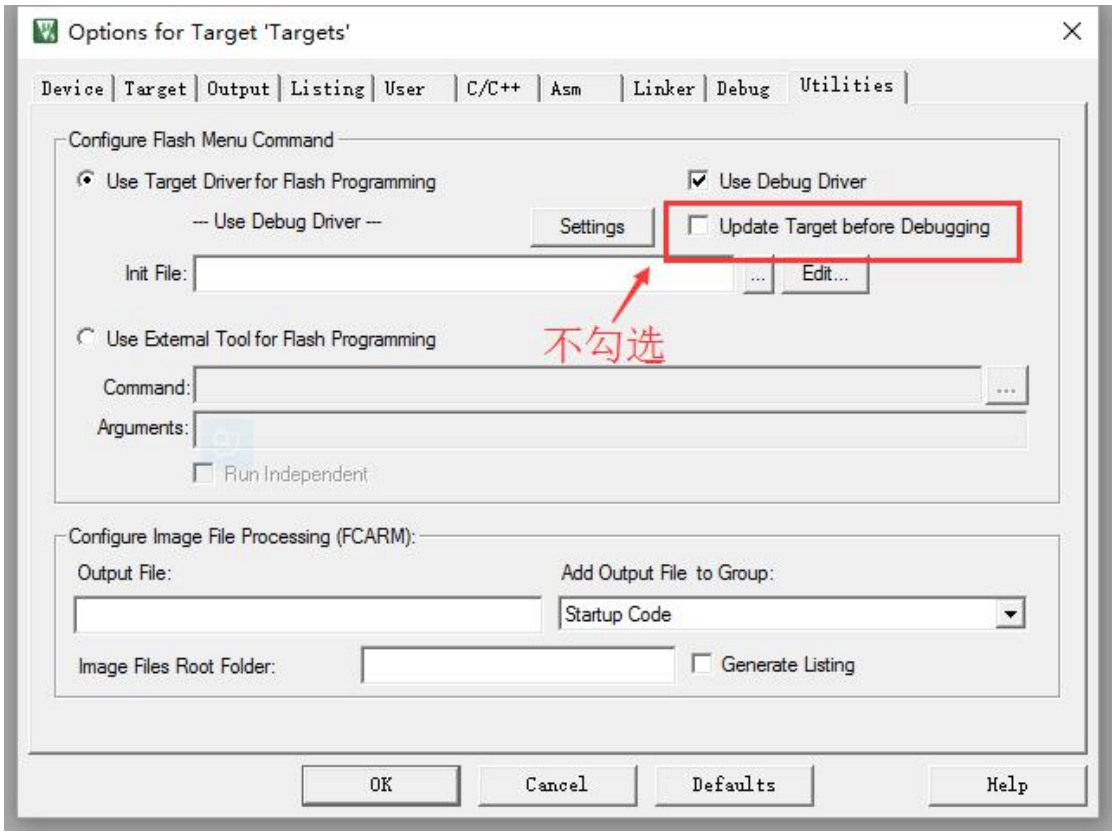


这个地方的配置很好理解，就是我们要将程序指针指向我们特定的地址（RAM）区域，这样好让程序执行我们指定地址里面的程序。

我们加载文件“CpuRAM.ini”，因此不需要勾选“Load Application at Startup”这个选项。

加载文件的名称“CpuRAM.ini”和网上一些教程命名一样，可以自己命名，只要后缀名一样就行。

4.调试不更新目标程序说明



这个地方其实就是在调试的时候更新（下载）芯片 FLASH 的代码，由于我们没有修改 FLASH 的烧写算法，这里就不勾选次选项。

当我们修改了 FLASH 的烧写算法（程序指向 RAM），这里可以勾选上。

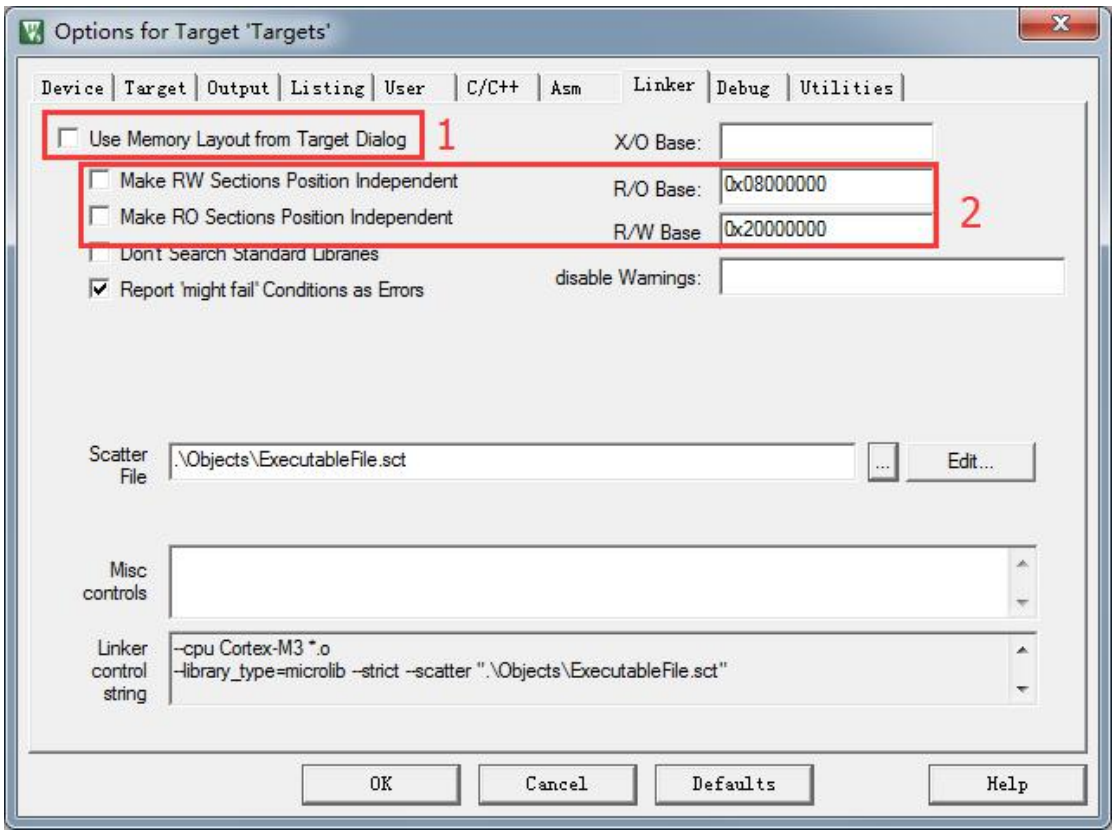
相比两者，我们选择不勾选该选项简单一点，因此这里选择不勾选。

V、网上配置说明

笔者开始学习 RAM 调试的内容时，也是参考网上很多的教程，但是经过笔者亲自，并且多次测试发现网上的有些教程存在不足之处。本节主要是提出网上某些教程存在的不足或者多余之处。【若有不对之处敬请谅解】

1.修改 Linker 地址

Project -> Options for Target -> Linker



网上的配置，这里的地址基本上都是修改了的。
1.去掉勾选； 2.再次修改地址。

我刚开始学习配置时也是修改了的，但后面我再次配置时发现一个问题：**地址前面为勾选**。什么意思呢，就是没有使用这个配置的地址。

于是我就不配置（不修改）这里的地址进行验证，结果还是可以在 RAM 中调试，大量测试也没发现什么问题。

我再次查看 Scatter File 文件“ExecutableFile.sct”，发现不修改地址也是一样的。其实“ExecutableFile.sct”文件的地址是我上一章节里面说的，由 Target 里面的地址决定的。

因此，网上所配置的这里其实是多余的配置。

2.配置向量表

有很多教程使用了在 main 函数开始配置向量表，也就是在 main 函数开始出增加一条语句：NVIC_SetVectorTable(NVIC_VectTab_RAM, 0x0);

这条语句其实是 system_stm32f10x.c 文件里面第 265 行的：SCB -> VTOR = SRAM_BASE | VECT_TAB_OFFSET;一样的意思。

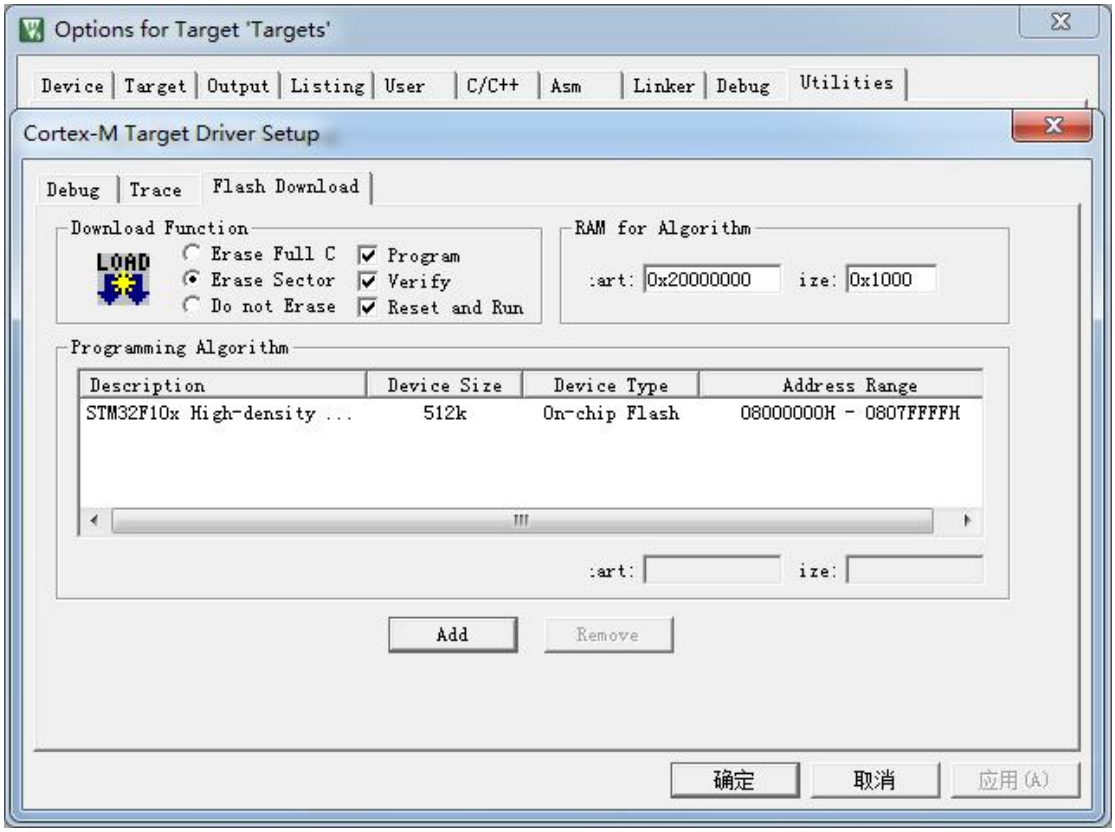
因此，我个人觉得，项目中的调试代码和真正运行的源代码不能有差异。定义了

VECT_TAB_SRAM 这个宏定义，就没必要还在 main 函数里面增加一条语句。

细心的朋友可能会发现，我在“III、RAM 调试配置方法”这一章节中没有修改源代码，只是修改了配置。这样就**保证了代码的一致性**。

3.修改编程地址

网上有很多教程是修改了下图中编程的地址，也就是修改了编程算法。



作为调试，本来就是运行在 RAM 中，还在这里配置，我觉得是多次一举。因此我们上面讲述的是没有勾选“Update Target Before Debugging”

VI、说明

STM32 内部 RAM 调试代码时，复位不起作用，需要复位请重新下载运行。关于 RAM 在线调试配置还有许多未讲述完，请亲自配置并测试验证，你或许会明白更多有用知识。

以上总结仅供参考，若有不对之处，敬请谅解。

Ⅶ、最后

我的博客：http://blog.csdn.net/ybhuangfugui
微信公众号：EmbeddDeveloper

本着免费分享的原则，方便大家业余利用手机学习知识，定期在微信公众号分享相关知识。如果觉得文章的内容对你有用，又想了解更多相关的文章，请用微信搜索“EmbeddDeveloper” 或者扫描下面二维码、关注，将有更多精彩内容等着你。