在 SRAM、FLASH 中調試代碼的配置方法(附詳細步驟)

https://www.cnblogs.com/zhengnian/p/11715696.html

因為 STM32 的 FLASH 擦寫次數有限（大概為 1 萬次），所以為了延長 FLASH 的使用時間，我們平時調試時可以選擇在 SRAM 中進行硬件調試。除此之外，SRAM 存儲器的寫入速度比在內部 FLASH 中要快得多，所以下載程序到 SRAM 中的速度較快。

所以我們很有必要建立兩個版本的工程配置，在 SRAM 中調試程序完畢後，再把代碼下載到 FLASH 中即可。這篇筆記主要分享在 keil5 中配置 FLASH 調試與 SRAM 調試的詳細配置方法及如何切換兩種配置。

本篇筆記以 STM32F103ZET6 為例。其 FLASH 大小為 512KB，SRAM 的大小為 64KB。FLASH 基地址為 0x08000000，SRAM 基地址為 0x20000000。在 STM32F10XXX 裡，可以通過 BOOT1、BOOT0 引腳來選擇三種不同的模式：

BOOT0	BOOT1	启动模式	说明
0	X	用户闪存存储器	用户闪存存储器，也就是FLASH启动
1	0	系统存储器	系统存储器启动，用于串口下载
1	1	SRAM启动	SRAM启动，用于在SRAM中调试代码

我們要在 FLASH 中進行硬件仿真調試還是在 RAM 中進行硬件仿真調試需要對這兩個 boot 腳進行對應的設置以及程序下載的地址進行設置。

在 FLASH 中進行硬件仿真調試

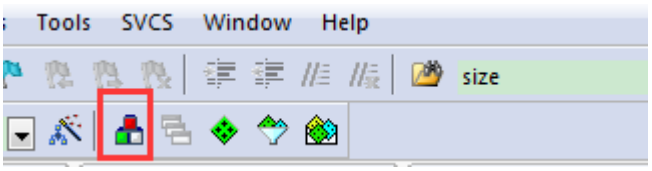
1、硬件設置

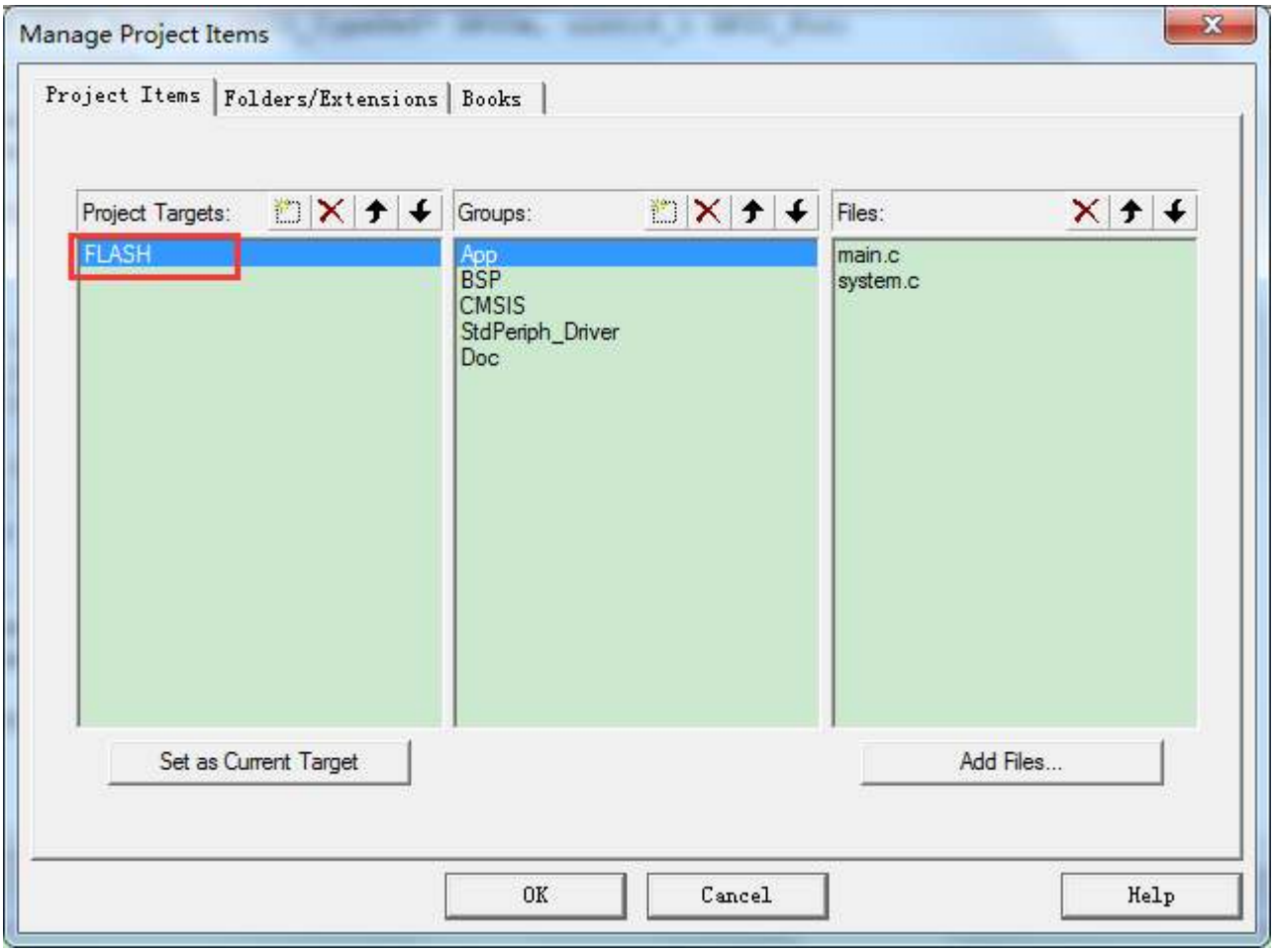
BOOT0 配置為 0，BOOT1 隨意設置。

2、keil 設置

本文以 keil5 為例。步驟如下：

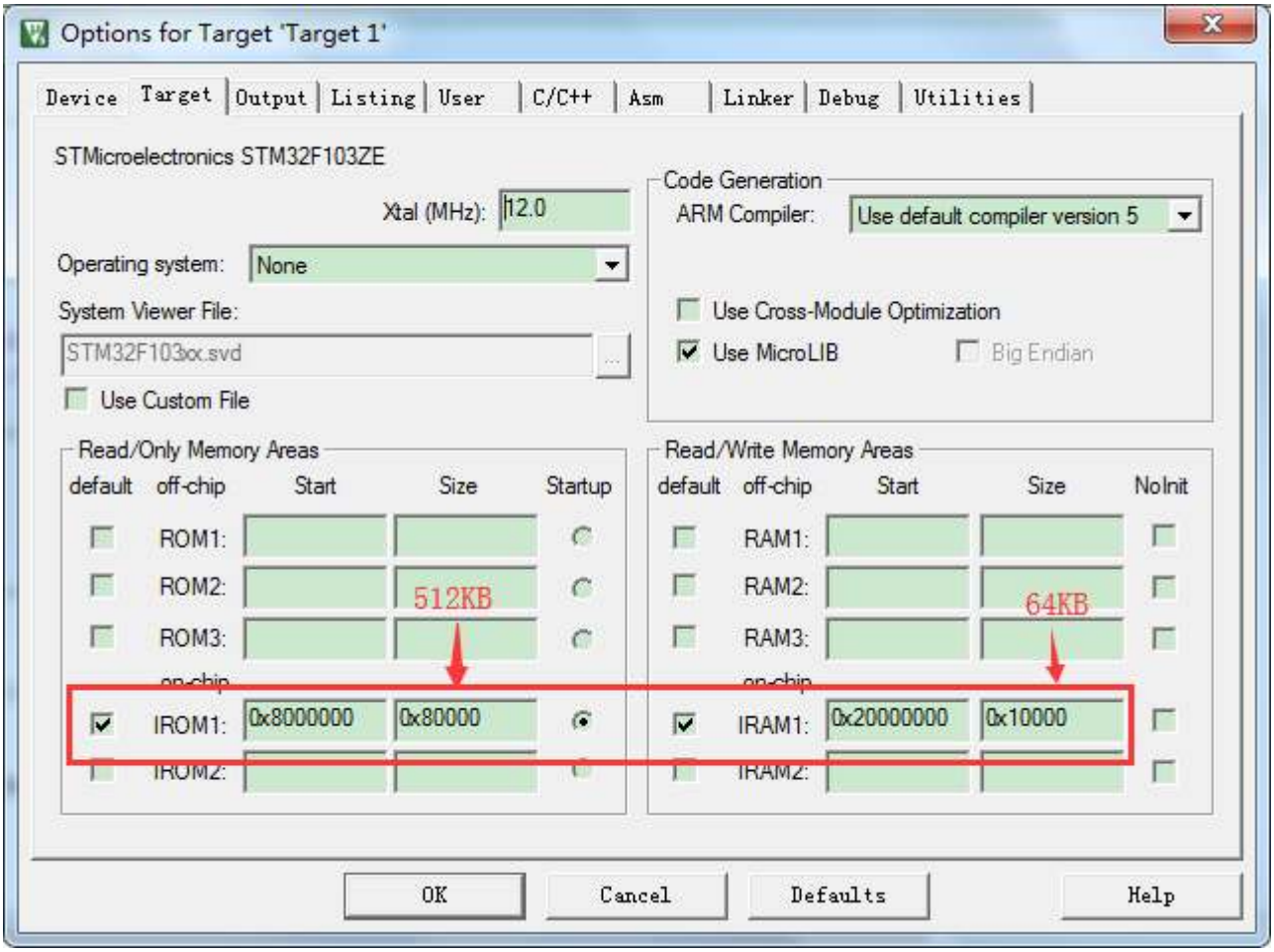
（1）點擊如下按鈕，修改 target 的名稱：





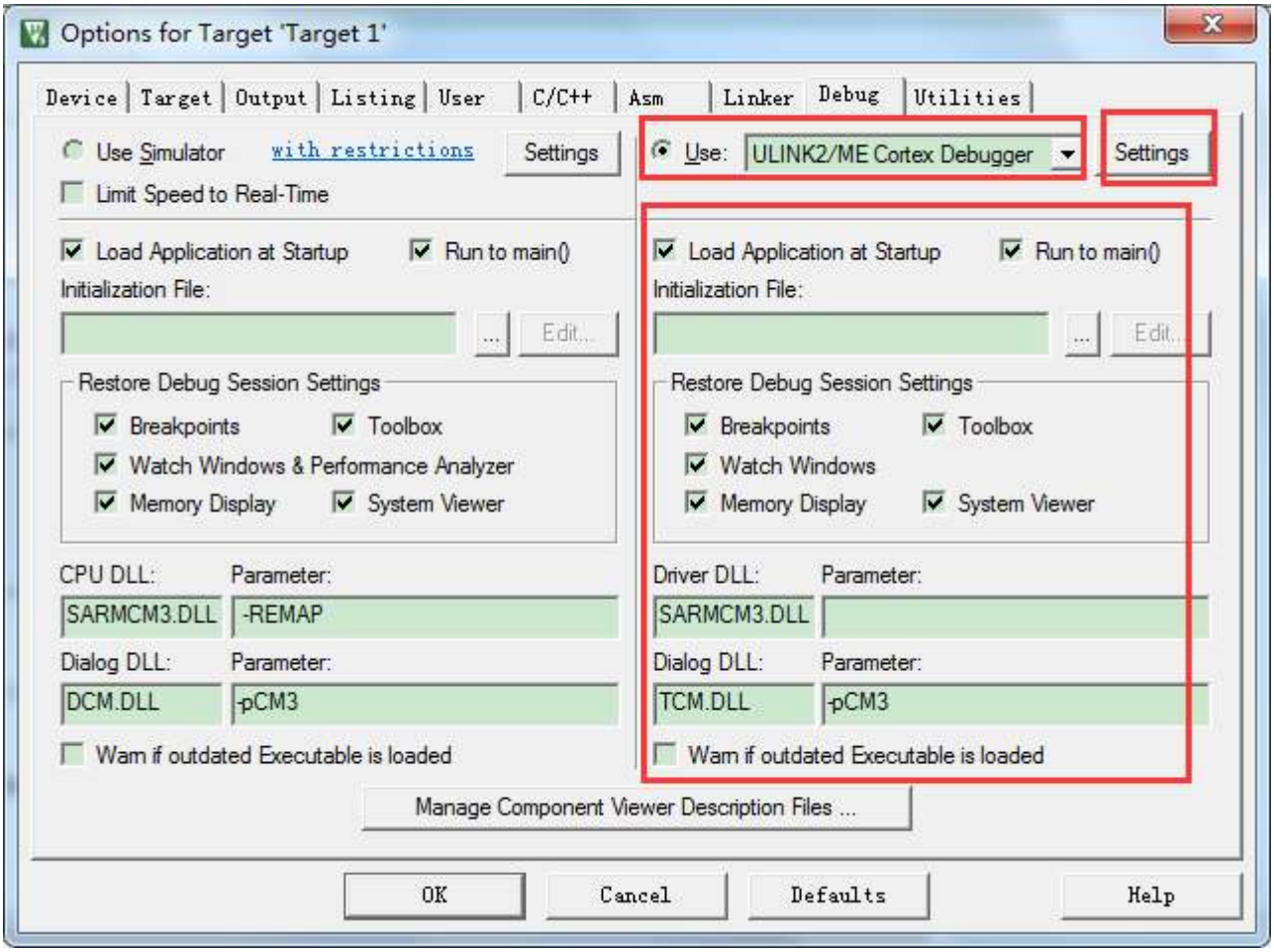
target 的名稱是可以隨意更改的，我們這裡改為 FLASH。

（2）點擊 Project->Options for Target Flash...（也可以點擊魔術棒那個圖標）進行配置。首先對 Target 選項卡設置：

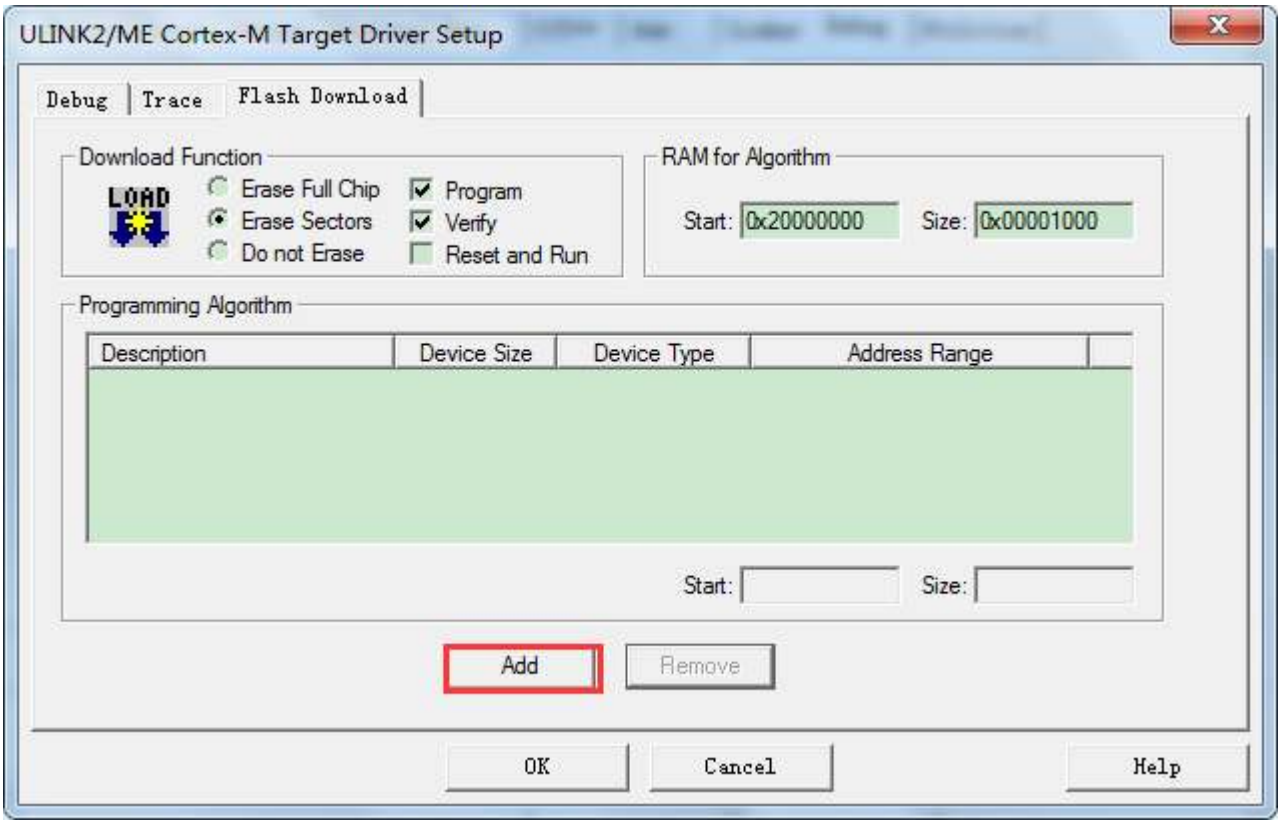


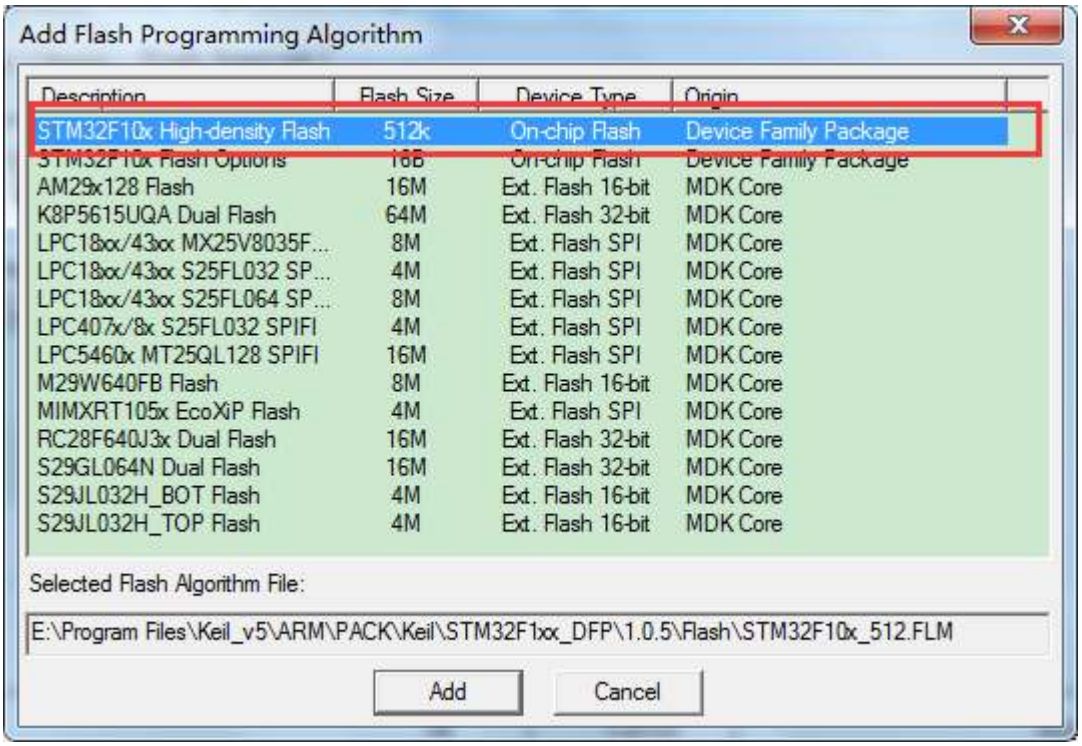
設置 IROM1 的起始地址為 0x8000000，大小為 0x80000，即 FLASH 的基地址與大小。設置 IRAM1 為 0x20000000，大小為 0x10000，即 SRAM 的基地址與大小。

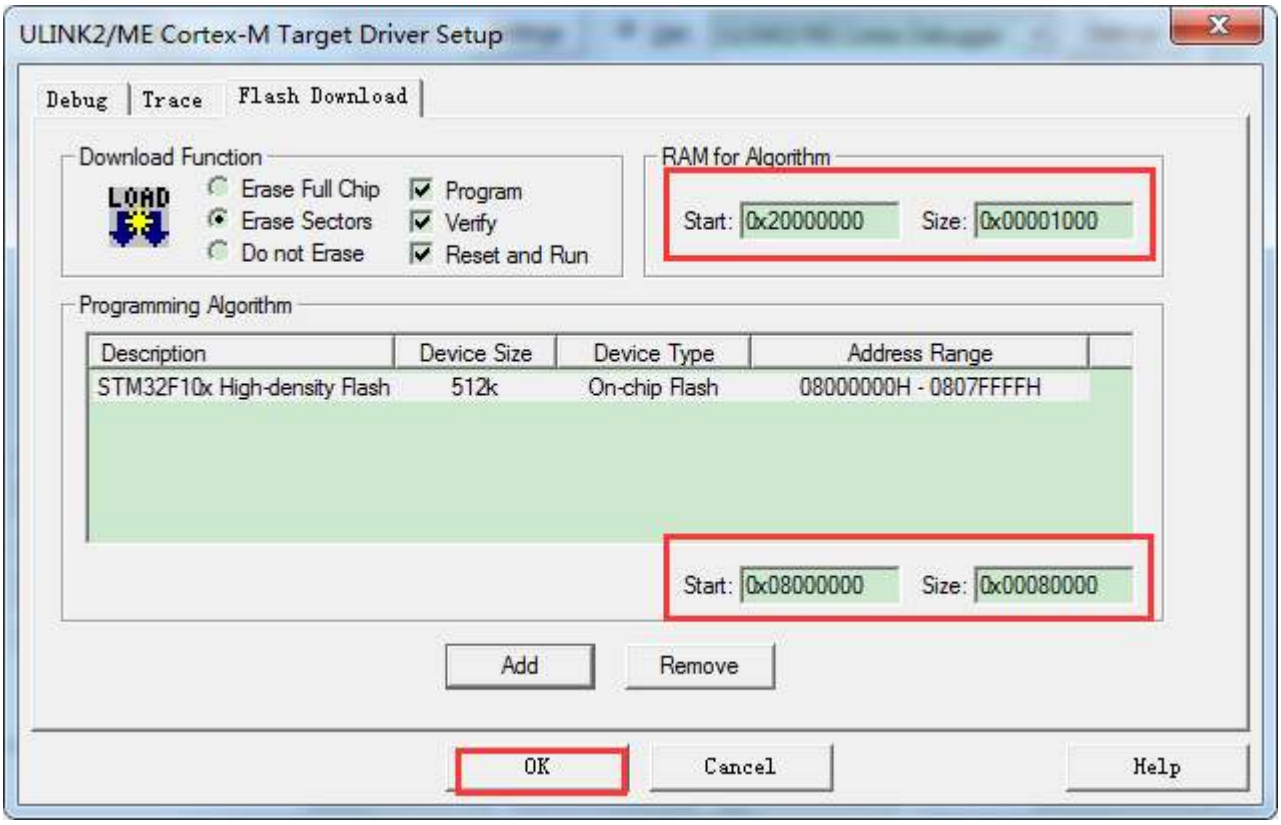
（3）Debug 選項設置：



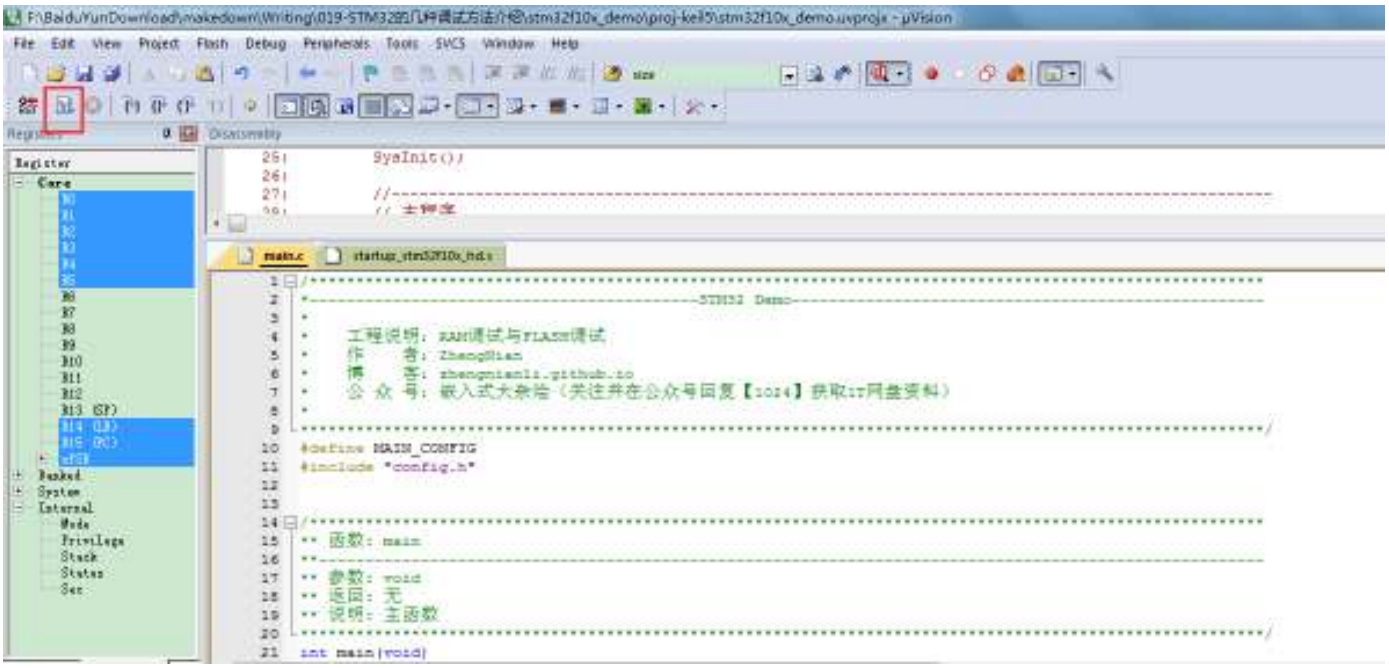
調試器根據實際進行選擇，我們這裡使用的調試器是 ULINK2。其它的按默認設置即可，然後點擊 Settings:


ULINK2/ME Cortex-M Target Driver Setup
Debug
Trace
Flash Download
Download Function
LOAD
Erase Full Chip
Erase Sectors
Do not Erase
Program
Verify
Reset and Run
RAM for Algorithm
Start: 0x20000000
Size: 0x00001000
Programming Algorithm
Description
Device Size
Device Type
Address Range
Start:
Size:
Add
Remove
OK
Cancel
Help

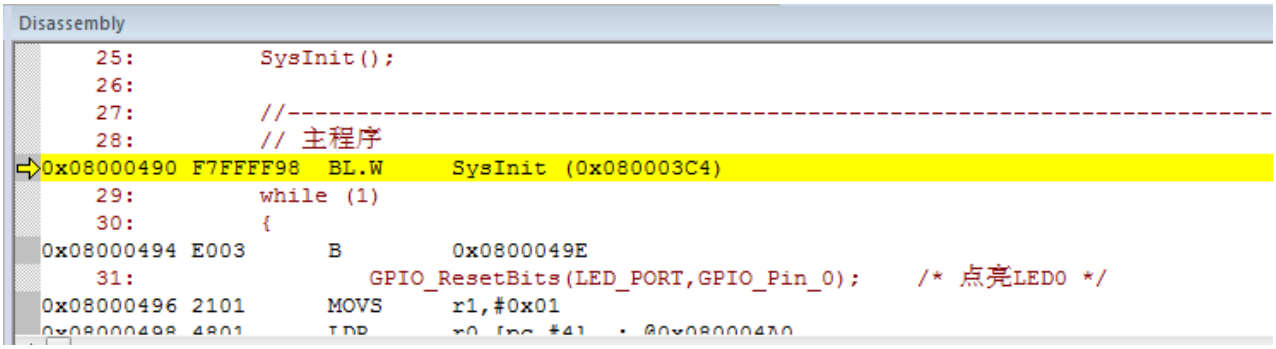

Add Flash Programming Algorithm
Description
Flash Size
Device Type
Origin
STM32F10x High-density Flash 512k On-chip Flash Device Family Package
STM32F10x Flash Options 16B On-chip Flash Device Family Package
AM29x128 Flash 16M Ext. Flash 16-bit MDK Core
K8P5615UQA Dual Flash 64M Ext. Flash 32-bit MDK Core
LPC18xx/43xx MX25V8035F... 8M Ext. Flash SPI MDK Core
LPC18xx/43xx S25FL032 SP... 4M Ext. Flash SPI MDK Core
LPC18xx/43xx S25FL064 SP... 8M Ext. Flash SPI MDK Core
LPC407x/8x S25FL032 SPIFI 4M Ext. Flash SPI MDK Core
LPC5460x MT25QL128 SPIFI 16M Ext. Flash SPI MDK Core
M29W640FB Flash 8M Ext. Flash 16-bit MDK Core
MIMXRT105x EcoXiP Flash 4M Ext. Flash SPI MDK Core
RC28F640J3x Dual Flash 16M Ext. Flash 32-bit MDK Core
S29GL064N Dual Flash 16M Ext. Flash 32-bit MDK Core
S29JL032H_BOT Flash 4M Ext. Flash 16-bit MDK Core
S29JL032H_TOP Flash 4M Ext. Flash 16-bit MDK Core
Selected Flash Algorithm File:
E:\Program Files\Keil_v5\ARM\PACK\Keil\STM32F1xx_DFP\1.0.5\Flash\STM32F10x_512.FLM
Add
Cancel



（4）編譯，然後按 Ctrl+F5 進入調試界面：



然後點擊全速運行：



在 Disassembly 窗口中可看到地址為 0x0800xxxx，說明代碼燒進了 FLASH 中，這時候就可以像使用其他 C 語言 IDE 調試 C 語言程序一樣打斷點、單步運行我們的 STM32 程序啦。

在 SRAM 中進行硬件仿真調試

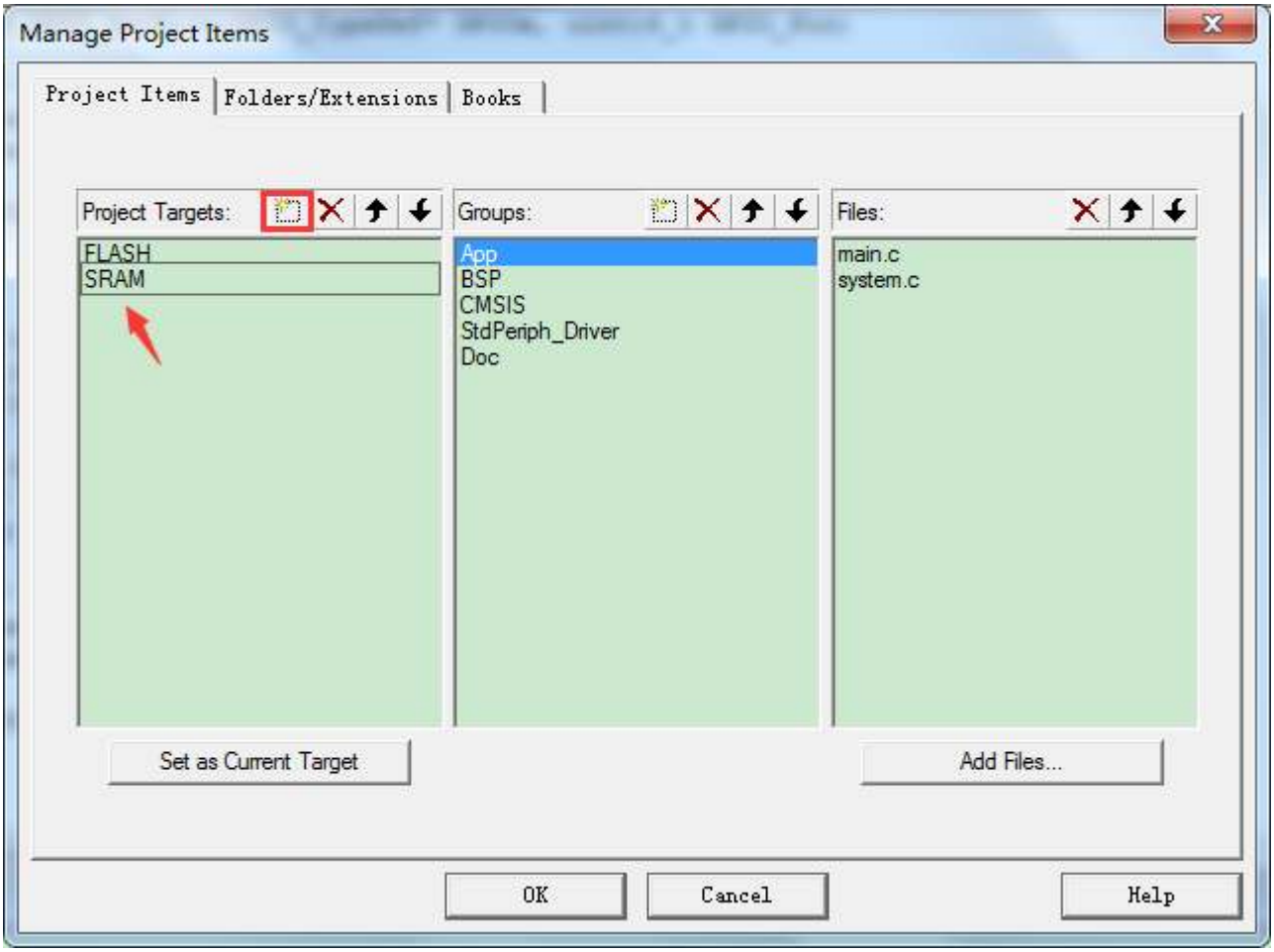
在 SRAM 的仿真調試配置比 FLASH 中的配置要麻煩一點，我配置的時候遇到不少問題~

1、硬件設置

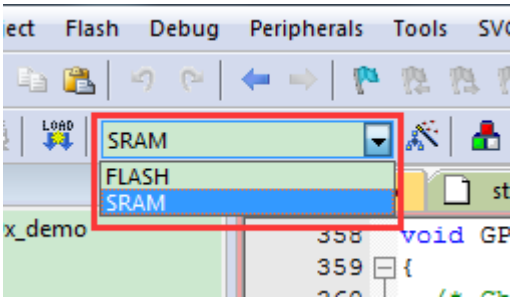
BOOT0 配置為 1，BOOT1 配置為 1。

2、keil 設置

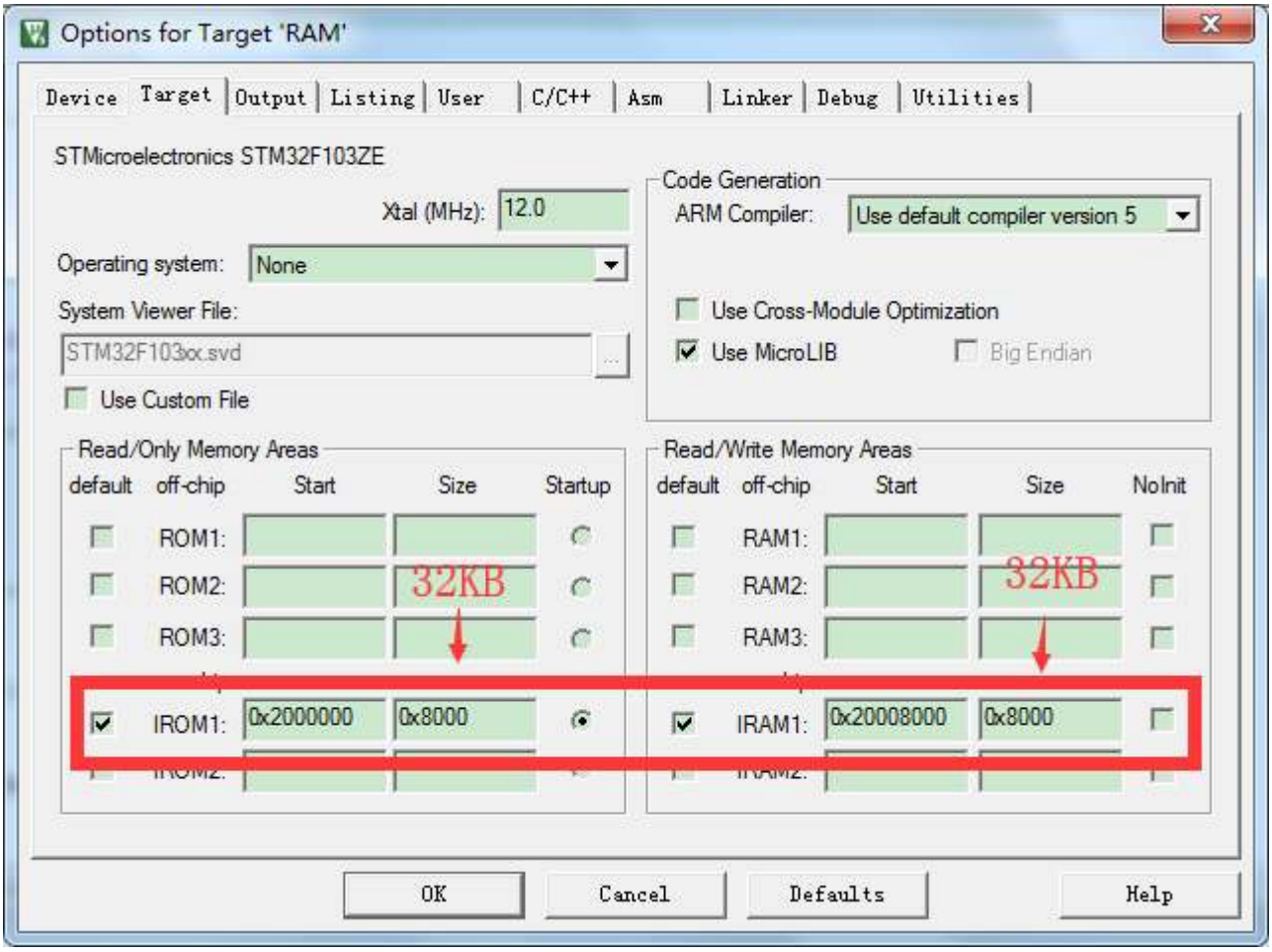
（1）新建一個 target，並修改名稱為 SRAM：



（2）切换至 SRAM Target：

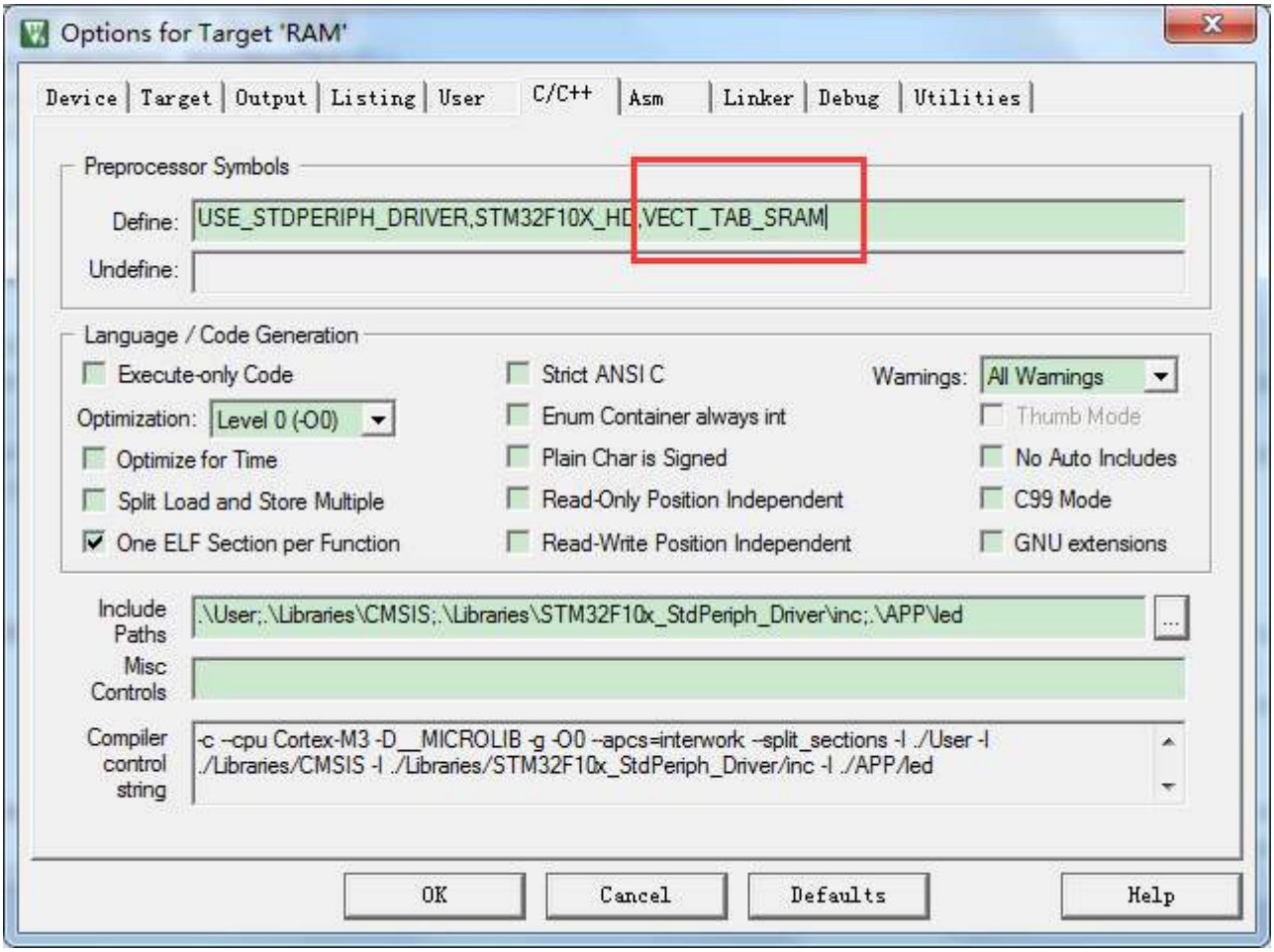


（3）點擊 Project->Options for Target SRAM ...（也可以點擊魔術棒那個圖標）Target 選項卡設置：



設置 IROM1 的起始地址為 0x2000000，大小為 0x8000（32KB）；設置 IRAM1 的起始地址為 0x2008000，大小為 0x8000（32KB）。即把 64KB 的 SRAM 分為 32KB 的 FLASH（當然這是 SRAM 虛擬出來的 FLASH，掉電易失）和 16KB 的 RAM。

（4）C/C++選項設置：



為什麼在 RAM 中調試要設置這個宏而在 FLASH 中調試卻不需要？這是因為我們的中斷向量表默認位於 FLASH 中，而此時我們要在 RAM 中進行調試，所以需要把中斷向量表拷貝到 RAM 中，相關代碼在 `system_stm32f10x.c` 的 `SystemInit` 函數中：

```
#ifdef VECT_TAB_SRAM
  SCB->VTOR = SRAM_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM. */
#else
  SCB->VTOR = FLASH_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal FLASH. */
#endif
}
```

其實 `system_stm32f10x.c` 文件中也有宏 `VECT_TAB_SRAM` 相關的代碼：

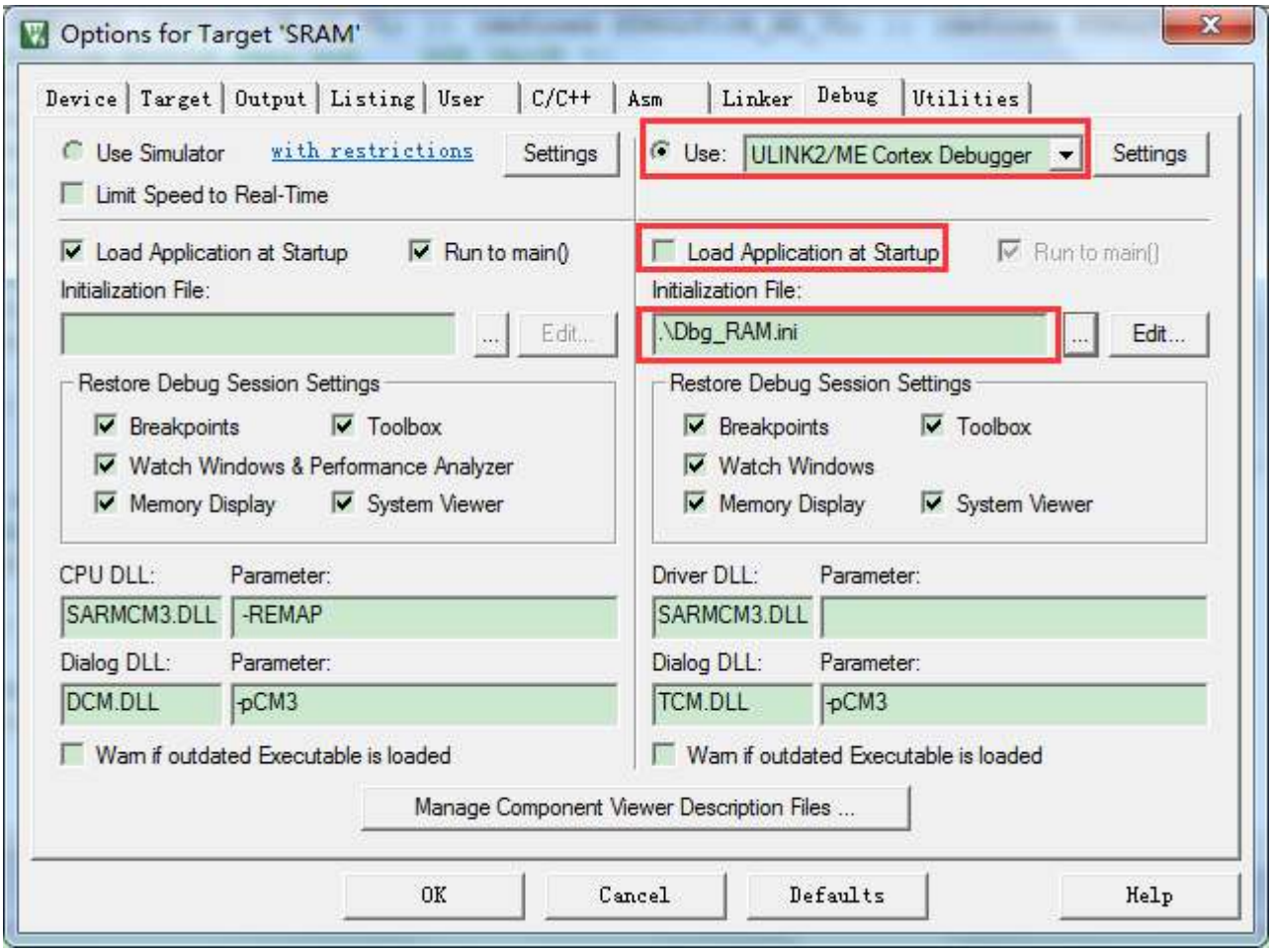
```
/*!< Uncomment the following line if you need to use external SRAM mounted
     on STM3210E-EVAL board (STM32 High density and XL-density devices) or on
     STM32100E-EVAL board (STM32 High-density value line devices) as data memory */
#if defined (STM32F10X_HD) || (defined STM32F10X_XL) || (defined STM32F10X_HD_VL)
/* #define DATA_IN_ExtSRAM */
#endif

/*!< Uncomment the following line if you need to relocate your vector Table in
     Internal SRAM. */
/* #define VECT_TAB_SRAM */
#define VECT_TAB_OFFSET  0x0 /*!< Vector Table base offset field.
                                  This value must be a multiple of 0x200. */
```

把這行代碼打開即可把中斷向量表拷貝到 RAM 中。但是這裡選擇在 C/C++選項選項裡添加宏，因為這樣可以保證 SRAM 版本與 FLASH 版本代碼的一致性。

（5）Debug 設置：

與在 FLASH 中調試不同的是，這裡需要加入.ini 文件：



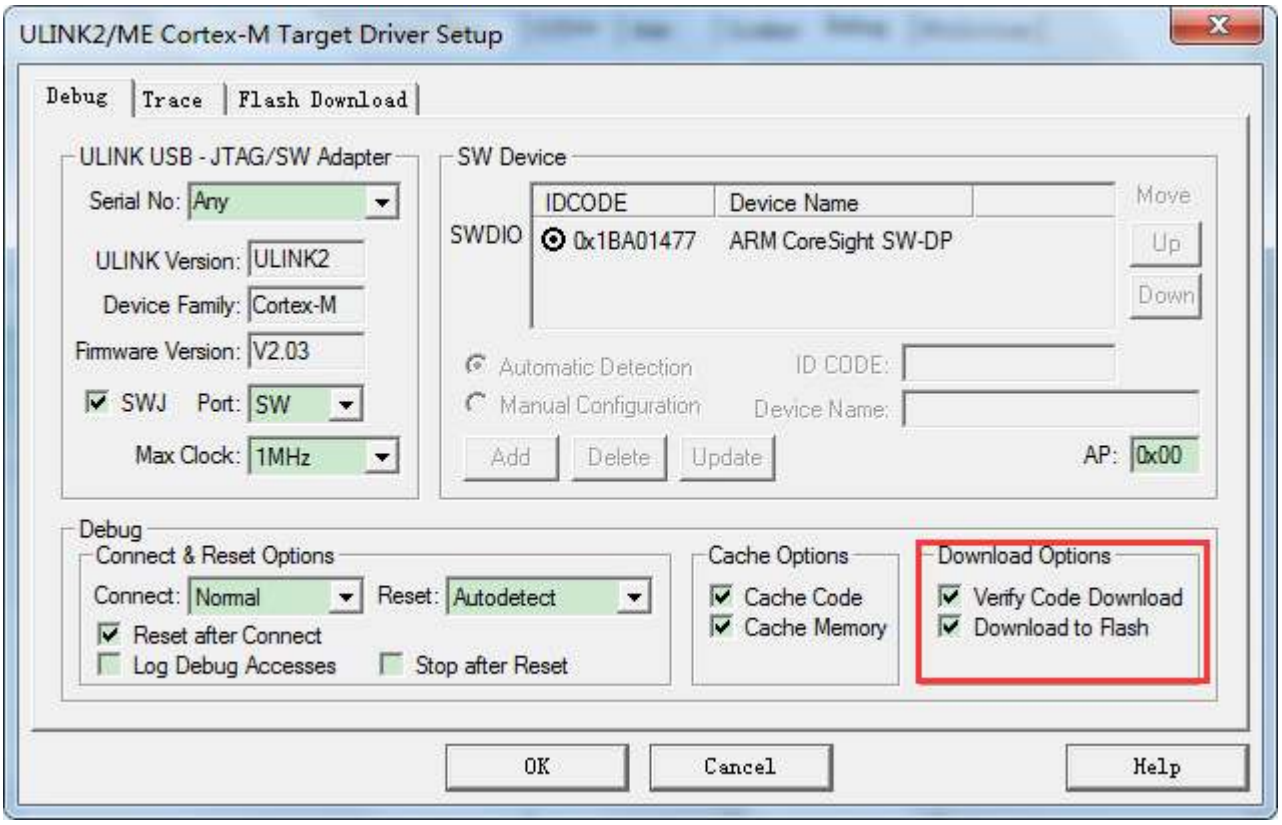
這個.ini 可以自己創建（也可以在芯片支持包裡找到），這裡我們建為 Dbg_RAM.ini。文件裡的內容如下：

```
1 FUNC void Setup (void) {
2   SP = _RDWORD(0x20000000);           // Setup Stack Pointer
3   PC = _RDWORD(0x20000004);           // Setup Program Counter
4   _WDWORD(0xE000ED08, 0x20000000);    // Setup Vector Table Offset Register
5 }
6
7 FUNC void OnResetExec (void)  {      // executes upon software RESET
8   Setup();                            // Setup for Running
9 }
10
11 load .\Objects\stm32f10x_demo.axf incremental
12
13 Setup();                            // Setup for Running
14
15 g, main
```

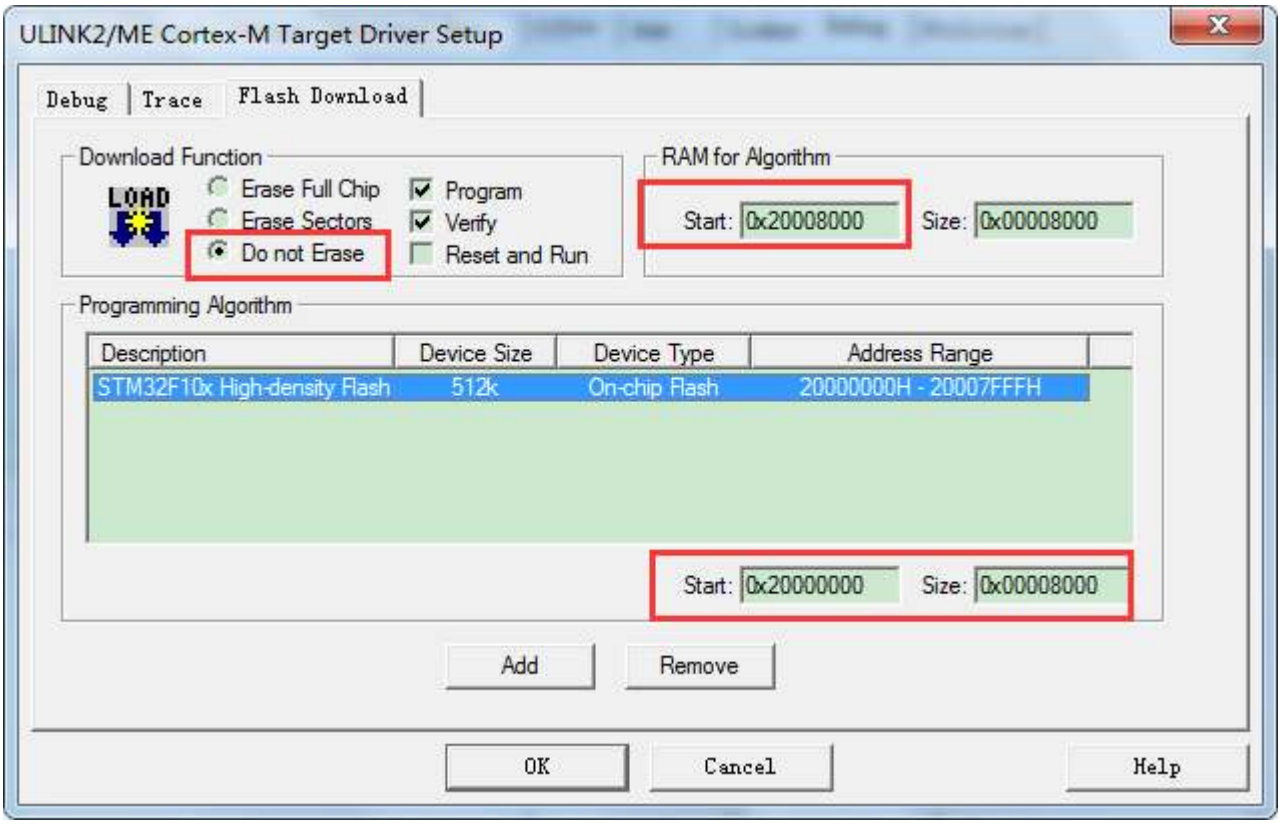
其中這裡的第 11 行是需要根據實際進行修改的，需要把工程編譯得出的.axf 格式文件的路徑及其文件名填到這裡。這裡因為我們這裡的的.ini 文件在.axf 的上一級目錄：

所以此處以./Objects 來表示。如果覺得麻煩的話，可以把.axf 文件與.ini 放在同一個目錄下。

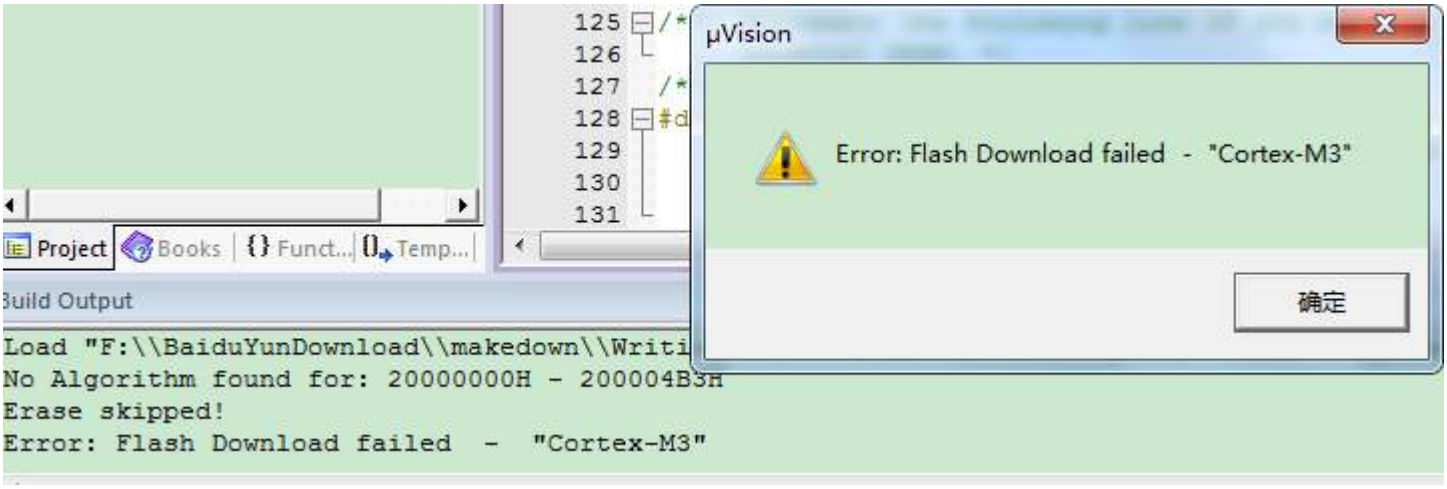
其它的按默認設置即可，然後點擊 Settings，並進行如下設置:



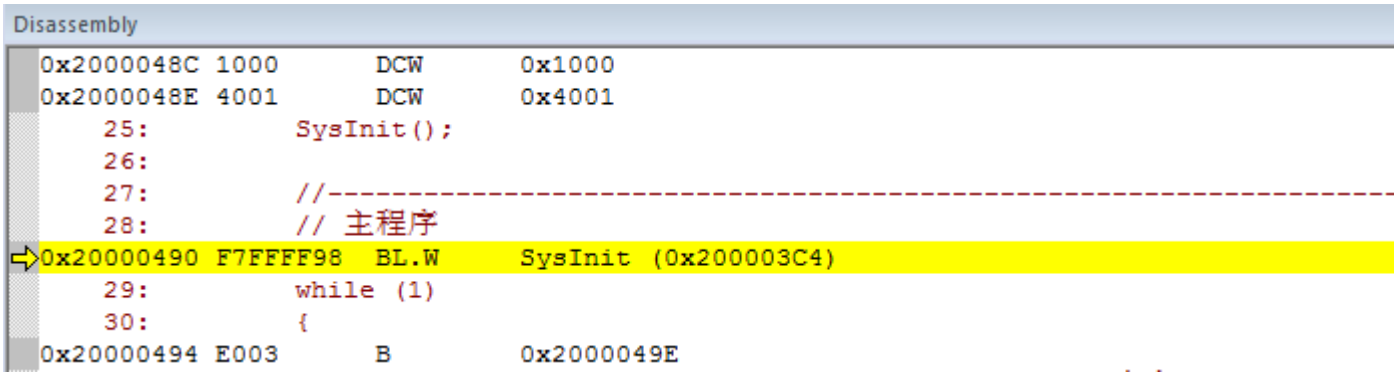
圖中我們需要勾選 Verify Code Download 及 Download to FLASH 選項，也就是說點擊調試按鈕後，本工程的程序會被下載到內部 SRAM 中，只有勾選了這兩個選項才能正常仿真。（至於為什麼 FLASH 版本的程序不需要勾選，不太清楚）。



Download Function 中的擦除選項配置為 Do not Erase。這是因為數據寫入到內部 SRAM 中不需要像 FLASH 那樣先擦除後寫入。 Programming Algorithm 的地址要與我們 Target 選項卡裡設置的地址一致，否則可能會出現如下錯誤：

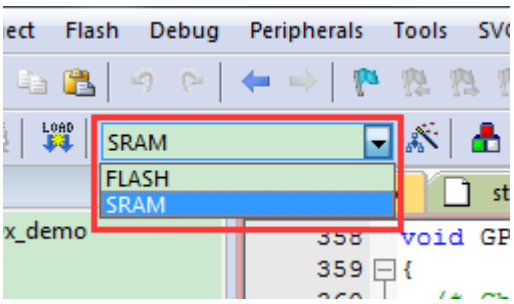


（6）編譯，然後按 Ctrl+F5 進入調試界面，然後點擊全速運行：



在 Disassembly 窗口中可看到地址為 0x2000xxxx，說明代碼燒進了 SRAM 中，這時候就可以像使用其他 C 語言 IDE 調試 C 語言程序一樣打斷點、單步運行我們的 STM32 程序啦。

以上就是在 FLASH 中調試與在 SRAM 中調試的設置方法，調試代碼時可以選擇 SRAM 版本的配置，調試完成再切換回 FLASH 版本的配置，把程序下載到 FLASH 中。切換方法：



在 RAM 中調試的優缺點

以下來自《【野火】零死角玩轉 STM32—F429 挑戰者 V2.pdf》。

優點：

1、載程序非常快。 RAM 存儲器的寫入速度比在內部 FLASH 中要快得多，且沒有擦除過程，因此在 RAM 上調試程序時程序幾乎是秒下的，對於需要頻繁改動代碼的調試過程，能節約很多時間，省去了煩人的擦除與寫入 FLASH 過程。另外，STM32 的內部 FLASH 可擦除次數為 1 萬次，雖然一般的調試過程都不會擦除這麼多次導致 FLASH 失效，但這確實也是一個考慮使用 RAM 的因素。

2、不改寫內部 FLASH 的原有程序。

3、對於內部 FLASH 被鎖定的芯片，可以把解鎖程序下載到 RAM 上，進行解鎖。

缺點：

1、存儲在 RAM 上的程序掉電後會丟失，不能像 FLASH 那樣保存。

2、SRAM 空間較小。

以上就是本次分享的關於 RAM 調試與 FLASH 調試的筆記，更多的相關原理、細節可查閱《【野火】零死角玩轉 STM32—F429 挑戰者 V2.pdf》。可在本公眾號嵌入式大雜燴聊天界面回復關鍵字：調試，進行獲取本筆記對應工程及《【野火】零死角玩轉 STM32—F429 挑戰者 V2.pdf》。本篇筆記如有錯誤歡迎指出！謝謝