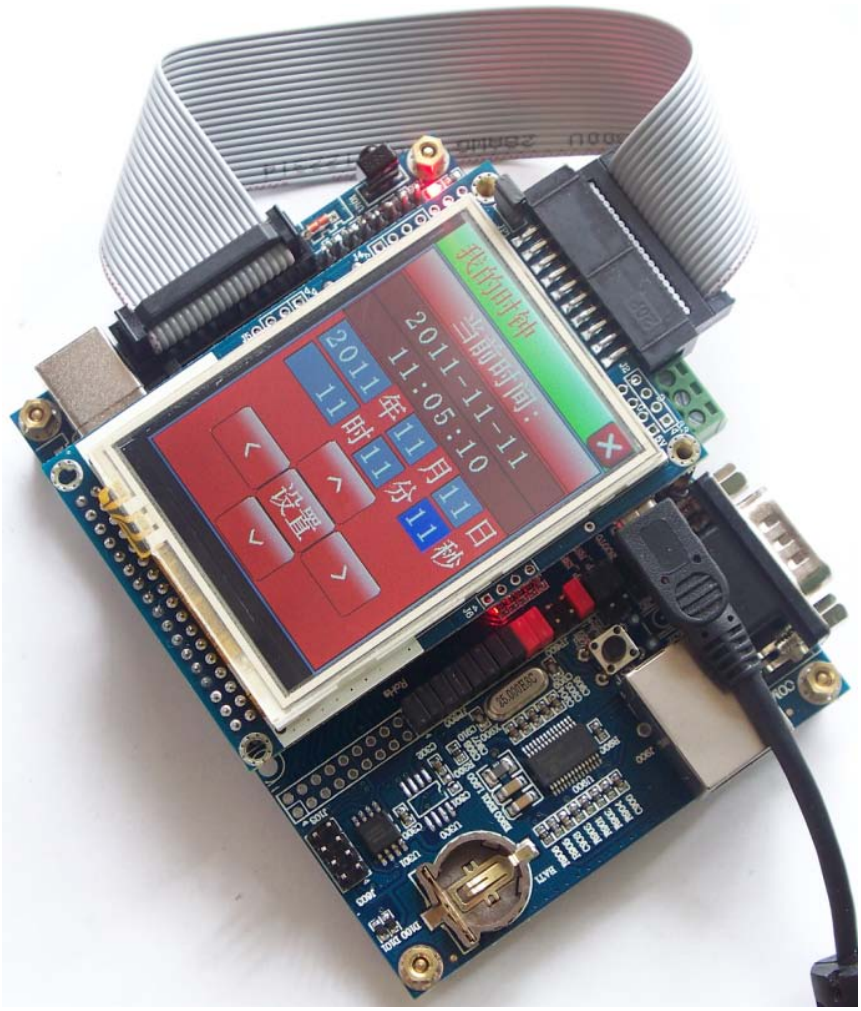
STM32+MDK 新手入门



本文是入门教程，高手请飘过…
介绍 MDK 入门使用，调试程序

QQ: 958664258
21IC 用户名：banhushui
交流平台：http://blog.21ic.com/user1/5817/index.html
Email: banhushui@163.com
淘宝店铺：http://shop58559908.taobao.com

半壶水

目录

1 启动选择

BOOT1	BOOT0	启动模式	说明
X	0	用户闪存存储器	用户闪存存储器被选为启动区域
0	1	系统存储器	系统存储器被选为启动区域（即进入 ISP 模式）
1	1	内嵌 RAM	内嵌 RAM 被选为启动区域

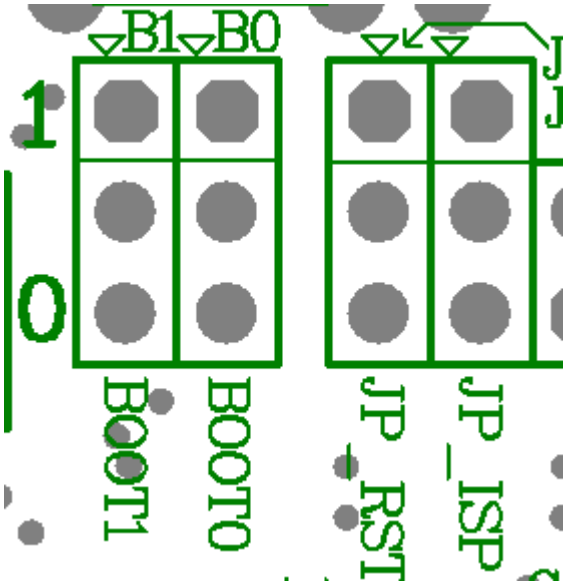
用户闪存存储器模式：上电运行你自己在 FLASH 中程序

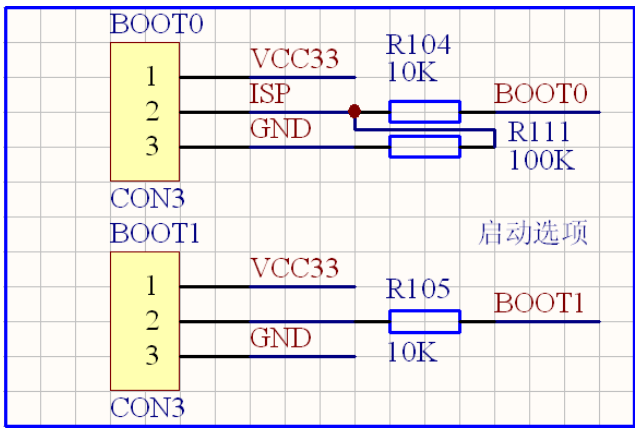
系统存储器模式：上电运行 ISP 程序，STM32 芯片出场是内部已经固化一段 ISP 程序用于在线编程，进入该模式你可以执行擦除芯片，编程芯片等操作

内嵌 RAM 模式：上电运行 RAM 中的程序

说明：

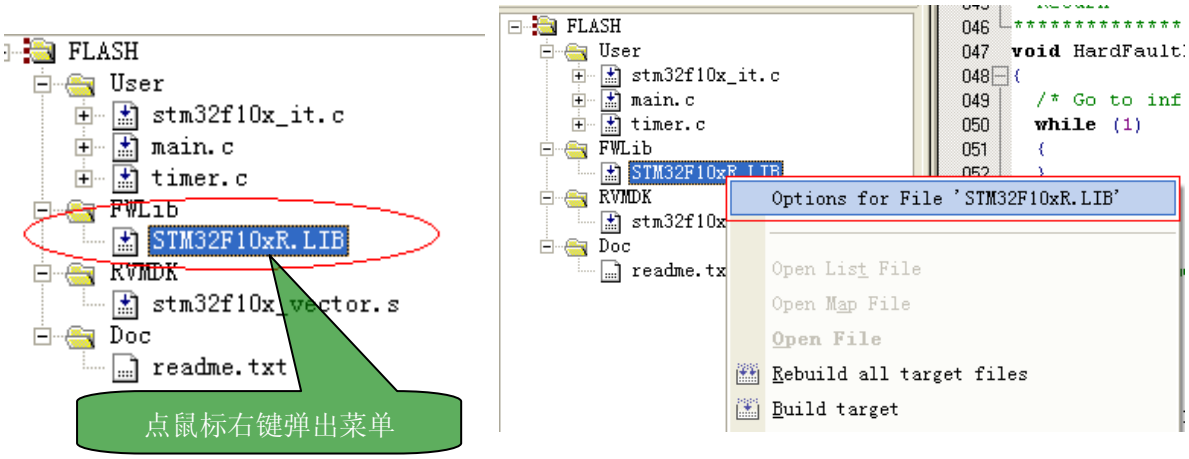
1. 出场默认设置 BOOT0=0，BOOT1=0。注意 BOOT0 已经通过电阻设置为 0
2. 由表可以看出要实现自动 ISP，设置 BOOT1=0 只需改变 BOOT0 状态就可以在用户模式和系统模式切换
3. 用仿真器调试时将忽略该设置
4. 很多人问 RAM 启动有什么作用，当我们在 RAM 中调试程序时，如果启动模式不是设置为 RAM 启动，也可以调试，但是当你按软件复位时，由于启动模式不是 RAM，那么你将不能继续调试程序，必须退出调试状态再重新进入调试才可以。如果你设置是 RAM 启动那么按软件复位后才能继续调试程序。我一般是懒得动跳线

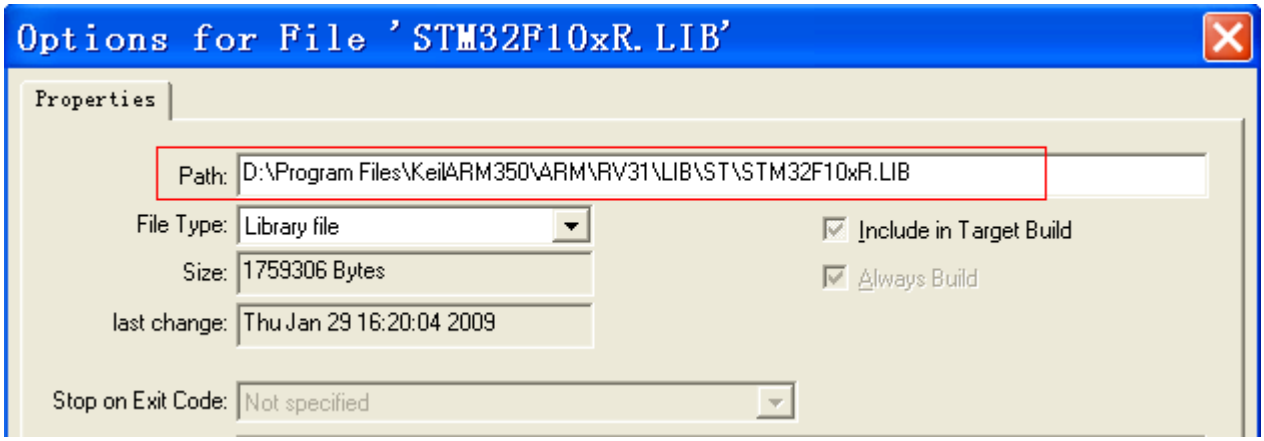




2 修改库文件地址

提供的例子使用了 MDK 的库，因为你的安装路径和我的可能不同，所以需要修改库文件路径





路径填你库文件 STM32F10xR.LIB 的路径
STM32F10xR.LIB 详细说明可以参考《STM32F101xx 与 STM32F103xx 固件函数库用户手册》

3　MDK 使用

本文关于 MDK 的说明均来自 KEIL 官方文档《UV3.chm》
另外关于 RTX，文件系统等参考《rlarm.chm》
这 2 个文件在安装路径的 HLP 文件下，另外光盘里也有中文版本
如果还没使用过 MDK 的同学，建议先看看《UV3.chm》，该文件让你很快熟悉 MDK 开发工具

3.1 MDK 常用工具及快捷方式

Project菜单和Project命令

Project 菜单	工具条	快捷键	功能描述
New Project...			创建一个新工程
Import µVision1 Project...			导入一个工程
Open Project...			打开一个工程
Close Project			关闭当前工程
Components, Environment, Books...			维护工程组件、配置工具环境及管理书
Select Device for Target			从设备库中选择CPU
Remove Item			从工程中移出组或文件
Options for Target			改变目标、组、文件的工具选项
		Alt+F7	改变当前目标的工具选项
	MCB251		选择当前目标
Build target		F7	翻译已修改的文件及编译应用
Rebuild all target files			重新翻译所有的源文件并编译应用
Translate...		Ctrl+F7	翻译当前文件
Stop Build			停止编译当前程序
1 - 9			打开最近使用的工程文件

Debug菜单和Debug命令

Debug 菜单	工具条	快捷键	功能描述
Start/Stop Debug Session		Ctrl+F5	启动或停止µVision3调试模式
Go		F5	运行到下一个活动断点
Step		F11	单步运行进入一个函数
Step Over		F10	单步运行跳过一个函数
Step Out of current Function		Ctrl+F11	从当前函数跳出
Run to Cursor Line			运行到当前行
Stop Running		ESC	停止运行
Breakpoints...			打开断点对话框
Insert/Remove Breakpoint			在当前行设置断点
Enable/Disable Breakpoint		Alt+F7	Enable/disable当前行的断点
Disable All Breakpoints			使程序中的所有断点无效
Kill All Breakpoints		F7	去除程序中的所有断点
Show Next Statement			显示下一条要执行的指令
Enable/Disable Trace Recording		Ctrl+F7	使能跟踪刻录
View Trace Records			浏览前面执行的指令
Execution Profiling			记录执行时间
Setup Logic Analyzer			打开逻辑分析仪对话框
Memory Map...			打开存储器映射对话框
Performance Analyzer...			打开性能分析仪对话框
Inline Assembly...			打开在线汇编对话框
Function Editor (Open Ini File)...			编辑调试函数及调试初始化文件

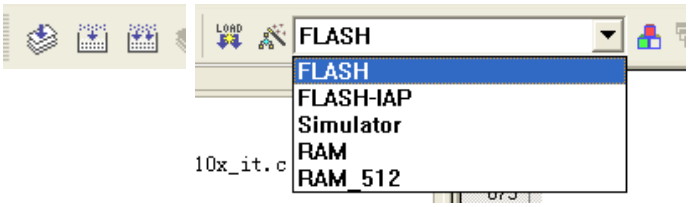
Debug 菜单	工具条	快捷键	功能描述
Download			按照配置下载到FLASH中

Debug 菜单	工具条	快捷键	功能描述
Reset CPU			重启CPU

Edit菜单和Edit命令

Edit 菜单	工具条	快捷键	功能描述
		Home	将光标移到当前行的开始
		End	将光标移到当前行的结束
		Ctrl+Home	将光标移到当前文件的开始
		Ctrl+End	将光标移到当前文件的结束
		Ctrl+Left Arrow	将光标移到当前单词的左侧
		Ctrl+Right Arrow	将光标移到当前单词的右侧
		Ctrl+A	选中当前文件中的所有内容
			把光标返回到执行'find'或'go to line'命令前的位置
			把光标移到到执行'find'或'go to line'命令后的位置
Undo		Ctrl+Z	撤消键入
Redo		Ctrl+Y	恢复键入
Cut		Ctrl+X	剪切
Copy		Ctrl+C	复制
Paste		Ctrl+V	粘贴
Indent Selected Text			向右缩进选定的文本
Unindent Selected Text			I向左缩进选定的文本
Toggle Bookmark		Ctrl+F2	在当前行设置标签
Goto Next Bookmark		F2	将光标移到下一个标签
Goto Previous Bookmark		Shift+F2	将光标移到前一个标签
Clear All Bookmarks		Ctrl+Shift+F2	消除所有的标签
Find	command	Ctrl+F	查找
		F3	重复向前查找
		Shift+F3	重复向后查找
		Ctrl+F3	在光标之下查找
Replace		Ctrl+H	替换
Find in Files...		Shift+Ctrl+F	在几个文件内查找
Incremental Find		Ctrl+I	增量查找
Outlining			有关源代码的命令
Advanced			编辑器命令
Configuration			改变着色、字体、快捷键

3.2 FLASH 中调试



下面以【SysTick_LED 点灯】例子来做介绍，例子都用 MDK+JLINK 说明

3.2.1. 选择 FLASH 项目编译

【FLASH】从 0x8000000 开始载入程序，

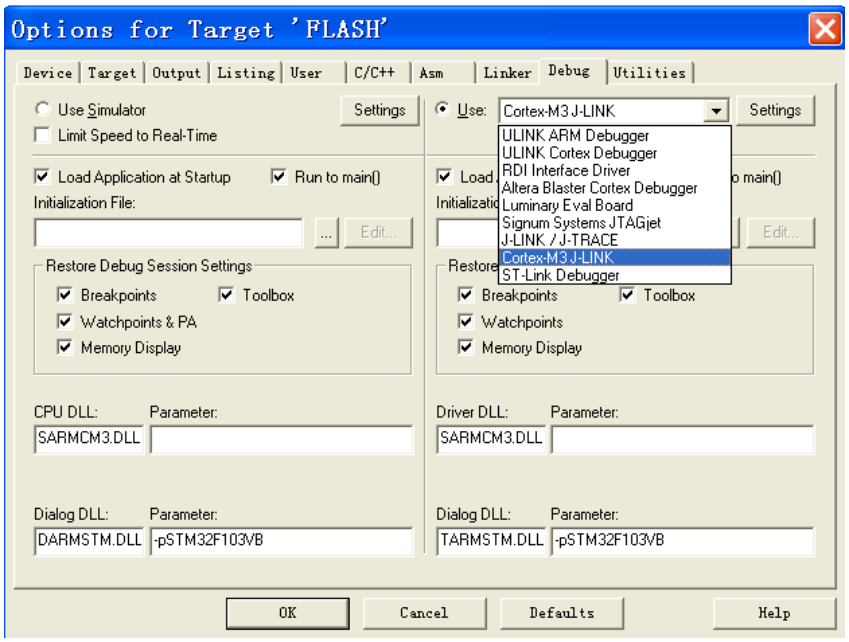
【Simulator】从 0x8000000 开始载入程序，该项是纯软件调试。例程部分外设是无法用软件调试的，请使用硬件仿真调试。

【FLASH-IAP】从 0x8002000 开始载入程序，

【RAM】从 0x20000000 开始载入程序，

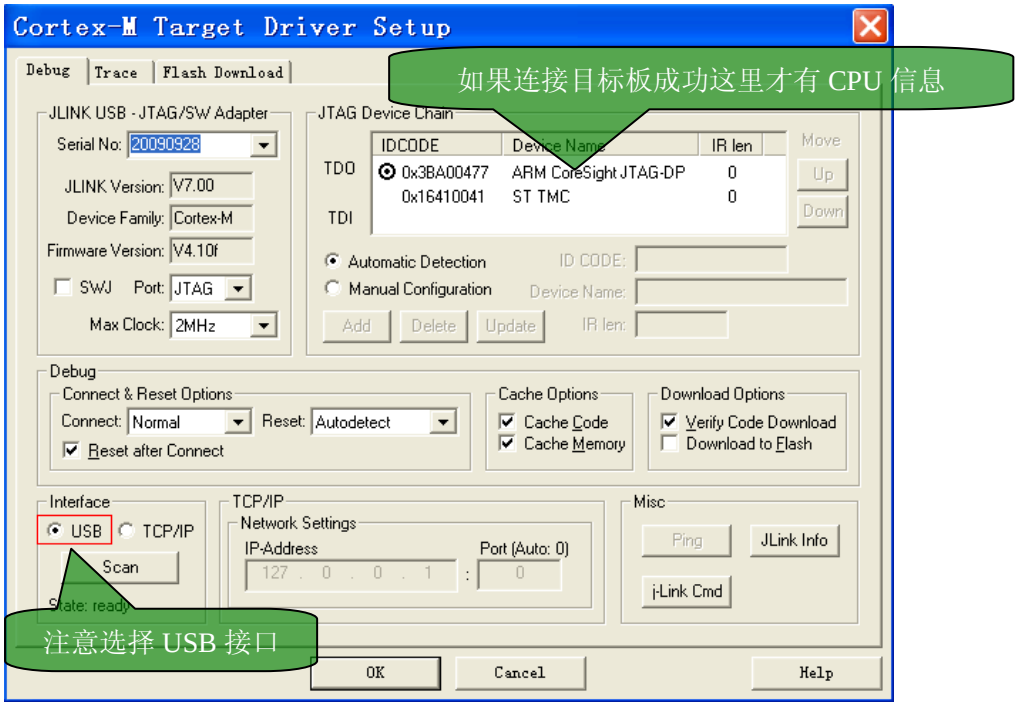
3.2.2. 配置项目

点击【项目配置】按钮弹出【项目配置】对话框，在[Debug]标签中选择【Cortex-M3 J-LINK】

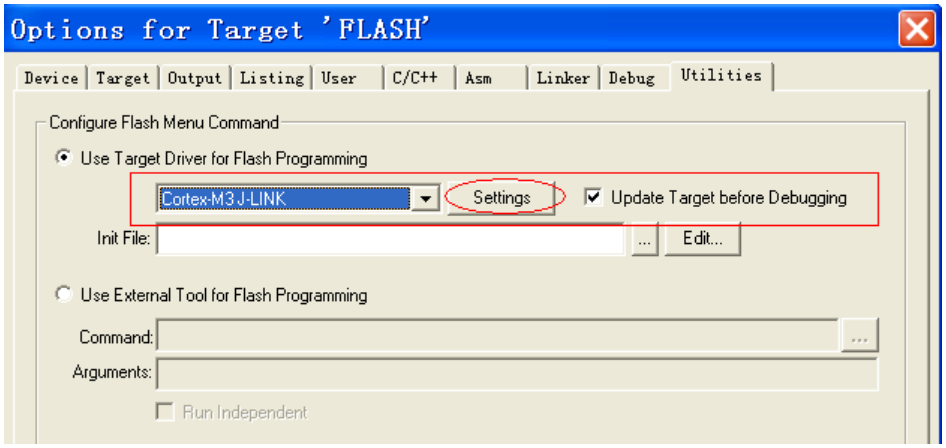


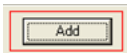
3.2.3. 设置 JLINK

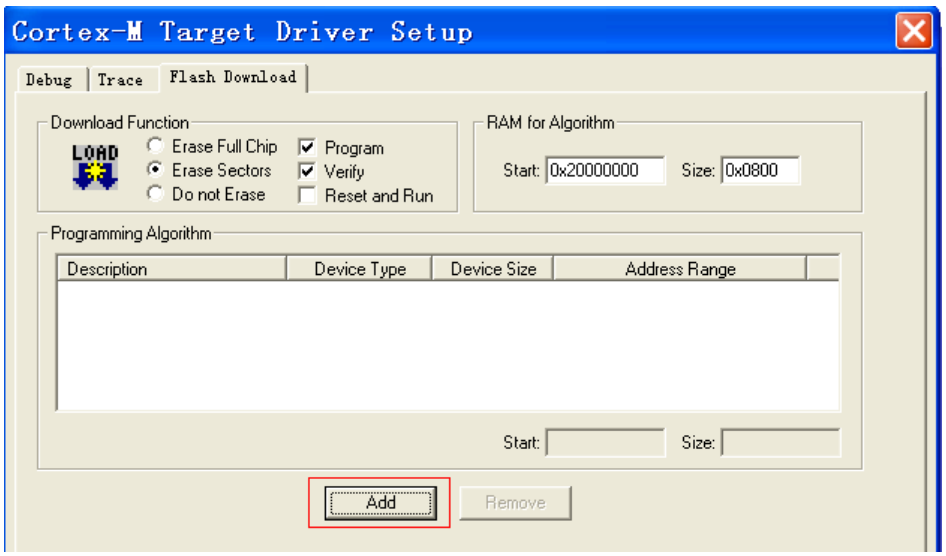
点击[settings]弹出设置对话框

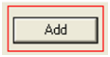


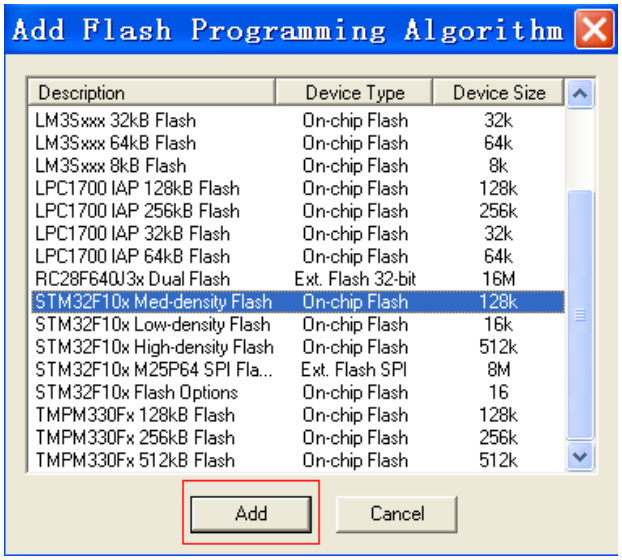
继续选择[Utilities]标签，同样选择【Cortex-M3 J-LINK】点击[Settings]



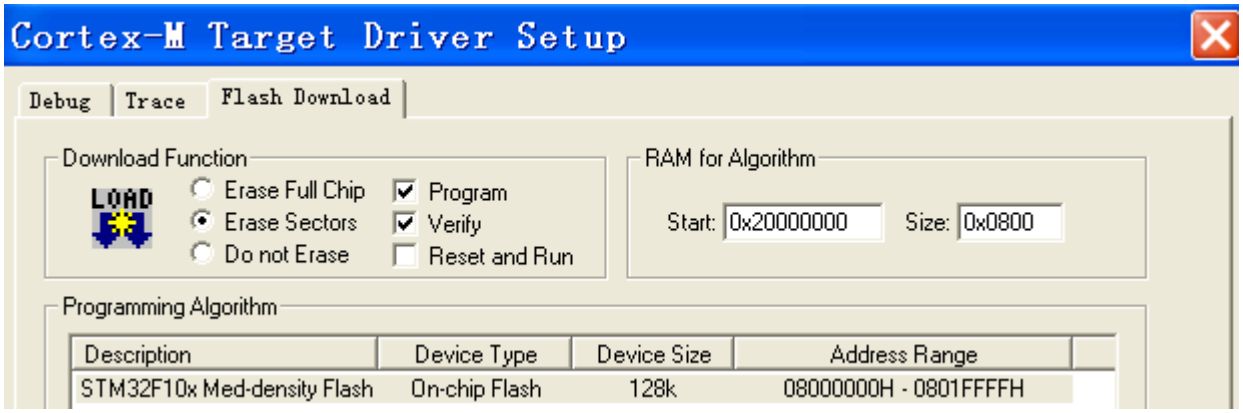
弹出目标板 CPU 加载对话框，点击[Add] 



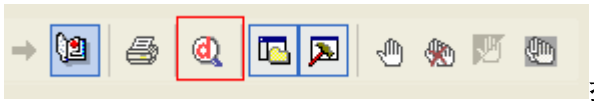
选择实际使用的 CPU 系列后，点击[Add] 



芯片加载成功



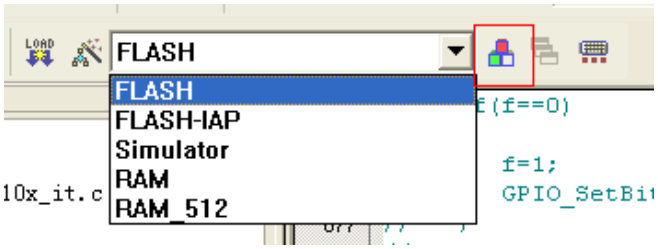
3.2.4. 仿真调试



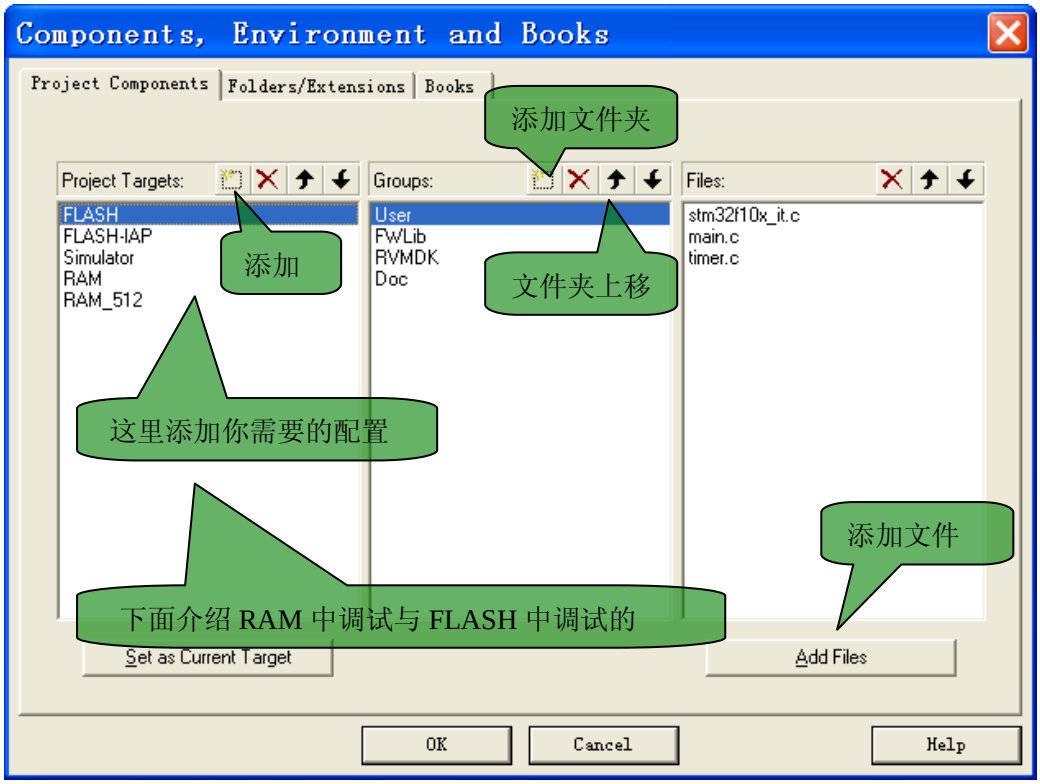
按下调试按钮进入调试状态

3.3 项目配置说明

有同学说自己新建的项目没有右边窗口选项，实际这是需要根据需求自己添加的，比如我提供的例子中一般包含在 FLASH，RAM 中调试，同时支持 IAP 功能等，实际这些都是通过不同配置实现的



点击  ，弹出如下对话框

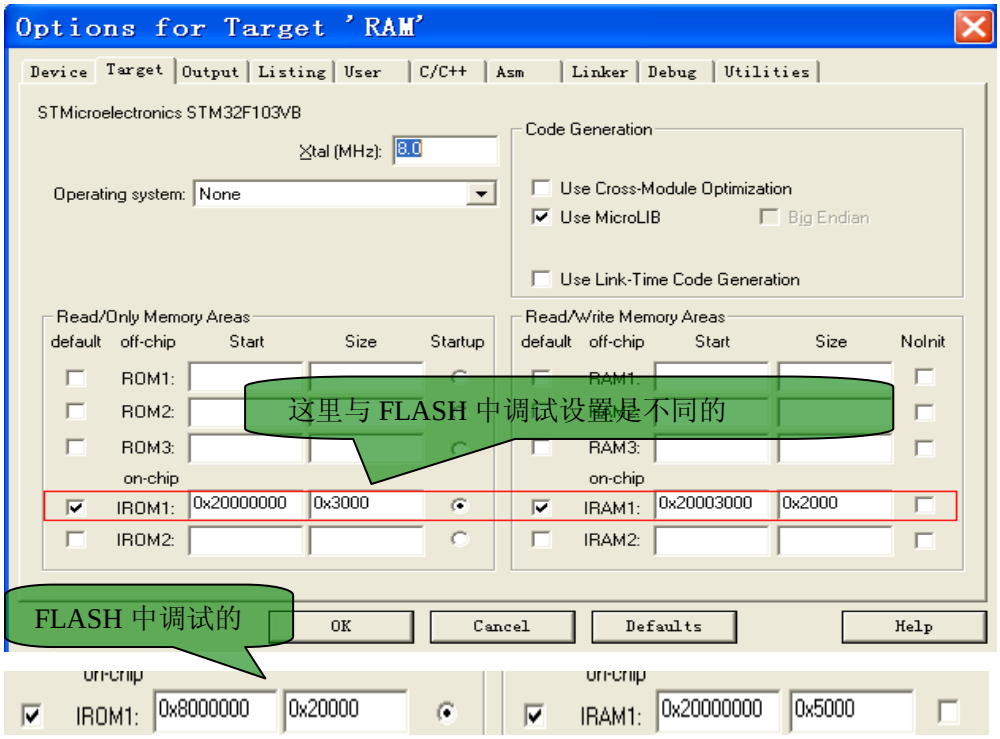


下面介绍 RAM 中调试，请注意与 FLASH 中调试配置的区别，那么你就知道该功能的具体作用了

3.4. 在 RAM 中调试

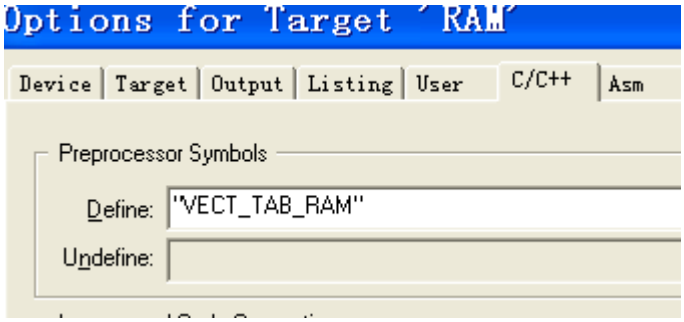
STM32 数据手册标明 FLASH 寿命是 1000 次，许多人都对这点耿耿于怀，其实大可不必。在做本开发板之前

我之前做过很多 STM32 的项目，从一开始根本就没用仿真器，基本都软件仿真，ISP，IAP 搞定。当然如果你觉得自己能力有限，那么 RAM 调试将帮你大忙，而且 RAM 中调试下载时比 FLASH 中调试快很多。本开发板提供的绝大部分例程都是在 RAM 中调试完成的，在项目也有 RAM 调试模板。



说明：IROM　IRAM 设置根据实际使用芯片 RAM 大小设置，本例是以 20K RAM 的芯片设置的其中 IROM 大小是你的程序容量

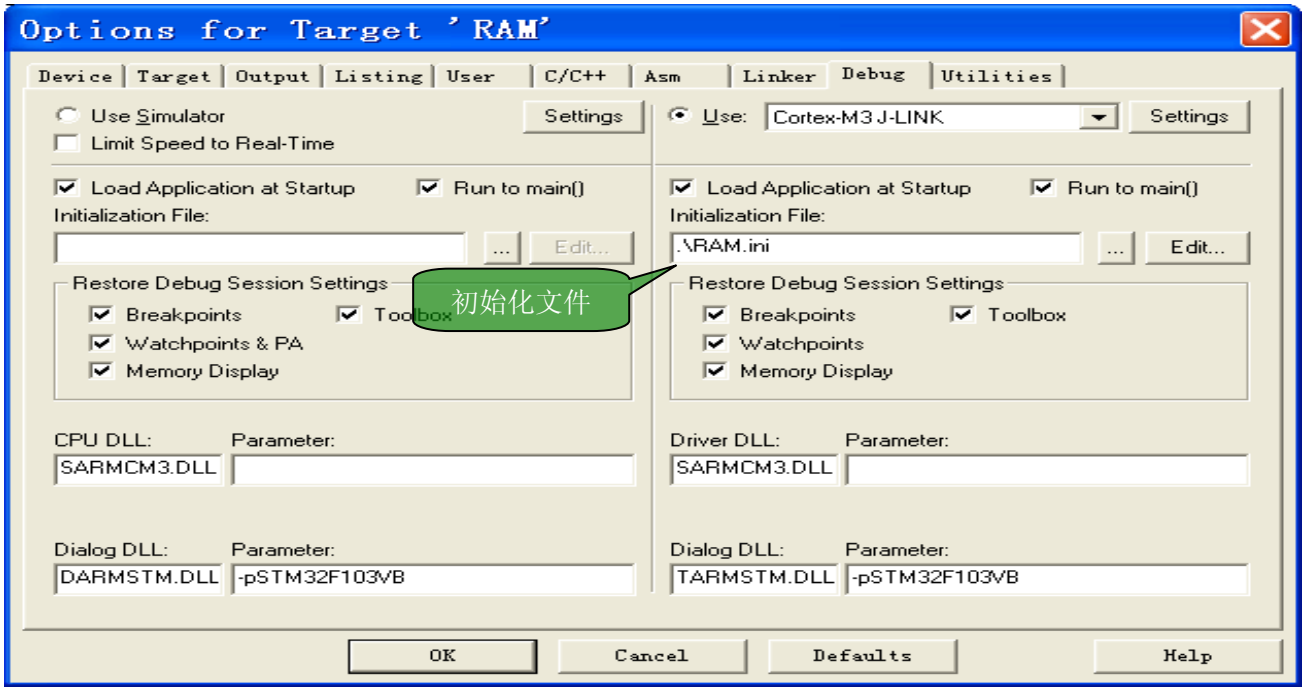
编译选项：该项配合代码重新定位中断向量表

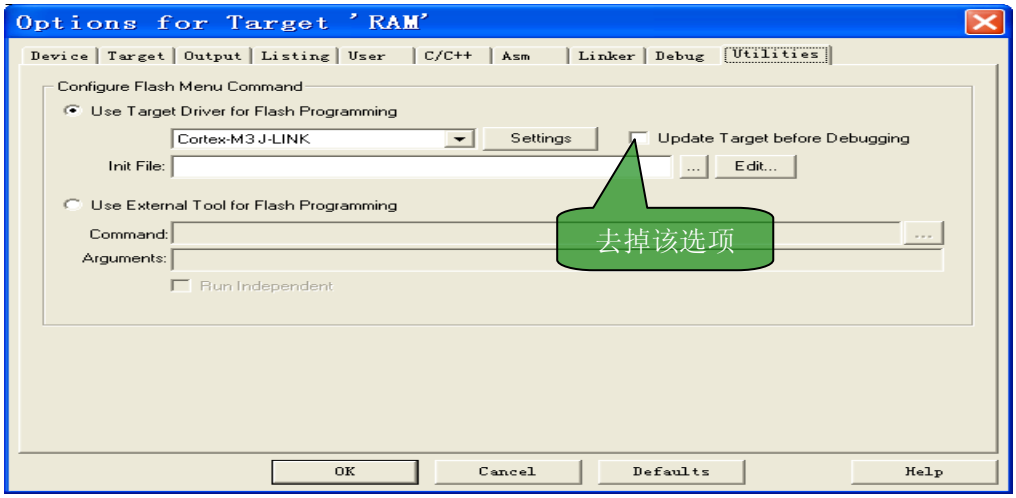


重定位中断向量表代码

```
#if defined (VECT_TAB_RAM)//RAM 中调试
  /* Set the Vector Table base location at 0x20000000 */
  NVIC_SetVectorTable(NVIC_VectTab_RAM, 0x0);
#elif defined(VECT_TAB_FLASH_IAP)//生成 IAP 文件
  NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x2000);
#else  /* VECT_TAB_FLASH  */
  /* Set the Vector Table base location at 0x08000000 */
  NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x0); //FLASH 中调试
#endif
```

载入初始化文件

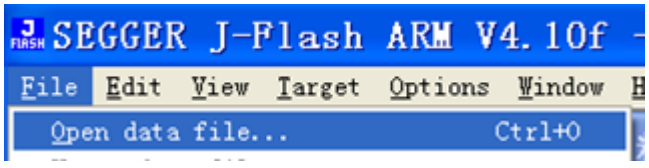




RAM 中调试大功告成，你再用 FLASH 中调试比较下，感觉 RAM 中调试速度是嗖嗖的，相当爽啊！让你都怀疑下载时到底有没完全下载下去。

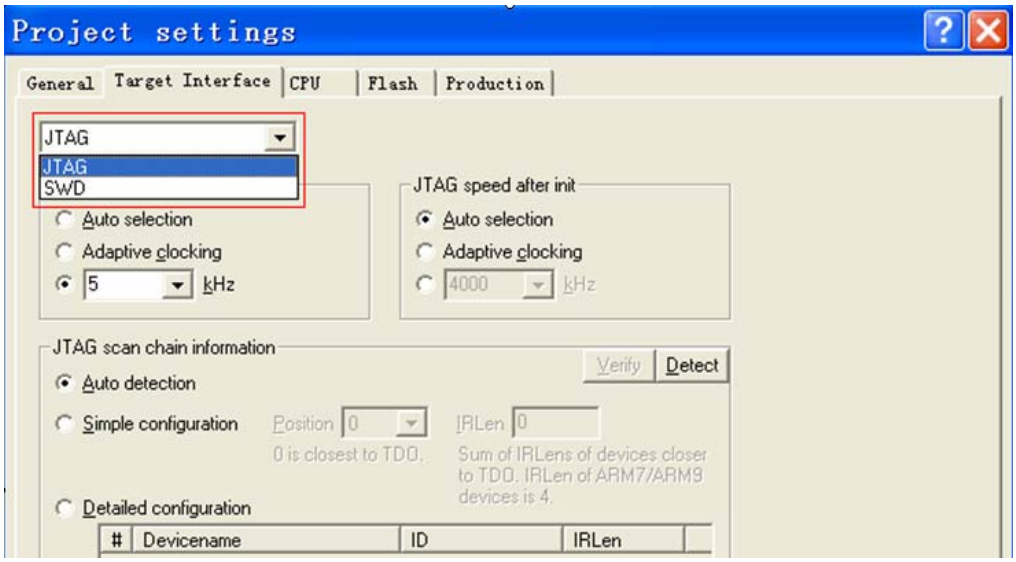
4 关于 JLINK

JLINK 官方文档《UM08001_JLinkARM.pdf》下载地址
http://www.segger.com/cms/admin/uploads/productDocs/UM08001_JLinkARM.pdf
使用 JLINK 最好到官方网站下载最新驱动
http://www.segger.com
JLINK 的软件工具使用可以参考《UM08001_JLinkARM.pdf》，我一般用的不是太多，偶尔用用 JFlash ARM 下面简单介绍用 JFlash ARM 编程 FLASH
1. 载入文件：选择菜单 File->Open data file…

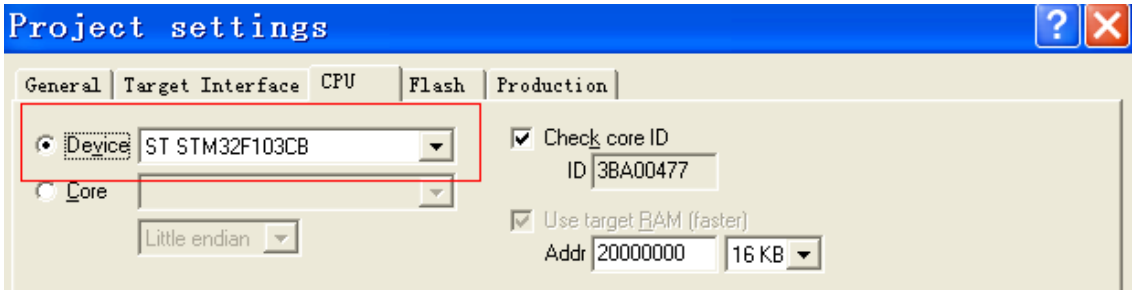


2. 项目设置：选择菜单 Options->Project settings…

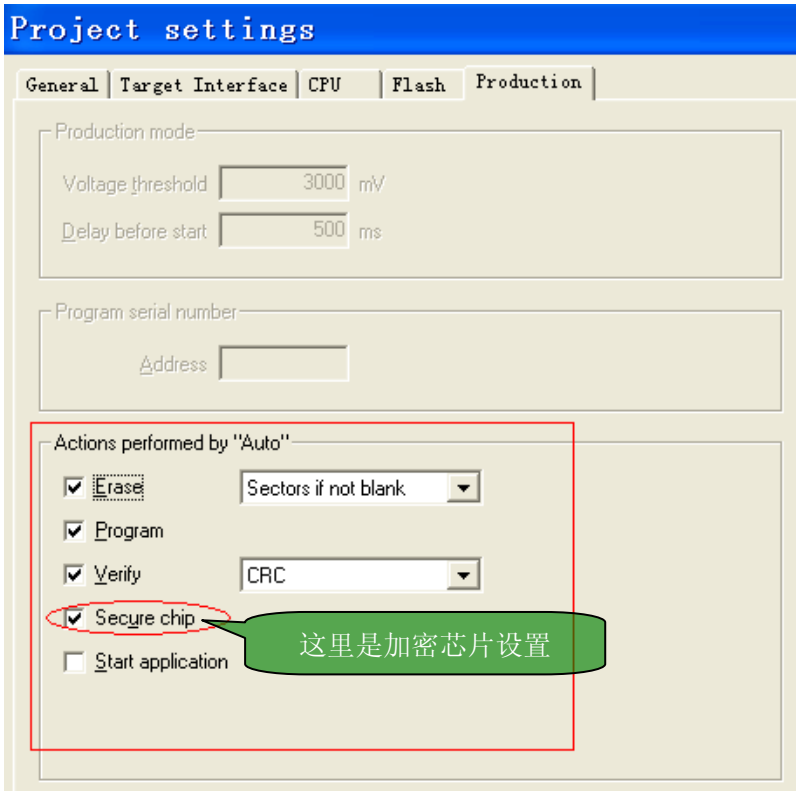
[Target Interface]选择 JTAG 或者 SWD 均可，具体根据你的链接方式



3. 选择 CPU，一定要选择对应型号，要不连不上目标板



4. 设置编程选项



5. 连接目标板：选择菜单 Target->Connect

下面是连接信息

```
Connecting ...
 - Connecting via USB to J-Link device 0
 - J-Link firmware: V1.20 (J-Link ARM-OB STM32 compiled Dec  3 2009 11:40:54)
 - JTAG speed: 5 kHz (Fixed)
 - Initializing CPU core (Init sequence) ...
    - Initialized successfully
 - JTAG speed: 2000 kHz (Auto)
 - Connected successfully
```

6. 编程：选择菜单 Target->Auto