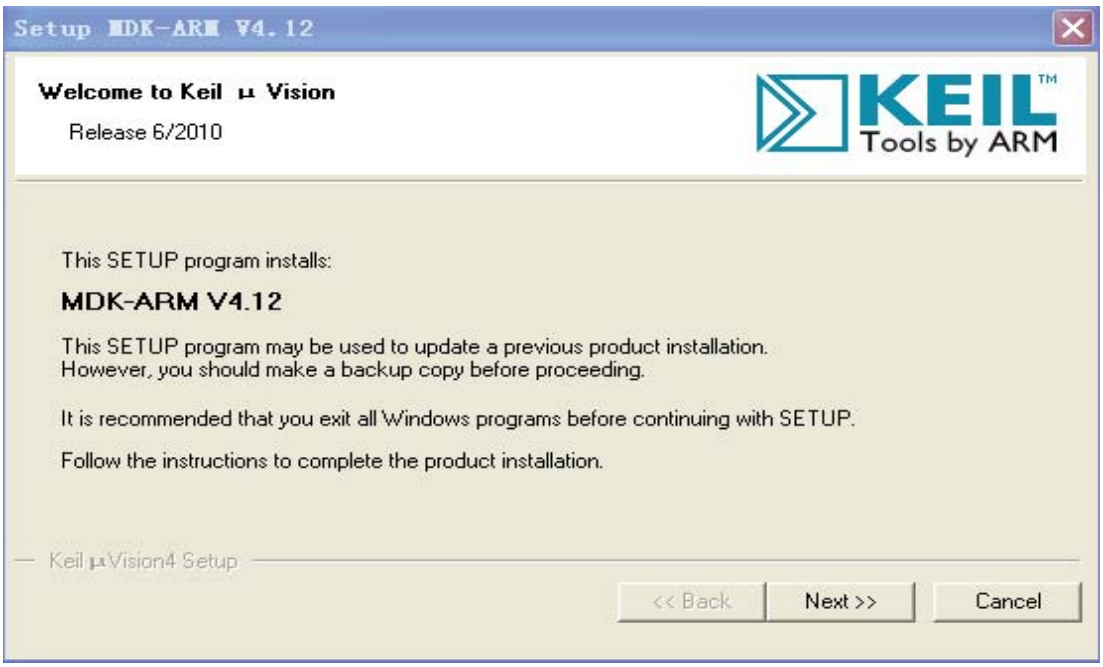
利用 STM32 的官方库在 RVMDK 中新建一个工程文件

作者	fire
E-Mail	firestm32@foxmail.com
QQ	313303034
博客	firestm32.blog.chinaunix.net
硬件平台	野火 STM32 开发板
库版本	ST3.0.0

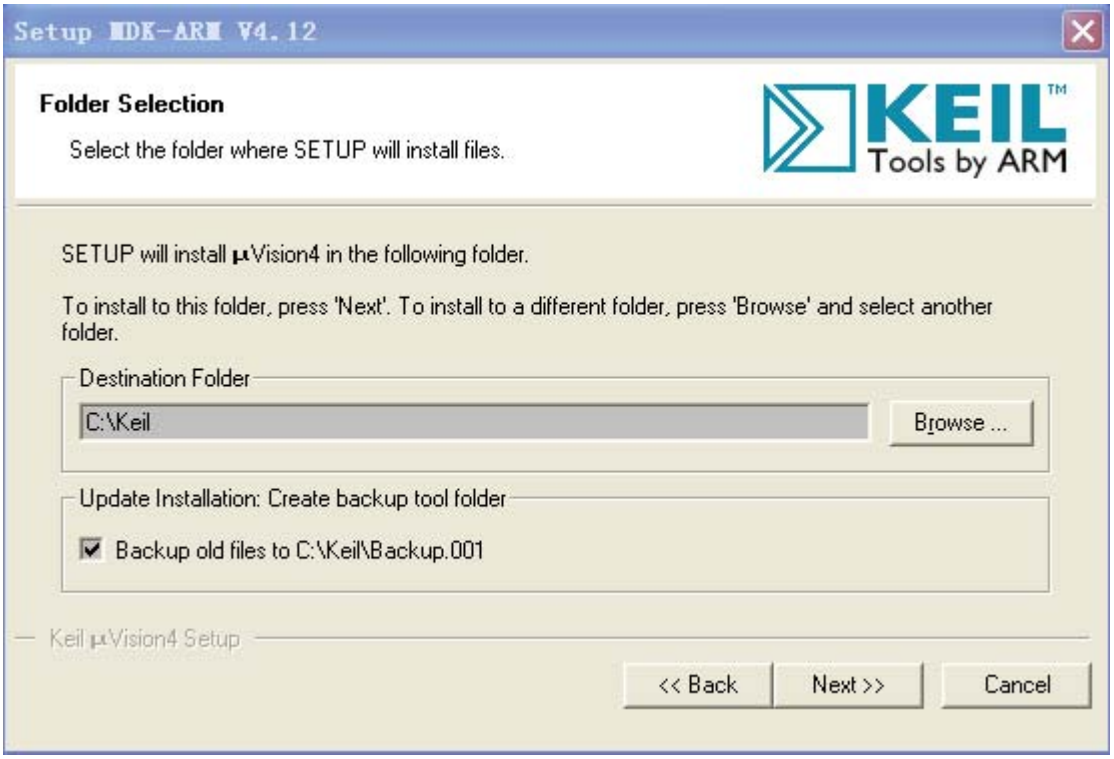
在新建工程之前我们先要把 RVMDK 这个软件安装好，这里用的版本是 V4.10，在安装完成之后可以在工具栏 help->about μVision 选项卡中查看到版本信息。μVision 是一个集代码编辑，编译，链接和下载于一体的集成开发环境（KDE），　　其支持我们常见的 arm7、arm9 和 arm 最新内核的 M3 系列，其前身就是 51 中的大名鼎鼎的 keil。　安装过程如下所示：

1、点击 Next。

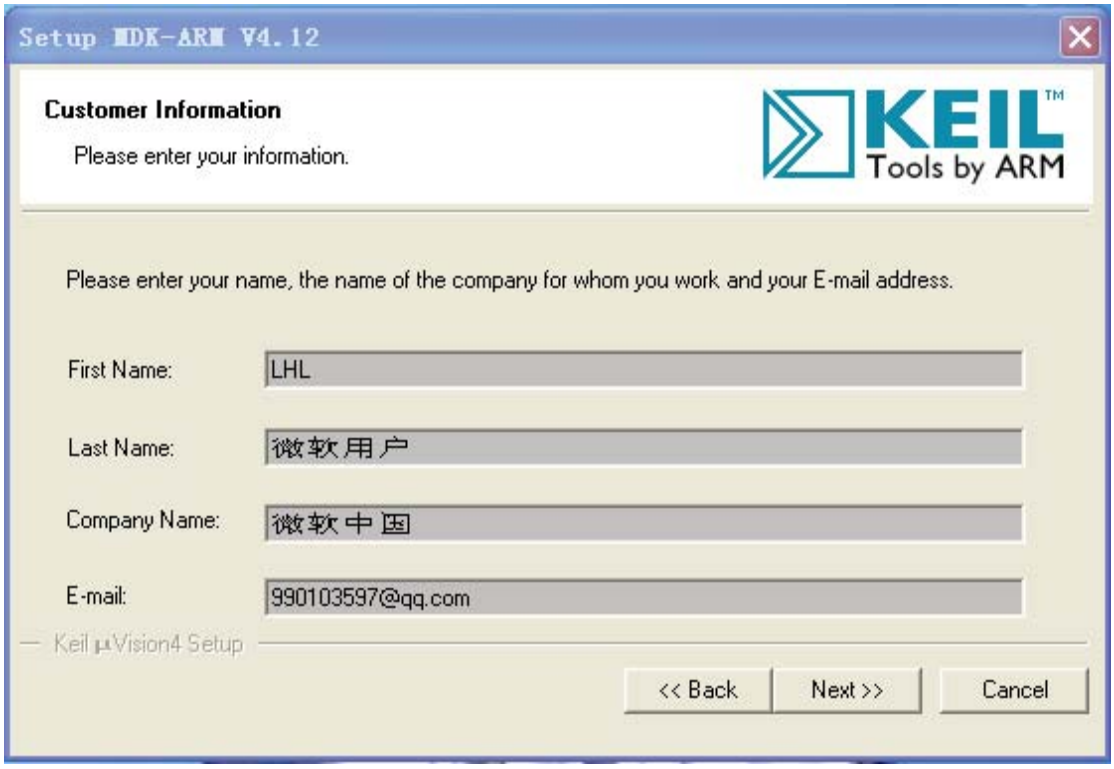


2、把勾勾上，点击 Next。

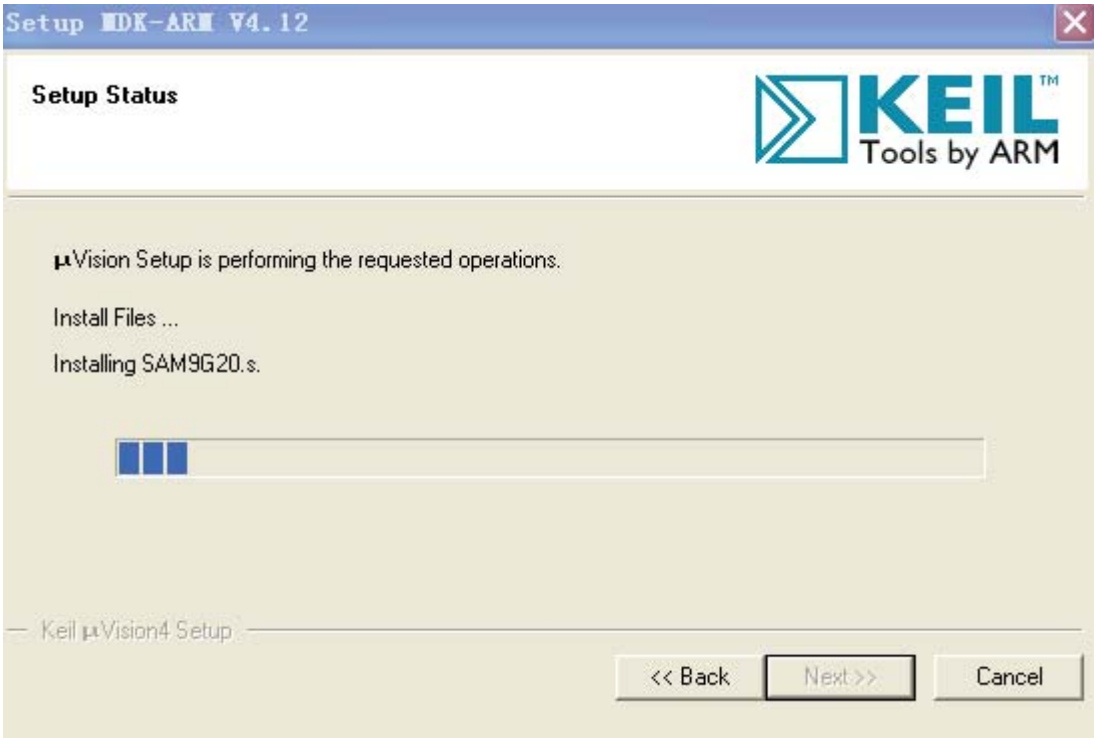
3、点击 Next，默认安装在 C:\keil 目录下。



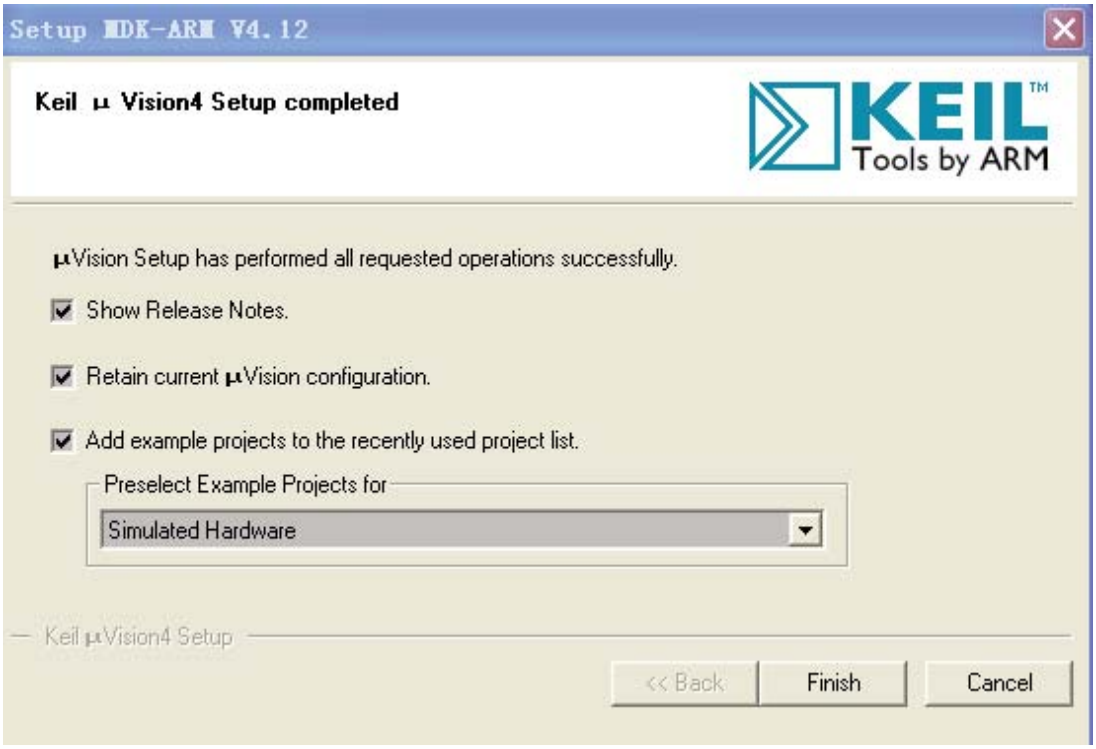
4、在用户名中填入名字（可随便写，可空格），在邮件地址那里填入邮件地址（可随便写，可空格），点击 Next。



5、正在安装，请耐心等待。



6、点击 Finish，安装完成。



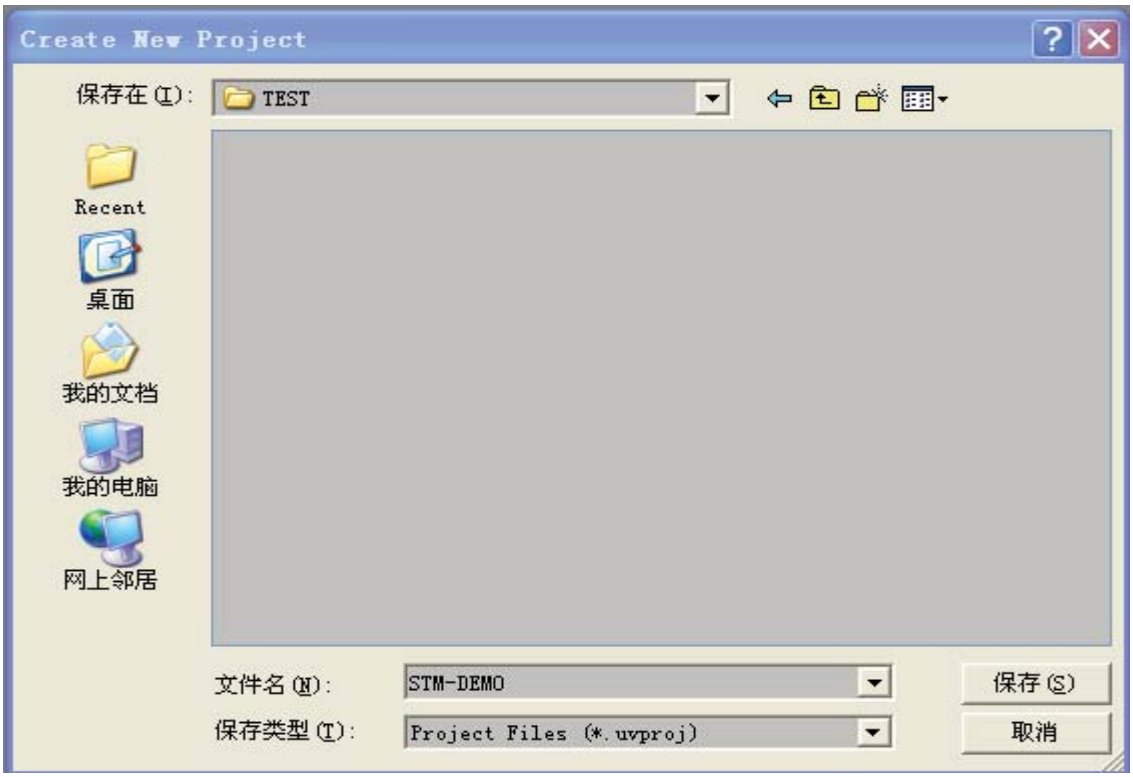
7、此时就可在桌面看到 µVision 的快捷图标，如下所示：

安装完成之后，先到 ST 的官方网站 http://www.st.com/mcu/familiesdocs-110.html（网址可能会随着时间有变动）下载 ST 的官方库，这里用的是 ST3.0.0 版本。

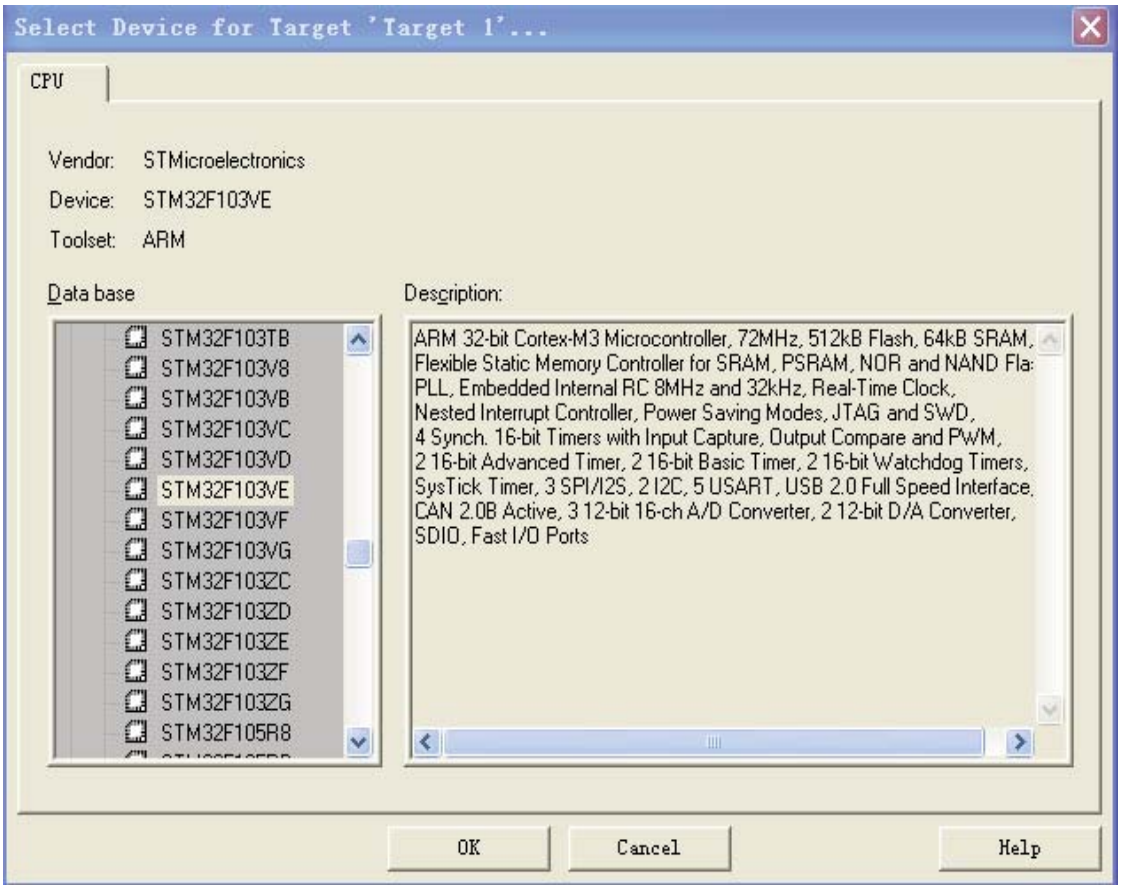
紧接着我们开始利用 STM32 的官方库来构建自己的工程模板。

1、点击桌面 µVision4 图标，启动软件。如果是第一次使用的话会打开一个自带的工程文件，我们可以通过工具栏 Project->Close Project 选项把它关掉。

2、在工具栏 Project->New µVision Project...新建我们的工程文件，我们将新建的工程文件保存在桌面的 TEST 文件夹下，文件名取为 STM-DEMO（英文 DEMO 的意思是例子），名字可以随便取，点击保存。



3、接下来的窗口是让我们选择公司跟芯片的型号，我们用的芯片是 ST 公司的 STM32F103VE，有 64K SRAM,512K Flash，属于高集成度的芯片。按如下选择即可。



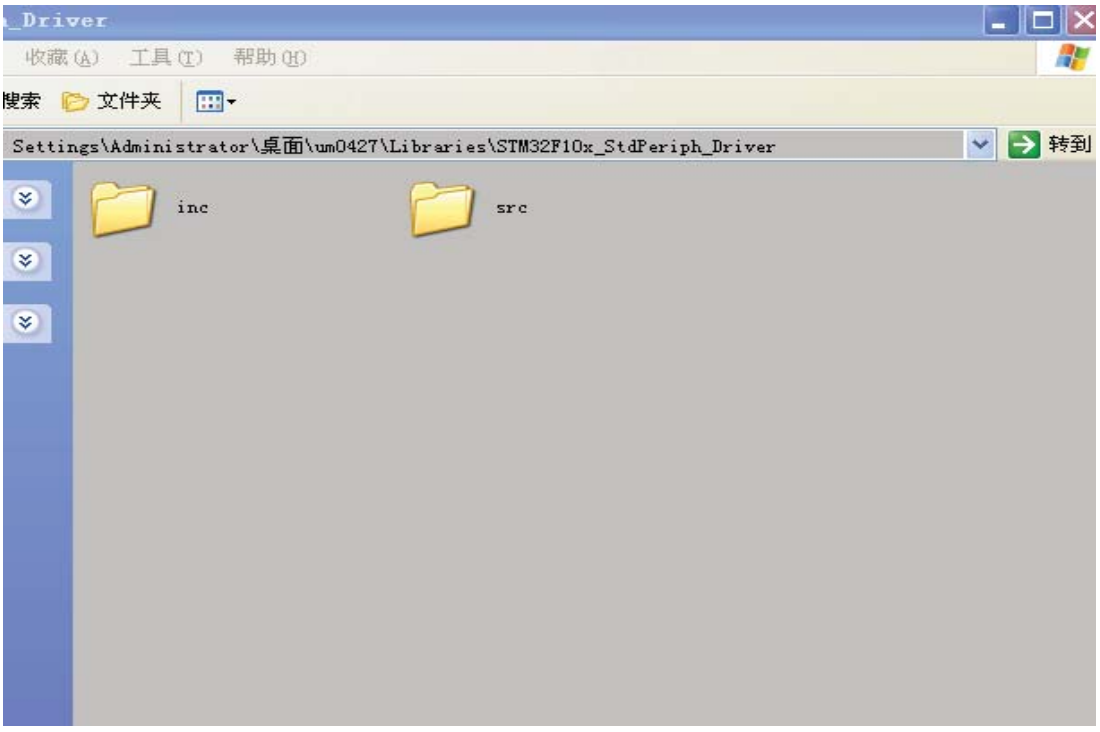
4、接下来的窗口问我们是否需要拷贝 STM32 的启动代码到工程文件中，这份启动代码在 M3 系列中都是适用的，一般情况下我们都点击是，但我们这里用的是 ST 的库，库文件里面也自带了这一份启动代码，所以为了保持库的完整性，我们就不需要开发环境为我们自带的启动代码了，稍后我们自己手动添加，这里我们点击否。

5、此时我们的工程新建成功，如下图所示。但我们的工程中还没有任何文件，接下来我们需要在我们的工程中添加所需文件。

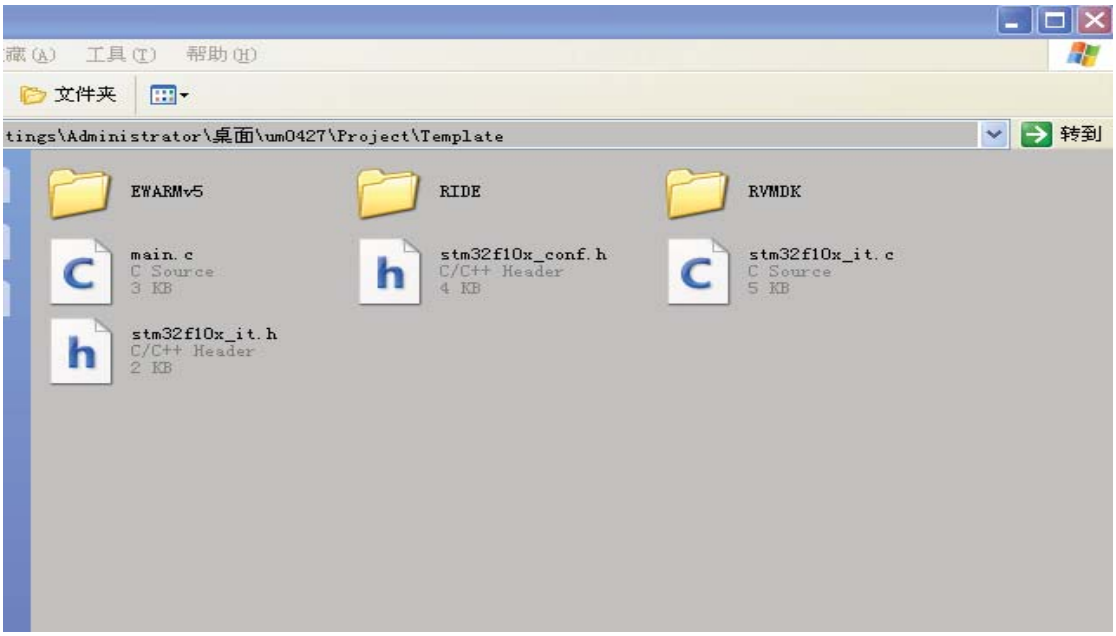
6、在 TEST 文件夹中，新建三个文件夹，分别为 USER、FWlib、CMSIS。USR 用来存放工程文件和用户代码，包括主函数 main.c，FWlib 用来存放 STM32 库里面的 inc 和 src 这两个文件，这两个文件包含了芯片上的所有驱动。CMSIS 用来存放库为我们自带的启动文件和一些 M3 系列通用的文件。CMSIS 里面存放的文件适合任何 M3 内核的单片机。CMSIS 的缩写为: Cortex Microcontroller Software Interface Standard，

是 ARM Cortex 微控制器软件接口标准，是 ARM 公司为芯片厂商提供的一套通用的且独立于芯片厂商的处理器软件接口。

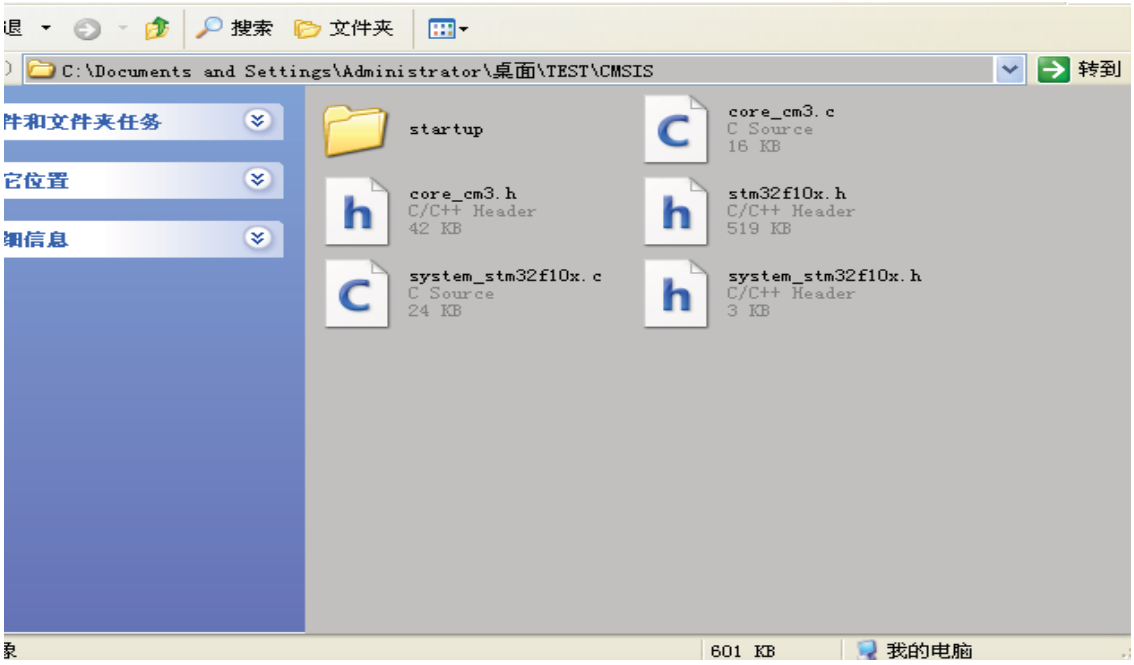
7、把库 um0427\Libraries\STM32F10x_StdPeriph_Driver 文件夹下的 inc 跟 src 这两个文件夹拷贝的 FWlib 文件夹中。



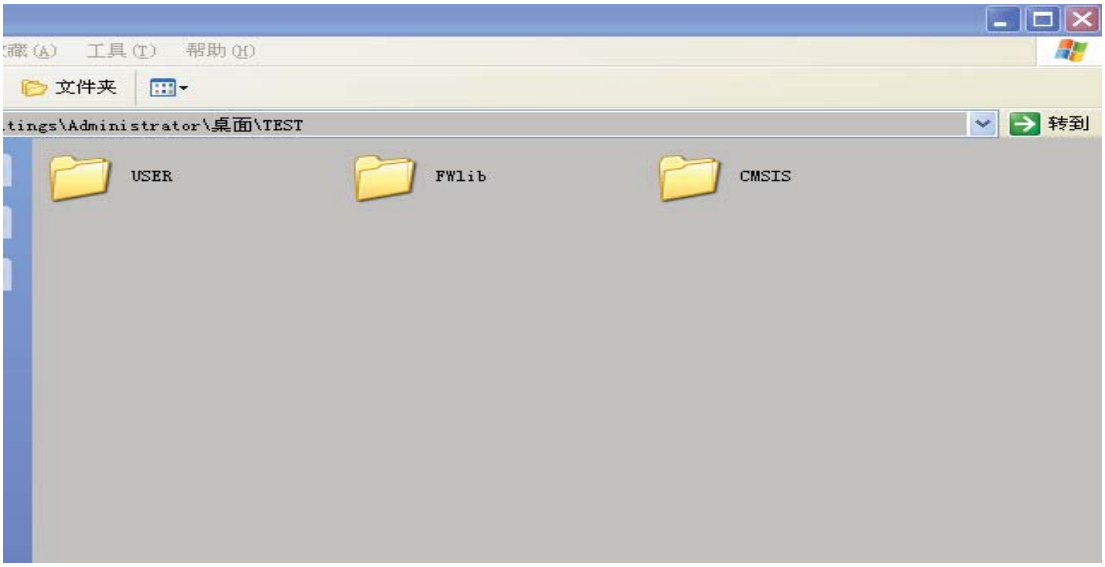
8、把库文件夹 um0427\Project\Template 下的 main.c、stm32f10x_conf.h、stm32f10x_it.h、 stm32f10x_it.c 拷贝到 USER 目录下。stm32f10x_it.h、和 stm32f10x_it.c 这两个文件里面是中断函数，里面为空，并没有写任何的中断服务程序。stm32f10x_conf.h 是用户需要配置的头文件，当我们需要用到芯片中的某部分外设的驱动时，我们只需要在该文件下将该驱动的头文件包含进来即可，片上外设的驱动在 src 文件夹中，inc 文件夹里面是它们的头文件。这三个文件是用户在编程时需要修改的文件，其他库文件一般不需要修改。



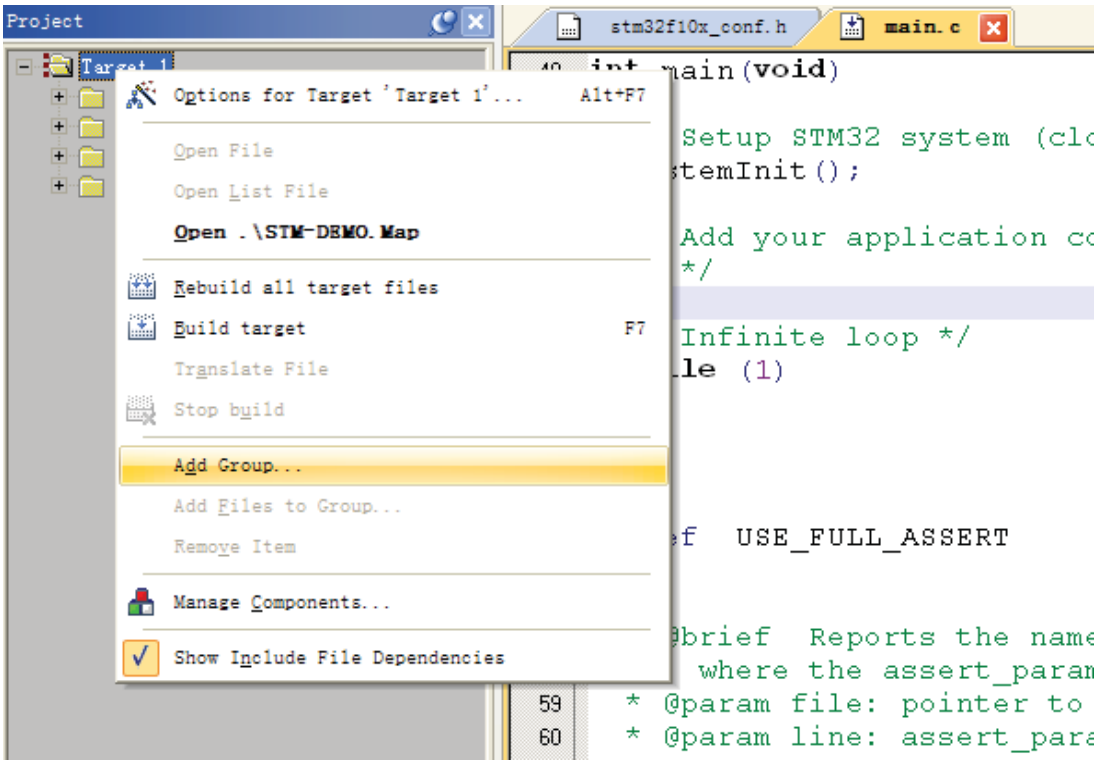
9、将库文件 um0427\Libraries\CMSIS\Core\CM3 文件夹下的全部文件拷贝到 CMSIS 文件夹中。Startup/arm 里面有三个启动文件，分别为 startup_stm32f10x_ld.s、startup_stm32f10x_md.s 、startup_stm32f10x_hd.s，按顺序是小容量、中容量、大容量 Flash 单片机的启动文件。我们这里用的是 STM32F103VE 有 512K Flash，属于大容量的，所以等下我们把 startup_stm32f10x_hd.s 添加到我们的工程中。具 ST 的官方资料：Flash 在 16 ~32 Kbytes 为小容量， 64 ~128 Kbytes 为中容量，256 ~512 Kbytes 为大容量，不同大小的 Flash 对应的启动文件不一样，这点要注意。其他几个文件也是 M3 内核单片机通用的，是独立于芯片厂商的，其功能由 ARM 公司决定，具体作用这里先不详述。



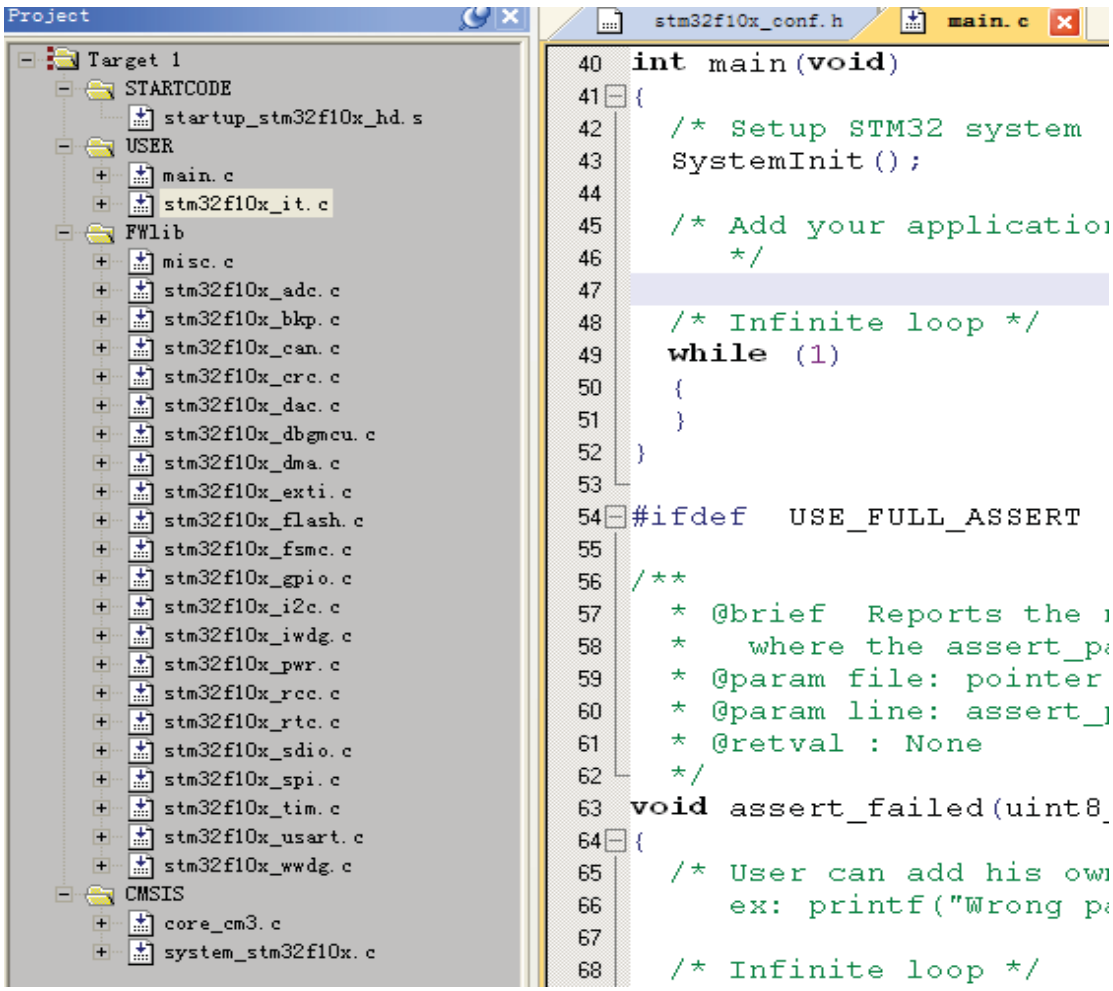
10、最后我们将我们的工程跟其他的一些编译出来的文件也放在 USER 目录下，这样看起来显得没那么乱。



11、回到我们的工程中，选中 Target 右键选中 Add Group...选项新建四个组，分别命名为 STARTCODE、USER、FWlib、CMSIS。STARTCODE 从名字就可以看得出我们是用它来放我们的启动代码的，USER 用来存放用户自定义的应用程序，FWlib 用来存放库文件，CMSIS 用来存放 M3 系列单片机通用的文件。



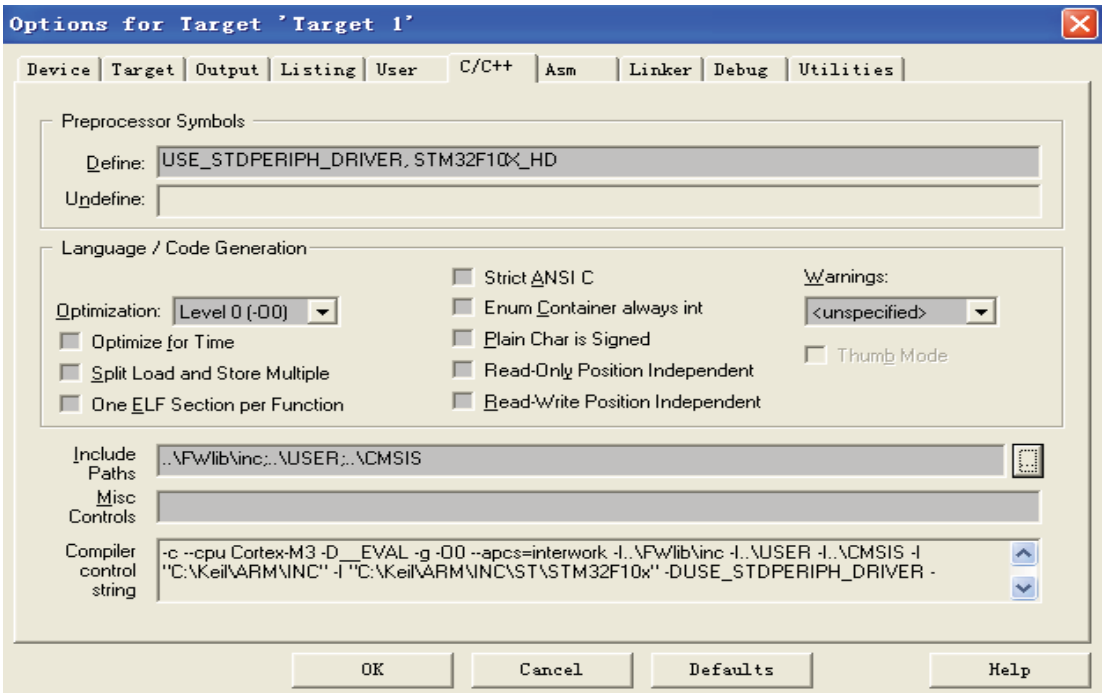
12、接下来我们往我们这些新建的组中添加文件，双击哪个组就可以往哪个组里面添加文件。我们在 STARTCOKE 里面添加 startup_stm32f10x_hd.s，在 USER 组里面添加 main.c 和 stm32f10x_it.c 这两个文件，在 FWlib 组里面添加 src 里面的全部驱动文件，当然，src 里面的驱动文件也可以需要哪个就添加哪个。这里将它们全部添加进去是为了后续开发的方便，况且我们可以通过配置 stm32f10x_conf.h 这个头文件来选择性添加，只有在 stm32f10x_conf.h 文件中配置的文件才会被编译。注意，这些组里面添加的都是汇编文件跟 C 文件，头文件是不需要添加的。最终效果如下图：



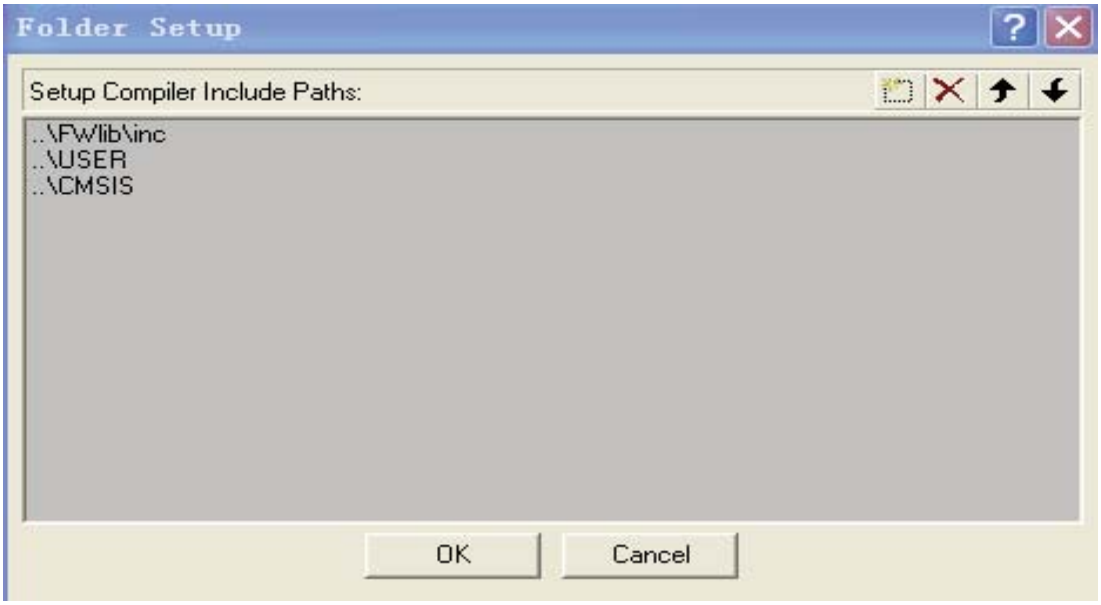
13、至此，我们的工程一基本建好，现在点击工具栏图标 来编译下，结果发现了 N 多的错误，如下图所示。究其原因是编译器在编译时搜索的库路径是：C:\Keil\ARM\INC\ST\STM32F10x ，这里面也有 ST 官方的驱动库的头文件，里面的文件跟刚刚我们的 inc 文件夹下的内容差不多，只是版本旧了点，在编译我们新版本的库时存在不兼容性。为了解决这个问题，我们需要屏蔽掉编译器默认库的搜索路径。

```
Build Output
C:\Keil\ARM\INC\ST\STM32F10x\stm32f10x_type.h:                                  ^
C:\Keil\ARM\INC\ST\STM32F10x\stm32f10x_type.h(62): error:  #101: "ERROR" has already been declarec
C:\Keil\ARM\INC\ST\STM32F10x\stm32f10x_type.h:   typedef enum {ERROR = 0, SUCCESS = !ERROR} ErrorS
C:\Keil\ARM\INC\ST\STM32F10x\stm32f10x_type.h:                ^
C:\Keil\ARM\INC\ST\STM32F10x\stm32f10x_type.h(62): error:  #101: "SUCCESS" has already been declar
C:\Keil\ARM\INC\ST\STM32F10x\stm32f10x_type.h:   typedef enum {ERROR = 0, SUCCESS = !ERROR} ErrorS
C:\Keil\ARM\INC\ST\STM32F10x\stm32f10x_type.h:                           ^
C:\Keil\ARM\INC\ST\STM32F10x\stm32f10x_type.h(62): error:  #256: invalid redeclaration of type nar
C:\Keil\ARM\INC\ST\STM32F10x\stm32f10x_type.h:   typedef enum {ERROR = 0, SUCCESS = !ERROR} ErrorS
C:\Keil\ARM\INC\ST\STM32F10x\stm32f10x_type.h:                                  ^
C:\Keil\ARM\INC\ST\STM32F10x\stm32f10x_conf.h(147): warning:  #47-D: incompatible redefinition of
C:\Keil\ARM\INC\ST\STM32F10x\stm32f10x_conf.h:   #define HSE_Value    ((u32)8000000) /* Value of t
C:\Keil\ARM\INC\ST\STM32F10x\stm32f10x_conf.h:          ^
C:\Keil\ARM\INC\ST\STM32F10x\stm32f10x_conf.h: ..\CMSIS\system_stm32f10x.c: 1 warning, 21 errors
Target not created
```

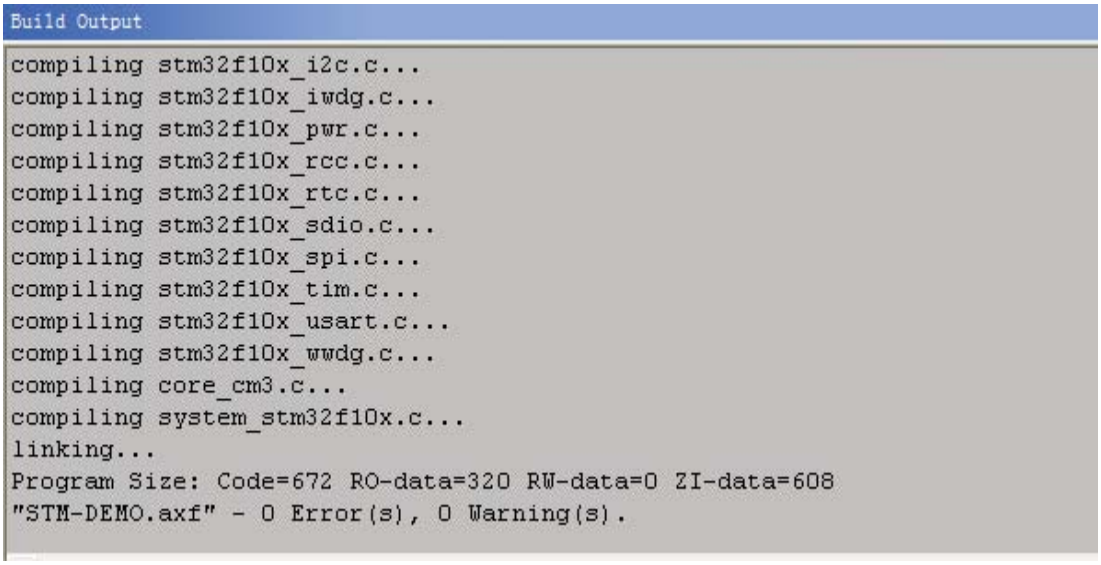
14、点击工具栏中的魔术棒按钮  ，在弹出来的窗口中选中 C/C++ 选项卡，在这里添加库文件的搜索路径，这样就可以屏蔽掉默认的搜索路径。但当编译器在我们指定的路径下 搜索不到的话还是会回到标准目录去搜索，就像有些 ANSIC C 的库文件，如 stdin.h 、stdio.h。



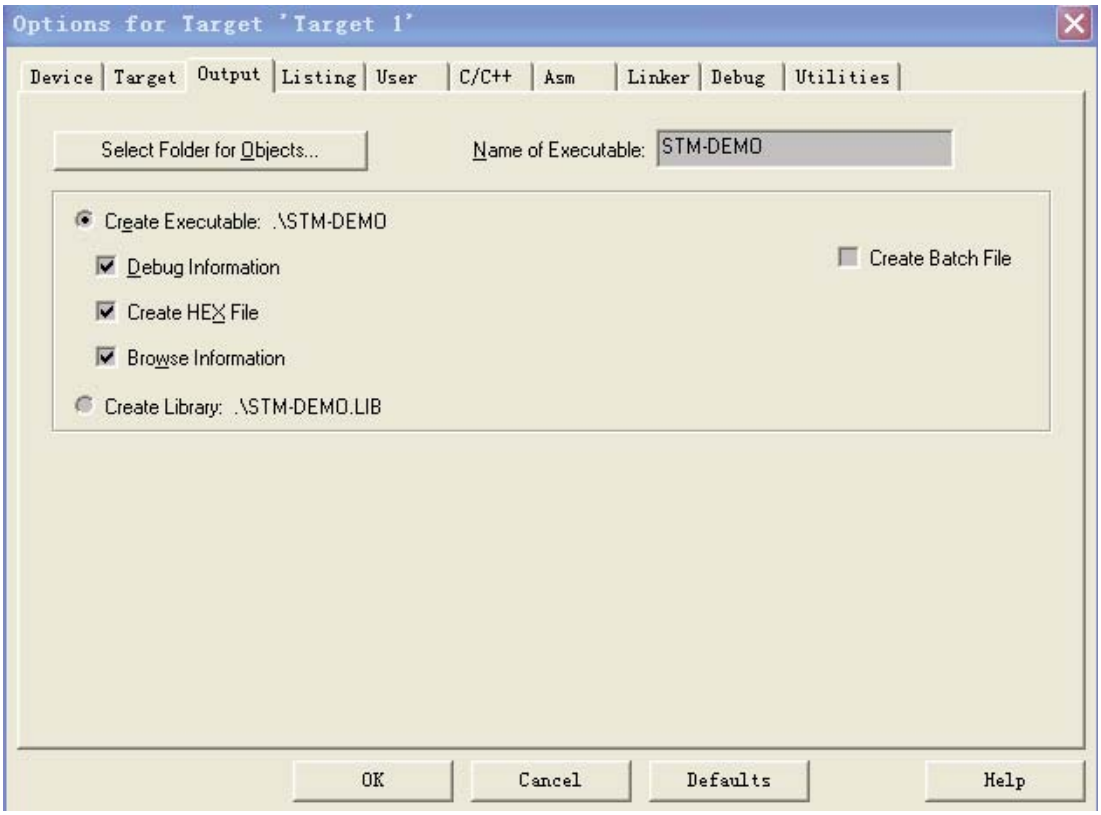
15、库文件路径修改成功之后如下所示：



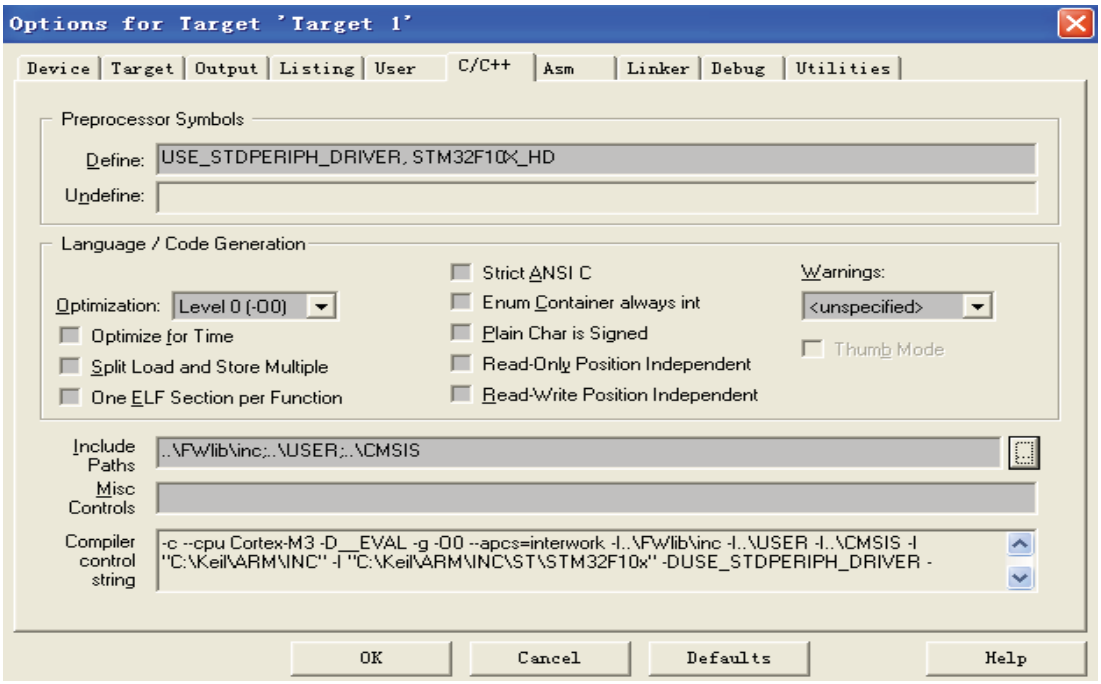
16、现在我们再编译下，就会发现刚刚出现的那些错误都没了，如下图所示：

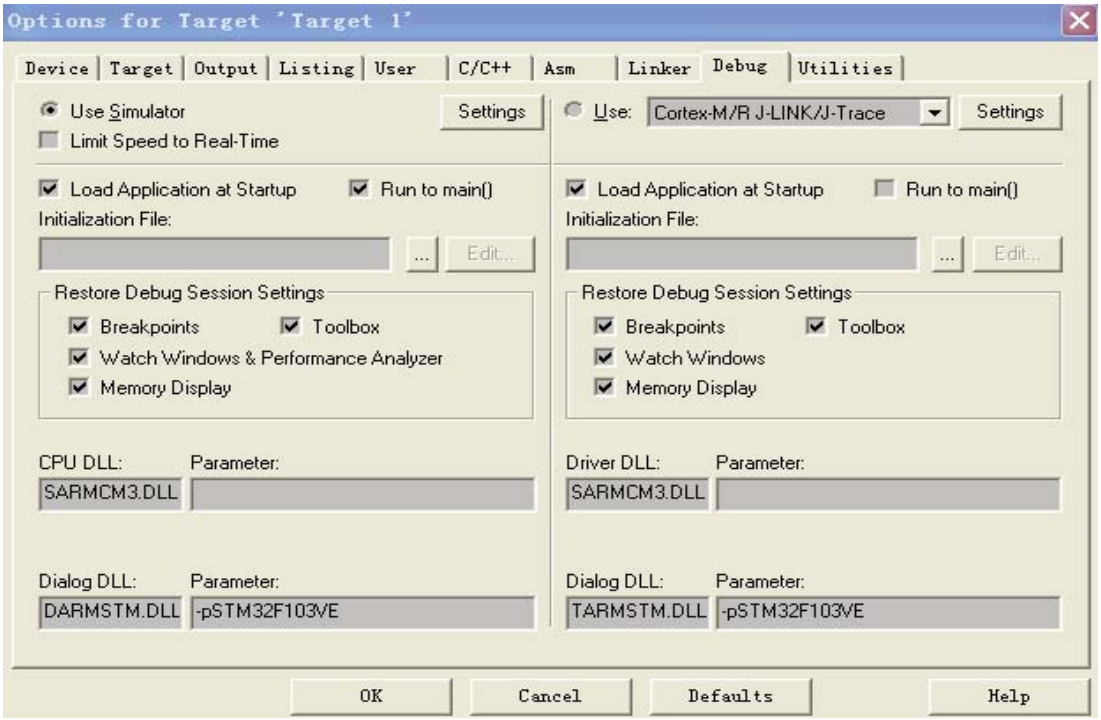


17、现在我们还需要再做一些工作我们的工程才算完成，主要修改如下几个选项卡，其他保持默认即可，如下图所示：



在 Define 里面添加 USE_STDPERIPH_DRIVER, STM32F10X_HD。添加 USE_STDPERIPH_DRIVER 是为了使用 ST 的库，添加 STM32F10X_HD 是因为我们用的芯片是大容量的，添加了 STM32F10X_HD 这个宏之后，库文件里面为大容量定义的寄存器我们就可以用了。芯片是小或中容量的时候宏要换成 STM32F10X_LD 或者 STM32F10X_MD。其实不管是什么容量的，我们只要添加上 STM32F10X_HD 这个宏即可，当你用小或者中容量的芯片时，那些为大容量定义的寄存器我不去访问就是了，反正也访问不了。


Options for Target 'Target 1'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Preprocessor Symbols
Define:
USE_STDPERIPH_DRIVER, STM32F10X_HD
Undefine:
Language / Code Generation
Optimization:
Level 0 (-O0)
Optimize for Time
Split Load and Store Multiple
One ELF Section per Function
Strict ANSI C
Enum Container always int
Plain Char is Signed
Read-Only Position Independent
Read-Write Position Independent
Warnings:
<unspecified>
Thumb Mode
Include Paths
..\FWlib\inc;..\USER;..\CMSIS
Misc Controls
Compiler control string
-c --cpu Cortex-M3 -D__EVAL -g -O0 --apcs=interwork -I..\FWlib\inc -I..\USER -I..\CMSIS -I "C:\Keil\ARM\INC" -I "C:\Keil\ARM\INC\ST\STM32F10x" -DUSE_STDPERIPH_DRIVER -
OK
Cancel
Defaults
Help


Options for Target 'Target 1'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Use Simulator
Limit Speed to Real-Time
Settings
Use:
Cortex-M/R J-LINK/J-Trace
Settings
Load Application at Startup
Run to main()
Initialization File:
Edit...
Restore Debug Session Settings
Breakpoints
Toolbox
Watch Windows & Performance Analyzer
Watch Windows
Memory Display
CPU DLL:
Parameter:
SARMCM3.DLL
Driver DLL:
Dialog DLL:
DARMSTM.DLL
-pSTM32F103VE
TARMSTM.DLL
OK
Cancel
Defaults
Help

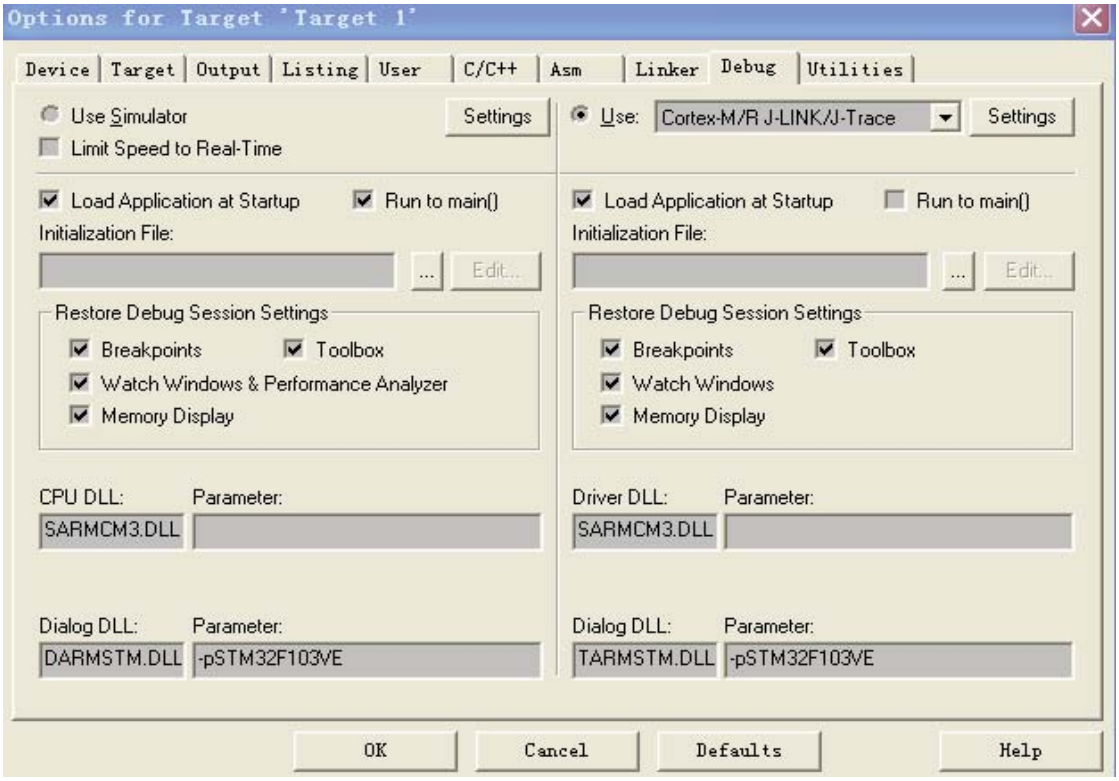
17、至此，大功告成，我们就可以在 main.c 函数中写自己的程序了。

stm32f10x_conf.h | main.c*

```
#include "stm32f10x.h"

/** @addtogroup Template_Project
  * @{
  */

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
/* Private macro -------------------------------------------------------------*/
/* Private variables ---------------------------------------------------------*/
/* Private function prototypes -----------------------------------------------*/
/* Private functions ---------------------------------------------------------*/
/**
  * @brief  Main program.
  * @param  None
  * @retval : None
  */
int main(void)
{
  /* Setup STM32 system (clock, PLL and Flash configuration) */
  SystemInit();

  /* Add your application code here
     */

  /* Infinite loop */
  while (1)
  {
  }
}
```

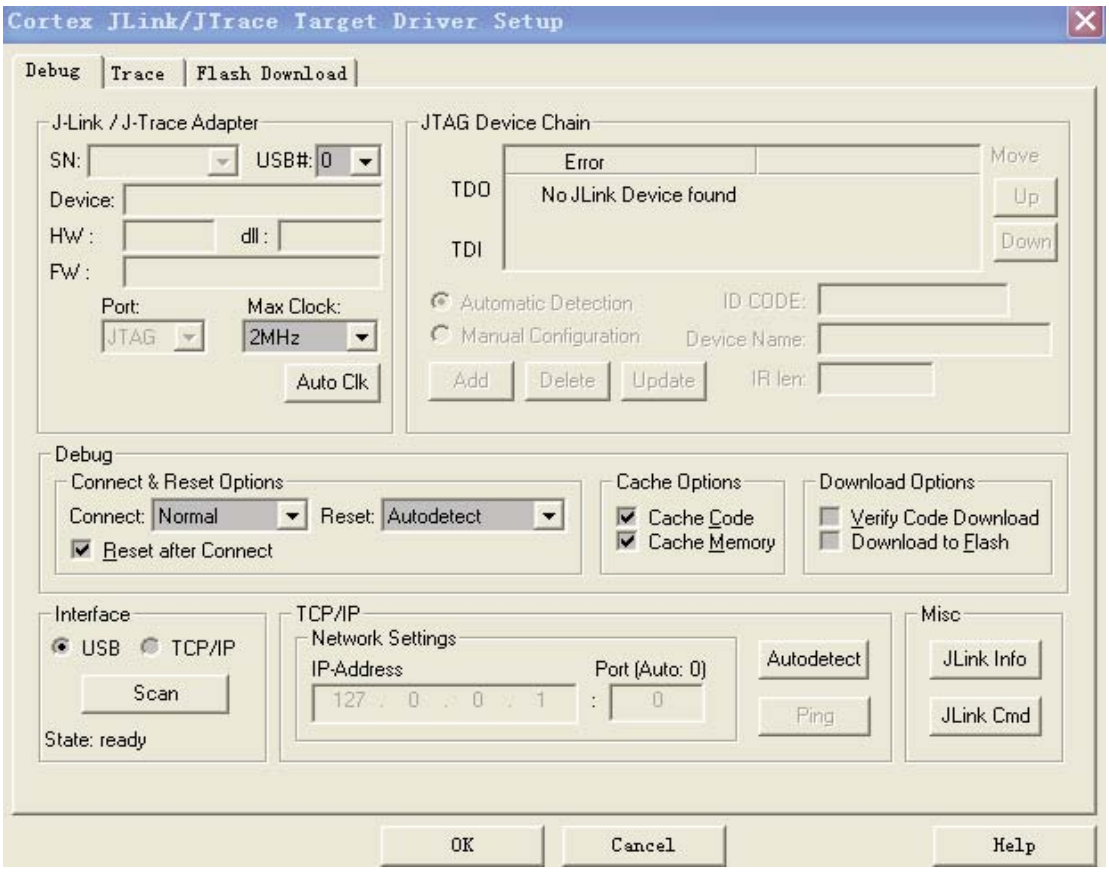
小结：学会新建工程是后续程序开发的一个非常重要的工作，如果工程没法建成功，那何来的开发呢？在建立工程时需要注意的是：1、因为我们用的是 ST 官方的新版本的库，跟编译器自带的库会存在不兼容性，所以我们需要修改库的搜索路径。2、这个工程我们是设置成软件仿真，如果是用开发板加 J-LINK 调试的话，还需要在开发环境中做如下修改。实际上，我们开发程序的时候 80%都是在硬件上调试的。

具体配置如下图所示：

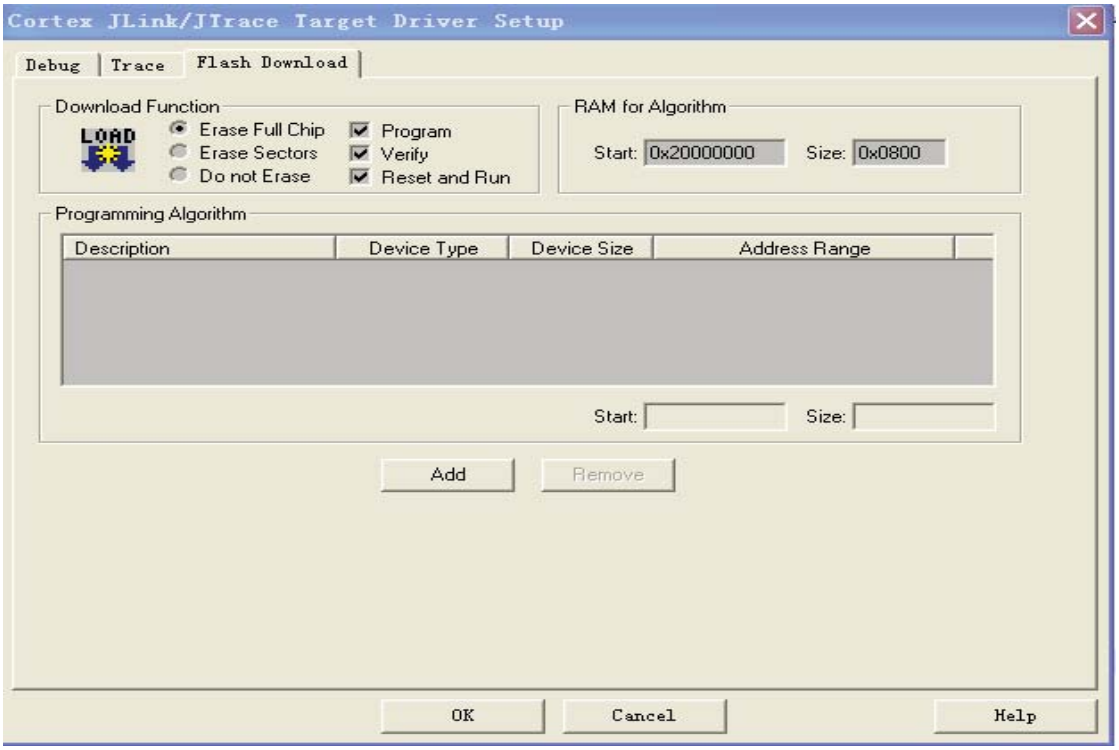


注意：以下三个图是 J-LINK 没接到开发板上且没上电时的截图

没检测到 CPU 的 ID。

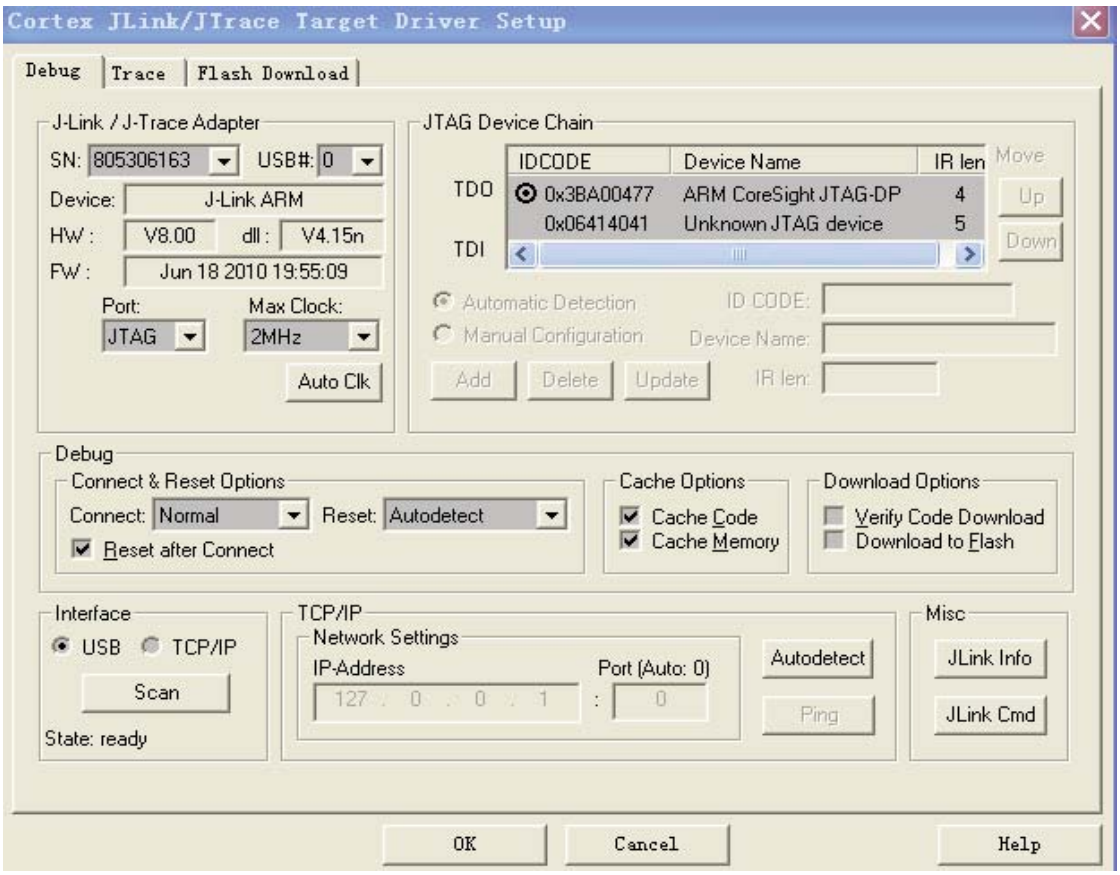


没有添加 CPU 支持的 flash。

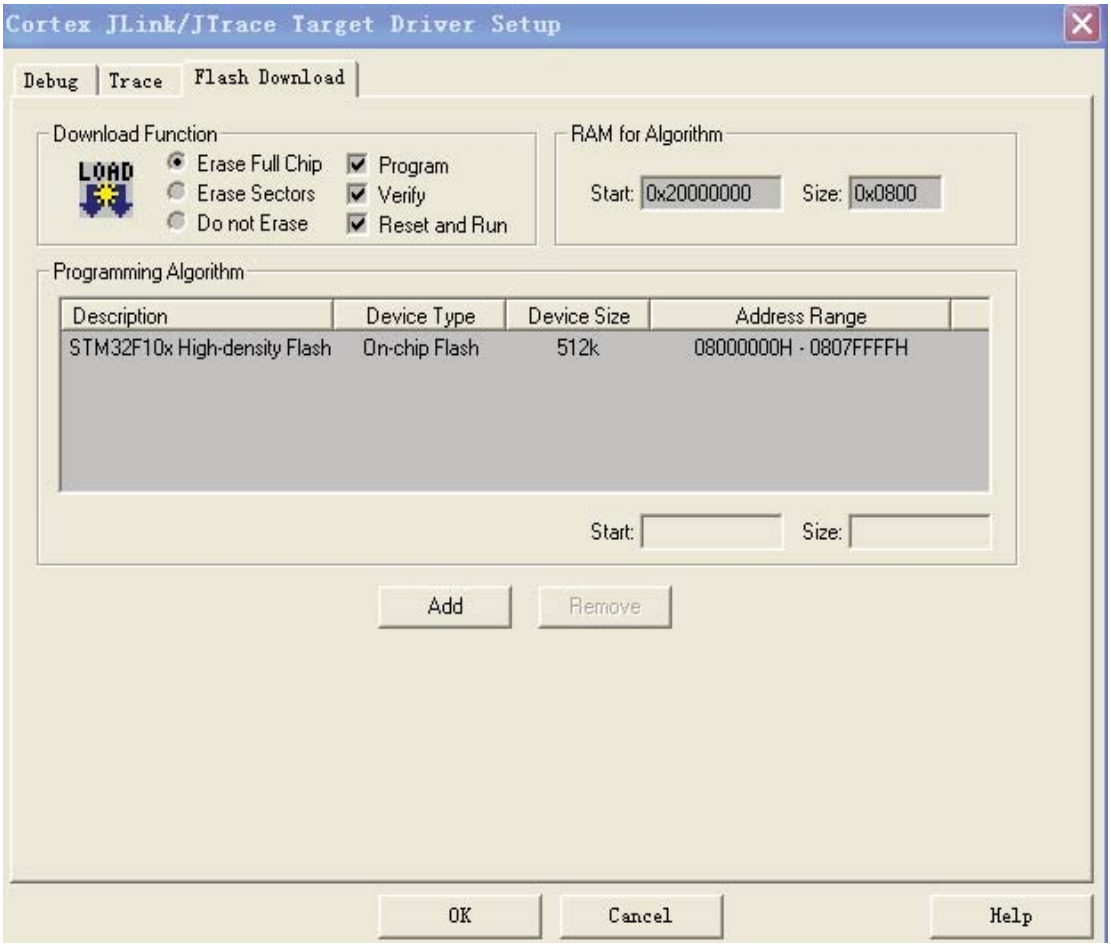


以下两幅是 J-LINK 接到开发板且上电时的截图：

检测到 CPU 的 ID



添加 CPU 支持的 flash，这一步很重要，否则程序将无法下载。



至此，一个真正完整的工程就算建立了。

实验讲解完毕，野火祝大家学习愉快^_^。