STM32开发基础

2019年3月

# 嵌入式软件开发介绍

STM32微处理器是ST（意法半导体）公司使用ARM公司的[cortex-M](https://www.baidu.com/s?wd=cortex-M&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "_blank)3为核心生产的32bit系列的单片机。STM32单片机基于ARM核，所以很多基于ARM的嵌入式开发环境都可用于STM32开发平台。选择合适的开发环境可以加快开发进度，节省开发成本。STM32常用的开发工具有Keil MDK和IAR EWARM，而基于Keil MDK的开发人员更多，学习参考资料更容易获得，我们将以Keil当前最新的版本 MDK-ARM V5.21a来进行STM32嵌入式开发。

# Keil开发环境准备

## 准备安装文件

准备好安装需要使用的文件，如图1所示。请先安装MDK，然后安装3个库文件，安装顺序：

1)安装MDK521A.exe；

2)安装Keil.STM32F1xx\_DFP.2.0.0.pack；

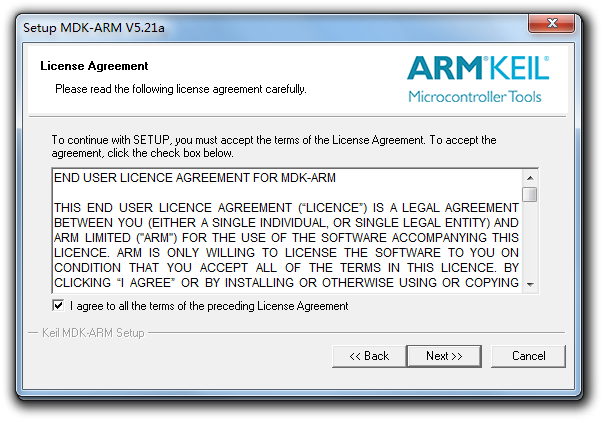
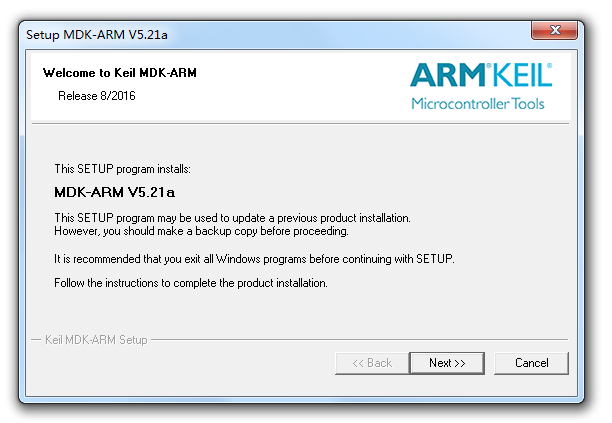
3)安装Keil.STM32F4xx\_DFP.2.9.0.pack ；

4)安装NordicSemiconductor.nRF\_DeviceFamilyPack.8.3.2.pack。

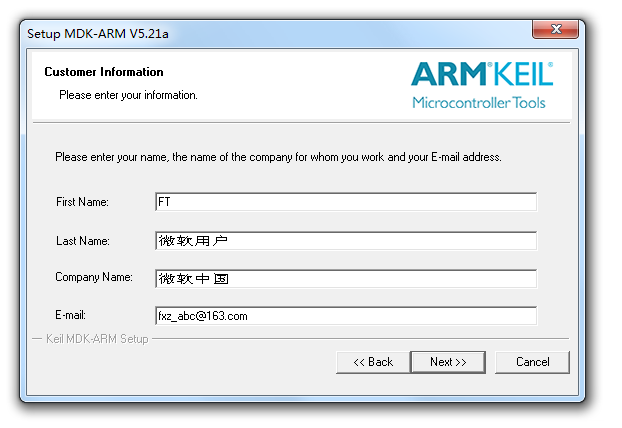
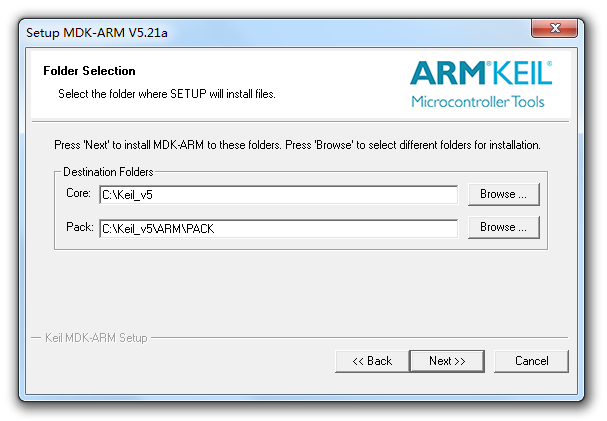


图1 MDK-ARM V5.21a的安装文件

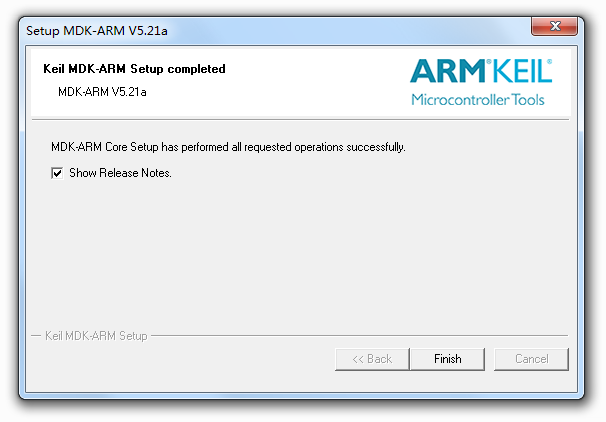
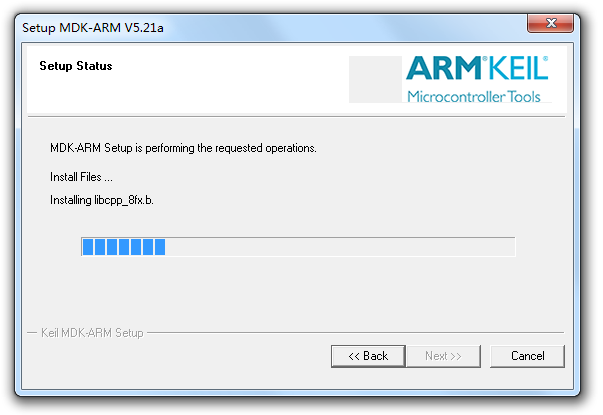
## 安装MDK521A.exe

运行Keil MDK的安装程序MDK521A.exe，具体安装过程如下：

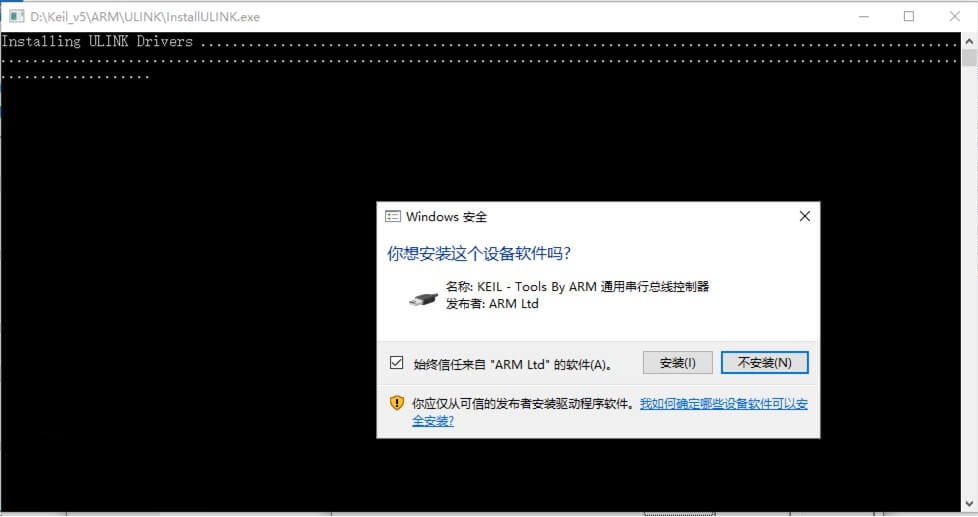
1）点击next 2）勾选复选框，点击next



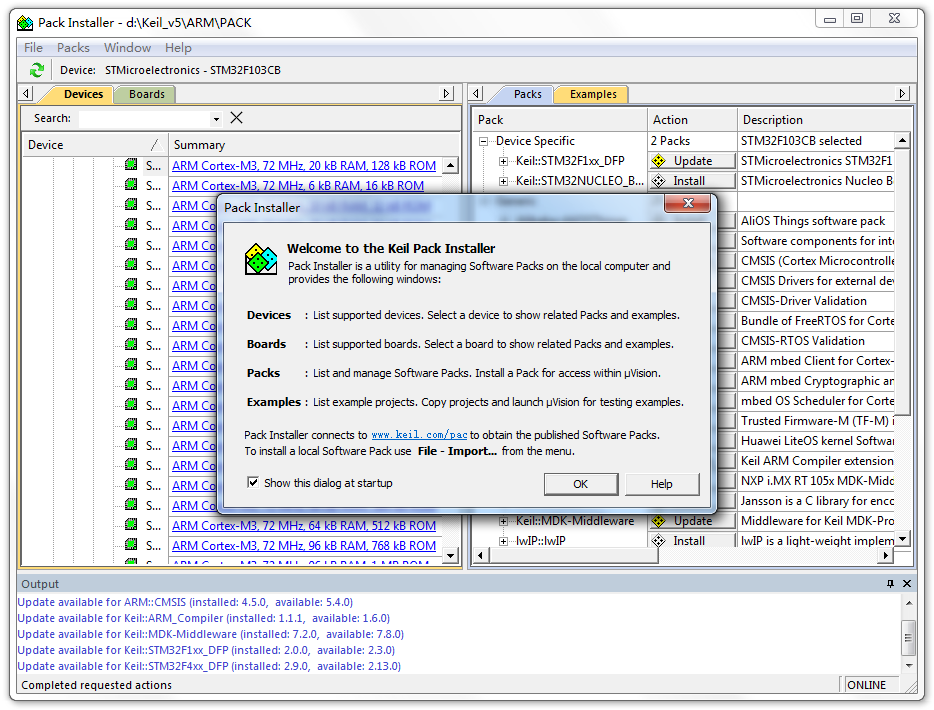
3）选择安装位置，点击next 4）填写first name和Email，点击next



5）等待安装完成 6）完成，点击finish



7）安装ULINK驱动，点击安装

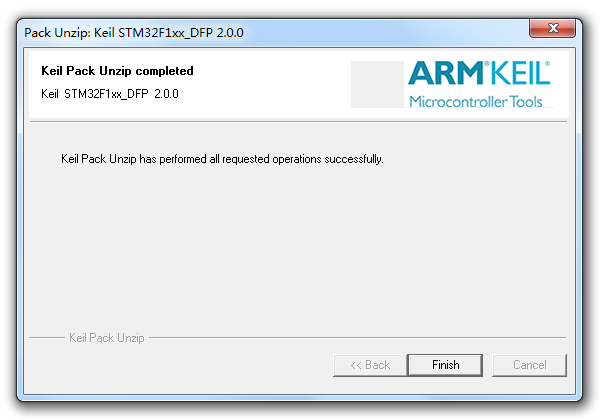
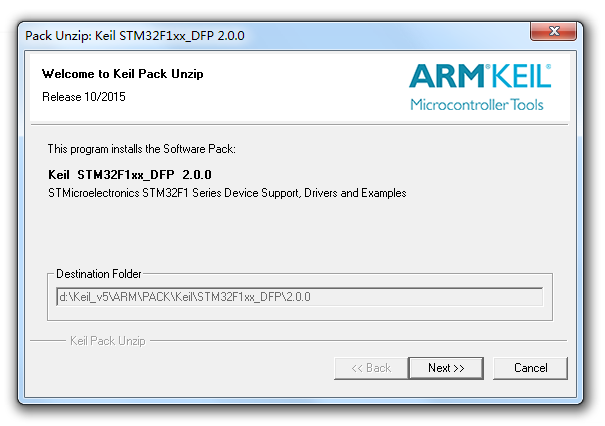


8）安装完成后会提示安装库文件包，库文件包可以根据需要安装，这里直接关闭退出

图2 安装 MDK521A.exe

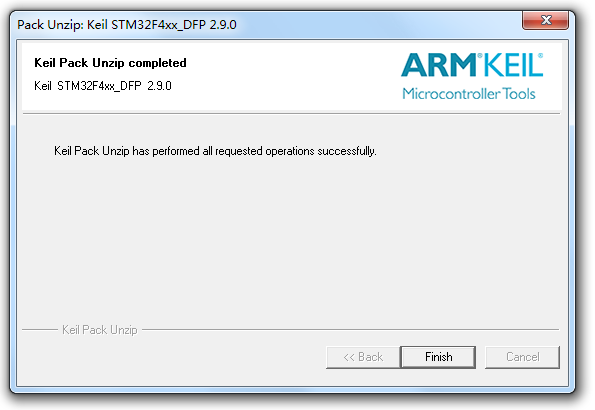
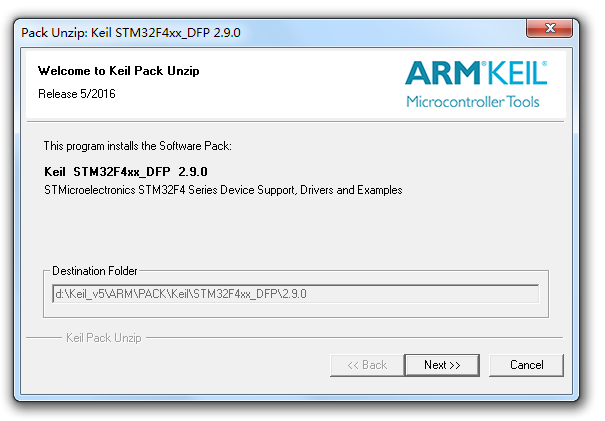
## 安装3个库文件包

库文件的安装过程如图3，图4，图5所示。



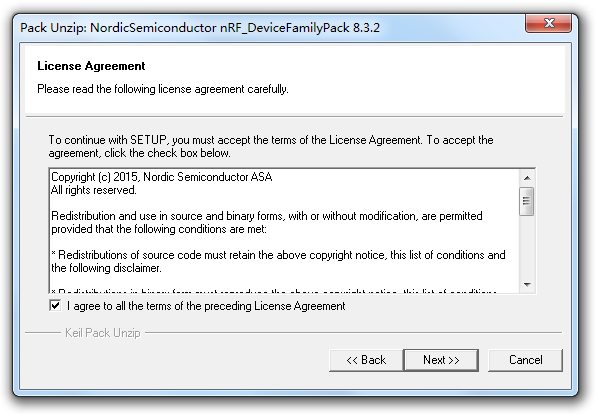
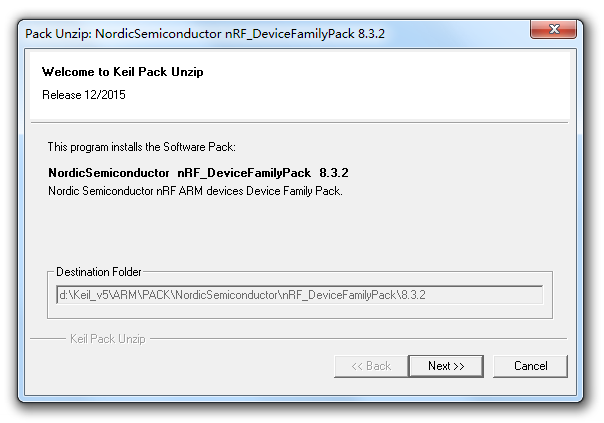
1）点击next 2）完成，点击finish

图3 安装Keil.STM32F4xx\_DFP.2.9.0.pack

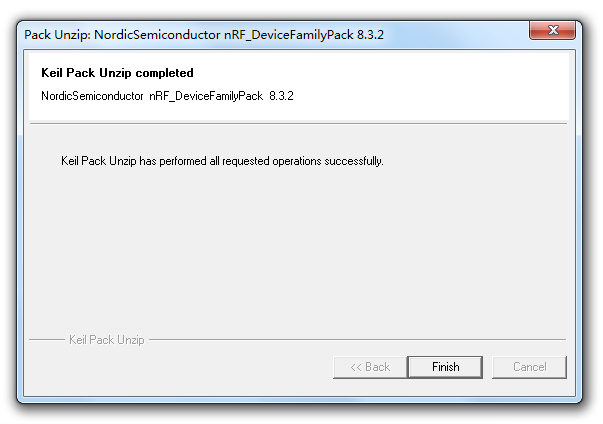


1）点击next 2）完成，点击finish

图4 安装Keil.STM32F1xx\_DFP.2.0.0.pack



1）点击next 2）点击next



3）完成，点击finish

图5 安装NordicSemiconductor.nRF\_DeviceFamilyPack.8.3.2.pack

## 破解MDK

破解方法如下：

1）打开MDK（注意：WIN7以及以上系统，请点击MDK图标，右键->以管理员权限打开）;

2）点击MDK中菜单：File->License Management，调出注册管理界面;

3）复制图 6界面的CID;

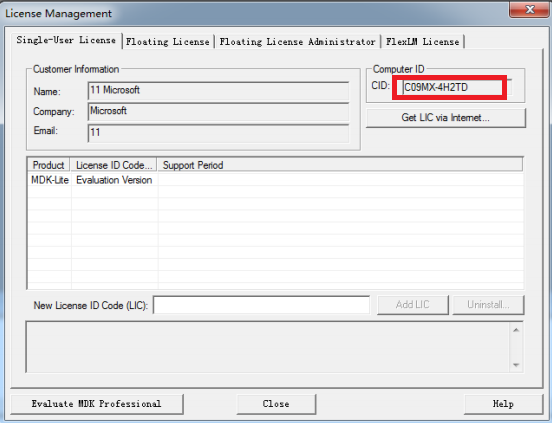


图6 MDK注册管理界面

4)运行Keygen.exe，CID选项里面填入上一步复制的CID，其他选项按图7填写。点击Generate，生成License ID并复制。



图7 Keygen.exe界面

5）把生成的Licese ID添加到MDK中，破解成功如图8所示。

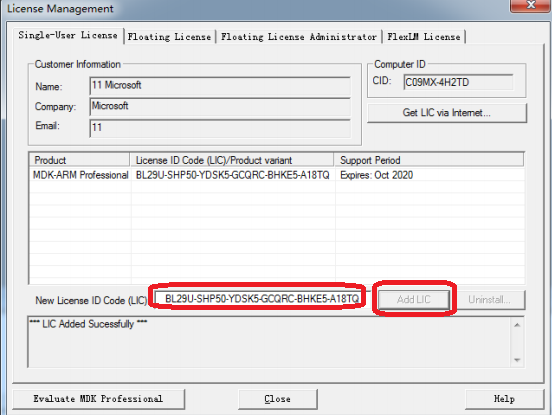


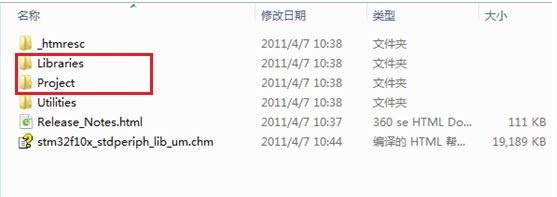
图8 MDK注册管理界面破解成功

# 嵌入式软件开发基础

下面我们以操作手柄的使用的STM32F103C8单片机为例，详细描述一个新工程的建立过程。

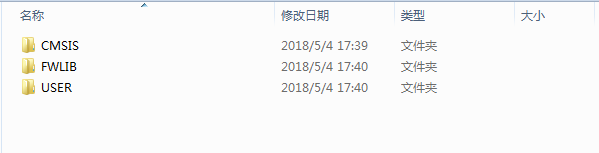
## 下载STM32F10x\_StdPeriph\_Lib\_V3.5.0.rar

下载ST官方库文件STM32F10x\_StdPeriph\_Lib\_V3.5.0.rar,解压后文件夹包含文件如下，真正用到的文件均在红色框标出的两个文件夹里。



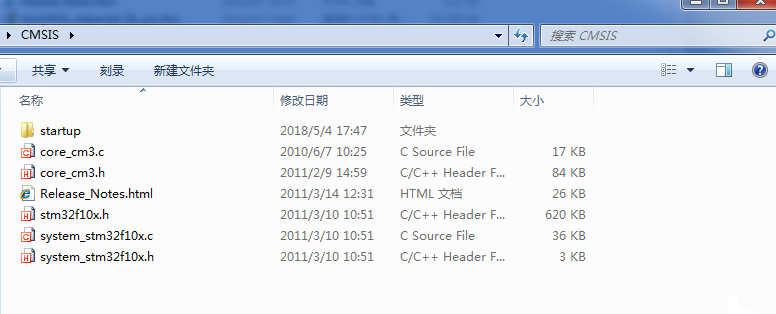
## 准备工程文件夹

在不包含中文目录的文件夹下新建一个项目文件夹，例如d:\study\test1，在test1文件夹下新建三个文件夹CMSIS（存放内核函数及启动[引导文件](https://www.baidu.com/s?wd=%E5%BC%95%E5%AF%BC%E6%96%87%E4%BB%B6&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "_blank)），FWLIB（存放库函数），USER（存放用户自己的函数），如下图所示。



* 准备CMSIS文件夹

将STM32F10x\_StdPeriph\_Lib\_V3.5.0\Libraries\CMSIS\CM3\CoreSupport中的文件复制到CMSIS中，再将STM32F10x\_StdPeriph\_Lib\_V3.5.0\Libraries\CMSIS\CM3\DeviceSupport\ST\STM32F10x中的文件全部复制到CMSIS中。最终CMSIS中的文件如下图所示。



* 准备FWLIB文件夹

将STM32F10x\_StdPeriph\_Lib\_V3.5.0\Libraries\STM32F10x\_StdPeriph

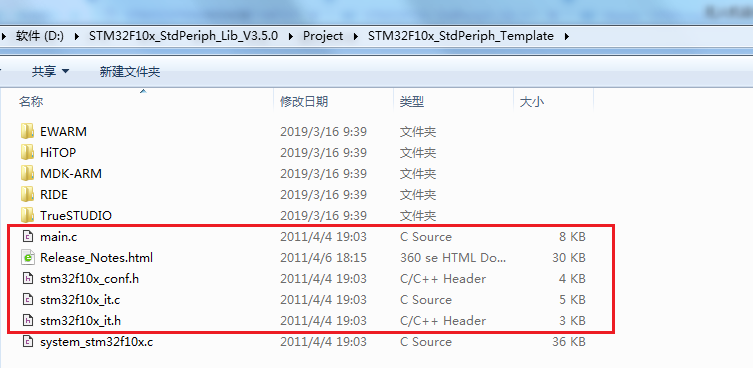
\_Driver中的 inc和src文件夹复制进FWLIB文件夹，如下图所示。



* 准备USER文件夹

 将STM32F10x\_StdPeriph\_Lib\_V3.5.0\Project\STM32F10x\_StdPeriph

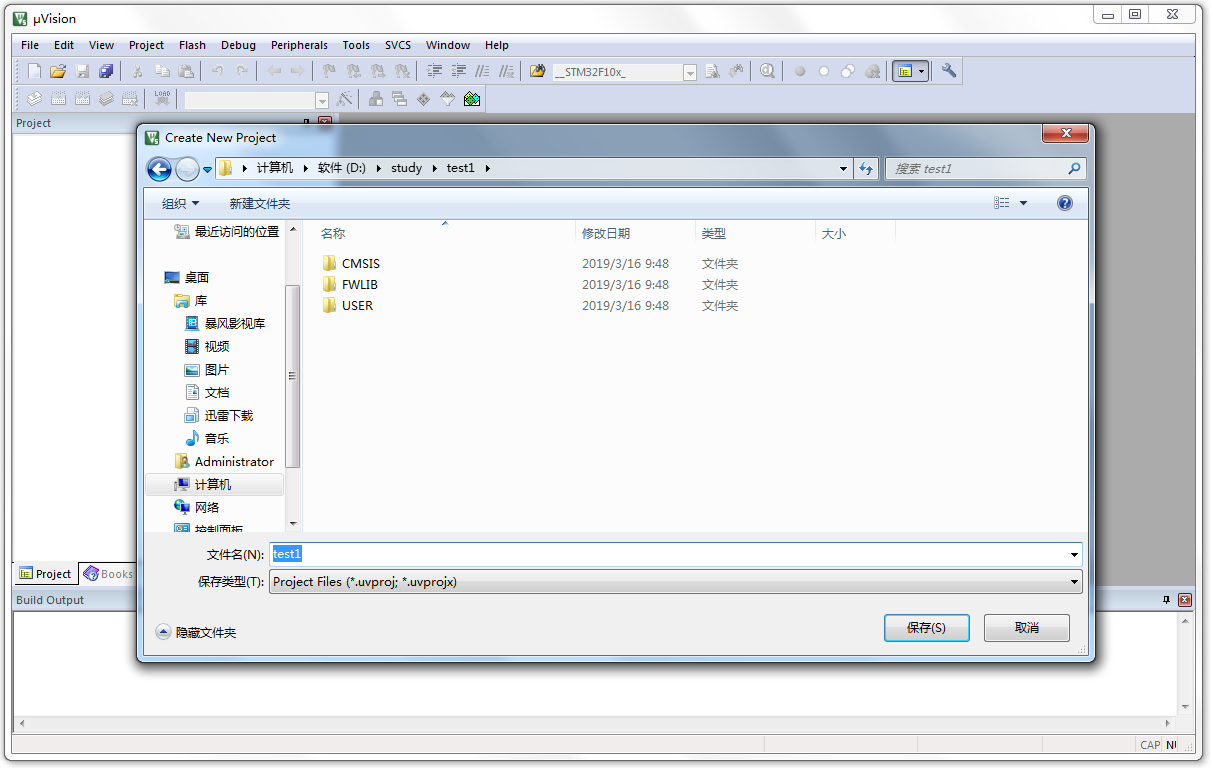
\_Template文件夹下图中红色框标出几个文件复制进USER。



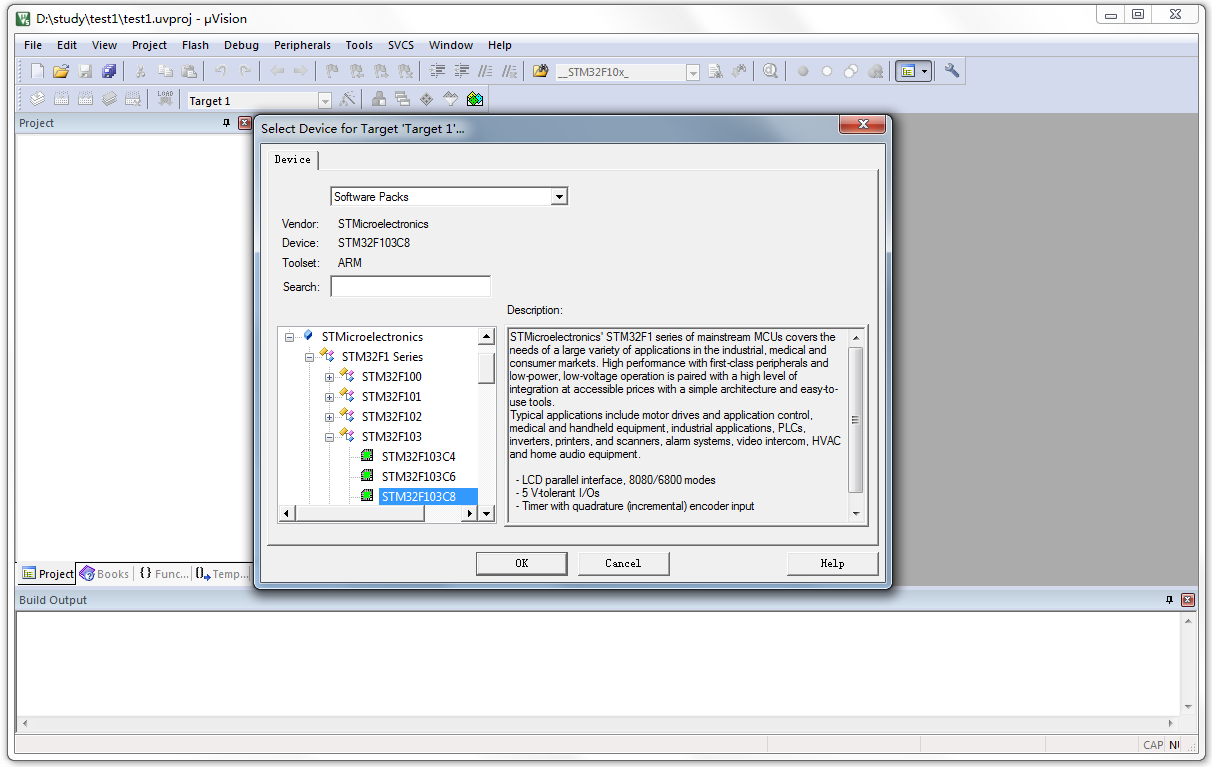
至此，文件夹准备工作完成。

## 新建工程，添加工程文件

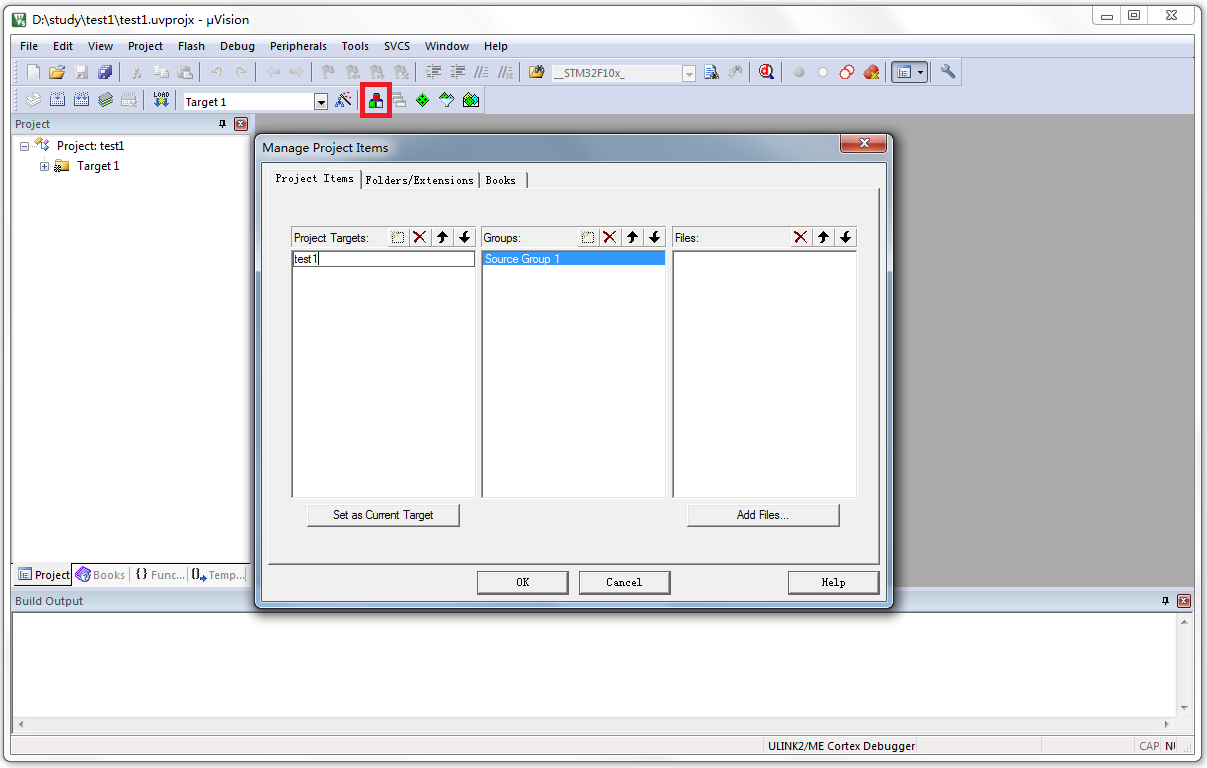
打开keil，点击菜单Project -> New uVision Project...,工程文件名填入test1，并保存在刚刚新建的d:\study\test1文件夹里。



点击保存后进行芯片选择，我们选择STM32F103C8。



点击OK之后跳出一个Manage Runtime Environment的窗口，直接关掉（这是让选择启动引导等的，我们已经下载的库里有，所以跳过这一步）。点红框标出的图标，进入Manage Project Items窗口。

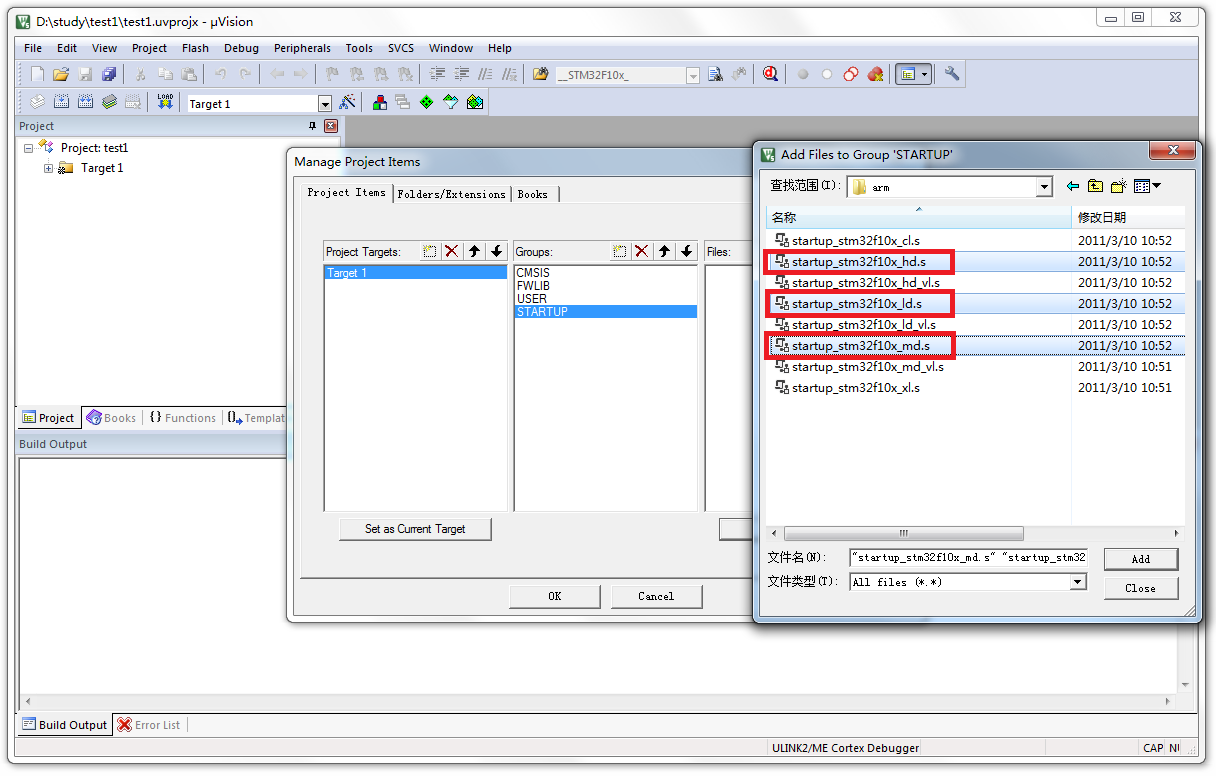


双击tegart1，将其改为test1。删除掉SourceGroup1。在Groups中依次添加CMSIS、FWLIB、USER、STARTUP，并依次单击Add Files按钮进行文件添加。具体添加细则如下：

CMSIS添加除startup文件夹以外的所有.c文件。

FWLIB中只需要添加src中的.c文件即可。

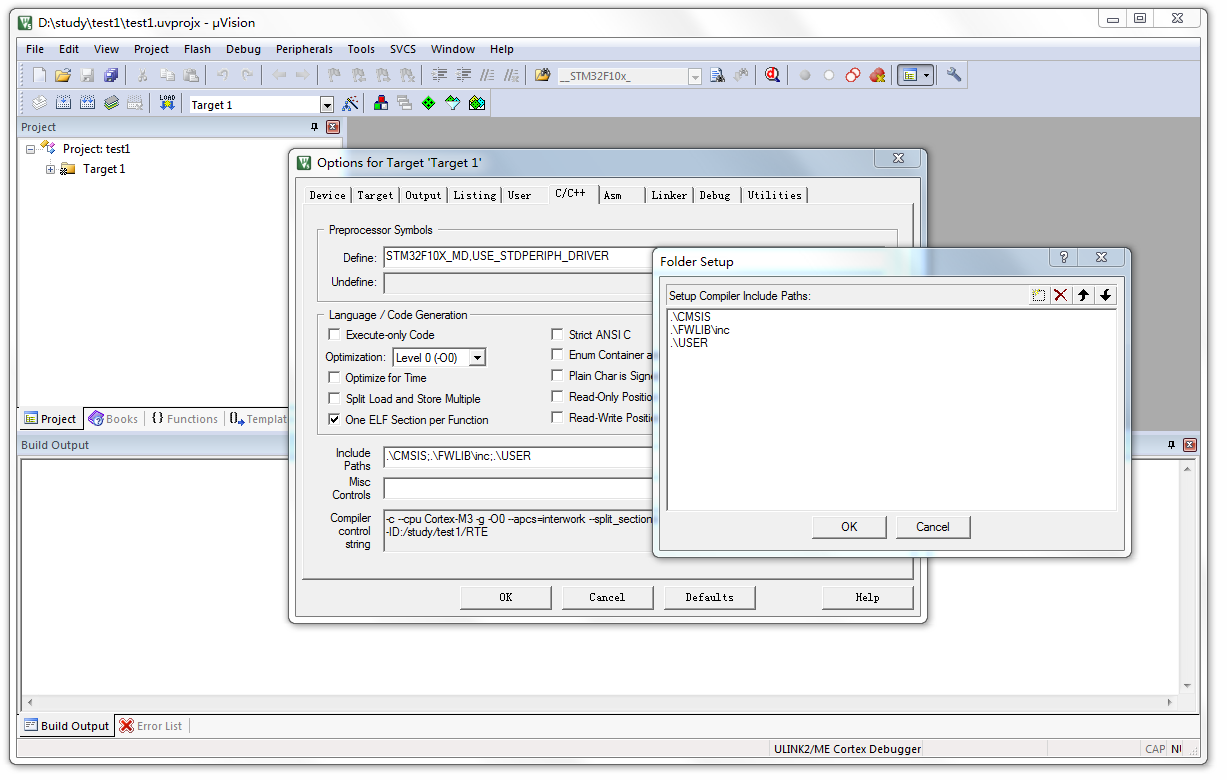
STARTUP首先将文件类型选择为“All Files”，然后选择NewProject\CMSIS\startup\arm中的startup\_stm32f10x\_hd.s、startup\_stm32f10x\_ld.s、startup\_stm32f10x\_md.s三个文件。添加完毕点击OK。



至此工程文件添加完毕。

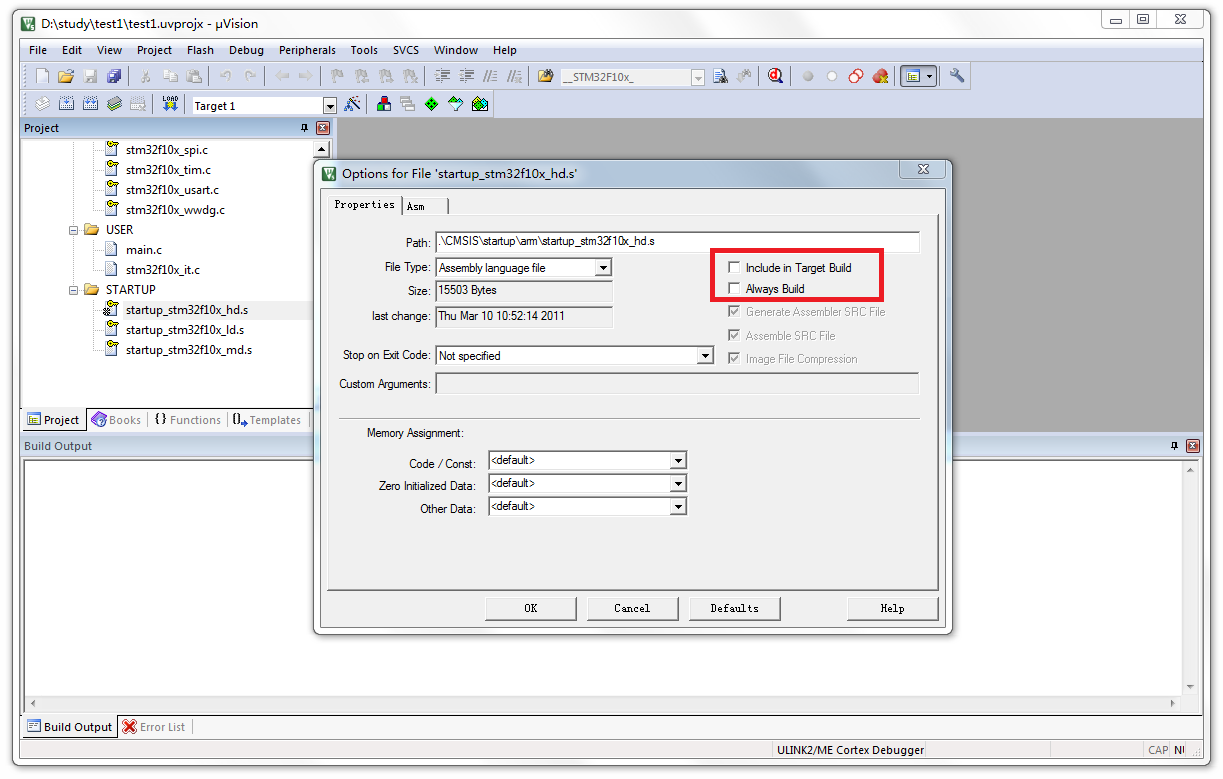
## 工程配置

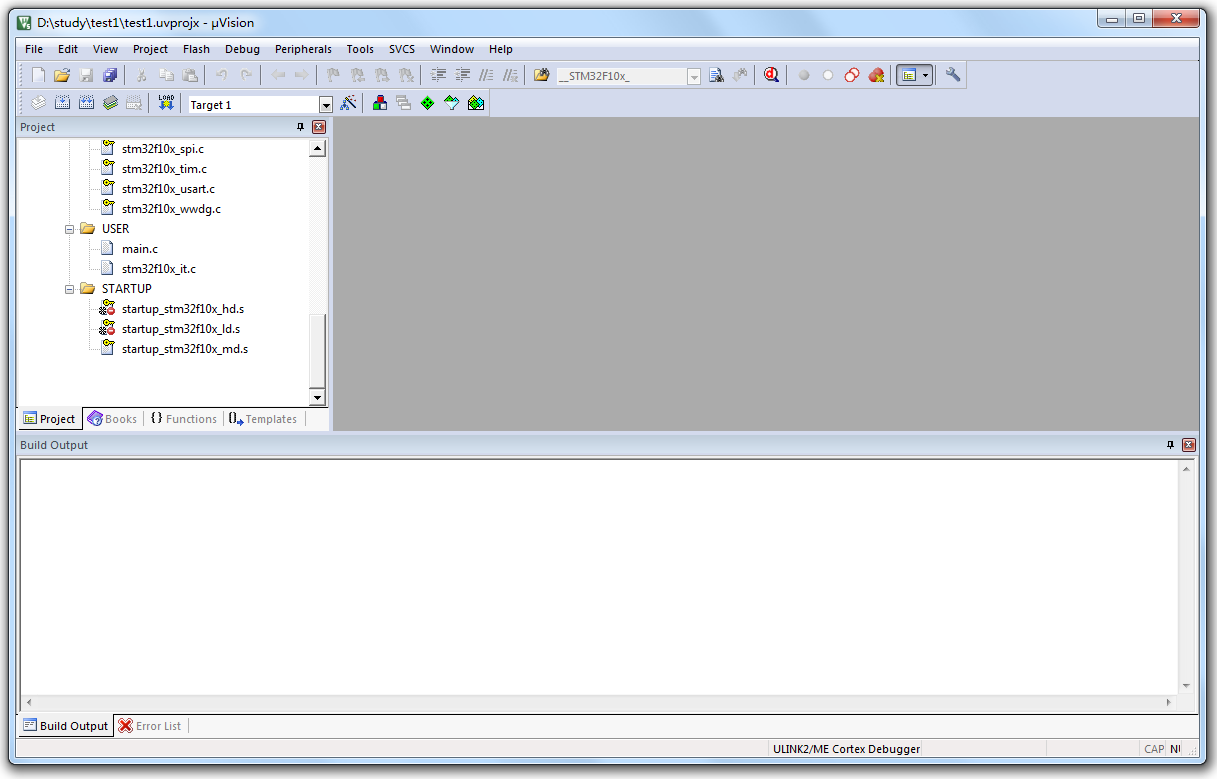
单击图中所示魔法棒图标，进入C/C++设置[界面](https://www.baidu.com/s?wd=%E7%95%8C%E9%9D%A2&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "_blank)，由于STM32F103C8为中等容量的STM32F10X的单片机，因此在define一栏输入STM32F10X\_MD,USE\_STDPERIPH\_DRIVER。然后在includepath栏加入头文件路径。单击OK后保存退出。



## 工程文件配置

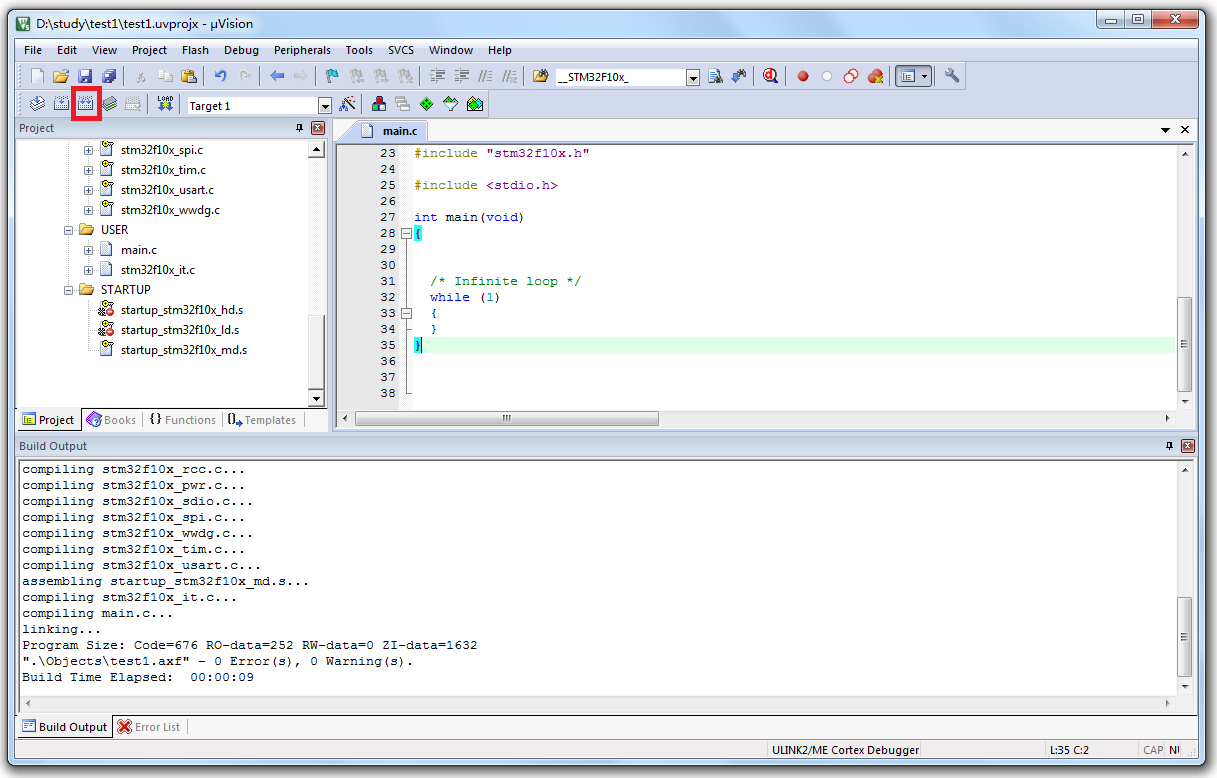
展开左侧Project管理菜单的STARTUP，右键依次单击startup\_stm32f10x\_hd.s、startup\_stm32f10x\_ld.s文件，选择Options for File’...’,将弹出窗口右上侧两个灰色对号去掉，单击OK。对上面个文件进行以上上操作后，Project菜单中这两个文件的前面会出现红色的杠杠。





## 编辑main.c文件、编译工程

双击打开main.c文件，删除其中不需要的代码，将文件修改为仅含有如图所示代码的空main函数。保存并点击编译按钮，".\Objects\test1.axf" - 0 Error(s), 0 Warning(s).编译成功！

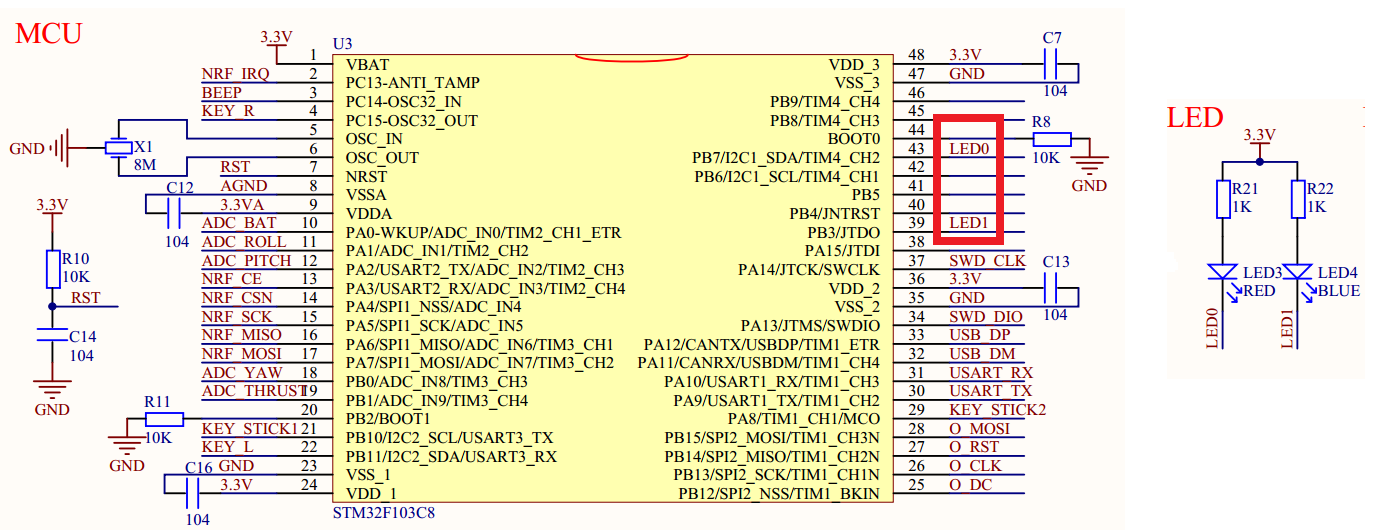


初次建立新工程的，可能会觉得过程有些繁琐，但第二步准备工作中的三个文件夹可以直接复制到其他新工程中使用，多练习几次新工程建立以后，keil5编译环境的使用也就不会觉得有什么困难了。

# 嵌入式C语言开发及技巧

完成空main函数的编译就证明工程文件和工程配置没有问题，现在就能够开始编写第一个自己的小程序了，首先我们编写一个简单的I/O口LED闪烁程序，在正点原子的MiniFly Remoter上来验证一下吧！

从正点原子的MiniFly Remoter的原理图上可以看到，板子上的两个LED分别与STM32F103C8的PB3和PB7管脚相连，当PB7为低电平时红色LED点亮，PB3为低电平时蓝色LED点亮，而当PB3和PB7为高电平时，这两个LED均为熄灭状态。



下面是在keil5上实现的红色LED和蓝色LED同时闪烁的例子，代码如下。

/\* Includes ------------------------------------------------------------------\*/

#include "stm32f10x.h"

#include <stdio.h>

/\* 自定义函数声明 ------------------------------------------------------------\*/

void RCC\_Configuration(void);

void GPIO\_Configuration(void);

void delay(vu32 n);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名： main

\*函数描述： main函数

\*输入参数： 无

\*输出参数： 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(void)

{

vu32 n = 2000000;

RCC\_Configuration();//RCC配置

GPIO\_Configuration();//GPIO配置

while(1)

{

/\*PB3、PB7输出低电平，点亮两个LED\*/

GPIO\_ResetBits(GPIOB, GPIO\_Pin\_3);

GPIO\_ResetBits(GPIOB, GPIO\_Pin\_7);

delay(n);//延时

/\*PB3、PB7输出高电平，熄灭两个LED\*/

GPIO\_SetBits(GPIOB, GPIO\_Pin\_7);

GPIO\_SetBits(GPIOB, GPIO\_Pin\_3);

delay(n);//延时

}

return 0;//实际程序不会运行至此

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名： RCC\_Configuration

\*函数描述： 设置系统时钟

\*输入参数： 无

\*输出参数： 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void RCC\_Configuration(void)

{

ErrorStatus HSEStartUpStatus;

RCC\_DeInit();

RCC\_HSEConfig(RCC\_HSE\_ON);

HSEStartUpStatus = RCC\_WaitForHSEStartUp();

if(HSEStartUpStatus == SUCCESS)

{

RCC\_HCLKConfig(RCC\_SYSCLK\_Div1);

RCC\_PCLK2Config(RCC\_HCLK\_Div1);

RCC\_PCLK1Config(RCC\_HCLK\_Div2);

FLASH\_SetLatency(FLASH\_Latency\_2);

FLASH\_PrefetchBufferCmd(FLASH\_PrefetchBuffer\_Enable);

RCC\_PLLConfig(RCC\_PLLSource\_HSE\_Div1, RCC\_PLLMul\_9);

RCC\_PLLCmd(ENABLE);

while(RCC\_GetFlagStatus(RCC\_FLAG\_PLLRDY) == RESET);

RCC\_SYSCLKConfig(RCC\_SYSCLKSource\_PLLCLK);

while(RCC\_GetSYSCLKSource() != 0x08);

}

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名： GPIO\_Configuration

\*函数描述： 配置GPIO各管脚工作模式

\*输入参数： 无

\*输出参数： 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void GPIO\_Configuration(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

/\*设置GPIO PB3和PB7为强推挽输出，最大翻转频率为50MHz\*/

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_3 | GPIO\_Pin\_7;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

GPIO\_Init(GPIOB, &GPIO\_InitStructure);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名： delay

\*函数描述： 延时

\*输入参数： vu32 n，改变n的大小可调整延时时长

\*输出参数： 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void delay(vu32 n)

{

while(n--);

}

# 嵌入式软件开发与调试技巧



