

- 1: 下载 STM32_V3.5的固件库 去论坛上找，很多
- 2: 准备 Keil uVision4 软件，并安装到电脑上。
- 3: 不要带板凳了，带上你的脑袋就行，因为板凳不会思考。

开始:

- 1: 首先解压缩下载的固件库（保留一个备份，你懂的）

里面有，

_htmresc : ST 的 logo 完全无用，不用理会，

Libraries: 比较重要的文件 包含 STM32的系统文件和大量头文件，也就是库文件了。

Project: 包含大量外设的例程，和各个软件版本的评估版工程模板。KEIL 对应的就是 MDK-ARM 文件下的工程模板。你也可以利用这个工程模板来修改，得到你自己的工程模块，本文不用此法。

Utilities: 就是评估版的相关文件：本文也不会用到，无视既可。

这四个文件，（先去掉文件的只读属性吧，相信你会的）

2: 按照一般的方法，建立工程模板先建立一些文件夹，比如工程模板要建在 D 盘，下面的 D:\STM32\PRO1（项目名字，自己随便定）再该文件夹下面新建以下文件夹 **Libraries**: 直接复制上述的 Libraries 文件夹，把其中的 CMSIS 剪切出来，放到 PRO1 目录下，直接成为另一个文件夹。另外把 STM32F10x_StdPeriph_Driver 下的 inc 和 src 文件夹剪切出来，放在 **Libraries** 目录下，STM32F10x_StdPeriph_Driver 文件夹就可以删除了。会发现里面就只剩下头文件了。

CMSIS: 就是从上面粘贴来的。在 CMSIS\CM3\DeviceSupport\ST\STM32F10x 目录下直接将 Startup 文件剪切出来，放在 **Libraries** 目录下，其他的不需要动。里面存放的就是重要的系统文件，先不要理会是什么作用吧，慢慢就明白了。

Startup 就是从上面粘贴来的。我们要用的比如是：STM32F103VC，只要把 startup\arm 目录下的 startup_stm32f10x_hd.s 文件剪切出来，放到 **Startup** 下面就好。**Startup** 只要这个文件，其它全删吧。这个就是启动文件，根据芯片存储容量来选择。

USER: 请在里面放个 main.c 的文件，里面就一个 WHILE (1) 循环就好，但请加上这个头文件：#include "stm32f10x.h"。另外从固件库 Project\STM32F10x_StdPeriph_Templat 目录下复制 stm32f10x_conf.h, stm32f10x_it.c, stm32f10x_it.h。这四个 h 文件是 **Libraries** 里面没有的，理论上要自己写好的，但是固件库里有就直接复制吧，呵呵。另外你还发现那个目录下有个 system_stm32f10x.c 文件，有人问，这个怎么不一起复制了，很简单，因为 **CMSIS** 这个文件夹里面已经有了，个人认为还是放在这里比较好，因为是系统文件。

Project : 这个文件夹用来存放项目文件。比如你等下要建立的工程名字 TEST.uvproj, 就放在这里。在该文件下存放一些项目的输出信息，OUTPUT, 和 LISTING 信息，可以建立这两个文件夹。等下就可以在 OPTIONS FOR TARGET 'PROJECT' 里面进行路径设置。

文件夹分配到此结束，为什么要这么分配文件夹呢，其实是个人理解的问题，我这样的目的是为了比较清楚，也符合一般程序员的设置方法。等你了解了，随便安排文件下都可以。下面开始建立工程模板。

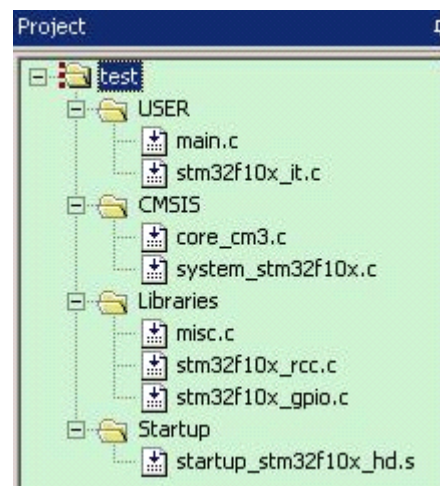
3: 打开 KEIL 软件，建立新工程。

比如建 TEST.uvproj, 保存在刚才的 **Project** 文件夹下面。

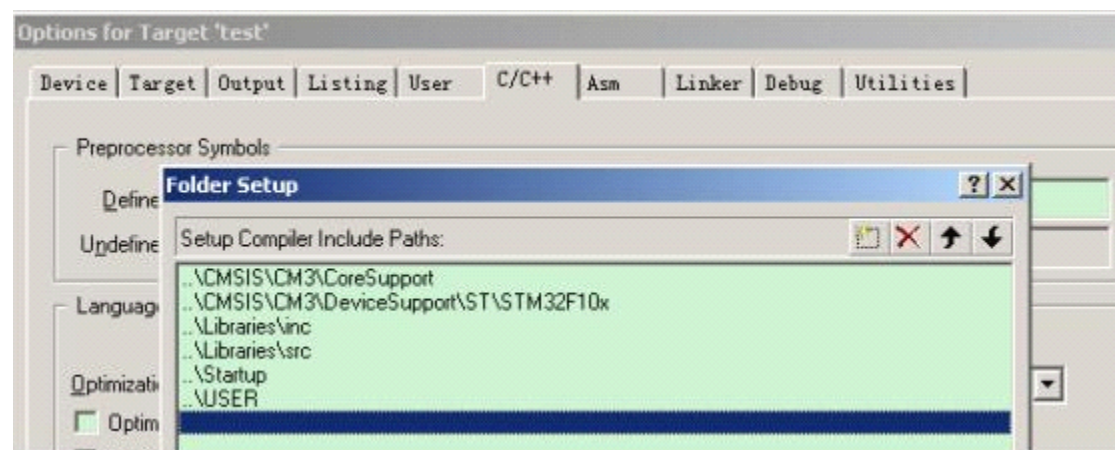
选择你使用的 STM32 芯片，比如我用的是 103VC 的，就选择那个，确定后，会跳出来个窗口，问你是否复制启动代码，选择否，因为我们已经弄好了，前面的准备工作不是白做的。

然后右键 TARGET 1，选择 Manage Components, 把文件给加进去吧。

来图片了：加好文件后，出现如下的文件架构。（按下图加文件吧，每个文件都在相应的文件夹里，前面的工作不是白做的。）



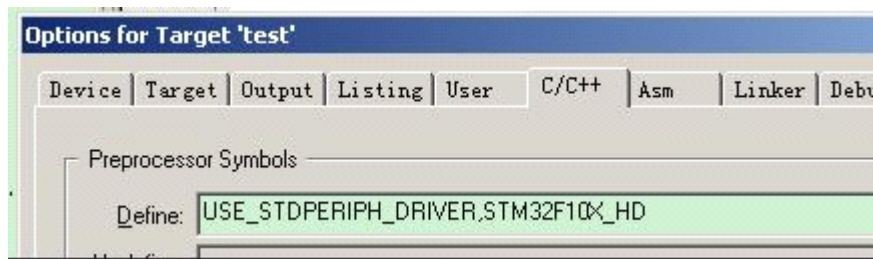
接下来要设置编译文件的路径，设置好后如下图：



现在编译，还是有错误的。可以先编译下，

因为 main.c 中有 `#include "stm32f10x.h"` 这句，在这个重要的头文件里，并没有定义好一些东西，分别在 stm32f10x.h 的 0070 行，取消对 `#define STM32F10X_HD` 的屏蔽，0105 行 取消对 `#define USE_STDPERIPH_DRIVER` 的屏蔽，这两个定义好，才能正常工作，第一个是选择存储容量的型号，每种芯片都不太一样，所以让你选择，第二个是使用标准库文件，因为我们是想要用库的，所以必须选择，默认是不用库的，直接进行寄存器的操作。像初学者是用库比较好，所以肯定选择的。

也可以直接如下图操作。



，作用是一样的。

选择好后，直接编译，恭喜你，成功了。

4: 最后要说明的就是:



这个库文件，是必须选择 misc.c 和 stm32f10x_rcc.c 的。

另外就是你选择用什么功能，就把什么功能的 c 文件加进去，我这里已经加了一个对 IO 口的应用的一个 c 文件。

5: 一些重要文件的说明吧

core_cm3.c/core_cm3.h

该文件是内核访问层的源文件和头文件，查看其中的代码多半是使用汇编语言编写的。

stm32f10x.h

该文件是外设访问层的头文件，该文件是最重要的头文件之一。例如向量等等。除了这些该头文件还定义了和外设寄存器相关的结构体，如：

```
typedef struct
{
    __IO uint32_t CRL;
    __IO uint32_t CRH;
    __IO uint32_t IDR;
    __IO uint32_t ODR;
    __IO uint32_t BSRR;
    __IO uint32_t BRR;
    __IO uint32_t LCKR;
} GPIO_TypeDef;
```

包含了那么多寄存器的定义，那么在应用文件中（例如自己编写的主源文件）只需要包含 stm32f10x.h 即可。

在 stm32f10x.h 8296行

```
#ifndef USE_STDPERIPH_DRIVER
    #include "stm32f10x_conf.h"
#endif
这里定义了 #include "stm32f10x_conf.h"
```

system_stm32f10x.c/h

该头文件也可以称为外设访问层的头文件和源文件。在该文件中可以定义系统的时钟频率，定义低速时钟总线 and 高速时钟总线的频率，其中最关键的函数就是 SystemInit()了，这个后面会详细介绍。总之这两个文件是新固件库 V3.5的重点，有了它也就大大简化了使用 stm32的初始化工作。

stm32f10x_conf.h

这个文件和 V2 版本的库的内容是一样的，需要使用哪些外设就取消哪些外设的注释。

stm32f10x_it.c/h

这两个文件包含了 stm32中断函数，在源文件和头文件中并没有把所有的中断入口函数都写出来，而只写了 ARM 内核的几个异常中断，其他的中断函数需要用户自己编写。stm32f10x_it.c 的最后给了这样一个模板。

```
/*
 * STM32F10x Peripherals Interrupt Handlers
 * Add here the Interrupt Handler for the used peripheral(s) (PPP), for the
 * available peripheral interrupt handler's name please refer to the startup
 * file (startup_stm32f10x_xx.s).
 */
/**
 * @brief This handles PPP interrupt request.
 * @param None
 * @retval None
 */
/*void PPP_IRQHandler(void)
{
}*/
```

从注释中的英文提示可以看出，中断向量的名称可以从相应的启动代码中找出，例如可以在 startup_stm32f10x_md.s 中找到 USART1 中断函数的名称——USART1_IRQHandler。其他的中断函

名可以以此类推，一一获得，在这里我就不一一复述了。

Libraries 文件夹

该文件夹有包含两个文件夹，一个是 src 文件夹，另一个是 inc 文件夹，顾名思义，一个里面放的是元件一个里面放的是头文件。这两个文件夹包含了所有的 STM32 的外设驱动函数，其实和 V2 版本也没有太大的变化。简单来说，外设的驱动相当于 windows 的驱动函数 API，这些驱动函数看到函数名基本就可以明白这个函数的作用，例如：GPIO_SetBits 可以置位某个 IO 口，相反 GPIO_ResetBits 则可以复位某个 IO 口。我个人认为熟练使用库可以大大提高编程的效率，同时规范使用库函数也可以提高程序的可读性，让团队中的其他程序员可以快速的明白代码的作用。