[第一章、建立工程模板 2](#_Toc18058)

[第二章、添加驱动底层 7](#_Toc15282)

[第一步、复制粘贴外设驱动底层到蓝桥模板的CODE文件夹里 7](#_Toc30147)

[第二步、打开已经创建好的模板，在CODE文件里添加这些底层文件的.c文件 7](#_Toc1572)

[第四步，在num.h文件里面添加如下头文件，以及宏定义方式,详情见注释 9](#_Toc14260)

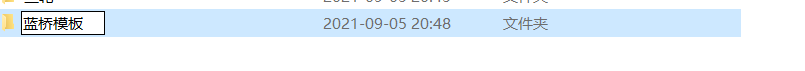
[第五步、将该头文件包含到main.c里，并且添加void main 函数 9](#_Toc24478)

[第六步、添加头文件路径 9](#_Toc17405)

[第七步，添加sys里的文件夹 11](#_Toc3258)

# 第一章、建立工程模板

1. 任意位置新建一个文件夹并命名，

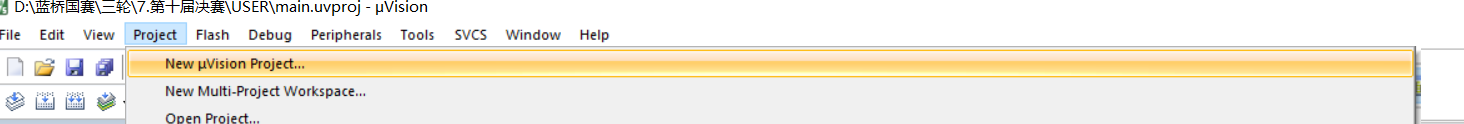


1. 在该文件夹里再任意建立三个文件夹并取名如下

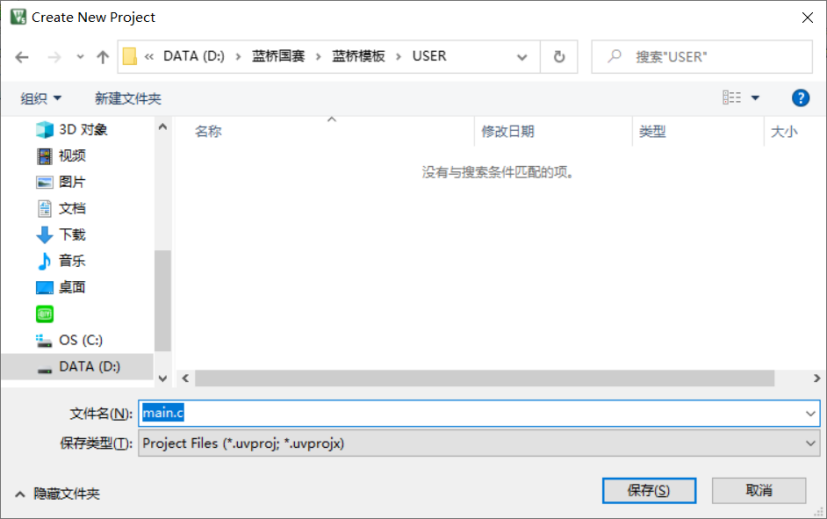


其中CODE文件夹的作用是用于存放外设底层，比如按键，数码管，led灯等底层文件，SYS文件夹用于存放一些定时器底层文件，公用的相关底层，USER文件夹用于存放系统主函数文件，这样分类的好处大家以后接触了嵌入式就明白了，方便程序移植和扩充，增强工程项目的适用性，比如好几个项目我就用一个底层模板，改一下里面的算法就可以完成了。

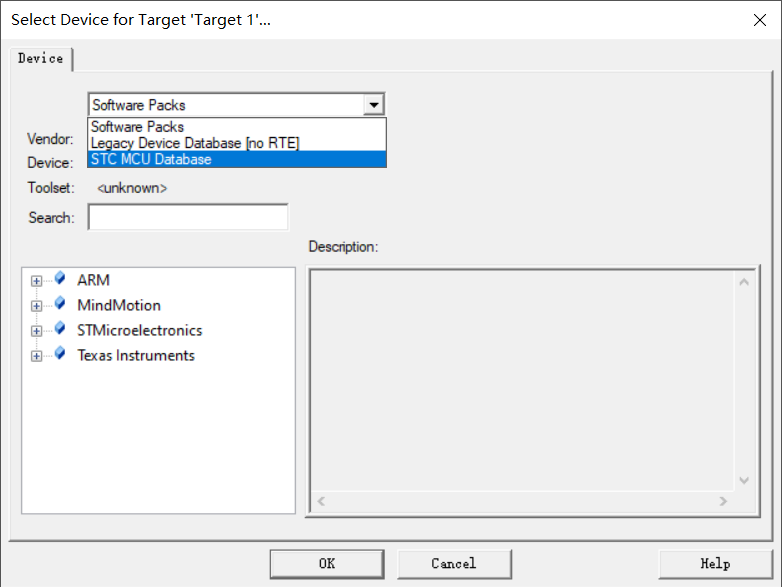
1. 打开keil新建一个工程项目



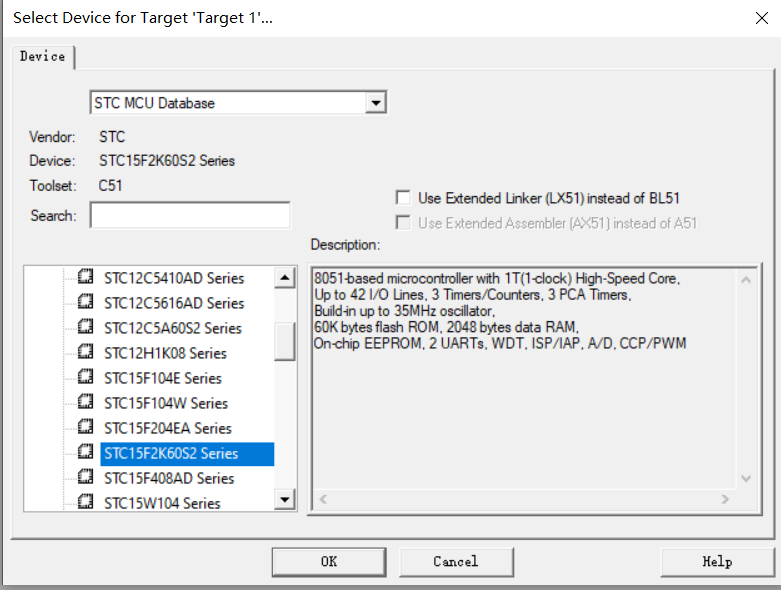
1. 命名为main.c，并存放在之前建立好的USER文件夹里



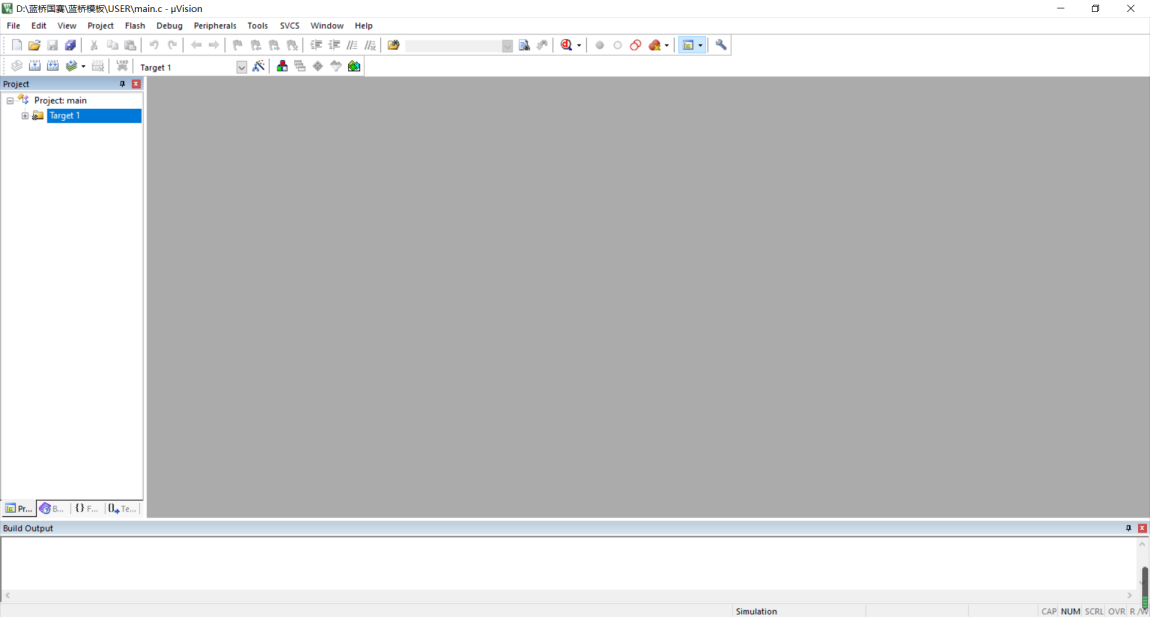
1. 点击保存后出现单片机型号选择界面如下



1. 选择单片机型号如下

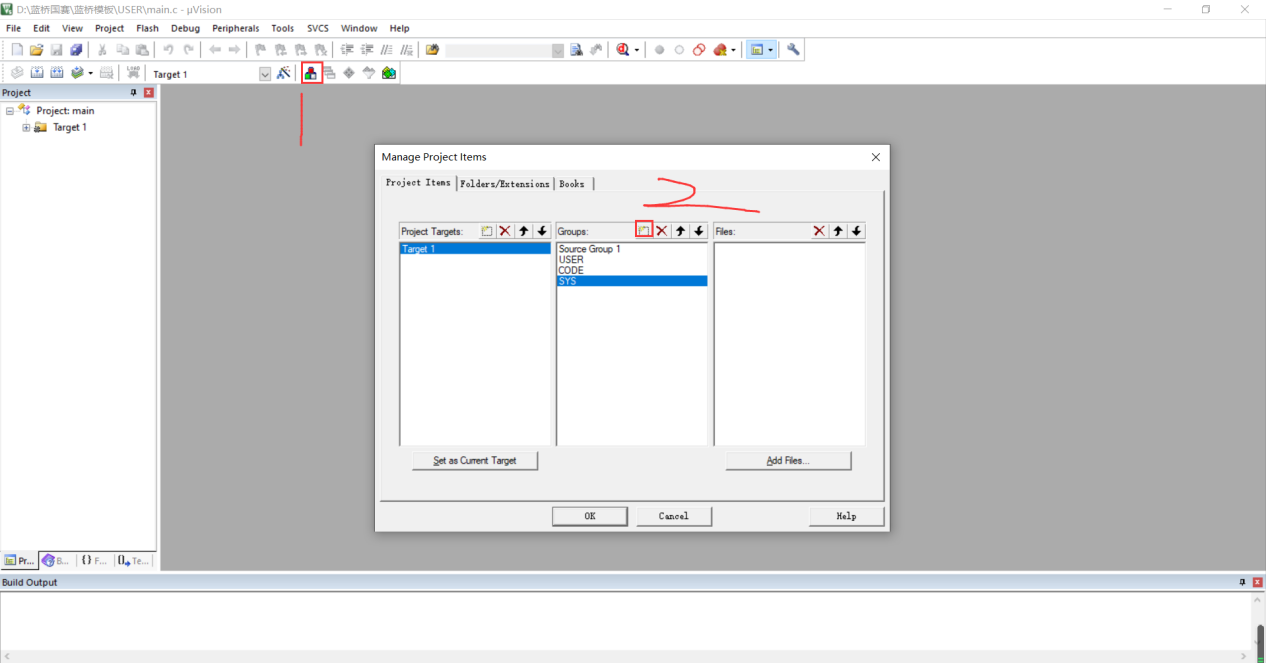


1. 点击ok，如下图

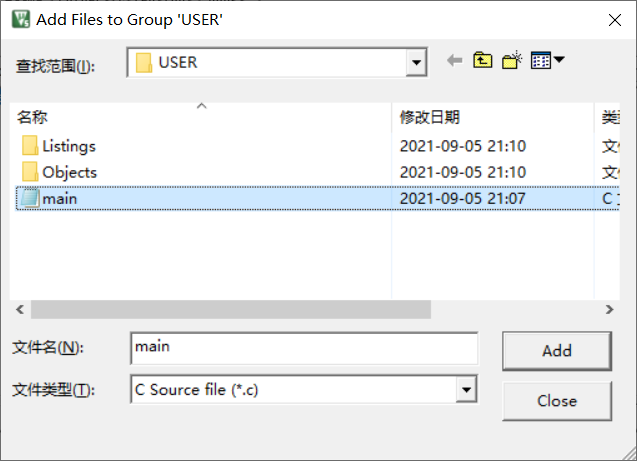


1. 添加USER文件夹里的main.c文件到工程里

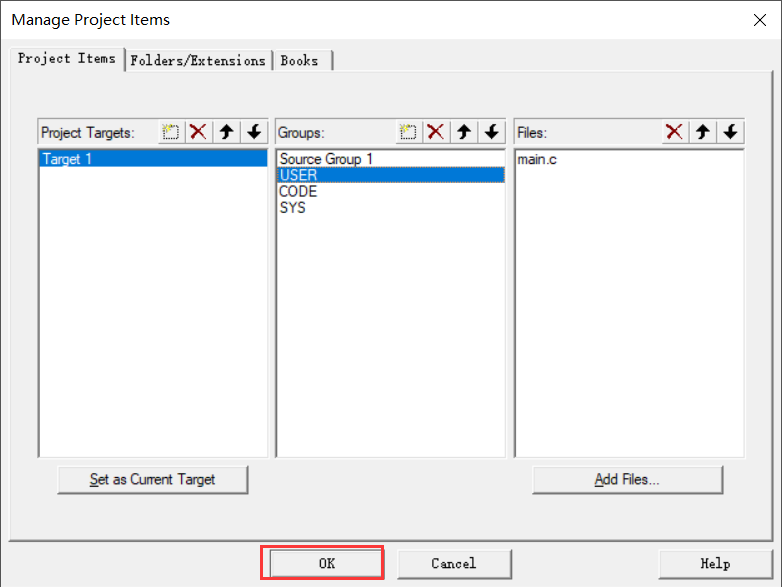
按如下步骤添加三个文件夹，命名与之前蓝桥模板里的三个文件夹命名一致，分别为USER,CODE,SYS



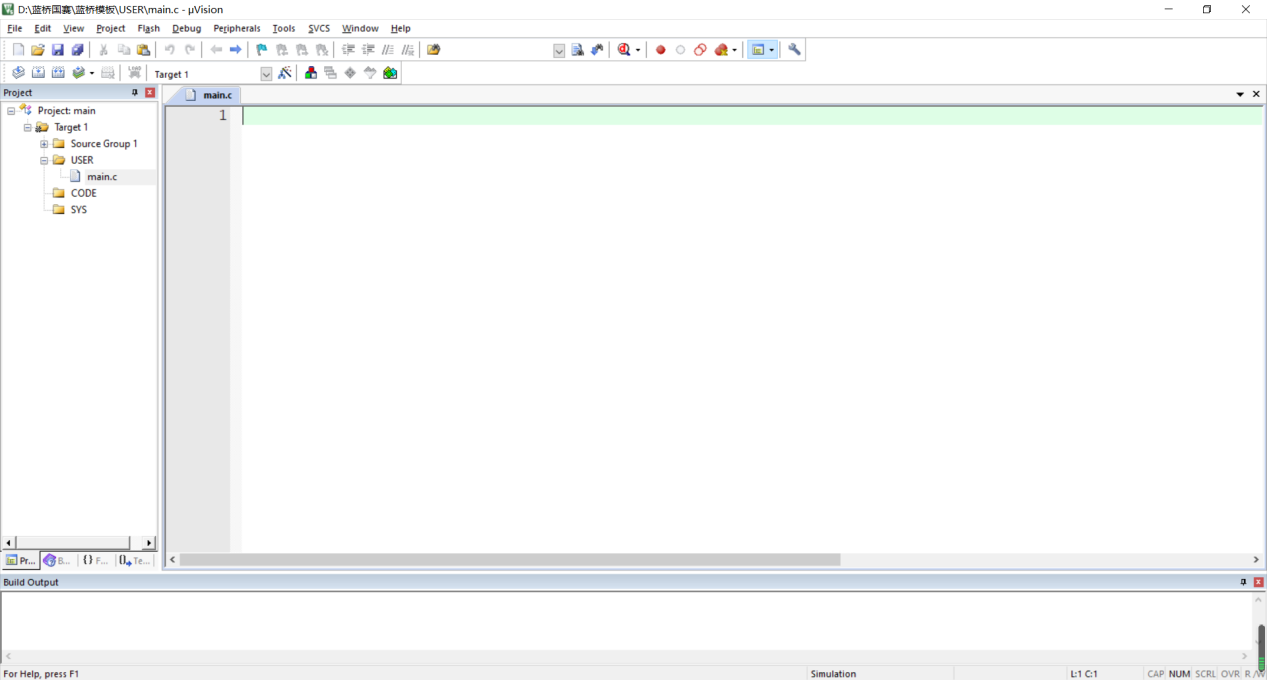
选中USER，点击 Add Files，将USER文件夹里的main.c文件添加进来



点击ok



完成了主函数c文件的添加，如图

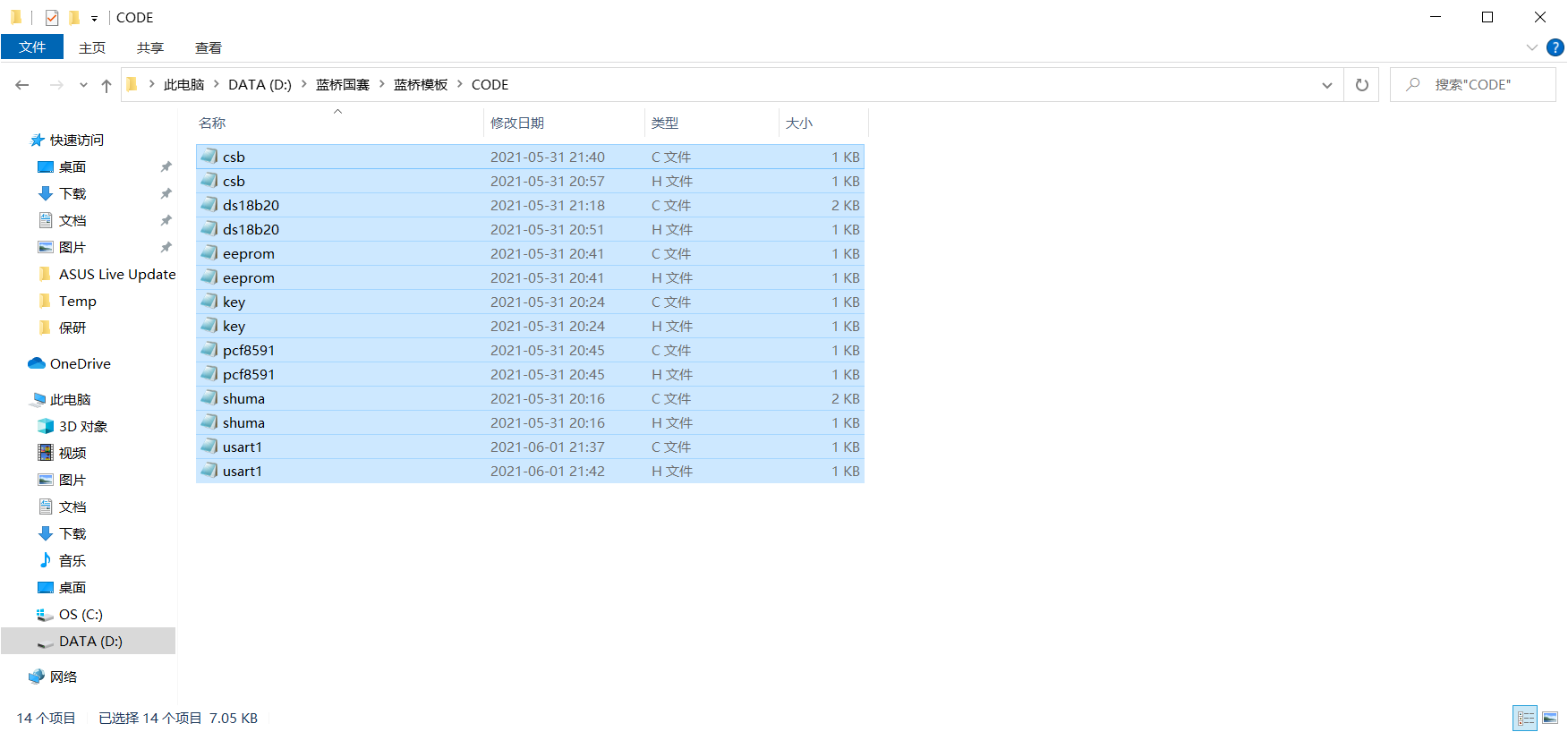


在这个界面下就可以肆意妄为开始编程了，不过为了让大家更好的入门单片机，以及以后学习嵌入式芯片，所以大家跟着我，系统的学习相关的编程方式。

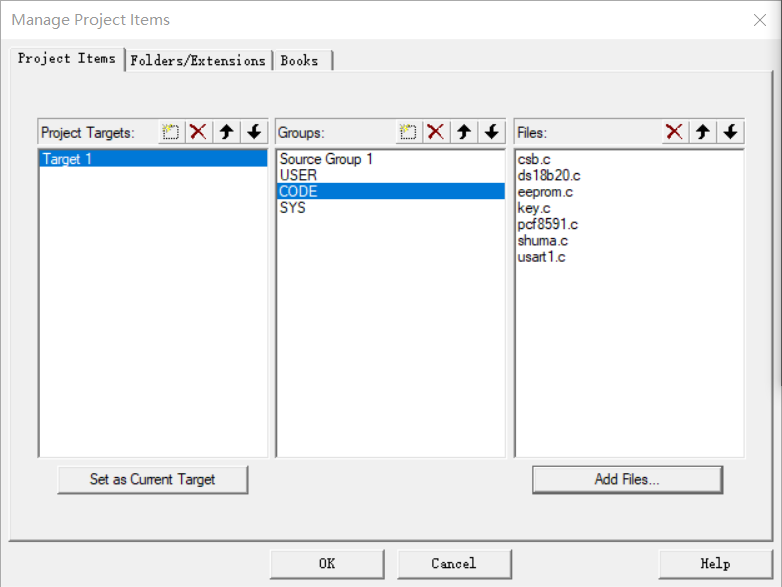
# 第二章、添加驱动底层

为了方便大家学习，所有外设驱动底层我已经写好了，大家现目前只需学会如何添加驱动底层，并调用相关模块来实现各种各样的功能，后续随着大家学习的深入，大家就能自己写底层了。所有驱动底层文件我会上发到培训群里，大家自行下载。

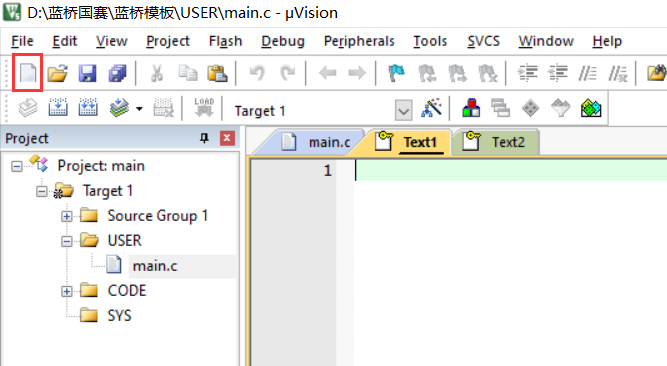
## 第一步、复制粘贴外设驱动底层到蓝桥模板的CODE文件夹里



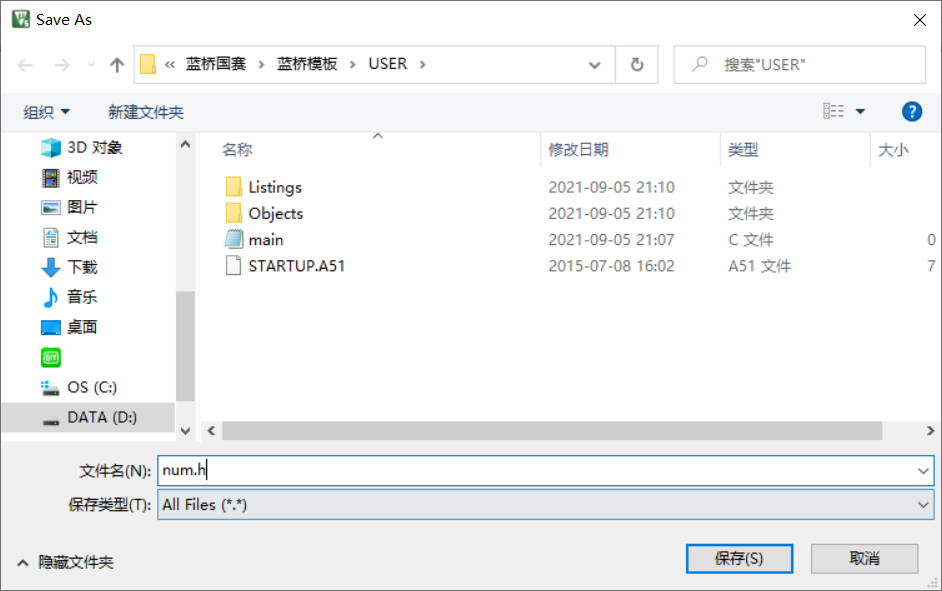
## 第二步、打开已经创建好的模板，在CODE文件里添加这些底层文件的.c文件



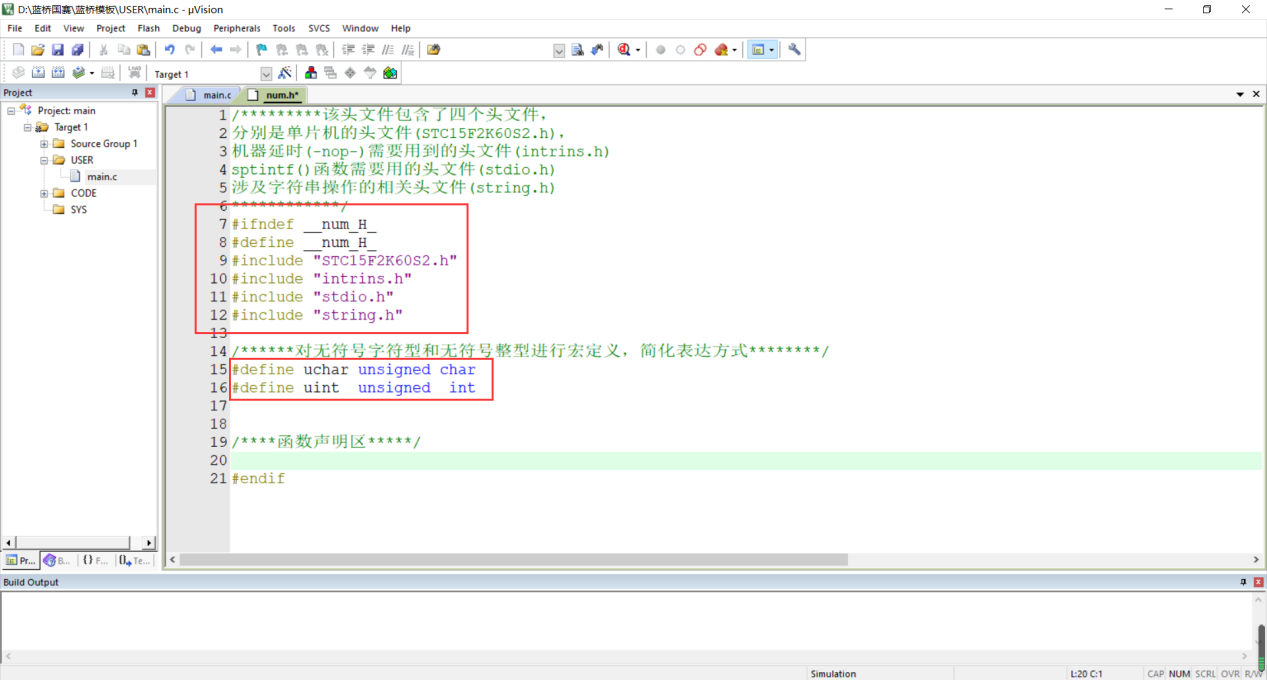
第三步、新建一个text文件



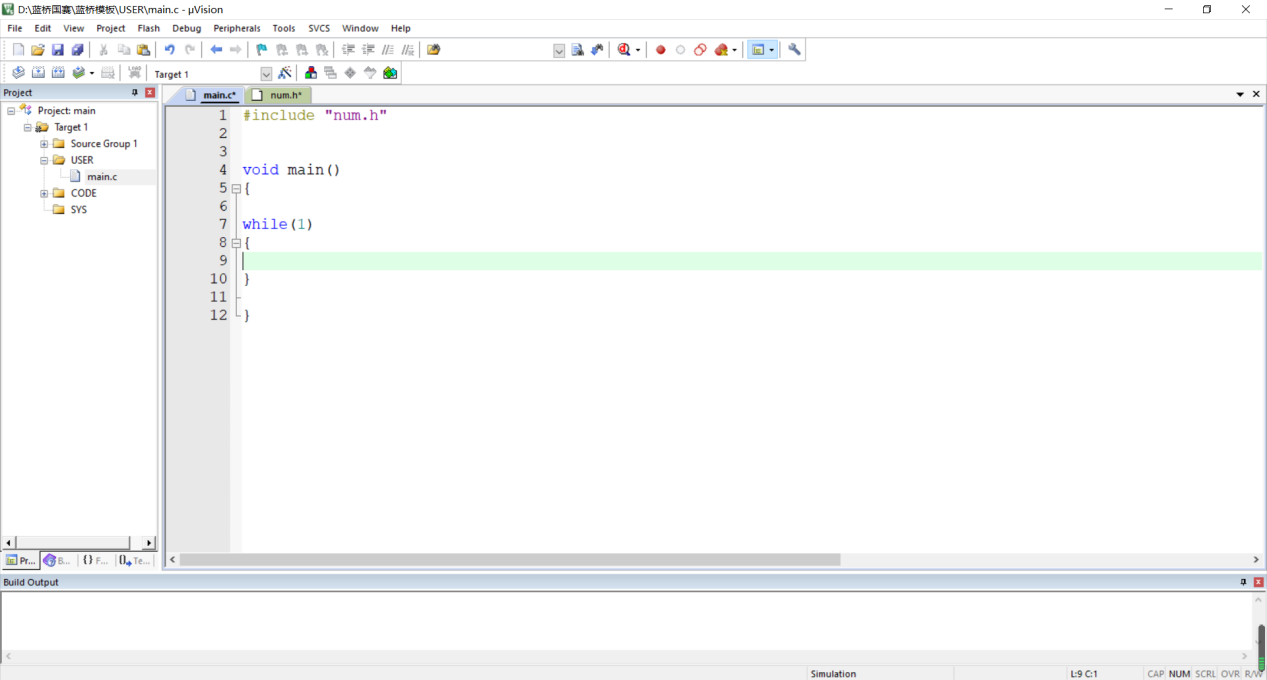
点击保存，并命名为num.h，并存放到USER文件夹下



## 第四步，在num.h文件里面添加如下头文件，以及宏定义方式,详情见注释

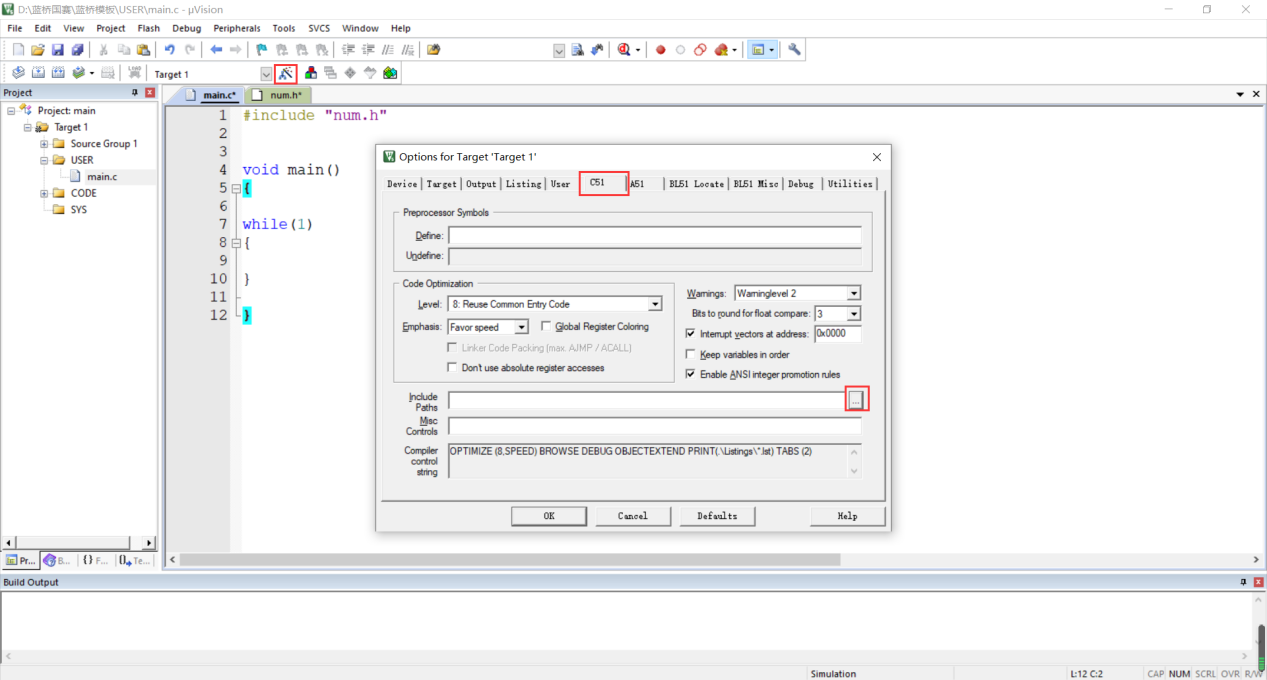


## 第五步、将该头文件包含到main.c里，并且添加void main 函数

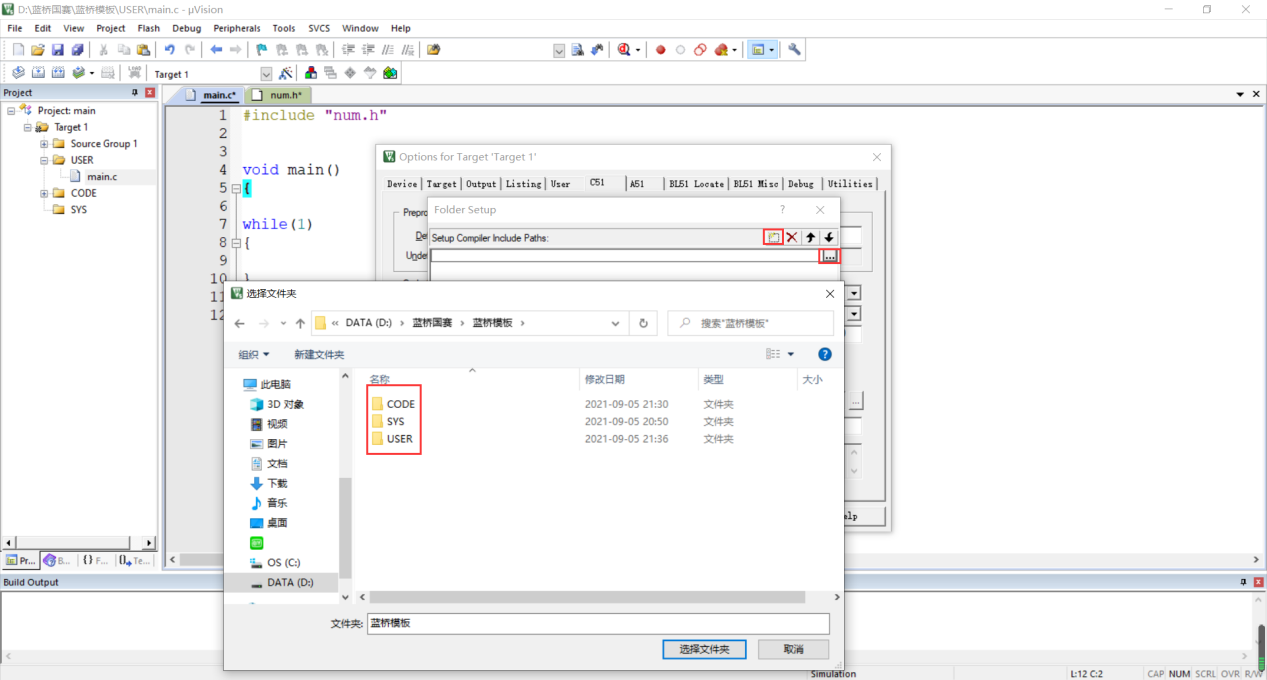


## 第六步、添加头文件路径

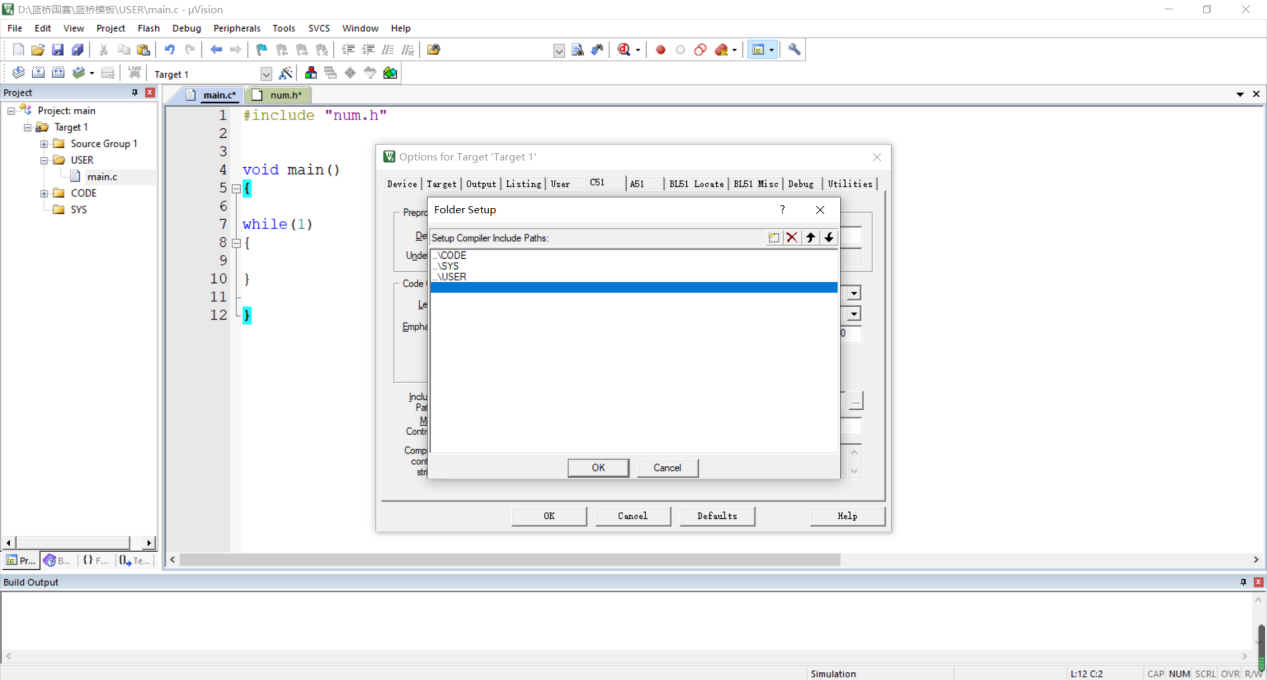
选中那个小锤子，点击C51，点击省略号



点击新建文件夹，点击省略号，将蓝桥模板里的三个文件夹逐次选中并添加

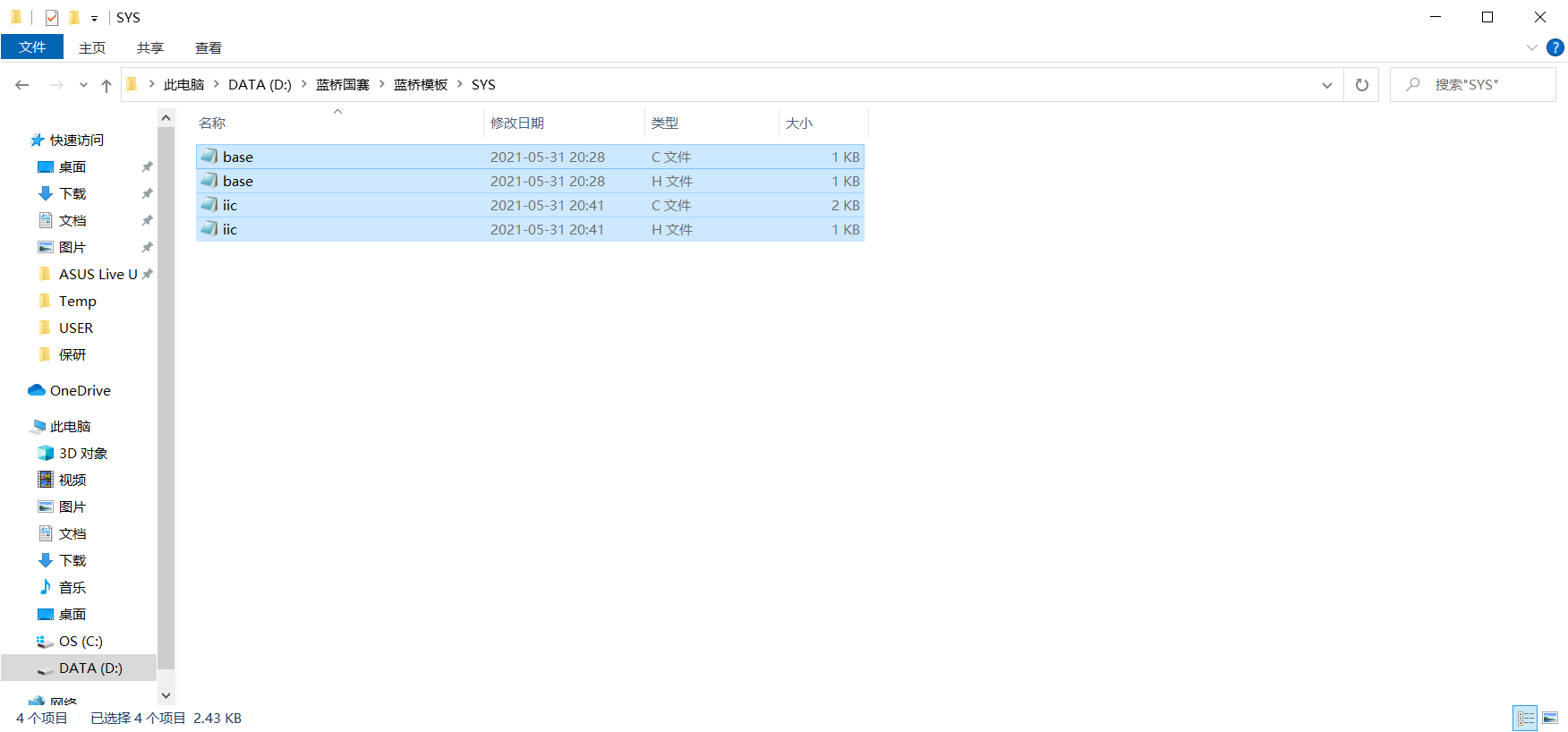


添加完后如图

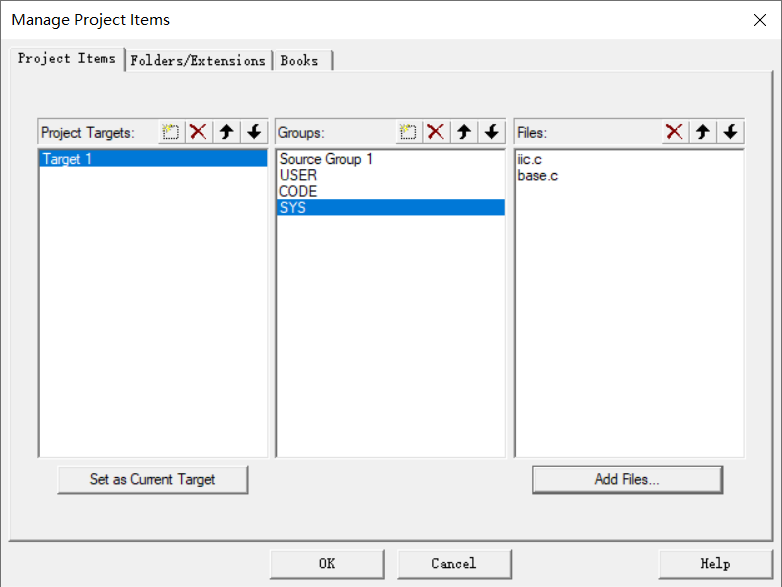


点击ok

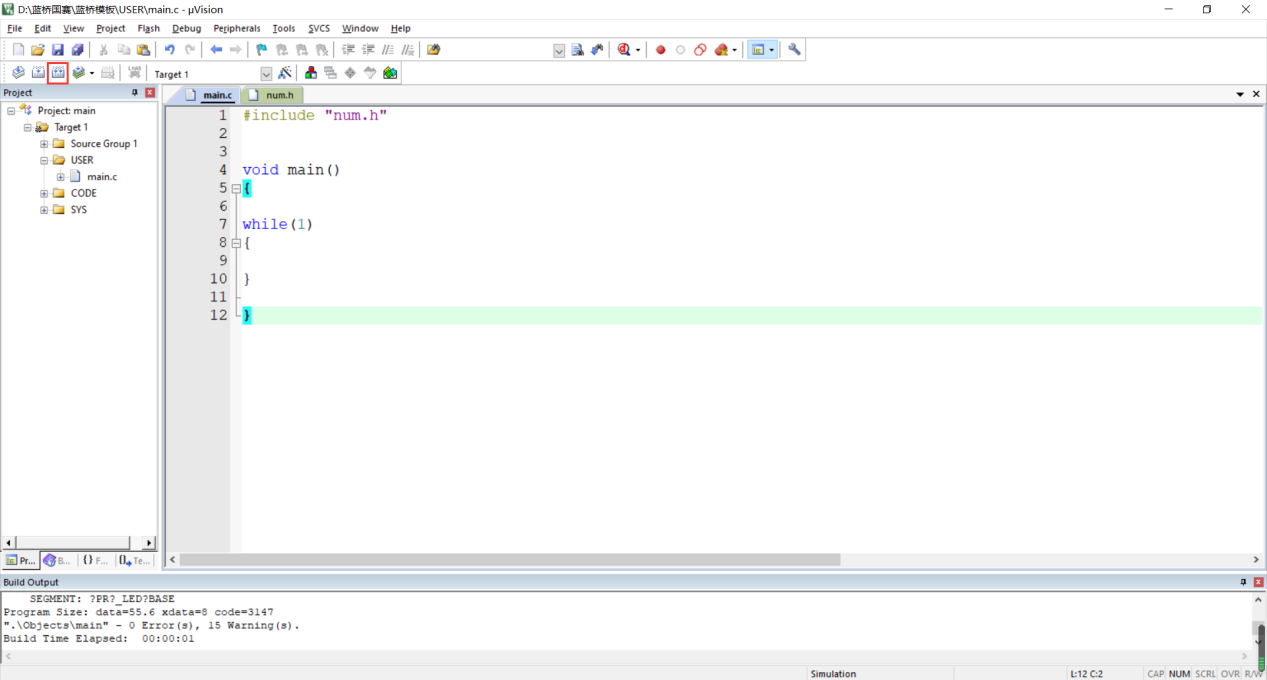
## 第七步，添加sys里的文件夹



同样的方法在keil对应位置里面添加对应的.c文件



点击编译



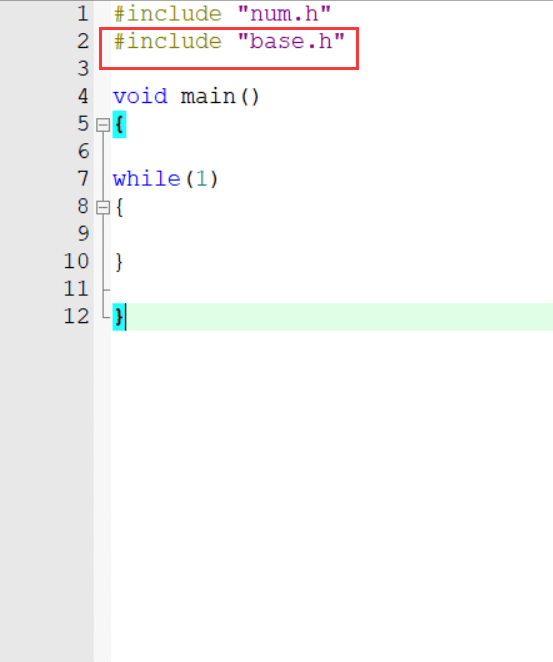
出现15个警告不用管，因为这个15个函数没有调用，不影响后续使用。

至此，模板建立完毕，以下章节是各个模板底层的使用方法。

# 各个外设底层的使用方法

## Led底层的使用

led相关底层封装在了base.c文件中，使用时只需在main.c里包含base.h头文件如图



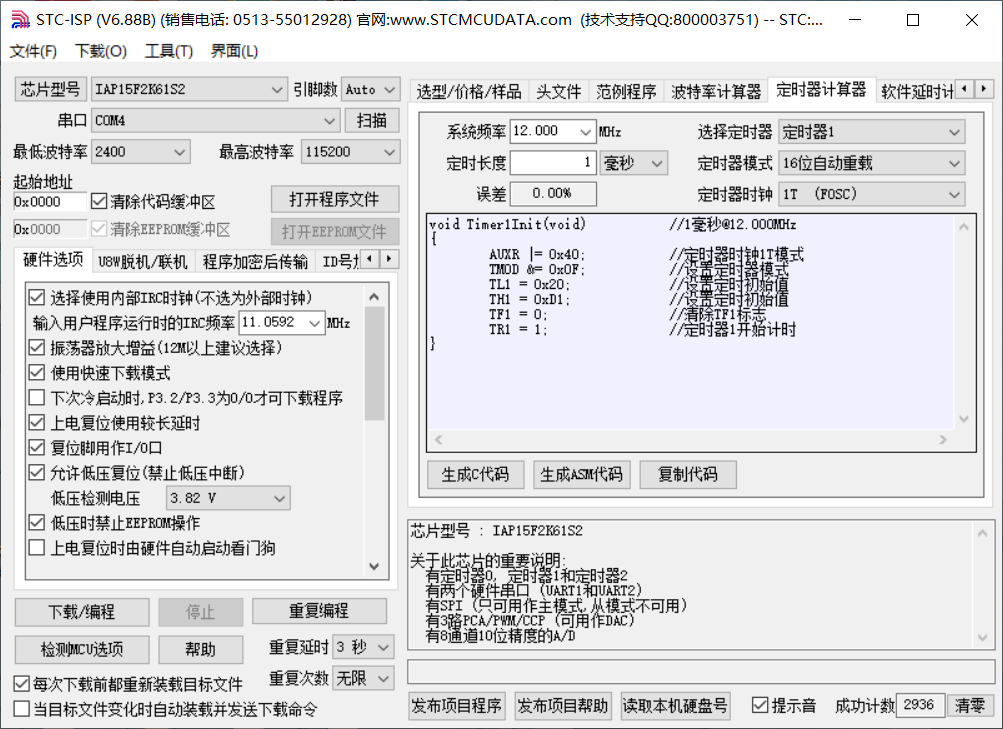
任何一个底层文件的使用首先都是包含头文件,然后主函数里初始化，然后根据相应的.c文件里封装的函数进行调用，这也是这个工程模板的好处之一，傻瓜式的调模块，接下来以led的使用为例，进行详解。

首先，使用之前我们要初始化一个定时器，这里我们为什么要初始化一个定时器呢，首先思考一下这个问题，这个单片机的主频我们比赛的时候一般使用的是12Mhz的，这个主频是比较高的了，这个主频，理论上来说足够我们完成大部分项目的设计了，但是我们得对我们的程序运行资源进行合理的规划，比如说每隔5ms更新一次led显示状态，每隔1ms更新一次数码管显示状态，每隔任意ms去执行一次其他程序比如进行温度采集，AD采集，按键扫描等等，当很多模块要同时让单片机处理的时候，如果不合理规划资源，那么性能再好的单片机也会带不动有些项目，所以这里引入一些操作系统的思想，即合理，分时的控制与管理程序运行。接下来，先详解如何配置一个定时器，其中涉及到一些中断知识，在后续章节会进行详解。

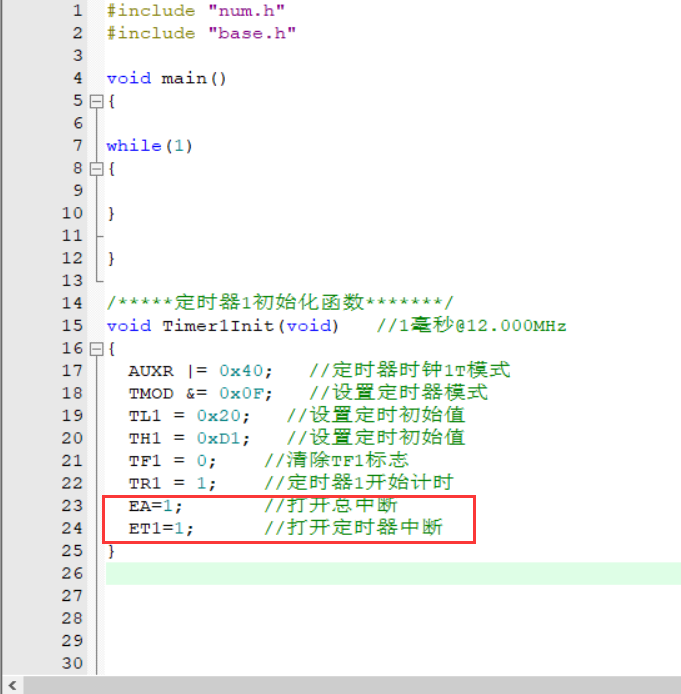
### 定时器的配置

第一步，编写定时器1初始化代码，为什么用定时器1呢？？我们要把定时器0和定时器2的资源预留出来，因为定时器0涉及到后面的ne555测频，定时器2用来给串口产生波特率，当同时用到ne555模块和串口模块时，这样只剩定时器1给我们用了，所以一开始训练的时候就养成这种提前预判和合理规划资源的习惯对我们是很有好处的。废话不多说，上图！

首先打开烧录软件，选择定时器计算器，配置系统频率为12Mhz，选择定时器1，16位自动重装载模式，定时长度为1ms。



复制这些代码到keil里面，



注意这里多了两行代码，因为我们接下来要编写定时器中断函数，所以需要开中断，总中断是管理所有中断资源的所以一但要用到中断必须打开，打开之后，需要用定时器中断就开定时器中断，需要用到什么中断就开什么中断。

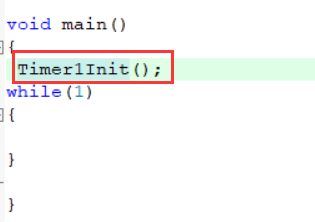
第二步编写定时器1中断服务函数，



注意：我们编写的是1ms中断，即每隔1ms单片机会跳转到中断服务函数里，并运行其中的代码，所以我们在中断服务函数里所运行的代码时间不能超过1ms，否则单片机就相当于一直运行中断里的程序，不会去运行主函数里的程序。这里是一个坑，所以要学会合理的规划程序资源。

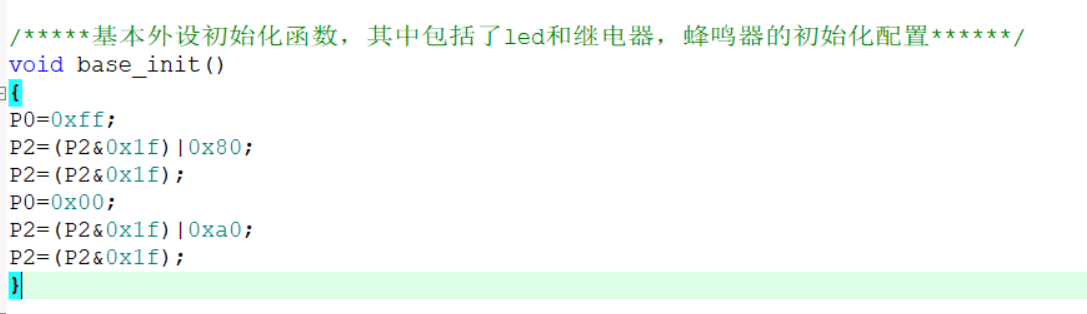
接下来再去num.h文件里面声明定时器初始化函数，然后主函数里调用即可。



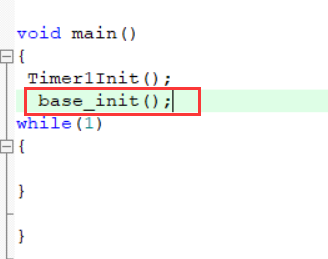


然后我们的定时器1就配置完成了。

接下里调用base.c里的初始化函数



如图调用，复制函数名去主函数里粘贴即可。



### 随心所欲实现led功能

之前的步骤都是一些初始化步骤，接下来我们来实现一些具体的功能。

#### 功能一：点亮任意一个led灯。

构建一个led处理函数，