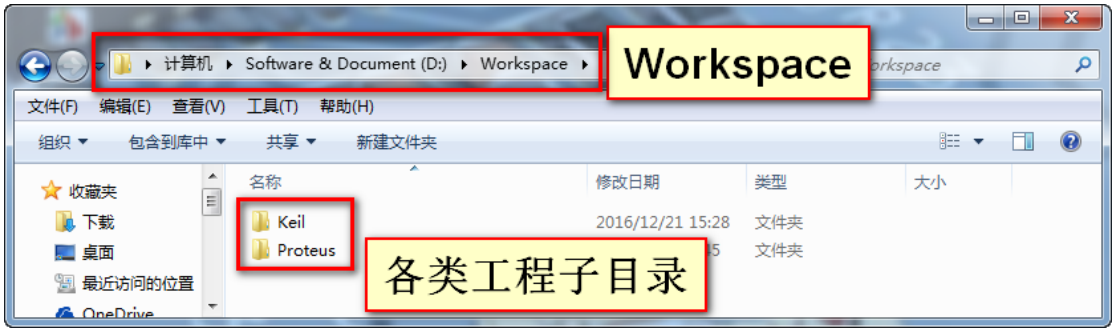
第二章 新建工程模板

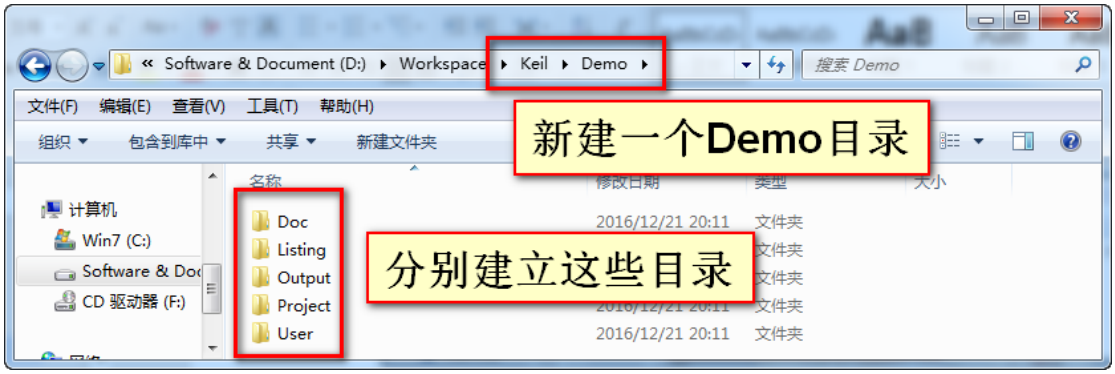
2.1 新建 Keil 工程模板

2.1.1 准备好工作空间

工作空间其实就是代码存放在电脑上的具体路径。在第一章中有要求说建立建立一个 Workspace 目录，用来专门的存储工程文件和代码。并且在 Workspace 目录下有各种工程的子目录，就像这样的：



既然是新建 Keil 工程，那我们进入 D:\Workspace\Keil 目录下,新建一个 Demo 目录。并在 Demo 目录下分别创建 Doc、Listing、Output、Project、User 目录。



简述一下各个目录大致存放的内容：

Doc：存放的是工程相关的参考资料。比如说芯片的数据手册，总结的笔记，工程的简介，程序使用时的注意事项。

Listing：存放 Keil 连接输出的连接文件。

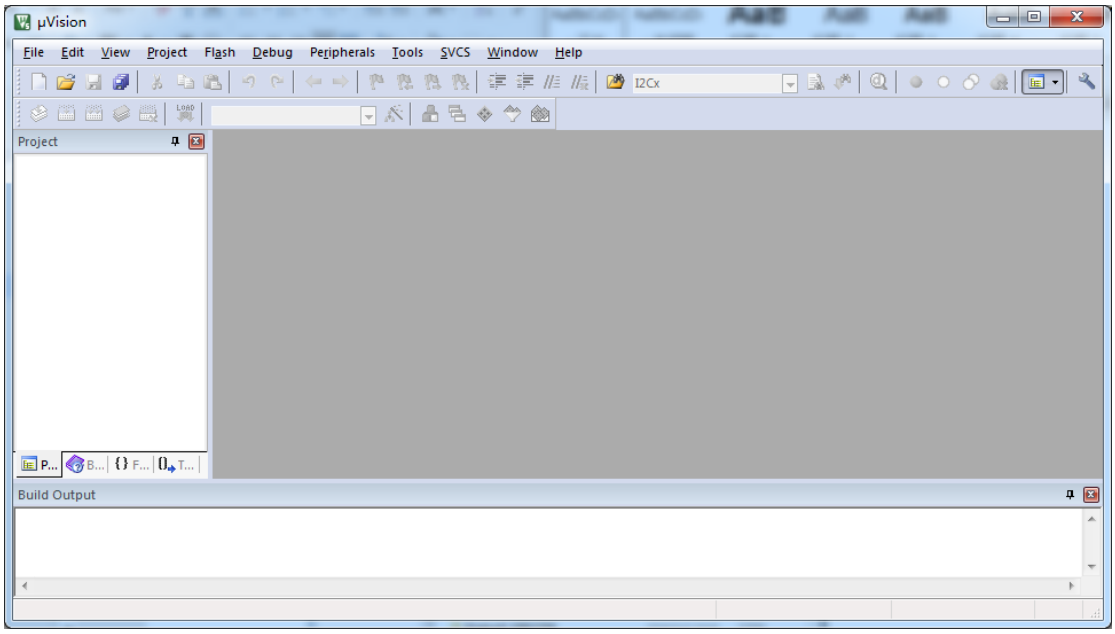
Output：存放 Keil 编译产生的输出文件。

Project：存放 Keil 的工程配置文件。

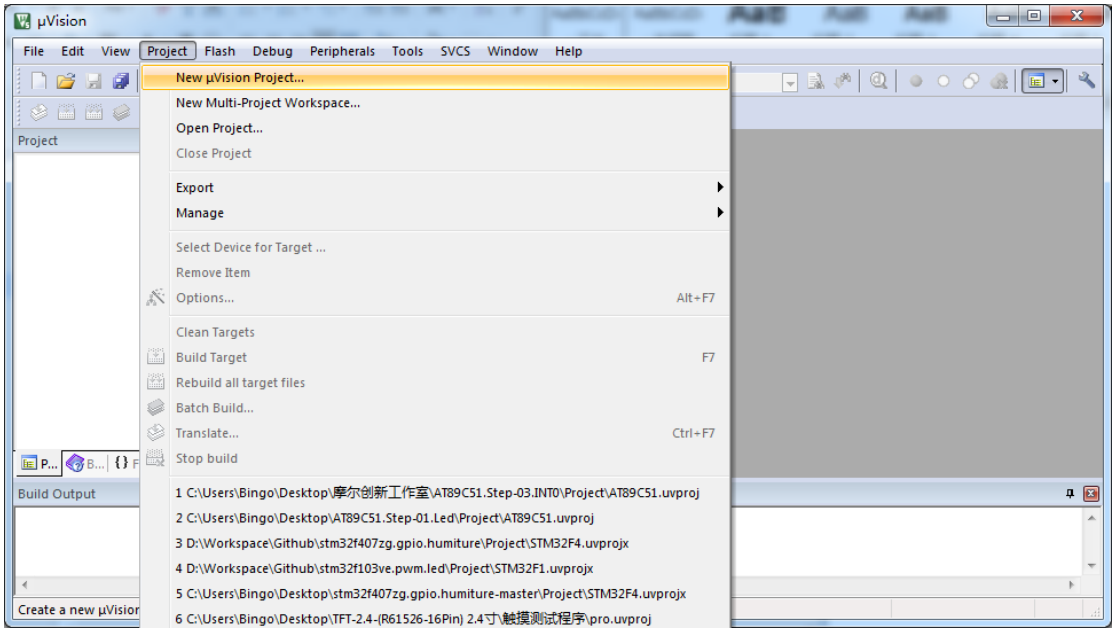
User：存放用户编写的驱动文件和测试文件。C 语言代码。

2.1.2 新建 Keil 工程

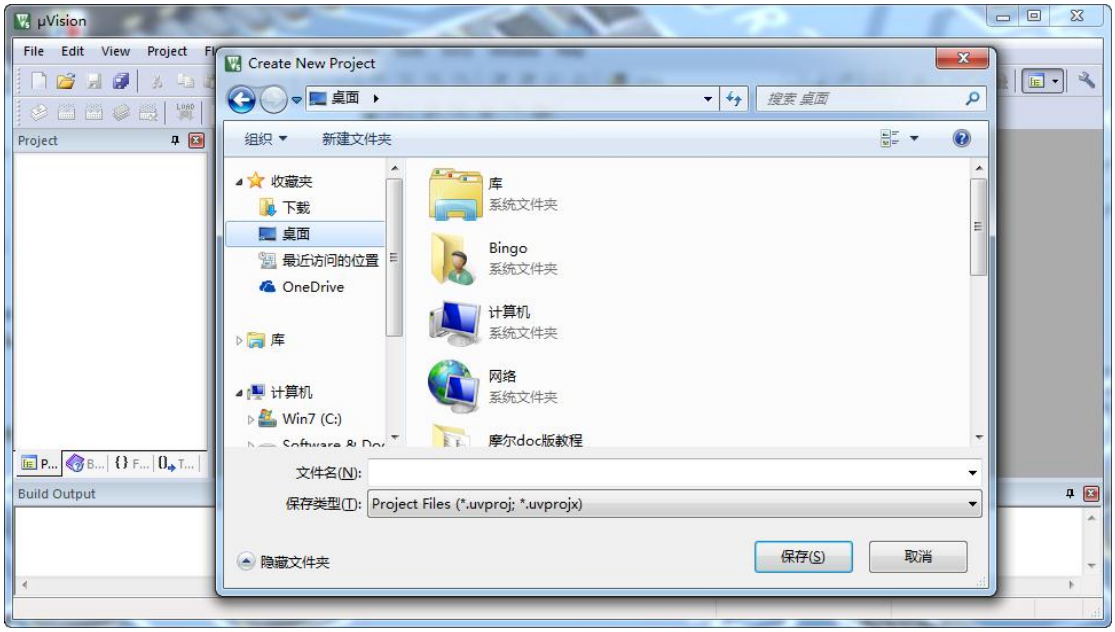
开始创建 Keil 工程，双击 Keil，进入 Keil 主界面。



点击工具栏 Project 按钮。

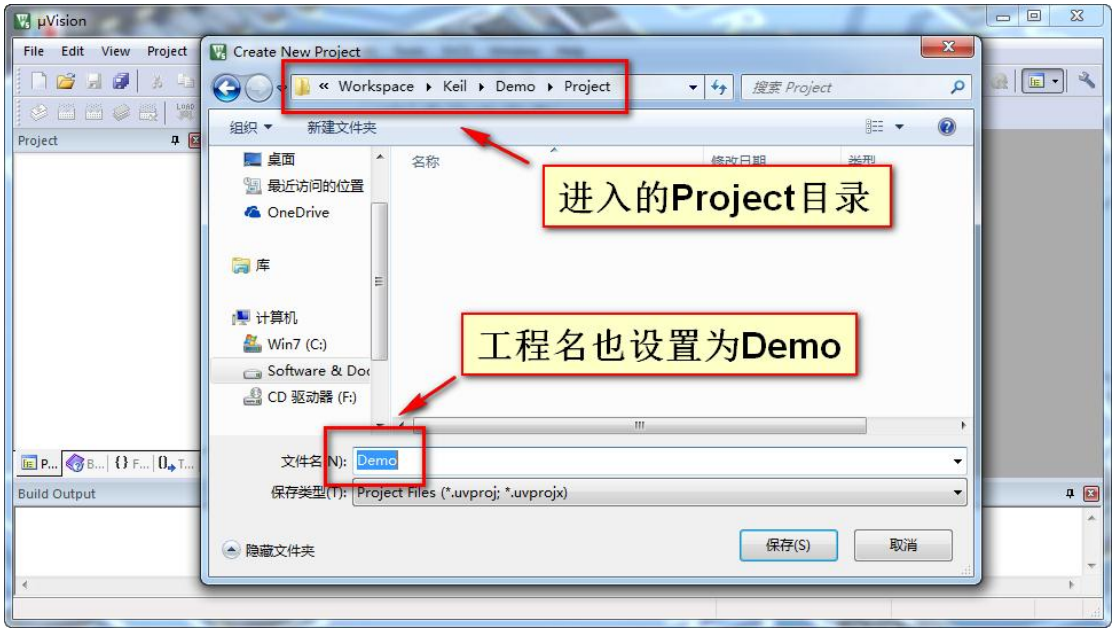


点击 New uVision Project，会弹出一个文件浏览框。

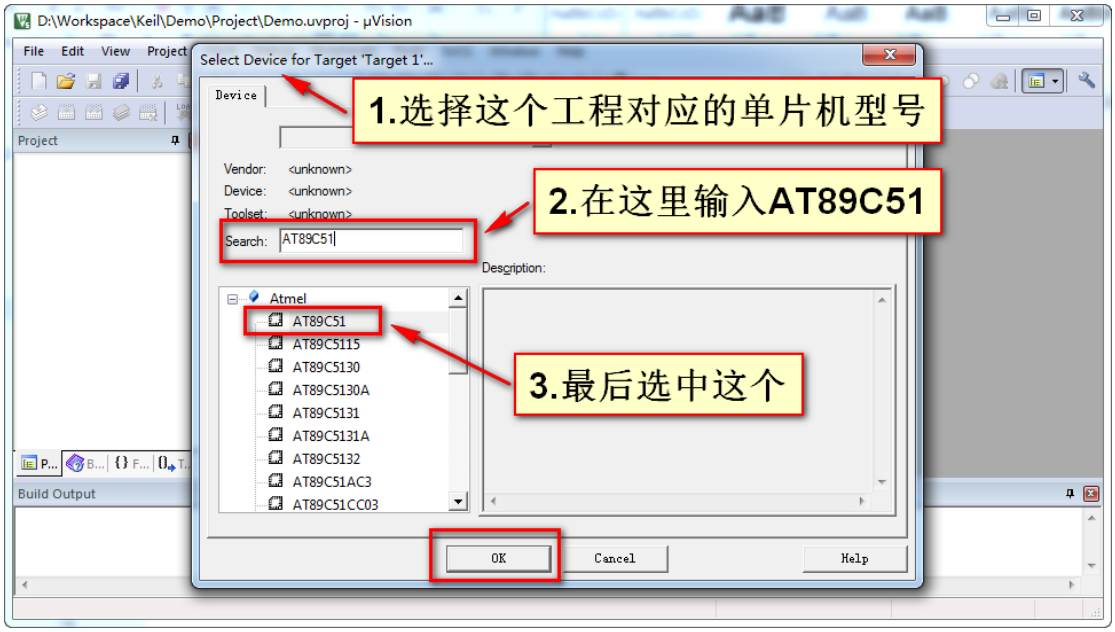


这个文件浏览框是选择工程存放的路径。

我们找到刚刚创建的 D:\Workspace\Keil\Demo\Project 目录，设置工程名也为 Demo。

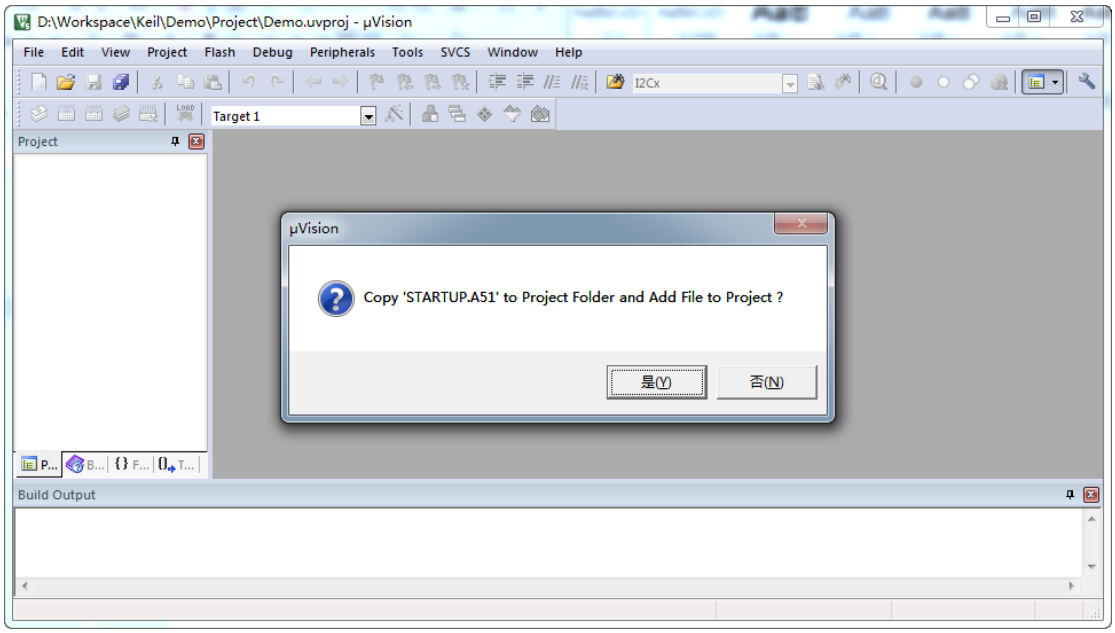


点击保存，之后出来一个器件选择界面。

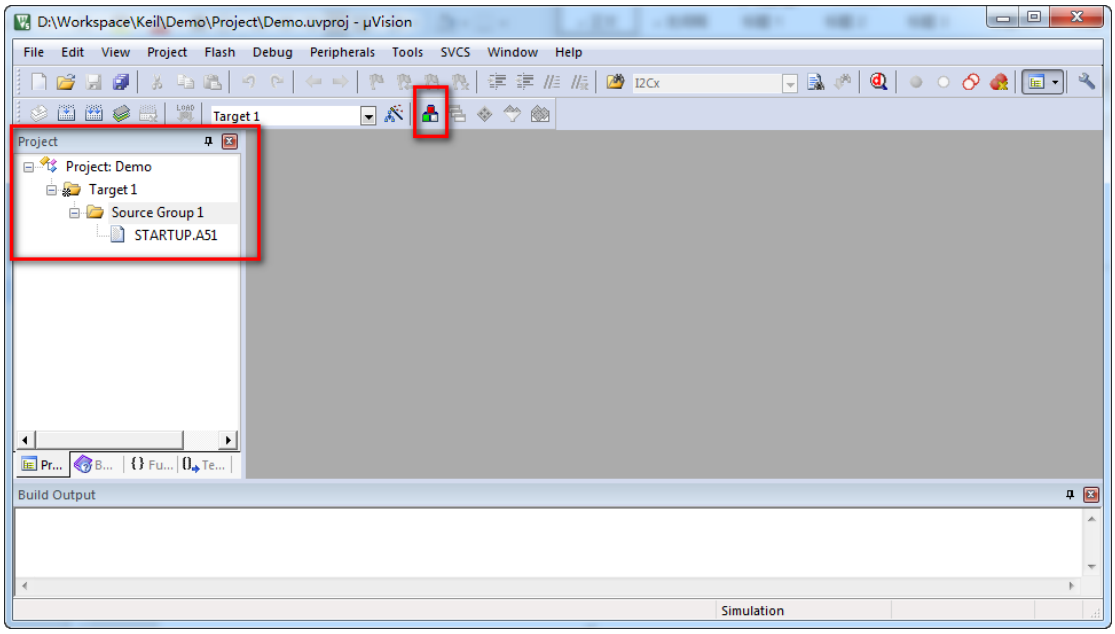


注意这里一定要先选中 AT89C51 才可以点击 OK。点击 OK 之后又会弹出一个提示的对话框。对话框是问要不要把 STARTUP.A51 文件添加进工程，这里一定要点是。

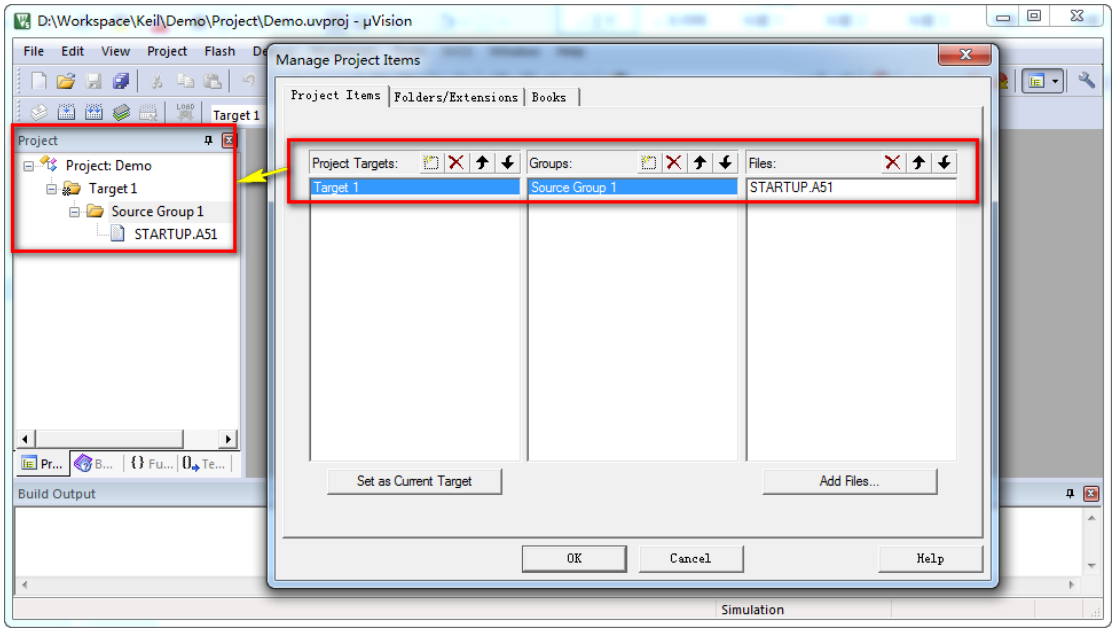
STARTUP.A51 文件是什么东西？是一段汇编编写的单片机启动代码，有了它我们的写的 C 程序的 main()函数才会被调用。



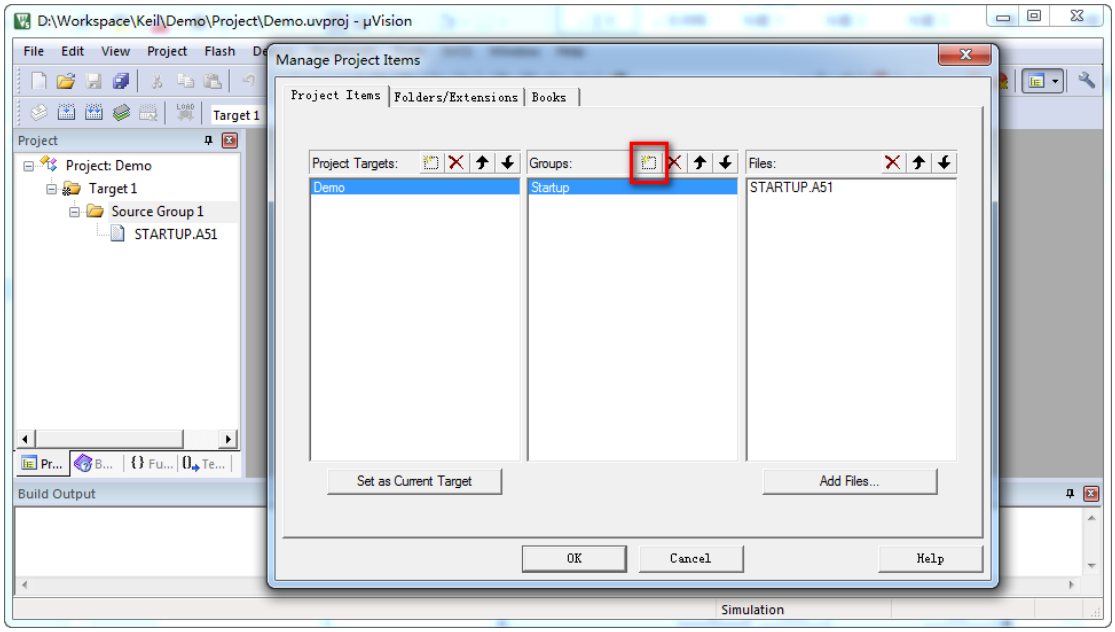
添加了启动代码之后，左边的 Project 工程文件管理面板会显示下图所示的工程文件。



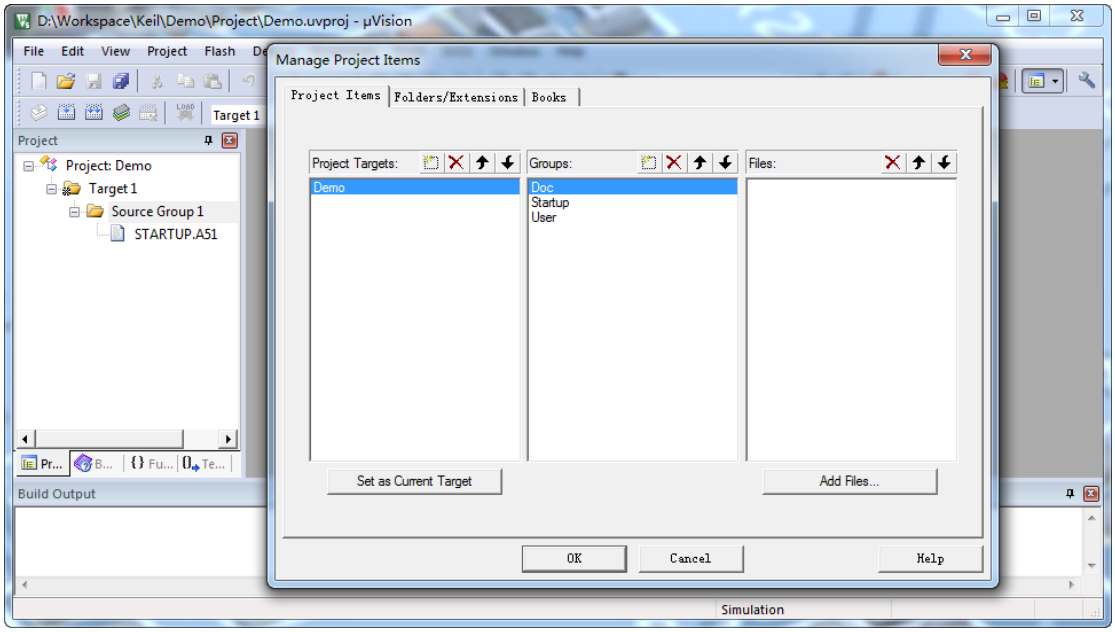
接下来点击图中圈出来的 **品** 字形的图标。



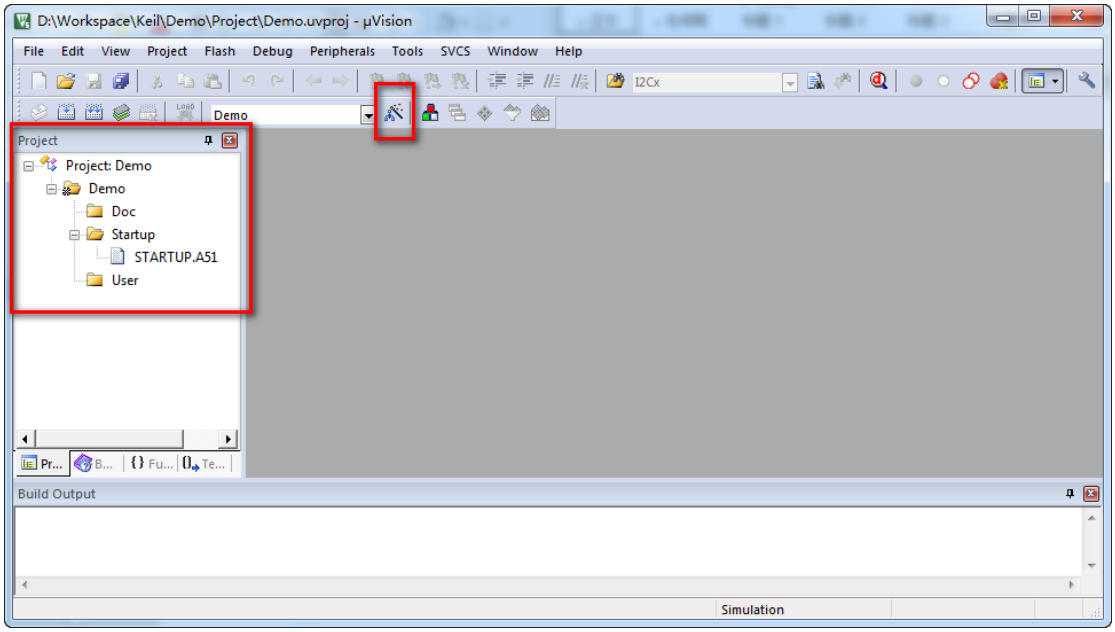
将其中的 Target 1 改成 Demo，Source Group 1 改成 Startup 。



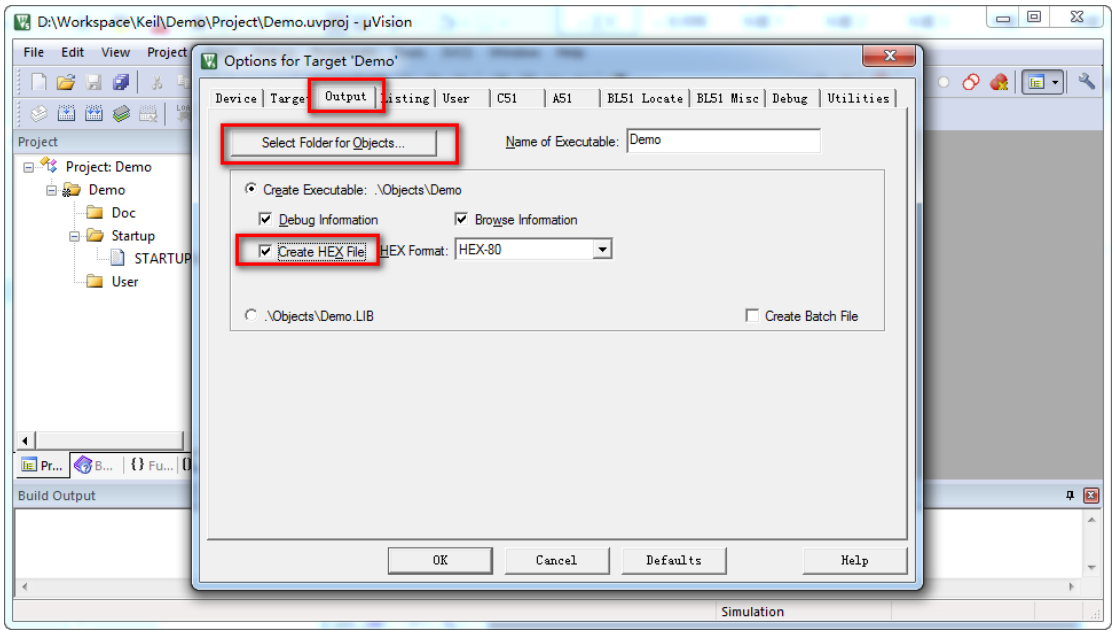
然后点击图中圈出来的图标，可以创建新的分组。创建下图的分组。上下图标可以移动分组的位置。



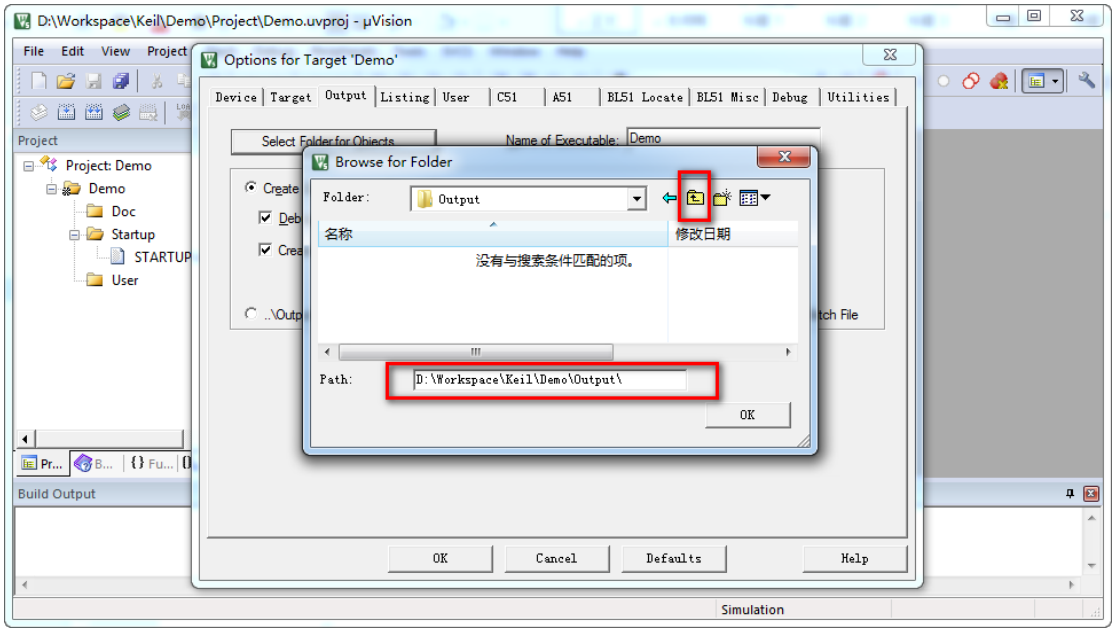
分组创建成功后 Project 面板是这样的。



接下来点击 魔术棒 按钮。进入 Output 选项卡，这里需要设置两个地方，先勾选 Create HEX File。这里的是 HEX File 文件是工程编译后生成的目标 16 进制文件。我们最终就是要把这个文件下载到单片机中，单片机就是按照这个文件的逻辑来运行的。

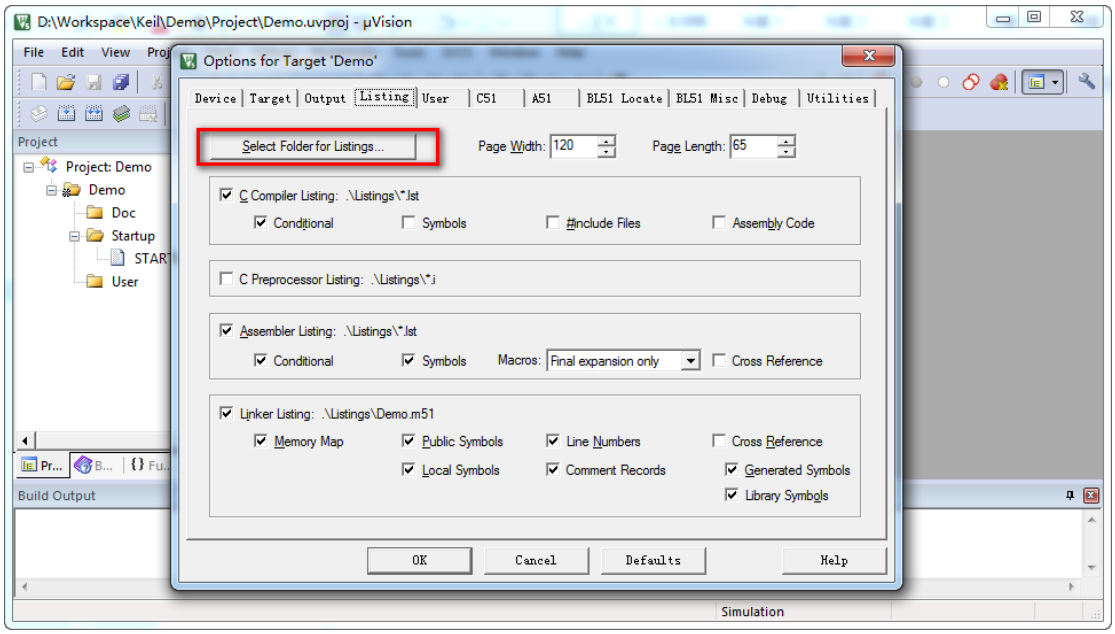


然后点击 Select Folder For Objects,这里是指定目标文件的输出路径。



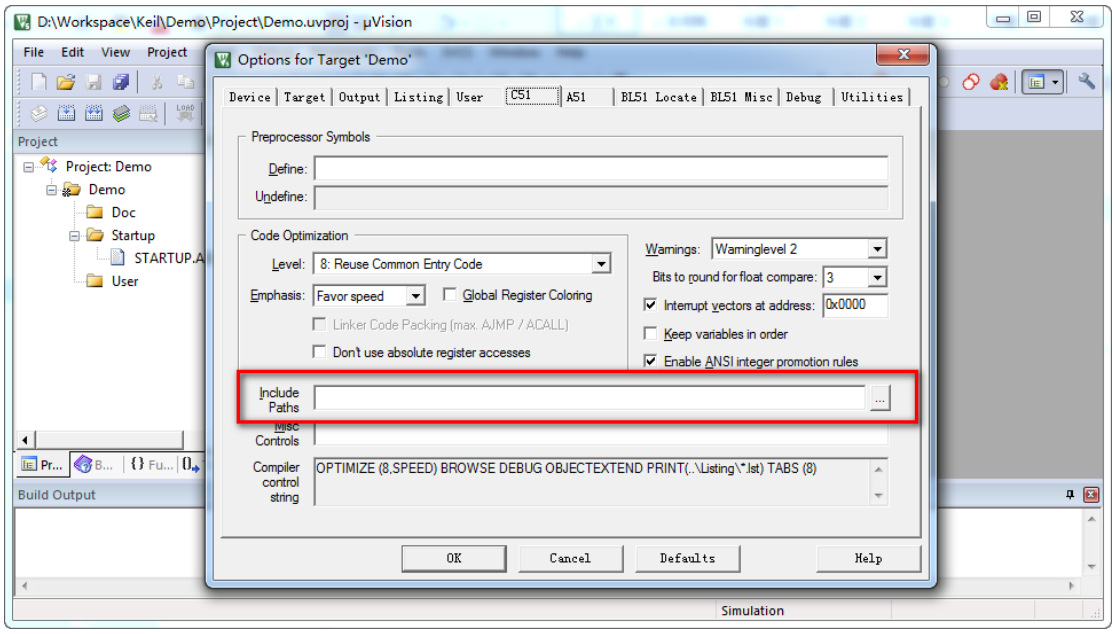
使用右上角的向上的箭头可以退出上一级。最后进入的是我们 Workspace 目录下的 Output 目录。最后看到 Path 为 D:\Workspace\Keil\Demo\Output\就对了。然后点击 OK。

然后进入 Listing 选显卡中。

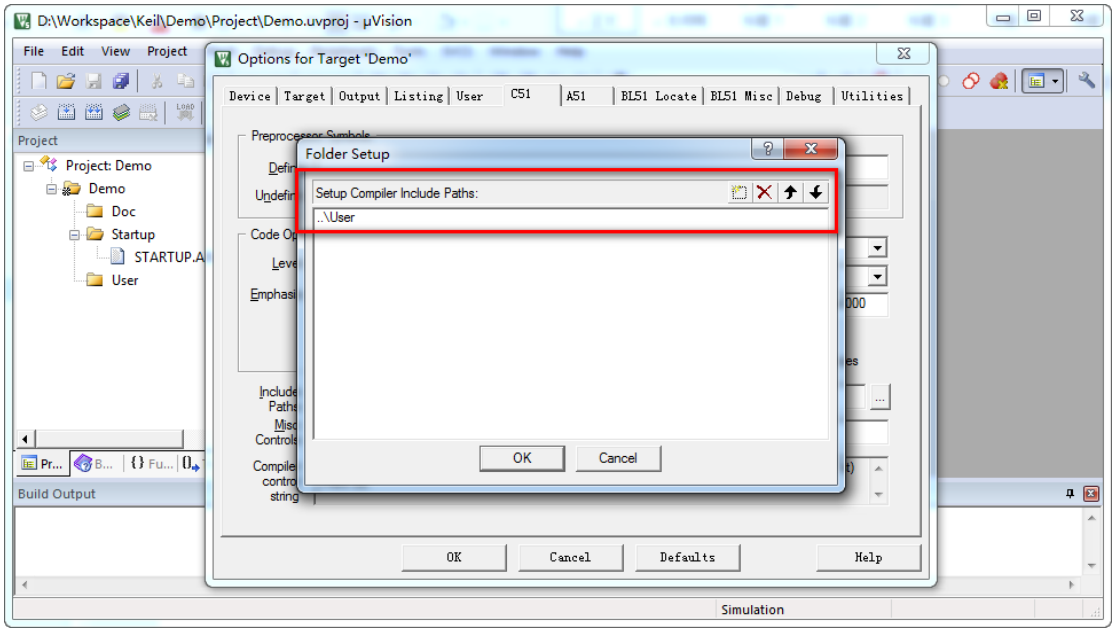


同样设置 Listing 文件的保存路径为 D:\Workspace\Keil\Demo\Listing\。这里就不贴图了，记得不要搞错了。不是 Project 目录下的 Listing。

然后进入 C51 选项卡。设置一下工程文件的头文件搜索路径，设置 Include Paths。

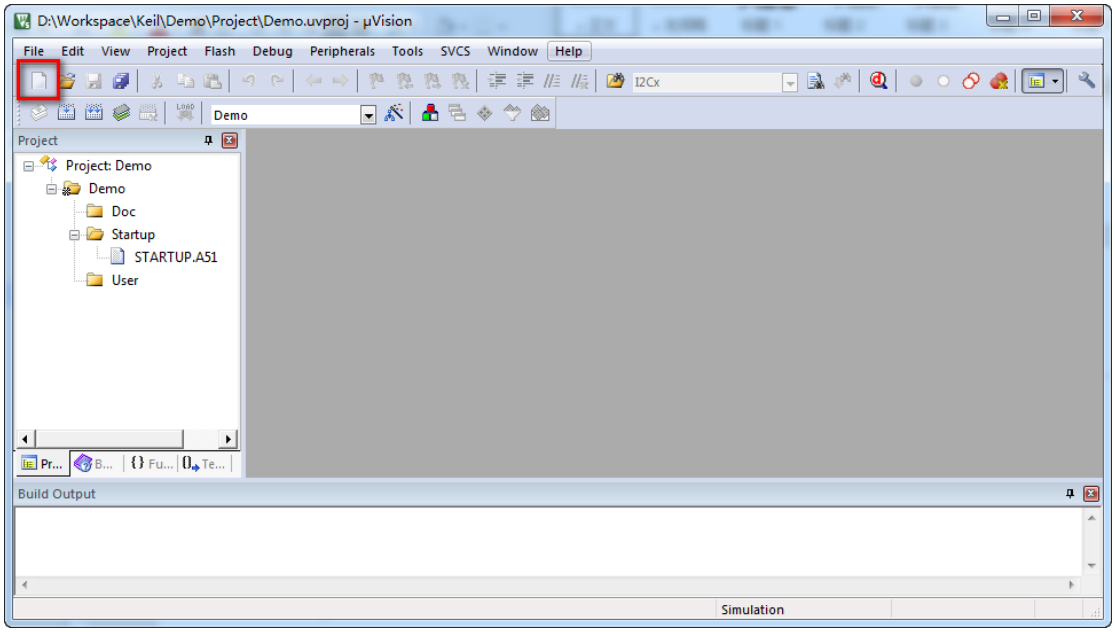


点击右边的点点点按钮，进入具体的设置界面。设置成下面的这个样子就可以了。

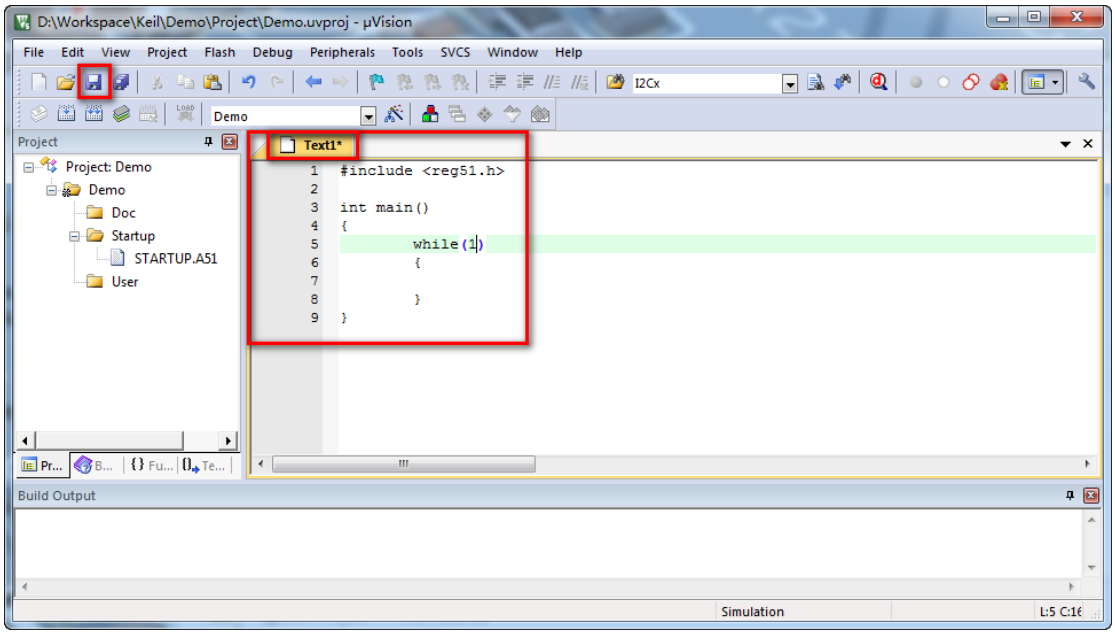


这里设置好了之后，以后头文件在 User 目录下，我们就可以直接#include 就可以了。具体写代码的时候你们就可以感受到了。

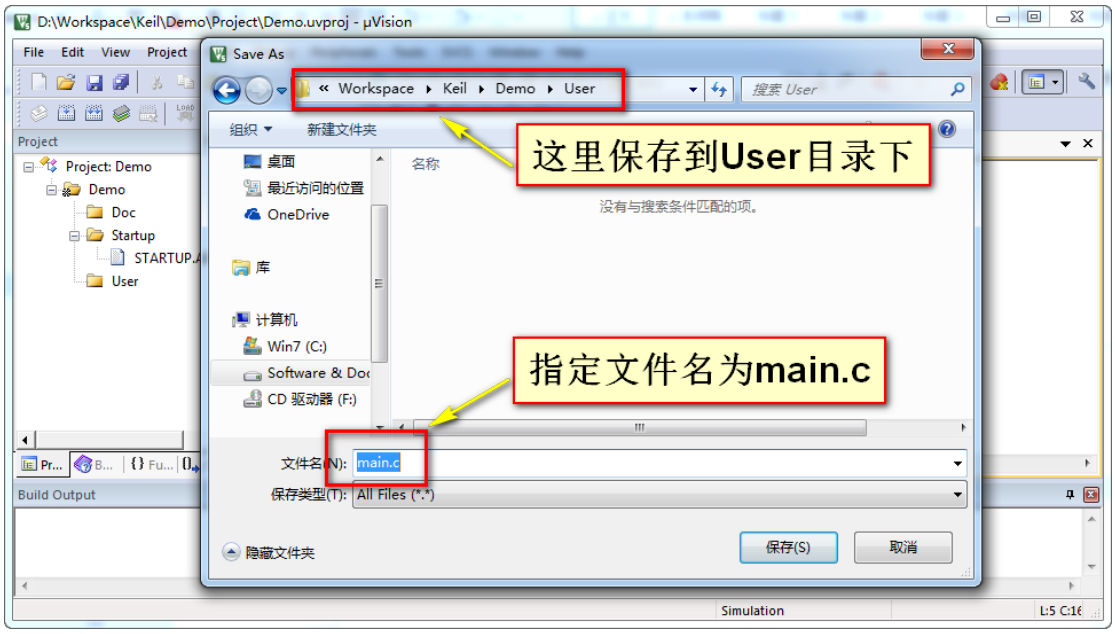
配置好了工程之后就要开始写代码了。



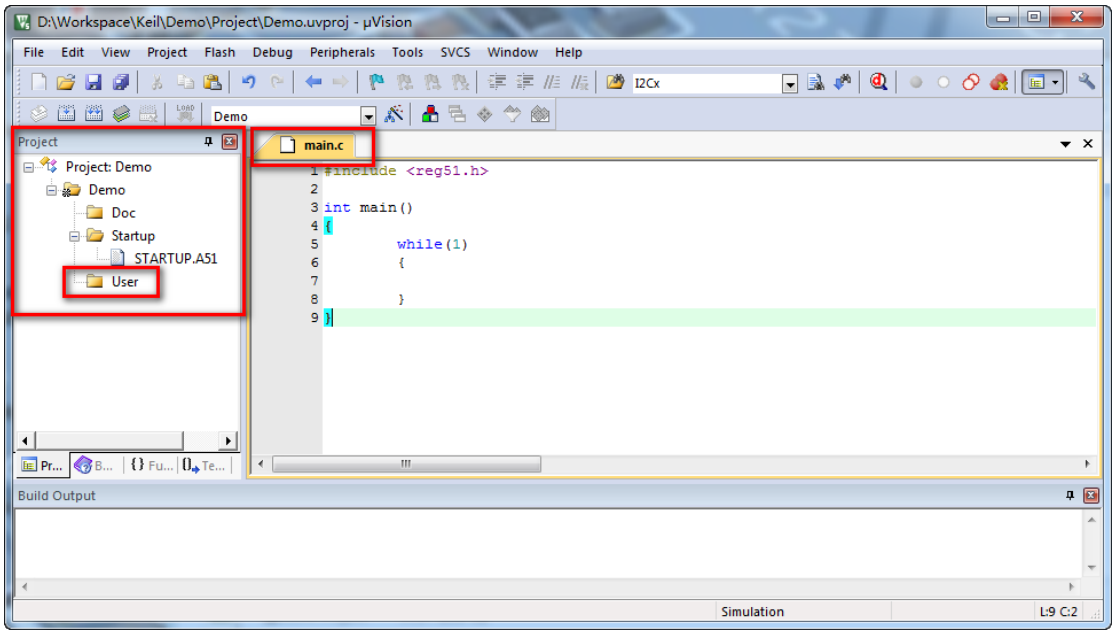
点击图中新建空白文件的按钮。填写内容如下图，具体代码的含义，后面会详细介绍，这里先把工程建立好。则灰色区域会出现一个 Text1，一旦你编辑了这个 Text1 并且尚未保存的话，就会有一个星号在 Text1 旁边，如图，表示编辑未保存。



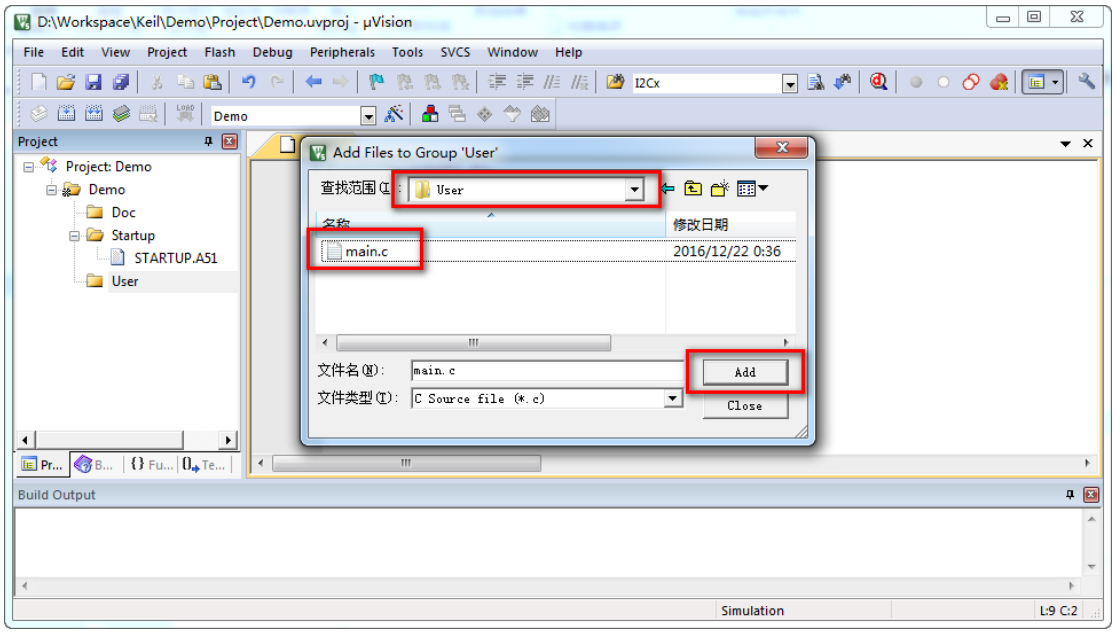
点击保存或者 Ctrl+S。之后又会弹出一个文件浏览框，这里是要指定当前代码文件存放在哪里。这里我保存到 User 目录下，文件名指定为 main.c，当然这个文件名是随便取得，但一定要是.c 格式。



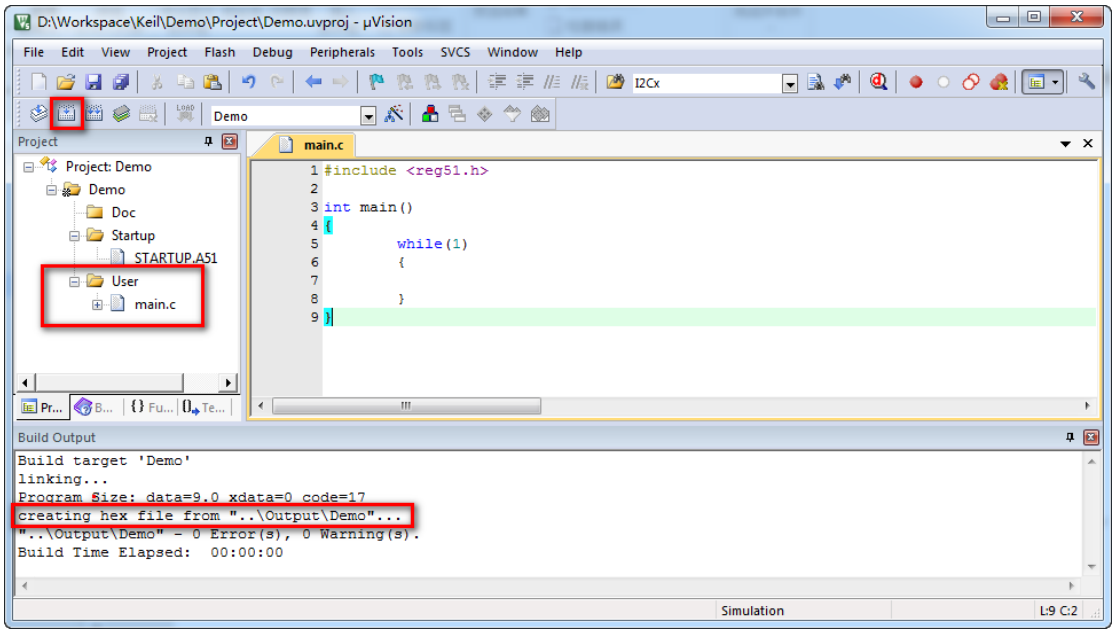
保存之后，Text1 就变成了 main.c 了，我们虽然创建了 main.c 文件，但是并没有将它添加到工程中。



接下来双击左边 Project 面板中的 User，会弹出一个添加文件到工程的对话框。切换到 User 目录，选中 main.c 点击 Add 按钮。或者直接双击 main.c。

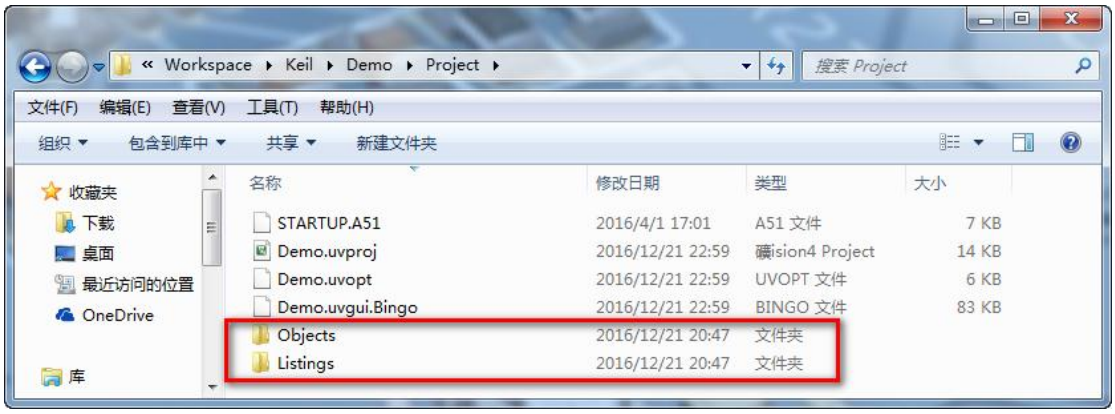


然后 main.c 就添加进了工程。如下图。以后写的其他 c 文件都要这样添加到工程中，h 文件后要在 Include Paths 中指定搜索路径。



最后就可以编译一下代码了。一定要看到下面有：creating hex file from "..\Output\Demo"...这句，说明了生成了单片机能加载的 16 进制文件。

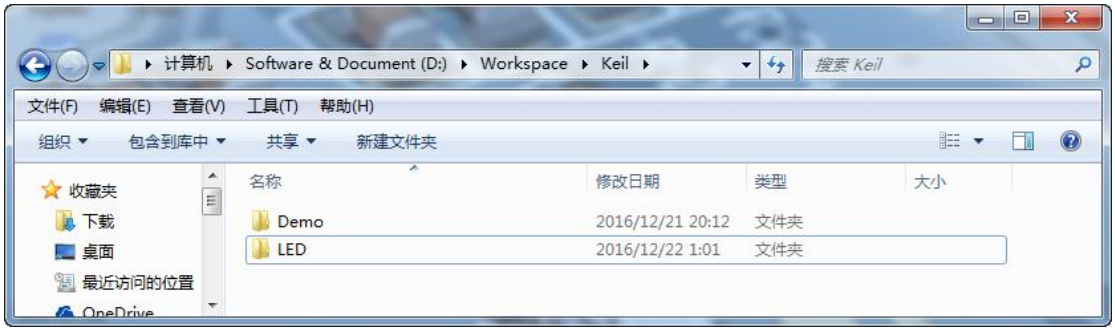
至此，工程就建立好了，其中在 D:\Workspace\Keil\Demo\Project 目录下，有两个没有用的文件夹，我们把它们删除就好了。



其实，这里的 STARTUP.A51 文件，一般我是会单独在 Demo 目录下创建一个 Startup 目录的，然后将它剪切到 Startup 目录下的。因为路径变了，所以需要重新添加进工程。这里你们愿意做就自己操作吧。

部分已经接触过单片机的小伙伴可能会很纳闷，为什么新建工程这么复杂，根本不必要。其实不是这样的，现在只是工程小，文件少，所以不必要说分好类，配置这么多，但是以后代码会越来越多，工程可能有成百上千个文件,哪个时候你们就知道代码分类存放的重要性了。

可能还有其他的小伙伴有疑问，建一个工程这么麻烦，那岂不是每次都要花好久在建工程上。其实工程模板一次建好了以后就不用再建了，每次新建一个工程的时候，只要在 Workspace\Keil\目录下，将 Demo 拷贝一份，并且重命名为工程名就可以了。比如说，我们要写一个点灯的程序，复制一份，将 Demo 目录名改为 Led 就可以了。

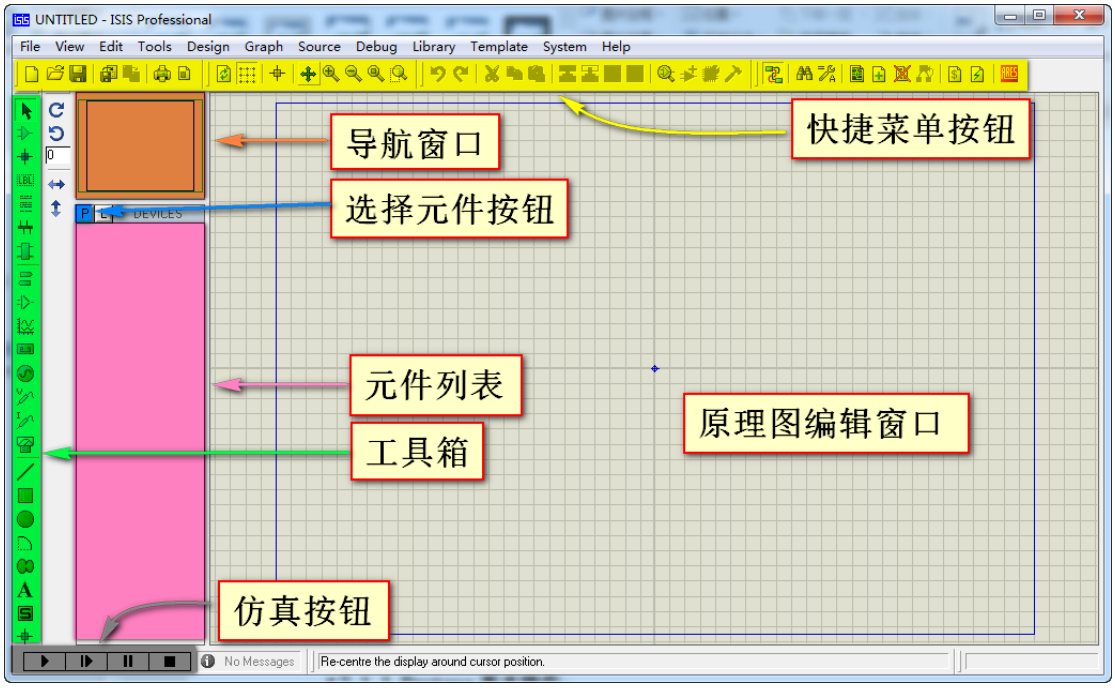


然后要打开 Led 这个工程也很简单，直接运行 Lcd 工程中 Project 目录下的 D:\Workspace\Keil\LED\Project\Demo.uvproj 文件就可以打开工程了，我们只需要将我们要编写的代码添加到主函数就可以了。

2.2 新建 Proteus 工程

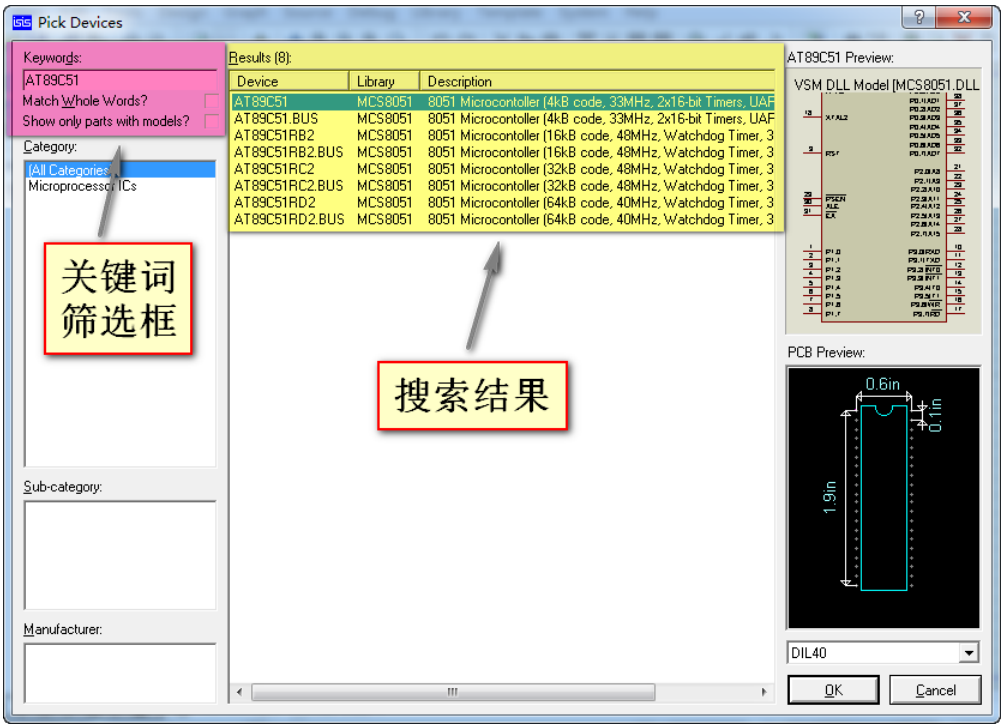
2.2.1 Proteus 界面介绍

双击运行 Proteus，进入主界面。这里简单介绍一下 Proteus 的界面，具体用法可以自行百度学习。



2.2.2 绘制一个 51 的最小系统电路原理图

运行 Proteus 程序，点击主界面的选择元件按钮 P。



在输入框中输入 AT89C51，双击右边的 AT89C51 元件，就可以将 AT89C51 这个元件添加到工程。

同理，分别添加下表元件到工程。

元器件名称	说明
AT89C51	AT89C51 单片机
RES	电阻
LED - RED	红色发光二极管 LED
BUTTON	按键
CRYSTAL	晶振(晶体振荡器)
CAP	无极性电容
CAP-POL	有极性电容

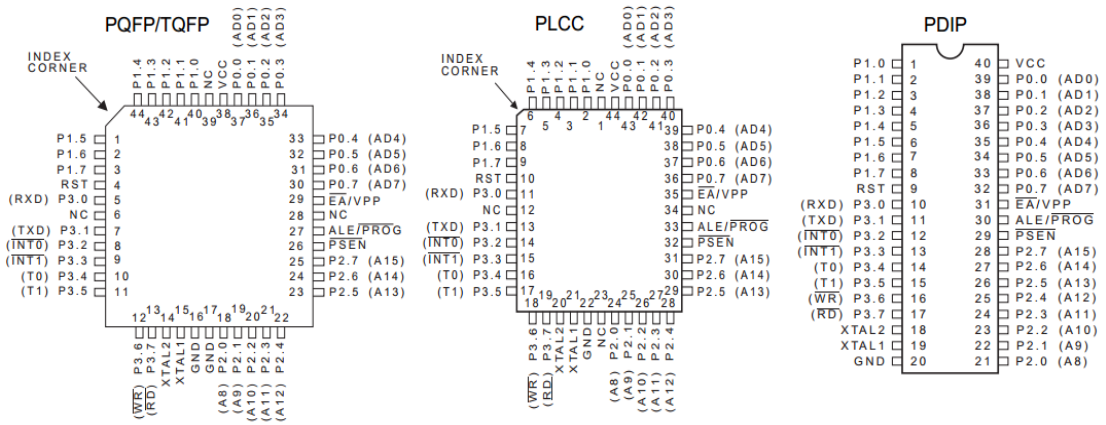
2.2.3 准备好工作空间

一进来默认是新建了一个空白的图纸，Proteus 建立一个工程并不复杂。不过我们还是准备一个目录存放 Proteus 的目录，在 Workspace\Proteus\目录下，创建一个 Demo 文件夹。到时候我们的工程文件就是保存在我们的 Demo 目录中。需要在保存的时候指定保存到这里文件夹。

2.2.4 51 单片机引脚介绍

一、AT89C51 单片机引脚介绍

AT89C51 有 PDIP、PLCC、TQFP 三种封装方式，其中最常见的就是采用 40Pin 封装的双列直接 PDIP 封装，外形结构下图。



芯片共有 40 个引脚，引脚的排列顺序为从靠芯片的缺口左边那列引脚逆时针数起，依次为 1、2、3、4...40，其中芯片的 1 脚顶上有个凹点。在单片机的 40 个引脚中，电源引脚 2 根，外接晶体振荡器引脚 2 根，控制引脚 4 根以及 4 组 8 位可编程 I/O 引脚 32 根。

1、主电源引脚（2 根）

VCC(Pin40)：电源输入，接＋5V 电源

GND(Pin20)：接地线

2、外接晶振引脚（2 根）

XTAL1(Pin19)：片内振荡电路的输入端

XTAL2(Pin20)：片内振荡电路的输出端

3、控制引脚（4 根）

RST/VPP(Pin9)：复位引脚，引脚上出现 2 个机器周期的高电平将使单片机复位。

ALE/PROG(Pin30)：地址锁存允许信号

PSEN(Pin29)：外部存储器读选通信号

EA/VPP(Pin31)：程序存储器的内外部选通，接低电平从外部程序存储器读指令，如果接高电平则从内部程序存储器读指令。

4、可编程输入/输出引脚（32 根）

AT89C51 单片机有 4 组 8 位的可编程 I/O 口，分别位 P0、P1、P2、P3 口，每个口有 8 位（8 根引脚），共 32 根。每一根引脚都可以编程，比如用来控制电机、交通灯、霓虹灯等，开发产品时就是利用这些可编程引脚来实现我们想要的功能，尽情发挥你的想象力吧，实现你想要的各种功能。

P0 口（Pin39～Pin32）：8 位双向 I/0 口线，名称为 P0.0～P0.7

P1 口（Pin1～Pin8）：8 位准双向 I/0 口线，名称为 P1.0～P1.7

P2 口（Pin21～Pin28）：8 位准双向 I/0 口线，名称为 P2.0～P2.7

P3 口（Pin10～Pin17）：8 位准双向 I/0 口线，名称为 P3.0～P3.7

2.2.5 51 最小系统

单片机最小系统或称为最小应用系统,是指用最少的元件组成的单片机可以工作的系统。

对 51 系列单片机来说,最小系统一般应该包括：单片机、电源、晶振电路、复位电路。

1、单片机

89C51 单片机一片

2、电源

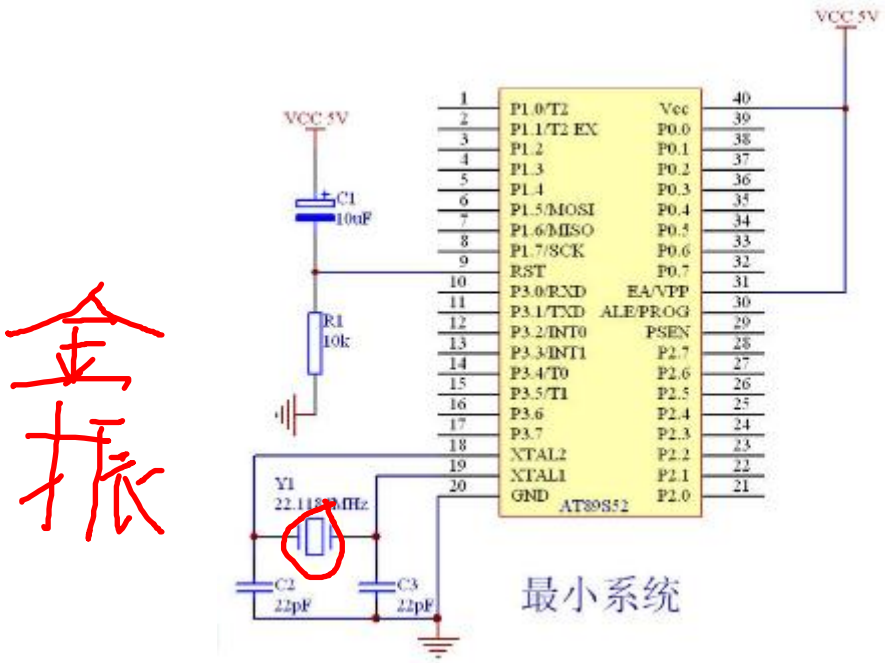
5V 直流电源 1 个

3、晶振电路

包括 12MHz 晶振 1 只、30pF 瓷片电容 2 只

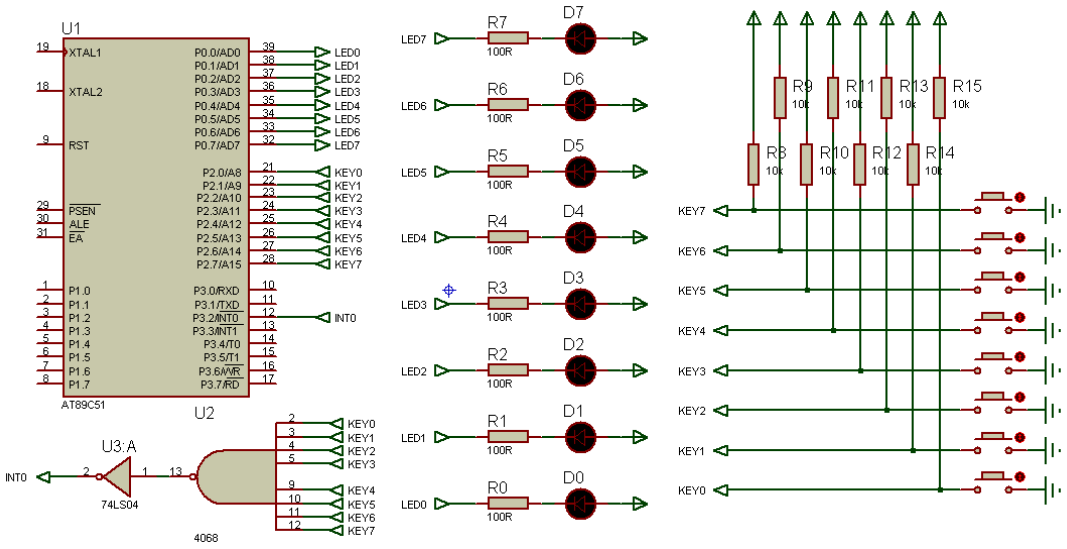
4、复位电路

10uF 电解电容 1 只，4k7 电阻 1 只。



2.2.6 网络标号

我们在看电路时候，经常发现很多没有连接在一起的电路模块。看起来好像没有连在一块，其实他们是通过网络标号连接在一起的。



相同的网络标号即表示相连。