

实验三 单片机系统设计和仿真

本实验的目的是在 Proteus 软件的设计环境中进行单片机系统的电路设计和仿真。实验中除了要使用 Proteus 软件，还要使用 Keil 软件进行单片机的应用程序开发。

图 1 是 8051 单片机最小系统，单片机采用的型号是 AT89C51。除了最小系统必要的电源、时钟、复位电路之外，在 P1 口外加一个数码管。P1 口是 8 位的并行口，此处用到了 P1.0~P1.6 七个引脚，它们在电路图中的标号为 P1_0~P1_6，分别连接的是数码管的段选信号 a、b、c、d、e、f、g。

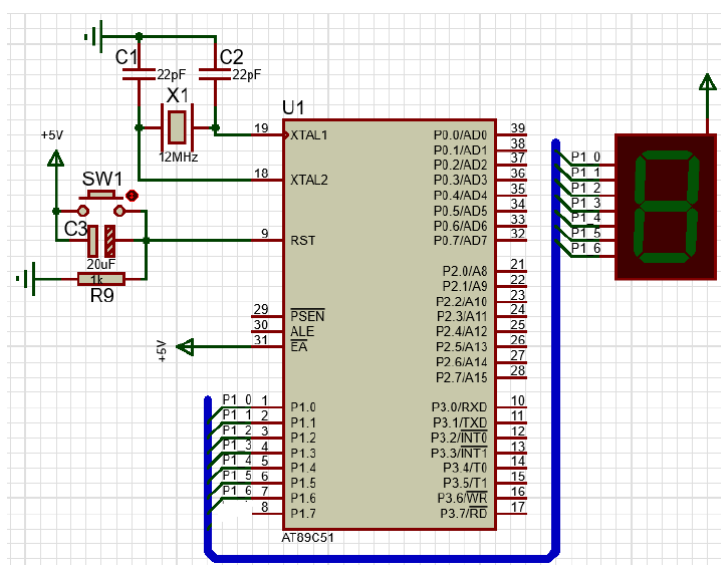


图 1

从图中可见，数码管是共阳极类型，共阳极端接电源，因此如果要使数码管某一段发光的话，就必须使相应的单片机引脚输出低电平。比如要在数码管上显示“0”，则 P1.6~P1.0 应输出“1000000”。

一、Keil 软件环境下的程序设计

(1) Keil 工程的建立

双击桌面上的 Keil μ Vision 集成开发环境图标 ，会出现以下运行界面：



图 2

随后，进入软件的开发界面如图 3 所示：

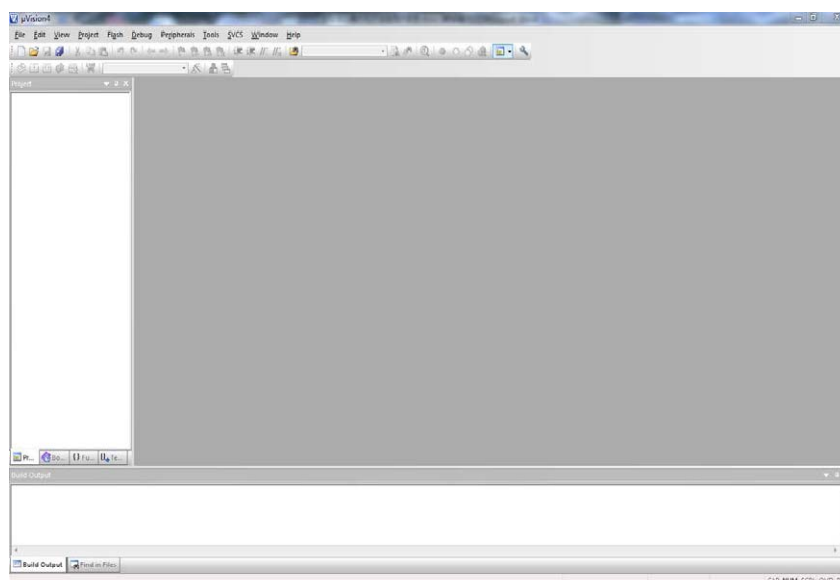


图 3

首先需要建立一个新的工程。点击菜单栏上的“Project→New uVision Project...”，如图 4 所示：

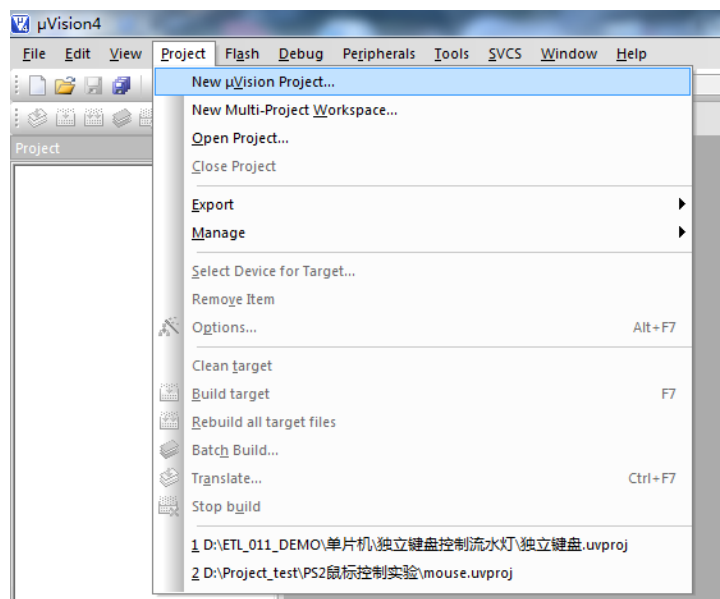


图 4

在弹出的对话框中选择新项目要保存的路径和文件名。Keil 的一个工程里通常含有很多小文件, 为了方便管理, 通常我们将一个工程放在一个独立文件夹下, 例如保存到 E:\mcu_project1 文件夹, 工程名为 mcu_project1, 如图 5 所示。Keil uVision4 的工程文件扩展名为.uvproj。

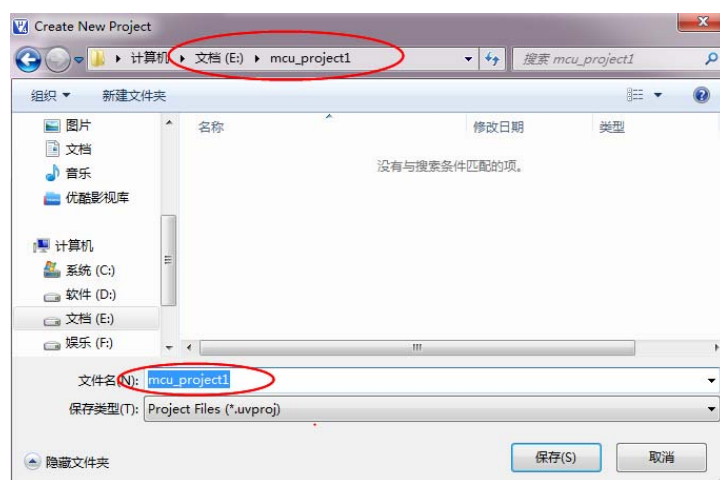


图 5

单击“保存”以后, 弹出如下界面:

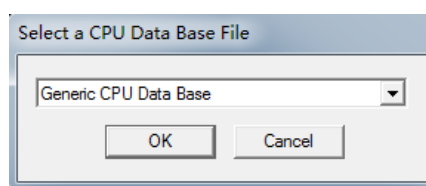


图 6

点击“OK”，会弹出选择 CPU 的界面，如图 7 所示。这里我们选择 Atmel 下的 AT89C51，如图 8 所示。

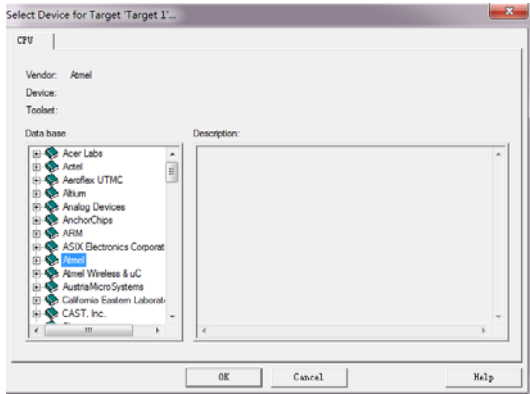


图 7

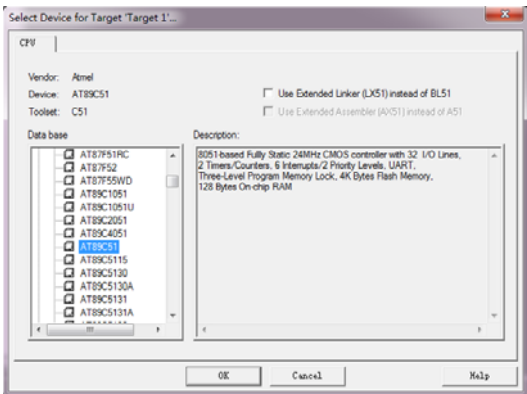


图 8

点击 OK 后会弹出提示框确认是否导入芯片启动代码到工程，在此我们选择 YES，这样一个新的工程就建立完成了。

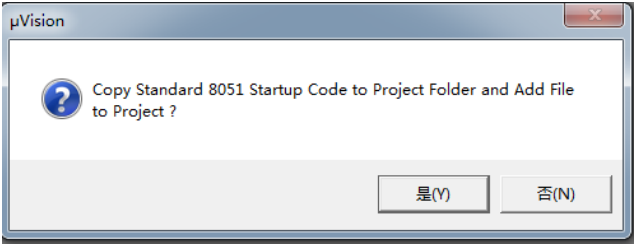


图 9

二、建立源文件

工程建立完成后，我们可以在左边的 Project 栏中看到如下图所示的信息。

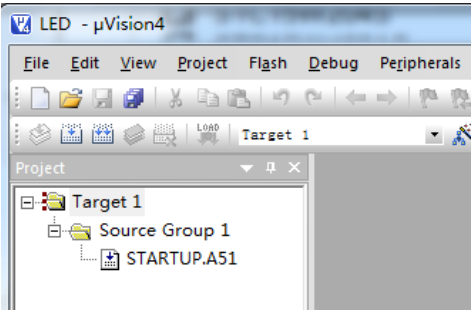


图 10

接下来单击“File”菜单中的“New”菜单项，如图 11 所示，进入编辑界面，如图 12 所示，此时光标会在编辑窗口中闪烁，可以输入用户的应用程序。

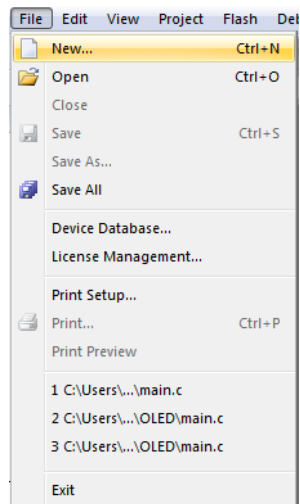


图 11

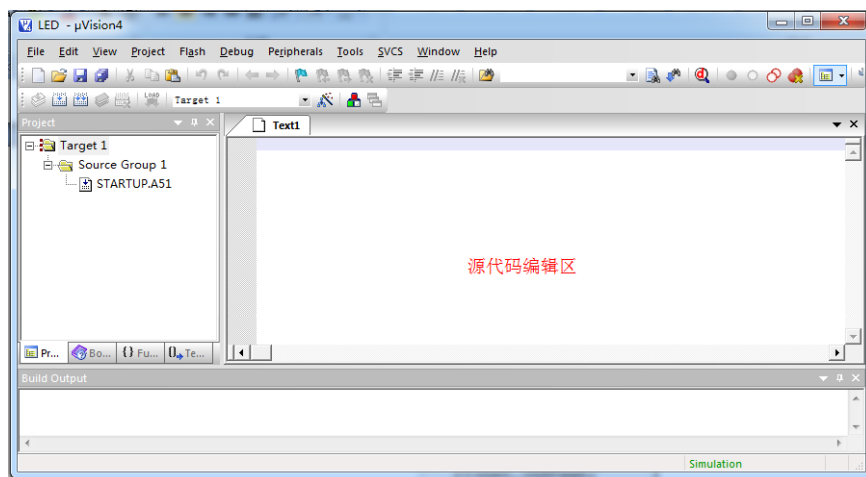


图 12

在源代码编辑区输入以下 C 语言源程序，注意：在输入程序时务必将输入法切换到英文半角状态。

```
#include <reg51.h>
#define uchar unsigned char
#define uint unsigned int


uchar code dis[] = {0xc0, 0xf9};

void delay(void)
{
    uint i, j;
    for (i=100; i>0; i--)    //注意没有分号
        for (j=500; j>0; j--);
}
```

```

void main()
{
    while(1)
    {
        P1 = dis[0];
        delay();
        P1 = dis[1];
        delay();
    }
}

```

代码输入完成以后，点击工具栏上的保存图标，弹出窗口界面如图 13 所示，在文件名编辑框中输入要保存的文件名，同时必须输入正确的扩展名（.c）。这里的文件名不一定要和工程名相同，用户可以随意填写文件名，然后单击“保存”即可。

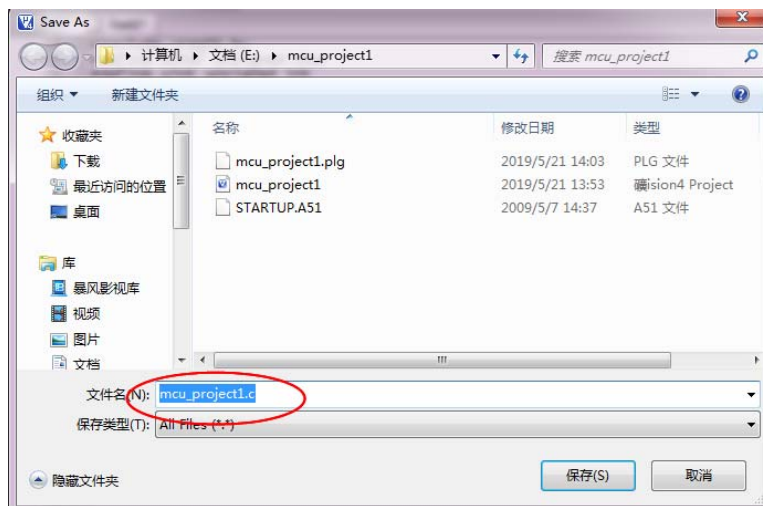


图 13

此时这个新建的源文件 mcu_project1.c 与我们刚才建立的工程还没有直接的联系，还需要我们把源文件 mcu_project1.c 添加到工程中去，以下为添加源文件的方法。

回到编辑界面，在“Source Group 1”选项上单击右键，弹出如图 14 所示的菜单，然后单击“Add Files to Group.....”菜单项，对话框如图 15 所示。

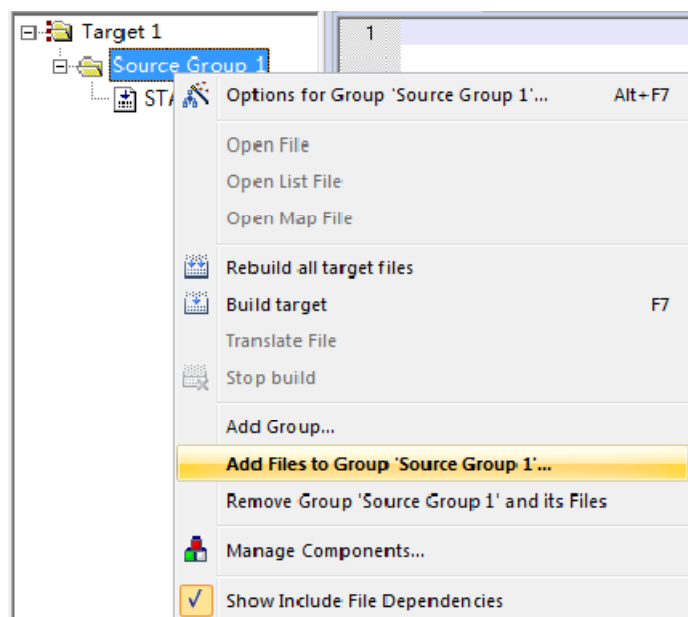


图 14

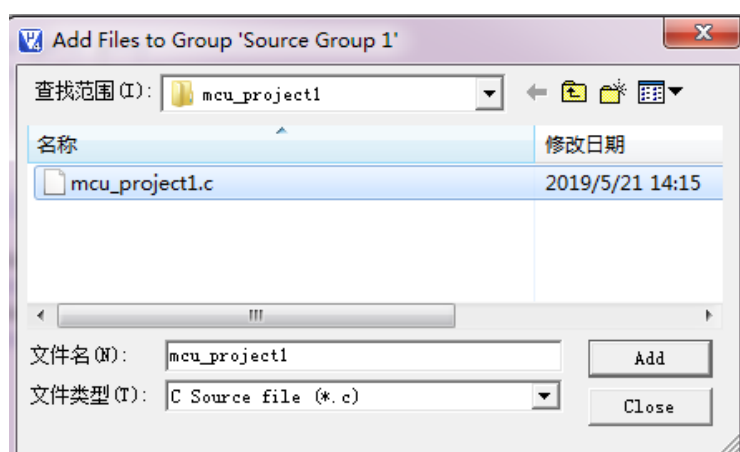



图 15

选中“mcu_project1.c”单击“Add”按钮，再单击“Close”按钮，然后我们就可以在左侧“Source Group 1”看到“mcu_project1.c”文件了。

源代码编写、添加完成之后，下面就要进行编译和下载了。点击图标，软件会自动编译当前文件。没有错误时，显示结果如下图所示。如果程序中有错误，可以按照提示进行修改。

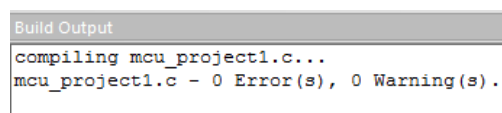



图 16

接下来点击图标，弹出如图 17 对话框，点击“Output”选项，勾选“Creat HEX File”选项，点击“OK”按钮。

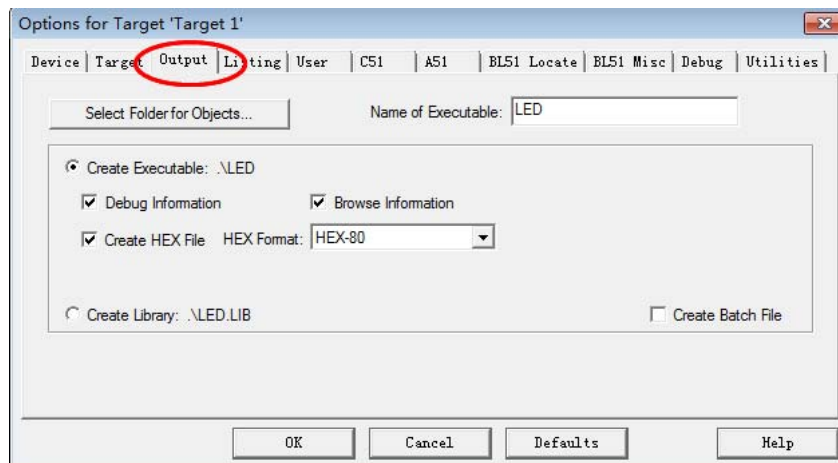



图 17

回到程序编辑窗口，点击图标 ，在下方的输出窗口显示如下信息，表示生成 HEX 文件成功。

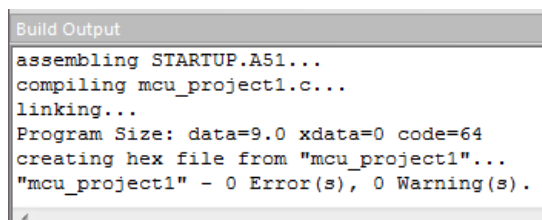


图 18


三、单片机系统的电路设计与仿真

按照实验一和实验二的方法，在 Proteus 软件中建立一个新的工程，将图 1 所示的单片机系统设计出来。图 1 中所用的元件可以采用直接查找法，元件的名称为：



- AT89C51
- 7SEG-COM-AN-GRN
- CAP
- CAP-ELEC
- CRYSTAL
- RES
- BUTTON


1. 放置总线

图 1 中单片机的 P1 端口的 7 根线采用的是总线画法。放置总线的步骤为：

- ① 从工具箱中选择总线“Bus”图标 ；

- ② 在总线起始端出现的位置单击鼠标左键；
- ③ 在总线路径的拐点处单击鼠标左键；
- ④ 在总线的终点双击鼠标左键，可结束总线放置。

单片机 P1 端口的每一个引脚连接到总线的连接线叫做总线分支。在 Proteus 中，总线分支既可以用总线命令 ，也可以用一般连线命令 。在使用总线命令画总线分支时，粗线自动变成细线。为了使电路图显得专业而美观，通常把总线分支画成与总线呈 45° 角的相互平行的斜线。

如图 1 所示，在 AT89C51 的 P1 口左侧先画一条自上而下的总线。确认主工具栏中的自动布线器  为选中状态(为画斜线准备)。在 P1.0 引脚单击鼠标左键后松开，拖动鼠标指针画线，距总线一个背景栅格时，单击鼠标左键确认，然后按住“Ctrl”键，并向左下移动鼠标，在与总线呈 45° 角相交时，单击鼠标左键确认，即完成一条总线分支的绘制。

其他总线分支的绘制不必这样复杂，只需在分支的起始点双击鼠标左键即可完成。比如，画下一条 P1.1 引脚至总线的分支，把鼠标指针放置在 P1.1 引脚口位置，出现一个红色小方框，开始双击鼠标左键，自动完成像 P1.0 引脚到总线的一系列动作。

2. 添加连线标签

- ① 从工具箱中选择“Wire Label”图标 。
- ② 把鼠标指针指向期望放置标签的总线分支位置，鼠标指针处出现一个“×”号，此时单击鼠标左键，出现“Edit Wire Label”对话框，如图 19 所示。
- ③ 在该对话框的“Label”选项卡中键入相应的文本，如“P1_0”，或者点击下拉选项选择一个标签。
- ④ 单击“OK”按钮，结束文本的输入。

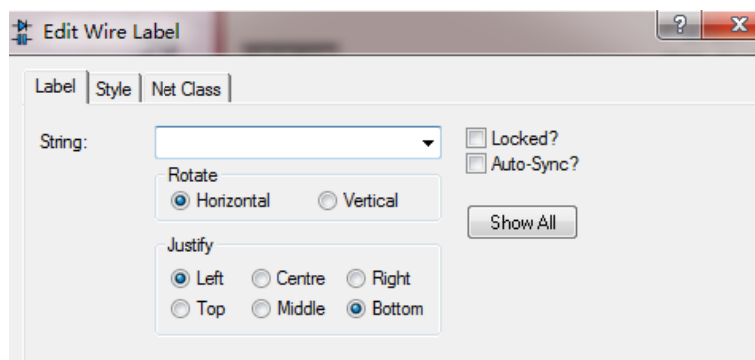


图 19

3. 电气检测

电路设计完成后，选择菜单栏中的 Tools-->Electrical Rules Check 进行电器检

测，则会弹出电气检测结果对话框，如图 20 所示，表示本例没有电气错误。

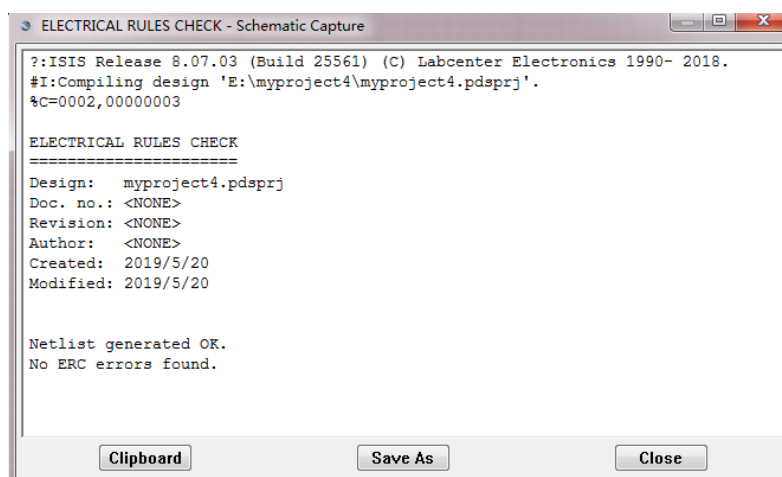


图 20

4. 单片机仿真

将鼠标放置在 AT89C51 上，双击左键，弹出编辑元器件的对话框，如图 21 所示。在“Program File”栏中选择前面用 Keil 软件设计生成的 mcu_project1.hex 文件。再点击 OK。

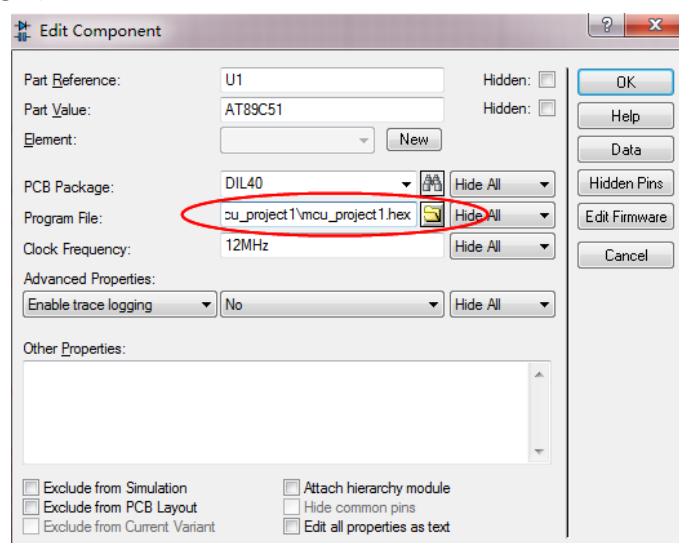


图 21

按下仿真运行按钮，可以看到数码管循环显示 0 和 1。

5. Proteus 和 Keil 的联调

以上第 4 步的单片机仿真方法中，用 Keil 设计单片机里的程序，生成.hex 文件，然后在 Proteus 中调用，Keil 中的程序不能进行调试。下面说明 Proteus 和 Keil 的联调方法。

在 Proteus 软件中打开图 1 电路的工程，在菜单栏选择 Debug-->Enable Remote Debug Monitor，如图 22 所示。

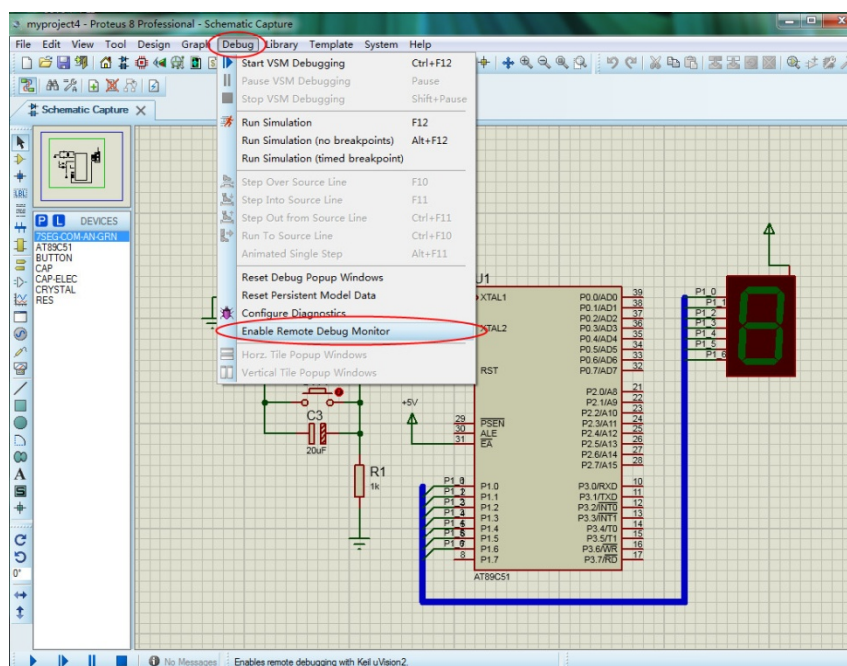



图 22

再打开 Keil 中的单片机程序，并在工具栏中选择 ，弹出如图 23 的窗口，选择 Debug 项，在仿真器的选择中选择“Proteus VSM Simulator”，然后点击 OK。

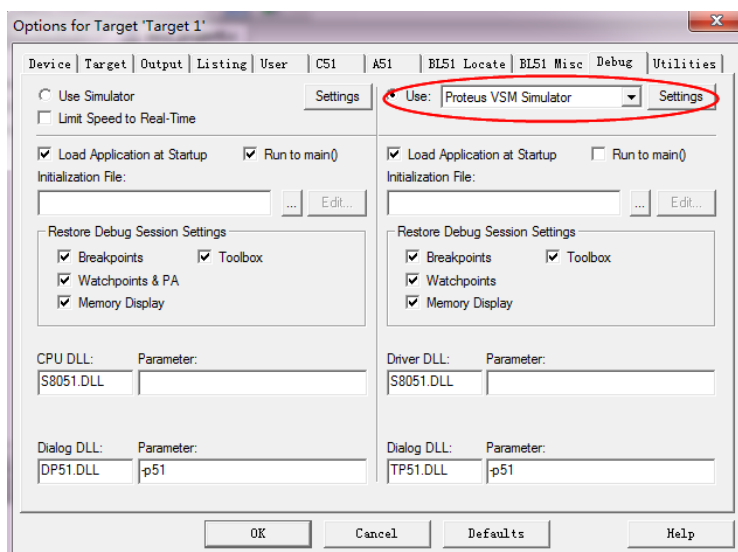



图 23

将 Proteus 和 Keil 都打开放于屏幕上。在 Keil 中点击 ，进入调试模式，可以看到 C 程序窗口。在两条赋值语句之前的灰色边带双击，可以放置断点，如图 24 所示。

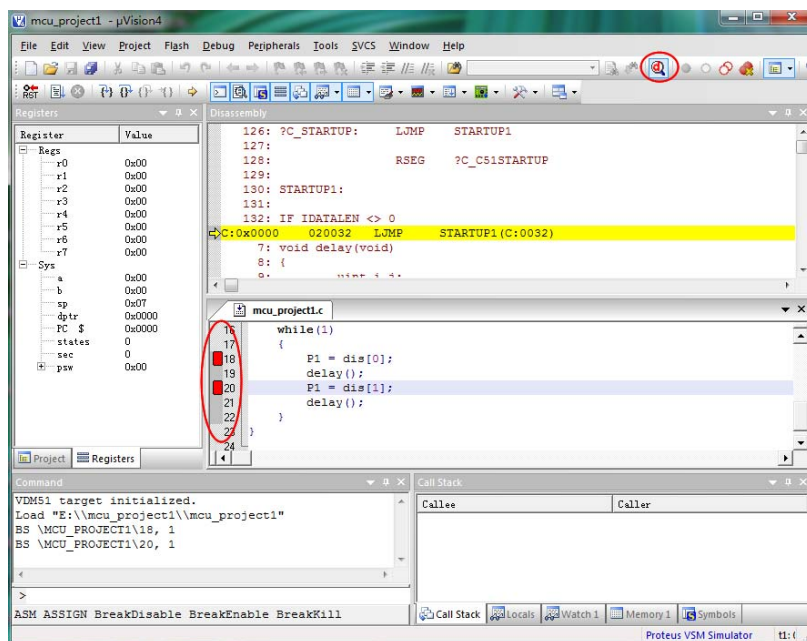



图 24

反复点击工具栏上的  或按 F10 执行程序，程序会在断点处暂停，此时可以看到 Proteus 中的数码管在数字 0 和 1 之间跳变。如图 25 所示。

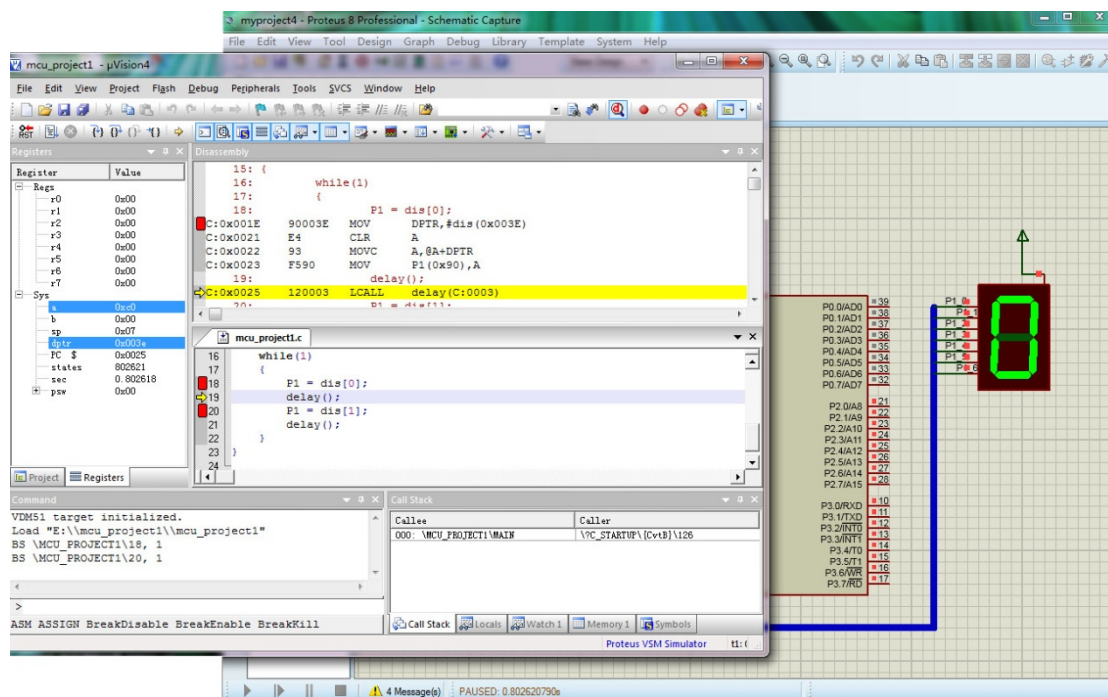
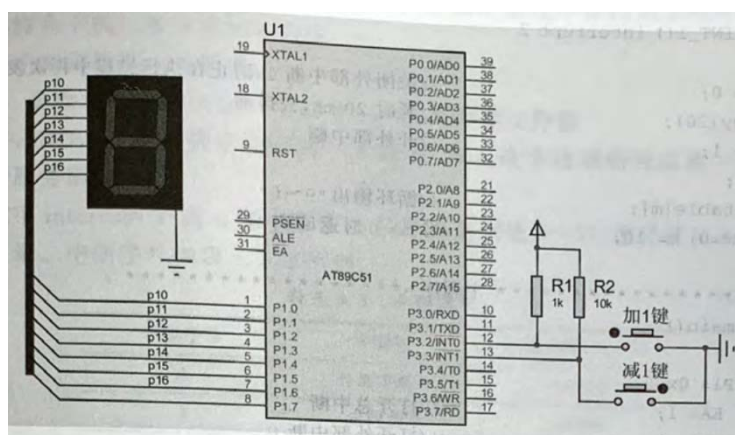


图 25

四、拓展练习

用 Proteus 软件设计图 22 所示的单片机系统，用 Keil 软件设计应用程序，并进行仿真。该系统实现的功能是用外部按键控制数码管显示的数字加 1 或减 1。



程序代码如下:

```

/*****必要的变量定义*****/
#include <reg51.h>
#define uint unsigned int
#define uchar unsigned char    //宏定义
uchar code table[] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F};
//共阴极数码管 0~9 编码表
uchar m = 0;
/*****延时子程序*****/
void delay(uchar c)
{
    unsigned char a, b;
    for (; c > 0; c--)    //注意没有分号
        for (b=142; b>0; b--)
            for (a=a; a>0; a--);
}
/*****外部中断 0 子程序*****/
void INT_0() interrupt 0
{
    EX0 = 0;    //关闭外部中断 0，防止在执行过程中再次发生中断
    delay(20); //延时 20ms，去抖动
    EX0 = 1;    //开外部中断 0
    m++;
    if(m==10) m=0; //当 m=10 时返回到 0
    else P1 = table[m]; //循环输出 0~9
}
/*****外部中断 1 子程序*****/
void INT_1() interrupt 2
{
    EX1 = 0;    //关闭外部中断 1，防止在执行过程中再次发生中断
    delay(20); //延时 20ms，去抖动
    EX1 = 1;    //开外部中断 1
}

```

```

        m--;
        P1 = table[m];    //循环输出 9~0
        if(m==0) m=10;    //当 m=0 时返回到 10
    }
    /*****主程序*****/
void main()
{
    P1 = 0x00;
    EA = 1;    //打开总中断
    EX0 = 1;    //打开外部中断 0
    IT0 = 1;    //设置中断触发方式为下降沿触发
    EX1 = 1;
    IT1 = 1;
    while(1)    //死循环
}

```