目 录

第 1 章　8086/8051 教学实验系统简介

1.1 简介

PROTEUS 教学实验系统（8086/8051）是我公司针对微机原理与接口技术课程、单片机课程的教学需求所研发的，其目的在于激发学生学习 8086/8051 的兴趣，提高教学质量，缩短教学与工程实际的差距，为社会培养出实践创新型人才。

PROTEUS 是本实验箱进行 8086 实验的必备软件，是电路设计、电路仿真与调试、程序编译的环境。PROTEUS 教学实验系统（8086/8051）主要由教学实验箱、实验指导书及其配套光盘组成。通过 USB 连接线把电脑与实验箱相连接，能完成针对 8086 的各种交互式仿真实验；通过 WWISP 下载器，可以对 8051 芯片进行 ISP 编程，进行单片机实验课程。

本教学实验箱摒弃以往的设计思想，采用模块化设计，总线器件都可以挂在总线上，只须要接上 CS 片选就可以实验，减少了实验过程中的接线问题，同时也可极大地提高学生的实验速度。结合 PROTEUS 的电路仿真功能，能够大大提高学生实验的动手设计能力。

1.2 硬件配置

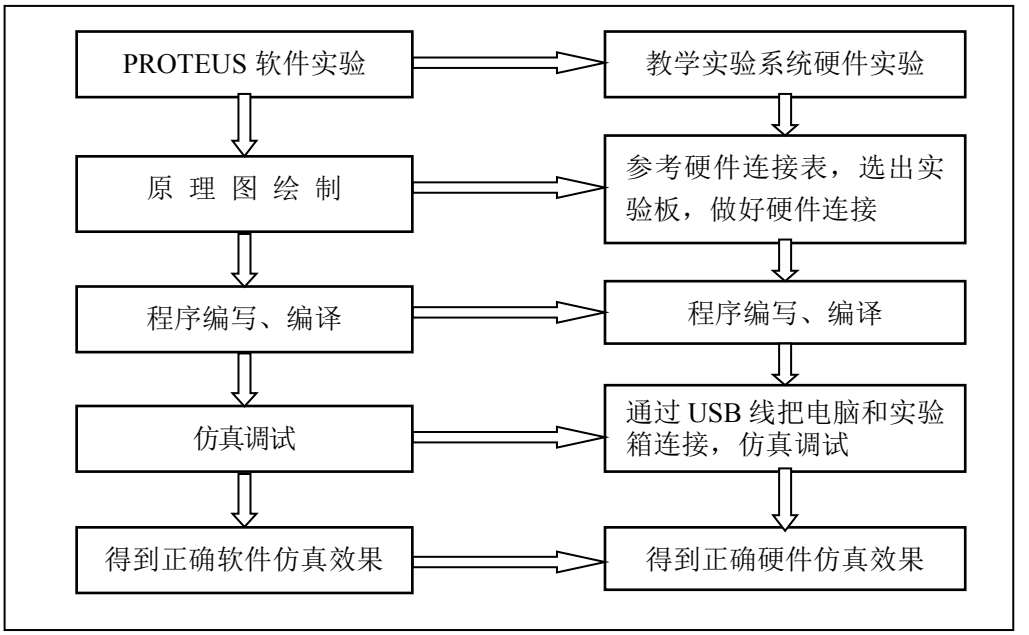
1、 箱体：
铝合金箱：440mm×280mm×130mm、配有交流 220V 转直流-5V、+5V、+12V 和-12V 电源适配器。
2、 核心模块：8086 仿真器，8051 核心板
3、 实验子电路模块
8255 可编程并行接口模块、8251 可编程串行通行接口模块、8253 可编程定时器/计数器模块、8259 中断控制器模块、8237 DMA 控制模块、RAM 存储器模块、数/模转换模块(DAC0832)、模/数转换模块(ADC0809)、8 位联体数码管、8 位独立发光二极管、8 位独立开关、LCD128*64 模块、LCD16*2 模块、温度传感器模块、直流电机模块、步进电机模块、继电器模块、RS232 串行通信模块、4X4 矩阵键盘模块、独立按键模块、4M 信号源模块、6 分频模块、逻辑笔模块、门电路电路模块、蜂鸣器模块、EEPROM 模块、时钟模块、电位器模块。
4、 配件
USB 连接线　　1 根
串口线　　1 根
220V 电源线　　1 根
可级联信号连接线 20 根

1.3 配套资料

1、 PROTEUS 教学实验系统（8086/8051）实验指导书
2、 PROTEUS VSM 详解
3、 所有实验源代码
4、 所有实验 PROTEUS DSN 设计文件
5、 PROTEUS 视频教程
6、 PROTEUS 技术讲座资料
7、 实验使用芯片 DATASHEET
8、 工具：MASM 应用程序、串口调试工具、虚拟串口软件、取模软件、汇编和 C 代码编译工具等。

1.4 实验操作

大部分实验的开展，我们都采用在 PROTEUS 平台下的交互式仿真，使用硬件平台与电脑软件仿真同时进行的方法，实验的开展流程如下：



在进行硬件实验中，有几点需要注意：

1、尽量保持线束的整齐，对于控制线少交叉缠绕。

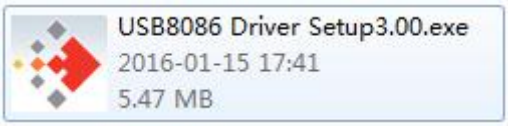
2、拔线时请逐根拔除，切忌强行硬拔整股连线（易造成整股损坏）。

3、液晶类实验涉及到液晶对比度的调节，请通过邻近电位器来调整。

完成驱动安装

1.5 8086 仿真器驱动及编译器安装

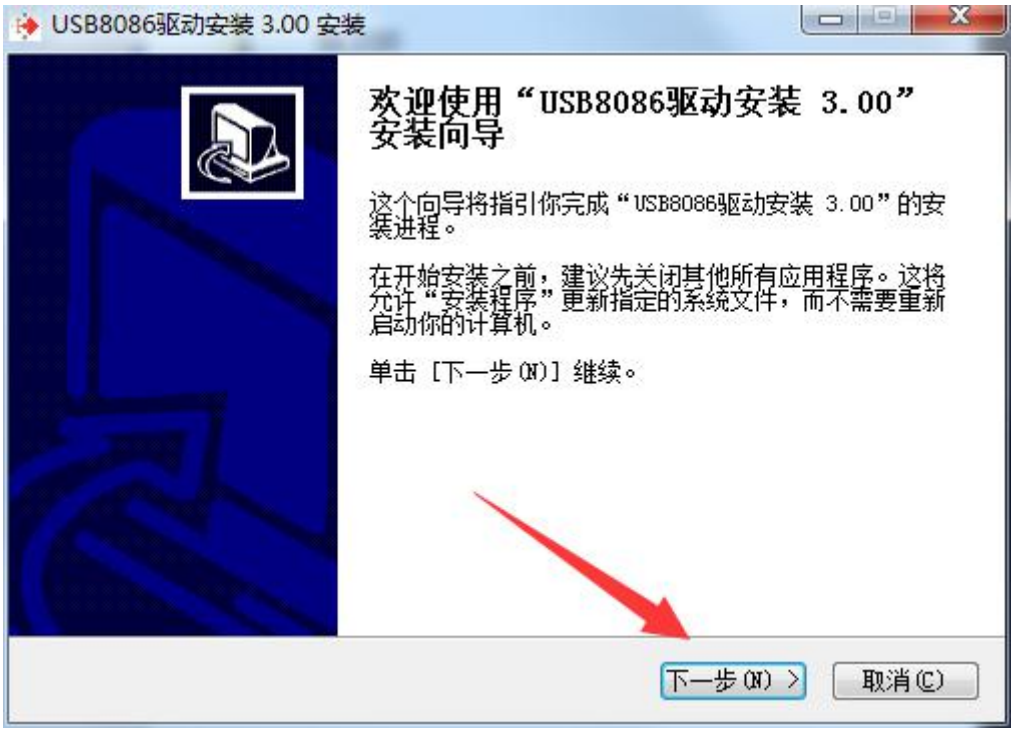
1、双击打驱动软件



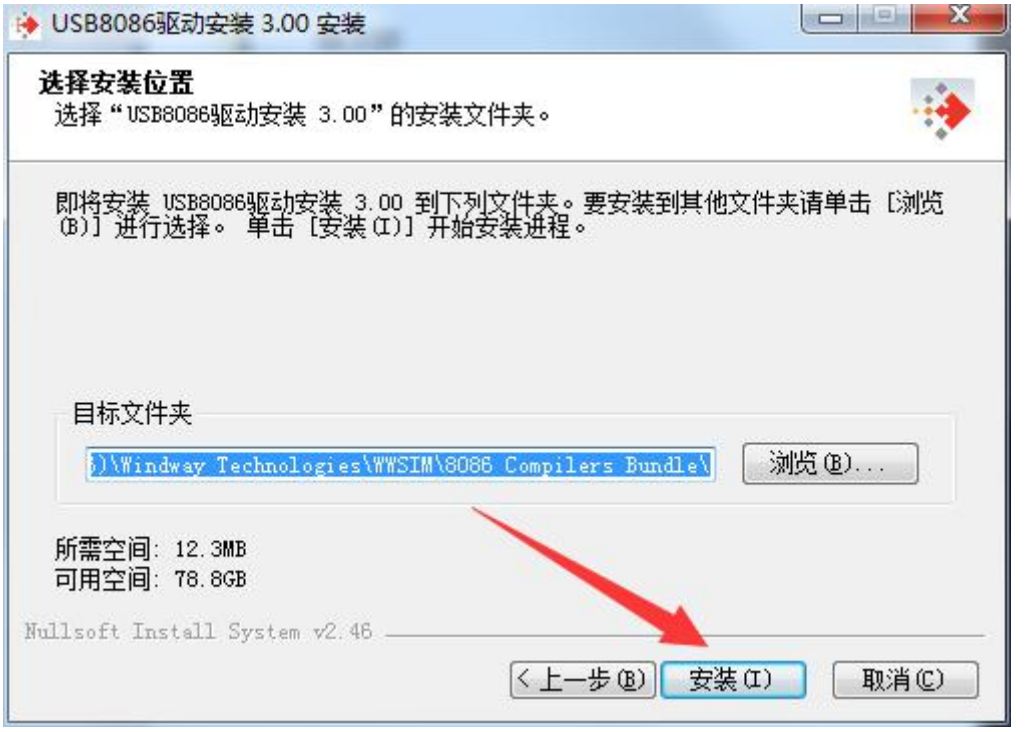
2、proteus 软件要提前关闭，如果软件已关闭点“是”，没有关闭请点“否”

3、语言选择，默认中文，点击“OK”

3、安装向导，点击下一步



4、驱动路径、默认安装，点击“安装”

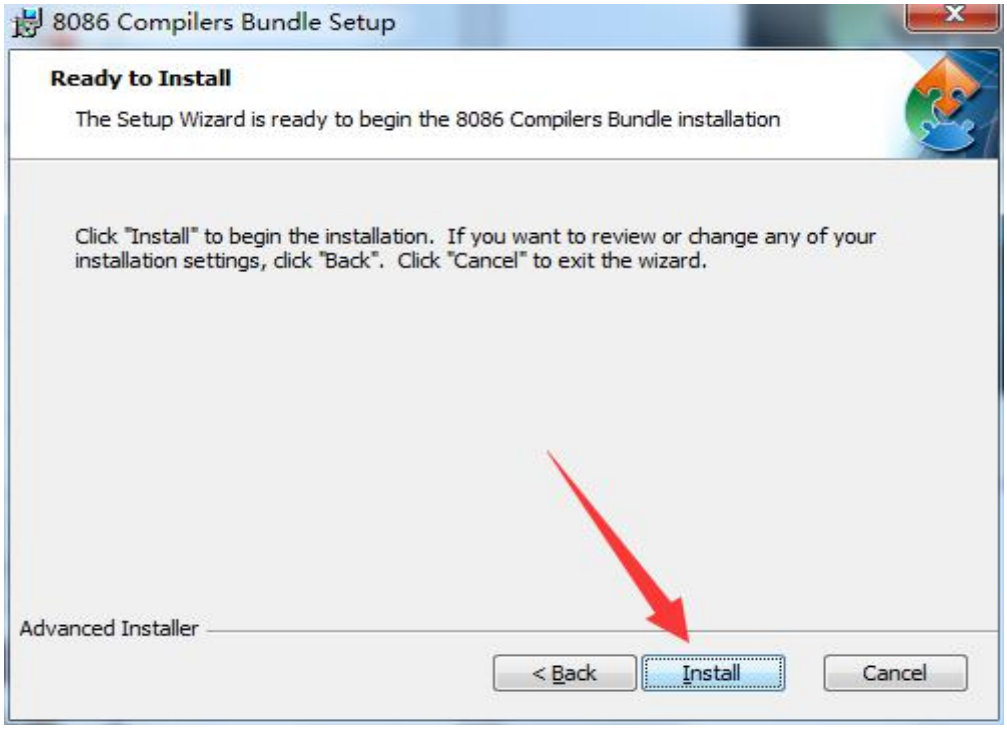


5、安装过程，稍作等待.....

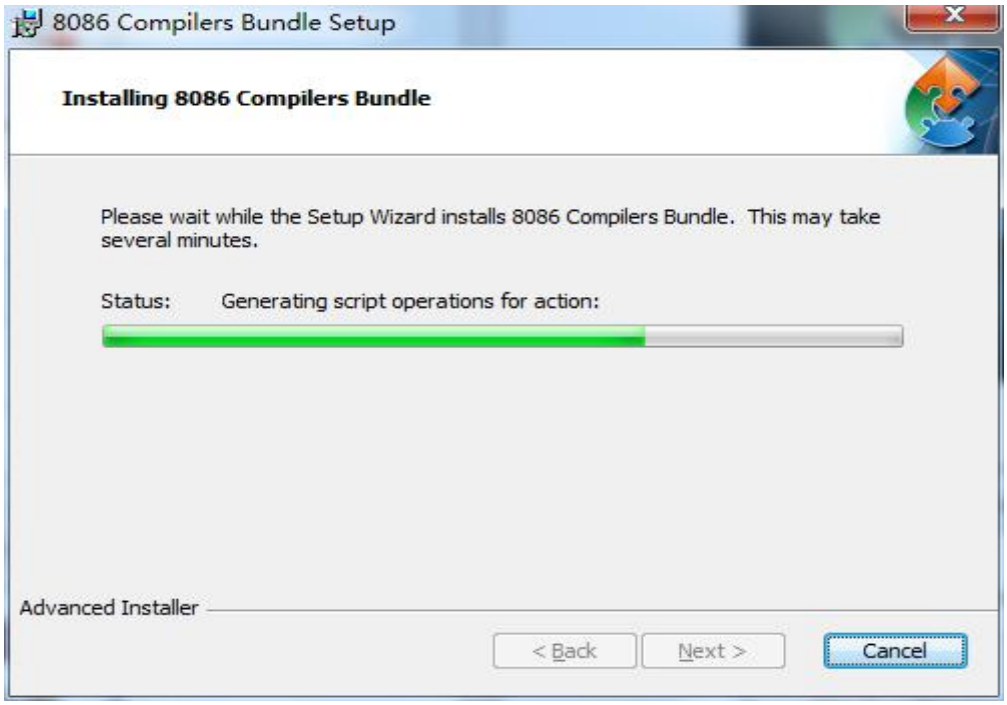
6、驱动安装完毕，还需要安装编译器，勾选对勾，点击“完成”

7、这里是编译器安装，点击“Next>”

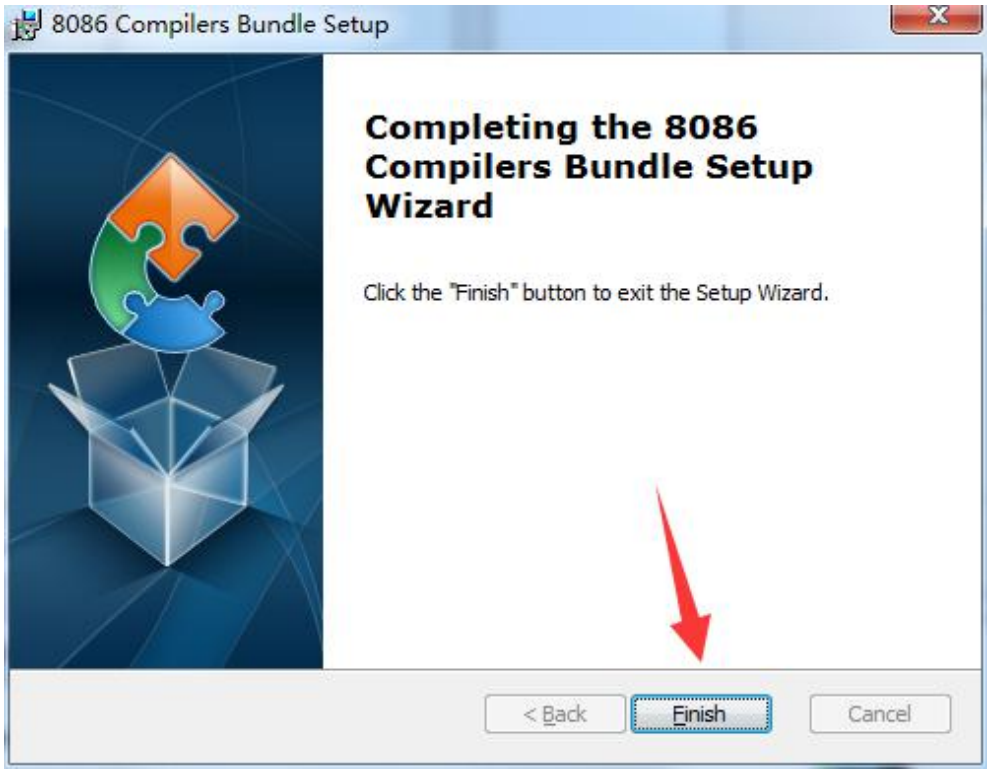
8、点击“Install”



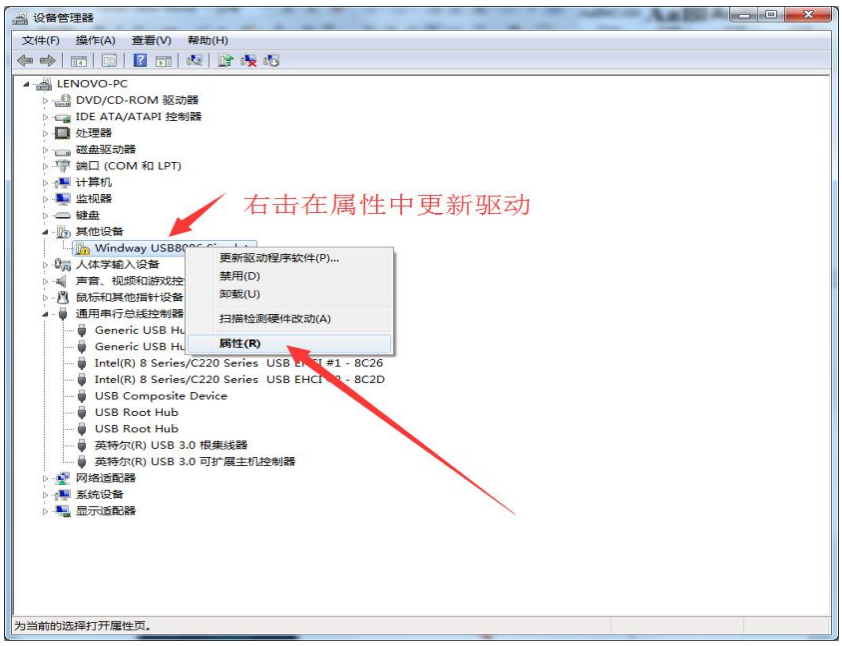
9、安装过程，稍作等待...



10、安装完成，点击“Finish”

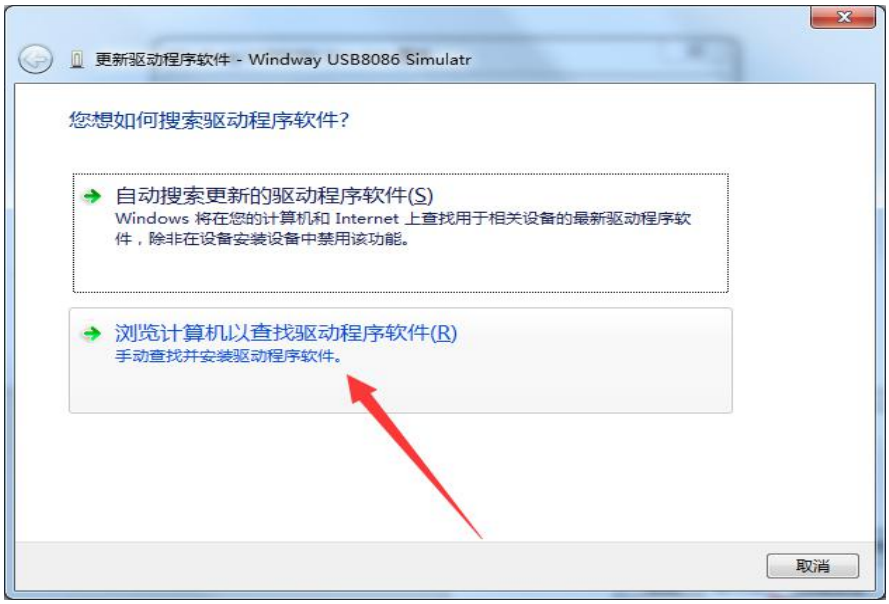


11、连接硬件，并上电，从设备管理器，查看硬件驱动情况，一般硬件会默认加载驱动，如果没有加载成功，如下图所示，请在驱动上“右击”-->“属性”进行驱动更新。

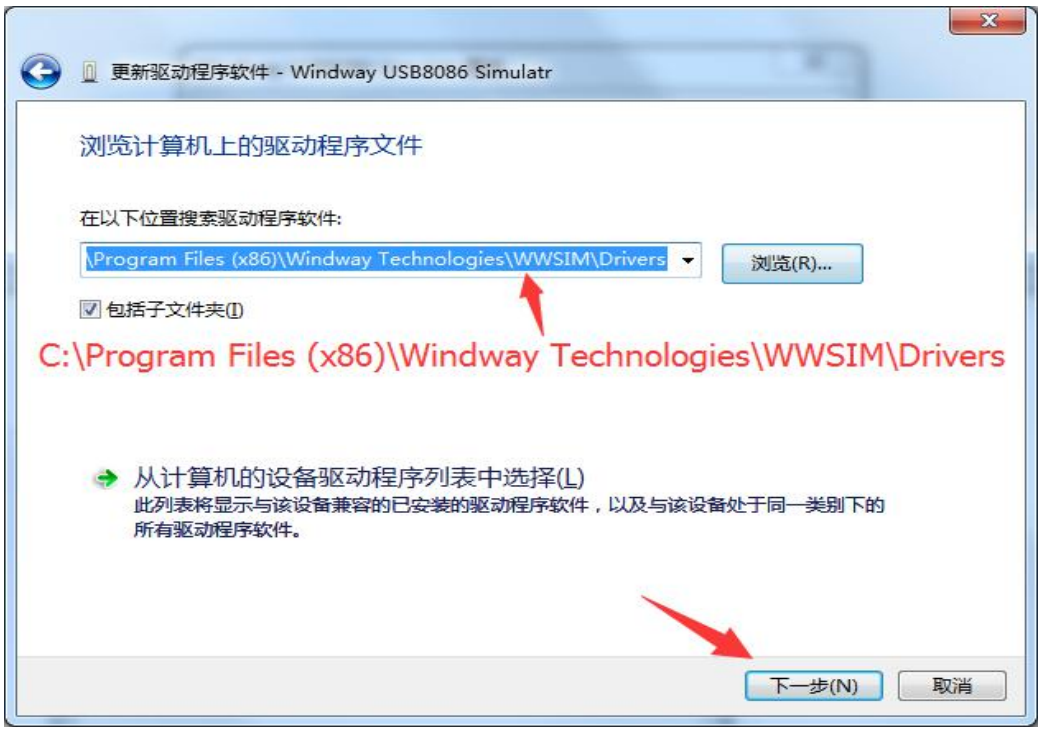


12、按图片所示，进行点击

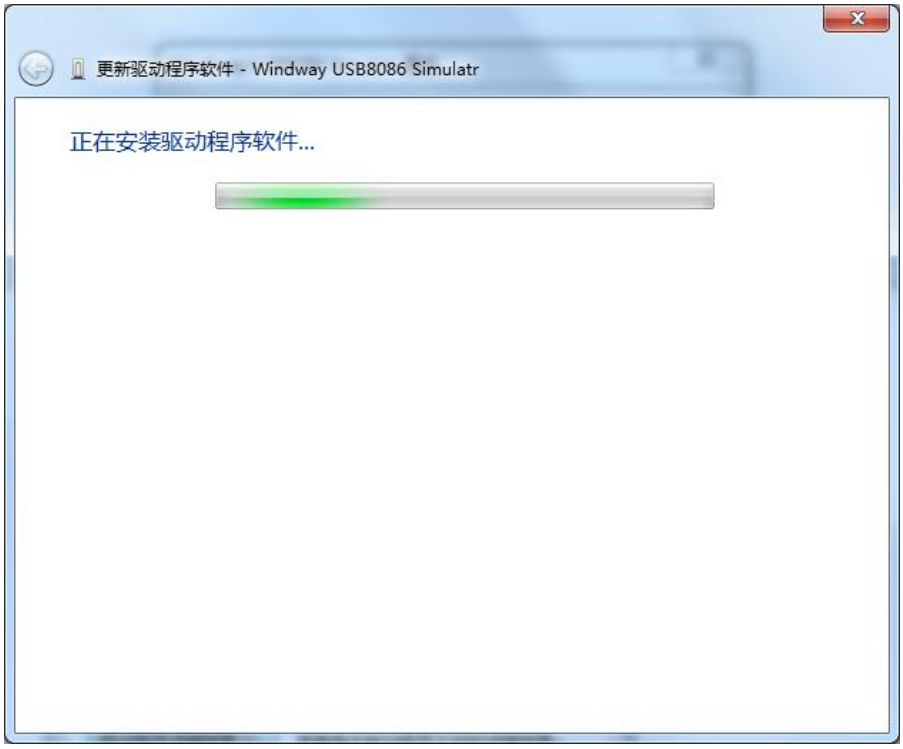
13、浏览本机安装路径加载驱动“C:\Program Files (x86)\Windway Technologies\WWSIM\Drivers”



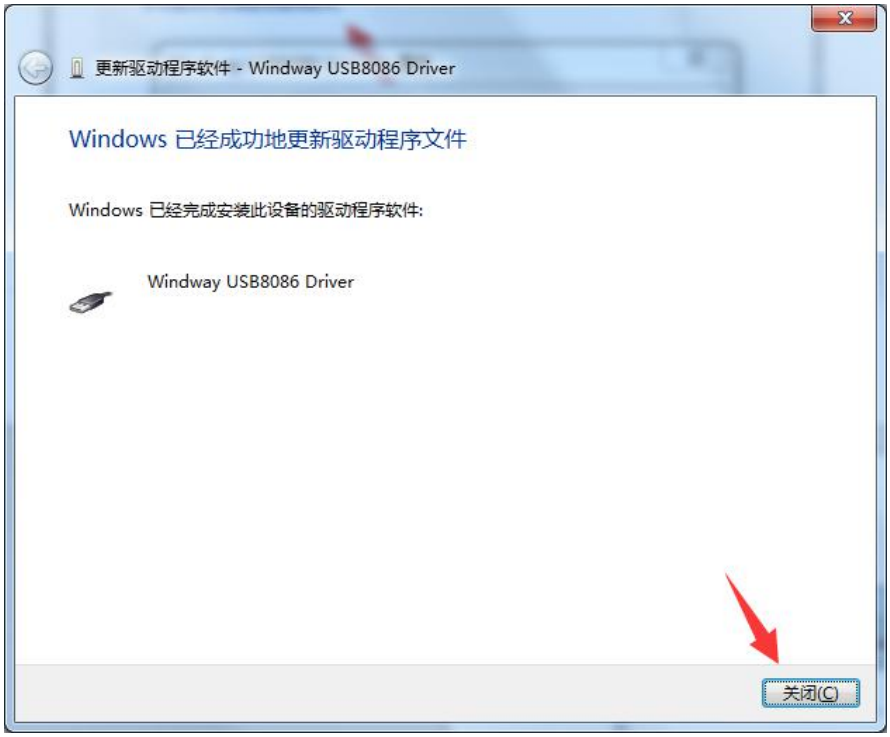
14、选定驱动路径，点击下一步进行安装。



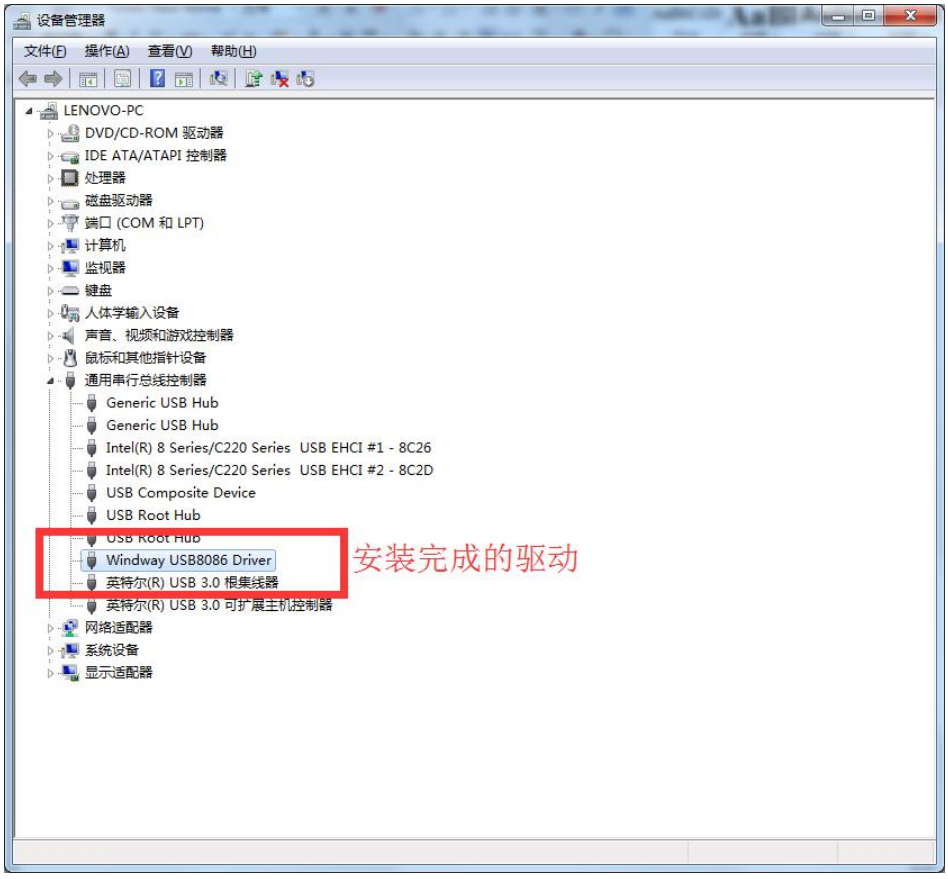
15、安装过程



16、安装完成



17、设备管理器查看驱动状态，下图为正常状态



第 2 章　8086 实验目录

2.1 新建工程与编译器配置

一、 实验要求

Proteus 本身不带有 8086 的汇编器和 C 编译器，因此必须使用外部的汇编器和编译器。汇编器有很多，如 TASM、MASM 等。C 编译器也有很多，如 Turbo C 2.0，Borland C，VC++，Digital Mars C Compiler 等。实验箱选用的是免费的 MASM 和 Digital Mars C Compiler。其中 MASM 的版本是 6.14.8444，Digital Mars C Compiler 的版本是 8.42n。我们之前在章节 1.5 8086 仿真器驱动及编译器安装中就已经安装完了编译器了。本实验就是让大家学会怎样在 Proteus8 版本中如何新建汇编工程，C 语言工程类似。

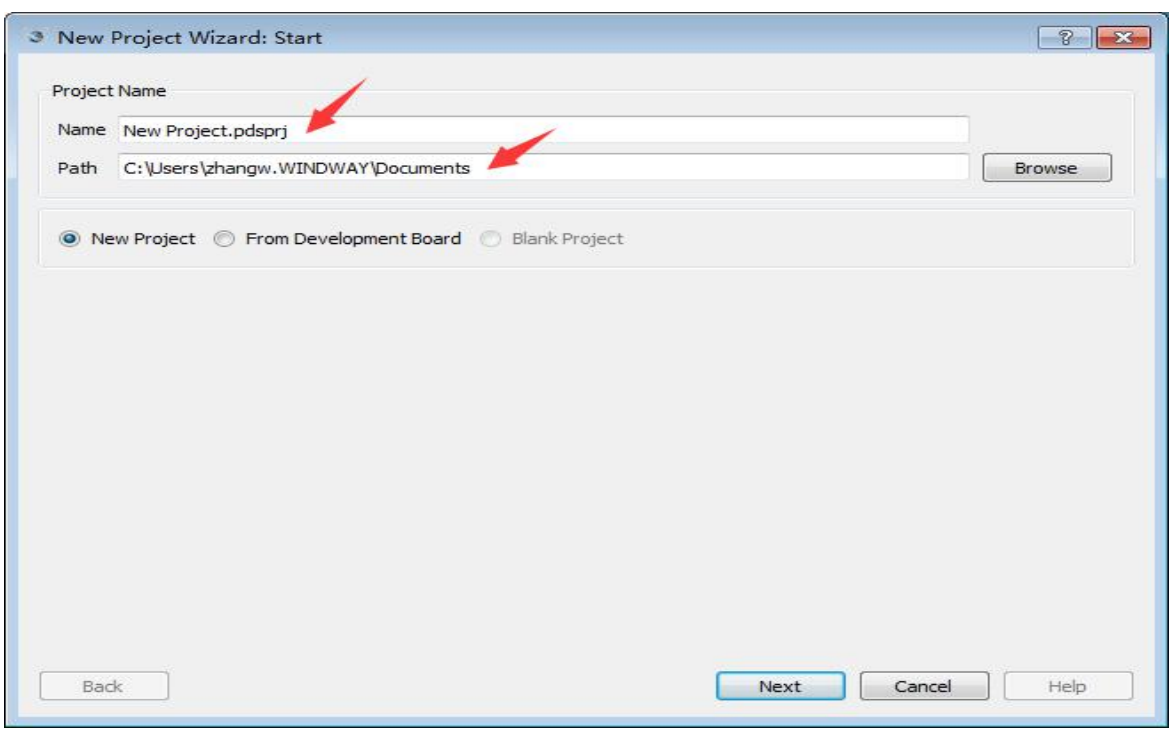
二、 实验目的

1、掌握 PROTEUS 如何新建和编译工程；

三、 系统环境

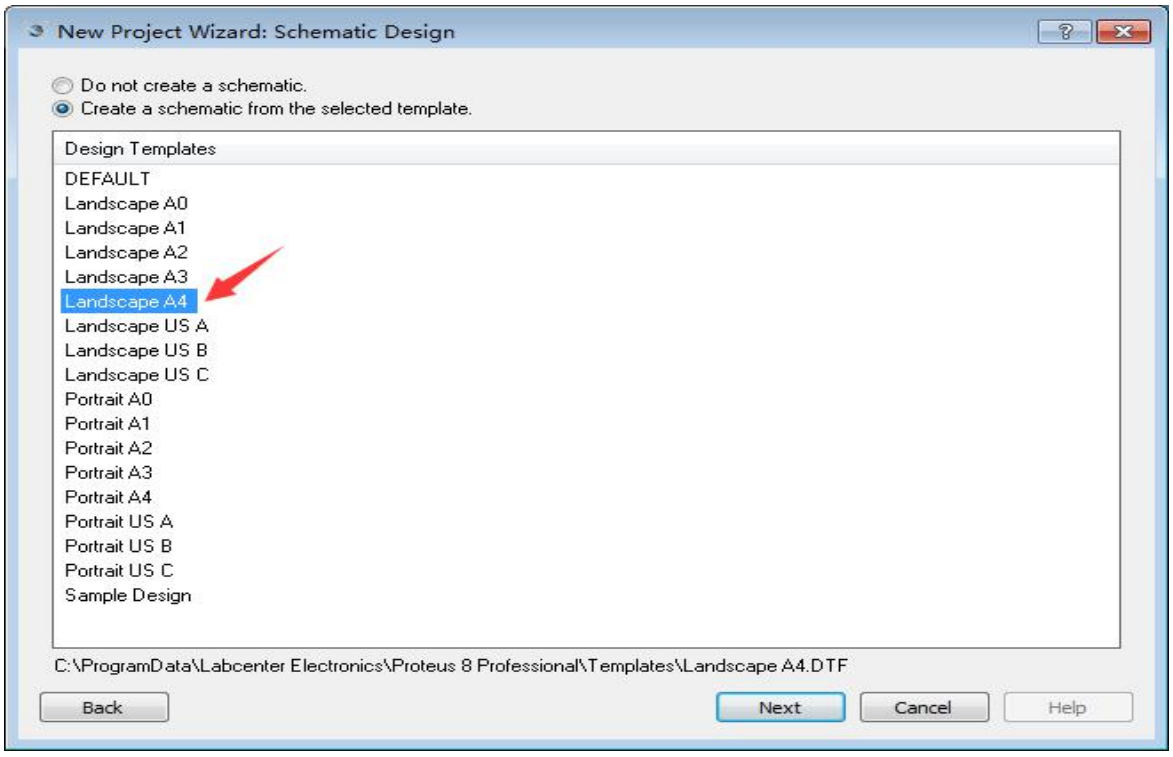
1、**新建工程**

首先，打开 PROTEUS 软件，打开菜单“File->New Project”。

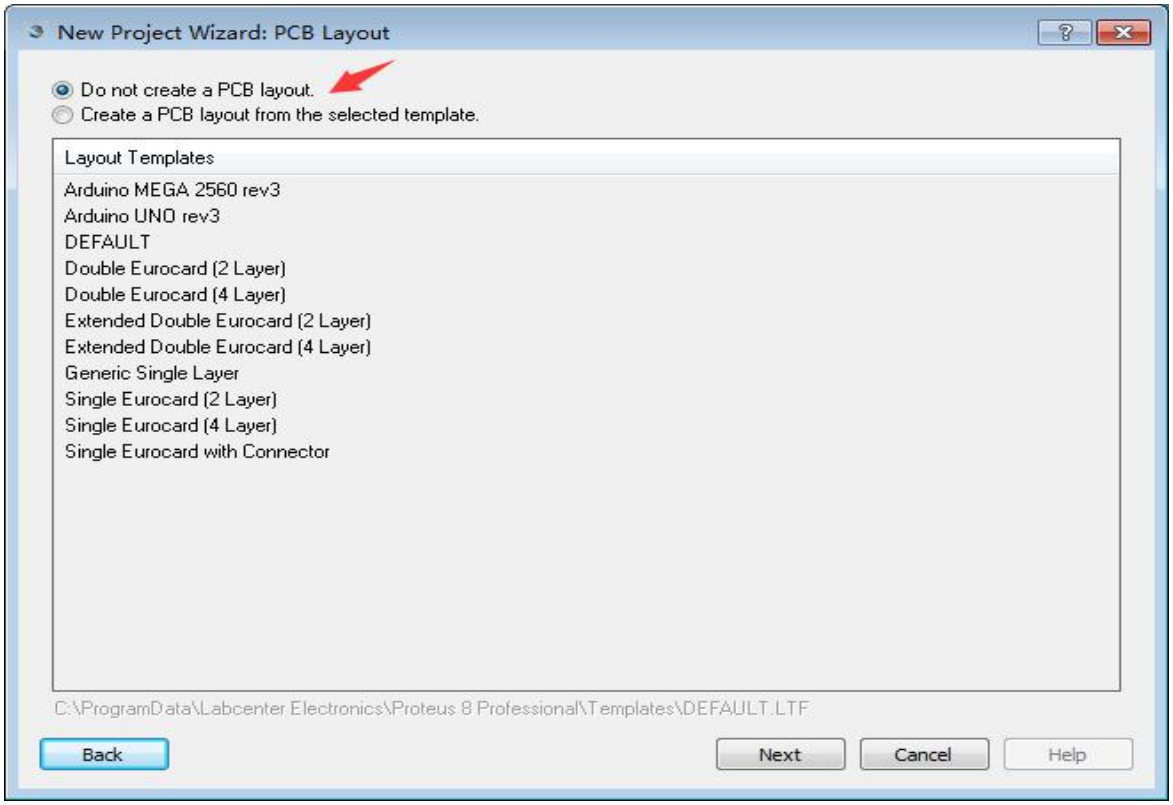


可以在 Name 下修改工程名，在 Path 下修改工程保存的路径。

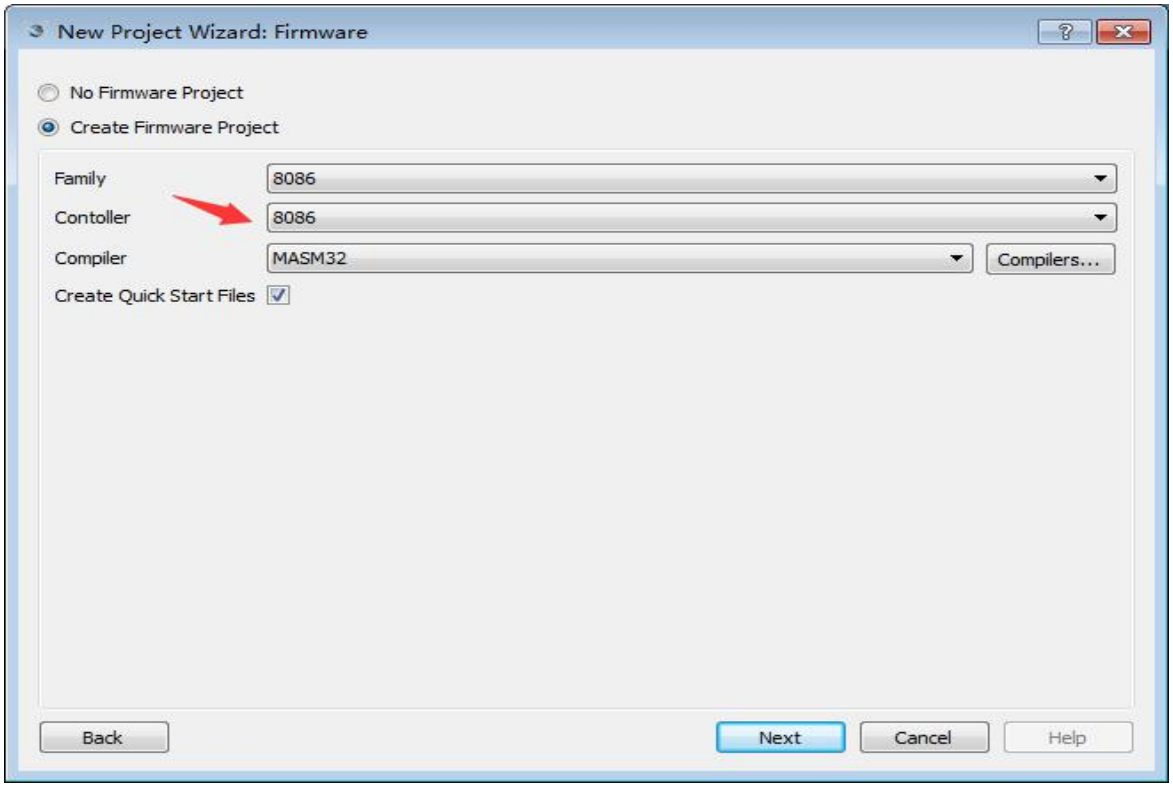
点击 Next，选择原理图的样式大小



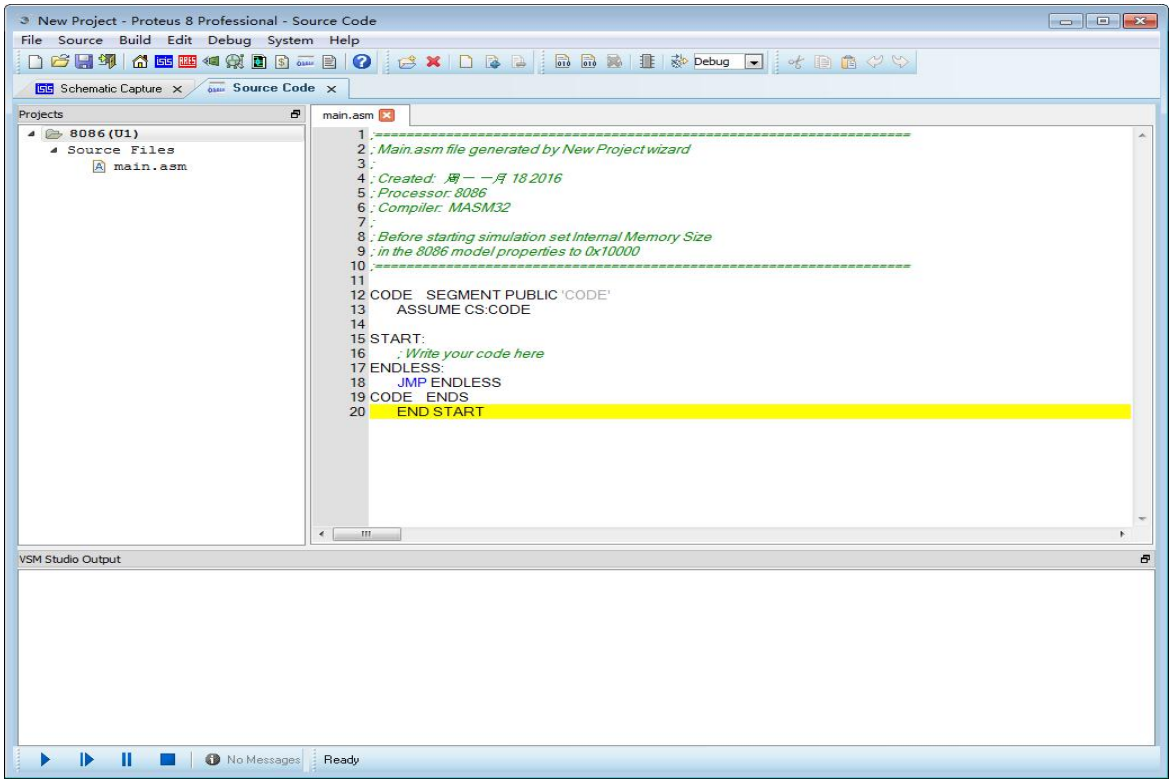
点击 Next，选择 PCB 模板，如不需要可选择默认选项



点击 Next，选择固件工程，选择好需要的控制器，选择编译器

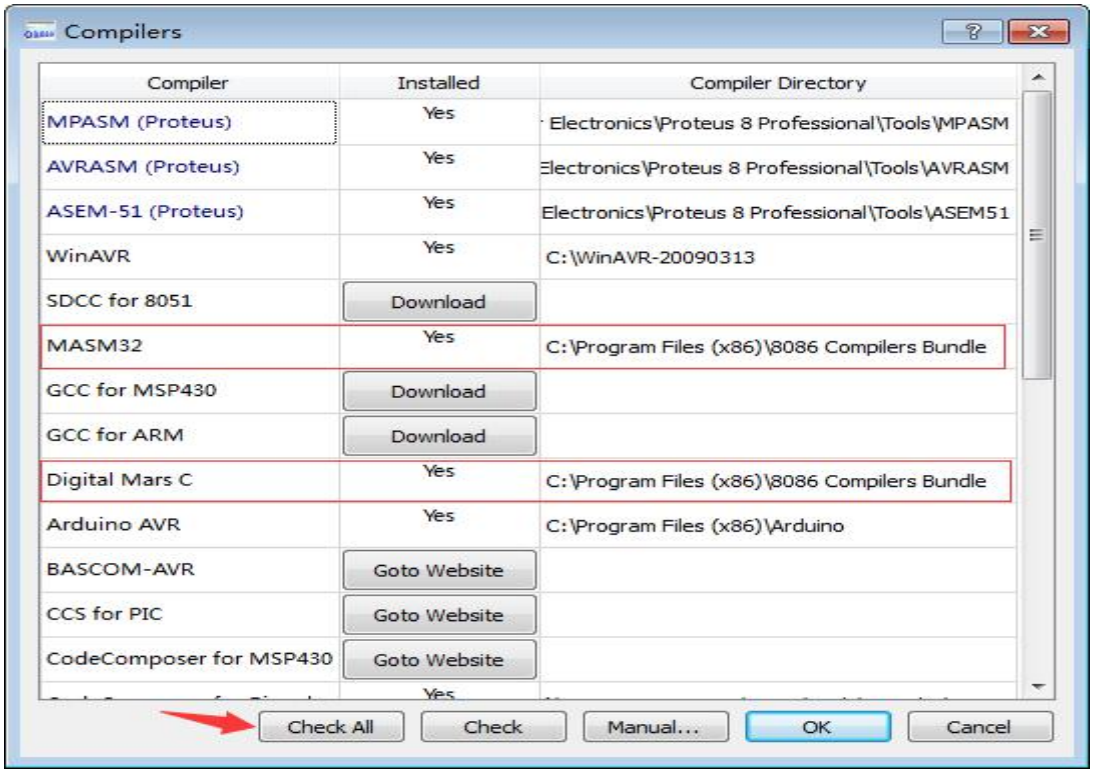


点击 Next，然后点击 Finish,汇编程序模板生成

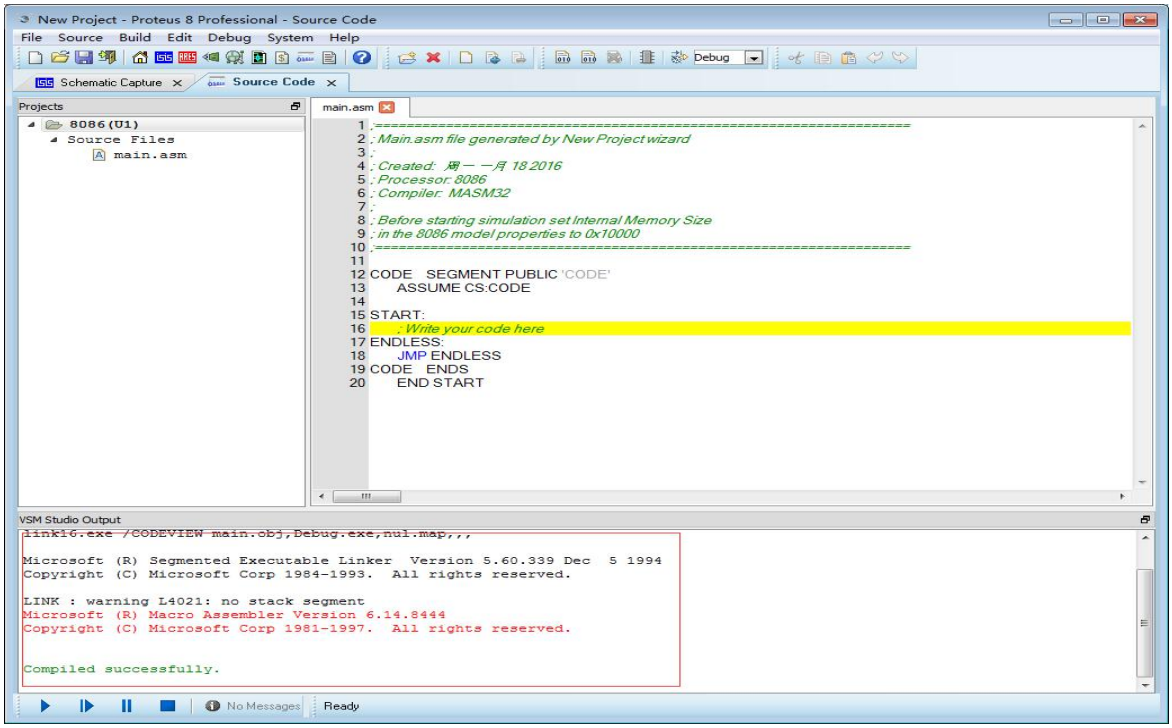


2、**编译工程**

点击菜单栏 System->Compilers Configurations，在对话框中，点击 Check All 能找到如下图的编译器则安装正确，点击 OK 关闭对话框

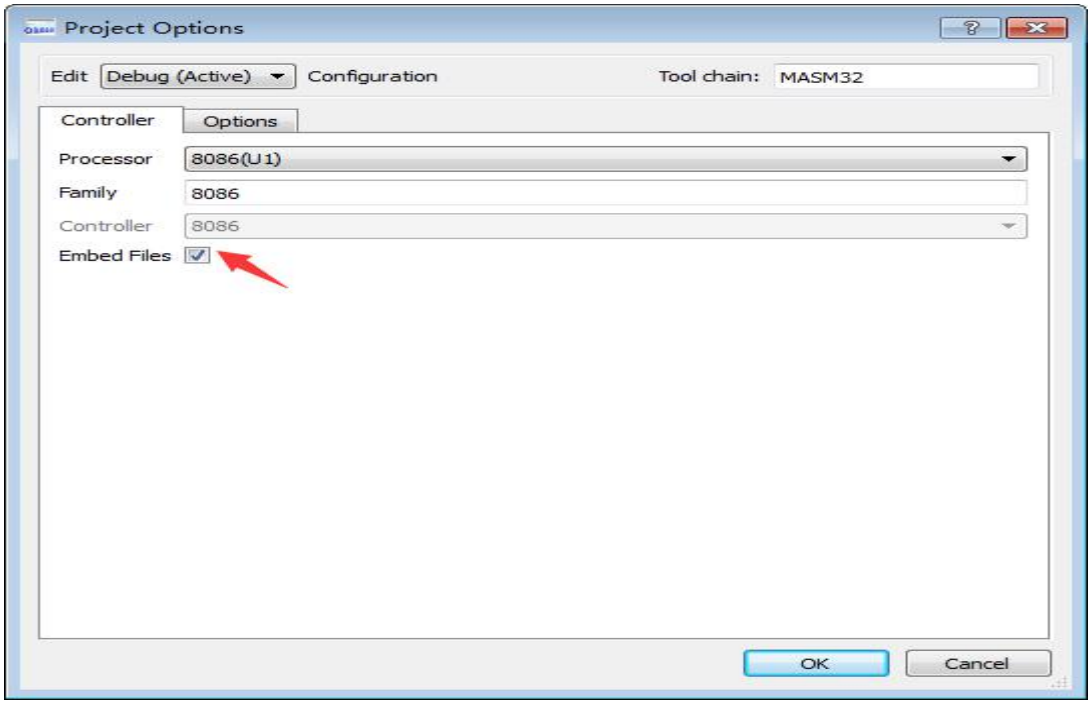


点击菜单栏 Build->Build Project,如果编译成功，如下图

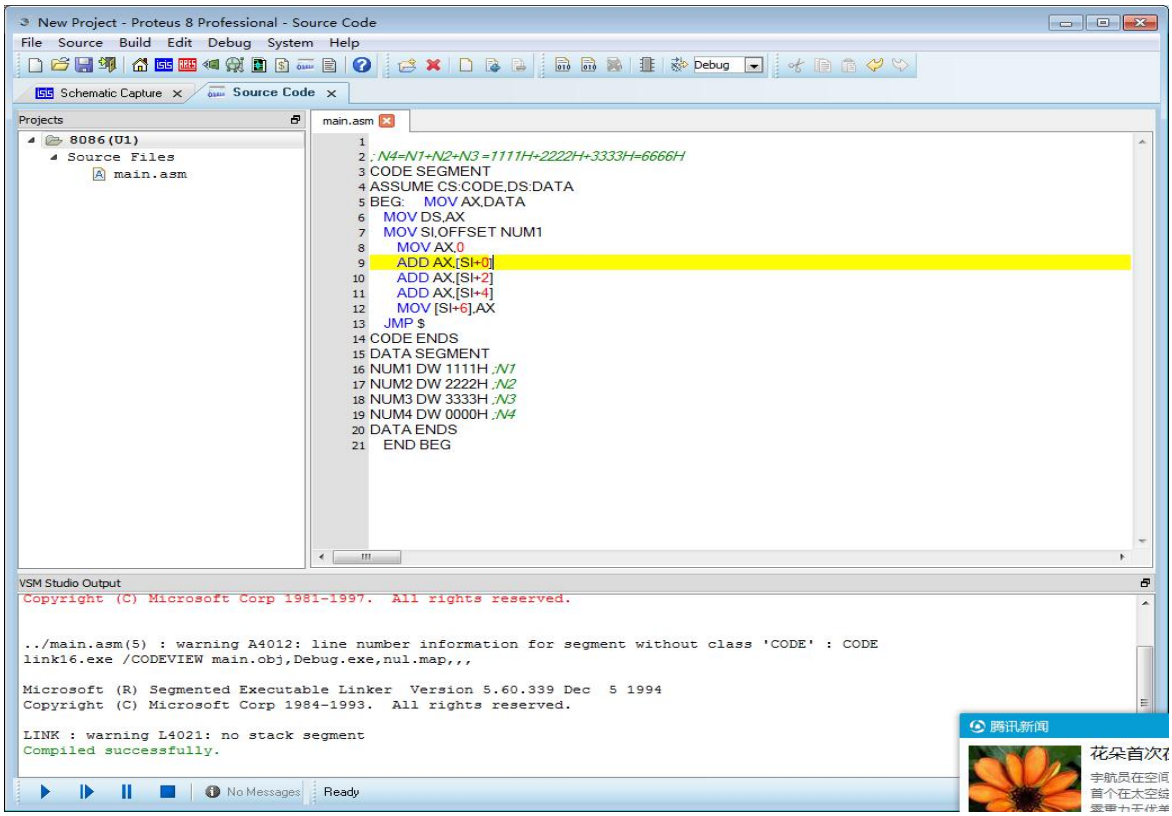


接下来我们就可以添加自己的代码了，不过还有个需要注意的地方

点击菜单栏Build->Project Settings,如果需要工程文件和源代码文件在同一路径，则需要取消 Embed Files 的复选勾，否则源代码会放到Ｐ ｒ ｏ ｔ ｅ ｕ ｓ默认路径



添加源代码，编译工程如下图，完成



四、 、实验结果和体会

五、 、建议

2.2 仿真调试技巧

Proteus 中提供了很多调试工具和手段，这些工具的菜单都放在 Proteus 的 Debug(调试)菜单下，如下图所示：

第一栏的菜单是仿真开始、暂停与停止的控制菜单，与 Proteus ISIS 左下角的仿真控制按钮的功能是一样的。

第二栏是执行菜单，可以执行一定的时间后暂停，也可以加断点执行和不加断点执行。

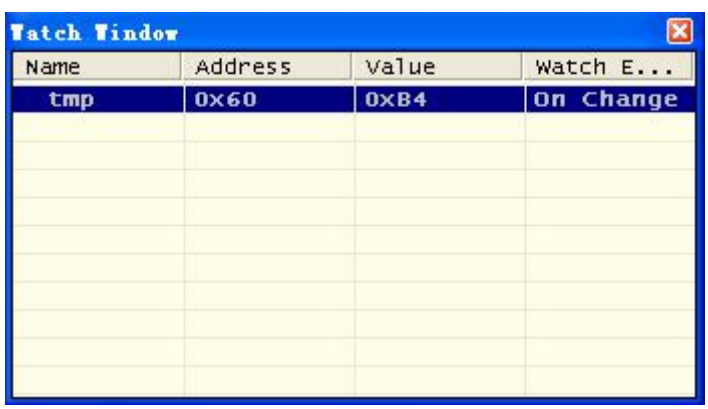
第三栏是代码调试菜单，有单步、连续单步，跳进/跳出函数，跳到光标处等功能。

第四栏是诊断和远程调试监控，但 8086 没有远程监控功能。诊断可以设置对总线读写，指令执行，中断事件和时序等进行跟踪。有四个级别，分别是取消、警告、跟踪和调试。级别的不同，决定事件记录的不同。例如，如果要对中断的整个过程进行详细的分析，则可以选择跟踪或者调试级别，ISIS 将会对中断产生的过程，响应的过程进行完整的记录，有助于学生加深中断过程的理解。

设置诊断选项

最后一栏是 8086 的各种调试窗口，包括观察窗口，存储器窗口，寄存器窗口，源代码窗口和变量窗口。

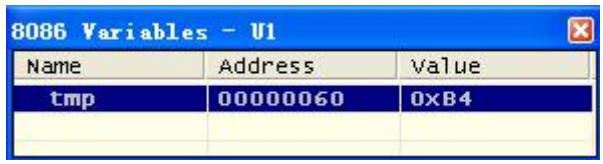
其中观察窗口可以添加变量进行观察，并且可以设置条件断点。这在调试程序的时候非常有用。



观察窗口

设置条件断点

变量窗口会自动把全局变量添加进来，并实时显示变量值，但不能设置条件断点。

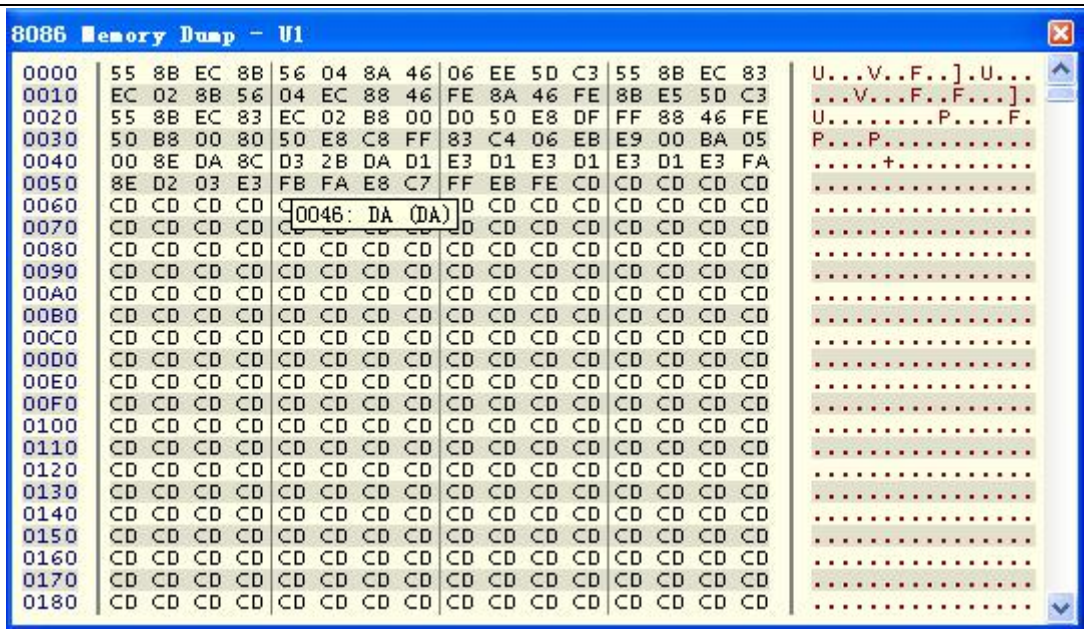


变量窗口

寄存器窗口实时显示 8086 各个寄存器的值。

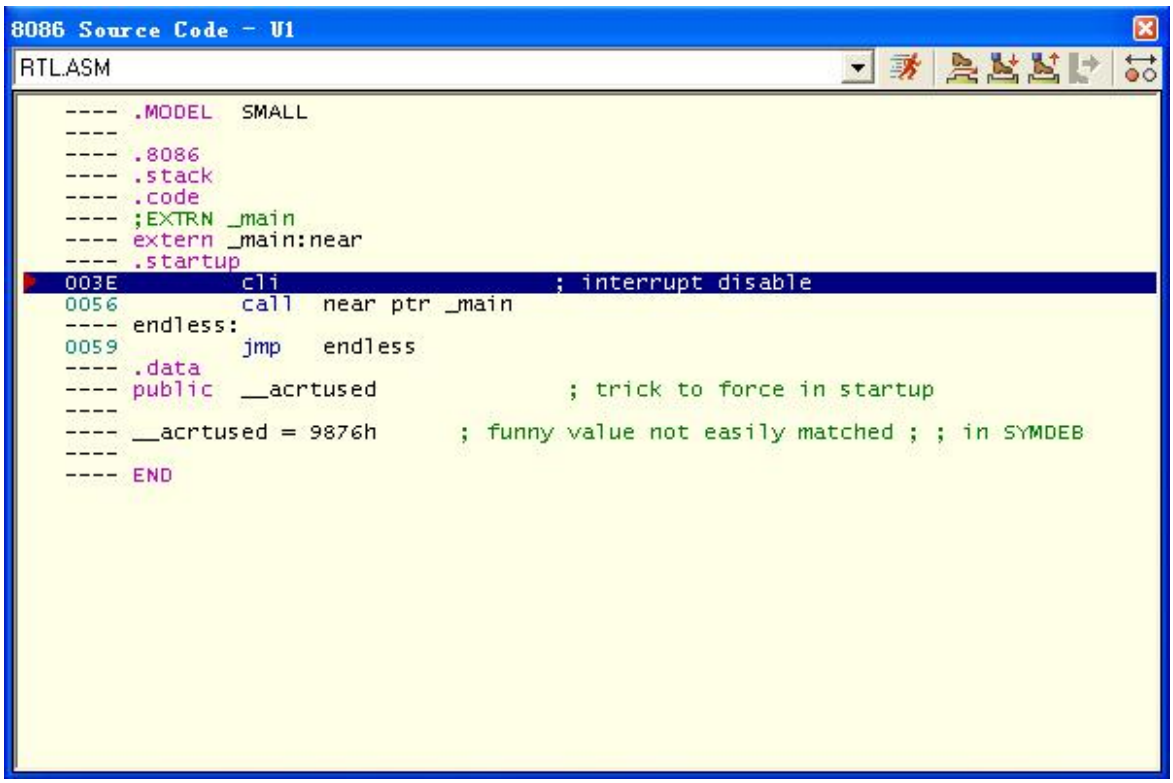
寄存器窗口

存储器窗口实时显示存储器的内容，仿真开始的时候，ISIS 会自动把可执行文件.exe 加载到 0x0000 地址开始的一段空间内。



存储器窗口

源代码调试窗口是最主要的调试窗口，在这里可以设置断点，控制程序的运行，如果是 C 程序，还可以进行反汇编。



以上几个工具配合起来，比起任何的 IDE 都要实用的多，可以大大提高学生的学习效率。

2.3 实验一（1） 循环程序实验

一、 实验要求

利用 PROTEUS 平台，建立 8086 的循环程序的例子。

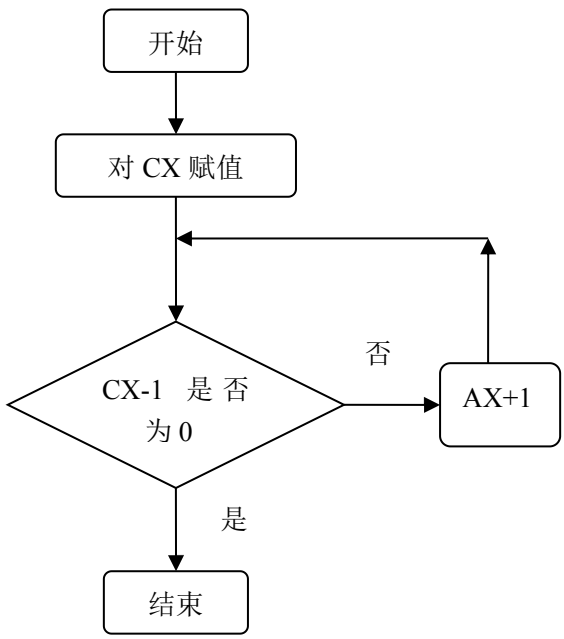
二、 实验目的

1、熟悉实验系统的编程和使用。
2、掌握使用 LOOP 判断转移指令实验循环的方法。
3、掌握使用 LOOP 与 CX 的组合。

三、 实验说明

由于本实验是通过给 CX 一个数值，再通过 LOOP 作一个判断 CX-1 是否为 0 的转移，实现程序的循环，循环的内容是执行 AX+1， 所以结果应该为 AX 最后大小为开始时给定 CX 的大小。

四、 实验程序流程图



五、 实验步骤

1、Proteus 仿真

a. 在 Proteus 中打开设计文档“循环程序.pdsprj”;
b. 单步运行，打开调试窗口进行调试。
参考程序：

```
CODE SEGMENT
```

```
        ASSUME CS:CODE
CON_A EQU   25
CON_B EQU   12
START:
            MOV AX,0
            MOV CX,5
INC_AX:NOP
            INC AX
            LOOP INC_AX
            JMP $
CODE ENDS
            END START
```

2、调试、验证

a．设置断点、单步运行程序，一步一步调试；

b．观察每一步运行时，8086 内部寄存器的数值变化；

c．改变 CX 的赋值大小，观察 AX 的变化；

d．检查验证结果。

六、 实验结果和体会

七、 建议

2.3 实验一（2） 分支程序实验

一、 实验要求

利用 PROTEUS 平台，建立 8086 的分支程序的例子。

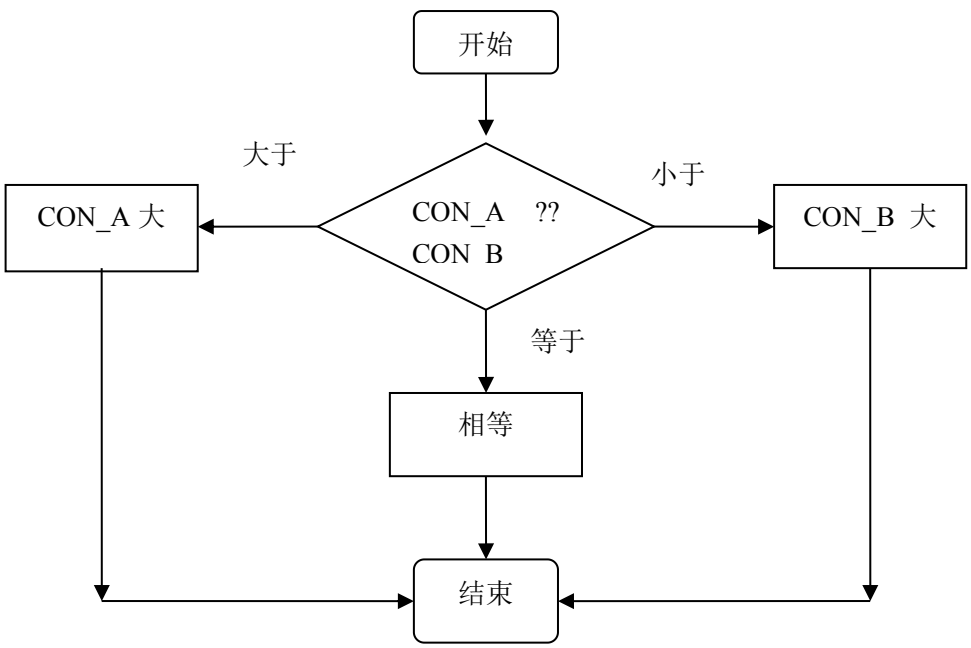
二、 实验目的

1、熟悉实验系统的编程和使用。
2、掌握使用转移类指令编程及调试方法。
3、掌握各种标志位的影响。

三、 实验说明

由于本实验是通过改变两个变量 CON_A 和 CON_B 的大小，实现用 CMP 指令对不同标示位的影响的一个转移，分别设有大于、等于和小于。

四、 实验程序流程图



五、 实验步骤

1、Proteus 仿真

a. 在 Proteus 中打开设计文档“分支程序.pdsprj”;
b. 单步运行，打开调试窗口进行调试。
参考程序：

```
CODE SEGMENT
```

```
        ASSUME CS:CODE
CON_A EQU   25
CON_B EQU   12
START:
            MOV AX,CON_A
            MOV BX,CON_B
            CMP AX,BX
            JNC MO_T    ;AX > BX  跳转
            JE  EQUA    ;AX = BX  跳转
            JC  LESS    ;AX < BX  跳转
MO_T:       JMP $
EQUA:       JMP $
LESS:       JMP $
CODE ENDS
            END START
```

2、调试、验证

a．设置断点、单步运行程序，一步一步调试；

b．观察每一步运行时，8086 内部寄存器的数值变化；

c．改变两个变量的大小，观察三程序跳转的实现；

d．检查验证结果。

六、 实验结果和体会

七、 建议

2.4 实验二 数据排列实验

一、 实验要求

利用 PROTEUS 平台，建立 8086 的由小到大的数据排列例子。

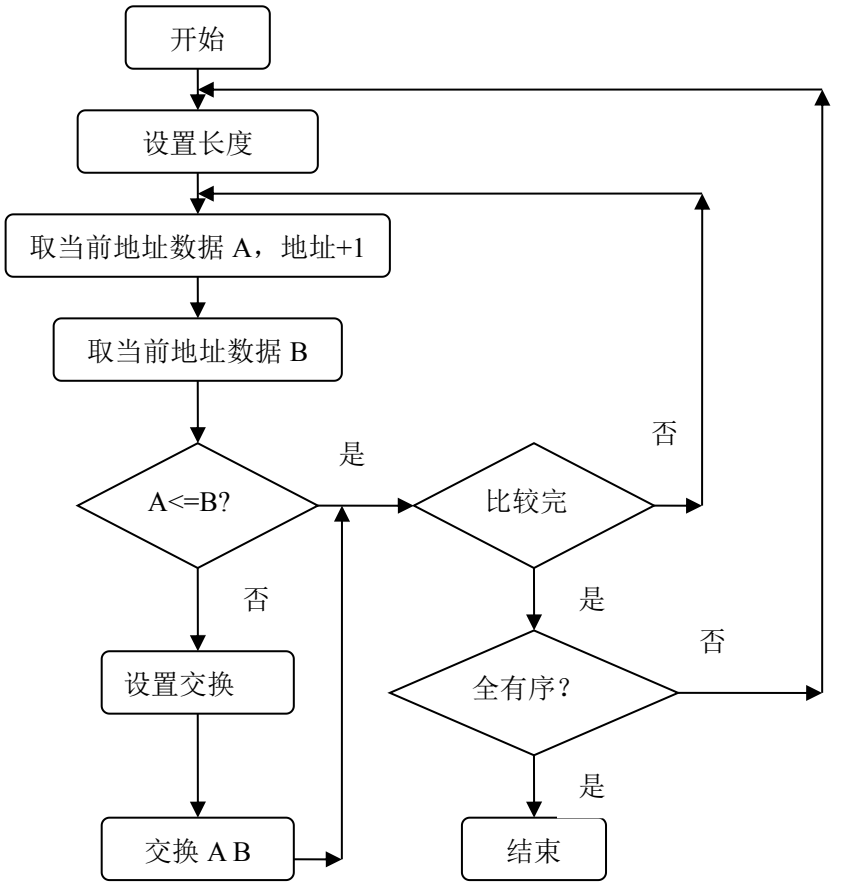
二、 实验目的

1、熟悉实验系统的编程和使用。
2、了解排列的简单算法。
3、了解“冒泡排序”法。

三、 实验说明

有序的数列更有利于查找。本程序用的是“冒泡排序”法，算法是将一个数与后面的数相比较，如果比后面的数大，则交换，如此将所有数比较一遍后，最大的数就会在数列的最后面。再进行下一轮比较，找出第二大数据，如此下去，直到全部数据由小到大排列完成。

四、 实验程序流程图



五、 实验步骤

1、Proteus 仿真

a. 在 Proteus 中打开设计文档“由小到大排列.pdsprj”;
b. 设置断点、运行程序，打开调试窗口进行调试。
参考程序：

```
CODE SEGMENT
        ASSUME CS:CODE,DS:DATA,SS:STACK
START:
                MOV AX,DATA
                MOV DS,AX
                MOV DX,COUNT-1
                MOV BL,0FFH
AGAINO:         CMP BL, 0
                JE DONE
                XOR BL,BL
                MOV CX,DX
                MOV SI,COUNT-1
AGAIN1:         MOV AL,ARRAY[SI]
                CMP AL,ARRAY[SI-1]
                JAE UNCH
EXCH: XCHG ARRAY[SI-1],AL
                MOV ARRAY[SI],AL
                MOV BL,0FFH
UNCH:           DEC SI
                LOOP AGAIN1
                DEC DX
                JNZ AGAINO
DONE:           JMP $
CODE ENDS
DATA SEGMENT
                ARRAY DB    25,46,3,75,5,30
                COUNT EQU $-ARRAY
DATA ENDS
STACK SEGMENT PARA STACK 'STACK'
                DB 60 DUP(?)
STACK ENDS
                END START
```

2、调试、验证

a．设置断点、单步运行程序；

b．观察程序运行到断点时，8086 内部寄存器的数值变化；

c．由于在 0040H 单元开始的 6 个字节 25,46,3,75,5,30 = 19H,2EH,03H,4BH,05H,1EH 所以由小到大排列后为：03H,05H,19H,1EH,2EH,4BH

d．检查验证结果是否由小到大的把数据区中的数据排列完成

六、 实验结果和体会

七、 建议

2.5 实验十 8255 并行 I/O 扩展实验

一、 实验要求

利用 8255 可编程并行口芯片，实现输入、输出实验，实验中用 8255PA 口作读取开关状态输入，8255PB 口作控制发光二极管输出

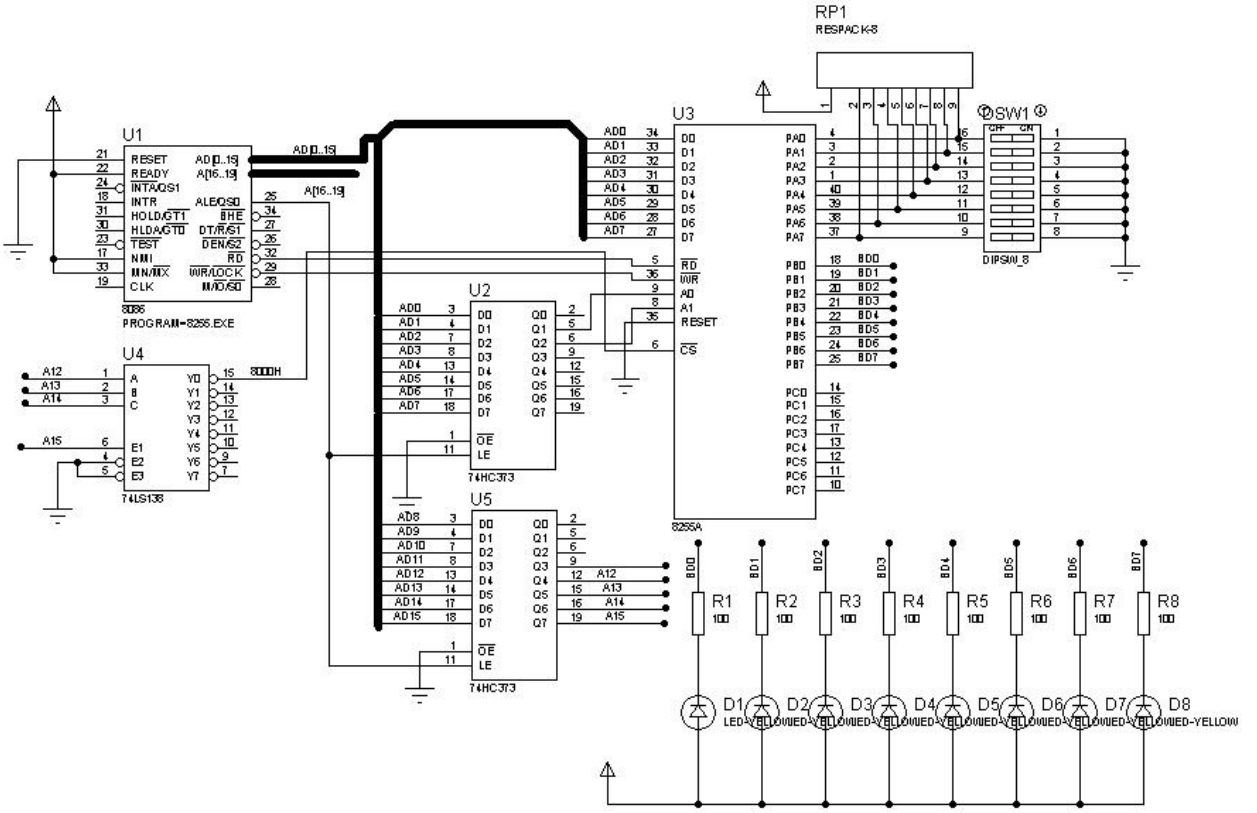
二、 实验目的

1、了解 8255 芯片结构及编程方法。

2、了解 8255 输入、输出实验方法。

三、 实验电路及连线

1、Proteus 实验电路



2、硬件验证实验

硬件连接表

接线孔 1	接线孔 2
8255 CS	8000H-8FFFH
PB0--PB7	D1--D8
PA0--PA7	SW1--SW8

四、 实验说明

1、8255A 芯片简介：8255A 可编程外围接口芯片是 INTEL 公司生产的通用并行接口芯片，它具有 A、B、C 三个并行接口，用+5V 单电源供电，能在以下三种方式下工作：

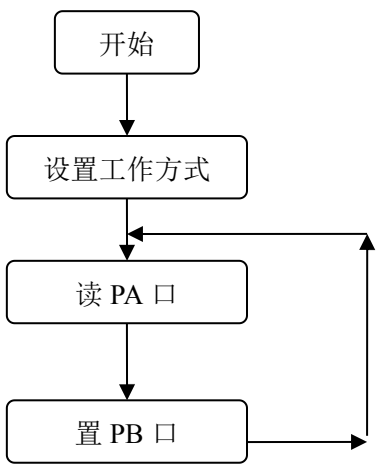
方式 0：基本输入/ 输出方式

方式 l：选通输入/ 输出方式

方式 2：双向选通工作方式

2、使 8255A 端口 A 工作在方式 0 并作为输入口，读取 Kl-K8 个开关量，PB 口工作在方式 0 作为输出口。

五、 实验程序流程图



六、 实验步骤

1、Proteus 仿真

a. 在 Proteus 中打开设计文档 8255_STM.DSN;

b. 建立实验程序并编译，仿真;

c. 如不能正常工作，打开调试窗口进行调试。

汇编语言参考程序：

```
CODE      SEGMENT ;
          ASSUME CS:CODE
IOCON   EQU 8006H
IOA       EQU 8000H
IOB       EQU 8002H
IOC       EQU 8004H
```

```
START:
        MOV AL,90H
        MOV DX,IOCON
        OUT DX,AL
        NOP
START1: NOP
        NOP
        MOV AL,0
        MOV DX,IOA
        IN AL,DX
        NOP
        NOP
        MOV DX,IOB
        OUT DX,AL
    JMP START1
CODE ENDS
        END START
```

C 语言参考程序：

```
#define IOCON 8006H
#define IOA   8000H
#define IOB   8002H
#define IOC   8004H

void outp(unsigned int addr, char data)
// Write a byte to the specified I/O port
  { __asm
      { mov dx, addr
        mov al, data
        out dx, al
      }
  }

char inp(unsigned int addr)
// Read a byte from the specified I/O port
  { char result;
    __asm
      { mov dx, addr
        in al, dx
        mov result, al
      }
    return result;
```

```
}

void main(void)
{
    char tmp;

    outp(IOCON, 0x90);

    while(1)
    {
        tmp = inp(IOA);
        outp(IOB, tmp);
    }
}
```

2、实验板验证

a．通过 USB 线连接实验箱

b．按连接表连接电路

c．运行 PROTEUS 仿真，检查验证结果

七、 实验结果和体会

Proteus 和实验箱联调时候，要将实验箱开关全部向下（高电平），反过来 proteus 中开关全部高电平。

八、 建议

2.6 实验四 可编程定时/计数器 8253 实验

一、 实验要求

利用 8086 外接 8253 可编程定时/计数器，可以实现方波的产生。

二、 实验目的

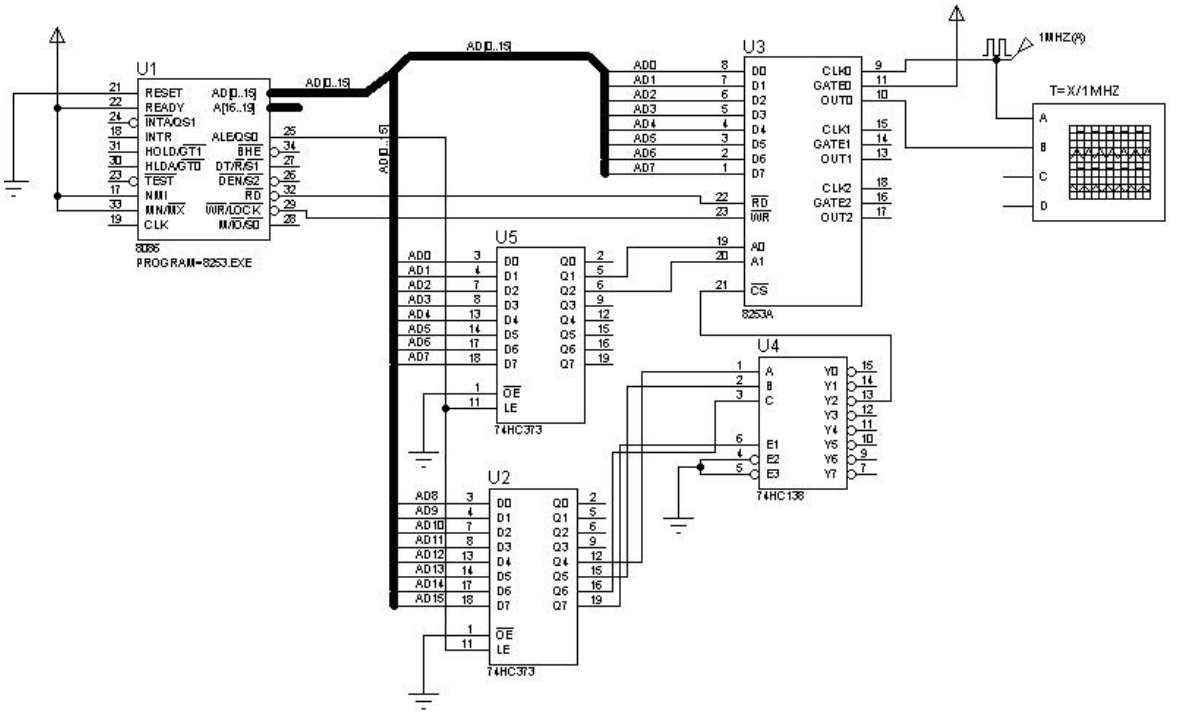
1、学习 8086 与 8253 的连接方法。

2、学习 8253 的控制方法。

3、掌握 8253 定时器/计数器的工作方式和编程原理

三、 实验电路及连线

1、Proteus 实验电路



2、硬件验证实验

硬件连接表

接线孔 1	接线孔 2
8253 CS	0A000H-0AFFFH
CLOCK_OUT	CLOUK_IN
1/4	CLK0
GATE0	+5V
OUT0	示波器

四、 实验说明

8253 芯片介绍

8253 是一种可编程定时/计数器，有三个十六位计数器，其计数频率范围为 0-2MHz， 用 +5V 单电源供电。

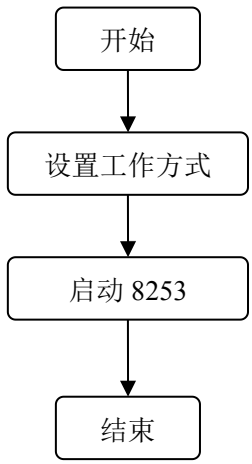
8253 的功能用途：

(1) 延时中断
(2) 可编程频率发生器
(3) 事件计数器
(4) 二进制倍频器
(5) 实时时钟
(6) 数字单稳
(7) 复杂的电机控制器

8253 的六种工作方式：

(1) 方式 0：计数结束中断
(2) 方式 l：可编程频率发生
(3) 方式 2：频率发生器
(4) 方式 3：方波频率发生器
(5) 方式 4：软件触发的选通信号
(6) 方式 5：硬件触发的选通信号

五、 实验程序流程图



六、 实验步骤

1、Proteus 仿真

a. 在 Proteus 中打开设计文档“8253_STM.DSN”;
b. 建立实验程序并编译，仿真;
c. 如不能正常工作，打开调试窗口进行调试。

汇编语言参考程序：

```
CODE      SEGMENT ;H8253.ASM
          ASSUME CS:CODE
START:    JMP TCONT
TCONTRO   EQU 0A06H
TCON0     EQU 0A00H
TCON1     EQU 0A02H
```

```
TCON2          EQU 0A04H
TCONT:         MOV DX,TCONTRO
               MOV AL,16H    ;计数器 0，只写计算值低 8 位，方式 3，二进制计数
               OUT DX,AL
               MOV DX,TCON0
               MOV AX,20     ;时钟为 1MHZ，计数时间=1us*20=20us，输出频率 50KHZ
               OUT DX,AL
               JMP $
CODE           ENDS
               END START
```

C 语言参考程序：

```
#define TCONTRO     0A006H
#define TCON0       0A000H
#define TCON1       0A002H
#define TCON2       0A004H

void outp(unsigned int addr, char data)
// Write a byte to the specified I/O port
  { __asm
      { mov dx, addr
        mov al, data
        out dx, al
      }
  }

char inp(unsigned int addr)
// Read a byte from the specified I/O port
  { char result;
    __asm
      { mov dx, addr
        in al, dx
        mov result, al
      }
    return result;
  }

void main(void)
{
     outp(TCONTRO,0x16);//计数器 0，只写计算值低 8 位，方式 3，二进制计数
     outp(TCON0,20);//时钟为 1MHZ ，计数时间=1us*20 =20 us   输出频率 50KHZ
     while(1){}
```

```
}
```

2、实验板验证

a．通过 USB 线连接实验箱

b．按连接表连接电路

c．运行 PROTEUS 仿真，检查验证结果

七、 实验结果和体会

八、 建议

2.7 实验二十二 外部中断实验（8259）

一、 实验要求

利用 8086 控制 8259 可编程中断控制器，实现对外部中断的响应和处理。要求程序中每次中断进行计数，并将计数结果用 8255 的 PA 口输出到发光二极管显示。

二、 实验目的

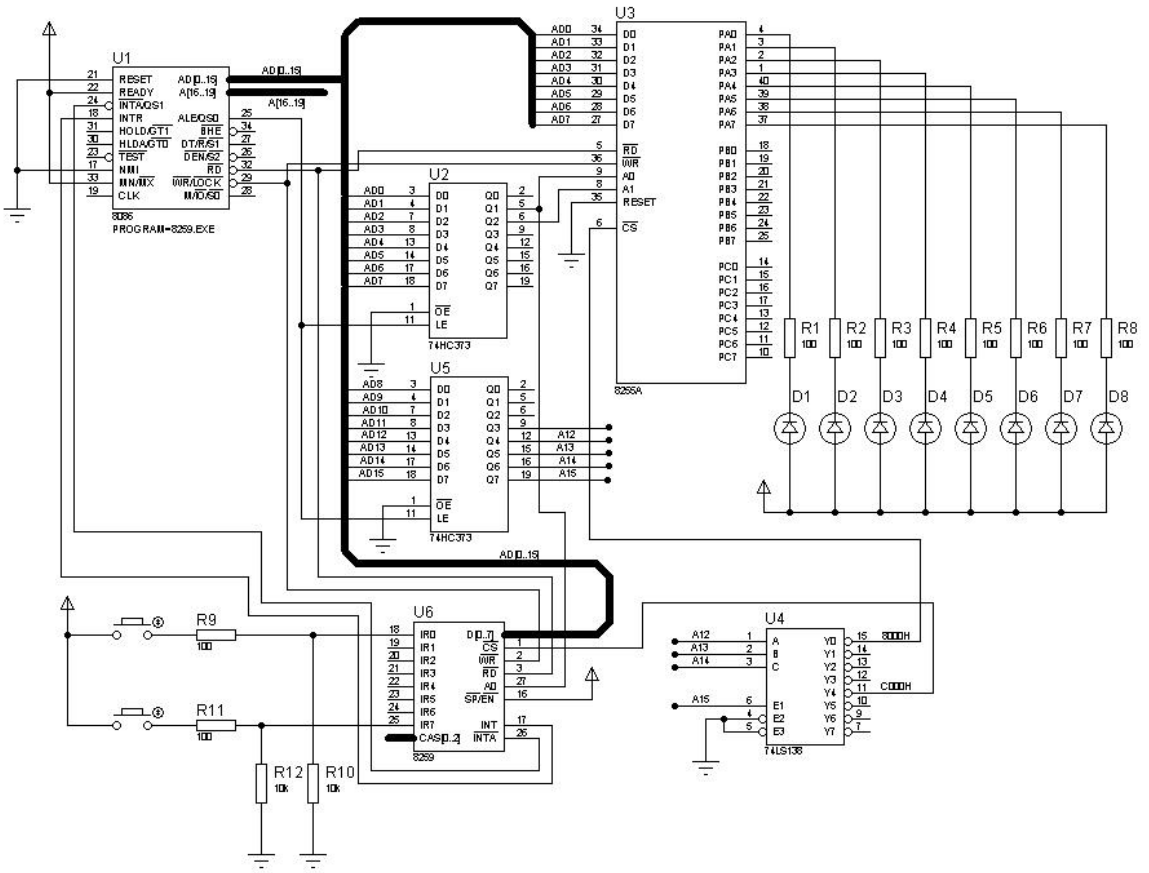
1、学习 8086 与 8259 的连接方法。

2、学习 8086 对 8259 的编程控制方法。

3、了解 8259 的多片级联。

三、 实验电路及连线

1、Proteus 实验电路



2、硬件验证实验

硬件连接表

接线孔 1	接线孔 2
8255 CS	08000H-08FFFH
8259 CS	0C00H-0CFFH
PA0—PA7	D1—D8
IR0	K1
IR7	K2
8959 $\overline{\text{INTA}}$	8086 $\overline{\text{INTA}}$
8259 INTR	8086 INTR

四、 实验说明

8086 的外部中断必须通过外接中断控制器才可以进行外部中断的处理。在编程时应注意：

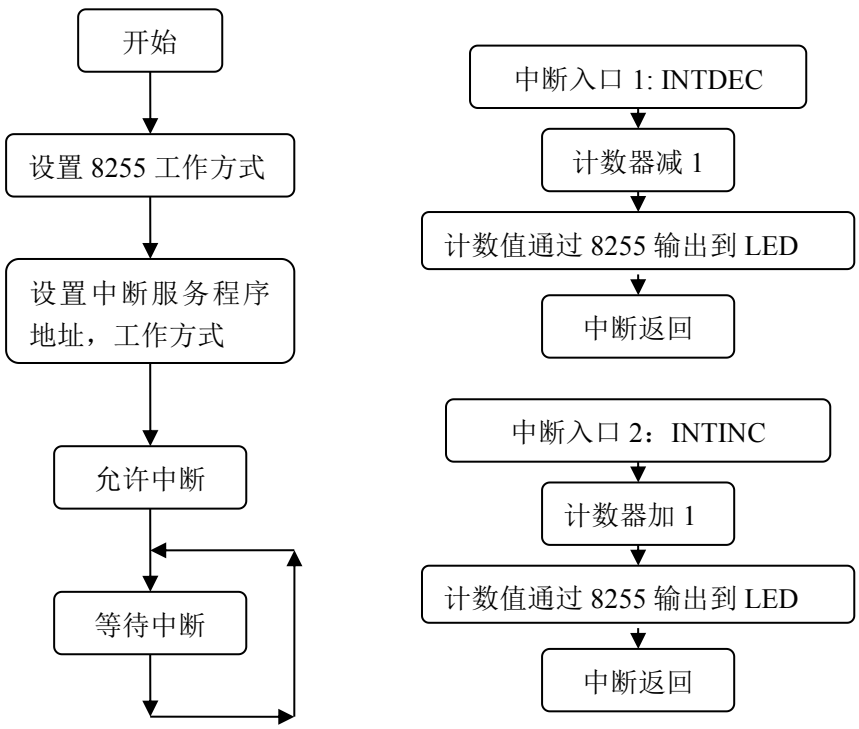
1、正确地设置可编程中断控制器的工作方式。

2、必须正确地设置中断向量表和中断服务程序的入口地址。

8259 可外接 8 个中断源，本实验响应 IR0 和 IR7 两个中断，在中断处理函数中，分别对计数器进行自增 1 和自减 1 运算，然后通过 8255 PA 口输出，由 LED 对计数结果进行显示。

将 K1、K2 信号接到 8259 的 IR0 和 IR7 脚。每次中断时，可以看到 LED 显示加 1 或减 1。

五、 实验程序流程图



六、 实验步骤

1、Proteus 仿真

a、Proteus 中打开设计文档 8259_STM.pdsprj ；

b、建立实验程序并编译，加载 hex 文件，仿真；

c、如不能正常工作，打开调试窗口进行调试。

汇编语言参考程序：

```
MODE       EQU     80H              ; 8255  工作方式
MODE       EQU     80H              ; 8255  工作方式
PA8255    EQU     8000H            ; 8255 PA 口输出地址
CTL8255 EQU     8006H

ICW1       EQU     00010011B        ; 单片 8259, 上升沿中断, 要写 ICW4
ICW2       EQU     00100000B        ; 中断号为 20H
ICW4       EQU     00000001B        ; 工作在 8086/88  方式
OCW1       EQU     00000000B        ; 只响应 INT0  中断
CS8259A EQU     0C000H           ; 8259 地址
CS8259B EQU     0C002H


CODE     SEGMENT
         ASSUME CS:CODE, DS: DATA,SS:STACK

ORG 800H

START:
          MOV AX, DATA
          MOV DS, AX

          MOV AX, STACK
          MOV SS, AX

          MOV AX, TOP
          MOV SP, AX

         MOV     DX, CTL8255
         MOV     AL, MODE
         OUT     DX, AL

         CLI
         PUSH    DS

         MOV     AX ,0
         MOV     DS ,AX
         MOV     BX, 128                  ;0X20 * 4    中断号
```

```
        MOV     AX, CODE
        MOV     CL, 4
        SHL     AX, CL                  ; X 16
        ADD     AX, OFFSET INTDEC       ; 中断入口地址（段地址为 0）
        MOV     [BX], AX

        MOV     AX, 0
        INC     BX
        INC     BX
        MOV     [BX], AX                ; 代码段地址为 0

        MOV     AX ,0
        MOV     DS ,AX
        MOV     BX, 156                 ;0X27 * 4   中断号

        MOV     AX, CODE
        MOV     CL, 4
        SHL     AX, CL                  ; X 16
        ADD     AX, OFFSET INTINC       ; 中断入口地址（段地址为 0）
        MOV     [BX], AX

        MOV     AX, 0
        INC     BX
        INC     BX
        MOV     [BX], AX                ; 代码段地址为 0

        POP DS
        CALL    IINIT

        MOV     AL, CNT                 ; 计数值初始为 0xFF,全灭
        MOV     DX, PA8255
        OUT     DX, AL
        STI
LP:                                     ; 等待中断，并计数。
        NOP
        JMP     LP
IINIT:
        MOV     DX, CS8259A
        MOV     AL, ICW1
        OUT     DX, AL
```

```
        MOV     DX, CS8259B
        MOV     AL, ICW2
        OUT     DX, AL

        MOV     AL, ICW4
        OUT     DX, AL

        MOV     AL, OCW1
        OUT     DX, AL
        RET

INTDEC:
        CLI
        MOV     DX, PA8255
        DEC     CNT
        MOV     AL, CNT
        OUT     DX, AL              ; 输出计数值

        MOV     DX, CS8259A
        MOV     AL, 20H             ; 中断服务程序结束指令
        OUT     DX, AL
        STI
        IRET

INTINC:
        CLI
        MOV     DX, PA8255
        INC     CNT
        MOV     AL, CNT
        OUT     DX, AL              ; 输出计数值

        MOV     DX, CS8259A
        MOV     AL, 20H             ; 中断服务程序结束指令
        OUT     DX, AL
        STI
        IRET

CODE    ENDS

DATA    SEGMENT
```

```
CNT       DB       0FFH
DATA      ENDS
STACK      SEGMENT 'STACK'
STA        DB    100 DUP(?)
TOP        EQU LENGTH STA
STACK      ENDS
          END START
```

无 C 语言参考程序。

2、实验板验证

a．通过 USB 线连接实验箱

b．按连接表连接电路

c．运行 PROTEUS 仿真，检查验证结果

七、 实验结果和体会

八、 建议

2.8 实验六 可编程串行通信控制器 8251A 实验

一、 实验要求

利用 8086 控制 8251A 可编程串行通信控制器，实现向 PC 机发送字符串“WINDWAY TECHNOLOGY!”。

二、 实验目的

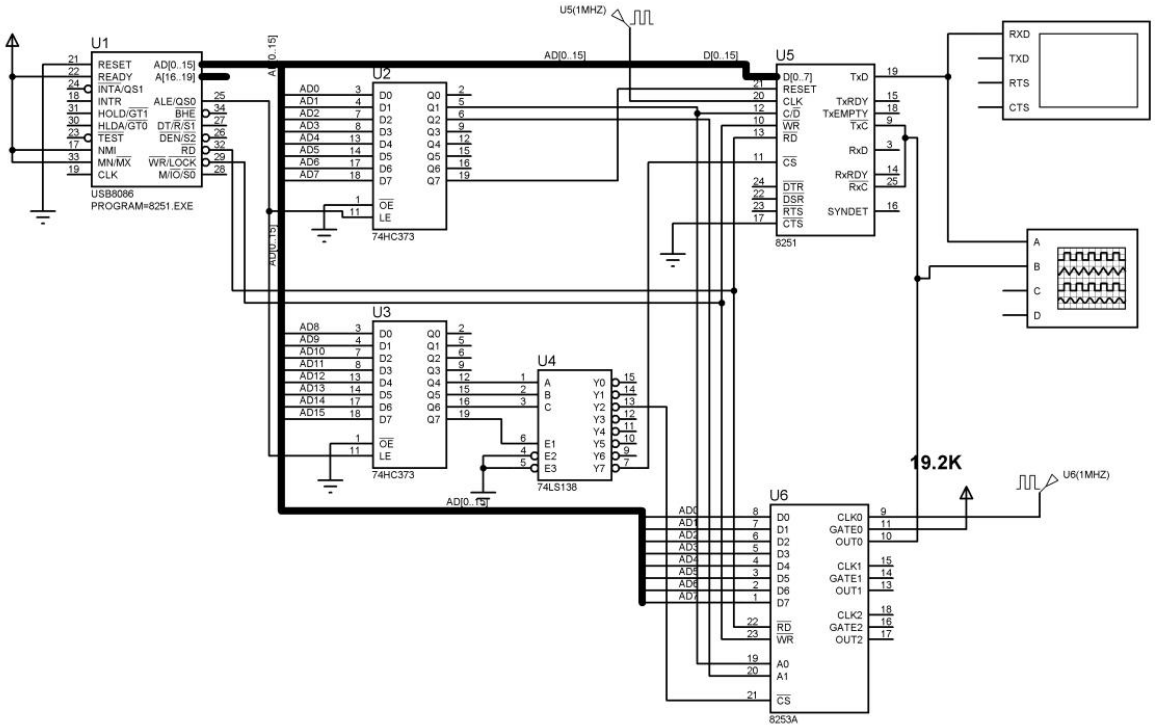
1、掌握 8086 实现串口通信的方法。

2、了解串行通讯的协议。

3、学习 8251A 程序编写方法。

三、 实验电路及连线

1、Proteus 实验电路



2、硬件验证实验

硬件连接表

接线孔 1	接线孔 2
8253 CS	A000H-AFFFH
8251 CS	F000H-FFFFH
分频 CLOCK_OUT	CLOUK_IN
分频 1/4	8253 CLK0

分频 1/4	8251 CLKM
8253 GATE0	+5V
8253 OUT0	8251 R/T_CLK
8251 TXD	TXD（RS-232 电路）

四、 实验说明

（1）8251 状态口地址：F002H，8251 数据口地址：F000H；

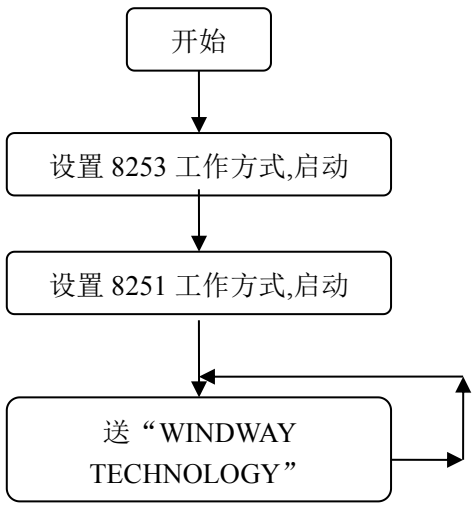
（2）8253 命令口地址：A006H，8253 计数器 0 口地址：A000H；

（3）通讯约定：异步方式，字符 8 位，一个起始位，一个停止位，波特率因子为 1，波特率为 19200；

（4）计算 T/RXC，收发时钟 fc，fc=1*19200=19.2K；

（5）8253 分频系数：计数时间=1us*50 =50 us 输出频率 20KHZ，当分频系数为 52 时，约为 19.2KHZ

五、 实验程序流程图



六、 实验步骤

1、Proteus 仿真

a. 在 Proteus 中打开设计文档“8251_STM.pdsprj”；

b. 建立实验程序并编译，仿真；

c. 如不能正常工作，打开调试窗口进行调试。

汇编语言参考程序：

```
CS8251R     EQU 0F080H    ; 串行通信控制器复位地址
CS8251D     EQU 0F000H    ; 串行通信控制器数据口地址
CS8251C     EQU 0F002H    ; 串行通信控制器控制口地址
TCONTRO     EQU 0A006H
TCON0       EQU 0A000H

CODE SEGMENT
```

```
                ASSUME DS:DATA,CS:CODE
START:
                MOV AX,DATA
                MOV DS,AX
                MOV DX,TCONTRO ;8253 初始化
                MOV AL,16H   ;计数器 0，只写计算值低 8 位，方式 3，二进制计数
                OUT DX,AL
                MOV DX,TCON0
                MOV AX,52   ;时钟为 1MHZ，计数时间=1us*50 =50 us  输出频率 20KHZ
                OUT DX,AL
                NOP
                NOP
                NOP
; 8251 初始化
                MOV     DX, CS8251R
                IN AL,DX
                NOP
                MOV     DX, CS8251R
                IN AL,DX
                NOP
                MOV     DX, CS8251C
                MOV     AL, 01001101b     ; 1 停止位,无校验,8 数据位, x1
                OUT     DX, AL
                MOV     AL, 00010101b     ; 清出错标志, 允许发送接收
                OUT     DX, AL
START4:MOV      CX,19
                LEA     DI,STR1
SEND:                   ; 串口发送' WINDWAY TECHNOLOGY   '
                MOV     DX, CS8251C
                MOV     AL, 00010101b     ; 清出错,允许发送接收
                OUT     DX, AL
WaitTXD:
                NOP
                NOP
                IN      AL, DX
                TEST    AL, 1                ; 发送缓冲是否为空
                JZ      WaitTXD
                MOV     AL, [DI]             ; 取要发送的字
                MOV     DX, CS8251D
                OUT     DX, AL               ; 发送
                PUSH    CX
```

```
                MOV     CX,8FH
                LOOP    $
                POP     CX
                INC DI
                LOOP SEND
                JMP START4
Receive:                        ; 串口接收
                MOV     DX, CS8251C
WaitRXD:
                IN      AL, DX
                TEST    AL, 2           ; 是否已收到一个字
                JE      WaitRXD
                MOV     DX, CS8251D
                IN      AL, DX          ; 读入
                MOV     BH, AL
                JMP START
CODE ENDS
DATA SEGMENT
STR1 db 'WINDWAY TECHNOLOGY!'
DATA ENDS
                END START
```

C 语言参考程序：

```
#define CS8251R    0F080h     // 串行通信控制器复位地址
#define CS8251D    0F000h     // 串行通信控制器数据口地址
#define CS8251C    0F002h     // 串行通信控制器控制口地址

#define TCONTRO      0A006H
#define TCON0        0A000H

unsigned char str[]="WINDWAY TECHNOLOGY!";
void outp(unsigned int addr, char data)
// Write a byte to the specified I/O port
  { __asm
      { mov dx, addr
        mov al, data
        out dx, al
      }
  }

char inp(unsigned int addr)
// Read a byte from the specified I/O port
```

```
 { char result;
    __asm
     { mov dx, addr
       in al, dx
       mov result, al
     }
   return result;
 }

void Send()
{
      unsigned char i=0;
      while(i<19){
      outp(CS8251C,0x15); //00010101b 清出错标志，允许发送接收
      while(inp(CS8251C)==0){}//发送缓冲是否为空
      outp(CS8251D,str[i++]);//发送
      }
}
char Receive()
{
      char a;
      while(inp(CS8251C)||0xfd){}//是否收到一个字节
      a=inp(CS8251D);//读入
      return a;
}
void main(void)
{
      char a;
      outp(TCONTRO,0x16);//8253 计数器 0，只写计算值低 8 位，方式 3，二进制计数
      outp(TCON0,52);  //时钟为 1MHZ ，计数时间=1us*50 =50 us  输出频率 20KHZ
      //以下为 8251 初始化
      a=inp(CS8251R);
      a=inp(CS8251R);
      outp(CS8251C,0x4d);//01001101b 1 停止位，无校验，8 数据，X1
      outp(CS8251C,0x15);//00010101b 清出错标志，允许发送接收
      while(1){
      Send();
      }
}
```

2、实验板验证

a．通过 USB 线连接实验箱

b．按连接表连接电路

c．运行 PROTEUS 仿真，检查验证结果

七、 实验结果和体会

八、 建议

2.9 实验七 D/A 数模转换实验(0832)

一、 实验要求

利用 DAC0832，编写程序生锯齿波、三角波、正弦波。用示波器观看。

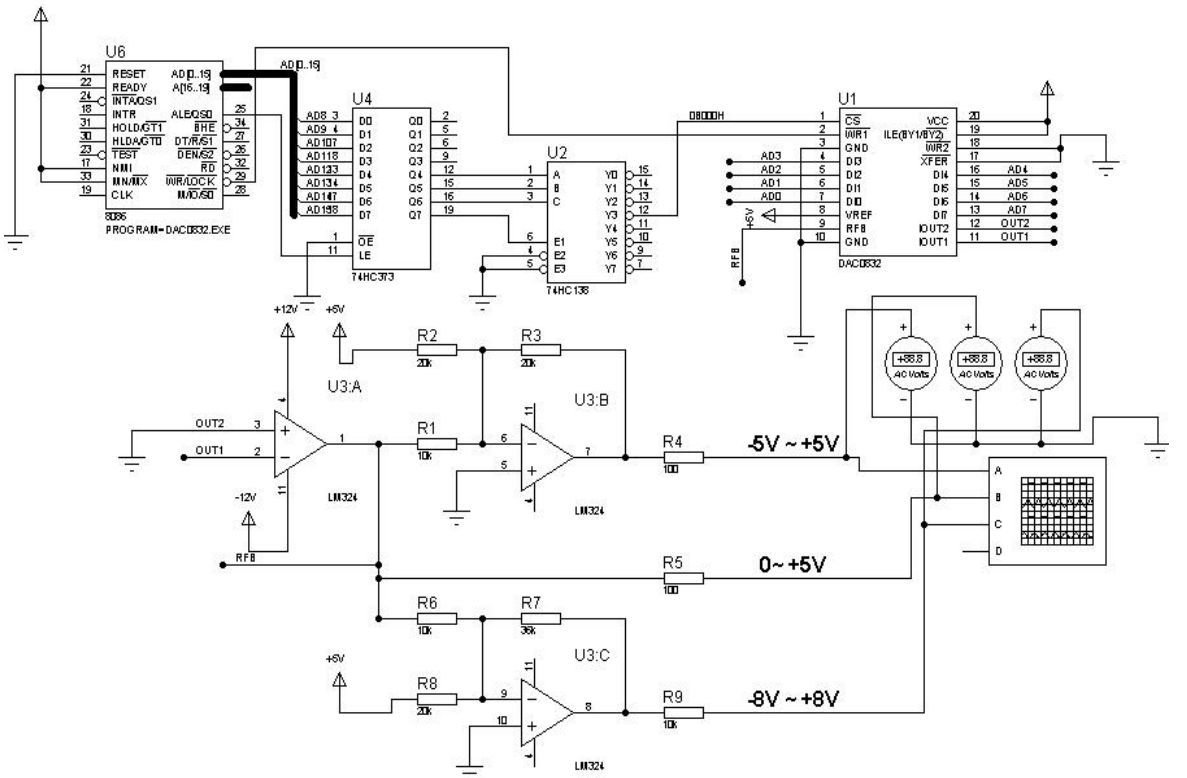
二、 实验目的

1、了解 D/A 转换的基本原理。

2、了解 D/A 转换芯片 0832 的编程方法。

三、 实验电路及连线

1、Proteus 实验电路图



2、硬件验证实验

硬件连接表

接线孔 1	接线孔 2
0832 CS	B000H-BFFFH
0-+5V	示波器
-5V-+5V	示波器
-8V-+8V	示波器

四、 实验说明

1、主要知识点概述：

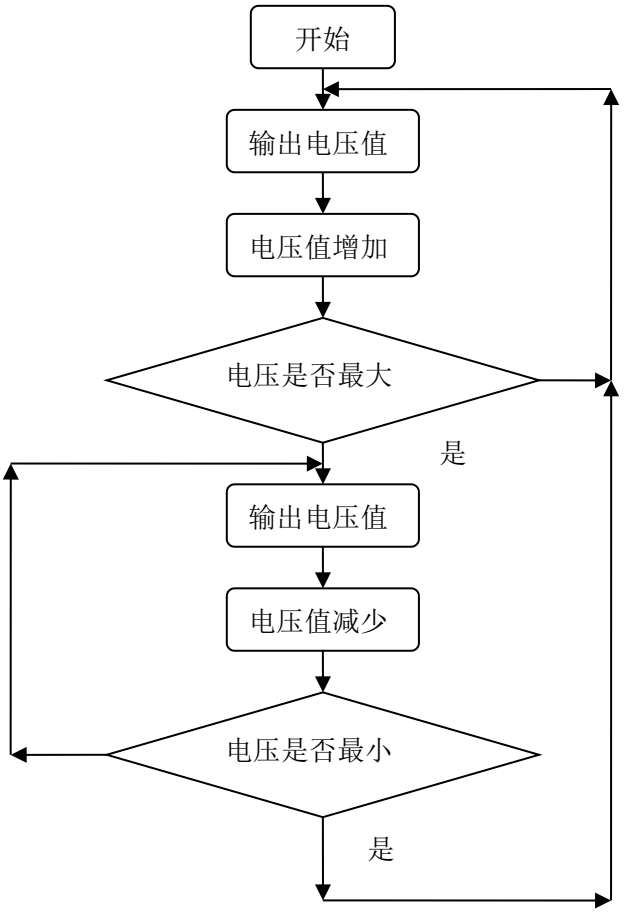
本实验用到的主要知识点是：DAC0832 的工作原理。

DAC0832 是采用先进的 CMOS 工艺制成的单片电流输出型 8 位 D/A 转换器。它采用的是 R-2R 电阻梯级网络进行 DA 转换。电平接口与 TTL 兼容。具有两级缓存。

2、实验效果说明：

通过电压表测量 DAC 转换出来的电压值

五、 实验程序流程图



六、 实验步骤

1、Proteus 仿真

a.在 Proteus 中打开设计文档　DAC0832_STM.pdsprj；

b.建立实验程序并编译，仿真；

c.如不能正常工作，打开调试窗口进行调试。

汇编语言参考程序：

```
CODE      SEGMENT
ASSUME CS:CODE
IOCON   EQU 0B000H
```

```
START:
                MOV AL,00H
                MOV DX,IOCON
OUTUP:          OUT DX,AL
                INC AL
                CMP AL,0FFH
                JE OUTDOWN
                JMP OUTUP

OUTDOWN:
DEC AL
                OUT DX,AL
                CMP AL,00H
                JE OUTUP
                JMP OUTDOWN

CODE ENDS
                END START
```

C 语言参考程序：

```
#define IOCON      0B006H

void outp(unsigned int addr, char data)
// Write a byte to the specified I/O port
  { __asm
      { mov dx, addr
        mov al, data
        out dx, al
      }
  }

char inp(unsigned int addr)
// Read a byte from the specified I/O port
  { char result;
    __asm
      { mov dx, addr
        in al, dx
        mov result, al
      }
    return result;
  }
```

```
void main(void)
{
    unsigned char tmp;
    tmp=0;
    while(1){
        while(tmp<0xff){
            outp(IOCON,tmp++);
        }
        while(tmp>0){
            outp(IOCON,tmp--);
        }
    }
}
```

2、实验板验证

a．通过 USB 线连接实验箱

b．按连接表连接电路

c．运行 PROTEUS 仿真，检查验证结果

七、 实验结果和体会

八、 建议

2.10 实验八 A/D 模数转换实验(0809)

一、 实验要求

利用实验箱上的 ADC0809 做 A/D 转换，实验箱上的电位器提供模拟量的输入，编写程序，将模拟量转换成二进制数据，用 74HC373 输出到发光二极管显示。

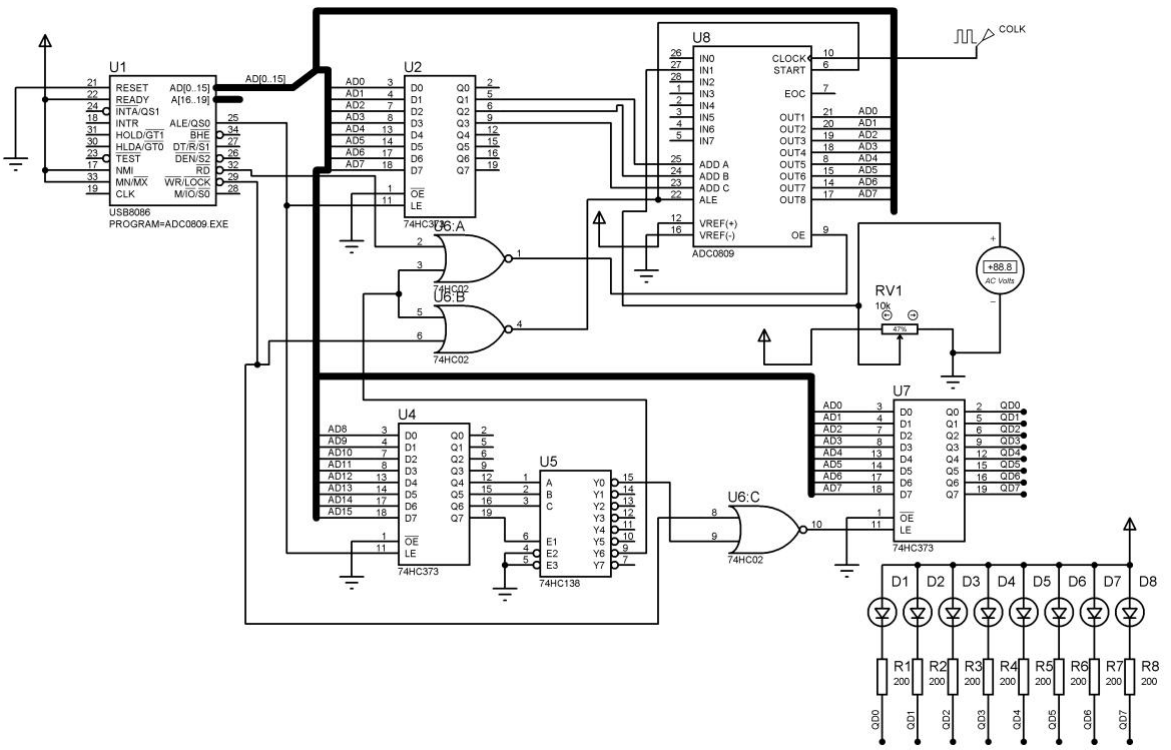
二、 实验目的

1、掌握 A/D 转换的连接方法。

2、了解 A/D 转换芯片 0809 的编程方法。

三、 实验电路及连线

1、Proteus 实验电路图



2、硬件验证实验

硬件连接表

接线孔 1	接线孔 2
0809_CS	E000H-EFFFH
373_CS	8000H-8FFFH
CLOCK_OUT	CLOCK_IN
分频 1/4	CLK
IN1	AD_IN

Q0--Q7	D1--D8

四、 实验说明

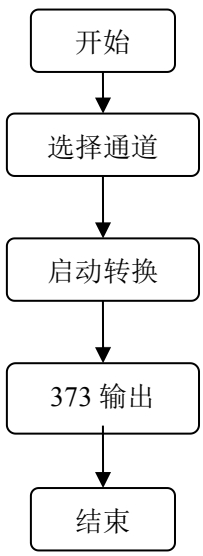
1、主要知识点概述：

A/D 转换器大致有三类：一是双积分 A/D 转换器，优点是精度高，抗干扰性好，价格便宜，但速度慢；二是逐次逼近 A/D 转换器，精度、速度、价格适中；三是并行 A/D 转换器，速度快，价格也昂贵。

2、实验效果说明：

实验用的 ADC0809 属第二类，是 8 位 A/D 转换器，每采集一次一般需100 μs 。本实验可采用延时方式或查询方式读入 A/D 转换结果，也可以采用中断方式读入结果，在中断方式下，A/D 转换结束后会自动产生 EOC 信号，将其与 CPU 的外部中断相接。调整电位计，得到不同的电压值，转换后的数据通过发光二级管输出

五、 实验程序流程图



六、 实验步骤

1、Proteus 仿真

a. 在 Proteus 中打开设计文档“ADC0809_STM.pdsprj”;

b. 建立实验程序并编译，仿真;

c. 如不能正常工作，打开调试窗口进行调试。

汇编语言参考程序：

```
CODE      SEGMENT
ASSUME CS:CODE
AD0809        EQU 0E002H
OUT373        EQU 8000H

START:
              MOV DX,8006H
```

```
                MOV AL,80H
                OUT DX,AL
START1:
                MOV AL,00H
                MOV DX,AD0809
                OUT DX,AL
                NOP
                IN   AL,DX

                MOV CX,10H
                LOOP   $

                MOV DX,OUT373
                OUT DX,AL
                JMP START1
CODE ENDS
                END START
```

C 语言参考程序：

```
#define AD0809     0E002H
#define OUT373     8000H

void outp(unsigned int addr, char data)
// Write a byte to the specified I/O port
  { __asm
      { mov dx, addr
        mov al, data
        out dx, al
      }
  }

char inp(unsigned int addr)
// Read a byte from the specified I/O port
  { char result;
    __asm
      { mov dx, addr
        in al, dx
        mov result, al
      }
    return result;
  }
```

```
void main(void)
{
    char i,in;
    while(1){
        for(i=10;i>0;i--){
          outp(AD0809,0);
              in=inp(AD0809);
        }
        outp(OUT373,in);
    }
}
```

2、实验板验证

a．通过 USB 线连接实验箱

b．按连接表连接电路

c．运行 PROTEUS 仿真，检查验证结果

七、 实验结果和体会

八、 建议