目 录

第1章 8086 教学实验系统简介

1.1 简介

PROTEUS 教学实验系统（8086）是我公司针对微机原理与接口技术课程的教学需求所研发的，其目的在于激发学生学习 8086 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 是本实验箱的电路设计、电路仿真与调试、程序编译的环境，需要另外购买。PROTEUS 教学实验系统（8086）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。

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

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

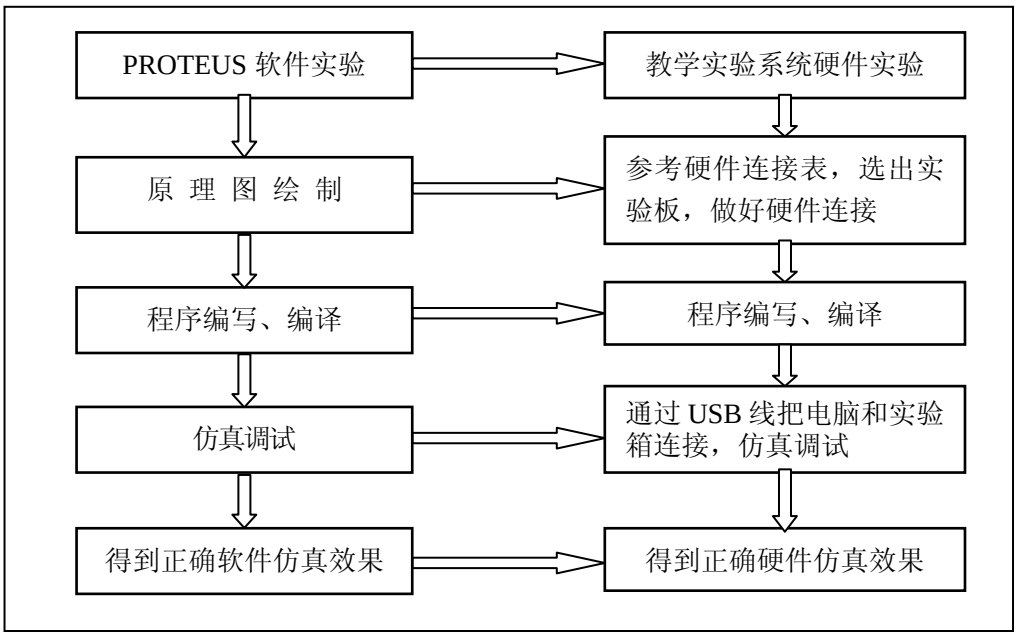
可级联信号连接线 20 根

1.3 配套资料

1、 PROTEUS 教学实验系统（8086）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

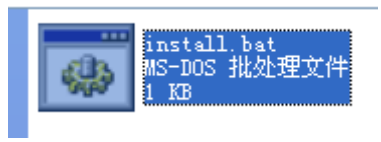
大部分实验的开展，我们都采用在 PROTEUS 平台下的交互式仿真，使用硬件平台与电脑软件仿真同时进行的方法，实验的开展流程如下：



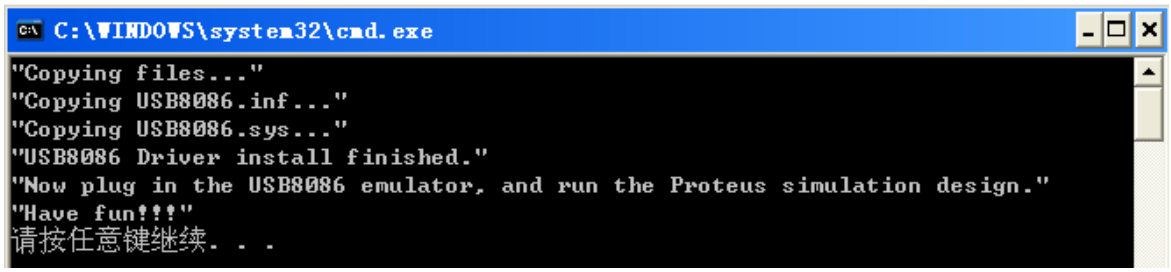
在进行硬件实验中，有几点需要注意：
1、尽量保持线束的整齐，对于控制线少交叉缠绕。
2、拔线时请逐根拔除，切忌强行硬拔整股连线（易造成整股损坏）。
3、液晶类实验涉及到液晶对比度的调节，请通过邻近电位器来调整。

1.5 驱动安装

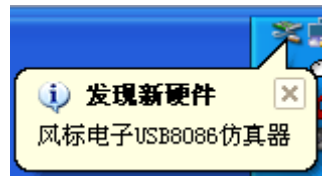
1. 在光盘的 Driver 目录下，找到 install.bat，安装驱动和仿真模型。



2. 安装完成后，显示如下，如果失败，请检查 Proteus 是否安装在默认的目录，如果不是，则请手动把 QtCore4.dll 和 USB8086.dll 复制到 Proteus 安装目录的 bin 文件夹下。



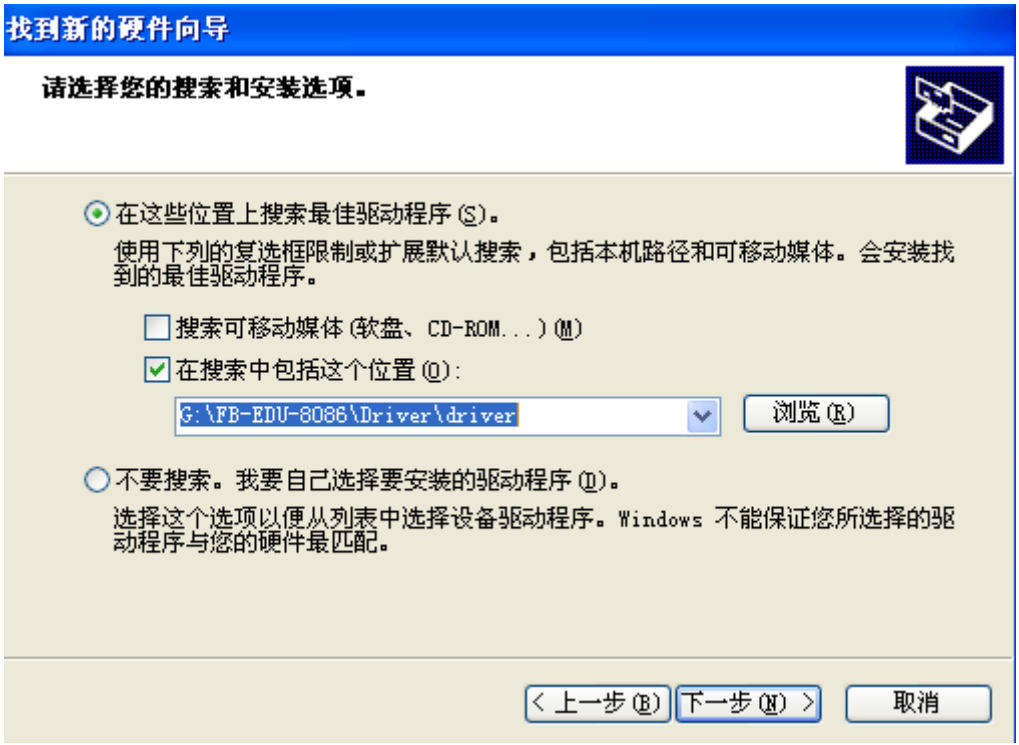
3. 通过 USB 线，把实验箱上的仿真器与电脑相连接，出现“发现新硬件 风标电子 USB8086 仿真器”的提示，如下图所示：



4. 自动弹出找到新的硬件向导，选择“从列表或指定位置安装（高级）（S），如下图所示：

找到新的硬件向导

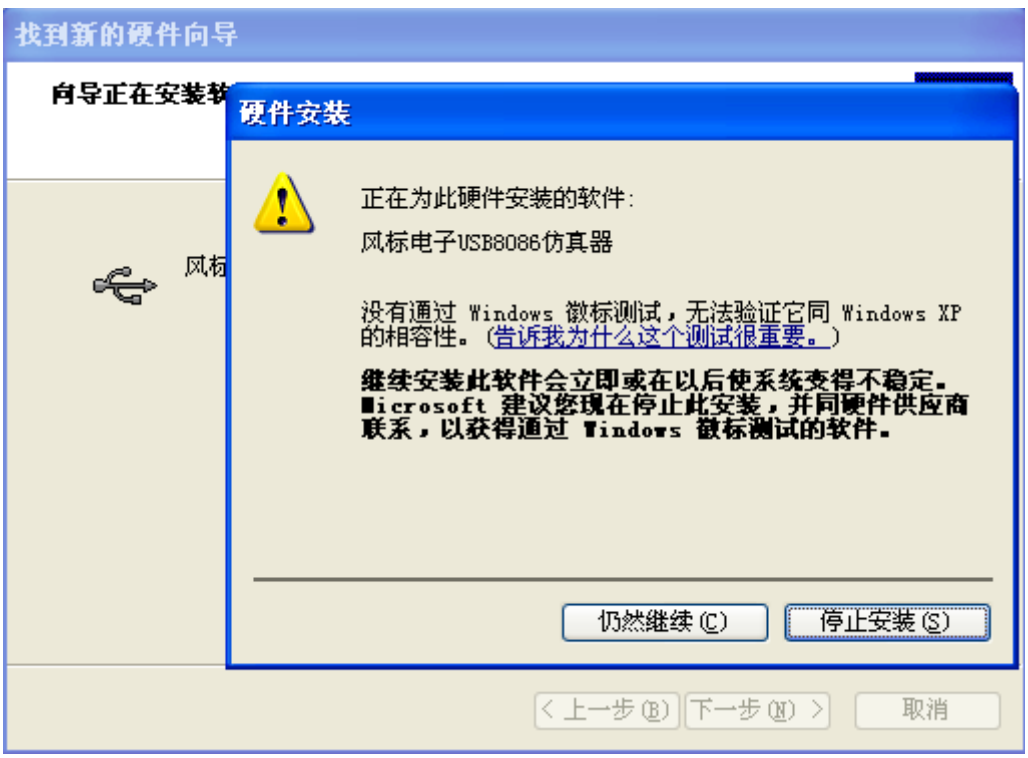
选择“在这些位置上搜索最佳驱动程序”，打勾“在搜索中包括这个位置（D）”，浏览到光盘目录下的 Driver 文件夹，如下图所示：



指定搜索文件夹


找到新的硬件向导
向导正在搜索，请稍候...
风标电子USB8086仿真器
< 上一步(B)
下一步(N) >
取消

搜索驱动


找到新的硬件向导
硬件安装
正在为此硬件安装的软件:
风标电子USB8086仿真器
没有通过 Windows 徽标测试，无法验证它同 Windows XP 的相容性。(告诉我为什么这个测试很重要。)
继续安装此软件会立即或在以后使系统变得不稳定。Microsoft 建议您现在停止此安装，并同硬件供应商联系，以获得通过 Windows 徽标测试的软件。
仍然继续(C)
停止安装(S)
< 上一步(B)
下一步(N) >
取消

选择仍然继续


找到新的硬件向导
完成找到新硬件向导
该向导已经完成了下列设备的软件安装:
风标电子USB8086仿真器
要关闭向导，请单击“完成”。
< 上一步(B)
完成
取消

完成驱动安装

第2章 8086 软件部分实验目录

2.1 系统环境配置与熟悉

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。在相应的 Projects(汇编)和 C_Projects(C 语言)目录下可以找到 Tools 目录，里面就有所需要的编译工具。其中 MASM 的版本是 6.14.8444，Digital Mars C Compiler 的版本是 8.42n。本实验就是让大家学会怎样在 Proteus 中调用外部的编译器进行编译，生成可执行文件.exe。

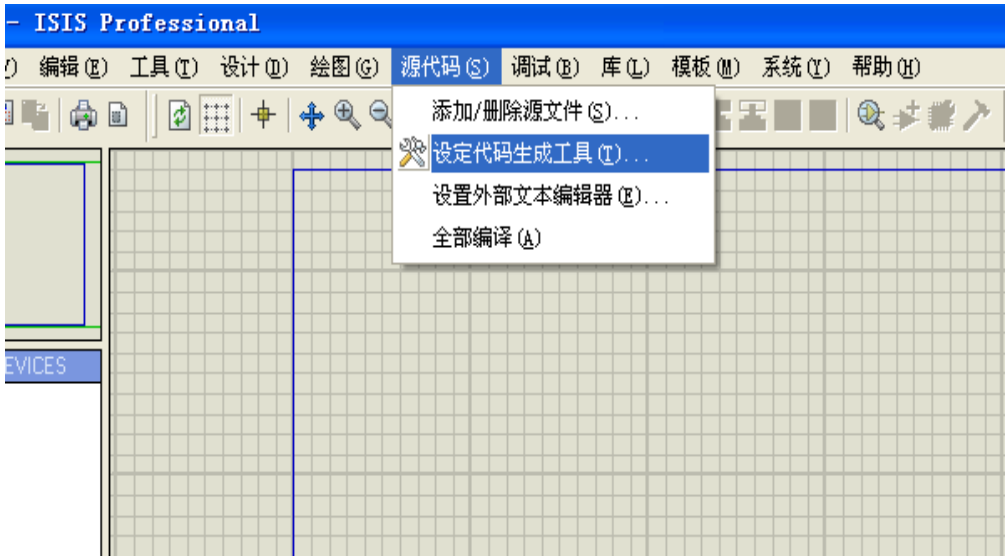
二、 实验目的

1、掌握 PROTEUS 调用外部编译器；
2、熟悉 PROTEUS 的程序编写环境。

三、 系统环境配置

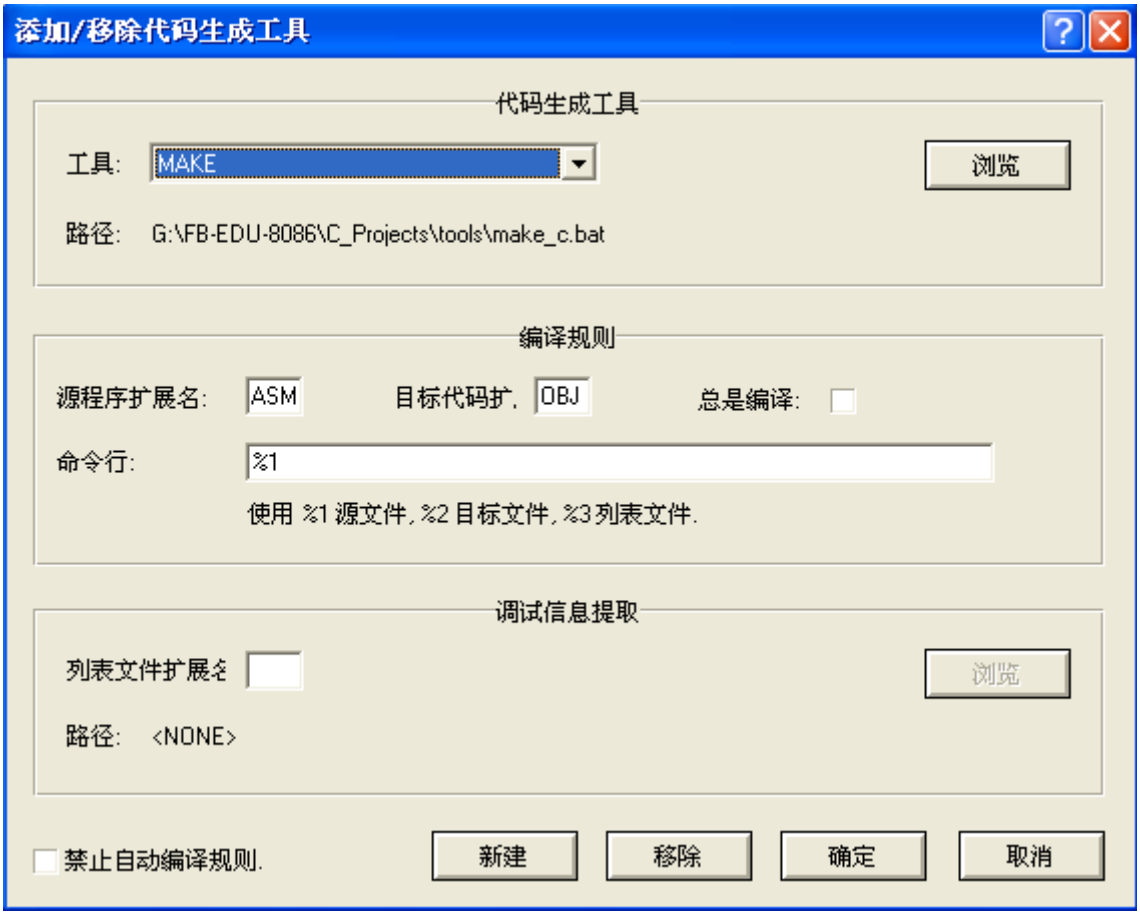
1、**PROTEUS 配置 8086 汇编编译工具**

首先，打开 PROTEUS 下的“源代码->设定代码生成工具”。



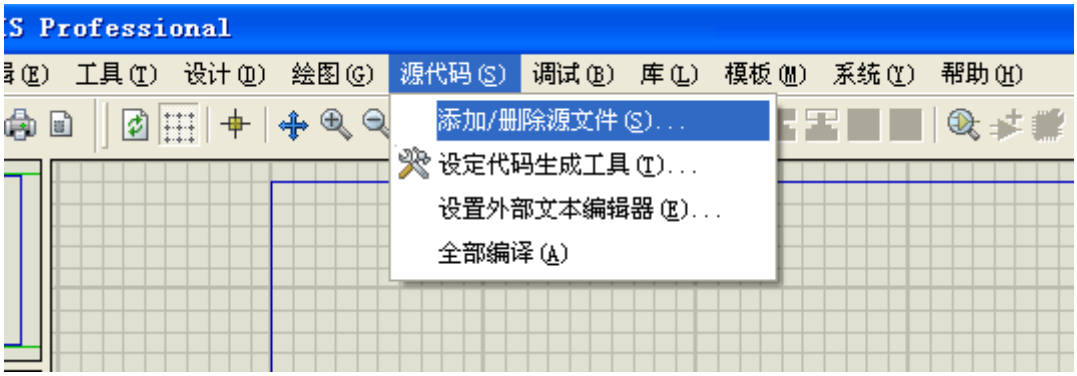
其次，在出现的对话框中点击新建，选择实验箱配送光盘下 Projects\tools 目录下的 make.bat 文件，然后在源程序扩展名下写入 ASM，目标代码扩展名写入

EXE，最后，点击确定配置完成，如下图所示：

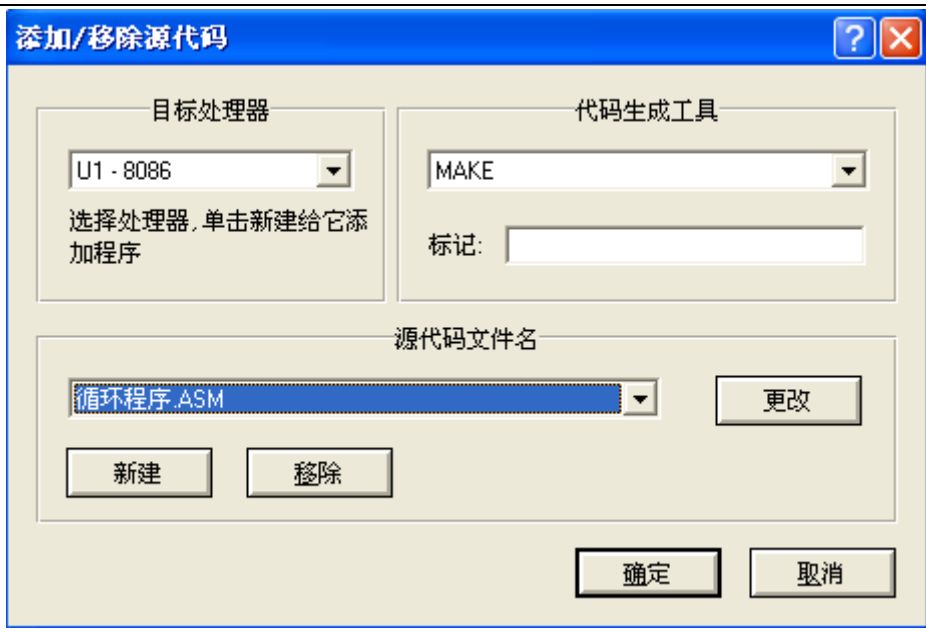


2、**编译 8086 汇编文件**

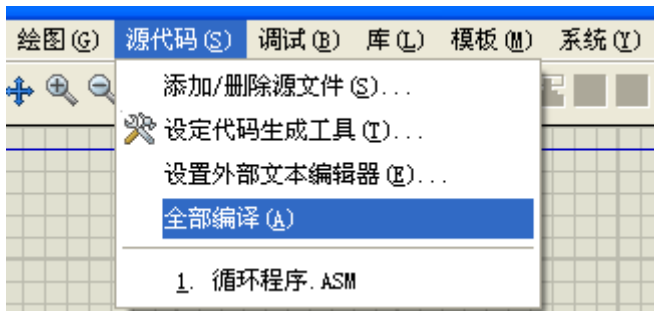
打开 PROTEUS 下的　源代码->添加/删除源文件



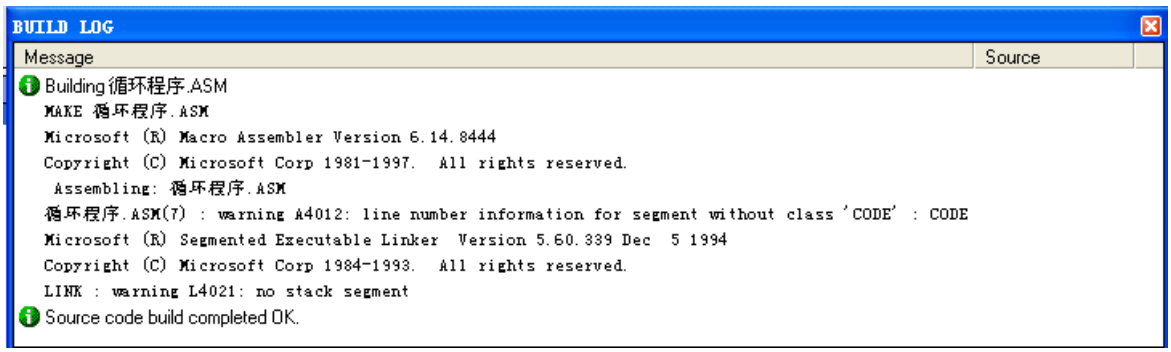
在出现的对话框中点击新建，加入之前做好的后缀为.ASM 的汇编文件，再选择代码生成工具，找到建好的 8086 汇编生成工具 MAKE。最后点确定。



编译代码操作如下



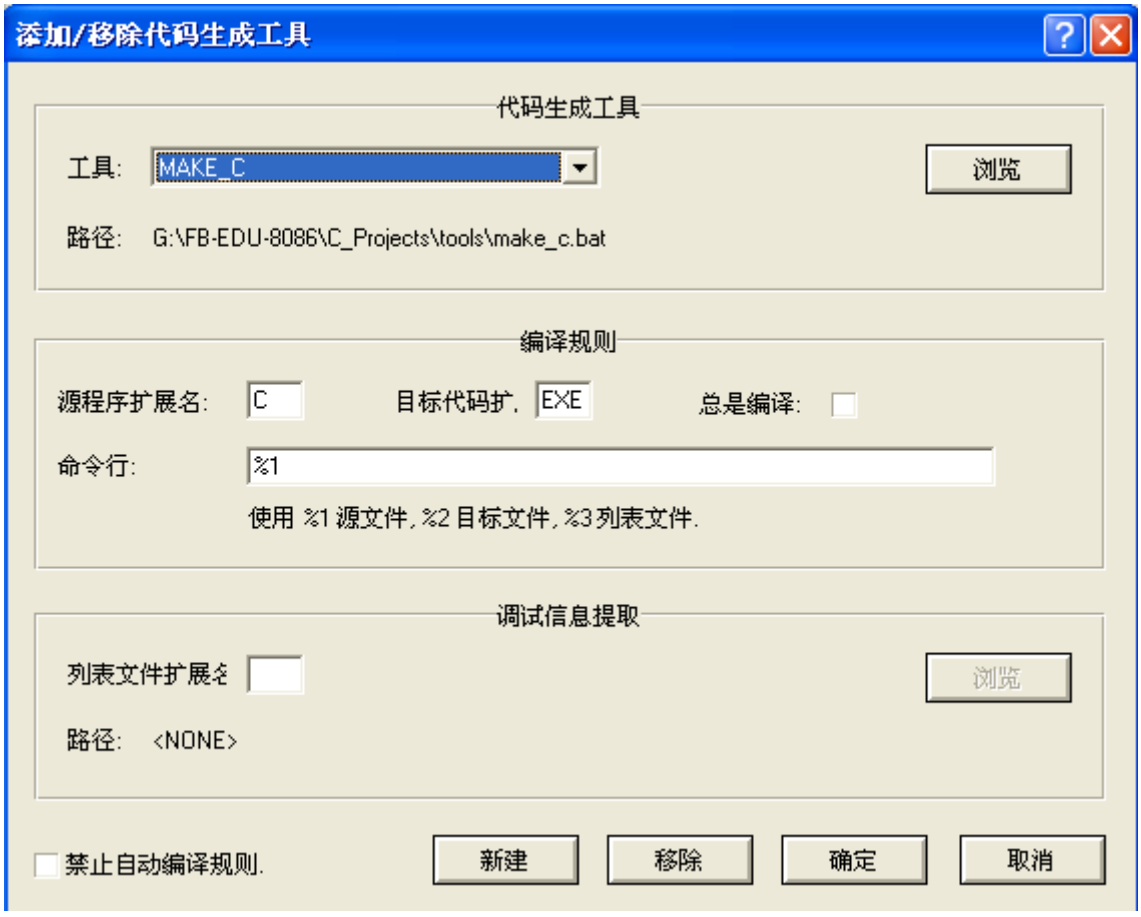
编译结果：



3、PROTEUS 配置 8086 C 编译工具

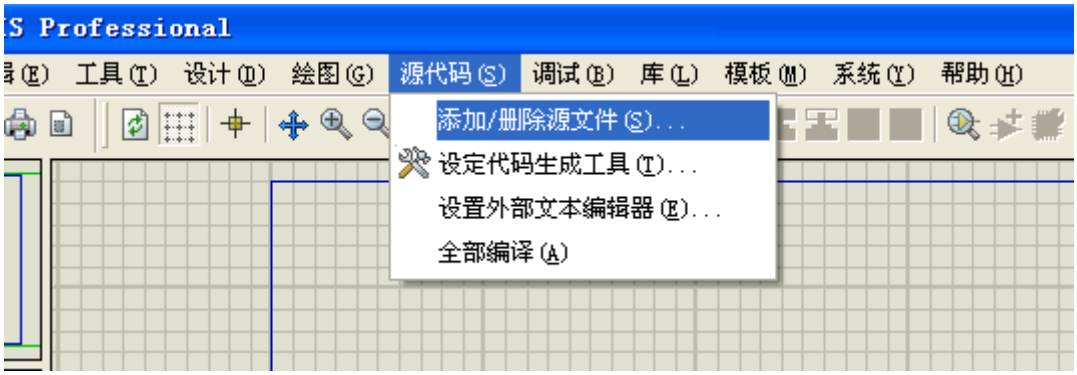
使用 Digital Mars C Compiler 编译 C 文件的设置过程也是类似的。首先，打开 PROTEUS 下的“源代码->设定代码生成工具”菜单。

其次，在出现的对话框中点击新建，选择实验箱配送光盘下\C_Projects\tools 下的 make_c.bat 文件，在源程序扩展名下写入 C，目标代码扩展名写入 EXE，最后，点击确定配置完成，如下图所示：



4、**编译 8086 C 文件**

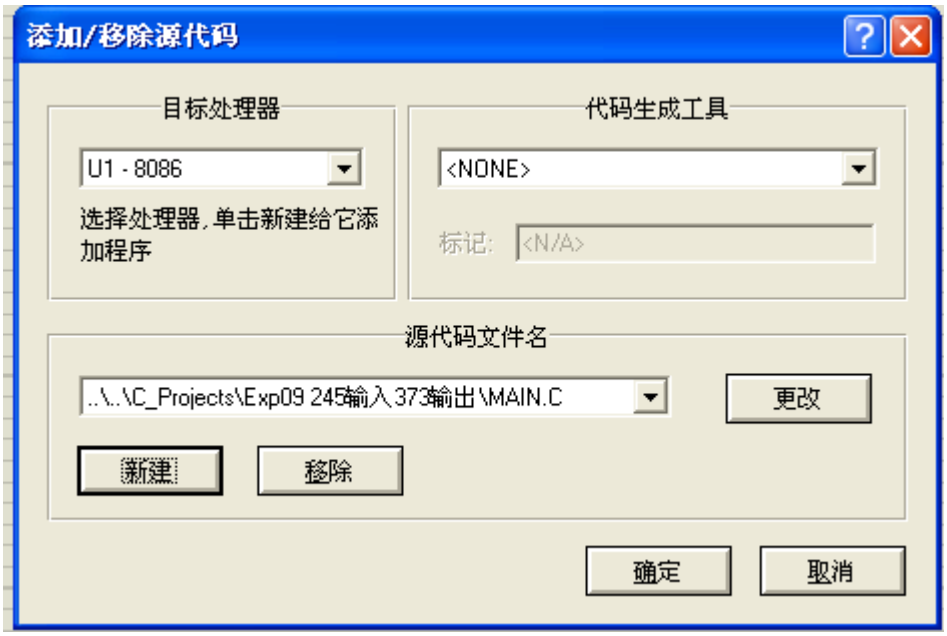
打开 PROTEUS 下的　源代码->添加/删除源文件



在出现的对话框中点击新建，加入之前做好的后缀为.C 的 C 文件，再选择代

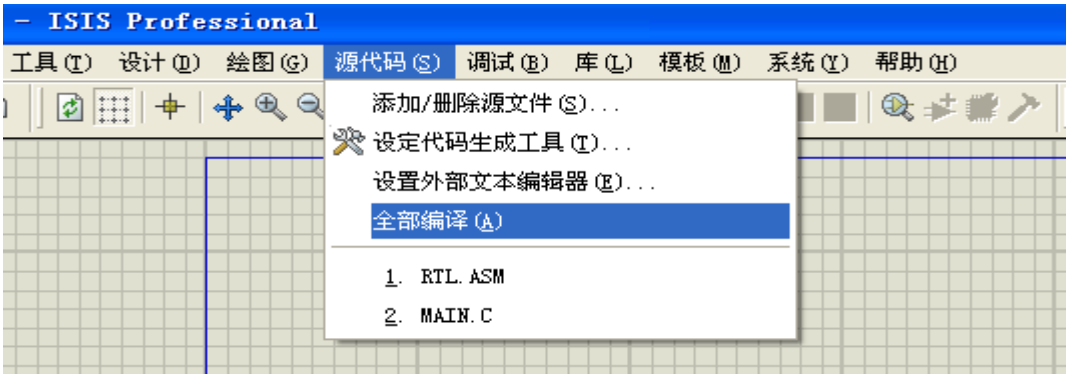
码生成工具，找到建好的 8086 汇编生成工具 MAKE_C,其中和汇编不同的是，这里还要加入一个汇编启动文件，但代码生成工具则为空。（加入的汇编启动文件为 RTL.ASM 如下图）

先加入 C 文件：

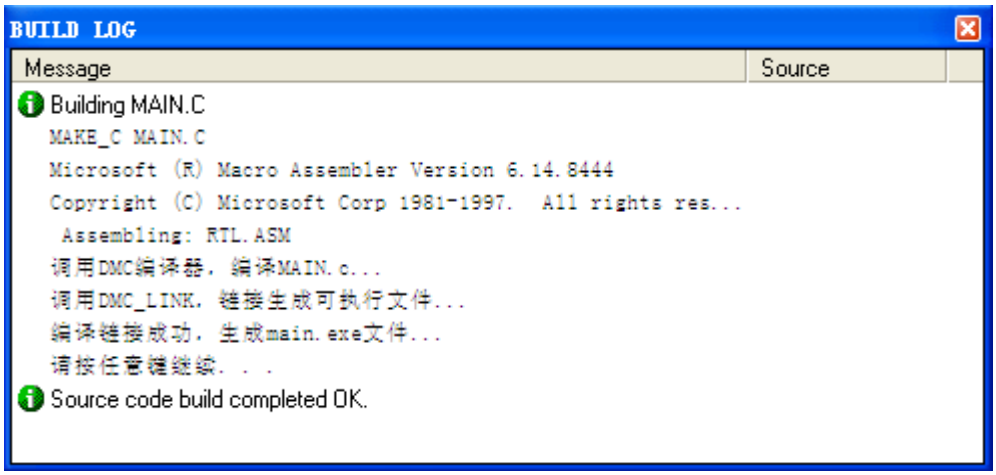


再加入 ASM 启动文件：

编译代码操作如下



编译结果：



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。

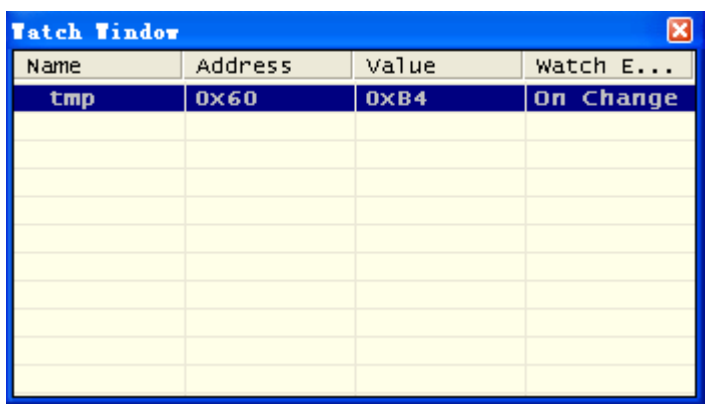
第三栏是代码调试菜单，有单步、连续单步，跳进/跳出函数，跳到光标处等功能。

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。

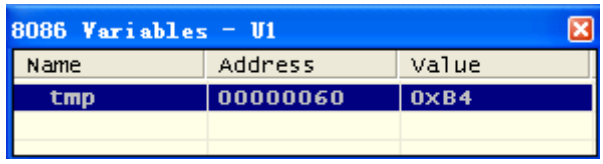
其中观察窗口可以添加变量进行观察，并且可以设置条件断点。这在调试程序的时候非常有用。



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。

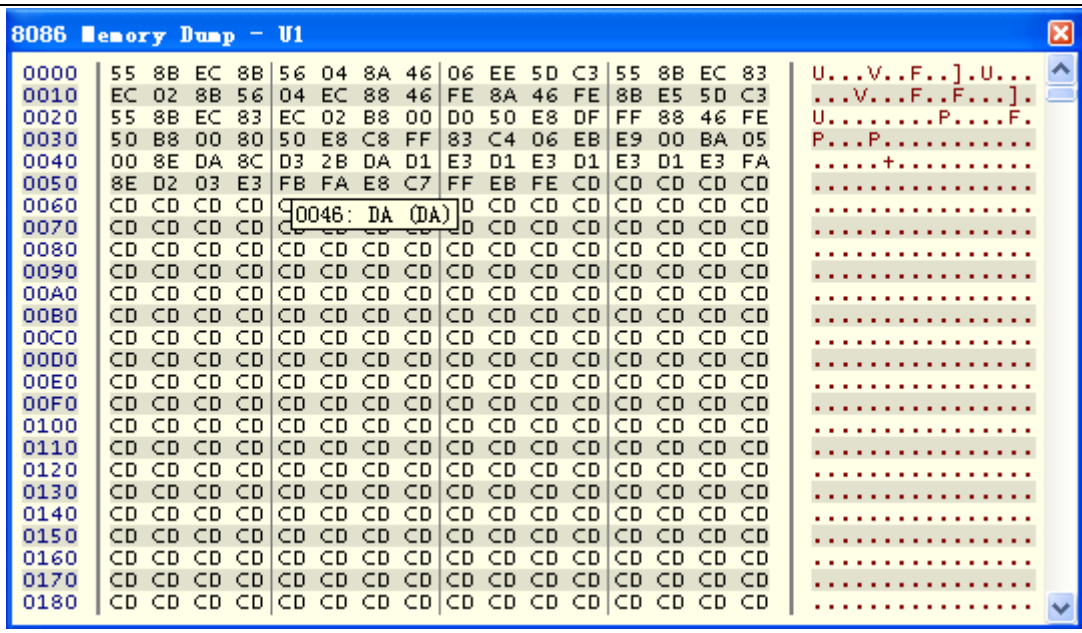


变量窗口

寄存器窗口实时显示 8086 各个寄存器的值。

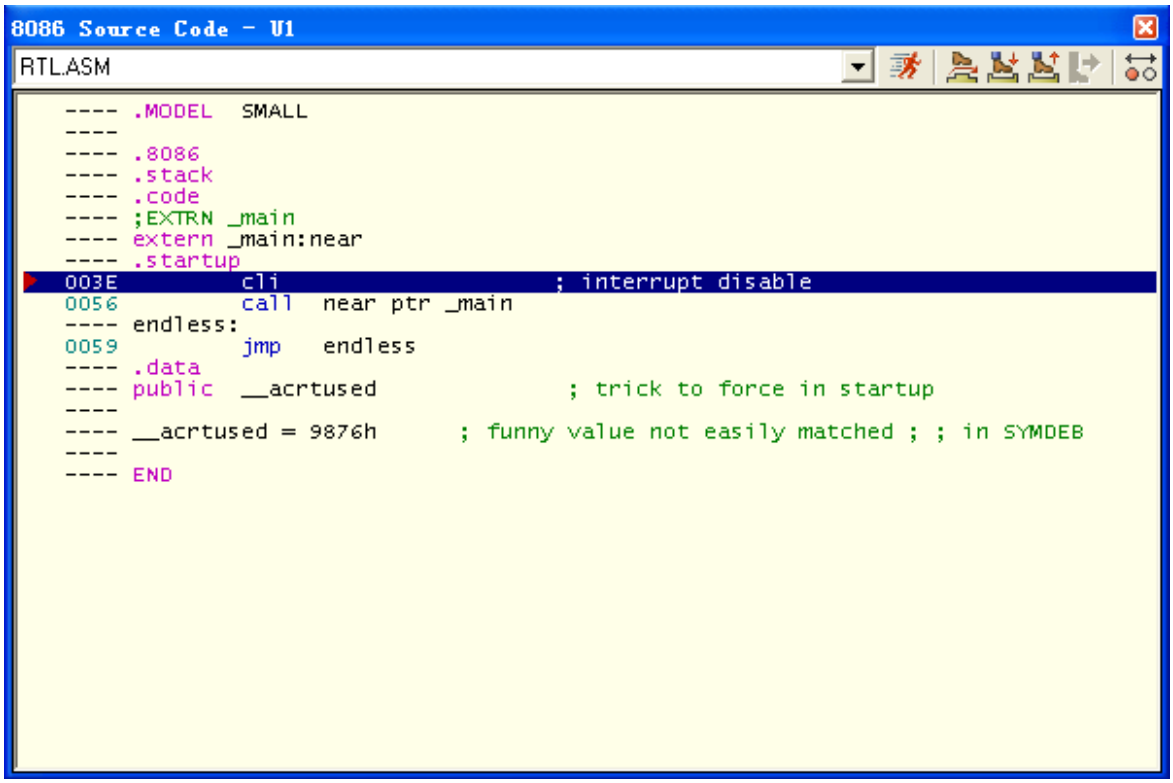
寄存器窗口

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 多位十六进制加法运算实验

一、 实验要求

利用 PROTEUS 平台，建立 8086 的多位十六进制加法运算的例子。

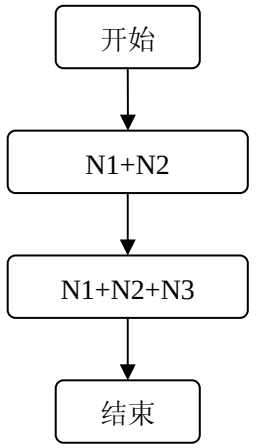
二、 实验目的

1、熟悉实验系统的编程和使用。
2、掌握使用加法类运算指令编程及调试方法。
3、掌握加法类指令对状态标志位的影响。

三、 实验说明

由于本实验是三个十六位二制数相加运算，N4 为存放结果，其中 N1 为 1111H、N2 为 2222H、N3 为 3333H 所以结果应该为 6666H

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“多位十六进制加法运算.DSN”;
b. 单步运行，打开调试窗口进行调试。
参考程序：

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
BEG:    MOV AX,DATA
        MOV DS,AX
        MOV SI,OFFSET NUM1
        MOV AX,0
```

```
                ADD AX,[SI+0]
                ADD AX,[SI+2]
                ADD AX,[SI+4]
                MOV [SI+6],AX
                JMP $
CODE ENDS
DATA SEGMENT
NUM1 DW 1111H ;N1
NUM2 DW 2222H ;N2
NUM3 DW 3333H ;N3
NUM4 DW 0000H ;N4
DATA ENDS
                END BEG
```

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．检查验证结果。

六、 实验结果和体会

七、 建议

2.4 实验二 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。

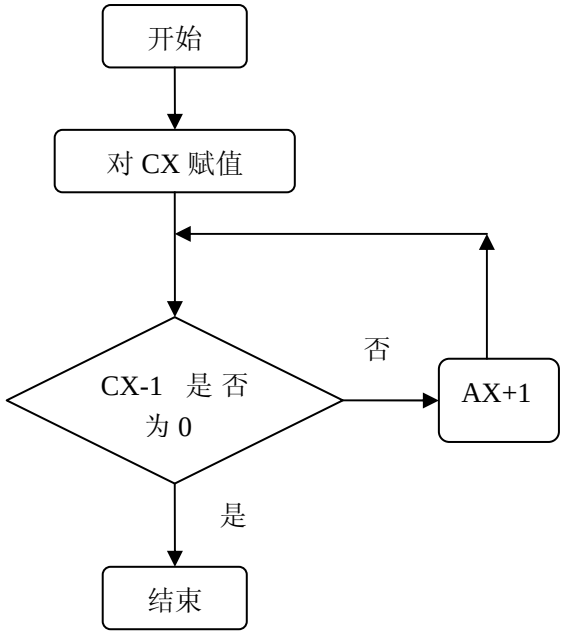
二、 实验目的

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.DSN”;
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

2.5 实验三 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。

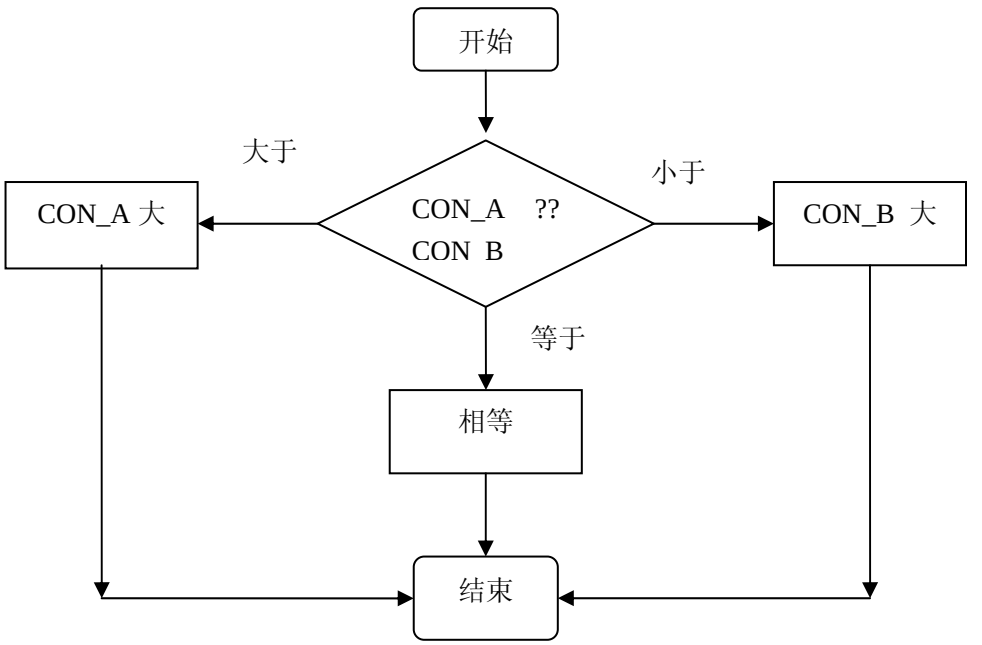
二、 实验目的

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.DSN”；
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

2.6 实验四 内存块移动实验

一、 实验要求

利用 PROTEUS 平台，建立 8086 的内存块移动的例子。

二、 实验目的

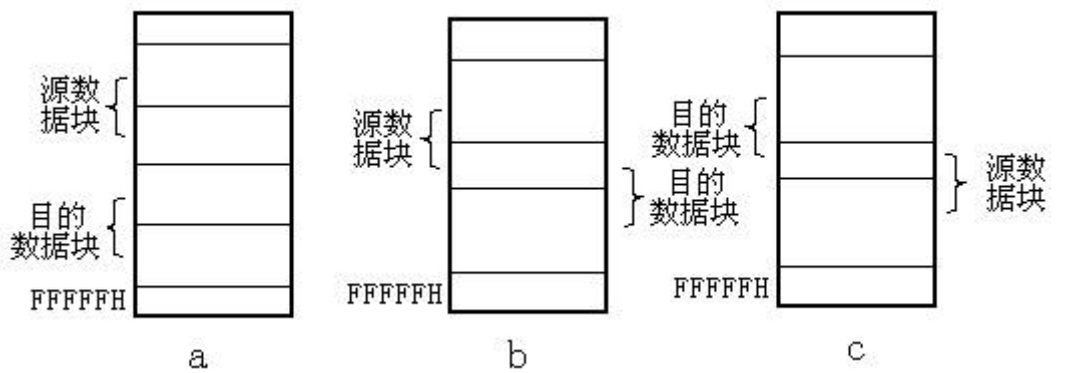
1、熟悉实验系统的编程和使用。

2、了解内存的移动方法。

3、加深对存储器读写的认识。

三、 实验说明

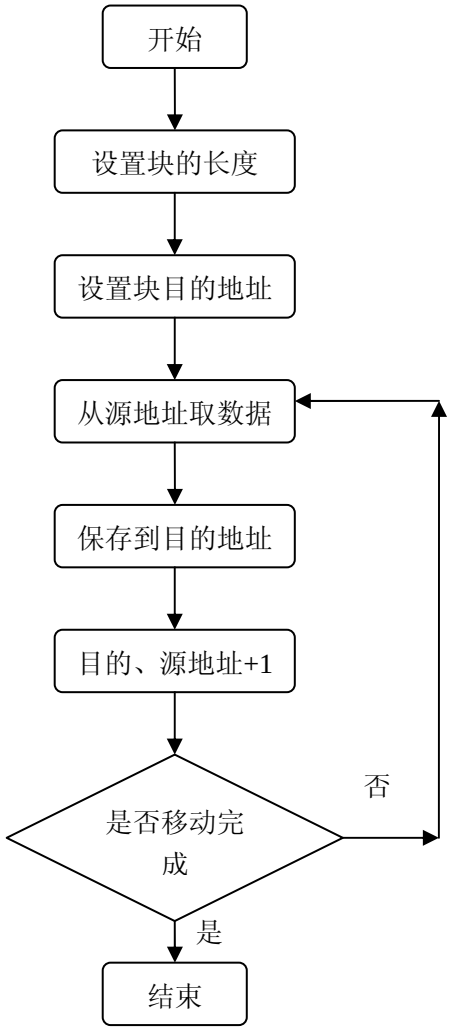
程序要求把内存中一数据区（称为源数据块）传送到内存另一数据区（称为目的数据块）。源数据块和目的数据块在存贮中可能有三种情况，如下图所示。



对于两个数据块分离的情况，如图（a），数据的传送从据块的首址开始，或者从数据块的末址开始均可。但对于有部分重叠的情况，则要加以分析，否则重叠部分会因“搬移”而遭破坏。

可以得出如下结论：当源数据块首址大于目的块首址时，从数据块首地址开始传送数据。当源数据块首址小于目的块首址时，从数据块末址开始传送数据。

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“内存块移动.DSN”;
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
START:
                MOV SI,1000H
                MOV CX,100
                MOV AL,1
PU_IN:  MOV [SI],AL ;先存入 1000H 开始的 100 个字节数据为 1 到 100
                INC AL
                INC SI
```

```
                LOOP PU_IN

                MOV CX,100
                MOV SI,1000H
                MOV DI,1100H ;

FADR:   MOV AL,[SI]
                MOV [DI],AL
                INC SI
                INC DI
                DEC CX
                JNE FADR
                JMP $
CODE    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．尝试改变源地址中的内容、长度，试试移动的结果；

d．检查验证结果。

六、 实验结果和体会

七、 建议

2.7　实验五 十六进制转BCD实验

一、 实验要求

利用 PROTEUS 平台，建立 8086 的十六进制转 BCD 例子。

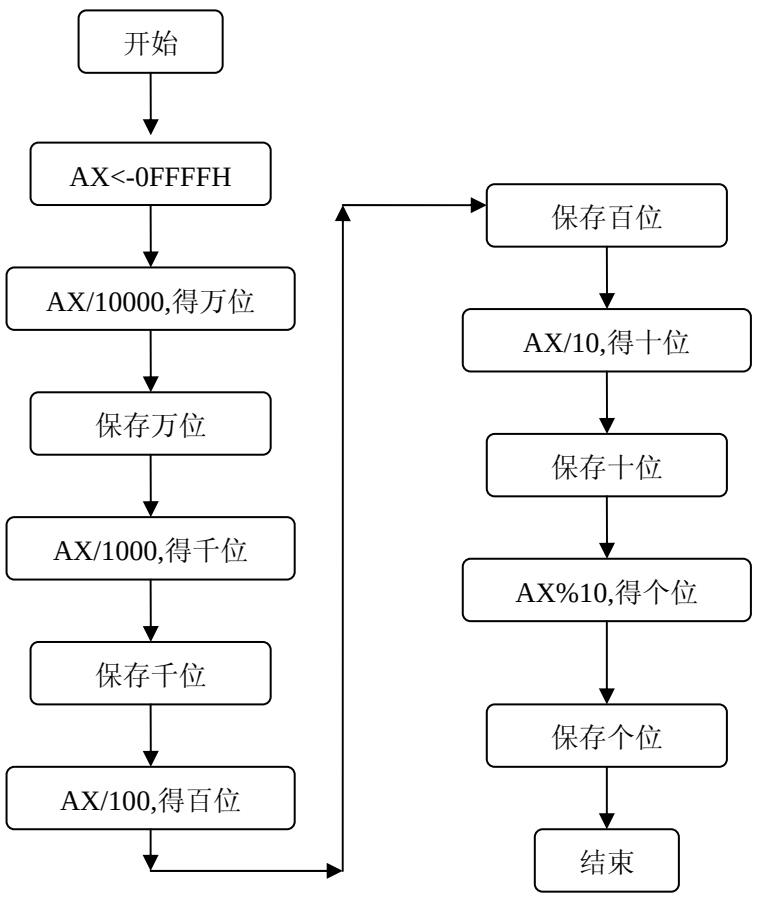
二、 实验目的

1、熟悉实验系统的编程和使用。
2、掌握简单的数值转换算法。
3、基本了解数值各种表达方法。

三、 实验说明

计算机中的数值有各种表达方式，这是计算机的基础。掌握各种数制之间的转换是一种基本功。有兴趣的同学可以试试将 BCD 转换成十六进制码。

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“十六进制转 BCD. DSN”;
b. 设置断点、运行程序，打开调试窗口进行调试。
参考程序：

```
;将 AX 拆为 5 个 BCD 码,并存入 RESULT 开始的 5 个单元
;AX=0FFFFH=65535
CODE SEGMENT
        ASSUME CS:CODE, DS:DATA
START:
        MOV AX, DATA
        MOV DS, AX
        MOV DX,0000H
        MOV AX, 65535
        MOV CX, 10000
        DIv CX
        MOV RESULT, AL ; 除以 10000, 得 WAN 位数
        MOV AX,DX
        MOV DX,0000H
        MOV CX, 1000
        DIv CX
        MOV RESULT+1, AL ; 除以 1000, 得 QIAN 位数
        MOV AX,DX
        MOV DX,0000H
        MOV CX, 100
        DIv CX
        MOV RESULT+2, AL ; 除以 100, 得 BAI 位数
        MOV AX,DX
        MOV DX,0000H
        MOV CX, 10
        DIv CX
        MOV RESULT+3, AL ; 除以 10, 得 SHI 位数
        MOV RESULT+4, DL ; 得 GE 位数
        JMP $
CODE ENDS
DATA SEGMENT
        RESULT Db 5 DUP(?)
DATA 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 AX 中给定数为 0FFFF，查看 BCD 码 Result 开始的 5 个单元，故其值应为 06、05、05、03、05。

六、 实验结果和体会

七、 建议

2.8 实验六 由 1 到 100 求和实验

一、 实验要求

利用 PROTEUS 平台，建立 8086 的 1 到 100 求和运算。

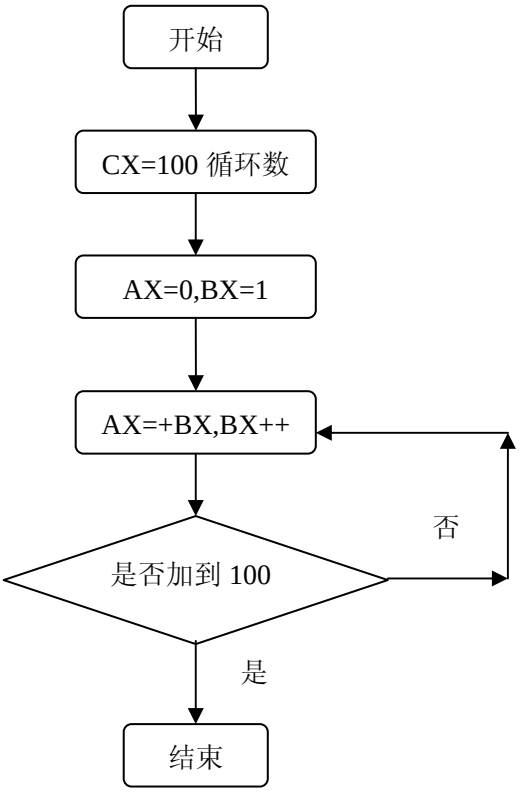
二、 实验目的

1、熟悉实验系统的编程和使用。
2、掌握使用加法类运算指令编程及调试方法。
3、掌握使用循环类指令编程及调试方法。

三、 实验说明

由于本实验是 1 到 100 的 100 个数想加，1+2+3+4+……+97+98+99+100=？求和

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 1 加到 100. DSN”;

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
BEG:        MOV AX,DATA
            MOV DS,AX
            MOV SI,OFFSET total
            MOV CX,100
            MOV AX,0
            MOV BX,1
add_100:    ADD AX,BX
            INC BX
            LOOP add_100
            MOV [SI],AX
            JMP $
CODE ENDS
DATA SEGMENT
total DW 0000H ;
DATA ENDS
            END BEG
```

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．total=1+2+3+4.....+99+100 = 5050 =13BA H(16 进制) 验证结果是否正确

六、 实验结果和体会

七、 建议

2.9 实验七 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。

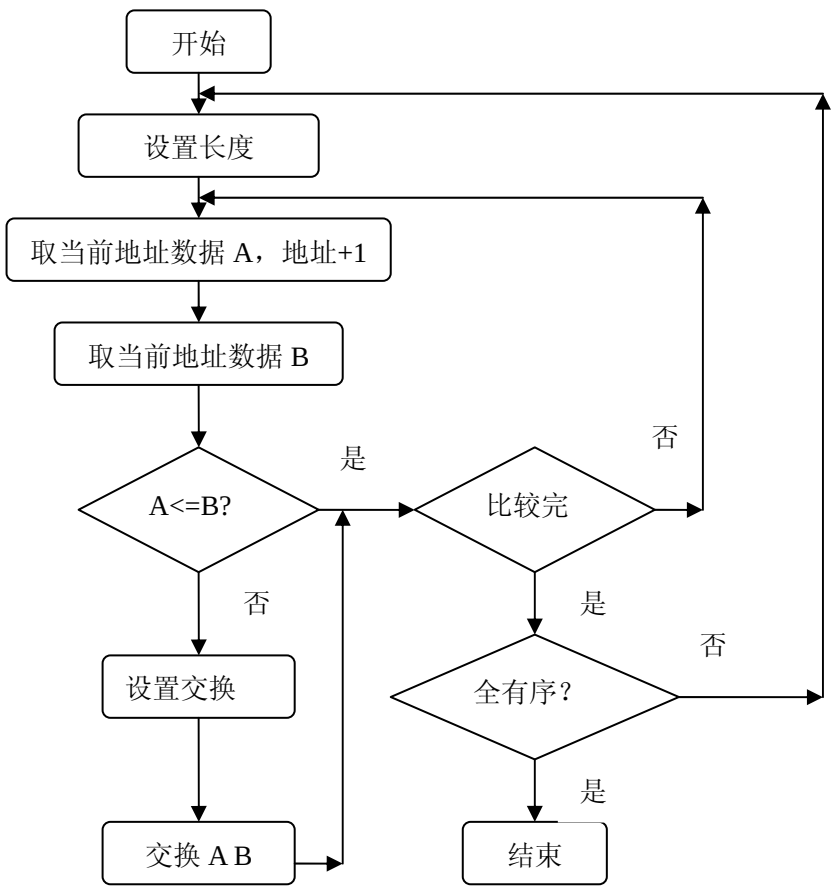
二、 实验目的

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“由小到大的数据排列.DSN”;
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

2.10 实验八 求表中正数_负数_0 的个数实验

一、 实验要求

利用 PROTEUS 平台，建立 8086 的查表中正数、负数与 0 的个数。

二、 实验目的

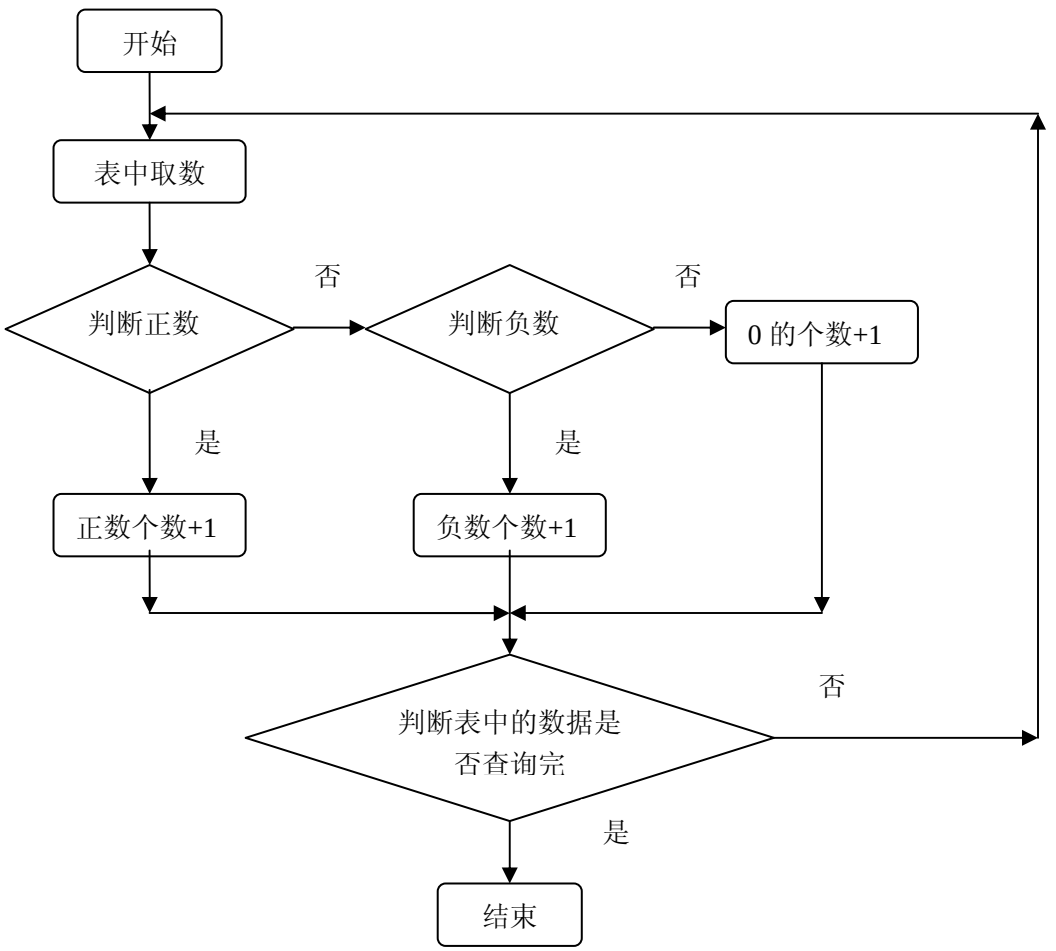
1、熟悉实验系统的编程和使用。

2、掌握查表方法。

三、 实验说明

由于本实验是先在表中存放数据，其它有正数、负数和 0，通过程序对表的查询，统计表中包含正数、负数和 0 的个数。

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“求表中正数_负数_0 的个数.DSN”;
b. 单步运行，打开调试窗口进行调试。
参考程序：

```
CODE SEGMENT
ASSUME   CS:CODE,DS:DATA
START: MOV   CX,DATA
                MOV   DS,CX
                MOV   DI,OFFSET TABLE
                XOR    BL,BL
                XOR    DL,DL
                XOR    AL,AL
                MOV   CX,20
                JCXZ DONE
                LEA   DI,TABLE+2
CMPNUM:    CMP WORD PTR[DI],0
                JGE A1
                INC BL
                JMP TONEXT
A1:             JE A2
                INC AL
                JMP TONEXT
A2:             INC   DL
TONEXT:    INC   DI
                INC   DI
                LOOP CMPNUM
DONE:
                MOV   NEGNUM,AX
                MOV   POSNUM,BX
                MOV   ZERONUM,DX
                JMP $
CODE ENDS
DATA SEGMENT
TABLE     DW   20   ;表中的数量
          DW -10,-9,-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9
NEGNUM   DW   0 ;存放正数的个数
POSNUM   DW   0 ;存放负数的个数
ZERONUM DW   0 ;存放 0 的个数
DATA ENDS
```

```
END START
```

2、调试、验证

a．设置断点，调试；

b．观察运行到断点时，8086 内部寄存器的数值变化；

c．检查验证结果；

六、 实验结果和体会

七、 建议

第3章 8086 硬件部分实验目录

3.1 实验九 IO口读写实验 (245、373)

一、 实验要求

利用板上集成电路上的资源，扩展一片 74HC245，用来读入开关状态；扩展一片 74HC373，用来作来输出口，控制 8 个 LED 灯。

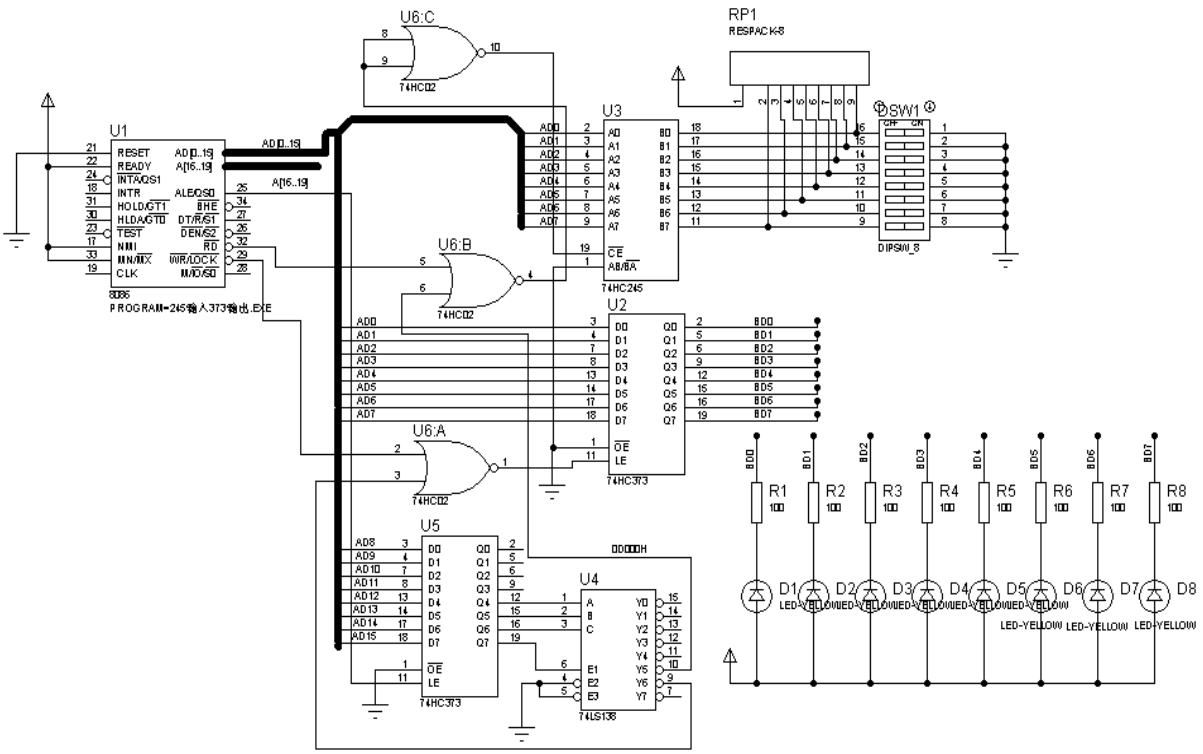
二、 实验目的

1、了解 CPU 常用的端口连接总线的方法。

2、掌握 74HC245、74HC373 进行数据读入与输出。

三、 实验电路及连线

1、Proteus 实验电路



2、硬件验证实验

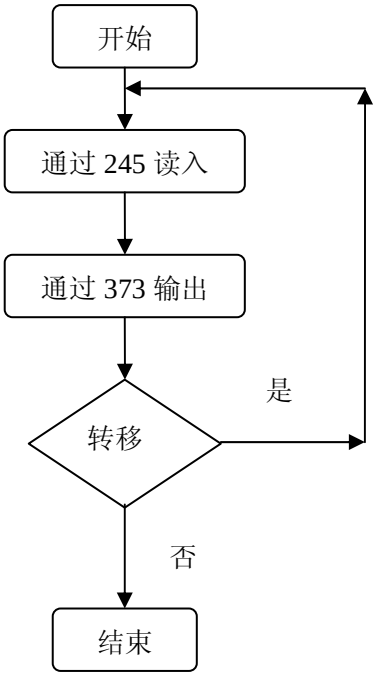
硬件连接表

接线孔 1	接线孔 2
245 CS	0D000H-0DFFFH
373 CS	8000H-8FFFH
B0—B7	K1—K8
Q0—Q7	D1—D8

四、 实验说明

一般情况下，CPU 的总线会挂有很多器件，如何使这些器件不造成冲突，这就要使用一些总线隔离器件，例如 74HC245、74HC373。74HC245 是三态总线收发器，本实验用它做输入，片选地址为 0D0000H-0DFFFFH。就是用于读入开关值。74HC373 是数据锁存芯片，通过它作数据的锁住输出。

五、 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 245 输入 373 输出_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。

参考程序：

```
CODE SEGMENT ;
                ASSUME CS:CODE
IN245 EQU 0D000H
```

```
OUT373 EQU 8000H

START:
        MOV DX,IN245
        IN   AL,DX
        MOV DX,OUT373
        OUT DX,AL
        JMP START
CODE ENDS
        END START
```

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

3.2 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

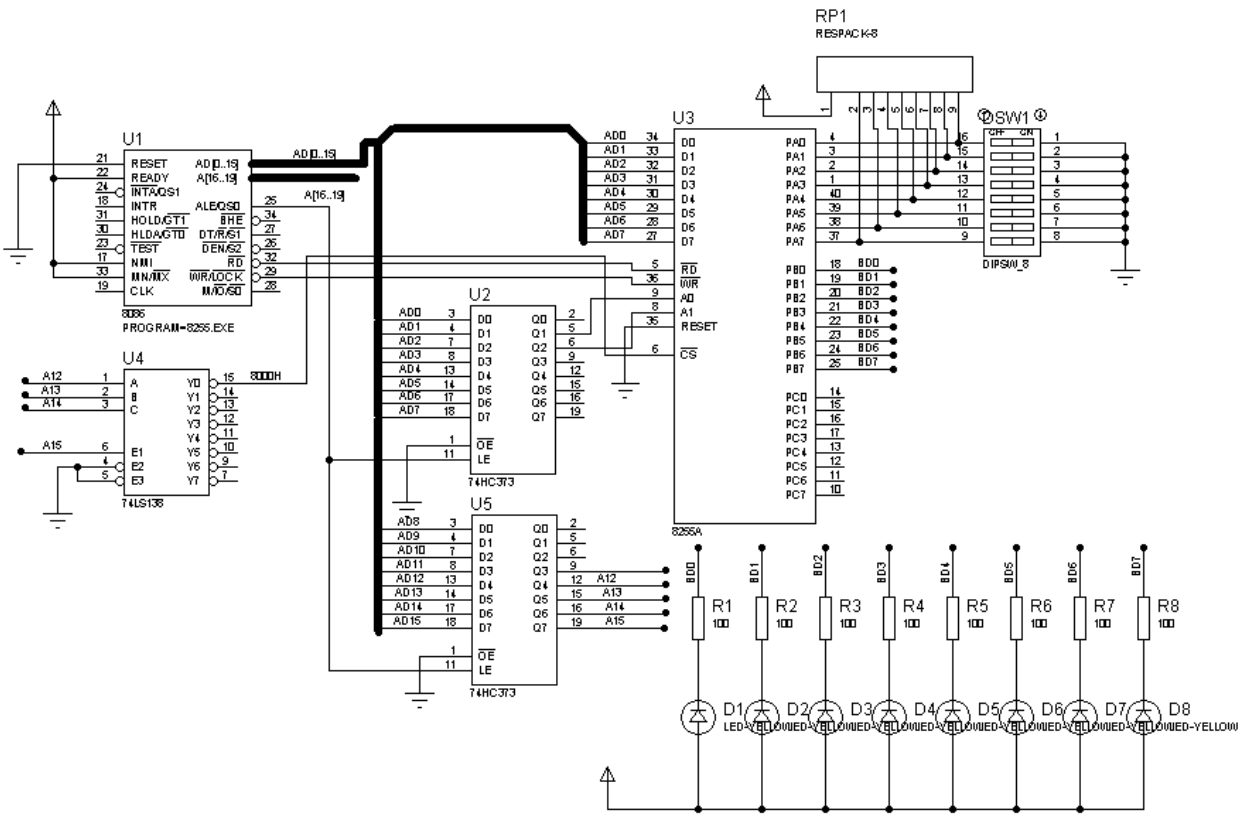
二、 实验目的

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	K1—K8

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：

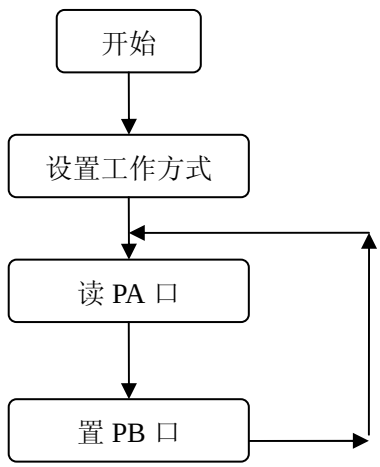
方式 0：基本输入/ 输出方式

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。

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
```

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

3.3 实验十一 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

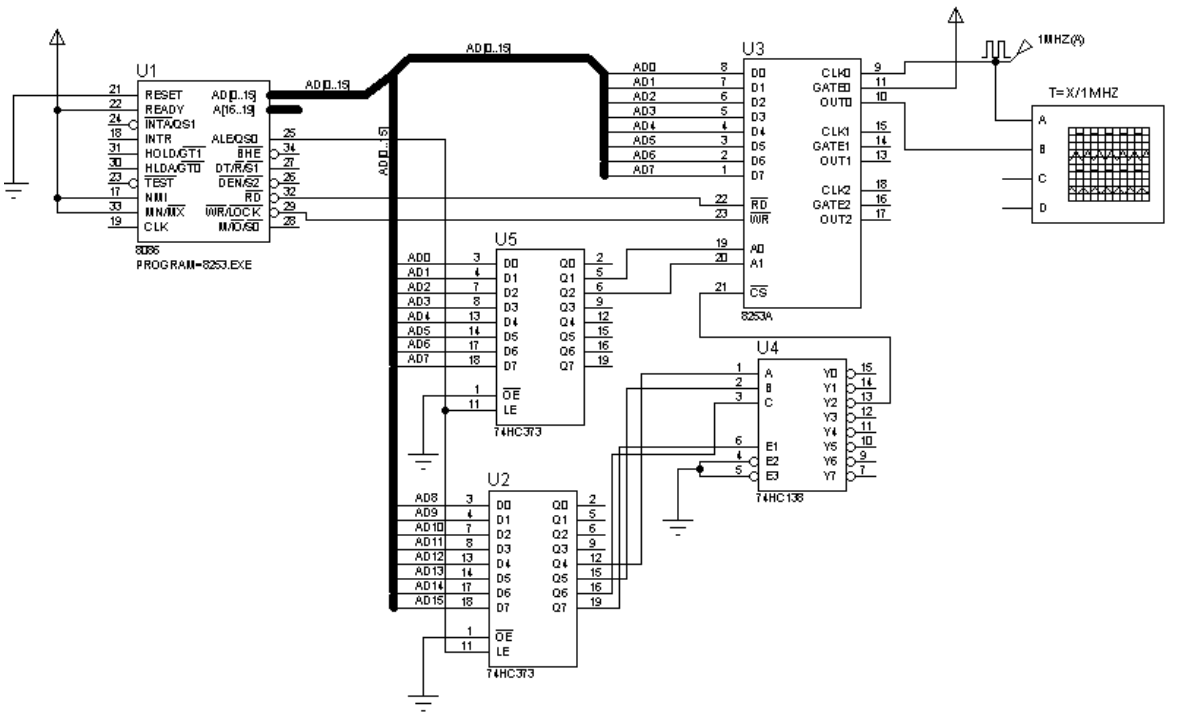
1、学习 8086 与 8253 的连接方法。

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	0A00H-0AFFH
CLOCK_OUT	CLOUK_IN
1/4	CLK0
GATE0	+5V

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：

⑴ 延时中断　　⑸ 实时时钟

⑵ 可编程频率发生器　　⑹ 数字单稳

⑶ 事件计数器　　⑺ 复杂的电机控制器

⑷ 二进制倍频器

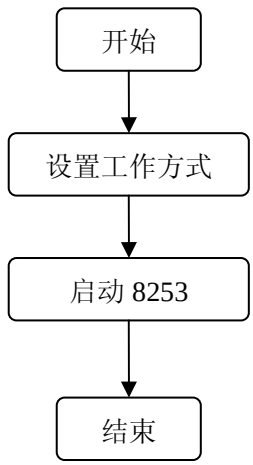
8253 的六种工作方式：

⑴ 方式 0：计数结束中断　⑷ 方式 3：方波频率发生器

⑵ 方式 l：可编程频率发生　⑸ 方式 4：软件触发的选通信号

⑶ 方式 2：频率发生器　⑹ 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。

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
```

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

3.4 实验十二 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

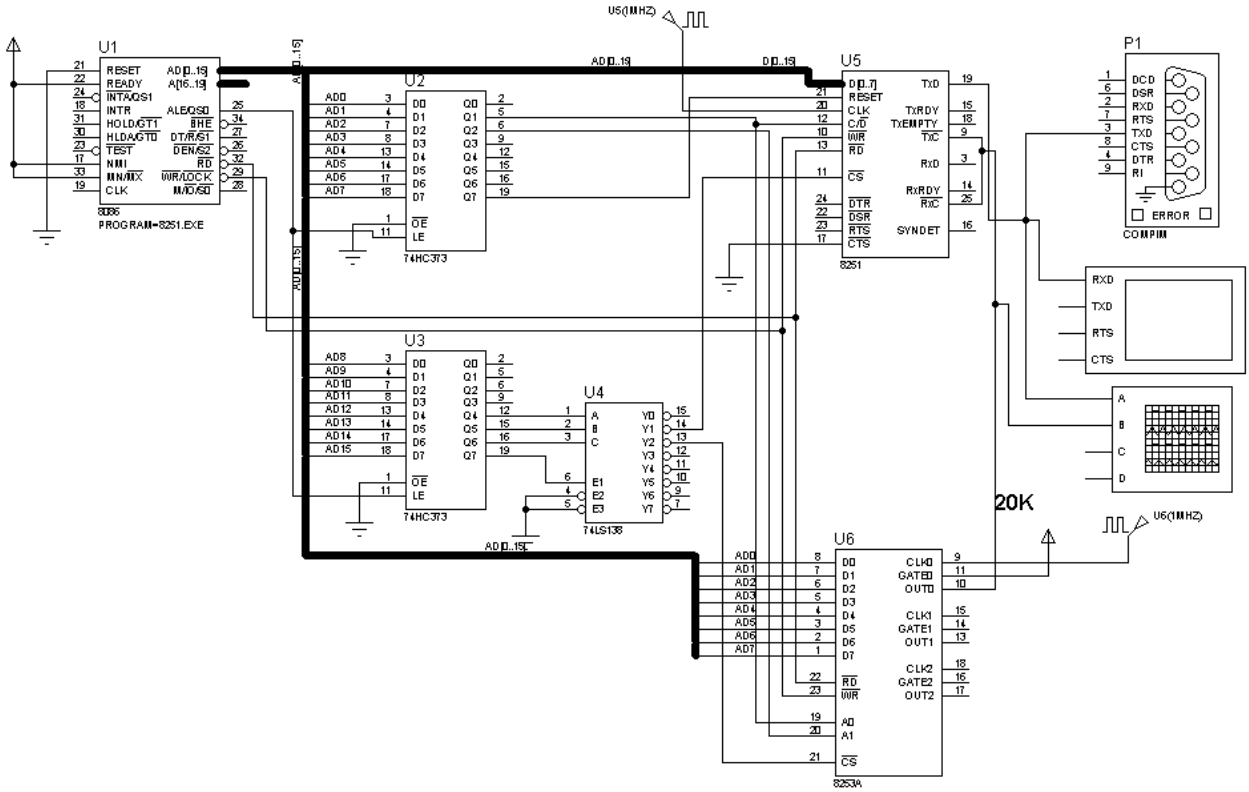
1、掌握 8086 实现串口通信的方法。

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	0A000H-0AFFFH

8251 CS	0F00H-0FFFH
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

四、 实验说明

（1）8251 状态口地址：F002H，8251 数据口地址：F000H；

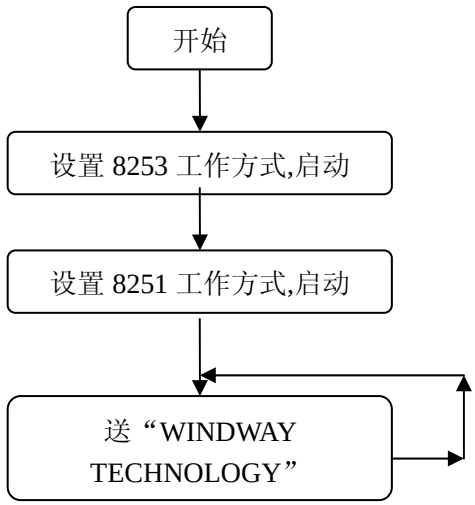
（2）8253 命令口地址：0A006H，8253 计数器 0 口地址：0A000H；

（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“8251A_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。

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
```

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

3.5　实验十三 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。

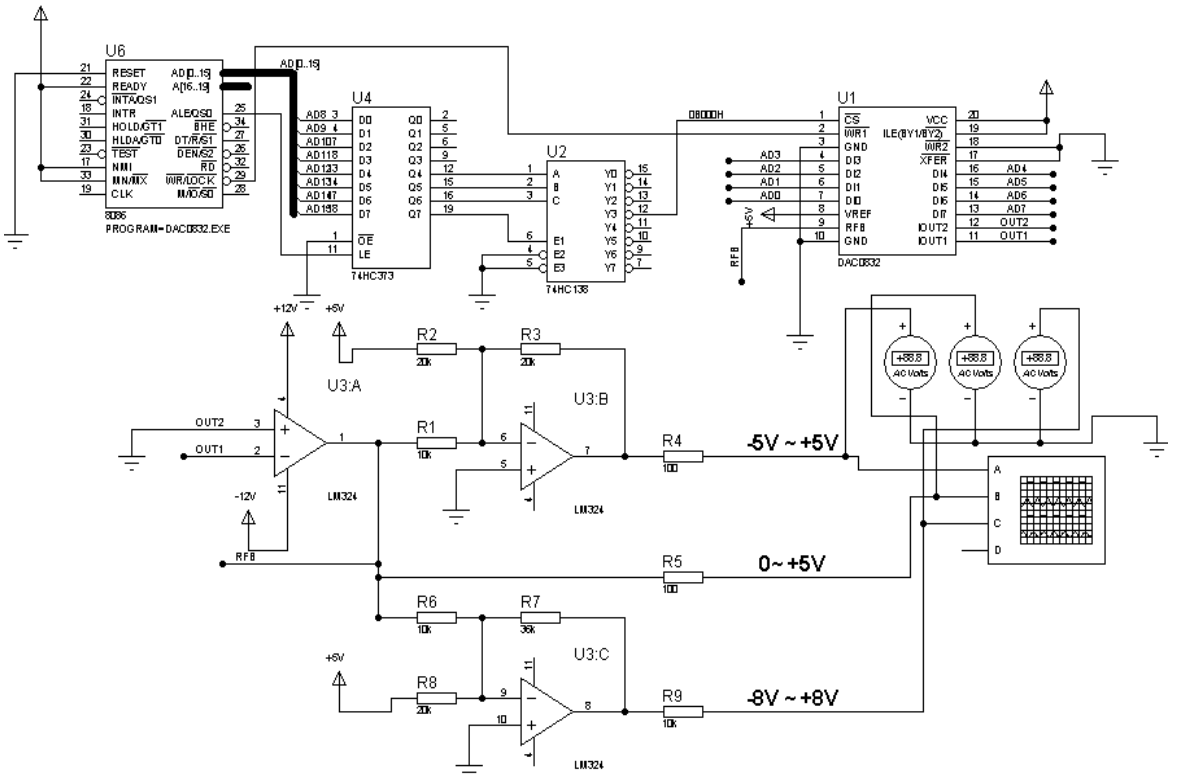
二、 实验目的

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	0B000H-0BFFFH
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：

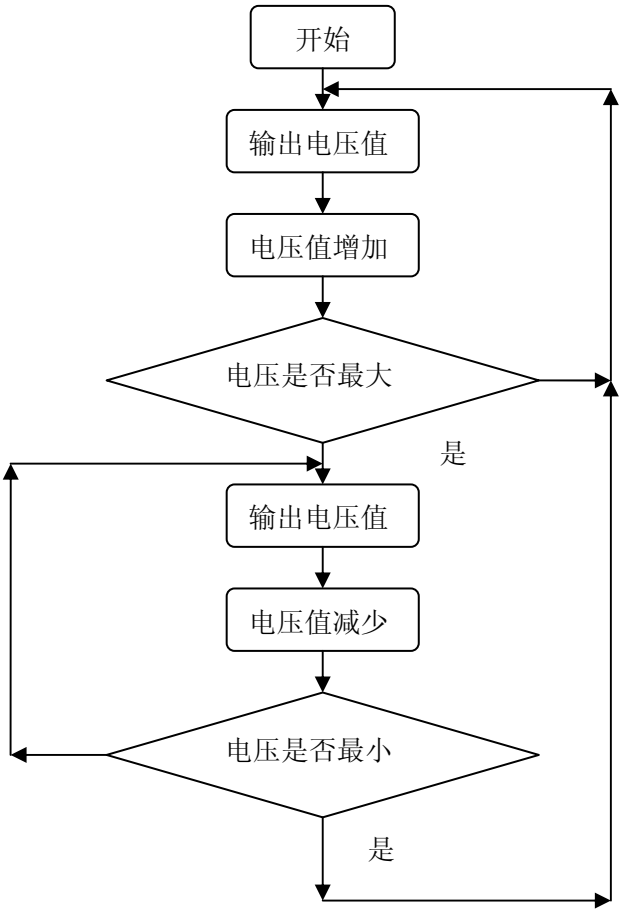
本实验用到的主要知识点是：DAC0832 的工作原理。

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.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。

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
```

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

3.6　实验十四 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。

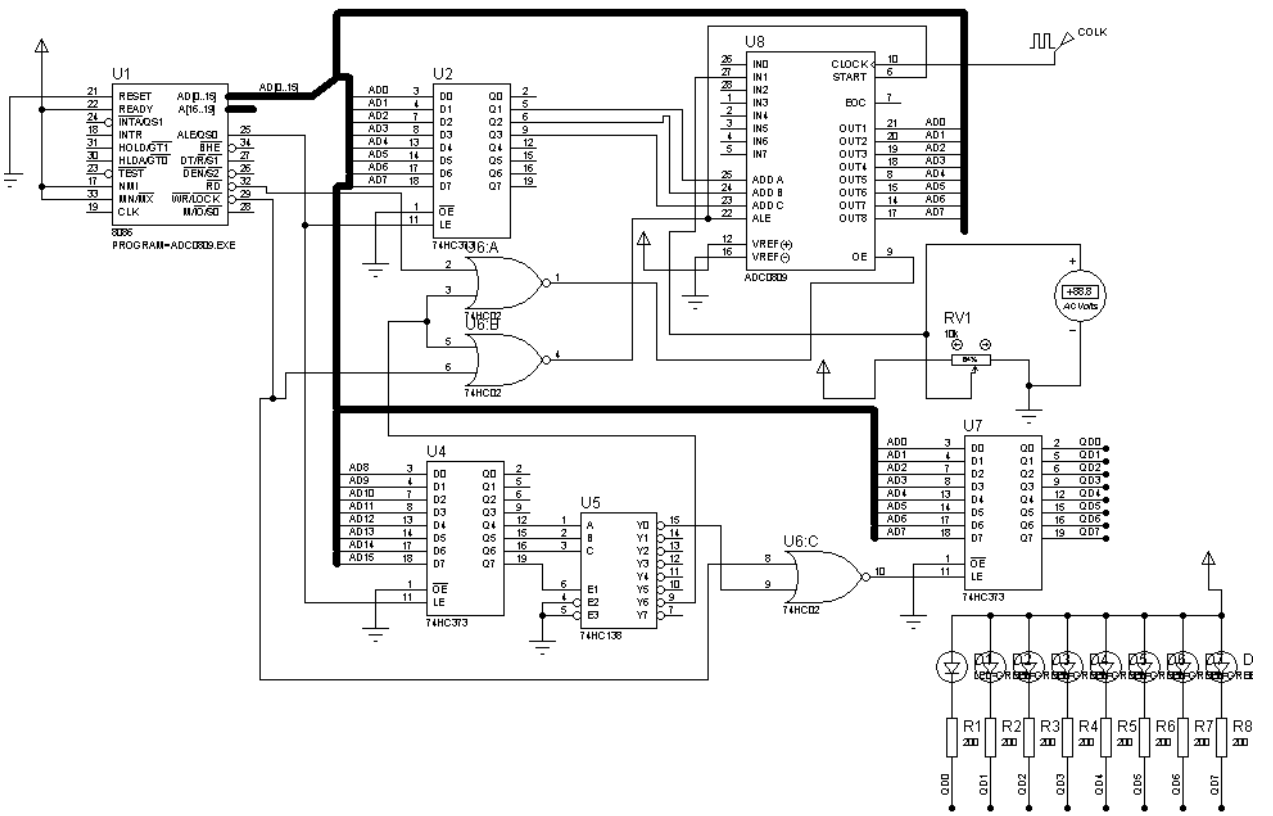
二、 实验目的

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	0E000H-0EFFFH
373　CS	8000H-8FFFH

CLOCK_OUT	CLOCK_IN
1/4	CLK
IN0	AD_IN
Q0--Q7	D1—D8

四、 实验说明

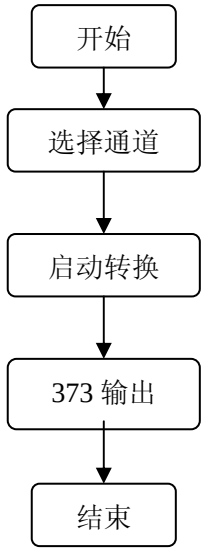
1、主要知识点概述：

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.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。

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
```

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

3.7　实验十五 1602 液晶显示的控制实验（44780）

一、 实验要求

利用实验板搭建液晶显示电路，编写程序控制输出显示数字和英文字符。

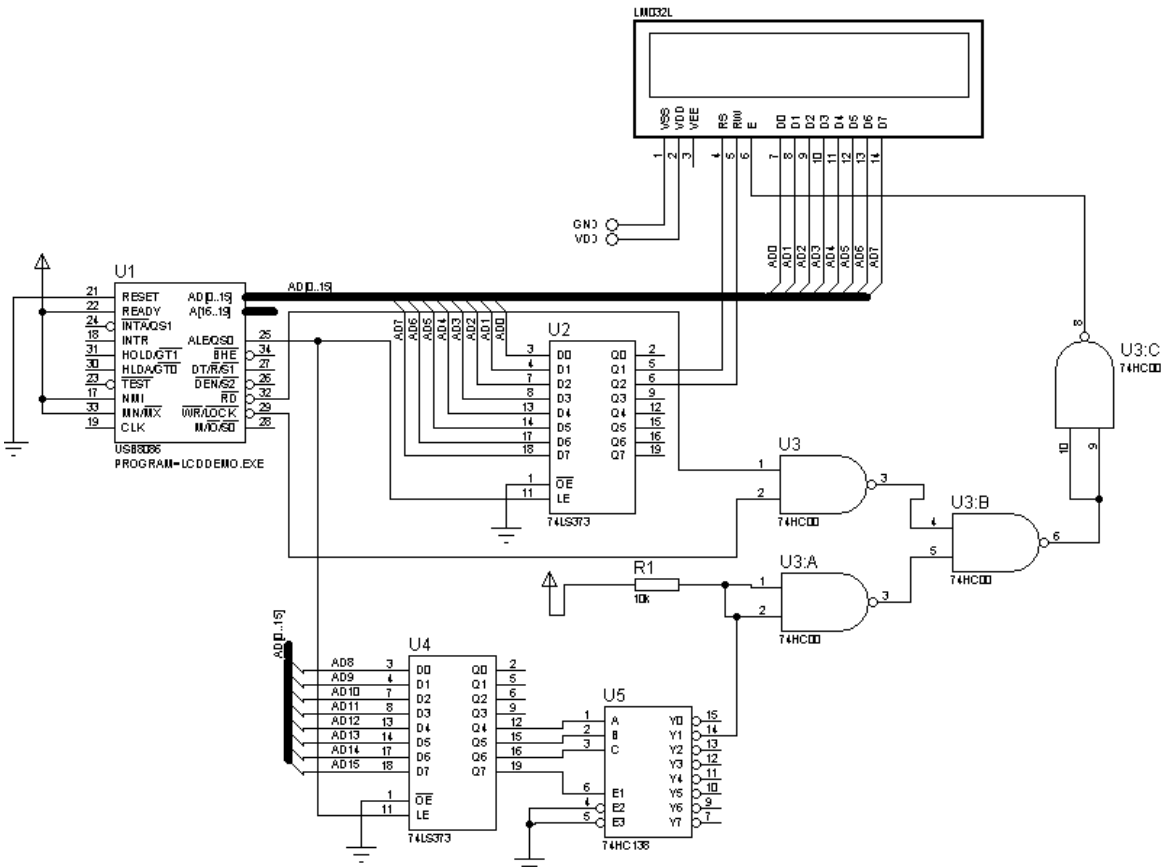
二、 实验目的

1、了解字符型液晶显示屏的控制原理和方法；

2、了解数字和字符的显示原理。

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
1602 CS	09000H-09FFFH

四、 实验说明

1、主要知识点概述：

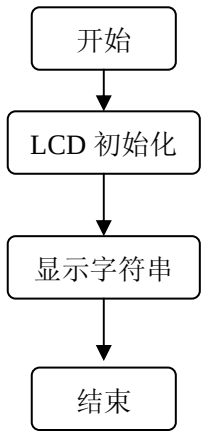
理解 44780 控制器的相关原理和控制命令。

2、实验效果说明：

本实验仪采用的液晶显示屏内置控制器为 44780，可以显示 2 行共 32 个 ASCII 字符。有关图形液晶显示屏的命令和详细原理，可参考有关的液晶模块资料。

实验效果：液晶动态显示“ 'WINDWAY TECHNOLOGY '
'!! A M A Z I N G !!'”字符。

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“LCD1602_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。
参考程序：

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

LCD_CMD_WR   EQU    9000H
LCD_DATA_WR  EQU    9002H
LCD_BUSY_RD  EQU    9004H
LCD_DATA_RD  EQU    9006H

START:
            MOV AX,DATA
            MOV DS,AX
            MOV AX,STACK
            MOV SS,AX
```

```
                MOV AX,TOP
                MOV SP,AX
                IN    AX,DX
                MOV AX,30H
                CALL WRCMD
                MOV AX,38H
                CALL WRCMD
                MOV AX,0CH
                CALL WRCMD
                MOV AX,01H
                CALL WRCMD
                MOV AX,06H
                CALL WRCMD

MAINLOOP:
                MOV AX,80H
                MOV CX,20
                LEA DI,str1
                CALL WRSTR
                MOV AX,0C0H
                MOV CX,20
                LEA DI,str2
                CALL WRSTR
                MOV AX,01H
                CALL WRCMD
                JMP MAINLOOP

WRCMD:      MOV DX,LCD_CMD_WR
                OUT DX,AX
                RET

;入口参数：
;AX-->行地址，第一行地址为 80H,第二行地址为 C0H
;CX-->字符数，不超过 20
;DI-->字符串首地址
WRSTR:      CALL WRCMD
                MOV DX,LCD_DATA_WR
WRBIT:      MOV AL,[DI]
                OUT DX,AL
                INC DI
```

```
                LOOP WRBIT
WRRET:          RET

CODE ENDS

STACK           SEGMENT 'STACK'
STA             DB    100 DUP(?)
TOP             EQU LENGTH STA
STACK ENDS

DATA            SEGMENT 'DATA'
str1 db ' WINDWAY TECHNOLOGY    '
str2 db ' !! A M A Z I N G !! '
DATA ENDS
                END START
```

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

3.8 实验十六 12864 液晶显示的控制实验（KS0108）

一、 实验要求

利用实验仪上的液晶屏(KS0108)电路，编写程序控制输出显示数字、英文字符、汉字或图形等。

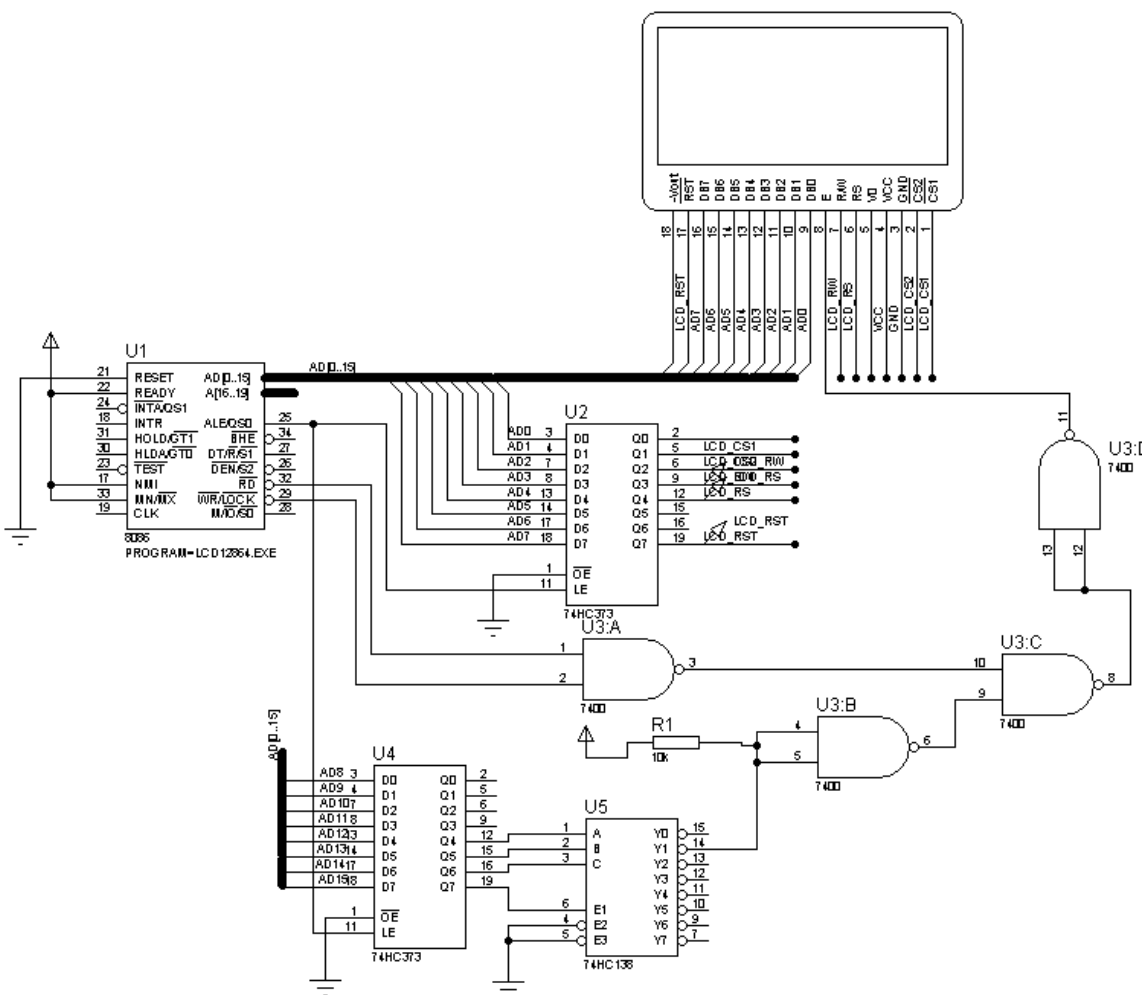
二、 实验目的

1、了解液晶显示屏的控制原理和方法；

2、了解数字、字符以及点阵汉字或图形的显示原理。

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
12864 CS	09000H-09FFFH

四、 实验说明

1、主要知识点概述：

本实验仪采用的液晶显示屏内置控制器为 KS0108，点阵为 128×64，需要两片 KS0108 组成，由 CS1、CS2 分别选通，以控制显示屏的左右两半屏。有关图形液晶显示屏的命令和详细原理，可参考有关的液晶模块资料。

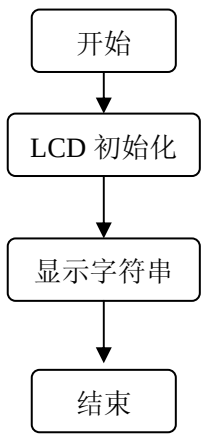
2、实验效果说明：

在该屏幕上显示 ”PROTEUS”

“电子设计与创新的”

“最佳平台”

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 LCD12864_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。

参考程序:

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

        LCD_CMD_WR    EQU    9086H
        LCD_DATA_WR1 EQU    9092H    ;写第一屏数据
        LCD_DATA_WR2 EQU    9094H    ;写第二屏数据
        LCD_DATA_WR0 EQU    9096H    ;写双屏数据
        LCD_DATA_RD    EQU    909EH
```

```
LCD_RST         EQU     9000H     ;复位
LCD_RST_OK      EQU     9080H     ;复位

LCD_ROW         EQU     0C0H      ;设置起始行
LCD_PAGE        EQU     0B8H      ;设置起始页
LCD_COLUMN      EQU     40H       ;设置起始列
LCD_PAGE_MAX    EQU     08H       ;页数最大值
LCD_COLUMN_MAX  EQU     40H       ;列数最大值

START:
        MOV AX, DATA
        MOV DS, AX
        MOV AX, STACK
        MOV SS, AX
        MOV AX, TOP
        MOV SP, AX
        CALL LCD_INIT
        CALL LCD_CLEAR

MAINLOOP:
        CALL WRSTR_PROTEUS
        CALL WRHZ
        JMP START

LCD_INIT:
        MOV DX, LCD_RST
        OUT DX, AL
        MOV AL, 00H
        MOV DX, LCD_RST_OK
        OUT DX, AL
        IN  AL, DX

        MOV AL, LCD_ROW
        CALL WRCMD

        MOV AL, LCD_COLUMN
        CALL WRCMD

        MOV AL, LCD_PAGE
        CALL WRCMD
```

```
                MOV AL, 3FH
                CALL WRCMD
                RET

LCD_CLEAR:
                MOV BL, 00H
LC_LOOP:        MOV AL, BL
                ADD AL, LCD_PAGE
                CALL WRCMD
                MOV AL, LCD_COLUMN
                CALL WRCMD
                MOV CX, LCD_COLUMN_MAX
                MOV AL, 00H
                MOV DX, LCD_DATA_WR0
LC_WDATA:
                OUT DX, AL
                LOOP LC_WDATA
                INC BL
                CMP BL, LCD_PAGE_MAX
                JL LC_LOOP
                RET

WRCMD:          PUSH DX
                MOV DX, LCD_CMD_WR
                OUT DX, AL
                POP DX
                RET

WRSTR_PROTEUS:
                MOV AX, 8
                MOV CX, 7
                MOV BX, 0028H
                LEA DI, ZF_P
WRSTR_LOOP:
                CALL WRCHAR
                ADD BX, AX
                ADD DI, 16
                LOOP WRSTR_LOOP
                RET

WRHZ:
```

```
                MOV AX, 16
                MOV CX, 8
                MOV BX, 0200H
                LEA DI, TABLE_HZ
WRHZ_LOOP:
                CALL WRCHAR
                ADD BX, AX
                ADD DI, 32
                LOOP WRHZ_LOOP
                MOV BX, 0420H
                MOV CX, 4
WRHZ_LOOP1:
                CALL WRCHAR
                ADD BX, AX
                ADD DI, 32
                LOOP WRHZ_LOOP1
                RET

;入口参数：
;AX-->字符宽度，ASCII 为 8，汉字为 16
;BL-->X 地址，0-127
;BH-->Y 地址，0-7
;DI-->字符首地址
WRCHAR:    PUSH AX
                PUSH BX
                PUSH CX
                PUSH DX
                PUSH DI
                MOV CX,AX
                MOV AH, 02H
                ADD BL, LCD_COLUMN
                ADD BH, LCD_PAGE
WRCHAR_P:
                MOV AL,BH
                CALL WRCMD
                TEST BL, 80H
                JNZ    WRCHAR_P0
                MOV AL, BL
                CALL WRCMD
                MOV DX, LCD_DATA_WR1
                JMP WRCHAR_P1
```

```
WRCHAR_P0:
                MOV AL, BL
                SUB AL, 40H
                CALL WRCMD
                MOV DX, LCD_DATA_WR2
WRCHAR_P1:
                PUSH CX
WRBIT1:         MOV AL, [DI]
                OUT DX, AL
                INC DI
                LOOP WRBIT1
                POP CX
                INC BH
                MOV AL,00H
                XCHG AL,AH
                DEC AX
                XCHG AL,AH
                JNZ WRCHAR_P
                POP DI
                POP DX
                POP CX
                POP BX
                POP AX
WRRET:          RET

CODE ENDS

STACK     SEGMENT 'STACK'
STA         DB    100 DUP(?)
TOP         EQU LENGTH STA
STACK     ENDS

DATA        SEGMENT 'DATA'
;--   文字:   P   --
;--   宋体 12;   此字体下对应的点阵为：宽 x 高=8x16     --
ZF_P DB 08H,0F8H,08H,08H,08H,08H,0F0H,00H,20H,3FH,21H,01H,01H,01H,00H,00H,

;--   文字:   R   --
;--   宋体 12;   此字体下对应的点阵为：宽 x 高=8x16     --
ZF_R DB 08H,0F8H,88H,88H,88H,88H,70H,00H,20H,3FH,20H,00H,03H,0CH,30H,20H,
```

```
;--  文字:  O  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=8x16    --
ZF_O DB 0E0H,10H,08H,08H,08H,10H,0E0H,00H,0FH,10H,20H,20H,20H,10H,0FH,00H,

;--  文字:  T  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=8x16    --
ZF_T DB 18H,08H,08H,0F8H,08H,08H,18H,00H,00H,00H,20H,3FH,20H,00H,00H,00H,

;--  文字:  E  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=8x16    --
ZF_E DB 08H,0F8H,88H,88H,0E8H,08H,10H,00H,20H,3FH,20H,20H,23H,20H,18H,00H,

;--  文字:  U  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=8x16    --
ZF_U DB 08H,0F8H,08H,00H,00H,08H,0F8H,08H,00H,1FH,20H,20H,20H,20H,1FH,00H,

;--  文字:  S  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=8x16    --
ZF_S DB 00H,70H,88H,08H,08H,08H,38H,00H,00H,38H,20H,21H,21H,22H,1CH,00H,

;--  文字:  电  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --
TABLE_HZ DB 0H,0H,0F8H,48H,48H,48H,48H,0FFH,48H,48H,48H,48H,0F8H,0H,0H,0H
         DB 0H,0H,0FH,04H,04H,04H,04H,3FH,44H,44H,44H,44H,4FH,40H,70H,00H

;--  文字:  子  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --
DB   00H,00H,02H,02H,02H,02H,02H,0E2H,12H,0AH,06H,02H,00H,80H,00H,00H
DB   01H,01H,01H,01H,01H,41H,81H,7FH,01H,01H,01H,01H,01H,01H,01H,00H

;--  文字:  设  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --
DB   40H,41H,0CEH,04H,00H,80H,40H,0BEH,82H,82H,82H,0BEH,0C0H,40H,40H,00H
DB   00H,00H,7FH,20H,90H,80H,40H,43H,2CH,10H,10H,2CH,43H,0C0H,40H,00H

;--  文字:  计  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --
DB   20H,21H,2EH,0E4H,00H,00H,20H,20H,20H,20H,0FFH,20H,20H,20H,20H,00H
DB   00H,00H,00H,7FH,20H,10H,08H,00H,00H,00H,0FFH,00H,00H,00H,00H,00H

;--  文字:  与  --
```

```
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --
DB   00H,00H,00H,00H,7EH,48H,48H,48H,48H,48H,48H,48H,48H,0CCH,08H,00H
DB   00H,04H,04H,04H,04H,04H,04H,04H,04H,24H,46H,44H,20H,1FH,00H,00H

;--  文字:  创  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --
DB   40H,20H,0D0H,4CH,43H,44H,48H,0D8H,30H,10H,00H,0FCH,00H,00H,0FFH,00H
DB   00H,00H,3FH,40H,40H,42H,44H,43H,78H,00H,00H,07H,20H,40H,3FH,00H

;--  文字:  新  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --
DB   20H,24H,2CH,35H,0E6H,34H,2CH,24H,00H,0FCH,24H,24H,0E2H,22H,22H,00H
DB   21H,11H,4DH,81H,7FH,05H,59H,21H,18H,07H,00H,00H,0FFH,00H,00H,00H

;--  文字:  的  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --
DB   00H,0F8H,8CH,8BH,88H,0F8H,40H,30H,8FH,08H,08H,08H,08H,0F8H,00H,00H
DB   00H,7FH,10H,10H,10H,3FH,00H,00H,00H,03H,26H,40H,20H,1FH,00H,00H

;--  文字:  最  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --
DB   40H,40H,0C0H,5FH,55H,55H,0D5H,55H,55H,55H,55H,5FH,40H,40H,40H,00H
DB   20H,20H,3FH,15H,15H,15H,0FFH,48H,23H,15H,09H,15H,23H,61H,20H,00H

;--  文字:  佳  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --
DB   40H,20H,0F0H,1CH,47H,4AH,48H,48H,48H,0FFH,48H,48H,4CH,68H,40H,00H
DB   00H,00H,0FFH,00H,40H,44H,44H,44H,44H,7FH,44H,44H,46H,64H,40H,00H

;--  文字:  平  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --
DB   00H,01H,05H,09H,71H,21H,01H,0FFH,01H,41H,21H,1DH,09H,01H,00H,00H
DB   01H,01H,01H,01H,01H,01H,01H,0FFH,01H,01H,01H,01H,01H,01H,01H,00H

;--  文字:  台  --
;--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --
DB   00H,00H,40H,60H,50H,48H,44H,63H,22H,20H,20H,28H,70H,20H,00H,00H
DB   00H,00H,00H,7FH,21H,21H,21H,21H,21H,21H,21H,7FH,00H,00H,00H,00H
DATA      ENDS
          END START
```

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

3.9 实验十七 七段数码管显示实验

一、 实验要求

利用 8255 的 IO 控制 8 位七段数码管显示实验，实现显示。

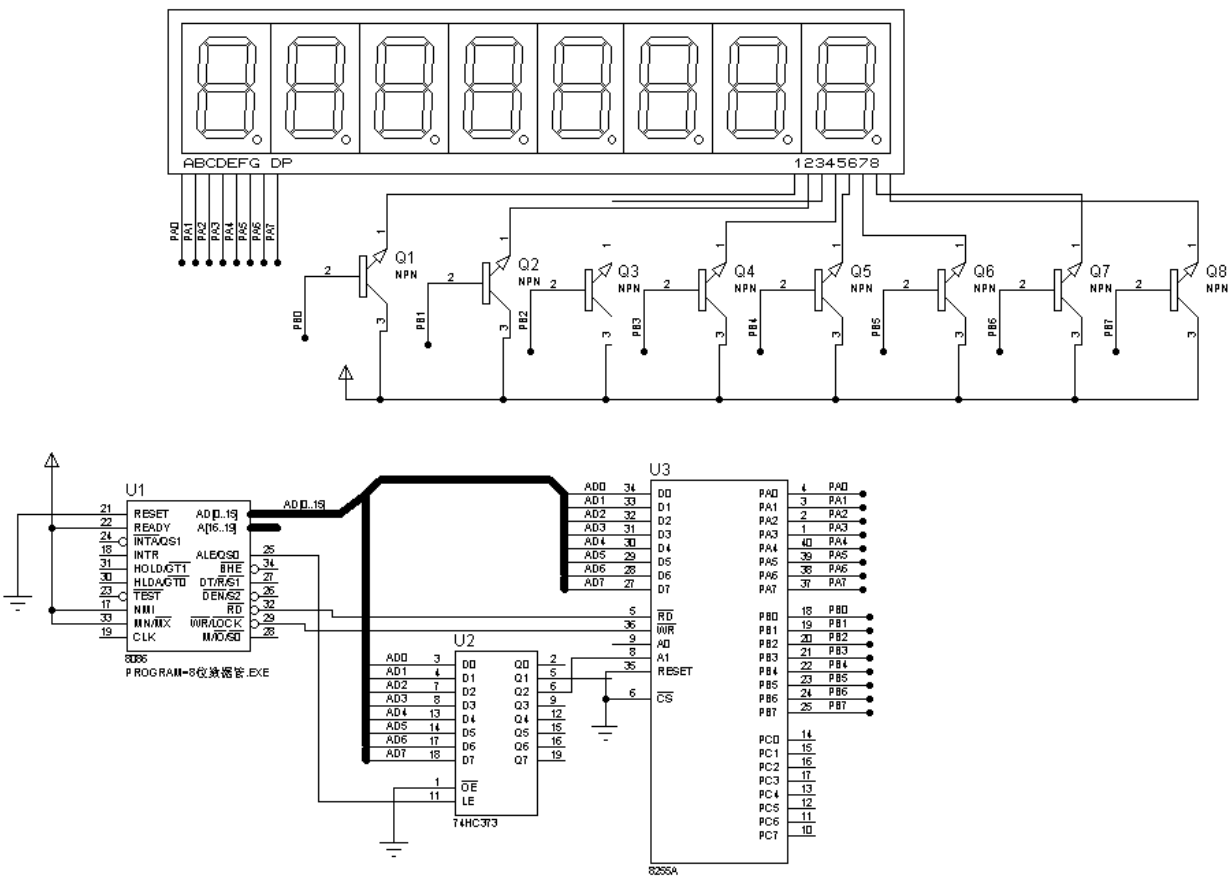
二、 实验目的

1. 了解数码管显示原理。

2. 掌握读表程序的编写。

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

COM_1—COM_8	PA0—PA7
COM_A—COM_DP	PB0—PB7

四、 实验说明

1．主要知识点概述：

1）LED 数码显示原理

七段 LED 显示器内部由七个条形发光二极管和一个小圆点发光二极管组成，根据各管的极管的接线形式，可分成共阴极型和共阳极型。

LED 数码管的 g~a 七个发光二极管因加正电压而发亮，因加零电压而不以发亮，不同亮暗的组合就能形成不同的字形，这种组合称之为字形码，下面给出共阳极的字形码见表 2

“0”	0C0H		“8”	80H
“1”	0F9H		“9”	90H
“2”	0A4H		“A”	88H
“3”	0B0H		“b”	80H
“4”	99H		“C”	0B6H
“5”	92H		“d”	0B0H
“6”	82H		“E”	86HH
“7”	F8H		“F”	8EH

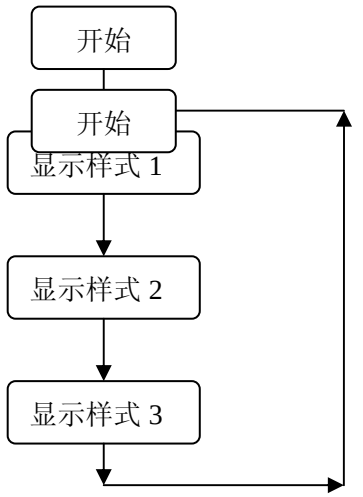
2) 段码表格

由于显示的数字 0－9 的字形码没有规律可循，只能采用查表的方式来完成我们所需的要求了。这样我们按着数字 0－9 的顺序，把每个数字的笔段代码按顺序排好！建立的表格如下所示：TABLE　DB　　0c0h,0f9h,0a4h,0b0h,99h,92h,82h,0f8h,80h,90h

2．实验效果说明：

数码管循环显示 0~9。

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“8 位数码管_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。
参考程序:

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

IOCON   EQU 8006H
IOA      EQU 8000H
IOB      EQU 8002H
IOC      EQU 8004H

START:
              MOV AX, DATA
              MOV DS, AX

              MOV AX, STACK
              MOV SS, AX

              MOV AX, TOP
              MOV SP, AX

TEST_BU:  MOV AL,80H
              MOV DX,IOCON
              OUT DX,AL
              NOP

              LEA DI,TABLE
              MOV CX,0AH
              MOV DX,IOB
              MOV AL,0FFH
              OUT DX,AL
              MOV DX,IOA

DISPLA1:  MOV AL,[DI]
              OUT DX,AL
              CALL DELAY
              INC DI
```

```
                LOOP DISPLA1
                LEA DI,TABLE
                MOV CX,08H
                MOV BH,80H

DISPLA2:        MOV DX,IOB
                MOV AL,BH
                OUT DX,AL
                MOV DX,IOA
                MOV AL,[DI]
                OUT DX,AL
                CALL DELAY
                INC DI
                MOV AL,BH
                SHR AL,1
                MOV BH,AL
                LOOP DISPLA2

                LEA DI,TABLE
                MOV CX,08H
                MOV BH,80H

DISPLA3:        MOV DX,IOB
                MOV AL,BH
                OUT DX,AL
                MOV DX,IOA
                MOV AL,[DI]
                OUT DX,AL
                CALL DELAY
                INC DI
                MOV AL,BH
                SAR AL,1
                MOV BH,AL
                LOOP DISPLA3
                JMP TEST_BU

DELAY:          PUSH CX
                MOV CX,3FFH
DELAY1:         NOP
                NOP
                NOP
```

```
                NOP
                LOOP DELAY1
                POP CX
                RET
CODE ENDS

STACK    SEGMENT 'STACK'
STA      DB   100 DUP(?)
TOP      EQU LENGTH STA
STACK    ENDS

DATA     SEGMENT 'DATA'
TABLE    DB 0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H
DATA     ENDS
         END START
```

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

3.10 实验十八 4x4 矩阵键盘

一、 实验要求

利用 4X4 16 位键盘和一个 7 段 LED 构成简单的输入显示系统，实现键盘输入和 LED 数码管显示实验。

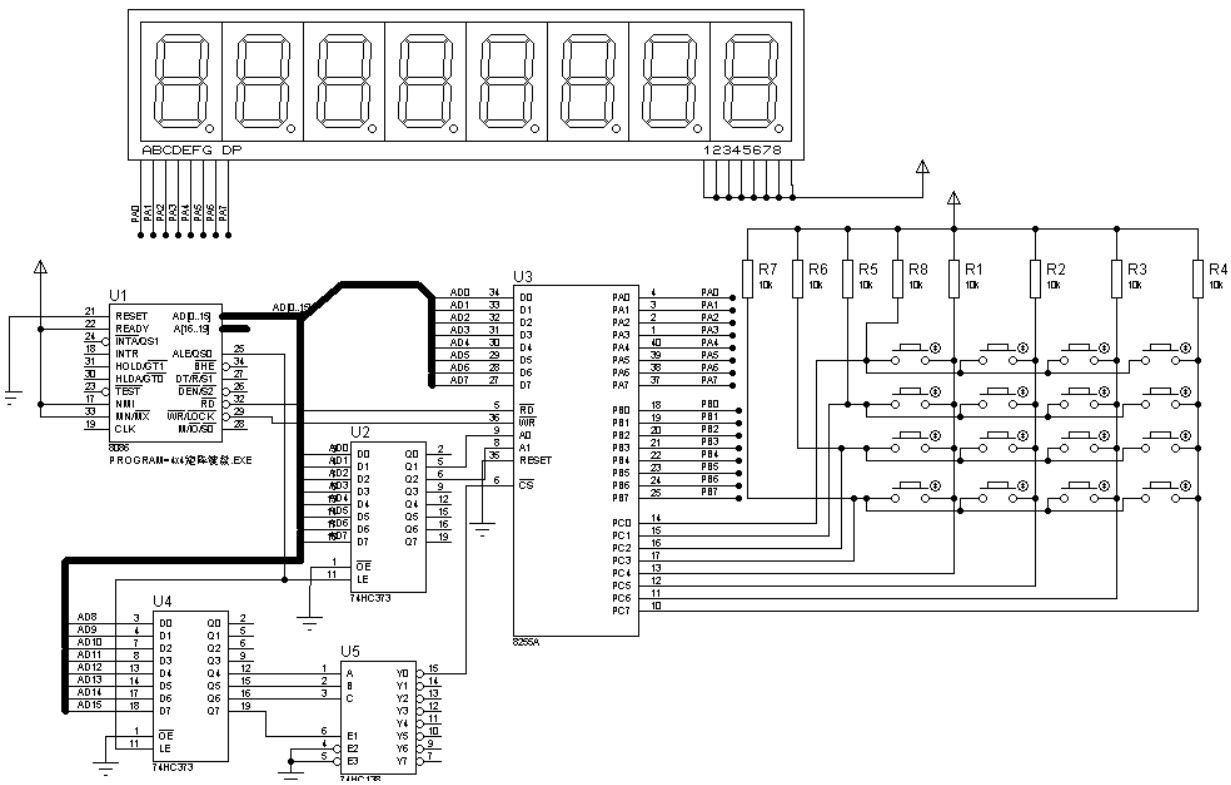
二、 实验目的

1、理解矩阵键盘扫描的原理；

2、掌握矩阵键盘与 8255 接口的编程方法。

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	08000H-08FFFH

COM_1—COM_8	+5V
COM_A—COM_DP	PA0—PA7
R0—R3	PC0—PC3
C0—C3	PC4—PC7

四、 实验说明

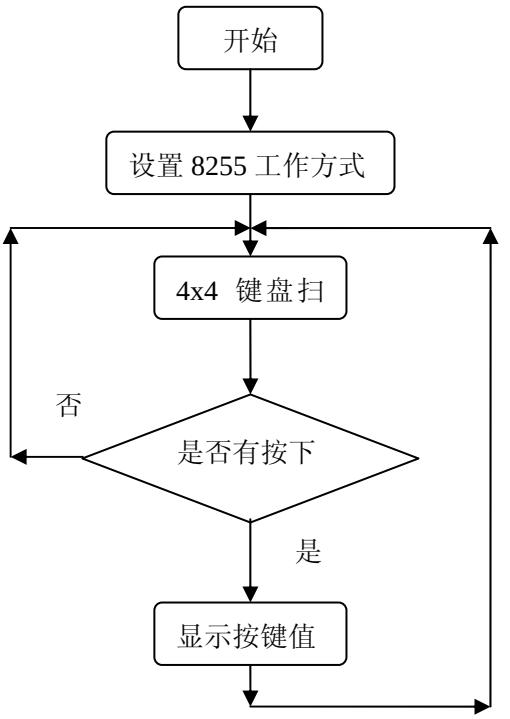
1、主要知识点概述：

本实验阐述了键盘扫描原理，过程如下：首先扫描键盘，判断是否有键按下，再确定是哪一个键，计算键值，输出显示。

2、实验效果说明：

以数码管显示键盘的作用。点击相应按键显示相应的键值。

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“4x4 矩阵键盘_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。

参考程序：

```
CODE      SEGMENT 'CODE'
ASSUME CS:CODE,DS:DATA
IOCON   EQU 8006H
```

```
IOA     EQU 8000H
IOB     EQU 8002H
IOC     EQU 8004H

START:      MOV AX, DATA
            MOV DS, AX

            LEA DI,TABLE

            MOV AL,88H
            MOV DX,IOCON
            OUT DX,AL
KEY4X4:
            MOV BX,0
            MOV DX,IOC
            MOV AL, 0EH
            OUT DX, AL

            IN   AL,DX
            MOV DX,IOC
            IN   AL,DX
            MOV DX,IOC
            IN   AL,DX

            OR   AL,0FH
            CMP AL,0FFH ; 0EFH,0DFH,0BFH,7FH
            JNE K_N_1 ;不等于转移
            INC BX

            MOV DX,IOC
            MOV AL, 0DH
            OUT DX, AL

            IN   AL,DX
            MOV DX,IOC
            IN   AL,DX
            MOV DX,IOC
            IN   AL,DX

            OR   AL,0FH
            CMP AL,0FFH ; 0EFH,0DFH,0BFH,7FH
```

```
                JNE K_N_1 ;不等于转移
                INC BX
                MOV DX,IOC
                MOV AL, 0BH
                OUT DX, AL
                IN   AL,DX
                MOV DX,IOC
                IN   AL,DX
                MOV DX,IOC
                IN   AL,DX
                OR   AL,0FH
                CMP AL,0FFH ; 0EFH,0DFH,0BFH,7FH
                JNE K_N_1 ;不等于转移
                INC BX
                MOV DX,IOC
                MOV AL, 07H
                OUT DX, AL
                IN   AL,DX
                MOV DX,IOC
                IN   AL,DX
                MOV DX,IOC
                IN   AL,DX
                OR   AL,0FH
                CMP AL,0FFH ; 0EFH,0DFH,0BFH,7FH
                JNE K_N_1 ;不等于转移
                JMP KEY4X4

K_N_1:          CMP AL,0EFH
                JNE K_N_2
                MOV AL,0
                JMP K_N

K_N_2:          CMP AL,0DFH
                JNE K_N_3
                MOV AL,1
                JMP K_N

K_N_3:          CMP AL,0BFH
                JNE K_N_4
                MOV AL,2
                JMP K_N
```

```
K_N_4:      CMP AL,7FH
            JNE K_N
            MOV AL,3

K_N:        MOV CL,2
            SHL BL,CL    ;BH X 2
            ADD AL,BL
            MOV BL,0
            MOV BL,AL
            MOV AL,[DI+BX]
            MOV DX,IOA
            OUT DX,AL
            JMP KEY4X4

CODE ENDS
DATA    SEGMENT 'DATA'
TABLE   DB
0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H,88H,83H,0C6H,0A1H,86H,8EH ;0-F
DATA    ENDS
            END START
```

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

3.11 实验十九 直流电机控制实验

一、 实验要求

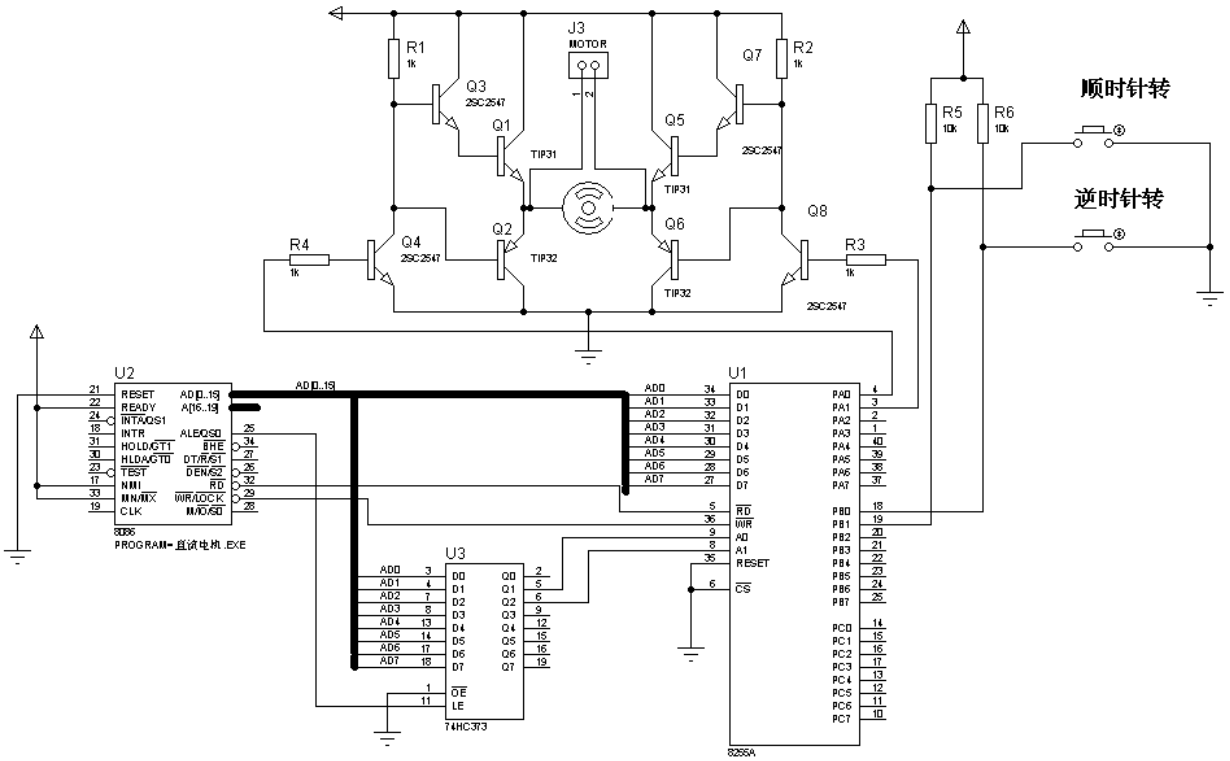
采用 8255 的 2 个 IO 口来控制直流电机，编写程序，其中一个 IO 口使用脉宽调制（PWM）对电机转速进行控制,另一个 IO 口控制电机的转动方向。

二、 实验目的

了解控制直流电机的基本原理；掌握控制直流电机转动的编程方法；了解脉宽调制的原理。

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
8255 CS	08000H-08FFFH
PB0--PB1	K1—K2
PWM	PA0
DIR	PA1

四、 实验说明

在实验中，我们改变 PWM 的占空比，然后查看对电机速度的影响。

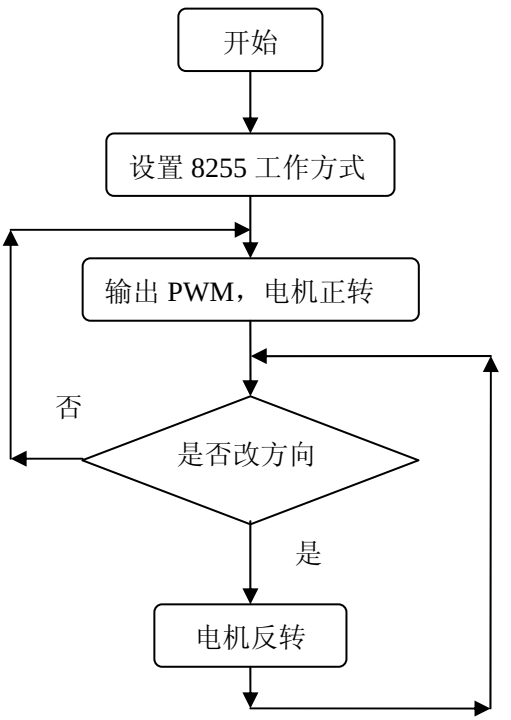
1、主要知识点概述：

本实验用到了两个主要知识点是：达林顿管的应用、PWM 波的产生方法。

2、实验效果说明：

通过两个按键改变直流电机的正反转。

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“直流电机_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。

参考程序：

```
CODE      SEGMENT 'CODE'
          ASSUME CS:CODE,SS:STACK,DS:DATA
IOCON         EQU 8006H
IOA           EQU 8000H
IOB           EQU 8002H
IOC           EQU 8004H
```

```
START:
        MOV AX, DATA
        MOV DS, AX
        MOV AX, STACK
        MOV SS, AX
        MOV AX, TOP
        MOV SP, AX

TEST_BU: MOV AL,82H
        MOV DX,IOCON
        OUT DX,AL
        NOP
        NOP
        NOP

MOT1:   MOV DX,IOA
        MOV AL,0FEH
        OUT DX,AL
        CALL DELAY
        MOV DX,IOB
        IN   AL,DX
        TEST AL,02H
        JE MOT2
        MOV DX,IOA
        MOV AL,0FFH
        OUT DX,AL
        CALL DELAY
        JMP MOT1

MOT2:   MOV DX,IOA
        MOV AL,0FDH
        OUT DX,AL
        CALL DELAY
        MOV DX,IOB
        IN   AL,DX
        TEST AL,01H
        JE MOT1
        MOV DX,IOA
        MOV AL,0FFH
        OUT DX,AL
        CALL DELAY
```

```
                JMP MOT2

DELAY:          PUSH CX
                MOV CX,0FH
DELAY1:         NOP
                NOP
                NOP
                NOP
                LOOP DELAY1
                POP CX
                RET

CODE ENDS

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

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

3.12 实验二十 步进电机控制

一、 实验要求

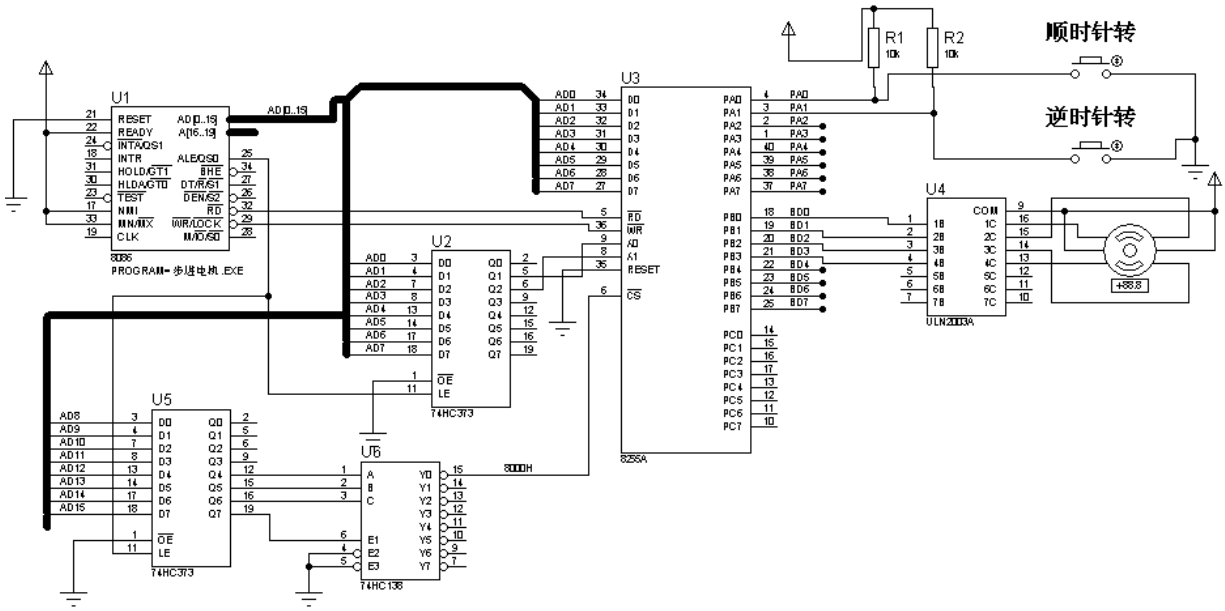
利用 8255 实现对步进电机的控制，编写程序，用四路 IO 口实现环形脉冲的分配，控制步进电机按固定方向连续转动。同时，要求按下 A 键时，控制步进电机正转；按下 B 键盘时，控制步进电机反转。

二、 实验目的

了解步进电机控制的基本原理；掌握控制步进电机转动的编程方法。

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

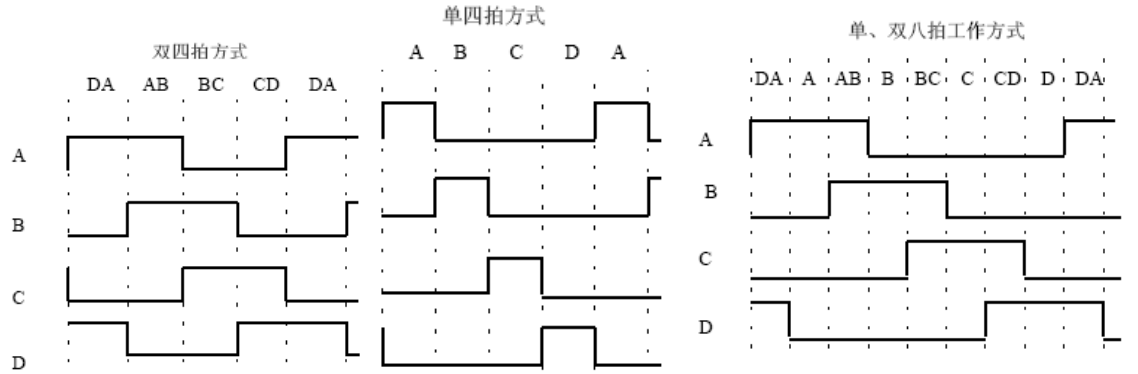
接线孔 1	接线孔 2
8255 CS	08000H-08FFFH
PA0--PA1	K1—K2
PB0—PB3	B1—B4

四、 实验说明

步进电机驱动原理是通过对每组线圈中的电流的顺序切换来使电机作步进式旋转。切换是通

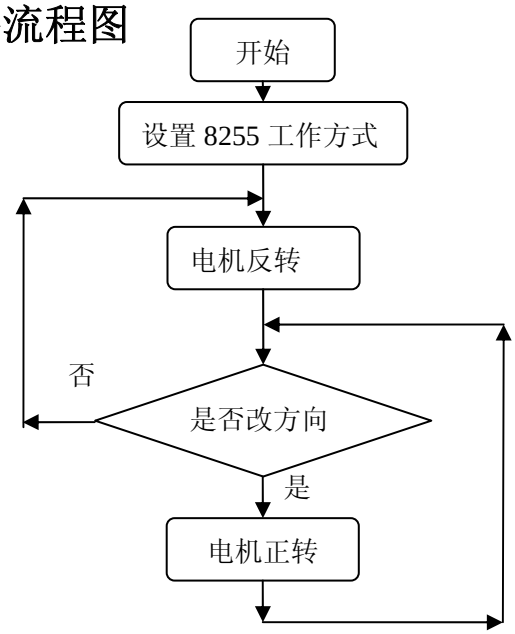
过单片机输出脉冲信号来实现的。所以调节脉冲信号的频率就可以改变步进电机的转速，改变各相脉冲的先后顺序，就可以改变电机的转向。步进电机的转速应由慢到快逐步加速。

电机驱动方式可以采用双四拍（$AB \rightarrow BC \rightarrow CD \rightarrow DA \rightarrow AB$）方式，也可以采用单四拍（$A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$）方式。为了旋转平稳，还可以采用单、双八拍方式（$A \rightarrow AB \rightarrow B \rightarrow BC \rightarrow C \rightarrow CD \rightarrow D \rightarrow DA \rightarrow A$）。各种工作方式的时序图如下：（高电平有效）：



上图中示意的脉冲信号是高电平有效，但实际控制时公共端是接在 VCC 上，所以实际控制脉冲是低电平有效。

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“步进电机_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。

参考程序：

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

```
IOCON   EQU 8006H
IOA     EQU 8000H
IOB     EQU 8002H
IOC     EQU 8004H
START:
        MOV AX, DATA
        MOV DS, AX
        MOV AX, STACK
        MOV SS, AX
        MOV AX, TOP
        MOV SP, AX
        MOV AL,90H
        MOV DX,IOCON
        OUT DX,AL
        NOP
        MOV AL,0FFH
MOT2:   MOV CX,08H
        LEA DI,STR2
IOLED2: MOV AL,[DI]
        MOV DX,IOB
        OUT DX,AL
        MOV DX,IOA
        IN  AL,DX
        TEST AL,01H
        JE MOT1   ; 为 0
        INC DI
        CALL DELAY
        LOOP IOLED2
        JMP MOT2

MOT1:   MOV CX,08H
        LEA DI,STR1
IOLED1: MOV AL,[DI]
        MOV DX,IOB
        OUT DX,AL
        MOV DX,IOA
        IN  AL,DX
        TEST AL,02H
        JE MOT2   ; 为 0
        INC DI
        CALL DELAY
```

```
                LOOP IOLED1
                JMP MOT1

DELAY:          PUSH CX
                MOV CX,0D1H
DELAY1:         NOP
                NOP
                NOP
                NOP
                LOOP DELAY1
                POP CX
                RET
CODE ENDS

STACK     SEGMENT 'STACK'
STA       DB    100 DUP(?)
TOP       EQU LENGTH STA
STACK     ENDS
DATA      SEGMENT 'DATA'
STR1      DB 02H,06H,04H,0CH,08H,09H,01H,03H ;控制数据表
STR2      DB 03H,01H,09H,08H,0CH,04H,06H,02H ;控制数据表
DATA      ENDS
                END START
```

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

3.13 实验二十一 16x16 点阵显示实验

一、 实验要求

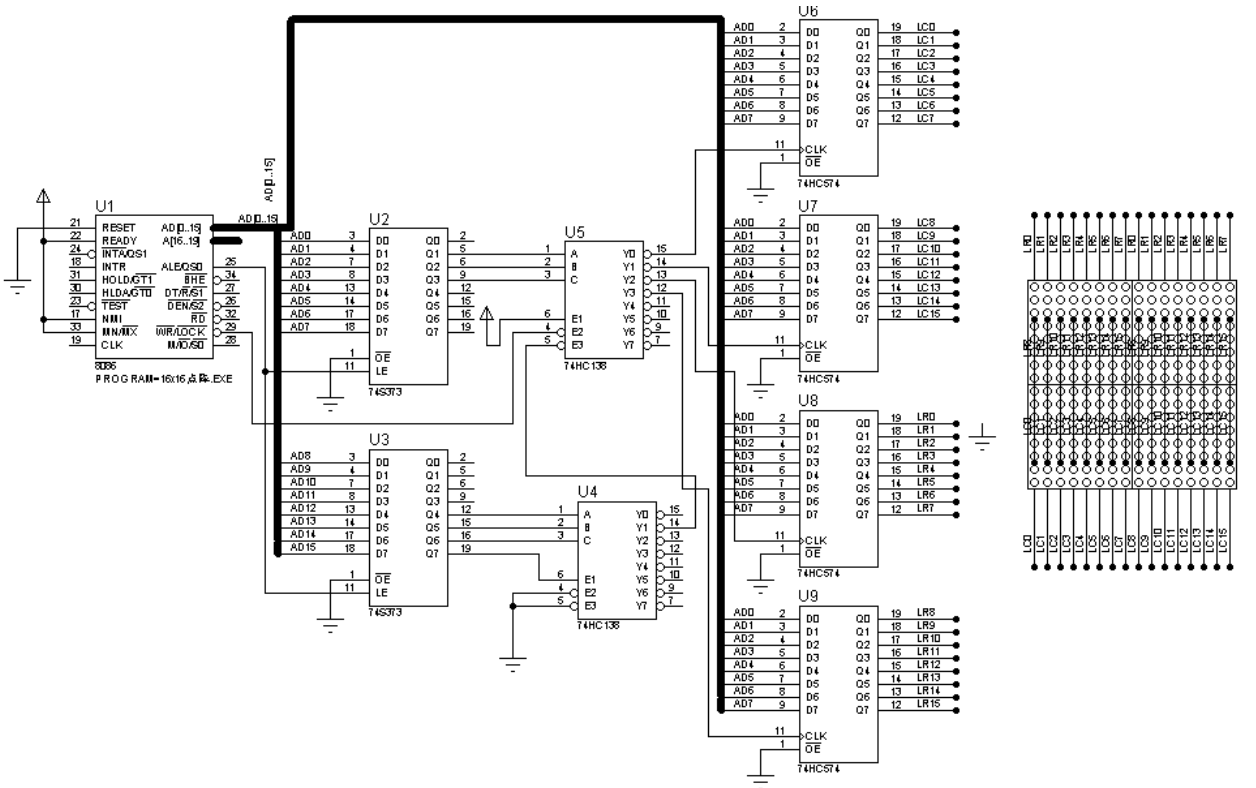
利用 8086 及 74HC574、74HC373、74HC138、16x16LED 屏，实现汉字的显示。

二、 实验目的

了解阵列 LED 扫描显示的原理。

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

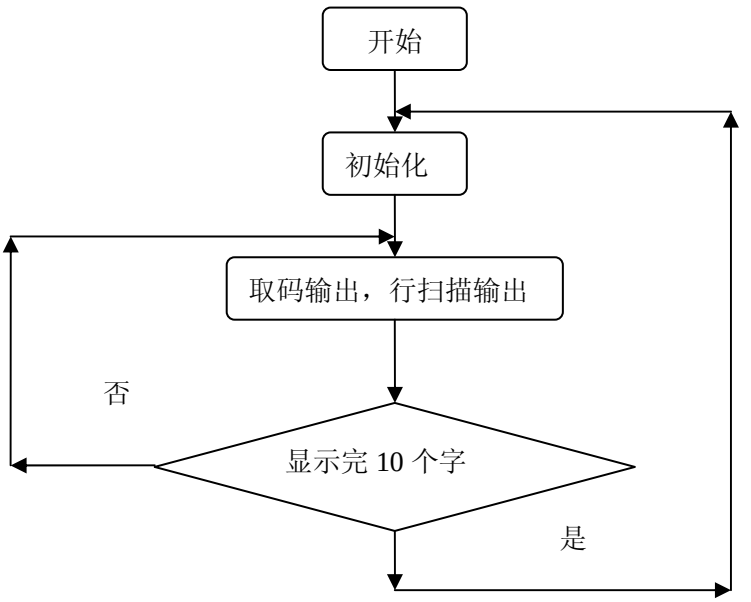
接线孔 1	接线孔 2
8255 CS	08000H-08FFFH

四、 实验说明

16X16 点阵共需要 256 个发光二极管组成，且每个发光二极管是放置在行线和列线的交叉点

上，当对应的某一列置 0 电平，某一行置 1 电平时，该点亮。

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“16x16 点阵_STM.DSN”;

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。

参考程序：

```
;LED16x16 的片选信号接主板 CS3,其它数据信号,地址信号,写信号接主板的相应信号.
RowLow     EQU    9004H;     行低八位地址
RowHigh    EQU    9006H;     行高八位地址
ColLow     EQU    9000H;     列低八位地址
ColHigh    EQU    9002H;     列高八位地址

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

START:         MOV      AX, DATA
               MOV      DS, AX
               MOV AX, STACK
               MOV SS, AX
               MOV AX, TOP
               MOV SP, AX
```

```
                MOV     SI, oFFset Font
main:
                MOV     AL, 0
                MOV     DX, RowLow
                OUT     DX, AL
                MOV     DX, RowHigh
                OUT     DX, AL

                MOV     AL, 0FFh
                MOV     DX, ColLow
                OUT     DX, AL
                MOV     DX, ColHigh
                OUT     DX, AL

n123:           MOV     CharIndex, 0
nextchar:
                MOV     DelayCNT, 10
LOOP1:          MOV     BitMask, 1
                MOV     ColCNT, 16
                MOV     BX, CharIndex
                MOV     AX, 32
                mul     BX
                MOV     BX, AX
nextrow:        MOV     AL, 0FFH
                MOV     DX, RowLow
                OUT     DX, AL
                MOV     DX, RowHigh
                OUT     DX, AL
                MOV     AX, [SI+BX]
                MOV     DX, ColLow
                OUT     DX, AL
                MOV     DX, ColHigh
                MOV     AL, ah
                OUT     DX, AL

                INC     BX
                INC     BX

                MOV     AX, BitMask
                MOV     DX, RowLow
                NOT     AL
```

```
                OUT     DX, AL
                MOV     DX, RowHigh
                MOV     AL, ah
                NOT AL
                OUT     DX, AL

                MOV     AX, BitMask
                ROL     AX, 1
                MOV     BitMask, AX

                NOP
                DEC    ColCNT
                JNZ    nextrow

                DEC    DelayCNT
                JNZ    LOOP1

                INC    CharIndex              ;指向下个汉字
                MOV    AX, CharIndex
                CMP    AX, 10
                JNZ    nextchar
                JMP    n123
delay :         PUSH   CX
                MOV     CX,1
delayl:         LOOP   delayl
                POP     CX
                RET

      CODE     ENDS
DATA     SEGMENT
Font:
;广
DB 080H, 000H, 000H, 001H, 0FCH, 07FH, 004H, 000H
DB 004H, 000H, 004H, 000H, 004H, 000H, 004H, 000H
DB 004H, 000H, 004H, 000H, 004H, 000H, 004H, 000H
DB 002H, 000H, 002H, 000H, 001H, 000H, 000H, 000H
;州
DB 010H, 020H, 010H, 021H, 010H, 021H, 010H, 021H
DB 010H, 023H, 032H, 025H, 052H, 025H, 052H, 029H
DB 011H, 029H, 010H, 021H, 010H, 021H, 008H, 021H
DB 008H, 021H, 004H, 021H, 004H, 021H, 002H, 020H
```

```
;风
DB 000H, 000H, 0F8H, 01FH, 008H, 010H, 008H, 012H
DB 028H, 016H, 048H, 012H, 088H, 012H, 008H, 011H
DB 008H, 011H, 088H, 012H, 048H, 056H, 024H, 054H
DB 014H, 064H, 002H, 060H, 001H, 040H, 000H, 000H
;标
DB 008H, 000H, 088H, 03FH, 008H, 000H, 008H, 000H
DB 03FH, 000H, 0C8H, 07FH, 01CH, 004H, 02CH, 004H
DB 0AAH, 014H, 08AH, 024H, 049H, 064H, 028H, 044H
DB 008H, 044H, 008H, 004H, 008H, 005H, 008H, 002H
;电
DB 080H, 000H, 080H, 000H, 080H, 000H, 0FCH, 01FH
DB 084H, 010H, 084H, 010H, 0FCH, 01FH, 084H, 010H
DB 084H, 010H, 084H, 010H, 0FCH, 01FH, 084H, 010H
DB 080H, 040H, 080H, 040H, 000H, 07FH, 000H, 000H
;子
DB 000H, 000H, 0FCH, 00FH, 000H, 004H, 000H, 002H
DB 000H, 001H, 080H, 000H, 080H, 000H, 080H, 020H
DB 0FFH, 07FH, 080H, 000H, 080H, 000H, 080H, 000H
DB 080H, 000H, 080H, 000H, 0A0H, 000H, 040H, 000H
;有
DB 080H, 000H, 080H, 000H, 0FEH, 07FH, 040H, 000H
DB 020H, 000H, 0F0H, 00FH, 018H, 008H, 014H, 008H
DB 0F2H, 00FH, 011H, 008H, 010H, 008H, 0F0H, 00FH
DB 010H, 008H, 010H, 009H, 010H, 00EH, 010H, 004H
;限
DB 000H, 000H, 0DFH, 01FH, 049H, 010H, 0C9H, 01FH
DB 045H, 010H, 045H, 010H, 0C9H, 01FH, 051H, 001H
DB 051H, 012H, 055H, 00AH, 049H, 004H, 041H, 004H
DB 041H, 008H, 041H, 071H, 0C1H, 020H, 041H, 000H
;公
DB 000H, 000H, 020H, 002H, 060H, 002H, 020H, 002H
DB 010H, 004H, 010H, 008H, 008H, 018H, 044H, 070H
DB 0C2H, 020H, 040H, 000H, 020H, 004H, 010H, 008H
DB 088H, 01FH, 0FCH, 018H, 008H, 008H, 000H, 000H
;司
DB 000H, 000H, 0FCH, 03FH, 000H, 020H, 000H, 020H
DB 0FEH, 027H, 000H, 020H, 000H, 020H, 0FCH, 023H
DB 004H, 022H, 004H, 022H, 0FCH, 023H, 004H, 022H
DB 004H, 020H, 000H, 028H, 000H, 010H, 000H, 000H
```

```
BitMask      DW       1
CharIndex DW       1
DelayCNT    DW       1
ColCNT       DW       1

DATA     ENDS
STACK     SEGMENT
STA         DB    100 DUP(?)
TOP         EQU LENGTH STA
STACK     ENDS

            END START
```

2、实验板验证

（只有仿真实验）

七、 实验结果和体会

八、 建议

3.14 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

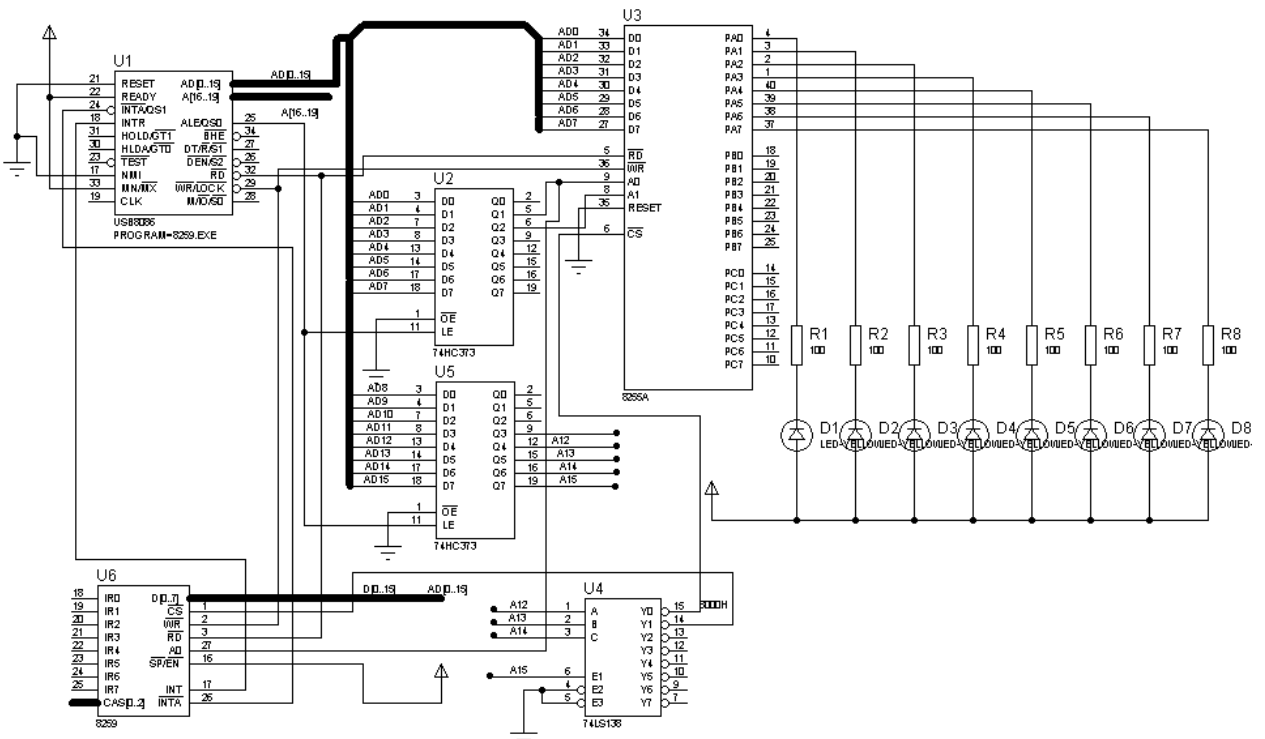
1、学习 8086 与 8259 的连接方法。

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

四、 实验说明

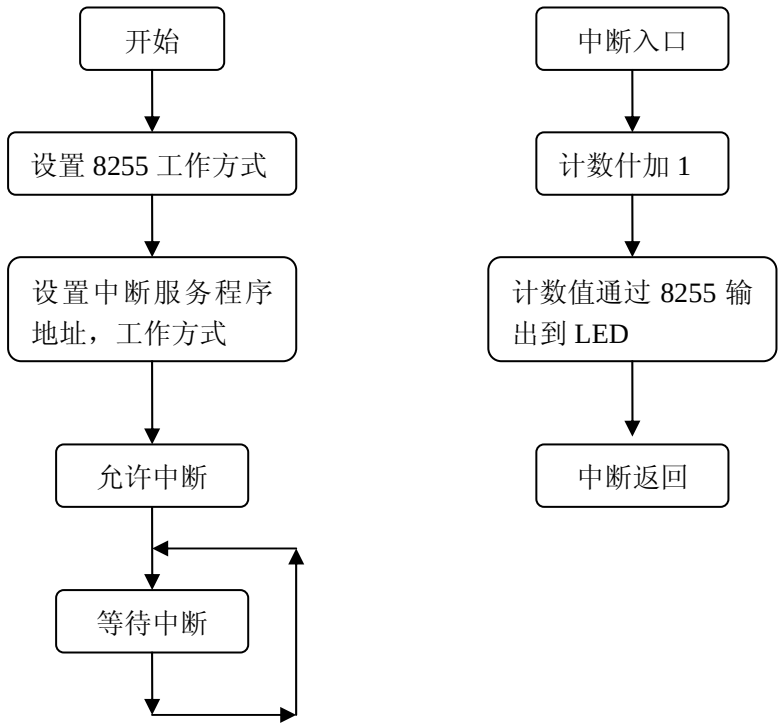
8086 的外部中断必须通过外接中断控制器才可以进行外部中断的处理。在编程时应注意：

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，本实验只响应 INT0 中断，8259 也可以多级连接以响应多个中断源。将单脉冲信号接到 8259 的 INT0 脚。每次中断时，可以看到 LED 显示加 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 16x16 点阵_STM.DSN ；

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。

参考程序：

```
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 00H
                JMP IEnTER
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 IEnTER     ; 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 0
                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

IEnTER:
                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
```

```
CODE ENDS

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

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

3.15 实验二十三 DMA传送实验（8237）

一、 实验要求

利用 8086 控制 8237 可编程 DMA 控制器，实现存储器中两个数据区之间的 DMA 块传送。

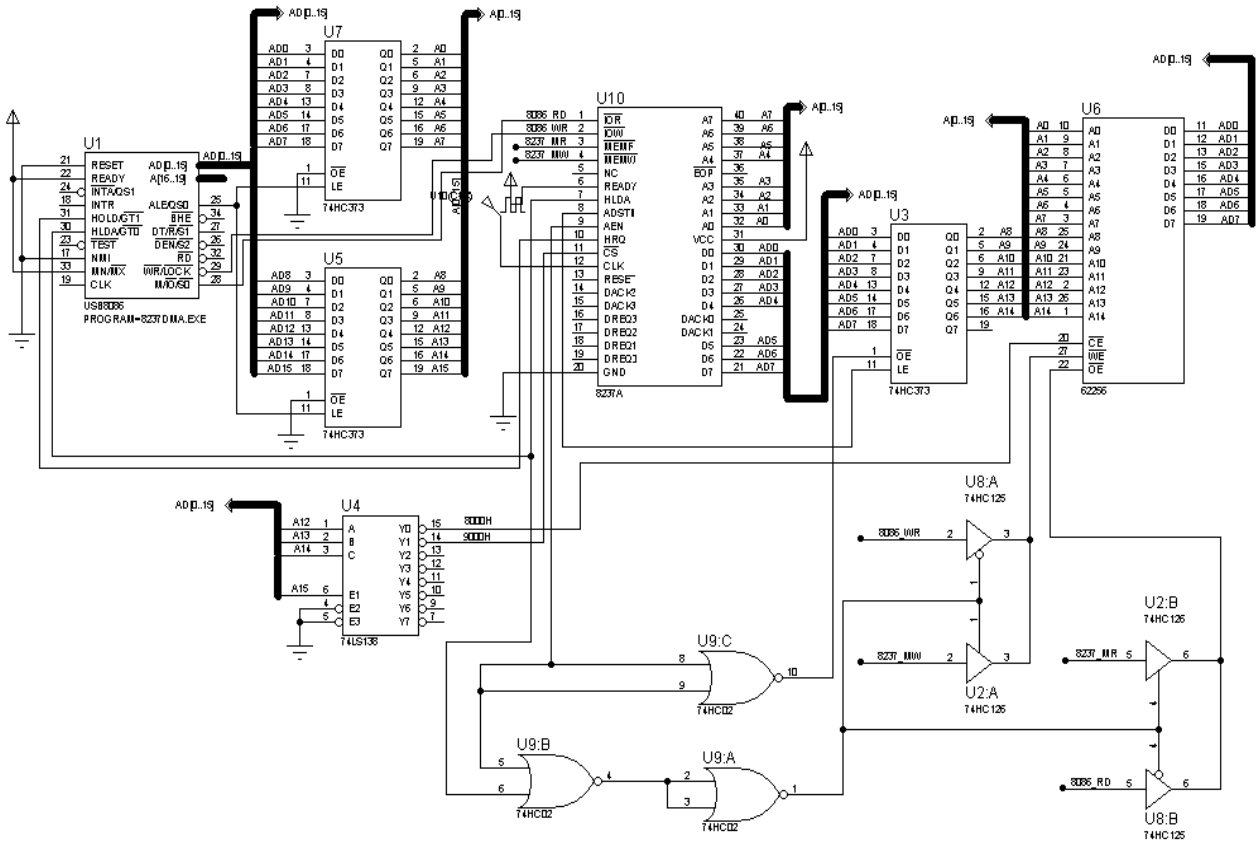
二、 实验目的

1、掌握 8237DMA 控制器。

2、学习 8237DMA 块传输的编程方法。

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
8237 CS	09000H-09FFFH
RAM CS	08000H-08FFFH

CLOCK_OUT	CLOCK_IN
1/2	CLK
HLDA	/HLDA
HRQ	HOLD

四、 实验说明

1、8086 通过外接的 8237 可编程 DMA 控制器实现 DMA 传输。

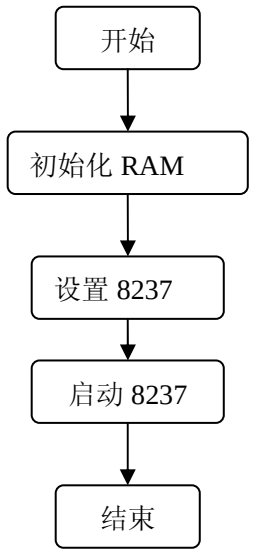
2、首先将存储器 8000H-80ffH 初始化。

3、设置 8237DMA，设定源地址 8000H，设定目标地址 8800H,设定块长度为 100H。

4、启动 8237DMA。

5、8237DMA 工作后 8286 暂停工作，总线由 8237DMA 控制，在 DMA 传输完 100H 个单元后，8237 将控制权还给 8286，CPU 执行 RET 指令。

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　16x16 点阵.DSN ；

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。

参考程序：

```
BLOCKFROM EQU 08000H    ; 块开始地址
BLOCKTO    EQU 08800H    ; 块结束地址
BLOCKSIZE EQU 100H       ; 块大小

LATCHB   EQU    9000H     ; LATCH B
CLEAR_F EQU    900CH     ; F/L  触发器
```

```
CH0_A    EQU    9000H     ; 通道 0 地址
CH0_C    EQU    9001H     ; 通道 0 记数
CH1_A    EQU    9002H     ; 通道 1 地址
CH1_C    EQU    9003H     ; 通道 1 记数
MODE     EQU    900BH     ; 模式 写工作方式
CMMD     EQU    9008H     ; 写命令
STATUS   EQU    9008H     ; 读状态
MASKS    EQU    900FH     ; 屏蔽 四个通道
REQ      EQU    9009H     ; 请求

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

START        MOV AX, DATA
             MOV DS, AX
             MOV AX, STACK
             MOV SS, AX
             MOV AX, TOP
             MOV SP, AX
             CALL   FILLRAM
             CALL   TRANRAM
             JMP    $              ; 打开数据窗口，检查传输结果

FILLRAM:  MOV    BX, BLOCKFROM
          MOV    AX, 10H
          MOV    CX, BLOCKSIZE
FILLLOOP: MOV    [BX], AL
          INC    AL
          INC    BX
          LOOP   FILLLOOP
          RET
TRANRAM:
          MOV    SI, BLOCKFROM
          MOV    DI, BLOCKTO
          MOV    CX, BLOCKSIZE

          MOV    AL, 0
          MOV    DX, LATCHB
          OUT    DX, AL
          MOV    DX, CLEAR_F
          OUT    DX, AL
```

```
        MOV     AX, SI          ; 编程开始地址
        MOV     DX, CH0_A
        OUT     DX, AL
        MOV     AL, AH
        OUT     DX, AL

        MOV     AX, DI          ; 编程结束地址
        MOV     DX, CH1_A
        OUT     DX, AL
        MOV     AL, AH
        OUT     DX, AL

        MOV     AX, CX          ; 编程块长度
        DEC     AX              ; 调整长度
        MOV     DX, CH0_C
        OUT     DX, AL
        MOV     AL, AH
        OUT     DX, AL

        MOV     AL, 88H         ; 编程 DMA 模式
        MOV     DX, MODE
        OUT     DX, AL
        MOV     AL, 85H
        OUT     DX, AL

        MOV     AL, 1           ; 块传输
        MOV     DX, CMMD
        OUT     DX, AL

        MOV     AL, 0EH         ; 通道 0
        MOV     DX, MASKS
        OUT     DX, AL
        MOV     AL, 4
        MOV     DX, REQ
        OUT     DX, AL          ; 开始 DMA 传输
        RET

DELAY:  PUSH    AX
        PUSH    CX
        MOV     AX, 100
```

```
DELAYLOOP:
        MOV     CX, 100
        LOOP    $
        DEC     AX
        JNZ     DELAYLOOP
        POP     CX
        POP     AX
        RET
CODE    ENDS

DATA    SEGMENT
DMA     EQU   00H
DATA    ENDS
STACK   SEGMENT 'STACK'
STA     DB    100 DUP(?)
TOP     EQU LENGTH STA
STACK   ENDS
        END START
```

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

第4章　8086 C语言实验

4.1　说明

本实验系统也支持使用 C 语言编写程序，并进行在线仿真。光盘里面只包含了硬件部分的 C 语言程序例程。

如果用户需要编写其它的例程，请注意以下几点：

1. 必须包含 RTL.ASM 文件到你新建个工程中。
2. 读写 IO 口，请使用以下这两个函数：

```
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;
}
```

3. 编译的方法请看第 2 章第 1 节的内容。

使用 C 语言编写程序进行仿真的过程与使用汇编语言的过程是完全相同的，这里也就不再重复。

第5章 32 位计算机接口技术实验

5.1 实验一 第一个MFC应用程序“Hello,world!”

一、 实验要求

编写出第一个 MFC 应用程序。

二、 实验目的

1、由于本实验箱所有的 32 位计算机接口技术实验都使用 Microsoft Visual Studio 2008 开发环境进行编写，因此在进行硬件实验之前，必须先熟悉 Microsoft Visual Studio 2008 开发环境。

2、熟悉 MFC 程序开发的流程。

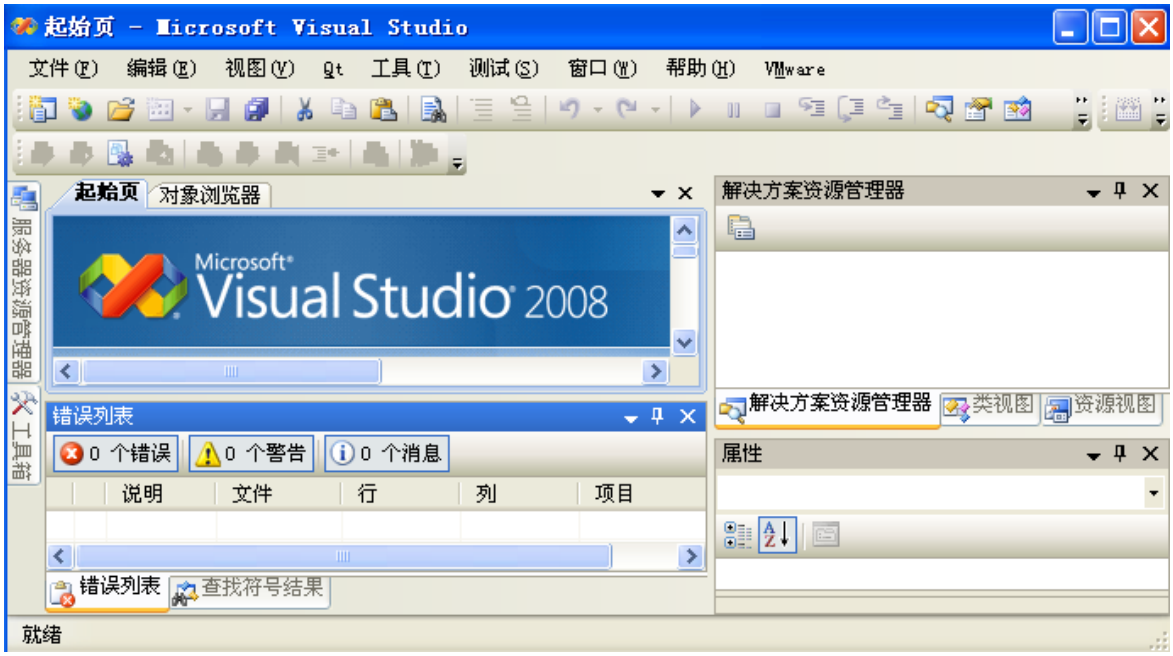
3、熟悉 MFC 程序的调试过程。

三、 实验电路及连线

此实验为纯软件实验，不需要连接电路。

四、 实验程序及说明

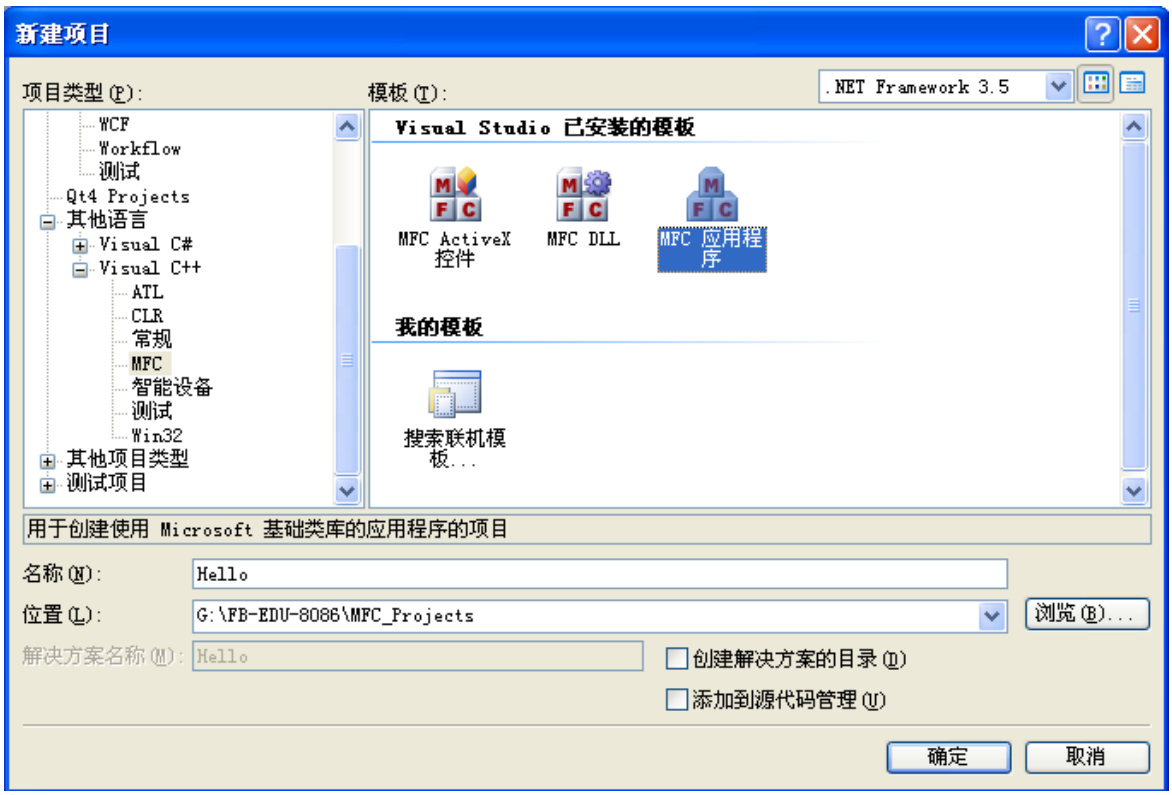
此实验中，我们将使用 Microsoft Visual Studio 2008 进行 MFC 程序的开发，如下图：



Microsoft Visual Studio 2008 开发环境

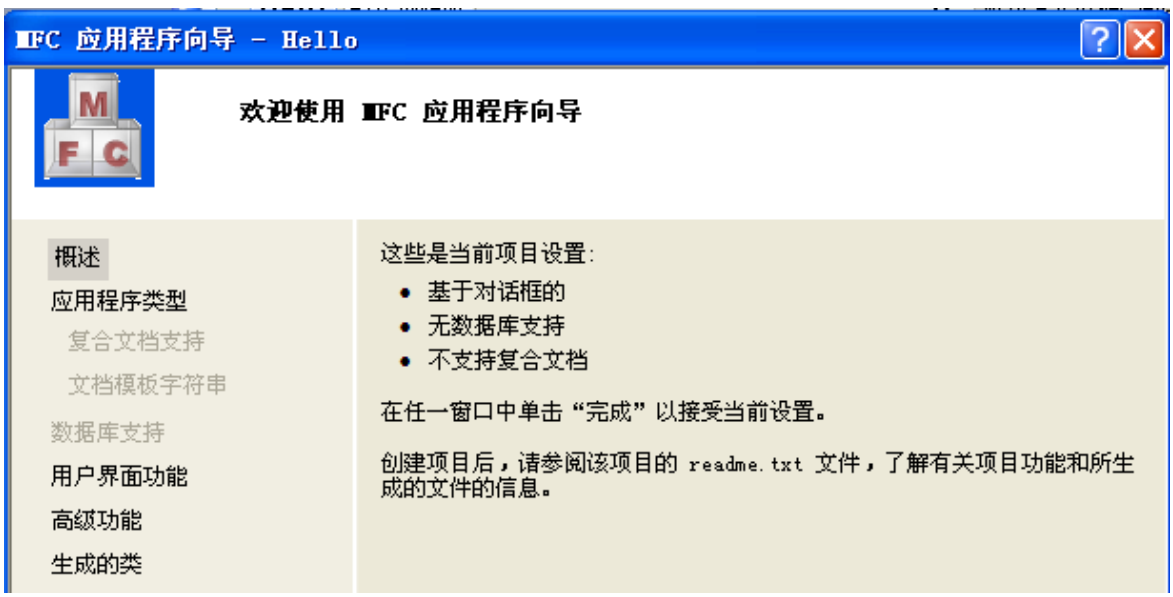
以下是此程序的编写过程，与 Microsoft Visual Studio 2008 相关的知识请参考相关书籍。

1. 打开 Microsoft Visual Studio 2008，新建一个项目。选择 MFC 应用程序，如下图所示，项目命名为 Hello。

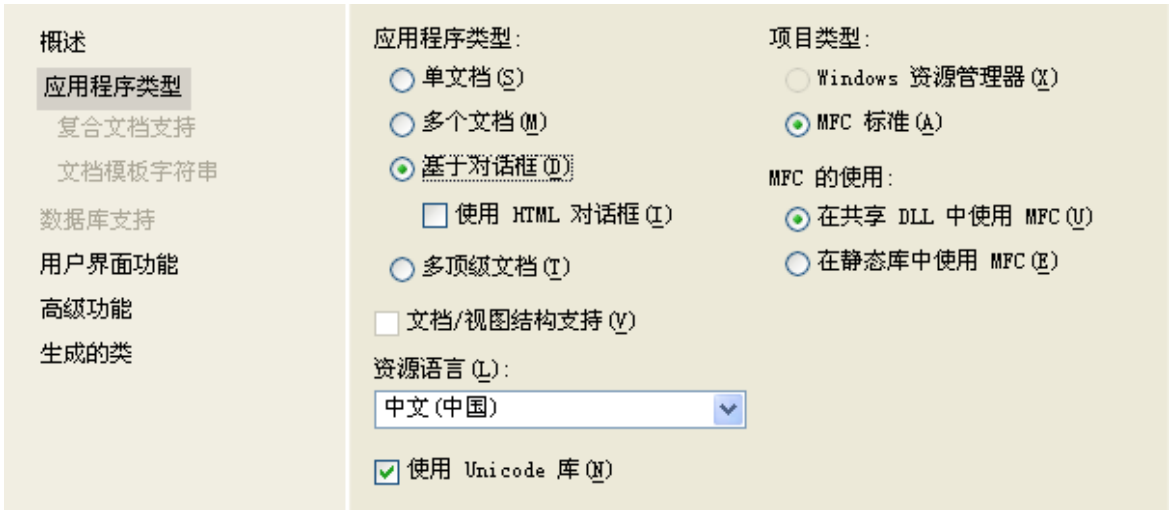


新建 MFC 应用程序项目

2. 进入 MFC 应用程序向导，如下图所示，依照向导，分别设置好以下各项参数，包括应用程序类型，用户界面功能，高级功能，生成的类等。



MFC 应用程序向导



应用程序类型设置

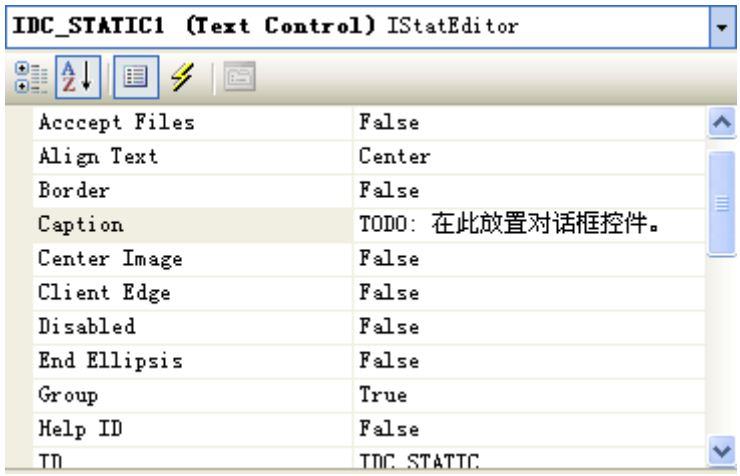
用户界面设置

高级功能设置

生成的类设置

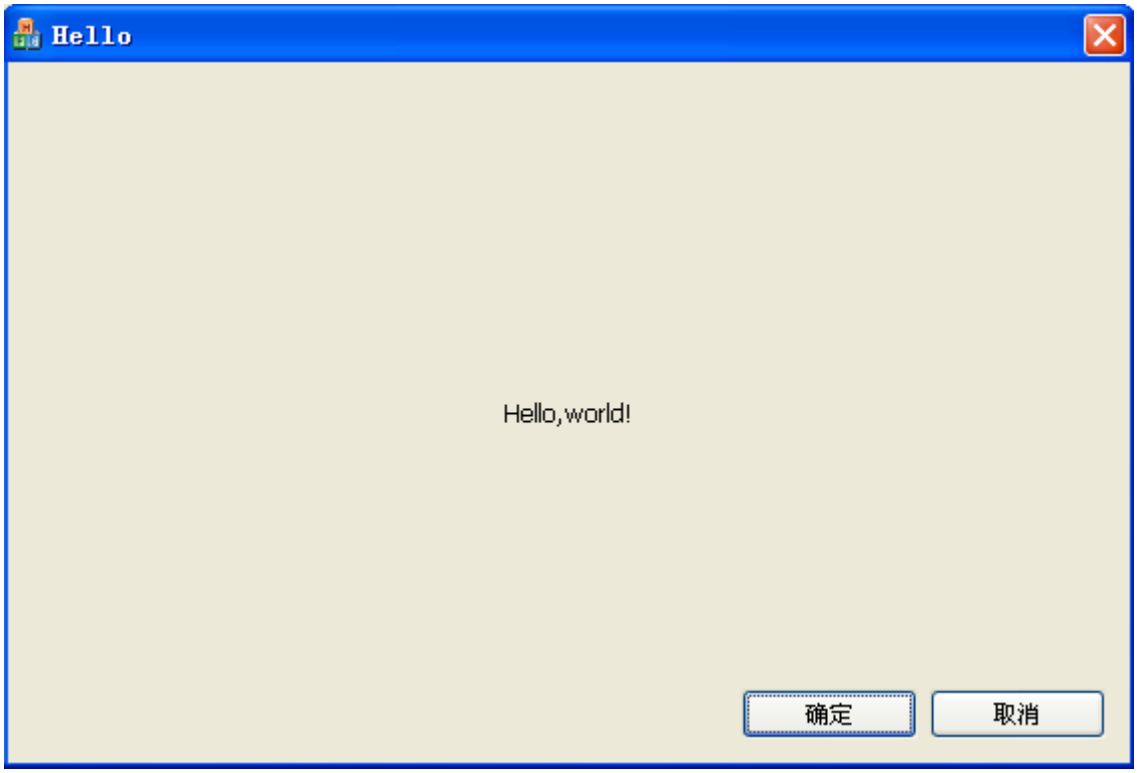
3. 在资源视图里面，打开 IDD_HELLO_DIALOG 对话框资源，如下图所示：

4. 单击“TODO: 在此放置对话框控件。”，选择 IDC_STATIC1 控件，或者从控件列表中直接选择，修改“TODO: 在此放置对话框控件。”为“Hello,world!”



控件属性列表

5. 最后点击编译，运行程序，我们的第一个 MFC 应用程序就诞生啦！由于我们并未编写其它的消息处理程序，这里的“确定”与“取消”两个按钮的处理都使用系统默认的处理流程。



Hello 程序运行中

五、 实验结果和体会

六、 建议

5.2 实验二 8255 简单I/O控制实验

一、 实验要求

通过 8255 简单 I/O 实验来说明如何使用风标电子的 USB8086 API。

二、 实验目的

1、使用实验箱的 8255A 并行接口扩展器件，实现 8 位开关输入和 8 位 LED 输出。

2、熟悉风标电子的 USB8086 API 的使用。

三、 实验电路及连线

电路原理和实验箱硬件连接请参照 8255 并行 I/O 扩展实验章节的内容。

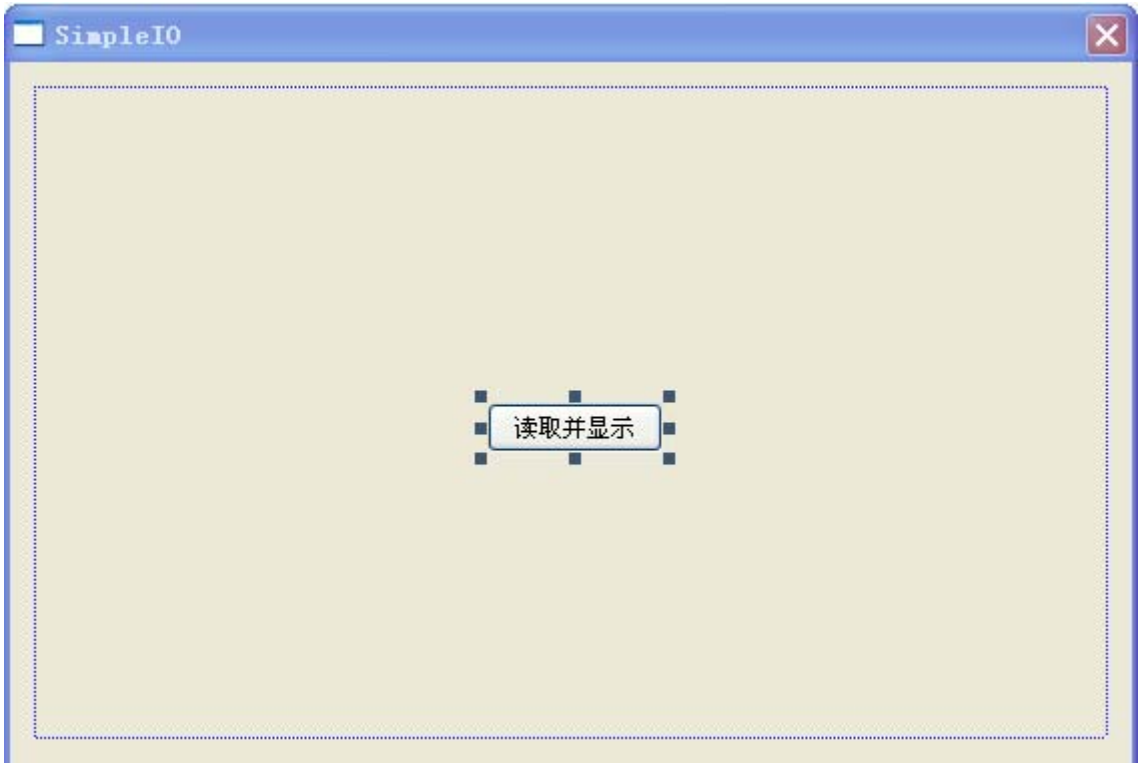
四、 实验程序及说明

1. 使用实验一中的流程，创建一个基于对话框的 MFC 应用程序，程序命名为“SimpleIO”。
2. 在资源视图里面，打开 IDD_SIMPLEIO_DIALOG 对话框资源，如下图所示：

3. 删除“TODO: 在此放置对话框控件。”和“确定”、“取消”按钮控件，从工具箱中拖取一个“Button”控件到对话框中，如下图所示：

选择 Button 控件

4. 修改该 Button 控件的 Caption 属性为“读取并显示”，ID 属性修改为“IDC_BUTTON_RW”。



SimpleIO 软件界面

5. 我们把风标电子 USB8086 API 的头文件加入 SimpleIODlg.h 文件代码中。另外，我们还需添加 CSimpleIODlg 类的解构函数~CSimpleIODlg()并添加一个 protected 类型变量 CUSB8086API* m_usb，这个就是 USB8086API 的接口类，用于控制实验箱上的 USB 接口。
6. 在 CSimpleIODlg.cpp 文件中，我们加入解构函数~CSimpleIODlg()的具体实现，如下：

```
CSimpleIODlg::~CSimpleIODlg()
{
    delete m_usb;
}
```

7. 在 BOOL CSimpleIODlg::OnInitDialog()函数中，创建我们的 USB8086API 类

的实例，如下：

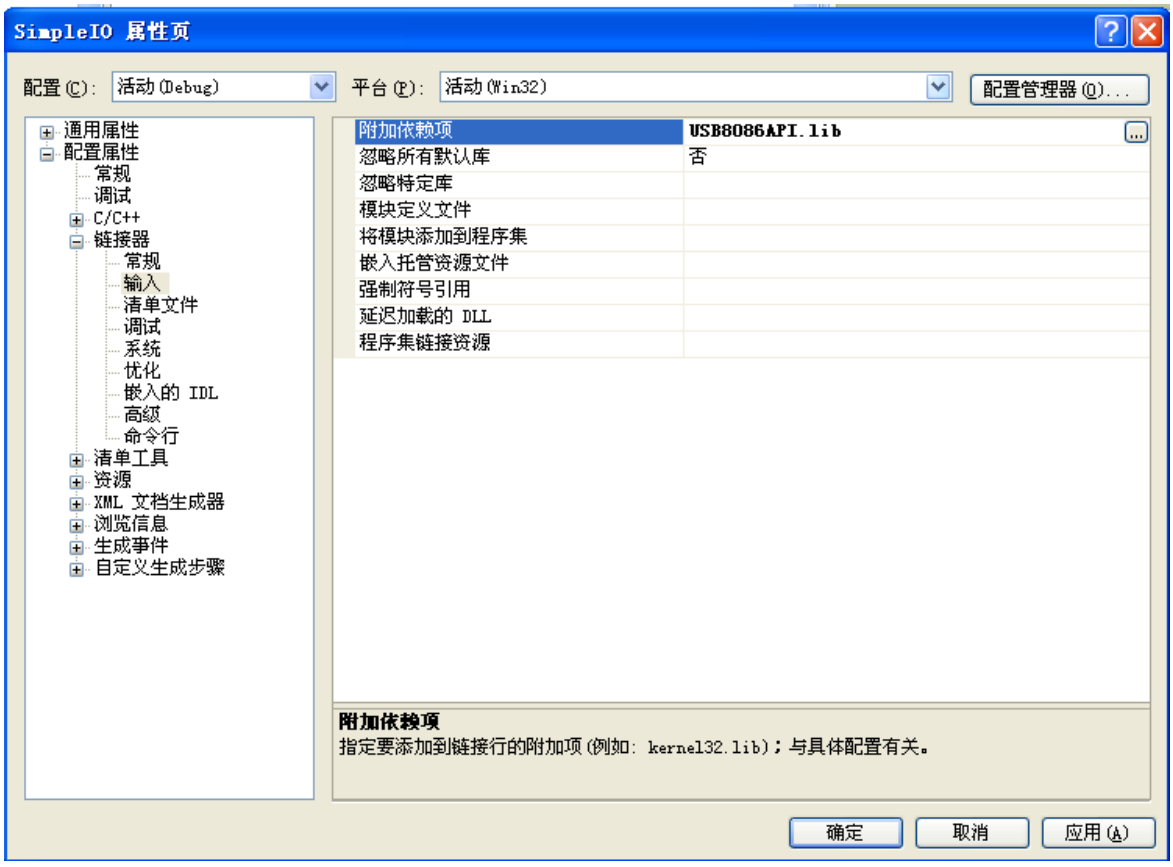
```
m_usb = new CUSB8086API();
```

8. 双击软件对话框中的“读取并显示”按键控件，打开源代码窗口，并自动定位到“void CSimpleIODlg::OnBnClickedButtonRw()”函数，我们在这里对单击事件进行响应。

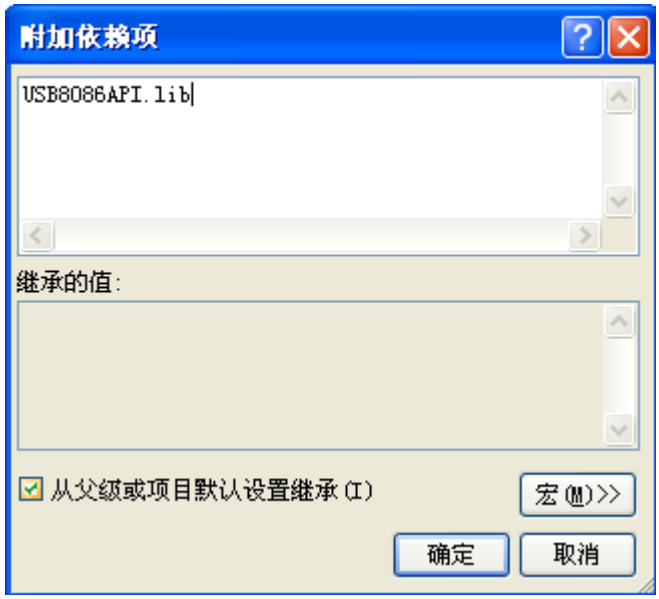
代码如下：

```
void CSimpleIODlg::OnBnClickedButtonRw()
{
    // TODO: 在此添加控件通知处理程序代码

    #define I8255_CON   0x8006
    #define I8255_IOA   0x8000
    #define I8255_IOB   0x8002
    #define I8255_IOC   0x8004

    int tmp;

    // 打开USB8086设备,返回0值表示成功，如果不为0，则出错。
    if(this->m_usb->open() == 0) {

        // 写控制字为x90
        this->m_usb->write(I8255_CON, 0x90);

        // 读PA口，PA口接了8位独立按键
        tmp = this->m_usb->read(I8255_IOA);

        // 写PB口，PB口接了8位LED
        this->m_usb->write(I8255_IOB, tmp);

        // 关闭USB8086设备
        this->m_usb->close();
    } else {
        AfxMessageBox(_T("无法连接USB8086设备！"));
    }
}
```

9. 程序的编写基本完成，我们还需要设置编译链接和运行的项目属性。
10. 在 SimpleIO 属性页中设置附加依赖项“USB8086API.lib”。



设置附加依赖项



添加附加依赖项

11. 我们既可以把 USB8086API 目录下的三个文件：USB8086API.h、USB8086API.lib 和 USB8086API.DLL 拷贝到工程文件目录下，也通过设置项目属性的方法，指定 USB8086API 目录，具体的设置可以参照光盘中的例程。

12. 编译并运行程序。

SimpleIO 程序运行中

五、 实验结果和体会

六、 建议

5.3 实验三 数码管动态扫描实验

5.4 实验四 步进电机驱动实验

5.5 实验五 0832 DA转换实验

5.6 实验六 0809 AD转换实验

5.7 实验七 8253 定时器/计数器实验

5.8 实验八 LCD1602 液晶显示实验

5.9 实验九 LCD12864 液晶显示实验