单片机原理及应用

实验指导书

罗钧　付丽 编

重庆大学光电工程学院

2012 年 4 月

目　录

实验一　单片机监控程序实验（3 学时）. 2

附 1.1：LAB2000P 实验仪 7

附 1.2：验证实验程序 8

附 1.3：KEIL 软件的使用步骤参考 15

实验二　A/D 转换实验（3 学时）. 19

附 2.1：验证实验程序 22

实验三　D/A 转换实验（2 学时）. 23

附 3.1：DA 转换实验程序 25

实验一　单片机监控程序实验（3 学时）

实验预习要求：

1．参考附1.3学习Keil软件的使用。

2．熟悉键盘和显示器接口及工作原理。

3．根据实验原理，读懂验证实验程序，写出设计性实验汇编程序。

4．完成下面的思考题：

（1）从附1.2监控程序可以看出：“8”字循环程序中，六位数码管显示数据的段码存放在单片机存储器哪些位置？

（2）参考图1.1A与监控程序，若键盘上某个键被按下，单片机怎样判断该键被按下？

一、实验目的

1．掌握 8031 系统中键盘和显示器的接口方法。

2．掌握键盘扫描和 LED 八段码显示器的工作原理。

3．掌握对单片机 IO 口的控制编程。

二、 实验器材

PC 机一台，Lab2000P 教学实验仪一台，导线数根。

三、实验内容

1．验证性实验

利用实验系统，把按键输入的键码在八位数码管上显示出来。并实现“8”字循环显示、8255 输出方波功能。

2．设计性实验

用 Lab2000P 教学实验系统（见附 2 图），使单片机 P1 口任意一位产生一矩形波（周期可自己设定），设计的程序加到验证实验程序中，存放在单片机程序空间 0280h 开始位置。

程序运行时，把产生矩形波的 P1 口（P1.0~P1.7 任意一位）接到示波器或者用导线连接到“逻辑笔”模块的“输入”，观察 P1 口输出的状态。

四、实验原理

1． 实验仪器简介

Lab2000P实验仪提供了多个模块，本次实验使用的主要模块为：单片机8031模块、8255模块。4×6键盘模块，六位LED显示模块。

2．键盘扫描显示原理

键盘与六位 LED 显示器连接电路图见图 1.1。

（1）芯片介绍

74HC245：高速 CMOS 型 8 位双向总线收发器（三态）。主要用于数据总线的同步双向通信，起总线隔离驱动作用。

74HC374：为八 D 触发器集成芯片，电路中起锁存、驱动作用。

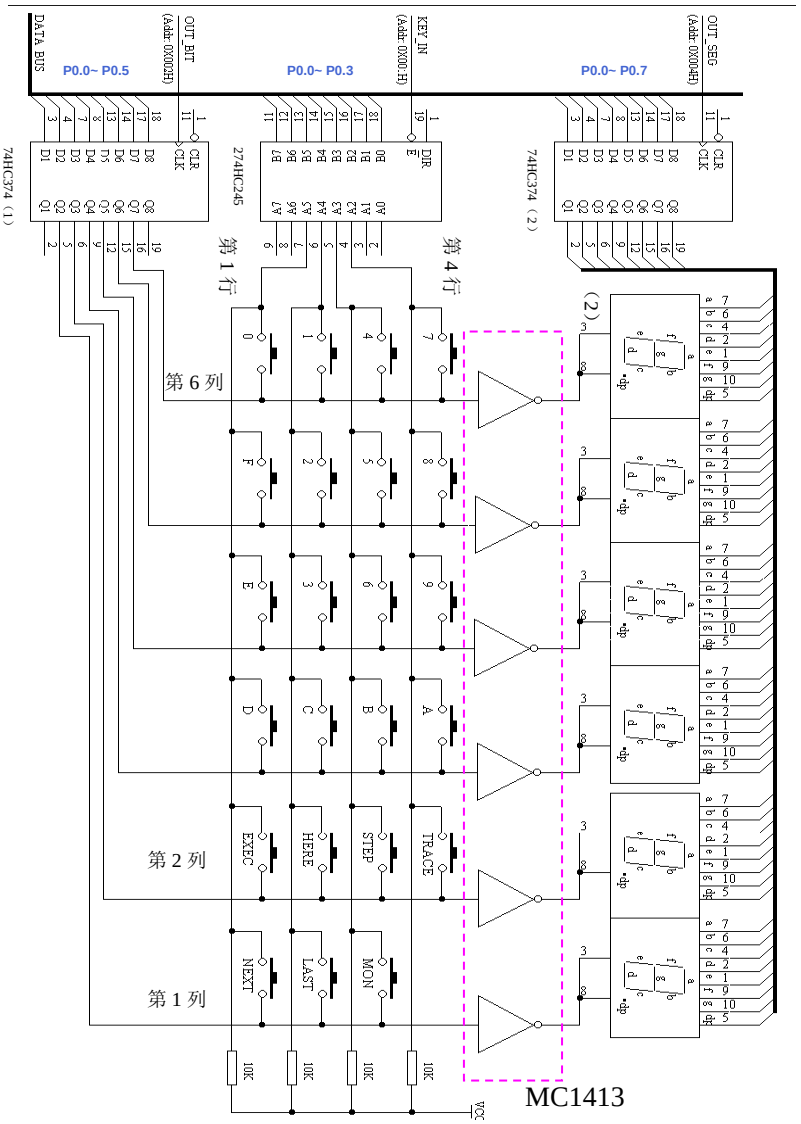


图 1.1(A) 键盘及 LED 显示电路

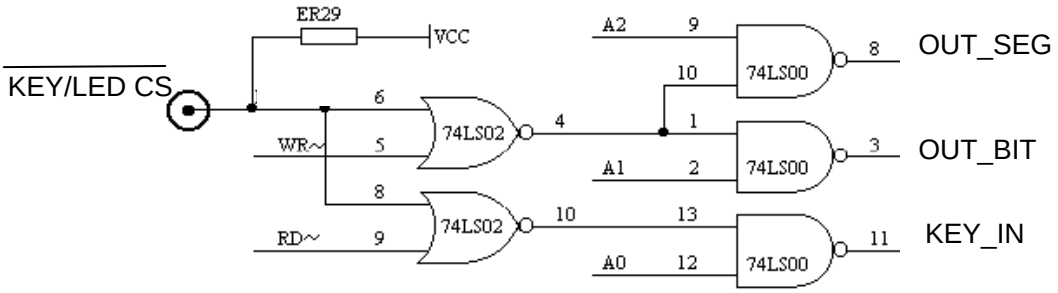


图 1.1(B) 键盘和 LED 显示的地址译码

图 1.1 键盘及 LED 显示电路

（2）扫描键盘和 LED 显示原理

本实验仪的 LED 显示电路和键盘电路如图 1.1（A）。图中 DATA_BUS 为单片机的 P0

口 8 位数据总线。显示控制的位码由芯片 74HC374（1）输出，经反向驱动后（MC1413 为反向驱动芯片），作为 LED 的位选通信号。位选通信号也同时作为键盘列扫描，键盘扫描的行数据从芯片 74HC245 读回单片机，用来判断是否有键被按下，以及按下的是什么键。如果没有键按下，由于上拉电阻的作用，经 245 读回的数据位均为高，如果有键按下，使 74HC374（1）输出的低电平经过按键被接到 245 的端口上，这样单片机从 245 读回的数据位就会是'0'，根据 74HC374（1）输出的列信号和 245 读回的行信号，就可以判断哪个键被按下。LED 显示的段码由 74HC374（2）输出。

键盘和LED显示的地址译码见图1.1(B)，图1.1中，标号相同的引脚连接到一起，图1.1(A)中74HC374(1)、74HC374(2)、74HC245三个芯片的控制引脚CLK、E分别连接到图1.1(B)中标号为OUT_BIT、OUT_SEG、KEY_IN的引脚上。图1.1(B)RD~、WR~分别和单片机的读写引脚相连。做键盘和LED显示实验时，需将 $\overline{\text{KEY / LED CS}}$ 连接接到相应的地址译码上。位码输出(OUT_BIT)的地址为0X002H，段码输出(OUT_SEG)的地址为0X004H，键盘行码(KEY_IN)读回的地址为0X001H，此处X是由 $\overline{\text{KEY / LED CS}}$ 决定。例如将 $\overline{\text{KEY / LED CS}}$ 接到地址译码的CS0（如图1.2）上，那么位码输出的地址就为08002H，段码输出的地址就是08004H，键盘行码读回的地址为08001H。

（3）Lab2000P 实验仪单片机系统 138 译码电路如下图所示，图中 A12-A15 对应连接到单片机的 P2 口的高 4 位，即 P2.4-P2.7。

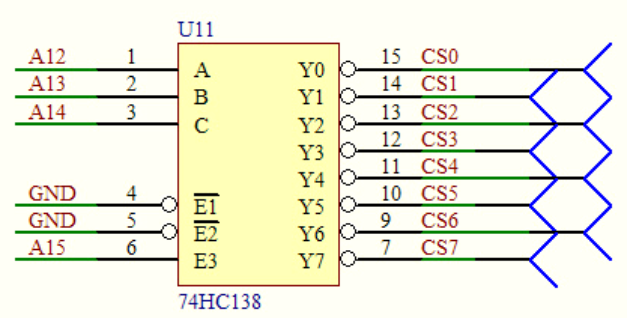


图1.2　译码电路图

其中：CS0: 08000H～08FFFH　CS1: 09000H～09FFFH　CS2: 0A000H～0AFFFH
CS3: 0B000H～0BFFFH　CS4: 0C000H～0CFFFH　CS5: 0D000H～0DFFFH
CS6: 0E000H～0EFFFH　CS7: 0F000H～0FFFFH

4. 段码表和键码表

（1）段码表

七段数码管的字符型代码表如表1.1所示。

（2）键码表

键码表如下，表中的前两行为功能键，其余为数字键。

KeyTable:　　；键码定义

16h, 15h, 14h, 0ffh　　分别对应按键 NEXT, LAST,MON, RST

13h, 12h, 11h, 10h　　分别对应按键 EXEC,HERE, MOVE, TRACE/MODE

0dh, 0ch, 0bh, 0ah　　分别对应按键 D，C，B，A

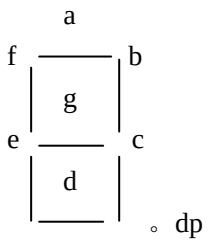
0eh, 03h, 06h, 09h　　分别对应按键 E，3，6，9

0fh, 02h, 05h, 08h　　分别对应按键 F，2，5，8

00h, 01h, 04h, 07h　　分别对应按键 0，1，4，7

表1.1 七段数码管的字符型代码表

显示字形	g	f	e	d	c	b	a	段码
0	0	1	1	1	1	1	1	3fh
1	0	0	0	0	1	1	0	06h
2	1	0	1	1	0	1	1	5bh
3	1	0	0	1	1	1	1	4fh
4	1	1	0	0	1	1	0	66h
5	1	1	0	1	1	0	1	6dh
6	1	1	1	1	1	0	1	7dh
7	0	0	0	0	1	1	1	07h
8	1	1	1	1	1	1	1	7fh
9	1	1	0	1	1	1	1	6fh
A	1	1	1	0	1	1	1	77h
b	1	1	1	1	1	0	0	7ch
C	0	1	1	1	0	0	1	39h
d	1	0	1	1	1	1	0	5eh
E	1	1	1	1	0	0	1	79h
F	1	1	1	0	0	0	1	71h



3. 验证实验程序流程

（1）主程序流程

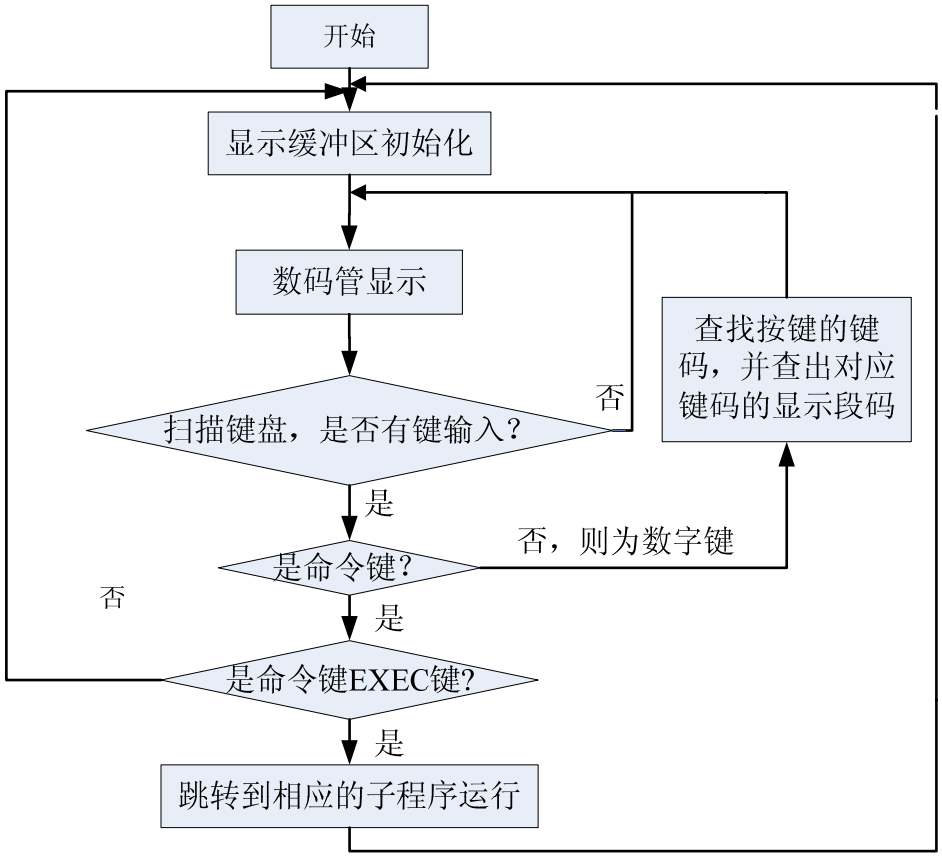


图 1.3 监控实验主程序流程图

（2）键盘扫描子程序流程图

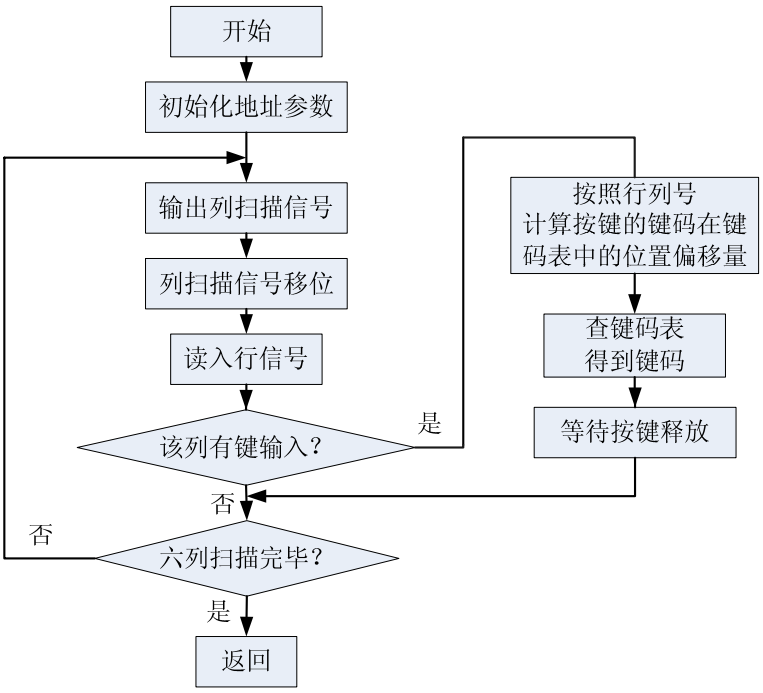


图1.4 键盘输入子程序流程图

五、实验步骤

1．运行Keil软件，按照附1.3 “Keil软件使用的参考步骤”步骤1～10，建立工程、编译、链接。

2、硬件连接：

（1)连接片选信号，将$\overline{\text{KEY/LED CS}}$连接到CS0即可(具体的电路原理图请参考图1.1)；8255的片选连接到CS1。

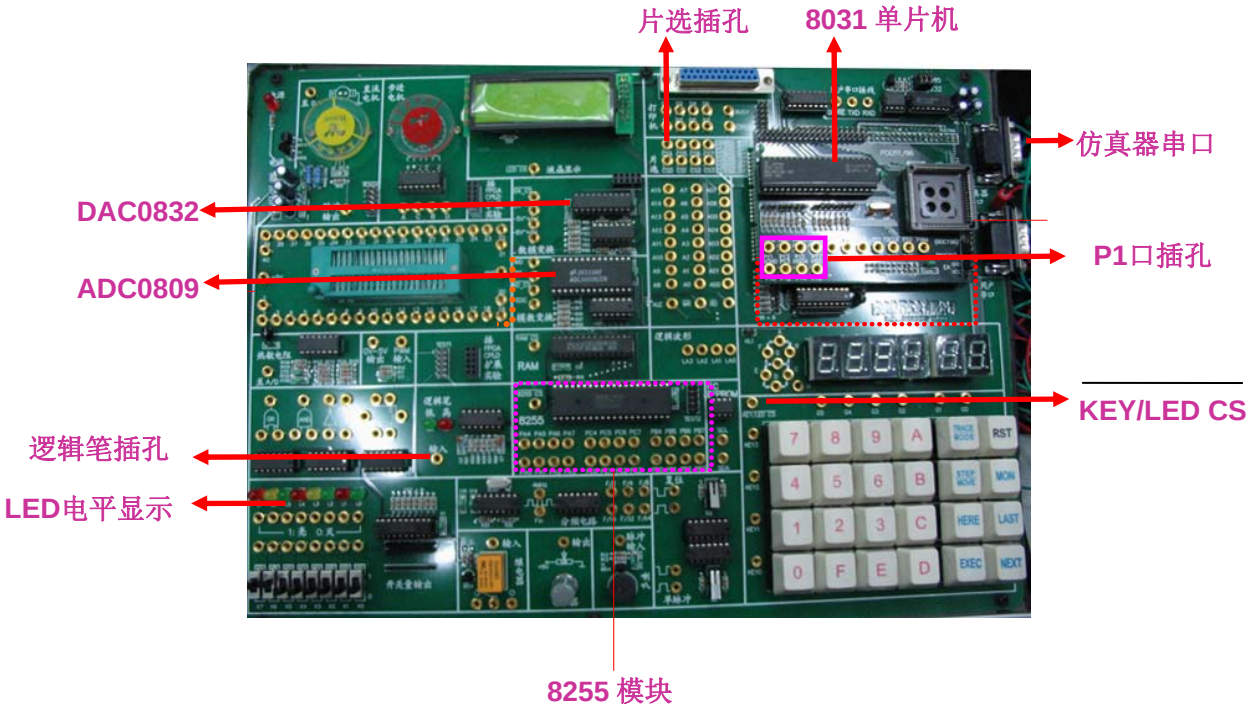
（2）用串口线连接Lab2000P实验仪（右上角标有“仿真器串口”处）到PC机。用电源线连接实验仪到220V电源，开启实验仪的电源（电源开关在实验仪的左侧）。

3、按照附1.3步骤11～12下载目标文件，并全速运行程序。

4、程序全速运行时，从实验仪上的键盘输入0200，再按“EXEC”键，观察实验结果。等到最左边的数码管显示“0”时，再从键盘输入0300，然后按“EXEC”键，用示波器观察8255的PA、PB、PC口输出的波形。或者把PA、PB、PC中的每位输出连接到实验仪中“逻辑笔”的输入，观察指示灯的变化，记录实验结果。

5、按照设计性实验要求完成设计实验。

附 1.1：lab2000P 实验仪



附 1.2：验证实验程序

```
/*监控程序：包括键盘扫描、8 字循环显示、8255 三个口输出方波*/

OUTBIT      equ   08002h   ；位控制口
OUTSEG      equ   08004h   ；段控制口
IN          equ   08001h   ；键盘读入口
Ctrl_8255   equ   09003h   ; 8255 控制口地址
PA_8255     equ   09000h   ; 8255 A 口地址

LEDBuf      equ     60h    ；键盘输入数字显示缓冲
LEDBuf1     equ     70h    ；六位数码管循环显示缓存
LEDBuf2     equ     50h    ；键码缓存

        ljmp    Start           ;程序运行的第一条指令

LEDMap:                     ；八段数码管显示码
        db      3fh, 06h, 5bh, 4fh, 66h, 6dh, 7dh, 07h      ; 0-7 的段码
        db      7fh, 6fh, 77h, 7ch, 39h, 5eh, 79h, 71h      ; 8-f 的段码


;============================== 延时子程序==============================
Delay:
        mov     r7, #0
DelayLoop:
        djnz    r7, DelayLoop
        djnz    r6, DelayLoop
        ret


;========================6 位数码管轮流显示子程序======================
DisplayLED:
        mov     r0, #020h
        mov     r1, #6              ；共 6 个八段管
        mov     r2, #00100000b ；从左边开始显示
Loop:
        mov     dptr, #OUTBIT
        mov     a, #0
        movx    @dptr, a            ；关所有八段管
        mov     a, @r0
        mov     dptr, #OUTSEG
        movx    @dptr,a
        mov     dptr, #OUTBIT
```

```
        mov     a, r2
        movx    @dptr, a            ; 显示一位八段管
        mov     a, r2               ; 显示下一位
        rr      a
        mov     r2, a
        inc     r0
        djnz    r1, Loop
        ret


;========================== 单片机读入键盘行的状态子程序 ===============
TestKey:
        mov     dptr, #OUTBIT
        mov     a, #0
        movx    @dptr, a            ; 输出线全置为 0
        mov     dptr, #IN           ; IN =08001h
        movx    a, @dptr            ; 读入键状态
        cpl     a
        anl     a, #0fh             ; 高四位不用
        ret

KeyTable:                           ; 键码定义
        db      16h, 15h, 14h, 0ffh   ;命令键的键码，分别对应 NEXT, LAST,MON, RST
        db      13h, 12h, 11h, 10h    ;命令键的键码，分别对应 EXEC,HERE, MOVE,
                                      ;TRACE/MODE
        db      0dh, 0ch, 0bh, 0ah    ; 分别对应 D，C，B，A
        db      0eh, 03h, 06h, 09h    ;分别对应 E，3，6，9
        db      0fh, 02h, 05h, 08h    ;分别对应 F，2，5，8
        db      00h, 01h, 04h, 07h    ;分别对应 0，1，4，7


;==========================扫描键盘子程序 ==========================
GetKey:
        mov     dptr, #OUTBIT

        mov     P2, dph
        mov     r0, #Low(IN)
        mov     r1, #00100000b
        mov     r2, #6
KLoop:
        mov     a, r1
        cpl     a
        movx    @dptr, a        ; 扫描第 1 列，使键盘第 1 列为低电平
```

```
        cpl     a
        rr      a
        mov     r1, a               ; 下一列
        movx    a, @r0              ; 读入行的状态
        cpl     a
        anl     a, #0fh
        jnz     Goon1               ; 该列有键入
        djnz    r2, KLoop
        mov     r2, #0ffh           ; 没有键按下，返回 0ffh
        sjmp    Exit
Goon1:
        mov     r1, a   ; 按下键的键码在键码表中的位置偏移量 =（列－1） X 4 + （行－1）
                        ;从第 6 列开始
        mov     a, r2
        dec     a
        rl      a
        rl      a
        mov     r2, a               ; r2 = (r2-1)*4
        mov     a, r1               ; r1 中为读入的行值
        mov     r1, #4
LoopC:
        rrc     a                   ; 移位找出所在行
        jc      Exit
        inc     r2                  ; r2 = r2+行值
        djnz    r1, LoopC
Exit:
        mov     a, r2               ;r2 中为按下键的键码在键码表中的位置偏移量
        mov     dptr, #KeyTable
        movc    a, @a+dptr
        mov     r2, a       ; 取出键码
WaitRelease:
        mov     dptr, #OUTBIT   ; 等键释放
        clr     a
        movx    @dptr, a
        mov     r6, #10
        call    Delay
        call    TestKey
        jnz     WaitRelease
        mov     a, r2
        ret


;================查找 A 中存放的数所对应 LED 显示段码子程序 ==============
ToLED:
```

```
        mov     dptr, #LEDMap
        movc    a, @a+dptr
        ret


;===========将键码(在 A 中) 存放到 LEDBuf2 所指向的内部存储器单元===========

ToKeyTable:
        mov     r1,LEDBuf2
        mov     @r1,a
        ret


;=========================初始化单片机内存子程序====================

InitLED:
        mov     20h, #3fh
        mov     21h, #0h
        mov     22h, #0h
        mov     23h, #0h
        mov     24h, #0h
        mov     25h, #0h
        ret

////////////////////////////////////////////////////  主程序  ////////////////////////////////////////////////////
Start:
        mov     LEDBuf,#020h
        mov     LEDBuf2,#30h
        call    InitLED
        mov     LEDBuf1+0, #0ffh ; 后面“8”字循环程序用到，0ffh 为 8.的段码
        mov     LEDBuf1+1, #0ffh
        mov     LEDBuf1+2, #0ffh
        mov     LEDBuf1+3, #0ffh
        mov     LEDBuf1+4, #0ffh
        mov     LEDBuf1+5, #0ffh
        mov     r4,#6
MLoop:
        call    DisplayLED
        call    TestKey             ;有键输入?
        jz      MLoop               ;无键输入，继续显示
        call    GetKey              ;读入键码
        jb      0e4h,commkey        ;如果键码的第 5 位即 ACC.4 为 1，则为命令键，转到
                                    ; commkey 处理
numkey:                             ;不是命令键，则为数字键
```

```
        mov     r0,a            ;保存数据 a
        call    ToKeyTable      ;键码保存在 LEDBuf2（初始值 30h）指向的单元中
        mov     a,r0
        anl     a, #0fh         ;数字键的键码只用到低 4 位，高 4 位不用，屏蔽掉高 4 位
        call    ToLED           ;查找 A 的段码并把段码保存在 A 中
        mov     r0,LEDBuf
        mov     @r0,a           ;段码保存在 LEDBuf（初始值 20h）指向的单元中
        inc     LEDBuf
        inc     LEDBuf2
        djnz    r4,MLoop
        mov     r4,#6
        mov     LEDBuf,#020h
        mov     LEDBuf2,#030h
        ljmp    MLoop
commkey:                        ;处理命令键
        cjne a,#13h,start       ;如果按下的键不是 EXEC，则转向程序开始
        call nextkey            ;如果按下的键是 EXEC，则跳转到键盘输入的地址（存储在单片机
                                ;内部 RAM30h-33h 单元中）去运行

;==========================把 dph、dpl 压入堆栈子程序====================

 nextkey:
        acall   pickdata
        push    dpl
        push    dph
        ret             ;dph 与 dpl 弹出堆栈赋给 PC，跳转到相应的地址去运行

;================================子程序============================;
; 取出单片机内部 RAM30h-33h 四个单元的数据（键盘输入的四个数据）分别赋给
; DPH，DPL
pickdata:
        mov     r0,#031h    ;将 30H、31H 单元中的低 4 位数据赋给 DPH
        acall   pickone
        mov     dph,a
        mov     r0,#033h    ;将 32H、33H 单元中的 4 位数据赋给 DPL
        acall   pickone
        mov     dpl,a
        ret

;============================取相邻内存单元低 4 位子程序===============
;取出 r0 与 r0-1 所指向的单元中低 4 位数据，并保存到 A 中，只取低 4 位，因为键盘上输
入的每位数据（0 到 F）最多只用到 4 位表示
pickone:
        mov     a,@r0
```

```
        anl     a,#0fh
        mov     r1,a
        dec     r0
        mov     a,@r0
        swap    a
        anl     a,#0f0h
        orl     a,r1
        ret

////////////// “8”字循环程序，即六位数码管轮流显示“8”字，从左到右共循环 10 次//////////////////////
        org     0200h
        mov     r0,#4
        clr a
        mov     r3,#10              ;循环次数设置，可修改
DisplayLED1:                        ;轮流显示
        mov     r0, #LEDBuf1
        mov     r1, #6              ; 共 6 个八段管
        mov     r2, #00100000b      ; 从左边开始显示
Loop1:
        mov     dptr, #OUTBIT
        mov     a, #0
        movx    @dptr, a            ; 关所有八段管
        mov     a, @r0
        mov     dptr, #OUTSEG
        movx    @dptr, a
        mov     dptr, #OUTBIT
        mov     a, r2
        movx    @dptr, a            ; 显示一位八段管
        mov     r6, #200
        call    Delay               ; 延迟值如果设定的太大的话，就会检测不到按键
        mov     a, r2               ; 显示下一位
        rr      a
        mov     r2, a
        inc     r0
        djnz    r1, Loop1
        djnz    r3, DisplayLED1     ;是否循环到 10 次
        acall   InitLED             ;重新从左边显示
        mov     r4,#6
        mov     LEDBuf,#020h
        mov     LEDBuf2,#030h
        ajmp    MLoop

//////////////////////////////////////8255 的 PA,PB,PC 口 分别循环输出方波程序/////////////////////////
        org     0300h
```

```
testPort:
        mov     dptr,#Ctrl_8255
        mov     a,#80h
        movx    @dptr,a
        mov     a,#55h
        mov     r3,#20              ;循环次数，可自行设置
testPortA:
        mov     dptr,#PA_8255   ;PA 口
        movx    @dptr,a
        inc     dptr
        movx    @dptr,a         ;PB 口
        inc     dptr
        movx    @dptr,a         ;PC 口
        rr      a
        mov     r6, #200        ;输出延时,可自行设置延时时间
        acall   Delay
        djnz    r3, testPortA
        ajmp    MLoop

         end
```

附 1.3：Keil 软件的使用步骤参考

1、 点击”Project→New Project……”，新建一个工程文件，在 Creat New Project 窗口中输入工程名。

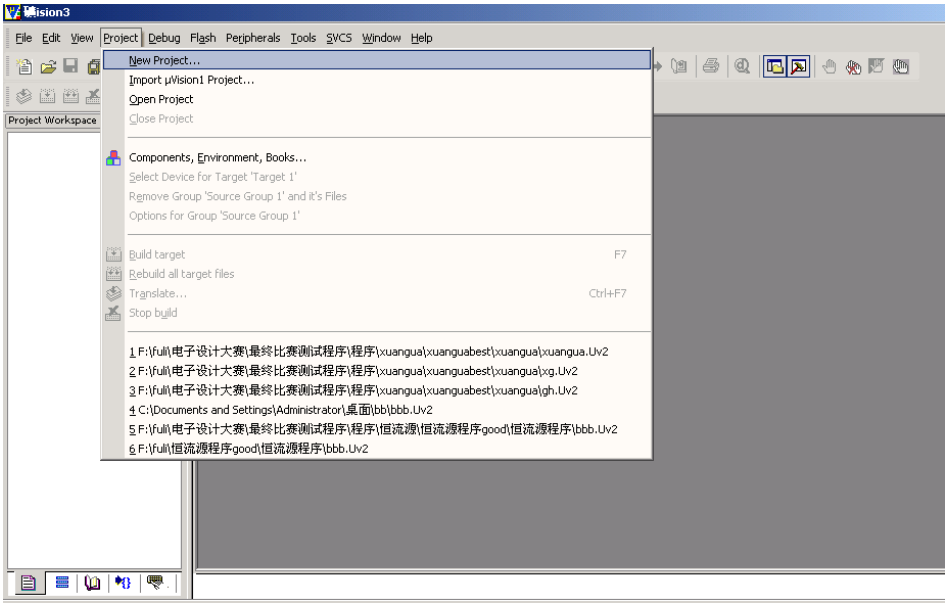


图 1.5 新建工程

2、Creat New Project 窗口中，点击“确定”，然后出现如下窗口，选择芯片类型，本实验选用芯片为 Intel 公司的 8051AH。

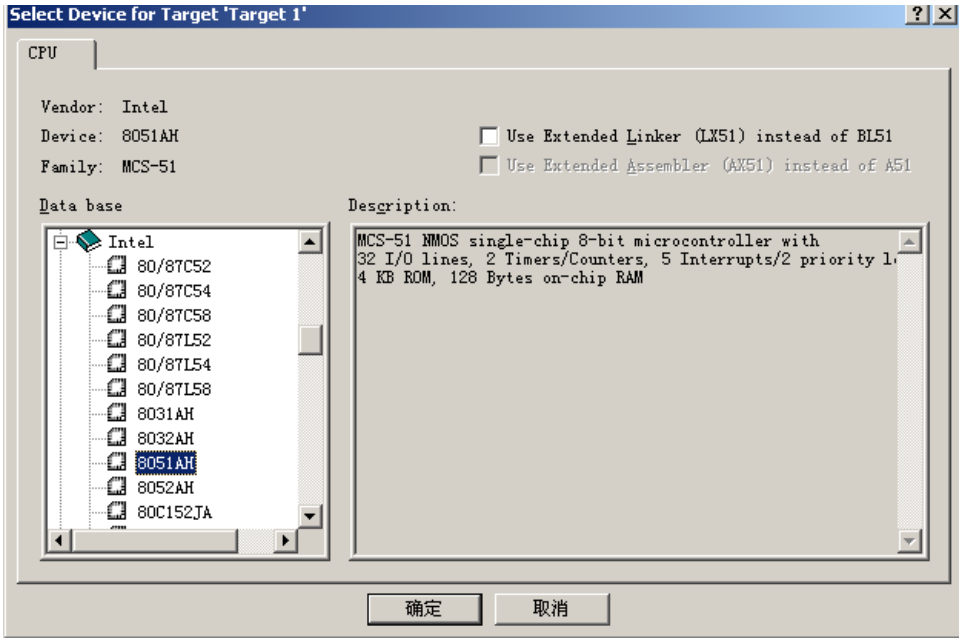


图 1.6 选择芯片

3、点击“确定”后，在出现的如下窗口中选择“否”。

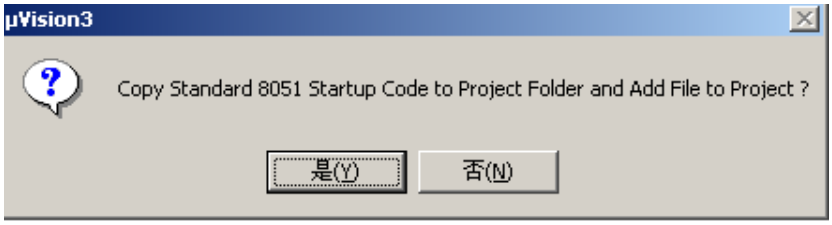


图 1.7 是否添加 Startup 文件到工程中

4、新建文件，File->New……或直接点击快捷按钮 Creat a New File ，然后编写程序。文件也可以在建立工程之前建立并编写。

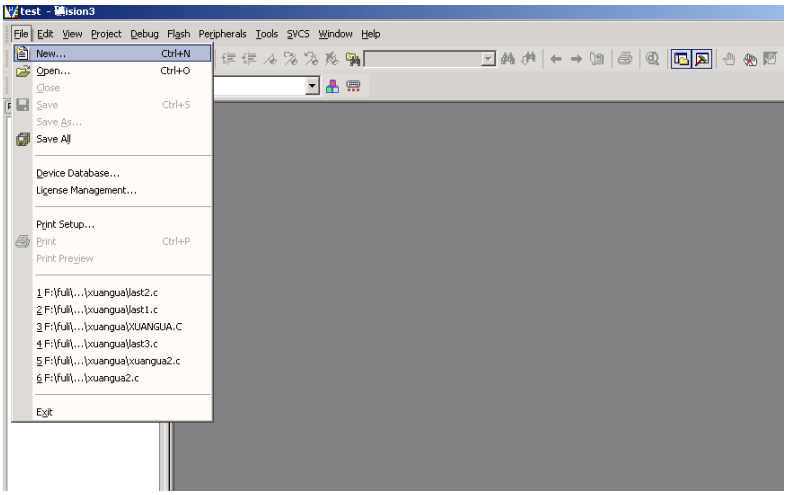


图 1.8 新建程序文件

5、保存文件，注意文件名必须有后缀名，若是汇编程序后缀名为.asm，C 程序为.c。

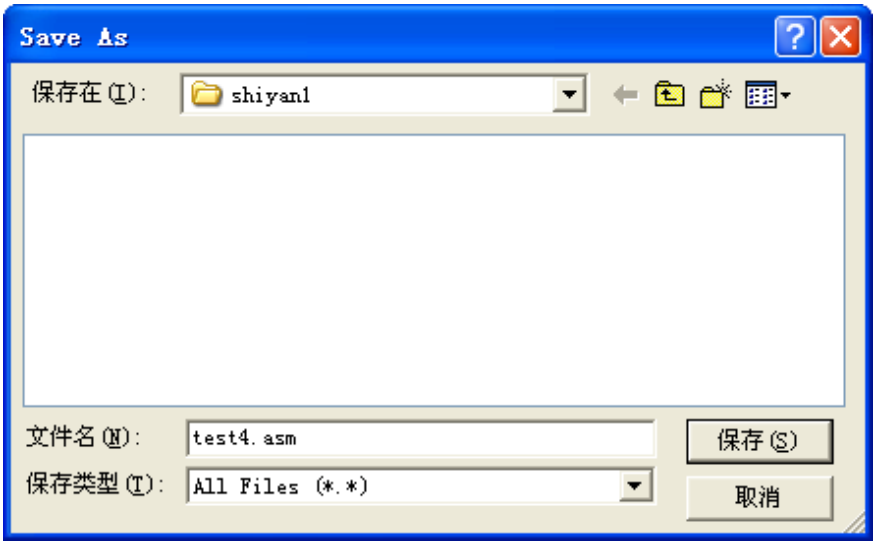


图 1.9 保存文件

6、把文件添加到建立的工程中，如下图，点击 Project Workspace 窗口中的 Source Group1，单击鼠标右键，在弹出的菜单中选择“Add Files Group “Source Group 1””

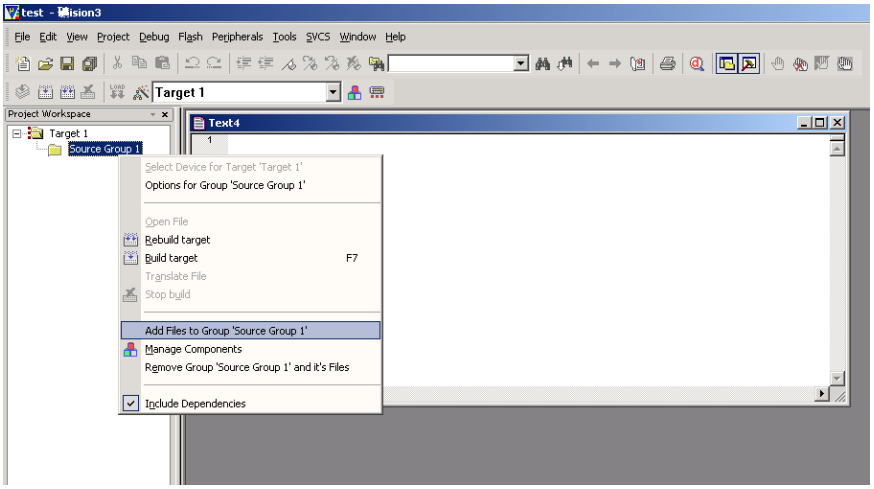


图 1.10 添加文件到工程

7、找到程序文件保存的目录，在出现如下图所示的窗口中，单击“ADD”按钮。

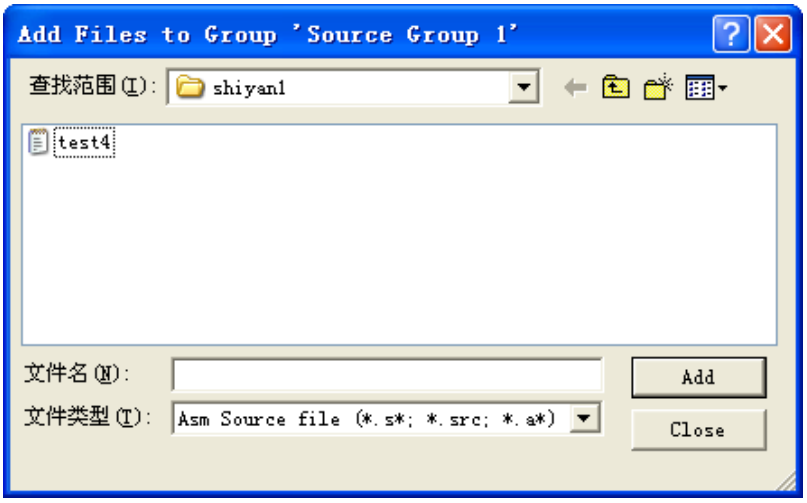


图 1.11 ADD Files to Group 窗口

8、按照下图 1.12 所示选择“Option for Target “Target 1””。

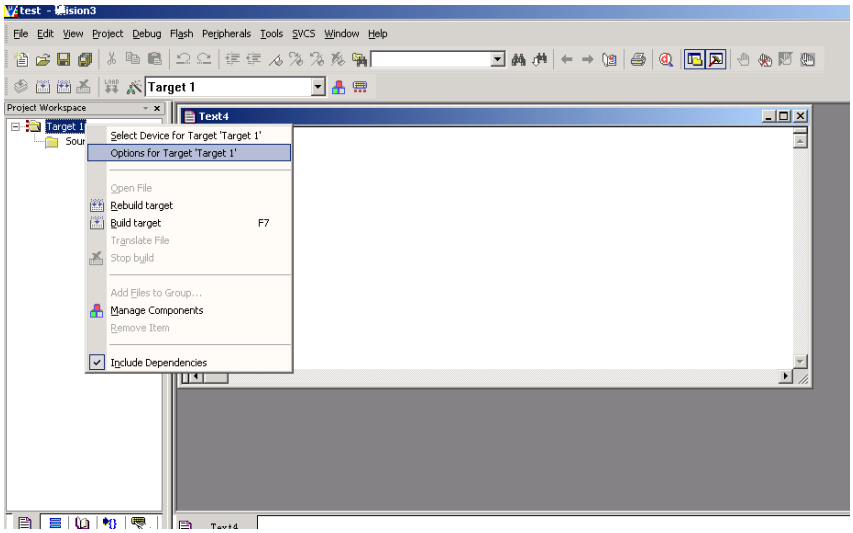


图 1.12 选择“Option for Target “Target 1””

9、单击选择 Option for Target “Target 1”窗口中的 Debug。设置.Use 为“WAVE Emulator Driver”，其余设置默认。

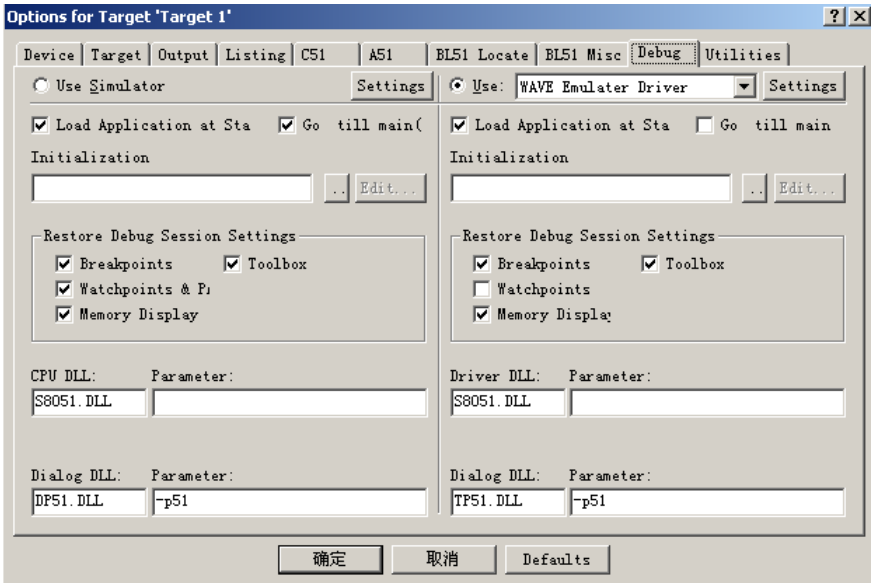
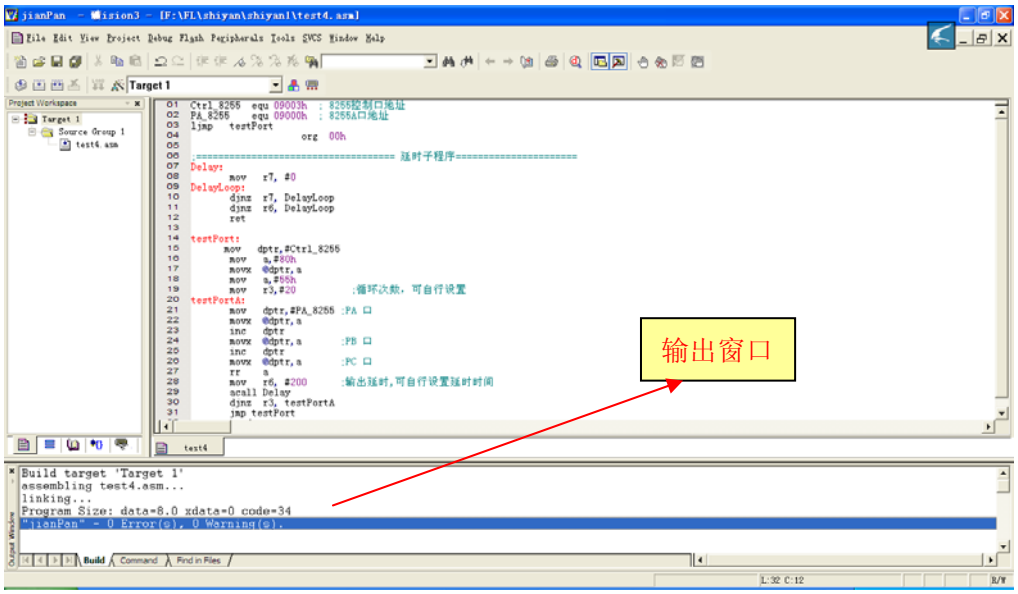


图 1.13　Option for Target “Target 1”窗口

10、Project->Rebuild all target files，工程编译链接。注意：观察输出窗口，看是否有错误提示，若有错误，需修改程序中的错误，然后再编译链接直至没有错误提示。



11、下载程序到仿真系统，选择 Debug->start/stop Debug Session。

12、运行程序 Debug->Go 或按快捷键 F5，若单步运行程序，可选择 Debug->Step。

提示：若需停止程序运行，选择 Debug->Stop Running，然后再选择 Debug->start/stop Debug Session

实验二　A/D 转换实验 (3 学时)

实验预习要求：

1．熟悉AD0809工作原理，熟悉单片机中断编程方法。

2．写出设计性实验源程序。

3．完成下面的思考题：

参考ADC0809转换电路图，ADC0809的控制引脚ALE、ENABLE、START怎样和单片机连接？叙述其原理。

一、实验目的

1．理解 A/D 转换的基本原理。

2．掌握 A/D 转换芯片 0809 的性能及编程方法。

3．掌握单片机系统中扩展 A/D 和 8255 的基本方法。

4．掌握单片机外部中断编程方法。

二、 实验器材

PC 机一台，Lab2000P 实验仪一台，导线数根

三、实验内容

1．验证性实验

利用实验系统中的 ADC0809 做 A/D 转换实验，其中系统中的电位器提供模拟量输入，验证程序实现将模拟量转换成二进制数字量，并用 8255 的 PA 口输出到发光二极管显示的功能。

2．设计性实验

ADC0809 芯片 A/D 转换结束后会自动产生 EOC 信号，将 EOC 与 CPU 的外部中断相接，在验证实验基础上，用中断方式编程读回 A/D 转换结果。

四、实验原理

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。

ADC0809 属第二类，是八位 A/D 转换器，每采集一次一般需 100us。本程序是用延时查询方式读入 A/D 转换结果。图 2.1 中/RD、/WR 分别和单片机的读写引脚相连，AD0~AD7 对应和单片机的 P0.0~P0.7 连接。

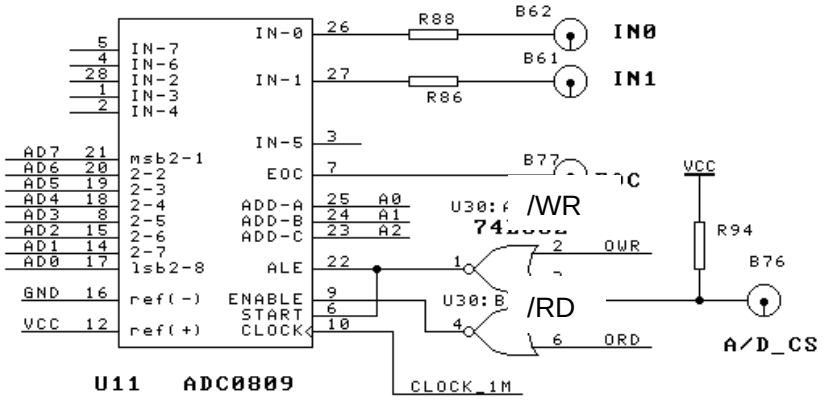


图 2.1　ADC0809 转换电路

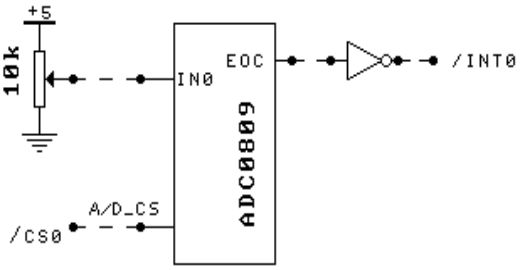


图 2.2 ADC0809 与单片机连接

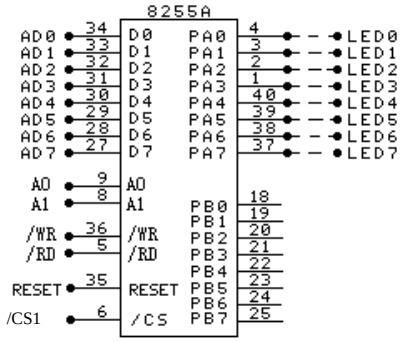


图 2.3 8255A 引脚图

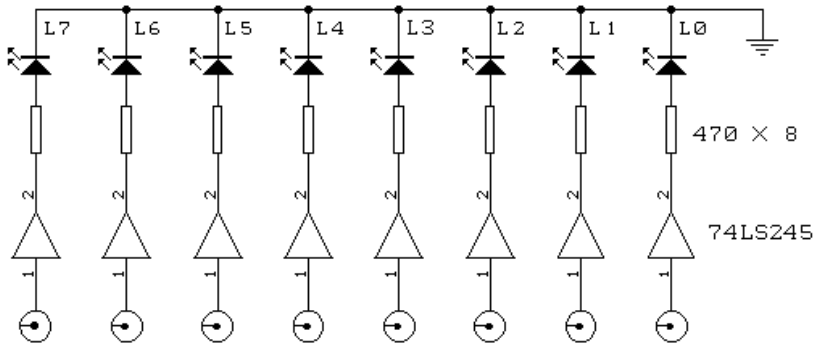


图 2.4　LED 电平显示电路

五、实验步骤

1．验证实验

（1）导线连接

a．$\overline{\mathrm{AD_CS}}$连接到$\overline{CS0}$（即连接到图1.2中的38译码器的输出CS0）；

b．8255片选$\overline{8255\ \mathrm{CS}}$连接到$\overline{CS1}$（即连接到图1.2中的38译码器的输出CS1）；

c．PA口PA0～PA7连接到LED电平显示模块中的L0～L7对应插孔；

d．ADC0809模拟输入连接到电位器模块中的“输出”。

（2）连接Lab2000P仿真器串口到PC机；连接实验仪的电源并开启。

（3）运行Keil软件，按照实验一附1.3“Keil的使用步骤参考”建立工程、添加实验程序、编译链接。注意：Project下Option for Target …－>Debug下设置use仿真器为WAVE Emulater Driver。

（4）程序下载后全速运行，用电位器调节ADC0809的模拟输入，观察发光二极管的状态，思考状态变化的原因。用万用表测量模拟输入的电压值并记录，从发光二极管的状态读出AD转换后的数字量并记录。再调节ADC0809的模拟输入，从发光二极管的状态读出AD转换后的数字量并记录。如此重复，记录至少10组数据。

2．设计实验

（1）导线连接：

a．将EOC和非门的输入连接；

b．该非门输出连接到单片机的中断引脚P3.2或P3.3。

（2）调试编写好的程序，并用实验仪进行仿真。

附 2.1：验证实验程序

AD0809程序：

```
mode     equ    082h     ; 082h  设为 8255 控制字，方式 0，PA，PC 输出，PB 输入
CS0809 equ    8000h
PortA    equ    9000h    ; Port A
PortB    equ    9001h    ; Port B
PortC    equ    9002h    ; Port C
CAddr    equ    9003h    ;  控制字地址

          ajmp    start
          org    0340h
start:    mov      dptr,#CAddr
          mov      a,#mode
          movx    @dptr,a

          mov      dptr, #CS0809
          mov      a, #0
          movx    @dptr, a       ;  起动  A/D

          mov      a, #40h
          djnz     ACC, $         ;  延时  > 100us

          movx    a, @dptr       ;  读入 A/D 转换结果
          mov      r7,#100
 dly:
          mov      dptr,#PortA
          movx    @dptr,a
          djnz     r7,dly          ;延时
          jmp     start
          end
```

实验三　D/A 转换实验 (2 学时)

实验预习要求：

1．熟悉DA0832的工作原理及其与单片机的接口方法。

2．写出设计性实验源程序。

3．查阅相关资料，写出8位DA转换器(DAC0832)的输入数据与输出电压的关系。

一、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的性能及编程方法。

3．了解单片机系统中扩展D/A转换的基本方法。

二、 实验器材

PC 机一台，Lab2000P 实验仪一台，导线数根

三、实验内容

1．验证性实验

一是利用实验系统中的 DAC0832，验证程序产生锯齿波，用示波器观察产生的波形。

2．设计性实验

编写利用实验系统中的 DAC0832 产生正弦波的设计程序（提示：可用 C 语言计算产生 1 组单周期正弦波曲线的数据表，然后用查表法输出周期性的正弦波），用示波器观察波形。

四、实验原理

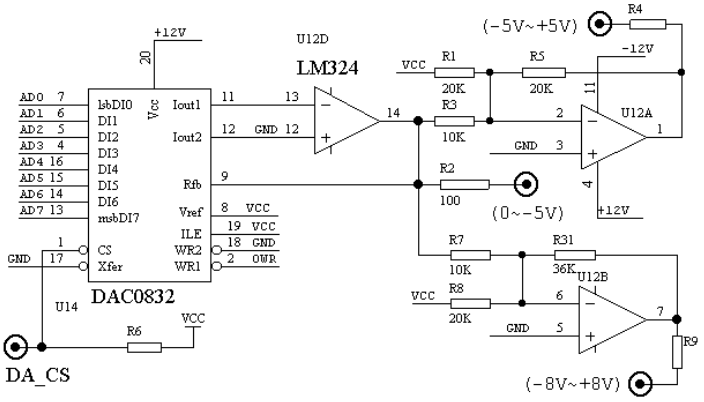


图3.1 实验原理图

实验仪上提供的 D/A 转换电路如图所示。可以通过软件编程控制 D/A 转换芯片 DAC0832 输出相应电流值，经过采样电路取出模拟量电压值，用电压表测量电压输出端子，

读出电压值。图 3.1 中 OWR 和单片机的写引脚相连，AD0~AD7 对应和单片机的 P0.0~P0.7 相连接。

五、实验步骤

1．验证实验

（1）连接 $\overline{\mathrm{DA\_CS}}$ 到 $\overline{CS2}$ 。

（2）POD51/96模块中跳线设置：S1：接80C51，EA：接地。

（3）连接Lab2000P仿真器串口到PC机；连接实验仪的电源并开启。

（4）运行Keil软件，开始实验，注意Project下Option for Target …一>Debug下设置use仿真器为WAVE Emulater Driver。

（5）程序编译链接无误后，下载目标文件到实验箱，并全速运行。

（6）用示波器观察D/A转换结果的模拟信号，记录实验结果。

2．设计实验

调试编写好的设计程序，并用实验仪进行仿真，用示波器观察D/A转换结果的模拟信号。

附 3.1：DA 转换实验程序

```
--－－－－－－－－－－－－－－－－－－－－产生锯齿波程序－－－－－－－－－－－－－－－－－－－－
        CS0832 equ 0a000h

          mov     a, #0
pp:       mov     dptr, #CS0832
          movx    @dptr, a    ;启动DA0832，并转换
          inc a
          ljmp    pp
          end
```