浙江工业大学

《单片机原理及应用》
课程实验报告

学生姓名 凌智城

指导教师 庄婵飞

专业班级 通信工程 1803 班

培养类别 全日制本科

所在学院 信息工程学院

提交日期 2021 年 1 月 4 日

目录

第一次实验：熟悉 Keil 软件系统和软件仿真、BCD 码加法、排序等

1.1 设计要求：

实验一：熟悉 Keil 软件系统和软件仿真，包括：

1) 启动 Keil 软件；
2) 使用 Keil 建立一个工程；
3) 新建一个工程；
4) 建立并保存新文件；
5) 把文件加入工程；
6) 输入并保存文件内容；
7) 软件仿真的设置；
8) Keil 仿真调试技巧（编译、链接、错误修改等）；
9) Debug 调试菜单（运行、单步、断点设置等）；

实验四：编程实现 4.多字节无符号数 BCD 码加法

思考题：试编写汇编源程序，编译后实现如下功能：

1) 若晶振频率是 12MHZ，轮流点亮 P1 口流水灯（从 01H-80H），间隔时间 0.5S。
2) 完成多字节 BCD 码加法，加数在 ROM 地址 0200H-0203H 单元中，被加数在 ROM 地址 0204H-0207H 单元中。
3) 将 ROM 中 0200H---020FH 存储单元的 16 个无符号数：22H,55H,0A0H,11H,0C0H,99H,00H,0B0H,44H,0F0H,77H,33H,0E0H,66H,88H,0D0H 转移到片内 RAM 的 40H---4FH 单元中；
   a) 找出其中最大数放在 R3 中；
   b) 将 40H---4FH 中的无符号数从小到大排列。

1.2 实验代码：

1) BCD_ADD.asm

```
        ORG     0000H
        LJMP    MAIN
        ORG     0100H
MAIN:
        MOV     SP,#60H
        CLR     C
        MOV     R2,50H
        MOV     51H,#00H;和的字节数清零
        MOV     R0,#30H
        MOV     R1,#40H
        LCALL BCD_ADD_BYTES
        SJMP   $

BCD_ADD_BYTES:
        MOV     A,@R0;取被加数
        ADDC    A,@R1;求和
        DA      A;十进制调整
        MOV     @R0,A;保存
        INC     51H;字节增 1
        INC     R0;地址增 1
        INC     R1
        DJNZ    50H ,BCD_ADD_BYTES;所有字节未加完继续，否则向下执行
        JC      NEXT;和的最高字节有进位则转移 next
        RET

NEXT:
        INC    51H
        MOV    @R0,#01H
        RET

        END
```

2) LED.asm

```
        ORG     0000H
        AJMP    START

START:
        MOV     DPTR,#TAB
LOOP1:
        CLR     A
        MOVC    A,@A+DPTR
        CJNE    A,#00H,SHOW
        AJMP    START
SHOW:
        MOV     P1,A
```

```
        ACALL   DELAY
        INC     DPTR
        AJMP    LOOP1
DELAY:  MOV     R5,#25
D1:     MOV     R6,#100
D2:     MOV     R7,#100
D3:     DJNZ    R7,D3
        DJNZ    R6,D2
        DJNZ    R5,D1
        RET
TAB:    DB      01H,02H,04H,08H,10H,20H,40H,80H,40H,20H,10H,08H,04H,02H,01H
        DB      00H
        END
```

3) tk2.asm

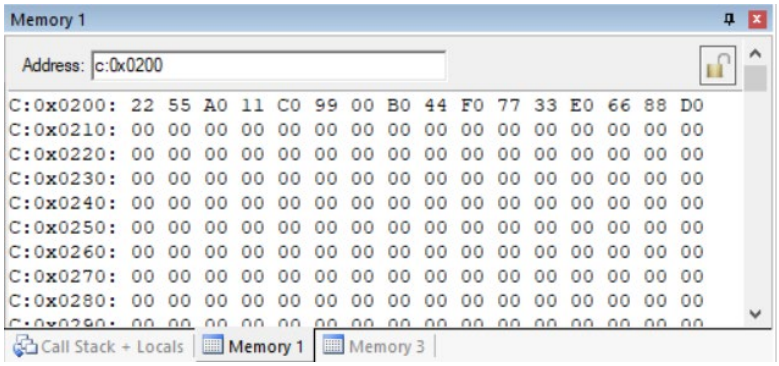
```
ORG     0000H
LJMP    MAIN
ORG     0100H
MAIN:
MOV     SP,#60H
CLR     C
MOV     DPTR,#2000H
MOV     51H,#00H
MOV     R1,#30H
MOV     R2,#04H
MOV     R3,#04H
LCALL   BCD_ADD_BYTES
SJMP    $

BCD_ADD_BYTES:
MOV     A,#00H
MOVC    A,@A+DPTR
MOV     R0,A
MOV     A,R3
MOVC    A,@A+DPTR
ADDC    A,R0
DA      A
MOV     @R1,A
MOV     A,R1
INC     A
MOV     R1,A
INC     DPTR
DJNZ    R2,BCD_ADD_BYTES
JC      S2
RET
```

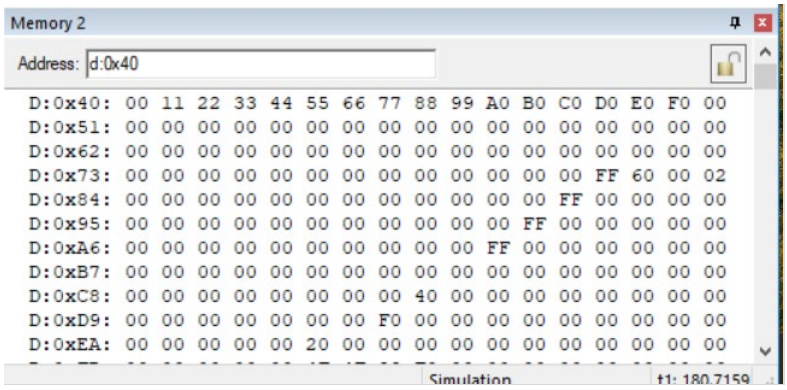
```
S2:
 MOV   @R1,#01H
 RET

 END
```

4) tk3.asm

```
ORG          0000H
LJMP         MAIN
ORG          0050H
MAIN:
        MOV     SP,#60H
        MOV     DPTR,#0200H
        MOV     R0,#40H
        MOV     R2,#00H
        MOV     R3,#00H
F1:
        MOV     A,R2
        MOVC    A,@A+DPTR
        MOV     @R0,A
        SUBB    A,R3
        JC      F2

        MOV     A,R2
        MOVC    A,@A+DPTR
        MOV     R3,A;如果没有借位则把最大数暂存 R3
    F2:
        CLR     C;循环十六次，先找出最大数，并先存入 40H--4FH
        INC     R0
        INC     R2
        CJNE    R2,#10H,F1

        MOV     R1,#40H
    PAIXU:
        MOV     R0,#40H
        MOV     A,@R0
        MOV     R3,A
    F3:
        INC     R0
        MOV     A,@R0
        SUBB    A,R3
        JNC     F4;若无借位，则高位比低位大，无需交换
        MOV     A,@R0;若有借位，交换操作
        MOV     R2,A
```

```
        MOV     A,R3
        MOV     @R0,A
        DEC         R0
        MOV     A,R2
        MOV     @R0,A
        INC     R0
        CLR     C
        CJNE    R0,#4FH,F3
    F4:
        MOV     A,@R0
        MOV     R3,A
        CLR     C
        CJNE    R0,#4FH,F3

        INC     R1
        CJNE    R1,#4FH,PAIXU

        SJMP $
        END
```





1.3 遇到的问题及解决方法：

第一次做单片机实验还是有很多问题，虽然提前做过准备，但是实际操作起来有很多不熟悉的地方，比如说 Memory 中 RAM 和 Rom 的修改，以及调试的过程，

都比较陌生，在实验中存在一些问题，导致第一次实验的效果不佳。

第二次实验：学习使用普中仿真器进行硬件仿真，LED 流水灯驱动汇编程序，静态、动态数码管驱动汇编程序等

2.1 设计要求：

学习使用普中仿真器进行硬件仿真，包括：

1) 普中仿真器的链接（实验十三）；
2) Keil 中设置仿真器；
3) 实验室五，LED 流水灯驱动汇编程序；
4) 实验十八，静态数码管显示驱动汇编程序（实现 0~9 循环播放或自由发挥，也可用 T0 或 T1 控制静态数码管显示速度）；
5) 实验十九，动态数码管显示驱动汇编程序（实现把显示缓冲区 30h~37h 的内容显示在显示器上）。

思考题：

1) 改变流水灯的显示顺序及速度；
2) 用定时器 T0 或 T1 控制静态数码管的显示速度。

2.2 实验代码：

1) liushui01.asm（使用逐个位操作和延时控制）

```
ORG     0000H
        AJMP    START
        ORG     0100H
START:
        CLR     P1.0
        ACALL   DELAY
        SETB    P1.0
        CLR     P1.1
        ACALL   DELAY
        SETB    P1.1
        CLR     P1.2
        ACALL   DELAY
        SETB    P1.2
        CLR     P1.3
```

```
        ACALL   DELAY
        SETB    P1.3
        CLR     P1.4
        ACALL   DELAY
        SETB    P1.4
        CLR     P1.5
        ACALL   DELAY
        SETB    P1.5
        CLR     P1.6
        ACALL   DELAY
        SETB    P1.6
        CLR     P1.7
        ACALL   DELAY
        SETB    P1.7
        ACALL   DELAY
        AJMP    START
DELAY:  MOV         R5,#20
D2:     MOV         R6,#20
D1:     MOV         R7,#248
D3:     DJNZ    R7,D3
        DJNZ    R6,D1
        DJNZ    R5,D2
        RET

        END
```

2) liushui02.asm（通过左移操作和延时）

```
led     EQU     P0
        ORG     0000H
        LJMP    MAIN
        ORG     0050H
MAIN:
        MOV     SP,#60H
LOOP:
        MOV     A,#01H
        MOV         R0,#08H
LEFTROTATE:
       MOV      led,A
        LCALL    DELAY1S
        RL      A
        DJNZ    R0,LEFTROTATE

        MOV     R0,#08H
RIGHTROTATE:
        RR      A
```

```
        MOV     led,A
        LCALL   DELAY1S
        DJNZ    R0,RIGHTROTATE

        SJMP    LOOP


DELAY1S:
        MOV  R7,#0A7H
DL1:
        MOV  R6,#0ABH
DL0:
        MOV  R5,#10H
        DJNZ    R5,$
        DJNZ    R6,DL0
        DJNZ    R7,DL1
        NOP
        RET

        END
```

3) liushui03.asm（使用查表操作）

```
        ORG     0000H
        AJMP    START
        ORG         0100H
START:  MOV         SP,#60H
        MOV         DPTR,#TAB
LOOP:   CLR     A
        MOVC    A,@A+DPTR

        CJNE    A,#0FFH,SHOW
        AJMP    START
SHOW:   MOV         P2,A
        LCALL   DELAY
        INC     DPTR
        AJMP    LOOP

DELAY:  MOV         R5,#20
D2:     MOV         R6,#20
D1:     MOV         R7,#50
D3:     DJNZ    R7,D3
        DJNZ    R6,D1
        DJNZ    R5,D2
        RET
```

```
TAB:    DB      0E7H,0DBH,0BDH,7EH,7EH,0BDH,0DBH,0E7H
        DB      0FFH

        END
```

4) StaticDisplay.asm

```
        ORG     0000H
        AJMP    START
        ORG     0100H
START:
        MOV     DPTR,#TAB
        MOV     R0,#00H         ;COUNT
S1:     MOV     P2,#0FFH        ;初始
S2:     MOV     A,R0
        MOVC    A,@A+DPTR
        MOV     P2,A
        LCALL   DELAY
        INC     R0;
        CJNE    R0,#10,S2
        MOV     R0,#00H
        LJMP    S1

DELAY:  MOV     R5,#100
D2:     MOV     R6,#20
D1:     MOV     R7,#248
D3:     DJNZ    R7,D3
        DJNZ    R6,D1
        DJNZ    R5,D2
        RET

TAB:    DB      0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H
        DB      00H
        END
```

5) DynamicDisplay.asm

```
        ORG     0000H
        AJMP    START
        ORG     0100H
START:
        MOV     SP,#60H

MAIN:   MOV     R7,#08H; 显示八位
                        //MOV   R6,#80H;1000‘0000B，第一次左移后从最低位开始显示
        MOV     R6,#0           //此处用 38 译码器，注释掉的部分是循环位移
        MOV     R0,#30H;显示缓存区 30H
```

```
LOOP:
                    //MOV   A,R6;读取显示位
                    //RL    A;显示位左移
                    //MOV   R6,A;放回 R6 暂存显示位
                    //MOV   P2,A;向 P2 输出显示位
        MOV     P2,R6
        INC     R6
        MOV     A,@R0;读取显示值
        MOV     P0,A;输出送入 P0 口
        LCALL   DELAY;调用延时子程序
        INC     R0;显示缓存区+1
        DJNZ    R7,LOOP;判断是否显示完八位，未完则继续 loop
        AJMP    MAIN

DELAY:  MOV     R2,#10;延时子程序 10*1*20*2=0.4ms
D2:     MOV     R3,#1
D1:     MOV     R4,#20
D3:     DJNZ    R4,D3
        DJNZ    R3,D1
        DJNZ    R2,D2
        RET

        END
```

2.3 遇到的问题及解决方法：

流水灯程序、静态数码管驱动程序编写都比较顺利，初次不清楚实验器材静态和动态数码管是共阴极还是共阳极显示，经过测试发现静态数码管是共阳极，动态数码管是共阴极的，与教材上有所不同；在动态数码管驱动程序的编写上出了一些问题，先是不清楚怎么找 30h~37h 缓冲区，然后是在延时的设置上出现了问题，不过最终通过检查程序解决了这些问题，结果第二次实验后对单片机汇编有了更深入的理解。

第三次实验：学习使用普中仿真器进行硬件仿真，独立键盘和矩阵式键盘等

3.1 设计要求：

1) 实验二十，独立键盘

   要求：完成 8 个独立式键盘的识别并显示在 LED 灯上；

2) 实验二十一，矩阵式键盘

   要求：

   ① 键值显示在静态数码管上；

   ② 实现循环动态显示的矩阵式键盘。

3.2 实验代码：

1) IndependentKeyboard_0.asm(实现独立键盘控制 LED 灯)

```
        ORG     0000H
        AJMP    START
        ORG     0100H

START:  MOV     SP,#60H
MAIN:   LCALL   KEY1;跳转至 KEY1
        LJMP    MAIN;重复循环
KEY1:   MOV     P1,#0FFH;P1 口全写入 1
        MOV     B,A;
        MOV     P2,B;
        MOV     A,P1;读入八个按键的状态，有键按下则为 0
        CPL     A;全部取反，有键按下则为 1
        JZ      KEY1;如果累加器 A 全为 0，则无键按下，跳转至 KEY1 重新开始
        LCALL   D10ms;去抖动
        MOV     A,P1;读入八个按键的状态，有键按下则为 0
        CPL     A;全部取反，有键按下则为 1
        JZ      KEY1;如果累加器 A 全为 0，则无键按下，跳转至 KEY1 重新开始
        JB      ACC.0,PK0
        JB      ACC.1,PK1
        JB      ACC.2,PK2
        JB      ACC.3,PK3
        JB      ACC.4,PK4
        JB      ACC.5,PK5
        JB      ACC.6,PK6
```

```
        JB      ACC.7,PK7
        RET

PK0:    LCALL   PKEY0
        LJMP    KEY1
PK1:    LCALL   PKEY1
        LJMP    KEY1
PK2:    LCALL   PKEY2
        LJMP    KEY1
PK3:    LCALL   PKEY3
        LJMP    KEY1
PK4:    LCALL   PKEY4
        LJMP    KEY1
PK5:    LCALL   PKEY5
        LJMP    KEY1
PK6:    LCALL   PKEY6
        LJMP    KEY1
PK7:    LCALL   PKEY7
        LJMP    KEY1

PKEY0:  MOV         B,A;
        MOV         P2,B;
        JNB     P1.0,PKEY0
        LCALL   D10ms
        JNB     P1.0,PKEY0
        RET
PKEY1:  MOV         B,A;
        MOV         P2,B;
        JNB     P1.1,PKEY1
        LCALL   D10ms
        JNB     P1.1,PKEY1
        RET
PKEY2:  MOV         B,A;
        MOV         P2,B;
        JNB     P1.2,PKEY2
        LCALL   D10ms
        JNB     P1.2,PKEY2
        RET
PKEY3:  MOV         B,A;
        MOV         P2,B;
        JNB     P1.3,PKEY3
        LCALL   D10ms
        JNB     P1.3,PKEY3
        RET
```

```
PKEY4:  MOV     B,A;
        MOV     P2,B;
        JNB     P1.4,PKEY4
        LCALL   D10ms
        JNB     P1.4,PKEY4
        RET
PKEY5:  MOV     B,A;
        MOV     P2,B;
        JNB     P1.5,PKEY5
        LCALL   D10ms
        JNB     P1.5,PKEY5
        RET
PKEY6:  MOV     B,A;
        MOV     P2,B;
        JNB     P1.6,PKEY6
        LCALL   D10ms
        JNB     P1.6,PKEY6
        RET
PKEY7:  MOV     B,A;
        MOV     P2,B;
        JNB     P1.7,PKEY7
        LCALL   D10ms
        JNB     P1.7,PKEY7
        RET

D10ms:  MOV     R7,#25
D1:     MOV     R6,#200
        DJNZ    R6,$
        DJNZ    R7,D1
        RET

        END
```

2) IndependentKeyboard_1.asm(实现独立键盘控制静态数码管灯)

```
        ORG     0000H
        AJMP    START
        ORG     0100H

START:  MOV     SP,#60H
MAIN:   MOV     DPTR,#TABLE
        LCALL   KEY1;跳转至 KEY1
        LJMP    MAIN;重复循环
KEY1:   MOV     P1,#0FFH;P1 口全写入 1
        MOV     A,#0FFH
        MOV     P2,A;
```

```
        MOV     A,P1;读入八个按键的状态，有键按下则为 0
        CPL     A;全部取反，有键按下则为 1
        JZ      KEY1;如果累加器 A 全为 0，则无键按下，跳转至 KEY1 重新开始
        LCALL   D10ms;去抖动
        MOV     A,P1;读入八个按键的状态，有键按下则为 0
        CPL     A;全部取反，有键按下则为 1
        JZ      KEY1;如果累加器 A 全为 0，则无键按下，跳转至 KEY1 重新开始
        JB      ACC.0,PK0
        JB      ACC.1,PK1
        JB      ACC.2,PK2
        JB      ACC.3,PK3
        JB      ACC.4,PK4
        JB      ACC.5,PK5
        JB      ACC.6,PK6
        JB      ACC.7,PK7
        RET

PK0:    LCALL   PKEY0
        LJMP    KEY1
PK1:    LCALL   PKEY1
        LJMP    KEY1
PK2:    LCALL   PKEY2
        LJMP    KEY1
PK3:    LCALL   PKEY3
        LJMP    KEY1
PK4:    LCALL   PKEY4
        LJMP    KEY1
PK5:    LCALL   PKEY5
        LJMP    KEY1
PK6:    LCALL   PKEY6
        LJMP    KEY1
PK7:    LCALL   PKEY7
        LJMP    KEY1

PKEY0:  MOV     A,#01H
        MOVC    A,@A+DPTR;静态显示
        MOV     P2,A
        JNB     P1.0,PKEY0
        LCALL   D10ms
        JNB     P1.0,PKEY0
        RET
PKEY1:  MOV     A,#02H
        MOVC    A,@A+DPTR;静态显示
        MOV     P2,A
```

```
        JNB     P1.1,PKEY1
        LCALL   D10ms
        JNB     P1.1,PKEY1
        RET
PKEY2:  MOV     A,#03H
        MOVC    A,@A+DPTR;静态显示
        MOV     P2,A
        JNB     P1.2,PKEY2
        LCALL   D10ms
        JNB     P1.2,PKEY2
        RET
PKEY3:  MOV     A,#04H
        MOVC    A,@A+DPTR;静态显示
        MOV     P2,A
        JNB     P1.3,PKEY3
        LCALL   D10ms
        JNB     P1.3,PKEY3
        RET
PKEY4:  MOV     A,#05H
        MOVC    A,@A+DPTR;静态显示
        MOV     P2,A
        JNB     P1.4,PKEY4
        LCALL   D10ms
        JNB     P1.4,PKEY4
        RET
PKEY5:  MOV     A,#06H
        MOVC    A,@A+DPTR;静态显示
        MOV     P2,A
        JNB     P1.5,PKEY5
        LCALL   D10ms
        JNB     P1.5,PKEY5
        RET
PKEY6:  MOV     A,#07H
        MOVC    A,@A+DPTR;静态显示
        MOV     P2,A
        JNB     P1.6,PKEY6
        LCALL   D10ms
        JNB     P1.6,PKEY6
        RET
PKEY7:  MOV     A,#08H
        MOVC    A,@A+DPTR;静态显示
        MOV     P2,A
        JNB     P1.7,PKEY7
        LCALL   D10ms
```

```
        JNB     P1.7,PKEY7
        RET

D10ms:  MOV     R7,#25
D1:     MOV     R6,#200
        DJNZ    R6,$
        DJNZ    R7,D1
        RET
TABLE:  DB      0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H

        END
```

3) MatrixKeyboard_0.asm(实现矩阵式键盘控制静态数码管灯)

```
        ORG     0000H
        AJMP    START
        ORG     0100H
START:  MOV     SP,#60H
        MOV     DPTR,#TABLE
        LCALL   KEY2
        LJMP    START

KEY2:   LCALL   KS;调用键盘检测子程序 ks
        JNZ     K1;无键按下则 A 为 0，有键按下则跳转至 K1
        LCALL   D10ms
        RET
K1:     LCALL   D10ms;去抖动 10ms
        LCALL   KS;再次检测
        JNZ     K2;仍然有键按下则跳转 K2，否则回到 KEY2 重新开始第一遍检测
        AJMP    KEY2

K2:     MOV     R2,#0EFH;1110'1111 暂存进 R2，列 P1.4 为 0
        MOV     R4,#00H;0 列号送入 R4 暂存
K3:     MOV     P1,R2;将列扫描值送入 P1 口，P1.4 为 0
L0:     JB      P1.0,L1;判 0 行线电平，P1.0 为 1 则无按下，跳转至 L1 检测 1 行
        MOV     A,#00H;若检测出，则将 0 行首键盘号送进 ACC
        AJMP    LK;跳转至 LK 计算行号+列号
L1:     JB      P1.1,L2
        MOV     A,#04H
        AJMP    LK
L2:     JB      P1.2,L3
        MOV     A,#08H
        AJMP    LK
L3:     JB      P1.3,NEXT;若 0~3 行均无检测出，跳入 NEXT，准备检测下一列
        MOV     A,#0CH
        AJMP    LK
```

```
NEXT:   INC     R4;列号加 1
        MOV     A,R2;将 R2 扫描值送入 A
        JNB     ACC.7,KEY2;判断 A 的最高位是否为 0，即是已经 0111’0000，若是则已
扫描完，则返回 KEY2
        RL      A;若不是，则 A 左移一位，扫描下一列
        MOV     R2,A;下一列的 A 返还给 R2
        AJMP    K3;回到 K3 开始扫描下一列

LK:     ADD     A,R4;列号 R4，行号 ACC
        MOVC    A,@A+DPTR
        MOV     P2,A
        ACALL   K4

K4:     LCALL   KS;检测是否键已经松开
        JNZ     K4
        LCALL   D10ms
        JNZ     K4
        LCALL   KEY2

KS:     MOV     P1,#0FH
        MOV     A,P1
        XRL     A,#0FH
        RET

D10ms:  MOV     R7,#25
D1:     MOV     R6,#200
        DJNZ    R6,$
        DJNZ    R7,D1
        RET

TABLE:  DB      0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H
        DB      80H,90H,88H,83H,0C6H,0A1H,86H,8EH
        DB      0FFH
        END
```

4) MatrixKeyboard_1.asm(实现矩阵式键盘控制动态码管灯循环显示)

```
//P0 接位选，P1 接矩阵式键盘，P2 接数值
        ORG     0000H
        AJMP    RESET
        ORG     0100H
RESET:  MOV     SP,#60H
        MOV     DPTR,#TABLE
        MOV     R5,#80H;8'b1000 0000
        MOV     R0,#08H
```

```
        MOV     B,#00H
MAIN:   MOV     P2,#00H
        LCALL   KEY2
        DJNZ    R0,MAIN
        LJMP    RESET

KEY2:   LCALL   KS;调用键盘检测子程序 ks
        JNZ     K1;无键按下则 A 为 0，有键按下则跳转至 K1(非 0 转移)
        LCALL   D10ms
        INC     R0;如果无键按下，R0 值需要恢复
        RET

K1:     LCALL   D10ms;去抖动 10ms
        LCALL   KS;再次检测
        JNZ     K2;仍然有键按下则跳转 K2，否则回到 KEY2 重新开始第一遍检测
        AJMP    KEY2

K2:     MOV     R2,#0EFH;1110'1111 暂存进 R2，列 P1.4 为 0
        MOV     R4,#00H;0 列号送入 R4 暂存
K3:     MOV     P1,R2;将列扫描值送入 P1 口，P1.4 为 0
L0:     JB      P1.0,L1;判 0 行线电平，P1.0 为 1 则无按下，跳转至 L1 检测 1 行
        MOV     A,#00H;若检测出，则将 0 行首键盘号送进 ACC
        AJMP    LK;跳转至 LK 计算行号+列号
L1:     JB      P1.1,L2
        MOV     A,#04H
        AJMP    LK
L2:     JB      P1.2,L3
        MOV     A,#08H
        AJMP    LK
L3:     JB      P1.3,NEXT;若 0~3 行均无检测出，跳入 NEXT，准备检测下一列
        MOV     A,#0CH
        AJMP    LK

NEXT:   INC     R4;列号加 1
        MOV     A,R2;将 R2 扫描值送入 A
        JNB     ACC.7,KEY2;判断 A 的最高位是否为 0，即是已经 0111’0000，若是则已
扫描完，则返回 KEY2
        RL      A;若不是，则 A 左移一位，扫描下一列
        MOV     R2,A;下一列的 A 返还给 R2
        AJMP    K3;回到 K3 开始扫描下一列

LK:     ADD     A,R4;列号 R4，行号 ACC，加到 A
        ACALL   LEFT
        MOVC    A,@A+DPTR
```

```
        MOV     P0,B
        MOV     P2,A
        ACALL   K4
        LCALL   DELAY
        DJNZ    R0,KEY2
        LJMP    RESET

K4:     LCALL   KS;检测是否键已经松开
        JNZ     K4
        LCALL   D10ms
        JNZ     K4
        RET

KS:     MOV     P1,#0FH
        MOV     A,P1
        XRL     A,#0FH
        RET

LEFT:   MOV     B,A
        MOV     A,R5
        RL      A
        MOV     R5,A
        MOV     A,B
        MOV     B,R5
        RET

TABLE:  DB      3FH,06H,5BH,4FH,66H,6DH,7DH,07H
        DB      7FH,6FH,77H,7CH,39H,5EH,79H,71H
        DB      00H

D10ms:  MOV     R7,#25
D1:     MOV     R6,#200
        DJNZ    R6,$
        DJNZ    R7,D1
        RET

DELAY:  MOV     R7,#10
D2:     MOV     R6,#20
D3:     DJNZ    R6,D3
        DJNZ    R7,D2
        RET

        END
```

3.3 遇到的问题及解决方法：

本次实验中独立键盘显示 LED 灯以及静态数码管的显示都比较简单，但是动态数码管由于上次实验的程序还没有在仿真器上测试过，所以出现了一些小问题，扫描程序，显示程序，缓冲区程序结合的时候出现了无法循环显示的问题，经过重新改进动态显示模块，最后成功完成动态数码管与矩阵式键盘的结合与循环显示。

第四次实验：学习使用普中 ISP 自动下载软件到 STC89C516 单片机中（实验十），单片机中断系统实验

4.1 设计要求：

学习使用普中 ISP 自动下载软件到 STC89C516 单片机中(参考实验十)，预习 AT89S51 单片机的中断系统，完成实验包括：

1) 实验二十八，外部中断 0；
2) 实验二十九，外部中断 1；
3) 预习中断优先级设置，完成实验思考题。

思考题，试编写汇编源程序，编译后实现如下功能：

1) 主程序相邻 2 个 LED 灯循环点亮（03H—C0H）1S；外部中断 0 后静态数码管间隔 1S 轮流点亮 0-F 后返回。
2) 主程序相邻 2 个 LED 灯循环点亮（03H—C0H）1S；外部中断 1 后 1 个 LED 灯循环点亮（01H-80H）后返回。
3) 中断优先级实验，主程序相邻 2 个 LED 灯循环点亮（03H—C0H）1S，外部中断 0 后静态数码管间隔 1S 轮流点亮 0-F 后返回，外部中断 1 后 1 个 LED 灯循环点亮（01H-80H）后返回，外部中断 0 优先级高于外部中断 1，实现中断嵌套。

4.2 实验代码：

1) tk1.asm

```
//P1-->>ABCDEF,P3.2 中断接线 k1,P0 接静态数码管
        ORG     0000H
        AJMP    START
        ORG     0003H;外部中断 0 的入口是 0003H
        AJMP    INTT0
        ORG     0100H

START:  MOV     SP,#60H
        SETB    EX0;开中断
        SETB    IT0
        SETB    EA
        MOV     IP,#01H;设置优先级
```

```
        MOV     P3,#0FFH;先将 P3 全部置为高电平
MAIN:   MOV     DPTR,#TABLE
        MOV     R0,#03H
        MOV     R2,#08H
        ACALL   LOOP
        LJMP    MAIN

LOOP:   MOV     A,R0
        MOV     P1,A
        RL      A
        MOV     R0,A
        LCALL   D1S
        DJNZ    R2,LOOP
        RET

INTT0:  MOV     A,P3
        CPL     A
        JZ      RETURN
        LCALL   D10ms   ;去抖动
        MOV     A,P3
        CPL     A
        JZ      RETURN
        JB      ACC.2,Pkey0
        RETI

Pkey0:  MOV     A,#00H
        MOVC    A,@A+DPTR
        CJNE        A,#0FFH,Pkey1
        LJMP    RETURN

Pkey1:  MOV     P0,A
        LCALL   D1S
        INC     DPTR
        LJMP    Pkey0

RETURN:     RETI

TABLE:  DB      0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H
        DB      80H,90H,88H,83H,0C6H,0A1H,86H,8EH
        DB      0FFH

D10ms:  MOV     R7,#25
D4:     MOV     R6,#200
        DJNZ    R6,$
```

```
        DJNZ    R7,D4
        RET

D1S:    MOV     R5,#20
D5:     MOV     R6,#100
D6:     MOV     R7,#248
D7:     DJNZ    R7,D7
        DJNZ    R6,D6
        DJNZ    R5,D5
        RET

        END
```

2) tk2.asm

```
//P3.3 接 k4，P1 接 LED 的 ABCDEF，
        ORG     0000H
        AJMP    START
        ORG     0013H;外部中断 1 的入口位 0013h
        AJMP    INTT1
        ORG     0100H

START:  MOV     SP,#60H
        SETB    EX1;开中断
        SETB    IT1
        SETB    EA
        MOV     IP,#04H;设置中断 1 为高优先级
        MOV     P3,#0FFH
MAIN:   MOV     R0,#03H
        MOV     R2,#08H
        ACALL   LOOP
        LJMP    MAIN

LOOP:   MOV     A,R0
        MOV     P1,A
        RL      A
        MOV     R0,A
        LCALL   D1S
        DJNZ    R2,LOOP
        RET

INTT1:  MOV     A,P3
        CPL     A
        JZ      RETURN
        LCALL   D10ms   ;去抖动
        MOV     A,P3
```

```
        CPL     A
        JZ      RETURN
        JB      ACC.3,Pkey0
        RETI

Pkey0:
        MOV     R1,#01H
        MOV     R3,#08H
        ACALL   Pkey1
        LJMP    RETURN

Pkey1:  MOV     A,R1
        MOV     P1,A
        RL      A
        MOV     R1,A
        LCALL   D1S
        DJNZ    R3,Pkey1
        RET

RETURN:     RETI

D10ms:  MOV     R7,#25
D4:     MOV     R6,#200
        DJNZ    R6,$
        DJNZ    R7,D4
        RET
D1S:        MOV     R5,#20
D5:     MOV     R6,#100
D6:     MOV     R7,#248
D7:     DJNZ    R7,D7
        DJNZ    R6,D6
        DJNZ    R5,D5
        RET
        END
```

3) tk3.asm

```
//P1 接 LED 的 ABCDEF，P3.2 接 K1，P3.3 接 K4，P0 接静态数码管 ABCDEF
        ORG     0000H
        AJMP    START

        ORG     0003H
        AJMP    INTT0

        ORG     0013H
        AJMP    INTT1
```

```
        ORG     0100H
START:  MOV     SP,#60H
        MOV     PSW,#00H
        SETB    EX0
        SETB    IT0
        SETB    EX1
        SETB    IT1
        SETB    EA
        MOV     IP,#01H
        MOV     P3,#0FFH
MAIN:   MOV     DPTR,#TABLE
        MOV     R0,#03H
        MOV     R2,#08H
        ACALL   LOOP
        LJMP    MAIN

LOOP:   MOV     A,R0
        MOV     P1,A
        RL      A
        MOV     R0,A
        LCALL   D1S
        DJNZ    R2,LOOP
        RET

INTT0:  PUSH    PSW
        PUSH    ACC
        MOV     PSW,#08H
        MOV     A,P3
        CPL     A
        JZ      RETURN
        LCALL   D10ms               //去抖动
        MOV     A,P3
        CPL     A
        JZ      RETURN
        JB      ACC.2,Pkey0
        LJMP    RETURN

INTT1:  PUSH    PSW
        PUSH    ACC
        MOV     PSW,#10H
        MOV     A,P3
        CPL     A
        JZ      RETURN
```

```
        LCALL   D10ms          //去抖动
        MOV     A,P3
        CPL     A
        JZ      RETURN
        JB      ACC.3,Pkey2
        LJMP    RETURN
/**********************/
Pkey0:  MOV     A,#00H
        MOVC    A,@A+DPTR
        CJNE    A,#0FFH,Pkey1
        LJMP    RETURN
Pkey1:  MOV     P0,A
        LCALL   D1S
        INC     DPTR
        LJMP    Pkey0
/**********************/
Pkey2:  MOV     R1,#01H
        MOV     R3,#08H
        ACALL   Pkey3
        LJMP    RETURN
Pkey3:  MOV     A,R1
        MOV     P1,A
        RL      A
        MOV     R1,A
        LCALL   D1S
        DJNZ    R3,Pkey3
        RET
/**********************/
RETURN:
        POP     ACC
        POP     PSW
        RETI

TABLE:  DB      0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H
        DB      80H,90H,88H,83H,0C6H,0A1H,86H,8EH
        DB      0FFH

D10ms:  MOV     R7,#25
D4:     MOV     R6,#200
        DJNZ    R6,$
        DJNZ    R7,D4
        RET

D1S:    MOV     R5,#20
```

```
D5:     MOV     R6,#100
D6:     MOV     R7,#248
D7:     DJNZ    R7,D7
        DJNZ    R6,D6
        DJNZ    R5,D5
        RET

        END
```

4.3 遇到的问题及解决方法：

单独使用外部中断 0 和外部中断 1 的时候比较简单，虽然有发现中断返回时的延迟比较高，但没有在意，知道结合使用外部中断 1 和外部中断 0 的时候发现中断返回时需要四五秒，便开始一步步调试程序检查问题，最后发现是因为使用的延迟程序中 R5，R6，R7 的问题，便想到了最开始学的程序状态字寄存器 PSW，通过设置 PSW 中的 RS1 和 RS0 选择四组工作寄存区，上电时默认为 0 区，可以在中断中选择另外的寄存区，来将使用的寄存器分开进而不会产生高延迟现象。LED 循环闪烁要将 74HC245 模块的 J21 跳线帽拔掉或者插最左边两个，如果用 74H138 译码器位选的话需要将 J15 和 J16 跳线帽插上。

第五次实验：单片机定时器系统实验

5.1 设计要求

1) 实验三十，定时器 0

用单片机和 LED 灯，编写定时器 0 中断程序，使用单片机内部定时器实现准确延时。循环点亮 小灯 1 秒，熄灭 1 秒。

2) 实验三十一，定时器 1

用单片机和静态数码管，编写定时器 1 中断程序，使用单片机内部定时器实现准确延时，静态数 码管间隔一秒循环显示 0-F。

3) 实验三十二，交通灯

用单片机、交通灯模块、动态数码管模块，利用定时器、十进制数拆分，编写交通灯控制程序， 用动态数码管显示秒倒计时。

5.2 实验代码

1) timer0.asm

```
//P2.0 接 LED 模块 J12 的 A
    ORG         0000H
RESET:
    LJMP    MAIN
    ORG         000BH;定时器中断 0 入口
    LJMP    T0_INT
    ORG         0100H

MAIN:
    MOV         SP,#60H
    MOV         TMOD,#01H;计时工作方式 1
    MOV         30H,#00H;定时器超时一百次，达到 64h 即一秒
    ;12MHz，机器周期 1μs，需要 10000 个计数，初值=65536-10000=55536=D8F0H
    MOV         TH0,#0D8H
    MOV         TL0,#0F0H
    SETB    P2.0
    SETB    ET0;开放 T0 中断和总中断
    SETB    EA
    SETB    TR0;启动 T0
    JNB     TF0,$;wait for TF0
```

```
T0_INT:
    PUSH    PSW;保护状态字寄存器
    PUSH    ACC
    CLR     EA;关闭总中断
    MOV         TH0,#0D8H
    MOV         TL0,#0F0H
    INC     30H
    MOV         A,30H
    CJNE    A,#100,CONTINUE_T0
    MOV         30H,#00H;定时器超时一百次，达到 64h 即一秒
    CPL     P2.0

CONTINUE_T0:
    SETB    EA;开总中断
    POP     ACC
    POP     PSW;保护状态字寄存器
    RETI
    END
```

2) timer1.asm

```
//P0 用杜邦线接到静态数码管 JP3
    ORG         0000H
RESET:
    LJMP    MAIN
    ORG         001BH;定时器中断 1 入口
    LJMP    T1_INT
    ORG         0100H

MAIN:
    MOV         SP,#60H
    MOV         DPTR,#TABLE
    MOV         TMOD,#10H;计时工作方式 1
    MOV         P0,#0FFH
    MOV         R0,#00H
    MOV         30H,#00H;定时器超时一百次，达到 64h 即一秒
    ;12MHz，机器周期 1μs，需要 10000 个计数，初值=65536-10000=55536=D8F0H
    MOV         TH1,#0D8H
    MOV         TL1,#0F0H
    SETB    ET1;开放 T1 中断和总中断
    SETB    EA
    SETB    TR1;启动 T1
    JNB     TF1,$;wait for TF1

T1_INT:
    PUSH    PSW;保护状态字寄存器
```

```
    PUSH    ACC
    CLR     EA;关闭总中断
    MOV         TH1,#0D8H
    MOV         TL1,#0F0H
    INC     30H
    MOV         A,30H
    CJNE    A,#100,CONTINUE_T0
    MOV         A,R0
    MOVC    A,@A+DPTR
    CJNE    A,#0FFH,SHOW
    MOV         R0,#00H

SHOW:
    MOV         P0,A
    INC     R0
    SETB    EA;开总中断
    POP     ACC
    POP     PSW;保护状态字寄存器
    RETI

CONTINUE_T0:
    SETB    EA;开总中断
    POP     ACC
    POP     PSW;保护状态字寄存器
    RETI

TABLE:
    DB      0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H
    DB      80H,90H,88H,83H,0C6H,0A1H,86H,8EH
    DB      0FFH

    END
```

3) traffic.asm

```
//交通灯模块、动态数码管模块，利用定时器、十进制数拆分
//用 8 根的排线连接单片机 P0 的 JP10 到动态数码管的 J12，用 8 根的排线连接单片机 P1 的 JP8 到 交通灯模块的 JP1
//单片机的 P22、P23、P24 分别接到 74HC138 模块的 A、B、C。JP165 跳线帽一定要拔掉，插在其中一根针上，以免丢掉。
//南北绿灯 30s，红灯 30s；黄灯均为 5s；东西绿灯 25s，红灯 35s，
//人行道不显示倒计时，车道显示倒计时
    ORG         0000H
RESET:
```

```
    AJMP    MAIN
    ORG     000BH           //定时器中断 0 入口
    AJMP    INTI0
    ORG     0100H

MAIN:
    MOV         SP,#60H         //计时工作方式 1
    MOV         TMOD,#01H
    MOV         DPTR,#TABLE
//12MHz，机器周期 1μs，需要 10000 个计数，初值=65536-10000=55536=D8F0H
    MOV         TH0,#0D8H
    MOV         TL0,#0F0H
    SETB    ET0             //开放 T0 中断和总中断
    SETB    EA
    SETB    TR0             //启动 T0
    MOV     R0,#01          //计数器初值为 01
    MOV         30H,#00H        //定时器超时一百次，达到 64h 即一秒
    MOV         31H,#30         //南北方向红黄绿时间，30，5，30
    MOV         32H,#35         //东西方向红黄绿时间，35，5，25
    MOV     P1,#11001101B   //110 东西红'011 南北绿'01 人行南北绿
    CLR     P3.0        //10 人行东西红
    SETB    P3.1
    LCALL   CALCU
    LJMP    DISPLAY

CALCU:
/***********南北方向倒计时计算*******************/
    MOV     R1,#10          //除数为 10
    MOV     A,31H           //倒计时数字放到 A
    MOV     B,R1            //除数 10 放到 B
    DIV     AB              //A 为商，B 为余数
    MOV     R4,A            //R4 存放商
    MOV     R5,B            //R5 存放余数
/***********东西方向倒计时计算*******************/
    MOV     R1,#10          //除数为 10
    MOV     A,32H           //倒计时数字放到 A
    MOV     B,R1            //除数 10 放到 B
    DIV     AB              //A 为商，B 为余数
    MOV     R6,A            //R6 存放商
    MOV     R7,B            //R7 存放余数
    RET

DISPLAY:
/***********南北方向倒计时显示*******************/
```

```
    CLR     P2.2
    CLR     P2.3
    CLR     P2.4
    MOV     A,R4            //LED0 输出商
    MOVC    A,@A+DPTR
    MOV     P0,A
    LCALL   D04MS

    SETB    P2.2
    MOV     A,R5            //LED1 输出余数
    MOVC    A,@A+DPTR
    MOV     P0,A
/***********东西方向倒计时显示*******************/
    CLR     P2.2
    CLR     P2.3
    SETB    P2.4
    LCALL   D04MS
    MOV     A,R6            //LED4 输出商
    MOVC    A,@A+DPTR
    MOV     P0,A
    LCALL   D04MS

    SETB    P2.2
    MOV     A,R7            //LED5 输出余数
    MOVC    A,@A+DPTR
    MOV     P0,A
/*****************************************/
    JNB     TF0,DISPLAY         //wait for TF0

INTI0:
    PUSH    PSW                 //保护状态字寄存器
    PUSH    ACC
    CLR     EA              //关闭总中断
    MOV         TH0,#0D8H       //定时 10ms
    MOV         TL0,#0F0H
    INC     30H
    MOV         A,30H
    CJNE    A,#100,OUT
    MOV         30H,#00H        //定时器超时一百次，达到 64h 即一秒
    INC     R0              //计数器+1
    DEC         31H
    DEC         32H
```

```
/**********四种状态互相转变******************/
JMP5SN:
    CJNE    R0,#31,JMP30EW
    MOV     P1,#11010110B    //110 东西红'101 南北黄'10 人行南北红
    CLR     P3.0             //10 人行东西红
    SETB    P3.1
    MOV         31H,#5
    LCALL   CALCU
    SJMP    OUT
JMP30EW:
    CJNE    R0,#36,JMP5EW
    MOV     P1,#01111010B    //011 东西绿'110 南北红'10 人行南北红
    SETB    P3.0             //01 人行东西绿
    CLR     P3.1
    MOV         31H,#30
    MOV         32H,#25
    LCALL   CALCU
    SJMP    OUT
JMP5EW:
    CJNE    R0,#66,JMP30SN
    MOV     P1,#10111010B    //101 东西黄'110 南北红'10 人行南北红
    SETB    P3.0             //01 人行东西绿
    CLR     P3.1
    MOV         32H,#5
    LCALL   CALCU
    SJMP    OUT
JMP30SN:
    CJNE    R0,#71,OUT
    MOV     R0,#1
    MOV     P1,#11001101B    //110 东西红'011 南北绿'01 人行南北绿
    CLR     P3.0             //10 人行东西红
    SETB    P3.1
    MOV         31H,#30
    MOV         32H,#35
    LCALL   CALCU
    SJMP    OUT

OUT:
    SETB    EA               //开总中断
    POP     ACC
    POP     PSW                  //保护状态字寄存器
    RETI

D04MS:                       //延时 0.4ms
```

```
    MOV         R3,#2
D1:
    MOV         R2,#10
D2:
    DJNZ        R2,D2
    DJNZ        R3,D1
    RET

TABLE:  DB      3FH,06H,5BH,4FH,66H,6DH,7DH,07H     //共阴极 0~9
        DB      7FH,6FH
        DB      00H
/*TABLE:    DB      3FH,06H,5BH,4FH,66H,6DH,7DH,07H     //共阴极 0~F
          DB      7FH,6FH,77H,7CH,39H,5EH,79H,71H
          DB      00H*/

/*TABLE:    DB      0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H//共阴极 0~9
          DB      80H,90H
          DB      0FFH*/

/*TABLE:    DB      0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H//共阴极 0~F
          DB      80H,90H,88H,83H,0C6H,0A1H,86H,8EH
          DB      0FFH*/

        END
```

5.3 遇到的问题及解决方法：

这次实验中前两个都比较简单，在定时时长上有出现过一些小问题，后来经过调试，晶振是 12MHz，一个机器周期是 1μs，方式 1 最多定时 65.536ms，故采用 10ms*100 的方式来达到 1s 的定时要求；刚开始在调试过程中没有设置断点时，单步执行，发现需要很长时间，后来在判断是否达到 100 次 10ms 计时语句的后面设置断点然后全速执行，刚好为 1s 左右，达到预期效果。

交通灯系统在观看了几个教学视频之后，可以主要分为几个部分：定时器模块，红绿灯状态转换模块，倒计时数值计算，动态显示模块，仿真器实物连接

倒计时数值计算刚开始没有考虑到有两条行车道，只做了一个倒计时，后来进行了更改；动态显示模块由于之前几次实验都是用的杜邦线直接连接，没有使用三八译码器，所以这次用三八译码器的时候出了点小问题，发现需要把动态显示 LED 的位选用短接帽短接，否则只有 LED 灯会亮，段码管不会亮。

第六次实验：单片机 IO 扩展-并转串 串转并 双机通信

6.1 设计要求

1) 单片机 IO 扩展-并转串-74LS165

   P1.6 置数，P1.7 数据，P3.6 位移时钟，JP165 跳线帽一定要街上

2) 单片机 IO 扩展-串转并-74HC595

   P3.4 串行数据输入，P3.6 移位脉冲，P3.5 数据输出，JP595 一定要接上

3) 甲机将内部 RAM 30h~4Fh 的内容发送给乙机

   - 波特率设置初始化：定时器 T1 模式 2 工作，计数常数 0F4H；
   - PCON 的 SMOD=1；
   - 串行口初始化：方式 1 工作，允许接收；
   - 内部 RAM 和工作寄存器设置：R0 存放发送的数据块首址；R7 存放发送的数据块长度； R6 为校验和寄存器。

6.2 实验代码

1) task1.asm

```
ORG 0000H//并转串
LJMP MAIN
ORG 0100H
MAIN:
    MOV R0,#08H
    CLR C
    MOV A,#00H
    ACALL GETDATA
    CJNE A,#0FFH,NEXT
    LJMP MAIN
NEXT:
    CPL A
    MOV P0,A
    LJMP MAIN
GETDATA:
    CLR P1.6
    NOP
    SETB P1.6
    NOP
LOOP:
```

```
    CLR P3.6
    NOP
    MOV C,P1.7
    RLC A
    SETB P3.6
    DJNZ R0,LOOP
    MOV R0,#08H
    RET
END
```

2) task2.asm

```
ORG 0000H   //串转并
LJMP MAIN
ORG 0100H
MAIN:
    MOV A,#01H
    MOV R0,#08H
NEXT:
    ACALL SENDDATA
    ACALL DELAY
    LJMP NEXT
SENDDATA:
    SETB P3.6
    SETB P3.5
LOOP:
    RLC A
    MOV P3.4,C
    CLR P3.6
    NOP
    NOP
    SETB P3.6
    DJNZ R0,LOOP
    MOV R0,#08H
    CLR P3.5
    NOP
    SETB P3.5
    RET
DELAY:
        MOV 20H,#100
DEL0: MOV 21H,#10
DEL1: MOV 22H,#200
DEL2: DJNZ 22H,DEL2
            DJNZ 21H,DEL1
            DJNZ 20H,DEL0
    RET
```

```
END
```

3) tx.asm

```
        ORG     0000H
        LJMP    START
        ORG     1000H

START:
        MOV     SP,#60H
        MOV     TMOD,#20H       //定时器 0，定时器工作模式 2 工作
        MOV     TH1,#0FAH       //模式 2 自动重装载的初值
        MOV     TH1,#0FAH
        SETB    TR1             //开始计时
        MOV     SCON,#50H       //方式 1 工作,允许接收-->4.8kbit/s
        MOV     PCON,#00H       //SMOD=0
        MOV     R0,30H          //存放发送的数据块首地址
        MOV     R7,#20H         //存放发送的数据块长度
        MOV     R6,#00H         //校验和，长度字节与数字字节的累加和

TX_ACK:
        MOV     A,#06H          //发送 06H 询问是否可以接收数据
        MOV     SBUF,A

WAIT1:
        JNB     TI,WAIT1        //等待发送完一个字节
        CLR     TI              //清除发送完毕 TI 标志位

RX_YES:
        JNB     RI,RX_YES       //等待乙机回答是否可以接收
        CLR     RI              //清除接收完毕 RI 标志位

NEXT1:
        MOV     A,SBUF          //接收到乙机发送过来的 ACK
        CJNE    A,#00H,TX_ACK //若为 00H 则表示可以接收数据，往下启动发送，
否则再次询问

TX_LENGTH:
        MOV     A,R7            //先发送字节长度数 R7
        MOV     SBUF,A

WAIT2:
        JNB     TI,WAIT2        //等待数据发送完毕
        CLR     TI              //清除发送完毕 TI 标志位
        MOV     R6,A            //增加校验和
```

```
TX_NEWS:                        //查询发送数据
        MOV     A,@R0
        MOV     SBUF,A

TX_NEWS_WAIT:
        JNB     TI,TX_NEWS_WAIT
        CLR     TI              //清除发送完毕 TI 标志位
        MOV     A,R6
        ADD     A,@R0
        MOV     R6,A
        INC     @R0
        DJNZ    R7,TX_NEWS      //如果没发送完 32 个数据，继续发送

TX_CHECK:
        MOV     A,R6
        ADD     A,@R0
        MOV     R6,A
        MOV     A,R6            //发送校验位
        MOV     SBUF,A

WAIT3:                          //等待发送完校验位
        JNB     TI,WAIT3
        CLR     TI

WAIT4:                          //等待接收完校验位
        JNB     RI,WAIT4
        CLR     RI

IF_0FH:
        MOV     A,SBUF          //将接收到校验是否正确的恢复存到累加器
        CJNE    A,#0FH,START

HERE:
        SJMP    HERE

        END
```

4) rx.asm

```
        ORG     0000H
        LJMP    START
        ORG     0023H
        LJMP    IF_06H
        ORG     1000H

START:
```

```
        MOV     SP,#60H
        MOV     TMOD,#20H       //定时器 0，定时器工作模式 2 工作
        MOV     TH1,#0FAH       //模式 2 自动重装载的初值
        MOV     TH1,#0FAH
        SETB    TR1             //开始计时
        MOV     SCON,#50H       //方式 1,允许接收-->4.8kbit/s
        MOV     PCON,#00H       //SMOD=0
        MOV     IE,#90H
        MOV     R0,30H          //存放发送的数据块首地址
        MOV     R7,#00H         //存放发送的数据块长度
        MOV     R6,#00H         //校验和，长度字节与数字字节的累加和

HERE:
        SJMP    HERE

IF_06H:
        PUSH    ACC
        PUSH    PSW
        CLR     RI              //清除接收完毕 RI 标志位
        MOV     A,SBUF          //核对握手信号是不是 06H
        CJNE    A,#06H,TX_15H   //如果是 06H 则发送 00H 应答，否则发送 15H 拒
绝

TX_00H:
        MOV     A,#00H
        MOV     SBUF,A          //发送出 00H 表示可以接收数据
        LJMP    HERE_RE

TX_15H:
        MOV     A,#15H          //发送 15H 表示不可以接收
        MOV     SBUF,A
        LJMP    RETURN

HERE_RE:
        JNB     TI,HERE_RE      //等待发送完毕，发送完毕准备接收
        CLR     TI

HAVE1:
        JNB     RI,HAVE1        //等待接收数据长度
        CLR     RI
        MOV     A,SBUF
        MOV     R7,A            //R7 存数据长度
        MOV     R6,A            //R6 存校验和
```

```
HAVE2:
        JNB     RI,HAVE2        //等待接收正式数据
        CLR     RI
        MOV     A,SBUF
        MOV     @R0,A
        MOV     A,R6
        ADD     A,@R0
        MOV     R6,A
        INC     @R0
        DJNZ    R7,HAVE2        //如果没接收完完 20 个数据，继续接收

RX_CHECK:                       //接收校验和
        JNB     RI,RX_CHECK
        CLR     RI
        MOV     A,SBUF
        MOV     39H,A
        CJNE    A,39H,TX_ERR    //如果校验正确，则继续向下 ok，错误则发送#F0H

TX_OK:                          //校验正确，发送 0FH
        MOV     A,#0FH
        MOV     SBUF,A
        LJMP    HERE_END

TX_ERR:                         //校验错误，发送 F0H
        MOV     A,#0F0H
        MOV     SBUF,A

HERE_END:                       //等待发送完毕
        JNB     TI,HERE_END
        CLR     TI

RETURN:
        POP     PSW
        POP     ACC
        RETI

        END
```

6.3 遇到的问题及解决办法

提前写好了双机通信的发送和接收模块，但是现场一个人在调试两台仪器有点手足无措。同时一开始接收模块没有设定中断接收，导致一直接收不到发送机发送出来的握手信号，浪费了比较多的时间。RI 和 TI 标志位清除一开始使用不习惯，

后来慢慢的在每个子程序段都差不多使用。DJNZ 自减 Ri 并判断的语句使用，一开始先把 R7 放到了累加器，然后累加器自减，再是判断是否为 0，增加了代码的繁琐程度。

第七次实验： A/D 转换

7.1 设计要求

实验三十七, AD 模数转换器

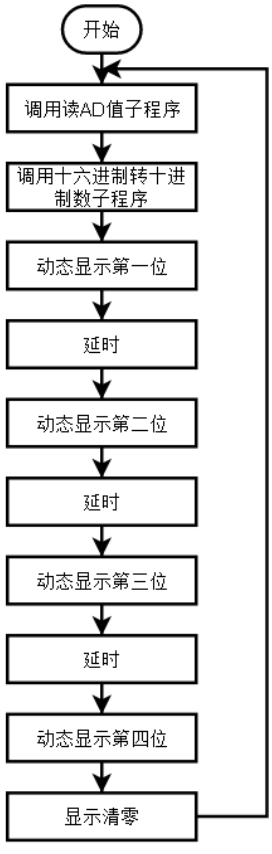
提示：1、AD 转换器 XPT2046 是 12 位的串行口（SPI）输出的可编程控制 AD 芯片，转换时间约 10uS,控制字分别为 94H 可调电位器、0A4H 光敏电阻、0D4H 热敏电阻。

2、采用 LED（4 位）动态显示，千位、百位、十位和个位。动态显示源程序要准备好。

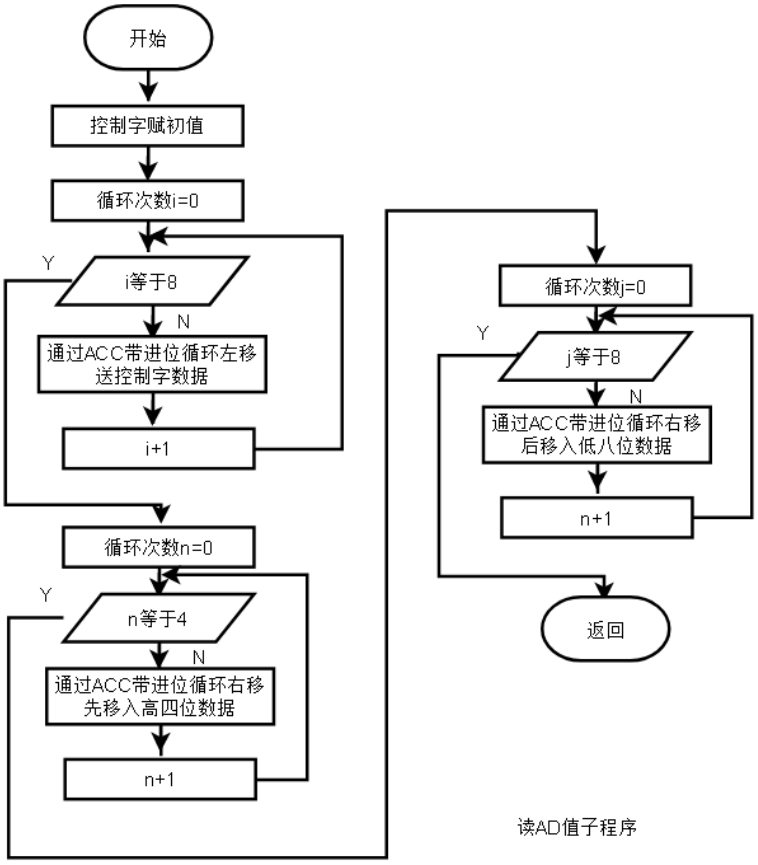
3、000H-FFFH 的温度数据需要转换成十进制数的仟佰拾个位，事先准备转换源程序，提示：可采用循环减法，千位数循环减 3E8H，百位数循环减 64H。

4、电压采集约 1S 采集一次。

7.2 流程图



主程序



读AD值子程序

7.3 实验代码

```
//连接单片机的 JP10 到动态数码管的 J12
//分别连接单片机的 P20、P21、P22 到 74HC138 模块的 A、B、C。
//分别连接单片机的 P34、P35、P36、P37 到 AD/DA 模块的 DI、CS、CLK、DO。
//将 NE555 模块的跳线帽 J11 跳开，跳线帽安装在其中一根上，以免丢掉。
ORG 0000H
LJMP MAIN

ORG 0200H
MAIN:
    MOV         SP,#60H
    ACALL   AD_CHANGE

    MOV     DPTR,#D_table           //动态显示四位

    MOV         A,#00H                  //38 译码器动态显示四位
    MOV     A,R0                    //放个位
    MOVC    A,@A+DPTR
    MOV     P2,#03
    MOV     P0,A
```

```
    ACALL   Delay

    MOV     A,#00H
    MOV     A,R1                    //放十位
    MOVC    A,@A+DPTR
    MOV     P2,#02
    MOV     P0,A
    ACALL   Delay

    MOV     A,#00H
    MOV     A,R2                    //放百位
    MOVC    A,@A+DPTR
    MOV     P2,#01
    MOV     P0,A
    ACALL   Delay

    MOV     A,#00H
    MOV     A,R3                    //放千位
    MOVC    A,@A+DPTR
    MOV         P2,#00
    MOV     P0,A
    ACALL   Delay

    MOV     P0,#00H
    LJMP    MAIN

AD_CHANGE:              //获取 AD 转换结果，共 12 位，串行读入单片机内，用 SPI 总线
    INC     R7
DELL:
    DJNZ    R6,DELL
    CJNE    R7,#0FFH,RETURN
    MOV     R7,#00H
    MOV     R6,#0FFH

    MOV     R0,#0d4H                //R0 控制字   0X94 或 0XB4 电位，0XD4 热敏，0XA4
光敏
    CLR     P3.5                //片选 CS 为低电平，选中 XPT2046
    CLR     P3.6                //时钟脚 I/O CLOCK 位低电平
    MOV     R2,#08H                 //设置循环读入的次数为 8
    MOV     A,R0                    //下一次转换的命令从 R0 送入 A
LOOP0:
    RLC     A
    MOV         P3.4,C
    CLR     P3.6
```

```
    NOP
    SETB    P3.6
    NOP
    DJNZ    R2,LOOP0
    MOV     A,#00H
    MOV     R2,#04H
    NOP
    NOP
    NOP
    NOP
LOOP1:
    MOV     C,P3.7              //读入上一次的转换结果中的 1 位
    RRC     A                   //带进位位的循环右移

    SETB    P3.6            //一个 CLK 时钟
    NOP
    CLR     P3.6
    NOP

    DJNZ    R2,LOOP1            //是否完成 8 次转换结果读入和命令输出？未完成则继续
    RRC     A
    RRC     A
    RRC     A
    RRC     A
    MOV     R1,A                //R1 存高 4 位数据
    MOV     A,#00H              //A 清“0”
    MOV     R2,#08H             //设置 R2 循环次数为 8，为移入 8 位数据准备
LOOP2:
    MOV     C,P3.7              //读入上一次的转换结果中的 1 位
    RRC     A                   //带进位位的循环右移

    CLR     P3.6            //一个 CLK 时钟
    NOP
    SETB    P3.6
    NOP

    DJNZ    R2,LOOP2            //是否完成 8 次转换结果读入和命令输出？未完成则继续

    SWAP    A
    MOV     R0,A                //R0 存低 8 位
    SETB    P3.6
    LJMP    DATA_HEX_DEC
RETURN:
    RET
```

```
DATA_HEX_DEC:                   //将获取的 12 位 2 进制数转换为十进制存在 R3 R2 R1 R0；千位 百位 十位 个位里
//    R0-->TL0
//   R1-->TH0
    MOV     A,R1
    ANL     A,#0FH
    MOV     R1,A
    CLR     A

    MOV     R2,A                //先清零
    MOV     R3,A
    MOV     R4,A
    MOV     R5,#16          //共转换十六位数
LOOP:
    CLR     C
    MOV     A,R0                //从待转换数的高端移出一位到 Cy
    RLC     A
    MOV     R0,A

    MOV     A,R1
    RLC     A
    MOV     R1,A

    MOV     A,R4                //送到 BCD 码的低端
    ADDC    A,R4                //带进位的自身相加，相当于左移一位
    DA      A                   //十进制调整，变成 BCD 码
    MOV     R4,A

    MOV     A,R3
    ADDC    A,R3
    DA      A
    MOV     R3,A

    MOV     A,R2
    ADDC    A,R2
    MOV     R2,A

    DJNZ    R5,LOOP
//已经把 TH1 TL1 中的数字，转换成 BCD 码，送到了 R2 R3 R4
//*************************************************//
//分别存入 R3 R2 R1 R0；千位 百位 十位 个位
    MOV         A,R4
    MOV     B,#16
```

```
    DIV     AB
    MOV     R1,A
    MOV     R0,B

    MOV     A,R2
    MOV     R4,A

    MOV     A,R3
    MOV     B,#16
    DIV     AB
    MOV     R3,A
    MOV     R2,B
    RET

Delay:                  //定时 0.4ms
    MOV     30H,#10
DEL0:
    MOV     31H,#1
DEL1:
    MOV     32H,#20
DEL2:
    DJNZ    32H,DEL2
    DJNZ    31H,DEL1
    DJNZ    30H,DEL0

    RET

D_table:
    DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH//共阴极 0~9
END
```

7.4 遇到的问题及解决办法

动态显示包括延时模块使用之前实验已经写好的模块，采用 R6 和 R7 的自减判断来采集模拟信号而没有采用中断定时采集信号；写控制字时刚开始不知道怎么弄，之后看了实验指导书上的 C 语言例程和芯片手册，将控制字循环左移八位写入；模拟信号十二位分两次读出，第一次高四位读到 R1，第二次低八位读到 R0；二字节十六进制转化十进制模块参考网上代码修改，经过验证可以使用，主要是先将二字节十六进制经过十六次循环移位变换出 3 位 BCD 码然后再转换成十进制存在 R3~R0 四个寄存器中，最后返回主程序动态显示数值。