硬件实验2：数码管显示实验

原创

nuttee 于 2017-05-30 16:18:24 发布 3994 收藏 7

分类专栏： 硬件 文章标签： 硬件 嵌入式 单片机 数码管 汇编

本文链接：https://blog.csdn.net/lrwwll/article/details/72810362

实验要求：

硬件实验2 数码管显示实验

一、实验要求

1. 实现数码管的循环显示控制。

二、实验目的

1. 学习单片机的基本接口技术。
2. 学习74HC595、74HC138使用及数码显示管的控制方法。

三、 实验电路及连线

8-LED 显示电路

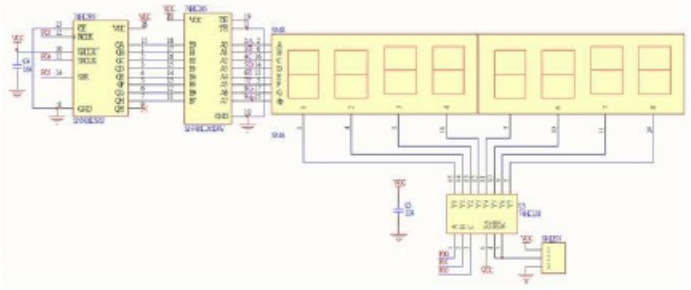


图 1： 8-LED 显示电路

四、实验说明

1. 关于74HC595、74HC138请参照器件手册的说明。
2. 延时程序参照上节课的代码。

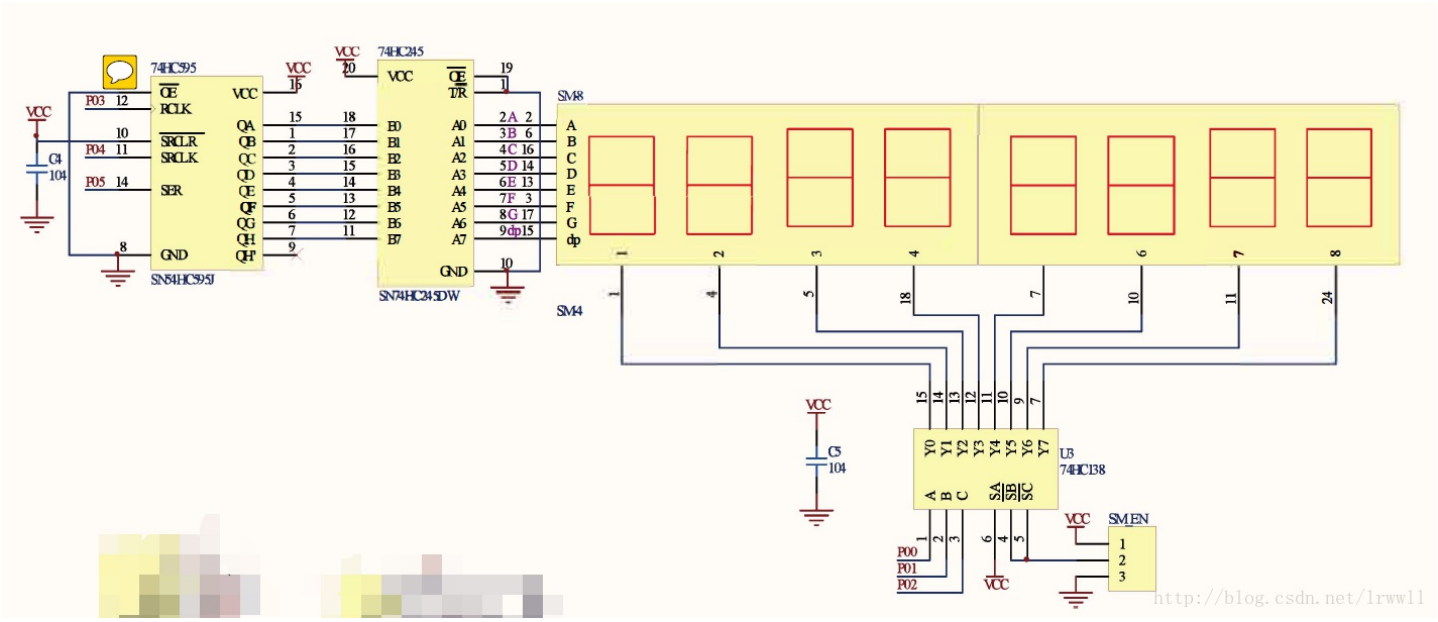
五、实验任务

实验程序，由学生独立完成。

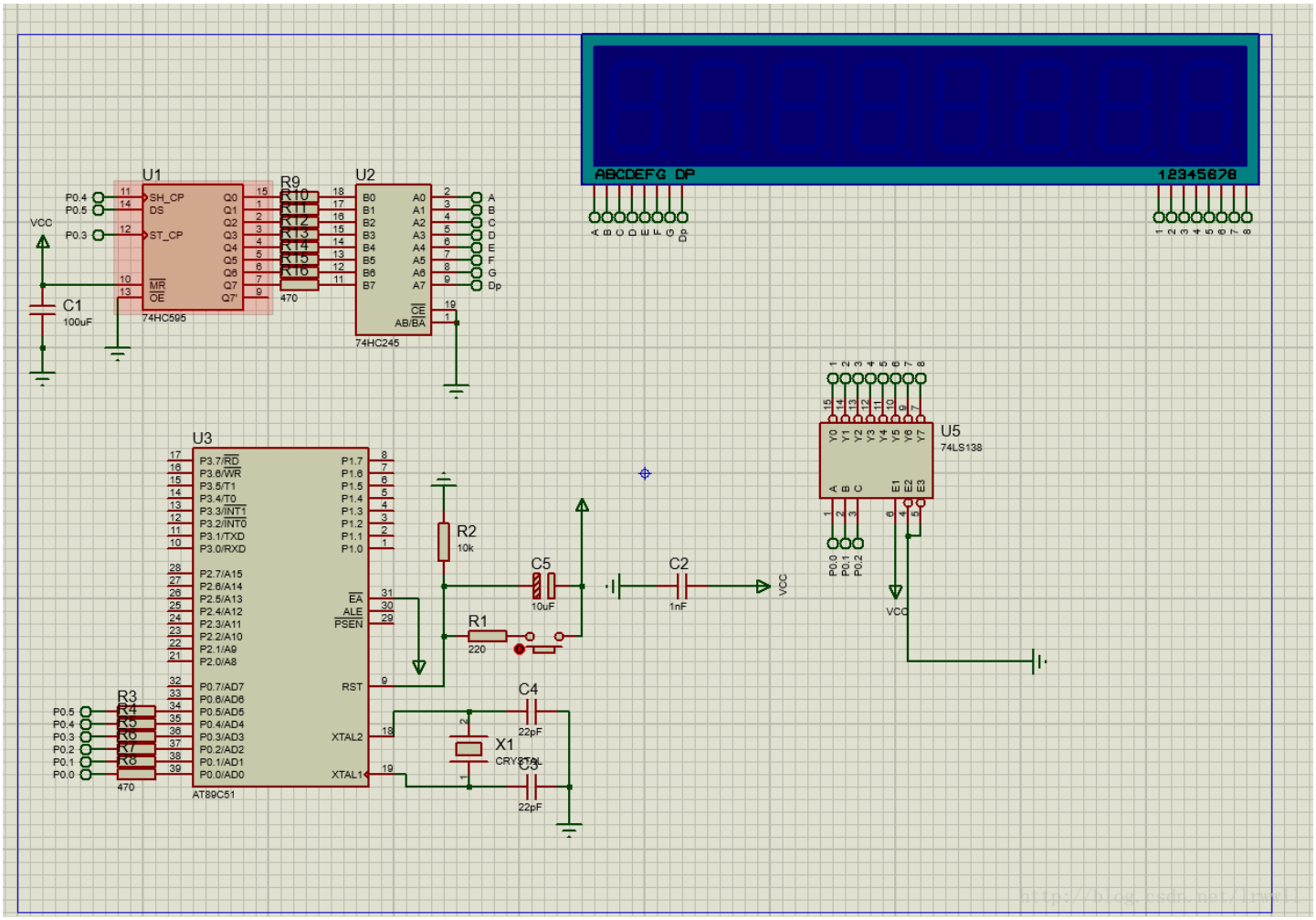
任务：用**汇编**语言设计程序，完成8个数码管的显示控制。

检查内容：程序启动后，8 个数码管依次显示 1→1 2→1 2 3→1 2 3 4→1 2 3 4 5→1 2 3 4 5 6→1 2 3 4 5 6 7→1 2 3 4 5 6 7 8；当 8 个数码管全亮时，进行循环移位显示 1 2 3 4 5 6 7 8→2 3 4 5 6 7 8 1→3 4 5 6 7 8 1 2→

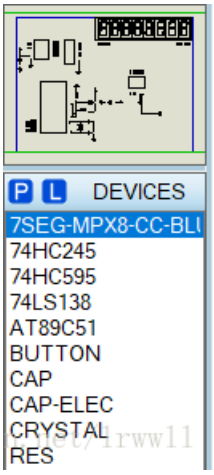
实验板：



我还是先介绍一下电路图吧，上面的电路图是我们学院做的板子的结构图，下面是我用proteus搭建的仿真工程：***proteus仿真工程我已经上传了，传送门：仿真工程下载***



里面用到的器件如下图所示：



我用的是74LS138，功能和74HC138一样，但是驱动更强。在搭建电路的时候，也遇到了一些坑，因为是第一次自己用proteus搭建仿真环境，所以遇到麻烦也无所谓了。主要是P0口最开始没有接上拉电阻，导致74HC595无法输出，Q0~Q7一直是高阻态，我查了一下网上的资料，说是通常在开漏输出的口线上需要上拉电阻，而P0口恰好就是开漏输出的口线。但是这个电阻也不能过大，否则压降全部都降在了电阻上面，就没有足够的电压来驱动芯片了，所以我选了一个经典的电阻值470欧姆。

晶振是12MHz的，由于74LS138是38译码器，一次只能选中一个数码管，也就是说不能够一次性的显示多位，但是实验显然有一次性显示多位的需求，因此为了能够利用人的视觉残留和LED的残影来达到“假装”一次显示多位的目的，我每隔**很短的时间**来选中下一个数码管并且显示。这个**很短的时间**经过我不断地调试发现14ms还不错，虽然理论上20ms以内人类的眼睛就不会察觉出来，但是随着板子的使用也会有所差异，所以需要根据自己的情况适当进行调整。

实验代码：

```
        ORG 00H
        LJMP    MAIN
        ORG 0BH     ;定时器0的中断服务程序入口
        LJMP    T0_INT
        ORG 30H


    NUMBER:
        ;;       1   2   3   4   5   6   7   8
        DB   6H,     5bH,    4fH,    66H,    6dH,    7dH,    7H,     7fH
        DB   6H,     5bH,    4fH,    66H,    6dH,    7dH,    7H,     7fH

    NUMADD      EQU 78h ;78H~7FH中放置的是每个LED值对应的打印
    T0Count     EQU     70h     ;;中断次数计数，每次50ms 10次为500ms
    DISCOUNT    EQU     71h     ;;显示的次数计数,前8次和为递增显示，后8次为循环显示。
    MAIN:
        MOV     A,#0h           ;A中数值为数码0
        MOV     R2,#8           ;一共8位
        MOV     R1,#NUMADD
    INILOOP:
        MOV     @R1,A
        INC     R1
        DJNZ    R2,INILOOP  ;初始化8个LED
    ;;*****************************************************
    ;;开启中断
        SETB    EA   ;开启所有的中断
        SETB    ET0   ;开启定时器中断
        MOV     TMOD,#0001B ;T0设定为16位计数器（定时方式）
        MOV     TH0,#03cH
```

```
    MOV     TL0,#0afH   ;(65535-50000=0x3caf)定义为50ms
    SETB    TR0  ;开启定时器0
    MOV     T0Count,#0    ;;中断次数计数，每次50ms 10次为500ms
    MOV     DISCOUNT,#0  ;;显示的次数计数,前8次和为递增显示，后8次为循环显示。
;;*********************************************************
;;显示8LED的内容，无限循环显示
LOOP:
    ;; 从DispNursAddr读值 在LED上显示
    MOV     P0,#00h
    MOV     R6,#8h
    MOV     R0,#NUMADD
NUMSHIFT: ;移动到下一个要显示的数
    MOV     R7,#8h  ;循环8次
    CLR     A       ;将Acc清零
    MOV     A,@R0   ;把要显示的数放到累加器A中
BITSHIFT:           ;74HC595：8位串行输入，8位并行输出
    RLC     A       ;带进位的左移
    MOV     P0.5,C  ;把移出位放到SER引脚
    ;; 在SRCLK上制造一个正跳变，接收SER的值
    CLR     P0.4
    SETB    P0.4
    DJNZ    R7,BITSHIFT
    ;结束8位串行输入
    INC     R0      ;移动到下一个要显示的值
    ;; 在RCLK上制造一个正跳变，接收整个8位值
    CLR     P0.3
    SETB    P0.3
    LCALL   DELAY  ;延迟
    INC P0              ;(P0.0~P0.2) 对应LED移动
    DJNZ    R6,NUMSHIFT
    /*依次输出12345678*/

    LJMP    LOOP;无限循环显示
;;*****************************************************************
T0_INT:;;T0的中断服务程序
    ;; 用64h ~ 69h保存中断服务程序中用到得寄存器的值
    MOV     69h,R0      ;push R0
    MOV     68h,R1      ;push R1
    MOV     67h,A       ;push A
    MOV     66h,DPH
    MOV     65h,DPL     ;push DPTR
    MOV     64h,R7      ;push R7

    MOV     A,T0Count
    INC     A
    MOV     T0Count,A       ;;读取中断的次数，且中断次数增一

    CJNE    A,#10,RET0   ;;如果已经产生了10次中断则就是500ms,对显示内容处理1次
    ;;
UPDATE:
    MOV     R1,#NUMADD
    MOV     DPTR,#NUMBER
    MOV     T0Count,#0  ;;中断次数清零

    MOV     A,DISCOUNT    ;;读取对显示内容处理的次数，前8次和后8次处理是不一样的
    INC     A
    MOV     DISCOUNT,A
    DEC     A
    MOV     B,#8
```

```
        DIV     AB              ;;余数B正好为循环闪烁时的起始位置

        JNZ    Disp2            ;;若商A为0则说明小于8，递增显示，不为0 则大于8，循环显示
    Disp1:   ;;递增显示的处理模块
        MOV     A,B              ;;从B处递增
        MOV     R0,A
        ADD     A,R1
        MOV     R1,A
        MOV     A,R0

        MOVC    A,@A+DPTR    ;用实际值取列表中的显示值
        MOV     @R1,A        ;把A中显示值放到LED要显示的对应字节中去
        LJMP    RET0
    Disp2:  ;;循环显示的处理模块
        MOV     R7,#8
        MOV     A,B          ;B处为循环闪烁的起点
        MOV     R0,A
    UpdateDisplay:
        MOVC    A,@A+DPTR    ;用实际值取列表中的显示值
        MOV     @R1,A        ;把A中显示值放到LED要显示的对应字节中去
        INC     R0          ;
        INC     R1          ;移动显示值指针
        MOV     A,R0
        DJNZ    R7,UpdateDisplay;一共有8个LED值要更新
    RedoT0: ;;重置定时器0
        MOV     TH0,#03cH
        MOV     TL0,#0afH       ;重置定时器的值
        SETB    TR0             ;开启定时器
        ;; 还原用到的寄存器的值
        MOV     R0,69h      ;pop R0
        MOV     R1,68h      ;pop R1
        MOV     A,67h       ;pop A
        MOV     DPH,66h
        MOV     DPL,65h     ;pop DPTR
        MOV     R7,64h      ;pop R7
        RETI
    ;;*************************************************************
    ;;LED显示切换
    DELAY:
        MOV     R4,#10H
    LOOP1:
        MOV     R5,#70
    LOOP2:
        DJNZ    R5,LOOP2
        DJNZ    R4,LOOP1
        RET
    END
```

我来讲一下思想吧

首先我们需要定义一个数值表，里面存的是12345678 12345678

↓

位置	0	1	2	3	4	5	6	7
表值	1	2	3	4	5	6	7	8
	1	2	3	4	5	6	7	8

然后我们再在指定的内存区中来存放我们即将赋予8个LED数码管的值

0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	2	0	0	0	0	0	0
1	2	3	0	0	0	0	0
1	2	3	4	0	0	0	0
1	2	3	4	5	0	0	0
1	2	3	4	5	6	0	0
1	2	3	4	5	6	7	0
1	2	3	4	5	6	7	8
2	3	4	5	6	7	8	1
3	4	5	6	7	8	1	2
4	5	6	7	8	1	2	3
5	6	7	8	1	2	3	4
6	7	8	1	2	3	4	5
7	8	1	2	3	4	5	6
8	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8

……

1.
2. 首先是初始化，将这些LED的值都初始化为0，数值表指针指向数值表的位置0
3. 然后，将指针所指向的数字放到LED数值表的对应位置上
4. 指针加一，LED数值表的指针也加一
5. 刷新显示LED数值表中的数字，若显示次数小于等于8，重复2、3步骤，否则，跳转到5
6. 数值表指针赋值为1
7. 重复2、3步骤
8. 数值表指针赋值为**(当前指针 + 1) % 8**，跳转到2
9. 结束
10.