电子系统设计基础

班级：信息 005
学号：2206113602
姓名：王靳朝

（1）：在班级、学号、姓名处补全个人信息
（2）：文档以“实验名称_姓名_班级”命名
（2）：报告完成后转换成 pdf，电子版提交到 1421759496@qq.com.
（3）：如发现雷同，0 分处理

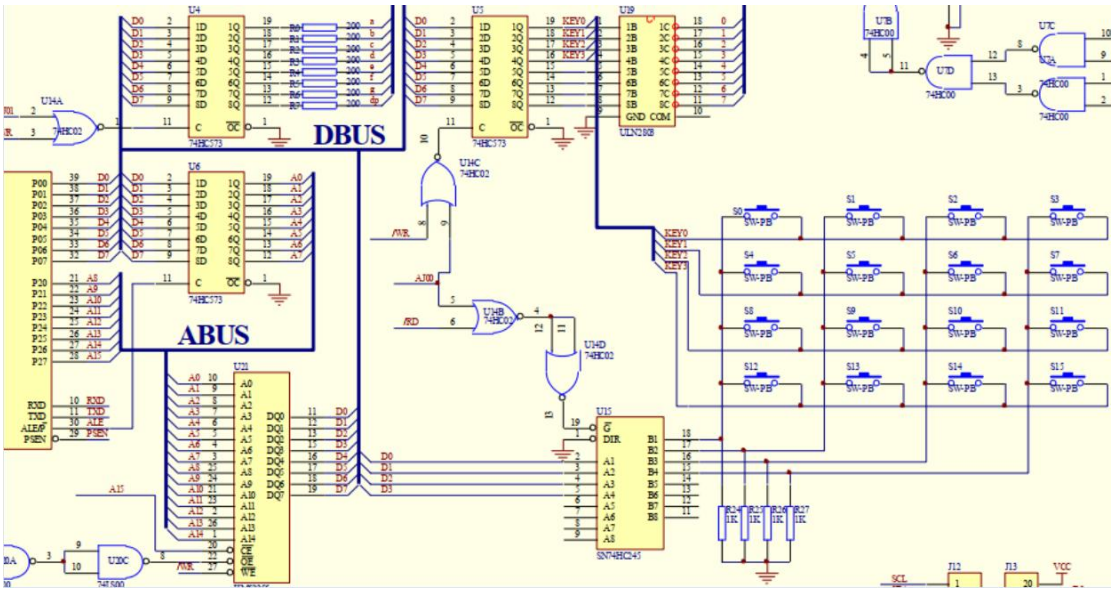
实验四：使用 51 单片机系统实现流水灯

一、实验内容（10 分）

1. 利用单片机实现按键键值获取，并将键值以流水灯的形式显示到七段数码管上。
2. 利用单片机实现按键键值获取，并使用 8 个七段数码管同时显示当前键值。
3. 8 个七段数码管滚动显示自己学号。

二、实验原理（40 分）

2.1 51 文具盒单片机系统按键检测原理（需结合硬件连接说明）



按键的检测原理为对于 4*4 的按键矩阵，按键的位置可以由行、列组成的数对确定。当某一行、某一列的按键被按下后，按键构成的开关导通，单片机的输出信号进入七段数码管，但此时其他按键构成的开关不导通，因此只有一个信号输入。例如当 S0 被按下时，第一行第一列导通，信号向上进入七段数码管。

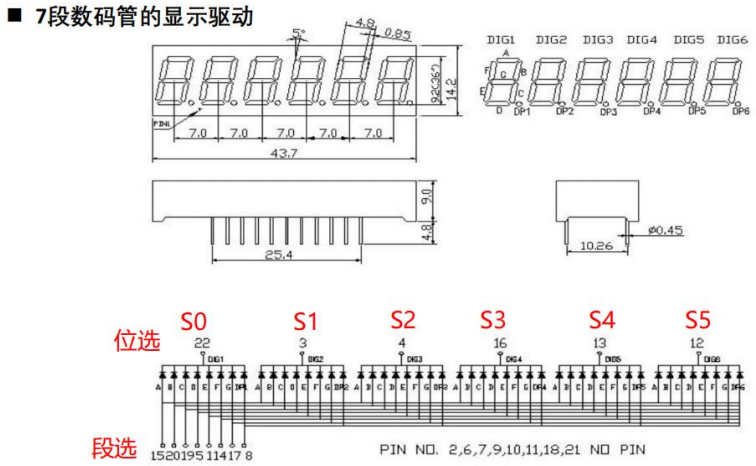
2.2 51 文具盒单片机系统按键检测程序设计思路

检测按键键值使用一个模块实现，主要思路如下：

首先定义行号、列号，初始化行扫描码为十六进制的 1，并定义键号。首先对行地址指针初始化幅值，并从列端口读取列值。当行扫描码低四位不全为 0 时进入循环，循环的作用为给行赋扫描码，并按行移动，当行低四位为 0 时跳出循环。跳出循环后将列码移动到高四

位，键值为列码和行码相或，得到高四位为列码，低四位为行码，从而得到键值获取模块。

2.3　51 文具盒单片机系统的数码管驱动原理（需结合硬件连接说明）



七段数码管共有 abcdefg 七段。对于共阴极七段数码管，当输入信号为高电平(1)时有效，该数段码管亮。对于获取到的不同简直，给数码管不同的位置赋为 1 并用二进制数表示即可，例如对于数码管想要显示“0”，即为 ABCDEF 六段亮，G 灭，由于 ABCDEFG 为从低到高，因此 16 进制表示为 0x6f,以此类推可以得到其他数字。

2.4　51 文具盒单片机系统的数码管驱动程序设计思路

由于本次实验的数码管需要按流水灯的形式进行移动显示，因此需要设计驱动数码管模块、数码管左右移动模块。

三、流水灯程序设计（30 分）

首先对头文件的引用如下：

```
#include <REG51.H>
#include <absacc.h>
#include <stdio.h>
```

3.1 流水灯形式显示当前按键值

由于需要按流水灯的形式显示键值，因此首先需要规定每一个状态持续的时间，即设计延时模块，利用运算本身的时延（比如加法运算、比较运算等）的叠加创造所需的时延。由于函数结构不同，左右移和其他功能需要的时延数量级不同，故分别写了两个时延函数：delay 和 delay2：

```
void delay(unsigned char p)                    //延时函数慢
{
    unsigned char i,j;
    for(;p>0;p--)
    {
        for(i=150;i>0;i--)
        {
            for(j=150;j>0;j--);
        }
    }
```

```
}
void delay2(unsigned char p)                    //延时函数快
{
      unsigned char i;
      for(;p>0;p--)
      {
            for(i=10000;i>0;i--);
      }
}
```

首先进行键值扫描获取，代码如下：

```
unsigned char getkeycode(void)       /*键盘扫描函数，返回获得键码*/
{
   unsigned char line=0x00;             /*行码*/
   unsigned char col=0x00;              /*列码*/
   unsigned char scancode=0x01;         /*行扫描码*/
   unsigned char keycode;               /*键号*/

//XBYTE[0x8000]:位选  XBYTE[0x9000]:段选
   XBYTE[0x8000]=0xff;                  //对单片机外部的内存单元 0x8000 写入 0xff 数据
   col=XBYTE[0x8000]&0x0f;              /*从列端口读入四位列码*/   //0x0f 为低四位代码的意思
   if (col==0x00)
       keycode=0x00;
   else
   {
     while((scancode&0x0f)!=0) /*取 scancode 的低四位，没变为全 0，循环*/
       {
  line=scancode;                        /*行号*/
          XBYTE[0x8000]=scancode;        /*给行赋扫描码，第一行为 0x01*/
          if((XBYTE[0x8000]&0x0f)==col) /*检测按键所在的行跳出循环*/
break;
          scancode=scancode<<1;          /*行扫描码左移一位，转下一行*/
       }
       col=col<<4;                       /*把列码移到高四位*/
       keycode=col|line;
     }
  return keycode;                       //keycode 高四位是列码，低四位是行码
}
```

获取键值后设计左移和右移函数模块，实现流水灯的功能。

```
void leftmove(void){       //左移函数
   int i=0,j=0;
   int k;
   while(j==0)
   {
      for(i=0;i<8;i++)
```

```
        {
         if(j==0)
         {
             delay(5);
             switch(getkeycode())
             {
                  case 0x11:k=1;break;                    /*第一行第 1 列*/
                  case 0x21:k=2;break;                    /*第一行第 2 列*/
                  case 0x41:k=3;break;                    /*第一行第 3 列*/
                  case 0x81:k=4;;break;                   /*第一行第 4 列*/

                  case 0x12:k=5;break;                    /*第二行第 1 列*/
                  case 0x22:k=6;break;                    /*第二行第 2 列*/
                  case 0x42:k=7;break;                    /*第二行第 3 列*/
                  case 0x82:k=8;break;                    /*第二行第 4 列*/
                  case 0x14:k=9;break;                    /*第三行第 1 列*/
                  case 0x88:j=1;break;
                  default:break;
             }
             XBYTE[0x8000]=0x01<<i;   //位选信号向左移 i 位
             XBYTE[0x9000]=led_table[k];
        }
        else
        {
             break;
        }
       }
     }
}

void rightmove(void){         //右移函数
    int i=0,j=0;
    int k;
    while(j==0)
    {
       i=0;
          for(i=0;i<8;i++)
          {
          if(j==0)
          {
              delay(5);
              switch(getkeycode())
              {
                  case 0x11:k=1;break;                        /*第一行第 1 列*/
```

```
                case 0x21:k=2;break;                    /*第一行第 2 列*/
                case 0x41:k=3;break;                    /*第一行第 3 列*/
                case 0x81:k=4;;break;                   /*第一行第 4 列*/

                case 0x12:k=5;break;                    /*第二行第 1 列*/
                case 0x22:k=6;break;                    /*第二行第 2 列*/
                case 0x42:k=7;break;                    /*第二行第 3 列*/
                case 0x82:k=8;break;                    /*第二行第 4 列*/
                case 0x14:k=9;break;                    /*第三行第 1 列*/
                case 0x88:j=1;break;
                default: break;
            }
            XBYTE[0x8000]=0x80>>i;
            XBYTE[0x9000]=led_table[k];
        }
        else
        {
          break;
        }
        }
    }
}
```

这两个函数都只显示一位数码管，所以不需要扫描，只需要每隔一段时间（肉眼可见）变换一下位选信号即可，段选信号如果不按按键变化数字的话可以不用变。这两个函数使用的是相对较长时间的 delay 函数。

3.2 数码管同时显示当前按键值

这个函数需要八位数码管同时亮同时灭，所以需要扫描，即一个周期中，分别让每一个段选信号有效一小段时间（肉眼几乎不可见），同时与该位对应的段选信号有效，在足够的频率下造成 8 位数码管同时亮的“错觉”。此时用到的 delay 相对较短，故用 delay2（）。

实现过程中设置了一个 flag 变量，当该变量为 1 时，利用循环实现延时（n，125 次），并在循环结束后将 flag 变为 0；当 flag 为 0 时，数码管的所有段选都为 0，不显示，利用循环实现相同的延时，并在循环结束后将 flag 变为 1：

```
void flash(void)
{
    unsigned int i,n,flag=0,k=0;
    while(k==0)
    {
        if(flag==0)
        {
            for(n=0;n<125;n++)                  //n：扫描信号
            {
                for(i=0;i<8;i++)            //i:位选
                {
```

```
                    XBYTE[0x9000]=0;
                    XBYTE[0x8000]=0x01<<i;
                    XBYTE[0x9000]=0;
                    delay2(10);
                }
            }
            flag=1;
        }
        else
        {
             for(n=0;n<125;n++)                //n：扫描信号
            {
                for(i=0;i<8;i++)              //i:位选
                {
                    XBYTE[0x9000]=0;
                    XBYTE[0x8000]=0x01<<i;
                    XBYTE[0x9000]=led_table2[i];
                    delay2(10);
                }
            }
            flag=0;
        }
        switch(getkeycode())
        {
            case 0x88:k=1;break;
            default: break;
        }
    }
}
```

3.3 滚动显示学号

设置的学号数组如下：

```
unsigned char code xuehao[]={                                    //学号
    0x40,0x5b,0x5b,0x3f,0x7d,0x06,0x06,0x4f,0x7d,0x3f,0x5b,0x40,0x40,0x00,0x00,0x40,0x40,
0x5b,0x5b,0x3f,0x7d,0x06,0x06,0x4f,0x7d,0x3f,0x5b,0x40,0x40,0x00,0x00,0x40};
```

分别根据 2206……等数字推出七段数码管对应的亮起位置，并在每次滚动之间设置{-- --}进行分割，函数体如下：

```
void number(void)
{
 unsigned int i,n,count,k=0;
 while(k==0)
 {
    for(count=0;count<16&&k==0;count++)    //count：滚动变化计数
    {
```

```
        for(n=0;n<125;n++)                  //n：扫描信号
        {
            for(i=0;i<8;i++)            //i:位选
            {

                XBYTE[0x9000]=0;
                XBYTE[0x8000]=0x01<<i;
                XBYTE[0x9000]=xuehao[8-i+count];   //学号：段选
                delay2(20);
            }
        }
        switch(getkeycode())
        {
            case 0x88:k=1;break;
            default: break;
        }
     }
   }
 }
```

左右移动函数整体上与前文中的相似，增加了变量 lr 作为控制左/右移的标志，当触及边界（i==7）时，该变量取反。代码如下：

```
void lrmove(void)
 {       //左右移函数
    int i=0,j=0;
    int k;
    int lr=0;
    while(j==0)
    {
        for(i=0;i<8;i++)
        {
        if(j==0)
        {
            delay(10);
            switch(getkeycode())
            {
                case 0x11:k=1;break;                    /*第一行第 1 列*/
                case 0x21:k=2;break;                    /*第一行第 2 列*/
                case 0x41:k=3;break;                    /*第一行第 3 列*/
                case 0x81:k=4;;break;                    /*第一行第 4 列*/

                case 0x12:k=5;break;                  /*第二行第 1 列*/
                case 0x22:k=6;break;                  /*第二行第 2 列*/
                case 0x42:k=7;break;                  /*第二行第 3 列*/
                case 0x82:k=8;break;                  /*第二行第 4 列*/
```

```
                case 0x14:k=9;break;                    /*第三行第 1 列*/
                case 0x88:j=1;break;
                default: break;
            }
            if(lr==0)
            {
                XBYTE[0x8000]=0x80>>i;
            }
            else if(lr==1)
            {
                XBYTE[0x8000]=0x01<<i;
            }
            if(i==7)
            {
                lr=1-lr;
                i=0;
            }
            XBYTE[0x9000]=led_table[k];
        }
        else
        {
            break;
        }
    }
  }
}
```

最后 main 函数进行整体调用即可，如下：

```
void main (void) {
  while (1)
  {
  switch(getkeycode())
  {
  case 0x11:leftmove();break;
  case 0x21:rightmove();break;
  case 0x41:flash();break;
  case 0x12:number();break;
  case 0x22:lrmove();break;
  default:break;
  }
 }
}
```

四、实验结果（10 分）

1.本次实验中设计了左右移动两种流水灯：

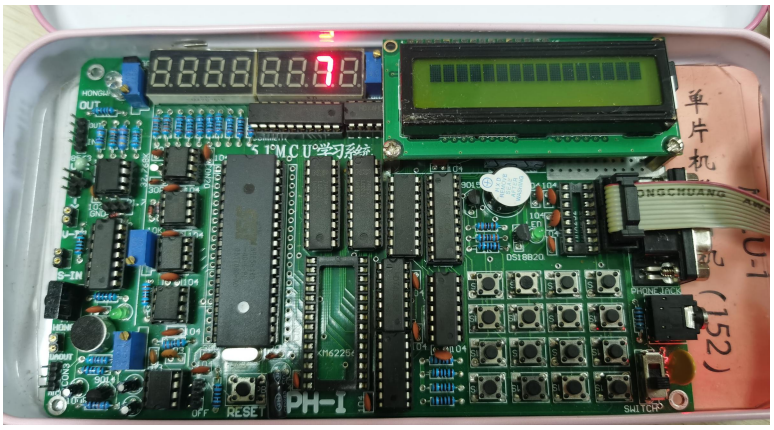
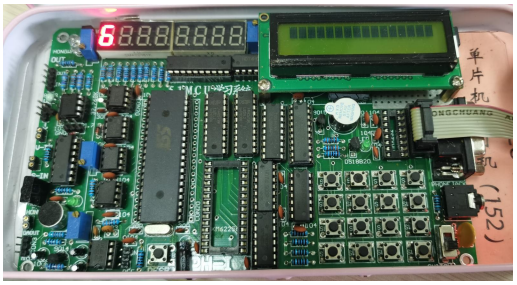


图 1 左移



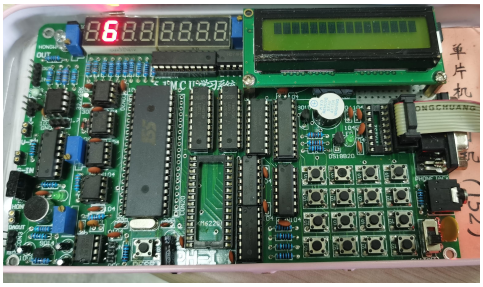
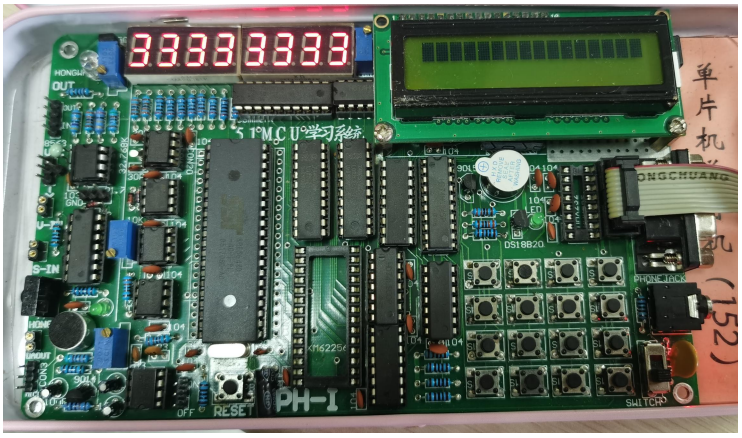
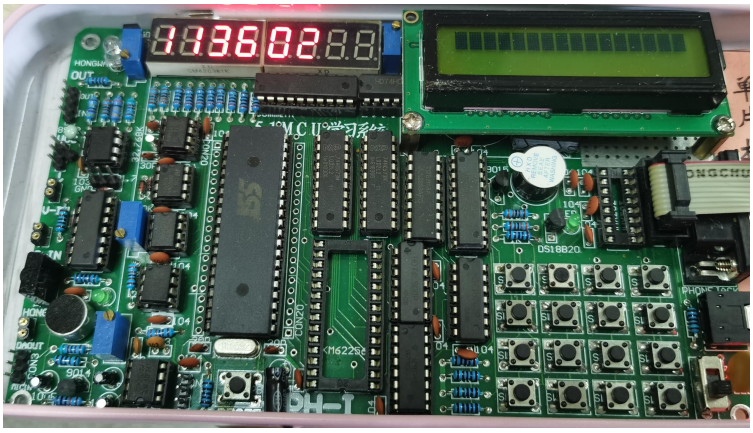


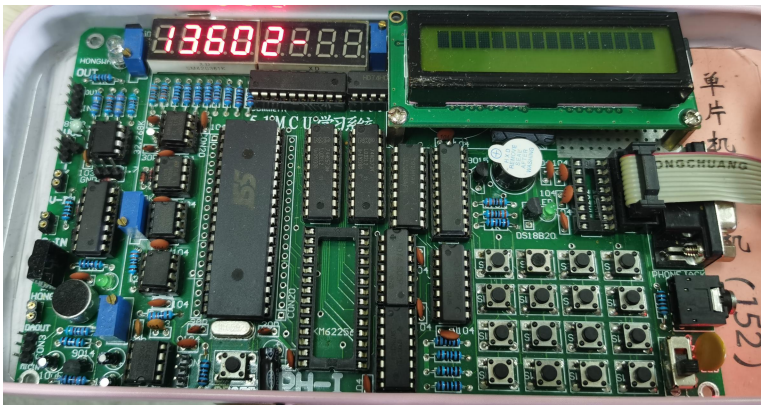
图 2 右移

2.八个数码管同时显示



3.滚动显示学号





可以观察到，学号按照肉眼可见的延时向左移动，功能完好。

五、总结（10 分）

1.问题：

在滚动显示学号模块，我最初的想法是写一个 16 位图形数组，分别让数码管显示{2206113602--　--}，然后让这个数组在每一位段选上差一位循环，从而实现滚动。但实际操作后发现了两个问题：

I）第一位“2”总是显示不出来，而是从“1”开始显示，可能是函数初态问题。

II）每个周期，由于前一半不需要计算，可以正常显示，但到后一半时，需要手动调整数组内的参数使其循环，否则会越界，这导致了前后显示周期不一致，所以后一半的显示是乱码。

为了解决这两个问题，又写了第二版函数，并调整了图形数组，由 16 位调整为 32 位，使其就算直接计算也不会越界，就不存在周期不一的问题；同时，将数组的首位设为-，之后才是 22061……（后略），避免第一位显示不出来。

2.部分功能优化：**模式退出功能（按键 0x88）**

发现每次进入了一个函数后都无法退回，如果想进入另一个函数，需要将整个程序停止运行，退出，初始化并重新启动才可以，十分不方便。所以在每个函数的运行过程中，每个周期都增加了一个判断，让单片机返回一个键值，如果键值为 0x88，则退出该函数，返回还没有按键的状态，此时可以通过按键再一次控制程序，决定进入哪一个函数。