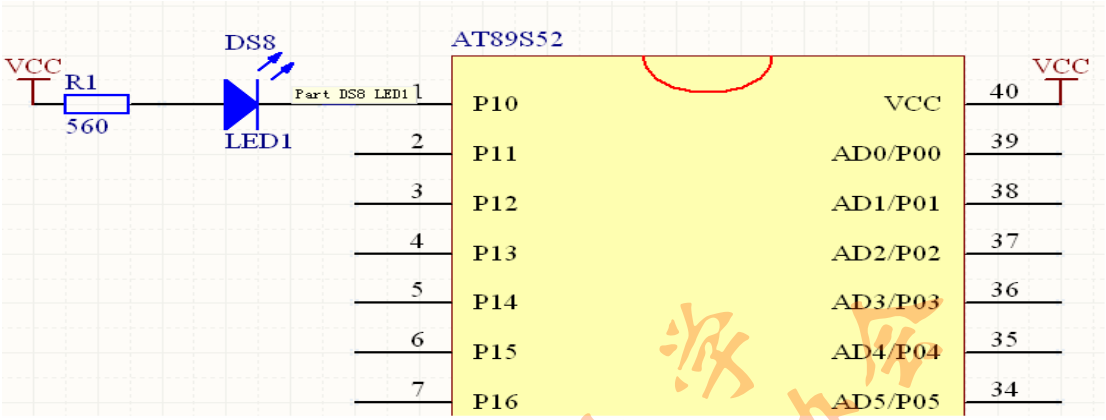
实验三:按键输入

一.实验目的:

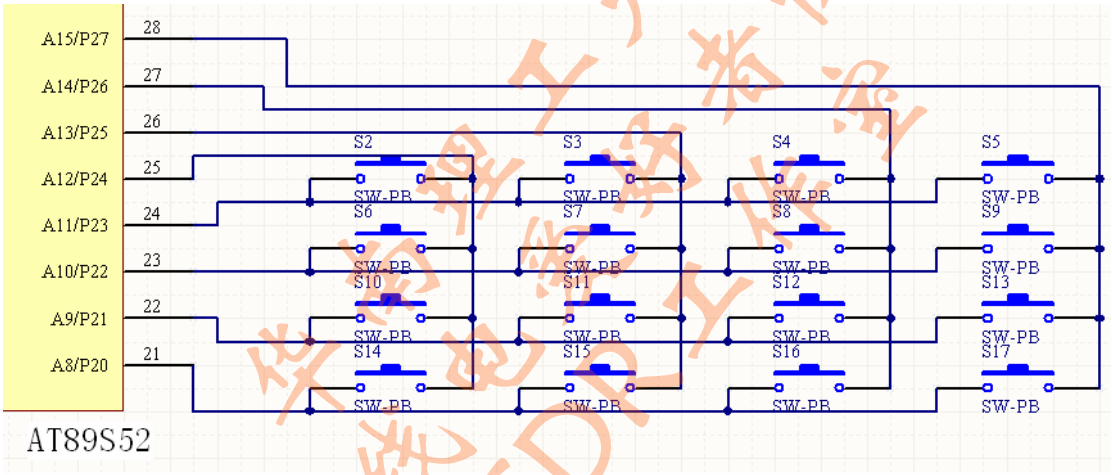
1,了解单片机IO口作为输入的操作

2,通过外部按键控制一个LED的亮灭

二,实验原理:



LED接线图



键盘接线图

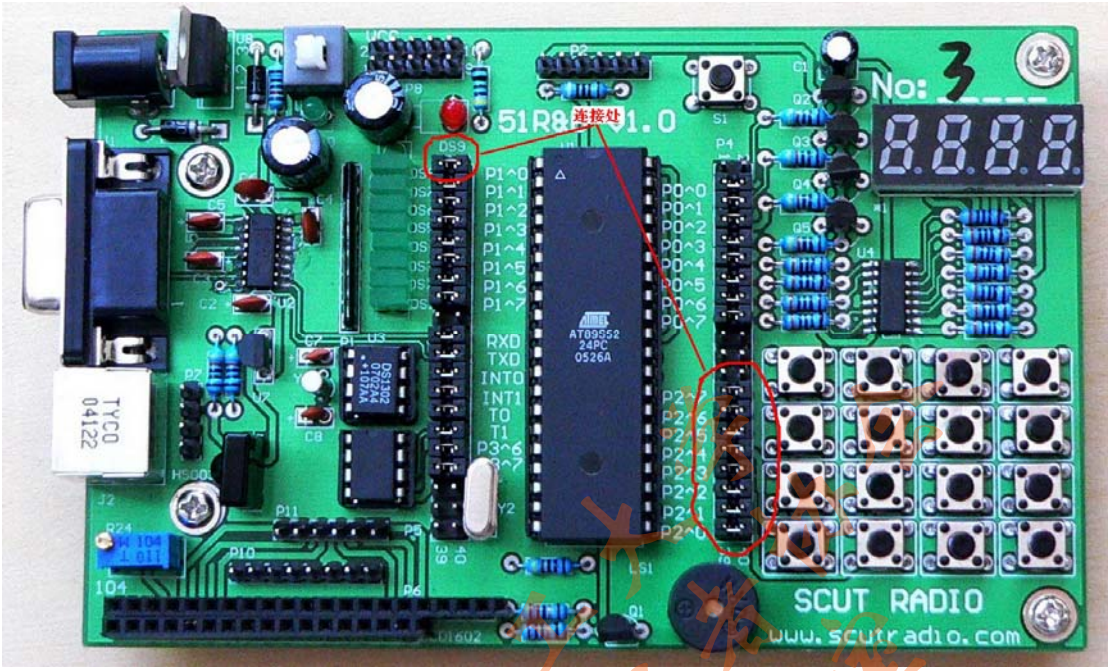
前面我们的学习,都是把单片机的IO口作为输出口使用.而在这一节我们把它做为输入口使用,有个问题必须**十分注意:51单片机的IO口作为输入口使用必须在读取IO口电平之前把该IO口置1,要不就可能读出错误的信息.**

此次实验我们把P2.0作为输入口,读取低电平,从上面的原理图可知,只要令P2=0X01,二进制的00000001B,则可以实现P2.0读取低电平,因为除P2.0之外其他都输出低电平,只要S14-S17中任意一个按键按下则可以把P2.0拉成低电平,从而实现P2.0读取低电平.

而在按键按下的过程中，由于机械抖动，将产生干扰，电平高低变化。在按下的过程中，一旦有干扰过来，可能造成误触发过程，这并不是我们所想要的。因此在按键按下的时候，要把我们按键的机械接触等干扰信号给滤除掉，我们可以采用软件滤波的方法去除这些干扰信号，一般情况下，一个按键按下的时候，总是在按下的瞬间高低电平不确定，若干毫秒之后就基本上进入了稳定的状态。我们在程序设计时，采用扫描判断的方法，CPU 在运行中不断判断P2.0的状态，一旦发现P2.0 为低电平，进入按键判断状态，先软件延时10ms，从而避开了干扰信号区域，再重新来检测P2.0状态，看按键是否真得已经按下，若真得已经按下，这时肯定输出为低电平，若这时检测到的是高电平，证明刚才是由于干扰信号引起的误触发,CPU 就认为是误触发信号而舍弃这次的按键识别过程。

另外为了不让按键重复触发,我们使用一个标记量off来标记,如果按下则令off=1,当按键未松开时,off不会恢复(off=1),屏蔽了按键,也就不会再次获得键值,只有按键松开才能使off=1,恢复按键功能所以就有效的控制了重复触发.

三,硬件连接:



连接图

四,实验代码:

```
#include <reg51.h>
sbit LED=P1^0; //定义 P1.0 为 LED 接口
sbit KEY=P2^0;
#define uchar unsigned char
/*------------------time ms 延时函数-----------------*/
void delay_ms(unsigned int time)
{
    unsigned char tres;
    for(;time>0;time--)
    {
        tres=150;
        while(tres--);
    }
}
void main()
{
    bit off=1;//按键松开标记
    P2=0X01;
    while(1)//死循环,不停的点亮和熄灭 LED
    {
        if(!KEY&&off)//判定进入获取键值的条件:key=0,off=1
        {
```

```
                delay_ms(10);//消除抖动
                if(!KEY)   //确实有键按下
                {
                    LED=!LED;//LED 取反
                    off=0;      //按键按下标记
                }
            }
            if(KEY)off=1;  //清除按下标记
        }
    }.
```

按硬件连接所示, 把硬件连接好(红色框内部分), 把程序编译之后,下载到单片机后,就可以通过最右边一列按键来控制DS8的亮灭了.