实验十六:PS2 鼠标控制

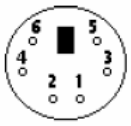
一. 实验目的:

1.了解 PS2 鼠标键盘协议
2.学会分析简单的数字信号和使用单片机捕捉及解码信号

二, 实验原理:

PS/2 鼠标键盘协议

引脚定义如下所示

Male 公的	Female 母的	6-pin Mini-DIN (PS/2):	6 脚 Mini-DIN(PS/2)
		1 - Data	1—数据
		2 - Not Implemented	2—未实现，保留
		3 - Ground	3—电源地
		4 - +5v	4—电源+5V
(Plug) 插头	(Socket) 插座	5 - Clock	5—时钟
		6 - Not Implemented	6—未实现，保留

PS/2 鼠标和键盘履行一种双向同步串行协议，换句话说，每次数据线上发送一位数据并且每在时钟线上发一个脉冲就被读入，键盘/鼠标可以发送数据到主机而主机也可以发送数据到设备，但主机总是在总线上有优先权。它可以在任何时候抑制来自于键盘/鼠标的通讯，只要把时钟拉低即可。从键盘/鼠标发送到主机的数据在时钟信号的下降沿，当时钟从高变到低的时候被读取，从主机发送到键盘/鼠标的数据在上升沿，当时钟从低变到高的时候被读取。不管通讯的方向怎样，键盘/鼠标总是产生时钟信号。如果主机要发送数据它。必须首先告诉设备开始产生时钟信号。最大的时钟频率是33kHz 而且大多数设备工作在10 20kHz 如果你要制作一个PS/2 设备我推荐你把频率控制在15kHz 左右这就意味着时钟应该是高40 微秒低40 微秒

所有数据安排在字节中每个字节为一帧包含了11个位，这些位的含义如下
1 个起始位总是为0
8 个数据位低位在前
1 个校验位奇校验
1 个停止位总是为1

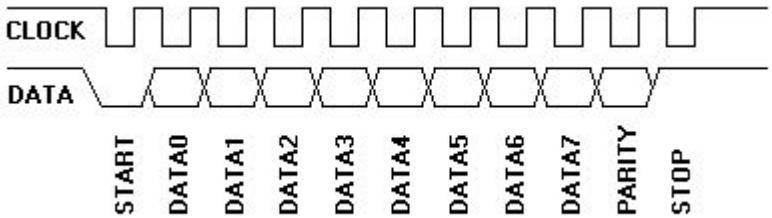
如果数据位中包含偶数个1,校验位就会置1 ;如果数据位中包含奇数个1,校验位就会置0,数据位中1的个数加上校验位总为奇数，这就是奇校验，这是用来错误检测。

当主机发送数据给键盘/鼠标时，设备回送一个握手信号来应答数据包已经收到，这个位不会出现在设备发送数据到主机的过程中，设备到主机的通讯过程数据和时钟线都是集电极开路结构，正常保持高电平当键盘或鼠标等待发送数据时它首先检查时钟以确认它是否是高电平，如果不是，那么是主机抑制了通讯，设备必须缓冲任何要发送的数据直到重新获得总线的控制权。键盘有16 字节的缓冲区而鼠标的缓冲区仅存储最后一个要发送的数据包，如果时钟线是高电平设备就可以开始传送数据。

键盘和鼠标使用一种每帧包含11 位的串行协议这些位含义是
1 个起始位总是为0
8 个数据位低位在前
1 个校验位奇校验
1 个停止位总是为1

每位在时钟的下降沿被主机读入,时钟频率为10--16.7kHz。从时钟脉冲的上升沿到一个数据转变的时间至少要有5 微秒，数据变化到时钟脉冲的下降沿的时间至少要有5 微秒并且不大于25 微秒，这个定时非常重要你应该严格遵循它！主机可以在第11 个时钟脉冲停止位之前把线拉低，导致设备放弃发送当前字节。这是非常罕见的。在停止位发送后，设备在发送下个包前至少应该等待50 毫秒，这将给主机时间，当它处理接收到的字节时抑制发送，主机

在收到每个包时通常自动做这个，在主机释放抑制后设备至少应该在发送任何数据前等50 毫秒



我推荐下面的过程发送一个单一字节从仿真键盘/鼠标到主机

1) 等待Clock = high
2) 延时 50 微秒
3) Clock s 仍旧为 high?

No—到第1 步

4) Data = high

No—放弃 (并且从主机读取字节)

5) 延迟 20 毫秒 (=40 微秒 to the time Clock is pulled low in sending the start bit。)
6) 输出起始位 (0)\ 在发送所有这些位的每一位后
7) 输出 8 个数据位 > 测试时钟确认主机是否把它拉低了
8) 输出校验位 / 这说明主机要放弃这次传送
9) 输出停止位 (1)
10) 延迟30 毫秒

按如下的过程发送单个位

1) 设置/复位数据
2) 延迟20 微秒
3) 把时钟拉低
4) 延迟40 微秒
5) 释放时钟
6) 延迟20 微秒

主机到设备的通讯

被发送的包有点不同于主机到设备通讯过程，首先PS/2 设备总是产生时钟信号。如果主机要发送数据它必须首先把时钟和数据线设置为请求发送状态，如下：

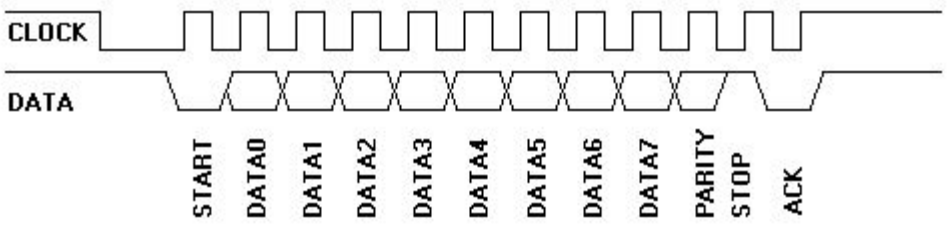
通过下拉时钟线至少100 微秒来抑制通讯

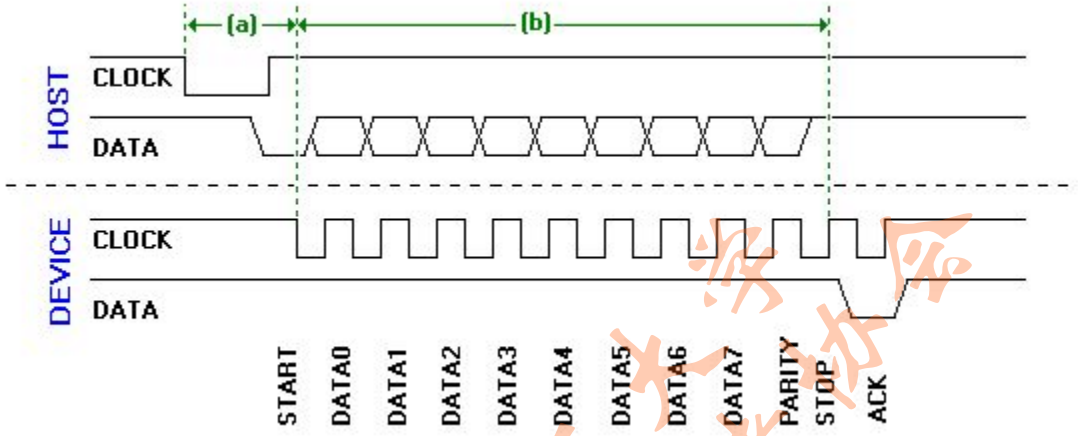
通过下拉数据线来应用请求发送然后释放时钟

设备应该在不超过10 毫秒的间隔内就要检查这个状态，当设备检测到这个状态它将开始产生时钟信号并且时钟脉冲标记下输入八个数据位和一个停止位，主机仅当时钟线为低的时候改变数据线，而数据在时钟脉冲的上升沿被锁存，这在发生在设备到主机通讯的过程中正好相反，在停止位发送后设备要应答接收到的字节就把数据线拉低并产生最后一个时钟脉冲，如果主机在第11 个时钟脉冲后不释放数据线，设备将继续产生时钟脉冲直到数据线被释放，然后设备将产生一个错误，主机可以在第11个时钟脉冲应答位前中止一次传送，只要下拉时钟线至少100 微秒，要使得这个过程易于理解主机必须按下面的步骤发送数据到PS/2 设备

1) 把时钟线拉低至少100 微秒
2) 把数据线拉低
3) 释放数据线
4) 等待设备把时钟线拉低
5) 设置/复位数据线发送第一个数据位
6) 等待设备把时钟拉高
7) 等待设备把时钟拉低
8) 重复 5-7 步 发送剩下的7 个数据位和校验位
9) 释放数据线

10) 等待设备把数据线拉低
11) 等待设备把时钟线拉低
12) 等待设备释放数据线和时钟线





设备产生的信号注意应答位时序的改变，数据改变发生在时钟线为高的时候，不同于其它11位是当它为低的时候，在主机最初把数据线拉低后，设备开始产生时钟脉冲的时间必须步大于15ms，数据包被发送的时间必须不大于2ms，如果这两个条件不满足，主机将产生一个错误，在包收到后主机为了处理数据立刻把时钟线拉低来抑制通讯，如果主机发送的命令要求有一个回应，这个回应必须在主机释放时钟线后20ms之内被收到，如果没有收到，则主机产生一个错误，在设备到主机通讯的情况中时钟改变后的5 微秒内不应该发生数据改变的情况，如果你要仿真一个鼠标或键盘我推荐你按如下的过程从主机读入数据，在你的主程序中至少每10 毫秒检测数据线是否为低，如果数据线已被主机拉低则从主机读取一个字节

1) 等待时钟线为高
2) 数据线仍然为低吗
不有错误发生放弃
3) 读入8 个数据位 \ 在读入这些位后
4) 读入校验位 > 测试时钟线数否被主机拉低
5) 读入停止位 / 这就意味着放弃这次传送
6) 数据线仍旧为0 吗
是保持时钟直到数据1 然后产生一个错误
7) 输出应答位
8) 检查校验位
如果校验位不正确则产生一个错误
9) 延迟45 微秒给主机时间抑制下次的传送
按如下次序读取每位8 个数据位检验位和停止位
1) 延迟 20 微秒
2) 把时钟拉低
3) 延迟 40 微秒
4) 释放时钟
5) 延迟20 微秒
7) 读数据线
按如下次序发送应答位

1) 延迟15 微秒
2) 把数据线拉低
3) 延迟5 微秒
4) 把时钟线拉低
5) 延迟40 微秒
6) 释放时钟线
7) 延迟5 微秒
8) 释放数据线

设备到主机的通信

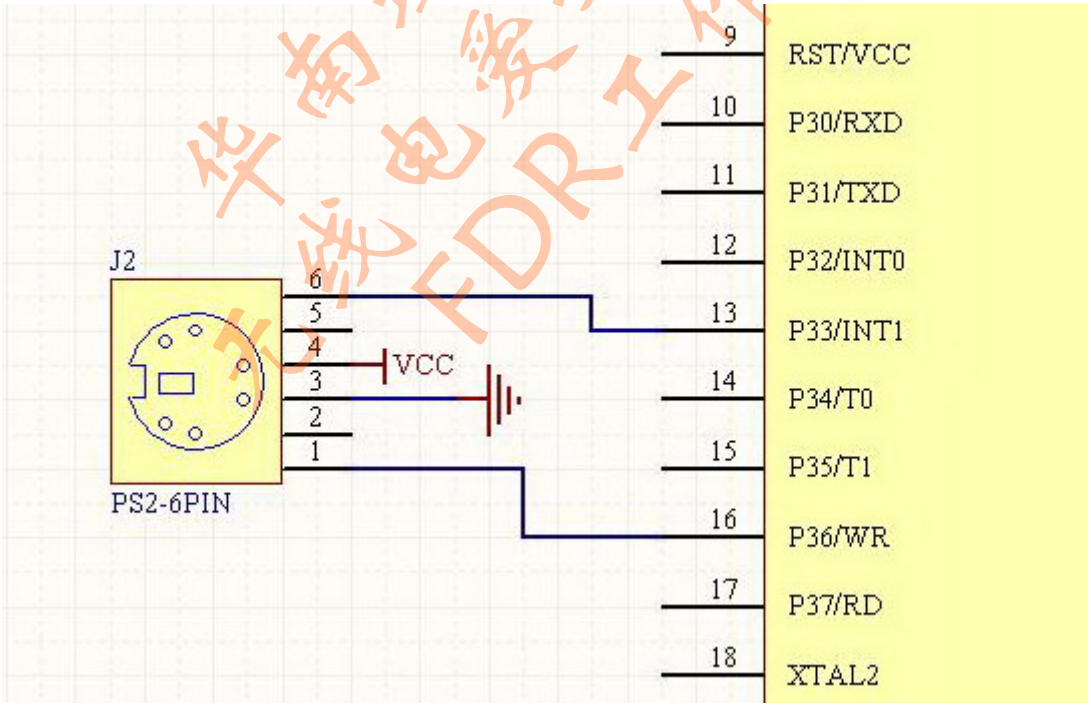
标准的PS/2 鼠标发送位移和按键信息给主机采用如下的3 字节数据包格式

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Y overflow	X overflow	Y sign bit	X sign bit	Always 1	Middle Btn	Right Btn	Left Btn
Byte 2	X Movement							
Byte 3	Y Movement							

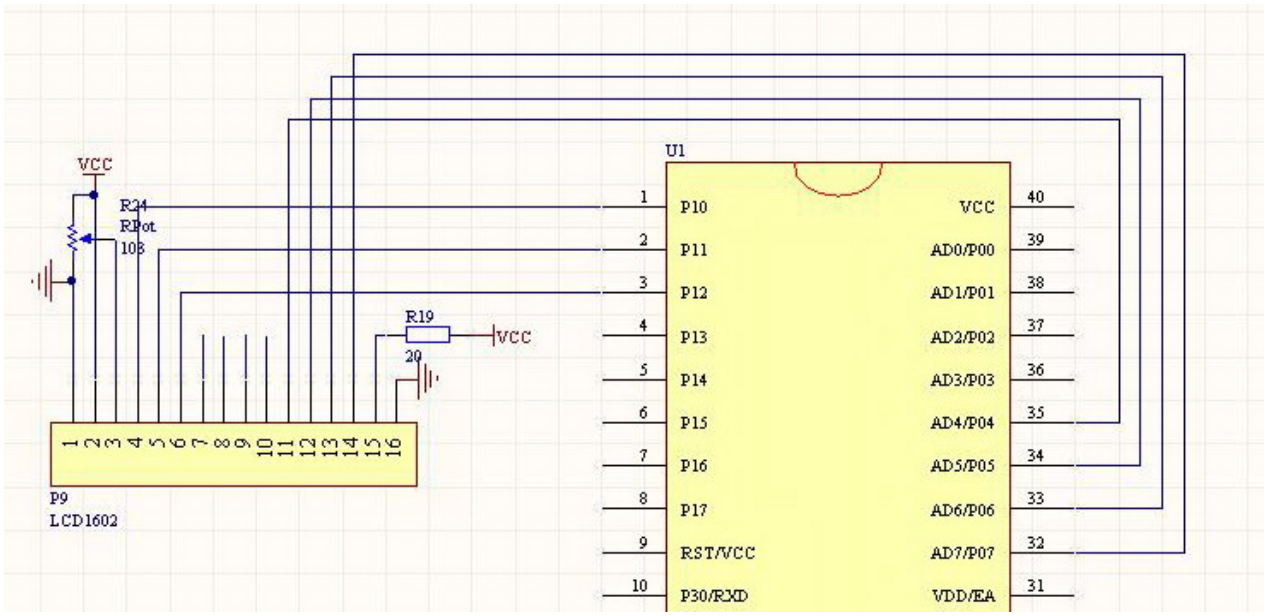
位移计数器是一个9 位2 的补码整数。它的最高位作为符号位出现在位移数据包的第一个字节里。这些计数器在鼠标读取输入发现有位移时被更新。这些值是自从最后一次发送位移数据包给主机后位移的累计量（即最后一次包发给主机后位移计数器被复位）。位移计数器可表示的值的范围是-255 到+255，如果超过了范围，相应的溢出位就被设置，并且在复位前，计数器不会增减。正如我前面提及的一旦位移数据包成功地发送给主机，位移计数器就会复位，同样鼠标在收到主机不是Resend 0xFE 命令外的其他命令，计数器也会复位。

接线图

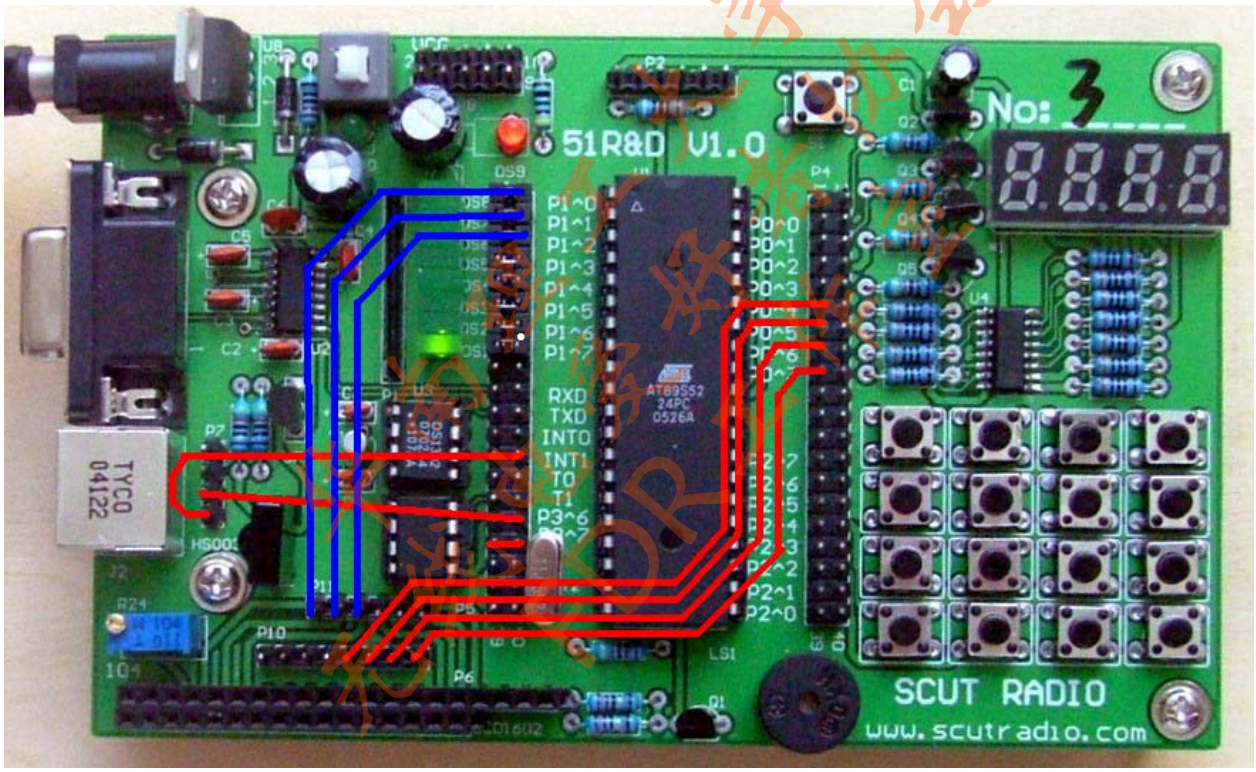
(1)Ps2座与单片机的连接



(2)单片机与液晶的连接



(3)硬件连接图



三.单片机程序:

**********主函数**********

```
#include<reg52.h>
#include"mouse.h"
#include"LCD1602_4.h"

sbit beep=P3^7;
```

```
void main()
{

    LCM1602_Init();
    Init_device();
    mouse_send_data(0xf4);//向鼠标发送0xF4命令发其开始工作
    EX1=0;//关掉外部中断以避开鼠标发回的应答
    delayms(100);
    EX1=1;//重开外部中断
    while(1)
    {
        CLEARSCREEN;//清屏
        LCM1602_write_string(0,0,"x:");
        num(0,2,move_x);//x坐标值
        LCM1602_write_string(0,8,"y:");
        num(0,10,move_y);//y坐标值
        if(mouse_data[0]&0x01)//如果点下左键
        {
            beep=0;
            LCM1602_write_string(1,0,"left");
        }
        else if(mouse_data[0]&0x02)//如果点下右键
        {
            beep=0;
            LCM1602_write_string(1,0,"right");
        }
        else if(mouse_data[0]&0x04)//如果点下中键
        {
            beep=0;
            LCM1602_write_string(1,0,"middle");
        }
        else
        {
            beep=1;
            LCM1602_write_string(1,0,"nothing");
        }
        delayms(50);
    }
}
```

**********鼠标驱动头文件**********

```
#include"delay52.h"
#ifndef MOUSE_H
#define MOUSE_H

sbit mouse_SDA=P3^6;//数据线P3_5
sbit mouse_CLK=P3^3;//时钟线P3_3

sbit led1=P1^3;
```

```
unsigned char bdata mouse_byte; //接收字节
sbit mouse_byte_bit0=mouse_byte^0;//mouse_byte第0位
sbit mouse_byte_bit1=mouse_byte^1;//mouse_byte第1位
sbit mouse_byte_bit2=mouse_byte^2;//mouse_byte第2位
sbit mouse_byte_bit3=mouse_byte^3;//mouse_byte第3位
sbit mouse_byte_bit4=mouse_byte^4;//mouse_byte第4位
sbit mouse_byte_bit5=mouse_byte^5;//mouse_byte第5位
sbit mouse_byte_bit6=mouse_byte^6;//mouse_byte第6位
sbit mouse_byte_bit7=mouse_byte^7;//mouse_byte第7位

unsigned char bdata mouse_fuction;//功能信息字节

unsigned char mouse_buffer[11];//接收位数据缓冲区
unsigned char mouse_buffer_bit=0;//mouse_buffer[mouse_buffer_bit]
unsigned char mouse_data[3];//接收鼠标数据缓冲区,分别存放:功能信息字节,x位移量,y
位移量
unsigned char mouse_data_bit=0;//mouse_data[mouse_data_bit]

unsigned int move_x=10000;//存放横坐标
unsigned int move_y=10000;//存放纵坐标

void Init_mouse(void)
{
     TCON=0x00;
     EA=1;
     EX1=1;//允许外部中断1
     ET0=0x01;//允许全局中断，允许定时器/计数器0溢出中断
     PX1=1;//设置中断优先级
}


/*********************************************************************
       发送数据
*********************************************************************/
 void mouse_send_data(unsigned char dat)
{
    unsigned char i;
    EX1=0;                                  /*关闭外部中断1*/
    ACC=dat;                                /*将要发送的数据放入A寄存器*/
    mouse_CLK=0;                           /*拉低时钟线*/
    delay10us(200);                          /*延时100us以上*/
    mouse_SDA=0;                            /*拉低数据线*/
    delay10us(40);
    mouse_CLK=1;                           /*释放时钟线*/
    for(i=0;i<=7;i++)                    /*低位在前，一次发送8个数据位*/
    {
        while(mouse_CLK==1);              /*等待设备拉低时钟线*/
        mouse_SDA=(dat>>i)&0x01;         /*发送数据位*/
        while(mouse_CLK==0);              /*等待设备释放时钟线*/
    }
```

```
    while(mouse_CLK==1);
    mouse_SDA=~P;                           /*发送校验位，奇校验*/
    while(mouse_CLK==0);
    while(mouse_CLK==1);
    mouse_SDA=1;                            /*发送停止位*/
    while(mouse_CLK==0);
    while(mouse_CLK==1);                /*应答位*/
    while(mouse_CLK==0);
    EX1=1;                                  /*打开外部中断1*/
}

/*********************************************
          奇校检
*********************************************/
unsigned char Checkout(void)
{
    ACC=mouse_byte;
    if(~P==mouse_buffer[9])
        return 1;
    else
        return 0;
}
/*************************************************************
          数据分析及处理
*************************************************************/
void data_analyse(void)
{
    //将收到的11位信号中截取8位数据放进mouse_byte
    mouse_byte_bit0=mouse_buffer[1];
    mouse_byte_bit1=mouse_buffer[2];
    mouse_byte_bit2=mouse_buffer[3];
    mouse_byte_bit3=mouse_buffer[4];
    mouse_byte_bit4=mouse_buffer[5];
    mouse_byte_bit5=mouse_buffer[6];
    mouse_byte_bit6=mouse_buffer[7];
    mouse_byte_bit7=mouse_buffer[8];
    if(Checkout())//如果校验正确
    {
        if(mouse_data_bit<3)
            mouse_data[mouse_data_bit++]=mouse_byte;
        if(mouse_data_bit==3)
        {
            mouse_data_bit=0;
            if(mouse_data[0]&0x10)//如果"X sign bit"为1,表示鼠标向左移
            {
                move_x-=(256-mouse_data[1]);//x坐标减
            }
            else
            {
                move_x+=mouse_data[1];//x坐标加
```

```
            }
            if(mouse_data[0]&0x20)
            {
                move_y-=(256-mouse_data[2]);//y坐标减
            }
            else
            {
                move_y+=mouse_data[2];//y坐标加
            }
        }
    }
}

/*****************************************************
                外部中断1
*****************************************************/
void ReceiveData(void)  interrupt 2
{
    if(mouse_buffer_bit<=10)
    {
        while(mouse_CLK==0);//等待设备拉高时钟线
        mouse_buffer[mouse_buffer_bit++]=mouse_SDA;//接收数据
    }
    if(mouse_buffer_bit==10)
    {
        data_analyse();//数据分析及处理
        mouse_buffer_bit=0;
    }
}
#endif
```

**********延时函数**********

```
//晶振11.0592M
#ifndef DELAY52_H
#define DELAY52_H
#define uchar unsigned char
#define uint unsigned int
#include<intrins.h>

//起始值delayus(1)=27us,间隔9.9us
void delay10us(unsigned int t)
{
    while(t--);
}

void delayms(unsigned int t)
{
    unsigned int i;
    unsigned char j;
    for(i=0;i<t;i++)
        for(j=0;j<125;j++);
```

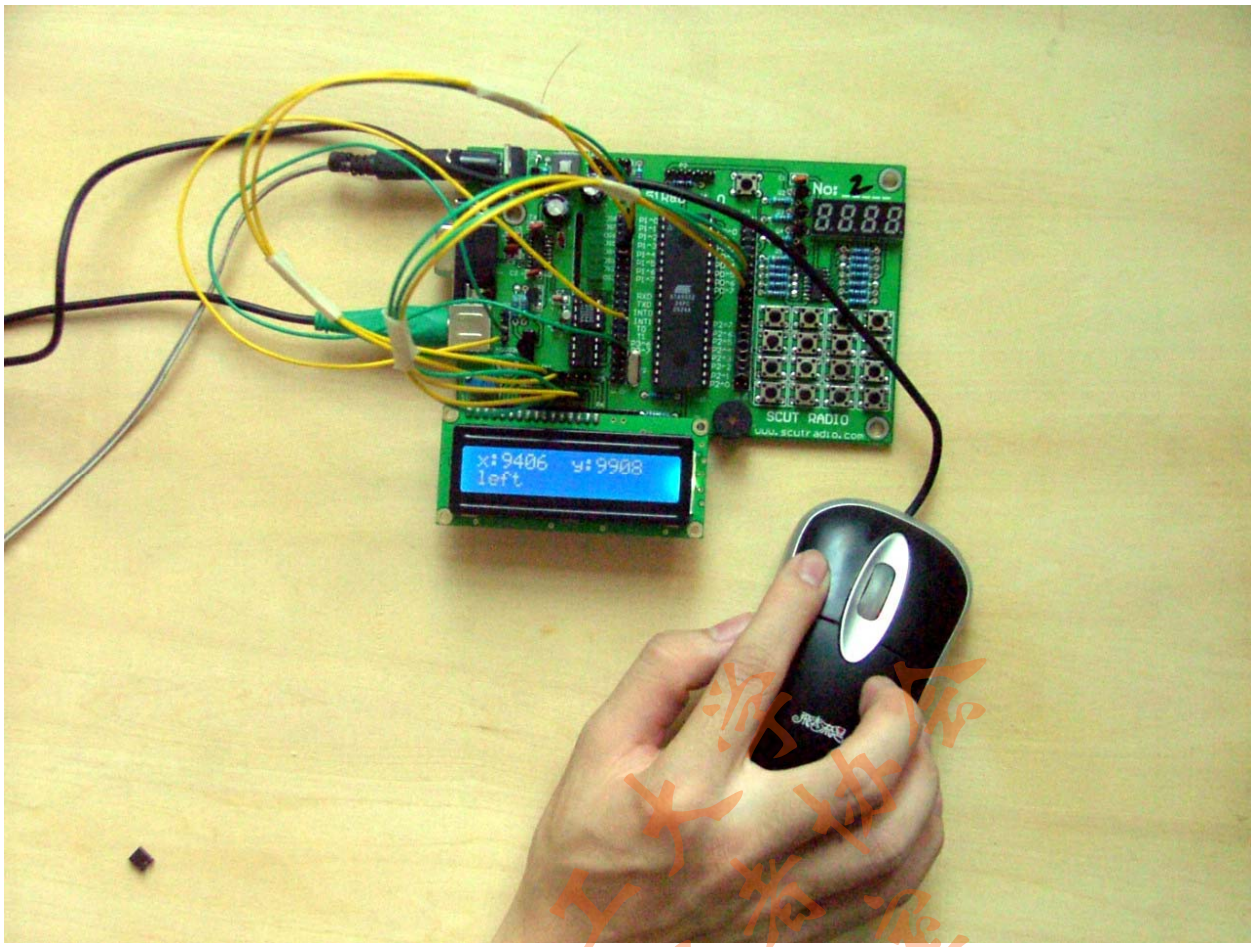
```
}
#endif

**********液晶1602驱动**********
#ifndef    LCD1602_4_H
#define LCD1602_4_H
#include <intrins.h>

#define  LCM1602_DATA    P0

sbit LCD1602_RS=P1^0;
sbit LCD1602_RW=P1^1;
sbit LCD1602_EN=P1^2;

//*************************************************************************
void LCM1602_Init(void);                                    //液晶初始化
void LCM1602_write_cmd(unsigned char command);              //写命令
void LCM1602_write_data(unsigned char temp);                //写数据
void LCM1602_set_xy(unsigned char x, unsigned char y);      //设置坐标
void LCM1602_write_char(unsigned x,unsigned char y,unsigned char dat);      //
写一个字符到第x行y列
void LCM1602_write_string(unsigned char x,unsigned char y,unsigned char *s);//写
字符串到第x行y列
void LCM1602_Read_BF(void);                                 //读忙信号
void num(unsigned char x,unsigned char y,unsigned int n); //在第x行,第y列显示整型
数字n
//void num16(unsigned char row,unsigned char col,unsigned int n);/在第row行,col
列处显示16进制数据n
//void num_float(unsigned char x,unsigned char y,float n);   //在第x行,第y列显示
数字n
//*************************************************************************

void LCM1602_Init(void)
{
    LCM1602_write_cmd(0x28);
    LCM1602_write_cmd(0x28);
    LCM1602_write_cmd(0x28);//设置4位数据传输模式
    LCM1602_write_cmd(0x0C);
    LCM1602_write_cmd(0x80);
    CLEARSCREEN;
}


void LCM1602_Read_BF(void)
{
    LCD1602_RW=1;     //RW 1
    LCD1602_RS=0;         //RS 0
    LCD1602_EN=1;     //EN 1 Read BF
    LCM1602_DATA=LCM1602_DATA&0x0F|0xf0;
    while(LCM1602_DATA&0x80);
```

```
    LCD1602_EN=0;
}
void LCD_en_write(void)          //EN端产生一个高电平脉冲，写LCD
  {
    LCD1602_EN=1;
    _nop_();
    LCD1602_EN=0;
  }

//*************************************
void LCM1602_write_cmd(unsigned char command)
{
    LCM1602_Read_BF();
    LCD1602_RS=0;     //RS 0
    LCD1602_RW=0;     //RW 0

    LCM1602_DATA&=0x0F;
    LCM1602_DATA=command&0xf0 | LCM1602_DATA&0x0f;
    LCD_en_write();
    command=command<<4;
    LCM1602_DATA&=0x0F;
    LCM1602_DATA=command&0xf0 | LCM1602_DATA&0x0f;
    LCD_en_write();
}
//*********************************
void LCM1602_write_data(unsigned char dat)
{
    LCM1602_Read_BF();
    LCD1602_RS=1;  //RS 1
    LCD1602_RW=0;  //RW 0
    LCM1602_DATA &=0x0F;
    LCM1602_DATA=dat&0xf0 | LCM1602_DATA&0x0f;
    LCD_en_write();
    dat=dat<<4;
    LCM1602_DATA &=0x0F;
    LCM1602_DATA=dat&0xf0 | LCM1602_DATA&0x0f;
    LCD_en_write();
}
//*************************************************
void LCM1602_set_xy(unsigned char x,unsigned char y)
{
    unsigned char address;
    y&=0x0f;
    if(!x)
        address=0x80+y;
    else
        address=0xc0+y;
    LCM1602_write_cmd(address);
}
//*****************************************************************
```

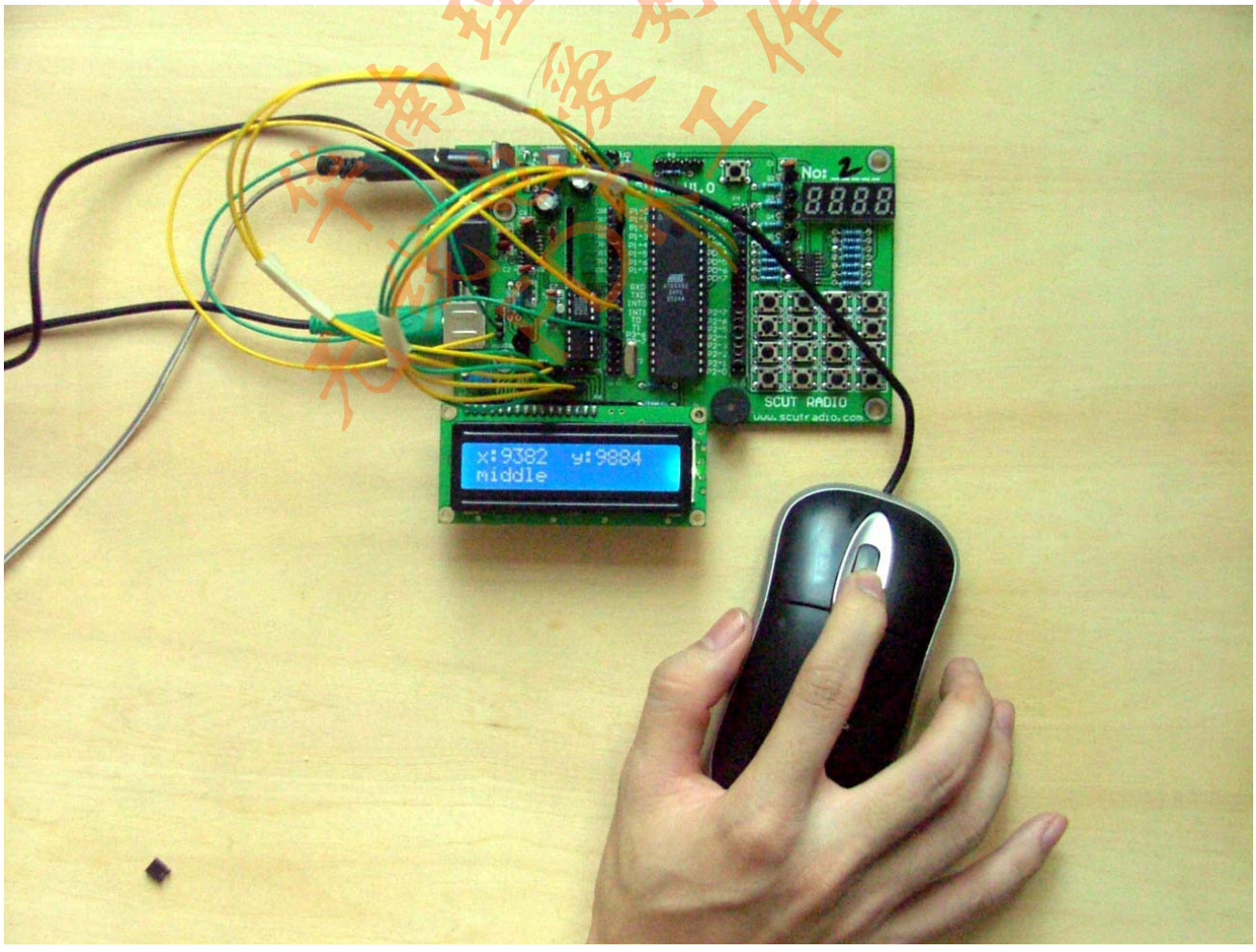
```
void LCM1602_write_char(unsigned x,unsigned char y,unsigned char dat)
{
    LCM1602_set_xy(x,y);
    LCM1602_write_data(dat);
}
//*********************************************************************
void LCM1602_write_string(unsigned char x,unsigned char y,const unsigned char *s)
{
    LCM1602_set_xy(x,y);
    while(*s)
    {
        LCM1602_write_data(*s);
        s++;
    }
}

void num(unsigned char x,unsigned char y,unsigned int n)
{
    unsigned char i,length,a[6]={0,0,0,0,0,0};
    unsigned int nx=n;
    if(n==0){LCM1602_write_char(x,y,'0');return;}
    for(i=0;i<6;i++)
    {
        if(nx>=1)length++;
        nx/=10;
    }
    nx=n;
    for(;length>0;length--)
    {
        a[length-1]=nx%10+48;
        nx/=10;
    }
    LCM1602_write_string(x,y,a);
}
#endif
```

按硬件连接所示,把硬件连接好,将程序编译后写进单片机之后就可进行试验,通过移动鼠标和按鼠标上的按键就可以到LCD1602显示屏上看到对应的变化了.

四,效果图:


No: 2
SCUT RADIO
x:9406 y:9908
left

图一


No: 2
SCUT RADIO
x:9382 y:9884
middle

图二