WTC6508BSI 八通道电容式触摸感应按键芯片

1　产品概述

8 个触摸感应按键，两线串口输出，带背光控制和蜂鸣器指示。NSOP16 封装.

2　订货信息

WTC6508BSI 提供两种按键反应模式，从型号上加以区分，用户订货时须提供完整的产品型号

产品型号	按键反应模式	应用注意事项
WTC6508BSI	SHIFT 模式，即多键顺序反应	可用作密集键盘
WTC6508BSI-M	任意三个按键同时按下，同时反应同时输出	不可用作密集键盘

3　主要应用

用于音响，各小家电，电视机，显示器，卫浴设备，工业控制，娱乐设备等

4　产品引脚定义

4.1　WTC6508BSI 的引脚图和引脚定义

图 1 是 WTC6508BSI 的引脚图

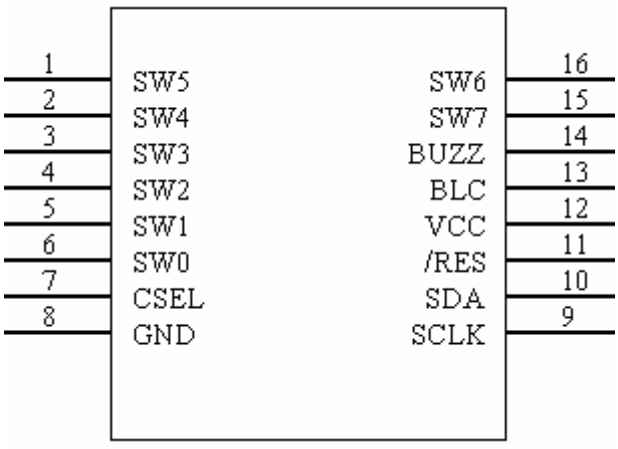


图 1　WTC6508BSI 引脚图

表1是WTC6508BSI引脚定义

表1

管脚序号	管脚名称	用法	功能描述
1	SW5	I	电容传感器（感应按键）接口5
2	SW4	I	电容传感器（感应按键）接口4
3	SW3	I	电容传感器（感应按键）接口3
4	SW2	!	电容传感器（感应按键）接口2

5	SW1	I	电容传感器（感应按键）接口1
6	SW0	I	电容传感器（感应按键）接口0
7	CSEL	I	灵敏度调整电容接口
8	GND	I	电源地
9	SCLK	I	数据传送的时钟输入脚
10	SDA	I/O	数据传送的数据输出脚

11	/RES	I	芯片复位脚
12	VCC	I	正电源输入
13	BLC	O	接近背光控制脚
14	BUZZ	O	蜂鸣器控制脚
15	SW7	I	电容传感器（感应按键）接口7
16	SW6	I	电容传感器（感应按键）接口6

4.2 WTC6508BSI 的工作电路图

WTC6508BSI 的外围电路很简单，只需少量阻容元件即可工作，图 2 是带有推荐串行接口电路的 WTC6508BSI 应用电路图。

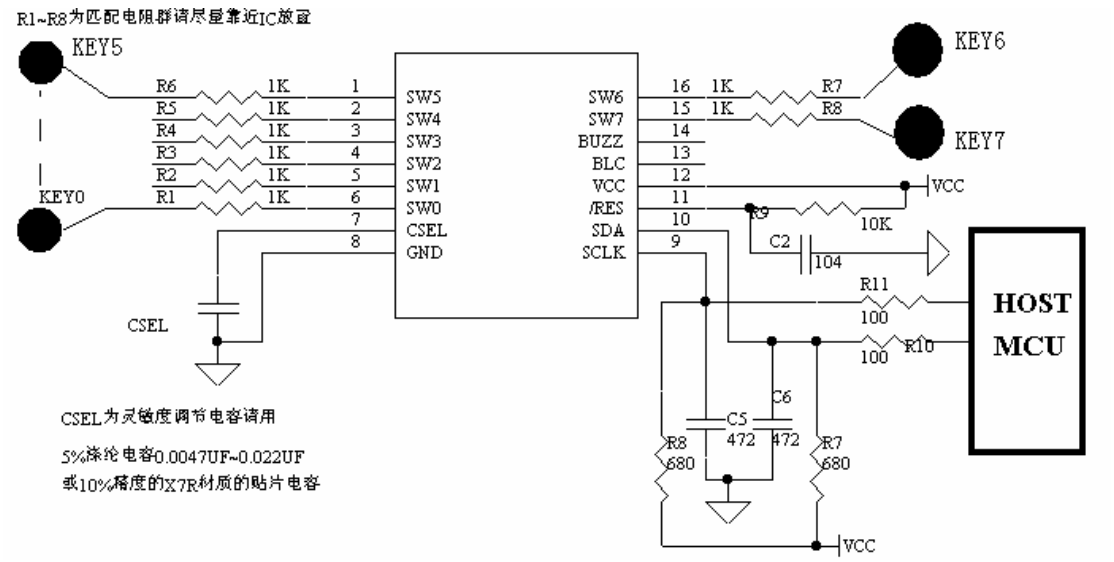


图 2;WTC6508BSI 的工作电路图

5 WTC6508BSI 蜂鸣器和背光控制信号

5.1　背光控制

WTC6508BSI 的 BLC 脚可以作为触控板的背光控制信号输出。当检测到手指接近感应盘时 BLC 输出高电平，当手指离开触控板后 9 秒钟后 BLC 恢复低电平。BLC 输出高电平时可以提供 4mA 的源电流驱动。如果 LED 背光所需电流超过 4mA 需外加驱动电路以免损坏 IC。

5.2　蜂鸣器控制

当检测到感应盘（sense element）上有有效触摸发生在 80ms 内 WTC6508BSI 的 BUZZ 脚输出 50ms 的低电平信号，客户可以利用这个信号来实现按键唤醒功能，还可以同时外接一个三极管推动一个直流蜂鸣器实现触摸按键的声音指示。

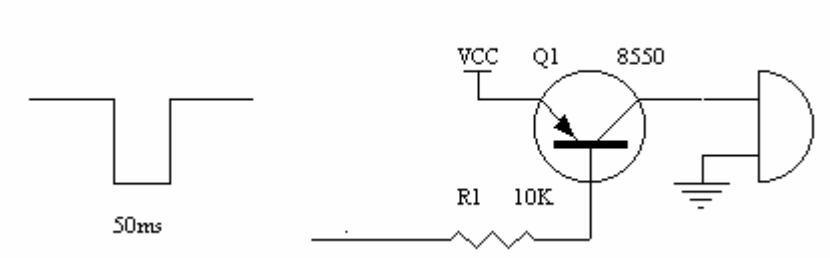


图 5

BUZZ 是触摸按键被触摸后输出的一个指示脉冲。并不表示键的触摸时间。触摸按键键值会一直保持到手指离开，只需要在程序中持续读取并判断就可以得到按键的实际状态。

6　输入输出接口

6.1　两线串行接口简介

WTC6508BSI 采用两线串行接口和主控 MCU 进行通信，主控 MCU 可以通过 SCLK 和 SDA 读取触摸按键的开关信息，并可以通过串行接口设置触摸按键的感应灵敏度。SCLK 是串行接口的时钟信号，SDA 是串行接口的数据信号。

SCLK 的速度

因为触摸芯片处理串行数据需要一定的延时时间，并且为了在总线上有偶发的噪声脉冲的情况下 WTC6508BSI 能够有自动总线复位能力。**建议主控 MCU 产生的 SCLK 方波速度为 20KHZ~2KHz 之间**，并且**每隔 15ms 以上的时间进行一次操作**。这样可以准确稳定的读写触摸芯片的串行总线。

建议主控 MCU 与 WTC6508BSI 的串行接口采用以下电路，如图 6，以尽量减少串行接口上的噪声，尤其是在信号线较长的情况下。

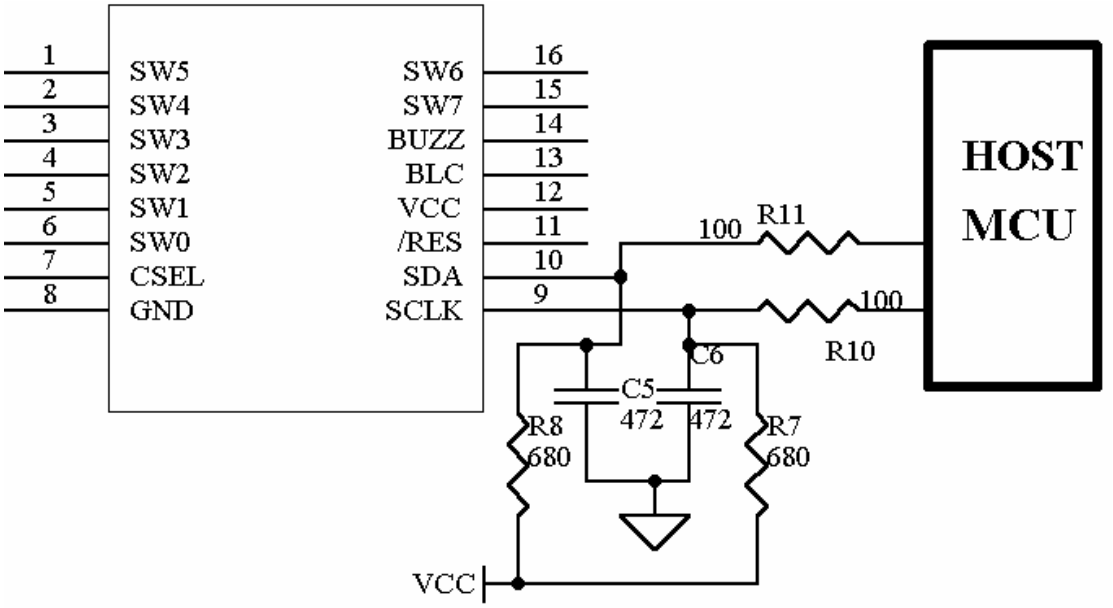


图 6：MCU 与 WTC6508BSI 的串行接口电路

R7，R8 为串行总线的上拉电阻，因为触摸芯片的 SCLK 和 SDA 平时是高阻输入状态，所以外部需要加上拉电阻。

R9,R10 和 C5,C6 构成两个 RC 滤波器，用于滤除 SCLK 和 SDA 上的“毛刺”噪声。

触摸芯片的 SCLK 脚总是保持高阻的输入状态，触摸芯片的 SDA 脚在主控 MCU 读取按键信息时是输出状态，其他任何时候都保持高阻的输入状态。

6.2 读取触摸按键信息

图 7 是主控 MCU 读取按键信息时触摸芯片的输出时序图

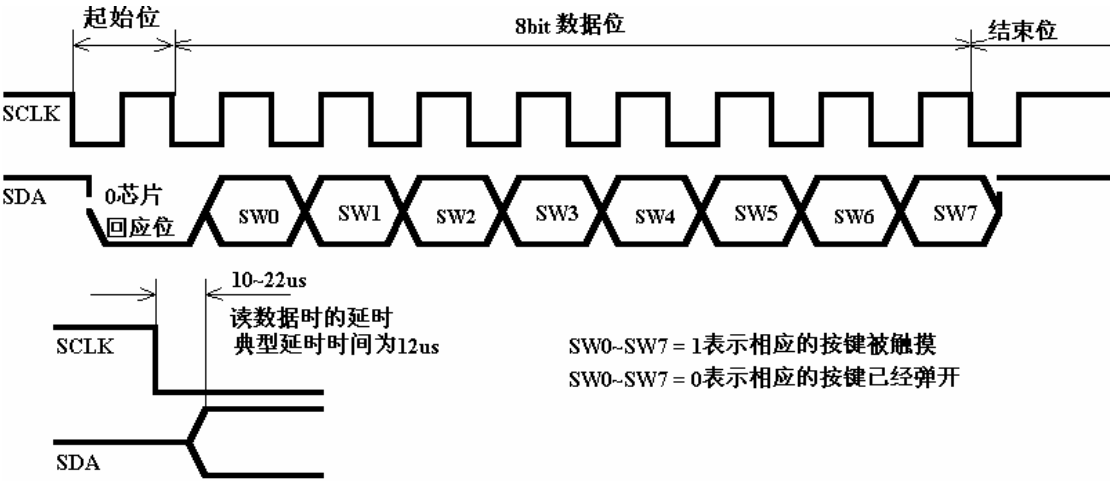


图 7：主控 MCU 读取按键信息时序

完成一次完整的读取按键信息操作，主控 MCU 需要在 SCLK 上产生 10 个低电平脉冲，分为 1 个起始位，8 个触摸按键数据位，1 个结束位。

1：产生起始位

触摸芯片的 SDA 脚平时为高阻的输入状态，当主控 MCU 也将 SDA 置为输入状态时，外部上拉电阻将 SDA 上拉为高电平，主控 MCU 将 SCLK 置为低电平，在 **10~22us 的延时后**触摸芯片会将 SDA 脚置为输出态并输出低电平作为开始传输按键信息的回应信号。主控 MCU 将 SCLK 置高完成起始位设定

2：读取 8 bit 触摸按键信息位

主控 MCU 再次将 SCLK 置为低电平，在 **10~22us 的延时后**触摸芯片会将 SW0 上连接的感应按键的状态放到 SDA 上。如果感应按键没有被触摸或已经弹开 SDA 为“0”，感应按键被触摸 SDA 为“1”。主控 MCU 读取 SDA 状态后将 SCLK 置高

依此类推主控 MCU 在 SCLK 上产生 8 个方波读取 8bit 的触摸按键信息。

3：产生 1 个结束位

MCU 读取 8bit 的触摸按键信息完成后再次产生一个脉冲以产生 1 个结束位，触摸芯片收到结束位后将 SDA 脚恢复置为高阻的输入状态。主控 MCU 完成一次读取操作。

MCU 不产生结束位的话触摸芯片将保持 SDA 的输出状态，这会影响芯片的工作。

6.3 设置触摸按键的感应灵敏度

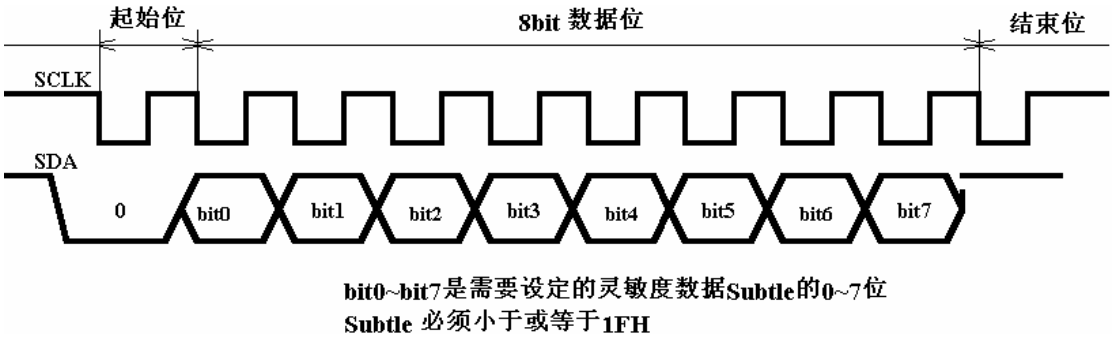


图 8：主控 MCU 设定触摸按键灵敏度操作时序

完成一次用软件设置触摸按键感应灵敏度的操作，主控 MCU 需要在 SCLK 上产生 10 个低电平脉冲，分为 1 个起始位，8 个灵敏度数据位，1 个结束位。

1：产生起始位

触摸芯片的 SDA 脚平时为高阻的输入状态，当主控 MCU 先将 SDA 置为低电平，然后再将 SCLK 置为低电平，在 **10~22us 的延时后**触摸芯片会转入接收数据的读取准备状态。主控 MCU 将 SCLK 置高完成起始位设定

2：将要设定的灵敏度级数传入灵敏度数据的暂存区

主控 MCU 先将需要设定的灵敏度级数的第 1 位放到 SDA 上，然后将 SCLK 置为低电平，在 **10~22us 的延时**后触摸芯片会将 SDA 上的数据读入暂存区的第 1 位。主控 MCU 将 SCLK 置高

以此类推，主控 MCU 将后面 7 位数据放到 SDA 上然后在 SCLK 上产生 7 个低电平脉冲，完成将数据传入暂存区的操作。

3：产生 1 个结束位

完成上述传输后主控 MCU 将 SDA 置为输入状态，同时在 SCLK 上产生一个低电平脉冲，以完成结束位的设定。

WTC6508BSI 的串行输入电路采用双缓冲结构，主控 MCU 将数据写入的 8bit 期间，电路将串行输入的数据送入独立的串入暂存区，触摸芯片的灵敏度不会变化。只有当主控 MCU 在 SCLK 上产生结束位后触摸芯片才会将触摸芯片重新复位并且使用本次设定的灵敏度级数重新设置内部相关参数。

6.4 软件设置触摸按键感应灵敏度的注意事项

主控 MCU 每次设定灵敏度后触摸芯片都会重新复位，复位需要 50ms 的时间。在触摸

芯片重新复位的 50ms 时间内，不论读写都不会得到正确的结果。所以主控 MCU 设定灵敏度完成后必须等待 50ms 以上的时间才能对 WTC6508BSI 的串行接口进行读写。

建议主控 MCU 在程序中不要频繁的对触摸芯片的灵敏度进行设定。只要在程序初始化时设定一次就可以了。

触摸按键感应灵敏度分为 32 级，相应的级数数据为 1～32。级数越高触摸按键的灵敏度就越高。但设定的级数数据如果超过了 32 或为 0，本次设置将无效，无效设置发送结束位后触摸芯片不会重新复位，也不会进行内部参数的调整，仍然会使用以前的灵敏度参数。

触摸芯片初次上电后灵敏度内部自动设定为第 29 级，用户也可以不使用软件调整灵敏度。直接使用芯片的默认参数。

7 组合键设计

7.1 标准 WTC6508BSI 的多键组合（SHIFT)工作模式

使用 WTC6508BSI 时，如果用户先后按下多个按键不释放，则多个按键都能依次作出反应。系统设计者可以据此设计出多种按键组合操作功能。

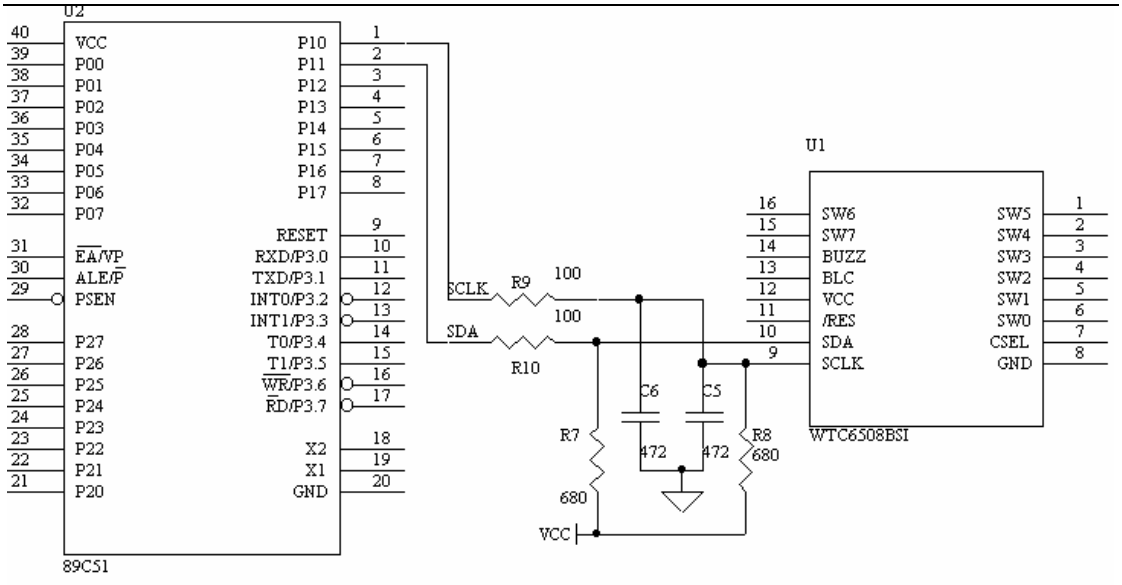
7.2 WTC6508BSI-M 可任意三键同时反应

使用 WTC6508BSI-M 任意三键可同时操作，数据可同步输出，此项特性可以让系统设计者能够满足同时触摸 2-3 键操作来实现组合键的客户需求。

8 WTC6508BSI 封装图及尺寸

9 WTC6508BSI 与 MCU 8051 的接口电路和操作程序范例

WTC6508BSI 与 8051 接口的典型应用：



典型操作程序程序对应电路

8051 与 WTC6508BSI 接口的典型操作程序

```
/*
项目描述：
        触摸芯片 WTC6508BSI 的 DEMO 板配套程序
DEMO 板功能：
        检测触摸芯片输出的信息，并点亮相应的 LED 完成显示
  MCU：          AT89C51
  晶振频率：     12MHz
  版本信息：     V1.0
*/

#include    <reg51.h>
#include     <INTRINS.H>
#define     uchar   unsigned char
#define     uint    unsigned int

#define     TIMER1_H    0xFC    //1ms 定时器初值
#define     TIMER1_L    0x17

//-------------------------------主机和触摸芯片的接口----------------------------------------
sbit    SDA   =   P1^1;       //数据线
sbit    SCLK  =   P1^0;       //时钟信号
```

```
sbit    LED0   =   P0^0;        //触摸按键对应的状态 LED 灯，低电平时点亮
sbit    LED1   =   P0^1;
sbit    LED2   =   P0^2;
sbit    LED3   =   P0^3;
sbit    LED4   =   P0^4;
sbit    LED5   =   P0^5;
sbit    LED6   =   P0^6;
sbit    LED7   =   P0^7;
//-------------------------------------the funtion define-----------------------------------
void    init(void);                 //89C51 的初始化程序
void    delay_24us(void);       //延时程序(延时 24us)

uchar   read_key_data(void);              //读触摸按键信息
void    set_subtle_SP(uchar subtle);    //用软件设置触摸芯片的触摸灵敏度

void    disp_key_led(uchar KeyValue); //用 LED 显示相应触摸按键的开关状态
//-------------------------------------the register define ---------------------------------

uchar       i,j;            //程序中使用的中间变量

uint    Counter1ms;     //timer1  每 1ms 中断一次对 Counter1ms 加 1
uchar   KeyValue;       //从触摸芯片读取的按键信息

//-----------------------------------------------------------------------------------------
// 主程序
//-----------------------------------------------------------------------------------------
void main(void)
{
  init();
  Counter1ms = 0;
  do
  {
  } while(Counter1ms <= 100);       //等待 100ms 确保触摸芯片完成复位
  set_subtle_SP(28);                    //设定触摸芯片的触摸灵敏度为 28 级

  Counter1ms ＝ 0;
  do
  {
  } while(Counter1ms <= 50);        //等待 50ms 确保触摸芯片完成重新复位
```

```
  while(1)
  {
        while(Counter1ms >= 15)        //每 15ms 读一次键值，并完成显示
        {
            Counter1ms = 0;            //15ms 计数器归零

            KeyValue = read_key_data();    //读触摸按键的按键信息

            disp_key_led(KeyValue);        /显示相应的 LED
        }
  }
}

//-------------------------------------------------------------------------------------
//功能：从触摸芯片读取触摸按键的信息
//入口：无
//返值：从触摸芯片读取到的触摸按键的开关信息
//-------------------------------------------------------------------------------------
uchar    read_key_data(void)
{
    uchar   KeyValue;
    TR1 =0;
    i = 0;

     //-----------------------------------------起始位的设定-------------------------
    SDA =1;                    //SDA 设置为输入状态,上拉电阻将 SDA 拉高置 1,表示要从
                               //触摸芯片读出触摸按键的按键信息
    SCLK = 0;                  //产生串行时钟的下降延信号
    delay_24us();

    if(SDA == 1)                    //发送起始位后检测触摸芯片是否发送了确认信号
    {                  //触摸芯片没有将 SDA 设置为"0".表明触摸芯片没有做好发送准备
        SCLK = 1;                   //SCLK 重新置为高电平
        return(0);             //函数直接返回 0 表示没有读到有效的键值
    }
     SCLK = 1;                      //产生串行时钟的上升延信号
    delay_24us();
     //----------------------读取触摸按键的开关信息（8 bit）--------------------
    do
```

```
    {

        KeyValue >>= 1;
        SCLK = 0;           //产生串行时钟的下降延信号
        delay_24us();
        if(SDA == 1)        //读取 SDA 上的数据
        {
            KeyValue |= 0x80;
        }
        else
        {
            KeyValue &= 0x7F;
        }
        SCLK = 1;               //产生串行时钟的上升延信号
        delay_24us();
        i++;
    }
    while(i < 8);          //读 8bit 数据
    //----------------------//发送结束位（总线恢复）
    SCLK = 0;               //产生串行时钟的下降延信号
    delay_24us();
    SCLK = 1;               //产生串行时钟的上升延信号
    delay_24us();
    TR1 =1;
    return(KeyValue);       //返回读取的触摸按键信息
}



/*-------------------------------------------------------------------------------------
//功能：设置触摸芯片的触摸灵敏度
//入口：subtle: 准备设定的触摸灵敏度级数(1~32)
//      subtle 的值在 1~32 内级数越高灵敏度越高.
//      如果 subtle > 32 或 subtle =0  本次设定操作无效.触摸芯片保持灵敏度不变
//返值：无
-------------------------------------------------------------------------------------*/
void set_subtle_SP(uchar subtle)        //用软件设置触摸芯片的触摸灵敏度
{
  TR1 = 0;
  //----------------------//发送 SDA 总线写数据的起始位----------
```

```
    SDA =0;             //起始位置 0,表示要向触摸芯片写入灵敏度数据
    SCLK = 0;           //产生串行时钟的下降延信号
    delay_24us();
    SCLK = 1;           //产生串行时钟的上升延信号
    delay_24us();

    //----------------------向触摸芯片写入需要设定的灵敏度数据-------
    i = 0;
    do
    {
      if((subtle & 0x01) == 0) {SDA = 0;} //准备写入的数据
      else                     {SDA = 1;}
      SCLK= 0;                          //产生串行时钟的下降延信号
      delay_24us();
      SCLK = 1;                         //产生串行时钟的上升延信号
      delay_24us();
      subtle >>= 1;
      i++;
    }
    while(i <8);        //写入 8 位数据
    //--------------------产生结束位,触摸芯片用新的灵敏度级数重新设定参数并重新复位-------
    SDA = 1;            //将 SDA 置为输入状态
    SCLK = 0;               //产生串行时钟的下降延信号
    delay_24us();
    SCLK = 1;                   //产生串行时钟的上升延信号
    delay_24us();
    TR1 = 1;
  }

/*-------------------------------------------------------------------------------------------------------
//功能:手指触摸按键时点亮被触摸按键上方的指示 LED,手指从触摸按键上移开后熄灭相应
按键上方的指示 LED
//入口：从触摸芯片读取到的按键信息
//返值：无
-------------------------------------------------------------------------------------------*/
void disp_key_led(uchar KeyValue)
{
        if((KeyValue & 0x01) != 0) {LED0 ＝ 0;}   //KEY0 被按下点
                                                  //亮相应的 LED
        else                          { LED0 ＝ 1;}   //KEY0 弹起,熄灭
```

```
        if((KeyValue & 0x02) != 0){ LED1 = 0;}   //KEY1 被按下
        else                      { LED1 = 1;}   //KEY1 弹起

        if((KeyValue & 0x04) != 0){ LED2 = 0;}   //KEY2 被按下
        else                      { LED2 = 1;}   //KEY2 弹起

        if((KeyValue & 0x08) != 0){ LED3 = 0;}   //KEY3 被按下
        else                      { LED3 = 1;}   //KEY3 弹起

        if((KeyValue & 0x10) != 0){ LED4 = 0;}   //KEY4 被按下
        else                      { LED4 = 1;}   //KEY4 弹起

        if((KeyValue & 0x20) != 0){ LED5 = 0;}   //KEY5 被按下
        else                      { LED5 = 1;}   //KEY5 弹起

        if((KeyValue & 0x40) != 0){ LED6 = 0;}   //KEY6 被按下
        else                      { LED6 = 1;}   //KEY6 弹起

        if((KeyValue & 0x80) != 0){ LED7 = 0;}   //KEY7 被按下
        else                      { LED7 = 1;}   //KEY7 弹起

}

//-----------------------------------------------------------------------------------------
void      init(void)
{
   EA   = 1;                    //open the gloabe interrupt enable
   EX0 = 0;                     //disable expend 0 interrupt
   EX1=1;                        //enable expend 1 interrupt for remote
   ET0 = 1;
   ET1 = 1;
   ES =   0;
   //IT0 = 0;                   //level tigger
   //IT1 = 0;                   //level tigger
   IT0 = 1;                     //edge tigger
   IT1 = 1;                     //edge tigger
   //TMOD   &= 0xF0;
   TMOD= 0x11;                  //timer0, timer1 work as 16 bit timer
```

```
    TH0 = 0x00;
    TL0 = 0x00;
    TH1 = TIMER1_H;
    TL1 = TIMER1_L;
    TR0 = 0;
    TR1 = 1;                                    //timer1 start work
}

//-------------------------------------------------------------------------------------
void delay_24us(void)
{
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
}

//-------------------------------------------------------------------------------------
void    timer1(void) interrupt 3 using 0        //8051 的 timer1 中断服务程序
{
     TH1 = 0xFC;       //1ms 定时器重新设置初值
     TL1 = 0x17;
     TR1 = 1;
     Counter1ms++;          //1ms 计数器加一
```

```
}
```