八通道触摸感应开关(I^2C)

RH6010

规格书

Revision 1.5 2011-11-25

目　　录

1.简介

RH6010 是一款带 I^2C 总线接口的 8 通道电容式触摸感应控制开关 IC，可以替代传统的机械式开关。

RH6010 可通过 I^2C 总线接口配置成多种模式，可广泛应用于灯光控制、玩具、家用电器等产品中。

RH6010 系列共有 2 个版本，其中

RH6010 面向低功耗；

RH6010A 面向高灵敏度。

2.特性

- 工作电压：2.0V~5.5V
- 典型工作电流 60uA，低功耗模式工作电流小于 10uA(均指 3.0V 供电且不带负载的条件下)
- 采用 I^2C 通信协议，可通过设置内部寄存器配置成多种模式
- 带有中断输出脚，可用于唤醒主控 MCU
- 内置去抖动算法可有效防止由噪声导致的误动作
- 可用于玻璃、陶瓷、塑料等介质表面
- 芯片 I^2C 地址：78H

3.封装示意图

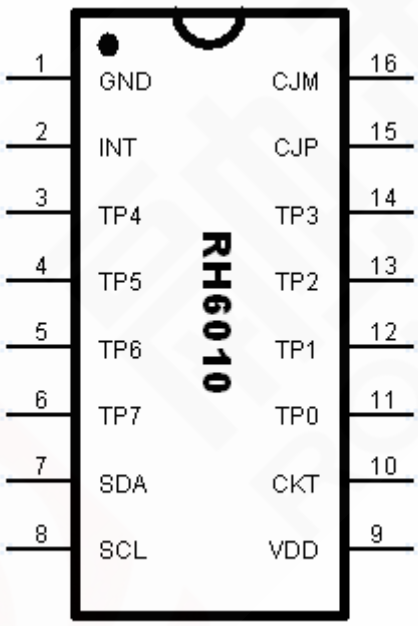


图 1 封装示意图**(SSOP16L(0.635-D1.40))**

表 1 引脚描述表

NO.	PADNAME	Descrption		NO.	PADNAME	Descrption
1	GND	负电源		**16**	CJM	外置电容 CJ 调节端
2	INT	中断指示标志		**15**	CJP	
3	TP4	触摸按键		**14**	TP3	触摸按键
4	TP5			**13**	TP2	
5	TP6			**12**	TP1	
6	TP7			**11**	TP0	
7	SDA	I^2C 通讯数据线		**10**	CKT	测试管脚
8	SCL	I^2C 通讯时钟线		**9**	VDD	正电源
注意：CKT 引脚在正常使用时悬空						

4.功能描述

通过设置内部寄存器，RH6010 既能配置为多种输出模式，也可灵活调节灵敏度和稳定度。

上电完成后若不需要对片内寄存器进行写操作，则设置为默认模式。

4.1 I2C总线

RH6010 采用标准 I²C 总线接口与主控 MCU 通讯，可通过该协议设置寄存器进行模式配置，也可读出按键状态值或寄存器值。

关于 I²C 详细信息请参考标准 I²C 通信协议资料。

注 1： RH6010 在工作中可动态多次配置。

注 2： I²C 地址：78H。

4.2 配置寄存器

可通过 I²C 总线的写指令写入寄存值。每次执行写寄存器指令后，芯片将更新配置寄存器值并自动重校准。

表 2 寄存器表

Reg.	Name	D7	D6	D5	D4	D3	D2	D1	D0
00H(R/W)	Config1	SCT[1:0]		SFT[5:0]					
01H(R/W)	Config2	OTC	MOT[1:0]		OLH	KEYT	LPM	RTC[1：0]	
02H (R)		Key7	Key6	Key5	Key4	Key3	Key2	Key1	Key0
03H(R/W)		*Reserved*							
注意： 02H 寄存器为只读									

4.2.1 灵敏度配置寄存器(SCT[1:0]& SFT[5:0])

SCT[1:0]可粗略地配置触摸传感器计数时长，计数时长越大，灵敏度越高。当 SCT[1:0]从 00B 向 11B 调节，灵敏度逐渐提高。

SFT[5:0]可较精细地配置触摸传感器计数时长。当 SFT[5:0]从 000000B 向 111111B 调节，灵敏度逐渐提高。

4.2.2 荡器频率配置寄存器(OTC)

OTC 配置触摸传感器的振荡器时钟输入有效频率。当 OTC 由 0 调节至 1，灵敏度提高。

表 3 OTC 寄存器

NAME	选项	功能	备注
OTC	=1	TCK	Default=1
	=0	TCK/2	

4.2.3 触摸最大开启时间寄存器(MOT[1:0])

最大开启时间指检测到触摸，且该触摸一直未释放(并无其余触摸通道被按下或释放)至 IC 自动重校准的时间。若触摸通道发生变化，则重新开始计数。

表 4 MOT 寄存器

NAME	选项	功能	备注
MOT[1:0]	00	约 100S	Default=11
	01	约 50S	
	10	约 20S	
	11	无穷大	
注意：最大开启时间为芯片在 VDD=3.0V 且无负载的条件下测得，不可作为精确时间			

4.2.4 中断标志(INT)电平配置寄存器(OLH)

INT 可设置触摸被检测到时输出的 INT 中断脉冲是高电平还是低电平脉冲有效 (脉冲宽度约 20us)。

表 5 OLH 寄存器

NAME	选项	功能	备注
OLH	=1	INT 输出高电平脉冲	Default=1
	=0	INT 输出低电平脉冲	

4.2.5 按键输出形式 KEY Type 配置寄存器(KEYT)

当 KEY Type=1 时为 multi-key 模式，即当同时有数个按键按住时，会同时承认几个按住的键；

当 KEY Type=0 时为 single-key 模式，即当有数个按键均按住时，只会承认最先被按下的键。

表 6 KEYT 寄存器

NAME	选项	功能	备注
KEYT	=1	multi-key 模式	Default=1
	=0	single-key 模式	

4.2.6 低功耗模式配置寄存器(LPM)

当 LPM=1 时，低功耗模式 disable，IC 会全速持续进行触摸检测；

当 LPM=0 时，低功耗模式 enable，IC 在一段时间没有检测到触摸则进入低功耗模式，但在此模式下仍会侦测触摸；当侦测到触摸，IC 进入全速模式。当所有触摸释放，IC 将继续保持全速模式检测一段时间，在这段时间内没有触摸则再次进入低功耗模式。

表 7 LPM 寄存器

NAME	选项	功能	备注
LPM	=1	低功耗模式 disable	Default=1
	=0	低功耗模式 enable	

4.2.7 重新自校准时间配置寄存器(RCT，Re-Calibration Time)

当低功耗模式 disable，重新自校准时间指 IC 未检测到触摸，重新自校正之间隔时间。

当低功耗模式 enable，重新自校准时间指 IC 检测到触摸并退出低功耗模式之后，当触摸释放，IC 先重新自校正，然后再进入低功耗模式之间隔时间。

表 8 RCT 寄存器

NAME	选项	功能		备注
		LPM=1	**LPM=0**	
RCT[1:0]	=00	约 1s	约 3.0s	Default=01
	=01	约 3s	约 6.3s	
	=10	约 6.3s	约 13s	
	=11	约 13s	约 26s	
注意：重新自校准时间为芯片在 VDD=3.0V 且无负载的条件下测得，不可作为精确时间				

4.2.8 上电默认配置

上述配置寄存器，在 IC 上电完成后，除 RCT[1:0] =01(4S)以外，其余寄存器均默认为 1。

4.3 键值寄存器

当 KEYn＝0 时，表示该触摸没有被检测到。
当 KEYn＝1 时，表示该触摸被检测到，且 INT 输出中断标志脉冲；松开后 KEYn 立即由 1 更新为 0(KEYn 键值为去抖动后更新键值)，该键值可通过 I^2C 读出。

5.绝对最大值

表 9 工作条件规格表

参数	符号	额定值	单位
工作电压	V_{DD}	-0.3～5.5	V
输入/输出电压	V_I / V_O	-0.5～V_{DD} +0.5	V
工作温度	T_{OPR}	-20 ~ 70	℃
储藏温度	T_{STG}	-40 ~ 125	℃

6.电气参数

表 10 电气参数表

参数	符号	测试条件	最小值	典型值	最大值	单位
工作电压	VDD		2.0	3.0	5.5	V
参考振荡器	MCK	VDD=3.0V 无负载	-	330K	-	Hz
传感器振荡器	TCK	VDD=3.0V 无负载	-	330K	-	Hz
工作电流	Iop	3.0V 无负载，全速模式	-	60.0	-	uA
		3.0V 无负载，低功耗模式	-	10	-	uA
若无特别说明，VDD 为 3.0V，环境温度为 25℃,芯片输出无负载						

7.应用电路

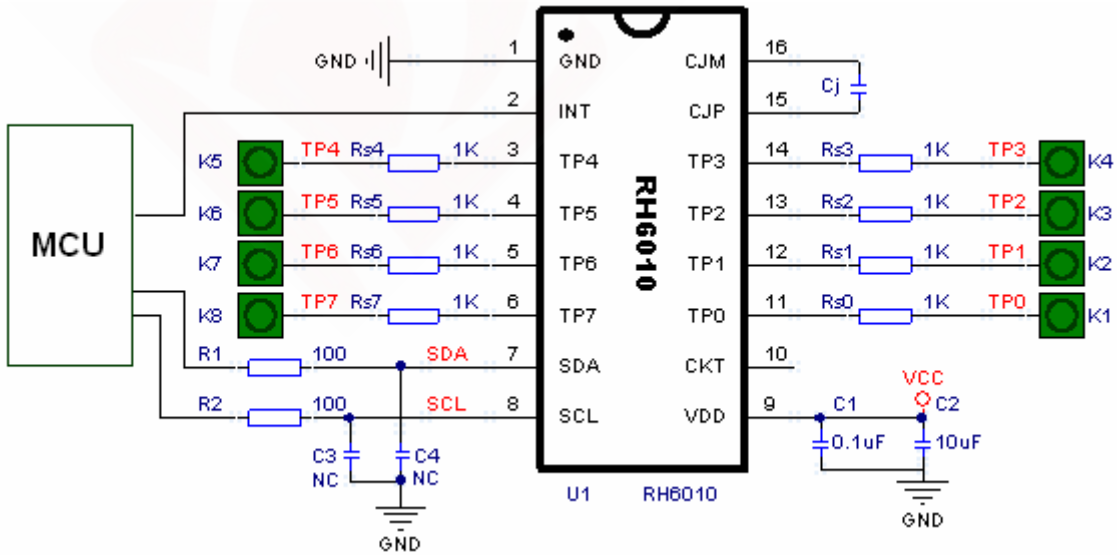


图 2 RH6010 应用示意图

说明：

1. CJ 指调节灵敏度的电容，电容值大小 0pF～50pF(电容值的增大将导致灵敏度降低)，可根据具体应用进行选择。
2. Rs 指在触摸电极和触摸输入脚之间串联的电阻，用于提高触摸的抗干扰能力，可根据具体应用进行选择。
3. VDD 与 GND 间需并联滤波电容以消除噪声。供电电源需稳定，如果电源电压漂移或者快速变化，可能引起灵敏度漂移或检测错误。
4. 请参看<RH60XX 应用指南>，以改善实际应用之可靠性。

8.穿透力应用说明

表11：穿透力与铺地、感应电极大小对应关系

感应电极面积	PCB顶层不铺地 底层不铺地	PCB顶层铺实铜 底层35%铺地
6×6mm	4mm	0.9 mm
7×7mm	5mm	1.1 mm
8×8mm	7mm	1.3 mm
10×10mm	8mm	1.8 mm
12×12mm	10mm	2.5 mm
15×15mm	13mm	3.5 mm

说明：

1. 此表仅供参考，具体焊盘大小应根据实际模具外壳厚度来调整。
2. 触摸焊盘面积越大，可穿透介质材料越厚。
3. PCB铺地比例越小，PCB点触焊盘与地之间的寄生电容越小，人体触摸后新生的手指电容相对PCB寄生电容变化越大，触摸灵敏度越高，可穿透介质越厚。
4. PCB铺地比例越小，越易受到外界干扰。
5. 建议实际应用时兼顾灵敏度和抗干扰设计PCB的铺地形式。如对穿透介质厚度要求不高，建议增加铺地比例以提高抗干扰性能。

9.I²C通讯及源代码

9.1 I²C总线的寻址

I2C总线协议明确规定：采用7位的寻址字节+1位读/写控制位。D7～D1位组成从机的地址，D0位是数据传送方向位，为“0”时表示主机向从机写数据，为“1”时表示主机由从机读数据。如下图：

位：	7	6	5	4	3	2	1	0
	从机地址							R/W

RH6010 I²C地址是0x78H，当主机向从机写数据时，寻址地址为0xF0；当主机向从机读数据时，寻址地址为0xF1。

9.2 RH6010 I²C通讯流程图

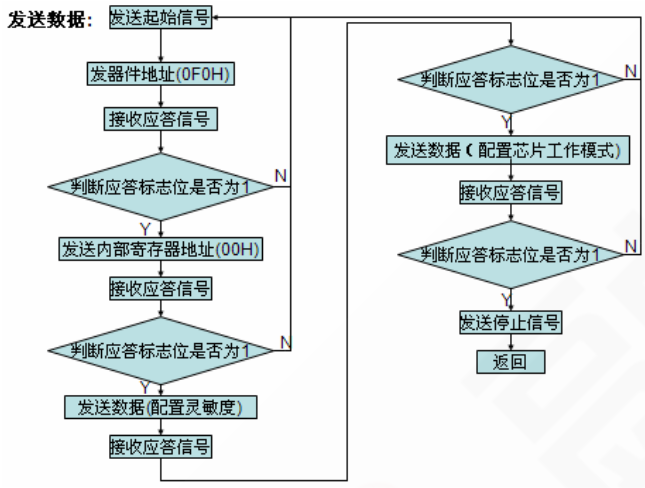


图3 发送数据流程图

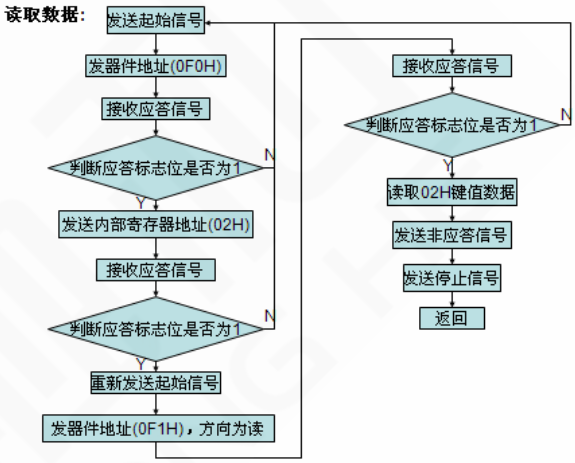


图4 读取数据流程图

9.3 RH6010源代码

9.3.1 汇编语言格式

```
    MOV     IIC_CONFIG, #0F5H
    ;单键模式、OTC最高频率、触摸最大开启时间无穷大、中断输出高电平、关闭低功耗模式、重新自校准时间约6.3s
    MOV     IIC_SLS     , #0FCH  ;设置灵敏度为0xFCH
;================================
IIC_SEND:           ;发送部分
    LCALL   IIC_START
    MOV     IIC_DATA,#0F0H   ;芯片地址
    LCALL   IIC_SEND_BYTE
    LCALL   IIC_REACK
    JB      ACK_BIT,IIC_SEND
    MOV     IIC_DATA,#00H       ;寄存器地址
    LCALL   IIC_SEND_BYTE
    LCALL   IIC_REACK
    JB      ACK_BIT,IIC_SEND
    MOV     IIC_DATA,IIC_SLS    ;灵敏度
    LCALL   IIC_SEND_BYTE
    LCALL   IIC_REACK
    JB      ACK_BIT,IIC_SEND
    MOV
IIC_DATA,IIC_CONFIG ;CONFIGURATION
    LCALL   IIC_SEND_BYTE
    LCALL   IIC_REACK
    JB      ACK_BIT,IIC_SEND
    LCALL   IIC_STOP
```

```
    RET
;------------------------------
IIC_RECEIVE:    ;读取部分
    LCALL   IIC_START
    MOV     IIC_DATA,#0F0H  ;芯片地址
    LCALL   IIC_SEND_BYTE
    LCALL   IIC_REACK

    JB      ACK_BIT,IIC_RECEIVE
    MOV     IIC_DATA,#02H       ;寄存器地址
    LCALL   IIC_SEND_BYTE
    LCALL   IIC_REACK

    JB      ACK_BIT,IIC_RECEIVE
    LCALL   IIC_START
    MOV     IIC_DATA,#0F1H  ;芯片地址
    LCALL   IIC_SEND_BYTE
    LCALL   IIC_REACK

    JB      ACK_BIT,IIC_RECEIVE
    LCALL   IIC_RECEIVE_BYTE ;读取02H寄存
器的内容
    LCALL   IIC_NACK
    LCALL   IIC_STOP
    MOV     TOUCH,IIC_RECE_TEMP
    RET

;------------------------------
IIC_SEND_BYTE: 发送一个字节
    MOV     A,IIC_DATA
    MOV     IIC_COUNTER,#08H
IIC_SEND_BYTE1:
    RLC A
    JC      WR_1
    LJMP        WR_0
IIC_SEND_BYTE2:
    DJNZ
    IIC_COUNTER,IIC_SEND_BYTE1
    RET
WR_1:
    SETB        IIC_SDATA
    SETB        IIC_SCL
    NOP
    NOP
    CLR IIC_SCL
    LJMP        IIC_SEND_BYTE2
WR_0:
    CLR IIC_SDATA
    SETB        IIC_SCL
    NOP
    NOP
    CLR IIC_SCL
    LJMP        IIC_SEND_BYTE2
;------------------------------
IIC_RECEIVE_BYTE:       ;接收一个字节
    MOV     IIC_COUNTER,#08H
    MOV     IIC_RECE_TEMP,#00H
IIC_RECEIVE_BYTE1:
    SETB        IIC_SDATA
    SETB        IIC_SCL
    NOP
    NOP
    MOV     C,IIC_SDATA
    MOV     A,IIC_RECE_TEMP
    CLR IIC_SCL
    RLC A
    MOV     IIC_RECE_TEMP,A
    DJNZ
    IIC_COUNTER,IIC_RECEIVE_BYTE1
    RET
;---------------------
IIC_START:  ;起始信号
    SETB        IIC_SDATA
    SETB        IIC_SCL
    NOP
    NOP
    CLR IIC_SDATA
    NOP
    NOP
    CLR IIC_SCL
    RET
;---------------------
IIC_STOP:   ;终止信号
    CLR IIC_SDATA
    SETB        IIC_SCL
    NOP
    NOP
    SETB        IIC_SDATA
```

```
    NOP
    NOP
    CLR IIC_SDATA
    RET
;----------------------
IIC_ACK:    ;应答信号
    CLR IIC_SDATA
    SETB    IIC_SCL
    NOP
    NOP
    CLR IIC_SCL
    SETB    IIC_SDATA
    RET
;----------------------
IIC_NACK:   ;非应答信号
    SETB    IIC_SDATA
    SETB    IIC_SCL
    NOP
    NOP
    CLR IIC_SCL
    CLR IIC_SDATA
    RET
;----------------------
IIC_REACK:  ;接收应答信号
    SETB    IIC_SDATA       ;作为输入
    NOP
    SETB    IIC_SCL     ;第九个时钟开始
    NOP
    MOV     C, IIC_SDATA;读SDA线
    MOV     ACK_BIT, C
    CLR IIC_SCL     ;时钟脉冲结束
    RET
```

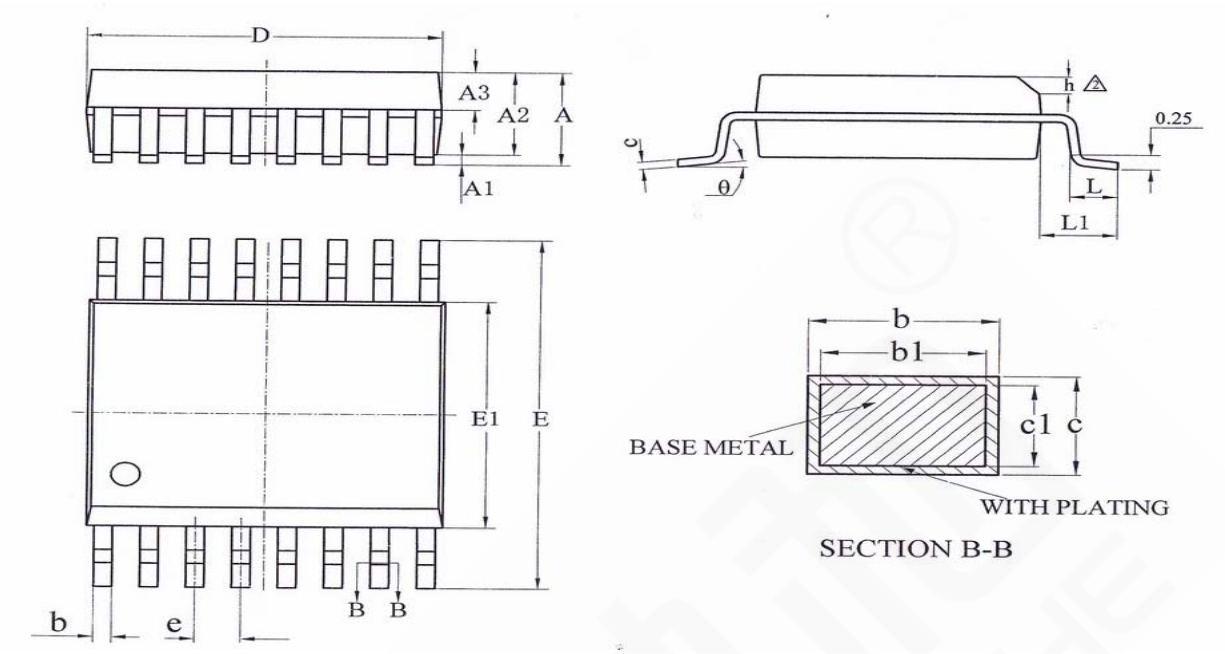
9.3.2 C程序格式

```
#include<reg52.h>
#include<I2C_C51.h>
#define uchar unsigned char /*宏定义*/
#define uint unsigned int
{   //主机写入数据
    ucFlagNeedWR = 0;
    do
      {
        while(1){
                Start();        //启动
                Send(0xf0);         //发送器件地址（7位）+0
                if(RecAck())   //判断返回，为0表示从机返回成功应答
                    break;
                Send(0x00);         //发送器内部子地址
                if(RecAck())
                    break;
                Send(ucSLS);
                if(RecAck())
                    break;
                Send(ucKeyTypeTemp);
                if(RecAck())
                    break;
                Stop();             //停止
                ucFlagGoodWR = 1;
                    break;
                }
            }while(!ucFlagGoodWR);
                ucFlagGoodWR = 0;
            }

;==============================

            while(1){ //主机读取数据
                Start();
                Send(0xf0);
                if(RecAck())
                    break;
                Send(0x02);
                if(RecAck())
                    break;
                Start();        //重启动
                Send(0xf1);   //发送器件地址（7位）+1（读）
                if(RecAck())
                    break;
                touch = Receive(); //读取数据
                NoAck();            //发送非应答，准备停止
                Stop();             //停止
                break;
            }
```

```
//I2C通讯IO口模拟程序
//只有在SCL线的时钟信号是低电平时才能改变
//起始条件: SCL 线是高电平时SDA 线从高电平向低电平切换
//停止条件: 当SCL 是高电平时SDA 线由低电平向高电平切换
//SDA 线上的数据必须在时钟的高电平周期保持稳定数据线的高或低电平状态
#include <reg51.h>
#include <intrins.h>

#define uchar unsigned char /*宏定义*/
#define uint unsigned int

sbit SDA = P1^1;
sbit SCL = P1^0;
//启动函数，在SCL为高时，SDA的下降沿为启动信号

void Start(void)
{
    SCL = 0;            //SCL处于低电平时,SDA才能改变
    SDA= 1;       // 一个"开始"状态,该状态必须在其他命令之前执行
    SCL = 1;            // 当scl为高电平时sda的下降沿表示开始状态
    _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); //给一个延时
    SDA= 0;             //给下降沿表示开始
    _nop_(); _nop_(); _nop_(); _nop_();
    SCL = 0;            //恢复低电平以改变sda的值
    SDA = 1;
}

//停止函数，在SCL为高时，SDA的上升沿为停止信号。
void Stop(void)
{
    SCL = 0;            //SCL处于低电平时,SDA改变数值 */
    SDA = 0;            //scl为高电平时，sda的上升沿表示停止，
    SCL = 1;            //scl为高电平时改变sda的状态表示启动，停止
    _nop_(); _nop_(); _nop_();_nop_();    //延时
    SDA = 1;
    _nop_(); _nop_(); _nop_();_nop_(); _nop_();
    SCL = 0;
}

//检查应答位
bit RecAck(void)
{
    SCL = 0;            //在scl为0的时候改变sda的值
    SCL = 1;            //在scl为1的时候等待sda值的变化，在器件接受到数据后会把sda拉低。
    _nop_(); _nop_(); _nop_(); _nop_();
    CY = SDA;           // 因为返回值总是放在CY中的
    _nop_();
    SCL = 0;
    return(CY);    //如果为CY为低则表示接受成功，如果为高，则表示接受失败。
}

/*
//对I2C总线产生应答（一般用在读操作中）

void Ack(void)
{
    SDA = 0;       // EEPROM通过在收到每个地址或数据之后,
    _nop_();_nop_();
    SCL = 1;       //置SDA低电平的方式确认表示收到读SDA口状态
    _nop_(); _nop_(); _nop_();
    _nop_(); _nop_(); _nop_();
    SCL = 0;
    _nop_();
    SDA = 1;
}
```

```
*/
//不对I2C总线产生应答

void NoAck(void)
{
    SDA = 1;
    SCL = 1;
    _nop_(); _nop_(); _nop_();
    _nop_(); _nop_(); _nop_();
    SCL = 0;
}


//向I2C总线写数据,每次写8位数据。

void Send(uchar sendbyte)
{
    uchar j = 8;
    do
    {
        SCL = 0;        //拉低scl准备给上升沿
        sendbyte <<= 1;    // 使
CY=sendbyte^7;
        SDA = CY;          // CY 进位标志位
        _nop_();_nop_(); _nop_();
        SCL = 1;           //给上升沿，发出
sda的状态值
        _nop_();_nop_();_nop_(); _nop_();
    }while(--j);
    _nop_();
    SCL=0;
}


//从I2C总线上读数据子程序 ，每次读8位数据

uchar Receive(void)
{
    register receivebyte;
    uchar i = 8;
    SCL = 0;
    SDA = 1;
    do{
        receivebyte <<= 1;
        SCL = 1;           //拉高scl准备给下
降沿
        _nop_();
        receivebyte |= SDA;     //接受值左移
一位把低位和sda相或得到sda的状态值
        SCL=0;             //给下降沿发出sda
的状态值
    }while(-- i);
    return(receivebyte);
}


//以下为头文件<I2C_C51.H>


#ifndef I2C_C51_H
#define I2C_C51_H

#ifndef uchar
#define uchar unsigned char
#endif

//启动函数
extern void Start(void);

//停止函数
extern void Stop(void);

//检查应答位
extern bit RecAck(void);

//对I2C总线产生应答
extern void Ack(void);

//不对I2C总线产生应答
extern void NoAck(void);

//向I2C总线写数据，每次8位
extern void Send(uchar sendByte);

//向I2C总线读数据，每次8位
extern uchar Receive(void);
#endif
```

10.封装信息（SSOP16L）



Symbol	Dimensions in mm		
	Min	Typ	Max
A	-	-	1.75
A1	0.10	-	0.225
A2	1.30	1.40	1.50
A3	0.50	0.60	0.70
b	0.24	-	0.30
b1	0.23	0.254	0.28
c	0.20	-	0.25
c1	0.19	0.20	0.21
D	4.80	4.90	5.00
E	5.80	6.00	6.20
E1	3.80	-	4.00
e	0.635BSC		
h	0.25	-	0.50
L	0.50	0.65	0.80
L1	1.05BSC		
θ	0	-	8°

注意！

规格如有更新，恕不另行通知。请在使用该 IC 前更新规格书至最新版本。