TC309 中文手册

目录

1．概述

人机接口要求更高的功能性和直观性，触摸式界面，迅速成为新的规范。
TC309 是一个 9 按键电容传感装置。该装置可以作为一个 9 键控制器。

2．特点

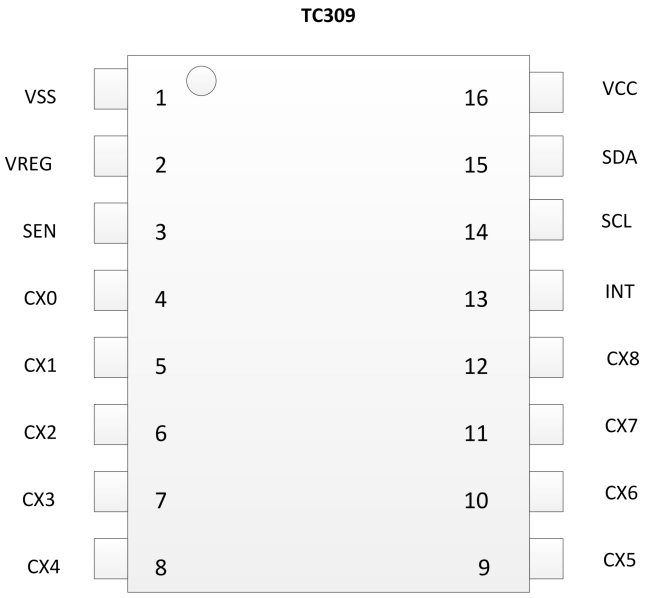
- ☐ 可以控制 9 个按键
- ☐ 自动灵敏度校正
- ☐ 系统低成本
- ☐ I2C 输出方式
- ☐ 降低系统复杂度提高稳定性

- ☐ 嵌入的共模干扰去除电路
- ☐ 空闲状态可以节省功耗
- ☐ RoHS 兼容的 SOP-16 封装

3. 应用

- ☐ 媒体播放器
- ☐ 消费类电子
- ☐ 家电应用
- ☐ 键盘
- ☐ 传统按键替换
- ☐ 密封控制面板

4. 管脚图示



5. 管脚描述

引脚	名称	输入/输出	描述
1	VSS	电源负极	地参考
2	VREG	模拟输出	内部参考源输出
3	SEN	模拟输入输出	灵敏度电容
4	CX0	模拟输入输出	感应输入 0（不使用时悬空）
5	CX1	模拟输入输出	感应输入 1（不使用时悬空）
6	CX2	模拟输入输出	感应输入 2（不使用时悬空）
7	CX3	模拟输入输出	感应输入 3（不使用时悬空）
8	CX4	模拟输入输出	感应输入 4（不使用时悬空）
9	CX5	模拟输入输出	感应输入 5（不使用时悬空）
10	CX6	模拟输入输出	感应输入 6（不使用时悬空）

11	CX7	模拟输入输出	感应输入 7（不使用时悬空）
12	CX8	模拟输入输出	感应输入 8（不使用时悬空）
13	INT	输出	通知主机有按键
14	SCL	输入	I2C 时钟输入
15	SDA	输入输出	I2C 数据输入输出
16	VCC	电源正极	供电电压输入

SEN

此管脚电容大小为10pf~100pf，电容越小灵敏度越高。

VREG

内部参考源输出，接4.7nf电容。

CX0~CX8

感应管脚 ,串联电阻是3KΩ。

INT

有按键时，输出低电平；无按键时，是高阻态。

SCL, SDA

SCL 是 I2C 时钟输入端口。SDA 是 I2C 数据输入输出端口。 SDA 端口有内部弱上拉。

6. 芯片功能

6.1 初始化时间

上电复位后，芯片需要300ms进行初始化，计算感应管脚的环境电容，然后才能正常工作。

6.2 灵敏度

灵敏度由SEN端口接的电容值决定。数值越小，灵敏度越高。

6.3 自校正

根据外部环境温度和湿度等的漂移，芯片会一直调整每个按键的电容基准参考值。从检测到按键开始，芯片会停止校正一段时间，这段时间大约50秒。然后芯片会继续自校正，也就是说检测按键有效的时间不会超过50秒。通过设置寄存器中的KVF位可以修改为一直输出有效。

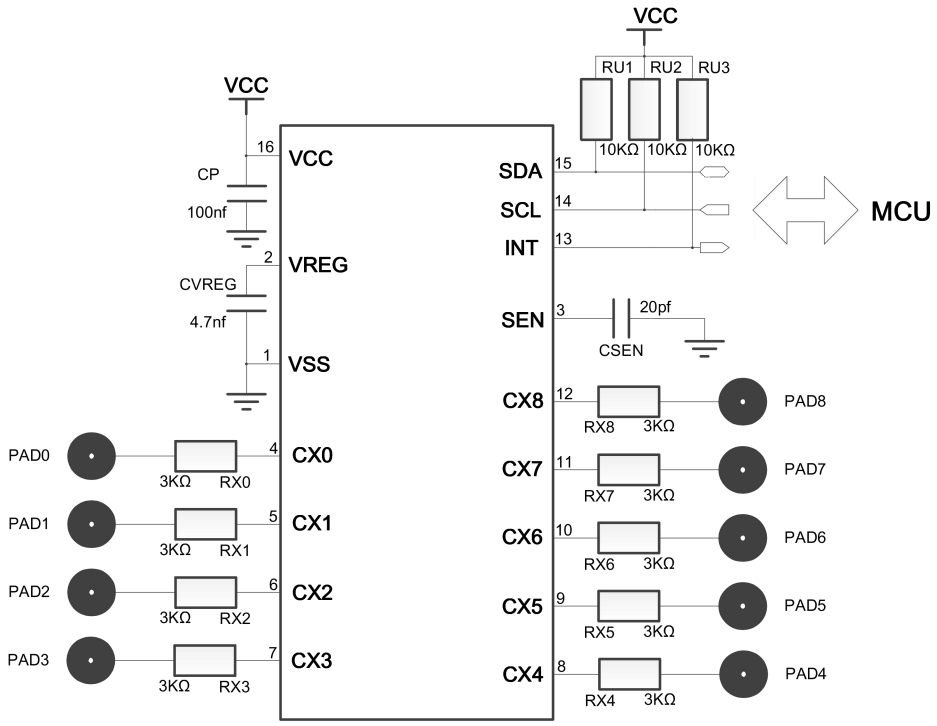
6.4 触摸反应时间

每个通道大约每隔12.5ms采样一次。经过按键消抖处理以后，检测到按键按下的反应时间大概是68毫秒，检测按键离开的反应时间大概是44毫秒。所以检测按键的最快频率大概是每秒9次。如果想提高反应速度，可以设置内部寄存器，详见**“8.6.2 控制寄存器 CTRL0中的RTM[1:0]”**

6.5 睡眠模式

如果在一段时间内（75秒左右）没有检测到按键并且SDA端口一直保持高电平，芯片会自动进入省电模式。只要让SDA保持高电平时间不超过75秒左右，芯片就不会进入睡眠模式。在睡眠模式中，按键的采样间隔会变长，电流消耗（Idd）会减小。如果检测到按键，芯片会马上离开睡眠模式，进入正常模式。

7. 应用原理图



注意:

CSEN 是灵敏度设置电容，取值范围是最小10pf，最大100pf，电容值越小灵敏度越高。

8. I2C 接口

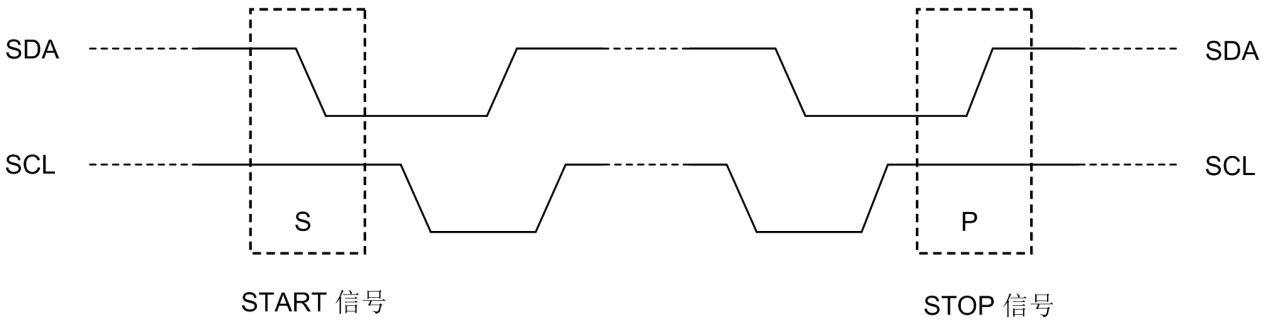
8.1 Start 和 Stop 信号

Start 信号(S)

当 SCL 是高电平时，SDA 由高到底变化，表示开始传输数据。

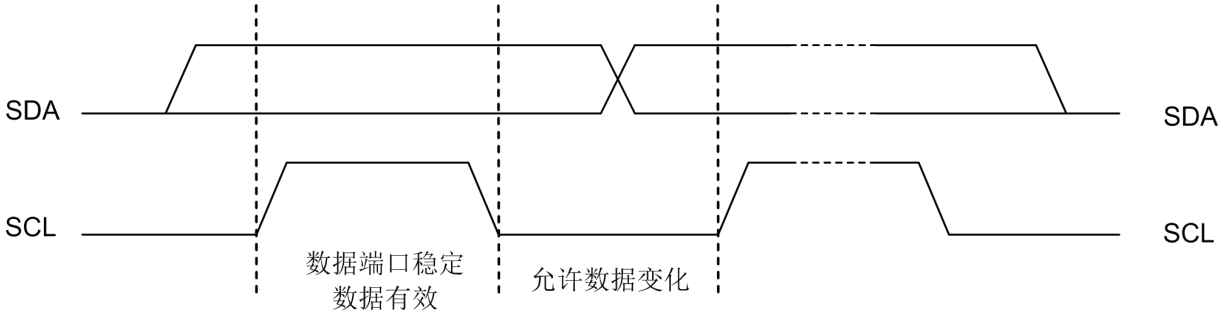
Stop 信号(P)

当 SCL 是高电平时，SDA 由低到高变化，表示结束数据传输。



8.2 数据有效

在 SCL 为高电平期间，SDA 必须保持稳定的电平。SDA 线上的高低电平变化只能在 SCL 为低电平期间。



8.3 字节格式

字节由 8 位数据和一个应答信号组成

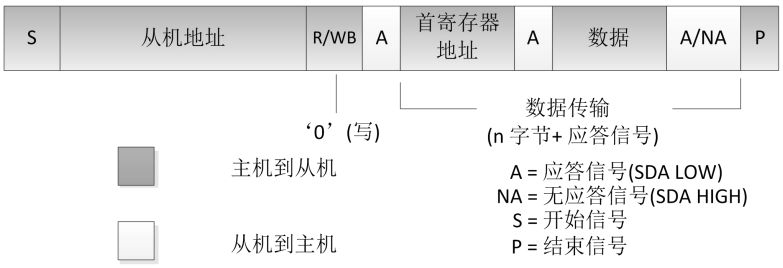
8.4 器件地址

地址 (A[6:0])	40H
读命令 (A[6:0]+RWB)	81H
写命令 (A[6:0]+RWB)	80H

8.5 操作模式

TC309 是从器件，支持读写两种操作模式：

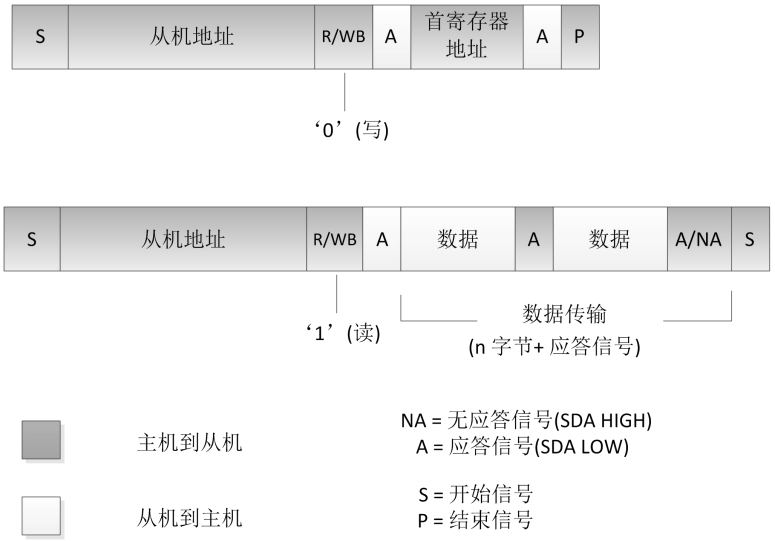
1) 写操作：
   - 首字节由 7 位从机地址和一位读写位组成（RWB=0）
   - 第二字节是要访问的内部寄存器地址
   - 下一个字节是要写入寄存器的内容
   - 继续写入下一个寄存器，直到 STOP 信号出现
   - 收到数据后 TC309 会发送应答信号



2) 读操作：

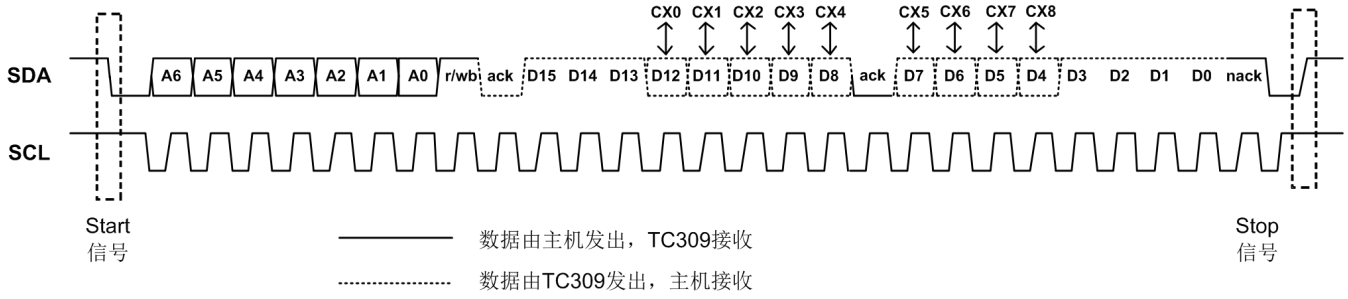
读操作的首寄存器地址由不含数据的写操作指定，由 STOP 信号结束。

然后主机送出开始信号，和器件地址和读取位(R/WB=1)，接下来的数据地址，是由首地址开始，然后地址依次加一。



3) 简化的读操作

TC309 的默认读寄存器地址为 08H。所以如果没有写过其它寄存器，就可以通过下面的时序直接读取按键信息。



8.6 TC309 控制寄存器列表

寄存器	地址（HEX）	读写	初始值（BIN）	寄存器功能描述							
				Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SenSet0	00H	W	0111 1001	SENSET0[7:0]							
SenSetCOM	01H	W	0111 1001	SENSETCOM[7:0]							
CTRL0	02H	W	100 0 0 0 11	SLPCYC[2:0]			SLPNOW	HOLD	KVF	RTM[1]	RTM[0]
Output1	08H	R	XXX1 1111				CH0	CH1	CH2	CH3	CH4
Output2	09H	R	1111 XXXX	CH5	CH6	CH7	CH8				
SAMPH	0AH	R	0000 0000	CHNUM[3:0]				SAMP[11:8]			
SAMPL	0BH	R	0000 0000	SAMP[7:0]							

8.6.1 灵敏度控制寄存器 SenSet0(地址 00H) SenSetCOM (地址 01H)

SENSET0[7:0] CX0 的灵敏度设置

SENSETCOM[7:0] 其余通道的灵敏度设置

共有 16 档灵敏度可以设置，由低到高为:【04H】【15H】【25H】【36H】【47H】【58H】【68H】

【79H】【8AH】【9BH】【ACH】【BCH】【CDH】【DEH】【EFH】【FFH】其中 79H 为初始值

CX0 单独设置灵敏度是可以把这个按键当做接近感应电极来用。如果用作普通按键，把 SENSET0[7:0]设成和 SENSETCOM[7:0]一样就可以了。

8.6.2 控制寄存器 CTRL0(地址 02H)

SLPCYC[2:0] 睡眠时，采样周期间隔设置

SLPCYC[2:0]	0	1	2	3	4（默认值）	5	6	7
采样间隔	无穷大	0.5T	1.5T	2.5T	3.5T	4.5T	5.5T	6.5T

T≈120ms

SLPNOW

SLPNOW	1	0（默认值）
	无按键马上进入睡眠	无按键 75S 进入睡眠

HOLD

HOLD	1	0（默认值）
	停止基准值校正	正常校正

KVF

KVF	1	0（默认值）
	按键后停止自校正	按键 50S 后开始自校正

RTM[1:0] 按键反应速度设置

RTM[1:0]	0	1	2	3（默认值）
按键有效判断	3 个采样周期	4 个采样周期	5 个采样周期	6 个采样周期
按键无效判断	1 个采样周期	2 个采样周期	3 个采样周期	4 个采样周期

8.6.3 按键信息寄存器 Output0 (地址 08H) Output0 (地址 09H)

CH[8:0] 分别对应 CX[8:0]的按键情况。无按键时为 1，有按键时为零。

8.6.4 采样值寄存器 SAMPH (地址 0AH) SAMPL (地址 0BH)

CHNUM[3:0] 采样值对应的通道

SAMP[11:0] 采样值

9. PCB 版图注意事项

1. VCC 和 VSS 电源线要单独走线，不能和其它芯片（单片机和 LCD 驱动芯片等）共用电源走线。以免使其它芯片的干扰信号通过电源线引到触摸芯片。

2. CP，CVREG，CSEN 三个电容必须靠近芯片放置。感应线上串联的 CX0~CX8 电阻，靠近芯片放置为宜。

3. 尽量大的铺地面积，可以提高抗干扰性。

4. 感应连线和感应焊盘优先布局。芯片靠近感应焊盘放置，感应连线直接引到感应焊盘（或弹簧焊盘），不同按键的感应连线不需要长度一致。感应连线线宽尽量小。感应连线周围不能走其他电源线和信号线。如果实在不能避免，其他走线要垂直跨过感应连线。感应焊盘之间至少留 5mm 间距，感应焊盘和铺地之间距离大于 1.5mm。

10. 额定值 *

工作温度	-40 ~ +85ºC
存储温度	-50 ~ +150ºC
电源电压	-0.3 ~ +6.5V
管脚最大电流	±20mA
管脚电压	-0.3V ~ (Vcc+ 0.3) Volts

* 注意 超出额定值可能会导致芯片永久损坏

11. 电气特性

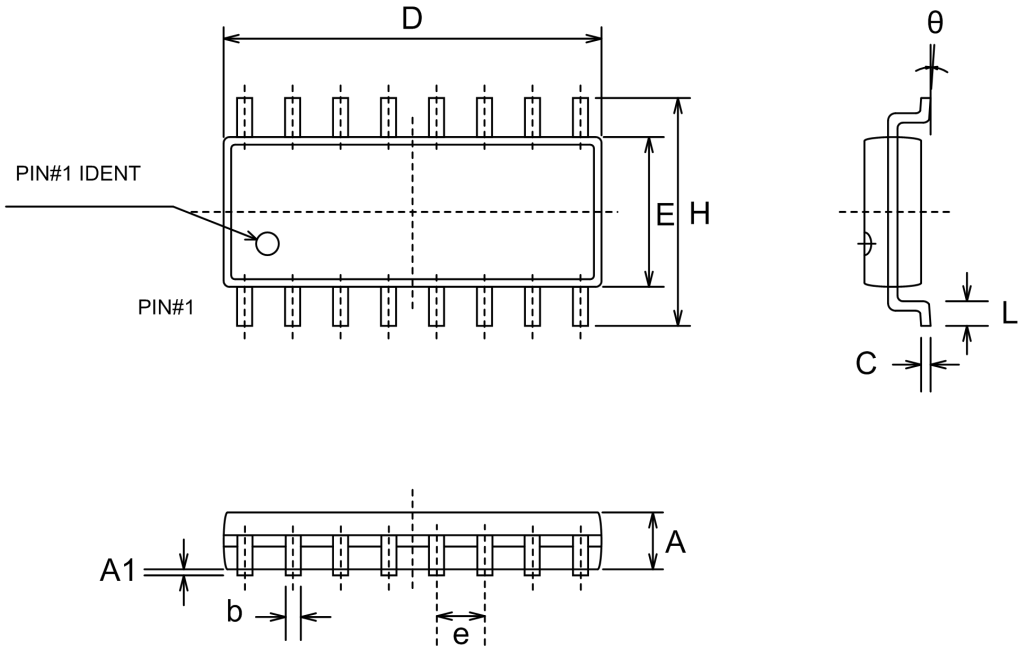
TA = 25℃

特性	符号	条件	最小值	典型值	最大值	单位
工作电压	Vcc		2.5		6.5	V
电流消耗	Idd	VCC=5.0V		1.15		mA
		VCC=3.0V		620		UA
		VCC=5.0V &SLEEP		40		UA
		VCC=3.0V &SLEEP		22		UA
上电稳定时间	Tini			300		ms
感应电容范围	CX				2.5*CSEN	
输出阻抗 (开漏输出)	Zo	低电平		50		Ohm
		高阻		100M		
输出灌电流	Isk	VCC=5V			10.0	mA
最小检测电容	delta_CX	CSEN=15pf		0.2		pF
采样周期	Tsi	正常工作状态		13		ms
进入空闲时间	Tidle	无按键		75		s

12. ESD 特性

模式	极性	最大值	参考
H.B.M	POS/NEG	8000V	VDD
		8000V	VSS
		8000V	P to P
M.M	POS/NEG	500V	VDD
		500V	VSS
		500V	P to P

13. 封装尺寸图 (SO-16)



Symbol	Dimensions In Millimeters			Dimensions In Inches		
	Min	Nom	Max	Min	Nom	Max
A	1.30	1.50	1.70	0.051	0.059	0.067
A1	0.06	0.16	0.26	0.002	0.006	0.010
b	0.30	0.40	0.55	0.012	0.016	0.022
C	0.15	0.25	0.35	0.006	0.010	0.014
D	9.70	10.00	10.30	0.382	0.394	0.406
E	3.75	3.95	4.15	.0148	0.156	0.163
e	--	1.27	--	--	0.050	--
H	5.70	6.00	6.30	0.224	0.236	0.248
L	0.45	0.65	0.85	0.018	0.026	0.033
θ	0°	--	8°	0°	--	8°

附录：通过 I2C 接口读写 TC309 的 C 语言演示程序

1) I2C 写控制寄存器函数

void i2c_write(unsigned dev_addr, unsigned reg_addr, unsigned char * src_buf, unsigned char len)

dev_addr 设置器件地址 TC309 固定唯一的器件地址是 0x40

reg_addr 设置寄存器地址

scr_buf 写入数据内容的首字节地址

len 写入数据内容的长度

举例：将数据 0x93 写入 TC309 的 CTRL0 寄存器(地址为 0x02)

```
char buffer;
buffer=0x93;
i2c_write(0x40, 2, &buffer, 1);
```

2) I2C 读取按键信息寄存器函数

void i2c_read_direct(unsigned dev_addr,unsigned char * dest_buf,unsigned char len)

dev_addr 设置器件地址 TC309 固定唯一的器件地址是 0x40

reg_addr 设置寄存器地址

dest_buf 读出数据的首字节地址

Len 读取数据的长度

举例：读取 TC309 按键信息 OUTPUT1 OUTPUT2 寄存器(地址为 0x08 0x09)

```
char buffer[2]={0,0};
i2c_write(0x40, 8, &buffer, 0);          //先设置读取寄存器地址为 0x08。如果上电后没有写过其它寄存器，这个步骤可以省略。因为芯片默认的读取地址就是 0x08
i2c_read_direct(0x40, &buffer, 2);    //读取 0x08 的内容放入 buffer[0]中,0x09 的内容放入 buffer[1]中。有按键时 buffer[2]相应位是 0，无按键时 buffer[2]相应位是 1
```

```
//*****************************************************************************
void i2c_delay()//简单延时函数
{
    char i;
    for(i=0;i<2;i++); }
//*****************************************************************************
void i2c_start()   //开始信号 SCL 在高电平期间，SDA 一个下降沿则表示启动信号
{
    SDA=1; //释放 SDA 总线
    i2c_delay();
    SCL=1;
    i2c_delay();
    SDA=0;
    i2c_delay();
    SCL=0;
    i2c_delay();
//  SDA=1;
}
//*****************************************************************************
void i2c_stop()    //停止 SCL 在高电平期间，SDA 一个上升沿则表示停止信号
{
//  SCL=0;
//    SDA=0;
//    i2c_delay();
    SCL=1;
//    i2c_delay();
//    SDA=1;
```

```
//      i2c_delay();
}
//*****************************************************************************
void i2c_checkack()   //应答  SCL 在高电平期间，SDA 被从设备拉为低电平表示应答
{
      bit   bit_temp;
      SCL=1;
      bit_temp=SDA;
/*    if(bit_temp)
            //无 ACK 回应
            ERR=0;
      else
            ERR=1;  */
      SCL=0;
//      i2c_delay();
}
void i2c_sendack()   //应答  SCL 在高电平期间，SDA 被从设备拉为低电平表示应答
{

      SDA=0;
      i2c_delay();
      SCL=1;
      i2c_delay();
      SCL=0;
      SDA=1;

}
void i2c_sendnack()   //应答  SCL 在高电平期间，SDA 被从设备拉为低电平表示应答
{

      SDA=1;
      i2c_delay();
      SCL=1;
      i2c_delay();
      SCL=0;

}
//*****************************************************************************
//*****************************************************************************
void i2c_write_byte(unsigned char date) //写一个字节
{
      unsigned char i,temp;
      temp=date;
```

```
    for(i=0;i<8;i++)
    {
        temp=temp<<1;
//          i2c_delay();
        SDA=CY;
        i2c_delay();
        SCL=1;//拉高 SCL，此时 SDA 上的数据稳定
        i2c_delay();
        SCL=0;//拉低 SCL，因为只有在时钟信号为低电平期间按数据线上的高低电平状态才允许变化；并在此时和上一个循环的 scl=1 一起形成一个上升沿
    }
//    SCL=0;//拉低 SCL，为下次数据传输做好准备
//    i2c_delay();
    SDA=1;//释放 SDA 总线，接下来由从设备控制，比如从设备接收完数据后，在 SCL 为高时，拉低 SDA 作为应答信号
    i2c_delay();
}
//*****************************************************************************
unsigned char i2c_read_byte()//读一个字节
{
    unsigned char i,k;

    for(i=0;i<8;i++)
    {
        i2c_delay();
        SCL=1;//上升沿时，IIC 设备将数据放在 sda 线上，并在高电平期间数据已经稳定，可以接收啦
//          i2c_delay();
        k=(k<<1)|SDA;
        SCL=0;
    }
    return k;
}
//*****************************************************************************
void i2c_write(unsigned dev_addr, unsigned reg_addr, unsigned char * src_buf, unsigned char len)
{
    char i;
    i2c_start();
    i2c_write_byte((dev_addr<<1));//发送发送从设备地址 写操作
    i2c_checkack();//等待从设备的响应
    i2c_write_byte(reg_addr);//发送发送寄存器地址 写操作
    i2c_checkack();//等待从设备的响应
    for(i=0;i<len;i++)
    {
```

```
            i2c_write_byte(src_buf[i]);//发送缓冲区数据 写操作
            i2c_checkack();//等待从设备的响应
        }
        i2c_stop();//停止
}
void i2c_read_direct(unsigned dev_addr,unsigned char * dest_buf,unsigned char len)
{
        char i;
        i2c_start();//启动
        i2c_write_byte((dev_addr<<1)+1);//发送发送从设备地址 读操作
        i2c_checkack();//等待从设备的响应
        dest_buf[0]=i2c_read_byte();//获取数据

        for(i=1;i<len;i++)
        {
            i2c_sendack();
            dest_buf[i]=i2c_read_byte();
        }
        i2c_sendnack();
        i2c_stop();//停止
}
```