2019.07

SC09B

9按键带自校正功能的容性触摸感应器

1. 概览

1.1 概述

SC09B 是带自校正的容性感应器，可以检测 9 个感应盘是否被触摸。它可以通过任何非导电介质（如玻璃和塑料）来感应电容变化。这种电容感应的开关可以应用在很多电子产品上，提高产品的附加值。

1.2 特征

◇ 9 个完全独立的触摸感应按键
◇ 保持自动校正，无需外部干预
◇ 按键输出经过完全消抖
◇ I^2C 串行接口
◇ 所有按键共用一个灵敏度电容
◇ 感应线长度不同不会导致灵敏度不同
◇ 2.5V ~ 6.0V 工作电压
◇ 符合 RoHS 指令的环保 SOP16 封装

1.3 应用

◇ 替代机械开关
◇ 家庭应用(电视机，显示器键盘)
◇ 玩具和互动游戏的人机接口
◇ 门禁按键
◇ 灯控开关
◇ 密封键盘面板

1.4 封装

SC09B采用SOP16封装

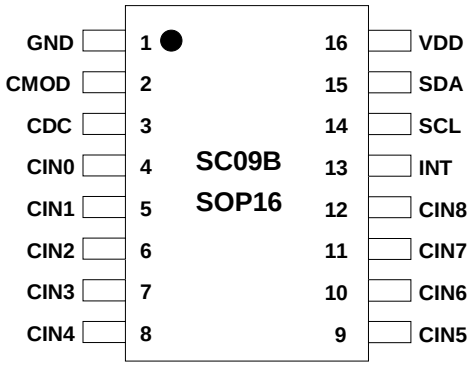


图1-1：封装简图

1.5 管脚列表

表1-1：管脚汇总

管脚编号	名称	类型	功能	不使用时
1	GND	**Pwr**	电源地	-
2	CMOD	**I/O**	接电荷收集电容	-
3	CDC	**I/O**	接灵敏度电容	-
4	CIN0	**I/O**	感应按键0检测输入	悬空
5	CIN1	**I/O**	感应按键1检测输入	悬空
6	CIN2	**I/O**	感应按键2检测输入	悬空
7	CIN3	**I/O**	感应按键3检测输入	悬空
8	CIN4	**I/O**	感应按键4检测输入	悬空
9	CIN5	**I/O**	感应按键5检测输入	悬空
10	CIN6	**I/O**	感应按键6检测输入	悬空
11	CIN7	**I/O**	感应按键7检测输入	悬空
12	CIN8	**I/O**	感应按键8检测输入	悬空
13	INT	**OD**	按键有效指示	悬空
14	SCL	**I**	I^2C 时钟输入	连接GND或VDD
15	SDA	**I/O**	I^2C 数据输入输出	连接GND或VDD 或者悬空
16	VDD	**Pwr**	电源正极	-

管脚类型　　I ：CMOS 输入　　I/O ：CMOS 输入/输出

O：CMOS 输出　　Pwr：电源 / 地

1.6 管脚说明

VDD, GND

电源正负输入端。

CMOD

电荷收集电容输入端，接固定值的电容，和灵敏度无关。

CDC

接灵敏度电容，电容范围是5pf ~100pf。根据使用环境选择合适的电容值，值越小，灵敏度越高。

CIN0~CIN8

接感应盘，是感应电容的输入检测端口。

INT

端口内部结构为NMOS开漏输出，输出高阻或低电平。有按键时输出低电平，无按键时输出高阻。

SCL, SDA

SCL 是I^2C时钟输入端口。SDA是I^2C数据输入输出端口。 SDA 端口有内部弱上拉。

2. 芯片功能

2.1 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。

2.2 灵敏度

灵敏度由CDC端口接的电容值决定。**电容范围是最小5pf，最大100pf。**数值越小，灵敏度越高。为了保证灵敏度的一致性，CDC电容要求使用10%或以上的精度的涤纶电容、NPO材质电容或者COG材质电容为最佳。务必在PCB布局时，将CDC电容尽量贴近IC放置。

2.3 自校正

根据外部环境温度和湿度等的漂移，按键电容基准参考值也会发生漂移，芯片会自动调整校正每个按键的电容基准参考值，以适应当前环境的变化。

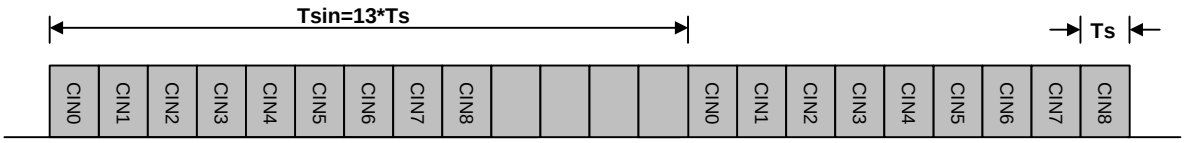
当检测到按键后，芯片会立即停止校正一段时间，这段时间大约 50 秒。停止校正时间一到，芯片会继续自校正，如果当前按键还是持续有效，按键信息会被当做环境的漂移立即被更新，也就是说检测按键有效的时间不会超过 50 秒。通过设置寄存器中的 KVF 位可以将按键修改为一直输出有效。

2.4 触摸反应时间

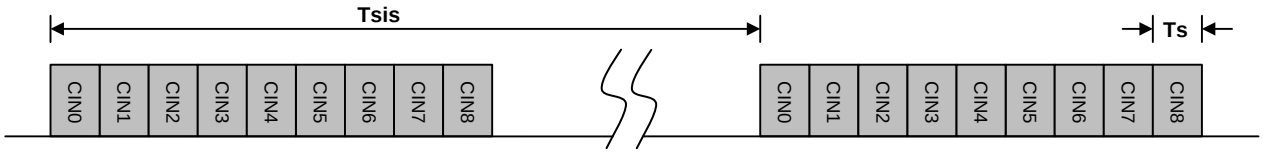
每个通道大约每隔12.5ms采样一次。经过按键消抖处理以后，检测到按键按下的反应时间大概是68毫秒，检测按键离开的反应时间大概是44毫秒。所以检测按键的最快频率大概是每秒9次。如果想要提高反应速度，可以设置内部寄存器，详情参考“**控制寄存器 CTRL0中的RTM[1:0]**”。

2.5 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，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。



正常模式下采样周期图示



睡眠模式下采样周期图示

图 2-1：正常和睡眠模式下采样周期图示

Ts：单个按键采样周期

Tsin：正常模式采样间隔

Tsis：睡眠模式采样间隔

Ts 大约是固定的950us左右。

正常模式下，采样间隔Tsin 是固定的大约12.5毫秒。

睡眠模式下，采样间隔Tsis通过**寄存器SLPCYC[2:0]** 配置，采样间隔越长，对应电流消耗Idds越低，但是唤醒的速度会相对变慢。

3. 应用

3.1 应用电路

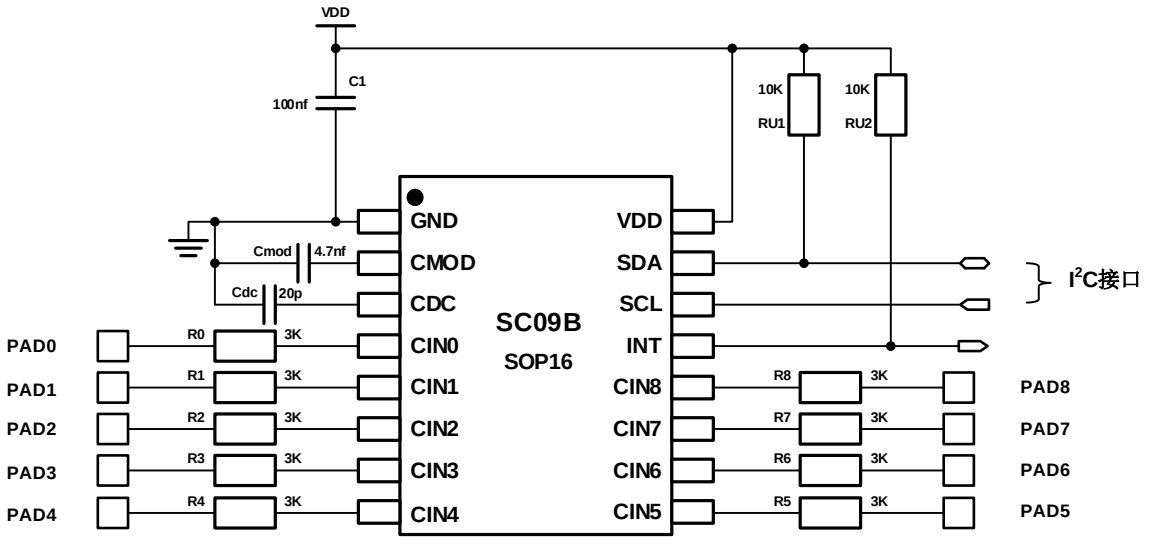


图3-1： 应用电路

注:

1. Cmod是电荷收集电容，取值范围是1nf~10nf。建议使用4.7nf。
2. Cdc 是灵敏度设置电容，取值范围是最小5pf，最大100pf，电容值越小灵敏度越高。

3.2 I²C 接口

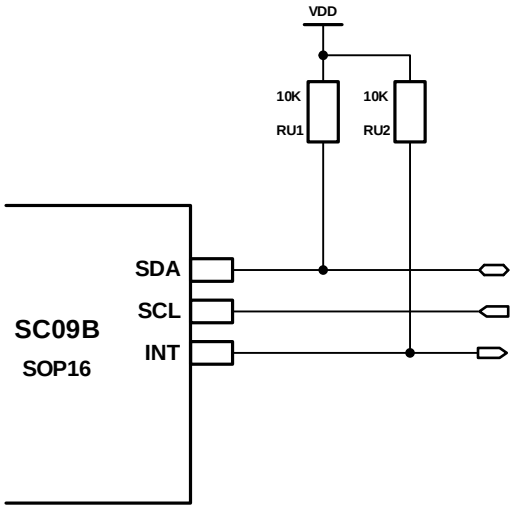


图3-2： I²C 中断方式

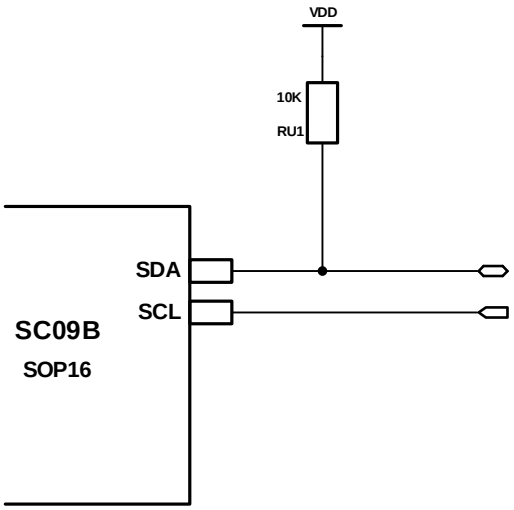


图3-3：I²C 查询方式

3.2.1 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。

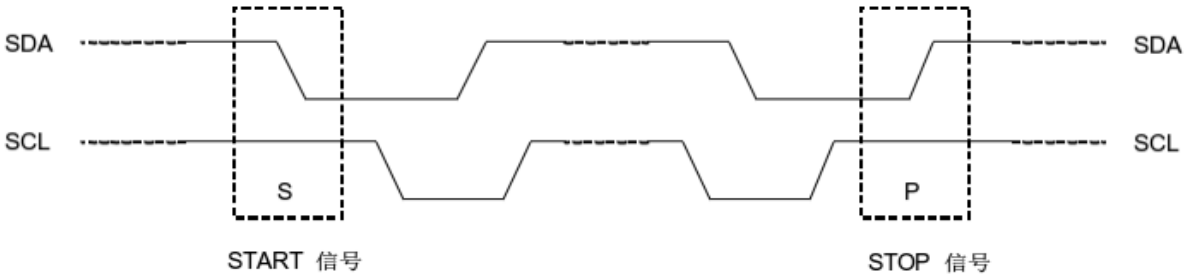


图3-4 ： Start和Stop信号

3.2.2 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。

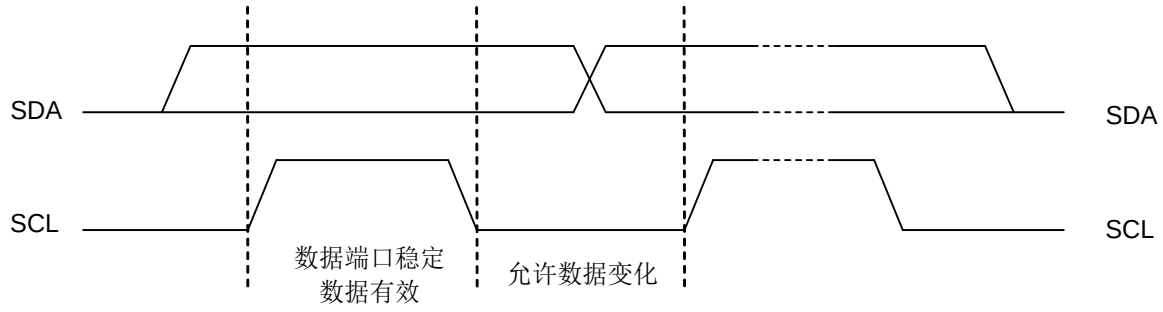


图 3-5 ： 有效数据

3.2.3 字节格式

字节由 8 位数据和一个应答信号组成

3.2.4 器件地址

SC09B 固定唯一的器件地址是 0x40。

表3-1：读写地址

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

3.2.5 操作模式

SC09B 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 接收到主机下达 STOP 信号出现
- 收到数据后 SC09B 会发送应答信号

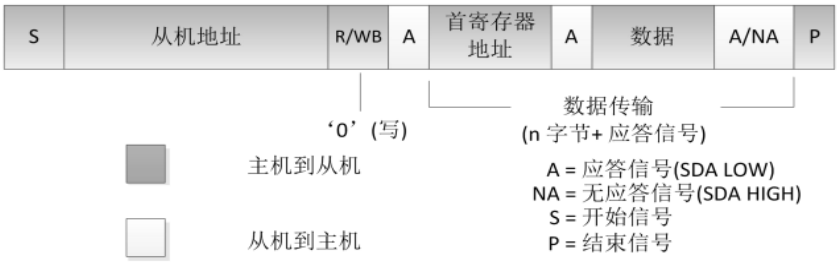


图 3-6 ： 写操作

（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。

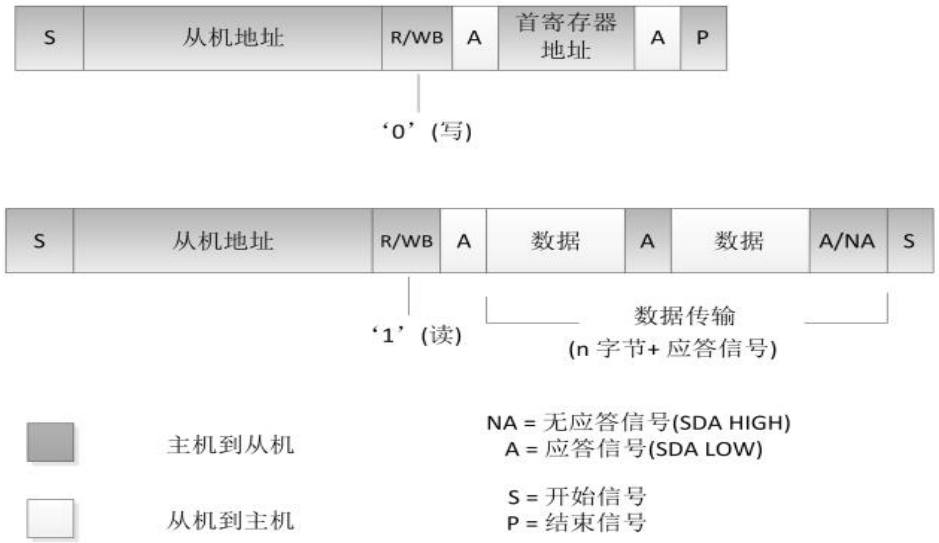


图 3-7 ： 读操作

(3) 简化的读操作

SC12B 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。寄存器 08H 的 D7~D5 和寄存器 09H 的 D3~D0 是固定低电平，寄存器 08H 的 D4~D0 与寄存器 09H 的 D7~D4 分别对应 CIN0~CIN8 是否有按键触摸。 例如，按键 CIN0 被触摸，寄存器 08H 的 D4 位将是高电平，如果 CIN0 没有被触摸，寄存器 08H 的 D4 位将是低电平。

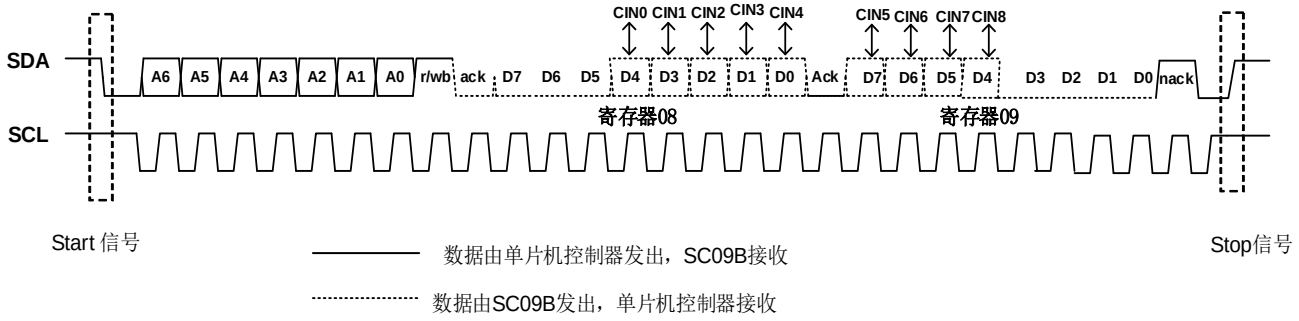


图 3-9： SC09B简化的 I2C 协议

3.2.6 操作模式

表 3-2：寄存器列表

寄存器	地址（HEX）	读写	默认值（BIN）	寄存器功能描述							
				Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SenSet0	00H	W	01111001	SENCH0[7:0]							
SenSetCOM	01H	W	01111001	SENCOM[7:0]							
CTRL0	02H	W	10000011	SLPCYC[2:0]			SLPNOW	HOLD	KVF	RTM[1]	RTM[0]
CTRL1	03H	W	00001000					CSEL3	CSEL2	CSES1	CSEL0
Output1	08H	R	00000000	0	0	0	CH0	CH1	CH2	CH3	CH4
Output2	09H	R	00000000	CH5	CH6	CH7	CH8	0	0	0	0
SAMPH	0AH	R	00000000	CS3	CS2	CS1	CS0	SAMP[11:8]			
SAMPL	0BH	R	00000000	SAMP[7:0]							

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

SENCH0[7:0] CIN0 的灵敏度设置

SENCOM[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。该寄存器涉及到手指触摸阈值及手指离开阈值，如无特殊运用，建议客户按照如上参数设置。

CIN0 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，或者隔空唤醒功能，如果用作普通按键，把SENCH0[7:0]设成和SENCOM[7:0]一样就可以了。

（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

（3） 控制寄存器 CTRL1(地址 03H)

CSEL3~CSEL0:内部基准通道电容的选择，默认值为 1000，对应的电容选择为8PF，该值一般用来修正外部通道的触摸感应量，该值可以设定范围0100~1111，对应值电容选择为4PF~15PF，如无特殊应用，建议设置默认值8PF。

（4）按键信息寄存器 Output0 (地址 08H) Output1 (地址 09H)

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

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

CS[3:0] 采样值对应的通道，采样时候对应是采样13个通道,而我们SC05B通道CIN0到CIN8对应是内部通道4到12。即当读取到CS值为4的时候，对应的SAMP值即为对应CIN0的采样值。

SAMP[11:0] 采样值

CS值	对应的通道	SAMP寄存器值
0	内部基准通道	0x0XXX
4	CIN0	0x4XXX
5	CIN1	0x5XXX
6	CIN2	0x6XXX
7	CIN3	0x7XXX

8	CIN4	0x8XXX
9	CIN5	0x9XXX
10	CIN6	0xAXXX
11	CIN7	0xBXXX
12	CIN8	0xCXXX

4. 详细参数

4.1 额定值 *

工作温度 .....-40 ~ +85ºC
存储温度.....-50 ~ +150ºC
最大Vdd电压.....-0.3 ~ +6.0V
管脚最大直流输出电流.....±10mA
管脚容限电压.....-0.3V ~ (Vdd + 0.3) Volts
* **注意:** 超出上述值可能导致芯片永久损坏

4.2 电气特性

表4-1: 电气参数 TA = 25℃

参数	符号	测试条件	最小值	典型值	最大值	单位
工作电压	Vdd		2.5		6.5	V
消耗电流 [1]	Idd	VDD=5.0V		1		mA
		VDD=3.0V		0.65		mA
		VDD=5.0V &SLEEP		30		uA
		VDD=3.3V &SLEEP		20		uA
芯片上电初始化时间	Tini			300		ms
感应管脚电容范围	Cin				2.5*Cdc²	
灵敏度电容	Cdc		5pf		100pf	
输出阻抗 （NMOS开漏）	Zo	delta Cin > 0.2pF		50		Ohm
		delta Cin < 0.2pF		100M		

输出灌电流	Isk	VDD=5V			10.0	mA
最小可检测电容	delta_Cin	CDC= 5pf		0.2		pF
I^2C 最大波特率	F_{br}	PullUp Res = 10K		400K		Bit/S
采样间隔时间	Tsin	Normal mode		12.5		ms

注： [1] **正常工作模式下与进入睡眠后的工作电流**

[2] **如果感应管脚寄生电容超过2.5倍的Cdc电容，芯片不能正常工作（绝大多数情况无需考虑这个限制）**

4.3 封装尺寸图 (SOP-16)

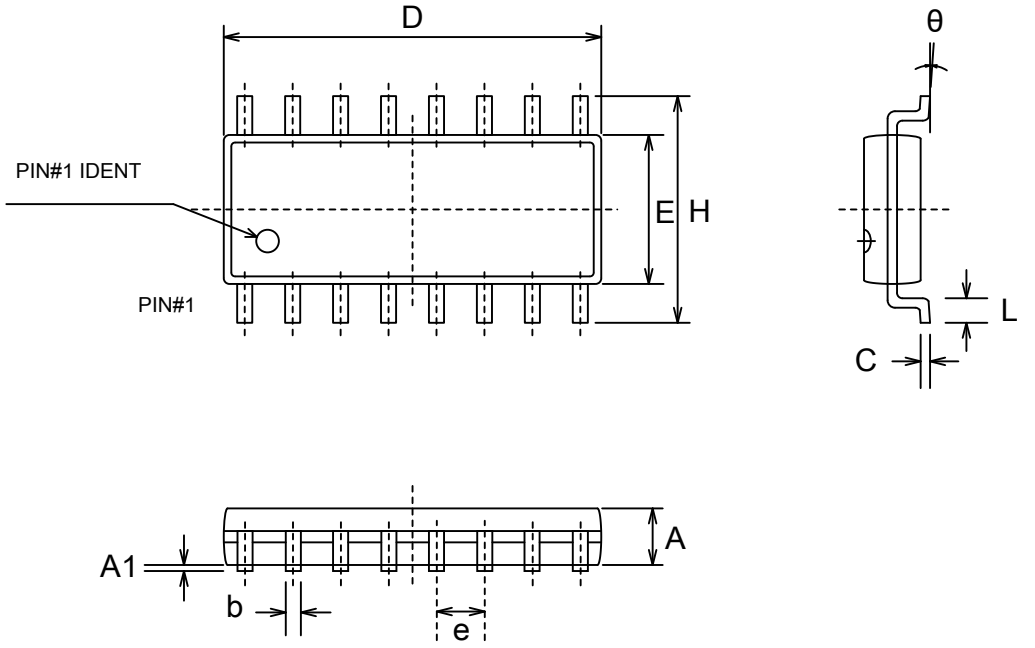


图 4-1： SOP16封装示例

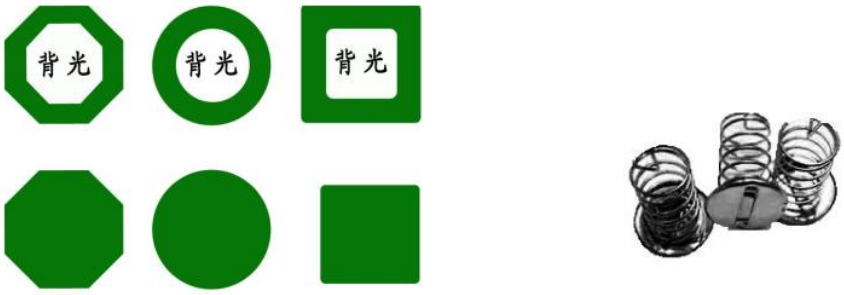
表4-2： 封装尺寸参数

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°

5. 电容按键传感器

5.1 触摸按键材料及形状

触摸按键可以是任何形状的导体，中间可以留孔或者镂空，但要保证一定的平面面积。建议使用直径大于12mm 的圆形或者方形，注意避免尖端效应。触摸感应盘可以用 PCB 铜箔、金属片、平顶圆柱弹簧、导电棉、导电油墨、导电橡胶、导电玻璃的 ITO 层等。图 5-1 所示：



按键感应盘可以是实心或中空的矩形、圆形，多边形

图 5-1： 不同形状按键感应盘示例

5.2 触摸感应盘的尺寸

触摸感应盘的尺寸大小：最小 4mmX4mm，最大 30mmX30mm。 实际面积大小根据灵敏度的需求而定，面积大小和灵敏度成正比。一般来说，按键感应盘的直径要大于面板厚度的 4 倍，并且增大电极的尺寸，可以提高信噪比。各个感应盘的形状、面积应该相同，以保证灵敏度一致。通常，在绝大多数应用里，12mmX12mm 是个典型值。

5.3 触摸触摸 PAD 和触摸面板的连接方式

（1）当用 PCB 的铜箔做触摸 PAD 时，直接将触摸 PAD 用两面胶粘在触摸面板上。
（2）使用带弹簧的贴片做触摸 PAD，必须将触摸 PAD 顶在面板上。
（3）使用导电橡胶或导电棉，导电橡胶或导电棉底端粘在 PCB 的铜箔上，顶端作为感应盘紧贴在面板上。
（4）导电油墨或 ITO 做成柔性 PCB，插在触摸端口的接口里。

5.4 触摸面板的选择

面板必须选用绝缘材料，可以是玻璃、聚苯乙烯、聚氯乙烯（pvc）、尼龙、树脂玻璃等。在生产过程中，要保持面板的材质和厚度不变，面板的表面喷涂必须使用绝缘的油漆。在触摸感应盘面积一定的情况下，面板的厚度和材质决定灵敏度。

通常面板厚度设置在 0~10MM 之间。不同的材料对应着不同的典型厚度， 按键感应盘表面要平整，且必须紧密贴在面板上，中间不能有空气间隙。

在实际应用的时候，客户根据实际需要，找到理想的折中值。下面的表格是 PAD 大小和不同材质面板厚度的推荐值。

表 5-1：PAD 大小与不同面板厚度的推荐值

PAD 直径（MM）	亚克力（介电常数 2.6~3.7）（MM）	树脂玻璃（介电常数 3.4）	ABS（介电常数 3.8~4.5)	云母片（介电常数 4~8）	普通玻璃（介电常数 7.6~8.0）
8	2.25	2.5	3	4.1	5
10	3.25	3.8	4.3	6.2	8
12	4.5	5.1	5.6	8	10
14	5.5	6	6.8	10	12.5

6. 电源

6.1 直流稳压器

SC 系列触摸芯片通过测量电容的微小变化反应触摸输出，因此要求电源的纹波和噪声要小，要注意避免由电源串入的外界强干扰。尤其应用于电磁炉、微波炉时，必须能有效隔离外部干扰及电压突变，因此要求电源有较高稳定度。建议采用如下图所示的 7805 组成的稳压电路。

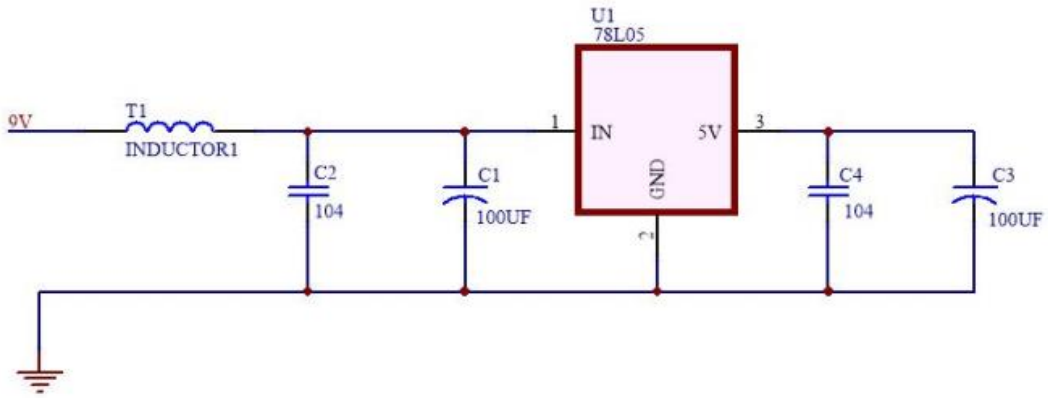


图 6-1： 7805 组成稳压电路

在 PCB 排版时，如果环境较恶劣，建议预留上图中电感 T1 焊盘，应对电磁炉等高噪声的干扰。在普通的应用中，可以不需要此电感。

6.2 稳压器件的放置

PCB LAYOUT 的时候，7805 电源组器件尽量靠近芯片的 VDD 和 GND 管脚。7805 电源组器件尽量与触摸芯片放在同一电路板上，并集中放置，杜绝电源连线过长带来噪声。

6.3 高噪声条件下的注意事项

在高噪声环境应用时，应避免高压(220V)、大电流、高频率操作的主板与触摸电路板上下重叠安置。如无法避免，应尽量远离高压大电流的器件区域或在主板上加屏蔽。

6.4 使用主机的 5V 电源

如果用户直接使用主机的 5V 电源，要接如下图的滤波电路，滤波电路中的 C3 电容和 C2 电容的放置规则和 6.2 相同。

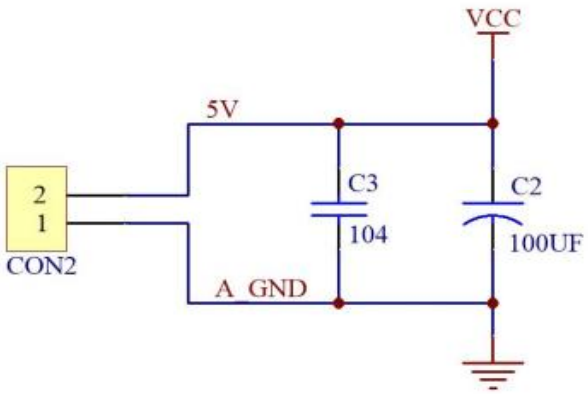


图 6-2： 5V 电源的滤波电路

7. 触摸感应电路 PCB 的设计

7.1 电源线的布线设计

触摸属于模拟敏感器件，同一系统的其他子单元的的电路要避免影响到触摸部分的电路，所以触摸电路部分的 VCC 电源线要单独走线，线长尽量短，走线要适当加粗。

7.2 地线的布线设计

触摸芯片的地线不要和其他电路共用，最好单独连到板子电源出入的接地点，也就是通常说的“星形接地”。电路的数字和模拟部分的电源和地分开用星型接法连接。

7.3 触摸应用电路外围元器件的布线设计

触摸芯片的退耦电容，CMOD 电容，CDC 电容及触摸限流电阻尽量要紧靠芯片放置，走线距离尽量短。

7.4 PAD 与 IC 的感应盘输入引脚之间的连线

触摸 IC 尽量要放在中心位置，尽量触摸 IC 到各个 PAD 之间的距离基本平衡。

PAD 输入端的走线，单面板走线建议是 8MIL~13MIL，双面板走线建议是 5~8mil。在工艺允许的情况下，建议越细越好。

PAD 输入端到触摸 IC 的连线不要跨越其他信号线。尤其不能跨越强干扰、高频的信号线。

PAD 输入端到触摸 IC 的连线周围 0.5MM 尽量不要走其他信号线。

7.4 铺地规则

触摸 IC 及其相关的外围电路要铺地，可以有效提高产品抗干扰能力。铺地的注意要点如下：

（1） 触摸 PAD 与铺地的距离推荐 1.5MM~2.0MM 之间，在这个距离区间内，可以有效平衡系统的抗干扰度和触摸的灵敏度。

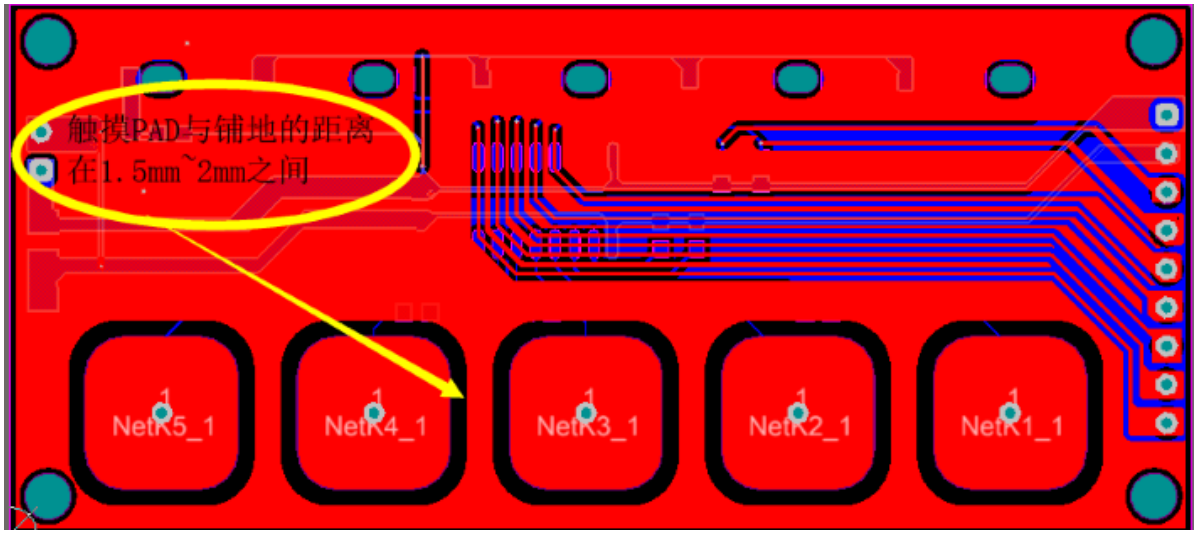


图 7-1：触摸 PAD 离铺地 1.5MM 以上

（2） 触摸 PAD 周围要铺地，触摸 PAD 正对反面的铺地要做镂空处理，减少寄生电容，改善灵敏度，且要尽量不要放置其他器件或者存在大面积铜箔，不走其他高频信号。

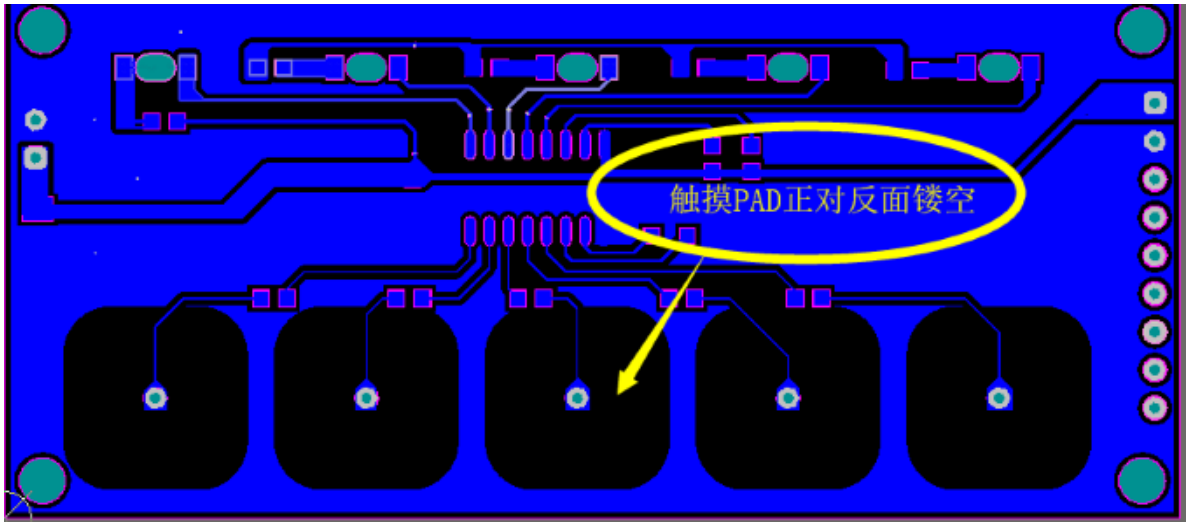


图 7-2：触摸 PAD 正背面镂空

（3）触摸信号线离铺地距离保持在 15mil 以上，且相邻触摸信号线之间也要尽量保持在 15mil 以上，避免产生串扰。 如下图所示：

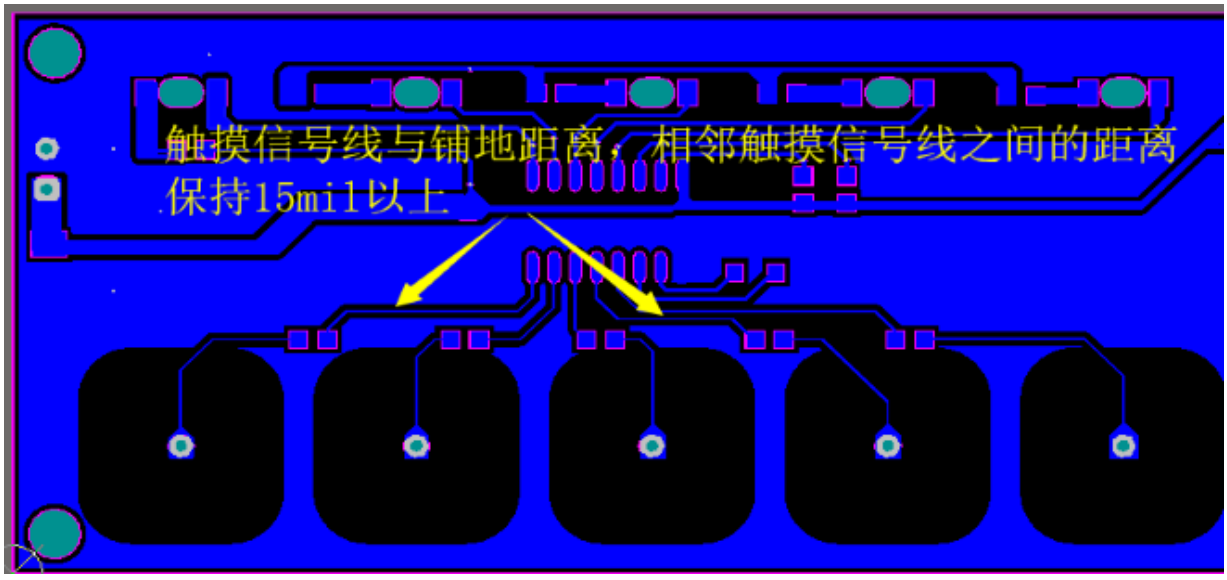


图 7-3：触摸信号线距离铺地及相邻触摸信号线保持 15mil 以上

（4）建议触摸 IC 及其相关的外围电路要用实铜铺地，增强芯片本身的抗干扰能力。

附录:

通过I²C接口读取SC09B的C语言演示程序

```
/*******************************************************************************
                            宏定义说明
*******************************************************************************/
//#define SPECIAL_APP                                   //需要特殊配置的运用
typedef enum //定义数据返回类型
{
   UNDONE = 0x00,
   DONE = 0x01
}Complete_Status;

#define SDA_OUT_OR_IN      TRISAbits.TRISA0          //定义 SDA 输入输出方向
#define SCL_OUT_OR_IN      TRISAbits.TRISA2          //定义 SCL 输入输出方向
#define SDA                LATAbits.LATA0            //定义 SDA 数据线输出
#define SDA_IN             PORTAbits.RA0             //定义 SDA 数据线读取
#define SCL                LATAbits.LATA2            //定义 SDA 数据线输出

#define SC09B_ADDR          0x40                     //SC09B 只有一个固定地址
//////////////////////////////////////////////////////////////////////////////Register ADDR//////////////////////////////////////////////////////////////////////////////
#define SenSet0_REG         0x00                     //CIN4 通道灵敏度的设置地址
#define SenSetCOM_REG       0x01                     //其他通道灵敏度的设置地址
#define CTRL0_REG           0x02                     //CTRL0 控制寄存器设置地址
#define CTRL1_REG           0x03                     //CTRL1 控制寄存器设置地址
#define Output_REG          0x08                     //触摸状态寄存器输出地址
#define SAMP_REG            0x0A                     //触摸数据值存器输出地址

#define RTM0                 0                       //3 个采样周期有效，1 个采样周期判断无效
#define RTM1                 1                       //4 个采样周期有效，2 个采样周期判断无效
#define RTM2                 2                       //5 个采样周期有效，3 个采样周期判断无效
#define RTM3                 3                       //6 个采样周期有效，4 个采样周期判断无效
#define KVF_STOP_CORREC     (1u<<2)                  // 按键有效，触摸不校准
#define KVF_50S_CORREC      (0u<<2)                  // 按下有效后，50S 开始校准
#define HOLD                (1u<<3)                  //基线保持不校准
#define NOTHOLD             (0u<<3)                  //基线持续校准
#define SLPCYC_LGT          (0u<<5)                  //无穷大
#define SLPCYC_0R5T         (1u<<5)                  //休眠后采样间隔 60MS
#define SLPCYC_1R5T         (2u<<5)                  //休眠后采样间隔 180MS
#define SLPCYC_2R5T         (3u<<5)                  //休眠后采样间隔 300MS
#define SLPCYC_3R5T         (4u<<5)                  //休眠后采样间隔 420MS
#define SLPCYC_4R5T         (5u<<5)                  //休眠后采样间隔 540MS
#define SLPCYC_5R5T         (6u<<5)                  //休眠后采样间隔 660MS
#define SLPCYC_6R5T         (7u<<5)                  //休眠后采样间隔 780MS
#define FAST_TO_SLEEP        (1u<<4)                 //快速进入休眠
#define SLOW_TO_SLEEP        (0u<<4)                 // 75S 进入休眠

/*******************************************************************************
                    * I2C 时钟延时函数
*******************************************************************************/
void Delay(unsigned char time)
{
      unsigned char a;
      for(a = time; a>0; a--);
}
/*******************************************************************************
                    * I2C 启动信号函数
*******************************************************************************/
void I2C_Start(void)
{
      SDA_OUT_OR_IN = 0;
```

```
        SCL_OUT_OR_IN = 0;
        SDA = 1;
        SCL = 1;
        Delay(1);
        SDA = 0;
        Delay(1);
        SCL = 0;
        Delay(1);
    }
/*****************************************************************************
            * 发送一个字节数据，并获取应答
*****************************************************************************/
    unsigned char SendByteAndGetNACK(unsigned char dataToSend)
    {
        unsigned char i;
        unsigned char ack;
        SDA_OUT_OR_IN = 0;
        for (i = 0; i < 8; i++) {
            SCL = 0;
            Delay(1);
            SDA = (dataToSend>>7) &0x01;
            Delay(1);
            SCL = 1;
            Delay(1);
            dataToSend <<= 1;
        }
        SCL = 0;
        Delay(3);
        SDA_OUT_OR_IN = 1;
        Delay(3);
        SCL = 1;
        Delay(1);
        i= 250;
        while(i--)
        {
            if(!SDA_IN){ SCL = 0; return 0;}
        }
        SCL = 0;
        return (1);
    }
/*****************************************************************************
            * 读取一个字节信号，并下发应答命令.
*****************************************************************************/
    void I2C_Respond(unsigned char ACKSignal)
    {
        SDA_OUT_OR_IN = 0;
        SDA = 0;
        SCL = 0;
        SDA = ACKSignal;
        Delay(1);
        SCL = 1;
        Delay(1);
        SCL = 0;
    }
/*****************************************************************************
                * 停止信号
*****************************************************************************/
    void I2C_Stop()
    {
        SCL = 0;
        SDA_OUT_OR_IN = 0;
```

```
        SDA = 0;
        Delay(1);
        SCL = 1;
        Delay(1);
        SDA = 1;
    }
/*****************************************************************************
                    * 读取一个字节函数
*****************************************************************************/
    unsigned char I2C_Receive8Bit(void)
    {
        unsigned char i,buffer;
        unsigned char ;
        SDA_OUT_OR_IN = 1;
        SCL = 0;
        for (i = 0; i < 8; i++)
        {
            Delay(1);
            SCL = 1;
            buffer = (buffer<<1)|SDA_IN;
            Delay(1);
            SCL = 0;
        }
        return (buffer);
    }
/*****************************************************************************
      * SC09B 初始化功能函数，如无特殊运用，无需初始化
*****************************************************************************/
    void SC09B_Init_Function(void)
    {
        unsigned char databuf;
        #ifdef SPECIAL_APP
        databuf = 0x79;
        I2C_Write_To_Device(SC05B_ADDR,SenSet0_REG,&databuf);
        databuf = 0x79;
        I2C_Write_To_Device(SC05B_ADDR,SenSetCOM_REG,&databuf);
        databuf = SLPCYC_3R5T | SLOW_TO_SLEEP | HOLD | KVF_50S_CORREC | RTM3;
        I2C_Write_To_Device(SC05B_ADDR,CTRL0_REG,&databuf);
        databuf =0b1000;
        I2C_Write_To_Device(SC05B_ADDR,CTRL1_REG,&databuf);
        #endif
    }
/*****************************************************************************************************************
        * SC09B 写寄存器参数运用函数
deviceAddr 设置器件地址    REG 设置寄存器地址   DAT8 写入数据内容的地址
*****************************************************************************************************************/
    Complete_Status I2C_Write_To_Device(unsigned char deviceAddr,unsigned char REG,unsigned char*DAT8)
     {
        I2C_Start();
        if (SendByteAndGetNACK((deviceAddr<<1) & ~0x01)) {
            I2C_Stop();
            return UNDONE;
         }
         if (SendByteAndGetNACK(REG)) {
            I2C_Stop();
            return UNDONE;
         }
         if (SendByteAndGetNACK(*DAT8)) {
            I2C_Stop();
            return UNDONE;
         }
```

```
        I2C_Stop();
        return DONE;
    }
/*****************************************************************************************************************
        * SC09B 简易读取按键值函数（默认直接读取）
        此函数只有初始化配置默认的情况下，直接调用，如果在操作前有写入或者其他读取不能调用默认
*****************************************************************************************************************/
    Complete_Status I2C_Simple_Read_From_Device(unsigned char deviceAddr,unsigned int* DAT16)
    {
        unsigned char buf1,buf2;
          I2C_Start();
        if (SendByteAndGetNACK((deviceAddr<<1) | 0x01)) {
          I2C_Stop();
          return UNDONE;
        }
        buf1 = I2C_Receive8Bit();
        I2C_Respond(0);
         Buf1 = I2C_Receive8Bit();
        I2C_Respond(1);
        I2C_Stop();
        *DAT16 = ((unsigned int)buf1 <<8)|buf2;
        return DONE;
    }
/*****************************************************************************************************************
    * SC09B 读取寄存器数值函数 （此函数主要是用来读取 SAMP 和 OutREG 值）
  deviceAddr 设置器件地址    REG 设置寄存器地址   DAT16 读取地址对应数据内容
*****************************************************************************************************************/
    Complete_Status I2C_Read_From_Device(unsigned char deviceAddr,unsigned char REG,unsigned int* DAT16)
    {
        unsigned char buf1,buf2;
        I2C_Start();
        if (SendByteAndGetNACK((deviceAddr<<1) & ~0x01)) {
          I2C_Stop();
          return UNDONE;
        }
        if (SendByteAndGetNACK(REG)) {
          I2C_Stop();
          return UNDONE;
        }
         I2C_Stop();
         I2C_Start();
        if (SendByteAndGetNACK((deviceAddr<<1) | 0x01)) {
          I2C_Stop();
          return UNDONE;
        }
        buf1 = I2C_Receive8Bit();
        I2C_Respond(0);
        buf2 = I2C_Receive8Bit();
        I2C_Respond(1);
        I2C_Stop();
        *DAT16 = ((unsigned int)buf1 <<8)|buf2;
        return DONE;
    }
```