CBM60xx
电容式触摸按键控制器

深圳芯邦科技股份有限公司
地址：广东省深圳市南山区科苑路科兴科学园B1栋16楼
邮编：518057
总机：(0755)86169690
邮箱：http://www.chipsbank.com
Email：info@chipsbank.com

产品概述

CBM60xx 系列芯片是芯邦科技推出的一款高性价比触控按键芯片，具有灵敏度调整方便、环境自校准、应用便捷等优点；满足家电、工业应用规格要求，可为用户在不同的应用环境中提供更可靠的性能。

CBM60xx 系列芯片主要包括 CBM6002，CBM6004C，CBM6008,如下列表所示：

型号	按键数目	通讯接口	封装	灵敏度调节方式
CBM6002	2 个	GPIO	SOP8	电容调整
CBM6002C	2 个	GPIO	SOP8	电容调整
CBM6004C	4 个	GPIO	SOP16	电容调整
CBM6008C	8 个	I2C	SOP16	电容调整/I2C 调整
CBM6008	8 个	I2C	SOP16	软件调整

产品特色

- 芯片 sensor 电容检测范围 1pF～40pF
- 触键灵敏度可通过外部校准电容灵活调整
- 支持 I2C Slave 标准通讯接口或 GPIO 通讯接口
- 支持 2 种工作模式 Normal/Idle，最低工作电流 500uA
- 面板溅水、溢水时触摸按键无异常反应，且能正常工作
- 最大支持 15mm 厚度的面板（与面板材料有关）
- Sensor 灵敏度自动校准功能，确保触摸按键灵敏度一致性
- 优良的抗电磁干扰能力
- 温度、湿度环境变化自适应调整
- EFT（4KV）、ESD（12KV）

典型应用

- 各种小家电设备
- 灯具开关控制
- 安防设备
- 工业智能控制仪器仪表
- 卫浴设备
- 音视频设备

工作电压

- 2.9V 到 5.5V

工作温度

- -40℃ 到 + 85℃

目录

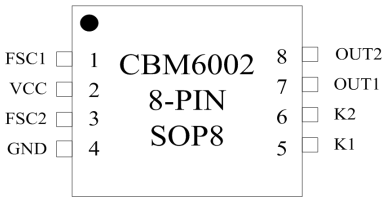
1. 介绍

CBM60XX 系列芯片包括 CBM6002、CBM6002C、CBM6004C、CBM6008C、CBM6008。其中 CBM6002、CBM6002C、CBM6004C、CBM6008C 无需烧录程序，通过外部校准电容或 I2C 命令调节灵敏度。而 CBM6008 支持软件调节灵敏度。CBM6008C、CBM6008 采用 I2C 标准接口；CBM6002、CBM6002C、CBM6004C 采用 GPIO 通讯接口，即一个按键对应一个 GPIO 输出。电容式触摸传感器可直接制作在 PCB 板上，外围器件少，系统总成本优于传统按键方案。

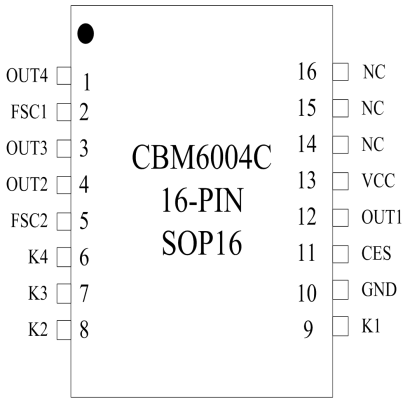
2. 封装

2.1. 封装形式

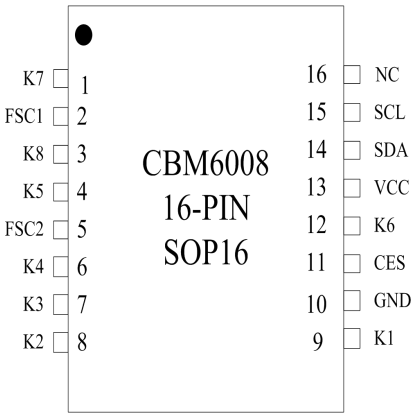
- **CBM6002/CBM6002C　SOP8 封装外观图:**



- **CBM6004C　SOP16 封装外观图:**



- **CBM6008C/CBM6008　SOP16 封装外观图:**



2.2. CBM6002/CBM6002C 引脚定义

序号	管脚名称	用法	功能描述
1	FSC1	PWR	接 4.7uF 滤波电容，增强抗 EFT、CS 等干扰
2	VCC	PWR	芯片电源输入 2.9V-5.5V
3	FSC2	PWR	接 4.7uF 滤波电容和 510Ω 稳压校准电阻，增强抗 EFT、CS 等干扰
4	GND	PWR	电源地
5	K1	I/O	按键感应通道 K1
6	K2	I/O	按键感应通道 K2
7	OUT1	I/O	按键信号输出 OUT1，高电平按键无效，低电平按键有效。CBM6002C 为 open drain PAD,需外接 4.7K 上拉电阻；CBM6002 无需外接上拉电阻，芯片内部有上拉电阻
8	OUT2	I/O	按键信号输出 OUT2，高电平按键无效，低电平按键有效。CBM6002C 为 open drain PAD,需外接 4.7K 上拉电阻；CBM6002 无需外接上拉电阻，芯片内部有上拉电阻

2.3. CBM6004C 引脚定义

序号	管脚名称	用法	功能描述
1	OUT4	I/O	按键信号输出 OUT4，默认高电平，低电平有效
2	FSC1	PWR	接 4.7uF 滤波电容，增强抗 EFT、CS 等干扰
3	OUT3	I/O	按键信号输出 OUT3，默认高电平，低电平有效

4	OUT2	I/O	按键信号输出 OUT2，默认高电平，低电平有效
5	FSC2	PWR	接 4.7uF 滤波电容和 510Ω稳压校准电阻，增强抗 EFT、CS 等干扰
6	K4	I/O	按键感应通道 K4
7	K3	I/O	按键感应通道 K3
8	K2	I/O	按键感应通道 K2
9	K1	I/O	按键感应通道 K1
10	GND	PWR	电源地
11	CES	I/O	灵敏度调整管脚
12	OUT1	I/O	按键信号输出 OUT1，默认高电平，低电平有效
13	VCC	PWR	芯片电源输入 2.9V-5.5V
14	NC		
15	NC		
16	NC		

2.4. CBM6008C/CBM6008 引脚定义

序号	管脚名称	用法	功能描述
1	K7	I/O	按键感应通道 K7
2	FSC1	PWR	接 4.7uF 滤波电容，增强抗 EFT、CS 等干扰
3	K8	I/O	按键感应通道 K8
4	K5	I/O	按键感应通道 K5
5	FSC2	PWR	接 4.7uF 滤波电容和 510Ω稳压校准电阻，增强抗 EFT、CS 等干扰
6	K4	I/O	按键感应通道 K4
7	K3	I/O	按键感应通道 K3
8	K2	I/O	按键感应通道 K2
9	K1	I/O	按键感应通道 K1
10	VSS	PWR	电源地
11	CES	I/O	CBM6008C 时，此脚为灵敏度调整管脚 CBM6008 时，此脚可以用来作为按键感应通道或 PWM 输出
12	K6	I/O	按键感应通道 K6
13	VCC	PWR	芯片电源输入 2.9V-5.5V
14	SDA	I/O	I2C 通讯接口，作为 I2C SDA 输入，外接上拉电阻
15	SCL	I/O	I2C 通讯接口，作为 I2C CLK 输入，外接上拉电阻
16	NC		

注意：

I: 输入管脚

O: 输出管脚
I/O: 输入输出管脚
PWR: 电源
NC： 未连接

3. 功能性能描述

3.1. 按键检测

在使用 CBM60xx 时，当手指靠近触摸感应盘时，该按键输入口的对地电容量会发生微小变化，当该值达到芯片的触发门槛时，判断为有按键动作，当该电容变化量被持续检测到超过 40ms，芯片认为按键有效，对应的输出端口会输出按键信息；当手指移开后，如果持续检测到非有效信号超过 50ms，认为按键无效，无按键信息输出。

按键检测最多支持同时有三个按键触发，当超过 3 个按键动作时，芯片将会输出无按键信息，并且超过 6 秒钟芯片会自校准。

3.2. 灵敏度调节

在使用 CBM60xx 时，灵敏度通过外部电容调整，电容值越大灵敏度越低，电容值越小灵敏度越高。调节电容的选取范围为 5pF-30pF，允许偏差≤5%，材质为 X7R/NPO。参考如下：

CES	5Pf	8pF	10 pF	15 pF	20 pF	27pF
灵敏度	最高	高	较高	适中	较低	低

在实际调试时可根据灵敏度情况逐次调整电容的大小，以达到最佳效果。

在使用 CBM6008C 时，客户还可以选择使用 I2C 命令调整按键灵敏度，详见 4.3.5， 4.3.6 说明。CBM6008 支持软件调整灵敏度。

3.3. 抗干扰性能

CBM60xx 内部集成了 CISC 处理器，采用独立的电容传感器输入通道检测触摸感应盘上的电容变化，通过高效的数据处理算法来保障识别手指的有效触摸，能够有效抵抗高低频噪音干扰和射频信号干扰、可以轻松对抗对讲机、GSM 手机在内的大部分射频信号对触键的干扰。

3.4. 防水性能

CBM60xx 具有自适应跟踪机制，当有水膜时可自适应调整按键的灵敏度，不会误触发，不影响正常使用，能够在溢水、溅水的面板环境下正常稳定工作。

3.5. 上电复位

当 CBM60xx 上电复位后，触控芯片首先需要进行初始化，包括 MCU 和模拟参数初始化，环境校准。在设备初始化完成之前，CBM60xx 将不会反馈按键信息。CBM60xx 上电初始化时间最大为 300ms，在初始化完成后即可响应有效的按键信息。

3.6. IDLE 模式

CBM60XX 中一项关键特性就是能够提供 IDLE 模式用于节省电源消耗。在触控系统中，当芯片上电 15 秒内无按键动作，芯片将自动进入低功耗模式，当有手触摸时会自动唤醒，全速运行。

4. 通讯接口

4.1. CBM6002/CBM6002C 通信接口

当 CBM6002/CBM6002C 检测到有效触发按键，按键所对应的 GPIO 口输出低电平，没有按键触发时默认为高电平，支持同时按下多个按键，对应关系如下：

按键	输出	按键有效	无按键（默认）
K1	Out1	低电平	高电平
K2	Out2	低电平	高电平

4.2. CBM6004C 通信接口

当 CBM6004C 检测到有效触发按键，按键所对应的 GPIO 口输出低电平，没有按键触发时默认为高电平，支持同时按下多个按键，对应关系如下：

按键	输出	按键有效	无按键（默认）
K1	Out1	低电平	高电平
K2	Out2	低电平	高电平
K3	Out3	低电平	高电平
K4	Out4	低电平	高电平

4.3. CBM6008C/CBM6008 通信接口

CBM6008C/CBM6008 支持 I2C slave 通信接口，具体特性如下：

- I2C slave clk: 20KHz - 100KHz
- I2C slave address: 0x22

4.3.1 I2C Slave 通讯接口

所有的地址包都是 9bit 长，包括 7bit 地址，1bit 读/写控制位和 1bit 应答位。当 CBM6008C/CBM6008 识别到 master 发送的地址与系统配置地址相同，将会在第 9 个 SCL 时钟周期通过下拉 SDA 应答（ACK）。所有的数据包也是 9bit 长，包括 8bit 数据和 1bit 应答位。如果接收方在第 9 个 SCL 时钟周期下拉 SDA，表示 ACK；如果接收方保持 SDA 为高，表示 NACK。I2C 每次读写周期必须通过 STOP 条件结束。

图4-1-1与图4-1-2表示I2C master读写I2C slave设备bit level波形示意图。当R/W bit被设置为0，I2C master可以写数据到地址有效的I2C slave设备。相反，当R/W bit被设置为1，I2C master可以从地址有效的I2C slave设备中读取数据。如果I2C slave设备地址验证失败，I2C slave将不会工作。

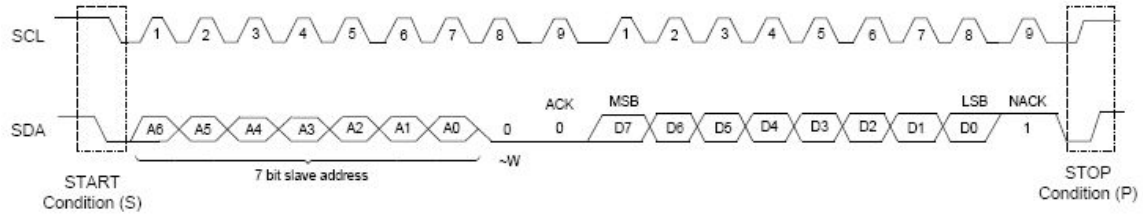


图4-1-1 I2C master写数据到I2C slave波形示意图（~W=0）

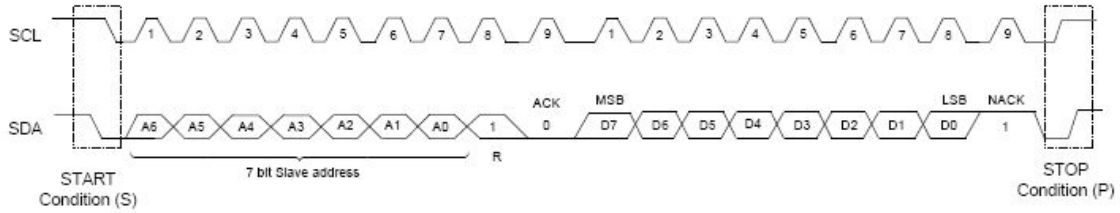


图4-1-2 I2C master读数据从I2C slave波形示意图（R=1）

CBM6008C/CBM6008 I2C通讯波形示意图如图4-1-1和图4-1-2 。CBM6008C/CBM6008控制器定义为slave，主机端定义为master，控制器设备地址定义为7bit 地址格式，默认值为0X22。如果有需要，可以通过修改I2C地址寄存器来改变设备地址，可配范围为0X00～0X7F。

图4-1-3显示了CBM6008C/CBM6008通过I2C slave通讯的系统框图。CBM6008C/CBM6008将触摸信息通过I2C实时上传给主机端，SCL与SDA需要接4.7K上拉电阻。主机端需要提供I2C serial clock（SCL）给CBM6008C/CBM6008。

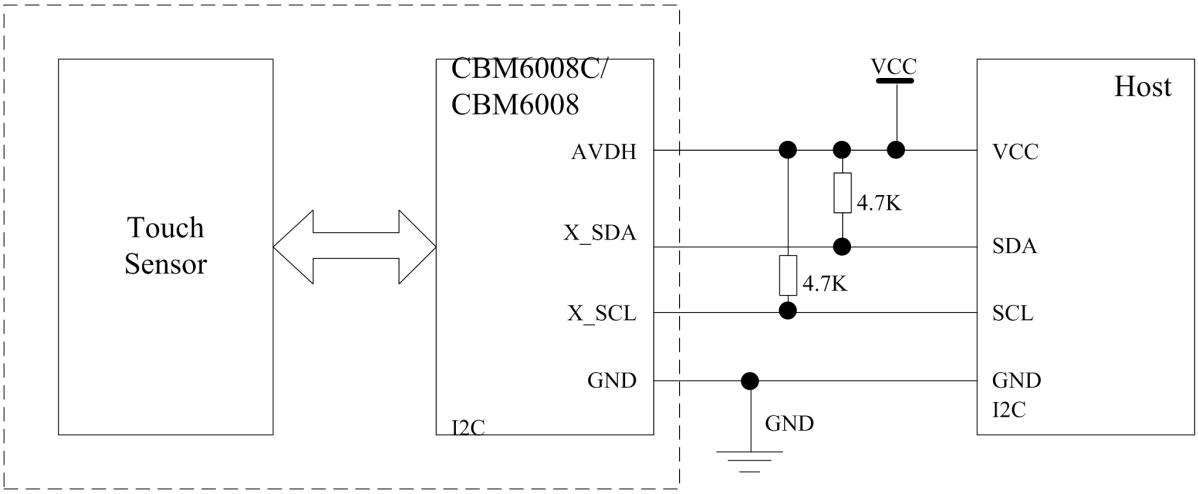


图 4-1-3 通过 I2C 通讯系统框图

包含有设备地址与读或者写位的地址包称为 Slave address+R 或者 Slave address+W。

写数据传输过程为：

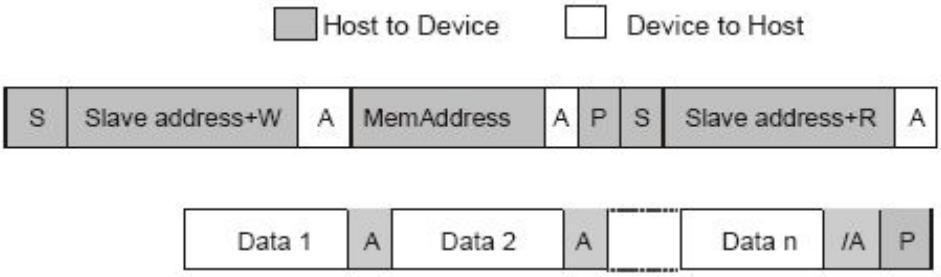
S	开始条件
Slave address+W	设备地址与写bit“0”
A	应答bit
Mem Address	CBM6008C/CBM6008内存地址
Data	写数据
P	停止条件

读数据传输过程有2种模式：

第一种为标准模式需要从机发送设备地址和存储地址；

第二种为快速模式主机只需要发送从机设备地址即可返回数据;

标准模式读数据格式：



快速模式读数据格式：

S	Slave address+R	A	Data 1	A	Data n	/A	P

S	开始条件
Slave address+W	设备地址与写bit“0”
A	应答bit
Mem Address	CBM6008C/CBM6008内存地址
Data	CBM6008C/CBM6008中数据
P	停止条件
Slave address+R	设备地址与读bit“1”
/A	没有应答，指示最后一个Byte数据发送完毕

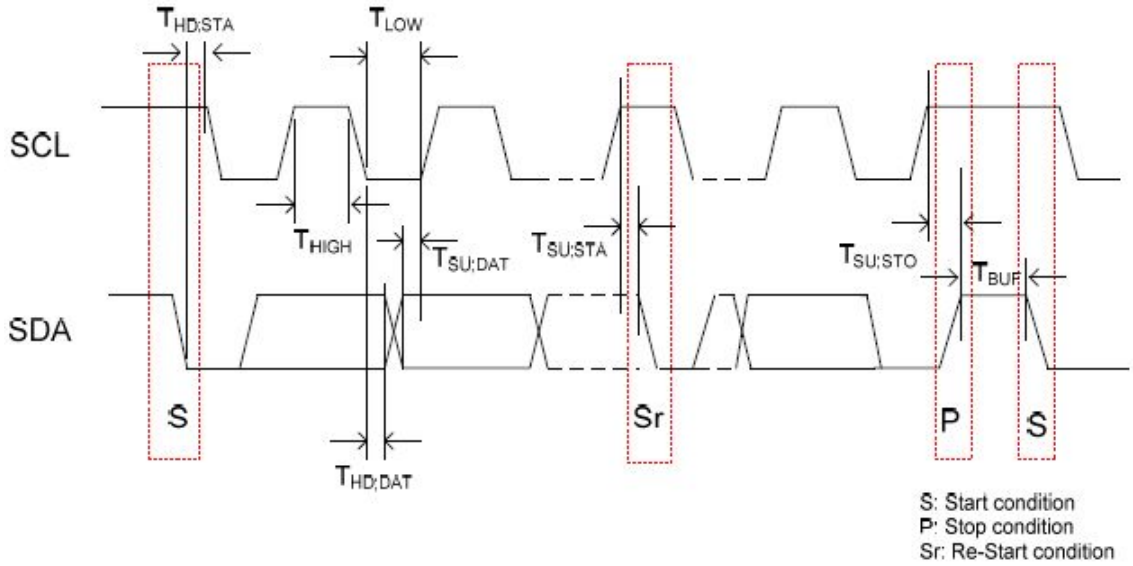


图4-1-4 I2C接口时序图

Table1: I2C SDA和SCL Pin脚特性

Symbol	Description	Standard Mode		Fast Mode		Units
		Min	Max	Min	Max	
F_{SCL}	SCL clock frequency	0	100	0	400	KHz
$T_{HD;STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	μs
T_{LOW}	LOW period of the SCL clock	4.7	-	1.3	-	μs
T_{HIGH}	High period of the SCL clock	4.0	-	0.6	-	μs
TSU;STA	Setup time for a repeated START condition	4.7	-	0.6	-	μs
$T_{HD;DAT}$	Data hold time	0	-	0	-	μs
$T_{SU;DAT}$	Data setup time	250	-	100	-	ns
$T_{SU;STO}$	Setup time for STOP condition	4.0	-	0.6	-	μs

T_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	μs

4.3.2 I2C 接口通信命令介绍

Address（HEX）	使用说明	访问模式	有效数据长度
0x01	芯片 ID	读	2byte 数据
0x0B	获取所有按键灵敏度参数	读	N byte 数据
0x21	设置单个按键灵敏度参数	写	2byte 数据
0x22	设置多个按键灵敏度参数	写	N byte 数据
0xA8	获取按键信息	读	1-3 byte 数据

4.3.3 读芯片 ID --0x01

Memory Address	Bytes	bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x01	Byte0	0xCB							
	Byte1	0x72							

功能：

读取芯片 ID，当主设备读取触控芯片 ID 时，需要传输正确的设备地址（设备地址默认为 0x22,若需改动可在量产软件中配置）。读到正确的 ID 后说明 I2C 主设备与从设备连接成功。

读数据长度：

2 bytes

备注：

0x72CB 即连续 2 字节 0xCB、0x72. 此数据包无校验码字节。

4.3.4 获取多个按键灵敏度 --0x0b

设备地址+W	Memaddr	设备地址+R	Data0	……	Data n
0x44	0x0b	0x45	Level PARM0	……	Level PARMn

说明：通过此命令读取所有按键灵敏度级数，**默认级数为 8**。

Level PARMn：每个按键灵敏度级数，级数范围 0---48 级。
n: 根据芯片的型号确定，CBM6008C 支持 8 个按键，n=7；
获取按键的灵敏度是得到全部按键的灵敏度级数，即包括有效按键和无效按键的灵敏度级数。

4.3.5 设置单个按键灵敏度 --0x21

设备地址+W	Memaddr	Data0	Data1
0x44	0x21	Key NO	Level PARM

说明：通过此命令可以调节单个按键灵敏度级数。
Key NO：按键序号.
CBM6008C 按键序号为 0---7，共 8 个按键。
Level PARM：灵敏度调节级数，级数范围 0---48 级。
当级数越大按键灵敏度越低，级数越小按键灵敏度越高。
如：调节第一个按键的灵敏度为 20 级时，参数为：Key NO=0x00, Level PARM=0x14。

4.3.6 设置多个按键灵敏度 --0x22

设备地址+W	Memaddr	Data0	…	Data n
0x44	0x22	Level PARM0	…	Level PARM n

说明：通过此命令可以调节多个按键灵敏度级数。
Level PARMn：每个按键灵敏度调节级数，级数范围 0---48 级。当级数越大按键灵敏度越低，级数越小按键灵敏度越高。
n: 根据芯片的型号确定，CBM6008C 支持 8 个按键，n=7；（无需烧录代码）。
Key0 对应的灵敏度级数为 Level PARM0，Key1 对应的灵敏度级数为 Level PARM1 … Keyn 对应的灵敏度级数为 Level PARMn。
没有使用按键的 Level PARM 设置为最大，即 48。已用的按键按实际状况设置。
例如：假设 CBM6008 使用的 5 个按键分别为 Key0,Key1,Key3,Key4,Key5，灵敏度级数为 17；而 Key2,Key6,Key7 未使用。此时配置参数为：

参数	Level PARM0	Level PARM1	Level PARM2	Level PARM3	Level PARM4	Level PARM5	Level PARM6	Level PARM7
级别	17	17	48	17	17	17	48	48

4.3.7 读按键信息 --0xA8

设备地址+W	Memaddr	设备地址+R	按键值		checksum
0x44	0xA8	0x47	Byte 0	Byte n	Byte n+1

说明：

1、默认按 bit 位返回键值信息。

2、bitX=1 表示第 X 个键被按下,当有多个键被按下时，将会有多个 bit 为 1 。
bitX=0 表示第 X 个键未按下或释放。

3、如下表格中 Key0，Key1 … Key7 表示对应的按键 0，按键 1 … 按键 7。

4、按键值通过查询命令方式获取按键值，查询周期大于 20ms。

5、各芯片按键返回值和对应的 Bit 关系，对应如下：

（1）CBM6008C 8 个按键返回的 bit 对应关系如下：

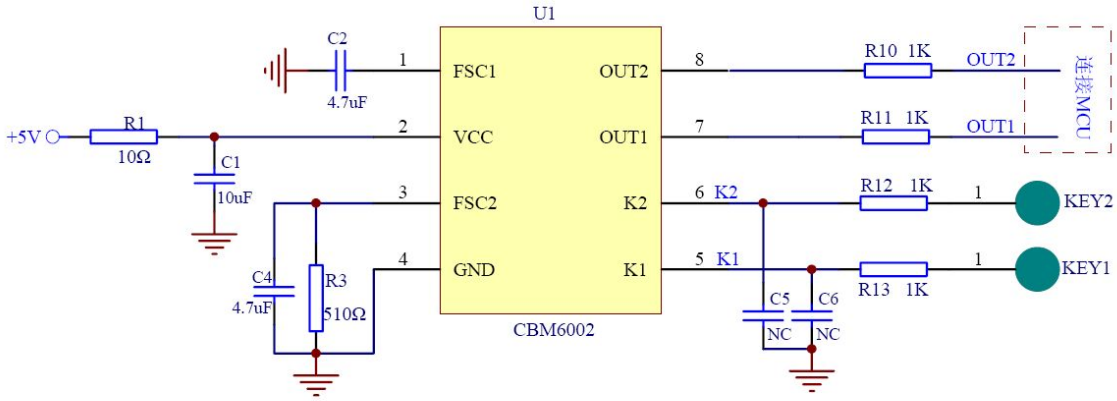
Memaddr	Bytes	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0xA8	Byte 0	Key7	Key6	Key5	Key4	Key3	Key2	Key1	Key0
	Byte 1	checksum							

checksum=Memaddr + Byte 0 的和的低 8 位。

5. 硬件设计指南

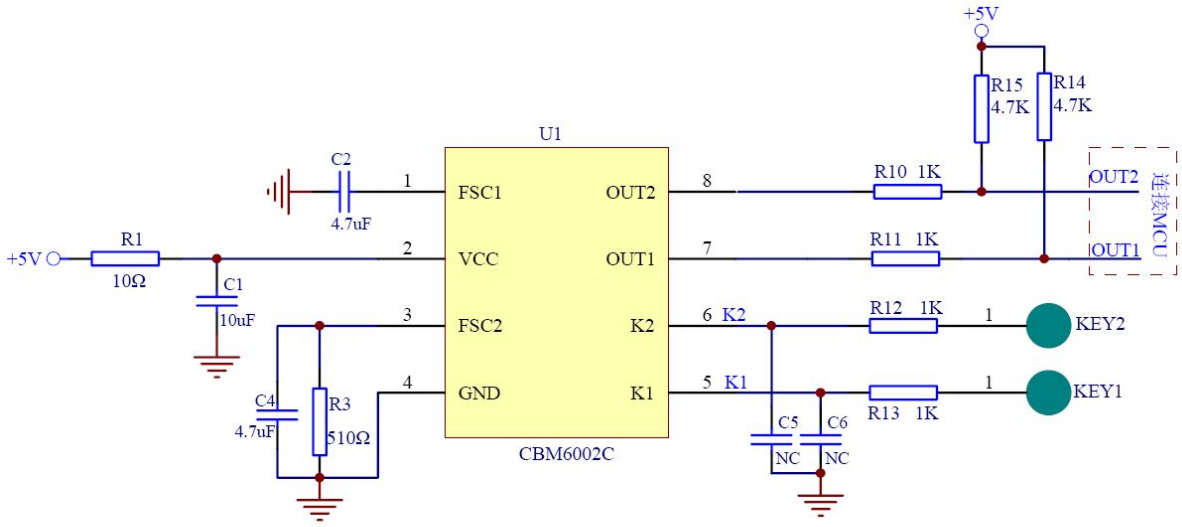
5.1 参考电路

(1) CBM6002 参考原理图



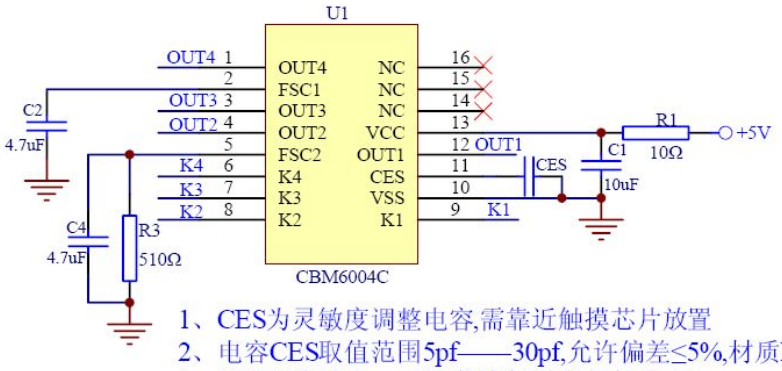
*根据按键个数自由选择感应通道，不用的悬空处理即可
*若测试环境恶劣，建议+5V增加100uF的电解电容
*C5,C6为灵敏度调整电容,需靠近触摸芯片放置,默认不焊接
*当灵敏度太高时，调整C5，C6值以达到最佳灵敏度。容值越大，灵敏度越低。
*灵敏度调整电容取值范围5pf——30pf,允许偏差≤5%,材质X7R/NPO

(2) CBM6002C 参考原理图

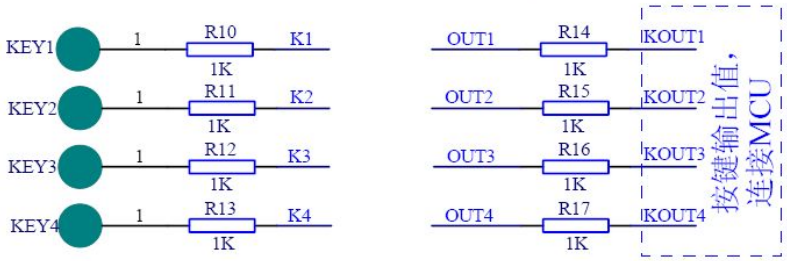


*根据按键个数自由选择感应通道，不用的悬空处理即可
*若测试环境恶劣，建议+5V增加100uF的电解电容
*C5,C6为灵敏度调整电容,需靠近触摸芯片放置,默认不焊接
*当灵敏度太高时，调整C5，C6值以达到最佳灵敏度。容值越大，灵敏度越低。
*灵敏度调整电容取值范围5pf——30pf,允许偏差≤5%,材质X7R/NPO

(3) CBM6004C 参考原理图

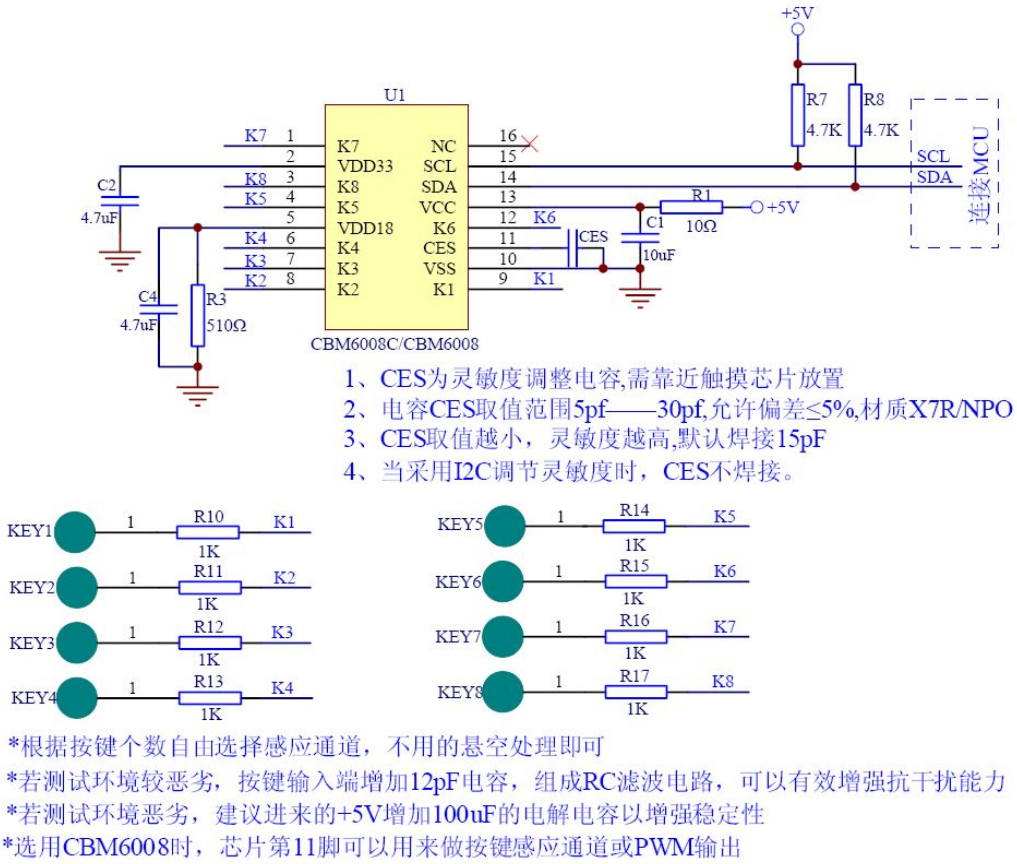


1、CES为灵敏度调整电容,需靠近触摸芯片放置
2、电容CES取值范围5pf——30pf,允许偏差≤5%,材质X7R/NPO
3、CES取值越小，灵敏度越高,默认焊接15pF



*根据按键个数自由选择感应通道，不用的悬空处理即可
*若测试环境恶劣，建议+5V增加100uF的电解电容
*若测试环境较恶劣，按键输入端增加12pF电容，组成RC滤波电路，可以有效增强抗干扰能力
*若测试环境较恶劣，按键输出端增加0.1uF电容，组成RC滤波电路，可以有效增强抗干扰能力

(4) CBM6008C/CBM6008 参考原理图



5.2 硬件设计注意事项

5.2.1 原理图设计注意事项

（1）芯片管脚 FSC1 和 FSC2 必须按照原理图接滤波电容和稳压电阻，以保证性能的稳定。

（2）按键上串接的电阻必须焊接，否则 EFT、ESD、CS 等抗干扰测试性能降低。

5.2.2 PCB 设计注意事项

(1)灵敏度调节电容的选取范围为 5pF-30pF,允许偏差≤5%,材质为 X7R/NPO。
(2)灵敏度调节电容尽量靠近芯片 CES 脚放置。
(3)按键走线要求 min=6mil，max=18mil，推荐 10mil。
(4)按键走线避免过孔和跳线进行连接。
(5)按键走线远离电源和高频信号 40mil 以上,越远越好。

(6)按键走线之间距离 40mil 以上，越宽越好。
(7)按键周围 80mil 内禁止铺地。
(8)制作在 PCB 上的按键，推荐圆形且直径为 12mm。

6. 电气特性

6.1 推荐操作条件

工作温度	-40℃ to +85℃
储存温度	-55℃ to +125℃
VCC 输入电压范围	2.9V to 5.5V
触摸按键电容检测范围	1pF to 40pF

6.2 直流参数

参数	描述	最小值	典型值	最大值	单位	注意
IDDR	工作时平均电流			20	mA	VCC = 5V OSC=20MHz
IDDS	IDLE 时平均电流			1	mA	VCC = 5V
VIL	输入低电平电压			0.3*VCC	V	2.9V < VCC <5.5V
VHL	输入高电平电压	0.7*VCC			V	2.9V < VCC <5.5V
VOL	输出低电平电压			0.3*VCC	V	
VOH	输出高电平电压	0.7*VCC			V	
AR	采样解析度	10	14	16	bits	

6.3 交流参数

参数	描述	最小值	典型值	最大值	单位	注意
TCLK	Internal oscillator	11.7	13	14.3	MHz	VCC = 5V
TVLCLK	Internal oscillator	36.1	32	41.2	KHz	VCC = 5V

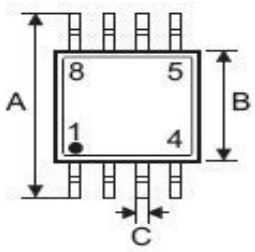
6.4 时间参数

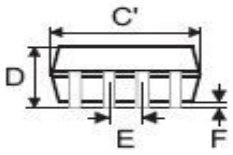
参数	描述	最小值	典型值	最大值	单位	注意
T_{EXTRST}	外部复位时间		200		us	

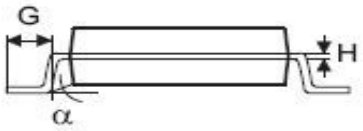
T_{POR}	开机启动时间		300		ms	

7. 封装外形图

7.1 CBM6002/CBM6002C SOP8 封装尺寸图

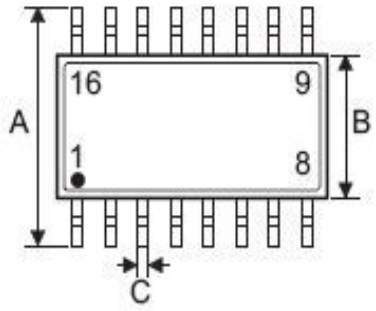


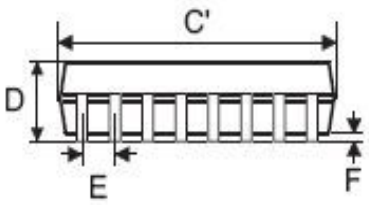


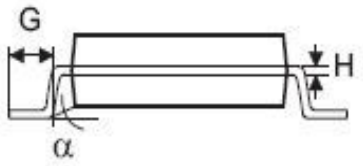


符号	说明	尺寸（mm）		
		最小	正常	最大
A	跨度	5.80	---	6.20
B	塑胶跨度	3.80	---	4.00
C	脚宽度	0.37	---	0.47
C’	塑胶体长	4.80	---	5.00
D	总高	1.35	---	1.70
E	脚间距	1.27		
F	站高	0.05	---	0.20
G	脚长	0.40	---	0.70
H	脚厚度	0.2		
a	脚角度	---	0	8

7.2 CBM6004C/CBM6008C/CBM6008 SOP16 封装尺寸图



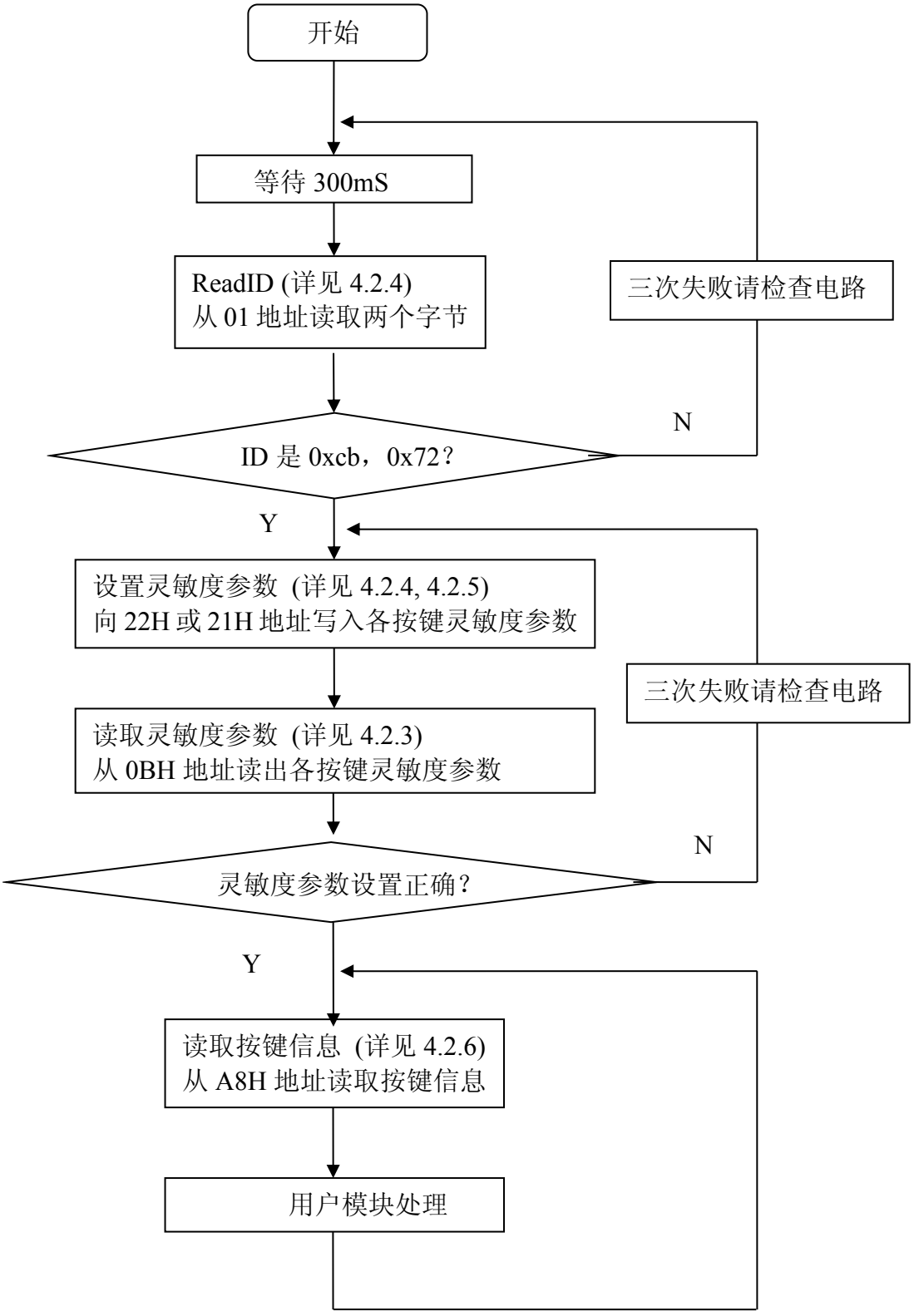




符号	说明	尺寸（mm）		
		最小	正常	最大
A	跨度	5.81	---	6.24
B	塑胶跨度	3.81	---	4.04
C	脚宽度	0.36	---	0.46
C’	塑胶体长	9.91	---	10.10
D	总高	1.40	---	1.70
E	脚间距	1.27		
F	站高	0.05	---	0.18
G	脚长	0.40	---	0.70
H	脚厚度	0.2		
a	脚角度	---	0	8

8. 附录一：

用户操作 CBM6008C 程序设计


开始
等待 300mS
ReadID (详见 4.2.4)
从 01 地址读取两个字节
三次失败请检查电路
ID 是 0xcb，0x72？
N
Y
设置灵敏度参数 (详见 4.2.4, 4.2.5)
向 22H 或 21H 地址写入各按键灵敏度参数
三次失败请检查电路
读取灵敏度参数 (详见 4.2.3)
从 0BH 地址读出各按键灵敏度参数
N
灵敏度参数设置正确？
Y
读取按键信息 (详见 4.2.6)
从 A8H 地址读取按键信息
用户模块处理

9. 附录二：

I2c 主程序（三星单片机）示例：

```
#define       SCL_L_1()       (BitClr(P4,BIT5)) //P4.5
#define       SCL_H_1()       (BitSet(P4,BIT5)) //P4.6
#define       SDA_L_1()       (BitClr(P4,BIT6))
#define       SDA_H_1()       (BitSet(P4,BIT6))
#define       SDA_TST_1       (BitTst(P4,BIT6))
#define       SDA_IN_1()       (P4CONH = 0x1a)
#define       SDA_OUT_1()     (P4CONH = 0x2a)

#define        get_ack          1
#define        send_ack         2
#define        send_nack        3

void delay_5us(void)
{
    NOP();NOP();NOP();NOP();NOP();
}
void init_i2c(void)
{
    //need init sck out
     SDA_OUT_1();
    SCL_H_1();
    delay_5us();
    SDA_H_1();
    delay_5us();
}
/*=====================================================================
name:          start_i2c
description:  i2c start status
input:         no
output:        no
=====================================================================*/
void start_i2c(void)
{
    SDA_H_1();
    delay_5us();
    SCL_H_1();
    delay_5us();
    SDA_L_1();
    delay_5us();
    SCL_L_1();
}
/*=====================================================================
name:          stop_i2c
```

```
description:  i2c stop status
input:        no
output:       no
=============================================================*/
void stop_i2c(void)
{
    SCL_L_1();
    delay_5us();
    SDA_L_1();
    delay_5us();
    SCL_H_1();
    delay_5us();
    SDA_H_1();
    delay_5us();
}
/*=============================================================
name:         ack_i2c
description:  i2c ack/nack status
input:        ack_mode
output:       temp
=============================================================*/
uchar ack_i2c(uchar ack_mode)              //应答
{
    uchar temp ,i;
    temp = 0;
    if (ack_mode == get_ack)
    {
        SCL_L_1();
        //delay_5us();
        SDA_IN_1();
        delay_5us();
        SCL_H_1();
        while((SDA_TST_1)&&(i<5))     i++;
        if (SDA_TST_1==1)       temp = 1;
        delay_5us();
        SCL_L_1();
        delay_5us();
    }
    else if (ack_mode == send_ack)
    {
        SCL_L_1();
        SDA_L_1();
        delay_5us();
        SCL_H_1();
        delay_5us();
    }
```

```
    else if (ack_mode == send_nack)
    {
        SCL_L_1();
        SDA_H_1();
        delay_5us();
        SCL_H_1();
        delay_5us();
    }
    SDA_OUT_1();
    return temp;
}
/*=====================================================================
name:                send_byte
description:  i2c sent 1 byte
input:               byte
output:              no
=====================================================================*/
void send_byte(uchar byte)              //发送字节
{
    uchar bit_count,data;
    data=byte;
    for(bit_count=0;bit_count<8;bit_count++)
    {
        SCL_L_1();
        SDA_OUT_1();
        //delay_5us();
        if(data&0x80)    SDA_H_1();
        else             SDA_L_1();
          data = data<<1;
        delay_5us();
        SCL_H_1();
        delay_5us();
    }
}
/*=====================================================================
name:          receive_byte
description:  i2c sent 1 byte
input:         no
output:        unsigned char retc
=====================================================================*/
uchar receive_byte(void)                //接收字节
{
    uchar retc;
    uchar bit_count;
    retc=0;
    SCL_L_1();
```

```
    //delay_5us();
    SDA_IN_1();
    delay_5us();
    for(bit_count=0;bit_count<8;bit_count++)
    {
        SCL_H_1();
        NOP();
        //delay_5us();
        retc<<=1;
        if (SDA_TST_1)      retc|=0x01;
        delay_5us();
        SCL_L_1();
        delay_5us();
    }
    SDA_H_1();
    SDA_OUT_1();
    //delay_5us();
    return(retc);
}
/*====================================================================
name:         read_ID
description:i2c read ID
input:        no
output:       unsigned char
====================================================================*/
uchar read_ID(void)
{
    WORD  r_id;
     r_key_buf1=0x00;                  //全局变量
    r_key_buf2=0x00;                   //全局变量      存放按键值
    uchar    device_addr =0x23;
    uchar    ret;
    start_i2c();
    send_byte(device_addr<<1);              //设备地址命令
    ack_i2c(get_ack);
    send_byte(1);                           //01:读 ID 的命令
    ack_i2c(get_ack);
        stop_i2c();
    start_i2c();
    send_byte((device_addr<<1)+1);          //设备地址命令+读使能
    ack_i2c(get_ack);

    r_id.byte.hi= receive_byte();           //第一个 byte=0xcb
    ack_i2c(send_ack);
    r_id.byte.lo = receive_byte();          //第二个 byte=0x72
```

```
    ack_i2c(send_nack);                    //最后一个 byte  返回 nack。
    stop_i2c();

     if(r_id.word==0xcb72)
        ret=0;
    else
        ret=1;

    return ret;
}
/*=============================================================
    name:        read_key
    description:i2c read key  （流程参考代码）
    input:       no
    output:
    note:
=============================================================*/
void read_key(void)
{
    uchar   buffer1,buffer2,buffer3;   //临时变量
    uchar   device_addr =0x23;         //设备地址
    uchar   memory_addr =0xa8;         //memory 地址
    r_key_buf1=0x00;                   //全局变量
    r_key_buf2=0x00;                   //全局变量      存放按键值
     reset_count=0x00;                 //全局变量      存放复位信息
    start_i2c();
    send_byte(device_addr<<1);         //发送设备地址
    if(1 == ack_i2c(get_ack))          //获取 ACK 信号，如是 NACK 则退出。
    {
        stop_i2c();
        return;
    }
   send_byte(memory_addr);             //发送 memory 地址
    ack_i2c(get_ack);
    stop_i2c();
    start_i2c();
    send_byte((device_addr<<1)+1);     //发送设备地址与读数据位
    ack_i2c(get_ack)

    buffer1= receive_byte();           //读取第一个 byte 数据
    ack_i2c(send_ack);
    buffer3 = receive_byte();          //读取第二个 byte 校验码数据
    ack_i2c(send_nack);                //最后一个 byte  返回 NACK。
    stop_i2c();
```

```
    if(buffer3==((buffer1+buffer2+0xa8)&0xff))   //校验数据包是否正确，如果正确则把数据赋给全局 buff，校验不正确或读数据异常全局 buff 为 0；
    {
        r_key_buf1=buffer1;
        r_key_buf2=buffer2;          //正常情况下将临时 buff 的键值赋给 key_buf1，做后续处理使用。
        reset_event=0;               //正常情况下此计数为 0；
    }
    return;
}
```

版本记录：

文档版本	更新日期	更新内容	修改人
V1.1	2013-12-1	增加功能项，初版	Touch Sensor Group
V1.1.1	2015-01-14	增加功能项，初版	Touch Sensor Group
V1.1.2	2015-01-14	增加功能项，初版	Touch Sensor Group
V1.1.3	2015-01-26	管脚定义更新	Touch Sensor Group
V1.1.4	2015-06-08	增加 CBM6002C,CBM6008C	Touch Sensor Group