永嘉微电
Vinka Microelectronics

VK3804 数据手册

触摸滑条I2C输出

Rev.1.1

知识产权说明

1 概述

此软件系提供给客户一个简易设定的滑条按键应用方案。客户只要使用IIC通讯格式，即可读取滑条按键及滑条地址的数据。

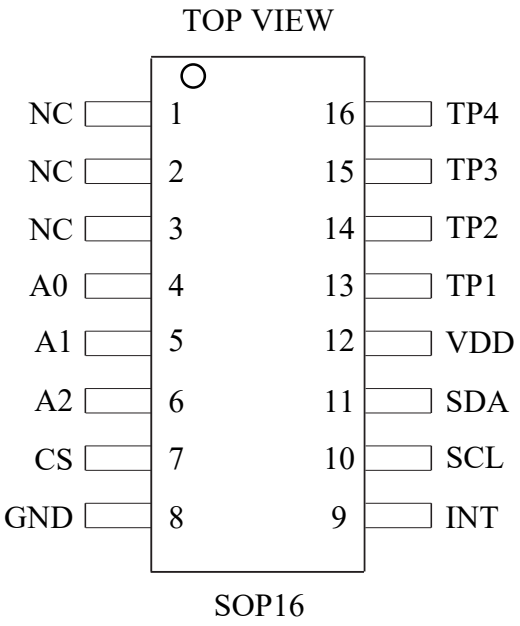
2 特点

- 此应用使用4个触摸键组合为一组滑条。
- 设计有省电模式。

3 应用领域

- 滑条触摸按键应用
- 门禁监控设备
- 消费类电子

4 管脚排列



有关详细信息，请参见 封装信息

4.1 VK3804/SOP16 管脚列表

脚位	管脚名称	输入/输出	功能描述
12	VDD	电源正	电源正
8	GND	电源地	电源地
7	CS	输入	灵敏度调节，接对地电容(1-100nF)
9	INT	输入	触摸状态输出，开漏输出需外接上拉电阻
10	SCL	输入	I2C 串行时钟脚，开漏输出需外接上拉电阻
11	SDA	输入/输出	I2C 串行数据输入/输出脚，开漏输出需外接上拉电阻
1-3	NC	—	悬空
4-6	A0-A3	输入	I2C从机地址选择脚
13-16	TP1-TP4	输入	触摸输入

表1. 管脚描述

5 电气特性

5.1 最大绝对额定值

参数	符号	条件	值	单位
工作温度	Top	——	-40~+85	℃
存放温度	T_{STG}	——	-50~+125	℃
电源电压	VDD	Ta=25℃	VSS-0.3~VSS+5.5	V
输入电压	V_{IN}	Ta=25℃	VSS-0.3~VDD+0.3	V
芯片抗静电强度HBM	ESD	——	>5	KV
备注：VSS代表系统接地				

5.2 DC/AC 特性：（测试条件为室温=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	-	5.5	V
系统震荡频率	F	VDD=5V	-	8M	-	HZ
工作电流	I_{OP}	工作，VDD=3V输出无负载	-	1.1	-	mA
	I_{OFF}	待机，VDD=3V输出无负载	5.3	6.8	10.0	uA

6 功能描述

触摸按键介绍：

触摸按键是利用测量人体接近导体时产生的电容变化，转换为数值判断的一种方式。此应用中所有的触摸按键都有Threshold 设定参数，用来调整触摸按键的灵敏度。

Threshold 依照按键的按压深度来做调整，数值越小越灵敏，但也越容易受到噪声干扰。

飞梭(滑条)介绍：

飞梭(滑条)按键的原理是利用在PCBLAYOUT上测得触摸的按压深度，来解析按压位置的一种方法。优点在可利用最少的按键解析出最多的按键地址。滑条图形主要分为环型跟直条两种应用，如下图:

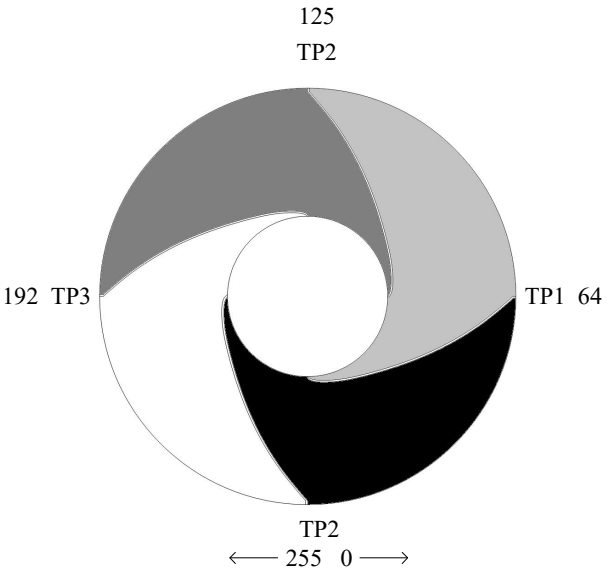


图1. 环形设计

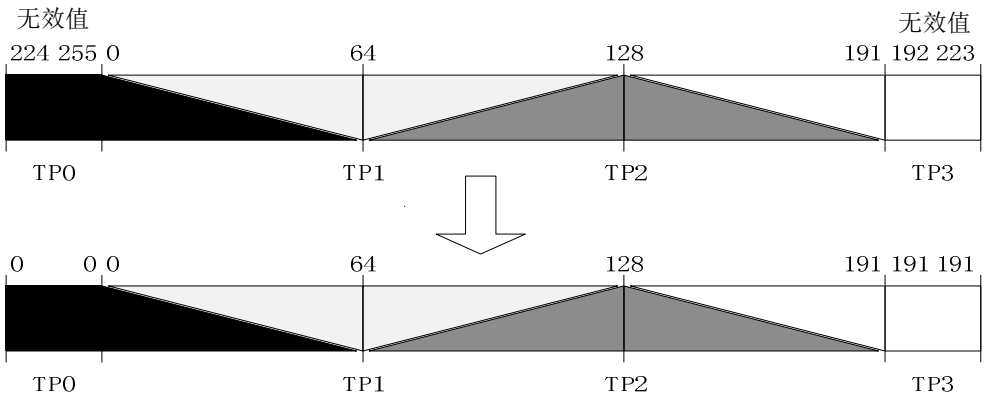


图2. 直条设计

注:直条设计中因为首尾不相连(非环形)，所以有效地址为0~191，若读取到值192~223或是224~255 则需要在主控端分别判断为是191或是0。

图3. 直条形设计

其原理是利用按压Touch Pad 时取得的数值变化，再使用内差法来计算其相对地址。因此需要最少3个按键，用以取得按压最深的按键与左右两边按键的差值来进行运算。

设计上建议按键与按键中心距离需小于30mm。齿与齿间的距离则约为0.4mm (如下图)，一般以3~4齿的设计为佳。

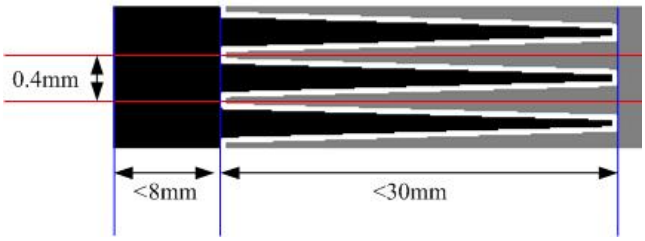


图4. Layout 设计要点

滑条按键为4Key（TP4~TP1)，如下表:

Slide	4Key
TP1	Slide _1
TP2	Slide _2
TP3	Slide _3
TP4	Slide _4

表2. 滑条按键定义

说明：滑条按键需要依照编号顺序排列，才能正确计算位置，禁止任意变换排列顺序。

7 IIC协议

IC使用IIC数据传输协议，两线式总线SCL、SDA来读写数据。INT脚位用来通知Master有按键状态变化。

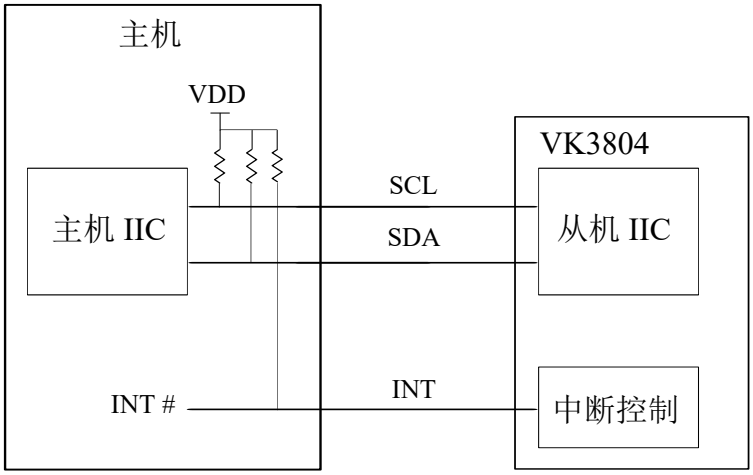


图5. 主机和VK3804通过IIC连接

INT在无按键状态变化时为高，当有按键时，INT脚位会拉低，无按键为高。

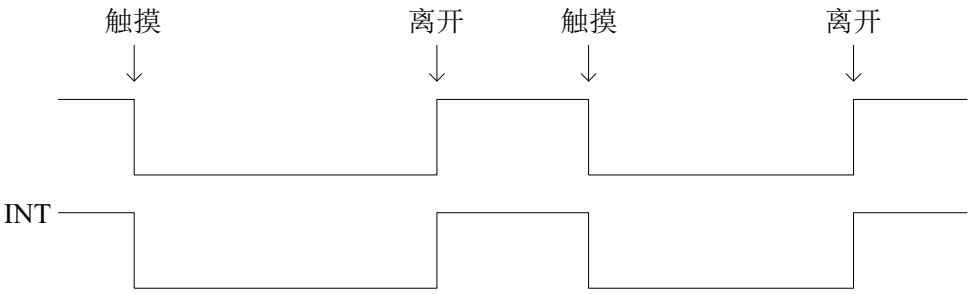
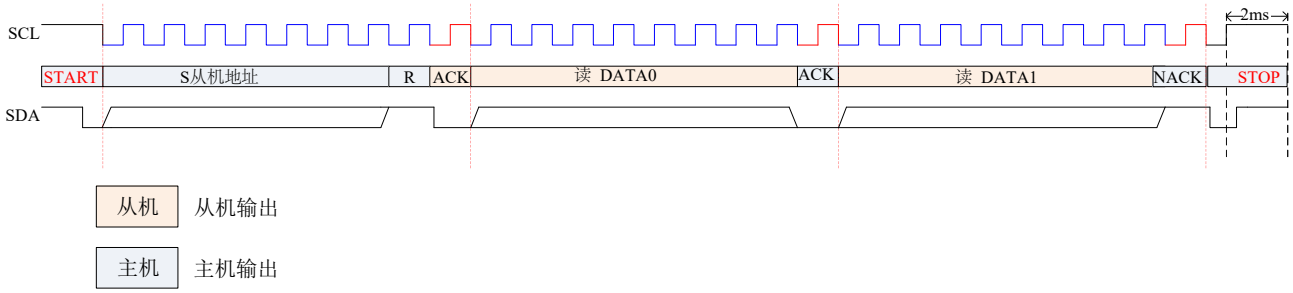


图6. INT 脚描述

7.1 数据流



7.2 从机地址

从机地址可通过 A2，A1 ,A0 脚选择，如下表：

A2	A1	A0	从机地址	读 (A6-A0,R)
0	0	0	50H	A1H
0	0	1	51H	A3H
0	1	0	52H	A5H
0	1	1	53H	A7H
1	0	0	54H	A9H
1	0	1	55H	ABH
1	1	0	56H	ADH
1	1	1	57H（默认）	AFH（默认）

说明：默认地址是A2/A1/A0脚悬空

7.3 读滑条位置

读数据	Bit 7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	C	0	0	WT	TP4	TP3	TP2	TP1
1	滑条位置(0~255)							

C	校准标志
0	正在校准
1	校准完成(默认)

WT	滑条按键标志
0	未触摸
1	触摸

TP4-TP1	触摸按键标志
0	无按键
1	有按键

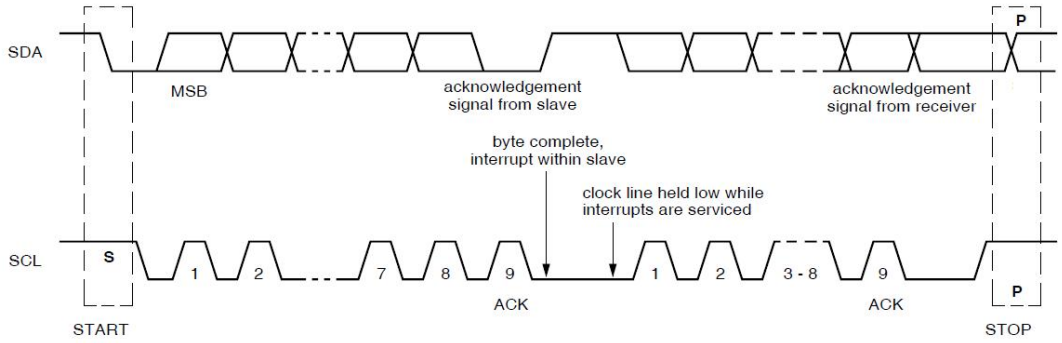
7.4 滑条位置

预设为 0，触摸滑条后输出按键位置，放开后保留最后按压位置。

Wheel Position								Position
7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	2	0	2
x	x	x	x	x	x	x	x	...
1	1	1	1	1	1	0	1	253
1	1	1	1	1	1	1	0	254
1	1	1	1	1	1	1	1	255

8 特别说明

1. VK3804的I2C接口有硬件的支持SCL可支持100KHz，但是译码为软件处理，处理约需20~100us视处理的情况而定，所以当Master发送完一组数据封包，应增加大于200us的延时，再发送下一组数据封包。



所以Master写程序时，需注意SCL拉Low的动作，若由硬件控制大多会支持此标准，若由程序控制IO脚，请增加对SCL输出High时要读回确认为High，才可让程序继续进行，若为Low应等待SCL为High后才可继续进行。C的程序如下：

```
SCL=1;
While(SCL!=1) { };
```

2. 若需要连续读取键值，建议读取完后暂停10ms以上，再读取下一次键值。否则会影响按键的反应速度。

3. 若开启睡眠模式，则禁止连续读取键值，因为每次读取键值时，都会清除进入睡眠的计时。，会导致系统无法睡眠。

4. 在系统进入睡眠模式时，会将IIC功能关闭。此时重新下IIC 指令可以唤醒系统，但是会收到no ACK的响应，需要等待系统唤醒后再重新下读写命令。

5. 按键阀值调整的步骤:

Step1.选择初始测试用的CS电容(见建议线路图):

建议使用10nF作为初始测试电容。

Step2.每个按键做按压测试:

以正常速度轻触按键或使用金属棒做测试条件，若在触摸到按键之前有按键输出，表示灵敏度太高，需要调高阀值，若触摸按键没有按键输出，或是要重压才有按键输出，表示灵敏度太低，需要降低阀值。

Step3.测试按键反应速度:

在判断按键灵敏度的时候，若觉得按键”不够灵敏”，需要进一步判断是按键响应速度不够快，还是按键灵敏度不够。判断方法是触摸停留一段时间(约1秒)，并检查是否有按键输出。若没有按键输出，则是按键不够灵敏，重新进行Step2调整，若有按键输出，则是按键响应速度不构快，则进行下一步。选择好适当的CS电容后需要回到Step2重新调整灵敏度。需要注意的是选择较小的CS电容，同时会降低滑条按键的精细度。

9 示范程序

```
/*
    项目名称:主控端对 VK3804 透过 IIC 控制的范例程序
    项目目的:1.透过软件模拟 IIC 主控端对 VK3804 写入设定参数
             2.透过软件模拟 IIC 主控端对 VK3804 读取按键状态
    主控 MCU:          C51
    Date & Version:    V1.0
*/
//-------------------------------------------------------------------------------------------
#include<reg51.h>
#include<intrins.h>
#define uint unsigned int
#define uchar unsigned char
#define address_W 0xa6 //从机的地址和写入标志
#define address_R 0xa7 //从机的地址和读取标志

sbit SINT=P0^0; //主控端与从机的 IIC 接口
sbit SDA=P0^1; //主控端与从机的 IIC 接口
sbit SCL=P0^2; //主控端与从机的 IIC 接口
uchar Write_Buffer[3]; //主控端的写入资料缓存
```

```
uchar Read_Buffer[3]; //主控端的读取资料缓存
//-------------------------------------------------------------------------------------------
//函数名称: void delay(uint x)
//函数功能: 程序延时
//函数输入: x
//函数输出: 无
//中间变量: i, j
//-------------------------------------------------------------------------------------------
void delay(uint x)
{
    uint i,j;
    for(i=x;i>0;i--)
        for(j=0x40;j>0;j--);
}
//-------------------------------------------------------------------------------------------
//函数名称: void sendStart()
//函数功能: IIC 的起始位
//函数输入: 无
//函数输出: 无
//中间变量: 无
//-------------------------------------------------------------------------------------------
void sendStart() //开始位
{
    SDA=1; /*发送起始条件的数据信号*/
    SCL=1;
    while(SCL!=1) { };
    SDA=0; /*发送起始信号*/
    _nop_();
    SCL=0; /*此位置只需要将 SCL 输出为 0 之后等待 4US 即可*/
```

```
}
//-------------------------------------------------------------------------------------------------
//函数名称: void sendStop()
//函数功能: IIC 的结束位
//函数输入: 无
//函数输出: 无
//中间变量: 无
//-------------------------------------------------------------------------------------------------
void sendStop() //停止位
{
    SCL=0;
    SDA=0; /*发送结束条件的数据信号*/
    _nop_();
    SCL=1;
    while(SCL!=1) { };
    _nop_();
    SDA=1;
} //-------------------------------------------------------------------------------------------------
-//函数名称: bit readACK()
//函数功能: 读取 IIC 的 acknowledge 标志位
//函数输入: 无
//函数输出: IIC 的 ACK 信号 返回 1 表示无acknowledge，0 表示acknowledge
//中间变量:
无 //-------------------------------------------------------------------------------------------------
bit readACK() //读取应答信号
{
SCL=0;
SDA=1; /*此处为释放 SDA 总线，由从从机发出低电平应答*/
_nop_();
SCL=1;
_nop_();
if(SDA)
```

```
            return 1; //no ACK
        else
            return 0; //ACK
}
//-------------------------------------------------------------------------------------------------
//函数名称: void sendACK()
//函数功能: 主控端送出应答信号
//函数输入: 无
//函数输出: 无
//中间变量: 无
//-------------------------------------------------------------------------------------------------
void sendACK() //输出应答信号
{
    SCL=0;
    SDA=0;
    _nop_();
    SCL=1;
}
//-------------------------------------------------------------------------------------------------
//函数名称: void sendNOACK()
//函数功能: 主控端送出无应答信号
//函数输入: 无
//函数输出: 无
//中间变量: 无
//-------------------------------------------------------------------------------------------------
void sendNOACK() //输出无应答信号
{
    SCL=0;
    SDA=1;
```

```
    _nop_();
    SCL=1;
}
//-------------------------------------------------------------------------------------------------
//函数名称: void sendByte(uchar dat)
//函数功能: 主控端写一个字节到从机
//函数输入: dat = 发送的字节
//函数输出: 无
//中间变量: i
//-------------------------------------------------------------------------------------------------
void sendByte(uchar dat) //写一个字节
{
    uchar i;
    for(i=0;i<8;i++)
    {
        SCL=0; /*钳住 I2C 总线，准备发送数据 */
        if(dat&0x80)
            SDA=1;
        else
    SDA=0;
_nop_(); /*如果需要在 SDA,SCL,INT 上串接电阻，根据电阻大小不同，电阻
越大建议将该时间适当加长，100KHZ 以内即可； */
    nop_();
SCL=1; /*此处由于 51 单片机的特性不需要做输入输出设置，
但如果是其他单片机需要先将其 IO 口改为输入上拉的设置，读到高之
后，SCL 转为输出为高。在读写完 ACK 后的第一个 clock 下降缘从机会钳住
SCL 脚做资料处理，
所以将 SCL 脚置为输入上拉，并等待 SCL 被释放。*/
```

```
        while(SCL!=1) { };
        dat<<=1;
      }
}
//--------------------------------------------------------------------------------------------
//函数名称: uchar readByte()
//函数功能: 主控端对从机读取一个字节
//函数输入: 无
//函数输出: 读取完成的字节
//中间变量: i, dat
//--------------------------------------------------------------------------------------------
uchar readByte() //读一个字节
{
      uchar i, dat=0;
      for(i=0;i<8;i++)
      {
         SCL=0;
         SDA=1;
_nop_(); /*如果需要在 SDA,SCL,INT 上串接电阻，根据电阻大小不同，电
阻越大建议将该

时间适当加长，100KHZ 以内即可；*/
            dat<<=1;
SCL=1; /*此处由于 51 单片机的特性不需要做输入输出设置，但如果是其他单
片机需要先将其 IO 口改为输入上拉的设置，读到高后，SCL 转为输出为高。
在读写完 ACK 后的第一个 clock 下降缘从机会钳住 SCL 脚做资料处理，所以
将 SCL 脚置为输入上拉，并等待 SCL 被释放。*/

      while(SCL!=1) { };
      if(SDA==1)
      dat|=0x01;
```

```
    }
return dat;
}
//-----------------------------------------------------------------------------------------------
//函数名称: bit writeIIC(uchar addrW, uchar *writeData, uchar length)
//函数功能: 主控端对从机数据写入
//函数输入: addrW = 从机地址及写入旗帜
//          *writeData = 预备写入数据的首个地址
//          length = 写入数据的长度(字节数)
//函数输出: 返回 IIC 通讯的 acknowledge 状态，若为 1，则停止并返回。若为 0，则完成通讯后返回 //中间变量: i, ACK
//-----------------------------------------------------------------------------------------------
bit writeIIC(uchar addrW, uchar *writeData, uchar length)
{
uchar i;
bit ACK;
sendStart();
sendByte(addrW); //传送地址与写入标记
ACK = readACK();
if (ACK)
{
    sendStop(); //地址不正确或装置未连接，送出停止信号
    return ACK;
}

for(i = 0; i<length; i++)
{
    sendByte(writeData[i]);
    ACK = readACK();
    if (ACK)
    {
```

```
                sendStop(); //未接收到 ACK，送出停止信号
                return ACK;
            }
        }
        sendStop(); //资料写入完成，送出停止信号
        return ACK;
}
//-------------------------------------------------------------------------------------------
//函数名称: bit readIIC(uchar addrR, uchar *readData, uchar length)
//函数功能: 主控端对从机数据读取
//函数输入: addrR = 从机地址及读取旗帜
//                *readData = 预备读取后存放数据的首个地址
//                length = 读取数据的长度(字节数)
//函数输出: 返回 IIC 通讯的 acknowledge 状态，若为 1，则停止并返回。若为
0，则完成通讯后返回 //中间变量: i, ACK
//-------------------------------------------------------------------------------------------
bit readIIC(uchar addrR, uchar *readData, uchar length)
{
        uchar i;
        bit ACK;
        sendStart();
        sendByte(addrR); //传送地址与读取标记
        ACK = readACK();
if (ACK)
{
        sendStop(); //地址不正确或装置未连接，送出停止信号
        return ACK;
        }

        for(i = 0; i<length; i++)
        {
```

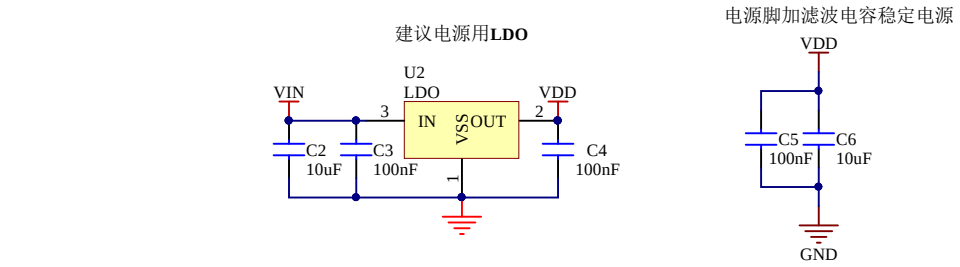
```
            readData[i] = readByte();
            if(i<length-1)
                sendACK();
            else
                sendNOACK(); //读取最后一笔资料，送出 No ACK
        }
        sendStop(); //资料读取完成，送出停止信号
        return ACK;
}
//-------------------------------------------------------------------------------------------
//函数名称: void setWrite_Buffer_3(uchar byte1, uchar byte2, uchar byte3)
//函数功能: 写入 3 个字节到写入缓存寄存器
//函数输入: byte1
//                byte2
//                byte3
//函数输出: 无
//中间变量: 无
//-------------------------------------------------------------------------------------------
void setWrite_Buffer_3(uchar byte1, uchar byte2, uchar byte3)
{
        Write_Buffer[0] = byte1;
        Write_Buffer[1] = byte2;
        Write_Buffer[2] = byte3;
}
void main()
{
bit ACK;
SINT = 1;
setWrite_Buffer_3(0xB1, 0xB9, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //MCU Setting
delay(1); //delay 1ms
setWrite_Buffer_3(0xC0, 0x10, 0x00);
```

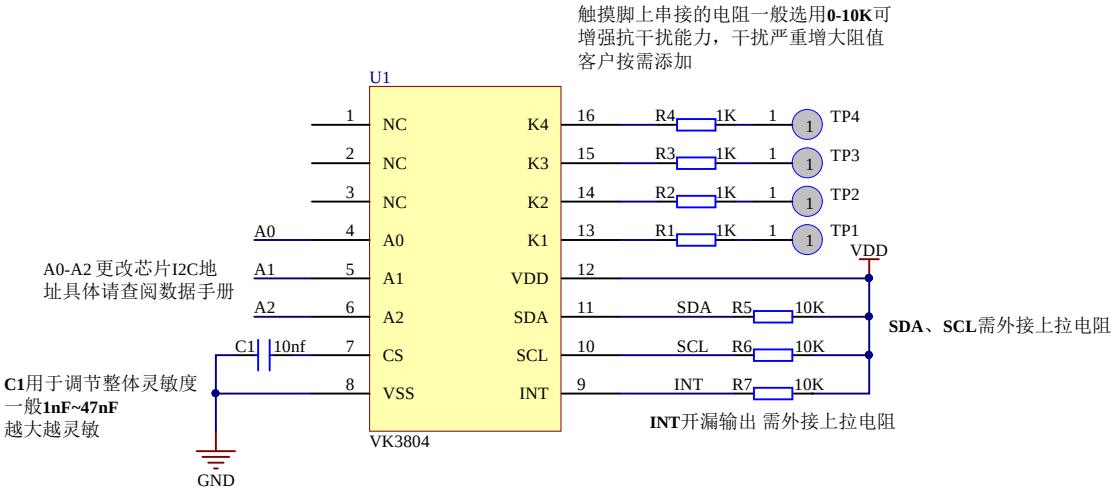
```
ACK = writeIIC(address_W, &Write_Buffer, 3); //TP0 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xC1, 0x10, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //TP1 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xC2, 0x10, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //TP2 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xC3, 0x10, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //TP3 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xC4, 0x10, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //TP4 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xC5, 0x10, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //TP5 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xC6, 0x10, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //TP6 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xC7, 0x10, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //TP7 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xC8, 0x10, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //TP8 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xC9, 0x10, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //TP9 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xCA, 0x10, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //TP10 Threshold
delay(1); //delay 1ms
```

```
setWrite_Buffer_3(0xCB, 0x10, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //TP11 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xD0, 0x02, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //Sleep Group 1 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xD1, 0x0F, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //Sleep Group 1
delay(1); //delay 1ms
setWrite_Buffer_3(0xD2, 0x02, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //Sleep Group 2 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xD3, 0xF0, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //Sleep Group 2
delay(1); //delay 1ms
setWrite_Buffer_3(0xD4, 0x02, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //Sleep Group 3 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xD5, 0x00, 0x0F);
ACK = writeIIC(address_W, &Write_Buffer, 3); //Sleep Group 3
delay(1); //delay 1ms
setWrite_Buffer_3(0xD6, 0x02, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //Sleep Group 4 Threshold
delay(1); //delay 1ms
setWrite_Buffer_3(0xD7, 0x00, 0x00);
ACK = writeIIC(address_W, &Write_Buffer, 3); //Sleep Group 4
delay(50);

while(1)
{
if(!SINT) /*等待读取请求，若关闭省电模式可以不用读取 SINT，但是
每次读取按键建议间格 30ms*/
                ACK = readIIC(address_R, &Read_Buffer, 3); //读取按键状态
    }
}
```

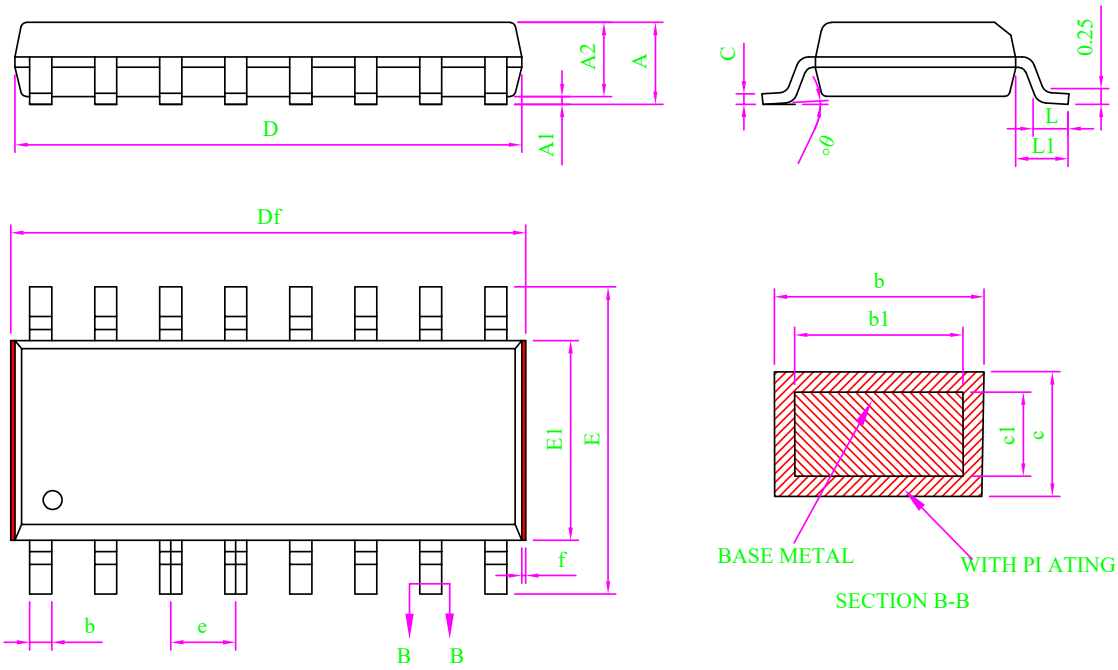
10 参考电路


建议电源用**LDO**
U2
LDO
VIN
VDD
IN
VSS
OUT
3
2
1
C2
10uF
C3
100nF
C4
100nF
电源脚加滤波电容稳定电源
VDD
C5
100nF
C6
10uF
GND


触摸脚上串接的电阻一般选用**0-10K**可
增强抗干扰能力，干扰严重增大阻值
客户按需添加
U1
NC
NC
NC
A0
A1
A2
CS
VSS
K4
K3
K2
K1
VDD
SDA
SCL
INT
R4 1K
R3 1K
R2 1K
R1 1K
TP4
TP3
TP2
TP1
VDD
A0-A2 更改芯片I2C地
址具体请查阅数据手册
C1 10nf
C1用于调节整体灵敏度
一般**1nF~47nF**
越大越灵敏
SDA R5 10K
SCL R6 10K
INT R7 10K
SDA、**SCL**需外接上拉电阻
INT开漏输出 需外接上拉电阻
VK3804
GND

11 封装信息

11.1 SOP16(9.9mm x 3.9mm PP=1.27mm)



Note:

1. All dimension are in mm.
   Dim D&El does not include plastic flash; Df includes plastic flash(f);
   Flash: Plastic residual around body edge after de junk/singulation.
2. Dim b does not include dambar protrusion/intrusion.
3. Plating thickness 0.007mm-0.020mm

MILLIMETER			
SYMBOL	MIN	NOM	MAX
A	-	-	1.75
A1	0.10	0.15	0.20
A2	1.35	1.45	1.55
b	0.39	-	0.47
b1	0.38	0.41	0.43
c	0.20	-	0.25
c1	0.19	0.20	0.21
D	9.80	9.90	10.00
Df	9.90	-	10.40
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
L	0.51	0.66	0.81
L1	0.95	1.05	1.15
θ	0	-	8°
f	0.05	-	0.20

12 免责说明

保修和责任 —— 本文档中的信息是正确可靠的，但我公司对于这些信息的准确性和完整性不作任何保证。对于此类信息的使用后果不负任何责任。在任何情况下，深圳市永嘉微电科技有限公司(以下简称本公司)不会承担任何间接 、意外发生、惩罚性的相关性的损害赔偿，不管这些损害赔偿是基于侵权（包括疏忽）、保修、违约合同或是其他法律理论。

变更的权利 —— 本公司有权在任何时间对此文件发布的信息做出任何改动。更改过的文件将会取代之前所有公布的信息。您可随时查看我们的官网：
https://www.szvinka.com/

适用性 —— 本公司的产品并非是为那些用于对生命和安全有重大关系的系统和设备而设计的。对于使用本公司的产品而导致的故障，造成的人身伤害、甚至死亡、或是严重的财产或环境损害的应用程序。如果本公司的产品应用在此类的设备或应用程序中，本公司对此造成的风险将不承担任何的责任，因此这些风险由客户自行承担。

应用 —— 在这里所有描述有关产品的任何应用程序仅用于说明的目的。在没有进一步测试或修改的情况下，本公司对该应用程序的指定用途是否合适不作任何表示或保证。本公司不负责协助应用程序或客户的产品设计。同时客户应自行负责决定我司的产品是否适合应用计划产品、计划的应用程序以及第三方客户的使用。客户应适当的提供设计和运行，保障措施以尽量减少其产品与应用的相关风险。如果因客户的应用或产品的弱点或缺陷所产生的，或因使用其他第三方的产品而造成的任何缺陷、损失、费用支出等问题，本公司不承担任何责任。客户应负责为其使用本公司产品的第三方客户做必要的产品或应用的测试，以避免使用不当而造成不必要的损失。本公司对在此方面不承担任何责任。

商业销售条件 —— 本公司的产品销售条款适用于通用的商业销售条款。如有其他要求可另出一份单独有效的书面协议，在此种情况下，将适用该单独有效的书面协议条款和条件。关于客户采购本公司的产品，本公司在此明确拒绝适用客户的通用条款和条件。

出口控制 —— 本文档描述的产品以及其项目可能受出口管制条例限制。出口可能需事先获得国家机关许可。

13 历史版本

No.	版本	日期	修订内容	检查
1	1.0	2024-03-22	原始版本	YES
2	1.1	2025-09-05	更新内容	YES

[1] 在开始或完成设计之前，请查阅最近发布的文件。
[2] 自本文档发布以来，本文档中描述的设备产品状态可能已经发生了变化，并且在多个情况下可能会有所不同。最新的产品状态信息可在互联网上查询，网址为 https://www.szvinka.com/