永嘉微电
Vinka Microelectronics

VK3816 数据手册

16通道触摸/3组滑条 I2C输出

Rev.1.1

知识产权说明

深圳市永嘉微电科技有限公司（以下简称“本公司”）已向国内外知识产权部门申请并获得了相关专利，享有这些专利的合法权益，并受到法律的严格保护。

本公司的产品及其相关专利权未经明确授权，任何公司、组织或个人均不得擅自使用。一旦发现任何侵权行为，本公司将采取一切必要的法律手段，坚决遏止此类不当行为，并追究侵权者因侵权行为给本公司造成的损失，或侵权者因此获得的不法利益。

本公司的名称和标识均为注册商标，受法律保护。未经本公司书面许可，任何单位或个人不得使用或仿冒。

在本公司知识产权的保护范围内，任何形式的许可证转让，无论是明示还是暗示，均不被允许。

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，即可设定并读取滑条按键及独立按键触摸数据。

2 特点

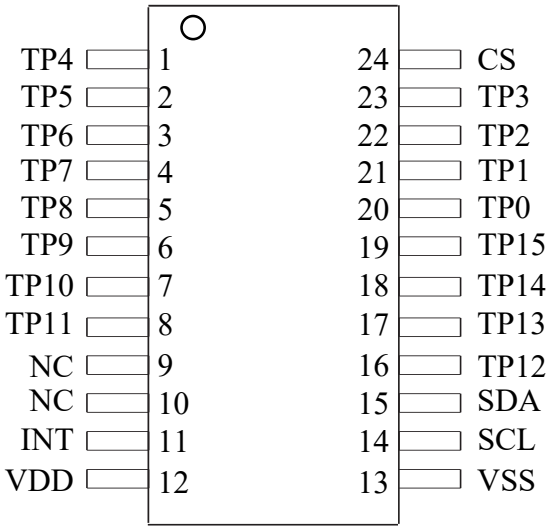
- 此应用使用 16 个 Touch Pad 让客户可依照设计上的需求进行规划。16 个 Touch PAD 可规划为滑条按键或是独立按键，滑条最多可规划为 3 组滑条。当无滑条按键设定时，16 Key 都可做独立按键使用。
- 修改设定参数的方式有两种，用户可配合 USB PCLink Board 来调整滑条按键触摸灵敏度等参数，然后利用 IIC 写入指令来修改设定参数。
- 程序的独立按键输出模式有两种 Multiple 以及Single，当选则为 Multiple 时会将所有按下的按键输出，而选择 Single 时只会输出第一个按下的按键，当按键被放开时，才会输出下一个按键。
- 设计有省电模式，适合用在如遥控器等需要长时间待机的应用。
- 有 4 组 Slave address，可由外部开关切换。
- 封装：
  SSOP24 (150mil) (8.65mm × 3.90mm PP=0.635mm)
  SSOP28(150mil) (9.9mm × 3.9mm PP=0.635mm)

3 应用领域

- 大小家电
- 门禁监控设备
- 消费类电子

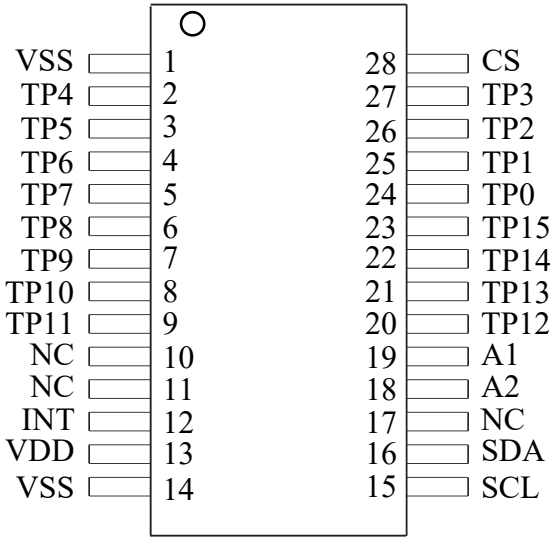
4 管脚排列

TOP VIEW



SSOP24

TOP VIEW



SSOP28

有关详细信息，请参见：34-35页

Ps：SSOP24 pin 封装无法选择 IIC 设备地址，只能用默认值。

CS 与 CAPN 为量测电容接脚，电容大小约 10nF~39nF。
TP0~TP15 是触摸按键的量测 PAD，VK3816 最多可侦测 16 个按键。 REFL 为环境值栓锁输入，空接为高电平，拉低时会停止环境值更新。
A1、A2 是 IIC 的 Slave ID 选择，在后面会说明设定方式。
SDA 是 IIC 的数据输出/输入脚。
SCL 是 IIC 的频率输入脚。

4.1 VK3816/SSOP28/SSOP24 管脚列表

脚位		管脚名称	输入/输出	功能描述
SSOP28	SSOP24			
13	12	VDD	电源正	电源正
14	13	VSS	电源地	电源地
28	24	CS	输入	触摸传感器输入
1	13	VSS	电源地	电源地
17	—	NC	—	悬空
12	11	INT	输入	触摸状态输出，开漏输出需外接上拉电阻
11-10	10-9	NC	—	悬空
15	14	SCL	输入	I2C 串行时钟脚，开漏输出需外接上拉电阻
16	15	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
18-19	—	A1-A2	输入	I2C从机地址选择脚
24-27 2-5	20-23 1-4	TP0-TP7	输入	触摸输入
6-9 20-23	5-8 16-19	TP8-TP15	输入	触摸输入

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	TSTG	——	-50~+125	℃
电源电压	VDD	Ta=25℃	VSS-0.3~VSS+5.5	V
输入电压	VIN	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	-	4M	-	HZ
工作电流	IOP	工作，VDD=3V输出无负载	-	1.1	-	mA
	IOFF	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，需要用户配合 USB PCLink Board 操作，并依照实际按压读取的数据来调整。

滑条介绍：

滑条按键的原理是利用在 PCB LAYOUT 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:

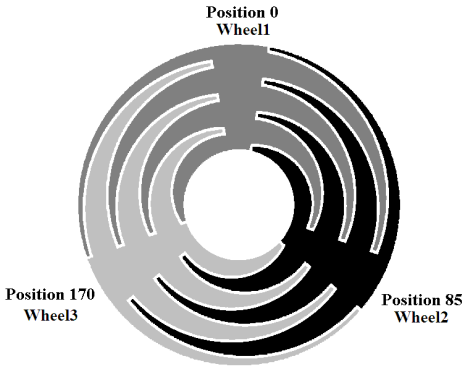


图1. 环形设计

图2. 直条设计

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。

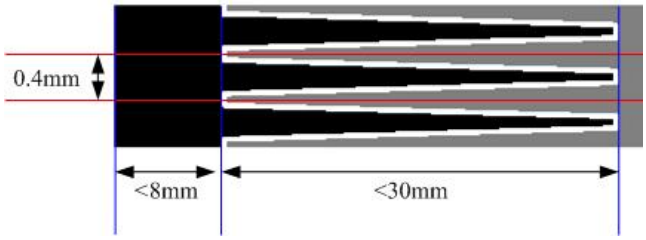


图3. Layout 设计要点

滑条按键可设置 3~16Key，当设置为 3Key 时使用的是 TP0~TP2，TP3~TP15 可规划为一般按键;若滑条按键设置为 4Key 则使用 TP0~TP3，TP4~TP15 可规划为一般按键。
如下表:

Slide 1	Disable	3Key	3Key	4Key	Disable	Disable	16Key
Slide 2	Disable	Disable	3Key	4Key	8Key	Disable	Disable
Slide 3	Disable	Disable	Disable	4Key	8Key	4Key	Disable
TP0	Key 1	Slide 1_1	Slide 1_1	Slide 1_1	Slide 2_1	Slide 3_1	Slide 1_1
TP1	Key 2	Slide 1_2	Slide 1_2	Slide 1_2	Slide 2_2	Slide 3_2	Slide 1_2
TP2	Key 3	Slide 1_3	Slide 1_3	Slide 1_3	Slide 2_3	Slide 3_3	Slide 1_3
TP3	Key 4	Key 1	Slide 2_1	Slide 1_4	Slide 2_4	Slide 3_4	Slide 1_4
TP4	Key 5	Key 2	Slide 2_2	Slide 2_1	Slide 2_5	Key 1	Slide 1_5
TP5	Key 6	Key 3	Slide 2_3	Slide 2_2	Slide 2_6	Key 2	Slide 1_6
TP6	Key 7	Key 4	Key 1	Slide 2_3	Slide 2_7	Key 3	Slide 1_7
TP7	Key 8	Key 5	Key 2	Slide 2_4	Slide 2_8	Key 4	Slide 1_8
TP8	Key 9	Key 6	Key 3	Slide 3_1	Slide 3_1	Key 5	Slide 1_9
TP9	Key 10	Key 7	Key 4	Slide 3_2	Slide 3_2	Key 6	Slide 1_10
TP10	Key 11	Key 8	Key 5	Slide 3_3	Slide 3_3	Key 7	Slide 1_11
TP11	Key 12	Key 9	Key 6	Slide 3_4	Slide 3_4	Key 8	Slide 1_12
TP12	Key 13	Key 10	Key 7	Key 1	Slide 3_5	Key 9	Slide 1_13
TP13	Key 14	Key 11	Key 8	Key 2	Slide 3_6	Key 10	Slide 1_14
TP14	Key 15	Key 12	Key 9	Key 3	Slide 3_7	Key 11	Slide 1_15
TP15	Key 16	Key 13	Key 10	Key 4	Slide 3_8	Key 12	Slide 1_16

轮垫和键垫定义

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协定

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。

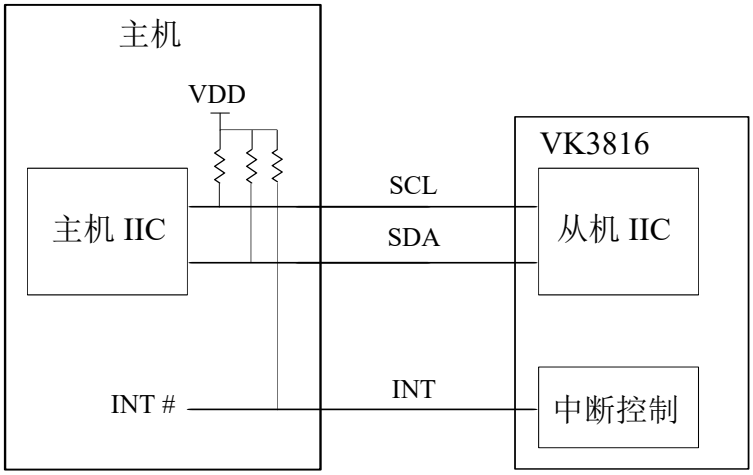


图4. 主机和VK3816通过IIC连接

INT 在无按键状态变化时为高，当有按键状态变化时，INT 脚位会拉底 100ms。若 Slave 接收到 Slave address 则会清除回复为高。

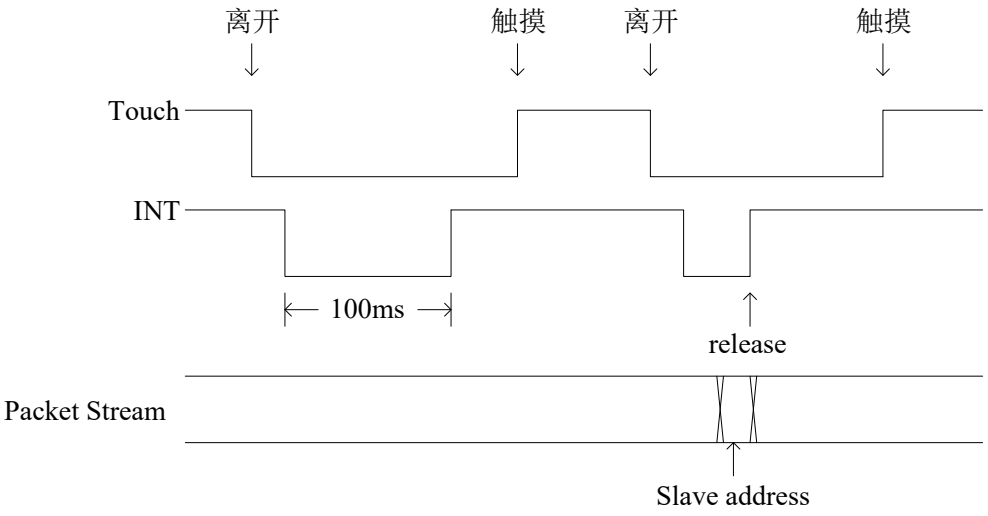


图5. INT 脚描述

7.1 开关的特点

符号	描述	最小值	最大值	单位
FSCL	SCL时钟频率	0	100	KHz
THDSTA	保持时间(重复)星型状态。 在这个周期之后，产生第一个时钟脉冲	4.0	-	us
TLOW	SCL时钟低周期	4.7	-	us
THIGH	SCL时钟的高周期	4.0	-	us
TSUSTA	重复启动条件的设置时间	4.7	-	us
THDDAT	数据保持时间	0	-	us
TSUDAT	数据设置时间	250	-	ns
TSUSTO	停止状态的设置时间	4.0		us
TBUF	在停止和开始之间的空闲时间条件	4.7	-	us
TSPI	脉冲的脉冲宽度被输入滤波器抑制	0	50	ns
TSPT	从处理器时间	10	75	us

VDD的IIC SDA和SCL引脚的交流特性

7.2 时间波形

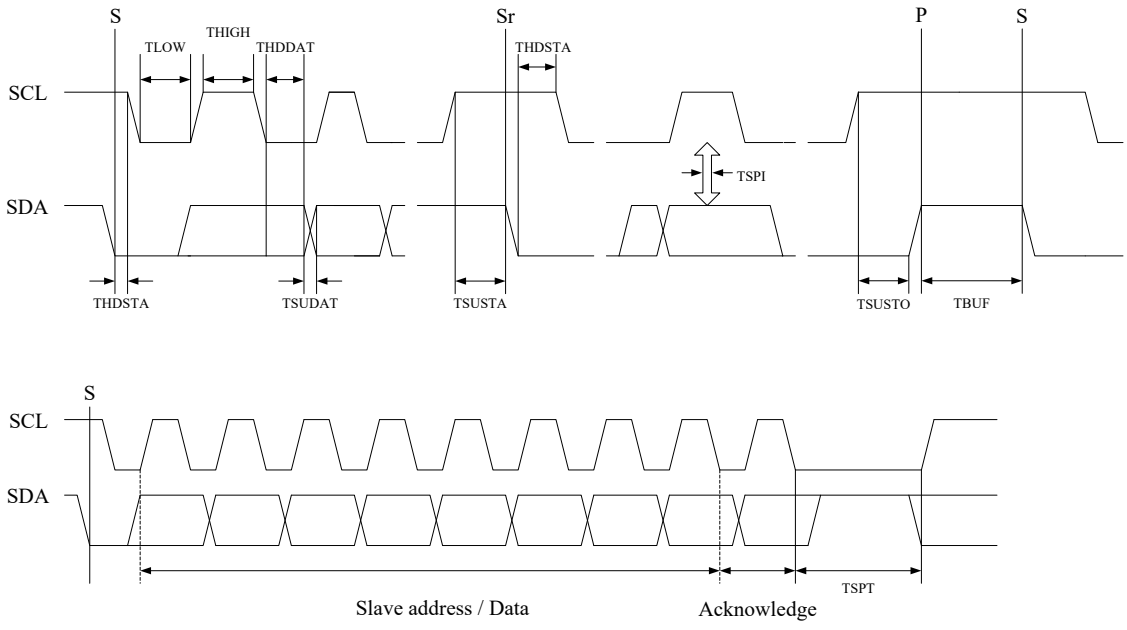


图6. IIC上快速/标准模式的时序定义

7.3 数据包流

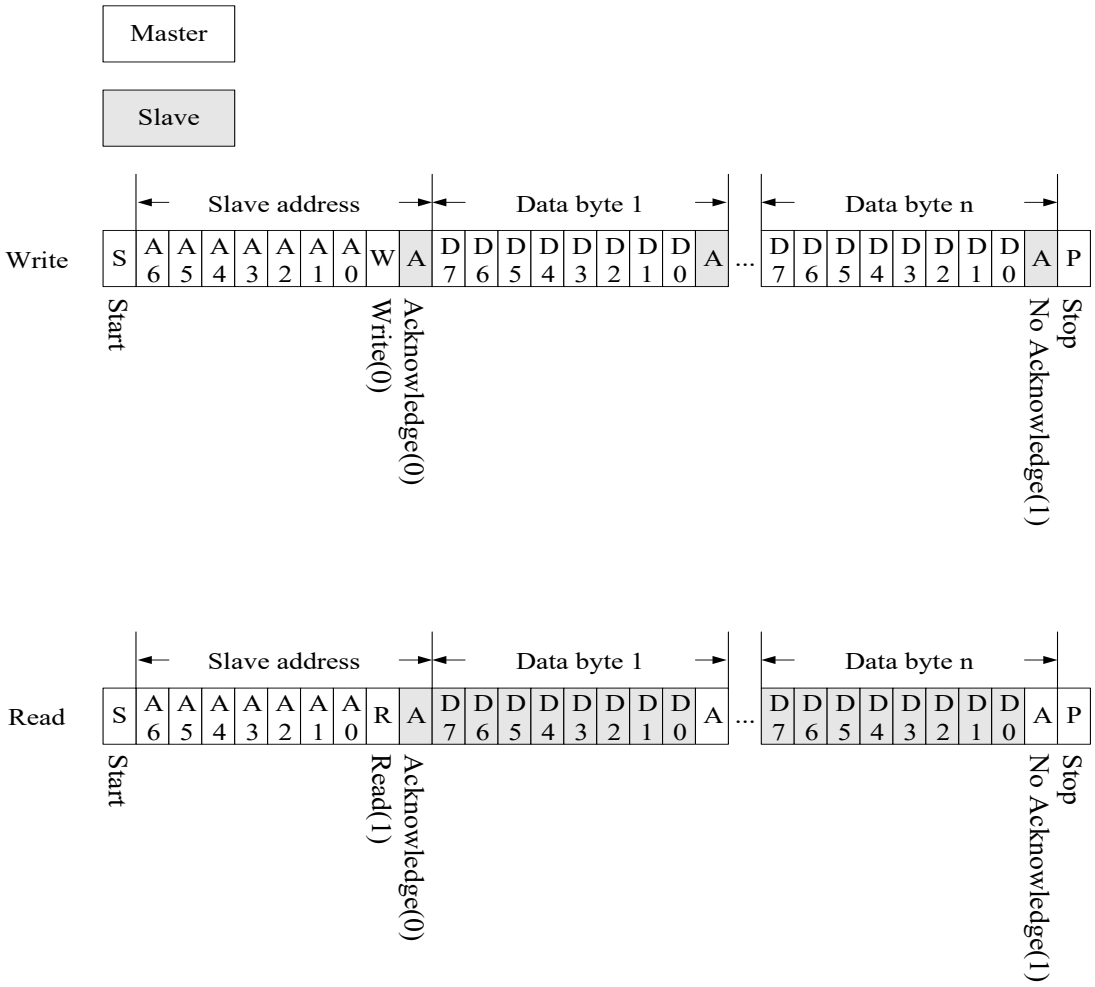


图7. 从I2C写入/读取字节

7.4 从设备地址

可透过 A1、A2 脚位做切换。如下表:

A1	A2	Slave address (A6-A0)	Write (A6-A0,R)	Read (A6-A0,R)
0	0	50H	A0H	A1H
0	1	51H	A2H	A3H
1	0	52H	A4H	A5H
1	1	53H	A6H	A7H

* A1、A2 空接时 Slave address 为 53H

7.5 数据流

软件设计有两种工作模式，一种是 PC Link 模式，另一种是滑条应用模式。PC Link 模式需要配合 USB PCLink Board 来读取触摸计数值，用以设定适当的按压阀值(Threshold)。滑条应用模式则可设定系统参数，并读取按键状态以及 Wheel 的输出。当写入数据第一个 Data Bytes 的 7 bit 为 0 时，设定系统为 PC Link 模式;若为 1，则设定为滑条模式。

在滑条应用模式，写入数据以 3~4 Data Bytes 为一组资料串流。当一笔数据串流写入完成后，系统会将数据覆写并进行系统重设。若写入被中断并重新写入，则前一笔数据会被放弃。

在每次写入完一组 3~4 Data Bytes 后，若要再次写入下一组设定，需要送出 Stop 讯号结束当前数据传输再重新写入下一组设定。

8 应用模式

写入数据

8.1 设置命令

Data byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	IICM=1	CT=0	KOM	AA	PSM	DT	ART	
2	Key Num					KAT		
3	Slide 2 Num				Slide 1 Num			
4	Key Off Num				Slide 3 Num			

8.1.1 IICM

IIC 数据模式选择。

IICM	IIC Mode
0	PC Link mode
1	Slide application mode (Define)

8.1.2 CT

在 Slide application 模式，写入数据区分成应用设定以及阀值设定，当 CT 为 0 时是写入应用设定，当 CT=1 时是写入阀值设定。

CT	Custom Threshold
0	Setting commands
1	Custom threshold commands

8.1.3 KOM

按键输出模式，有多个按键输出以及单一按键输出两种模式。此选项是对普通按键的输出设定，滑条按键则不受影响。单一按键输出模式时只会输出第一个被按下的按键，当按键放开后才会承认其它按键。

KOM	Key Output Mode
0	Multiple
1	Single (Define)

8.1.4 AA

基准值自动调整，当无按键时，自动更新基准值。

AA	Auto Adjust
0	Disable
1	Enable (Define)

8.1.5 PSM

省电模式，无按键 4 秒后进入睡眠模式。

PSM	Power Save Mode
0	Disable
1	Enable (Define)

静态电流为：6.8uA@3.0V 工作电流：1.1mA@3.0V

8.1.6 DT

动态阀值，功能开启时，滑调按键的阀值会依照按压位置自动调整。

DT	Dynamic Threshold
0	Disable (Define)
1	Enable

8.1.7 ART

自动重置时间设定，在按键位置没有改变时开始计时，时间到自动重置。

ART		Auto Reset Time
0	0	Disable
0	1	15 second (Define)
1	0	30 second
1	1	60 second

8.1.8 KAT

按键消抖时间。

KAT			Key Acknowledge Times
2	1	0	
0	0	0	1 times
0	0	1	2 times
0	1	0	3 times
0	1	1	4 times (Define)
1	0	0	5 times
1	0	1	6 times
1	1	0	7 times
1	1	1	8 times

8.1.9 Key Num

按键数设定，当滑条设定 Disable 时普通按键最大按键数为 16 Keys。当普通按键数设定 16Keys 时，若有滑条按键设定，则以最大按键数 16 减去滑条按键数，为普通按键数。

Key Num					Key Number
4	3	2	1	0	
0	0	0	0	0	Disable
0	0	0	0	1	1 key
0	0	0	1	0	2 keys
0	0	0	1	1	3 keys
0	0	1	0	0	4 keys
0	0	1	0	1	5 keys
0	0	1	1	0	6 keys
0	0	1	1	1	7 keys
0	1	0	0	0	8 keys
0	1	0	0	1	9 keys
0	1	0	1	0	10 keys
0	1	0	1	1	11 keys
0	1	1	0	0	12 keys
0	1	1	0	1	13 keys
0	1	1	1	0	14 keys
0	1	1	1	1	15 keys
1	0	0	0	0	16 keys (Define)

8.1.10 Slide x Num

滑条按键数设定，最多为 16 Keys。

Slide x Num				Slide x Number
3	2	1	0	
0	0	0	0	Disable (Define)
0	0	0	1	Disable
0	0	1	0	3 keys Slide
0	0	1	1	4 keys Slide
0	1	0	0	5 keys Slide
0	1	0	1	6 keys Slide
0	1	1	0	7 keys Slide
0	1	1	1	8 keys Slide
1	0	0	0	9 keys Slide
1	0	0	1	10 keys Slide
1	0	1	0	11 keys Slide
1	0	1	1	12 keys Slide
1	1	0	0	13 keys Slide
1	1	0	1	14 keys Slide
1	1	1	0	15 keys Slide
1	1	1	1	16 keys Slide

8.1.11 Key Off Num

多按键重置设定，最多为 16 Keys。
以 Slide1、Slide2、Slide3、Normal 个别按压按键数做判断。

Key Off Num				Key Off Number
3	2	1	0	
0	0	0	0	Disable (Define)
0	0	0	1	2key reset
0	0	1	0	3 keys reset
0	0	1	1	4 keys reset
0	1	0	0	5 keys reset
0	1	0	1	6 keys reset
0	1	1	0	7 keys reset
0	1	1	1	8 keys reset
1	0	0	0	9 keys reset
1	0	0	1	10 keys reset
1	0	1	0	11 keys reset
1	0	1	1	12 keys reset
1	1	0	0	13 keys reset
1	1	0	1	14 keys reset
1	1	1	0	15 keys reset
1	1	1	1	16 keys reset

8.2 自定义阈值命令

阀值则是设定按键承认的门坎。分为按键阀值，睡眠唤醒阀值两种。

8.2.1 Item

选择切换不同的写入参数的设定。

Item		Item
0	0	TPx setting
0	1	Sleep setting
1	0	-
1	1	-

8.2.2 TPx Setting

Data byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	IICM=1	CT=1	Item=0		TP Num			
2	TPx Threshold M				TPx Threshold L			
3					TPx Threshold H			

8.2.2.1 TPx Threshold M

TPx Threshold : 按键承认阀值。(Define : 010H)

按键承认阀值越小灵敏度越高，越大灵敏度越低。预设的阀值为 010H，建议的最小值为 008H，若调整到 008H 按键灵敏度仍然不够，则建议加大 CS 电容，CS 电容的值则建议小于 39nF。

8.2.2.2 TP Num

按键的期待值与阀值设定是依照触摸按键的脚位编排，若滑条按键 3 keys 普通按键 13keys，则TP0 – TP2 为滑条按键，TP3 - TP15 为普通按键。

数据写入的按键编号。

TP NUM				TP Number
3	2	1	0	
0	0	0	0	TP0
0	0	0	1	TP1
0	0	1	0	TP2
0	0	1	1	TP3
0	1	0	0	TP4
0	1	0	1	TP5
0	1	1	0	TP6
0	1	1	1	TP7
1	0	0	0	TP8
1	0	0	1	TP9
1	0	1	0	TP10
1	0	1	1	TP11
1	1	0	0	TP12
1	1	0	1	TP13
1	1	1	0	TP14
1	1	1	1	TP15

8.2.3 Sleep Setting

Data byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	IICM=1	CT=1	Item=1		-			
2	TPSLP Threshold M				TPSLP Threshold L			
3					TPSLP Threshold H			

TPSLP Threshold : 省电模式唤醒阀值。(Define : 002H)

8.3 Read Data

Data byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	C	WSET				S3T	S2T	S1T
2	Key 8	Key 7	Key 6	Key 5	Key 4	Key 3	Key 2	Key 1
3	Key 16	Key 15	Key 14	Key 13	Key 12	Key 11	Key 10	Key 9
4	S1 Position							
5	S2 Position							
6	S3 Position							

8.3.1 C

系统校正标志，当值为 0 时，表示系统校正中，键值读取无效。当值为 1 时，键值有效。

C	Calibrate
0	Calibrating
1	Calibrate Finish

8.3.2 WSET

系统写入标志，上电为 1，写入设定后该标志设置为 0。

WSET	Have write setting
0	Have write setting
1	No write setting

8.3.3 SxT

滑条按键标志，无按键为 0，有按键为 1。

SxT	Slide x Touch
0	No touch
1	Touch

8.3.4 K1…K16

触摸按键标志，无按键为 0，有按键为 1。

K1…K16	Key1…Key16
0	No touch
1	Touch

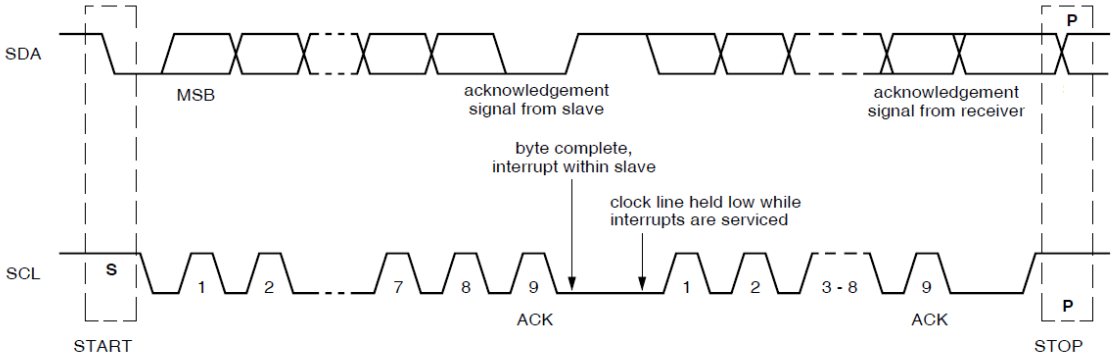
8.3.5 Sx Position

滑条位置标志，预设为 0，触摸滑条后输出按压位置，放开后保留最后按压位置。

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

9 特别说明

1. VK3816的I2C界面有硬件的支持SCL可支持100KHz，但是译码为软件处理，所以当Master的第9个SCL为Low时，VK3816会马上将SCL的bus拉Low，表示VK3816进入busy的状态，同时VK3816 内部会产生中断处理I2C的解码，处理约需20~100us视处理的情况而定，等处理完就会释放SCL，一般主控的SCL控制脚为Nmos的输出，需外加上拉电阻，以免主控无法将SCL拉High。



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，若由过程控制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):
先确定设计中是否使用滑条功能，若使用滑条功能，则建议 33nF 作为 CS 充放电电容，若仅做一般按键使用，则建议使用 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。

滑条按键因为锯齿状设计，在不同位置灵敏度也会不同，故建议做灵敏度测试时，以两个锯齿按键中间位置做灵敏度测试，避免滑动效果不佳。

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。

Step4. 调整按键反应速度:

按键消抖时间(KAT)预设为 4，若按键反应速度不构快，可以下修值到 3。

若下修到 3 反应速度仍然不够，则建议将 CS 电容减小。

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。

10 示范程序

```
/*
    项目名称:主控端对 VK3816 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 VK3816 写入设定参数
             2.透过软件模拟 IIC 主控端对 VK3816 读取按键状态
主控 MCU:AT89C51
    Date & Version:    2018/01/06 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[4]; //主控端的写入资料缓存
uchar Read_Buffer[6]; //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 表示有 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) { };
        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;
```

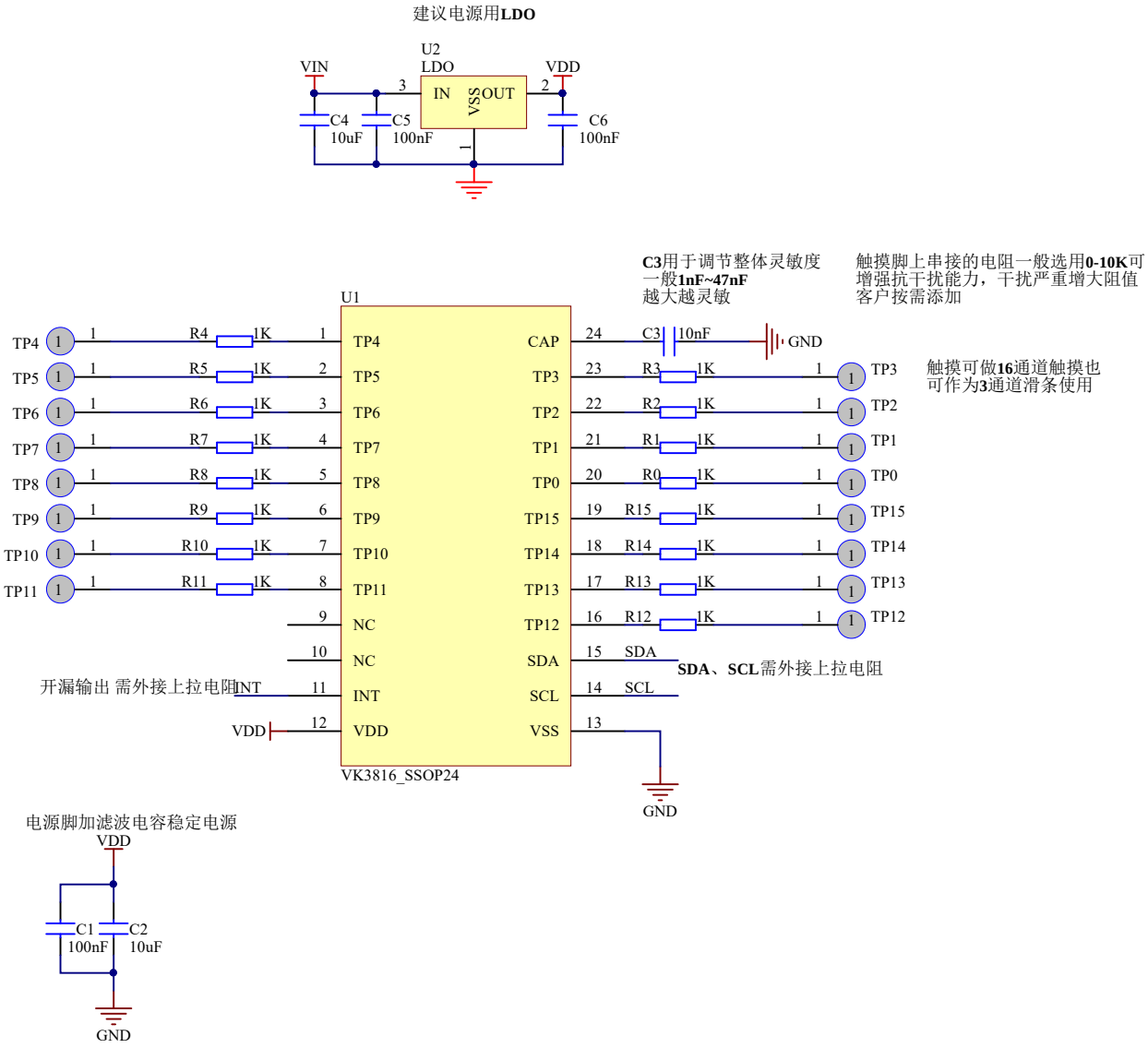
```
    }

    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_4(uchar byte1, uchar byte2, uchar byte3, uchar byte4)
//函数功能: 写入 4 个字节到写入缓存寄存器
//函数输入: byte1
//          byte2
//          byte3
//          byte4
//函数输出: 无
//中间变量: 无
//-------------------------------------------------------------------------------------------
void setWrite_Buffer_4(uchar byte1, uchar byte2, uchar byte3, uchar byte4)
{
    Write_Buffer[0] = byte1;
    Write_Buffer[1] = byte2;
    Write_Buffer[2] = byte3;
    Write_Buffer[3] = byte4;
}
//-------------------------------------------------------------------------------------------
//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_4(0xB1, 0x23, 0x33, 0x03);
    ACK = writeIIC(address_W, &Write_Buffer, 4); //MCU Setting
    setWrite_Buffer_3(0xC0, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP0 Threshold
    setWrite_Buffer_3(0xC1, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP1 Threshold
    setWrite_Buffer_3(0xC2, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP2 Threshold
    setWrite_Buffer_3(0xC3, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP3 Threshold
    setWrite_Buffer_3(0xC4, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP4 Threshold
    setWrite_Buffer_3(0xC5, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP5 Threshold
    setWrite_Buffer_3(0xC6, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP6 Threshold
    setWrite_Buffer_3(0xC7, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP7 Threshold
    setWrite_Buffer_3(0xC8, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP8 Threshold
    setWrite_Buffer_3(0xC9, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP9 Threshold
    setWrite_Buffer_3(0xCA, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP10 Threshold
    setWrite_Buffer_3(0xCB, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP11 Threshold
    setWrite_Buffer_3(0xCC, 0x10, 0x00);
```

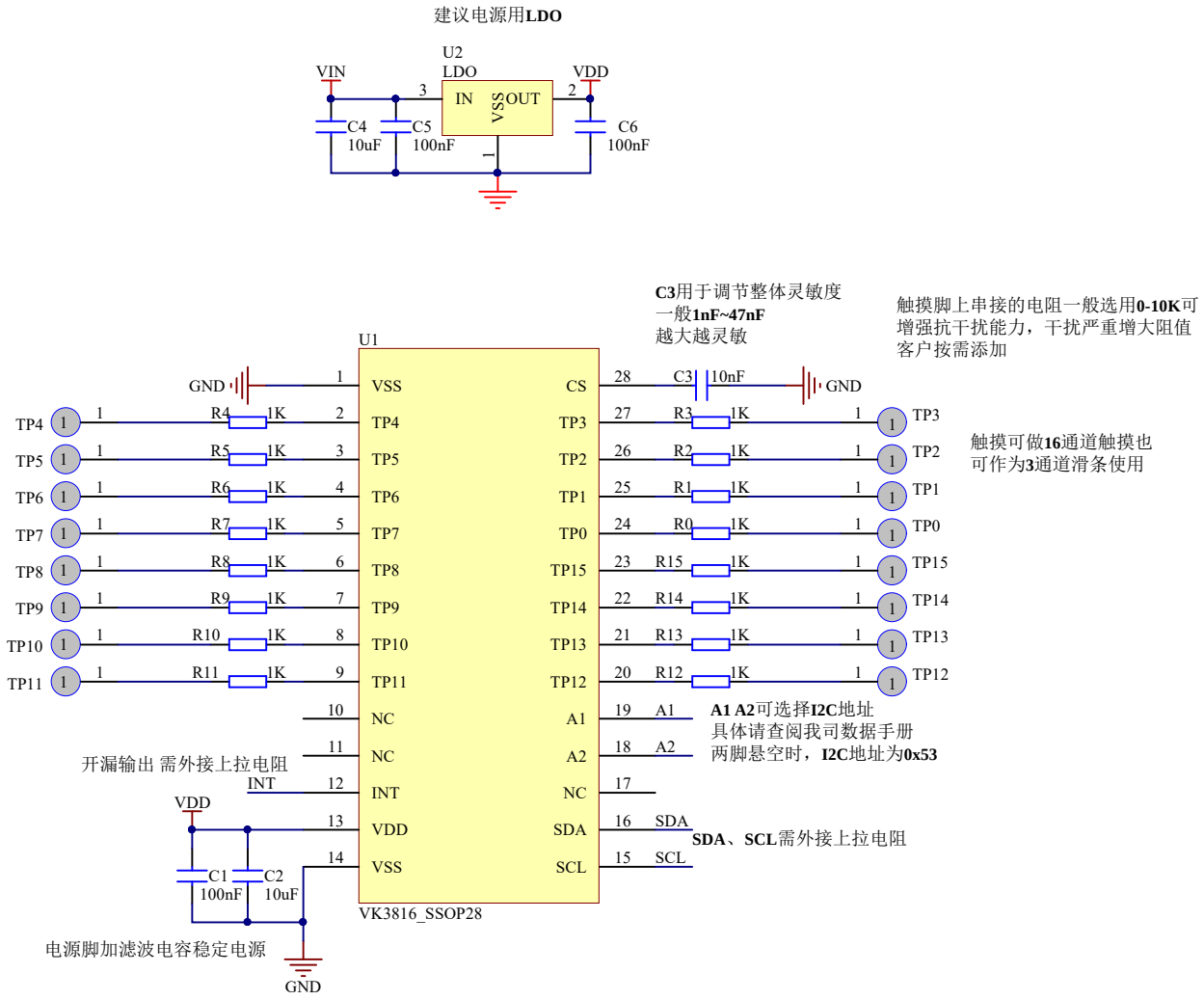
```
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP12 Threshold
    setWrite_Buffer_3(0xCD, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP13 Threshold
    setWrite_Buffer_3(0xCE, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP14 Threshold
    setWrite_Buffer_3(0xCF, 0x10, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //TP15 Threshold
    setWrite_Buffer_3(0xD0, 0x02, 0x00);
    ACK = writeIIC(address_W, &Write_Buffer, 3); //Sleep Threshold
    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, 6); //读取按键状态
    }
}
```

11 参考电路

11.1 SSOP24

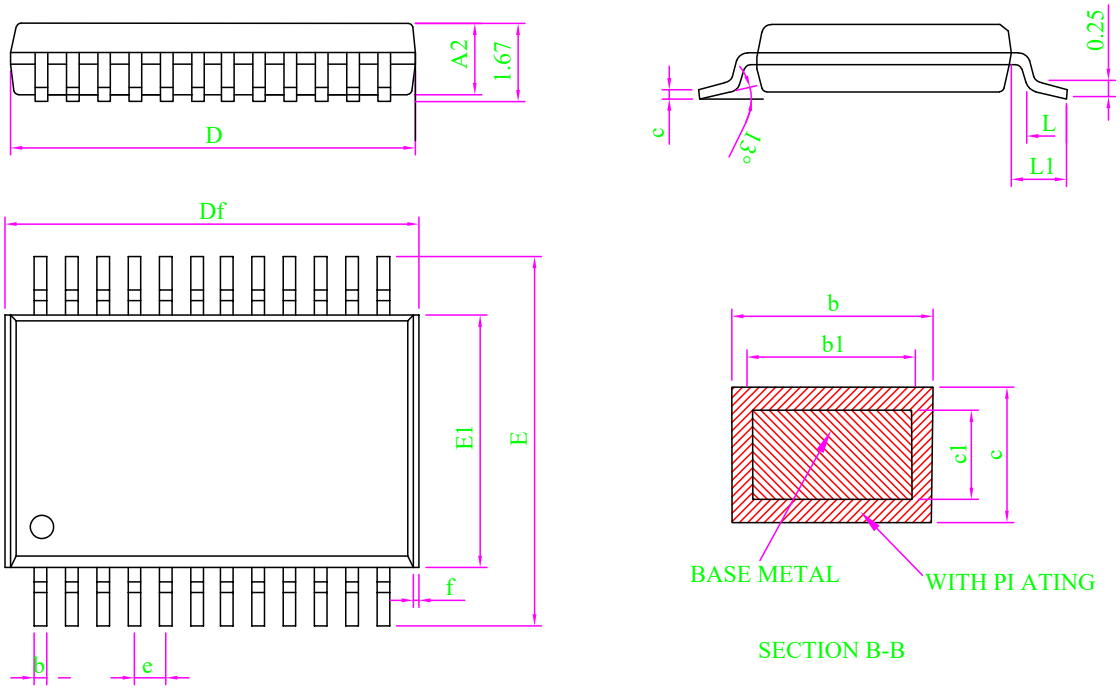


11.2 SSOP28


建议电源用**LDO**
U2
LDO
VIN
VDD
IN
VSS
OUT
C4
10uF
C5
100nF
C6
100nF
C3用于调节整体灵敏度
一般**1nF~47nF**
越大越灵敏
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
GND
VSS
CS
C3
10nF
GND
TP4
R4
1K
TP4
TP3
R3
TP3
TP5
R5
TP5
TP2
R2
TP2
触摸可做**16**通道触摸也可作为**3**通道滑条使用
TP6
R6
TP6
TP1
R1
TP1
TP7
R7
TP7
TP0
R0
TP0
TP8
R8
TP8
TP15
R15
TP15
TP9
R9
TP9
TP14
R14
TP14
TP10
R10
TP10
TP13
R13
TP13
TP11
R11
TP11
TP12
R12
TP12
NC
A1
A2
A1 A2可选择**I2C**地址
具体请查阅我司数据手册
两脚悬空时，**I2C**地址为**0x53**
开漏输出 需外接上拉电阻
INT
VDD
SDA
SCL
SDA、**SCL**需外接上拉电阻
C1
100nF
C2
10uF
VK3816_SSOP28
电源脚加滤波电容稳定电源
GND

12 封装信息

12.1 SSOP24 (150mil) (8.65mm × 3.90mm PP=0.635mm)

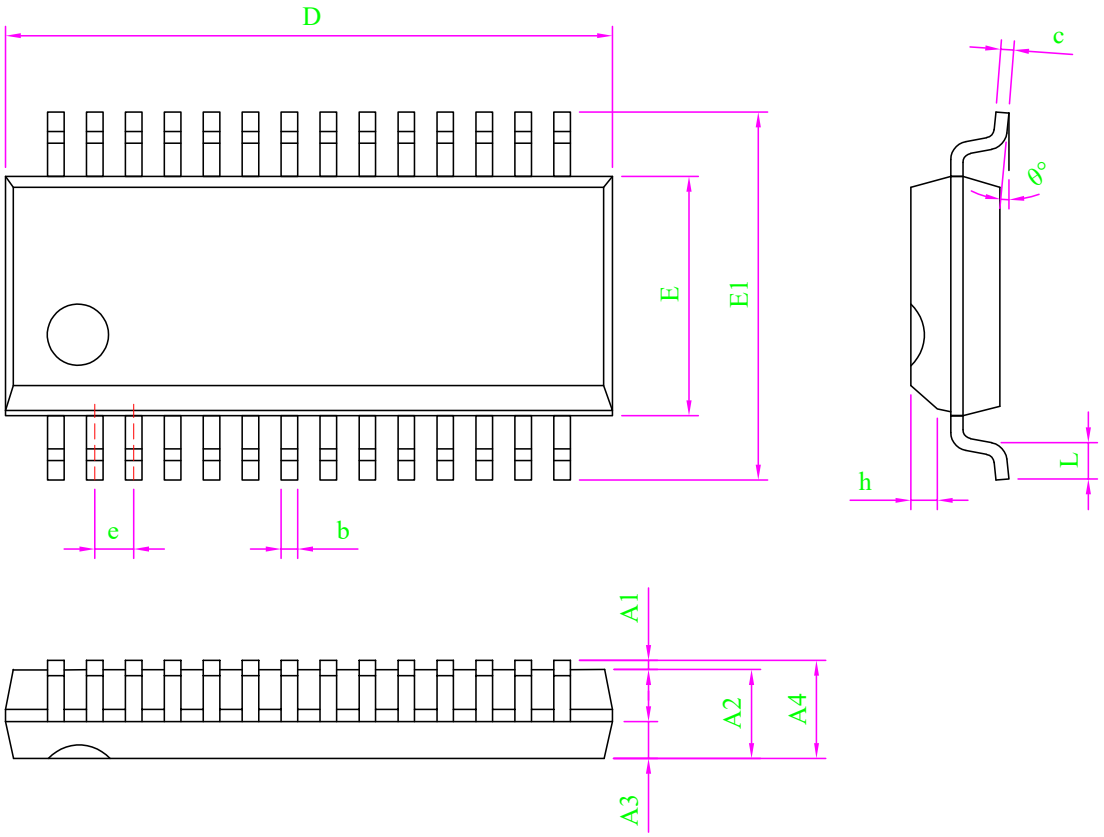


Note:

1. All dimension are in mm.
2. Dim D&El does not include plastic flash; Df includes plastic flash(f);
Flash: Plastic residual around body edge after de junk/singulation.
3. Dim b does not include dambar protrusion/intrusion.
4. Plating thickness 0.007mm-0.015mm

SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	-	-	1.75
A1	0.10	0.15	0.20
A2	1.35	1.45	1.55
b	0.23	-	0.31
b1	0.22	0.25	0.28
c	0.20	-	0.24
c1	0.19	0.20	0.21
D	8.56	8.66	8.76
Df	8.66	-	9.16
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	0.635 BSC		
L	0.51	0.66	0.81
L1	0.95	1.05	1.15
θ	0	-	8°
f	0.05	-	0.20

12.1 SSOP28(150mil) (9.9mm × 3.9mm PP=0.635mm)



SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	-	-	1.75
A1	0.05	-	0.23
A2	1.35	1.45	1.55
A3	0.60	0.65	0.70
b	0.23	-	0.31
c	0.19	-	0.25
D	9.80	9.90	10.00
E	3.90	3.90	4.00
E1	5.80	6.00	6.20
e	0.635 BSC		
h	0.30	-	0.50
L	0.60	-	0.80
θ	0	-	8°

13 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。

14 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	2025-09-08	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/