

TTY6955 3 Slide Application

规格书 V1.2

● 软件简介:	2
● 特色:	2
● 产品应用范围:	2
● 封装脚位图与功能说明:	3
● 脚位定义:	4
● 电气特性:	5
1 最大绝对额定值	5
2 DC/AC 特性: (测试条件为室温=25℃)	5
● 功能描述:	5
● 特别说明:	20
● 示范程序:	22
● 建议线路:	31
● 封装说明:	32

● 软件简介:

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。

● 特色:

- 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。

● 产品应用范围:

- 大小家电
- 门禁监控设备
- 消费类电子

● 封装脚位图与功能说明:

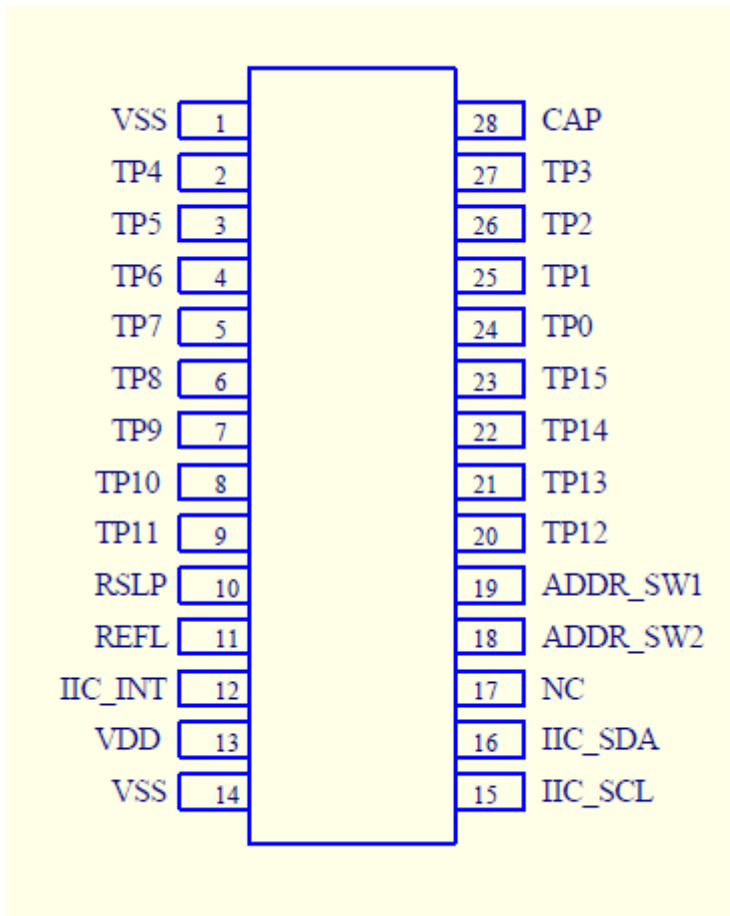


Figure5. pin define for IC package

CAPN 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，TTY6955 最多可侦测 16 个按键。

RSLP 为省电模式输出，平时保持高电平，进入省电模式后则输出低电平。

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。

ADDR_SW1、ADDR_SW2 是 IIC 的 Slave ID 选择，在后面会说明设定方式。

IIC_SDA 是 IIC 的数据输出/输入脚。

IIC_SCL 是 IIC 的频率输入脚。

● 脚位定义:

Pin	Define	I/O	Pin Description
-	RSTB	I	External reset input, active low 50kΩ pull-up(5v)
13	V _{DD}	Power	Positive power supply
14	V _{SS}	Power	Negative power supply, ground
28	CAP	I	Touch sensor input
1	V _{SS}	Power	
17	NC	-	
12	IIC_INT	IO	IIC interrupt pin
11	REFL	I	Reference Lock
10	RSLP	O	Read Sleep enter
-	-	-	
15	IIC_SCL	IO	IIC clock pin
16	IIC_SDA	IO	IIC data pin
18	ADDR_SW1	I	IIC slave address select
19	ADDR_SW2	I	IIC slave address select
24	TP0	IO/I	touch pad input
25	TP1	IO/I	touch pad input
26	TP2	IO/I	touch pad input
27	TP3	IO/I	touch pad input
2	TP4	IO/I	touch pad input
3	TP5	IO/I	touch pad input
4	TP6	IO/I	touch pad input
5	TP7	IO/I	touch pad input
6	TP8	IO/I	touch pad input
7	TP9	IO/I	touch pad input
8	TP10	IO/I	touch pad input
9	TP11	IO/I	touch pad input
20	TP12	IO/I	touch pad input
21	TP13	IO/I	touch pad input
23	TP14	IO/I	touch pad input
24	TP15	IO/I	touch pad input

Table1. TTY6955 pin description

● 电气特性:

1 最大绝对额定值

参数	符号	条件	值	单位
工作温度	Top	——	-40~+85	°C
存放温度	T _{STG}	——	-50~+125	°C
电源电压	VDD	Ta=25°C	VSS-0.3~VSS+5.5	V
输入电压	V _{IN}	Ta=25°C	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				

2 DC/AC 特性: (测试条件为室温=25°C)

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	I _{OP}	待机, 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

● 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：

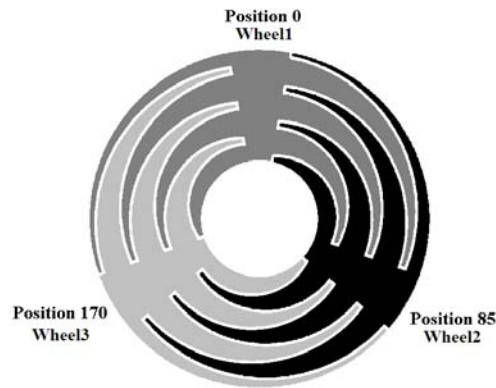


Figure1. 环形设计



Figure2. 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。

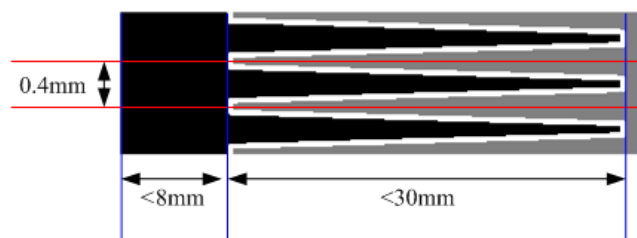


Figure3. Layout 设计要点

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

Table2. Wheel pad and key pad define

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。

● 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。

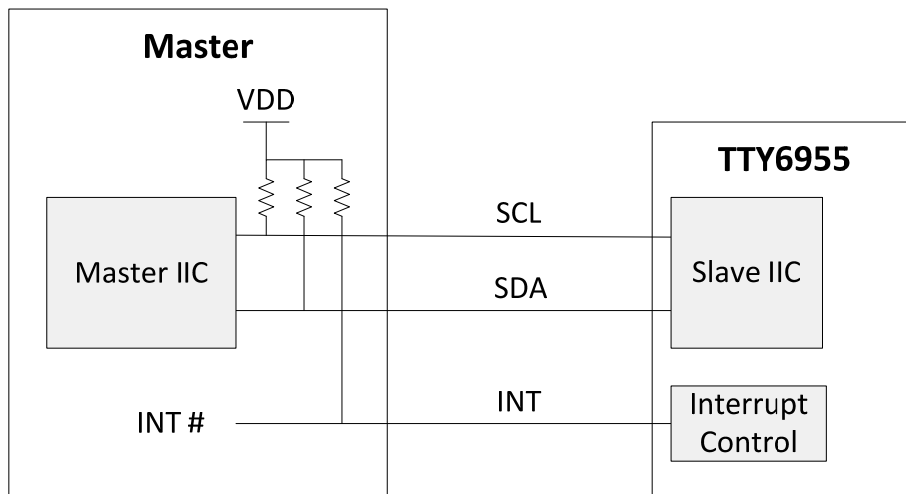


Figure6. IIC connect for master and TTY6955

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

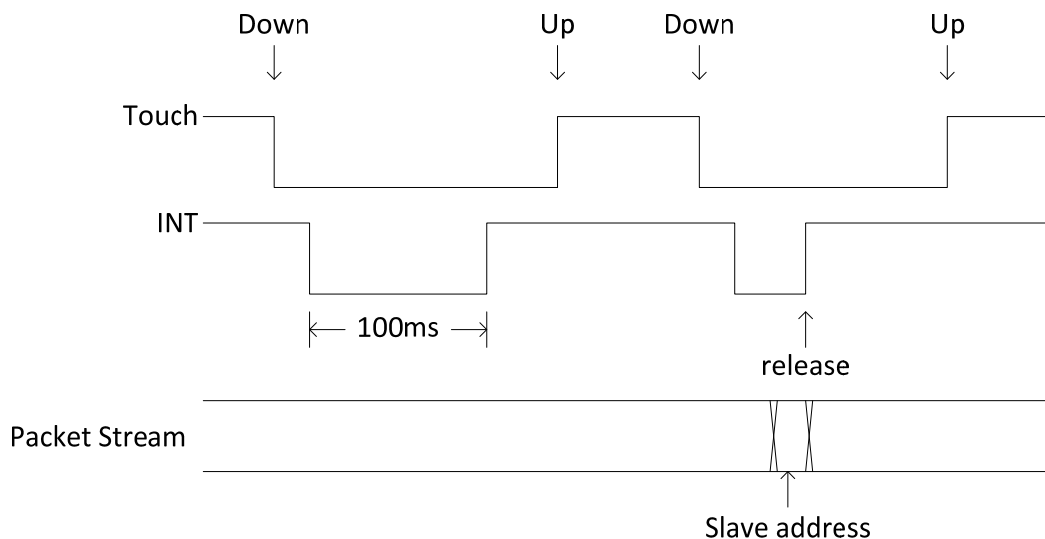


Figure7. INT pin describe

在 Slave Address、Data Byte 传送或接收的第 9 clock 结束时(下拉)，Slave (TTY6955)会将 SCL 拉 Low 20~100us 的时间来处理数据，待处理完成后才会释放 SCL。因此 Master 需要等待 SCL 释放后才能继续读写数据。

简单的设定方式是在每次 Master 将 SCL 拉 High 后，读取并等待 SCL 为 High。或是在第 9 clock 结束时(下拉)拉低并停留 100us。

Switching Characteristics

Symbol	Description	Min	Max	Units
FSCL	SCL clock frequency.	0	100	KHz
THDSTA	Hold time(repeated) star condition. After this period, the first clock pulse is generated.	4.0	-	us
TLOW	Low period of the SCL clock.	4.7	-	us
THIGH	High period of the SCL clock.	4.0	-	us
TSUSTA	Set-up time for a repeated start condition.	4.7	-	us
THDDAT	Data hold time.	0	-	us
TSUDAT	Data set-up time.	250	-	ns
TSUSTO	Set-up time for stop condition.	4.0	-	us
TBUF	Bus free time between a stop and start condition.	4.7	-	us
TSPI	Pulse width of spikes are suppressed by the input filter.	0	50	ns
TSPT	Slave processor time	10	75	us

Table3. AC characteristics of the IIC SDA and SCL pins for vdd

Timing Waveform

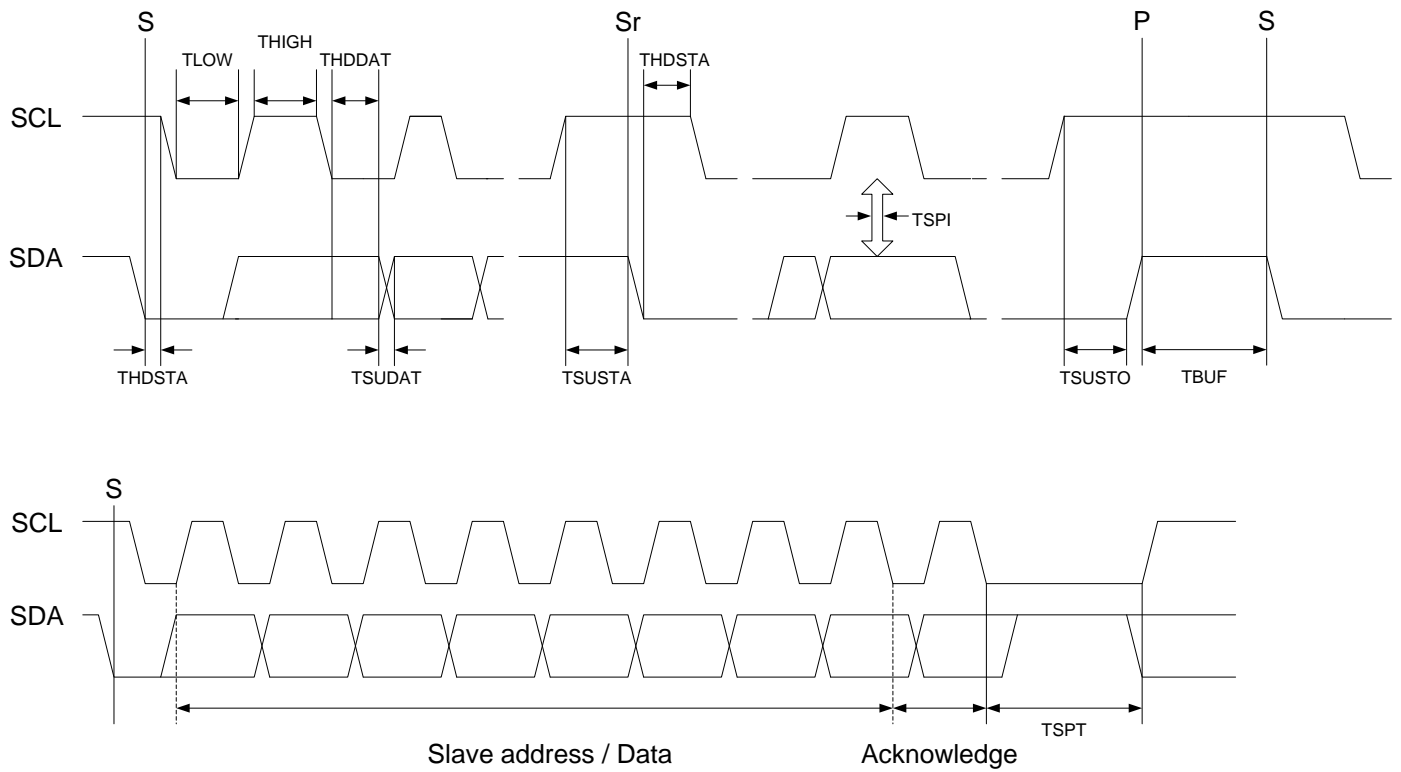


Figure8. Definition for timing for fast/standard mode on the IIC

Packet Stream

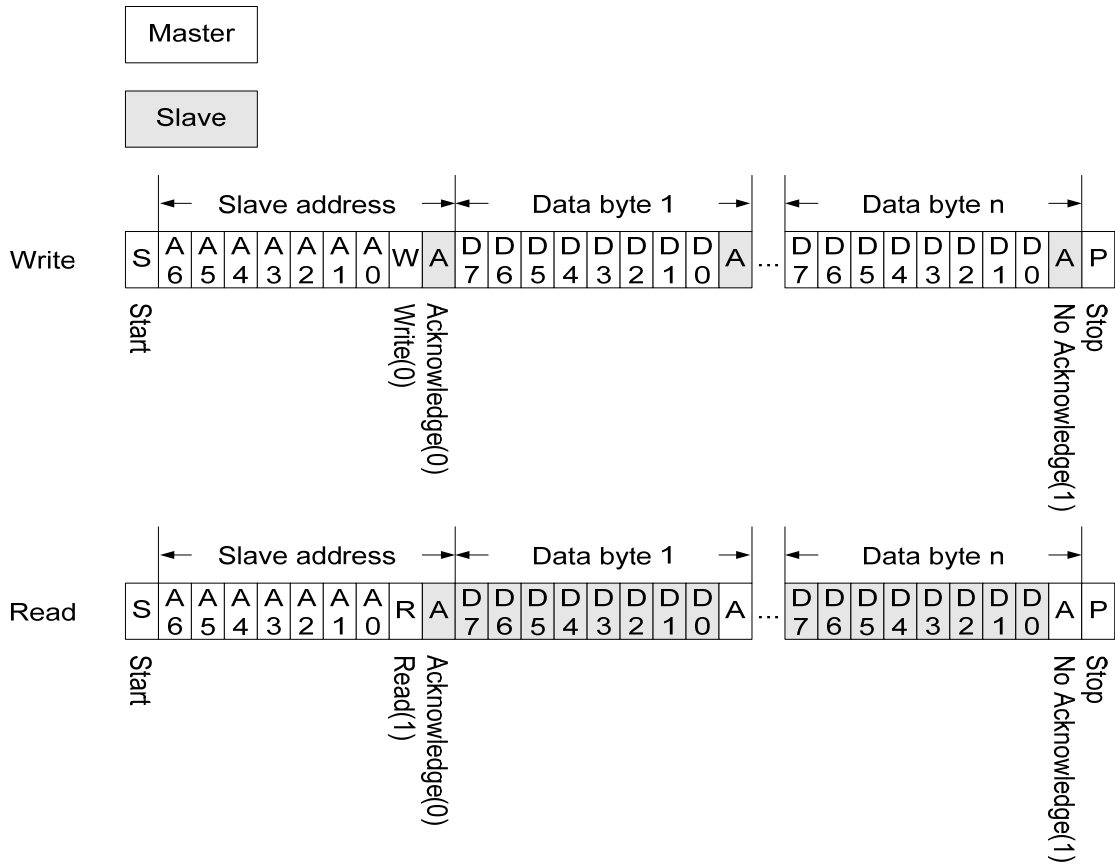


Figure9. Write / Read byte form I2C

Slave address

可透过 ADDR_SW1、ADDR_SW2 脚位做切换。如下表:

ADDR_SW1	ADDR_SW2	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

* ADDR_SW1、ADDR_SW2 空接时 Slave address 为 53H

Table 4. Slave address select

Data Stream:

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阈值(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。

Slide Application mode

Write Data

1. Setting commands

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			

IICM

IIC 数据模式选择。

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

CT

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

CT	Custom Threshold
0	Setting commands
1	Custom threshold commands

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)

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)

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

DT

动态阈值，功能开启时，滑调按键的阈值会依照按压位置自动调整。

DT	Dynamic Threshold
0	Disable (Define)
1	Enable

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

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

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)

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

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

2. Custom threshold commands

阈值则是设定按键承认的门坎。分为按键阈值，睡眠唤醒阈值两种。

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	-

● 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			

TPx Threshold：按键承认阈值。(Define：010H)

按键承认阈值越小灵敏度越高，越大灵敏度越低。预设的阈值为 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。

按键的期待值与阈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。

TP Num

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

● 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：省电模式唤醒阈值。(Define：002H)

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							

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

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

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

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

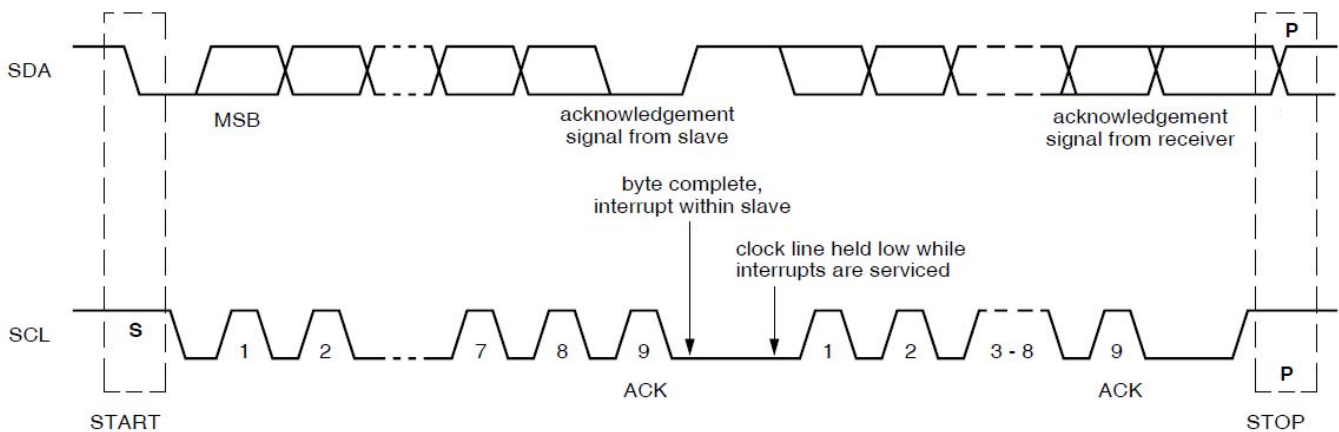
Sx Position

滑条位置标志，预设值为 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

● 特别说明:

1. TTY6955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，TTY6955会马上将SCL的bus拉Low，表示TTY6955进入busy的状态，同时TTY6955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. 按键阈值调整的步骤:
 - 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，需要调高阈值，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，需要降低阈值。

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够快，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，若按键反应速度不够快，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。

● 示范程序:

```

/*
项目名称:主控端对 TTY6955 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 TTY6955 写入设定参数
          2.透过软件模拟 IIC 主控端对 TTY6955 读取按键状态
主控 MCU:AT89C51
Date & Version: 2016/01/06 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()

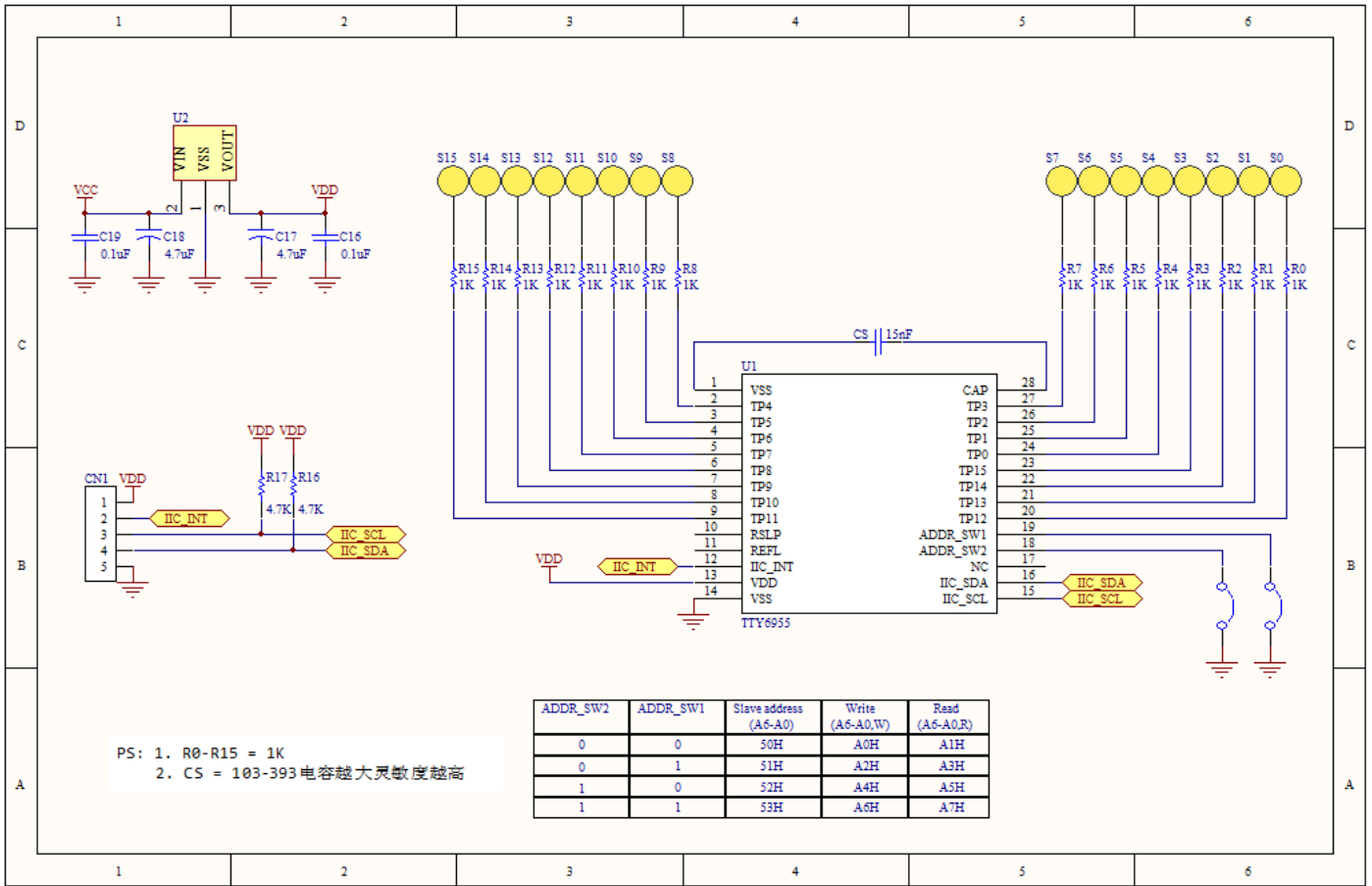
```

```
{
    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隔 30ms*/
        ACK = readIIC(address_R, &Read_Buffer, 6); //读取按键状态
    }
}
```

● **建议线路:**



R0-R15 1Kohm 是可以增强抗手机和对讲机的干扰, 一般情况是可以省略!

Cs 外接电容与压克力厚度关系: (预设阈值 010H)

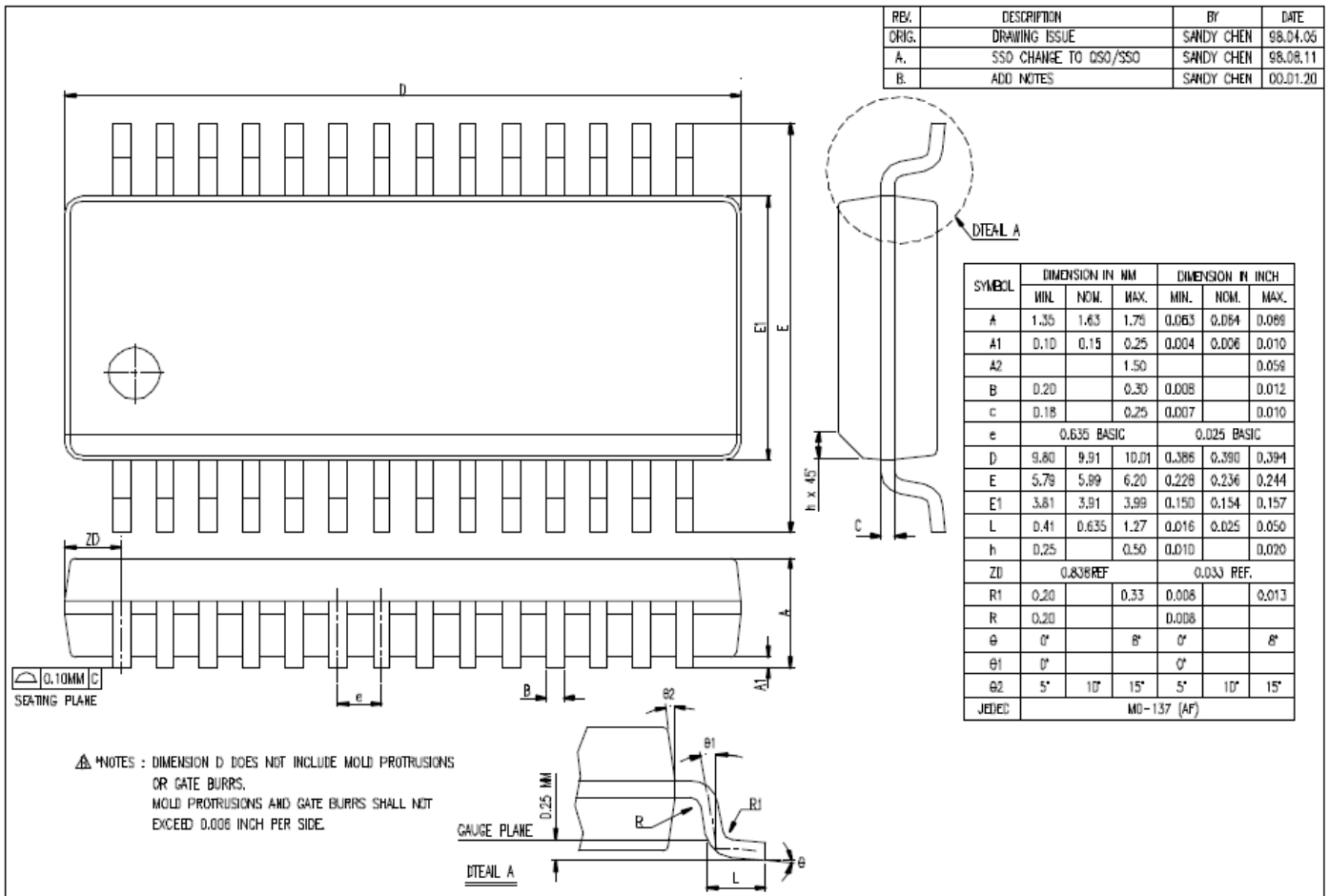
以铁片弹簧键, 圆型实心直径 12 MM 为例, 压克力厚度与 CS 电容的关系如下:

压克力厚度(mm)	CS
1	682
2	882
4	103
6	153
8	223
10	223

此表格仅供参考, 不同的 PAD 大小, PCB layout 皆会影响。

● 封装说明:

(28-SSOP)



订购信息

1. TTY6955

- a. 封装型号 : TTP259-ASFN

修订记录

1. 2015/10/26 - Version: 1.00
2. 2016/01/11 - Version: 1.10:
 - a. 增加示范程序
 - b. 增加 Cs 电容与亚克力之间的关系
3. 2016/03/10 - Version: 1.20:
 - 增加阈值调整相关说明