TTY6952 16Key + IIC
规格书 V1.1

● *产品描述:*

此方案提供客户一个简单的 1~16 键 IIC 输出应用。

● *特色:*

- 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。

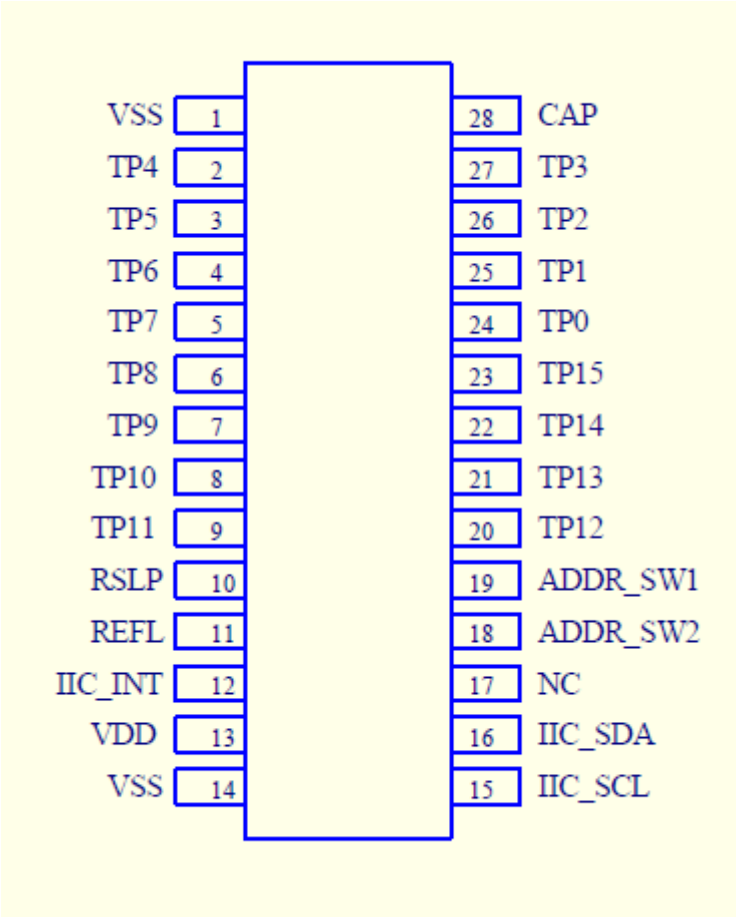
- 设计有省电模式，适合用在如遥控器等需要长时间待机的应用。

- 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:***



*未标示的脚位请保持空接

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，TTY6952 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:*

28pin	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	VREG	I	LDO voltage output.
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. TTY6952 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	℃
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				

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	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。

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。

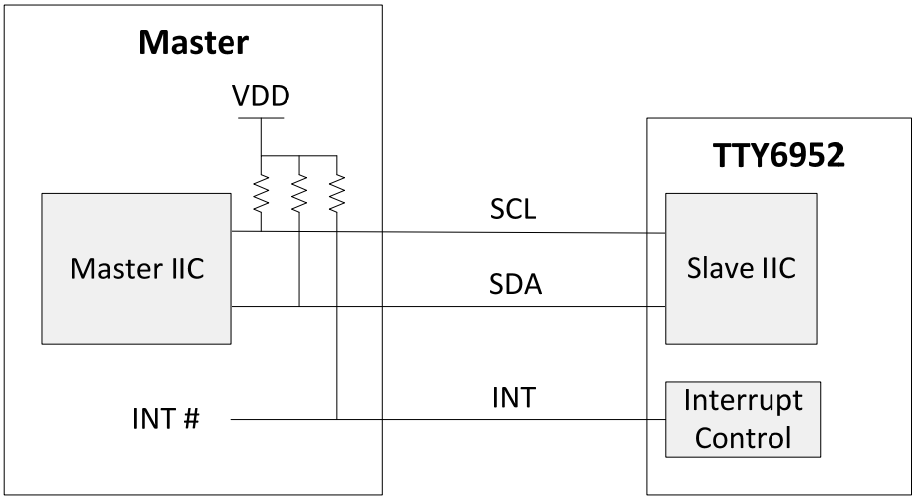


Figure6. IIC connect for master and TTY6952

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。

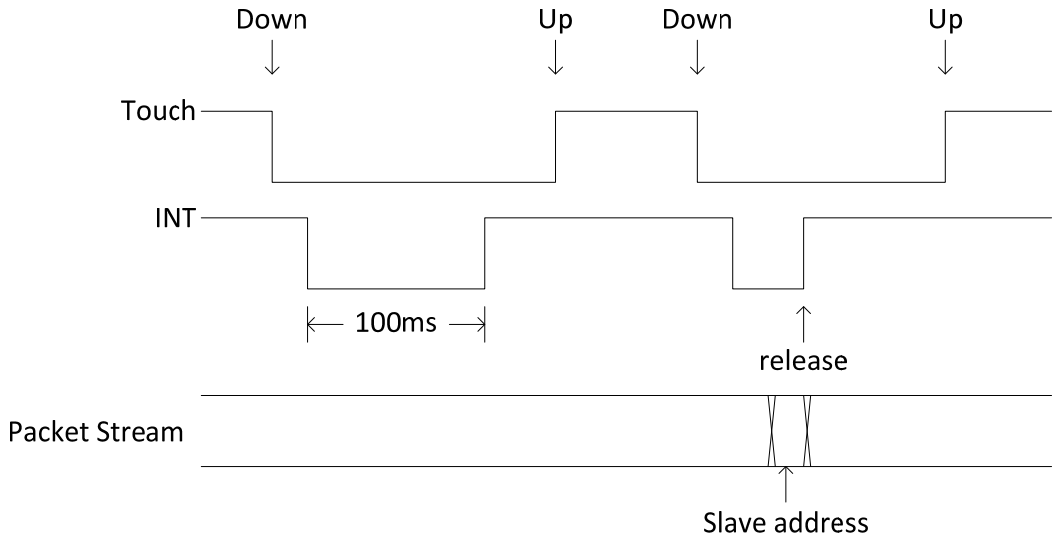


Figure7. INT pin describe

在 Slave Address、Data Byte 传送或接收的第 9 clock 结束时(下拉) ，Slave (TTY6952)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。

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

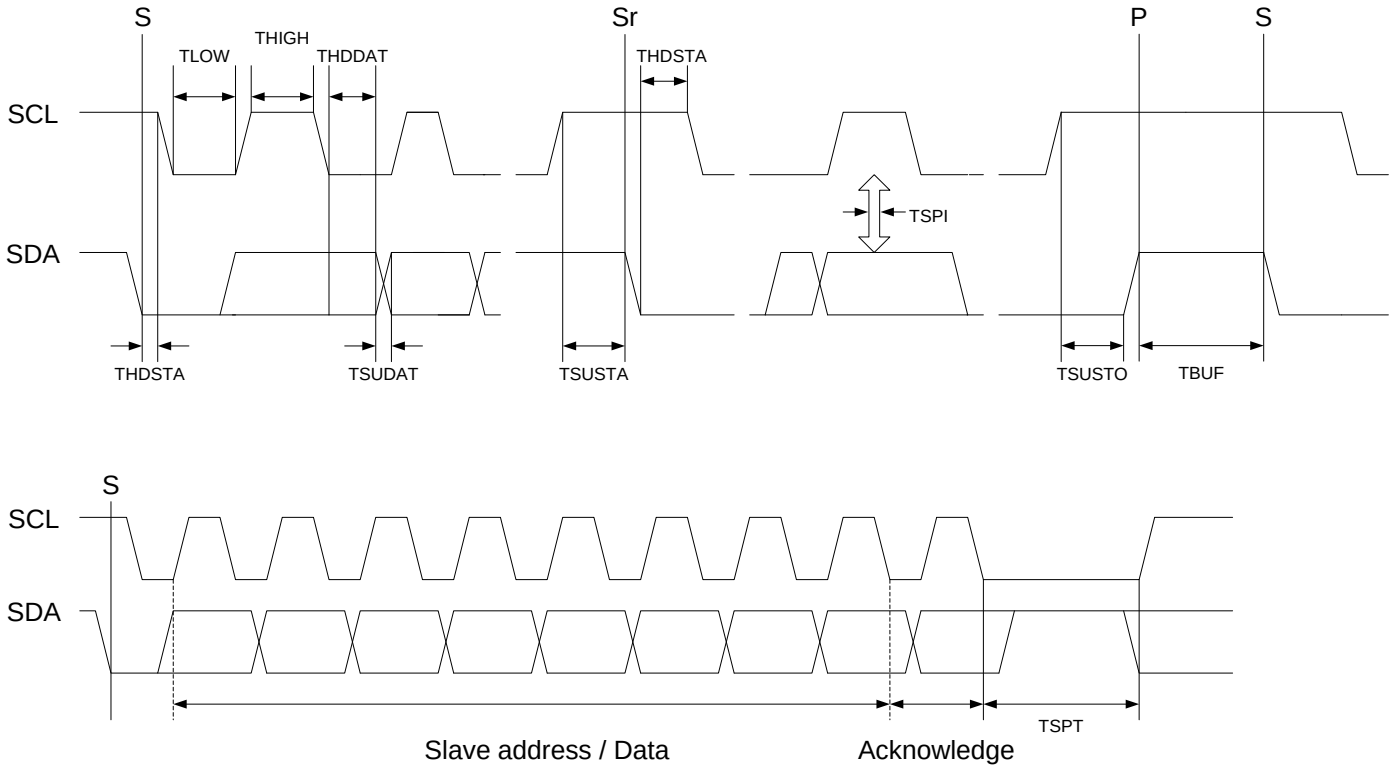


Figure8. Definition for timing for fast/standard mode on the IIC

Packet Stream

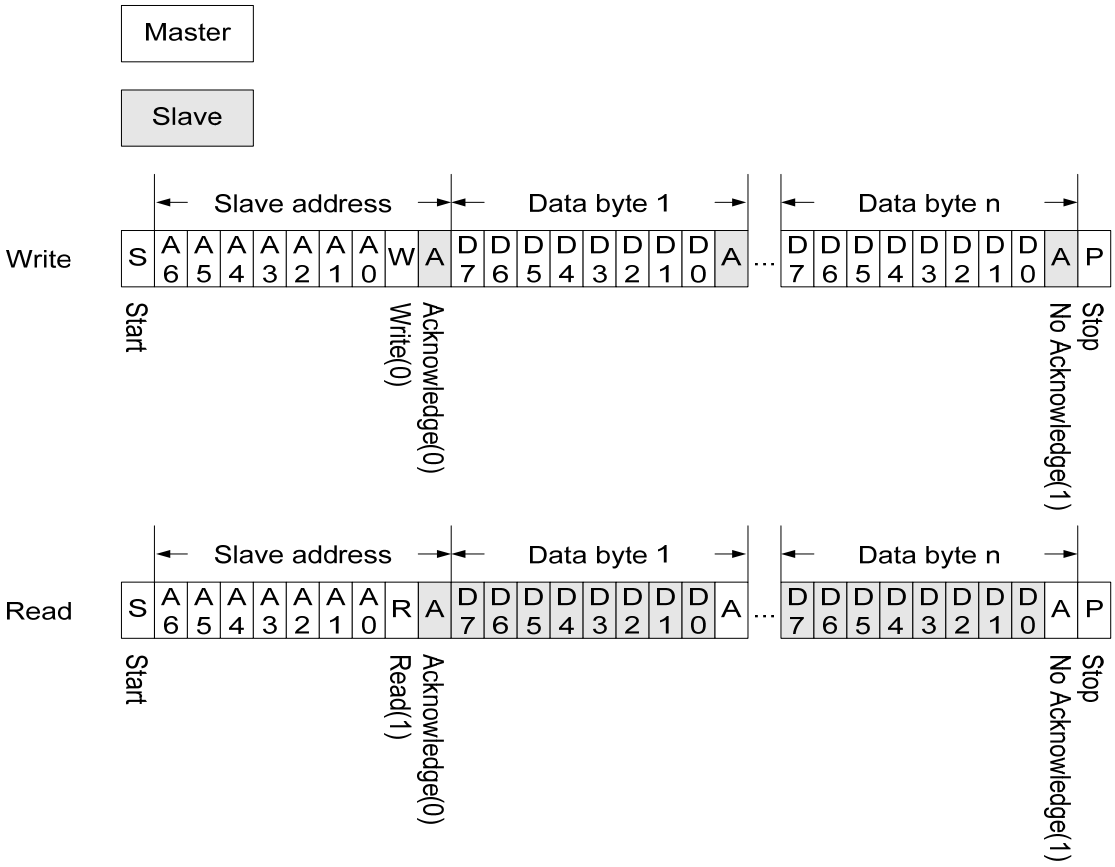


Figure9. Write / Read byte form I2C

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 **16 键应用模式**。PC Link 模式需要配合 USB PCLink Board 来读取触摸计数值，用以设定适当的期待值以及按压深度的模式。按键应用模式则可设定调整参数期待值以及承认值等设定，并读取按键输出。**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 16 键模式**。

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	-	ART	
2	Key Num					KAT		
3	Key Off Num							

IICM

IIC 数据模式选择。

IICM	**IIC Mode**
0	PC Link mode
1	Slide application mode

CT

在 Wheel 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

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

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

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

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
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
1	0	0	5 times
1	0	1	6 times
1	1	0	7 times
1	1	1	8 times

Key Num

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

Key Off Num

多按键重置设定，最多为 16 Keys。

Key Off Num				Key Off Number
3	2	1	0	
0	0	0	0	Disable
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

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。

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 : 按键承认阀值。

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 : 省电模式唤醒阀值。

Read Data

Data byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	C	WSET						
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

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

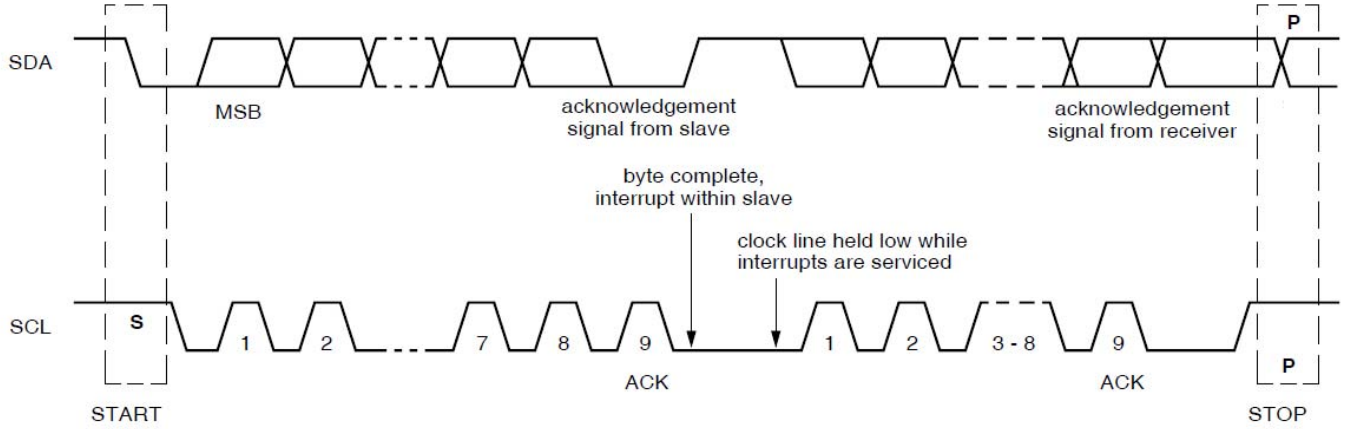
K1…K16

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

● *特别说明:*

1. TTY6952的I2C接口有硬件的支持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，TTY6952会马上将SCL的bus拉Low，表示TTY6952进入busy的状态，同时TTY6952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。
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。

- *示范程序:*

```
/*
    项目名称:主控端对 TTY6952 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 TTY6952 写入设定参数
             2.透过软件模拟 IIC 主控端对 TTY6952 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[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;
```

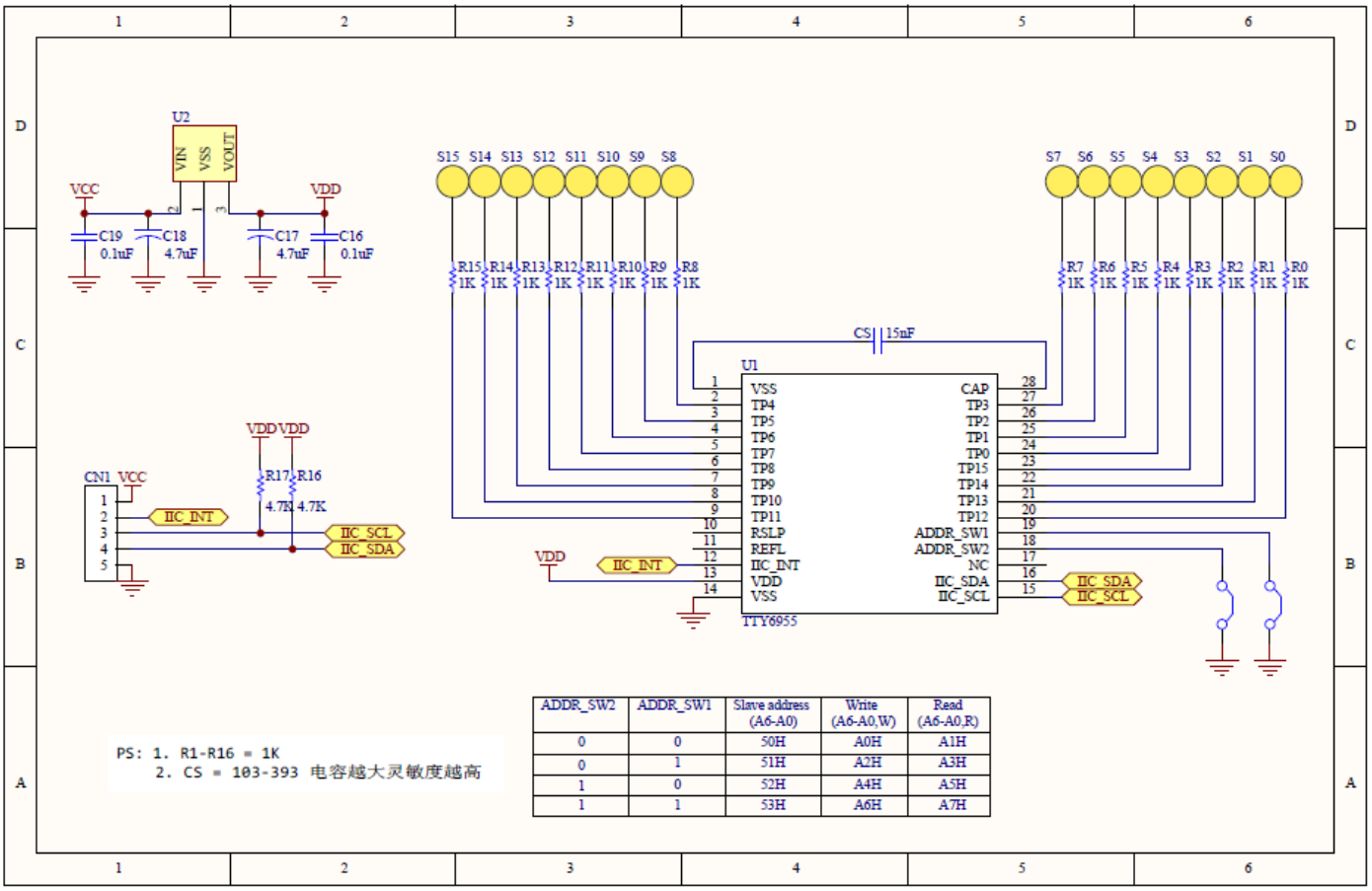
```
            SDA=1;
            _nop_(); /*如果需要在 SDA,SCL,INT 上串接电阻，根据电阻大小不同，电阻越大建议将该
时间适当加长，100KHZ 以内即可；*/
            dat<<=1;
            SCL=1; /*此处由于 51 单片机的特性不需要做输入输出设置，
但如果是其他单片机需要先将其 IO 口改为输入上拉的设置，读到高之后，SCL 转为输出为高。
在读写完 ACK 后的第一个 clock 下降缘从机会钳住 SCL 脚做资料处理，
所以将 SCL 脚置为输入上拉，并等待 SCL 被释放。*/
            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_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, 0x83, 0x00);
        ACK = writeIIC(address_W, &Write_Buffer, 3); //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, 3); //读取按键状态
    }
}
```

- ***建议线路:***



ADDR_SW2	ADDR_SW1	Slave address (A6-A0)	Write (A6-A0,W)	Read (A6-A0,R)
0	0	50H	A0H	A1H
0	1	51H	A2H	A3H
1	0	52H	A4H	A5H
1	1	53H	A6H	A7H

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：

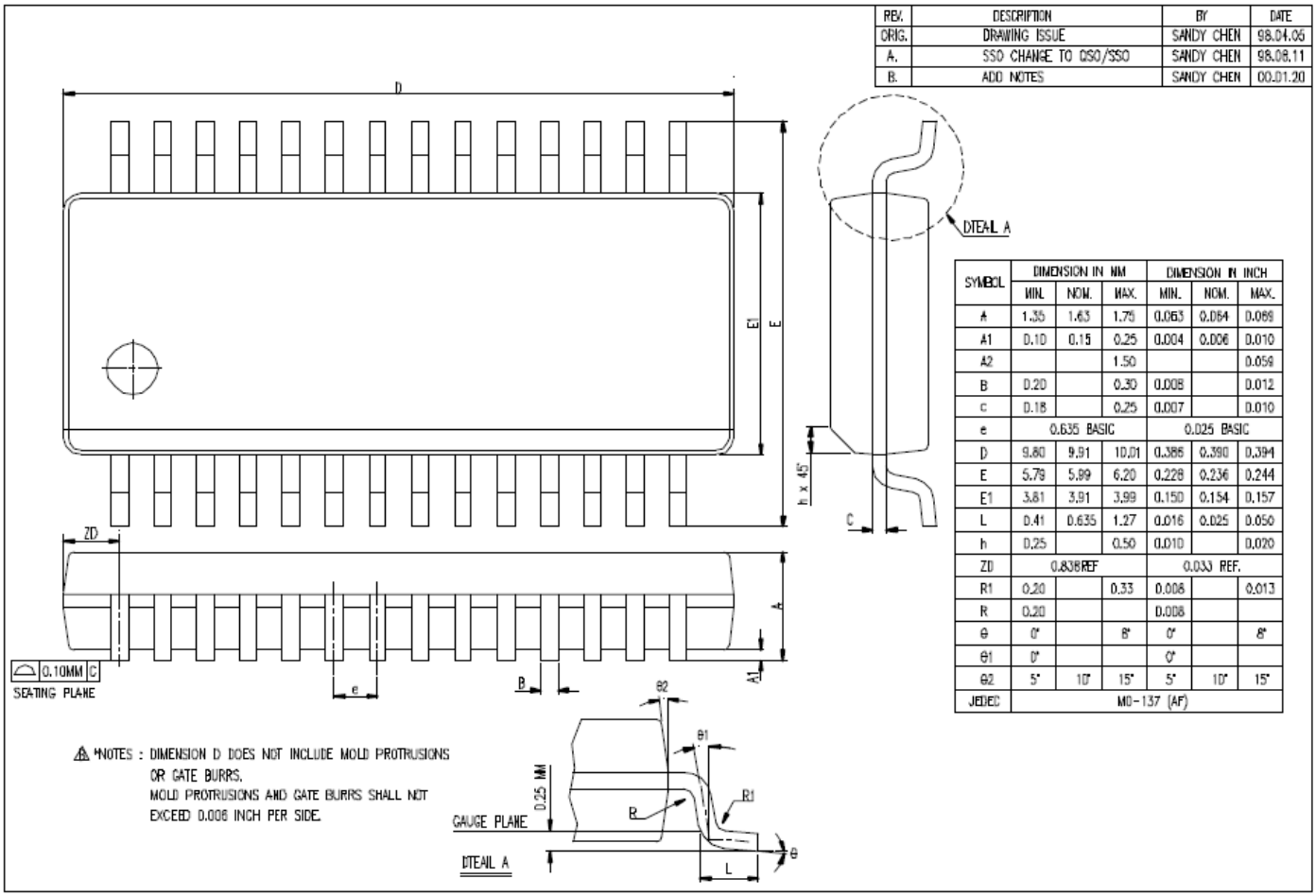
以铁片弹簧键，圆型实心直径 12 MM 为例，压克力厚度与 CS 电容的关系如下:

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）



SYMBOL	DIMENSION IN MM			DIMENSION IN INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	1.35	1.63	1.78	0.053	0.064	0.069
A1	0.10	0.15	0.25	0.004	0.006	0.010
A2			1.50			0.059
B	0.20		0.30	0.008		0.012
c	0.18		0.25	0.007		0.010
e	0.635 BASIC			0.025 BASIC		
D	9.80	9.91	10.01	0.386	0.390	0.394
E	5.79	5.99	6.20	0.228	0.236	0.244
E1	3.81	3.91	3.99	0.150	0.154	0.157
L	0.41	0.635	1.27	0.016	0.025	0.050
h	0.25		0.50	0.010		0.020
ZD	0.838REF			0.033 REF.		
R1	0.20		0.33	0.008		0.013
R	0.20			0.008		
θ	0°		8°	0°		8°
θ1	0°			0°		
θ2	5°	10°	15°	5°	10°	15°
JEDEC	MO-137 (AF)					

订购信息

1. TTY6952
   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