TTY6952 16 Keys + IIC

Technical Specifications V1.2

Contents

General Description

This solution provides customers with a simple 1-16 keys IIC output application.

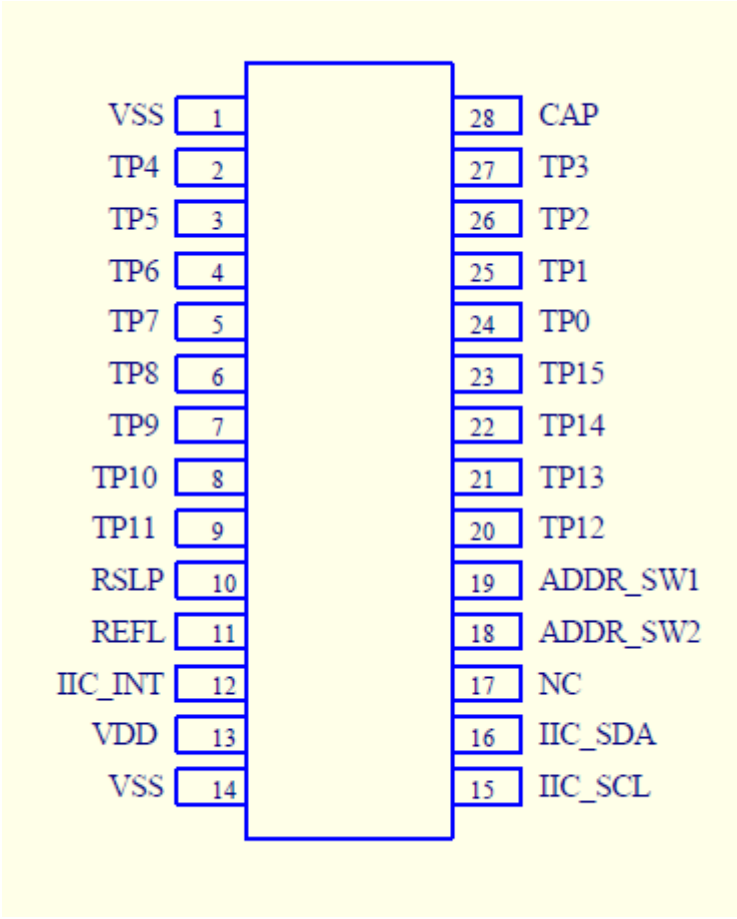
Features

- There are two ways to set parameters. Users can adjust the sensitivity parameter of slider keys with a USB PCLink Board and use IIC commands to modify and set parameters.
- There are two output modes for independent keys : multiple and single. When selecting “multiple”, all pressed keys will output signals at the same time. When “single” is selected, only the first pressed key will output signals. Other keys will output signals in turn after the previous one is released. That is, one key at a time.
- The power-saving mode (PSM) is suitable for use on remote controls and other devices requiring long standby time.
- Four slave addresses for external on/off control.

Application

- All kinds of home appliances.
- Access control devices
- Consumer electronics

- **Pin Package Diagram**



*Leave not indicated pins unconnected.

Figure 5 Pin definition for IC package

- CAPN and CAPN are for measuring capacitance: approx.10μF-39μF.
- TP0-TP15 are for measuring touch pads, the TTY6952 detects up to 16 keys.
- RSLP is the output in PSM. It normally maintains at High and reduces to Low in PSM.
- REFL is the ambient value latch input. It maintains at High when is not connected. When power is low, it stops updating ambient values.
- ADDR_SW1 and ADDR_SW2 are the Slave ID options of the IIC. Setup will be described below.
- IIC_SDA is the data I/O pins of the IIC.
- IIC_SCL is the frequency input pin of the IIC.

Pin Description

Pin No	Pin Name	Type	Pin Definition
-	RSTB	I	External reset input, active low 50kΩ pull-up (5v)
13	VDD	Power	Positive power supply
14	VSS	Power	Negative power supply, ground
28	CAP	I	Touch sensor input
1	VSS	Power	
17	VREG	I	LDO voltage output.
12 11 10 -	IIC_INT REFL RSLP -	IO I O -	IIC interrupt pin Reference Lock Read Sleep enter
15 16 18 19	IIC_SCL IIC_SDA ADDR_SW1 ADDR_SW2	IO IO I I	IIC clock pin IIC data pin IIC slave address select IIC slave address select
24 25 26 27	TP0 TP1 TP2 TP3	IO/I IO/I IO/I IO/I	touch pad input touch pad input touch pad input touch pad input
2 3 4 5	TP4 TP5 TP6 TP7	IO/I IO/I IO/I IO/I	touch pad input touch pad input touch pad input touch pad input
6 7 8 9	TP8 TP9 TP10 TP11	IO/I IO/I IO/I IO/I	touch pad input touch pad input touch pad input touch pad input
20 21 23 24	TP12 TP13 TP14 TP15	IO/I IO/I IO/I IO/I	touch pad input touch pad input touch pad input touch pad input

Table 1 TTY6952 Pin Description

- **AC/DC Characteristics**

1. Absolute maximum ratings

Item	Symbol	Condition	Rating	Unit
Operating Temperature	Top	——	-40~+85	°C
Storage Temperature	T_{STG}	——	-50~+125	°C
Supply Voltage	VDD	Ta=25°C	VSS-0.3~VSS+5.5	V
Input Voltage	V_{IN}	Ta=25°C	VSS-0.3~VDD+0.3	V
Human Body Model (HBM)	ESD	——	>5	KV
Note: VSS represents ground.				

2. DC/AC Characteristics (condition : room temperature=25°C)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Unit
Operating voltage	VDD		2.5	-	5.5	V
System clock	F	VDD=5V	-	4M	-	Hz
Operating current	I_{OP}	Standby, VDD=3V, no output load	-	1.1	-	mA
	I_{OFF}	Standby, VDD=3V, no output load	5.3	6.8	10.0	μA

- **Features**

  Touch Keys

  - Touch keys judge the value change based on the capacitance change occurred when the human body approaches a conductor. Therefore, we can adjust touch sensitivity by adjusting the **threshold** value of each key.
  - The threshold value is adjusted according to the depth of pressing, and the smaller the value, the higher the sensitivity. However, this will expose to higher noise interference. Users should adjust the threshold value of touch keys with a USB PCLink Board based on the actual press data.

IIC Protocol

The IC transfers data with the IIC protocol and reads and writes data with a two-wire bus: SCL and SDA. The INT pin byte is used to notify Master of a press state change.

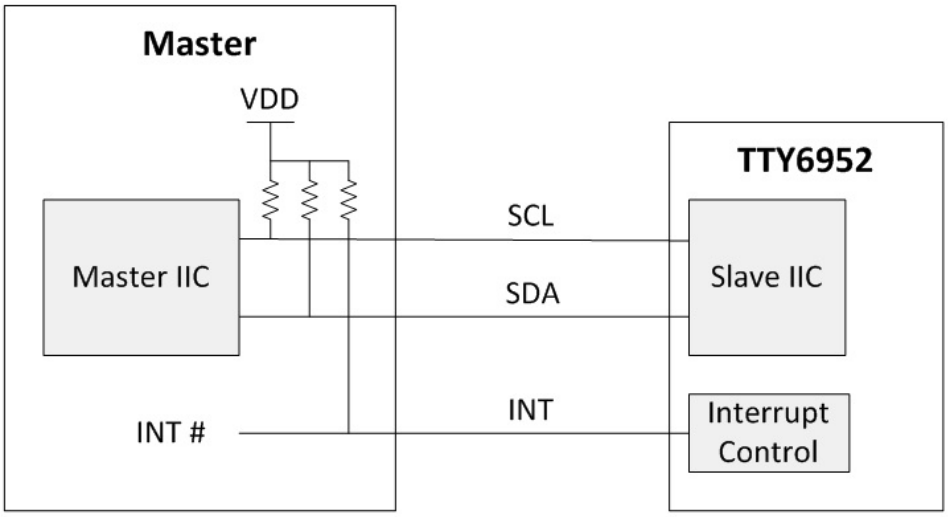


Figure 6 IIC connection for Master and TTY6952

When the key state is unchanged, INT output is high. When the key state changes, the INT output will reduce to Low for 100ms. After receiving a slave address, the Slave will clear and reset output to High.

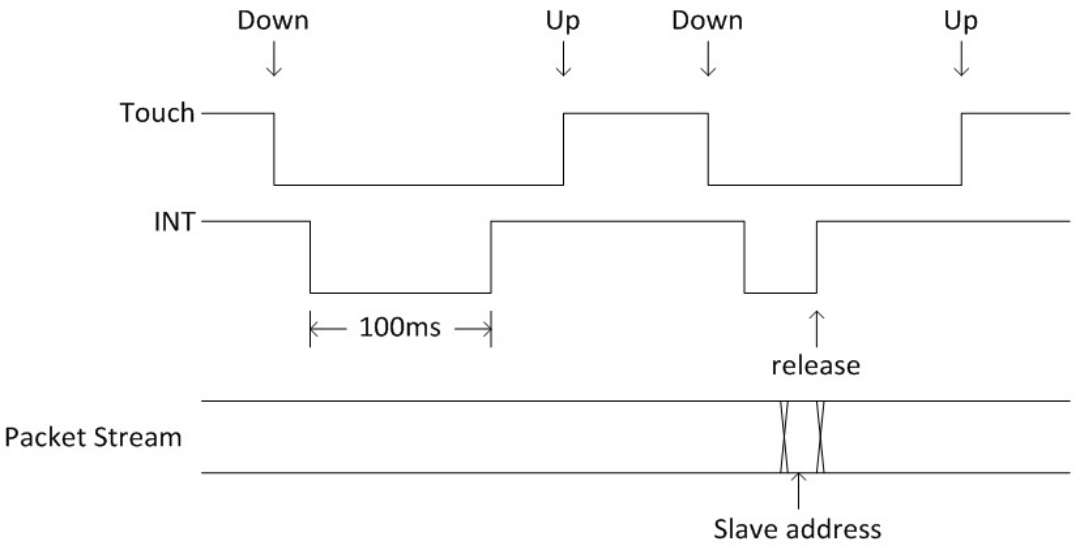


Figure 7 INT pin description

When clock 9 ends the transfer or receipt of a slave address and data byte (output reduces), Slave (TTY6952) will reduce output for 20-100μs to process data and release SCL after completing data processing. Therefore, Master must wait for SCL release to continue writing or reading data.

The simple setup is, after Master increases SCL output, read data and wait for SCL output to increase. Or, when clock 9 ends, output reduces and holds for 100μs.

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	us
TSPT	Slave processor time	10	75	us

Table 3 AC characteristics of the IIC SDA and SCL pins for vdd

Timing Waveform

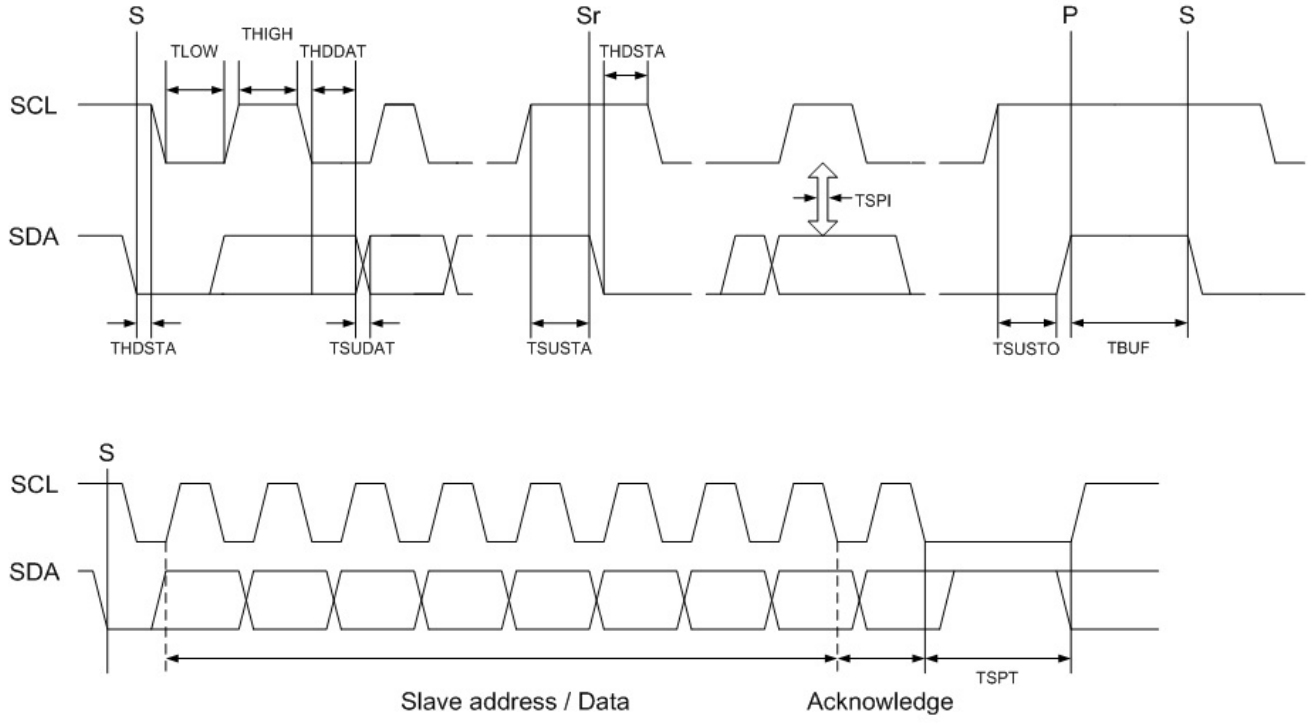


Figure 8 Definition of timing for fast/standard mode on the IIC

Packet Stream

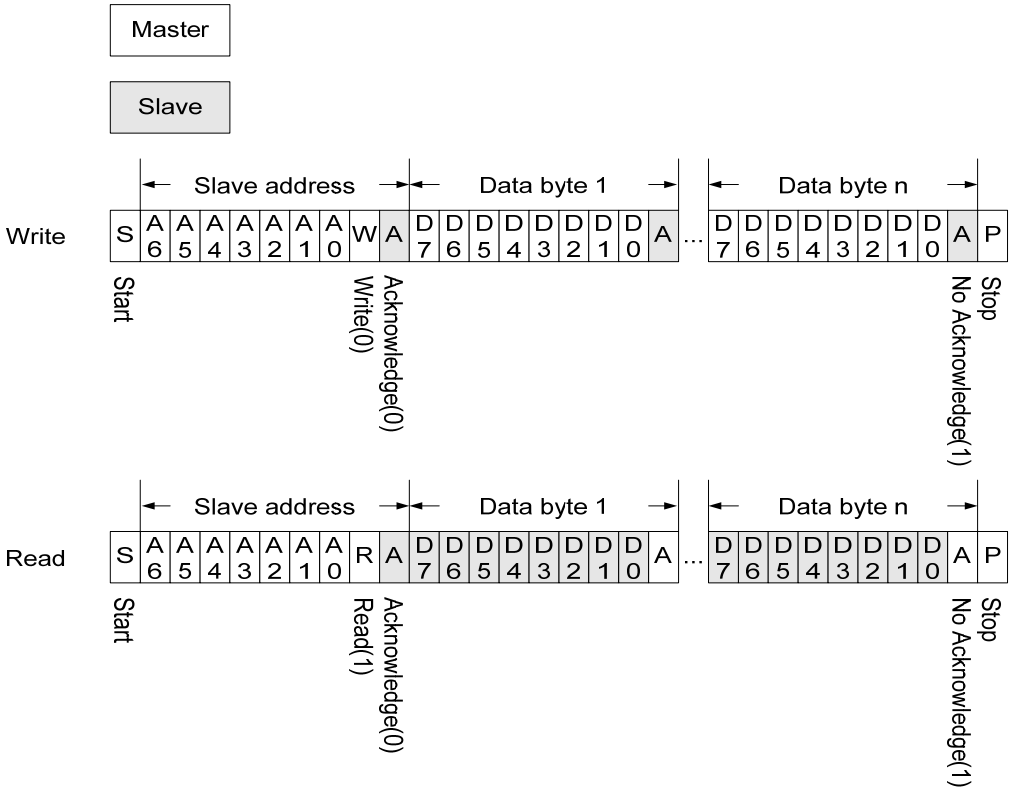


Figure 9 Write/read byte form I2C

Slave address

Switch from pins ADDR_SW1 and ADDR_SW2 as shown below :

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

* Slave address is 53H when ADDR_SW1 and ADDR_SW2 are not connected.

Table 4 Slave Address Selection

Data Stream :

There are two operating modes for software programming: **PC Link** and **16-key application modes**. The PC Link mode requires a USB PCLink Board to read the touch count value for setting the appropriate expected value and press depth mode. The key application mode enables adjustment of the expected value and acknowledgement value of parameters and reads key outputs. **If bit 7 of the first written date byte is "0", the system is set to the PC Link mode. If bit 7=1, it is the 16-key mode.**

In the 16-key application mode, each write data stream consists of three data bytes. After writing a data stream, the system will cover the data and **reset the system**. If writing is interrupted and re-does, the previous data stream will be neglected.

After it needs to write another data stream after finishing one, it needs to send a stop signal to end the transfer of the previous data stream before writing the new data stream.

16 Key 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	-	ART	
2	Key Num					KAT		
3	Key Off Num							

IICM

IIC data mode selection

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

CT

In the wheel application mode, data writing blocks are divided into application setup and threshold setup. When CT=0, it is write application setup. When CT=1, it is write threshold setup

CT	Custom Threshold
0	Setting commands
1	Custom threshold commands

KOM

There are two options for the key output mode: Multiple and Single modes. These options are for configuring the key output settings, and slider keys are unaffected. When “Single” is selected, only the signal of the first pressed key will be sent, and other keys will be acknowledged after the first pressed key is released.

KOM	Key Output Mode
0	Multiple
1	Single (Define)

AA

Auto adjustment (AA) of base value : When no key is pressed, base value is updated automatically.

AA	Auto Adjust
0	Disable
1	Enable (Define)

PSM

PSM : Enters the sleep mode when no key is pressed after four seconds.

PSM	Power Save Mode
0	Disable
1	Enable (Define)

ART

Auto reset time (ART) setup : Timeout countdown starts when the key position is unchanged and resets automatically when time is up.

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

KAT

Key acknowledge times.

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

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)

Key Off Num

Multiple keys reset, up to 16 keys.

Key Off Num				Key Off Number
3	2	1	0	
0	0	0	0	Disable (Define)
0	0	0	1	2 key 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

Threshold setup enables users to configure the key acknowledgment threshold, including the key threshold and wakeup threshold.

Item

Select a parameter for setup.

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 : Key acknowledgement threshold. (Define : 010H)

In key acknowledgement threshold, **the smaller the value, the higher the sensitivity**, and the greater the value, the lower the sensitivity. The default threshold value is 010H. The recommended smallest value is 008H. If sensitivity is still not high enough at 008H, increase CS capacitance. Recommended CS capacitance is smaller than 39nF.

TP Num

Data write key number.

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 : PSM wakeup threshold value. (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						
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

System calibration mark : When the value=0, calibration is in progress, and key value read is invalid. When value=1, key is valid.

C	Calibrate
0	Calibrating
1	Calibrate Finish

WSET

System write mark : Value=1 when power is on; value=0 after writing in settings.

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

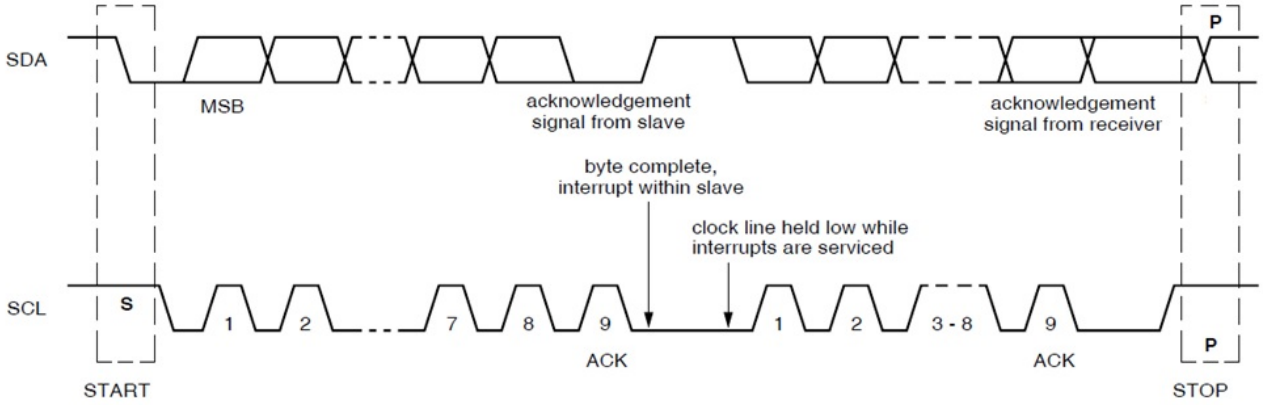
K1…K16

Touch key mark : “0” for none and “1” with keys.

K1…K16	Key1…Key16
0	No touch
1	Touch

- **Precautions**

1. The I2C interface of the TTY6952 supports hardware SCL up to 100KHz, but it needs a software decoder. Therefore, when SCL 9 of Master is low, the TTY6952 will immediately reduce SCL bus to low. This means the TTY6952 enters a busy state. Meanwhile, the TTY6952 will generate a signal to interrupt I2C decoding. The processing time is about 20-100μs, depending on the situation. After finishing processing, it will release SCL. In general, the Master's SCL pin is Nmos output and needs additional raise impedance to ensure Master can raise SCL to High.



Hardware control usually supports this standard. When controlling the I/O pin with a program, please add a command to confirm SCL output high to continue the next step. If SCL output is Low, wait for SCL output changes to High before continue the next step. The C program is as follows :

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

2. If it needs to continuously read key values, users are recommended to hold for at least 10ms between reading the value between keys. Otherwise, this will affect key response time.
3. If the sleep mode is enabled, continuous key value reading is prohibited, because each time of reading will clear the sleep mode timeout, and the system will not be able to enter the sleep mode.
4. The IIC function will shut down after the system enters the sleep mode. At this moment, give a new command to IIC to wake up the system. However, this will receive a no ACK response. Wait for system wakeup to give a new read/write command..
5. Key threshold adjustment procedure :

   Step 1 Select CS capacitance for initializing the test (see the recommended layout) First, confirm is the slider function is enabled. If it is enabled, users are recommended to use a 33nF capacitor for CS discharge. For general key test, use a 10nF capacitor to initialize the test.

Step 2 Press test for each key :
When touching a key at a normal speed or with a metal rod, if there is output before touching the key, this means sensitivity is too high and it needs to adjust the threshold value. If there is no output after touching the key or there is output when users need to press the key harder, this means sensitivity is too low and it needs to adjust the threshold value.
As a slider is notched, sensitivity varies with notches. In a sensitivity test, users are recommended to run the sensitivity test with two keys set at the center of the slider to prevent poor sliding effect.

Step 3 Key response time test :
When judging key sensitivity, if the key is “not sensitive enough,” users will need to judge if key response is not fast enough. The method is to press and hold a key for some time (approx. 1 second) and check if there is output from the key. If there is no output, key sensitivity is low and repeat Step 2 to adjust key sensitivity. If there is output, this means the key response time is not fast enough. Run the next step to adjust.

Step 4 Key response time adjustment

- If default KAT is “4” and the key response time is not fast enough, adjust KAT to “3”.
- If the key response time is still not fast enough after adjusting KAT to “3”, please reduce CS capacitance.
- After selecting appropriate CS capacitance, repeat Step 2 to adjust sensitivity.
- Please note that lower CS capacity will reduce slider accuracy.

- **Demo Procedures**

```
/*
        Item: Demo program for Master control of the TTY6952 through IIC
        Purpose:
        1. To write settings in the TTY6952 from Master with software analog IIC.
        2. To read key state of the TTY6952 from Master with software analog IIC.
        Master 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 //Slave address and writing mark
#define address_R 0xa7 // Slave address and reading mark

sbit SINT=P0^0; //IIC interface on Master and Slave
sbit SDA=P0^1; // IIC interface on Master and Slave
sbit SCL=P0^2; // IIC interface on Master and Slave
uchar Write_Buffer[3]; //Master write buffer
uchar Read_Buffer[3]; //Master read buffer
//-----------------------------------------------------------------------------------------------
//Function name: void delay(uint x)
//Function functions: Program delay
//Function input: x
//Function output: None
// Intermediate variable: i, j
//-----------------------------------------------------------------------------------------------
void delay(uint x)
{
    uint i,j;
    for(i=x;i>0;i--)
        for(j=0x40;j>0;j--);
}
//-----------------------------------------------------------------------------------------------
//Function name: void sendStart()
//Function functions: IIC start position
//Function input: None
//Function output: None
// Intermediate variable: None
//-----------------------------------------------------------------------------------------------
void sendStart() //start position
{
    SDA=1; /*Send initialization condition data signal*/
    SCL=1;
    while(SCL!=1) { };
    SDA=0; /* Send initialization signal*/
    _nop_();
    SCL=0; /*At this position, only set SCL=0 and wait for 4μs*/
}
//-----------------------------------------------------------------------------------------------
//Function name: void sendStop()
```

```
//Function functions: IIC stop position
//Function input: None
//Function output: None
// Intermediate variable: None
//-----------------------------------------------------------------------------------------------
void sendStop() //Stop position
{
	SCL=0;
	SDA=0; /* Send ending condition data signal */
	_nop_();
	SCL=1;
	while(SCL!=1) { };
	_nop_();
	SDA=1;
}
//-----------------------------------------------------------------------------------------------
//Function name: bit readACK()
//Function functions: Read IIC acknowledgement mark byte
//Function input: None
//Function output: IIC ACK signal response=1 means not acknowledged and ACK signal
response=0 means acknowledged.
// Intermediate variable: None
//-----------------------------------------------------------------------------------------------
bit readACK() //Read ACK signal
{
	SCL=0;
	SDA=1; /*Release the SDA bus by sending a low output level from Slave*/
	_nop_();
	SCL=1;
	_nop_();
	if(SDA)
		return 1; //no ACK
	else
		return 0; //ACK
}
//-----------------------------------------------------------------------------------------------
//Function name: void sendACK()
//Function functions: Master sends an acknowledgement signal
//Function input: None
//Function output: None
// Intermediate variable: None
//-----------------------------------------------------------------------------------------------
void sendACK() //Output an acknowledgement signal
{
	SCL=0;
	SDA=0;
	_nop_();
	SCL=1;
}
//-----------------------------------------------------------------------------------------------
//Function name: void sendNOACK()
//Function functions: Master sends a no acknowledgement signal
//Function input: None
//Function output: None
```

```
// Intermediate variable: None
//-------------------------------------------------------------------------------------------------
void sendNOACK() //Output a no acknowledgement signal
{
    SCL=0;
    SDA=1;
    _nop_();
    SCL=1;
}
//-------------------------------------------------------------------------------------------------
//Function name: void sendByte(uchar dat)
//Function functions: Master writes a byte to Slave.
//Function input: dat = sent byte
//Function output: None
// Intermediate variable: i
//-------------------------------------------------------------------------------------------------
void sendByte(uchar dat) //Write a byte
{
    uchar i;
    for(i=0;i<8;i++)
    {
        SCL=0; /*Clamp the I2C bus and prepare to send data */
        if(dat&0x80)
            SDA=1;
        else
            SDA=0;
        _nop_(); /*When connecting resistors to SDA, SCL, and INT, extend time appropriately
according to impedance: the greater the impedance, the longer the time, within 100KHz;*/
        _nop_();
        SCL=1; /*No I/O setup is required due to the characteristics of the 51 single chip
machine. For other single chip machines, however, change the I/O port to increase the input level.
After reading High, SCL will change output to High. After reading ACK, clock 1 will reduce from
Slave to clamp the SCL pin for data processing. Therefore, set SCL pin to output High and wait for
SCL release.*/
        while(SCL!=1) { };
        dat<<=1;
    }
}
//-------------------------------------------------------------------------------------------------
//Function name: uchar readByte()
//Function functions: Master reads a byte from Slave.
//Function input: None
//Function output: Read completed byte stream.
// Intermediate variable: i, dat
//-------------------------------------------------------------------------------------------------
uchar readByte() //Read a byte stream
{
    uchar i, dat=0;
    for(i=0;i<8;i++)
    {
        SCL=0;
        SDA=1;
        _nop_(); /*When connecting resistors to SDA, SCL, and INT, extend time appropriately
according to impedance: the greater the impedance, the longer the time, within 100KHz;*/
```

```
            dat<<=1;
            SCL=1; /*No I/O setup is required due to the characteristics of the 51 single chip
machine. For other single chip machines, however, change the I/O port to increase the input level.
After reading High, SCL will change output to High. After reading ACK, clock 1 will reduce from
Slave to clamp the SCL pin for data processing. Therefore, set SCL pin to output High and wait for
SCL release.*/
            while(SCL!=1) { };
            if(SDA==1)
            dat|=0x01;
      }
      return dat;
}
//-----------------------------------------------------------------------------------------------
//Function name: bit writeIIC(uchar addrW, uchar *writeData, uchar length)
//Function functions: Master writes data to Slave.
//Function input: addrW = Slave address and write flag.
                  *writeData = the first address of data ready for writing.
                  length = data length (bytes)
//Function output: Return to the acknowledge state of IIC communication. If it is “1”, it stops and
returns. If it is “0”, it completes communication and returns.
// Intermediate variable: i, ACK
//-----------------------------------------------------------------------------------------------
bit writeIIC(uchar addrW, uchar *writeData, uchar length)
{
      uchar i;
      bit ACK;
      sendStart();
      sendByte(addrW); //Send address and write mark.
      ACK = readACK();
      if (ACK)
      {
            sendStop(); //If the address is incorrect or no device is connected, send the Stop signal.
            return ACK;
      }

      for(i = 0; i<length; i++)
      {
            sendByte(writeData[i]);
            ACK = readACK();
            if (ACK)
            {
                  sendStop(); //If the ACK signal is not received, send the Stop signal.
                  return ACK;
            }
      }
      sendStop(); //After data writing is completed, send the Stop signal.
      return ACK;
}
//-----------------------------------------------------------------------------------------------
//Function name: bit readIIC(uchar addrR, uchar *readData, uchar length)
//Function functions: Master reads data from Slave.
//Function input: addrW = Slave address and read flag.
                  *writeData = the first address for storing the read data.
                  length = data length (bytes)
```

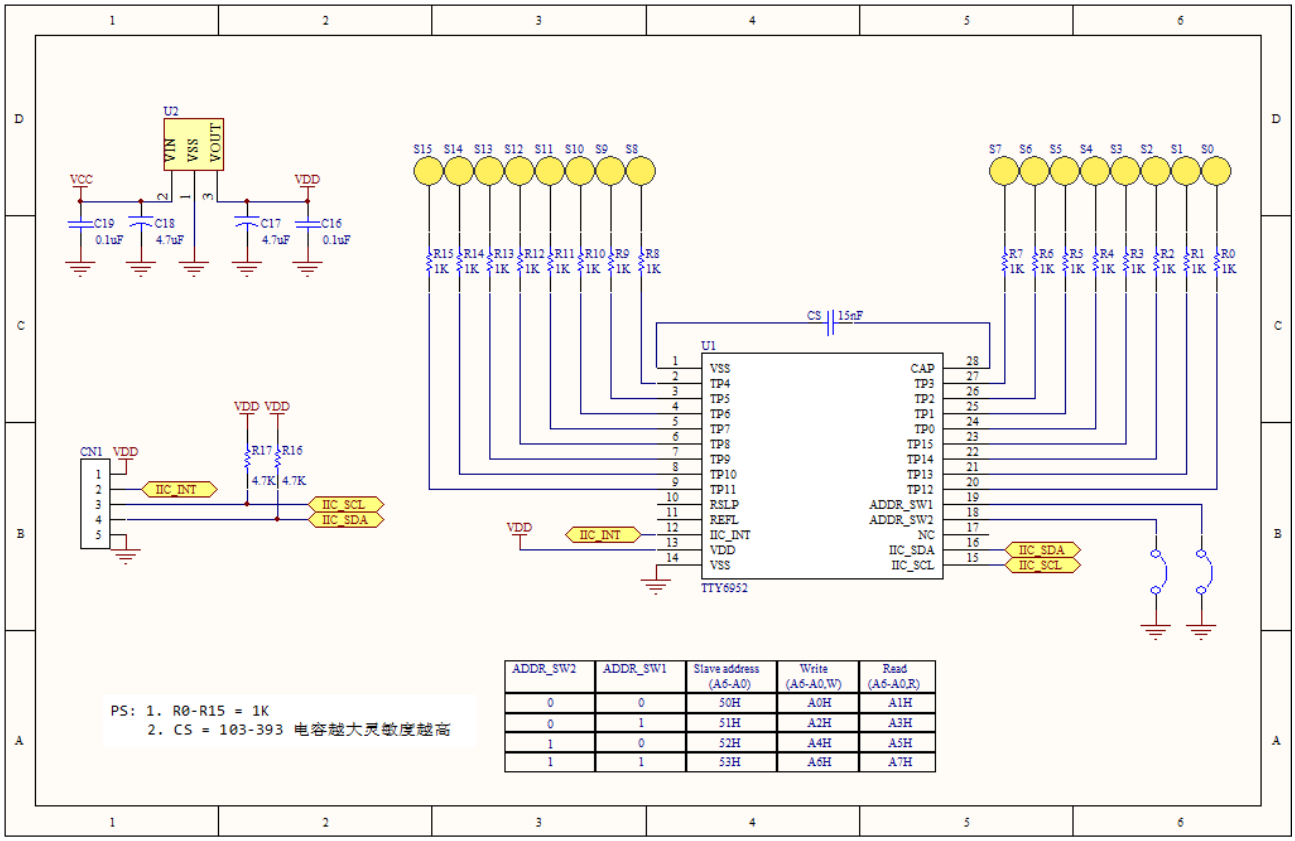
```
//Function output: Return to the acknowledge state of IIC communication. If it is “1”, it stops and
returns. If it is “0”, it completes communication and returns.
// Intermediate variable: i, ACK
//-----------------------------------------------------------------------------------------------
bit readIIC(uchar addrR, uchar *readData, uchar length)
{
    uchar i;
    bit ACK;
    sendStart();
    sendByte(addrR); //Send address and read mark.
    ACK = readACK();
    if (ACK)
    {
        sendStop(); //If the address is incorrect or no device is connected, send the Stop signal.
        return ACK;
    }

    for(i = 0; i<length; i++)
    {
        readData[i] = readByte();
        if(i<length-1)
            sendACK();
        else
            sendNOACK(); //Read the last piece of data and send No ACK.
    }
    sendStop(); //After data writing is completed, send the Stop signal.
    return ACK;
}
//-----------------------------------------------------------------------------------------------
//Function name: void setWrite_Buffer_3(uchar byte1, uchar byte2, uchar byte3)
//Function functions: Write 3 bytes to the buffer.
//Function input: byte1
                  byte2
                  byte3
//Function output: None
// Intermediate variable: None
//-----------------------------------------------------------------------------------------------
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) /*Wait for read request. No need to read SINT in PSM. Recommended interval
for each time of reading is 30ms*/
            ACK = readIIC(address_R, &Read_Buffer, 3); //Read key state
    }
}
```

- **Recommended Layout**



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

PS :

1. R0-R15 = 1K
2. CS = 103-393, the greater the capacitance, the higher the sensitivity

R0-R15 1Kohm can enhance resistance to mobile and walkie-talkie interference. In general, it can be omitted!

Correlations between external capacitance (Cs) and acrylic thickness

(default threshold=010H)

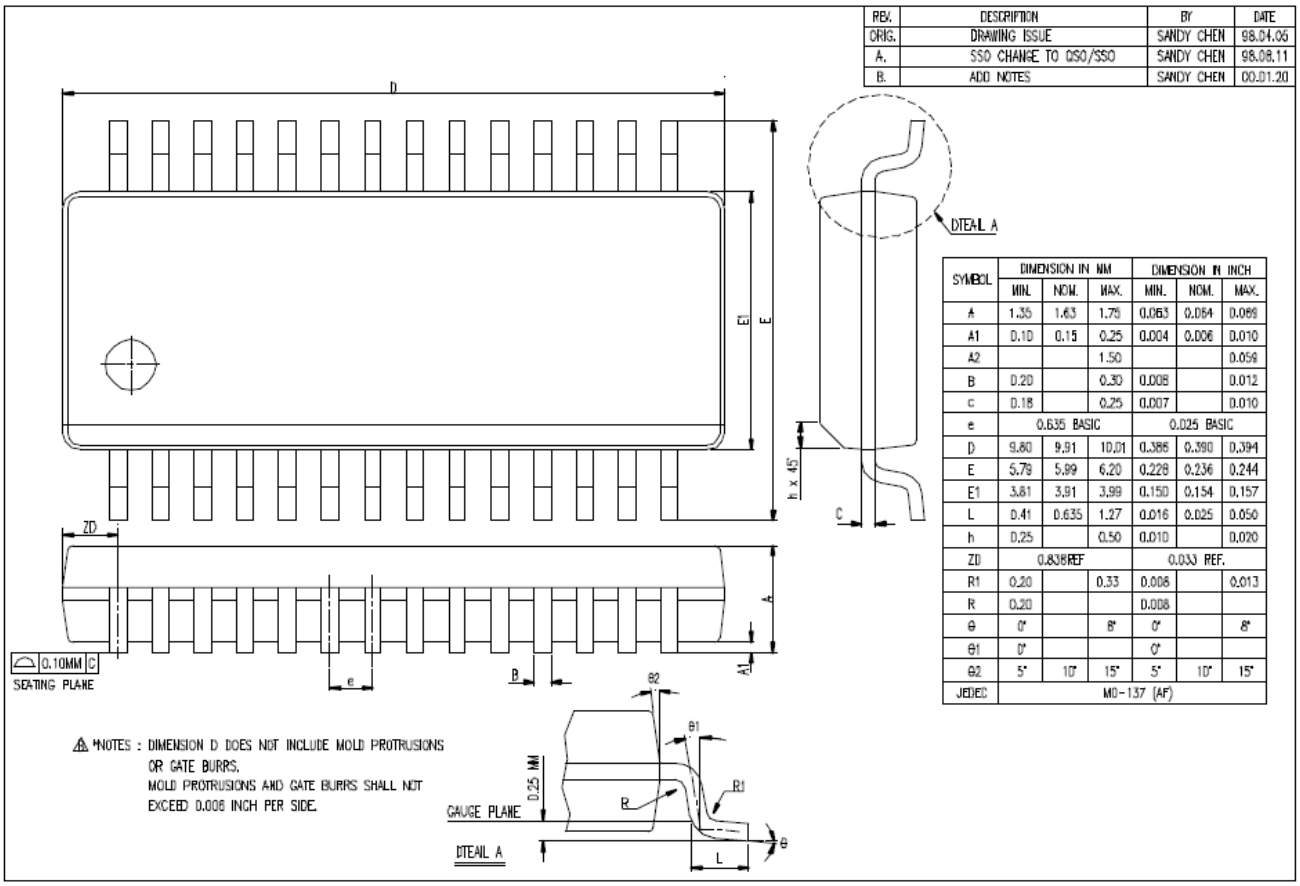
Correlations between acrylic thickness and capacitance (Cs) with an example of round metal flat spring at 12mm dia. :

Acrylic thickness (mm)	Cs
1	682
2	882
4	103
6	153
8	223
10	223

Values in the above table are for reference only. Pad size and PCB layout will affect actual results.

- **Package Description**

(28-SSOP)



SYMBOL	DIMENSION IN MM			DIMENSION IN INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	1.35	1.63	1.75	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)					

Ordering Information

TTY6952		
Package Type	**Chip Type**	**Wafer Type**
TTP259-ASFN	—	—

Revision History

1. 2015/10/26 – Version : 1.00
2. 2016/01/11 – Version : 1.10
   a. Added the demo program.
   b. Added the correlations between Cs capacitance and acrylic thickness.
3. 2016/03/10 - Version: 1.20 :
   Added the description for threshold value adjustment