Freescale Semiconductor
Application Note

Document Number:AN4431
Rev. 1, 11/2012

TSI module application on the S08PT family

by: Wang Peng

Contents

1 Introduction

The S08PT family are the first S08 MCUs that include the Touch Sensor Input (TSI) module to detect capacitive touch sensor.

Capacitive touch sensing has become one of the de facto input technologies for user input in human-machine interfaces (HMI). It now has a place in all types of markets, from industrial control panels to portable consumer devices. Though capacitive touch sensing is not the only touch sensing method, it is one of the most common and most practical to implement.

This application note serves as a guide to design engineers who use S08PT family devices to design products with TSI applications.

2 Touch sensing electrode model

The basic element in capacitive touch sensing is the electrode. In this case, the electrode is an area of conductive material with dielectric material on the top, usually plastic or glass. This is what the user touches. This conductive area plus the dielectric material effectively create a capacitor referenced to the system ground. By touching the dielectric on top of the electrode, the user effectively changes the electrode capacitance both by adding a second conductive area that is grounded (the conductive part of the finger shown in Figure 1) and by increasing the dielectric of the original capacitor. The sensor, or the TSI module in this case, uses a capacitive sensing method to measure changes in the electrode capacitance.

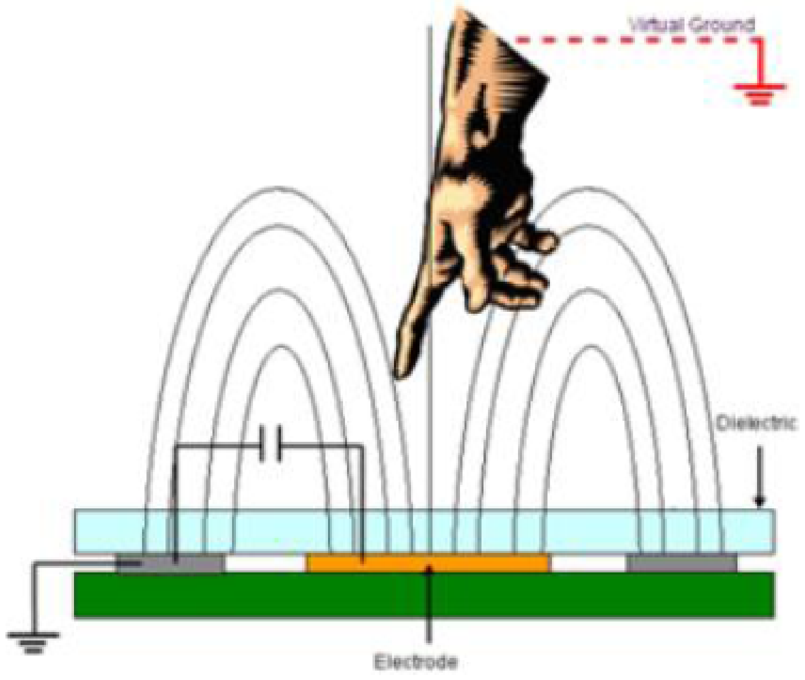

Figure 1. Capacitive touch sensing electrode model

The TSI module provides capacitive touch sensing detection with high sensitivity and enhanced robustness. Each TSI pin implements the capacitive measurement by a current source scan, charging and discharging the electrode, once or several times. A reference oscillator ticks the scan time and stores the result in a 16-bit register when the scan completes. See the following figure.

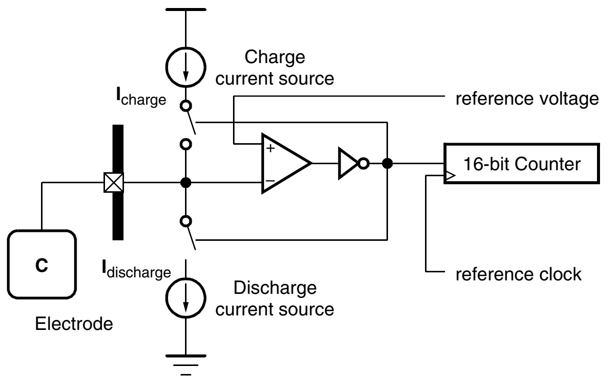

Figure 2. TSI circuit model

When the user touches the electrode, the capacitance of the electrode will increase and charge and discharge time will change. The following figure depicts the change curve.

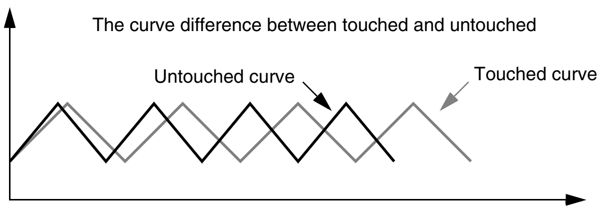

Figure 3. Touched and untouched curve

When a touch is detected, the frequency of the oscillator will become slow. By comparing the counter value changes, both the touched and untouched action states, can be distinguished.

3 Application

There are several parameters with the TSI module which are configurable, such as internal or external constant current, scan number, and so on. To get good performance, the user needs to configure these parameters correctly and optimize the settings. The following sections introduce how to configure TSI registers and optimize configuration, in detail.

3.1 Initialize the TSI module

The TSI module can support up to 16 external electrodes and act as general-purpose input/output (GPIO) pin by default . The user needs to enable these I/O pins as TSI pins by configuring TSI_PEN0 and TSI_PEN1 registers as shown in the following code snippet:

```
TSI_PEN0 = 0x0f; // enable TS0 ~ TS3 external electrodes
```

Constant current to internal or external oscillators can be tuned from 500 nA to 64 µA. The current is directly proportional to the frequency, that is, for large currents, the frequency would also be high. The larger current of the external oscillator will reduce the sensitivity of measuring. See Sensitivity for detailed description.

DVOLT (ΔV) indicates the oscillator’s voltage rails, which can be changed to tune frequency. The delta voltage is inversely proportional to the frequency, as can be seen from the following equation:

Equation 1:

$$F = {}^{I}/_{2C} * \nabla V$$

The corresponding register is as below:

Address: TSI_CS2 is 8h base + 2h offset = 0Ah

Bit	7	6	5	4	3	2	1	0
Read / Write		REFCHRG		DVOLT			EXTCHRG	
Reset	0	0	0	0	0	0	0	0

The following is the code snippet to configure parameters for the internal reference:

```
TSI_CS2_REFCHRG = 0x01; // setting reference oscillator current to 1 µA
TS1_CS2_DVOLT = 0x01; // setting Vp = 0.7Vref, Vm = 0.3Vref
```

The external oscillator can be configured as below:

```
TSI_CS2_EXTCHRG = 0x04; // setting reference oscillator current to 8 µA
```

The user can configure the prescaler of the output of the electrode oscillator by P_s, and change the scan number for each electrode by N_{scan}.

The configuration is as below:

```
TSI_CS1_PS = 2; //configure prescaler to 4
TSI_CS1_NSCA = 0x0f; // scan number is 16.
```

The equation to calculate discharge and charge duration with 16-bit counter is given below:

Equation 2:

$$N = C_{elec} * P_s * N_{scan} * I_{ref} / (I_{ext} * C_{ref})$$

C_{elec}	External electrode capacitance
P_s	Prescaler of the external oscillator
N_{scan}	Scan number of each electrode
I_{ref}	Internal oscillator current
I_{ext}	External oscillator current
C_{ref}	Internal reference capacitor

There are two ways to trigger the TSI module:

- Software trigger: If software trigger is selected, setting TSI_CS0[SWTS] will start a scan for channels specified by TSI_CS3[TSICH]. The following code snippet is used to start a scan with software trigger.

  ```
  // specify the scanning channel
  TSI_CS3_TSICH = 0x01;//select channel 1 start scanning
  // start scanning
  TSI_CS0_SWTS = 1; // start a scanning
  ```
- Hardware trigger mode: When a hardware trigger is applied or, when TSI_CS0[STM] is set, the TSI will not start scanning until the hardware trigger event occurs. The result from the 16-bit counter can be read by register TSI_CNTH:TSI_CNTL. Following is the code snippet to read data from the register.

  ```
  // specify the scanning channel
  TSI_CS3_TSICH = 0x01;//select channel 1 start scanning
  // start scanning
  TSI_CS0_SWTS = 1; // start a scanning
  ```

A sample code to initialize TSI module is as below:

```
void TSI_Init( void ){
        TSI_CS1_PS = 2; //configure prescaler to 4
        TSI_CS1_NSCA = 0x0f; // scan number is 16.
        TSI_CS2_EXTCHRG = 0x04; // setting external oscillator current to 8 µA
        TSI_CS2_REFCHRG = 0x04; // setting reference oscillator current to 8 µA
        TS1_CS2_DVOLT = 0x01; // setting Vp = 0.7Vref, Vm = 0.3Vref
        // enable TSI electrode 0 ~ 4
        TSI_PEN0 = 0x0f;
        // enable TSI module in Stop mode for low-power
        TSI_CS0_STPE = 1;
       // enable interrupt,when scanning complete,generate a interrupt
       TSI_CS0_TSIIEN = 1;
       // enable TSI module
       TSI_CS0_TSIEN = 1;
       }
```

3.2 Electrodes scan

The TSI module supports hardware and software trigger to start a scan. It provides the flexible method to implement the electrode scan. Based on different application requirements, such as low-power application, the hardware trigger can be used to wake up the MCU and complete electrode scan. The following sections will introduce the usage for two trigger modes.

3.2.1 Software trigger

- Clear TSI_CS0[STM] to enable software trigger.
- Set TSI_CS0[SWTS] to start a scan.
- In order to scan all the electrodes and get capacitance change, the user can use the Interrupt or the Polling mode to get result from the TSI_CNT register.
- Store the results of all the electrodes to a buffer array.

The following is a code snippet if polling method is used to get result.

```
static  unsigned int m_uiScanValue[ELEC_NUM];         //result buffer array
static   unsigned char m_ucScanChannel[ELEC_NUM]; //channel buffer array
void TSI_Scan( void ){
       unsigned char i;
       unsigned int uiScanValue[ELEC_NUM];
       for(i=0;i< ELEC_NUM;i++)
      {
        // specify the scanning channel
           TSI_CS3_TSICH = m_ucScanChannel[i];
           // start scanning
           TSI_CS0 |= 0x01;
           // wait scan complete
           while(!TSI_CS0_EOSF);
           TSI_CS0_EOSF = 1;
           m_uiScanValue[i] = TSI_CNT;
       }
}
```

There are two ways to implement electrodes scan:

- Polling mode:
  a. Scan all the electrodes one by one.
  b. When scanning is complete, store the result to buffer array in order.
- Interrupt mode:
  a. Enable the TSI interrupt and trigger the first channel to be scanned.
  b. When scanning is complete, continue to trigger the next channel in the interrupt service routine until all the channels are completely scanned.
  c. Store the result to buffer array in the order.

The following is a code snippet to interrupt service routine:

```
 void TSI_IntStartScan( void ) { //
        m_ucChannelIndex = 0;
        //select the first channel to scan
        TSI_CS3_TSICH = g_ElecWorkInfo[m_ucChannelIndex].Channel;
        // start channel scanning,wait generate interrupt
        TSI_CS0 |= 0x01;
 }
interrupt VectorNumber_Vtsi void TSI_ISR ( void ){ //TSI interrupt service routine
   if( TSI_CS0_EOSF ){
           TSI_CS0_EOSF = 1;
           m_uiScanValue[m_ucChannelIndex++] = TSI_CNT;
           if( m_ucChannelIndex < ELEC_NUM ){
                   //select next channel scan
                   TSI_CS3_TSICH = g_ElecWorkInfo[m_ucChannelIndex].Channel;
                   // start next channel scanning
                   TSI_CS0 |= 0x01;
```

```
    }
     else{
            //all of channels scan completed,setting flag
            g_bScanCompleteFlag = 1;
    }
 }
}
```

The user can start a scan using the function TSI_IntStartScan. When all channel scan is completed, implement processing program and determine if there has been a touch or not in any of the TSI electrodes.

3.2.2 Hardware trigger

Hardware trigger source is a result of real-time counter (RTC) overflow, which can periodically trigger the TSI module to start scan. In order to enable hardware trigger, first configure the TSI module to hardware trigger mode.

```
TSI_CS0_STM = 1;
```

TSI_CS3[TSICH] is used to set the scanning of channels. When RTC generates overflow event, the TSI starts to scan. The following is procedure for hardware trigger to the TSI module:

1. Initialize the TSI module and enable hardware trigger mode.
2. Set the TSI_CS3[TSICH] channel to be scanned.
3. Initialize RTC.
4. Wait for the RTC to generate overflow, and trigger to scan the channel that is specified by TSI_CS3[TSICH].

The following code snippet shows how to implement TSI hardware trigger with RTC overflow event:

```
// enable hardware trigger for TSI
TSI_CS0_STM = 1;
// setting channel
TSI_CS3_TSICH = 0x00;
void RTC_Init( void ){
       RTC_MOD = 10; // 1S
       RTC_SC1_RTIE = 1; // enable RTC interrupt
       RTC_SC2_RTCLKS = 0x01; //select clock source to 1 kHz
       RTC_SC2_RTCPS = 0x06; // clock prescaler is 100
}
interrupt VectorNumber_Vrtc void RTC_ISR ( void ){
       if( RTC_SC1_RTIF ){
       RTC_SC1_RTIF = 1;
       }
}
```

3.3 Improve the performance

For different applications, there are many factors which determine the performance, such as sensitivity, responding time, and stability. For example, a proximity sensor which requires higher sensitivity and longer distance, but has no effect of responding time, can implement lower scan speed to provide increased sensitivity at all distances, whereas other applications may require rapid response time, such as slide, rotary, and so on. Therefore, optimum parameters need to be configured according to different applications.

The following sections explain how to improve the performance.

3.3.1 Sensitivity

Sensitivity is an important parameter to determine that the TSI module can measure the minimum capacitance change. According to Equation 2 in Initialize the TSI module, the minimum capacitance that can be measured is shown as below.

Equation 3:

$$C_{\min} = I_{\mathrm{ext}} * C_{\mathrm{ref}} / (P_s * N_{\mathrm{scan}} * I_{\mathrm{ref}})$$

From the equation given above, it can be inferred that:

- Sensitivity is inversely proportional to $C_{\min}$, that is, smaller $C_{\min}$ makes the module more sensitive.
- To improve sensitivity, decrease I_{ext} or, increase P_s or N_{scan}, or I_{ref}.

In some cases, where the material on the top of the touch is thick, or when the TSI needs to sense longer distances, it is necessary to improve sensitivity.

3.3.2 Stability

The sample from the TSI electrode is subject to be interfered from self-board circuit or other application circumstances. To reduce the impact of this interference, and improve the robustness, it is required to consider how to design TSI application with respect to two aspects, hardware and software.

3.3.2.1 Hardware

Hardware design includes the PCB layout, touch sensors and the trace connect to MCU. While starting the layout, depart from the strong interference signals such as oscillator, clock, bus, or DC-DC as well as other high-frequency signals.

The following things must be considered while designing touch sensing electrodes for the TSI:

- Trace width: Keep the trace width as thin as possible. 5-7 mil traces are recommended. The base capacitance increases with the width of the traces.
- Clearance: Leave a minimum clearance of 10 mil. At the trace connection to the MCU, the pitch is lower than 10 mil, therefore use bottleneck mode.
- Trace length: Keep trace length as short as possible. As traces become longer, the baseline capacitance increases and is also more susceptible to coupled noise.
- Electrode traces must be routed in a different layer from the one containing the electrodes.
- Electrodes and trace must stay far away other strong interference signals. It is recommended to isolate touch sensors from other signals with ground.
- Components and traces must not be placed directly underneath the electrodes area. Good results can be obtained by keeping the number of components behind the electrodes to minimum and running as few traces as possible.
- Avoid large, solid ground-fill areas inside the proximity sensor, as this decreases the sensitivity.

In certain applications, where conducted emissions or, electrostatic discharge (ESD) is a concern, external protective components can be added. The idea is to use only a transient voltage suppression (TVS) diode designed for ESD suppression and a low value (100–470 Ω) resistor as protection for current that might flow into the MCU.

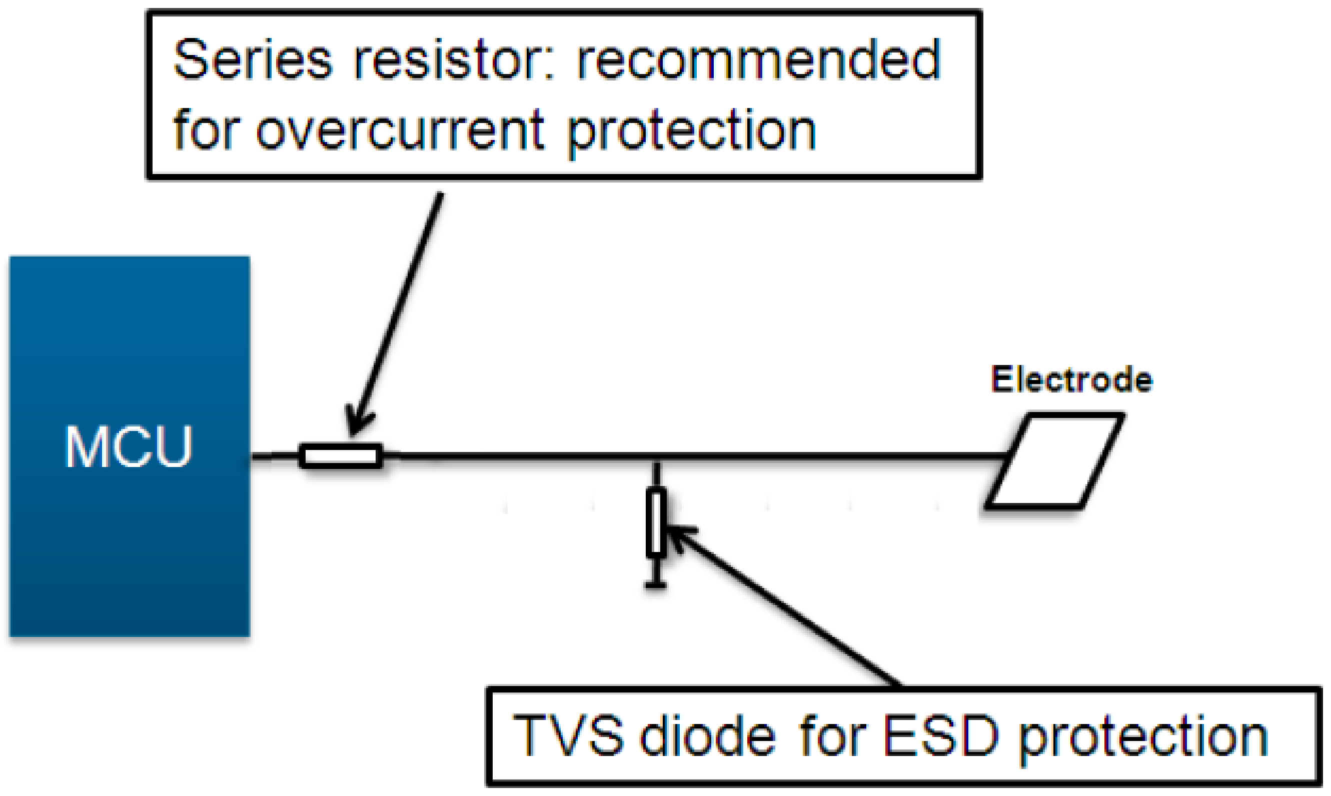
Series resistor: recommended for overcurrent protection
MCU
Electrode
TVS diode for ESD protection

Figure 4. Protected components to electrode

For further information on designing electrodes and in-depth considerations on hardware and electrode design, see AN3863: Designing Touch Sensing Electrodes, available at **freescale.com**.

3.3.2.2 Software

- **Debounce**: Based on count value from scanning all the electrodes, the key state, touched or untouched, can be determined. But sometimes, the result from one sample is variable due to finger bouncing, just like mechanical buttons have electrical disturbances. In capacitive touch sensors, as the finger approaches the electrode, variations in capacitance due to finger approaching or moving away, may falsely trigger more than one touch. Therefore, debouncing can be done to avoid such mistakes. The following figure shows the debouncing flowchart.

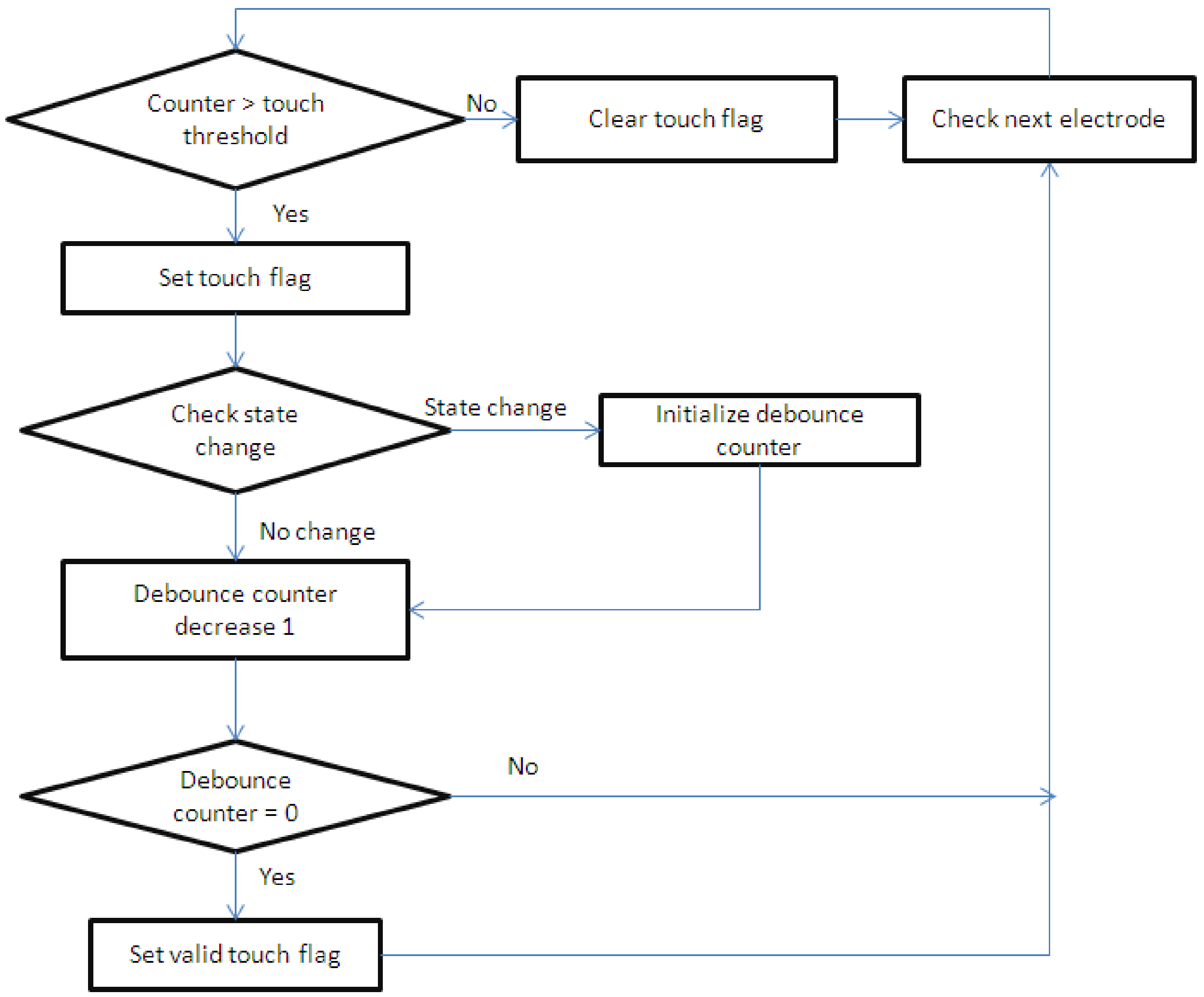

Figure 5. Debounce flowchart

- **Software filter**: In order to reduce noise interference, from software point of view, some processing can be done to decrease the impact of noise arising from some strong interference sources. Therefore, it is recommended to run the TSI scanning at the time when noise is weaker or, these interference sources are inactive. This would reduce noise interference from the controllable noise sources. Apart from this, software filter such as IIR algorithm can be used to reduce noise impact. It is a low-pass filter that is less sensitive to sudden changes in the raw count.
- **Baseline tracking**: The temperature changes, humidity, and noise are some of the factors that can influence the electrode capacitance. These variations are usually small, but sometimes can lead to spurious detection of the electrodes. Therefore, an algorithm is required to avoid erratic behavior in the electrodes. The TSS library implements the baseline tracking algorithm. The algorithm determines if the change in the electrode capacitance was caused by a touch, or an environmental factor. The library also allows to configure the rate at which the baseline is updated. For more information on TSS library, please visit: **freescale.com/TSS**.

4 Stop mode

The TSI can work in Stop mode, when enabled, and can generate the Scan Complete interrupt to wake up the MCU.

In some low-power applications, if no touch lasts certain times, the MCU will enter Stop mode to save power, with RTC overflow event to periodically scan electrodes. When a touch event is detected on any electrode, it wakes up the MCU and restores to Normal Run mode, otherwise continues to enter the Stop mode.

To enter the Stop mode:

- Enable the TSI module by setting TSI_CS0[STPE].
- Configure as hardware trigger.
- Initialize RTC.
- Wait for the RTC overflow event to trigger a scan.

The following figure shows a flowchart for Stop mode application.

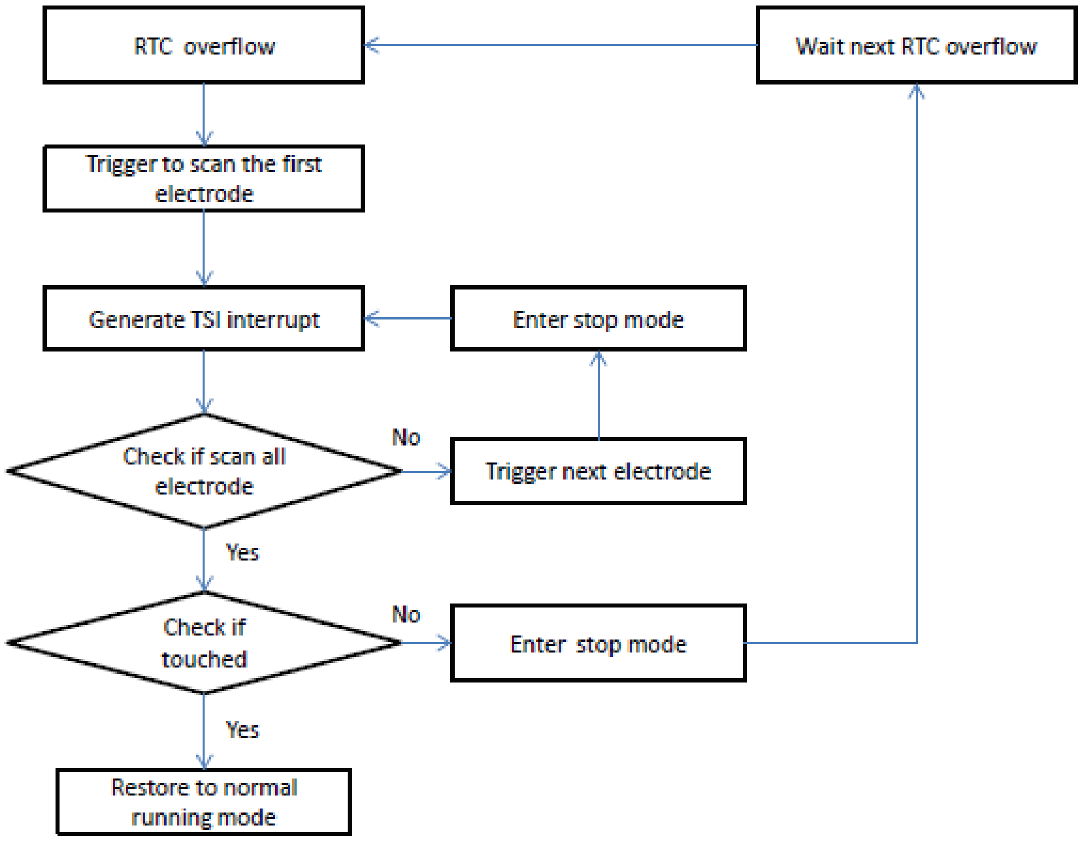

Figure 6. Stop mode flowchart

The following is the code snippet to work in stop mode:

```
void TSI_EnterIntoStopMode( void ) {
         while(TSI_CS0_SCNIP); // wait TSI is idle,
         // change mode to hardware trigger
         TSI_CS0_STM = 1;
         m_ucChannelIndex = 0;
}
interrupt VectorNumber_Vtsi void TSI_ISR ( void ) {
        if( TSI_CS0_EOSF ){
                TSI_CS0_EOSF = 1;
                m_uiScanValue[m_ucChannelIndex++] = TSI_CNT;
                if( m_ucChannelIndex < ELECTRODE_NUM )
                {
                        //select next channel scan
                        TSI_CS3_TSICH = m_ElecWorkInfo[m_ucChannelIndex].Channel;
                        // change mode to soft trigger
                        TSI_CS0_STM = 0;
                        // start next channel scanning
                        TSI_CS0_SWTS = 1;
                        m_bScanCompleteFlag = 0;
                }
                 else {
                        //all of channels scan completed,setting flag
                        m_bScanCompleteFlag = 1;
                        m_ucChannelIndex = 0;
                        //select next channel scan
                        TSI_CS3_TSICH = m_ElecWorkInfo[m_ucChannelIndex].Channel;
                 }
} }
```

Generally, in Normal Run mode, the TSI is configured as software trigger. When entering the Stop mode, call *TSI_EnterIntoStopMode* function to switch the TSI from software trigger to hardware trigger mode, then wait for the RTC overflow event, which will trigger the TSI to start scan. If no touch occurs, the MCU will enter the Stop mode again.

5 Conclusions

This application note describes how to use the TSI module on S08PT MCU, including basic module initialization, scan trigger modes, performance improvement as well as providing a solution for typical low-power applications, which need wakeup from Stop mode by a touch pad event.

Contrary to other S08 MCUs, S08PT family adds the dedicated TSI module which is very convenient to implement HMIs. In addition, Freescale also provides a wide range of production and tools supporting touch sensor inputs. The TSS library is one of these tools compatible with S08PT TSI module. There are also a lot of design guidelines from Freescale on how to design and layout electrodes, most common touch sensing topologies including keys silders, rotary switches, and different keyboard arrangement. With these enablements, it is easy for the user to add the TSI application on production.

6 References

The following reference materials are available at **freescale.com**.

- MC9S08PT60/32 Reference Manual
- TSSMCU: Xtrinsic Touch Sensing for MCUs
- AN1259: System Design and Layout Techniques for Noise Reduction in MCU-based Systems
- AN1985: Application note titled Touch Panel Applications Using the MC34940/MC33794 E-Field IC
- AN3863: Designing Touch Sensing Electrodes for Electrical Considerations and Recommended Layout Patterns
- AN3747: Pad Layout Application Note

7 Glossary

BDM	Background Debug Mode
CRC	Cyclic Redundency Check
ECC	Error Correction Codes
FCCOB	Flash Common Command Object
RTC	Real Time Counter
TPM	Timer/PWM Module
FTM	FlexTimer Module
MTIM	Modulo Timer

Table continues on the next page...

WDOG	Watchdog
TSI	Touch Sense Input

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Document Number: AN4431
Rev. 1, 11/2012