Freescale Semiconductor
Application Note

Document Number: AN4330
Rev. 0, July 2011

Writing Touch Sensing Software Using TSI Module

by: **Daniel Martínez**
Microcontroller Solutions Group
Guadalajara

Contents

1 Before you begin..1
1.1 Reference material..1
1.2 Acronyms and abbreviations..1
2 Touch Sensing Input (TSI) module..2
2.1 What is Touch Sensing Input?..2
2.1.1 How TSI works..2
2.1.2 TSI advantages over software scan methods..2
3 Developing a TSI project using the TSS library..3
3.1 Reasons to use TSS library instead of TSI module directly..3
3.2 Create TSS project..3
3.2.1 How to select TSI module for electrodes..3
3.2.2 Trigger source..4
3.2.3 Low power function..5
3.2.4 TSI configuration variables..6
4 TSI tuning..7
4.1 Configure TSI resolution..7
4.2 Evaluate delta values and baseline..7
4.3 Change the sensitivity value..9
4.4 Continue with your application..10

1 Before you begin

1.1 Reference material

Use this document in conjunction with:

- *Touch Sensing Software API Reference Manual* (document TSSAPIRM)
- *Touch Sensing Software Users Guide* (document TSSUG)
- *K40 Sub-Family Reference Manual* (document K40P144M100SF2RM)

All these documents are available at freescale.com.

1.2 Acronyms and abbreviations

TSS	Touch Sensing Software
TSI	Touch Sensing Input
MCU	Microcontroller Unit
SSC	System Setup Creator
TWRPI	Tower Plug-ins

© 2011 Freescale Semiconductor, Inc.

2 Touch Sensing Input (TSI) module

2.1 What is Touch Sensing Input?

TSI is an electrode capacitive scan method based on hardware. Not all Freescale MCUs include a TSI module — it is a specific peripheral present only in certain MCU families like Coldfire+ and Kinetis. Besides all the advantages offered by having a dedicated module for touch sensing solutions, our TSS library has complete support for TSI. This makes the configuration and use of the module much easier.

2.1.1 How TSI works

Basically TSI uses two oscillating signals, one internal and another external. Each signal can be configured by the module. The external signal must be slower — this allows counting the number of times that the fast signal oscillates during a single external period.

When a touch changes the capacitance, the internal signal remains oscillating at the same frequency as before, but the external signal frequency becomes slower. For this reason there is an increment of the internal oscillation count during the external period.

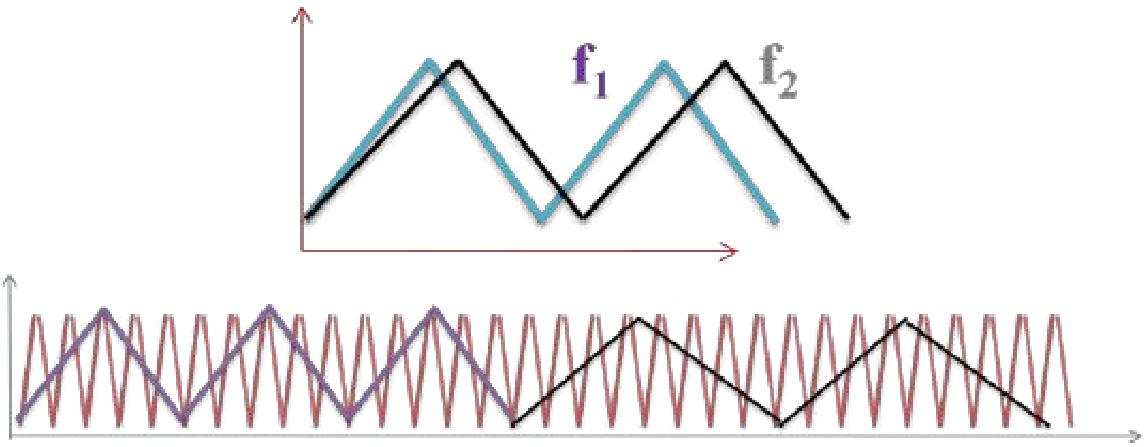

Figure 1. Internal reference oscillations (Blue) vs. external reference oscillations (Black)

The value thus generated must be recorded on a TSI module register and will be actualized at the end of each scan. In this way the TSS library or the application will know when the touch has occurred.

In the next sections some configurable parameters, including the external and internal reference oscillations, will be explained and shown in demo code.

NOTE

For further information about TSI operations, please refer to the TSI chapter in the appropriate MCU reference manual.

2.1.2 TSI advantages over software scan methods

Below is a list of the most important advantages that TSI offers versus a software scan method.

- Reduced MCU processing time
- More robustness against electrical noise
- More sensitivity
- Ability to work in, and wake up from, low power modes
- No need for external hardware, except electrode
- More configuration possibilities
- Interrupts for simplified real-time processing

In addition to these improvements, there are also all the advantages already offered by the TSS library.

To better understand what TSS offers, please refer to Freescale document TSSAPIRM, *Touch Sensing Software API Reference Manual*, which explains this library in detail.

3 Developing a TSI project using the TSS library

3.1 Reasons to use TSS library instead of TSI module directly

The TSS library is modular software that provides a framework for Freescale touch sensing solutions. It is structured only for linking the user application with the modules used in the application code. The TSI sensor method is one of the integrated low level methods using the TSI hardware (HW) peripheral module.

TSS covers these functions for the TSI periphery:

- TSI HW module initialization
- TSI HW module auto-calibration
- Waking the MCU from low power mode by detecting a touch
- Triggering in all available modes (ALWAYS, software (SW) and AUTO)
- Starting the capacitance measurement on the pin and handling measured values

Over the measured signal you can use all other TSS functions, such as:

- Baseline tracking
- Debouncing
- TSS decoding of key pad, slider, and rotary controls
- IIR and noise amplitude filters
- Visualization and debugging tools
- Possibility to create a mixed solution combining software scan methods and TSI

3.2 Create TSS project

Below is a description of the new TSS options when using the TSI module. Freescale document TSSUG, *Touch Sensing Software Users Guide,* has a complete step-by-step description of how to create a new TSS project, if needed to complement this description. This guide is intended to help understand the configurable parameters in further detail.

3.2.1 How to select TSI module for electrodes

When the electrodes are to be assigned, the TSI option should be chosen as the scan method (Figure 2 for System Setup Creator, Figure 3 for Processor Expert). Make sure that the correct channel is selected.

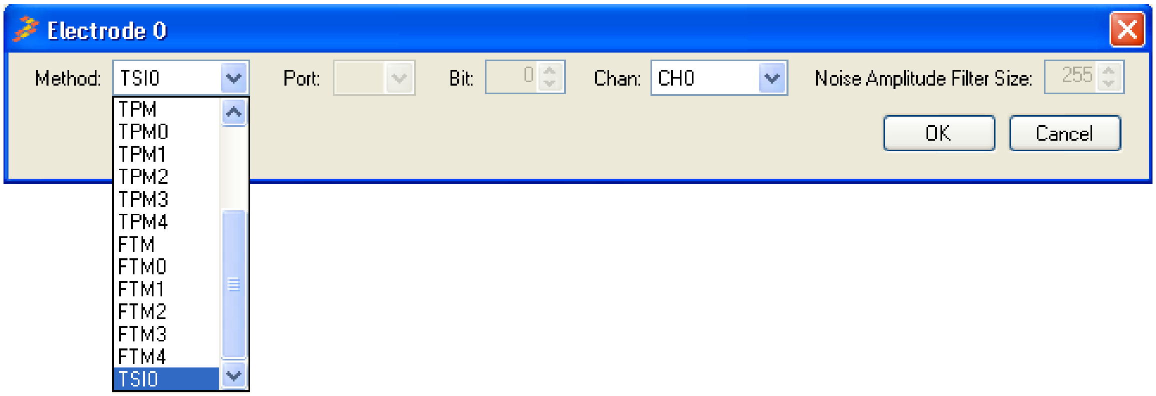

Figure 2. Electrode pin assignment in System Setup Creator

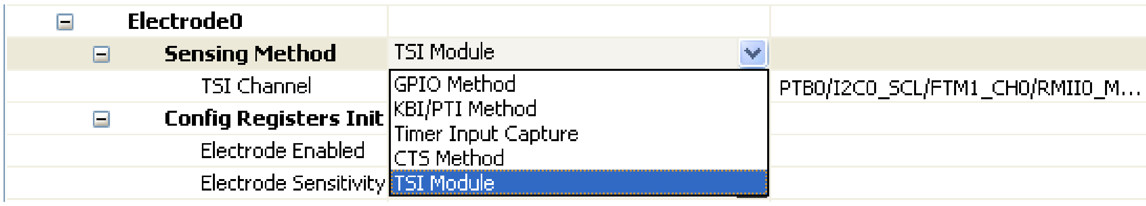

Figure 3. Electrode pin assignment in Processor Expert

The TSS_SystemSetup.h file should be written something like this.

```
#define TSS_E0_TYPE         TSI0_CH0      /* Electrode measurement method specification */
#define TSS_E1_TYPE         TSI0_CH6      /* Electrode measurement method specification */
#define TSS_E2_TYPE         TSI0_CH7      /* Electrode measurement method specification */
#define TSS_E3_TYPE         TSI0_CH8      /* Electrode measurement method specification */
#define TSS_E4_TYPE         TSI0_CH13     /* Electrode measurement method specification */
#define TSS_E5_TYPE         TSI0_CH14     /* Electrode measurement method specification */
#define TSS_E6_TYPE         TSI0_CH15     /* Electrode measurement method specification */
#define TSS_E7_TYPE         TSI0_CH5      /* Electrode measurement method specification */
#define TSS_E8_TYPE         TSI0_CH9      /* Electrode measurement method specification */
#define TSS_E9_TYPE         TSI0_CH11     /* Electrode measurement method specification */
#define TSS_E10_TYPE        TSI0_CH10     /* Electrode measurement method specification */
#define TSS_E11_TYPE        TSI0_CH12     /* Electrode measurement method specification */
```

In this case the twelve electrodes were assigned to TSI channels. The electrode number assignment will be important when the decoder control option is selected — see the next example.

```
#define TSS_N_CONTROLS                          2

#define TSS_C0_TYPE           TSS_CT_KEYPAD     /* Control type */
#define TSS_C0_ELECTRODES     10                 /* Number of electrodes in the control */
#define TSS_C0_STRUCTURE      cKey0             /* Name of the C&S struct to create */
#define TSS_C0_CALLBACK       TSS1_fCallBack0   /* Identifier of the user's callback */

#define TSS_C1_TYPE           TSS_CT_KEYPAD     /* Control type */
#define TSS_C1_ELECTRODES     2                 /* Number of electrodes in the control */
#define TSS_C1_STRUCTURE      cKey1             /* Name of the C&S struct to create */
#define TSS_C1_CALLBACK       TSS1_fCallBack1   /* Identifier of the user's callback */
```

In this case there are two keypad controls. The first one will control electrodes E0 to E9 and the second one electrodes E10 and E11.

3.2.2 Trigger source

In the software scan method the measurement processing is asynchronous, which means that the scan process is carried out every time TSS_Task function is called. The trigger source is a new TSI feature that allows you to configure the scan process in three different modes:

- Always mode: the electrode scanning never stops — as soon as the scan is finished it immediately starts again.
- Software mode: the scan is performed only if the code directs it do so.
- Auto mode: after the scan finishes, there is a configured wait time, and then the scan will start again.

Note that this feature only affects the hardware scan process. This means that TSS_Task has to be called to perform the touch detection and all the software TSS functions.

Below is an example of how to configure the trigger source in auto mode. The trigger source is also affected by the clock source, clock divider and clock prescaler. These variables are explained in the next section.

NOTE

For further information about the trigger source, please refer to *Touch Sensing Software Users Guide* (TSSUG) and *Touch Sensing Software API Reference Manual* (TSSAPIRM).

This next code line in TSS_SystemSetup.h will enable the trigger source.

```
#define TSS_USE_TRIGGER_SOURCE     TSI0    /*Identifier of the Trigger source device*/
```

This is the code to configure the trigger source. The scan period is an 8-bit configurable register. Values between 1–255 are valid.

```
/* Auto Trigger Config */
TSS_SetSystemConfig(System_SystemTrigger_Register, TSS_TRIGGER_MODE_AUTO);
/* TSS_TRIGGER_MODE_AUTO/TSS_TRIGGER_MODE_ALWAYS/TSS_TRIGGER_MODE_SW */
TSS_SetSystemConfig(System_AutoTriggerModuloValue_Register, 255);
```

3.2.3 Low power function

The low power function attempts to wake the MCU from low power mode if the defined source device detects a touch. The configurable variables for low power are:

- Selectable electrode, which is any TSI electrode from the application
- Low power scan period (to reduce power consumption, the scan period should be slower than normal scans)
- Sensitivity, which generally will be the same or less as in run mode

The electrode functionality will depend on which low power mode was configured. Deeper low power modes allow functionality only from the selected electrode. Other operation modes allow operation from all electrodes on the application. Please refer to the appropriate MCU reference manual to clarify the low power modes available.

This next macro in the TSS_SystemSetup.h file will enable the use of low power.

```
#define TSS_USE_LOWPOWER_CONTROL_SOURCE  TSI0  /* Identifier of the Low
                                         Power control source device */
```

The code below shows the lines needed to configure the low power mode parameters.

```
/* Low Power Config */
TSS_SetSystemConfig(System_LowPowerScanPeriod_Register, 0x08);
TSS_SetSystemConfig(System_LowPowerElectrode_Register, 5u);
TSS_SetSystemConfig(System_LowPowerElectrodeSensitivity_Register, 0x1A);
```

NOTE

TSS only enables TSI parameters to work in low power, but the appropriate low power configuration and low power entry must be made by the application.

3.2.4 TSI configuration variables

The TSI module has different configuration options to customize your application depending on what you need. The TSS gives you the possibility to configure those variables. Figure 4 and Figure 5 show the help tools on System Setup Creator and Processor Expert respectively.

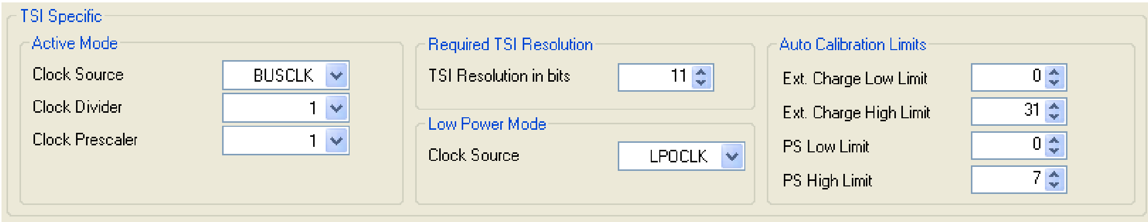

Figure 4. TSI options on System Setup Creator

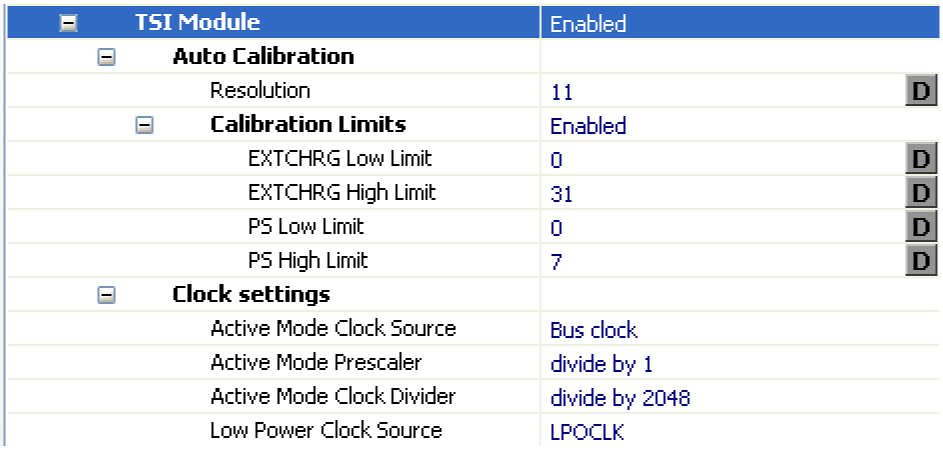

Figure 5. TSI options on Processor Expert

For active mode

Clock source: The selection of the clock will depend on your application and the operation modes you want to use. To verify which specific clock you need for each operation mode, please refer to the appropriate MCU reference manual.

```
#define TSS_TSI_SCANC_AMCLKS     0      /* Input Active Mode Clock Source
                             (0=BUSclk, 1=MCGIRCLK, 2=OSCERCLK, 3-NA)*/
```

Clock divider: This variable directly affects the clock source that has been selected. It is in effect only if auto mode on trigger source is enabled.

```
#define TSS_TSI_SCANC_AMCLKDIV  1     /* Input Active Mode Clock Divider
                                   ( 0 = divide 1, 1 = divide 2048 ) */
```

Clock prescaler: This register is in effect only if auto mode is enabled on trigger source. The configuration will depend on the scan period required by the application.

```
#define TSS_TSI_SCANC_AMPSC    0     /* Input Active Mode Clock Prescaler
                                   ( 0 = divide 1, 7 = divide 128 ) */
```

For Low power mode

Clock source: This variable is relevant only if you use low power mode. It will be selected depending on the current consumption and the available chip clock for the application.

```
#define TSS_TSI_GENCS_LPCLKS    0         /* Low Power Mode Clock Source
                                       ( 0 = LPOCLK, 1 = VLPOSCCLK ) */
```

TSI resolution

TSI resolution in bits: This variable is not a TSI module register, but depending on this value most of the TSI parameters will be calculated. The larger this value is, the more sensitive the application will be, but the smaller it is, the more robust the application will be against noise.

```
#define TSS_TSI_RESOLUTION   11  /*Required Bit resolution of the TSI*/
```

TSI auto-calibration limits

External OSC charge current low and high limit: This value controls the current from the oscillator. If the current is low the sample charge time will be slower, which allows more sensitivity. But if the current is low, the possibility of being affected by electrical noise is bigger.

```
#define TSS_TSI_SCANC_EXTCHRG_LOW_LIMIT      1         /* Low Limit of External OSC Charge
Current register value for TSI autocalibration */
#define TSS_TSI_SCANC_EXTCHRG_HIGH_LIMIT     31        /* High Limit of External OSC Charge
Current register value for TSI autocalibration */
```

External OSC prescaler low and high limit: This variable acts like a sample multiplier, which means that the higher this value is, the higher the number of samples will be too. This is the best way to reach the required sensitivity.

```
#define TSS_TSI_GENCS_PS_LOW_LIMIT           0         /* Low Limit of External OSC Prescaler
register value for TSI autocalibration */
#define TSS_TSI_GENCS_PS_HIGH_LIMIT          7         /* High Limit of External OSC
Prescaler register value for TSI autocalibration */
```

4 TSI tuning

One of the most complicated parts of development for a touch sensing design is the tuning of the electrodes. This section provides a description of how to perform this process.

To make this activity easier and more informative we will use the FreeMaster tool, which shows useful TSS variables in real time while tuning the design. You can also check these variables via the CodeWarrior debugger, but CodeWarrior does not always have the capability to show the variables in real time.

NOTE

Touch Sensing Software Users Guide (TSSUG) has a guide to enable FreeMaster on your project. You can also use the examples in the TSS library as a guide.

4.1 Configure TSI resolution

This first setting may be configured at the start of the the project with any TSS tool, or you can modify this value directly in the TSS_SystemSetup.h file if your project is already complete. For this example the resolution will be eight, which is the least sensitive but the most robust for noisy environments.

```
#define TSS_TSI_RESOLUTION   8  /*Required Bit resolution of the TSI*/
```

4.2 Evaluate delta values and baseline

Run the program and look at the electrode baseline (`tss_au16ElecBasline[]`) and delta (`tss_ai8InstantDelta[]`). The value of this variable should increment during a touch event.

If there is no delta or if the change to be detected is very small, increment the OSC prescaler register variable until the delta value will be around 20–70. The higher the value then the higher the sensitivity, but the tradeoff is scanning time. Doing more scans means more time to get a measurement.

Figure 6 is an example where the sensitivity is too low. The evaluated electrode on this example is E3, the maximum delta value on touch events was around 8, and the baseline was around 169.

Relevant variables:

```
(void)TSS_SetSystemConfig(System_NSamples_Register, 8);
#define TSS_TSI_GENCS_PS_LOW_LIMIT           1
#define TSS_TSI_GENCS_PS_HIGH_LIMIT          7
```

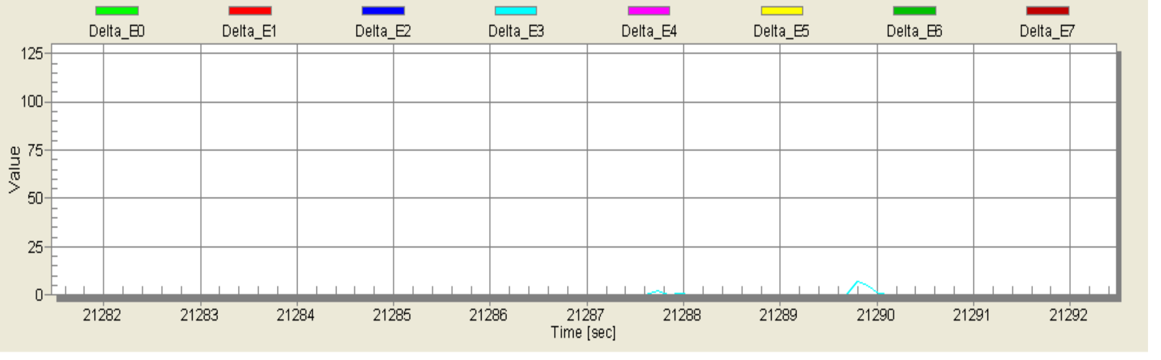

Figure 6. Delta values with low sensitivity

If the sensitivity is too high then there can be crosstalk in touch events, meaning that approaching one electrode may trigger a touch in another one. Reduce the OSC prescaler value to reduce sensitivity.

Figure 7 is an example of a configuration that has too much sensitivity. The evaluated electrode is E3, baseline value is around 6885, a touch event delta value is 127, which means it has overflowed. The problem here is that some electrodes show crosstalk. Also note that by having too much sensitivity, electrical noise will have a bigger impact.

Relevant variables:

```
(void)TSS_SetSystemConfig(System_NSamples_Register, 8);
#define TSS_TSI_GENCS_PS_LOW_LIMIT           7
#define TSS_TSI_GENCS_PS_HIGH_LIMIT          7
```

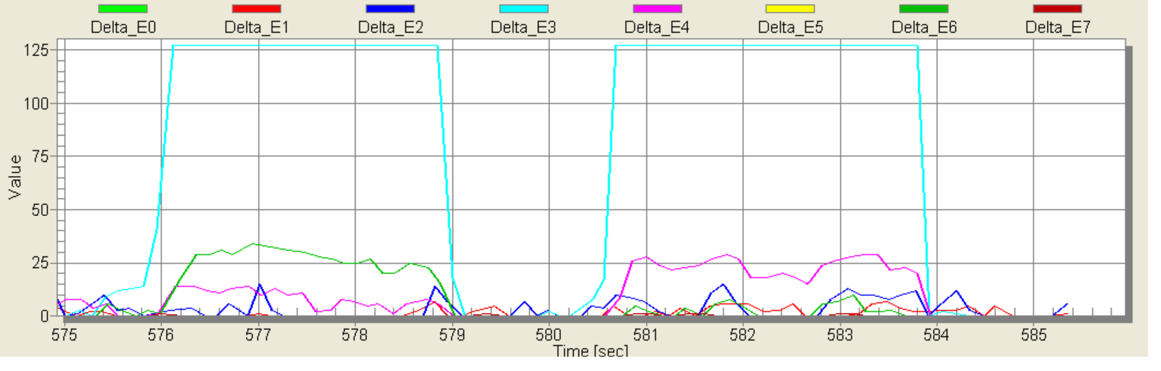

Figure 7. Delta value with high sensitivity

In Figure 8, the evaluated electrode is E3, its baseline is 896, and delta value is around 40. In this case a single increment on OSC prescaler may be greater than we want. The appropriate variable to change is the sample number, which modifies the sensitivity but to a smaller degree than the OSC prescaler.

Relevant variables:

```
(void)TSS_SetSystemConfig(System_NSamples_Register, 8);
#define TSS_TSI_GENCS_PS_LOW_LIMIT            4
#define TSS_TSI_GENCS_PS_HIGH_LIMIT           7
```

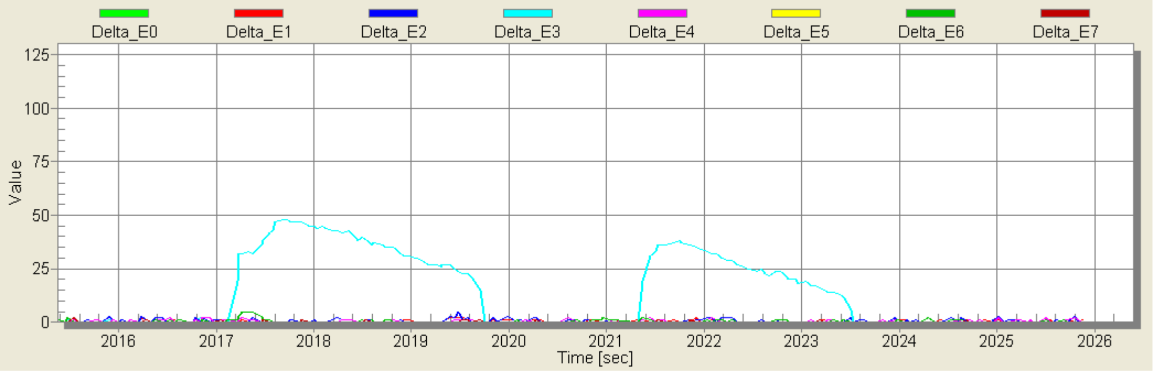

Figure 8. Delta value that required a small change

Figure 9 shows the evaluated E3 electrode with a delta value of approximately 100, baseline is 1722. There is no crosstalk and there is no overflow on the delta value. For this reason noise should not have a significant impact on the electrodes.

Relevant variables:

```
(void)TSS_SetSystemConfig(System_NSamples_Register, 16);
#define TSS_TSI_GENCS_PS_LOW_LIMIT            4
#define TSS_TSI_GENCS_PS_HIGH_LIMIT           7
```

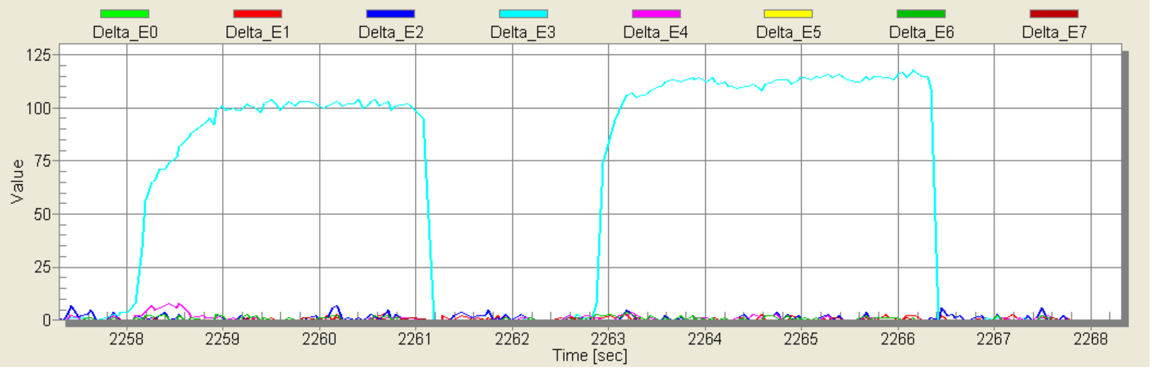

Figure 9. Appropriate delta value

4.3 Change the sensitivity value

Once the corresponding changes have been made to delta values, it is time to change the sensitivity values for the application.

The first step is to check the maximum delta value for each electrode. For the previous example, E3 maximum delta is around 110. The recommendation is to use 90% of the maximum delta — for this example that means configure sensitivity to 99 . After evaluating the user experience and tuning the application, the final value for the example is 90. This is important, because a bigger margin allows setting the sensitivity threshold above the signal-to-noise-ratio (SNR), meaning there is less risk of false touch triggers.

Figure 10 show the sensitivity threshold value. In this case TSS will interpret a valid touch when the delta value reaches or exceeds the sensitivity threshold (90).

```
(void)TSS_SetSystemConfig(System_Sensitivity_Register + 3, 90);
```

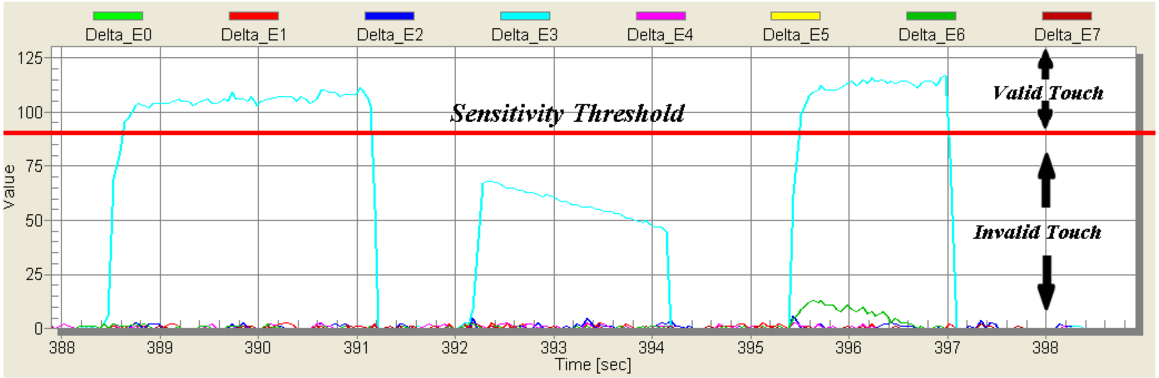

Figure 10. Sensitivity threshold

4.4 Continue with your application

After completing the previous steps, continue with the development of the application. Remember that TSS offers many options to create a complete project. Again, for more information refer to TSSAPIRM, *Touch Sensing Software API Reference Manual.*

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

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductors products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claims alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-complaint and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to http://www.freescale.com/epp.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2011 Freescale Semiconductor, Inc.

Document Number: AN4330
Rev. 0, July 2011