Tuning a touch sensing application on MCUs

Introduction

This document provides guidelines on how to tune touch sensing applications on MCUs, with the STM-Studio tools (STM-STUDIO-STM32).

This application note details a methodology to configure the STMTouch library parameters, and shows how to trim firmware parameters and adjust hardware components to optimize the application performance. All values given in this document are for guidance only (refer to the related datasheet to get guaranteed values).

STMicroelectronics is providing free STMTouch touch sensing firmware libraries, directly integrated into the corresponding STM32Cube package (such as STM32CubeL4).

Table 1. Applicable products

Type	Product series
Microcontrollers	STM32F0 Series, STM32F3 Series, STM32L0 Series, STM32L1 Series, STM32L4 Series, STM32L4+ Series, STM32L5 Series, STM32U5 Series, STM32WB Series

1 STM-Studio overview

The STM-Studio tools are free software to debug and diagnose Arm$^{®}$-based STM32 microcontrollers running applications, by reading and displaying their variables in real-time.

Running on a PC with Windows operating system, the STM-Studio tools interface with the MCU via standard development tools such as ST-LINK/V2. These STM-Studio tools are non-intrusive, preserving the real-time behavior of the applications and they perfectly complement traditional debugging tools to fine tune the applications.

The STM-Studio tools are especially well suited for debugging applications that cannot be stopped, such as motor control.

Different graphic views are available to match the needs of debugging and diagnosis or to demonstrate the application behavior. The STM-Studio tools work with STM32 microcontrollers through JTAG or SWD (serial wire debug) interface.

Refer to the *STM-STUDIO-STM32* release note (RN0058) and to the user manual

Getting started with STM-Studio (UM1025) for more details.

arm

Note: *Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.*

2 Monitoring STMTouch driver variables using STM-Studio

The main parameters to trim a touch sensing application are the following:

- Channel reference: "Ref" element of an array of TSL_ChannelData_T structure
- Channel delta: "Delta" element of an array of TSL_ChannelData_T structure
- Object state: "StateId" element of an array of TSL_TouchKeyData_T structure or a TSL_LinRotData_T structure

This list is not exhaustive and depends on the application.

To import these variables, the steps listed below must be followed:

1. Open the STM-Studio tool corresponding to the used MCU.
2. Right-click in the *Display Variables* tab and select *Import* or select the *File/Import* Variables menu.
3. In the *Import variables from executable* window (shown in the figure below):
   a. Select the application Elf file (.elf, .out or .axf) through the *Executable file* field using the *Browse* button
   b. Check the *Expand table elements* box.
   c. Check the *Store executable path relatively to the user settings file* box to use a relative path.
   d. Enter 'Ref' in the *Show symbol containing...* text box.
   e. Select *Add variables to the display variables table* in the *Variables list* box.
   f. Select the '.Ref' ended variables and click on the *Import* button or Ctrl+Click to operate an noncontinuous multi-selection.
   g. Repeat step d) and f) with 'Delta'.
   h. Repeat step d) and f) with 'StateId'.
   i. Click on the *Close* button.

**Figure 1. STM-Studio variable selection window**
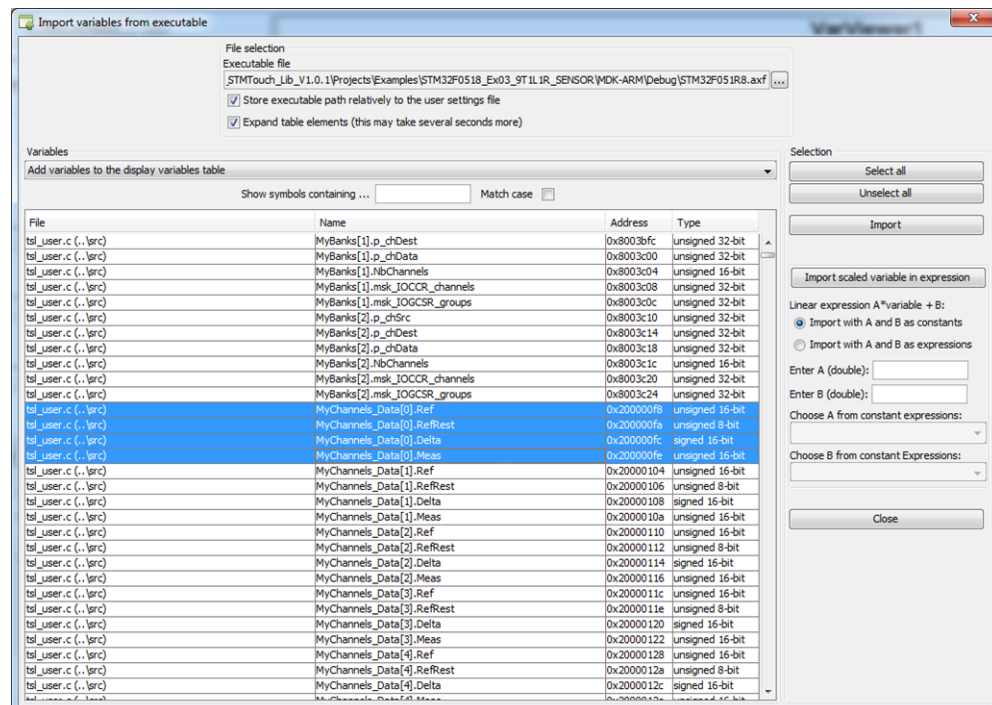
Once imported, the variables must be assigned to the viewer in order to be displayed. Follow the steps listed below:

1. As described previously, select some variables in the *Display Variables settings* table.
2. Right-click in the table and select *Send To →VarViewer1* or drag them directly to the right viewer.
3. In the *Viewers settings* window dock, right-click in the greyed part and select *New VarViewer*. A new *VarViewer2* tab appears.

To ease the navigation, the *VarViewer* windows can be renamed with the name of the monitored variables (right click and rename).

Variables can be displayed as a curve, as a bar graph or in a table. The display in table is recommended for variables with very slow variation. Curve and bar graph suit for variables with quick variation (see examples in the figure below).

Figure 2. VarViewers with variable name
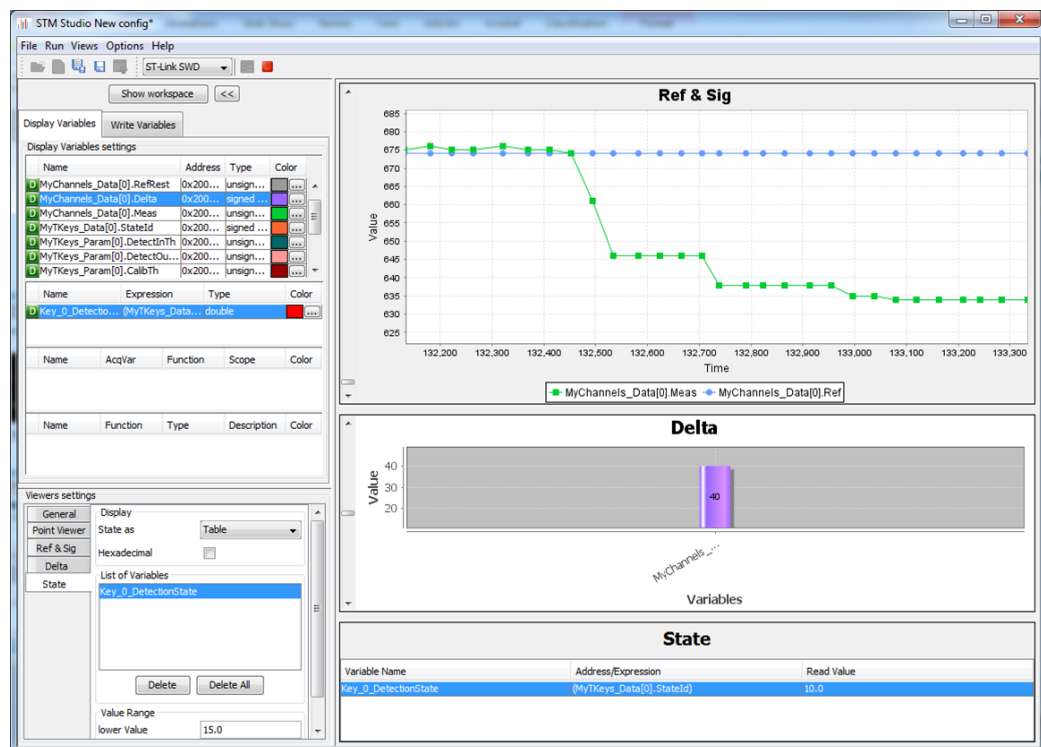
The value range for each *VarViewer* window can be adjusted:

- The Delta depends on the application sensitivity and can be positive or negative.
- The State varies from 0 to 19 (refer to TSL_StateId_enum_T in tsl_types.h for the meaning of this value).
- The reference depends on C_X/C_S.

Connect a PC to the application with the selected binary code downloaded in the microcontroller (through a USB cable and the appropriate hardware tool such as a ST-LINK). Click on the green arrow or select the "Run/Start" menu to start the monitoring.

Data can be stored in a file with the below actions (see the figure below):

1.   Open *Options* and *Acquisition Settings* window.
2.   Check the *Log to file* box and set the log file path.

Figure 3. Data log setting

3 Tuning the thresholds

This section details how to select reliable thresholds depending on the application use cases.

Capacitive touch sensing applications are sensitive to earth coupling. The parameter tuning must be done in the same environment as the final application. The hardware tool connected to the application may change the earth coupling. This is especially true for example in battery power applications. ST is providing a galvanic insulated hardware tool (ST-LINK/V2-ISOL) to minimize this effect.
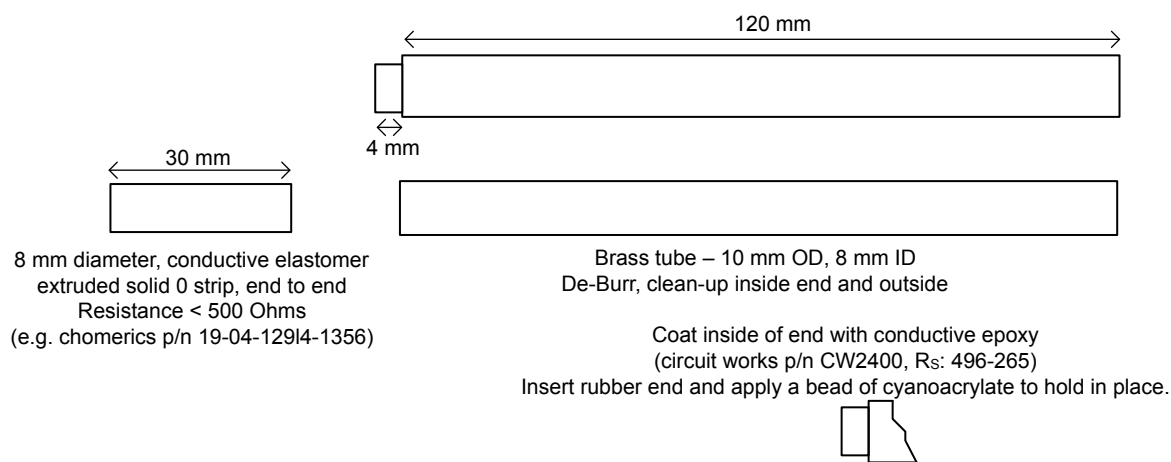
3.1 Use of a standard test finger

In order to build an application working with the widest range of people finger characteristics, the use of a standard test finger (see the figure below) gives a worst case but also allows the test repetition without human dependency (to test finger size, pressure and contact area or skin conductivity for example). An electrically conductive pen-shape tool with a flat rubber end is the most adequate for repetitive tests. The flat end made of conductive rubber allows a constant contact surface with the touchkey. The operator performing the validation must take care to center the contact area on the touchkey. A final validation with a panel of users may be planned.

Figure 4. Standard finger (8 mm diameter)

The figure below describes a recommended standard test finger.

Figure 5. Recommended standard finger

120 mm

4 mm

30 mm

8 mm diameter, conductive elastomer
extruded solid 0 strip, end to end
Resistance < 500 Ohms
(e.g. chomerics p/n 19-04-129l4-1356)

Brass tube – 10 mm OD, 8 mm ID
De-Burr, clean-up inside end and outside

Coat inside of end with conductive epoxy
(circuit works p/n CW2400, R_S: 496-265)
Insert rubber end and apply a bead of cyanoacrylate to hold in place.

Notes: General tolerance +/- 0.5 mm. Remove all sharp corners and edges.

## 3.2 Threshold definitions

3.2.1 Touchkeys thresholds

To tune the detection thresholds, it is first necessary to determine the sensitivity of each touchkey, with the steps listed below:

1. Connect the final hardware to a PC through ST-LINK/V2 and power the application.
2. Download the firmware used in the final application with the final parameters of the STMTouch driver. The default detection thresholds can be set to a low value but must be kept higher than the noise level.
3. Launch the STM-Studio tool and configure it as explained in Section 1 STM-Studio overview.
4. Use the standard finger as described in Section 3.1 Use of a standard test finger.
5. Touch a touchkey and move the finger in order to find the maximum delta. Write down this value, then repeat the operation for each touchkey. If around the maximum delta there is still significant jittering on the measure, then compute the average and use it as baseline.

The collected values are the thresholds baseline. If a significant variation exists between the baseline of the application touchkeys, it is recommended to set a specific threshold for each touchkey.

The detection threshold that must be exceeded in order to report the touchkey as detected, must be set between 55 % and 65 % of the baseline. The end-of-detection threshold, below which the key is not detected anymore, must be set between 35 % and 45 % of the baseline (see the figure below).

Figure 6. Threshold position
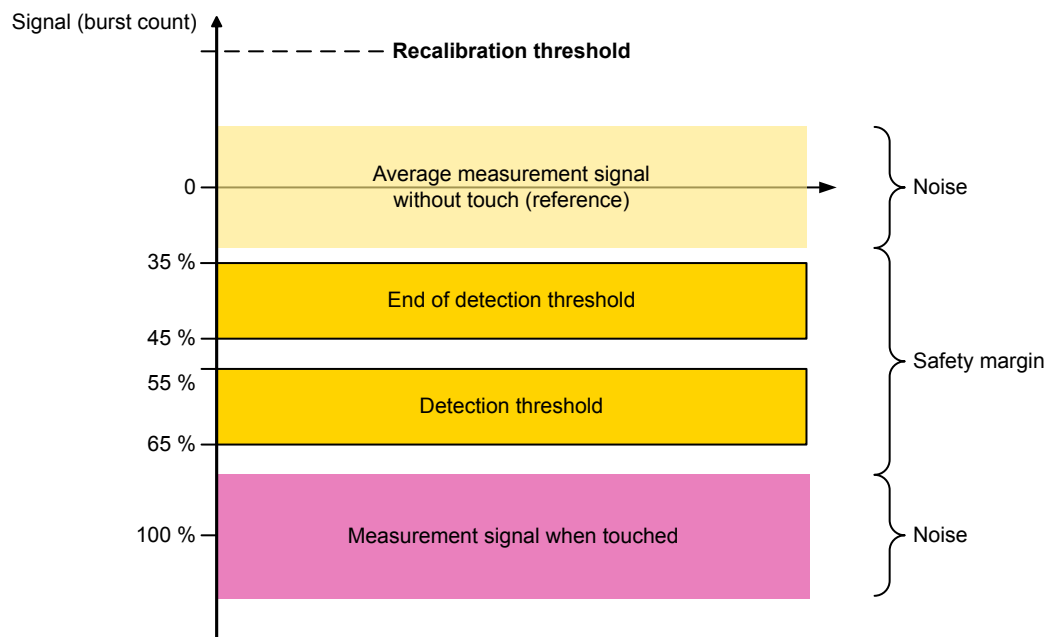
An example of threshold firmware adjustment, with a baseline measured at 80 and a threshold adjusted between ~65 % and ~55 % of the baseline is given below:

```
MyTKeys[0].p_Param->DetectInTh = 50; // Key 1 detection threshold
MyTKeys[0].p_Param->DetectOutTh = 30; // Key 1 end of detection threshold
```

The calibration threshold (TSLPRM_TKEY_CALIB_TH) can be common to all keys and set to 60 % of the maximum baseline.

If one of these thresholds must be greater than 255, the TSLPRM_COEFF_TH must be set in order to bring the value in the correct range [0;255]. This is obtained by dividing by two all thresholds (except the calibration threshold) to the power of TSLPRM_COEFF_TH. In that case, the divided coefficients values (except for the calibration threshold) are the values that need to be configured in the firmware. A compensation factor of two to the power of TSLPRM_COEFF_TH is then applied to all coefficients (except the calibration threshold) by the firmware.

3.2.2 Linear and rotary touch sensors thresholds

The approach is different for these sensors as they are composed of several channels. The standard test finger must be moved along the whole sensor and a log of the delta must be recorded using the STM-Studio tool.

Detection thresholds adjustment

A rotary sensor log example is shown is the figure below.

Figure 7. Rotary sensor log

Worst case delta to consider for a rotary touchkey threshold tuning
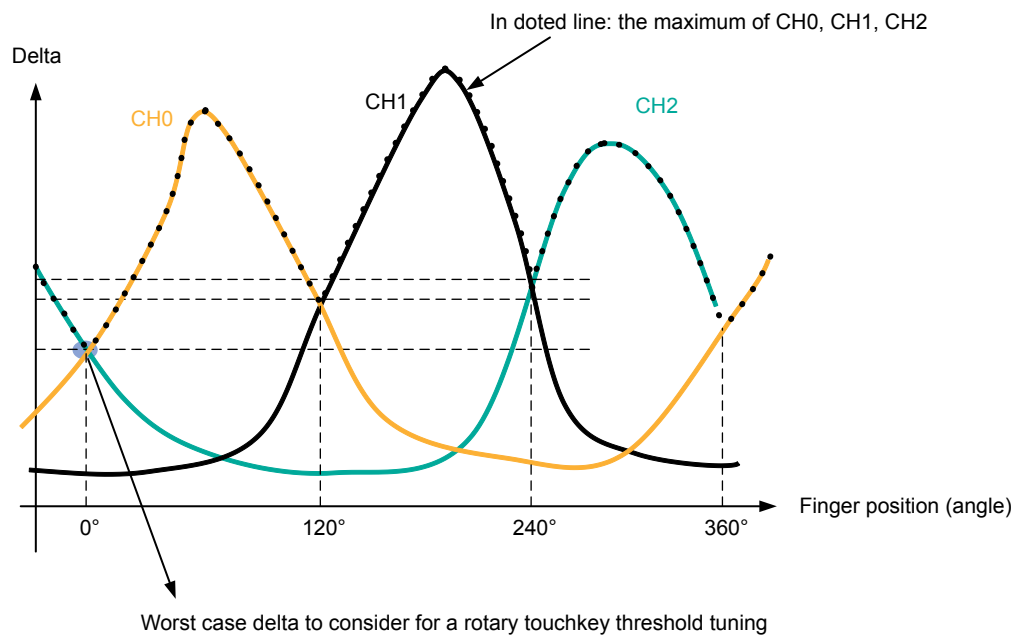
The worst case delta as shown in the figure above must be considered as the baseline to compute the threshold for this sensor. This threshold must be reached in order to trig the computation of the position and to report a detection on this sensor.

The same ratio used for a touchkey can be applied to this baseline, so between 65 % and 55 % to enter in detection and between 35 % and 45 % to stop reporting a detection.

Balancing between channels

While a significant difference of sensitivity appear on the channels, the STMTouch driver provides a way to balance the delta in order to minimize the error on the position computation.

The sensitivity is determined with the maximum delta on each channel.

The figure below shows a log from a rotary sensor with an excessive difference of sensitivity.

Figure 8. Sensor log before balancing
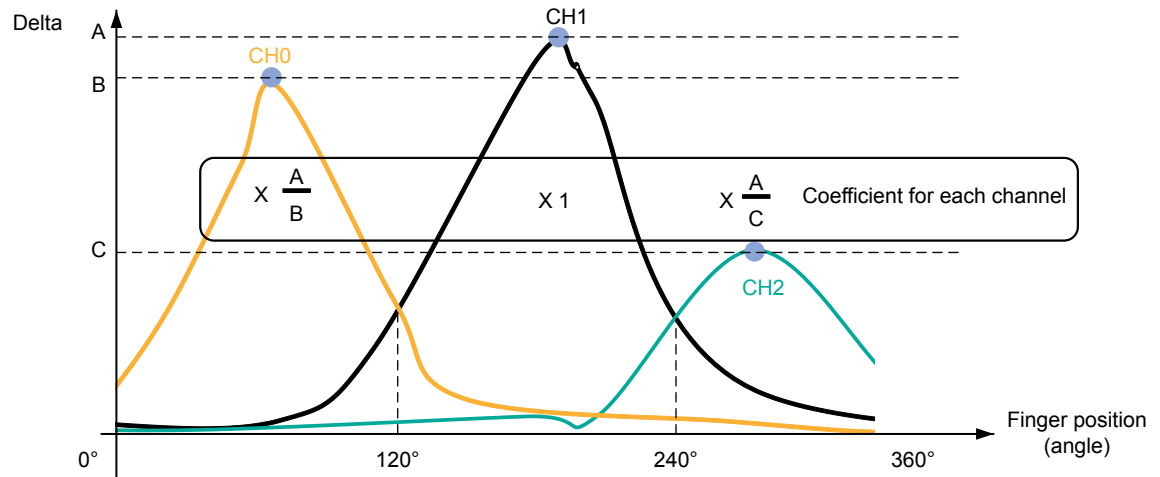
To get well-balanced channels, a coefficient must be applied to each channel delta. This coefficient is the ratio between the maximum deltas of each channel. The reference channel is the one with the highest delta (A for channel 1), the others channels have their maximum delta in B for channel 0 and C for channel 2. The coefficient of channel 0 must be set to A/B and the one of channel 2 to A/C. Channel 1 is not changed and gets its coefficient at 1.

These coefficients must be multiplied by 255 as they are used as a fraction of 255.

For instance:

- With A/B = 1.75, the coefficient is 488 = 0x1C0.

- With A/C = 3, the coefficient is 768 = 0x300 then the code is the following one:

```
CONST uint16_t MyLinRot0_DeltaCoeff[LINROT_CHANNELS] =
{
    0x1C0, 0x100, 0x300,// CH0, CH1, CH2
};
```

MyLinRot0_DeltaCoeff is pointed by the p_DeltaCoeff item in the declaration of the TSL_LinRot_T or TSL_LinRotB_T structure (see the code below).

```
CONST TSL_LinRotB_T MyLinRots[TSLPRM_TOTAL_LNRTS] =
{
    {
    &MyLinRots_Data[0],
    &MyLinRots_Params[0],
    &MyChannels_Data[CHANNEL_16_DEST],
    (TSL_tNb_T)LINROT_CHANNELS,
    MyLinRot0_DeltaCoeff,
    (TSL_tsignPosition_T *)TSL_POSOFF_3CH_LIN_INTERLACED,
    (TSL_tIndex_T) TSL_SCTCOMP_3CH_LIN_INTERLACED,
    (TSL_tIndex_T) TSL_POSCORR_3CH_LIN_INTERLACED
  }
};
```

3.2.3 Proximity

To define the proximity thresholds, the designer must consider the noise sensitivity and the expected detection distance, but also the minimum surface to detect.

3.3 Debounce settings

In order to improve the robustness of the application, the STMTouch driver provides the debounce feature. To validate a touch detection, the delta must exceed the threshold during a certain number of consecutive samples as shown in the figure below. This is to avoid a false-detection due to a noise peak.

Figure 9. Debouncing example
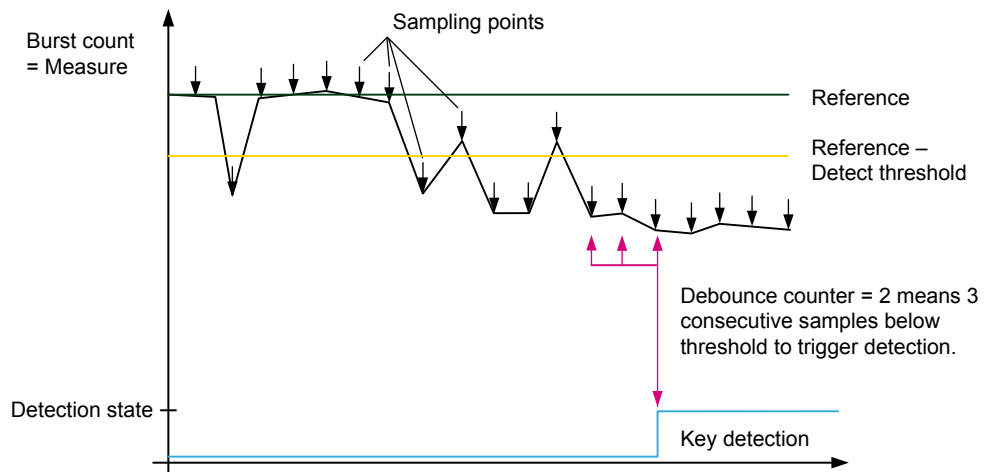
Due to the use of a down-counter, the DEBOUNCE preprocessor constant must be set to n-1 while n consecutive sample must be measured below the detection threshold before triggering a detection (see the code below).

```
#define TSLPRM_DEBOUNCE_DETECT (2) //3 consecutive samples needed to enter in Detection.
```

The debounce feature is configurable for each state transition listed below:

- TSLPRM_DEBOUNCE_PROX: while switching from release state to proximity state
- TSLPRM_DEBOUNCE_DETECT: while switching from release state or proximity to touch detection state
- TSLPRM_DEBOUNCE_RELEASE: while switching from touch or proximity state to release state
- TSLPRM_DEBOUNCE_CALIB: while switching from release state to calibration state (compute again the reference)
- TSLPRM_DEBOUNCE_ERROR: while switching from any state to error state

4 Charge transfer period tuning

The acquisition is based on the measurement of the sensor channel capacitance (or a set of sensors in the case of linear and rotary sensors). The more charged this sensor capacitance is, the more accurate the measure and the better the noise immunity.

To ensure that the capacitance is correctly charged, it is necessary to monitor the pin connected to the sensor plate or through a metallic coin put on the sensor (see the figure below).

Figure 10. Metallic coin probe

The signal must show a square wave that indicates a fully loaded capacitance.

Figure 11 shows an example of ideal charge transfers and Figure 12 an example of non-ideal transfers.

Figure 11. Ideal charge transfer
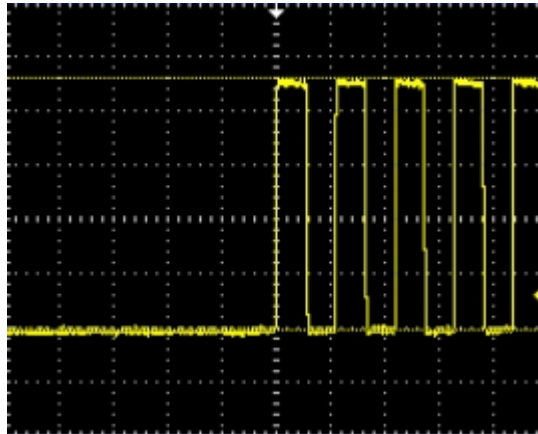
Figure 12. Non-ideal charge transfers
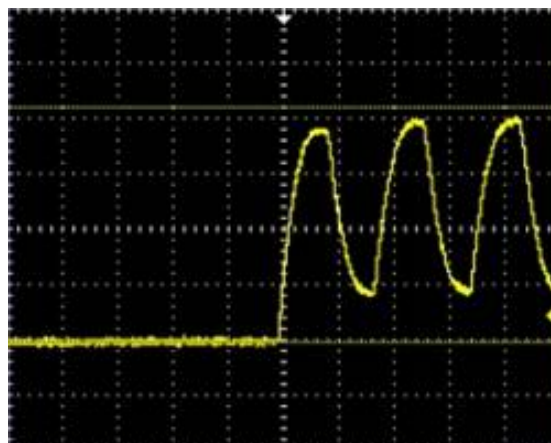
In case of an uncompleted charge, the user must increase the charge transfer period.

Depending on the product and type of acquisition, different adjustments are necessary.

The following involved parameters are in the MCU PARAMETERS section of the tsl_conf_stmxxxx.h file:

- For microcontrollers of STM32F0, STM32F3, STM32L0, STM32L1 and STM32WB Series embedding the TSC peripheral, the trimming is done by increasing TSLPRM_TSC_CTPH for the charge period and/or TSLPRM_TSC_CTPL for the transfer period, and optionally TSLPRM_TSC_PGPSC to divide the pulse generator frequency by a power of two.
- For STM32L1 products, the charge transfer period is set through TSLPRM_CT_PERIOD and TSLPRM_TIMER_FREQ parameters.

Refer to reference manuals of the products for more details on registers.

5 Hardware trimming

5.1 C_S trimming

The C_S capacitance is a key parameter for sensitivity. For touchkey sensors, the C_S value is usually between 8.7 nF to 22 nF. For linear and rotary touch sensors, the value is between 47 nF and 100 nF. These values are given as reference for an electrode fitting a human finger tip size across a few millimeters dielectric panel.

The signal delta for a touchkey is usually above 20 while it is around 100 for linear and rotary touch sensors. These values are given for a normalized test finger.

5.2 Shield adjustment

The efficiency of the shield depends on the waveforms matching between the shielded channel and the shield burst pulses. The parameters to adapt the shield waveform are Cs and Rs, which can be adjusted through the two following steps:

1. Active shield Cs trimming: The burst envelop of the channels belonging to a same bank must end at the same voltage level V_{IH} (see the figure below).

Figure 13. Active shield Cs trimming

C_S must be trimmed to reach V_{IH} at the end of the burst.
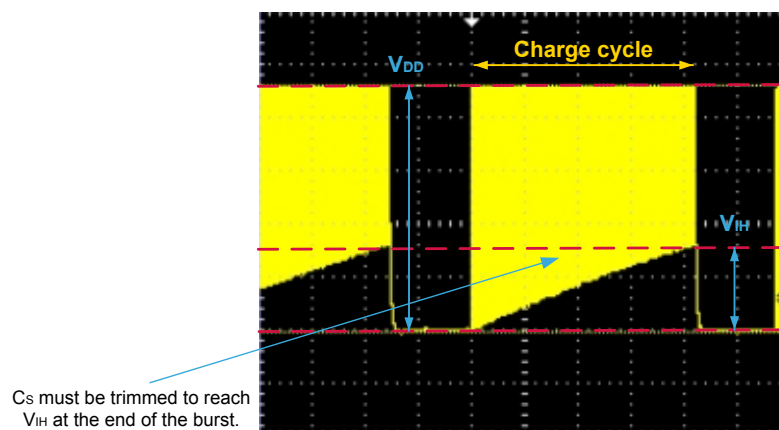
2. Active shield Rs trimming: Rs must be trimmed to ensure shield capacitance is fully charged (see the figure below).

Figure 14. Active shield Rs trimming

R_S must be trimmed to obtain the same pulse waveform.
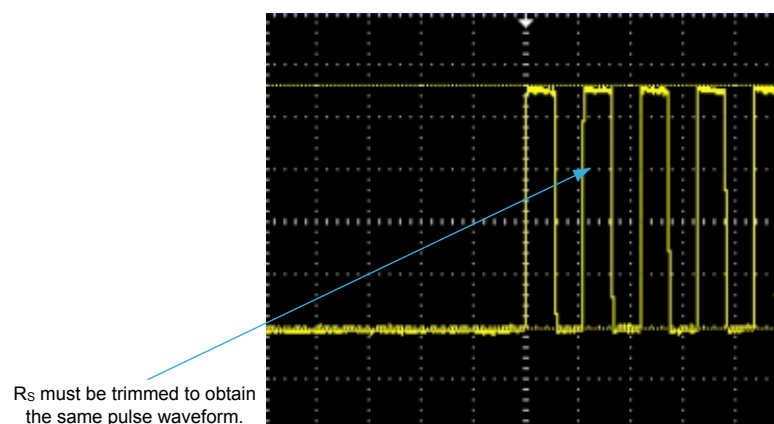
6 Performance comparison

In order to compare the performance of a whole touch sensing application, the usual parameter is the signal-to-noise ratio (SNR). Each company, each team and even each engineer as its own method to compute it.

Here is a proposal to compute the SNR but, to get correct results, the user must compare apples to apples. The test conditions and the method of computation must be under control and reproducible else it is meaningless to compare two results.

In the SNR calculation, the signal is the average of the delta measurements during a touch and the noise is the amplitude (delta max - delta min) without touch (see the figure below).

Figure 15. **SNR computation**
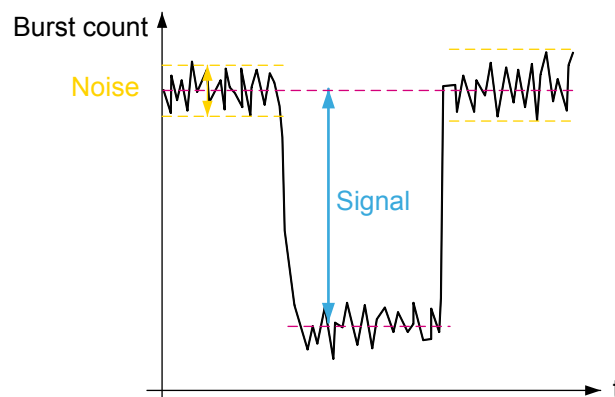
In order to make meaningful comparisons, the various SNR measurements must be performed under similar test conditions.

The following parameters affect the results of the SNR:

- Hardware application: layout, panel (such as dielectric, thickness or glue), capacitance value and quality
- Firmware application: acquisition configuration (such as frequency or reference), threshold settings
- Test conditions: object used to touch (such as standard test finger or genuine finger), the way the panel is touched (such as pressure or slope), the applied noise if any

7 Conclusion

In order to get the best performance for any STMTouch-based application, the designer must tune it correctly. STMicroelectronics provides the STM-Studio free tools to help performing this task. It is important to have the thresholds and the debounce values set according to the application environment. To get the best performance, the charge transfer must be operated completely.

Revision history

Table 2. Document revision history

Date	Version	Changes
4-Mar-2014	1	Initial release.
11-Jun-2014	2	Added support for STM32L0 series and STM8AL series.
15-Oct-2015	3	Added support for STM32L4 series.
24-Jan-2019	4	Updated: • Title of the document • All section numbers (as the Introduction is now not numbered) • Introduction and Table 1 • Section 2 Monitoring STMTouch driver variables using STM-Studio • Section 4 Charge transfer period tuning Appendix A removed and corresponding figure moved to Section 3.1 Use of a standard test finger.
12-Aug-2021	5	Updated: • Table 1. Applicable products • Removed all information relative to STM8 MCUs

Contents

List of figures

IMPORTANT NOTICE – PLEASE READ CAREFULLY