Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CE228931 – PSoC 4 CapSense CSX Tuning

Objective

This code example demonstrates how to manually tune a mutual capacitance (CSX)-based button widget in PSoC® 4 devices using CapSense® Tuner GUI.

Requirements

Tools: PSoC Creator™ 4.2, PSoC Programmer

Programming Language: C (Arm® GCC 5.4.1)

Associated Parts: All PSoC 4 MCU parts that supports CapSense CSX sensing

Related Hardware: CY8CKIT-149 PSoC 4100S Plus Prototyping Kit, CY8CKIT-041-41XX PSoC 4100S Pioneer Kit, CY8CKIT-041-40XX PSoC 4000S Pioneer Kit, CY8CKIT-145-40XX PSoC® 4000S CapSense Prototyping Kit

Overview

This example demonstrates how to manually tune a CSX- based button using CapSense Tuner GUI.

This document details the following:

1. A high-level overview of the CSX widget Tuning flow
2. An example to show how to manually tune a CSX button widget
3. How to use the CapSense Tuner GUI to monitor the CapSense raw data and fine-tune the CSX button for optimum performance such as reliability, power consumption, and response time

The code scans a single button widget using the CSX sensing method. The code also sends the CapSense raw data over an I^2C interface to the on-board KitProg which in turn enables reading the data from CapSense Tuner GUI.

Hardware Setup

The example project is set up for CY8CKIT-149 PSoC 4100S Plus Prototyping Kit. The project can be migrated to any supported kits as listed in Table 1 by changing the target device with Device Selector called from the project's context menu.

Table 1. Supported Development Kits

Development Kit	Series	Device
CY8CKIT-149	PSoC 4100 S Plus	CY8C4147AZI-S475
CY8CKIT-041-40XX	PSoC 4000 S	CY8C4045AZI-S413
CY8CKIT-041-41XX	PSoC 4100 S	CY8C4146AZI-S433
CY8CKIT-145	PSoC 4000 S	CY8C4045AZI-S413

The pin assignments for the supported kits are provided in Table 2. For these kits, the project includes control files to automatically assign pins with respect to the kit hardware connections during the project build. To change the pin assignments, override the control file selections in the Pin Editor of the Design Wide Resources by selecting the new port or pin number.

Table 2. Pin Assignments

Pin Name	Development Kit			
	CY8CKIT – 041 – 40XX PSoC 4S Series Pioneer Kit	CY8CKIT-041-41XX PSoC 4100S Pioneer Kit	CY8CKIT-145-40XX PSoC 4000S CapSense Prototyping Kit	CY8CKIT-149 PSoC 4100S Plus Prototyping Kit
CapSense:CintA	P4[2]	P4[2]	P4[2]	P4[2]
CapSense:CintB	P4[3]	P4[3]	P4[3]	P4[3]
CapSense:Rx (BTN0_Rx0)	P0[1]	P0[1]	P1[4]	P4[6]
CapSense:Tx (BTN0_Tx)	P3[7]	P3[7]	P1[3]	P0[2]
EZI2C:scl	P3[0]	P3[0]	P1[0]	P3[0]
EZI2C:sda	P3[1]	P3[1]	P1[1]	P3[1]

Note: If you are using the CY8CKIT-145-40XX PSoC 4000S CapSense Prototyping kit, the pin assignment must be done manually.

Software Setup

Ensure that you have all the software tools as mentioned in Requirements section installed.

Pre-requisites

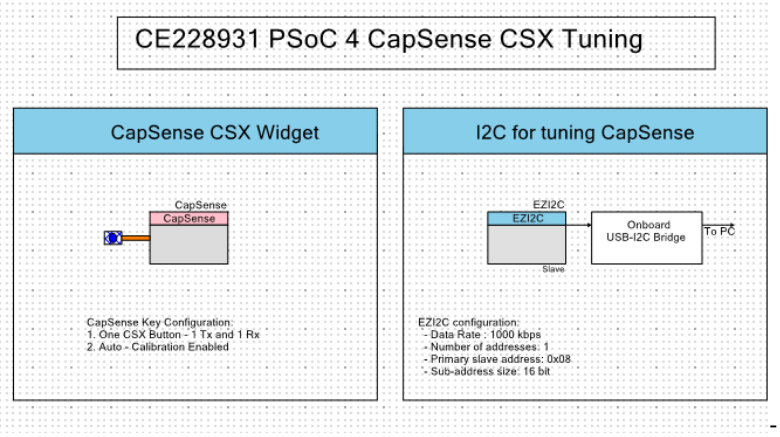
The user of this code example is assumed to be familiar with CapSense technology and basic terms such as raw counts, baseline, and difference counts. If you are new to CapSense, see the application note AN85951 – PSoC 4 and PSoC 6 MCU CapSense Design Guide.

Design and Implementation

This code example has a single workspace, CE228931.

Figure 1 shows the PSoC Creator schematics for the “CE228931_PSoC4_CapSense_CSX_Tuning” code example. This code example uses CapSense and SCB (configured as EZI2C slave) Components.

Figure 1. PSoC Creator Schematics



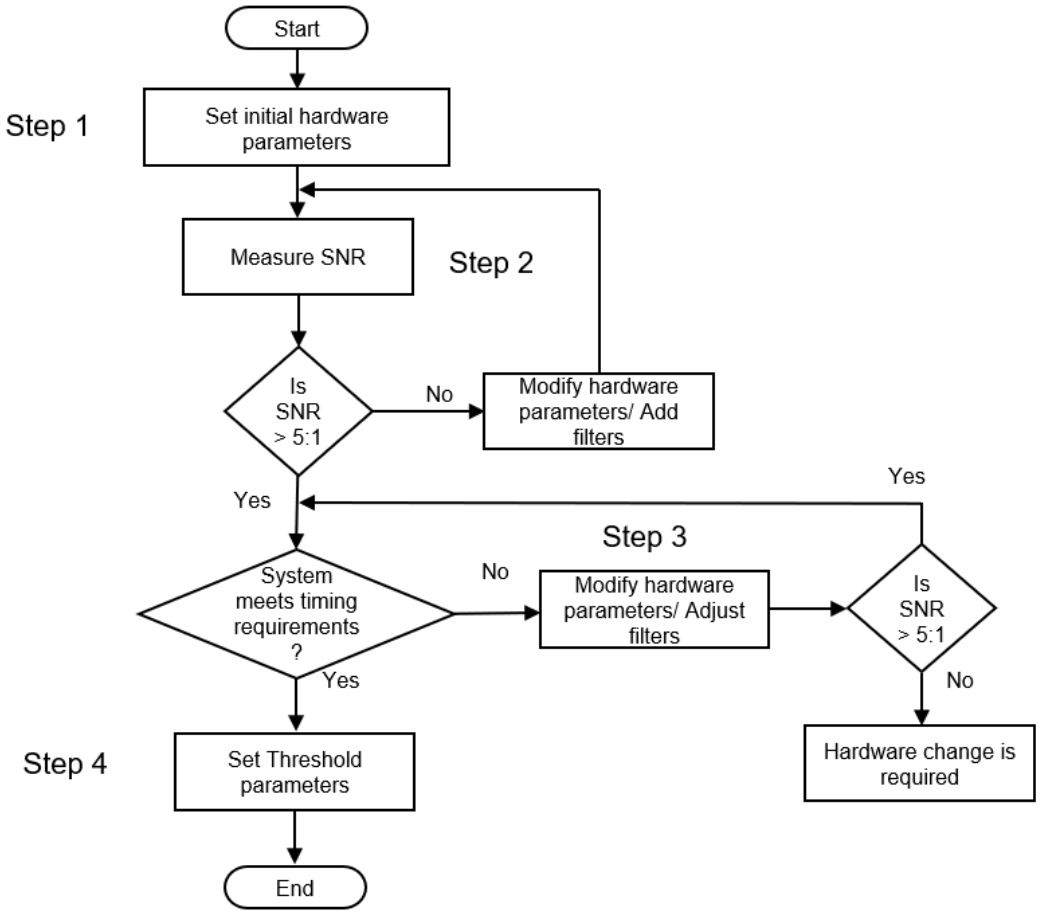
The project’s top design contains a CapSense Component that has a single button widget configured in CSX sensing mode. See the section “CapSense CSX Sensing Method” in the PSoC 4 and PSoC 6 MCU CapSense Design Guide. The step-by-step instructions to configure the other settings of the CapSense Component is explained in the Operation section of this document.

The CapSense Component provides a GUI-based tuner application for debugging and tuning the CapSense system. The CapSense Tuner application works with the EZI2C and UART Communication Components. This project has an SCB block configured in EZI2C mode to establish communication with the on-board KitProg, which in turn enables reading the CapSense raw data by the Tuner GUI. See Configure EZI2C Component.

The CapSense data structure that contains the CapSense raw data is exposed to the Tuner GUI by setting up the I2C communication data buffer with the CapSense data structure. This enables the Tuner GUI to access the CapSense raw data for tuning and debugging CapSense data.

Tuning Flow Summary

The following flowchart gives a high-level summary on how to tune a CSX based CapSense button in PSoC 4 devices. See the "Manual Tuning" section in AN85951 – PSoC 4 and PSoC 6 MCU CapSense Design Guide for information on the hardware and threshold parameters that determines the CapSense touch performance.



To manually tune CSX button, see the Operation section.

Operation

Set Initial Hardware Parameters

1. Power the kit by plugging the kit board into your computer's USB port.
2. In the **Workspace Explorer**, double-click on the *TopDesign.cysch* file to open the schematic editor window.
3. Double-click the CapSense Component, or right-click the CapSense Component and select **Configure** to open the CapSense Component configuration window.

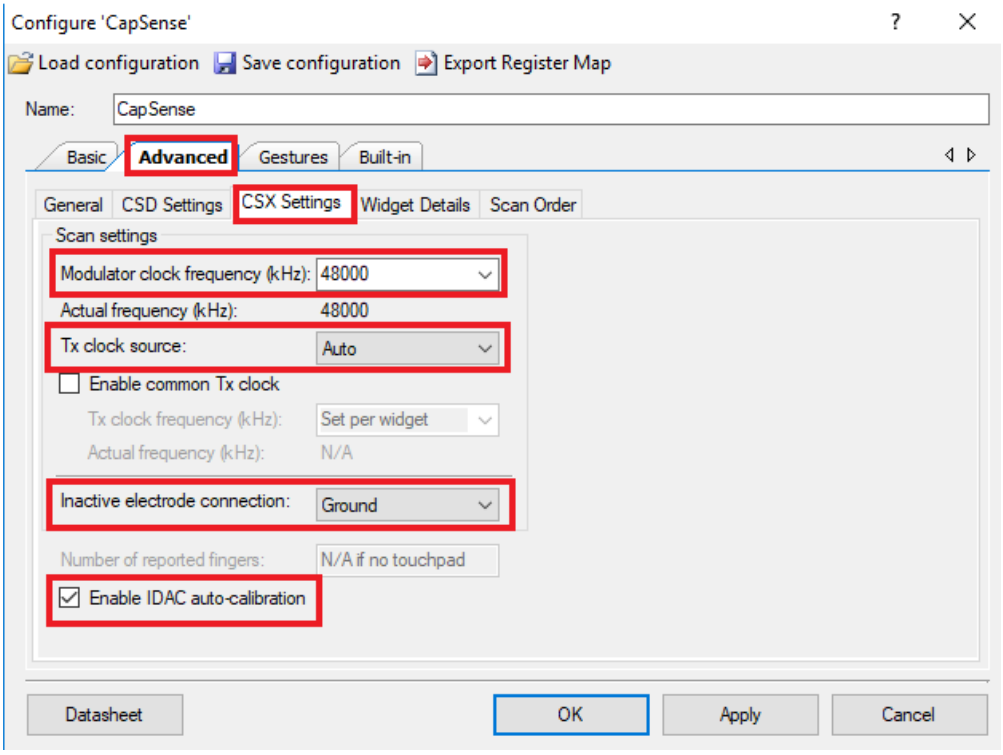
   In the **Basics** tab, you will find a single widget 'BTN0' configured as a CSX button.

   Note: In this example, a single sensor is implemented. Thus, the number of Rx Sensing element(s) has been kept at 1. For multiple sensors, which use the same Tx Sensing element, the Rx Sensing element(s) count can be increased. That is, one Tx element can be used to measure multiple Rx elements.
4. Navigate to the **Advanced** tab - **General** settings sub-tab. Leave all the filter parameters at their default settings. Filters will be enabled depending on the SNR and system time requirements in Step 19.
5. Select **Advanced** tab - **CSX Settings** sub-tab and configure the parameters as Table 3 and Figure 2 show.

Table 3. CSX Settings

Parameter	Value	Remarks
Modulator clock frequency (kHz)	Maximum allowed by the selected device	Higher modulator clock frequency reduces Flat spot , increases measurement accuracy and sensitivity. Thus, it is recommended to select the highest possible available Modulator clock frequency.
Tx clock source	Auto	Enabling Auto (chooses between spread spectrum clock or direct clock). SSCx clock source option is available only for PSoC 4 S-Series, PSoC 4100S Plus and PSoC 4100PS. It helps to deal with EMC/EMI. For other devices, the Tx clock is always Direct.
Enable common Tx clock	Unchecked	When selected, all CSX widgets share the same Tx clock with the frequency specified in the Tx clock frequency (kHz) parameter. Otherwise, the Tx clock frequency is entered separately for each CSX widget in the Widget Details tab. If the electrode capacitance of all the CSX widgets is the same, using the common Tx clock for all CSX widgets results in lower power consumption and optimized memory usage; it is the recommended setting for CSX widgets. However, if the electrode capacitance is significantly different for each widget, a common Tx clock may not produce the optimal performance.
Inactive electrode connection	Ground	Inactive sensors are connected to ground to provide good shielding from noise sources.
Enable IDAC auto-calibration	Checked	Enabling auto-calibration allows the device to automatically choose the optimal IDAC calibration point (for CSX, this is 40 percent of the maximum value).

Figure 2. Advanced - CSX Settings



Note: The Modulator Clock Frequency can be changed to 48,000 kHz only after the IMO clock is changed to 48 MHz. Under **Design Wide Resources**, click **Clocks** > **System IMO**. Select the drop down on the IMO Clock and select **48 MHz**.

6. Navigate to the **Advanced** tab > **Widget Details** window. Set the Tx clock frequency in such a way that it completely charges and discharges the sensor parasitic capacitance for maximum sensitivity. It can be verified by checking the signal in an oscilloscope or it can be set using the Equation 1.

Equation 1. Condition for Selecting Tx Clock Frequency

$$F_{Tx} < \frac{1}{10 R_{SeriesTx} C_{PTx}}$$

C_{PTx} is the parasitic capacitance of the Tx electrode. $R_{SeriesTx}$ is the total series-resistance, including the 500-ohm resistance of the internal switches, the recommended external series resistance of 2 kΩ (connected on PCB trace connecting sensor pad to the device pin), and trace resistance if using highly resistive materials (example ITO or conductive ink); i.e., a total of 2.5 kΩ plus the trace resistance.

To calculate the maximum Tx Clock frequency:

a. Estimate the Cp of the Tx electrode and the $R_{SeriesTx}$. The C_P of the Tx electrode can either be measured using an LCR meter or estimated using the Built-in-Self-test (BIST) API; `CapSense_GetSensorCapacitance()`. See the CapSense Component datasheet for details.

   Note: BIST feature is available only from CapSense Component v3.10 and above.

b. Calculate the maximum Tx clock frequency using Equation 1.

c. Set the nearest possible Tx clock frequency as supported by the device in the CapSense Configurator. The maximum value of F_{Tx} depends on the selected device. For the PSoC 4 S-Series, PSoC 4100S Plus, and PSoC 4100P family of devices, the maximum F_{Tx} is 3000 kHz and for other devices it is 300 kHz.

Using BIST, the C_P of the Tx electrodes was estimated to be as shown in Table 4. Set the Tx Clock frequency in the configurator as Table 4 shows.

Table 4. Cp of Tx Electrode, Calculated Maximum Frequency and Tx Frequency Setting in Configurator

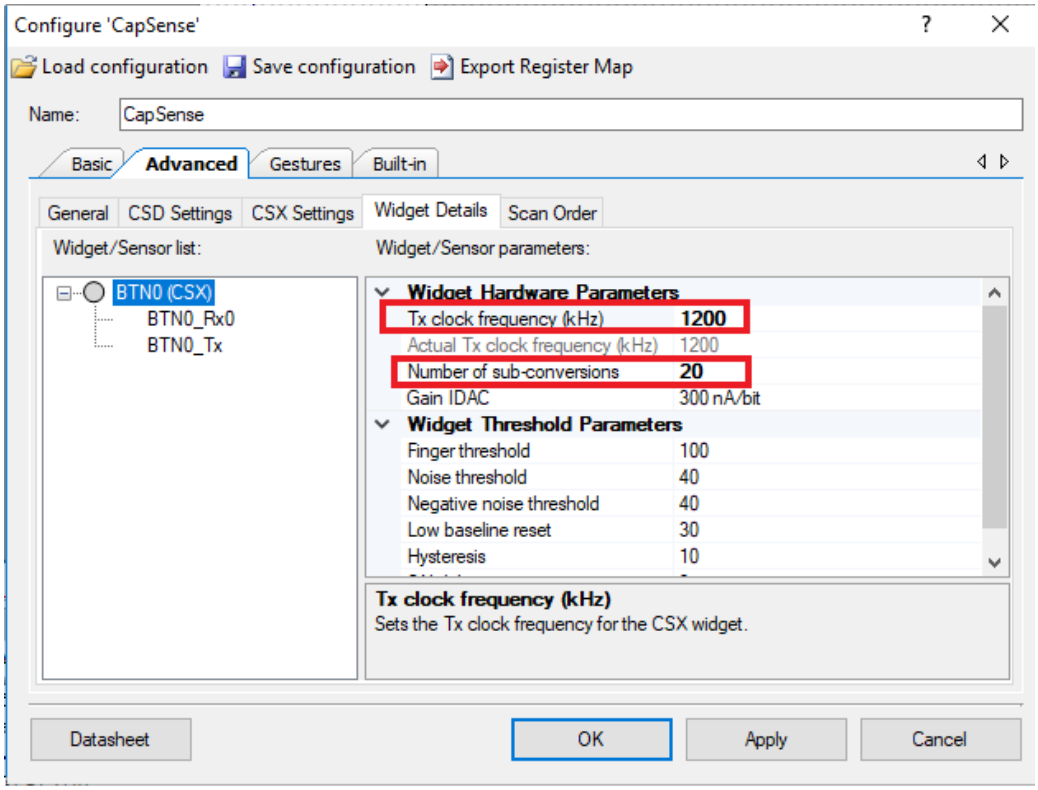
Development kit	C_P of Tx Electrode (pF)	Calculated Maximum Frequency (kHz)	Tx Clock Setting in Configurator (kHz)
CY8CKIT-149	31	1290	1200
CY8CKIT-041-41XX	37	1081	1000
CY8CKIT-041-41XX	37	1081	1000
CY8CKIT-145	21	1904	1500

In addition to the condition mentioned above, you should ensure the following:

- The auto-calibrated IDAC code should lie in the mid-range (for example, 30-90) for the selected F_{Tx}. If the auto-calibrated IDAC code lies out of the recommended range, F_{Tx} is tuned such that IDAC falls in the recommended range which will be explained in section Tuning Tx Clock Frequency to Get Desired IDAC Value.
- If you are using the SSCx clock source, ensure that you select the Tx clock frequency that meets the conditions mentioned in CapSense Component datasheet in addition to these conditions.

7. Set the number of sub-conversions to an initial low value of 20. This will be modified in Step 19 and 20 based on Signal-to-Noise ratio (SNR) and system timing requirements. Leave all other values in the tab to their default settings.

Figure 3. Advanced tab - Widget Details Sub-Tab



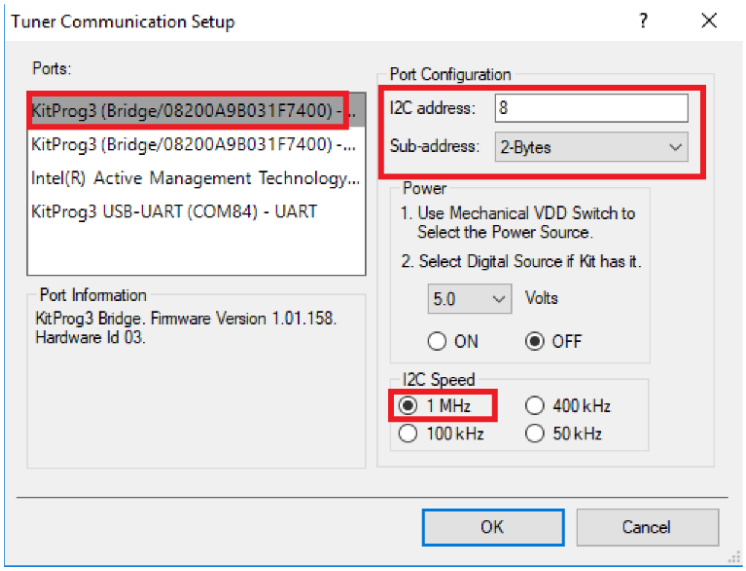
Build and Program

8. Build and Program the project onto the kit. Select **Debug** > **Program.** This builds the project and then programs your kit.

Set up CapSense Tuner GUI to View Sensor Data

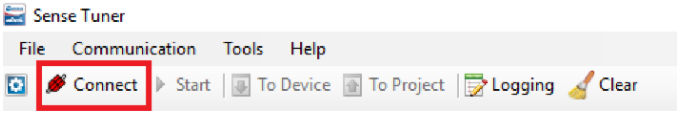
9. In the *TopDesign.cysch* file, right-click the **CapSense** Component and select **Launch Tuner**.
10. Go to **Tools** > **Tuner Communication Setup**… and set the parameters as Figure 4 shows and click **OK**.

Figure 4. Tuner Communication Setup



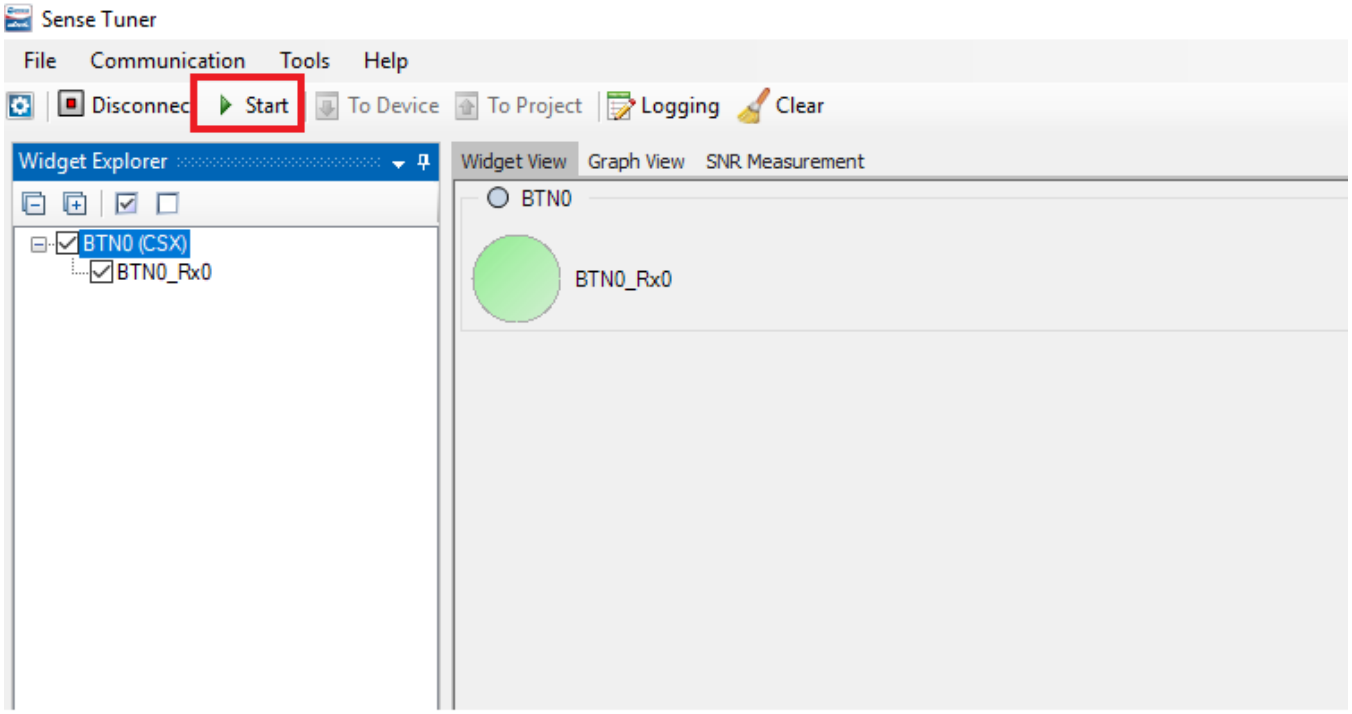
11. Click **Connect** as Figure 5 shows.

Figure 5. CapSense Tuner Window



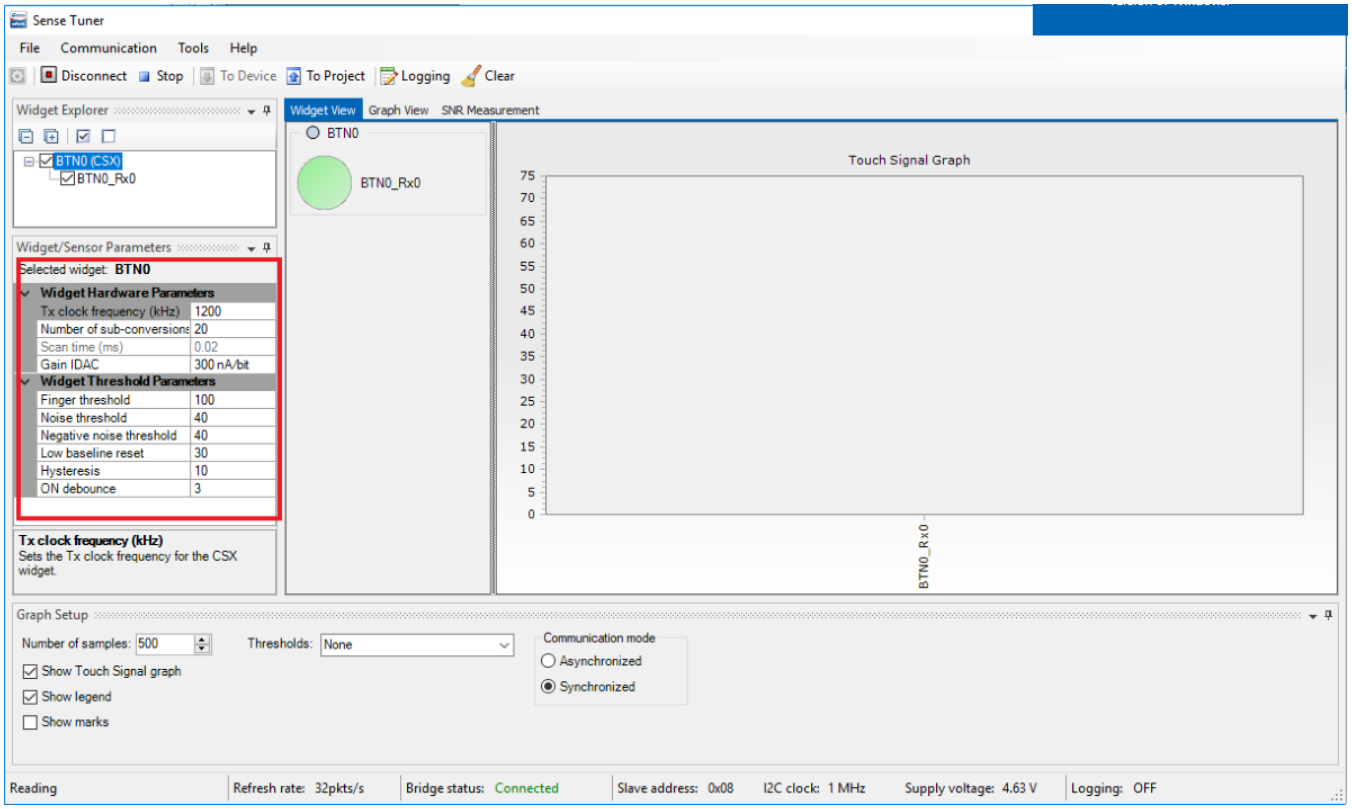
12. Click **Start** as Figure 6 shows.

Figure 6. Starting Tuning with CapSense Tuner



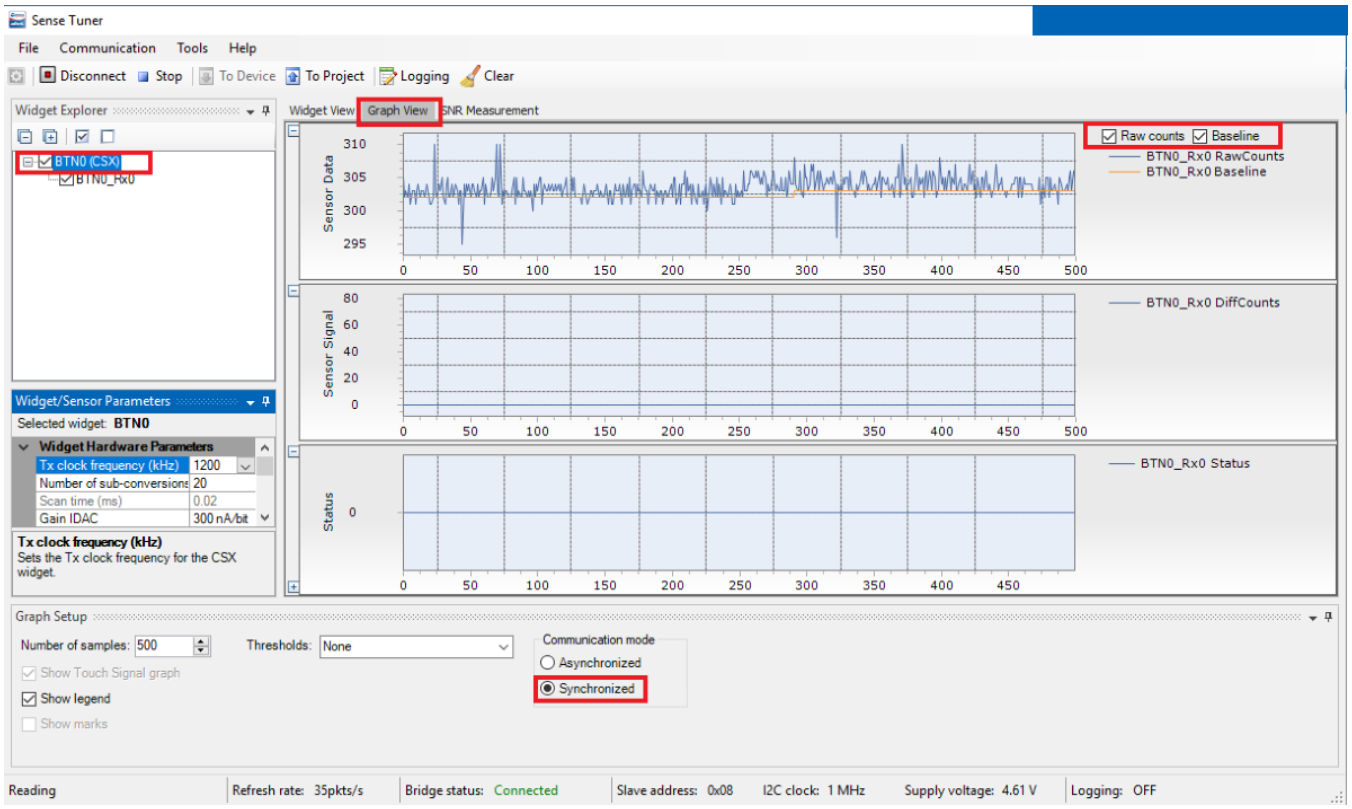
13. Figure 7 shows that the Widget/Sensor Parameters get updated with the parameters configured in the CapSense Component Configuration window in steps 5 to 7.

Figure 7. Widget/Sensor Parameters Updated in Sense Tuner



14. Select the **BTN0_Rx0** check box, select **Synchronized** under **Communication mode**, and then navigate to the Graph View as Figure 8 shows. The graph view displays the raw counts and baseline for Button0_Rx0 in the Sensor Data window. Ensure that the **Raw Counts** and **Baseline** checkboxes are selected to view the sensor data.

Figure 8. Graph View of Tuner GUI

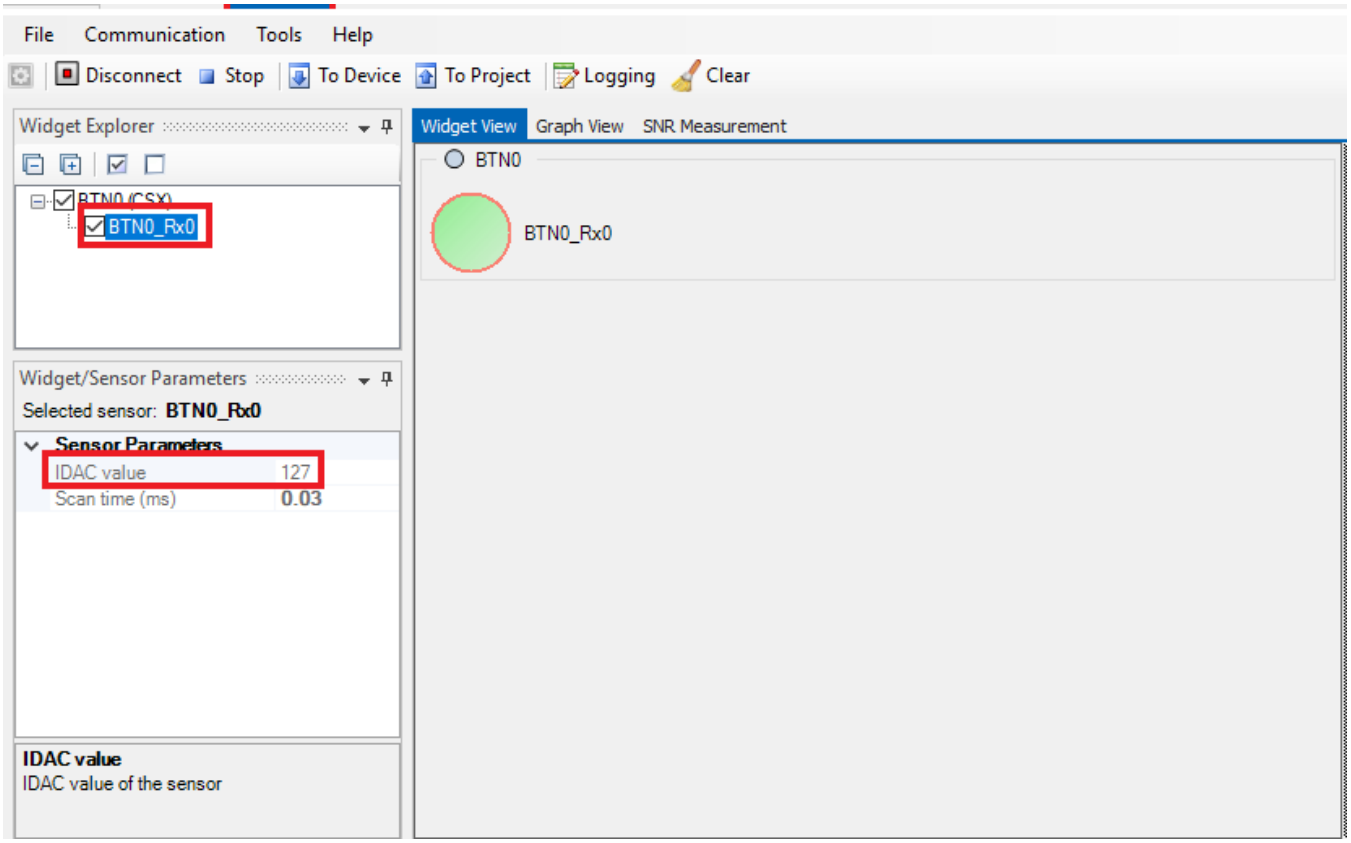


Note: At this point, when the configured button is touched, you may or may not notice the touch signal in the Sensor Signal graph. There are chances that the sensor may false trigger which can be seen in the touch status going from 0 to 1 in the Status window.

Tuning Tx Clock Frequency to Get Desired IDAC Value

15. As discussed in Step 6, the Tx Clock frequency will be tuned to bring IDAC code to the recommended range in this step. Click on **BTN0_Tx0** in the Widget Explorer to view the IDAC value in the sensor parameters window as shown in Figure 9. If the IDAC value is within the range 30 to 90, skip to Step 17; if not, modify Tx clock frequency until the IDAC is in the desired range.

Figure 9. IDAC Value



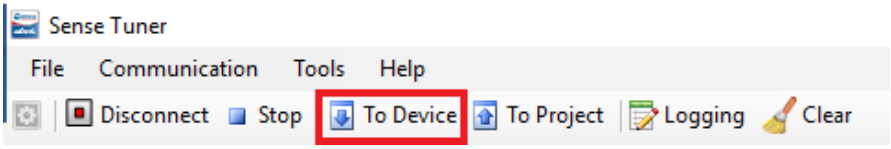
16. Do the following to modify the Tx Clock frequency:

    a. Click on **BTN0** in the Widget Explorer.

    b. Increase or decrease the Tx Clock frequency in the **Widget Hardware Parameters** window. Decreasing the Tx clock frequency will decrease the IDAC value for a fixed IDAC gain and calibration percent and vice versa.

    c. Click **To Device** to apply the changes to the device as shown in Figure 10.

Figure 10. Apply Changes to Device



    d. Observe the IDAC value in the Sensor Parameters section of Widget/Sensor Parameters window.

    Repeat steps 17 a to 17 d until you obtain the IDAC value in the range of 30 to 90. After performing these steps, you will arrive at the following Tx Clock frequency for different development kits as shown in Table 5.

Table 5. Final Tx Clock Frequency Setting for Different Development Kits

Development Kit	Tx Clock Frequency (kHz)
CY8CKIT-149	600
CY8CKIT-041-41XX	300
CY8CKIT-041-41XX	300
CY8CKIT-145	300

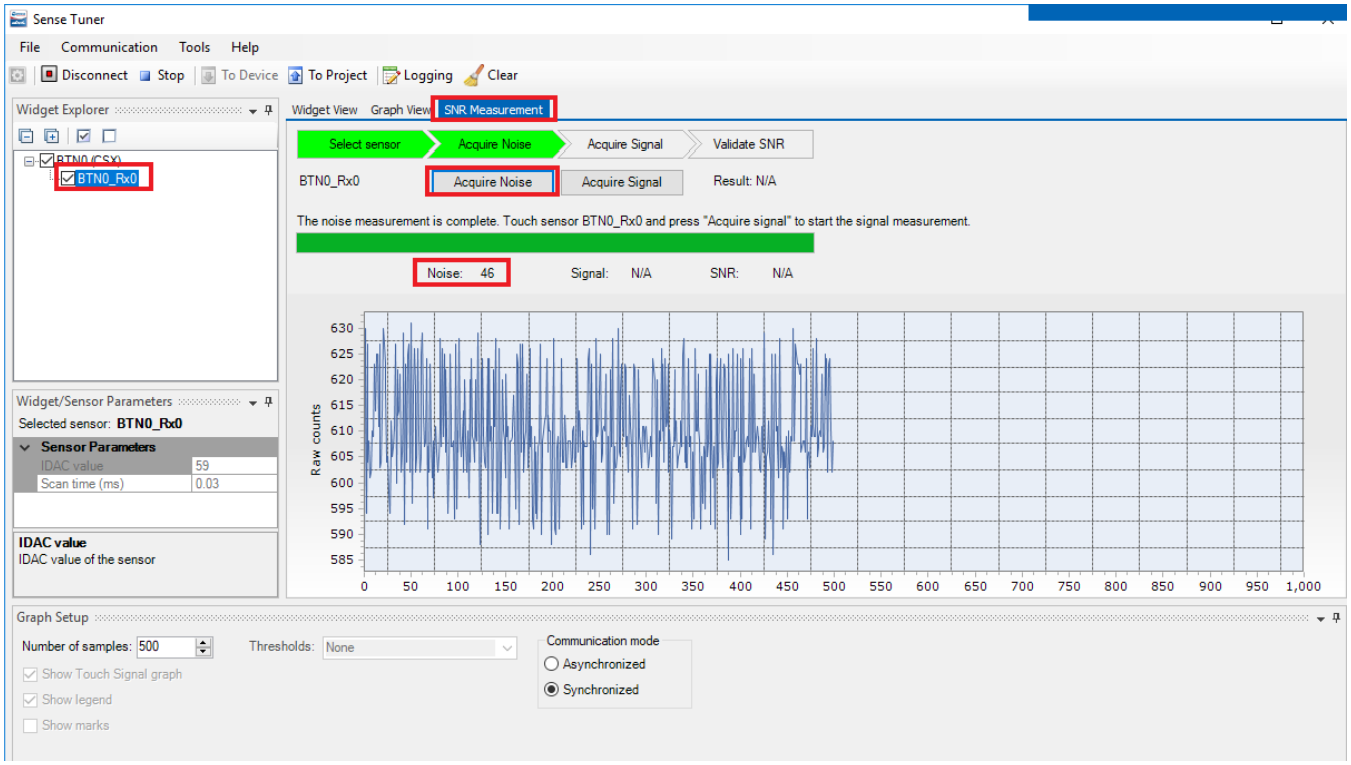
Note: Decreasing the Tx Clock frequency will increase the scan time of the sensor. If your system doesn't meet the scan time requirements, you can bring the IDAC value to the recommended range by varying the IDAC gain parameter in the **Widget Hardware Parameters** window.

Use CapSense Tuner to Fine-tune Sensitivity for 5:1 SNR

The CapSense system may be required to work reliably in adverse conditions such as a noisy environment. Buttons need to be tuned with SNR > 5 :1 to avoid triggering false touches and ensure that all intended touches are registered in these adverse conditions.

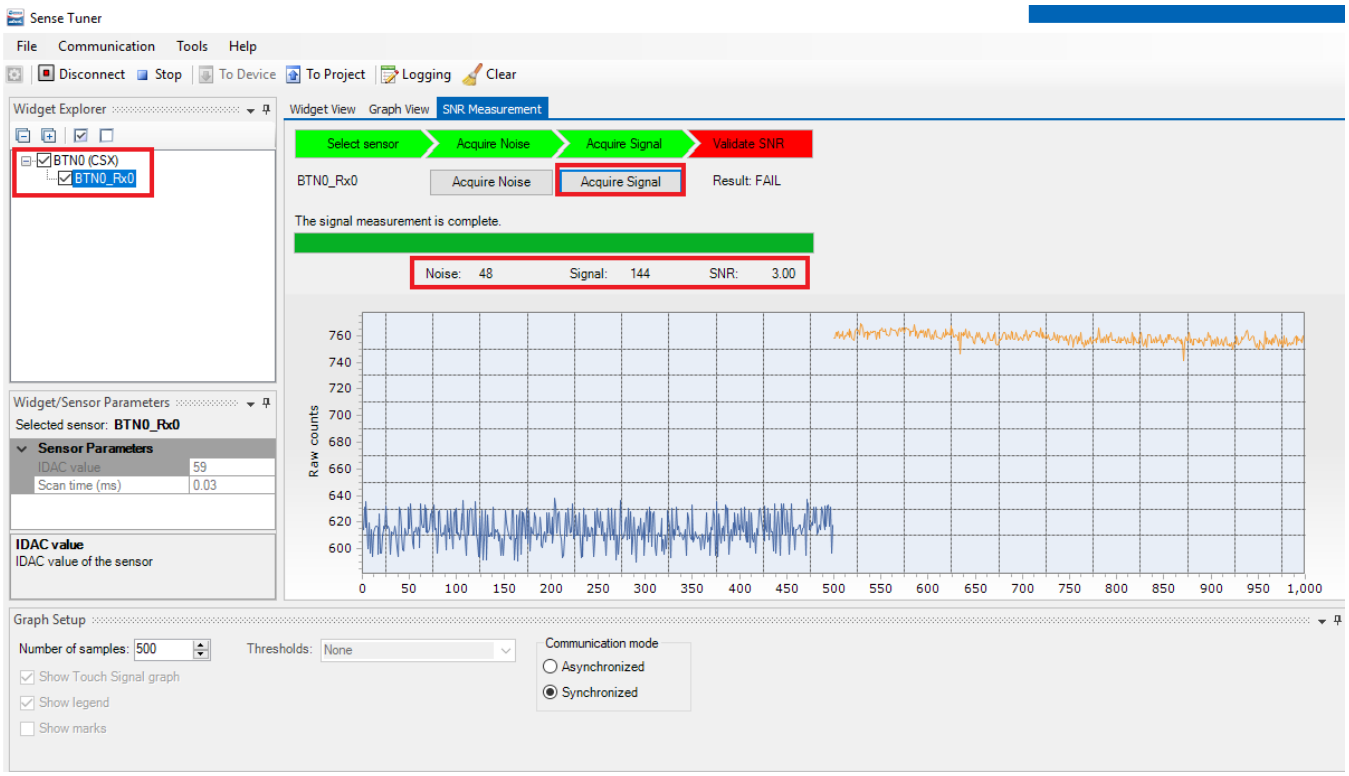
17. Switch to the **SNR Measurement tab**, select the **BTN0_Rx0** sensor, and then click **Acquire Noise** as Figure 11 shows.

Figure 11. SNR Measurement Using Tuner GUI



18. Once the noise is acquired, touch the button on the kit, and then click **Acquire Signal**. Ensure that the button is touched as long as the signal acquisition is in progress. You will now be able to see the calculated SNR on this button as Figure 11 shows. Based on your end-system design, test with a finger that matches the size of your normal use case. Typically, finger size targets are ~8 to 9 mm.

Figure 12. Acquiring Signal and Calculating SNR Using Tuner



19. If the SNR is greater than 5, skip to Step 20. If the SNR is less than 5, you must optimize the parameters to achieve a minimum of 5:1 SNR.

    Number of sub-conversions: Increasing the number of sub-conversions increases the signal without increasing the noise at the same rate. Increasing the sub-conversions will in turn will increase the sensor scan time.

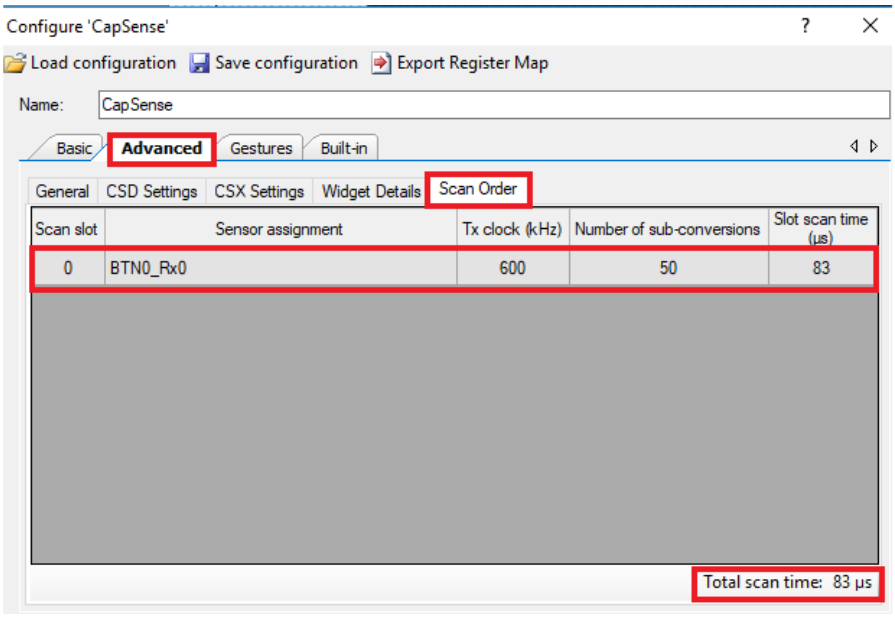
    Filters - Filters help reduce the noise without increasing the signal. Adding a filter adds to the processing time and memory usage, because this is implemented in firmware and gets executed by the CPU. This also results in increased power consumption. For more details on filters and selecting the filter coefficients, see the CapSense Component datasheet.

    It is best to find a balance between the number of sub-conversions and filters to achieve proper overall tuning. If your system is very noisy (counts > 20), you may want to prioritize adding a filter. On the other hand, if your system is relatively noise-free (counts < 10), you will want to focus on resolution, as this will increase the sensitivity and signal of your system.

    Note: The number of sub-conversions can be updated directly in the **Widget/Sensor Parameters** tab of the CapSense Tuner GUI, but to adjust the filter settings, you will need to open up the CapSense Configurator and select the appropriate filter in Step 4, and reprogram the device to update filter settings.

20. If the total sensor scan time meets your requirements, skip to Step 22 in Use CapSense Tuner to Tune Threshold Parameters. Otherwise, adjust the tuning to speed up the scan time. If SNR is greater than 10 on any sensor, you can lower the number of sub-conversions or remove filters to decrease scan time but keep the SNR greater than 5:1. The hardware scan time can be seen from the **Scan Order** tab in the **Advanced** settings of the CapSense Component as shown in Figure 13.

Figure 13. Hardware Scan Time of the Sensor



21. Use Table 6 to set the final hardware tuning parameters for the supported kits to achieve 5:1 SNR.

Table 6. Final Hardware Tuning Parameters to Achieve 5:1 SNR on Different Development Kits

Development Kit	Tx Clock Frequency (kHz)	Number of Sub-conversions
CY8CKIT-149	600	50
CY8CKIT-041-40XX	300	175
CY8CKIT-041-41XX	300	175
CY8CKIT-145	300	35

Use CapSense Tuner to Tune Threshold Parameters

After you have confirmed that your design meets the timing parameters, and the SNR is greater than 5:1, set your threshold parameters as follows:

22. Switch to the **Graph View** tab and ensure that **BTN0 (CSX)** is selected.

    a. Touch the sensor and monitor the touch signal in the **Sensor Signal** graph.

       The Sensor Signal graph should show the signal as Figure 14 shows. Ensure that you observe the difference count (that is **signal** output) in the **Graph View** tab in Figure 14, *not* the raw count output for setting these thresholds. Based on your end system design, test the signal with a finger that matches the size of your normal use case. Typically, finger size targets are ~8 to 9 mm. Consider testing with smaller sizes that should be rejected by the system to ensure that they do not reach the finger threshold.

    b. When the signal is measured, set the thresholds according to the following recommendations:

       Finger Threshold = 80 percent of signal

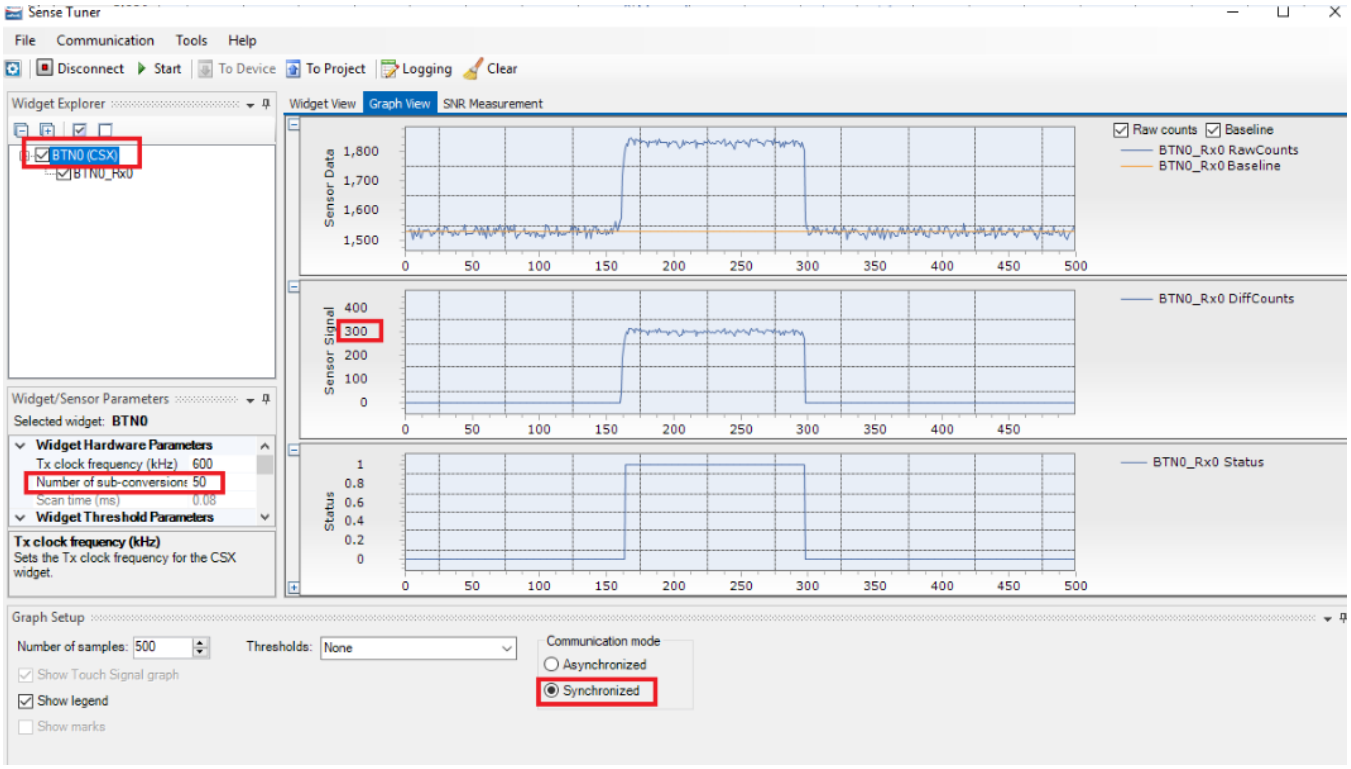
       Noise Threshold = 40 percent of signal

       Negative Noise Threshold = 40 percent of signal

       Hysteresis = 10 percent of signal

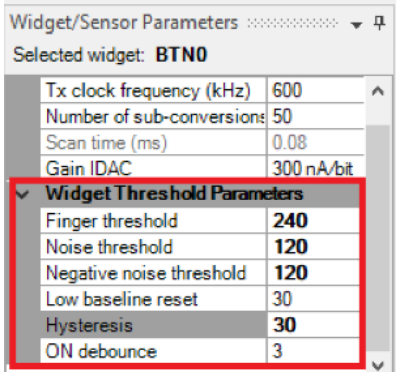
       Debounce = 3

Figure 14. Sensor Signal when the Sensor Is Touched



23. Set the threshold parameters in the **Widget/Sensor Parameters** section of the Tuner GUI, as Figure 15 shows.

Figure 15. Widget Threshold Parameters



See Table 7 to set the threshold parameters in the Tuner GUI for different development kits.

Table 7. Threshold Parameters for Different Development Kits

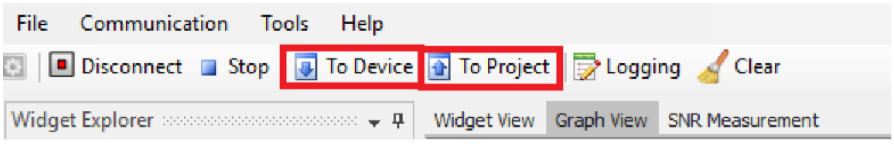
Development Kit	Finger Threshold	Noise Threshold	Negative Noise Threshold	Hysteresis	Low Baseline Reset	Debounce
CY8CKIT-149	240	120	120	30	30	3
CY8CKIT-041-40XX	160	80	80	20	30	3
CY8CKIT-041-41XX	160	80	80	20	30	3
CY8CKIT-145	140	70	70	18	30	3

If your sensor is tuned correctly, you will observe the touch status go from 0 to 1 in the **Status** sub-window of the **Graph View** window as Figure 14 shows.

Apply Settings to Firmware

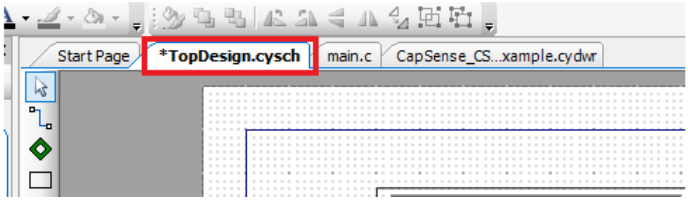
24. Apply the settings to the device and to the project by clicking on **To Device** and **To Project** as Figure 16 shows, and close the tuner.

Figure 16. Apply to Project



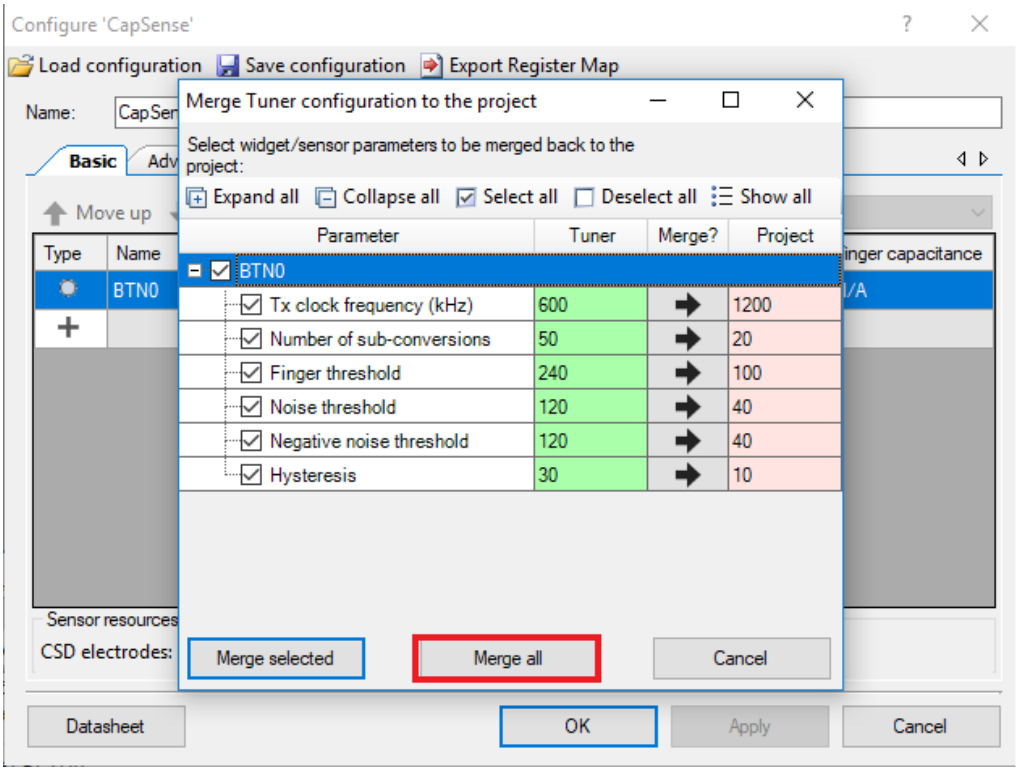
25. Notice the * sign on the **TopDesign** file, indicating that TopDesign has been updated as Figure 17 shows.

Figure 17. TopDesign Gets Updated when To Project Is Clicked on Tuner



26. Double-click the CapSense Component to open the configuration window and choose **Merge All** to accept the changes applied by the CapSense Tuner, as Figure 18 shows. Click **OK** to close the Component Configuration window.

Figure 18. Accept Tuner Modified Parameters in CapSense Component



Components and Settings

See the Operation section.

Configure EZI2C Component

Figure 19 shows the settings for the EZI2C Component.

Figure 19. EZI2C Component Settings

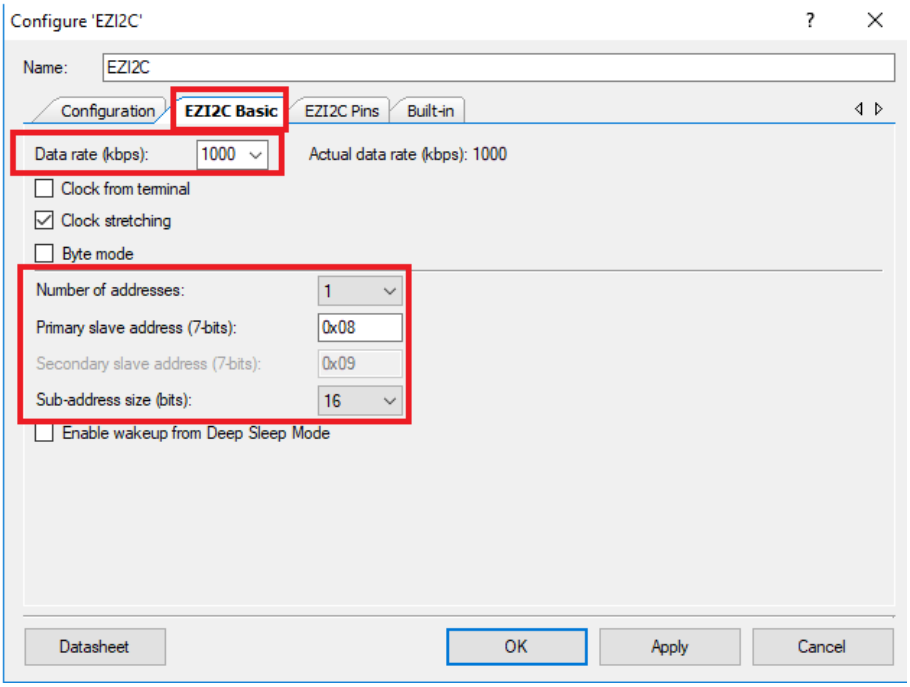


Table 8 lists the PSoC Creator Components used in this example.

Table 8. PSoC Creator Components

Component	Instance Name	Version	Hardware Resources
CapSense	CapSense	v7.0	One CSD
EZI2C Slave (SCB mode)	EZI2C	v4.0	One Serial Communication Block (SCB) and two GPIOs

Reusing This Example

This example can be easily ported to the kits mentioned in the Requirements section. Refer to the Hardware Setup section for correct pin assignments.

To port the code example to a device that is not listed, in PSoC Creator, select **Project** > **Device Selector** and change to the target device.

This code example can be used in all CapSense-enabled devices in the PSoC 4 family of devices that support the CSX sensing method, including:

- **Third-generation CapSense**: PSoC 4000, PSoC 4100, PSoC 4200, PSoC 4100M, PSoC 4200M, PSoC 4200L, PSoC 4100 BLE and PSoC 4200 BLE.
- **Fourth-generation CapSense**: PSoC 4000S, PSoC 4100S and PSoC 4100 S Plus.

Related Documents

Application Notes		
AN85951	PSoC 4 and PSoC 6 MCU CapSense Design Guide	Describes how to design capacitive touch sensing applications with PSoC 4 and PSoC 6 MCU
AN79953	Getting Started with PSoC 4	Introduces the PSoC 4 device and explains how to build a PSoC Creator project
PSoC Creator Component Datasheets		
CapSense		Supports various interfaces such as Button, Matrix Buttons, Slider, Touchpad, and Proximity Sensor
EZI2C Slave		Supports one or two address decoding with independent memory buffers
Pins		Supports connection of hardware resources to physical pins.
I2C		Supports I^2C Slave, Master, Multi-Master, and Multi-Master-Slave configurations
Device Documentation		
PSoC 4100S Plus Datasheet		PSoC 4100S Plus Architecture Technical Reference Manuals PSoC 4100S Plus Register Technical Reference Manuals
PSoC 4100S Datasheet		PSoC 4100S Architecture Technical Reference Manuals PSoC 4100S Register Technical Reference Manuals
PSoC 4000S Datasheet		PSoC 4000S Architecture Technical Reference Manuals PSoC 4000S Register Technical Reference Manuals
Development Kit (DVK) Documentation		
CY8CKIT-149 PSoC 4100S Plus Prototyping Kit		
CY8CKIT-041-41XX PSoC 4100S Pioneer Kit		
CY8CKIT-041-40XX PSoC 4000S Pioneer Kit		
CY8CKIT-145-40XX PSoC® 4000S CapSense Prototyping Kit		

Document History

Document Title: CE228931 – PSoC 4 CapSense CSX Tuning

Document Number: 002-28931

Revision	ECN	Date	Description of Change
**	6824937	03/05/2020	New code example

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709