Procedure

We will start this lab from a template project which already has all the components placed in the top design and corresponding pins selected in the design wide resources (**.cydwr**) file. The **main.c** file in this template project has the necessary firmware code to scan the button sensor, communicate with the tuner and control the LED state.

In this lab, we will configure the necessary components in the template project. Then, we will use the CapSense Tuner to tune the **Finger capacitance** parameter by choosing the maximum value for the **Finger capacitance** that ensures at-least 5:1 SNR and also ensures a finger touch on the button changes the sensor status to 1 i.e. ON.

Open the Template Project

1. Open a new instance of PSoC Creator 3.3 SP2. It is located in the **All Programs -> Cypress -> PSoC Creator 3.3** folder in the Windows start menu.
2. Open the template project by going to **File -> Open -> Project/Workspace** and browsing to the path where **Lab1.cywrk** file is stored in the **template Labs** folder.
3. As Figure 10 shows, in the **Workspace Explorer**, double click on the **TopDesign.cysch** file to open the schematic editor. The components placed in the template project and the functionality of each of these components is listed in Table 2.

Figure 10. Opening Schematic Editor in the Project

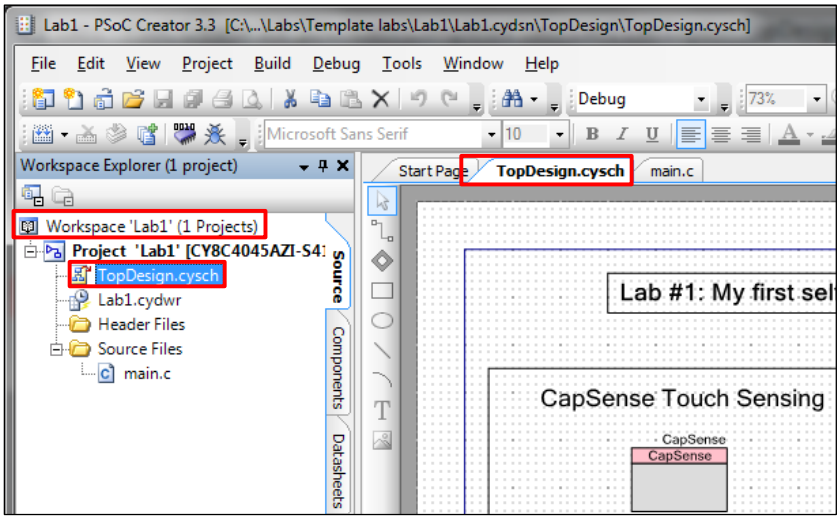


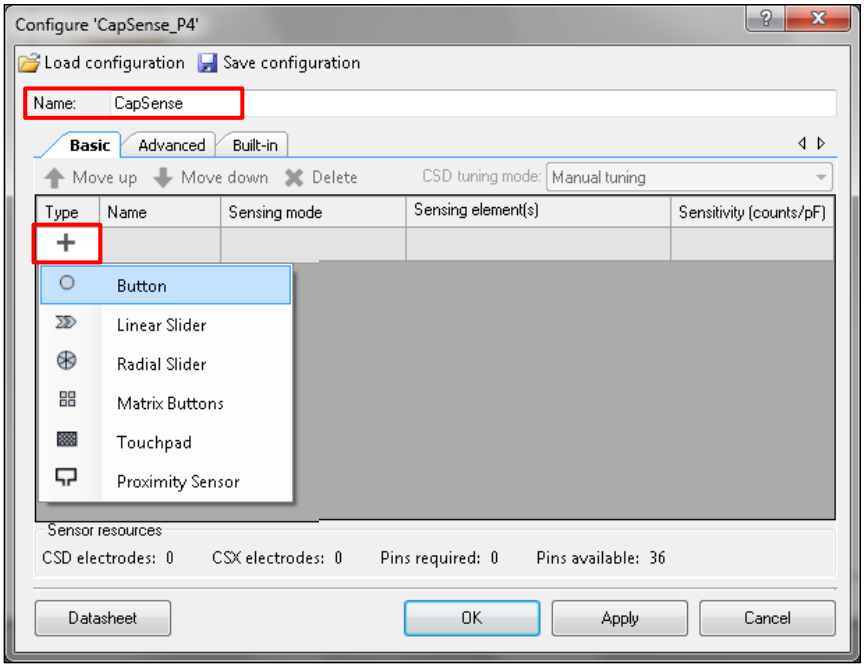
Table 2. Lab1 Template Project Components

Component	Instance Name	Purpose
CapSense	CapSense	To scan the button sensor and check the sensor status
EZI2C Slave (SCB mode)	EZI2C	To communicate with the CapSense Tuner to observe sensor data graphically, for tuning purpose
Digital Output Pin	Pin_LED	To control the state of LED connected to this pin

Configure CapSense and EZI2C Components

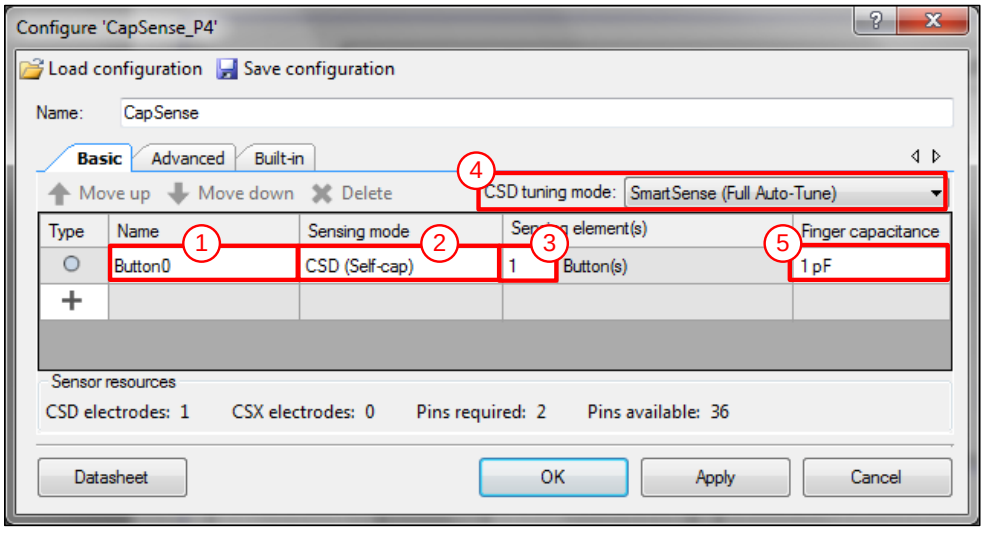
4. Double-click the **CapSense** Component to open its Component Configuration Tool. Note, you can also right-click the Component and select **Configure…**.
   As Figure 11 shows, in the **Configure 'CapSense_P4'** window that appears, add a Button sensor by clicking on the **+** sign in the **Type** column and selecting **Button**.

Figure 11. Adding a Button Sensor in CapSense Component Basics Tab



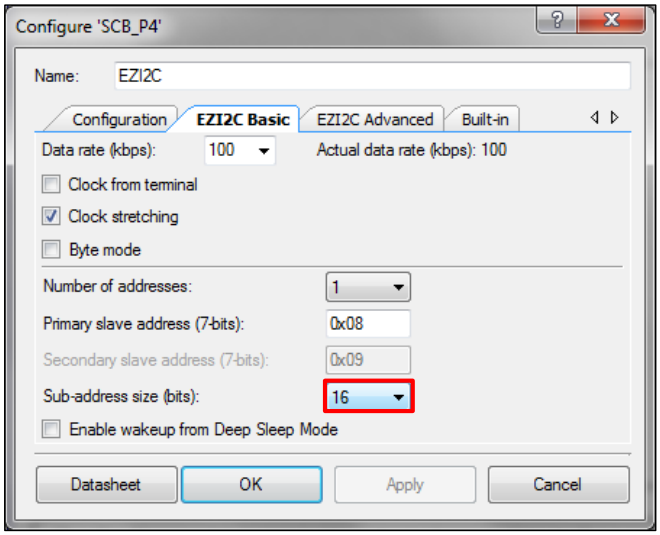
5. Select the sensor parameters as Figure 12 shows. Leave parameters in all other tabs at their default values and click **OK** to apply the changes and close the Component Configuration Tool.
   Note: The rationale behind specifying the parameter values shown in Figure 12 is listed in Table 3 on page 25 in the Appendix section.

Figure 12. CapSense Basic Tab Parameters



6. Double-click the **EZI2C** Component to open its Component Configuration Tool. As Figure 13 shows, change the **Sub-address size (bits)** to **16** and click **OK** to apply the changes and close window.

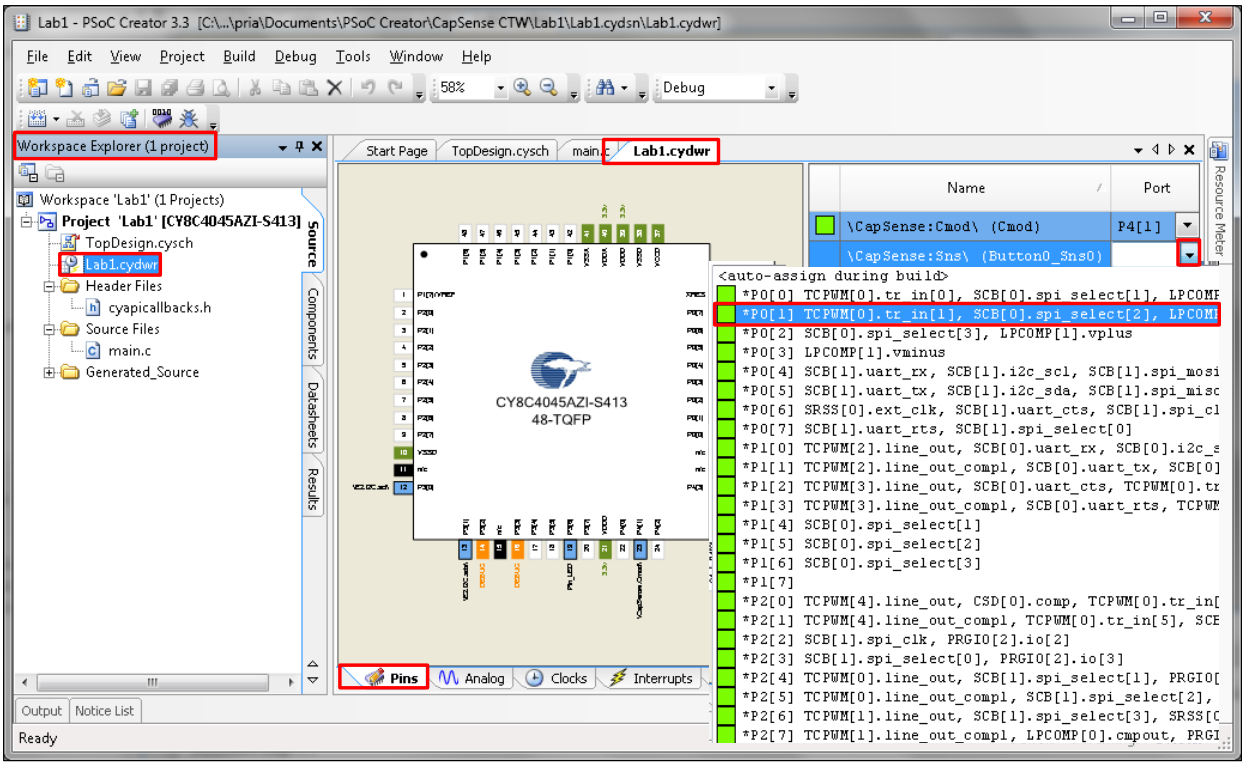
Figure 13. EZI2C Component configuration



Select Sensor Pin

7. As Figure 14 shows, in the **Workspace Explorer**, double-click on the **Lab1.cydwr** file to open the **Pins** tab in **Lab1.cydwr** window and select **P0[1]** for **Button0**.

Figure 14. Pin Assignment in Design Wide Resources



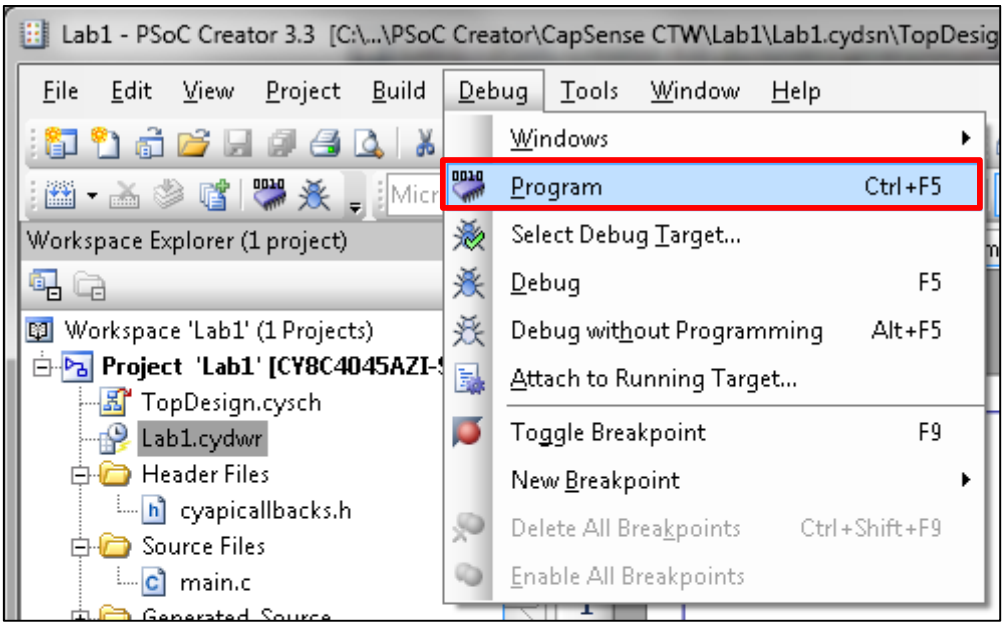
Implement Firmware

8. Go to the **Workspace Explorer**, and double click the **main.c** file to view the code. The firmware code has already been added in the template project. Review this code to understand the typical code flow for CapSense designs.

Build and Program

9. Click the menu item **Debug -> Program** as shown in Figure 15. Clicking **program** builds the project (if there are any changes from previous build or if the project has not been built before) and then programs your kit.

Figure 15. Programming a Project



Note. You may also see a pop-up window asking you to confirm which device to program. Simply choose the **KitProg2** (the PSoC 5-based programmer and debugger on the baseboard) and click **Port Acquire**, see

Figure 16 on page 14. Then select the device and click **Connect** and **OK**, as Figure 17 on page 14 shows.

Figure 16. PSoC Creator “Select Debug Target” Window – Port Acquire

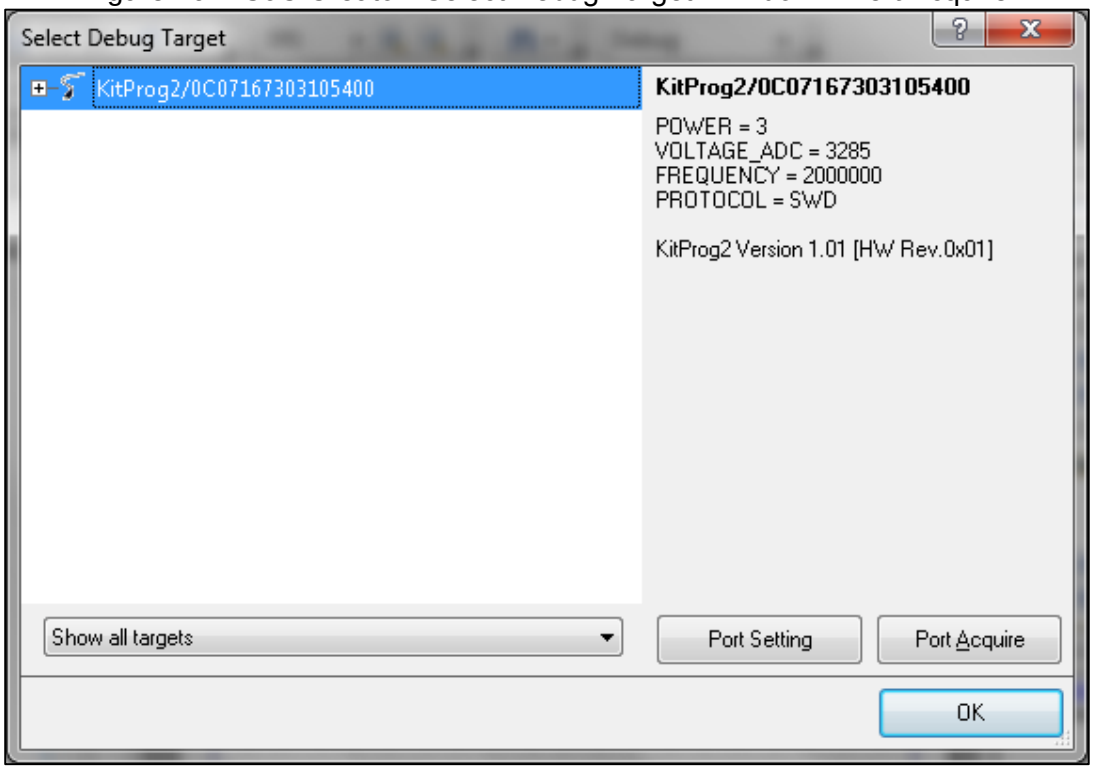
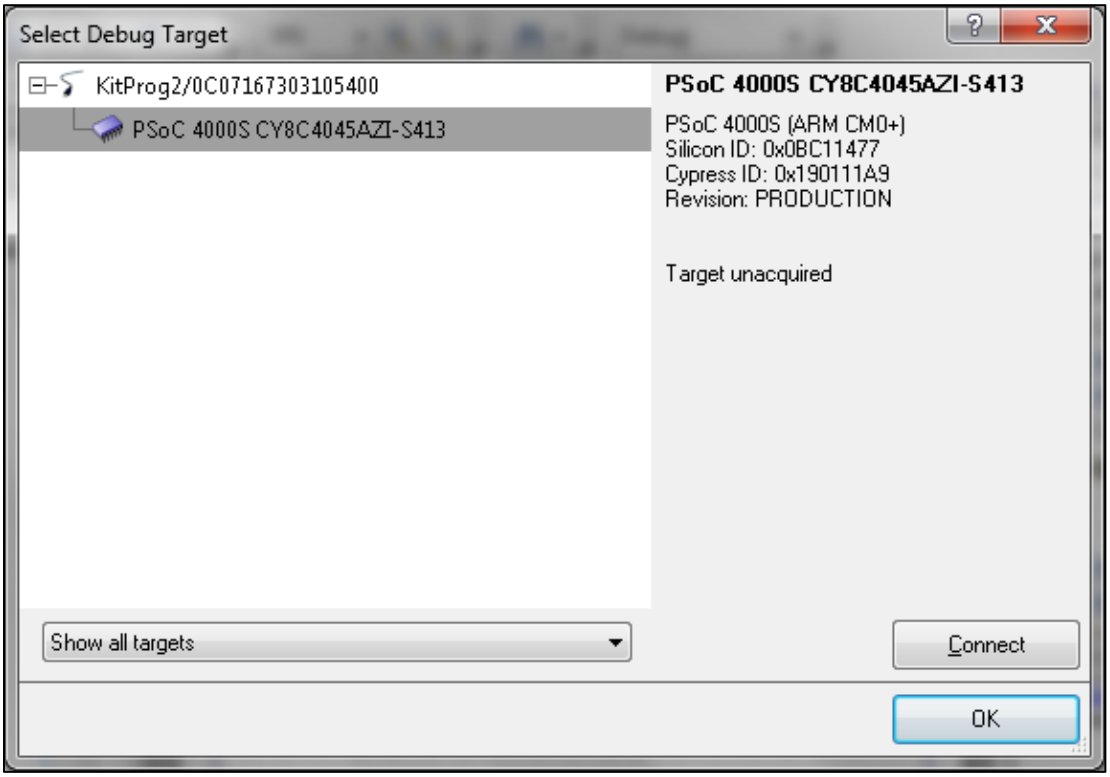


Figure 17. PSoC Creator “Select Debug Target” Window – Connect



Use CapSense Tuner to view sensor data

10. In the **TopDesign.cysch** file, right-click the **CapSense** component and select **Launch Tuner** as Figure 18 shows. This will open the **CapSense Tuner** as Figure 19 shows.

Figure 18. Launching CapSense Tuner

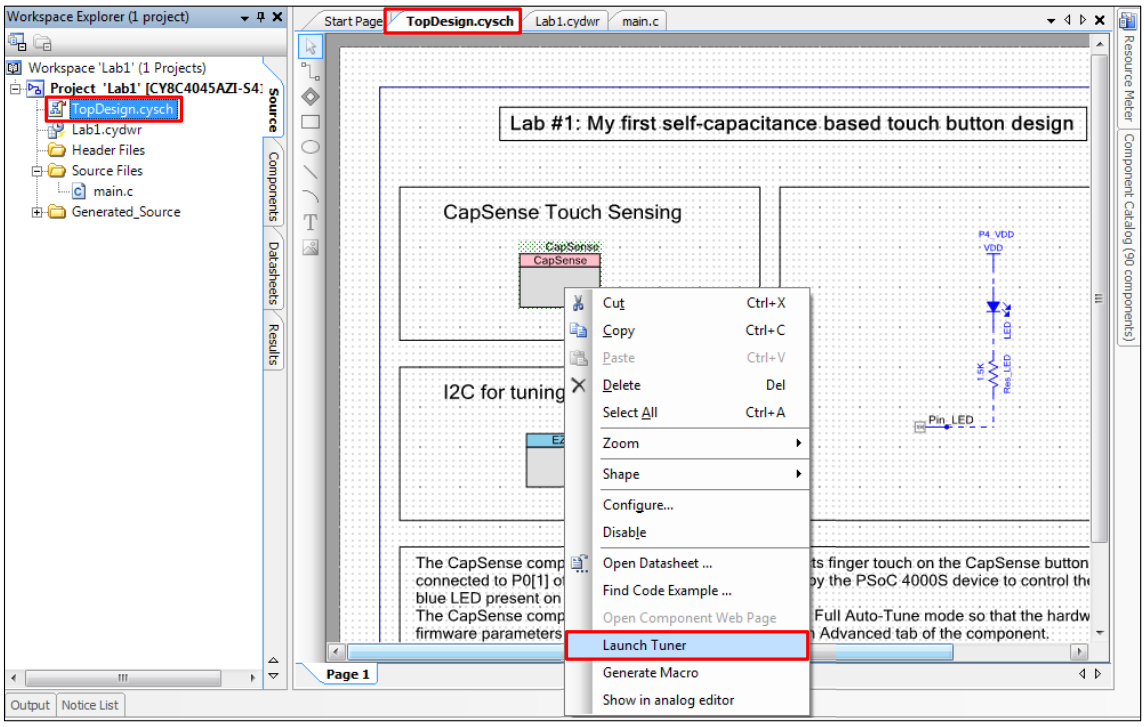
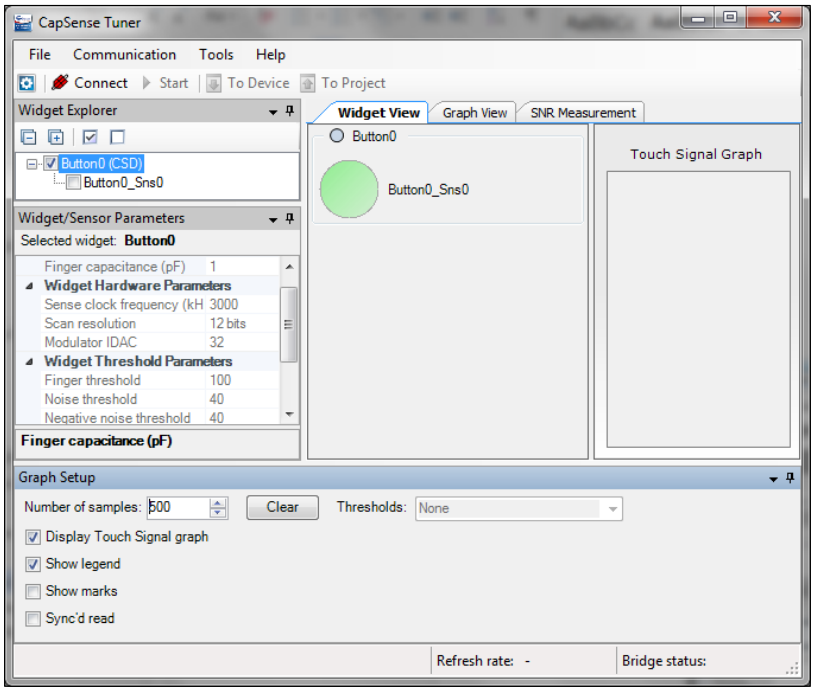
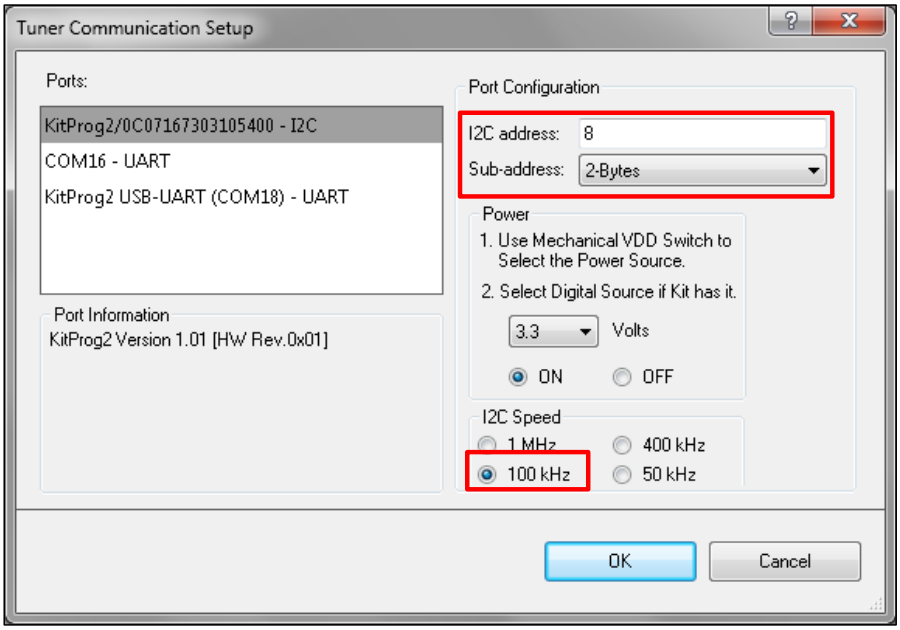


Figure 19. CapSense Tuner Window



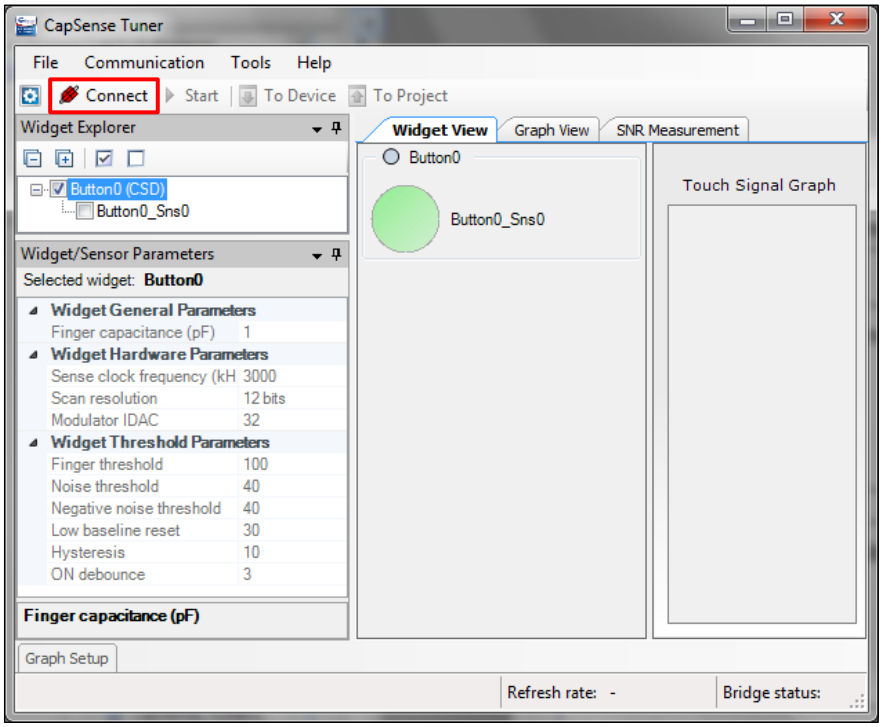
11. In the CapSense Tuner window, go to **Tools -> Tuner Communication Setup…** to open the **Tuner Communication Setup window**, and click on the I2C port to set the parameters as Figure 20 shows. Note that the **I2C address**, **Sub-address**, and **I2C speed** selected in this window match the parameters **Primary slave address (7-bits)**, **Sub-address size (bits)**, and **Data rate (kbps)** in the EZI2C Component configuration window. Click **OK**, this closes the Tuner Communication Setup.

Figure 20. Tuner Communication Setup



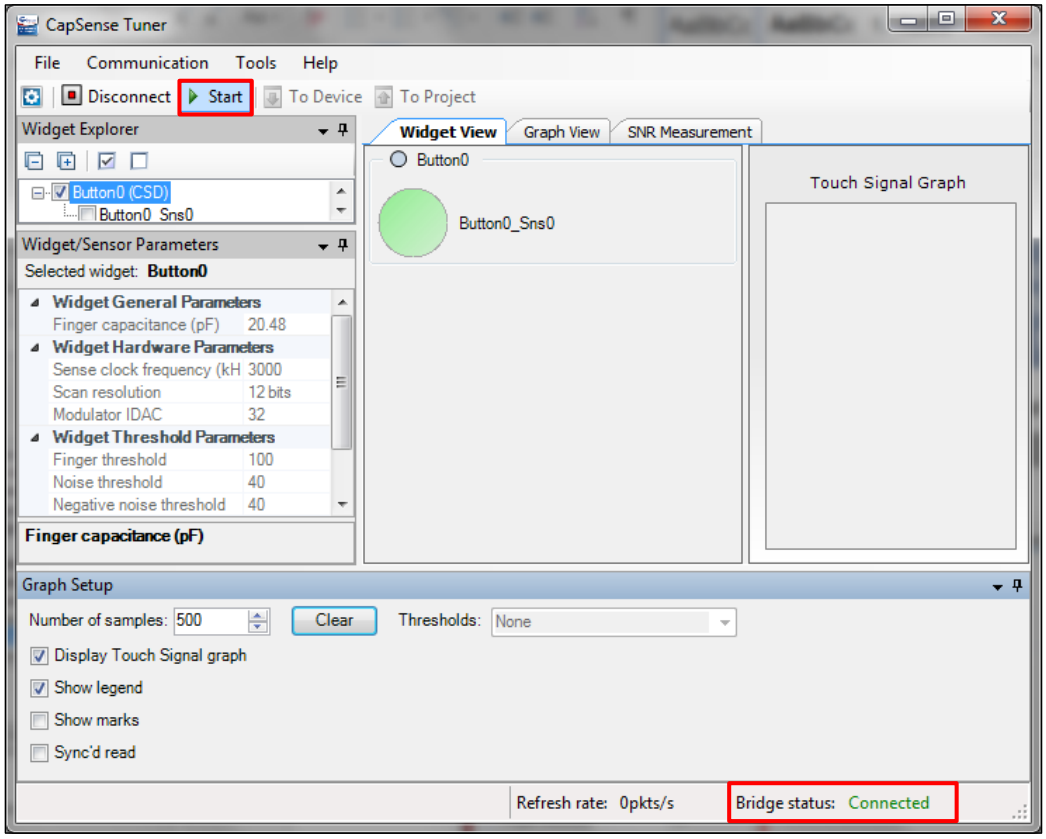
12. As Figure 21 shows, click on **Connect** in the **CapSense Tuner** window.

Figure 21. CapSense Tuner Window



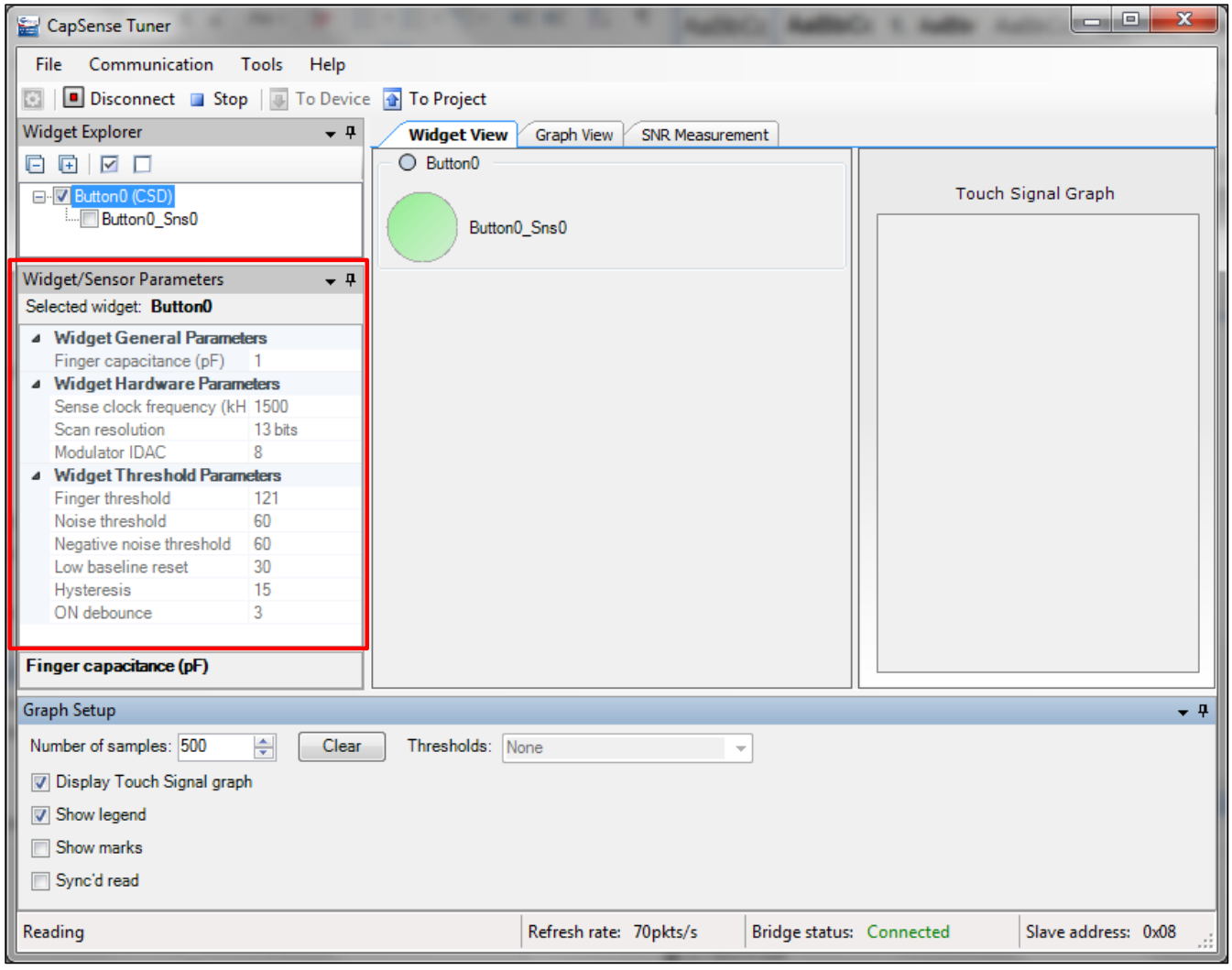
13. The **Bridge Status** now shows as **Connected** as Figure 22 shows. Now click on **Start** as Figure 22 shows.

Figure 22. Starting tuning with CapSense Tuner



14. Notice that the **Widget/Sensor Parameters** get updated, see Figure 23. The SmartSense tuning method automatically calculates the required **Widget Hardware parameters** and also continuously calculates and updates the **Widget Threshold parameters** based on the **Finger capacitance** to be sensed and the noise on raw counts. Even if you plan on using manual tuning this is a quick method of determining initial parameters.

Figure 23. Widget Parameters Update in CapSense Tuner

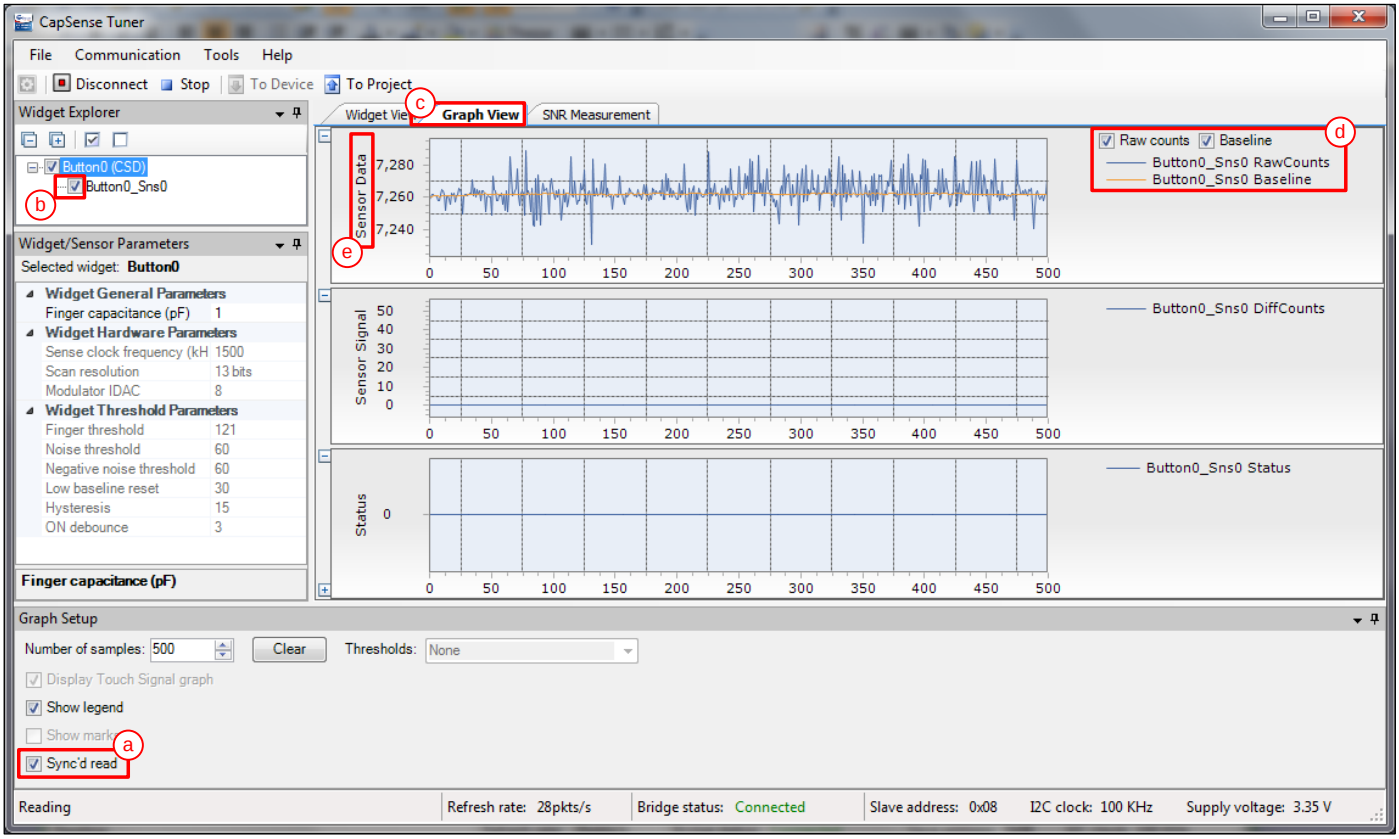


15. As Figure 24 shows, select the **Sync'd read** option (a) and the **Button0_Sns0** (b) and then go to the **Graph View** (c)**.** You will now be able to see the **Raw counts** and **Baseline** (d) for **Button0_Sns0** in the **Sensor Data** (e) window in **graph view**.

    Note that enabling the **Sync'd read** enables synchronized data read from/to device, this ensures that there is no noise seen in Tuner graphs due to asynchronous I2C data reads.

    Also, enabling **Sync'd read** option is a must to be able to change the tunable CapSense parameters from the tuner itself. Enable the **Sync'd read** so that the **Finger Capacitance** parameter is not greyed out in the tuner.

Figure 24. Sensor Data in Graph View of CapSense Tuner

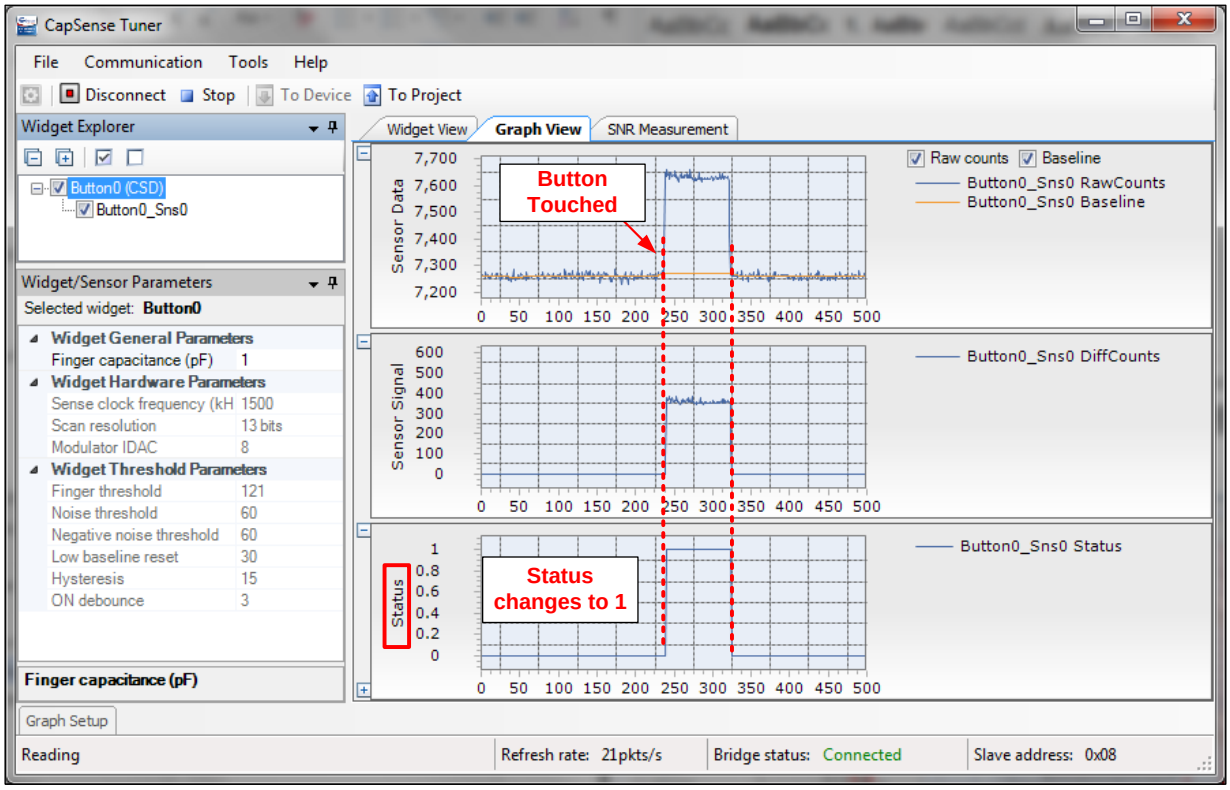


Coarsely tune the Finger Capacitance Parameter

16. As Figure 25 shows, touch the button on the kit. Check if the **Status** in **CapSense Tuner** changes from '0' to '1' (see Figure 26). If status does not change to '1', go to step 17 on page 21, else if the status changes to '1' on touch, skip to step 20 on page 22.

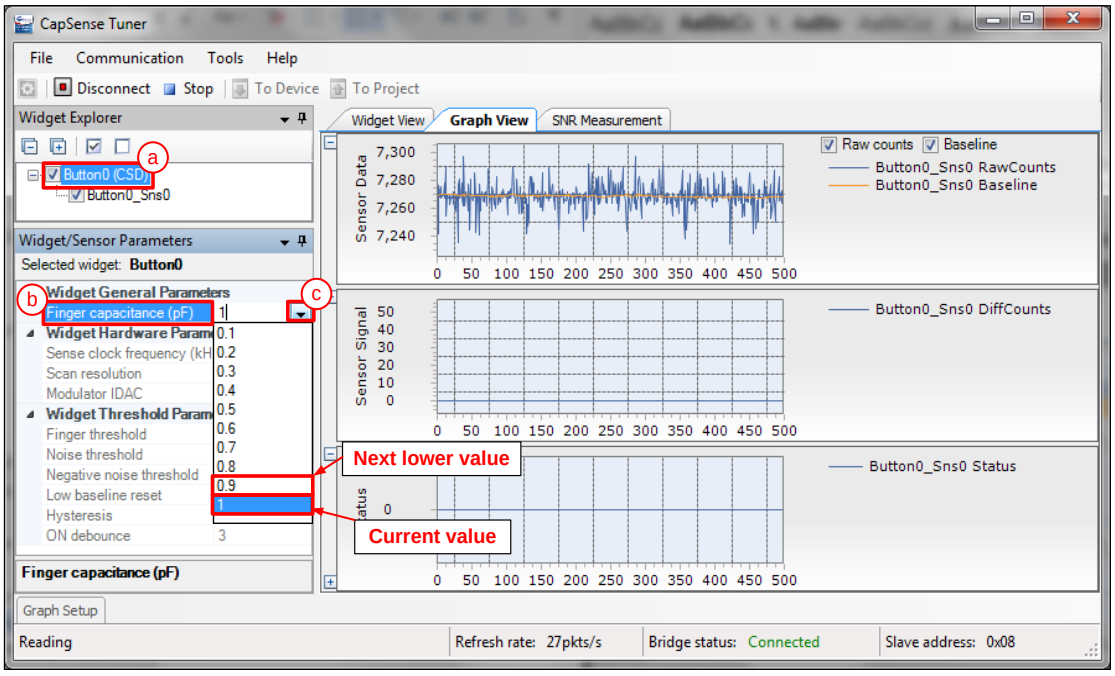
Figure 25. Touching the Button on the Kit

Figure 26: Sensor Data When Button is Touched



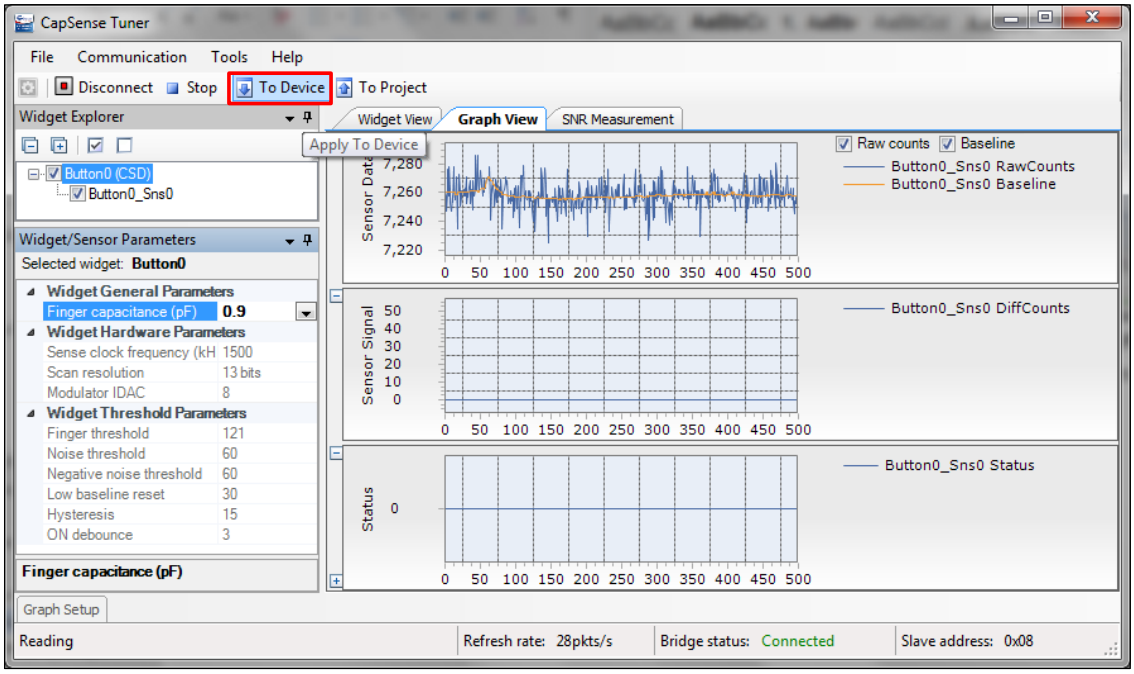
17. If the **status** does not change to '1' on touch, click on widget, **Button0(CSD)** (a) and then **Finger capacitance** (b), and chose the next lower value of **Finger capacitance** (c) from the available options in the drop-down list as Figure 27 shows.

Figure 27: Choose Next Lower Value of Finger Capacitance from Drop-Down List



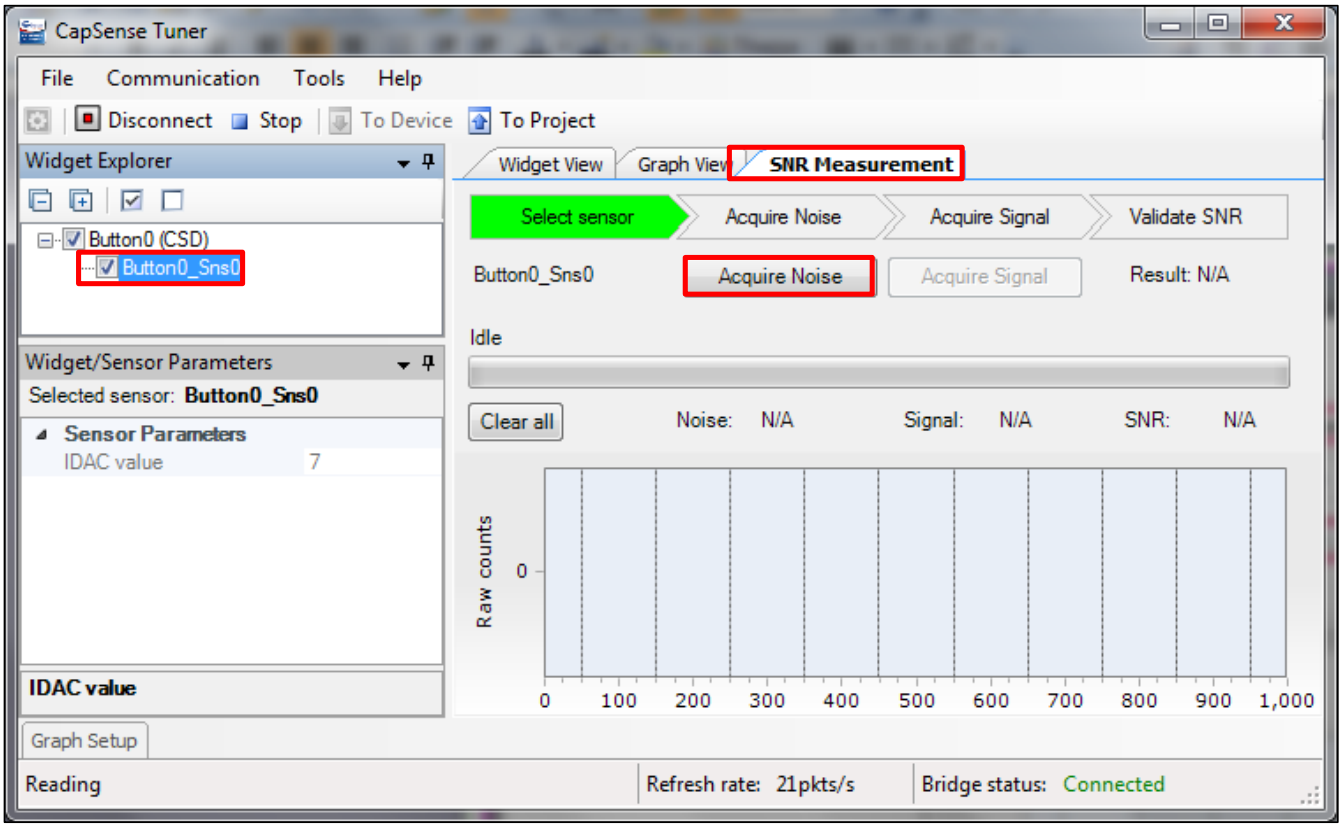
18. Click on **To Device** as Figure 28 shows. This applies the updated **Finger capacitance** to the device in the kit.

Figure 28: Apply Updated Parameters to Device



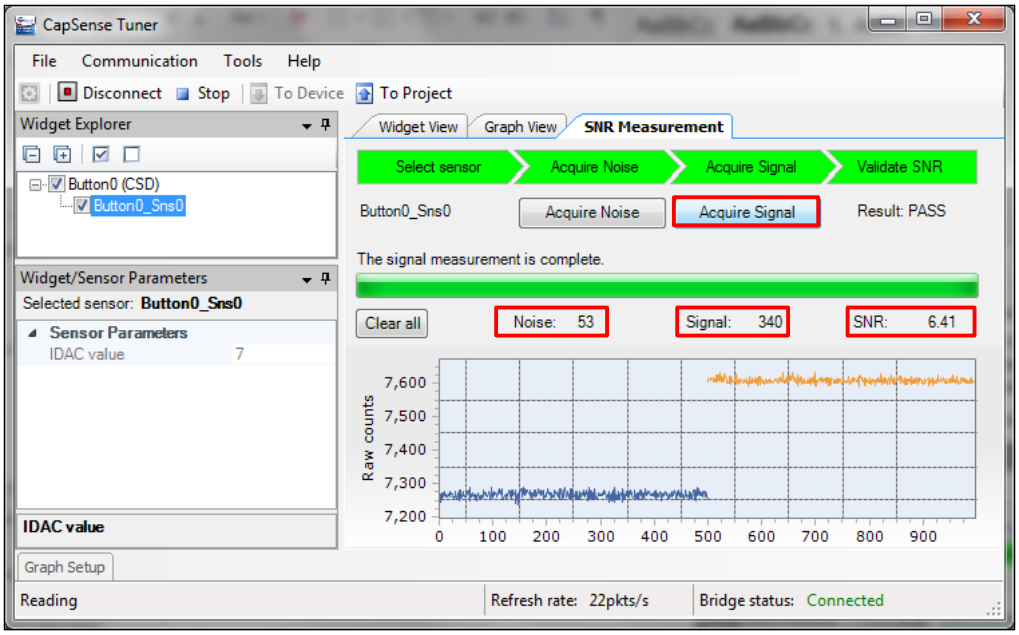
19. Touch the button again and check if the **Status** in **CapSense Tuner** changes from '0' to '1'. If the sensor status does not change to '1' on button touch, keep repeating steps 17 to 19 (on page 21 to 22) i.e. keep decreasing the **Finger capacitance** to the next lower value and applying to the device until the sensor **status** changes to '1' on touch.
20. If the Status changes to '1' on touch, switch to the **SNR Measurement** tab, select **Button0_Sns0** sensor and then click on **Acquire Noise** as Figure 29 shows. Do not touch the sensor during noise acquire period.

Figure 29. Noise Measurement Using CapSense Tuner



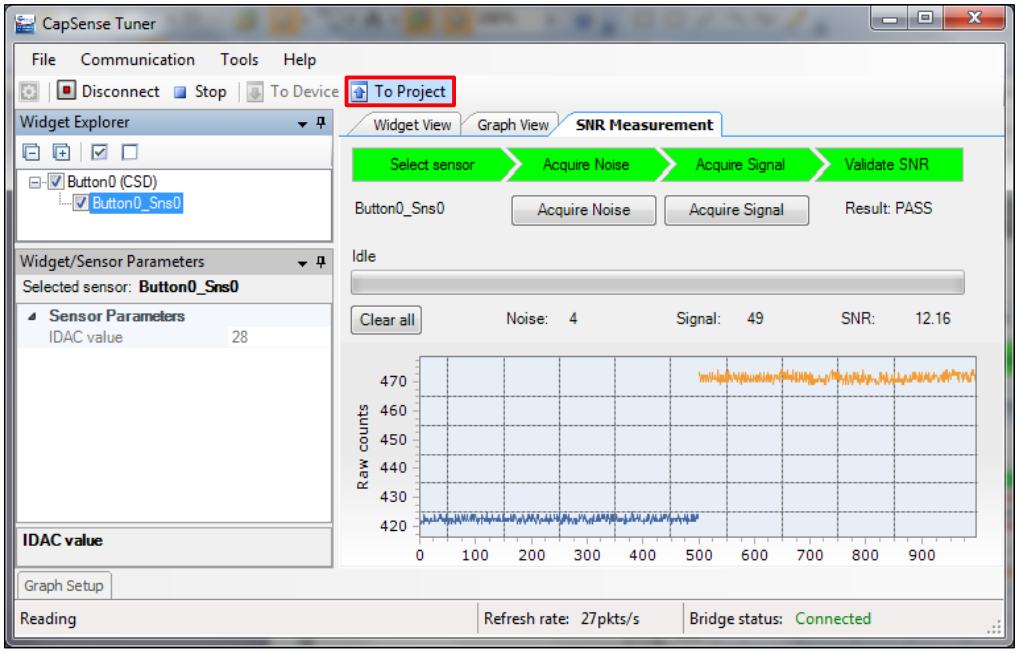
21. Once the noise is acquired, touch the button on the kit and then click on **Acquire Signal**. Ensure that the button is touched for as long as the signal acquisition is in progress. You will now be able to see the calculated **Signal** and **SNR** for this button as Figure 30 shows.

Figure 30. SNR Calculated by CapSense Tuner



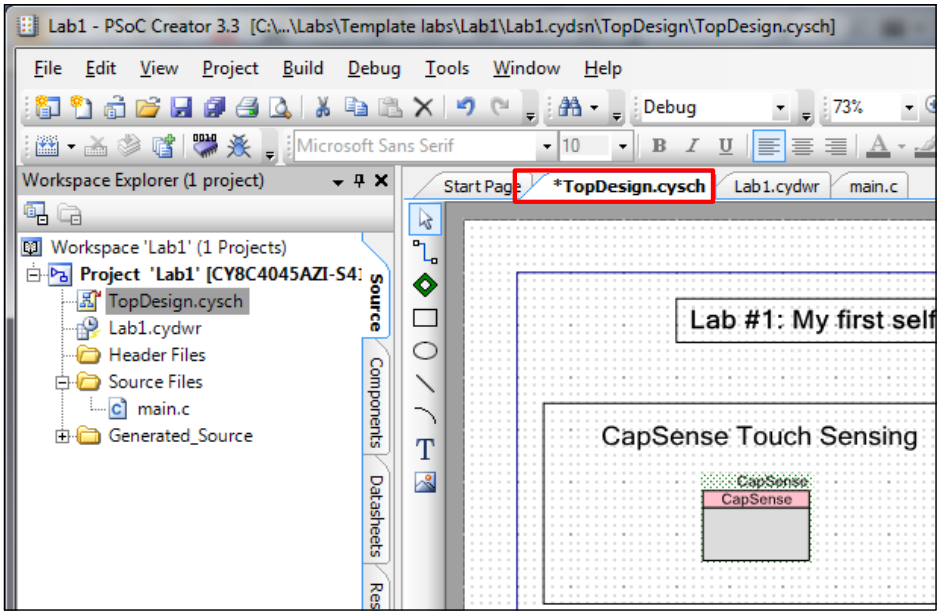
22. If SNR > 5, click on **To Project** as Figure 31 shows. However, if SNR is less than 5, keep repeating steps 17 to 21 (on page 21 to 23) i.e. keep decreasing the **Finger capacitance** to the next lower value and applying to the device until both the following conditions become true:
    a. Sensor **status** changes from '0' to '1' on touch, and
    b. SNR > 5.

Figure 31. Apply Updated Settings to Project



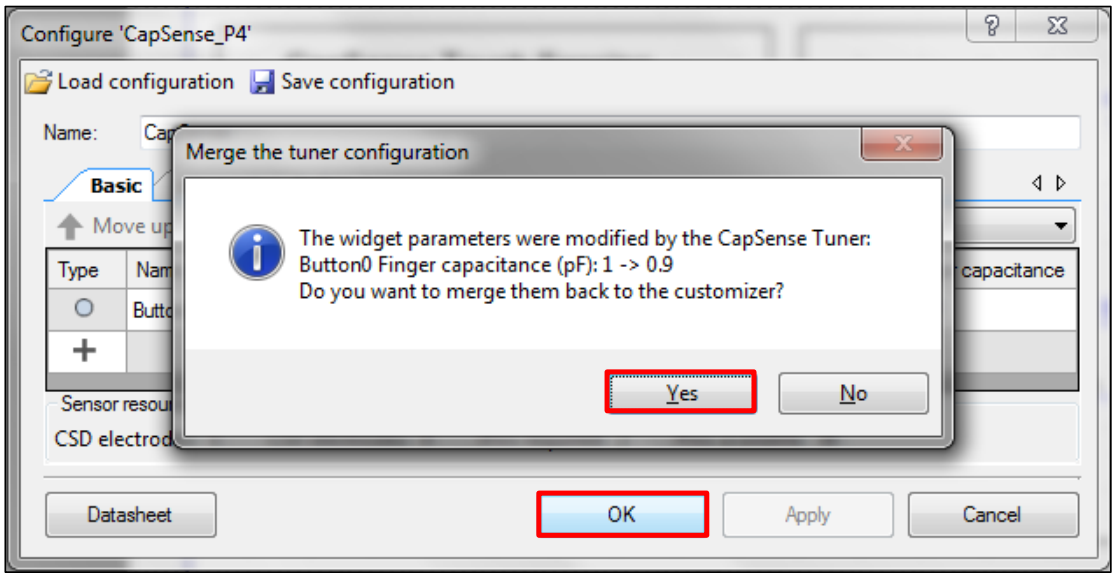
23. Close the tuner by clicking on the **x** sign on the top right corner of the window, or click **File -> Exit.**
24. Notice the ***** sign on the **TopDesign** file, indicating that the file has been updated (see Figure 32).

Figure 32. TopDesign Gets Updated When **To Project** is Clicked on Tuner



25. Double-click the CapSense component to open the Configuration window. If a change has been made to the **finger capacitance** parameter by the tuner, a **merge tuner Configuration** window will open as Figure 33 shows. Click **Yes** to accept the changes and click **OK** to close the component configuration window.
    Note that if **Finger capacitance** parameter has not been updated in tuner, **merge tuner Configuration window** will not open, just click **OK** and close the component configuration window in this case.

Figure 33. Accepting Tuner Modified Parameters in CapSense Component



Build and program the project again and test

26. Build and Program the Lab1 template project onto the kit by clicking on the menu item **Debug -> Program.**
27. Touch the button and observe that the LED glows when the button is touched.

Appendix

Table 3. Rationale Behind Recommended Settings for CapSense Basic Tab Parameters

Parameter	Value	Rationale
Name	Button0	We can give this sensor any relevant name. For this lab, we have retained the default name.
Sensing mode	CSD (Self-cap)	CSD (Self-Cap) is chosen for self-capacitance based touch sensing. Mutual-capacitance based sensing is explored in a later lab.
Sensing element(s)	1	Here, we specify the number of sensors in this widget. Since we want to sense touch on only one sensor, we have specified this value as 1. Note that his option is useful in applications where there are multiple sensors that are identical, that is, have the same C_P and sensor area. The SmartSense method will tune all the sensors under a single widget for same finger Capacitance. Having multiple sensors under a single widget saves processing time and reduces memory footprint.
CSD tuning mode	SmartSense (Full Auto-Tune)	We use **SmartSense (Full Auto-Tune)** to allow the component to automatically tune all of the CapSense hardware and software parameters, based on the specified **Finger Capacitance** value.
Finger capacitance	Highest value from drop-down	Use the highest value for the initial setting. We will tune this number to the correct value later in section Coarsely tune the Finger Capacitance Parameter, based on the button's response to finger touch.