Monitoring and Tuning CapSense CSD Component – PSoC® 3 and PSoC 5

EP58265

Project Name: CapSense_CSD_Tuning
Programming Language: C
Associated Part Families: CY8C38xx/CY8C55xx
Software Version: PSoC® Creator™
Related Hardware: CY8CKIT-001, MiniProg3 Debug and Evaluation Device
Prerequisites: Example Project: CapSense Button and Slider Example - EP56173
Author: Pavankumar Vibhute

Project Objective

This project demonstrates monitoring CapSense® output and tuning of CapSense_CSD component through I2C communication.

Overview

The CapSense component has several parameter settings that should be tuned to make it work for particular design. The CapSense_CSD component has 'Auto Tuning' (Smart Sense) feature, where the firmware tunes all the parameters automatically to required sensitivity. With Auto Tuning, you need not worry about any parameter setting. The component also gives option of Manual Tuning, if you want to tune the CapSense parameters manually.

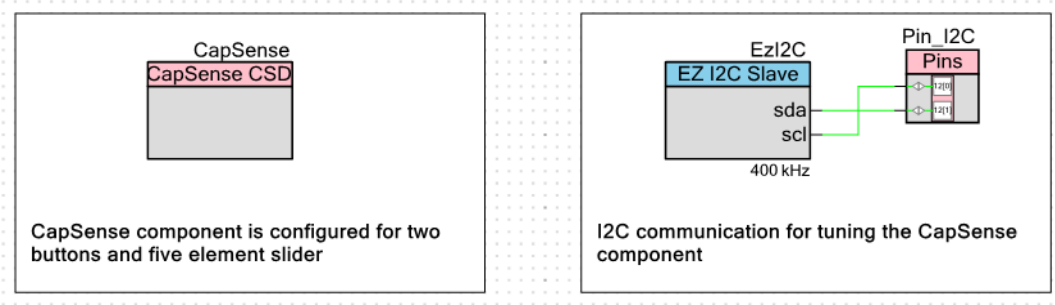
This example project shows how to tune different CapSense_CSD parameters using Tune Helper GUI manually. The Tune Helper GUI is provided with CapSense_CSD component and it uses the I2C for communication between device and GUI. The CapSense parameters are set in the GUI and result of the settings are observed on graphs in the GUI. In case of Auto Tuning, the parameters will be automatically set, therefore GUI provides only monitoring of CapSense output.

Component List

Instance Name	Component Name	Version	Component Category	Comments
CapSense	CapSense_CSD	2.0	Analog → CapSense	The CapSense is configured to have two buttons and a five element slider.
EzI2C	EZ I2C Slave	1.5	Communication	It sends the CapSense variables and receives the parameter settings
Pin_I2C	Digital Output Pin	1.5	Ports and Pins	These pins are used for EzI2C SDA and SCL. Configured as Bidirectional pins.

Top Design

The following figure illustrates the components and their routing.



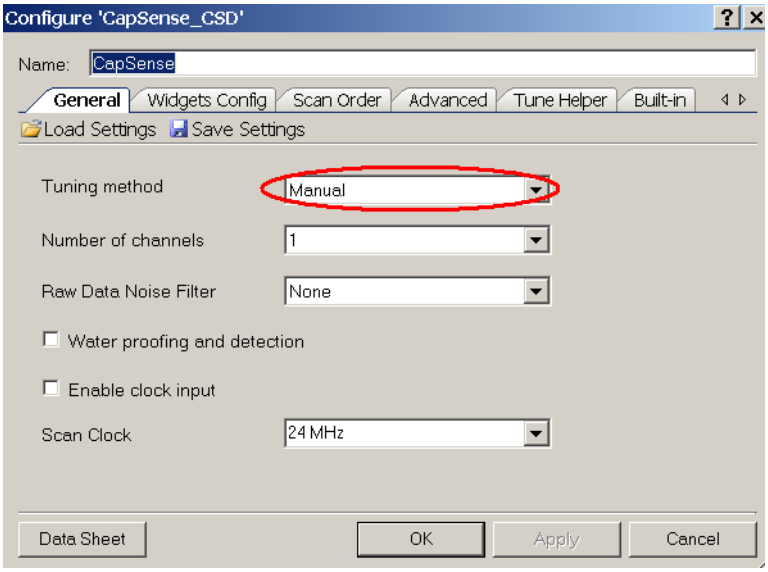
The following figure shows pin placement (as in .cydwr file). The Cmod capacitor should be configured as P2[7] for a PSoC® 3 processor module and P15[5] for a PSoC 5 processor module.

Alias	Name	Pin
LinearSlider0_e4__LS	\CapSense:PortCH0\[6]	P0[4]
LinearSlider0_e3__LS	\CapSense:PortCH0\[5]	P0[3]
LinearSlider0_e2__LS	\CapSense:PortCH0\[4]	P0[2]
LinearSlider0_e1__LS	\CapSense:PortCH0\[3]	P0[1]
LinearSlider0_e0__LS	\CapSense:PortCH0\[2]	P0[0]
Button1__BTN	\CapSense:PortCH0\[1]	P0[6]
Button0__BTN	\CapSense:PortCH0\[0]	P0[5]
Cmod_CH0	\CapSense:CmodCH0\	P15[5]
sda	Pin_I2C[1:0]	P12[1:0]

Component Configuration

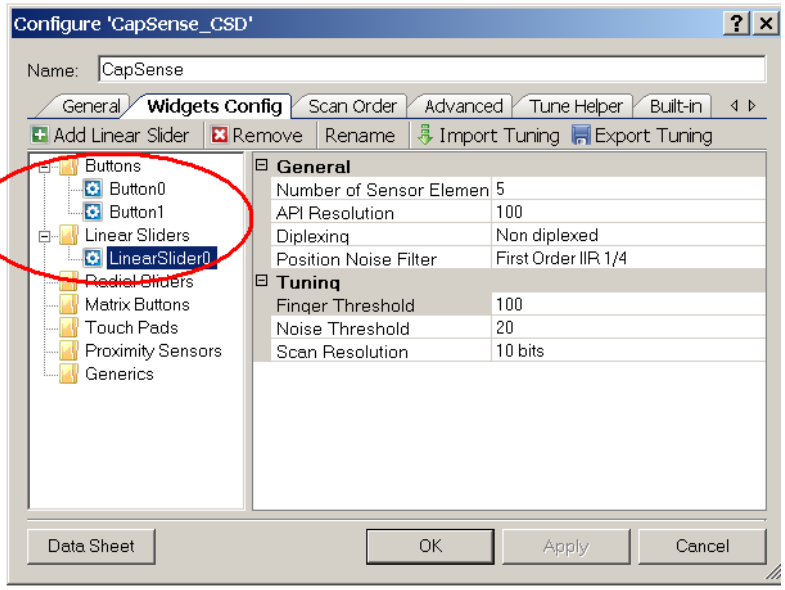
CapSense

General Tab



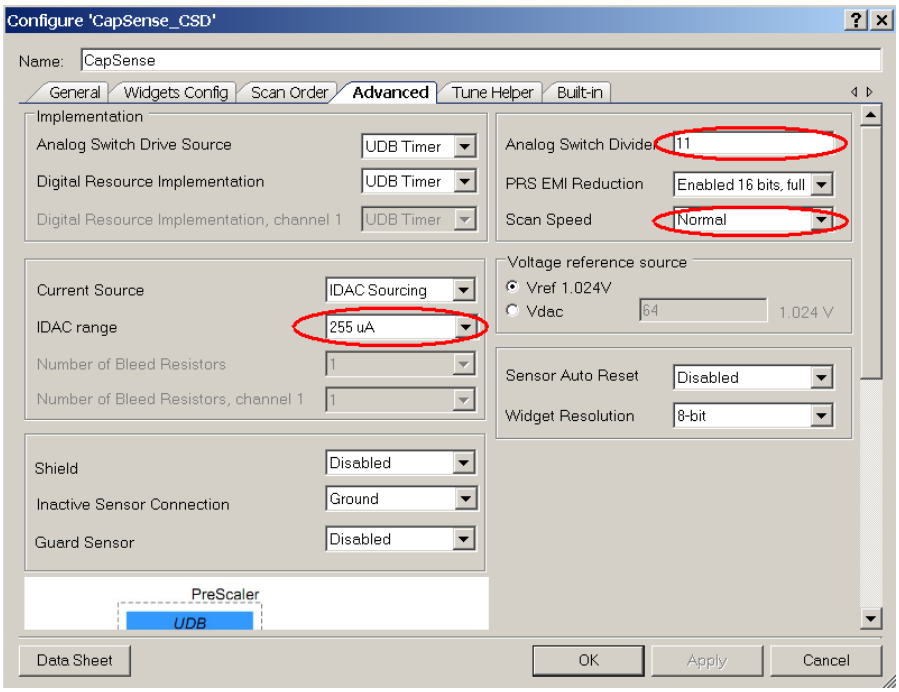
Tuning is set to Manual Tuning. Using the Tune Helper GUI, all the parameters can be manually tuned.

Widget Config Tab



Two buttons and a five-element slider are added in this tab. These sensors are assigned to the CapSense buttons and slider pins on the development kit.

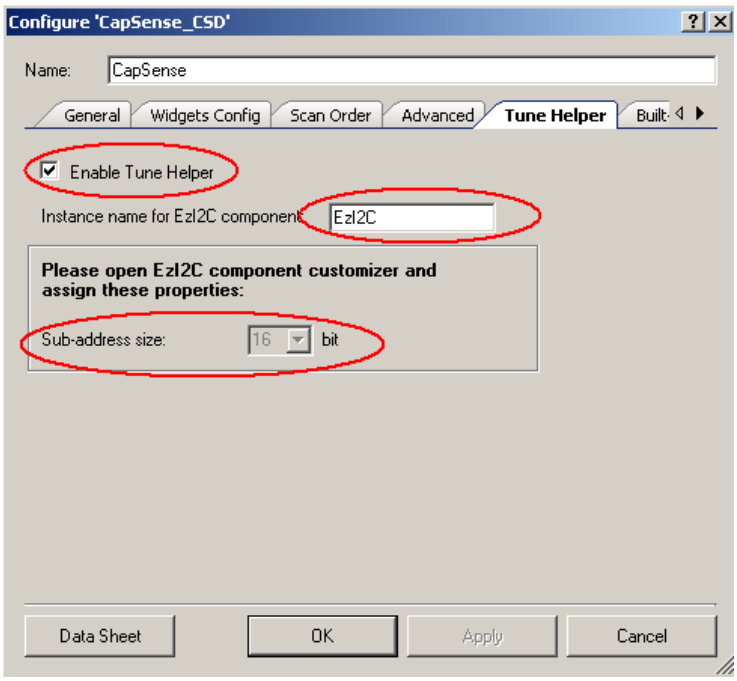
Advanced Tab



Analog Switch Divider, IDAC Range and Scan Speed parameters cannot be changed in Tuner GUI. These parameters should be set only here. These are common parameters to all the sensors.

During the tuning process if these parameters need to be altered then new values should be entered in this tab and project should be rebuilt and reprogrammed to the device.

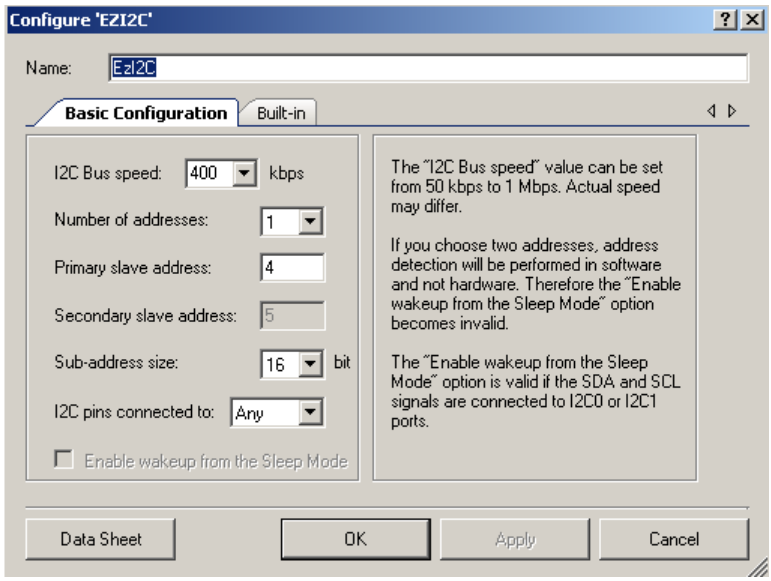
Tune Helper Tab



Enabling the Tune Helper provides the Tune Helper GUI. In GUI, all the CapSense parameter settings which are set by 'Auto Tuning' can be monitored using I2C communication. Also, the variables for different sensors such as Raw count, Baseline can be monitored.

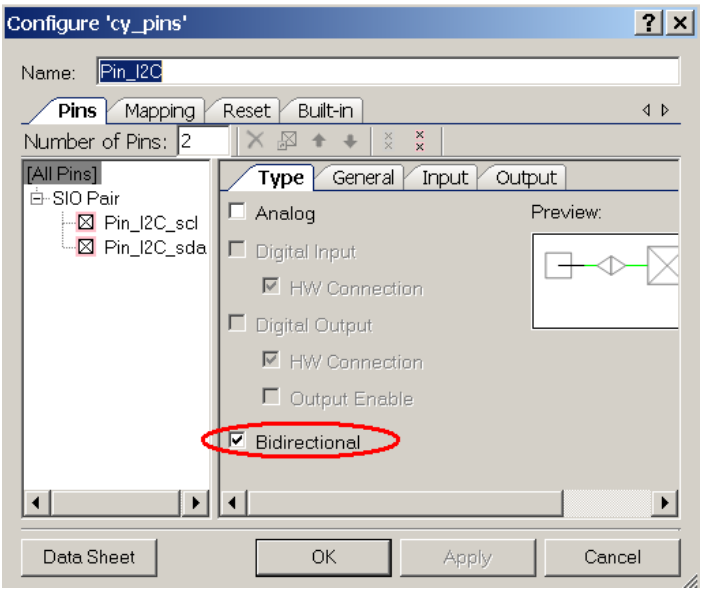
If tuner is enabled, then EzI2C Slave component needs to be placed in the Top Design and should be renamed using the same name as it is in the Tune Helper tab. Also, the sub address size in the EzI2C component should be set as 16 bit as it is in the Tune Helper tab.

EzI2C



The name of the component EzI2C should be same as it is displayed in the Tune Helper tab in the CapSense_CSD component. The Sub-Address size should also match the value in the CapSense_CSD component. Remaining settings are done as follows.

Pin_I2C



For I2C communication, the pins are configured as 'Bidirectional'.

Design Wide Resources

This project uses the default configuration. Refer to *Capsense_Example.cydwr* for the default configuration.

Operation

The tuning method is set to 'Manual Tuning' in the component. The Tune Helper GUI is used for tuning process; see Tuning Process at the end of this document. Tuning process involves monitoring of CapSense results in the GUI and setting different parameters in the GUI to obtain optimum CapSense results. After tuning the parameters for all the sensors, the settings are applied to CapSense_CSD component.

Afer the tuned parameters are applied to the component, the tuning method can be set to 'None'. When the tuning method is set to 'None', the Creator™ defines all the parameters as constants and hence reduces the RAM size required for the CapSense application. After changing the tuning method to 'None', the code should be modified with the normal APIs to scan all the sensors, project should be built again and reprogrammed to the device.

The following code scans the CapSense sensors continuously and communicates between the GUI and device. The APIs used here will work only when the Tune Helper GUI is enabled.

```
      CYGlobalIntEnable; /* Uncomment this line to enable global interrupts. */

    /* Intializes the CapSense, EzI2C components. Also initializes the baseline and
    * starts the scanning loop */
    CapSense_TunerStart();

    for(;;)
    {
        /* This API imports settings from the GUI, applies to CapSense sensors
       * scans all the sensors and sends the data to GUI through I2C communication.
*/
        CapSense_TunerComm();
}
```

Hardware Connections

The project is tested with PSoC Development Kit CY8CKIT-001. The kit is used with jumpers in the default state. Refer to the PSoC Development Kit Board Guide, provided with the kit.

MiniProg3 is used for I2C-USB communication. The Pin_I2C_scl and Pin_I2C_sda are connected to MiniProg3 I2C_SCL and I2C_SDA pins. The MiniProg3 pins are shown in the following figure.

GND of DVK is connected to MiniProg3 GND.

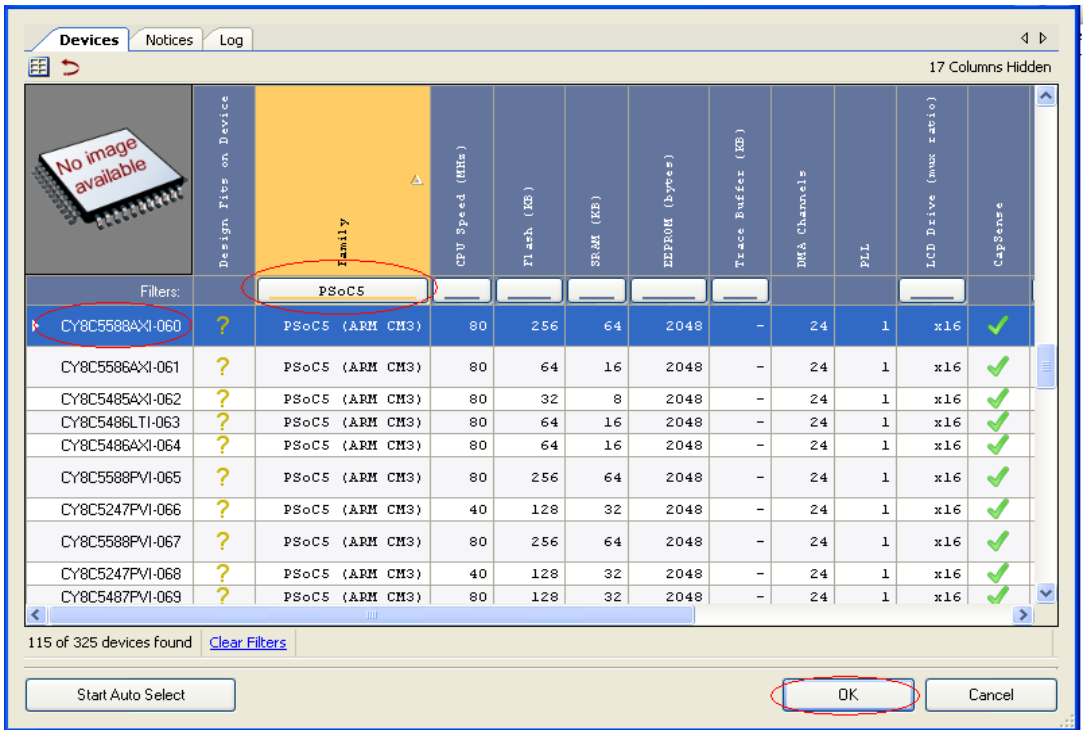
Mini-Prog3 I2C connector

Output

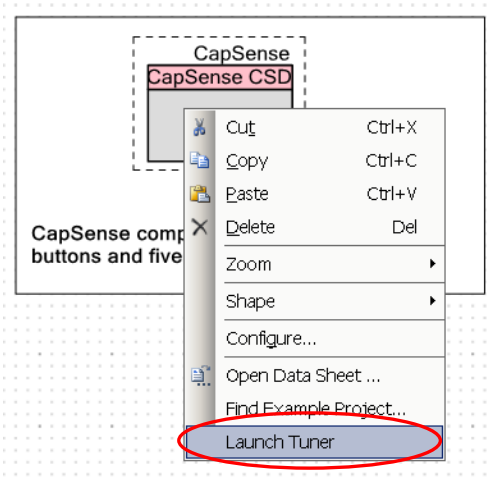
1. Build the project and program the chip.

   Note The default device selection is PSoC 3 (CY8C3866AXI-040), for using this project with PSoC 5 family follow the steps shown as follows.

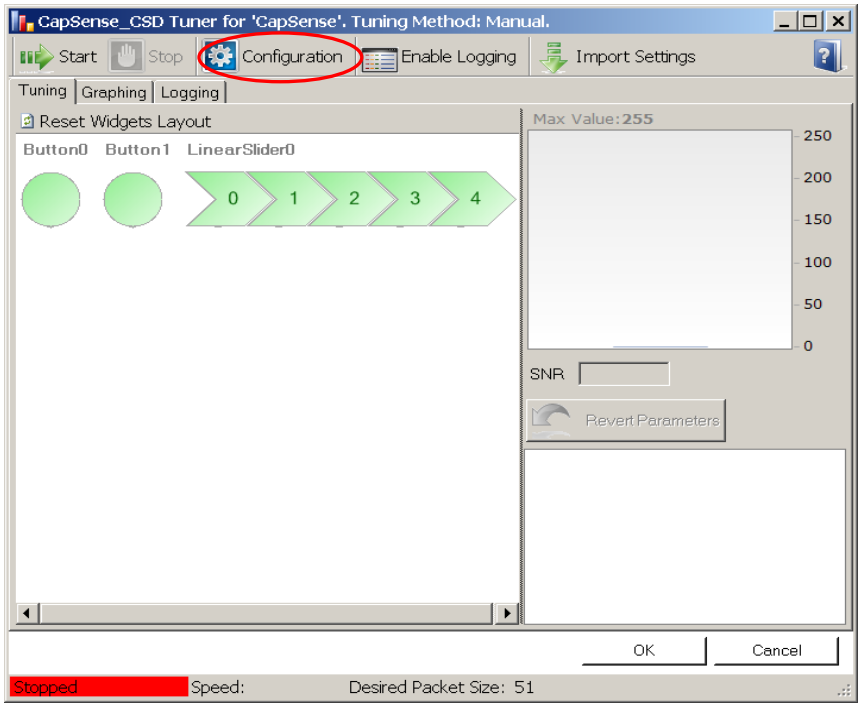
   Go to **Project → Device Selector →** Select **PSoC 5** device (CY8C5588AXI-060), build the project again and program the PSoC 5 device. This is shown in following figure.



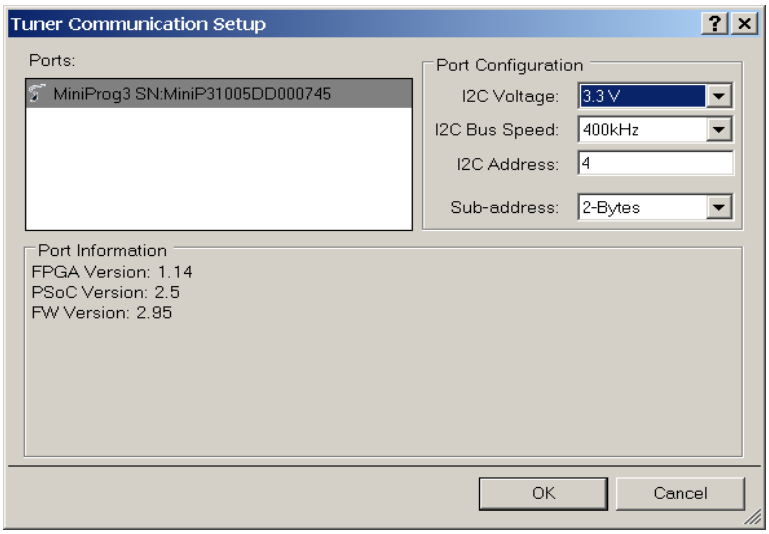
2. Right click on the CapSense component in the TopDesign in PSoC Creator and launch the Tuner as shown in the following figure.



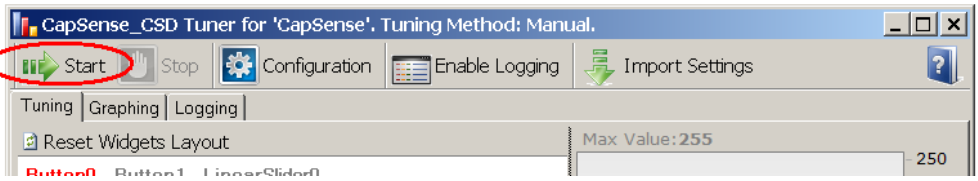
3. Tuner GUI opens as shown in the following figure. Refer to CapSense_CSD component datasheet, “Tuner GUI User Guide” section. Click on the configuration button to open the I2C communication set up window.



4. Set the I2C communication parameters inside the window. Note that the I2C Bus Speed, I2C Address and Sub address size should be same as they are set in the EzI2C component configuration. Click OK to close the window.

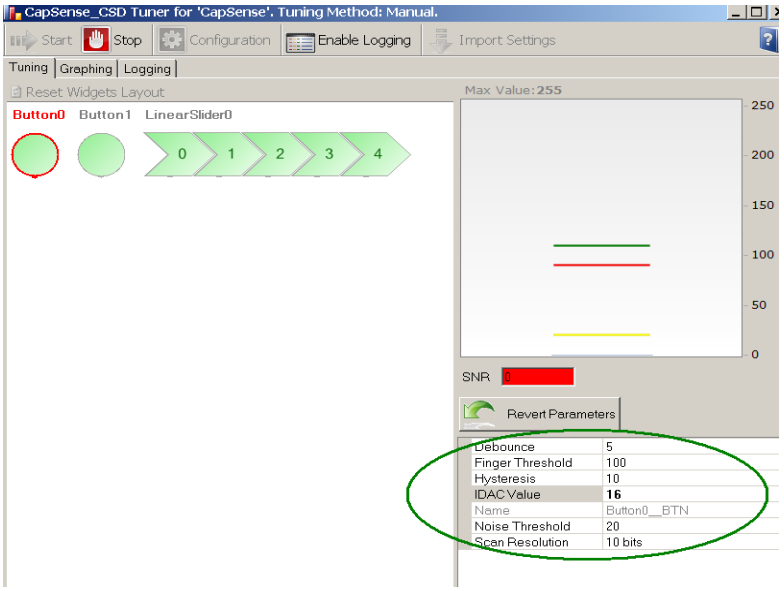


5. Start the tuning by clicking the Start Button in the GUI.



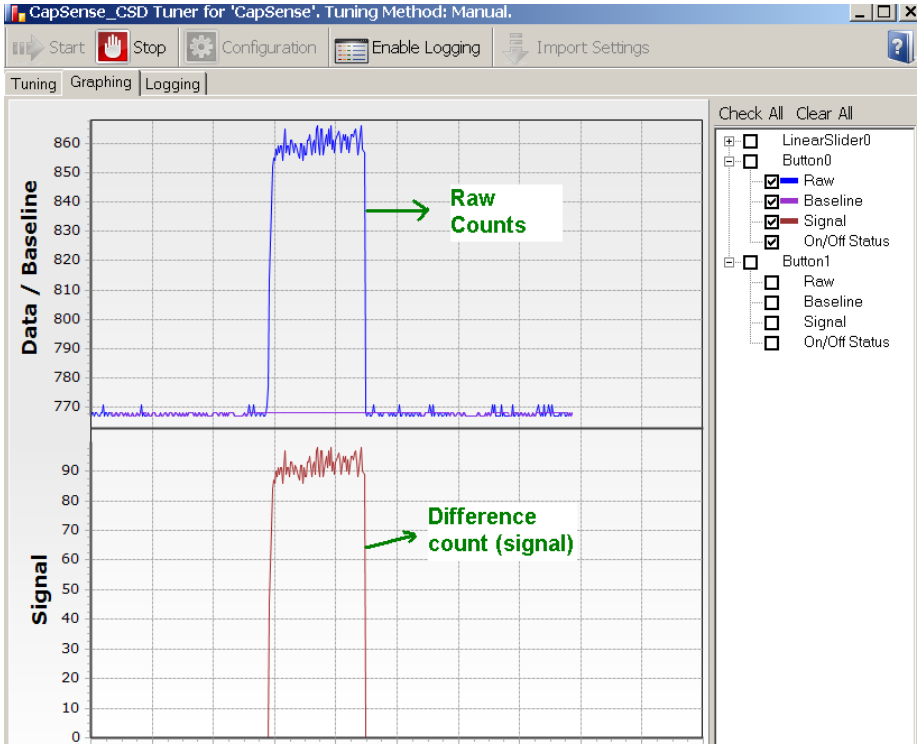
6. Click on any sensor, the parameters belonging to it are shown on the right side (highlighted in green color).

   These parameters can be altered. The following figure shows Button 0 settings.



7. After setting the sensor parameters, now the effect of it can be seen in the ‘Graphing’ tab.

   Different CapSense output variables like ‘Raw counts’, ‘Baseline counts’, ‘Signal (Differnece counts)’ and ‘On/Off Status of sensors’ can be monitored as Graphs.



Tuning Process

a. Analog Switch Divider (PreScaler value):

   Analog Switch Divider sets the frequency at which the sensor capacitance is switched. For CapSense_Sigma Delta (CSD) to work, the sensor capacitor should charge and discharge completely.

   If the sensor is not charging completely to the reference voltage and not discharging completely to the ground then the switching frequency should be reduced.

   Observe the sensor voltage on the oscilloscope by probing on the sensor pin. Note that while observing the voltage on the sensor capacitor with oscilloscope probe, the probe capacitance will be added to the sensor parasitic capacitance. Using probes in 10x mode reduces the capacitance of the probes.

   Make sure that the sensor is charging completely charging and discharging, if not, reduce the ‘Analog Switch Divider’ value in the component configuration. This parameter cannot be changed in the Tuner GUI and it should be set in the component configuration. Therefore when this value is changed in component configuration window, the project should be rebuilt again and programmed on the device.

b. Resolution and Scan speed:

   These parameters decide the sensitivity. Higher resolution and lower scan speed increases the sensitivity of the sensor. This also increases the scan time, therefore there is a tradeoff between sensitivity and scan time.

   In this project, the scan speed is set to Normal in the component configuration. This parameter is global parameter, common to all sensors and cannot be tuned in Tuner GUI. Therefore, if this parameter is changed in component configuration then the project should be built again and reprogrammed to the device.

   Resolution can be tuned in the Tuner GUI, initially set it to 10 bits. If the required sensitivity is not met at the end of the tuning process, it can be increased.

c. IDAC value:

Change the IDAC Value in the GUI such that the raw count lies between 50% and 80% of full-scale reading. Full-scale reading is given by (2^N)-1, where N is resolution set for that sensor. When the IDAC value is increased, the raw counts decrease and vice versa.

If all the IDAC values between 0-255 fail to bring the raw count within 50% to 80%, change the IDAC range in the component configuration. The IDAC Range cannot be altered in the Tuner GUI, it is set in the component configuration. When the IDAC range is changed in the component configuration, the project needs to be built again and programmed on device.

d. SNR:

Measure the Signal to Noise ratio as explained in the application note, Capacitance Sensing - Signal-to-Noise Ratio Requirement for CapSense Applications - AN2403. The requirement for good CapSense application is SNR should be greater than 5:1. If the requirement is not met, increase the resolution to meet the same and re-tune raw count as explained in step c.

e. Firmware related parameters:

Set other sensor parameters finger threshold, noise threshold, hysteresis and debounce in the GUI, as mentioned in the application note, Capacitance Sensing - Signal-to-Noise Ratio Requirement for CapSense Applications - AN2403.

f. Apply the settings to the component:

After tuning all the parameters, click ‘OK’ on the Tuner GUI to apply all the settings to CapSense_CSD component.

Manual tuning needs all the parameters set manually and most of the parameters are set in the GUI while monitoring the CapSense results while some should be set in the CapSense component configuration before the device is programmed with the project.

Parameters set in the Component Configuration window:

1. Analog Switch Divider
2. Scan Speed
3. IDAC Range

Parameters set in the Component GUI:

1. Scan Resoluiton for each sensor
2. IDAC Value for each sensor
3. Finger threshold for each sensor
4. Noise threshold for each sensor
5. Debounce and Hysteresis for each sensor

Note In case of Auto Tuning, the Tune Helper GUI provides the monitoring of CapSense output but you cannot alter the parameter settings as it is automatically done by firmware.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com/